МАТЕРИАЛЫ ЗАДАНИЙ
Всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада»

по профилю «Спутниковые системы»

2024/25 учебный год

ntcontest.ru

УДК 373.5.016:629.783
ББК 74.263.0
С74

Авторы:
С. А. Аретинский, Н. С. Ванаг, О. В. Зубков, Н. Ю. Кузнецов, Б. М. Никитин, А. В. Слоква, Н. М. Яковлева

С74 Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада». Учебно-методическое пособие
Том 18 **Спутниковые системы**
— М.: Ассоциация участников технологических кружков, 2025. — 328 с.

ISBN 978-5-908021-17-3

Данное пособие разработано коллективом авторов на основе опыта проведения всероссийской междисциплинарной олимпиады школьников 8–11 класса «Национальная технологическая олимпиада» в 2024/25 учебном году, а также многолетнего опыта проведения инженерных соревнований для школьников. В пособии собраны основные материалы, необходимые как для подготовки к олимпиаде, так и для углубления знаний и приобретения навыков решения инженерных задач.

В издании приведены варианты заданий по профилю Национальной технологической олимпиады за 2024/25 учебный год с ответами, подробными решениями и комментариями. Пособие адресовано учащимся 8–11 классов, абитуриентам, школьным учителям, наставникам и преподавателям учреждений дополнительного образования, центров молодежного и инновационного творчества и детских технопарков.

Методические материалы также могут быть полезны студентам и преподавателям направлений, относящихся к группам:
11.00.00 Электроника, радиотехника и системы связи
27.00.00 Управление в технических системах

ISBN 978-5-908021-17-3

УДК 373.5.016:629.783
ББК 74.263.0

Оглавление

1. Введение

1.1. Национальная технологическая олимпиада

Всероссийская междисциплинарная олимпиада школьников 8–11 класса «Национальная технологическая олимпиада» (далее — Олимпиада, НТО) проводится в соответствии с распоряжением Правительства Российской Федерации от 10.02.2022 № 211-р при координации Министерства науки и высшего образования Российской Федерации и при содействии Министерства просвещения Российской Федерации, Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации, Министерства промышленности и торговли Российской Федерации, Ассоциации участников технологических кружков, Агентства стратегических инициатив по продвижению новых проектов, АНО «Россия — страна возможностей», АНО «Платформа Национальной технологической инициативы» и Российского движения детей и молодежи «Движение Первых».

Проектное управление Олимпиадой осуществляет структурное подразделение Национального исследовательского университета «Высшая школа экономики» — Центр Национальной технологической олимпиады. Организационный комитет по подготовке и проведению Национальной технологической олимпиады возглавляют первый заместитель Руководителя Администрации Президента Российской Федерации С. В. Кириенко и заместитель Председателя Правительства Российской Федерации Д. Н. Чернышенко.

Национальная технологическая олимпиада — это командная инженерная Олимпиада, позволяющая школьникам работать в самых передовых инженерных направлениях. Она базируется на опыте Олимпиады Кружкового движения НТИ и проводится с 2015 года, а с 2016 года входит в перечень Российского совета олимпиад школьников и дает победителям и призерам льготы при поступлении в университеты.

Всего заявки на участие в десятом юбилейном сезоне (2024–25 гг.) самых масштабных в России командных инженерных соревнованиях подали более 140 тысяч школьников. Общий охват олимпиады с 2015 года превысил 880 тысяч участников.

НТО способствует формированию профессиональной траектории школьников, увлеченных научно-техническим творчеством и помогает им:

- определить свой интерес в мире современных технологий;
- получить опыт решения комплексных инженерных задач;
- осознанно выбрать вуз для продолжения обучения и поступить в него на льготных условиях.

Кроме того, НТО позволяет каждому участнику познакомиться с перспективными направлениями технологического развития, ведущими экспертами и найти единомышленников.

Ценности НТО

Национальная технологическая олимпиада — командные инженерные соревнования для школьников и студентов. Олимпиада создает уникальное пространство, основанное на общих ценностях и смыслах, которыми делятся все участники процесса: школьники, студенты, организаторы, наставники и эксперты. В основе Олимпиады лежит представление о современном технологическом образовании как новом укладе жизни в быстро меняющемся мире. Эта модель предполагает:

- доступность качественного обучения для всех, кто стремится к знаниям;
- возможность непрерывного развития;
- совместное формирование среды, где гуманитарные знания и новые технологии взаимно усиливают друг друга.

Это — образ общества будущего, в котором участники Олимпиады оказываются уже сегодня.

Решать прикладные задачи, нацеленные на умножение общественного блага

В заданиях Олимпиады используются актуальные вызовы науки и технологий, адаптированные под уровень школьников. Они имеют прикладной характер и отражают реальные потребности общества, а системное и профессиональное решение подобных задач способствует развитию общего блага. Олимпиада предоставляет возможность попробовать себя в этом направлении уже сегодня и найти единомышленников.

Создавать, а не только потреблять

Стремление к созданию нового ценится выше потребления готового, а ориентация на общественную пользу — выше личной выгоды. Это не исключает заботу о собственных интересах, но подчеркивает: творчество приносит больше удовлетворения, чем пассивное потребление. Олимпиада — совместный труд организаторов, партнеров и участников, в котором важнее стремление решать общие задачи, чем критика чужих усилий.

Работать в команде

Командная работа рассматривается не только как эффективный способ достижения целей, но и как основа для формирования сообщества, объединенного общими ценностями. Команда помогает раскрыть индивидуальность каждого, при этом сохраняя уважение к другим. Такие горизонтальные связи необходимы для реализации амбициозных технологических проектов. Олимпиада способствует формированию подобного сообщества и приглашает к его созданию всех заинтересованных.

Осваивать и ответственно развивать новые технологии

Сообщество Национальной технологической олимпиады — часть Кружкового движения НТИ, объединенные интересом к современным технологиям, стремлением

к их пониманию и созданию нового. Возможности технологий постоянно расширяются, однако развитие должно сопровождаться ответственностью. Этика инженера и ученого предполагает осознание последствий своих решений. Главное правило — создавая новое, не навредить.

Играть честно и пробовать себя

Ценится честная победа, достигнутая в рамках установленных правил. Это предполагает отказ от списывания, давления и манипуляций. Честная игра означает уважение к себе, команде и соперникам. Олимпиада поддерживается как безопасное пространство, где каждый может пробовать новое, не опасаясь ошибок, и постепенно становиться сильнее и увереннее в себе.

Быть человеком

Соревнования — это сложный и эмоционально насыщенный процесс, в котором особенно важны порядочность, вежливость и чуткость. Эмпатия, уважение и забота делают участие полезным и комфортным. Высоко ценится бережное отношение к людям и их труду, отказ от токсичной критики и готовность нести ответственность за слова и поступки. Участие в общем деле помогает не только окружающим, но и самому человеку.

Организационная структура НТО

НТО — межпредметная олимпиада. Спектр соревновательных направлений (профилей НТО) сформирован на основе актуального технологического пакета и связан с решением современных проблем в различных технологических отраслях. С полным перечнем направлений (профилей) можно ознакомиться на сайте НТО: https://ntcontest.ru/tracks/nto-school/.

Соревнования в рамках НТО проводятся по четырем трекам:

1. НТО Junior для школьников (5–7 классы).
2. НТО школьников (8–11 классы).
3. НТО студентов.
4. Конкурс цифровых портфолио «Талант НТО».

В 2024/25 учебном году 21 профиль НТО включен в Перечень олимпиад школьников, ежегодно утверждаемый Приказом Министерства науки и высшего образования Российской Федерации, а также в Перечень олимпиад и иных интеллектуальных и (или) творческих конкурсов, утверждаемый приказом Министерства просвещения Российской Федерации. Это дает право победителям и призерам профилей НТО поступать в вузы страны без вступительных испытаний (БВИ), получить 100 баллов ЕГЭ или дополнительные 10 баллов за индивидуальные достижения. Преимущества при поступлении победителям и призерам НТО предлагают более 100 российских вузов.

НТО для школьников 8–11 классов проводится в три этапа:

- Первый отборочный этап — заочный индивидуальный. Участникам предлагаются предметный тур, состоящий из задач по двум предметам, связанным

с выбранным профилем, а также инженерный тур, задания которого погружают участников в тематику профиля; образовательный модуль формирует теоретические знания и представления.

- Второй отборочный этап — заочный командный. На этом этапе участники выполняют как индивидуальные задания на проверку компетенций, так и командные задачи, соответствующие выбранному профилю.
- Заключительный этап — очный командный. В течение 5–6 дней команды участников со всей страны, успешно прошедшие оба отборочных этапа, соревнуются в решении комплексных прикладных инженерных задач.

Профили НТО 2024/25 учебного года и соответствующий уровень РСОШ

Профили II уровня РСОШ:

- Автоматизация бизнес-процессов.
- Автономные транспортные системы.
- Беспилотные авиационные системы.
- Водные робототехнические системы.
- Инженерные биологические системы.
- Наносистемы и наноинженерия.
- Нейротехнологии и когнитивные науки.
- Технологии беспроводной связи.
- Цифровые технологии в архитектуре.
- Ядерные технологии.

Профили III уровня РСОШ:

- Анализ космических снимков и геопространственных данных.
- Аэрокосмические системы.
- Большие данные и машинное обучение.
- Геномное редактирование.
- Интеллектуальные робототехнические системы.
- Интеллектуальные энергетические системы.
- Информационная безопасность.
- Искусственный интеллект.
- Летающая робототехника.
- Спутниковые системы.
- Кластер «Виртуальные миры»:
    - ◇ Разработка компьютерных игр.
    - ◇ Технологии виртуальной реальности.
    - ◇ Технологии дополненной реальности.

Профили без уровня РСОШ:

- Инфохимия.
- Квантовый инжиниринг.
- Новые материалы.
- Программная инженерия в финансовых технологиях.

- Современная пищевая инженерия.
- Умный город.
- Урбанистика.
- Цифровые сенсорные системы.
- Разработка мобильных приложений.

Обратите внимание на то, что в олимпиаде 2025/26 учебного года список профилей, в т. ч. входящих в РСОШ, и уровни РСОШ могут поменяться.

Участие в НТО старшеклассников может принять любой школьник, обучающийся в 8–11 классе. Чаще всего Олимпиада привлекает:

- учащихся технологических кружков, интересующихся инженерными и робототехническими соревнованиями;
- школьников, увлеченных олимпиадами и предпочитающих межпредметный подход;
- энтузиастов передовых технологий;
- активных участников хакатонов, проектных конкурсов и профильных школ;
- будущих предпринимателей, ищущих команду для реализации стартап-идей;
- любознательных школьников, стремящихся выйти за рамки школьной программы.

Познакомить школьников с НТО и ее направлениями, а также мотивировать их на участие в Олимпиаде можно с помощью специальных мероприятий — Урока НТО и Дней НТО. Методические рекомендации для педагогов по проведению Урока НТО и организации Дня НТО в образовательной организации размещены на сайте: https://nti-lesson.ru. Здесь можно подобрать и скачать готовые сценарии занятий и подборки материалов по различным направлениям Олимпиады.

Участвуя в НТО, школьники получают возможность работать с практико-ориентированными задачами в области прорывных технологий, собирать команды единомышленников, погружаться в профессиональное сообщество, а также заработать льготы для поступления в вузы.

По всей стране работают площадки подготовки к НТО, которые помогают привлекать участников и проводят мероприятия по подготовке к этапам Олимпиады. Такие площадки могут быть открыты на базе:

- школ и учреждений дополнительного образования;
- частных кружков по программированию, робототехнике и другим технологическим направлениям;
- вузов;
- технопарков и других образовательных и научно-технических организаций.

Любое образовательное учреждение, ученики которого участвуют в НТО или НТО Junior, может стать площадкой подготовки к Олимпиаде и присоединиться к Кружковому движению НТИ. Подробные инструкции о том, как стать площадкой подготовки, размещены на сайте: https://ntcontest.ru. Условия регистрации и требования к ним актуализируются с развитием Олимпиады, а обновленная информация публикуется перед началом каждого нового цикла.

Наставники НТО

В Национальной технологической олимпиаде большое внимание уделяется работе с **наставниками** — людьми, сопровождающими участников на всех этапах подготовки и участия в Олимпиаде. Наставник оказывает поддержку как в решении организационных вопросов, так и в развитии технических и социальных навыков школьников, включая умение работать в команде.

Наставником НТО может стать любой взрослый, готовый помогать школьникам развиваться и готовиться к участию в инженерных соревнованиях. Это может быть:

- учитель школы или преподаватель вуза;
- педагог дополнительного образования;
- руководитель кружка;
- родитель школьника;
- специалист из технологической области или представитель бизнеса.

Даже если наставник сам не обладает достаточными знаниями в определенной области, он может привлекать к подготовке коллег и экспертов, а также оказывать поддержку и организовывать процесс обучения для самостоятельных учеников. Сегодня сообщество наставников НТО насчитывает более **7 000 человек** по всей стране.

Главная цель наставника — **организовать системную подготовку к Олимпиаде в течение всего учебного года**, поддерживать интерес и мотивацию участников, а также помочь им справляться с возникающими трудностями. Также наставник фиксирует цели команды и каждого участника, чтобы в дальнейшем можно было проанализировать развитие профессиональных и личных компетенций.

Основные направления работы наставника

Организационные задачи:

- Информирование и мотивация: наставник рассказывает учащимся об НТО, ее этапах и преимуществах, помогает с выбором подходящего профиля, ориентируясь на интересы и способности школьников.
- Составление программы подготовки: формируется расписание и план занятий, организуется работа по освоению необходимых знаний и навыков.
- Контроль сроков: наставник следит за календарем Олимпиады и напоминает участникам о сроках решения заданий отборочных этапов.

Содержательная подготовка:

- Оценка компетенций участников: наставник помогает определить сильные и слабые стороны учеников и подбирает задания и материалы для устранения пробелов.
- Подготовка к отборочным этапам: помощь в изучении рекомендованных материалов, заданий прошлых лет, онлайн-курсы по профилям.
- Подготовка к заключительному этапу: разбираются задачи заключительных этапов прошлых лет, отслеживаются подготовительные мероприятия (очные и дистанционные), в которых наставник рекомендует ученикам участвовать.

Развитие личных и командных навыков:

- Формирование команд: наставник помогает сформировать сбалансированные команды для второго отборочного и финального этапов, распределить роли, при необходимости ищет участников из других регионов и организует онлайн-коммуникацию.
- Анализ прогресса и опыта: после каждого этапа проводится совместная рефлексия, обсуждаются успехи и трудности, выявляются зоны роста и направления для дальнейшего развития.
- Поддержка и мотивация: наставник поддерживает интерес и энтузиазм участников (особенно в случае неудачных результатов), помогает справиться с разочарованием и сохранить настрой на дальнейшее участие.
- Построение индивидуальной образовательной траектории: наставник помогает школьникам осознанно планировать дальнейшее обучение: выбирать курсы, участвовать в конкурсах, определяться с вузами и направлениями подготовки.

Поддержка наставников НТО

Работе наставников посвящен отдельный раздел на сайте НТО: https://ntcontest.ru/mentors/.

Для систематизации знаний и подходов к работе наставников в рамках инженерных соревнований разработан курс «Дао начинающего наставника: как сопровождать инженерные команды»: https://stepik.org/course/124633/. Курс формирует общие представления об их работе в области подготовки участников к инженерным соревнованиям.

Для совершенствования профессиональных компетенций по направлениям профилей создан курс «Дао начинающего наставника: как развивать технологические компетенции»: https://stepik.org/course/186928/.

Для организации занятий с учениками педагогам предлагаются образовательные программы, разработанные на основе многолетнего опыта организации подготовки к НТО. В настоящий момент они представлены по передовым технологическим направлениям:

- компьютерное зрение;
- геномное редактирование;
- водная, летающая и интеллектуальная робототехника;
- машинное обучение и искусственный интеллект;
- нейротехнологии;
- беспроводная связь, дополненная реальность.

Программы доступны на сайте: https://ntcontest.ru/mentors/education-programs/.

Регистрируясь на платформе НТО, наставники получают доступ к личному кабинету, в котором отображается расписание отборочных соревнований и мероприятий по подготовке, требования к знаниям и компетенциям при решении задач отборочных этапов.

Сообщество наставников НТО существует и развивается. Ежегодно Кружко-

вое движение НТИ проводит Всероссийский конкурс технологических кружков: https://konkurs.kruzhok.org/. Принять участие в конкурсе может каждый наставник.

В 2022 году было выпущено пособие «Технологическая подготовка инженерных команд. Методические рекомендации для наставников». Методические рекомендации предназначены для учителей технологий, а также наставников и педагогов кружков и центров дополнительного образования. Рекомендации направлены на помощь в процессе преподавания технологий в школе или в кружке. Пособие построено на примерах из реального опыта работы со школьниками, состоит из теоретических положений, посвященных популярным взглядам в педагогике на тему подготовки инженерных команд к соревнованиям. Электронное издание доступно по ссылке: https://journal.kruzhok.org/tpost/pggs3bp7y1-tehnologicheskaya-podgotovka-inzhenernih.

В нем рассмотрены особенности подготовки к пяти направлениям:

- Большие данные.
- Машинное обучение.
- Искусственный интеллект.
- Спутниковые системы.
- Летающая робототехника.

Для наставников НТО разработана и постоянно пополняется страница с материалами для профессионального развития: https://nto-forever.notion.site/c9b9cbd21542479b97a3fa562d15e32a.

1.2. Спутниковые системы

Основная направленность профиля Спутниковые системы — проектирование искусственных спутников Земли, включая разработку его конструктивных компонентов, программирование и проработку аспектов космической миссии.

В течение десятилетий основные задачи космонавтики решались большими аппаратами со сложнейшими инженерными системами, которые создавались исключительно крупными корпорациями и национальными космическими агентствами. Теперь, в эпоху миниатюризации электроники, стали все чаще появляться спутники столь компактные, что человек может взять их в руку. Космический аппарат такого класса вполне под силу сделать коммерческой компании, университету или группе энтузиастов. При этом могут достигаться профессиональные цели, такие как, например, съемка всей земной поверхности в рамках проекта PlanetLab. На этапах профиля Спутниковые системы школьники сталкиваются с технологическими вызовами разработки и применения именно таких сверхмалых космических аппаратов.

Командная задача состоит из нескольких взаимосвязанных компонентов, требующих разносторонних инженерных компетенций. Ключевыми являются навыки конструирования, программирования микроконтроллеров семейства STM32, схемотехники и орбитальной механики. Большая часть этих навыков востребована не только в космической отрасли, но и в любой инженерной сфере.

На проходящих в течение года вовлекающих мероприятиях, таких как «Урок НТО», все желающие могут ознакомиться с назначением космических аппаратов, их составом, а также чуть глубже узнать об особенностях перемещения спутников в пространстве и о том, как это влияет на их использование и роль в глобальных коммуникациях.

На схеме (рис. 1.2.1) показана взаимосвязь этапов цикла профиля на примере задачи сезона 2024–25 гг.

Для понимания принципов работы космического аппарата на орбите требуется подготовка по темам: кинематика, динамика и электромагнетизм, а для написания алгоритмов управления необходимо знать базовый синтаксис языка программирования (C, JavaScript) и понимать принципы построения алгоритмов в целом. Поэтому наиболее близкими школьными предметами к тематике профиля являются физика и информатика, по которым идет отбор на первом этапе.

В 2024–25 гг. **на первом этапе** в рамках инженерного тура проводится подготовка по выделенным ролям в команде: физик-баллистик, программист полезной нагрузки, программист платформы спутника и инженер связи.

На втором этапе начинается погружение в специфику отрасли: задания становятся не только инструментом отбора, но и содержат образовательные элементы (рекомендации, вебинары и пр.). Это помогает участникам проходить самоподготовку по востребованным в профиле компетенциям.

Участие в заключительном этапе, как отмечают сами школьники, дает немало возможностей по проверке и развитию приобретенных компетенций, а также незаменимый опыт командной деятельности и разработки технического проекта.

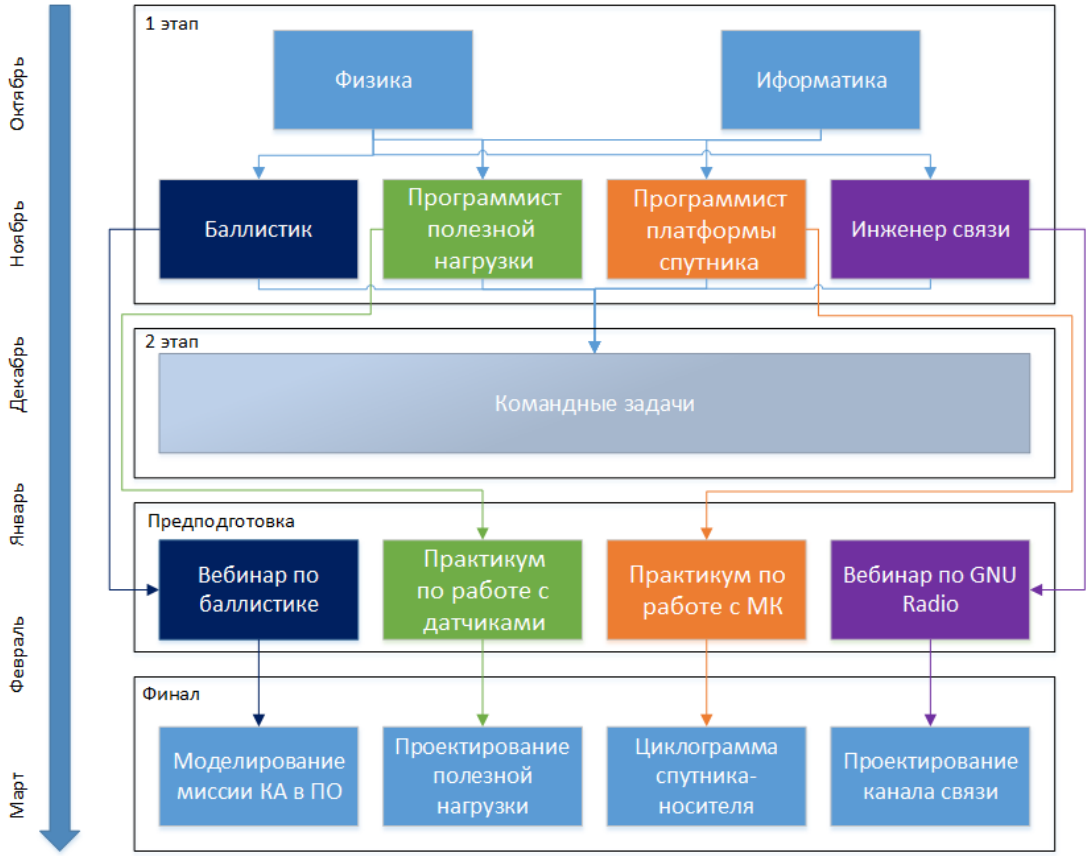


Рис. 1.2.1

Выпускники профиля впоследствии поступают в такие вузы, как МГТУ им. Баумана, МАИ, НГУ и пр. Приобретенный опыт позволяет им начать работать как в профессиональной, так и в образовательной сферах (проектные смены на базе площадок дополнительного образования, операторы на заключительном этапе Олимпиады).

2. Первый отборочный этап

2.1. Работа наставника НТО на этапе

Педагог-наставник играет важную роль в подготовке участника к первому отборочному этапу Национальной технологической олимпиады. На этом этапе школьникам предстоит справиться как с предметными задачами, соответствующими профилю, так и с заданиями инженерного тура, погружающими в выбранную технологическую область.

Наставник может организовать подготовку участника, используя разнообразные форматы и ресурсы:

- Разбор заданий прошлых лет. Совместный анализ задач отборочного этапа предыдущих лет позволяет понять структуру, уровень сложности и типичные подходы к решению. Это формирует у школьника устойчивые стратегии работы с олимпиадными заданиями.
- Мини-соревнования. Проведение тренировочных турниров с заданиями предметных олимпиад муниципального уровня помогает развить соревновательный навык, тренирует скорость и уверенность при решении задач в ограниченное время.
- Углубленные занятия. Наставник может выстроить образовательную траекторию, опираясь на рекомендации разработчиков профиля, и провести занятия по ключевым темам. Это особенно важно для системного понимания предметной области.
- Использование онлайн-курсов. Для самостоятельной подготовки и проверки знаний участник может использовать предметные курсы НТО, размещенные на платформах Степик и Яндекс Контест. Наставник может также организовать занятия с использованием этих материалов в рамках групповой или индивидуальной подготовки.
- Привлечение внешних экспертов. Если у наставника нет достаточной экспертизы в какой-либо предметной области, он может пригласить других педагогов или специалистов для проведения тематических занятий.
- Поддержка в инженерном туре. Инженерный тур включает теоретические материалы и задания, помогающие глубже погрузиться в тематику профиля. Наставник может сопровождать изучение курса, помогать в разборе теоретических вопросов и тренировать участника на практических задачах.

Таким образом, наставник не только помогает систематизировать подготовку, но и мотивирует участника, создавая для него комфортную и продуктивную образовательную среду.

2.2. Предметный тур. Информатика

2.2.1. Первая волна. Задачи 8–11 класса

Задачи первой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63452/enter/.

Задача 2.2.1.1. Ускорение ускорения (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим модель движения тела. Будем фиксировать такие параметры, как координата, скорость, ускорение и ускорение ускорения (рывок). Если некоторый параметр равен a и имеет скорость изменения v, то в следующий момент времени этот параметр будет равен $a + v$.

Например, если тело имело координату, равную 10, скорость, равную 20, ускорение, равное 30 и ускорение ускорения, равное 40, то в следующий момент оно будет иметь координату 30, скорость 50 и ускорение 70. Ускорение ускорения будем считать в этой задаче постоянной величиной.

Задача довольно проста: тело в начальный момент времени 0 находится в точке с координатой 0, скоростью 0 и ускорением 0. На это тело действует постоянное ускорение ускорения, равное 6. Требуется определить, в точке с какой координатой окажется это тело в момент времени t.

Формат входных данных

В единственной строке находится одно число t, где $0 \leqslant t \leqslant 10^6$.

Формат выходных данных

Вывести одно число — координату, в которой окажется тело в момент времени t.

Примеры

Пример №1

Стандартный ввод
6
Стандартный вывод
120

Пример №2

Стандартный ввод
2
Стандартный вывод
0

Пример №3

Стандартный ввод
1000000
Стандартный вывод
999997000002000000

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int t;
    cin >> t;
    cout << ((t * (t - 1)) * (t - 2)) << endl;
}
```

Задача 2.2.1.2. Двойное остекление (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У деда Василия есть два прямоугольных куска стекла. Один из них имеет размеры $a \times b$, другой — $c \times d$. Дед собирается из этих кусков сделать окно с двойным остеклением. Он хочет, чтобы окно было обязательно квадратным и как можно большим по размеру. Дед должен вырезать из имеющихся у него прямоугольников два одинаковых квадрата максимально возможного размера. Нужно написать программу, которая по заданным a, b, c, d найдет максимальные размеры квадратного окна. Имейте ввиду, что оба квадрата могут быть вырезаны и из одного прямоугольного куска стекла.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника a, b через пробел, во второй — размеры второго прямоугольника c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — максимальную сторону квадратного двойного окна, которое можно вырезать из заданных на входе прямоугольных кусков стекла. Ответ может быть нецелым, требуется вывести его с точностью 1 знак после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 10 9 6
Стандартный вывод
5

Пример №2

Стандартный ввод
4 10 9 6
Стандартный вывод
4.5

Комментарий

Второй пример показывает, что иногда лучше вырезать оба квадрата из одного и того же куска стекла.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    double a, b, c, d;
    cin >> a >> b >> c >> d;
    double a0 = min({a, b, c, d});
    double a1 = min(max(a, b) / 2.0, min(a, b));
    double a2 = min(max(c, d) / 2.0, min(c, d));
    double ans =  max({a0, a1, a2});
    if( (int)ans == ans ){
        int ians = ans;
        cout << ians << endl;
        return 0;
    }
    cout.precision(1);
    cout << fixed<< ans << endl;
}
```

Задача 2.2.1.3. О золотой рыбке и... досках (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

После событий известной сказки А. С. Пушкина старик решил принципиально не пользоваться услугами золотой рыбки. Поэтому для того чтобы изготовить новое корыто, он честно заготовил n одинаковых досок.

Но гостивший в это время у старика со старухой внук решил, что ему нужно научиться пилить. И, не сказав ничего своему деду, внук быстро распилил каждую из досок на две части. В итоге у старика оказались $2n$ кусков досок. Самое интересное, что все эти куски оказались разными по длине, но имели целочисленные размеры. К сожалению, старик забыл, какова была исходная длина целых досок.

Формат входных данных

В первой строке задается целое число n — исходное количество целых досок, где $1 \leqslant n \leqslant 10^5$.

Во второй строке заданы $2n$ целых чисел d_i — длины всех кусков, которые получились после «тренировки» внука, где $1 \leqslant d_i \leqslant 10^9$. Гарантируется, что эти числа попарно различны, и их можно разбить на пары одинаковых по сумме чисел.

Все эти части досок пронумерованы от 1 до $2n$ в том порядке, в котором они заданы на входе.

Формат выходных данных

В первую строку вывести одно число — исходную длину целых досок.

В следующих n строках вывести пары номеров кусков досок, которые составляют по длине целые доски. Номера выводить через один пробел, внутри пары сначала должен идти меньший номер, затем больший. Пары должны быть выведены в порядке возрастания первых номеров в парах.

Примеры

Пример №1

Стандартный ввод
3 4 8 2 3 6 7

Стандартный вывод
10 1 5 2 3 4 6

Комментарий

Отсортируем куски и далее будем брать один из начала и второй к нему из конца.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    vector<pair<int, int> > v(2 * n);
     for(int i = 0; i < 2 * n; i++){
        int d;
        cin >> d;
        v[i] = {d, i + 1};
    }
    sort(v.begin(), v.end());
    vector<pair<int, int> > ans(n);
    for(int i = 0; i < n; i++){
```

```
16          ans[i] = {v[i].second, v[2 * n - i - 1].second};
17          if(ans[i].first > ans[i].second){
18              swap(ans[i].first, ans[i].second);
19          }
20      }
21      sort(ans.begin(), ans.end());
22      cout << v[0].first + v.back().first<< endl;
23      for(int i = 0; i < n; i++){
24          cout << ans[i].first<<' '<< ans[i].second<< endl;
25      }
26  }
```

Задача 2.2.1.4. Бонусы и экономия (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Технология производства некоторой металлической детали предполагает вытачивание ее из металлической заготовки. При этом образуются стружки, которые не стоит выкидывать. Ведь из a комплектов стружек (оставшихся после обработки a заготовок) можно бесплатно выплавить еще одну заготовку, которую снова можно использовать для выточки детали и создания еще одного комплекта стружек.

Заготовки можно купить на оптовом складе, при этом в целях привлечения клиентов, проводится акция «купи b заготовок, тогда еще одну получишь бесплатно».

Требуется изготовить c деталей. Нужно определить минимальное число заготовок, которые нужно купить за деньги, чтобы с учетом бонусных заготовок и экономии на стружках можно было изготовить требуемое число деталей.

Формат входных данных

В одной строке через пробел заданы три целых числа a, b, и c такие, что $2 \leqslant$ $\leqslant a \leqslant 10^{18}$, $1 \leqslant b$, $c \leqslant 10^{18}$.

Формат выходных данных

Вывести одно целое число — минимальное количество заготовок, которые нужно купить, чтобы с учетом всех бонусов и экономии выточить c конечных деталей.

Примеры

Пример №1

Стандартный ввод
4 5 41
Стандартный вывод
26

Примечания

В примере из условия нужно закупить 26 заготовок. Тогда за каждые пять купленных заготовок будет предоставлена одна бесплатная, итого по акции добавится еще пять заготовок, то есть получится 31 заготовка. Далее из 31 заготовки выточится 31 деталь, останется 31 комплект стружек. Из каждых четырех комплектов выплавится дополнительная заготовка, получится семь заготовок и три комплекта стружек. Из семи заготовок выточится семь деталей и останется семь комплектов стружек, три комплекта стружек осталось с первого шага, итого 10 комплектов стружек. Из них выплавится еще две заготовки, дающие две детали и два комплекта стружек. Собрав эти два комплекта с двумя, оставшимися от 10, получим еще одну заготовку, из которой выточится еще одна деталь. Останется один комплект стружек, который уже никак не получится использовать. Итого будет произведена $31 + 7 + 2 + 1 = 41$ деталь.

Комментарий

Методом бинарного поиска можно подобрать минимальное необходимое количество исходных заготовок.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int f1(int M, int a){
    int res = 0, z = 0;
    while(1){
        if(M == 0 && z < a){
            return res;
        }
        res += M;
        M = M + z;
        z = M % a;
        M = M / a;
    }
}
```

```
16 int f2(int M, int b){
17     return M + M / b;
18 }
19 signed main(){
20     int a, b, c;
21     cin >> a >> b >> c;
22     int L = 0, R = 1;
23     while(f1(R, a) <= c){
24         R *= 2;
25     }
26     while(R - L > 1){
27         int M = (R + L) / 2;
28         if(f1(M, a) < c){
29             L = M;
30         }
31         else{
32             R = M;
33         }
34     }
35     int z = R;
36     L = 0, R = 1;
37     while(f2(R, b) <= z){
38         R *= 2;
39     }
40     while(R - L > 1){
41         int M = (R + L) / 2;
42         if(f2(M, b) < z){
43             L = M;
44         }
45         else{
46             R = M;
47         }
48     }
49     cout << R << endl;
50 }
```

Задача 2.2.1.5. Сон таксиста (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному таксисту приснился красочный сон. Во сне он живет и работает в некотором городе, где абсолютно все улицы с односторонним движением. Эти улицы устроены так, что невозможно проехать с какого-либо перекрестка так, чтобы вернуться обратно на этот же перекресток, то есть в дорожной сети города нет циклов.

Таким образом, если с перекрестка A можно попасть по направлению движения улиц на перекресток B, то люди вызывают такси, иначе их везет специальный муниципальный подземный транспорт бесплатно.

В связи с такими странными правилами, таксистам в этом городе разрешено законом везти пассажира по любому маршруту, не нарушающему направления движения. Все в этом городе привыкли к такой ситуации и абсолютно спокойно относятся к тому, что таксисты везут их самым длинным путем. Разумеется, заработок таксиста за одну поездку прямо пропорционален ее длине. Для упрощения будем считать, что стоимость 1 км поездки составляет ровно 1 руб.

Схема дорог города задана. Перекрестки города пронумерованы числами от 1 до n. Таксист в своем сне находится на перекрестке номер S. Напишите программу, которая подскажет ему, сколько он максимально сможет заработать, когда ему придет заказ от клиента. Так как он не знает, куда попросит его везти клиент, нужно для каждого перекрестка от 1 до n указать максимальную стоимость поездки до этого перекрестка из пункта S на такси. Если по правилам на такси добраться из пункта S до какого-то перекрестка нельзя, вывести -1.

Формат входных данных

Дорожная сеть задана следующим образом: в первой строке находятся два числа через пробел n и m — число перекрестков и число улиц в городе, где $2 \leqslant n, m \leqslant 2 \cdot \cdot 10^5$.

В следующих m строках задана очередная односторонняя улица в виде трех чисел A, B, d через пробел, где A — начало улицы, B — конец улицы и d — ее длина. $1 \leqslant A, B \leqslant n$, $1 \leqslant d \leqslant 10^9$. Гарантируется, что в этой дорожной сети нет циклов. Некоторые пары перекрестков могут быть соединены двумя и более односторонними улицами. Дорожная сеть может быть неплоской за счет мостов и тоннелей.

В последней строке ввода содержится номер стартового перекрестка S, $1 \leqslant S \leqslant \leqslant n$.

Формат выходных данных

Вывести n чисел в одну строку через пробел. i-е число обозначает длину самого длинного пути с перекрестка номер S до перекрестка номер i. Если до перекрестка номер i от S нельзя доехать, не нарушая правила движения, вывести -1.

Примеры

Пример №1

Стандартный ввод
10 20 9 10 15 9 8 3 8 10 7 7 8 4 7 10 10 5 8 2 5 9 10

Стандартный ввод

```
5 6 5
7 6 5
4 6 8
3 6 4
3 4 6
5 3 2
2 5 2
2 3 3
3 1 5
1 4 2
2 1 7
4 7 4
6 8 1
5
```

Стандартный вывод

```
7 -1 2 9 0 18 13 19 10 26
```

Комментарий

Задача решается методом динамического программирования на ориентированном ациклическом графе.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
int n, m;
vector<vector<pair<int, int> > > G;
vector<int> order, used;
void dfs(int a){
    used[a] = 1;
    for(auto to : G[a]){
        if(!used[to.first]){
            dfs(to.first);
        }
    }
    order.push_back(a);
}
signed main(){
    cin >> n >> m;
    G.resize(n + 1);
    used.resize(n + 1, 0);
    for(int i = 0; i < m; i++){
        int a, b, d;
        cin >> a >> b >> d;
        G[a].push_back({b, d});
    }
```

```
25      int s;
26      cin >> s;
27      dfs(s);
28      reverse(order.begin(), order.end());
29      vector<int> dp(n + 1, -1);
30      dp[s] = 0;
31      for(auto el : order){
32          for(auto to : G[el]){
33              dp[to.first] = max(dp[to.first], dp[el] + to.second);
34          }
35      }
36      for(int i = 1; i <= n; i++){
37          cout << dp[i] <<' ';
38      }
39  }
```

2.2.2. Вторая волна. Задачи 8–11 класса

Задачи второй волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63454/enter/.

Задача 2.2.2.1. Игра на планшете (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маленький Андрей изучает геометрические фигуры при помощи игры на планшете. У него есть прямоугольные треугольники четырех цветов и ориентаций: желтые, зеленые, красные и синие. Для каждой разновидности треугольников есть заданное количество экземпляров этих треугольников. Более точно: у Андрея есть a желтых, b зеленых, c красных и d синих треугольников. Помимо этого у него есть прямоугольная таблица $n \times m$.

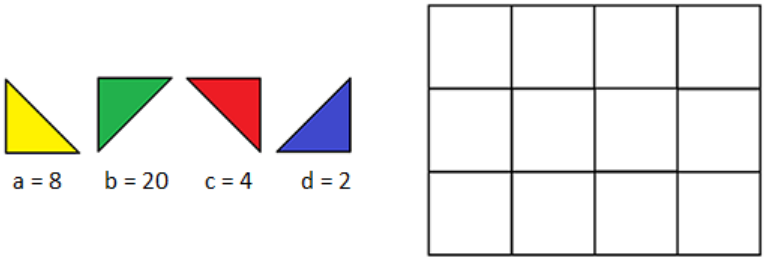


Рис. 2.2.1

Треугольники одного цвета имеют одну и ту же ориентацию, которую нельзя поменять. Андрей может только взять очередной треугольник и переместить его параллельным сдвигом в одну из ячеек этой прямоугольной таблицы. При этом в одну ячейку можно поместить либо вместе желтый и красный треугольники, либо вместе зеленый и синий, либо один любой треугольник из имеющихся.

Андрей хочет расположить в ячейках таблицы как можно больше треугольников из тех, что у него имеются. Нужно подсказать ему максимальное количество треугольников, которые получится разместить в таблице.

Формат входных данных

В первой строке содержатся четыре целых числа a, b, c и d через пробел — количество желтых, зеленых, красных и синих треугольников соответственно.

Во второй строке содержатся два целых числа n и m через пробел — размеры прямоугольной таблицы.

Все числа в пределах от 1 до 10^9.

Формат выходных данных

Вывести одно число — максимальное количество треугольников, которые можно при заданных условиях разместить в таблице.

Примеры

Пример №1

Стандартный ввод
`8 20 4 2` `3 4`

Стандартный вывод
`18`

Примечания

На рис. 2.2.2 представлен один из примеров размещения 18 треугольников из 34 заданных на входе.

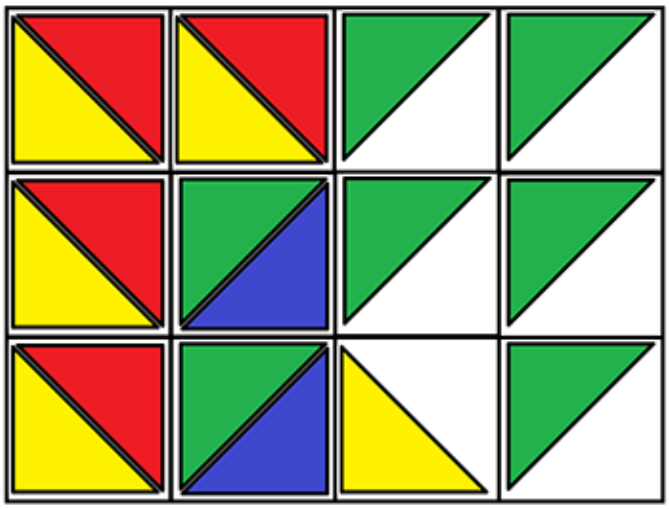


Рис. 2.2.2

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d, n, m;
    cin >> a >> b >> c >> d >> n >> m;
    if(a > c){
        swap(a, c);
    }
    if(b > d){
        swap(b, d);
    }
    int f = a + b;
    int k = n * m;
    if(k <= f){
        cout << k * 2;
        return 0;
    }
    k -= f;
    c -= a;
    d -= b;
    cout << f * 2 + min(k, c + d) << endl;
}
```

Задача 2.2.2.2. Старая задача на новый лад (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одна старая задача имеет следующий вид:

«Разбить число 45 на сумму четырех слагаемых так, что если к первому прибавить 2, из второго вычесть 2, третье умножить на 2, а четвертое разделить на 2, то получится одно и то же число».

Ответ к этой задаче — четыре числа 8, 12, 5 и 20. Можно убедиться, что в сумме они дают число 45, а если с каждым из них проделать соответствующую арифметическую операцию, то получится одно и то же число 10.

Необходимо решить чуть более общую задачу: даны числа n и k. Нужно представить число n в виде суммы четырех целых неотрицательных слагаемых $a + b + c + d$ таких, что $a + k = b - k = c \cdot k = d/k$. Гарантируется, что для заданных n и k такое разбиение существует.

Формат входных данных

В одной строке через пробел два числа n и k, где $1 \leqslant n \cdot k \leqslant 10^{18}$.

Формат выходных данных

Вывести через пробел в одну строку четыре целых неотрицательных числа a, b, c, d таких, что $a + b + c + d = n$ и $a + k = b - k = c \cdot k = d \,/\, k$.

Примеры

Пример №1

Стандартный ввод
`45 2`

Стандартный вывод
`8 12 5 20`

Пример №2

Стандартный ввод
`128 7`

Стандартный вывод
`7 21 2 98`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, k;
    cin >> n >> k;
    int x = (k * n) / (k * k + 2 * k + 1);
    cout << x - k <<' '<< x + k <<' '<< x / k <<' '<< x * k << endl;
}
```

Задача 2.2.2.3. Ладья и обязательная клетка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Шахматная ладья находится в левом верхнем углу прямоугольного поля, разбитого на клетки размером $n \times m$. n обозначает число строк, m — число столбцов. Она хочет попасть в правую нижнюю клетку этого поля кратчайшим путем. Ладья может передвигаться либо вправо, либо вниз на любое количество клеток. Ладья обязана посетить заданную клетку с координатами (x, y), где x — номер строки этой клетки, а y — номер ее столбца.

Требуется найти количество способов построить путь ладьи из левого верхнего угла в правый нижний, которые проходят через обязательную клетку с заданными координатами.

Формат входных данных

В первой строке находятся два числа через пробел: n — число строк и m — число столбцов прямоугольного поля, $2 \leqslant n, m \leqslant 25$. Во второй строке через пробел находятся координаты (x, y) обязательной для посещения клетки, где $1 \leqslant x \leqslant n$, $1 \leqslant y \leqslant m$. Координаты x и y не совпадают с координатами левой верхней и правой нижней клеток.

Формат выходных данных

Вывести одно число — количество кратчайших путей ладьи из верхней левой в правую нижнюю клетку, проходящих через заданную клетку.

Примеры

Стандартный ввод
3 4 2 3

Стандартный вывод
6

Примечания

На рис. 2.2.3 представлены шесть путей, которыми ладья может пройти по полю размером 3×4, обязательно посещая по пути клетку $(2, 3)$.

Комментарий

Задачу можно решить как комбинаторными методами (произведение биномиальных коэффициентов), так и динамическим программированием.

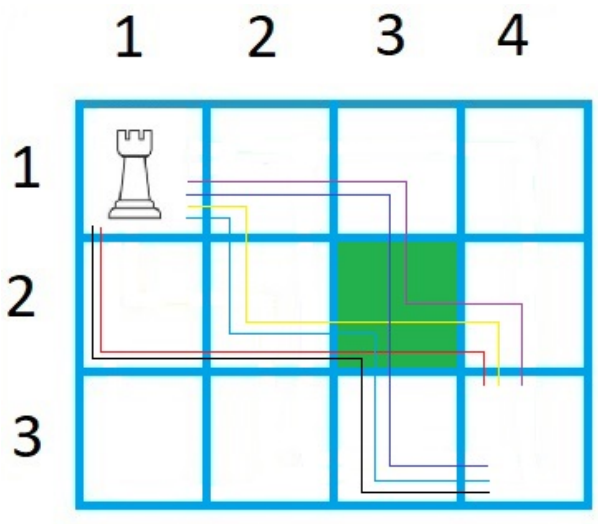


Рис. 2.2.3

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    vector<vector<int> > bc(51, vector<int>(51, 0));
    bc[0][0] = 1;
    for(int i = 1; i <= 50; i++){
        for(int j = 0; j < 51; j++){
```

```
            bc[i][j] += bc[i - 1][j];
            if(j - 1 >= 0){
                bc[i][j] += bc[i - 1][j - 1];
            }
        }
    }
    int n, m, x, y;
    cin >> n >> m >> x >> y;
    int d1 = bc[x - 1 + y - 1][x - 1];
    int d2 = bc[n - x + m - y][n - x];
    int ans = d1 * d2;
    cout << ans << endl;
}
```

Задача 2.2.2.4. Танец с цифрами (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Десять танцоров репетируют на сцене новый танец. Каждый танцор одет в футболку, на которой написана одна из цифр от 1 до 9, цифры могут повторяться. Изначально они стоят в некотором порядке слева направо, и их цифры образуют некоторое десятизначное число A. Далее во время всего танца участники либо разбиваются на пять пар рядом стоящих танцоров и одновременно меняются местами внутри своих пар, либо самый левый танцор перемещается на самую правую позицию и становится самым правым танцором.

Сын постановщика танца от скуки на бумаге выписывает все получающиеся при каждом перемещении десятизначные числа. Так как танец длинный, то в итоге на бумаге окажутся все возможные числа, которые в принципе могут появится при этих условиях. Нужно найти разницу между самым большим и самым маленьким из этих чисел.

Формат входных данных

На вход подается одно десятизначное число A, обозначающее начальное расположение танцоров. В числе могут встречаться цифры от 1 до 9, некоторые из них могут повторяться.

Формат выходных данных

Вывести одно число, равное разности самого большого и самого маленького из чисел, которые могут быть получены во время танца.

Примеры

Пример №1

Стандартный ввод
1456531355
Стандартный вывод
5182160085

Примечания

Самое маленькое число, которое можно получить в примере, равно 1353155456, самое большое равно 6535315541.

Покажем, как получить эти числа из исходного числа 1456531355. Сначала получим самое большое следующим образом: две левых цифры, 1 и 4, переместим вправо, получим 5653135514, потом поменяем в парах цифры местами и получим самое большое — 6535315541. Далее опять поменяем порядок в парах и в числе 5653135514 переместим три левых цифры 5, 6 и 5 вправо, получим 3135514565 и здесь снова поменяем порядок в парах, получим самое маленькое — 1353155456. Таким образом, искомая разница равна 5182160085.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    string s;
    cin >> s;
    string mx = s, mn = s;

    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 10; j++){
            mx = max(mx, s);
            mn = min(s, mn);
            if(j < 9){
                s = s.substr(1) + s[0];
            }
        }
        for(int j = 0; j < 5; j++){
            swap(s[2 * j], s[2 * j + 1]);
        }
    }
    stringstream ssmn;
    ssmn << mn;
    int imn;
    ssmn >> imn;
    stringstream ssmx;
```

```
26      ssmx << mx;
27      int imx;
28      ssmx >> imx;
29      cout << imx - imn << endl;
30  }
```

Задача 2.2.2.5. Трудная сортировка (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Иннокентий работает в отделе сортировки перестановок, подотделе сортировки вставками. Его задача заключается в сортировке перестановок, предоставленных заказчиками. Перестановкой длины n называется такая последовательность чисел, в которой встречаются все числа от 1 до n без повторений в некотором порядке.

Перестановка считается отсортированной, если в ней все числа расположены по возрастанию, то есть она имеет вид $1, \ldots, n$.

Иннокентий начинает рабочий день с пустой последовательности чисел. За день он сортирует вставками перестановку длины n. В начале каждой операции вставки он получает очередное число a_i из перестановки заказчика, после чего обрабатывает его, вставляя в отсортированную последовательность из ранее полученных чисел. После каждого такого добавления последовательность уже обработанных чисел должна быть отсортирована по возрастанию.

Перед тем как вставить число a_i в последовательность, он может выбрать, с какого края последовательности начать вставку. Далее он устанавливает число a_i с этого края и последовательно меняет вставляемое число с рядом стоящим числом b_j до тех пор, пока число a_i не встанет на свое место. На каждую перестановку вставляемого числа a_i с числом b_j Иннокентий тратит b_j единиц энергии.

Дана перестановка длины n из чисел a_i в том порядке, в котором Иннокентий их будет обрабатывать. Подскажите ему, какое минимальное количество энергии ему потребуется потратить, чтобы отсортировать всю перестановку.

Формат входных данных

В первой строке находится одно целое число n — длина перестановки, где $1 \leqslant \leqslant n \leqslant 2 \cdot 10^5$.

Во второй строке содержится n целых чисел a_i через пробел в том порядке, в котором они поступают на обработку Иннокентию. Гарантируется, что эти числа образуют перестановку длины n, то есть каждое число от 1 до n содержится в заданном наборе ровно один раз.

Формат выходных данных

Вывести одно число — минимальные суммарные энергозатраты Иннокентия для сортировки вставками заданной на входе перестановки.

Примеры

Пример №1

Стандартный ввод
9 2 9 1 5 6 4 3 8 7
Стандартный вывод
43

Примечания

Первым устанавливается число 2. Оно ни с чем не меняется местами, поэтому затрат нет.

Далее устанавливается число 9. Выбираем правый край и ставим его туда без потерь энергии.

Затем устанавливаем число 1. Выбираем левый край, ставим его туда и снова потерь нет.

Теперь нужно вставить число 5. Если его вставлять с правого края, придется менять местами с 9, а если с левого, то с 1 и 2, что суммарно явно лучше. Итого затраты на вставку 5 равны 3.

Число 6 снова лучше вставить слева, затраты на его вставку равны 8.

Число 4 вставим слева за 3.

Число 3 так же слева за 3.

А вот число 8 лучше вставить справа за 9.

И осталось число 7. Если вставлять слева, то затратим 21, а если справа, то всего 17.

Итого на сортировку заданной перестановки потратили: $0 + 0 + 0 + 3 + 8 + 3 + 3 + 9 + 17 = 43$.

Комментарий

Построим дерево отрезков на сумму, при обработке числа a будем находить, какая сумма на данный момент меньше: от 1 до $a - 1$ или от $a + 1$ до n. Прибавим ее к ответу и поместим в позицию a это число a.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int LG = 19;
int N = (1 << LG);
vector<int> tr(2 * N, 0);
void upd(int pos, int x){
    pos += N;
    tr[pos] = x;
    pos /= 2;
    while(pos){
        tr[pos] = {tr[2 * pos]+ tr[2 * pos + 1]};
        pos /= 2;
    }
}
int get(int l, int r){
    l += N;
    r += N;
    int res = 0;
    while(l <= r){
        if(l % 2 == 1){
            res += tr[l];
        }
        if(r % 2 == 0){
            res += tr[r];
        }
        l = (l + 1) / 2;
        r = (r - 1) / 2;
    }
    return res;
}
signed main(){
    int n, a;
    cin >> n;
    int ans = 0;
    for(int i = 0; i < n; i++){
        cin >> a;
        int sl = get(0, a - 1);
        int sr = get(a + 1, N - 1);
        ans += min(sl, sr);
        upd(a, a);
    }
    cout << ans << endl;
}
```

2.2.3. Третья волна. Задачи 8–11 класса

Задачи третьей волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63456/enter/.

Задача 2.2.3.1. Туннель (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим классическую задачу прохождения группы с одним фонариком по туннелю. Есть четыре человека, и у них есть один фонарик. Нужно перевести всю группу на другой конец туннеля. По туннелю можно проходить только с фонариком и только либо вдвоем, либо в одиночку. По этой причине придется сделать пять рейсов по туннелю: три рейса туда и два рейса обратно. Туда идут двое, обратно — один, возвращая фонарик еще не прошедшей части группы. У каждого из четырех человек своя скорость передвижения по туннелю, но некоторые скорости могут совпадать. Двое идут со скоростью самого медленного в этой паре. Нужно найти минимальное время, за которое можно перевести группу по туннелю.

Здесь, в зависимости от скоростей персонажей, есть две стратегии. Проиллюстрируем их на примерах.

Пусть есть люди A, B, C, D. У A — время прохождения туннеля 1 мин, у B — 4 мин, у C — 5 мин, у D — 10 мин. Здесь работает наиболее очевидная стратегия: самый быстрый переводит текущего и возвращается с фонариком обратно за следующим. При этой стратегии нужно проходить так:

- A, B туда, затрачено 4 мин;
- A обратно, затрачена 1 мин;
- A, C туда, затрачено 5 мин;
- A обратно, затрачена 1 мин;
- A, D туда, затрачено 10 мин.

Общее время $4 + 1 + 5 + 1 + 10 = 21$ мин.

Но не всегда эта стратегия оптимальна. Уменьшим время прохождения туннеля персонажем B до 2 мин. По вышеопределенной стратегии будет 19 мин ($2 + 1 + 5 + + 1 + 10 = 19$), но имеется более быстрое решение:

- A, B туда, затрачено 2 мин;
- A обратно, затрачена 1 мин;
- C, D туда, затрачено 10 мин;
- B обратно, затрачено 2 мин;
- A, B туда, затрачено 2 мин.

Общее время $2 + 1 + 10 + 2 + 2 = 17$ мин.

Заметим, что для предыдущего примера такая стратегия не работает: $4+1+10+ + 4 + 4 = 23$ мин.

Если же персонаж B проходит туннель за 3 мин (а все остальные так же, как и в примерах), то независимо от стратегии будет затрачено 20 мин. В этом случае

считаем, что работает первая стратегия.

Поразмыслив, станет понятно, от какого условия зависит выбор стратегии. Далее будем всегда считать, что A движется не медленнее B, B движется не медленнее C, C движется не медленнее D.

Дано время прохождения туннеля персонажами A, C, D. Нужно найти границу `border` для B такую, что если определить для B время прохождения строго меньшее, чем `border`, то выгодна вторая стратегия, иначе — первая.

Формат входных данных

В одной строке задано три целых чисел через пробел — время прохождения туннеля персонажами A, C, D. Времена даны по неубыванию. Все числа на входе в пределах от 1 до 100.

Формат выходных данных

Вывести одно число — границу `border` для B такую, что если определить время прохождения им туннеля строго меньше, чем `border`, нужно использовать вторую стратегию, иначе — первую. Ответ может быть нецелым, поэтому вывести его нужно с одним знаком после десятичной точки.

Примеры

Пример №1

Стандартный ввод
`1 5 10`
Стандартный вывод
`3`

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int A, C, D;
    cin >> A >> C >> D;
    cout.precision(1);
    cout << fixed << (A + C) / 2.0 << endl;
}
```

Задача 2.2.3.2. Математический пазл (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

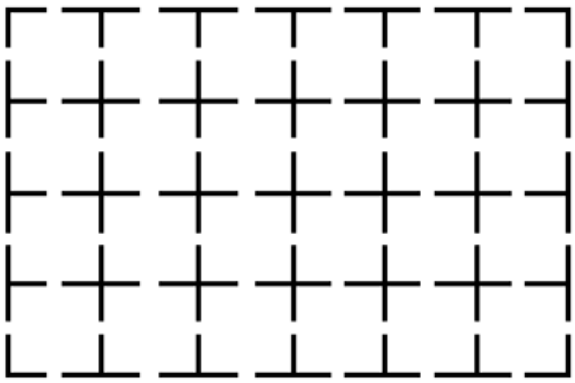


Рис. 2.2.4

Компания по производству пазлов решила освоить принципиально новый тип головоломок. Для этого берется прямоугольная решетка размера $n \times m$, каждый ее столбец и строка разрезаются посередине пополам. После этого образуются фигуры трех типов: четыре уголка, $2 \cdot (n+m-2)$ т-образных фигур и $(n-1) \cdot (m-1)$ крестиков.

Тому, кто решает головоломку, требуется сложить из этих фигур исходную прямоугольную решетку. При этом необходимо использовать абсолютно все имеющиеся в наличии фигуры.

Формат входных данных

В первой строке заданы через пробел два числа a — количество т-образных фигур и b — количество крестиков, которые находятся в одном из пазлов. При этом в наборе всегда есть еще четыре уголка. Известно, что этот комплект позволяет собрать прямоугольную решетку размера $n \times m$, где $1 \leqslant n,\ m \leqslant 10^9$.

Формат выходных данных

Требуется по числам a и b найти размеры исходной решетки n и m. Будем всегда считать, что $n \leqslant m$, то есть нужно вывести в одну строку через пробел два числа, первое из которых не превосходит второго, и вместе они задают размеры загаданной решетки.

Примеры

Пример №1

Стандартный ввод
16 15
Стандартный вывод
4 6

Пример №2

Стандартный ввод
0 0
Стандартный вывод
1 1

Комментарий

Задачу можно решить либо бинарным поиском, либо при помощи квадратного уравнения.

Решение

Ниже представлено решение на языке C++ при помощи бинпоиска.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b;
    cin >> a >> b;
    int L = 0, R = a / 4 + 1;
    while(R - L > 1){
        int M = (R + L) / 2;
        int D = a / 2 - M;
        if(M * D <= b){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout << L + 1 << ' ' << a / 2 - L + 1 << endl;
}
```

Задача 2.2.3.3. Восемь пирогов и одна свечка (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Мечта Карлсона наконец-то сбылась! Мама Малыша испекла восемь пирогов прямоугольной формы и в один из них воткнула свечку. После того как Карлсон съел семь пирогов, он решил-таки поделиться кусочком оставшегося восьмого пирога с Малышом. Но, будучи в хорошем настроении, он вынул из пирога свечу и предложил ему решить задачку.

«Так как я самый щедрый Карлсон в мире, то делить оставшийся пирог будешь ты. Но учти, ты должен разрезать пирог одним прямым разрезом так, чтобы линия прошла через один из углов и точку, где стояла свечка. После этого я выберу себе один из двух кусочков, а оставшийся, так и быть, достанется тебе».

Малыш не против этого замысла, однако считает, что разрезать пирог нужно как можно более справедливо, то есть так, чтобы разница между меньшим и большим кусками была как можно меньше. Подскажите Малышу, какой минимальной разницы между площадями кусков он сможет добиться.

Формат входных данных

В первой строке находятся два числа n и m через пробел — размеры прямоугольного пирога. Пирог размещен на координатной плоскости так, что его левый нижний угол находится в точке $(0, 0)$, а правый верхний — в точке (n, m), где $2 \leqslant n,\ m \leqslant 1\,000$.

Во второй строке находятся два числа x и y через пробел — координаты свечки, где $1 \leqslant x \leqslant n - 1$, $1 \leqslant y \leqslant m - 1$, то есть свечка находится строго внутри пирога.

Формат выходных данных

Вывести одно вещественное число с точностью не менее трех знаков после десятичной точки — минимальную разницу между площадями двух получающихся после разрезания кусков, которую сможет получить Малыш.

Примеры

Пример №1

Стандартный ввод
8 5 7 2
Стандартный вывод
12.571

Пример №2

Стандартный ввод
2 2 1 1
Стандартный вывод
0.000

Примечания

На рис. 2.2.5 представлены четыре варианта разделения пирога для первого примера из условия. Можно видеть, что самый близкий к справедливому способ разделения связан с разрезом из левого верхнего угла. Площадь треугольника в этом случае будет равна $96 / 7$, площадь четырехугольника равна $184 / 7$, и разница равна $88 / 7$, что при округлении до трех знаков равно 12,571.

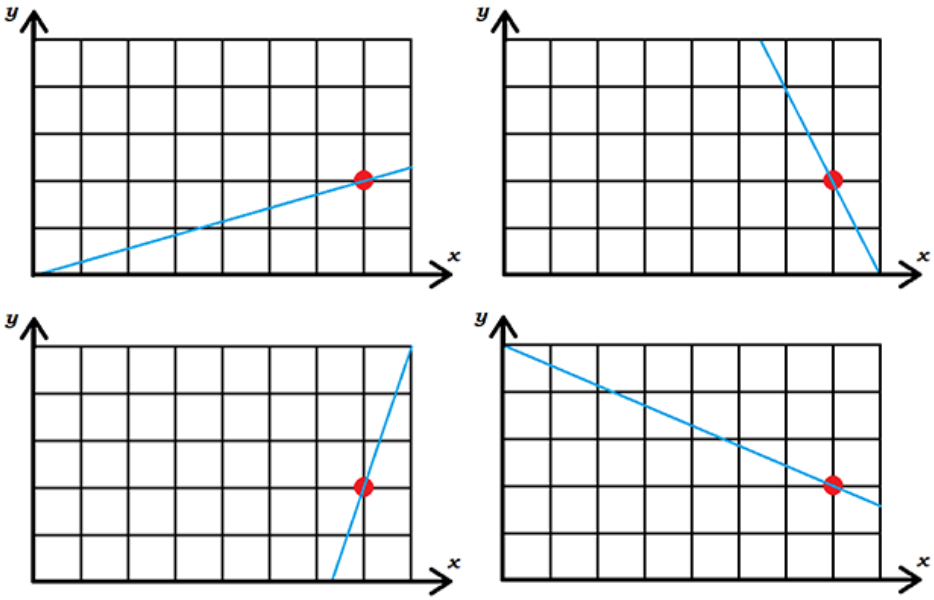


Рис. 2.2.5

Комментарий

Геометрия: для каждого из четырех случаев аккуратно находим катеты прямоугольного треугольника при помощи пропорции, затем находим площадь этого треугольника и, вычитая из всего прямоугольника эту площадь, находим площадь второго куска. Далее выбираем наиболее оптимальное отношение площадей.

Решение

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
double katy(double x, double y, double n){
    return n * y / x;
}
double n, m, x, y;
double ans = INF;
double k1, k2;
void upd(){
    if(k1 < m){
        double st =k1 * n / 2;
        ans = min(ans, n * m - 2 * st);
    }
    else{
        double st =k2 * m / 2;
        ans = min(ans, n * m - 2 * st);
    }}
signed main(){
    cin >> n >> m >> x >> y;
    k1 = katy(x, y, n);
    k2 = katy(y, x, m);
    upd();
    k1 = katy(n - x, y, n);
    k2 = katy(y, n - x, m);
    upd();
    k1 = katy(x, m - y, n);
    k2 = katy(m - y, x, m);
    upd();
    k1 = katy(n - x, m - y, n);
    k2 = katy(m - y, n - x, m);
    upd();
    cout.precision(3);
    cout << fixed << ans<< endl;
}
```

Задача 2.2.3.4. Плетенка (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

У Маши есть n полосок бумаги. i-я полоска имеет ширину 1 и длину a_i. Маша разделит эти полоски на две части и покрасит некоторые в желтый, а оставшиеся — в зеленый цвет. Она сама выберет, какие полоски как покрасить. Далее она хочет из этих полосок сплести максимально большую плетенку. Она расположит полоски одного цвета в некотором порядке горизонтально, а полоски другого цвета в некотором порядке вертикально. После этого она переплетет горизонтальные и вертикальные полоски так, что они будут чередоваться то сверху, то снизу, образуя в местах пересечения шахматную раскраску. Наконец, она обрежет выступающие края полосок так, что останется прямоугольная плетенка с ровными краями. Каждая клетка полученной плетенки должна иметь два слоя.

Маша хочет сплести максимально большую по площади прямоугольную плетенку. Подскажите ей, плетенку какой площади она сможет сделать. Заметим, что она может при создании плетенки использовать не все имеющиеся у нее полоски.

Формат входных данных

В первой строке на вход подается число n — количество полосок бумаги у Маши, где $2 \leqslant n \leqslant 2 \cdot 10^5$. Во второй строке через пробел заданы n целых чисел a_i через пробел — длины полосок, где $1 \leqslant a_i \leqslant 10^9$.

Формат выходных данных

Вывести одно число — площадь прямоугольника, форму которого может иметь самая большая плетенка Маши.

Примеры

Пример №1

Стандартный ввод
8 3 6 5 4 4 5 5 2

Стандартный вывод
12

Примечания

На рис. 2.2.6 представлен один из вариантов получения самой большой плетенки для полосок из примера. Синим обозначена граница полученной максимальной плетенки. Ее размер 3×4, и ее площадь 12. При ее создании Маша не должна использовать полоску номер 8, по этой причине неважно, как она раскрашена.

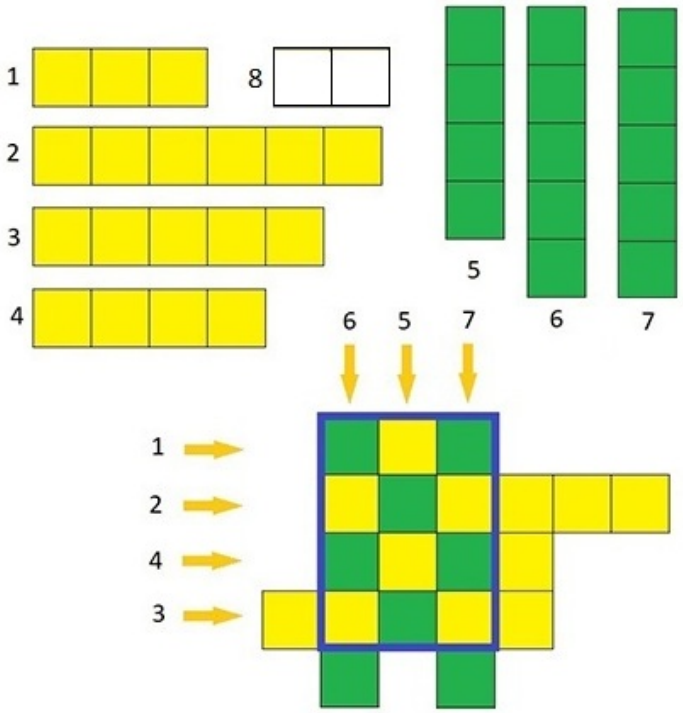


Рис. 2.2.6

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n;
    cin >> n;
    deque<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    sort(v.begin(), v.end());
    int ans = 0;
    int cnth = 0, minh;
    while(1){
        if(v.size() == 0){
            break;
        }
        cnth++;
        minh = v.back();
        v.pop_back();
        while(v.size() > 0 && v[0] < cnth){
            v.pop_front();
        }
        ans = max(ans, cnth * min(minh, (int)v.size()));
    }
    cout << ans << endl;
}
```

Задача 2.2.3.5. Английский в игровой форме (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 3 с.

Ограничение по памяти: 64 Мбайт.

Условие

Маша и Витя запоминают слова английского языка в оригинальной игровой форме. За день им нужно выучить n слов, где $20 \leqslant n \leqslant 100$, каждое из которых имеет длину от 5 до 8 символов. Маша выбирает из этого набора наугад несколько попарно различных слов (также от 5 до 8) и собирает их в одну строку без пробелов. Далее она переставляет буквы в этой строке так, что слова оказываются полностью перепутанными, и дает эту строку Вите. Теперь Витя должен восстановить все слова, которые выбрала Маша.

Но у Вити плохо получается, а Маша уже забыла, какие слова она выбрала. Нужно им помочь — написать программу, которая восстановит слова, выбранные Машей.

Формат входных данных

В первой строке находится строка, которую Маша предложила Вите. Во второй строке содержится число n — количество слов, которые нужно выучить детям, $20 \leqslant$ $\leqslant n \leqslant 100$.

В следующих n строках содержатся эти слова по одному в строке. Все слова в этом наборе различны. Слова отсортированы в лексикографическом (алфавитном) порядке. Все слова состоят из маленьких букв от a до z. Обратите внимание, что в тестах к этой задаче все заданные слова реально существуют в английском языке и случайным образом выбраны из словаря.

Гарантируется, что длина каждого слова из предложенного набора (словаря) в пределах от 5 до 8, строка, которую получила Маша, может быть получена путем перестановки букв некоторых различных слов из предложенного словаря, причем, набор выбранных Машей слов определяется по ней однозначно. Количество слов, из которых составлена Машина строка, находится в пределах от 5 до 8.

Формат выходных данных

Вывести все слова, выбранные Машей, в алфавитном порядке по одному в строке.

Примеры

Пример №1

Стандартный ввод
stirbaexsudueoeidgomttcrnrwlunapntetacwri 24 bridge cranky document drawing farmer fighter figurine gravy havoc minimum reactant reply republic sonata soprano split subset tailor texture tomorrow trout vicinity wrist writer

Стандартный вывод
document drawing republic sonata texture wrist

Комментарий

В случае, выделенном в условии (слова являются случайными, взятыми из английского словаря), задача решается рекурсией с перебором вариантов.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
string frs;
int n;
vector<string> dict;
vector<int> msk(26, 0);
int cnt = 0;
vector<vector<int> > amsk;
vector<string> ans;
bool bigok = 0;
void p(int pos){
   if(!bigok){
        if(cnt == 0){
            sort(ans.begin(), ans.end());
            bigok = 1;
            return;
        }
        for(int i = pos; i < n; i++){
            string ts = dict[i];
            bool ok = 1;
            for(int j = 0; j < 26; j++){
                if(amsk[i][j] > msk[j]){
                    ok = 0;
                }
            }
            if(ok){
                ans.push_back(ts);
                for(int j = 0; j < 26; j++){
                    msk[j] -= amsk[i][j];
                    cnt -= amsk[i][j];
                }
                p(i + 1);
                if(!bigok){
                for(int j = 0; j < 26; j++){
                    msk[j] += amsk[i][j];
                    cnt += amsk[i][j];
                }
                ans.pop_back();
                }
            }
        }
   }
}
signed main(){
    cin >> frs;
    cin >> n;
    amsk.resize(n, vector<int>(26, 0));

    string ts;
    for(int i = 0; i < n; i++){
        cin >> ts;
        dict.push_back(ts);
    }
    for(int i = 0; i < n; i++){
        for(auto el : dict[i]){
            amsk[i][el - 'a']++;
        }
    }
```

```
60      for(auto el : frs){
61          msk[el - 'a']++;
62          cnt++;
63      }
64      p(0);
65      for(auto el : ans){
66          cout << el << endl;
67      }
68  }
```

2.2.4. Четвертая волна. Задачи 8–11 класса

Задачи четвертой волны предметного тура по информатике открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63457/enter/.

Задача 2.2.4.1. Квадратный флаг (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Одному портному заказали сделать одноцветный флаг. Особенность этого флага в том, что он должен быть квадратным. У портного есть два прямоугольных куска ткани заданного цвета. Один из них имеет размеры $a \times b$, другой — $c \times d$. Так как клиент будет платить пропорционально площади изготовленного флага, портной хочет сначала сшить имеющиеся у него прямоугольные куски, соединив их двумя какими-то сторонами, а затем из полученного полотна вырезать и сделать флаг с максимально большой стороной. Определить сторону получившегося у него флага.

Формат входных данных

На вход подаются две строки. В первой строке находятся размеры первого прямоугольника — целые числа a, b через пробел, во второй — размеры второго прямоугольника, также целые числа c, d через пробел, где $1 \leqslant a, b, c, d \leqslant 10^9$.

Формат выходных данных

Вывести одно число — сторону самого большого квадрата, который можно получить по условию задачи.

Примеры

Пример №1

Стандартный ввод
2 4 3 6

Стандартный вывод
4

Пример №2

Стандартный ввод
2 2 3 6

Стандартный вывод
3

Примечания

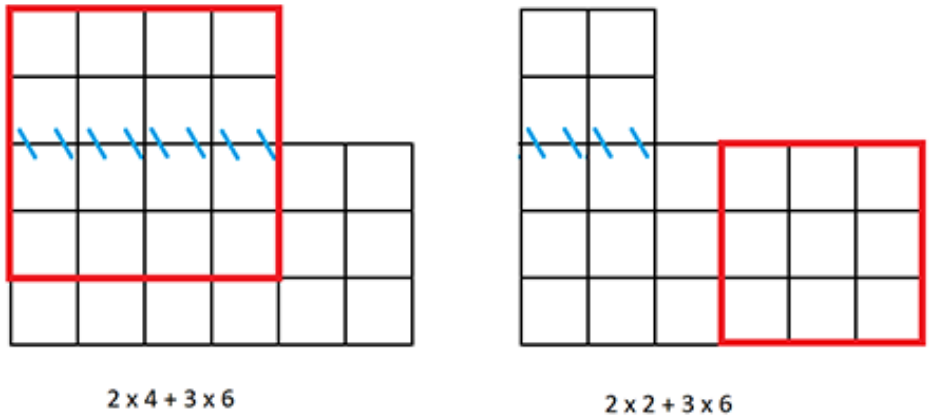


Рис. 2.2.7

На рис. 2.2.7 представлены иллюстрации для тестов из условия. Синими штрихами обозначено место сшивки двух кусков. Красный квадрат выделяет один из вариантов вырезания максимального квадрата.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    int ans = max(min(a, b), min(c, d));
    int p1 = min(a + c, min(b, d));
    int p2 = min(a + d, min(b, c));
    int p3 = min(b + c, min(a, d));
    int p4 = min(b + d, min(a, c));
    ans = max({ans, p1, p2, p3, p4});
    cout << ans << endl;
}
```

Задача 2.2.4.2. Потерянная ДНК (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

В данной задаче будем упрощенно считать, что ДНК представляется строкой длины от 10 до 100, состоящей из букв A, C, G, T.

Пусть даны две ДНК D_1 и D_2 одной и той же длины n. Выберем некоторое произвольное число i от 1 до $n-1$ и поменяем местами префиксы (начала) этих ДНК длины i. Будем говорить, что полученные новые две строки образованы путем скрещивания двух исходных по префиксу длины i.

Например, пусть $D_1 =$ **AACGGTAGGT**, а $D_2 =$ TCCCGGAACA. Выберем $i = 4$ и поменяем местами префиксы длины 4. Получим две новые ДНК, одна из которых будет иметь вид **AACG**GGAACA, а вторая — TCCC**GTAGGT**. Для наглядности были выделены части первой из них.

Полученные новые ДНК снова могут быть скрещены по любому префиксу длины от 1 до $n-1$.

Теперь можно рассмотреть популяцию из нескольких ДНК. Выберем из них две, произведем их скрещивание по префиксу какой-либо длины и поместим две новые ДНК в исходную популяцию. В данной задаче будем считать, что количество ДНК не увеличивается, то есть старые две ДНК заменяются на новые две ДНК.

Дана исходная популяция из m ДНК, каждая имеет одну и ту же длину n. После некоторого количества попарных скрещиваний была получена новая популяция. Но при итоговой обработке данных сведения об одной ДНК из новой популяции были потеряны. Задача состоит в отыскании этой потерянной ДНК по оставшимся $m-1$ ДНК из новой популяции.

Формат входных данных

В первой строке через пробел даны два числа n — длина ДНК и m — количество ДНК в исходной популяции, где $10 \leqslant n \leqslant 100$, $2 \leqslant m \leqslant 100$.

В следующих m строках содержится описание исходной популяции ДНК, каждая задается строкой длины n, состоящей из символов A, C, G и T.

Далее следует разделяющая строка, содержащая n символов «$-$».

Далее следует еще $m - 1$ строк, описывающих новую (заключительную) популяцию без одной ДНК.

Гарантируется, что данные верны, то есть $m - 1$ последняя ДНК является некоторой новой популяцией ровно без одной ДНК, полученной из исходной популяции, заданной в m первых строках.

Формат выходных данных

Вывести недостающую утерянную ДНК.

Примеры

Пример №1

Стандартный ввод
10 2 AACGGTAGGT TCCCGGAACA ---------- TCCCGTAGGT

Стандартный вывод
AACGGGAACA

Пример №2

Стандартный ввод
10 4 AACCGGTTAA ACGTACGTAC AAACCCGGGT CATTACTGGA ---------- AAGCGCTTAA CCACACGTGC AACTAGGGGT

Стандартный вывод
AATTCCTGAA

Комментарий

Для каждой позиции нужно найти недостающую букву из первого набора ДНК. Для этого удобнее всего использовать функцию `xor`.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
signed main(){
    int n, m;
    cin >> n >> m;
    vector<string> v1(m);
    for(int i = 0; i < m; i++){
        cin >> v1[i];
    }
    string d;
    cin >> d;
    vector<string> v2(m - 1);
    for(int i = 0; i < m - 1; i++){
        cin >> v2[i];
    }
    for(int j = 0; j < n; j++){
        int ss = 0;
        for(int i = 0; i < m; i++){
            ss ^= (int)(v1[i][j]);
        }
        for(int i = 0; i < m - 1; i++){
            ss ^= (int)(v2[i][j]);
        }
        cout << (char)(ss);
    }
    cout << endl;
}
```

Задача 2.2.4.3. Утомленные туристы (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Рассмотрим следующий вариант известной задачи на перемещение по туннелю группы из четырех человек. В общем виде она выглядит так: четыре туриста хотят пройти по темному туннелю. Имеется один фонарик. По туннелю можно перемещаться либо вдвоем, либо по одному, при этом у тех, кто движется в туннеле,

должен быть фонарик в руках. По этой причине движение должно быть следующим: двое переходят туда, один возвращается обратно и приносит фонарик тем, кто еще не перешел. После этого указанный маневр повторяется снова.

У каждого участника своя скорость движения в туннеле. Пусть участники проходят туннель за A, B, C и D мин. Если идут двое, то они движутся со скоростью того, кто идет медленнее. Требуется по заданным временам прохождения туннеля каждого из участников перевести их максимально быстро через туннель.

Немного усложним данную задачу. Введем фактор усталости. А именно, любой участник, пройдя по туннелю, устает и в следующий раз идет уже медленнее. После каждого прохождения туннеля время прохождения любого участника увеличивается на E мин. Например, если участник до начала движения проходит туннель за 1 мин, а показатель усталости E равен 3 мин, то первый раз участник пройдет туннель за 1 мин, второй раз — за 4 мин, третий раз — за 7 мин и т. д.

По заданным A, B, C, D и E узнать, за какое минимальное время можно провести всю группу через туннель согласно указанным правилам.

Формат входных данных

На вход подаются пять чисел. В первой строке через пробел четыре числа A, B, C и D — время прохождения туннеля каждым из четырех участников до того, как они начали движение. Во второй строке содержится число E — величина, на которую увеличивается время прохождения туннеля каждым участником после каждого перемещения. При этом $1 \leqslant A, B, C, D \leqslant 1\,000$, $0 \leqslant E \leqslant 1\,000$.

Формат выходных данных

Вывести одно число — минимальное время прохождения туннеля всей группой.

Примеры

Пример №1

Стандартный ввод
8 9 10 1 3
Стандартный вывод
44

Пример №2

Стандартный ввод
8 9 10 1 0
Стандартный вывод
29

Примечания

В первом примере при прохождении туннеля каждый турист устает и движется медленнее на 3 мин. Покажем, как перевести группу при этом за 44 мин.

Каждую ситуацию будем обозначать следующим образом: слева от двоеточия находятся туристы, которые стоят в начале туннеля, а справа — те, что стоят в конце туннеля. Туриста будем обозначать при помощи числа, соответствующего его текущему времени прохождения туннеля.

Тогда исходная ситуация имеет вид 1, 8, 9, 10 :.

Сначала идут туристы 1 и 8, каждый после перехода устает на 3 мин, получим ситуацию 9, 10 : 4, 11, затрачено 8 мин.

Обратно возвращается турист 4, он устает еще на 3 мин. Ситуация становится 7, 9, 10 : 11, затрачено $8 + 4 = 12$ мин.

Теперь идут туристы 7 и 9, получится ситуация 10 : 10, 11, 12, затрачено $8 + 4 + 9 = 21$ мин.

Возвращается турист 10, получится 10, 13 : 11, 12, затрачено $8 + 4 + 9 + 10 = 31$ мин.

Наконец, оставшиеся двое туристов 10 и 13 за 13 мин переходят туннель, итого затрачено $8 + 4 + 9 + 10 + 13 = 44$ мин.

Комментарий

Задача решается рекурсивным перебором всех вариантов прохождения.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
vector<int> v(4);
int e, ans = INF;
void p(vector<int> &vl, vector<int> &vr, int tv){
    if(vl.size() == 2){
        ans = min(ans, tv + *max_element(vl.begin(), vl.end()));
        return;
    }
    for(int i = 0; i < vl.size() - 1; i++){
        for(int j = i + 1; j < vl.size(); j++){
            vector<int> vl1;
            for(int k = 0; k < vl.size(); k++){
                if(k != i && k != j){
                    vl1.push_back(vl[k]);
                }
            }
            vector<int> vr1 = vr;
```

```
21              vr1.push_back(vl[i] + e);
22              vr1.push_back(vl[j] + e);
23              int tmp = max(vl[i], vl[j]);
24              sort(vr1.rbegin(), vr1.rend());
25              vl1.push_back(vr1.back() + e);
26              vr1.pop_back();
27              p(vl1, vr1, tv + tmp + vl1.back() - e);
28          }
29      }
30  }
31  signed main(){
32      for(int i = 0; i < 4; i++){
33          cin >> v[i];
34      }
35      sort(v.begin(), v.end());
36      cin >> e;
37      vector<int> vl = v, vr;
38      p(vl, vr, 0);
39      cout << ans;
40  }
```

Задача 2.2.4.4. Проектируем мост (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

При постройке моста используются два типа пролетов: П-образные (они прочные, но дорогие) и Т-образные (они дешевле, но менее надежные). Мост должен начинаться и заканчиваться П-образными пролетами. Любой Т-образный пролет должен иметь хотя бы один П-образный пролет в качестве соседнего.

Длина проектируемого моста — n пролетов. Муниципалитет выделил средства на постройку a П-образных и b Т-образных пролетов. При этом $a + b = n$. Требуется выяснить, сколькими способами при этих условиях можно скомпоновать мост. Два способа компоновки моста отличаются, если в одной на некоторой позиции стоит П-образный пролет, а в другой на этой же позиции стоит Т-образный пролет.

Формат входных данных

В одной строке через пробел заданы два числа: a — число П-образных пролетов и b — число Т-образных пролетов, на постройку которых выделены средства, где $2 \leqslant a \leqslant 10^6$, $0 \leqslant b \leqslant 10^6$.

Формат выходных данных

Вывести одно число — количество вариантов компоновки моста. Так как ответ может быть очень большим, требуется вывести остаток от его деления на 1 000 000 007 ($10^9 + 7$).

Примеры

Пример №1

Стандартный ввод
4 3

Стандартный вывод
7

Примечания

Для примера из условия имеется 7 вариантов компоновки моста (пробелы добавлены для лучшего восприятия вариантов):

П Т Т П Т П П

П Т Т П П Т П

П Т П Т Т П П

П Т П П Т Т П

П П Т П Т Т П

П П Т Т П Т П

П Т П Т П Т П

Комментарий

При заданных ограничениях задача решается только при помощи комбинаторики с вычислениями по модулю.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int INF = 1e18;
const int MOD = 1e9 + 7;
vector<int> f(2e6 + 1, 1);
```

```
int binpow (int a, int n) {
        int res = 1;
        while (n > 0) {
                if (n % 2 == 1)
                        (res *= a) %= MOD;
                (a *= a) %= MOD;
                n /= 2;
        }
        return res;
}

int bc(int n, int k){
    int res = f[n];
    int p1 = binpow(f[k], MOD - 2);
    int p2 = binpow(f[n - k], MOD - 2);
    (res *= p1) %= MOD;
    (res *= p2) %= MOD;
    return res;
}
signed main(){
    for(int i = 1; i <= 2e6; i++){
        f[i] = (f[i - 1] * i) % MOD;
    }
    int a, b;
    int ans = 0;
    cin >> a >> b;
    a--;
    for(int i = 0; i < a + 1; i++){
        if(2 * i <= b){
            int d = bc(a, i);
            if(b - 2 * i <= a - i){
                (d *= bc(a - i, b - 2 * i) ) %= MOD;
                (ans += d) %= MOD;
            }
        }
    }
    cout << ans << endl;
}
```

Задача 2.2.4.5. Джентльмены на прогулке (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 8 с.

Ограничение по памяти: 64 Мбайт.

Условие

По прямому участку улицы, которую будем считать отрезком AB длины d, прогуливаются n джентльменов. i-й джентльмен движется со скоростью v_i. Скорости всех джентльменов попарно различны. Дойдя до любого конца улицы, каждый джентльмен поворачивает и идет в обратную сторону.

При каждой встрече два джентльмена приветствуют друг друга, приподнимая

головной убор. Приветствие происходит и в том случае, когда один джентльмен обгоняет другого. Если два джентльмена встречаются в момент их одновременного поворота, то происходит два приветствия: одно до поворота, другое — после поворота. Если происходит одновременная встреча трех и более джентльменов, то они приветствуют друг друга попарно, то есть каждый каждого. Допустим, если одновременно встретились четыре джентльмена где-то посреди улицы, произойдет шесть попарных приветствий. Если же эти четыре джентльмена встретились в момент их одновременного поворота, произойдет уже двенадцать приветствий.

В этой задаче считаем, что все действия происходят без остановок, то есть и повороты и приветствия происходят мгновенно. Джентльмены одновременно начинают свою прогулку из точки A в момент 0. В этот момент они уже производят свои первые попарные приветствия, то есть в момент 0 уже произведено $n \cdot (n-1)/2$ приветствий. Момент старта не считается моментом поворота, то есть на старте число приветствий не удваивается. Джентльмены гуляют достаточно долго, чтобы произошло любое заданное количество приветствий.

Требуется найти момент, в который было произведено k-е по порядку приветствие.

Формат входных данных

В первой строке ввода через пробел содержится два целых числа: d — длина отрезка AB и n — количество прогуливающихся джентльменов, где $1 \leqslant d \leqslant 200$, $2 \leqslant n \leqslant 2\,000$.

Во второй строке находятся n целых чисел v_i через пробел — скорости каждого джентльмена, где $1 \leqslant v_i \leqslant 2\,000$. Гарантируется, что все скорости попарно различны. Скорости даны в порядке возрастания, то есть $v_1 < v_2 < \ldots < v_n$.

В третьей строке содержится одно целое число k — номер требуемого приветствия, для которого нужно найти момент, когда оно произойдет, где $1 \leqslant k \leqslant 10^9$.

Формат выходных данных

Вывести одно вещественное число — время, когда произойдет k-е по порядку приветствие. Ответ вывести с точностью не менее двух знаков после десятичной точки.

Примеры

Пример №1

Стандартный ввод
5 4 2 5 8 10 6

Стандартный вывод
0.000

Пример №2

Стандартный ввод
5 4 2 5 8 10 7
Стандартный вывод
0.556

Пример №3

Стандартный ввод
5 4 2 5 8 10 11
Стандартный вывод
1.000

Пример №4

Стандартный ввод
5 4 2 5 8 10 15
Стандартный вывод
1.429

Пример №5

Стандартный ввод
5 4 2 5 8 10 17
Стандартный вывод
1.667

Пример №6

Стандартный ввод
5 4 2 5 8 10 19
Стандартный вывод
1.667

Пример №7

Стандартный ввод
5 4 2 5 8 10 21
Стандартный вывод
2.000

Примечания

На рис. 2.2.8 приведено положение джентльменов из примеров в моменты времени 0, 1 и 2. Джентльмены обозначены своими скоростями. Стрелками обозначены направления их движения в соответствующий момент. Перечислим и пронумеруем в порядке возрастания моменты попарных приветствий этих джентльменов до момента времени 2 включительно. Если два и более приветствия происходят одновременно, неважно какое из них конкретно имеет номер k, главное, что они происходят в один и тот же определенный момент времени.

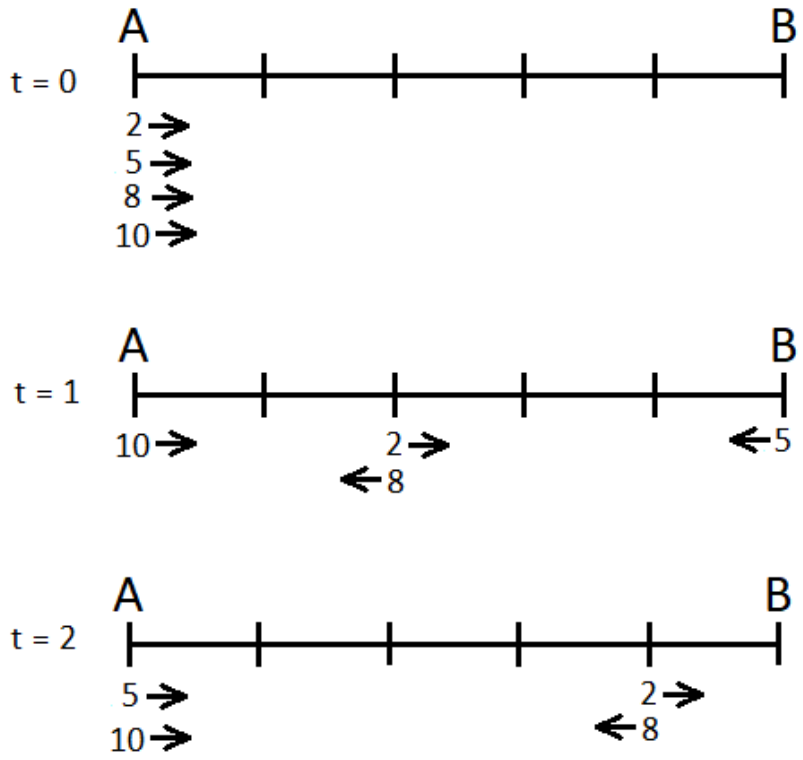


Рис. 2.2.8

1. 2 и 5 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
2. 2 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
3. 2 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
4. 5 и 8 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
5. 5 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).

6. 8 и 10 приветствуют друг друга в момент 0 (изображено на рис. 2.2.8).
7. 8 и 10 приветствуют друг друга в момент 0.556.
8. 5 и 10 приветствуют друг друга в момент 0.667.
9. 5 и 8 приветствуют друг друга в момент 0.769.
10. 2 и 10 приветствуют друг друга в момент 0.833.
11. 2 и 8 приветствуют друг друга в момент 1.000 (изображено на рис. 2.2.8).
12. 8 и 10 приветствуют друг друга в момент 1.111.
13. 2 и 10 приветствуют друг друга в момент 1.250.
14. 5 и 10 приветствуют друг друга в момент 1.333.
15. 2 и 5 приветствуют друг друга в момент 1.429.
16. 5 и 8 приветствуют друг друга в момент 1.538.
17. 2 и 8 приветствуют друг друга в момент 1.667.
18. 2 и 10 приветствуют друг друга в момент 1.667.
19. 8 и 10 приветствуют друг друга в момент 1.667 (в момент 1.667 встретятся одновременно три джентльмена 2, 8 и 10).
20. 2 и 8 приветствуют друг друга в момент 2.000 (изображено на рис. 2.2.8).
21. 5 и 10 приветствуют друг друга в момент 2.000 (до поворота).
22. 5 и 10 приветствуют друг друга в момент 2.000 (после поворота, изображено на рис. 2.2.8).

Комментарий

Задача решается при помощи бинпоиска с квадратичным нахождением ответа в каждой его итерации.

Решение

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const double EPS = 1e-7;
double x(double M, int V, int d){
    double dst = V * M;
    int cnt = floor((dst + EPS) / d);
    double pin = dst - cnt * d;
    if(cnt % 2 == 0){
        return pin;
    }
    else{
        return d - pin;
    }
}
int F(double M, vector<int> &v, int d){
    int res = 0;
    for(int i = 0; i < v.size(); i++){
        double dst = v[i] * M;
```

```
            int cnt = floor((dst + EPS) / d);
            res += cnt * i;
            double tx = x(M, v[i], d);
            for(int j = 0; j < i; j++){
                double txj = x(M, v[j], d);
                if(cnt % 2 == 0){
                    res += txj <= tx + EPS;
                }
                else{
                    res += txj >= tx - EPS;
                }
            }
        }
        return res;
}
signed main(){
    int d, n;
    cin >> d >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    int k;
    cin >> k;
    double L = 0, R = 1;
    while(F(R, v, d) <= k){
        R *= 2;
    }
    R *= 2;
    while(R - L > 1e-4){
        double M = (R + L) / 2.0;
        if(F(M, v, d) < k){
            L = M;
        }
        else{
            R = M;
        }
    }
    cout.precision(10);
    cout << fixed << L << endl;
}
```

2.3. Предметный тур. Физика

2.3.1. Первая волна. Задачи 8–9 класса

Задачи первой волны предметного тура по по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63463/enter/.

Задача 2.3.1.1. Калориметр (10 баллов)

Условие

Внутренний стакан калориметра представляет собой цилиндр с радиусом $R = 8$ см и высотой $3R$. Внешний стакан также имеет форму цилиндра, стенки которого (как боковые, так и торцы) отстоят от стенок внутреннего на расстояние R. Какая масса теплоизоляционного материала с плотностью $\rho = 25$ кг/м^3 необходима, чтобы полностью заполнить пространство между стаканами?

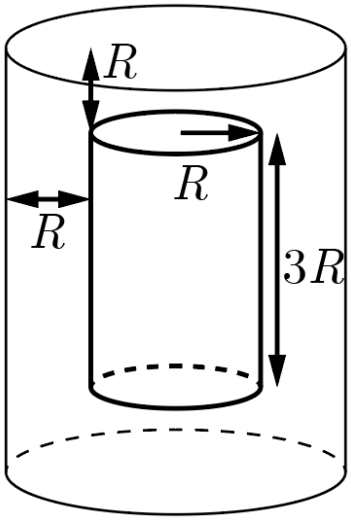


Рис. 2.3.1

Решение

Чтобы найти массу m теплоизоляционного материала, необходимо его плотность умножить на занимаемый им объем V:

$$m = \rho V. \tag{2.3.1}$$

Объем области, заполняемой теплоизоляцией, удобнее всего найти, вычтя из объема V_1 большого внешнего стакана объем V_2 маленького внутреннего. Для любого кругового цилиндра с высотой h и радиусом r объем может быть найден по

формуле $V = \pi r^2 h$. В случае большого цилиндра $r = 2R$ и $h = 5R$, следовательно,

$$V_1 = 5R \cdot \pi(2R)^2 = 20\pi R^3. \tag{2.3.2}$$

Аналогично, для маленького $r = R$, $h = 3R$ и, следовательно,

$$V_2 = 3\pi R^3. \tag{2.3.3}$$

Подставляя (2.3.2), (2.3.3) в (2.3.1), получим, что искомая масса составляет:

$$m = \rho(V_1 - V_2) = 17\pi R^3 \rho \approx 0{,}68\,\text{кг}. \tag{2.3.4}$$

Погрешность 0,01 кг.

Ответ: $m = 17\pi R^3 \rho = (0{,}68 \pm 0{,}01)$ кг.

Задача 2.3.1.2. Нить накала (15 баллов)

Условие

Нити накала ламп изготавливают из вольфрама, удельное сопротивление которого сильно зависит от температуры. По мере прогрева нити оно возрастает от $\rho_0 = 5{,}5 \cdot 10^{-8}$ Ом · м до $\rho_1 = 1{,}1 \cdot 10^{-6}$ Ом · м. Определите электрическую мощность, потребляемую лампой в первый момент после ее включения, если в рабочем режиме (полностью прогревшись) лампа потребляет от той же сети мощность $P_1 = 30$ Вт.

Решение

Электрическая мощность P, потребляемая нитью накала, может быть вычислена по закону Джоуля – Ленца, который для фиксированного напряжения U в сети удобно записать как

$$P = U^2/R. \tag{2.3.5}$$

При этом сопротивление R нити легко выразить через ее удельное сопротивление ρ, длину l и площадь поперечного сечения S

$$R = \frac{\rho l}{S}. \tag{2.3.6}$$

Подставляя (2.3.6) в (2.3.5), получим

$$P = \frac{U^2 S}{\rho l},$$

где только ρ зависит от температуры. В результате придем к выводу, что выделяющаяся в лампе мощность обратно пропорциональна ее удельному сопротивлению, откуда окончательно следует

$$P_0 = P_1 \frac{\rho_1}{\rho_0} \approx 600\,\text{Вт}.$$

Погрешность 1 Вт.

Ответ: $P_0 = (600 \pm 1)$ Вт.

Задача 2.3.1.3. Свая (20 баллов)

Условие

Бетонная свая высотой $h = 1{,}4$ м и массой $m = 160$ кг полностью погружена в грунт так, что ее верхний торец совпадает с уровнем почвы. К сожалению, сваю понадобилось извлечь. Определите, какую работу для этого необходимо совершить, если сила трения со стороны грунта, действующая на сваю, прямо пропорциональна площади соприкосновения ее боковой стороны с землей и в начальный момент ее извлечения равна $F = 4$ кН. Ускорение свободного падения $g \approx 9{,}8$ м/с^2.

Решение

Работа A, необходимая для извлечения сваи, складывается из увеличения потенциальной энергии сваи на величину mgh и работы по преодолению силы трения $A_{\text{тр}}$. Последняя должна быть найдена с учетом постепенного уменьшения силы трения $F_{\text{тр}}$ от максимального значения F до нуля. Поскольку это уменьшение происходит линейно, общая работа оказывается строго вдвое меньше, чем при постоянном значении $F_{\text{тр}} = F$ (аналогично тому, как вычисляется значение работы сил упругости пружины). В результате

$$A = mgh + \frac{Fh}{2} \approx 5\,\text{кДж}. \tag{2.3.7}$$

Погрешность $0{,}1$ кДж.

Ответ: $(5{,}0 \pm 0{,}1)$ кДж.

Задача 2.3.1.4. Двое из ларца (25 баллов)

Условие

Два дрона одновременно вылетают с общей пусковой станции и движутся по прямолинейным траекториям. Первый дрон на начальном этапе движения перемещается с постоянной скоростью $v_1 = 15$ м/с, а через время $\tau = 40$ с быстро переключается на движение с постоянной скоростью $v_2 = 20$ м/с. Второй дрон — наоборот, сначала движется со скоростью v_2, а через время τ переключается на скорость v_1. Наконец, дроны одновременно заканчивают полет. Определите, как долго длился этот полет, если по его итогам средняя путевая скорость первого дрона оказалась на $\Delta v = 1$ м/с выше, чем средняя путевая скорость второго.

Решение

По определению средняя путевая скорость — это отношение общего пройденного пути S к общему времени движения t:

$$v = \frac{S}{t}. \tag{2.3.8}$$

Для первого дрона это уравнение принимает вид

$$v_a = \frac{v_1\tau + v_2(t-\tau)}{t}, \tag{2.3.9}$$

а для второго, соответственно,

$$v_b = \frac{v_2\tau + v_1(t-\tau)}{t}. \tag{2.3.10}$$

Из условий задачи известно, что $v_a - v_b = \Delta v$. Подставляя в это уравнение формулы (2.3.9) и (2.3.10), а также домножая его на t, получим

$$v_1\tau + v_2(t-\tau) - v_2\tau - v_1(t-\tau) = \Delta v t. \tag{2.3.11}$$

Перегруппировав слагаемые, получим

$$v_2 t - 2v_2\tau - v_1 t + 2v_1\tau = \Delta v t, \tag{2.3.12}$$

откуда окончательно

$$t = \frac{2(v_2 - v_1)\tau}{v_2 - v_1 - \Delta v} = 100\,\text{с}. \tag{2.3.13}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.1.5. Призма (30 баллов)

Условие

Для тонкого контроля параметров призмы используется следующая установка: отмечается точка, в которую падает лазерный луч, направленный на экран строго под прямым углом (пунктирный на рисунке). Затем на пути луча устанавливается исследуемая призма так, что задняя (первая по ходу распространения луча) ее грань оказывается строго перпендикулярна лучу, и измеряется расстояние d, на которое в результате этого смещается пятно лазера.

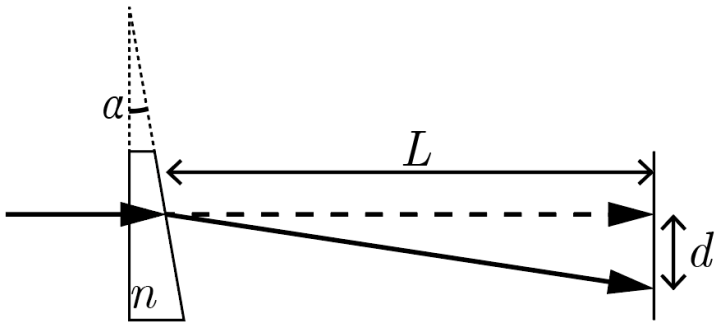


Рис. 2.3.2

Определите показатель преломления стекла, из которого изготовлена призма, если расстояние от передней грани призмы до экрана $L = 3$ м, смещение пятна при установке призмы $d = 12$ см, а угол между передней и задней поверхностями призмы $\alpha = 3°$. Используйте приближение малых углов.

Решение

На первой по ходу распространения луча грани призмы свет не преломляется, поскольку падает на нее под прямым углом. Следовательно, угол падения луча на переднюю грань призмы равен α. Тогда по закону Снеллиуса

$$n = \frac{\sin\beta}{\sin\alpha}, \tag{2.3.14}$$

где β — угол преломления луча, что с учетом приближения малых углов $\sin\alpha \approx \alpha \approx \approx \operatorname{tg}\alpha$ (в радианах) принимает форму

$$n \approx \frac{\beta}{\alpha}. \tag{2.3.15}$$

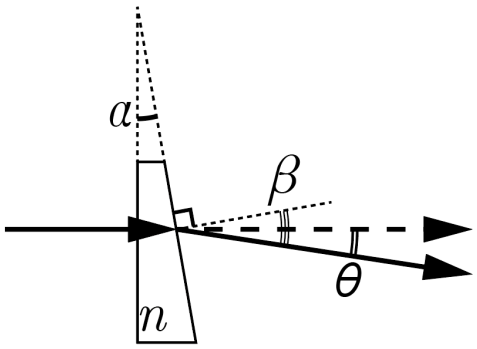


Рис. 2.3.3

Из геометрии рисунка легко видеть, что $\beta = \alpha + \theta$, где θ — угол, который преломленный луч составляет с направлением своего распространения до установки призмы. При этом $\operatorname{tg}\theta = \frac{d}{L}$, откуда

$$n\alpha \approx \beta = \alpha + \operatorname{arctg}\frac{d}{L} \Rightarrow n \approx 1 + \frac{d}{\alpha L} \approx 1{,}76. \tag{2.3.16}$$

Погрешность 0,02.

Ответ: $1{,}76 \pm 0{,}02$.

2.3.2. Первая волна. Задачи 10–11 класса

Задачи первой волны предметного тура по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63480/enter/.

Задача 2.3.2.1. Беспилотник (10 баллов)

Условие

Беспилотник, двигаясь равномерно и прямолинейно и обладая при этом импульсом $p_0 = 10^4$ кг $\cdot$ м/с, преодолевает дистанцию $L = 20$ км ровно за 1,5 мин. За какое

время преодолеет ее этот же беспилотник, двигаясь также равномерно и прямолинейно, но обладая на $\Delta p = 10^3$ кг $\cdot$ м/с меньшим импульсом?

Решение

Импульс p тела — это произведение его массы на его скорость, поэтому скорость беспилотника легко может быть вычислена как

$$p_0 = \frac{mL}{t_0}, \tag{2.3.17}$$

где t_0 — время в пути с импульсом p_0. Искомое время t можно аналогично выразить через скорость v беспилотника во второй рассмотренной ситуации как

$$t = \frac{L}{v} = \frac{Lm}{p_0 - \Delta p}. \tag{2.3.18}$$

Выражая массу беспилотника из 2.3.17 и подставляя ее в 2.3.18, получим:

$$t = \frac{\cancel{L} p_0 t_0}{\cancel{L}(p_0 - \Delta p)} = 100\,\text{с}. \tag{2.3.19}$$

Погрешность 1 с.

Ответ: (100 ± 1) с.

Задача 2.3.2.2. Грузовик (15 баллов)

Условие

На плоское горизонтальное дно кузова транспортного грузовика погрузили большой грузовой контейнер и забыли его закрепить. Благодаря силе трения контейнер оставался в покое относительно грузовика до тех пор, пока ускорение последнего не превосходило $a_0 = 2{,}0\,\text{м/с}^2$, и начинал скользить при превышении этого значения. Совершая маневр, грузовик приобрел ускорение $a = 2{,}12\,\text{м/с}^2$, направленное по ходу движения. Какое время длился маневр, если в результате контейнер сдвинулся на $l = 1{,}5\,\text{м}$ относительно грузовика?

Решение

Исходя из того, что контейнер остается на месте при ускорении грузовика до a_0, можно, по второму закону Ньютона, заключить, что сила трения покоя, обеспечивающая это ускорение для контейнера, не превышает значения

$$F = ma_0, \tag{2.3.20}$$

где m — масса контейнера. При любом маневре, при котором контейнер начинает скользить, на него действует сила трения скольжения, равная F и, следовательно, его ускорение относительно дороги оказывается равно a_0.

Во время маневра ускорение контейнера относительно грузовика равно (по модулю) $a - a_0$, а пройденное контейнером относительно грузовика расстояние может быть выражено по законам кинематики как

$$l = \frac{(a - a_0)t^2}{2}, \tag{2.3.21}$$

где t — искомое в задаче время. Преобразуя эту формулу, получим

$$t = \sqrt{\frac{2l}{a - a_0}} = 5\,\text{с}. \tag{2.3.22}$$

Погрешность $0{,}1$ с.

Ответ: $(5{,}0 \pm 0{,}1)$ с.

Задача 2.3.2.3. Методичка (20 баллов)

Условие

На лабораторной работе по физике в распоряжении школьников оказались резисторы двух номиналов: с сопротивлениями x и y кОм, а также конденсаторы двух номиналов: с емкостями x и y мкФ, при этом $x > y$. В старой методичке, посвященной этой лабораторной работе, была изображена схема, приведенная на рисунке, чернила на которой сильно затерлись. В результате Витя решил, что изображенные элементы являются резисторами, и, соединив их согласно схеме, получил элемент с эквивалентным сопротивлением $R = 5$ кОм. Таня же решила, что это конденсаторы и, соединив их, получила элемент с эквивалентной емкостью $C = 2$ мкФ. Определите, чему равнялось число y.

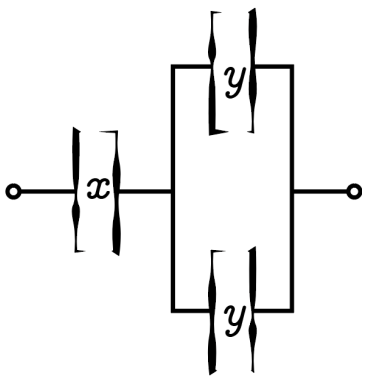


Рис. 2.3.4

Решение

При последовательном соединении резисторов их сопротивления складываются, а при параллельном — складываются обратные сопротивлениям величины. Поэтому сопротивление R схемы Вити через числа x и y в килоомах (кОм) может быть выражено по формуле

$$R = x + \frac{y}{2}. \tag{2.3.23}$$

Для конденсаторов правила поиска эквивалентной емкости при их последовательном и параллельном соединениях в точности обратные, поэтому емкость схемы Тани может быть выражена в микрофарадах (мкФ) по формуле

$$C = \frac{2xy}{x + 2y}. \tag{2.3.24}$$

Выразим x из уравнения (2.3.23) и подставим в (2.3.24):

$$C = \frac{2(R - y/2)y}{R + 3y/2}. \tag{2.3.25}$$

Домножив полученное уравнение на знаменатель дроби и раскрыв скобки, получим

$$RC + \frac{3Cy}{2} = 2Ry - y^2. \tag{2.3.26}$$

Поскольку величины x и y имеют разную размерность в разных частях задачи, имеет смысл сразу перейти к численным значениям

$$10 + 3y = 10y - y^2. \tag{2.3.27}$$

Это квадратное уравнение, которое легко привести к канонической форме

$$y^2 - 7y + 10 = 0 \tag{2.3.28}$$

и решить с помощью дискриминанта

$$y_{1,2} = \frac{7 \pm \sqrt{49 - 40}}{2} = \frac{7 \pm 3}{2}. \tag{2.3.29}$$

Снова воспользовавшись уравнением (2.3.23), легко определить, что при $y = 5$ (корень квадратного уравнения с плюсом) $x = 2{,}5$, что не удовлетворяет условию $x > y$. В то же время, при $y = 2$ (корень с минусом) $x = 4$, что удовлетворяет этому условию. Стало быть, верный корень — $y = 2$.

Погрешность $0{,}1$.

Ответ: $2{,}0 \pm 0{,}1$.

Задача 2.3.2.4. Морозилка (25 баллов)

Условие

Морозильная установка работает по циклу Карно, обходимому в обратном направлении. Какую работу должна потребить такая установка, чтобы заморозить $m = 0{,}4$ кг воды, взятой при ее температуре замерзания, передав полученную от нее теплоту в помещение, температура θ которого равна 30 °C? Удельная теплота плавления и кристаллизации воды $\lambda = 333$ кДж/кг, абсолютный ноль температур $T_0 = -273$ °C.

Решение

Идеальный тепловой двигатель (машина Карно) работает, как известно, по циклу, состоящему из двух изотерм и двух адиабат, и имеет КПД

$$\eta = \frac{T_2 - T_1}{T_2}, \tag{2.3.30}$$

где T_1 — минимальная, а T_2 — максимальная температуры в цикле. По определению КПД тепловой машины он равен отношению работы A, совершаемой газом за цикл, к теплоте Q_2, получаемой им от нагревателя

$$\frac{T_2 - T_1}{T_2} = \eta = \frac{A}{Q_2}. \tag{2.3.31}$$

Отсюда Q_2 может быть выражено как

$$Q_2 = \frac{AT_2}{T_2 - T_1}. \tag{2.3.32}$$

Поскольку за полный цикл внутренняя энергия не изменяется, количество теплоты Q_1, которую газ отдает холодильнику такой машины, равна разнице

$$Q_1 = Q_2 - A = \frac{AT_1}{T_2 - T_1}. \tag{2.3.33}$$

В рассматриваемой задаче тепловой двигатель заменен холодильной машиной, для чего его цикл необходимо обходить в обратном направлении. При этом тепловой резервуар с температурой T_2 начинает получать тепло, а с температурой T_1 — отдавать, но по модулю количества теплоты, которыми газ обменивается с этими тепловыми резервуарами, не изменяются. Остается заметить, что в описанной в условиях холодильной машине $T_1 = 0\,^\circ\mathrm{C} = 273\,\mathrm{K}$, $T_2 = \theta$, $Q_1 = \lambda m$, поскольку забираемая у теплового резервуара с меньшей температурой теплота идет на замораживание в нем воды. Подставив эти данные в (2.3.33), получим

$$\lambda m = \frac{AT_1}{\theta - T_1}, \tag{2.3.34}$$

откуда окончательно выразим ответ

$$A = \frac{\lambda m(\theta - T_1)}{T_1} \approx 14{,}6\,\text{Дж}. \tag{2.3.35}$$

Погрешность: $0{,}5$ Дж.

Ответ: $(14{,}6 \pm 0{,}5)$ Дж.

Задача 2.3.2.5. Электромагнит (30 баллов)

Условие

Для большого промышленного электромагнита критически важной стала проблема охлаждения. Было установлено, что при пропускании через электромагнит

тока $I_1 = 10$ А он нагревается до температуры $t_1 = 70\,^\circ$C, после чего перестает увеличивать свою температуру, а при пропускании через него тока $I_2 = 20$ А — до температуры $t_2 = 205\,^\circ$C.

Определите температуру θ в помещении цеха, в котором используется электромагнит, если известно, что основным механизмом, отвечающим за охлаждение магнита, выступает теплопроводность, мощность которой прямо пропорциональна разнице температур между телами, обменивающимися теплом.

Решение

Как указано в условиях, мощность теплопроводности прямо пропорциональна разнице температур между магнитом и окружающим его воздухом в помещении цеха. Обозначим коэффициент этой пропорциональности κ

$$\begin{cases} P_1 = \kappa(\theta - t_1), \\ P_2 = \kappa(\theta - t_2). \end{cases} \tag{2.3.36}$$

Повышение температуры останавливается, когда мощность производимого катушкой тепла и мощность тепла, уходящего от катушки, благодаря теплообмену, оказываются равны. Первую можно выразить из закона Джоуля – Ленца

$$\begin{cases} P_1 = I_1^2 R, \\ P_2 = I_2^2 R, \end{cases} \tag{2.3.37}$$

где R — сопротивление катушки.

Разделим друг на друга уравнения системы (2.3.36) и уравнения системы (2.3.37), а затем приравняем эти отношения:

$$\frac{P_1}{P_2} = \frac{\theta - t_1}{\theta - t_2} = \frac{I_1^2}{I_2^2}. \tag{2.3.38}$$

Тривиальными алгебраическими преобразованиями выразим θ

$$(\theta - t_1)I_2^2 = (\theta - t_2)I_1^2 \Rightarrow \theta = \frac{t_1 I_2^2 - t_2 I_1^2}{I_2^2 - I_1^2} = 25\,^\circ\text{C}. \tag{2.3.39}$$

Погрешность: $0{,}1\,^\circ$C.

Ответ: $(25{,}0 \pm 0{,}1)$.

2.3.3. Вторая волна. Задачи 8–9 класса

Задачи второй волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63464/enter/.

Задача 2.3.3.1. Аккумулятор тепла (10 баллов)

Условие

Для печи отопления требуется разработать аккумулятор тепла, представляющий собой емкость фиксированного объема, заполненную тем или иным минералом. Используя таблицу 2.3.1 плотностей ρ и удельных теплоемкостей $c_{\text{уд}}$ различных подходящих для этого пород, расположите их в порядке увеличения количества теплоты, которое может быть запасено в таком аккумуляторе при его нагреве до одной и той же температуры θ. Считайте, что θ заведомо меньше температур, при которых любой из этих минералов начнет плавиться или химически разрушаться, а тепловое расширение этих минералов при нагреве до θ пренебрежимо мало.

Таблица 2.3.1. Плотности и удельные теплоемкости

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)
A	Кварц	2,6	0,75
B	Базальт	2,8	0,85
C	Талькохлорит	2,75	0,98
D	Нефрит	3	1,1
E	Порфирит	1,45	0,83

Введите в поле ответа последовательность букв, соответствующих выбранным минералам, без пробелов, от наименьшего к наибольшему количеству запасаемой теплоты.

Решение

Количество тепла Q, которое может быть запасено в тепловом аккумуляторе фиксированного объема V, удобно выразить через массу m материала этого аккумулятора

$$Q = c_{\text{уд}} m(\theta - t_0), \tag{2.3.40}$$

где t_0 — начальная температура теплоаккумулятора. В свою очередь, масса m элементарно выражается через плотность вещества и объем V

$$m = \rho V, \tag{2.3.41}$$

откуда

$$Q = c_{\text{уд}} \rho V(\theta - t_0). \tag{2.3.42}$$

Поскольку величины V, θ, t_0 независимы от выбранного вещества, задача сводится к расположению в порядке возрастания произведений $c_{\text{уд}}\rho$. Найдем эти произведения для всей таблицы 2.3.1.

Таблица 2.3.2

	Минерал	ρ, г/см^3	$c_{\text{уд}}$, кДж/(кг · °C)	$c_{\text{уд}}\rho$, кДж/(м^3 · °C)
A	Кварц	2,6	0,75	1 950
B	Базальт	2,8	0,85	2 380
C	Талькохлорит	2,75	0,98	2 695
D	Нефрит	3	1,1	3 300
E	Порфирит	1,45	0,83	1 204

Ответ: EABCD.

Задача 2.3.3.2. Изображения (15 баллов)

Условие

Тонкая собирающая линза имеет фокусное расстояние $F = 20$ см. Вдоль ее оптической оси перед линзой расположено плоское зеркало, на расстоянии $h = 2$ см от которого и $d = 60$ см от линзы размещен светодиод S (рис. 2.3.5). Найдите расстояние между двумя действительными изображениями светодиода, формируемыми этой оптической системой.

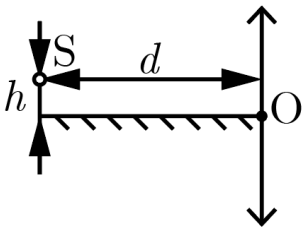


Рис. 2.3.5

Решение

Зеркало формирует мнимое изображение S_1 источника, расположенное в противоположном от него полупространстве на таком же расстоянии от зеркала, как и сам источник. В силу перпендикулярности зеркала и линзы, это мнимое изображение также окажется на расстоянии d от плоскости линзы. Далее линза формирует два действительных изображения: одно непосредственно от источника S (на рис. 2.3.6 оно обозначено S_2) и другое — от его мнимого изображения S_1 (S_3 на рисунке).

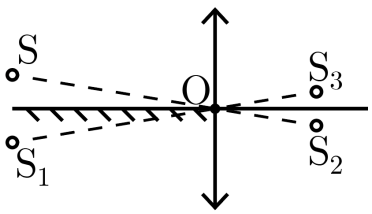


Рис. 2.3.6

Расстояние f, на котором находятся оба действительных изображения от плоскости линзы, легко найти по формуле тонкой линзы

$$\frac{1}{f}+\frac{1}{d}=\frac{1}{F}\Rightarrow f=\frac{Fd}{d-F}. \tag{2.3.43}$$

Искомое расстояние l между изображениями S_2 и S_3, благодаря подобию треугольников $\triangle \mathrm{OSS_1}$ и $\triangle \mathrm{OS_2S_3}$, относится к расстоянию $2h$ между источником и его мнимым изображением S_1 так же, как расстояния от соответствующих изображений и источников до плоскости линзы, являющиеся высотами указанных треугольников

$$\frac{l}{2h}=\frac{f}{d}=\frac{F}{d-F}. \tag{2.3.44}$$

Отсюда окончательно находим

$$l=2h\frac{F}{d-F}=2\,\text{см}. \tag{2.3.45}$$

Погрешность $0{,}1$ см.

Ответ: $l=(2{,}0\pm 0{,}1)$ см.

Задача 2.3.3.3. Пила (30 баллов)

Условие

Циркулярная пила представляет собой пильный диск диаметром $D=19$ см, вращающийся с частотой $4\,500$ об/мин. Определите среднюю силу сопротивления заготовки вращению полотна пилы, если за один пропил, длившийся $t=3$ с, выделилось $Q=5$ кДж тепла, а пила соприкасалась с заготовкой только узкой полоской своей внешней кромки.

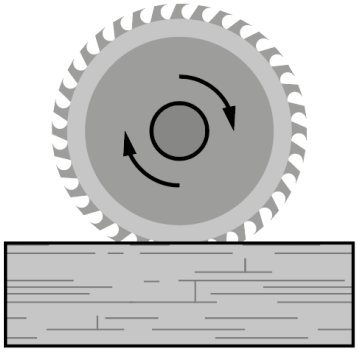


Рис. 2.3.7

Решение

При вращении пилы диссипативные силы (трения различных типов) переводят механическую энергию пильного диска в тепловую. При этом количество выделяющегося тепла равно по модулю работе A этих сил. Последнюю легко найти из ее определения

$$A = FS = Fvt, \tag{2.3.46}$$

где S — путь точек соприкосновения диска с заготовкой, v — скорость этих точек. При вращении диска скорость точек его кромки удобно выразить через период вращения T этого диска

$$v = \frac{\pi D}{T} = \pi D\nu, \tag{2.3.47}$$

где ν — частота вращения диска в оборотах в секунду. Подставляя (2.3.47) в (2.3.46) и учитывая $Q = A$, получим окончательно

$$Q = \pi FD\nu t \Rightarrow F = \frac{Q}{\pi D\nu t} \approx 37\,\text{Н}. \tag{2.3.48}$$

Погрешность 1 Н.

Ответ: $F = (37 \pm 1)\,\text{Н}$.

Задача 2.3.3.4. Питстоп (25 баллов)

Условие

Транспортный робот перемещается из города A в город B, двигаясь практически все время с некоторой постоянной скоростью v. Однако один раз за маршрут ему необходима остановка для заправки и краткого технического обслуживания. Инженеры установили, что при уменьшении длительности этой остановки вдвое скорость движения робота на остальной части маршрута можно будет снизить на $\delta = 10\%$, сохранив при этом его среднюю путевую скорость, что поможет повысить безопасность и экономичность движения. Определите, на сколько процентов (от исходного значения) удалось бы снизить скорость v без изменения средней путевой, если бы от технической остановки удалось полностью отказаться?

Решение

Средняя путевая скорость определяется как отношение всего пути ко всему времени, которое этот путь занимает. Поскольку расстояние между городами неизменно, сохранение средней путевой скорости означает и сохранение общего времени в пути (включая остановку). Следовательно, уменьшение длительности остановки на Δt эквивалентно увеличению времени непосредственного движения на ту же величину. Обозначим общее время робота в пути t, исходную длительность его остановки τ, а исходную скорость движения v_0. Тогда путь робота может быть выражен до и после снижения времени остановки как

$$S = v_0(t - \tau) = v_0(1 - \delta)\left(t - \frac{\tau}{2}\right). \tag{2.3.49}$$

Сократив v_0 и перегруппировав слагаемые, получим

$$\delta t = \tau\left(1 - \frac{1}{2} + \frac{\delta}{2}\right) = \tau\frac{\delta+1}{2}. \tag{2.3.50}$$

Полностью избавившись от остановки, таким образом, робот будет двигаться в течение времени

$$t = \tau\frac{\delta+1}{2\delta}. \tag{2.3.51}$$

Аналогично, время движения $t - \tau$ при наличии остановки удобно записать как

$$t - \tau = \tau\left(\frac{\delta+1}{2\delta} - 1\right) = \tau\frac{1-\delta}{2\delta}. \tag{2.3.52}$$

Обозначив v_1 скорость, которой можно добиться, исключив остановку, запишем через эти выражения путь и приравняем его в случаях с остановкой и без

$$v_1 t = v_0(t-\tau) \Rightarrow v_1\tau\frac{\delta+1}{2\delta} = v_0\tau\frac{1-\delta}{2\delta}. \tag{2.3.53}$$

Отсюда окончательно

$$\frac{v_1}{v_0} = \frac{1-\delta}{1+\delta} \approx 0{,}818. \tag{2.3.54}$$

Итого скорость движения без остановки может составлять приблизительно $81{,}8\%$ от исходной скорости, то есть ниже ее на $18{,}2\%$.

Погрешность $0{,}5\%$.

Ответ: $(18{,}2 \pm 0{,}5)\%$.

Задача 2.3.3.5. Сцепка (30 баллов)

Условие

Два абсолютно одинаковых ползунковых реостата, сопротивления которых могут изменяться в пределах от 0 до $R_0 = 2\,\text{кОм}$, размещены параллельно на печатной плате и соединены как изображено на рис. 2.3.8. Из-за ошибки в процессе пайки изоляция их ползунков слиплась таким образом, что ползунки всегда занимают одно и то же положение на обоих реостатах, но электрический контакт между ними отсутствует (это соединение обозначено на рисунке пунктиром). Найдите разницу между максимальным и минимальным сопротивлениями полученной батареи.

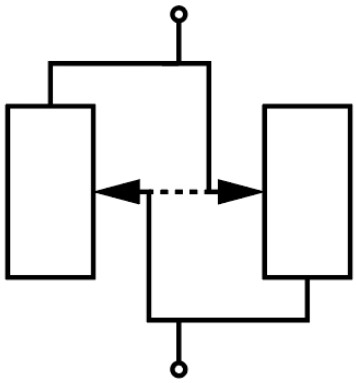


Рис. 2.3.8

Решение

Реостаты на схеме соединены параллельно, поэтому общее сопротивление схемы R может быть выражено через сопротивления $R_{1,2}$ каждого из реостатов по формуле

$$R = \frac{R_1 R_2}{R_1 + R_2}. \tag{2.3.55}$$

Несложно видеть из схемы, что когда ползунок находится в крайнем верхнем положении, левый реостат имеет нулевое сопротивление, а правый — сопротивление R_0 и наоборот. Величина сопротивления находится в линейной зависимости от длины провода. Из этого можно заключить, что при любом положении ползунка

$$R_2 = R_0 - R_1. \tag{2.3.56}$$

Подставляя этот результат в (2.3.55), получим

$$R = \frac{R_1(R_0 - R_1)}{R_0} = R_1 - \frac{R_1^2}{R_0}. \tag{2.3.57}$$

График полученной функции является параболой. Его минимумы и максимумы могут лежать либо на границах диапазона изменения R_1, либо в вершине соответствующей параболы. Поскольку коэффициент перед квадратным слагаемым отрицательный, парабола «повернута» ветвями вниз, то есть на границах диапазона (при $R_1 = 0$ или $R_1 = R_0$) сопротивления батареи минимальны и равны 0, а в ее вершине (которая, как легко видеть из симметрии или непосредственно по формуле $x_{max} = -b/(2a)$, лежит в центре диапазона, при $R_1 = R_2 = \dfrac{R_0}{2}$) сопротивление батареи равно $\dfrac{R_0}{4}$.

Таким образом,

$$R_{max} - R_{min} = \frac{R_0}{4} - 0 = \frac{R_0}{4} = 0{,}5\,\text{кОм}. \tag{2.3.58}$$

Погрешность 0,01 кОм.

Ответ: $l = (0{,}50 \pm 0{,}01)$ кОм.

2.3.4. Вторая волна. Задачи 10–11 класса

Задачи второй волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63481/enter/.

Задача 2.3.4.1. Зарядка (10 баллов)

Условие

Для увеличения ресурса аккумулятора его зарядка происходит по специальной программе, учитывающей внешние условия, интенсивность использования прибора и другие факторы. Рассчитав оптимальный режим, зарядное устройство в течении $\tau = 10$ мин подавало на аккумулятор ток, линейно возраставший со временем от нуля до максимального значения $I_0 = 3$ А, затем в течение $2{,}5\tau$ поддерживало постоянное значение этого тока и, наконец, на протяжении $\tau/2$, также линейно опускало ток от максимального значения до нуля. Какой общий заряд поступил на положительную клемму аккумулятора за все это время?

Решение

Один из способов решения задачи состоит в обнаружении аналогии между током и движением. Подобно тому, как скорость описывает темп изменения координаты, сила тока описывает темп поступления заряда на аккумулятор. Из кинематики известно, что при равномерном увеличении этого темпа (равноускоренном движении) от нуля, либо при равномерном снижении этого темпа (равнозамедленном движении) до нуля тело проходит вдвое меньшее расстояние, чем при движении с постоянной скоростью, равной максимальной на рассматриваемом участке. Применяя этот результат к току, заметим, что за время $2{,}5\tau$ постоянного тока зарядки на аккумулятор поступил заряд

$$q_1 = 2{,}5\tau I_0, \tag{2.3.59}$$

а за общее время $1{,}5\tau$ увеличения и уменьшения силы тока — заряд

$$q_2 = \frac{1{,}5\tau I_0}{2} = 0{,}75\tau I_0. \tag{2.3.60}$$

Складывая эти заряды, получим окончательно

$$q = 3{,}25\tau I_0 = 5850\,\text{Кл}. \tag{2.3.61}$$

Задача также может быть решена графически, изображением зависимости $I(t)$ и вычислением площади под ней. Фактически такое решение также является применением аналогии, но геометрической, а не кинематической.

Погрешность 50 Кл.

Ответ: (5850 ± 50) Кл.

Задача 2.3.4.2. Принтер (15 баллов)

Условие

Печатающая головка 3D-принтера может перемещаться вдоль направляющей (координата x), которая также может смещаться в перпендикулярном направлении (координата y) под действием двух сервоприводов. Для изготовления детали на принтер была передана программа, согласно которой сервоприводы должны перемещать головку по законам $x(t) = 0{,}2\sin(t/10)$; $y(t) = 0{,}1 + 0{,}2\cos(t/10)$, где t — время, а все величины даны в основных единицах СИ. Найдите величину ускорения печатающей головки при выполнении этой программы.

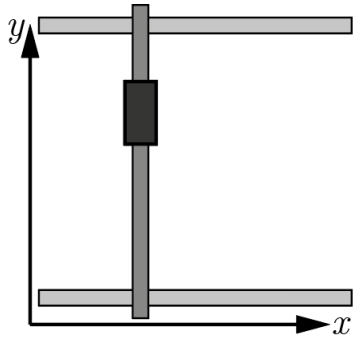


Рис. 2.3.9

Решение

Прежде всего заметим, что уравнения приведенного типа параметрически задают окружность. Это следует из самого определения синуса и косинуса. Радиус R этой окружности равен множителю перед синусом и косинусом (т. к. в математическом определении тригонометрических функций используется единичная окружность), то есть $R = 0{,}2$ м. Здесь было учтено, что основными единицами СИ для измерения длины являются метры.

Поскольку зависимость угла на окружности (аргумента синуса и косинуса) от времени линейна, модуль скорости v печатающей головки постоянен. Чтобы найти его, определим период обращения. Печатающая головка описывает полную окружность, когда аргумент синуса и косинуса меняется на 2π, что происходит при достижении t значения $T = 20\pi$ с. Тогда

$$v = \frac{2\pi R}{T}. \tag{2.3.62}$$

Окончательно заметим, что при неизменной по модулю скорости головки единственное ее ускорение — центростремительное, которое может быть найдено как

$$a = \frac{v^2}{R} = \frac{4\pi^2 R}{T^2} = \frac{4\cancel{\pi^2}\cdot 0{,}2}{400\cancel{\pi^2}} \approx 2\,\text{мм/с}^2. \tag{2.3.63}$$

Погрешность $0{,}1\,\text{мм/с}^2$.

Ответ: $(2{,}0 \pm 0{,}1)\,\text{мм/с}^2$.

Задача 2.3.4.3. Плита (20 баллов)

Условие

Квадратная плита ABCD со стороной $a = 40\,\text{см}$ шарнирно закреплена в одной точке и вращается вокруг оси, перпендикулярной ее плоскости с постоянной угловой скоростью. При этом в некоторый момент времени скорость вершины C этой плиты направлена строго на вершину D, а ускорение вершины A — строго на вершину B (см. рис. 2.3.10). На каком расстоянии от центра пластины находится шарнир?

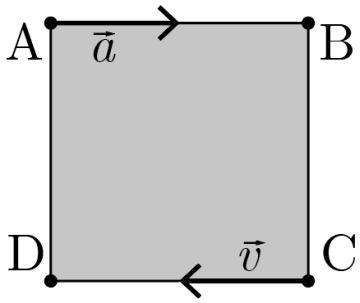


Рис. 2.3.10

Решение

При плоском вращении скорость каждой точки тела направлена перпендикулярно направлению из этой точки на ось вращения, а ускорение (центростремительное) — непосредственно на эту ось. Поэтому, проведя одну прямую через точку C перпендикулярно ее скорости, а другую — через точку A вдоль ее ускорения, можно найти ось вращения как точку пересечения этих прямых. Такой точкой будет, разумеется, вершина B, а расстояние от нее до центра пластины равно

$$l = \frac{\sqrt{2}}{2}a \approx 28{,}3\,\text{см}. \tag{2.3.64}$$

Погрешность $0{,}5\,\text{см}$.

Ответ: $(28{,}3 \pm 0{,}5)\,\text{см}$.

Задача 2.3.4.4. Прямоугольники (25 баллов)

Условие

К проекту модельного теплового двигателя, рабочим телом которого является идеальный одноатомный газ, прилагается pV-диаграмма его рабочего цикла, представляющая собой прямоугольник 1234. К сожалению, автор не указал ни давления, ни объемы характерных точек, а ограничился «площадями» (в энергетических единицах) некоторых прямоугольников на данной диаграмме, которые указаны на рис. 2.3.10. Да к тому же самая важная площадь — площадь внутри цикла 1234, стерлась. Тем не менее определите КПД данного двигателя.

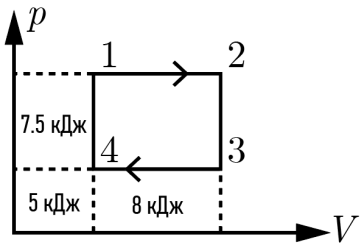


Рис. 2.3.11

Решение

КПД η теплового двигателя определяется как отношение работы A газа за один цикл этого двигателя к количеству теплоты Q, получаемой газом за этот цикл от нагревателя:

$$\eta = \frac{A}{Q}. \tag{2.3.65}$$

Первая — есть площадь прямоугольника 1234. Найти ее просто, если заметить, что поскольку высота (вдоль оси p) у верхних двух прямоугольников одинакова, их площади относятся так же, как и их ширины (вдоль оси V), и то же верно для нижней пары прямоугольников

$$A = 7{,}5\frac{8}{5} = 12\,\text{кДж}. \tag{2.3.66}$$

Чтобы найти теплоту Q, воспользуемся первым началом термодинамики $Q = A + \Delta U$ и заметим, что газ получает теплоту на процессах 41 и 12. Работа за эти два процесса равна полной площади под отрезком 12

$$A_{412} = 12 + 8 = 20\,\text{кДж}, \tag{2.3.67}$$

а внутренняя энергия может быть удобно вычислена по формуле

$$U = \frac{3}{2}PV, \tag{2.3.68}$$

из которой следует, что внутренняя энергия U_4 газа в состоянии 4 равна

$$U_4 = \frac{3}{2}5 = 7{,}5\,\text{кДж}, \tag{2.3.69}$$

поскольку произведение PV для этого состояния есть площадь одного соответствующего прямоугольника. Аналогично, внутренняя энергия U_2 газа в состоянии 2 равна

$$U_2 = \frac{3}{2}(7{,}5 + 12 + 5 + 8) = 48{,}75\,\text{кДж}, \tag{2.3.70}$$

поскольку в этом состоянии соответствующее произведение PV равно общей площади всех прямоугольников на диаграмме.

Подставляя все найденные величины в исходное уравнение (2.3.65), получим окончательно

$$\eta = \frac{A}{A_{412} + U_2 - U_4} = \frac{12}{20 + 48{,}75 - 7{,}5} \approx 19{,}6\%. \tag{2.3.71}$$

Погрешность 1%.

Ответ: $(19{,}6 \pm 1{,}0)\%$

Задача 2.3.4.5. Трюм (30 баллов)

Условие

Трюм грузового судна представляет собой призму с основанием в виде равностороннего треугольника вершиной вниз. Длина внешней стороны треугольника $a = 15\,\text{м}$, толщина его стенок $d = 1{,}5\,\text{м}$, длина киля судна (высота призмы) $l = 40\,\text{м}$. Инженерами было рассчитано, что для сохранения устойчивости судна центр тяжести сыпучего груза, перевозимого в трюме, должен быть хотя бы на $h = 2\,\text{м}$ ниже центра тяжести вытесняемой трюмом воды при его полном погружении. Какой максимальный объем груза можно разместить в трюме, если при насыпании его центр тяжести занимает самое низкое доступное положение?

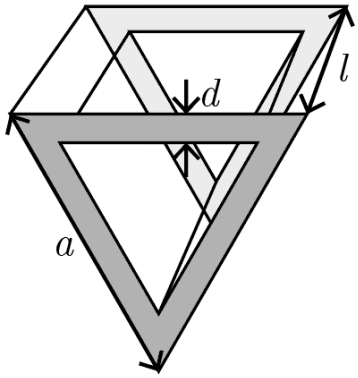


Рис. 2.3.12

Решение

Сыпучий груз занимает объем, форма которого также является призмой с основанием в виде равностороннего треугольника, во всяком случае, при необходимости понизить центр тяжести. В силу симметрии, центр тяжести равностороннего треугольника находится в его геометрическом центре. Поскольку центр тяжести груза должен быть на h ниже, чем центр тяжести вытесненной воды, центр треугольника, формируемого сечением насыпанного груза (темный на рис. 2.3.13), на h ниже центра трюма.

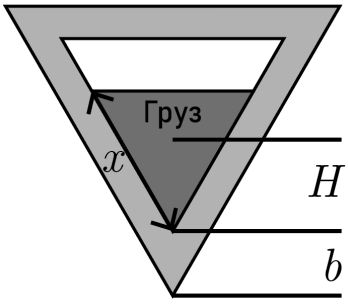


Рис. 2.3.13

Отсчитывая от киля (нижней вершины треугольника) высоту $y_{\text{в}}$, на которой

находится центр тяжести вытесненной воды, легко выразить как

$$y_{\text{в}} = \frac{\sqrt{3}}{3}a, \tag{2.3.72}$$

поскольку радиус описанной окружности для равностороннего треугольника в $\sqrt{3}/3$ раз меньше его стороны.

Высота $y_{\text{г}}$ центра тяжести груза может быть найдена как

$$y_{\text{г}} = b + H, \tag{2.3.73}$$

где b — толщина стенки вдоль соответствующего направления (см. рис. 2.3.13), а H — высота центра тяжести груза от нижней внутренней точки трюма. Обе эти величины вычисляются из тригонометрии

$$b = 2d; \quad H = \frac{\sqrt{3}}{3}x, \tag{2.3.74}$$

где x — сторона треугольника, являющегося поперечным сечением груза.

Учитывая $y_{\text{г}} + h = y_{\text{в}}$, получим

$$\frac{\sqrt{3}}{3}x + 2d + h = \frac{\sqrt{3}}{3}a, \tag{2.3.75}$$

откуда

$$x = a - \sqrt{3}(2d + h) \approx 6{,}34\,\text{м}. \tag{2.3.76}$$

Объем, занимаемый грузом при такой длине его стороны, находится как произведение площади равностороннего треугольника на длину киля

$$V = \frac{\sqrt{3}}{4}x^2 l \approx 696\,\text{м}^3. \tag{2.3.77}$$

Погрешность 10 м^3.

Ответ: (696 ± 10) м^3.

2.3.5. Третья волна. Задачи 8–9 класса

Задачи третьей волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63465/enter/.

Задача 2.3.5.1. Башня (10 баллов)

Условие

В гидравлической системе используется башня, заполненная минеральным маслом с плотностью $\rho = 900\,\text{кг/м}^3$. Верхний уровень масла расположен на $h = 60\,\text{м}$ выше, чем смотровое окно в трубе с маслом, закрепленное на ней при помощи $n = 16$ одинаковых болтов. Определите, какую нагрузку должен выдерживать каждый из этих болтов на разрыв, чтобы обеспечить трехкратный запас прочности? Площадь смотрового окна $S = 0{,}1\,\text{м}^2$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

Давление p масла на уровне окна элементарно вычисляется по формуле гидростатического давления

$$p = \rho g h. \tag{2.3.78}$$

Исходя из определения всякого давления p как отношения силы F к площади S, на которую действует эта сила, найдем силу со стороны жидкости, «пытающуюся выдавить» смотровое окно

$$F = pS = \rho g h S. \tag{2.3.79}$$

Искомая рассчетная нагрузка f каждого из болтов может быть получена домножением этой силы на 3 (требуемый запас прочности) и делением на число болтов n, по которым распределяется нагрузка

$$f = \frac{3F}{16} = \frac{3\rho g h S}{16} \approx 9{,}9\,\text{кН}. \tag{2.3.80}$$

Погрешность $0{,}1$ кН.

Ответ: $(9{,}9 \pm 0{,}1)$ кН.

Задача 2.3.5.2. Реостат (15 баллов)

Условие

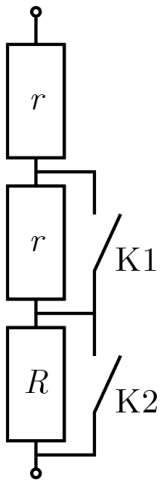


Рис. 2.3.14

Имея в своем распоряжении резисторы только двух различных номиналов, начинающий радиолюбитель изготовил ступенчатый реостат оригинальной конструкции, позволяющий, переключая ключи, получить четыре различных значения сопротивления. Увы, на приложенной к прибору схеме он забыл указать сопротивления отдельных резисторов, указав только, какие из них совпадают. В техническом паспорте устройства остались данные о том, что при замыкании ключа K_1 и размыкании ключа K_2 оно имеет сопротивление $R_1 = 2$ кОм, а напротив, при замыкании ключа K_2 и размыкании ключа K_1 — сопротивление $R_2 = 3$ кОм. Найдите максимальное сопротивление, которое можно получить, используя этот реостат.

Решение

Когда параллельно резистору коротко замыкается цепь, этот резистор фактически перестает работать, поскольку сопротивление провода пренебрежимо мало. Следовательно, замыкание ключа K1 фактически эквивалентно замене среднего резистора на отрезок провода, а ключа K2 — такой же замене нижнего. Учитывая это и применяя формулу эквивалентного сопротивления последовательно соединенных резисторов, легко получим

$$\begin{cases} R_1 = r + R, \\ R_2 = 2r. \end{cases} \tag{2.3.81}$$

Решая эту систему, находим $r = \frac{R_2}{2}$ и $R = R_1 - \frac{R_2}{2}$. Теперь точно так же составим выражения для оставшихся двух конфигураций реостата: R_3 с обоими замкнутыми ключами и R_4 с обоими разомкнутыми

$$\begin{cases} R_3 = r = \dfrac{R_2}{2} = 1{,}5\,\text{кОм}, \\ R_4 = 2r + R = R_1 + \dfrac{R_2}{2} = 3{,}5\,\text{кОм}. \end{cases} \tag{2.3.82}$$

Погрешность $0{,}01$ кОм.

Ответ: $(3{,}50 \pm 0{,}1)$ кОм.

Задача 2.3.5.3. Теплоноситель (20 баллов)

Условие

В некоторых типах ядерных реакторов в качестве теплоносителя используются жидкие металлы. Определите, сколько теплоты за 1 с забирает у реактора жидкий свинец с удельной теплоемкостью $c = 155$ Дж/(кг·°C) и средней плотностью $\rho = 10^4$ кг/м^3, если, двигаясь в трубе диаметром $d = 10$ см со скоростью $v = 20$ м/с, он нагревается от $t_1 = 400$ °C до $t_2 = 900$ °C.

Решение

Обозначим рассматриваемый промежуток времени (1 с) τ. Двигаясь со скоростью v, свинец успевает за это время пройти в трубе дистанцию $l = v\tau$. Учитывая площадь сечения трубы $S = \pi d^2/4$, можно заключить из этого, что за время τ в реактор поступает и из реактора уходит объем

$$V = Sl = \frac{\pi}{4} d^2 v\tau \tag{2.3.83}$$

расплавленного свинца. Количество теплоты Q, которое этот свинец забирает у реактора, задается выражением

$$Q = cm(t_2 - t_1) = c\rho V(t_2 - t_1) = \frac{\pi}{4} c\rho d^2 v\tau (t_2 - t_1) \approx 122\,\text{МДж}, \tag{2.3.84}$$

где m — масса поступившей и ушедшей порции свинца.

Погрешность 2 МДж.

Ответ: (122 ± 2) МДж.

Задача 2.3.5.4. Шкала (25 баллов)

Условие

Шкала вольтметра, используемого в эксперименте, имеет вид, представленный на рис. 2.3.15, и общую длину $l = 15$ см (от минимальной до максимальной отметки). Экспериментатор, глядя на прибор под углом $45°$ к плоскости шкалы, считал показания вольтметра как $U_1 = 1{,}2$ В ровно, однако на самом деле стрелка прибора находилась напротив отметки $U_2 = 0{,}8$ В. Определите, на какое расстояние отстоит стрелка от шкалы, если известно, что глаза экспериментатора находились со шкалой строго на одном уровне высоты, а деления расположены на шкале равномерно.

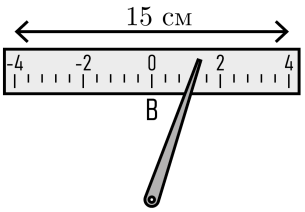


Рис. 2.3.15

Решение

Согласно условиям, глаза экспериментатора находятся на одной высоте со шкалой, поэтому удобно изобразить систему в горизонтальной плоскости (вид сверху), см. рис. 2.3.16. Поскольку угол α, под которым наблюдатель смотрит на стрелку, равен $45°$, искомое расстояние x в точности равно расстоянию y между точкой A действительных показаний прибора и точкой B считанных экспериментатором показаний (треугольник ABC, где C — стрелка — прямоугольный и равнобедренный).

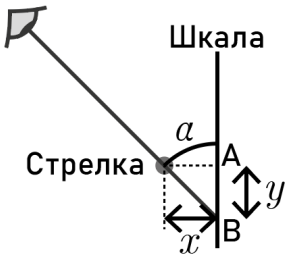


Рис. 2.3.16

Тогда остается вычислить расстояние y между отметками шкалы, соответствующими значениям U_1 и U_2. Поскольку шкала равномерная, это расстояние относится к полной длине шкалы так же, как величина абсолютной ошибки к ее разнице между ее верхним U_{max} и нижним U_{min} пределами измерений

$$\frac{y}{l} = \frac{U_2 - U_1}{U_{max} - U_{min}}. \tag{2.3.85}$$

Отсюда окончательно

$$x = y = l\frac{U_2 - U_1}{U_{max} - U_{min}} = 7{,}5 \text{ мм}. \tag{2.3.86}$$

Погрешность 0,1 мм.

Ответ: $(7{,}5 \pm 0{,}1)$ мм.

Задача 2.3.5.5. Площадка (30 баллов)

Условие

Три робота одновременно стартуют в углу A прямоугольной площадки ABCD. Все они движутся с постоянными по модулю скоростями и все заканчивают движение в точке D одновременно. Но первый робот движется по прямой вдоль стороны AD, второй — по трехзвенной ломаной ABCD, а третий — по двузвенной: сначала вдоль диагонали AC, а затем — по стороне CD. Во сколько раз средняя путевая скорость третьего робота выше, чем первого, если средняя путевая скорость второго робота выше, чем первого, в 2,5 раза? Временем на разгон, остановку и развороты роботов можно пренебречь.

Решение

Обозначив длину стороны AB (и, соответственно, CD) прямоугольника a, а длину стороны BC (и, соответственно, DA) — b, можно легко выразить через эти стороны пути $S_{1,2,3}$ всех трех роботов

$$\begin{cases} S_1 = b, \\ S_2 = 2a + b, \\ S_3 = \sqrt{a^2 + b^2} + a. \end{cases} \tag{2.3.87}$$

Поскольку время движения всех роботов совпадало, отношения их путей точно такие же, как и средних путевых скоростей:

$$\frac{v_2}{v_1} = \frac{S_2}{S_1} = \frac{2a}{b} + 1 = 2{,}5, \tag{2.3.88}$$

откуда легко найти

$$\frac{2a}{b} = 1{,}5 \Rightarrow b = \frac{4}{3}a. \tag{2.3.89}$$

Теперь, пользуясь той же логикой, найдем ответ на вопрос задачи как отношение путей третьего и первого роботов

$$\frac{v_3}{v_1} = \frac{S_3}{S_1} = \frac{\sqrt{a^2 + b^2} + a}{b} = \frac{\sqrt{\dfrac{25a^2}{9}} + a}{\dfrac{4a}{3}} = \frac{\dfrac{8a}{3}}{\dfrac{4a}{3}} = 2. \tag{2.3.90}$$

Погрешность 0,01.

Ответ: $2{,}00 \pm 0{,}01$.

2.3.6. Третья волна. Задачи 10–11 класса

Задачи третьей волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63482/enter/.

Задача 2.3.6.1. На коленке (10 баллов)

Условие

На конференции, посвященной освоению труднодоступных северных регионов, был представлен проект теплового двигателя, для работы которого используются два тепловых резервуара. Их можно собрать «на коленке»: в качестве холодильника выступает емкость с мокрым снегом, а в качестве нагревателя — котелок с кипящей водой. При этом, по заверениям авторов проекта, КПД двигателя только в $\alpha = 1{,}6$ раз уступает КПД идеальной тепловой машины с такими же холодильником и нагревателем. Найдите этот КПД. Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$. Двигатель используется при нормальном атмосферном давлении на уровне моря.

Решение

Мокрый снег представляет собой смесь льда и воды, поэтому может существовать только при температуре плавления льда, $t_\text{х} = 0\,^\circ\text{C}$. Аналогично, при атмосферном давлении температура кипящей воды может быть равна только $t_\text{н} = 100\,^\circ\text{C}$. КПД идеальной тепловой машины (машины Карно) с известными абсолютными термодинамическими температурами $T_\text{н}$ и $T_\text{х}$ холодильника задается выражением

$$\eta_0 = \frac{T_\text{н} - T_\text{х}}{T_\text{н}}. \tag{2.3.91}$$

Чтобы дать ответ на вопрос задачи, таким образом, остается разделить этот КПД на α и перевести температуры в шкалу Кельвина

$$\eta = \frac{\eta_0}{\alpha} = \frac{t_\text{н} - t_\text{х}}{\alpha(t_\text{н} - T_0)} \approx 16{,}8\%. \tag{2.3.92}$$

Погрешность: $0{,}2\%$.

Ответ: $(16{,}8 \pm 0{,}2)\%$.

Задача 2.3.6.2. Патруль (15 баллов)

Условие

Корабль береговой охраны движется с постоянной скоростью $v = 12\,\text{м/с}$ относительно поверхности воды. Наблюдательный дрон запрограммирован летать на постоянной высоте по траектории, в системе отсчета корабля представляющей собой окружность с радиусом $R = 2\,\text{км}$ и центром на этом корабле, двигаясь в этой

системе отсчета равномерно и совершая полный оборот за время $T = 10$ мин. Во сколько раз максимальная скорость дрона относительно поверхности воды выше его минимальной скорости относительно нее же?

Решение

Согласно правилу сложения скоростей, скорость $\vec{v}_{\text{дв}}$ дрона относительно воды равна (векторной) сумме его скорости $\vec{v}_{\text{дк}}$ относительно корабля и скорости $\vec{v}_{\text{кв}}$ корабля относительно воды. Поскольку направление вектора $\vec{v}_{\text{кв}}$ неизменно, а вектор $\vec{v}_{\text{дк}}$ в ходе движения дрона принимает все возможные в горизонтальной плоскости направления, обязательно найдутся такие моменты времени, когда эти два вектора сонаправлены и такие, когда они противонаправлены. Эти два случая и будут соответствовать максимальному и минимальному значениям модуля суммы этих векторов, равным $v_{max} = |\vec{v}_{\text{дк}}| + |\vec{v}_{\text{кв}}|$ и $v_{min} = ||\vec{v}_{\text{дк}}| - |\vec{v}_{\text{кв}}||$ соответственно.

Модуль вектора $\vec{v}_{\text{кв}}$ дан напрямую: он равен v. Модуль вектора $\vec{v}_{\text{дк}}$ легко получить, разделив путь дрона в СО корабля на период его обращения в этой СО

$$v_{\text{дк}} = \frac{2\pi R}{T}. \tag{2.3.93}$$

Отсюда получим окончательно

$$\frac{v_{max}}{v_{min}} = \frac{vT + 2\pi R}{|vT - 2\pi R|} \approx 3{,}68. \tag{2.3.94}$$

Погрешность 0,02.

Ответ: $3{,}68 \pm 0{,}02$.

Задача 2.3.6.3. Конденсатор (20 баллов)

Условие

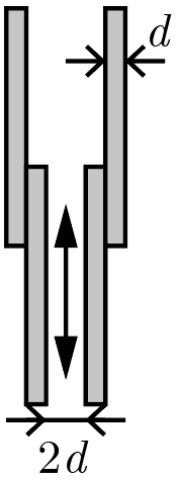


Рис. 2.3.17

Переменный конденсатор состоит из двух пар металлических пластинок толщиной d и площадью $S \gg d^2$, разделенных воздушными зазорами. При этом одна из пар (внутренняя) может частично или полностью входить в зазор другой (внешней), плотно прилегая к ней так, что электрический контакт между соответствующими пластинами никогда не нарушается, но при полном выдвижении площадь этого контакта пренебрежимо мала в сравнении с S.

Определите, во сколько раз максимальная емкость такого конденсатора превосходит минимальную, если зазор между пластинами внутренней пары имеет ширину $2d$.

Решение

Рассматриваемый конденсатор может быть представлен как батарея из двух параллельно соединенных конденсаторов, один из которых (внутренняя пара пластин) всегда имеет зазор $2d$ и площадь обкладок S, а другой (внешняя пара) имеет зазор $4d$ и площадь обкладок, которая может изменяться в пределах от 0 до S. Поскольку эти конденсаторы соединены параллельно, эквивалентная емкость батареи равна сумме их емкостей, а значит, максимальна, когда пары пластин максимально раздвинуты и минимальна, когда они полностью вдвинуты одна в другую. Используя формулу емкости плоского конденсатора, найдем, что емкость внутренней пары пластин (она же минимальная емкость всей батареи) равна

$$C_1 = \frac{S\varepsilon_0}{2d}, \tag{2.3.95}$$

где ε_0 — диэлектрическая постоянная.

Аналогично, емкость полностью выдвинутой внешней пары равна

$$C_2 = \frac{S\varepsilon_0}{4d}, \tag{2.3.96}$$

а максимальная емкость батареи составляет, соответственно, $C_1 + C_2$.

Тогда для искомого отношения максимальной и минимальной емкостей получим:

$$\frac{C_{max}}{C_{min}} = \frac{C_1 + C_2}{C_1} = \frac{\cancel{S\varepsilon_0}/(4\cancel{d}) + \cancel{S\varepsilon_0}/(2\cancel{d})}{\cancel{S\varepsilon_0}/(2\cancel{d})} = \frac{1/4 + 1/2}{1/2} = 1{,}5. \tag{2.3.97}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.6.4. Аэростат (25 баллов)

Условие

Горелка теплового аэростата способна поддерживать среднюю температуру воздуха в его оболочке не более, чем на $\Delta t = 70\,^\circ\text{C}$ выше, чем температура окружающего шар воздуха. Аэростат имеет объем $V = 645$ м^3, а общая масса его оболочки, корзины и полезной нагрузки $M = 150$ кг. Определите, при какой максимальной температуре окружающей среды аэростат сможет взлететь?

Абсолютный ноль температур $T_0 = -273\,^\circ\text{C}$, атмосферное давление $p_0 = 100\,\text{кПа}$, молярная масса воздуха $\mu = 29\,\text{г/моль}$, универсальная газовая постоянная $R = 8{,}31\,\text{Дж/(моль}\cdot\text{К)}$.

Решение

Оболочки монгольфьеров (тепловых аэростатов) представляют собой открытые сосуды, поэтому давление внутри и снаружи оболочки должно совпадать (и равняться p_0). Тогда из уравнения Менделеева – Клапейрона

$$p_0 V = \frac{m}{\mu} RT, \tag{2.3.98}$$

где T — абсолютная термодинамическая температура газа, m — его масса.

Легко выразить массу m_1 воздуха внутри оболочки и массу m_2 вытесненного атмосферного воздуха

$$\begin{aligned} m_1 &= \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \\ m_2 &= \frac{\mu p_0 V}{RT_0}, \end{aligned} \tag{2.3.99}$$

где T_0 — искомая температура окружающего воздуха.

Для того чтобы аэростат мог подняться в воздух, необходимо, чтобы вес вытесняемого им воздуха превысил его общий вес (включая вес газа в оболочке)

$$m_2 \cancel{g} = M\cancel{g} + m_1\cancel{g} \Rightarrow \frac{\mu p_0 V}{RT_0} = M + \frac{\mu p_0 V}{R(T_0 + \Delta t)}, \tag{2.3.100}$$

где g — ускорение свободного падения (сразу сокращающееся во всех слагаемых).

Домножим это выражение на $RT_0(T_0 + \Delta T)$ и получим квадратное уравнение относительно T_0

$$\mu p_0 V(\cancel{T_0} + \Delta t) = MRT_0(T_0 + \Delta t) + \cancel{\mu p_0 V T_0}. \tag{2.3.101}$$

В канонической форме:

$$T_0^2 + T_0 \Delta t - \frac{\mu p_0 V \Delta t}{MR} = 0. \tag{2.3.102}$$

Остается найти (методом дискриминанта) единственный положительный корень этого уравнения

$$T_0 = -\frac{\Delta t}{2} + \frac{1}{2}\sqrt{\Delta t^2 + \frac{4\mu p_0 V}{MR}\Delta t} \approx 291\,\text{К} \approx 18\,^\circ\text{C}. \tag{2.3.103}$$

Погрешность $0{,}1\,^\circ\text{C}$.

Ответ: $(18{,}0 \pm 0{,}1)\,^\circ\text{C}$.

Задача 2.3.6.5. Спутник (30 баллов)

Условие

Спутник, движущийся вокруг Земли по высокой круговой орбите, перевели на другую круговую орбиту, в результате чего его кинетическая энергия увеличилась на 5%. На сколько процентов увеличился модуль потенциальной энергии взаимодействия спутника с планетой, если она считается равной нулю на бесконечном удалении от планеты?

Решение

Обозначим радиус орбиты спутника R, его орбитальную скорость v, его массу m, а массу планеты M. На спутник действует сила всемирного тяготения

$$F = G\frac{mM}{R^2}, \tag{2.3.104}$$

где G — гравитационная постоянная, связанная с центростремительным ускорением v^2/R спутника вторым законом Ньютона

$$\cancel{m}\frac{v^2}{\cancel{R}} = G\frac{\cancel{m}M}{R^{\cancel{2}}}. \tag{2.3.105}$$

Из этого выражения легко видеть, что квадрат орбитальной скорости спутника v^2 обратно пропорционален радиусу орбиты. Разумеется, кинетическая энергия спутника $mv^2/2$ прямо пропорциональна этому квадрату скорости и, следовательно, тоже обратно пропорциональна R.

Чтобы понять, как потенциальная энергия спутника зависит от радиуса его орбиты, проще всего обратить внимание на аналогию между гравитацией и электростатическими силами. Сила Кулона взаимодействия двух точечных зарядов зависит от расстояния между ними и обратно пропорциональна квадрату разделяющего их расстояния, точно так же, как сила всемирного тяготения. Одновременно потенциальная энергия взаимодействия этих зарядов обратно пропорциональна первой степени расстояния между ними, следовательно, то же справедливо и для гравитации. В результате видно, что модуль потенциальной энергии обратно пропорционален R, как и кинетическая энергия. Следовательно, при изменении орбиты спутника он изменится ровно во столько же раз.

Погрешность $0{,}01\%$.

Ответ: $(5{,}00 \pm 0{,}01)\%$.

2.3.7. Четвертая волна. Задачи 8–9 класса

Задачи четвертой волны предметного тура по физике за 8–9 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63466/enter/.

Задача 2.3.7.1. Расплав (10 баллов)

Условие

Расплавленная соль предлагается как эффективный аккумулятор тепла для некоторых типов теплоцентралей. Удельная теплота плавления и кристаллизации соли $\lambda = 28{,}7$ кДж/кг, ее теплоемкость в твердой форме $c_1 = 0{,}92$ кДж/(кг°C), а в жидкой — $c_2 = 1{,}5$ кДж/(кг°C), температура ее плавления $\theta = 800$ °C. Определите, какую массу соли необходимо взять, чтобы при ее нагреве от $t_0 = 20$ °C до $t_1 = 1200$ °C запасти $Q = 1$ МДж тепла.

Решение

Количество теплоты, требуемое на нагрев твердой соли до температуры плавления, задается выражением

$$Q_1 = c_1 m(\theta - t_0), \tag{2.3.106}$$

где m — масса нагреваемой соли.

Количество теплоты, уходящее непосредственно на плавление

$$Q_2 = \lambda m. \tag{2.3.107}$$

Наконец, количество теплоты, уходящее на нагрев расплава соли,

$$Q_3 = c_2 m(t_1 - \theta). \tag{2.3.108}$$

Складывая все эти порции тепла, получим, что общая запасаемая теплота

$$Q = Q_1 + Q_2 + Q_3 = m(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta)), \tag{2.3.109}$$

откуда окончательно

$$m = \frac{Q}{(c_1(\theta - t_0) + \lambda + c_2(t_1 - \theta))} \approx 743\,\text{г}. \tag{2.3.110}$$

Погрешность 1 г.

Ответ: (743 ± 1) г.

Задача 2.3.7.2. Катафот (15 баллов)

Условие

Катафот представляет собой два одинаковых квадратных зеркала, соединенных общей гранью под прямым углом друг к другу. При падении на него видимого света каждое зеркало поглощает $\eta = 20\%$ достигающей его световой энергии, а остальную — отражает. Параллельно биссектрисе образованного зеркалами угла в плоскости, перпендикулярной к их общему ребру, на середину одного из зеркал падает узкий лазерный луч, переносящий мощность $P = 5$ мВт. Какое количество световой энергии поглотит второе зеркало за $\tau = 10$ с?

Решение

Прежде всего отметим, что любой луч, падающий на катафот параллельно его биссектрисе, отразится последовательно от обоих зеркал катафота, как изображено на рис 2.3.18. При этом после первого отражения мощность луча снизится в $(1 - \eta)$ раз, и доля η от этой оставшейся мощности будет поглощена вторым зеркалом. В результате связь между изначальной P и поглощаемой P_1 мощностями имеет следующий вид:

$$P_1 = (1 - \eta)\eta P. \tag{2.3.111}$$

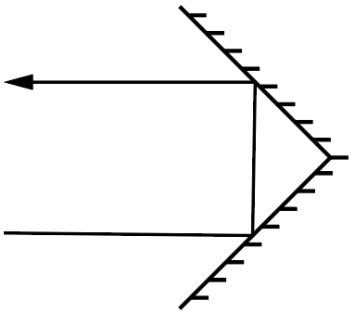


Рис. 2.3.18

Теперь остается лишь вспомнить определение мощности, как отношения энергии (в данном случае переносимой лазерным лучом или поглощаемой зеркалом) ко времени, чтобы получить окончательный ответ

$$Q = P_1\tau = (1 - \eta)\eta P\tau \approx 8\,\text{мДж}. \tag{2.3.112}$$

Погрешность $0{,}1$ мДж.

Ответ: $(8{,}0 \pm 0{,}1)$ мДж.

Задача 2.3.7.3. Соты (20 баллов)

Условие

Композитный материал изготавливают, вырезая из алюминия (плотность $\rho_1 = 2{,}7\,\text{г/см}^3$) строго периодическую вдоль двух взаимно перпендикулярных осей квадратную сетку с толщиной стенки d и длиной внутренней стороны ячейки $4d$, фрагмент которой изображен на рис. 2.3.19. Затем полости заполняют смолой, после затвердевания имеющей плотность $\rho_2 = 1{,}2\,\text{г/см}^3$. Найдите среднюю плотность большого листа из такого материала.

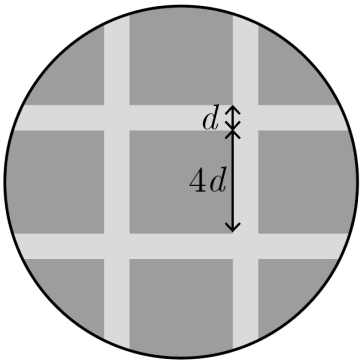


Рис. 2.3.19

Решение

По мере увеличения размеров листа материала роль его краев в общей плотности постепенно снижается, поэтому среднюю плотность большого листа следует вычислять как среднюю плотность одного элемента периодичности, границы которого изображены пунктиром на рис. 2.3.20. Объем V этого элемента равен $25dh$, где h — толщина листа материала.

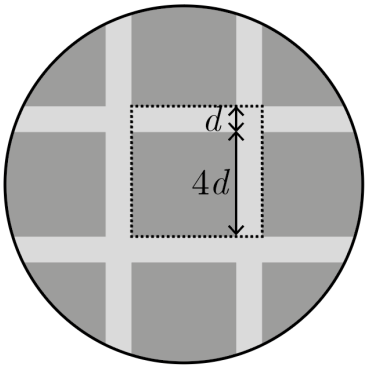


Рис. 2.3.20

Массу элемента найдем, сложив массы алюминия (индексы 1) и смолы (индексы 2)

$$m = m_1 + m_2 = \rho_1 V_1 + \rho_2 V_2 = \rho_1 9dh + \rho_2 16dh. \tag{2.3.113}$$

Тогда искомая плотность окончательно равна

$$\rho = \frac{m}{V} = \frac{\rho_1 9\cancel{dh} + \rho_2 16\cancel{dh}}{25\cancel{dh}} \approx 1{,}74\,\text{г/см}^3. \tag{2.3.114}$$

Погрешность $0{,}01\,\text{г/см}^3$.

Ответ: $(1{,}74 \pm 0{,}01)\,\text{г/см}^3$.

Задача 2.3.7.4. Домкрат (25 баллов)

Условие

На рис. 2.3.21 приведена схема устройства гидравлического домкрата. Его поршни представляют собой цилиндры с радиусами $R = 21\,\text{см}$ и $r = 3\,\text{см}$. Чтобы поднимать при помощи этого домкрата груз $m = 1{,}4\,\text{т}$, установленный на платформе большого цилиндра, к точке A рычага необходимо приложить силу не менее $F = 87{,}5\text{Н}$. Определите отношение $AB : BC$. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$. На рисунке точный масштаб не сохранен.

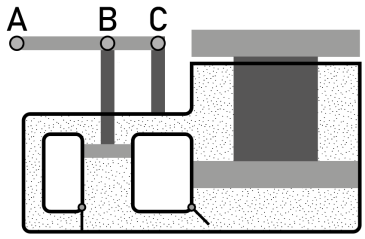


Рис. 2.3.21

Решение

Изображенный домкрат дает выигрыш в силе благодаря двум механизмам: рычагу и гидравлическому прессу. Выигрыш в силе, обеспечиваемый прессом, равен отношению площадей его цилиндров

$$\frac{mg}{F_1} = \frac{\pi R^2}{\pi r^2} \Rightarrow F_1 = mg\frac{r^2}{R^2}, \tag{2.3.115}$$

где F_1 — сила давления малого поршня.

В свою очередь, выигрыш в силе, обеспечиваемый рычагом, равен отношению его плеч, но так как рычаг закреплен в точке C, нужно сравнивать плечи AC и BC

$$\frac{F_1}{F} = \frac{AC}{BC} = \frac{AB + BC}{BC} = 1 + \frac{AB}{BC}. \tag{2.3.116}$$

Совместив эти уравнения, получим окончательно

$$\frac{AB}{BC} = \frac{F_1}{F} - 1 = \frac{mgr^2}{FR^2} - 1 = 2{,}2. \tag{2.3.117}$$

Погрешность $0{,}1$.

Ответ: $2{,}2 \pm 0{,}1$.

Задача 2.3.7.5. Номинальная мощность (30 баллов)

Условие

Изучая электронагреватель прямого действия, ученик заметил, что увеличение подаваемой на него силы тока на $\Delta I = 0{,}1\,\text{А}$ над номинальным значением приводит к увеличению тепловой мощности, выделяемой прибором, на $\Delta P_1 = 44\,\text{Вт}$,

а уменьшение силы тока на ту же величину от номинальной, приводит к уменьшению мощности на $\Delta P_2 = 36\,\text{Вт}$. Найдите номинальную мощность прибора, считая его сопротивление независимым от температуры.

Решение

Согласно закону Джоуля – Ленца, тепловая мощность электронагревателя равна

$$P = I^2 R, \tag{2.3.118}$$

где I — сила пропускаемого через него тока, а R — сопротивление прибора. Последнее по условиям задачи можно считать неизменным, поэтому, введя обозначения I_0 для номинальной силы тока и P_0 для номинальной мощности прибора, можно составить пропорции

$$\begin{cases} \dfrac{P_0 + \Delta P_1}{P_0} = \left(\dfrac{I_0 + \Delta I}{I_0}\right)^2, \\ \dfrac{P_0 - \Delta P_2}{P_0} = \left(\dfrac{I_0 - \Delta I}{I_0}\right)^2. \end{cases} \tag{2.3.119}$$

Обозначив $\frac{\Delta I}{I_0}$ буквой x и частично сократив дроби, приведем их к виду

$$\begin{cases} 1 + \dfrac{\Delta P_1}{P_0} = (1+x)^2, \\ 1 - \dfrac{\Delta P_2}{P_0} = (1-x)^2. \end{cases} \tag{2.3.120}$$

Эту систему можно решить, вычитая второе уравнение из первого

$$\frac{\Delta P_1 + \Delta P_2}{P_0} = 4x \Rightarrow x = \frac{\Delta P_1 + \Delta P_2}{4P_0} \tag{2.3.121}$$

и подставляя результат в любое уравнение системы (2.3.120)

$$\cancel{1} + \frac{\Delta P_1}{P_0} = \cancel{1} + \frac{\Delta P_1 + \Delta P_2}{2P_0} + \frac{(\Delta P_1 + \Delta P_2)^2}{16P_0^2}. \tag{2.3.122}$$

Домножим на $16P_0^2$

$$16\Delta P_1 P_0 = 8P_0(\Delta P_1 + \Delta P_2) + (\Delta P_1 + \Delta P_2)^2. \tag{2.3.123}$$

и выразим окончательно

$$P_0 = \frac{(\Delta P_1 + \Delta P_2)^2}{8(\Delta P_1 - \Delta P_2)} = 100\,\text{Вт}. \tag{2.3.124}$$

Погрешность 1 Вт.

Ответ: 100 ± 1 Вт.

2.3.8. Четвертая волна. Задачи 10–11 класса

Задачи четвертой волны предметного тура по по физике за 10–11 класс открыты для решения. Соревнование доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/63483/enter/.

Задача 2.3.8.1. Шестеренки (10 баллов)

Условие

В сложной трансмиссии две шестеренки А и Б вращаются в различных частях механизма так, что угловая скорость вращения шестеренки А в 3 раза выше, чем шестеренки Б, но линейная скорость зубцов шестеренки Б в 2 раза выше, чем шестеренки А. Найдите отношение центростремительного ускорения зубцов шестеренки А к центростремительному ускорению зубцов шестеренки Б.

Решение

Два хорошо известных выражения для центростремительного ускорения a

$$a = \frac{v^2}{R} = \omega^2 R, \tag{2.3.125}$$

где R — радиус траектории, v — линейная скорость, ω — угловая скорость. Перемножив эти выражения, получим

$$a^2 = \frac{v^2}{\cancel{R}}\omega^2\cancel{R} \Rightarrow a = v\omega. \tag{2.3.126}$$

Таким образом, центростремительное ускорение равно произведению линейной скорости на угловую, из чего следует

$$\frac{a_\text{А}}{a_\text{Б}} = \frac{v_\text{А}}{v_\text{Б}} \cdot \frac{\omega_\text{А}}{\omega_\text{Б}} = \frac{1}{2} \cdot \frac{3}{1} = 1{,}5. \tag{2.3.127}$$

Погрешность 0,01.

Ответ: $1{,}50 \pm 0{,}01$.

Задача 2.3.8.2. Маятник (15 баллов)

Условие

Заряженный металлический шарик закреплен на конце тонкой шелковой нити и несет заряд $q = 20$ мкКл. Другой конец нити закреплен к потолку. Определите массу шарика, если при помещении такого маятника в однородное электрическое поле, вектор напряженности которого направлен строго горизонтально и равен по модулю $E = 2{,}5$ кВ/м, сила натяжения нити после установления равновесия оказывается равна $T = 130$ мН. Ускорение свободного падения $g = 9{,}8$ м/с^2.

Решение

На шарик действуют три силы: горизонтально направленная сила электростатического взаимодействия $q\vec{E}$, вертикально вниз направленная сила тяжести $m\vec{g}$ и направленная вдоль нити сила ее натяжения $\vec{T}$. Разумеется, маятник может быть в равновесии, только если векторная сумма этих сил равна нулю, то есть эти три вектора образуют замкнутый треугольник

$$q\vec{E} + m\vec{g} + \vec{T} = \vec{0}. \tag{2.3.128}$$

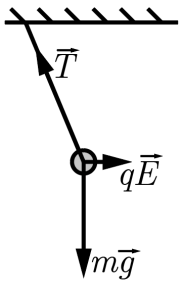


Рис. 2.3.22

Поскольку направления $\vec{E}$ и $\vec{g}$ известны, этот треугольник прямоугольный, а $\vec{T}$ — его гипотенуза. Тогда из теоремы Пифагора

$$q^2E^2 + m^2g^2 = T^2 \Rightarrow m = \frac{\sqrt{T^2 - q^2E^2}}{g} \approx 12{,}2\,\text{г}. \tag{2.3.129}$$

Погрешность $0{,}2$ г.

Ответ: $(12{,}2 \pm 0{,}2)$ г.

Задача 2.3.8.3. Автопилот (20 баллов)

Условие

Автомобиль с автопилотом запрограммирован таким образом, что при движении по прямой на трассе он всегда старается поддерживать дистанцию между собой и движущимся непосредственно перед ним автомобилем ровно такой же, как между собой и движущимся непосредственно за ним автомобилем. В некоторый момент движения скорости всех трех этих автомобилей были равны. Определите ускорение автомобиля, движущегося непосредственно за автопилотируемым, если модули ускорения самого автопилотируемого автомобиля и движущегося непосредственно перед ним в этот момент оба оказались равны $a = 0{,}2\,\text{м/с}^2$, но дистанция между ними при этом начала сокращаться. Колеса автомобилей движутся без проскальзывания, и автопилоту удается соблюдать требования своей программы.

Решение

Программа автомобиля означает, что (до тех пор, пока это позволяет мощность двигателя и сцепление колес) координата x вдоль оси, совпадающей с дорогой, ав-

томобиля с автопилотом равна среднему арифметическому координат x_1 идущего впереди и x_2 идущего позади

$$x = \frac{x_1 + x_2}{2}. \tag{2.3.130}$$

Поскольку такая кинематическая связь справедлива в любые два момента времени t_1 и t_2, для средней скорости v автомобиля на автопилоте в проекции на $0x$ на любом промежутке времени можно записать

$$v_x = \frac{x(t_2) - x(t_1)}{t_2 - t_1} = \frac{x_1(t_2) + x_2(t_2) - x_1(t_1) - x_2(t_1)}{2(t_2 - t_1)} = \frac{v_{x1} + v_{x2}}{2}, \tag{2.3.131}$$

где использована та же система индексов. Таким образом, средняя скорость на любом промежутке времени и, следовательно, мгновенная скорость в любой момент времени автомобиля на автопилоте в проекции на $0x$ равна среднему арифметическому мгновенной скорости в этой же проекции впереди и позади идущих автомобилей. Повторение этих рассуждений приводит к аналогичному результату для ускорений

$$a_x = \frac{a_{x1} + a_{x2}}{2}. \tag{2.3.132}$$

Выразим из этого уравнения проекцию на $0x$ искомого ускорения замыкающего автомобиля a_{x2}

$$a_{x2} = 2a_x - a_{x1}. \tag{2.3.133}$$

По условиям задачи в рассматриваемый момент скорости всех трех автомобилей равны, а ускорения a_1 и a совпадают по модулю, но дистанция начинает сокращаться. Это возможно только если передний автомобиль тормозит — имеет отрицательную проекцию ускорения на направление движения, а автопилотируемый автомобиль, напротив, ускоряется (имеет положительную проекцию). Тогда напрямую из (2.3.133) получим

$$a_{x2} = 2a_x - a_{x1} = 2a - (-a) = 3a = 0{,}6\,\text{м/с}^2. \tag{2.3.134}$$

Погрешность $0{,}01\,\text{м/с}^2$.

Ответ: $(0{,}60 \pm 0{,}01)\,\text{м/с}^2$.

Задача 2.3.8.4. Лифт (25 баллов)

Условие

В лифте, движущемся вверх с некоторым ускорением a, сонаправленным его скорости, уронили без начальной скорости относительно лифта мячик с высоты $h_1 = 0{,}6\,\text{м}$ над уровнем пола лифта. Мячик абсолютно упруго ударился о пол, но своим ударом спровоцировал срабатывание системы аварийной остановки, в результате чего в момент удара ускорение лифта резко поменяло направление на противоположное, а его модуль возрос втрое. После отскока мячик поднялся до высоты $h_2 = 1{,}8\,\text{м}$. Определите a. Ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

Решение

В системе отсчета, связанной с лифтом, начальное ускорение мяча равно $g + a$. Проходя с этим ускорением расстояние h_1, мяч приобретает скорость (относительно лифта) v, которую легко вычислить из соотношения

$$\frac{v^2}{2} = (g + a)h_1. \tag{2.3.135}$$

В момент удара резко меняется ускорение, но не скорость лифта, поэтому значение v при абсолютно упругом ударе по модулю остается неизменным.

В процессе подъема мяч уже имеет относительно лифта ускорение $g - 3a$, что позволяет записать аналогично

$$\frac{v^2}{2} = (g - 3a)h_2. \tag{2.3.136}$$

Совмещая эти равенства, получим окончательно

$$(g + a)h_1 = (g - 3a)h_2 \Rightarrow a(h_1 + 3h_2) = g(h_2 - h_1) \Rightarrow a = g\frac{h_2 - h_1}{h_1 + 3h_2} = 1{,}96\,\text{м/с}^2. \tag{2.3.137}$$

Погрешность $0{,}02$ м/с^2.

Ответ: $(1{,}96 \pm 0{,}02)$ м/с^2.

Задача 2.3.8.5. Эффект Лейденфроста (30 баллов)

Условие

Капля воды при температуре немного ниже температуры кипения упала на раскаленную поверхность, в результате чего $\alpha = 10^{-5}$ ее массы практически мгновенно испарилось. Известно, что $\eta = 3\%$ полученной каплей энергии пошло на работу расширяющегося пара над оставшейся частью капли.

На какую высоту «подпрыгнет» капля вертикально вверх в результате такого испарения, если сопротивлением воздуха ее движению, а также потерями тепла в окружающую среду и работой пара против воздуха можно пренебречь?

Удельная теплота парообразования воды равна $L = 2{,}26$ МДж/кг, ускорение свободного падения $g = 9{,}8\,\text{м/с}^2$.

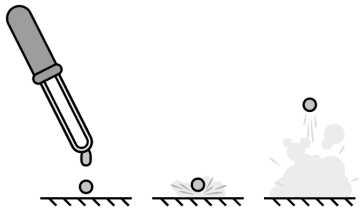


Рис. 2.3.23

Решение

Согласно первому началу термодинамики та теплота, полученная каплей, которая не пошла на работу расширяющегося пара над жидкой частью капли, ушла на изменение ее внутренней энергии; в данном случае — на испарение. Таким образом, можно записать для для этой части энергии

$$\Delta U = (1-\eta)Q = Lm\alpha, \tag{2.3.138}$$

где m — масса капли до испарения, а Q — общая полученная каплей теплота. Отсюда легко найти Q

$$Q = \frac{Lm\alpha}{1-\eta}. \tag{2.3.139}$$

В то же время работа пара над каплей полностью идет на увеличение ее механической энергии, которая в верхней точке траектории чисто потенциальна

$$A = \eta Q = (1-\alpha)mgh. \tag{2.3.140}$$

Выражая из этого уравнения h и подставляя в него Q, получаем

$$h = \frac{\eta Q}{(1-\alpha)mg} = \frac{L\alpha\eta}{(1-\alpha)(1-\eta)g} \approx 7{,}1\,\text{см}. \tag{2.3.141}$$

Разумеется, если величину $1-\alpha$ в этом выражении считать просто единицей, ответ не изменится в пределах любой разумной погрешности.

Погрешность $0{,}5$ см.

Ответ: $(7{,}1 \pm 0{,}5)$ см.

2.4. Инженерный тур

2.4.1. Задачи по компетенции Программирование микроконтроллеров

Этот небольшой курс посвящен началу работы с микроконтроллерами серии `STM32`. В курсе содержится необходимая теоретическая информация, а также вопросы для самопроверки, которые не дают баллов, но помогают проверить полученные знания. Их можно использовать для самоподготовки.

Чтобы лучше усвоить курс, желательно немного знать язык программирования C/C++. Также для более полного погружения в курс желательно скачать и установить на ПК программное обеспечение для работы с микроконтроллерами семейства `STM32` — `STM32CubeIDE`. Скачать ПО можно по ссылке: https://disk.yandex.ru/d/Xa1IOzaG2j__7Q.

Программирование микроконтроллеров `STM32` — неотъемлемая компетенция профиля Спутниковые системы, и готовиться к ней можно начать уже сейчас. Умение работать с микроконтроллерами пригодится на заключительном этапе для ролей программист микроконтроллеров и инженер связи. К тому же `STM32` — одна из самых востребованных серий среди современных микроконтроллеров.

Тест сделан по версии `STM32CubeIDE v1.11`.

Задания-тесты созданы в большей степени в виде гайда, который предлагается пройти, чтобы познакомиться с ПО. Тесты несложны, но для ответа на вопросы необходимо выполнить несколько последовательных действий в самой IDE. Чтобы прохождение гайдов не вызывало больших затруднений, желательно иметь представлении о программировании на C/C++ и начальное понимание о работе микроконтроллеров.

Задача 2.4.1.1. Микроконтроллер — маленький цифровой повелитель (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Наверное, слово «микроконтроллер» хоть раз было у каждого на слуху. И не без причины, ведь сейчас микроконтроллеры используются в большинстве современных устройств.

Микроконтроллер представляет собой небольшую микросхему, однако в устройствах, от которых не требуется сложных вычислительных операций, именно он играет управляющую роль. Прежде всего микроконтроллеры (как видно из их названия) предназначены для управления периферийными устройствами, такими как различные сенсоры, двигатели, сервоприводы и пр.

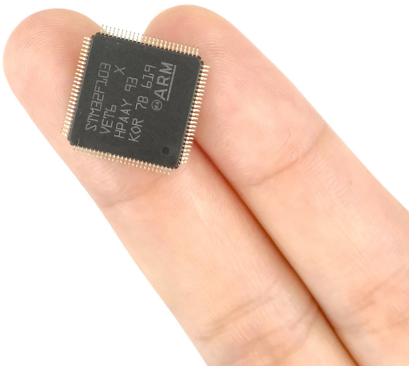


Рис. 2.4.1

Микроконтроллеров существует множество, однако в заданиях уделяется внимание серии `STM32`, а точнее, микроконтроллеру `STM32F103`. В упрощенном виде его структура представлена на схеме ниже (структура других микроконтроллеров будет немного отличаться, однако общие принципы почти у всех аналогичны).

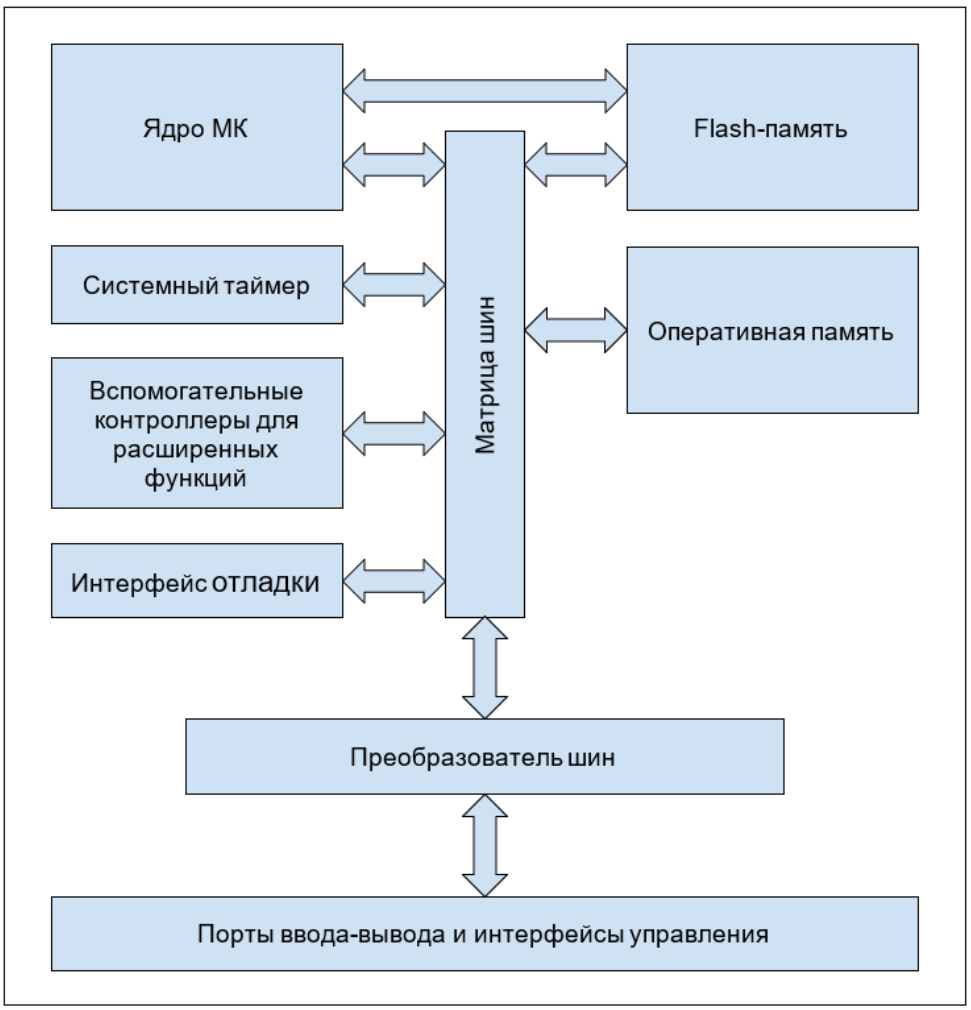


Рис. 2.4.2

Ядро микроконтроллера управляет входящими запросами и доступом к памяти.

Во flash-память пользователь может записать свой программный код, состоящий из базовых команд, который с помощью микроконтроллера будет управлять подключенными к нему устройствами.

Базовые команды определяются архитектурой микроконтроллера. В большинстве случаев код для микроконтроллера пишется на языках Assembler или C/C++, однако встречаются и другие варианты.

Перед загрузкой на микроконтроллер написанный код компилируется, то есть проверяется на синтаксические ошибки, переводится в массив из 0 и 1 и собирается определенным образом. После этого код передается на ядро микроконтроллера по соответствующему интерфейсу (об этом будет сказано подробнее дальше) и записывается определенным образом в память микроконтроллера.

В области flash-памяти есть область, в которой находится загрузчик (bootloader), позволяющий многократно перезаписывать программу пользователя во flash-память. В настройках микроконтроллера по умолчанию эта область памяти остается неперезаписываемой.

Матрица шин контролирует передачу данных в необходимые области. Также по ней передаются данные от микроконтроллера для периферийных устройств и обратно. Преобразователь шин нужен, чтобы микроконтроллер смог подстроиться под использование того интерфейса, по которому будет общаться с периферийным устройством.

Помимо возможности записи управляющей программы во flash-память, существует также опция записать ее в оперативную память. Это может понадобиться в тех случаях, когда необходимо, чтобы код отработал всего один раз: при перезагрузке микроконтроллера оперативная память очистится, в отличие от flash-памяти.

Системный таймер, прежде всего, нужен для отсчета времени самого микроконтроллера и синхронизации различных процессов: при работе с некоторыми устройствами синхронизация крайне важна. Тактирование — отдельная большая тема в микроконтроллерах, к которой мы вернемся чуть позже.

В зависимости от сложности периферийного устройства, для его управления может понадобиться как простой порт ввода-вывода общего назначения, представляющий собой всего один контакт с регулируемым уровнем напряжения, так и более сложный определенный интерфейс информационного обмена, состоящий из нескольких линий (несколько из них будет рассматриваться дальше). Микроконтроллеры, как правило, поддерживают основные способы информационного обмена. В программе управления, записываемой во flash-память, можно инициализировать тот или иной способ информационного обмена.

Здесь дано краткое описание структуры микроконтроллера серии STM32. Более подробно можно ознакомиться по ссылке: https://robocraft.ru/arm/711.

Условие

Космокот Кас решил загрузить программу на свой новый, только что приобретенный микроконтроллер. Однако ничего не получилось. Если учесть, что у космокота все подключено правильно, то где возможна ошибка в самом микроконтроллере? Выберите один вариант ответа, самый вероятный:

1. загрузчик;
2. оперативная память;
3. порты ввода-вывода;
4. матрица шин.

Ответ: 1.

Задача 2.4.1.2. Память микроконтроллера — это не монолит (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Память микроконтроллера — не монолитное пространство. Она разделена на множество ячеек, называемых регистрами. В случае с `STM32` регистр — это ячейка памяти длиной 32 б. С помощью регистров происходит управление портами ввода-вывода, интерфейсами, таймерами и прочим.

Каждый регистр имеет свой адрес. Чтобы осуществить управление нужным узлом (например, включить или выключить его), необходимо записать в регистр определенную комбинацию из нулей и единиц. А чтобы, например, узнать состояние интересующего узла, нужно считать значение соответствующего регистра.

Чтобы осуществлять чтение или запись в регистр, необходимо знать его адрес, по которому к нему можно обратиться. Весь перечень регистров, их назначение и то, какая последовательность нулей и единиц задает определенное управление, описывается разработчиком микроконтроллера в документации (`register map`, или карте регистров). Ниже для примера приведен скриншот из даташита на микроконтроллер `STM32F103C8`, на котором в первой колонке указаны адреса регистров, а во второй колонке — узел, которым они управляют (`ADC` — аналого-цифровой преобразователь, `GPIO` — порты ввода/вывода).

0x4001 2800 - 0x4001 2BFF	ADC2
0x4001 2400 - 0x4001 27FF	ADC1
0x4001 2000 - 0x4001 23FF	GPIO Port G
0x4001 1C00 - 0x4001 1FFF	GPIO Port F
0x4001 1800 - 0x4001 1BFF	GPIO Port E
0x4001 1400 - 0x4001 17FF	GPIO Port D
0x4001 1000 - 0x4001 13FF	GPIO Port C
0x4001 0C00 - 0x4001 0FFF	GPIO Port B
0x4001 0800 - 0x4001 0BFF	GPIO Port A

На схеме (рис. 2.4.3) представлено более подробное устройство памяти микроконтроллера.

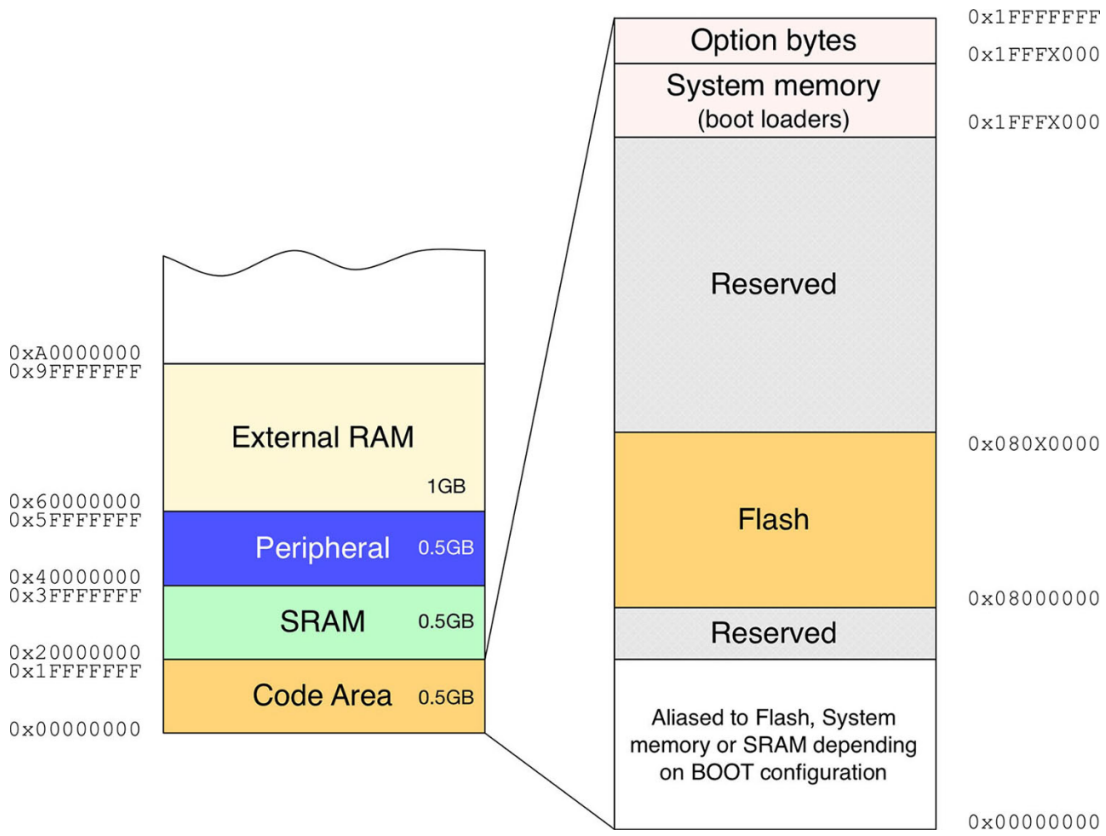


Рис. 2.4.3

Область `flash`-памяти занимает адреса с `0x00000000` до `0x1FFFFFFF`. На адресах `0x1FFFX000` находится загрузчик. По умолчанию программный код пользователя загружается в пространство памяти с адресами `0x08000000`–`0x080X000`.

Обычно для управления и настройки одного узла используется несколько регистров. Чтобы с самими регистрами было удобнее работать, им дают имена (и не только самим регистрам, но и битам в регистрах тоже). Эти имена можно найти все также в даташите. Рассмотрим это на примере системного таймера, который упоминался раньше. Для его настройки используется всего четыре регистра.

Адрес регистра	**Имя регистра**	**Назначение регистра**
`0xE000E010`	`SYST_CSR`	Регистр управления и статуса
`0xE000E014`	`SYST_RVR`	Регистр, хранящий загружаемое в таймер значение (максимально возможное количество тиков), от которого он считает до нуля, после чего перезагружается
`0xE000E018`	`SYST_CVR`	Регистр, хранящий текущее значение счетчика системного таймера
`0xE000E01C`	`SYST_CALIB`	Регистр калибровки

Работу таймера можно настроить на разную частоту, и от этого будет зависеть, сколько отсчетов он будет делать в секунду. Некоторые программные операции выполняются не просто за доли секунд, а за всего за несколько микросекунд. И чтобы обеспечить необходимую точность, таймер должен работать с нужной частотой.

Один из битов регистра `SYST_CSR` как раз позволяет настроить точность работы таймера.

Чтобы узнать, сколько времени прошло с момента запуска работы микроконтроллера, следует считать значение с регистра `SYST_CVR`.

Более подробно о системном таймере можно почитать по ссылкам:

- http://easyelectronics.ru/arm-uchebnyj-kurs-systick-sistemnyj-tajmer.html.
- https://www.rotr.info/electronics/mcu/arm_systick.htm.

Обычно существует стандартный набор библиотек, который позволяет упростить процесс программирования и не писать код управления микроконтроллером, непосредственно задавая значения регистров: вместо этого используются типовые функции. Однако иногда прямого обращения к регистрам не избежать, поэтому необходимо понимать, что это такое. При компиляции программа управления приобретает бинарный или шестнадцатибитный формат, который и позволяет записать в микроконтроллер всю необходимую информацию в соответствующие регистры. Внизу на скриншоте показано, как выглядит сформированный файл программы в том виде, как он будет записан в регистры микроконтроллера.

Рис. 2.4.4

Обратите внимание на то, что, как и говорилось выше, первый доступный для записи адрес `flash`-памяти — `0x08000000`.

Более подробно о регистрах можно почитать по ссылке (если материал оказался сложным, можно вернуться к нему чуть позже, после прохождения следующего раздела): https://habr.com/ru/post/407083/.

Условие

Какое значение и в какой регистр таймера нужно записать космокоту Касу, если ему необходимо, чтобы таймер обнулялся каждые 15 363 тиков? Ответ укажите в шестнадцатеричной системе счисления (без указателя 0x, незначащих нулей

и пробелов).

Подсказка: к записываемому в соответствующий регистр значению всегда автоматически будет добавляться. Учтите это при выборе ответа.

Ответ: `E000E014 <- 4E90`.

Задача 2.4.1.3. Управлять и властвовать (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

В программе управления, если сильно упрощать, можно выделить две основные части: часть для преднастройки параметров микроконтроллера и основную исполняемую часть, которая выполняется циклически, пока на микроконтроллере есть питание.

Для разработки программ для микроконтроллеров используют специальные среды разработки (IDE — integrated development environment). Для `STM32` можно использовать несколько таких IDE, но впоследствии мы преимущественно будем использовать официальную `STM32CubeIDE`. Для выполнения следующих заданий нелишним будет ее установить. Напомним, что ПО можно скачать по ссылке: https://disk.yandex.ru/d/Xa1IOzaG2j__7Q.

После установки `STM32CubeIDE` запустите ее. На экране появится форма для выбора рабочего пространства. Можно выбрать его по умолчанию, предлагаемое самой программой. Здесь будут храниться проекты участников.

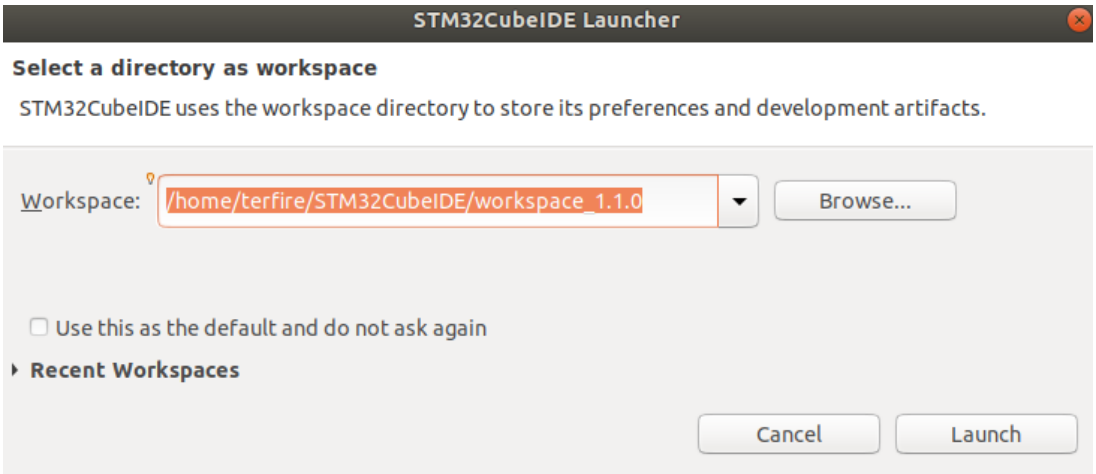


Рис. 2.4.5

В открывшемся окне в боковом меню выберите **Start New STM32 project**. Далее на экране появится список выбора микроконтроллера или отладочной платы для нового проекта, в первой вкладке которого предлагается выбрать тип микроконтроллера, а во второй — название отладочной платы. Выберите из списка микроконтроллер, с которым предстоит работать: **stm32f103c8t6**.

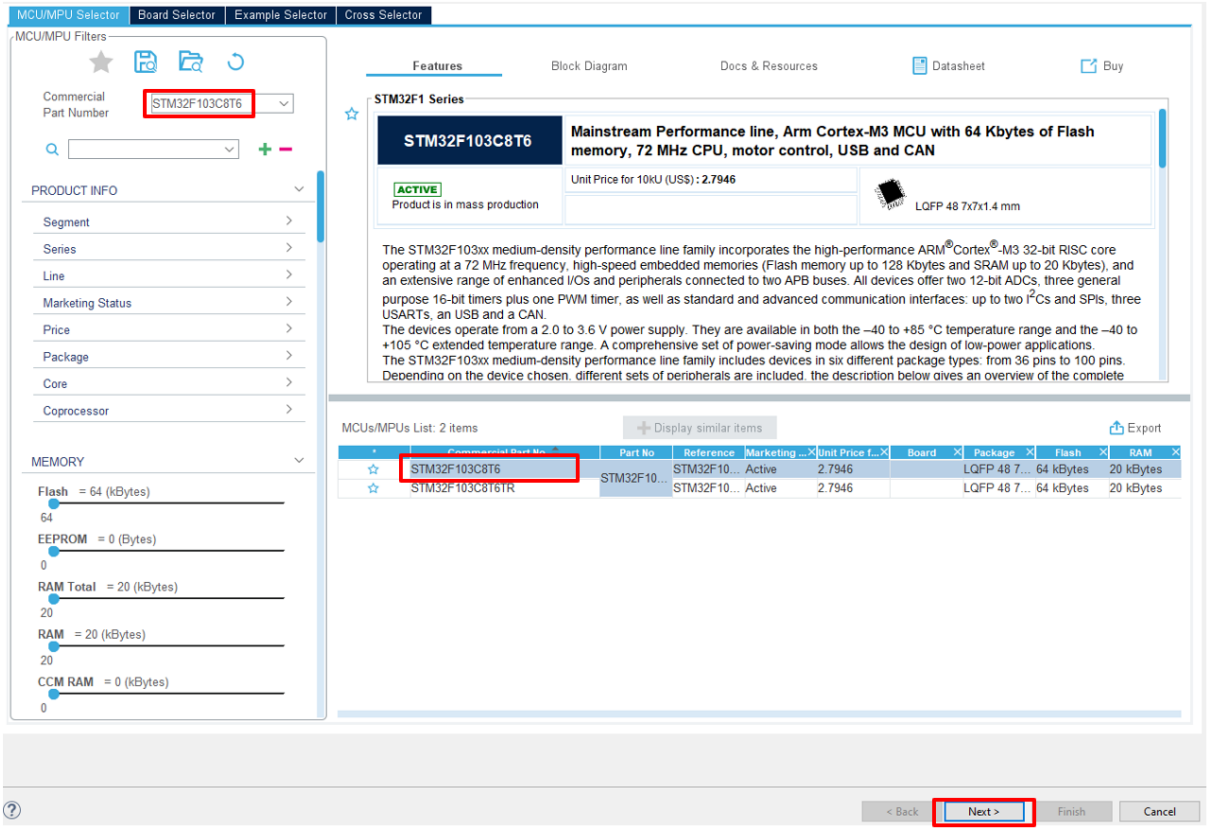


Рис. 2.4.6

Далее необходимо будет ввести название проекта и нажать **Finish**.

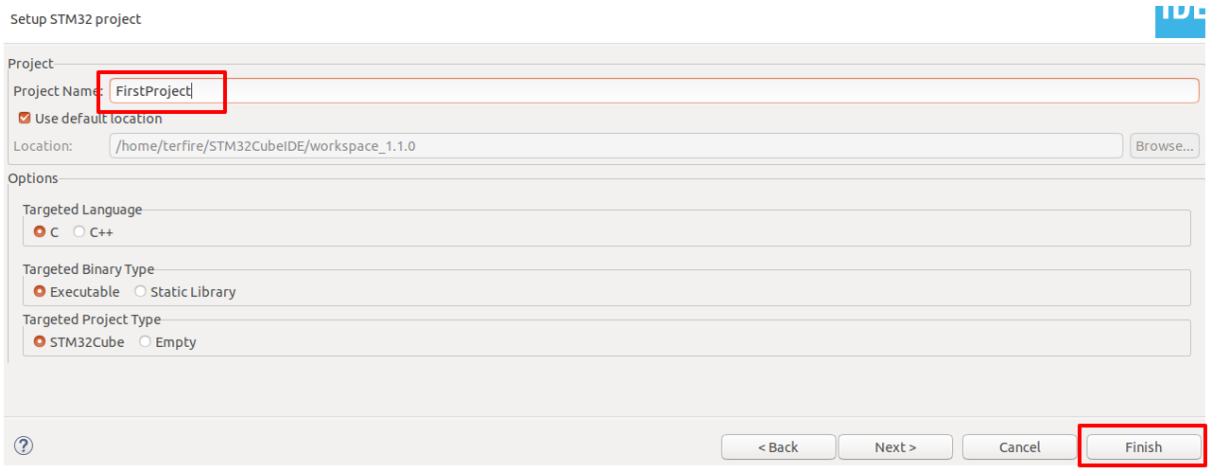


Рис. 2.4.7

После этого `STM32CubeIDE` предложит подкачать необходимые для работы файлы. Согласитесь на подкачку файлов, необходимо будет некоторое время подождать (лучше использовать `vpn`). Затем откроется файл с расширением `.ioc`, в котором будет представлен выбранный микроконтроллер.

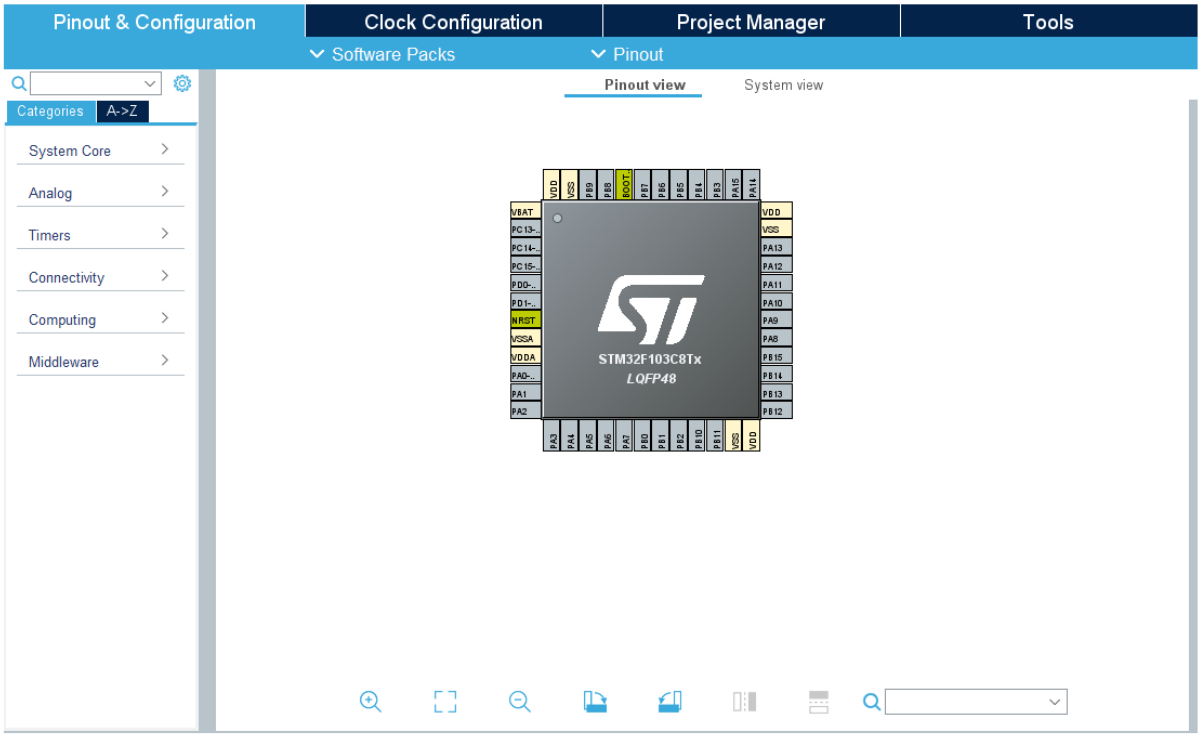


Рис. 2.4.8

Почти каждый вывод микроконтроллера (пин) на представленной схеме можно настроить. Для этого нужно кликнуть по выводу левой кнопкой мыши и выбрать нужный режим настройки вывода. К настройке пинов вернемся чуть позднее. Пока просто соберем пустой проект и посмотрим на него. Для этого нажмите на значок шестеренки в верхней панели меню. Это действие сгенерирует код в соответствии с настройками по умолчанию.

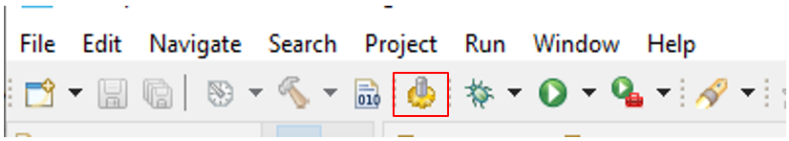


Рис. 2.4.9

Генерация проекта займет немного времени. После того как она закончится, слева отобразится дерево проекта.

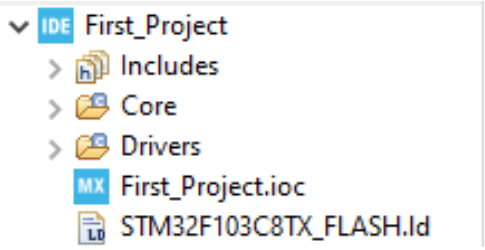


Рис. 2.4.10

Файлы проекта, как правило, представляют собой пары заголовочных файлов с расширением `.h` и исполняемых файлов с расширением `.c`. По общепринятым

правилам в заголовочных файлах объявляются константы и определяются функции, а в исполняемых прописывается рабочий код для функций. Пара файлов `.h` и `.c` способна образовывать библиотеку, которая может использоваться в основном рабочем файле `main.c`. Так как IDE прежде всего облегчает работу с микроконтроллером и позволяет программировать его, не запоминая множество регистров и их необходимых значений, то в проекте по умолчанию подключено множество библиотек, начиная со стандартных для языка C, заканчивая специализированными библиотеками для микроконтроллеров `STM32`, называемых `HAL`.

Если развернуть содержание папки `core`, то можно увидеть основные заголовочный и исполняемый файлы: `main.h` и `main.c`. Помимо этого подключается несколько файлов, которые по умолчанию задают работу системного таймера и прочих узлов микроконтроллера.

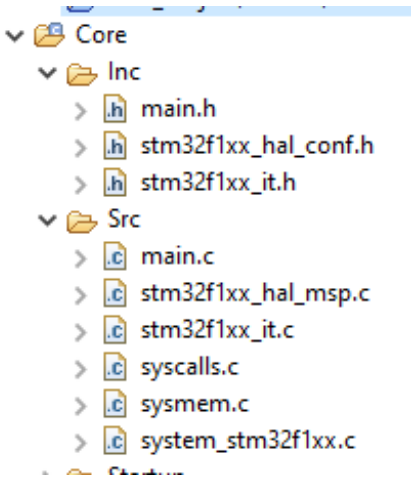


Рис. 2.4.11

Рассмотрим подробнее файл `main.c`. Двойное нажатие по наименованию файла откроет его в IDE. На начальный момент большая часть этого файла — комментарии. Обратите внимание, многие комментарии содержат слова `USER CODE`. Это те области, в которые нужно писать код. Тогда при новой генерации проекта он не будет стерт.

Помимо комментариев в этом файле еще содержатся:

- строка `#include "main.h"`, которая подключает заголовочный файл к исполняемому (стандартное действие при работе с несколькими файлами для языка C);.
- функция `void SystemClock_Config(void)`, задающая конфигурацию тактирования микроконтроллера, в том числе и работу системного таймера;
- функция `int main(void)` — самая основная функция в данном файле, именно она используется для программирования микроконтроллера и именно с ней впоследствии предстоит работать.

Обратите внимание, что в функции `int main(void)` есть следующие строки:

1. `HAL_Init();`
   Эта строчка вызывает инициализацию функций `HAL` — типовых функций, которыми мы будем пользоваться.
2. `SystemClock_Config();`
   Эта строчка вызывает функцию инициализации настроек тактирования.

3. `while(1)`
   Бесконечный цикл, который в большей степени определяет работу микроконтроллера.

При работе микроконтроллера вызывается функция `main` со всеми указанными настройками, а цикл `while(1)` будет повторяться до тех пор, пока питание не будет отключено.

Если возвращаться к тезису, что программа для микроконтроллера состоит из двух частей, то первая часть с преднастройками будет выполняться один раз, и она описывается в функции main до цикла `while`. Само тело вызываемых функций выносится в другие части файла, как сделано, например, с функцией `SystemClock_Config`.

Цикл `while`, в данном случае, вторая часть программы, исполняемая. Внутри цикла под код отведено место после строки `/* USER CODE BEGIN 3 */`.

Условие

Космокот Кас решил вынести инициализацию необходимых настроек для своих устройств в отдельную функцию, задав ее следующим образом (строка № 50).

C++

```
47  /* Privite function prototypes -----------------------------*/
48  void SystemClock_Config(void);
49  /* USER CODE BEGIN PFP */
50  void Init_All_Devices(void);
51  /* USER CODE BEGIN PFP */
```

В каком блоке функции `main` можно вызвать функцию `Init_All_Devices`?

Варианты ответа:

1. `USER CODE 1`;
2. `USER CODE Init`;
3. `USER CODE SysInit`;
4. `USER CODE 2`;
5. `USER CODE 3`.

Ответ: 4.

Задача 2.4.1.4. Жук в сапогах или что такое `Debug` и `Boots` (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Для удобства работы с микроконтроллером его размещают на отладочной плате. Отладочная плата содержит необходимые электронные компоненты для стабильной работы микроконтроллера и выведенные его контакты для подключения периферии.

Одной из широко распространенных отладочных плат является `Blue Pill`.

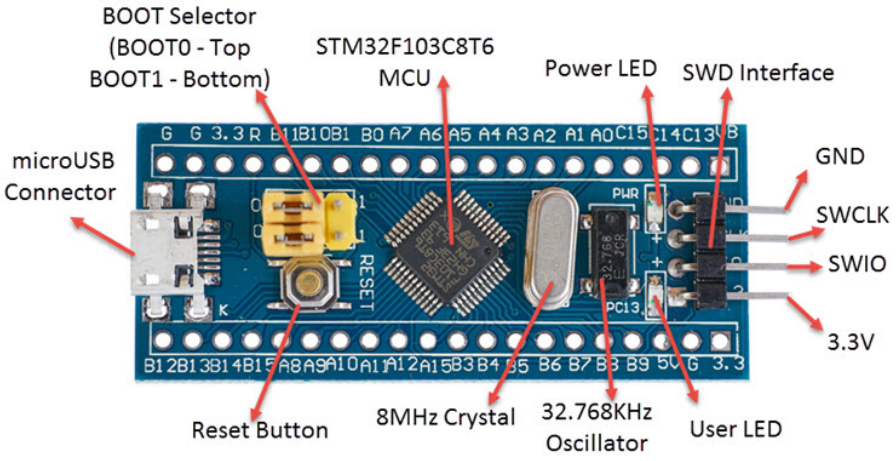
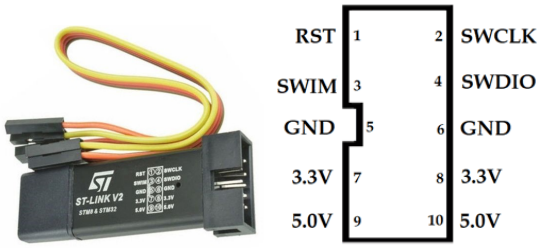
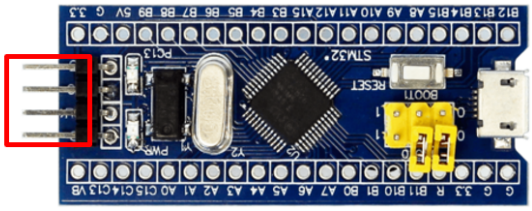


Рис. 2.4.12

Помимо самого микроконтроллера на отладочной плате можно найти вспомогательные светодиоды для индикации, кварцевые резонаторы на 8 МГц и 32 КГц и перемычки для изменения доступа к памяти.

`SWD` (Serial Wire Debug, четыре контакта на схеме справа: `SWCLK`, `SWIO`, питание и земля) — интерфейс для прошивки и отладки микроконтроллера. Он дает возможность с помощью `STM32CubeIDE` посмотреть, как работает программа, позволяя остановить ее в любом месте и посмотреть построчно, что происходит при ее выполнении. Такой режим очень полезен при поиске ошибок. Помимо этого, через `SWD` можно и прошить микроконтроллер.

Для прошивки микроконтроллеров используются специальные устройства — программаторы. Для `STM32`, как правило, выполняет функцию программатора `ST-Link`. В случае с платой Blue Pill он подключается по четырем проводам по интерфейсу `SWD`. С помощью `ST-Link` можно и прошивать микроконтроллер, и отлаживать загружаемую в него программу.

ST-Link V2	**Blue Pill**
	
(2) `SWCLK` >— (4) `SWDIO` >— (5) или (6) `GND` >— (7) или (8) `3.3V` >—	—> `SCK` —> `DIO` —> `GND` —> `3.3V`

Кварцевые резонаторы являются внешними источниками тактирования для микроконтроллера. Более подробно о них можно почитать по ссылке: http://myprac

tic.ru/urok-5-sistema-taktirovaniya-stm32.html (здесь используется CubeMX, однако STM32CubeIDE сделан на его основе, поэтому можно также попрактиковаться и в программе; для перехода на вкладку Clock Configuration дважды кликните по файлу с расширением .ioc в дереве вашего проекта и выберите соответствующую вкладку вверху окна с видом микроконтроллера).

Перемычки BOOT0 и BOOT1 управляют тем, какая область памяти будет запускаться при подаче питания на плату. Изначально обе перемычки стоят в нулевом положении, как показано на фото.

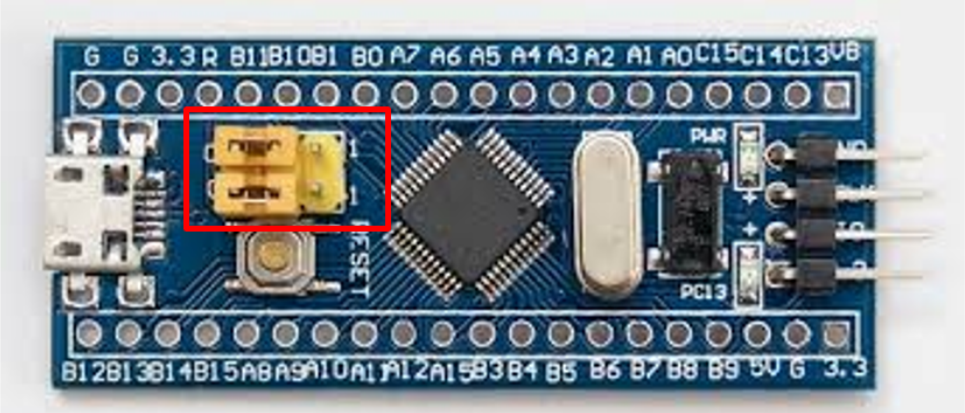


Рис. 2.4.13

Этот режим соответствует тому, что при подаче питания на плату выполнение программы начнется со считывания flash-памяти микроконтроллера и последующему ее выполнению. Режим стандартный, он используется в большинстве случаев. Когда плата прошивается через ST-Link, перемычки не нужно переставлять.

Если переставить нижнюю перемычку BOOT1 в положение 1, то при подаче питания на плату будет запущен загрузчик, который позволит загрузить программу во flash-память без использования ST-Link. Этот режим бывает полезен, если под рукой не оказалось самого ST-Link или нет возможности использовать стандартный интерфейс SWD (вместо SWD можно заменить на UART).

Если переставить обе перемычки на 1, то это позволит запустить работу с оперативной памятью. Такой режим используется достаточно редко, и в дальнейшем нам не понадобится.

Условие

Космокот Кас не смог прошить плату, используя ST-Link. В чем может быть неполадка? Выберите все возможные варианты:

1. на плате отсутствует разъем microusb;
2. неисправен один или несколько контактов интерфейса SWD;
3. неправильно подключен ST-Link;
4. на плате отсутствуют светодиоды;
5. не подключен UART.

Ответ: 2, 3.

Задача 2.4.1.5. Начинаем управлять (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

GPIO (General Purpose Input-Output) — это выводы общего назначения, доступные для прямого управления. На плате Blue Pill пины микроконтроллера выведены по бокам платы и имеют подписи в виде заглавной буквы и номера (A1, A5, B6 и т. д.).

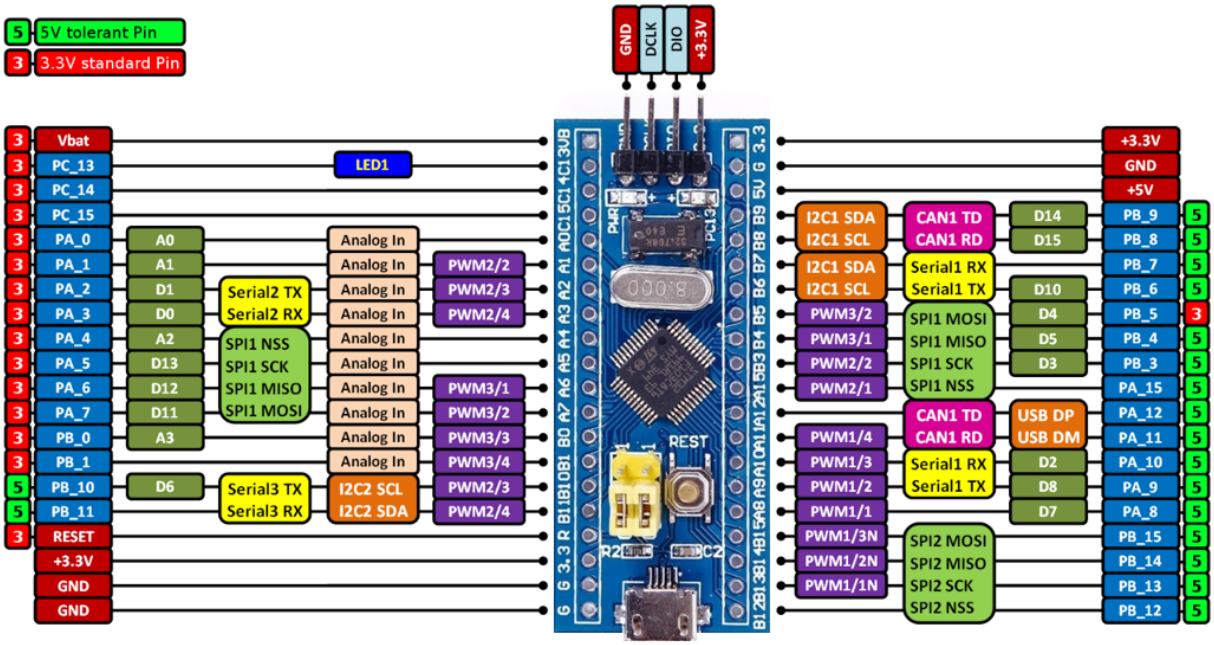


Рис. 2.4.14

Почти все пины могут работать как выводы общего назначения (GPIO), управление которыми сводится к двум действиям: подать высокий уровень напряжения (в случае с STM32 — 3,3 В) или подать низкий уровень напряжения (0 В). Таким образом можно зажигать и гасить подключенные к пинам светодиоды, работать с транзисторами, а также с некоторыми драйверами. Обобщая вышесказанное, можно сказать, что GPIO — это цифровой сигнал, в котором 0 соответствует 0 В, а 1 — 3,3 В (хотя иногда бывает наоборот).

Помимо GPIO, есть также более сложные способы управлять устройствами, например, широтно-импульсная модуляция или ШИМ (PWM), а также такие интерфейсы, как UART (Serial), I2C, SPI, CAN. Некоторые из них будут рассмотрены позднее.

Работа каждого пина GPIO может быть задана как исходящий или входящий режим. Когда пин GPIO работает в исходящем режиме, это значит, что микроконтроллер управляет им, подавая и убирая с него напряжение в соответствии с программным кодом. Если же работа GPIO задана во входящем режиме, микроконтроллер будет измерять на заданном пине уровень напряжения и выдавать логический 0, если напряжения нет, или логическую 1, если напряжение на пине есть.

Все пины разделены на группы A, B, C и D. В каждой группе есть свое количество пинов GPIO. Поэтому полный номер пина выглядит, например, как PA8. Это

значит, что будет использоваться восьмой пин из группы A. Далее эта информация понадобится для написания кода для управления пинами.

Более подробно о GPIO можно прочитать, например, здесь: http://mypractic.ru/urok-6-porty-vvoda-vyvoda-stm32.html.

Рассмотрим работу GPIO на классическом примере. На плате Blue Pill есть встроенный светодиод, который уже по умолчанию подключен к пину PC13 (на фото он горит зеленым).

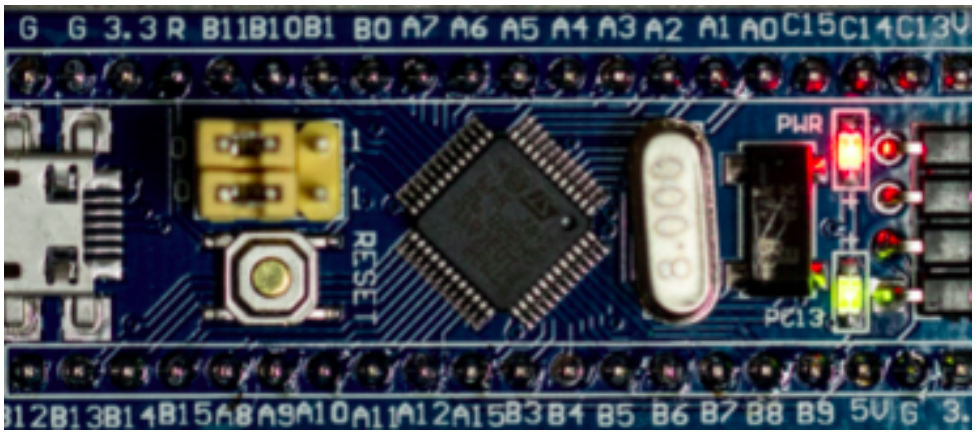


Рис. 2.4.15

Просто так он гореть не будет: на него нужно подать напряжение. И сделать это можно программно с помощью стандартной функции.

C++

```
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
```

Функция HAL_GPIO_WritePin() и будет задавать напряжение на светодиоде. Типовой аргумент функции GPIOC указывает, что используются порты из группы C, а GPIO_PIN_13 говорит о том, что в группе используется тринадцатый пин. Состояние пина GPIO_PIN_SET соответствует наличию напряжения на нем, а GPIO_PIN_RESET — его отсутствию.

Если этот код немного дополнить, то светодиод будет мигать.

C++

```
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
HAL_Delay(1000);
HAL_GPIO_WritePin(LED_PIN_GPIO_Port, LED_PIN_Pin, GPIO_PIN_RESET);
HAL_Delay(1000);
```

Функция HAL_Delay(1000) задает задержку в 1 с, то есть целую секунду программа просто считает тики по своему системному таймеру.

Условие

Космокот Кас запутался в функция управления GPIO, надо помочь ему соотнести, что делает функция и ее наименование:

- Функция установки состояния порта вывода GPIO.

- Функция инверсии состояния порта вывода GPIO.
- Функция чтения состояния порта ввода GPIO.

Ответ:

- функция установки состояния порта вывода GPIO — HAL_GPIO_WritePin;
- функция инверсии состояния порта вывода GPIO — HAL_GPIO_TogglePin;
- функция чтения состояния порта ввода GPIO — HAL_GPIO_ReadPin.

Задача 2.4.1.6. Продолжаем властвовать (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Предположим, что имеется простая система, в которой в зависимости от дискретного сигнала (принимающего значения 0 или 1) фоторезистора происходит переключение состояния реле (вкл или выкл). Проще говоря, логика работы следующая:

- если на фоторезистор падает свет, то он выдает значение, равное 1;
- если на фоторезистор находится в тени, то он выдает значение, равное 0;
- если сигнал с фоторезистора равен 1, то реле переходит в состояние вкл;
- если сигнал с фоторезистора равен 0, то реле переходит в состояние выкл.

В данном случае фоторезистор выступает в качестве источника входного сигнала для микроконтроллера, а контакт управления реле является выходным сигналом. Для реализации необходимой инициализации пинов воспользуемся режимом GPIO — ввода/вывода данных. Пусть для реализации такой схемы работы будут использоваться пины PA0 и PA1.

Зададим PA0 как входной сигнал. Для этого необходимо дважды кликнуть в дереве проекта на файл с расширением .ioc. Это вернет нас на вкладку с микроконтроллером. Кликните по пину PA0 и выберите из списка GPIO_Input, это определит его работу как входного сигнала. Затем кликните по пину PA1 и определите его как выходной, выбрав GPIO_Output.

Теперь нужно заново пересобрать проект, нажав на значок шестеренки. После этого снова откройте файл main.c. Обратите внимание, что добавилась новая функция static void MX_GPIO_Init(void). В ней есть строки, отвечающие за инициализацию пинов PA0 и PASTM32CubeIDE самостоятельно сконфигурировала эти части кода, чтобы упростить программирование микроконтроллера. Если потребуется изменить настройки пинов, то при следующей генерации проекта эти части кода также поменяются.

C++

```
/*Configure GPIO pin Output Level*/
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

/*Configure GPIO pin : PA0 */
GPIO_InitStruct.Pin = GPIO_PIN_0;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
8  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
9
10 /*Configure GPIO pin : PA1 */
11 GPIO_InitStruct.Pin = GPIO_PIN_1;
12 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
13 GPIO_InitStruct.Pull = GPIO_NOPULL;
14 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
15 HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

Каждая строчка кода определяет режим работы пина. По первым двум строчкам и по последней можно понять, что PA0 задается как входной (GPIO_MODE_INPUT), а PA1 — как выходной (GPIO_MODE_OUTPUT). Дополнительно для выходного пина указывается скорость его работы (сейчас это неважно, однако в некоторых проектах скорость реакции пина может влиять на качество работы всей системы).

Дополнительно узнать о том, что такое GPIO_NOPULL и о других режимах GPIO можно по ссылке: http://we.easyelectronics.ru/STM32/prakticheskiy-kurs-stm32-urok-1---gpio-porty-vvoda-vyvoda.html.

Вернемся к логике работы системы. Ее можно описать следующей блок-схемой.

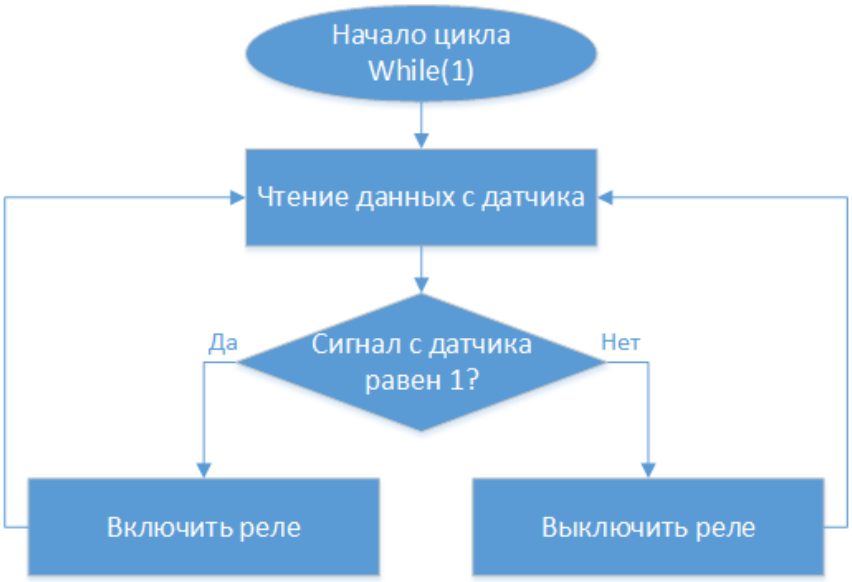


Рис. 2.4.16

Действие «Чтение данных с датчика» (в роли датчика выступает как раз фоторезистор) в данном случае будет задаваться типовой функцией HAL_GPIO_ReadPin. Единице соответствует состояние, когда на пине PA0 есть напряжение (то есть уровень напряжения 3,3 В), а нулю — когда напряжения нет (то есть уровень напряжения — 0 В).

Включение и выключение реле происходит за счет подачи на пин PA1 напряжения. Например, при напряжении в 3,3 В на пине PA1 реле будет включаться, а при напряжении 0 В — выключаться. Чтобы подать напряжение на пин, можно воспользоваться функцией HAL_GPIO_WritePin.

Код управления можно задать следующим образом (напомним, что писать код стоит в блоках, обозначенных как USER CODE, иначе каждая генерация проекта будет стирать блоки кода, написанные вне обозначенных блоков).

C++

```
while(1)
{
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
    if( HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET ) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
    }
    else {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
    }
}
```

GPIOA указывает обеим функциям, что используются порты из группы A, к которой и принадлежат пины PA0 и PA1, это указано в их названии (например, для пинов PB0 и PB1 нужно было бы указать GPIOB). Переменная, содержащая в названии сочетание GPIO_PIN_, указывает номер пина в группе. Состояние пина GPIO_PIN_SET соответствует наличию напряжения на нем, а GPIO_PIN_RESET — его отсутствию.

Приведенный код можно прописать в цикле while(1) в качестве небольшой тренировки. После того как это будет сделано, нужно кликнуть на значок молотка в верхнем меню, чтобы собрать проект.

Рис. 2.4.17

Если все функции прописаны правильно, то сообщения об ошибках не возникнет.

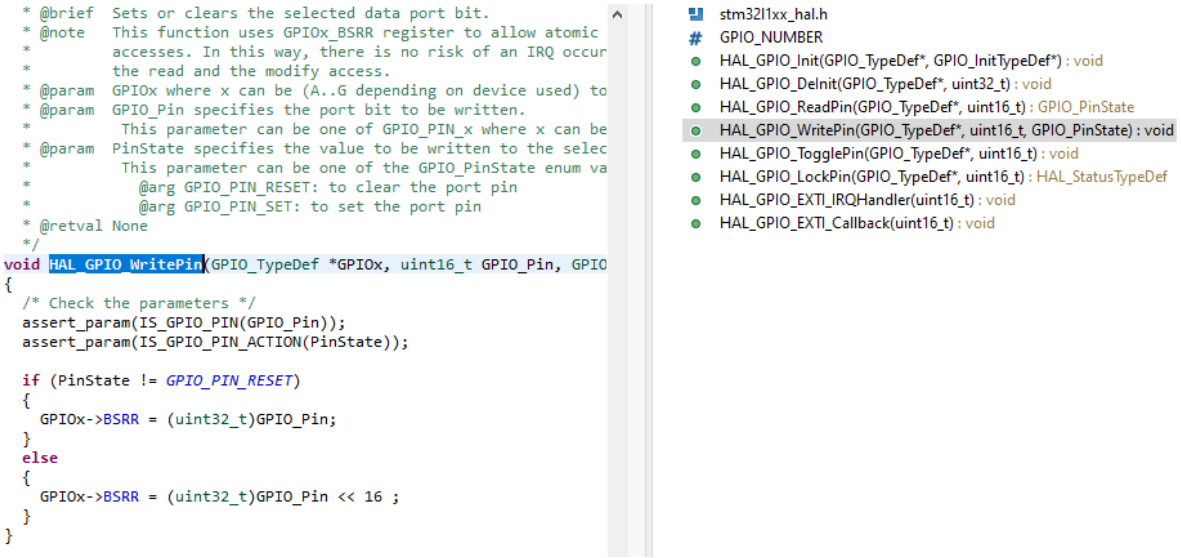


Рис. 2.4.18

После этого кликните правой кнопкой мыши по функции **HAL_GPIO_ReadPin** и выберите **Open Declaration**. Откроется новый файл с описанием функций **HAL**

для управления пинами **GPIO**. Весь перечень функций отобразится в окне справа (рис. 2.4.18).

Найдите там функцию **HAL_GPIO_TogglePin** и попробуйте по описанию разобраться, что она делает.

Условие

Определите, какое управление системой задает код ниже.

C++

```
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
  if (HAL_GPIO_ReadPin(GPIOA, GPIO_Pin_0) || HAL_GPIO_ReadPin(GPIOA,
   GPIO_Pin_0) == GPIO_PIN_SET)
  {
    HAL_GPIO_TogglePin(GPIOA, GPIO_Pin_1);
  }
}
/* USER CODE END 3 */
```

Варианты ответа:

1. при наличии света на датчике реле включается, а при отсутствии света — выключается;
2. при наличии света на датчике реле выключается, а при отсутствии света — включается;
3. состояние реле стало зафиксированным;
4. при отсутствии света на датчике реле включается, а при отсутствии света — выключается;
5. при отсутствии света на датчике реле выключается, а при отсутствии света — включается;
6. при наличии света на датчике реле меняет свое состояние на противоположное;
7. при отсутствии света на датчике реле меняет свое состояние на противоположное;
8. логика работы системы не изменилась.

Ответ: 6.

Задача 2.4.1.7. Время универсальности или что такое UART (1 балл)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Когда речь идет о работе с датчиками, часто возникает необходимость в выводе данных на экран в удобном формате, которые с него приходят. Для того чтобы

данные с датчика, считываемые микроконтроллером, отобразились на экране монитора, необходимо настроить обмен данным. Часто обмен идет по так называемому протоколу UART. По нему, например, данные можно передать через подключение по USB-порту.

UART — Universal Asynchronous Receiver-Transmitter — универсальный асинхронный приемо-передатчик. Этот интерфейс соединяет между собой два устройства четырьмя проводами и позволяет вести передачу данных в обоих направлениях. Интерфейс UART — широко распространенный способ построения обмена информацией между устройствами.

Как правило, в любом интерфейсе присутствует два стандартных провода: питание и земля (VCC и GND). В случае с UART они тоже есть. Однако помимо этого, UART задается двумя каналами: Rx (Receiver), который принимает данные, и Tx (Transmitter), который отправляет данные.

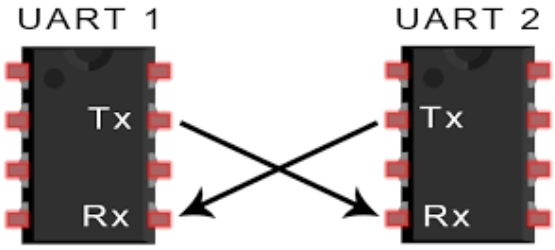


Рис. 2.4.19

Rx одного устройства соединяется с Tx другого устройства, и наоборот. Обмен по UART представляет собой передачу стандартной структуры бит, предполагающую небольшую вариативность. На рисунке ниже побитово представлена такая структура.

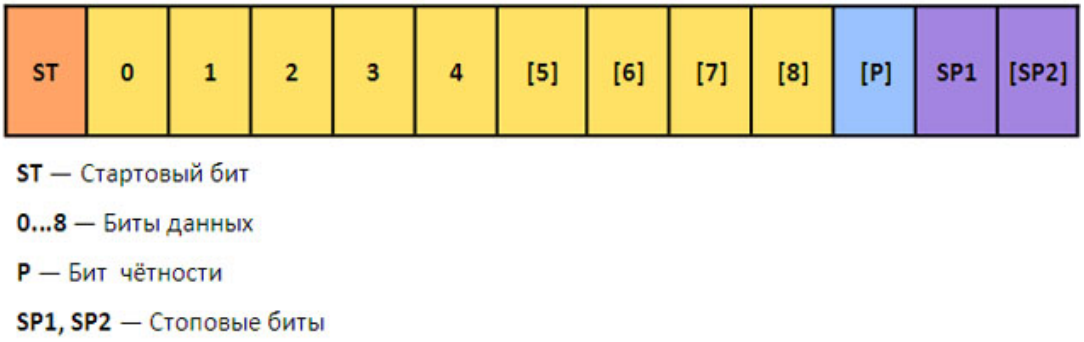


Рис. 2.4.20

Стартовый бит определяет начало передающегося байта данных, а стоповые — его конец. Это необходимо для синхронизации передачи и последующей расшифровки переданных данных, чтобы передаваемые байты не смешивались друг с другом, если вдруг по какой-то причине один из битов потеряется или за счет помехи поменяет свое значение.

Бит четности служит для проверки корректности полученных данных. Он равен нулю, если сумма всех единиц дает четное число, и единице, если сумма всех единиц дает нечетное число.

С точки зрения электрического уровня передаваемый массив из 0 и 1 выглядит примерно следующим образом.

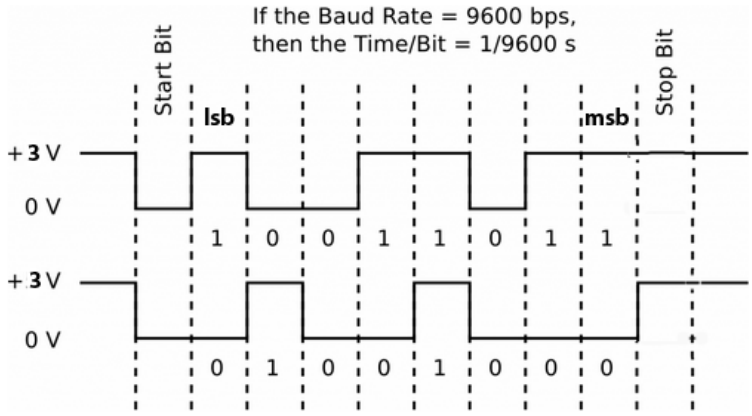


Рис. 2.4.21

Также одной из важных характеристик `UART` является его скорость. Для упрощения синхронизации двух устройств во время передачи значения скорости обмена по `UART` стандартизированы и могут принимать фиксированные значения: 4800 б/с, 9600 б/с, 19,2 кбит/с, 57,6 кбит/с и 115,2 кбит/с.

Для успешной передачи информации по UART нужно нужно, чтобы оба устройства, участвующие в информационном обмене, были синхронизированы и по скорости, и по структуре передаваемых данных. Все это задается настройками для обоих устройств.

К каждому `UART`, как правило, подключают всего одно устройство. У микроконтроллера `STM32F103C8T6` есть возможность задействовать несколько каналов `UART`. Если снова вернуться к схеме с распиновкой платы, то можно увидеть, какие именно пины можно использовать для `UART` (на схеме они обозначены как `Serial`).

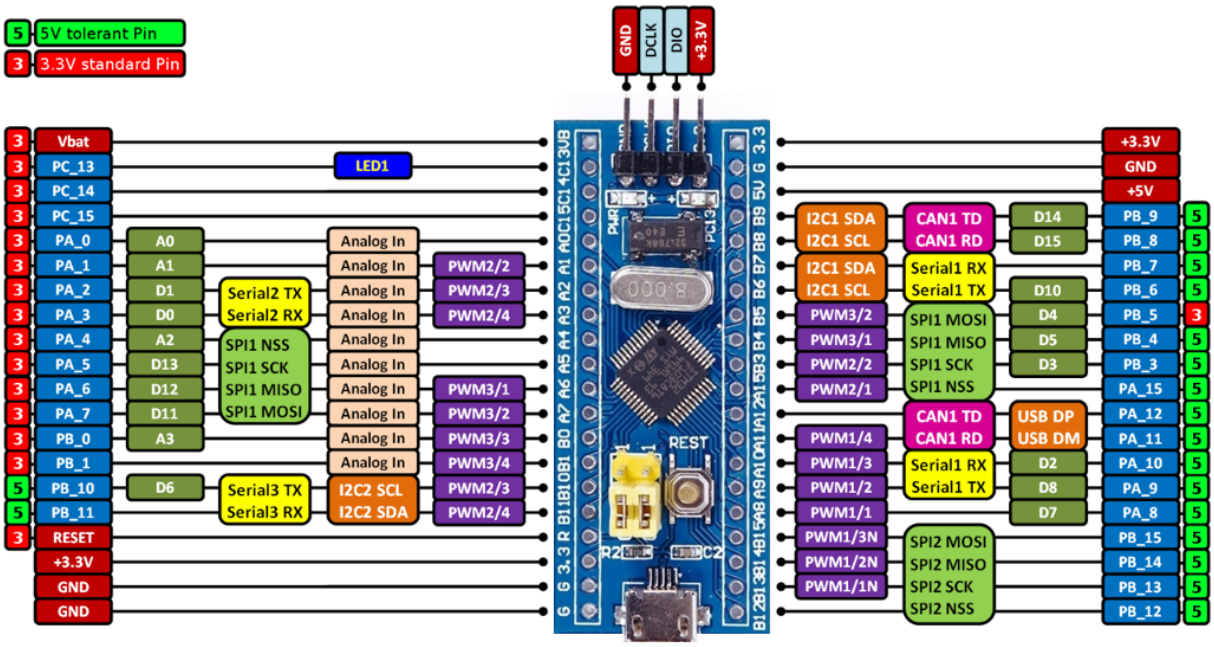


Рис. 2.4.22

Можно увидеть, что всего у микроконтроллера есть три `UART`, причем первый из них можно задать двумя способами. Такая вариативность позволяет подойти с некоторой гибкостью при распределении подключения множества устройств к выводам микроконтроллера. Будем пользоваться пинами `PA9` и `PA10`, так как они (и по умолчанию никакие другие пины `UART`) позволяют еще и прошивать микроконтроллер альтернативным способом.

Однако чтобы управлять обменом по этому интерфейсу, необходимо его для начала инициализировать. Чтобы это сделать с помощью `STM32CubeIDE`, нужно вернуться к вкладке с распиновкой микроконтроллера. Если она закрыта, то следует дважды кликнуть на файл с расширением `.ioc` слева в дереве проекта.

Перейдите в меню слева в пункт **Connectivity** и выберите **USART1** (пункт 1 на скриншоте). В появившемся меню настроек выберите **Mode** → **Asynchronous** (пункт 2). После этого можете наблюдать, как еще два пина инициализируются на схеме микроконтроллера (обычно `PA9` и `PA10`). У пинов также появятся надписи `USART1_RX` и `USART1_TX` (пункт 3).

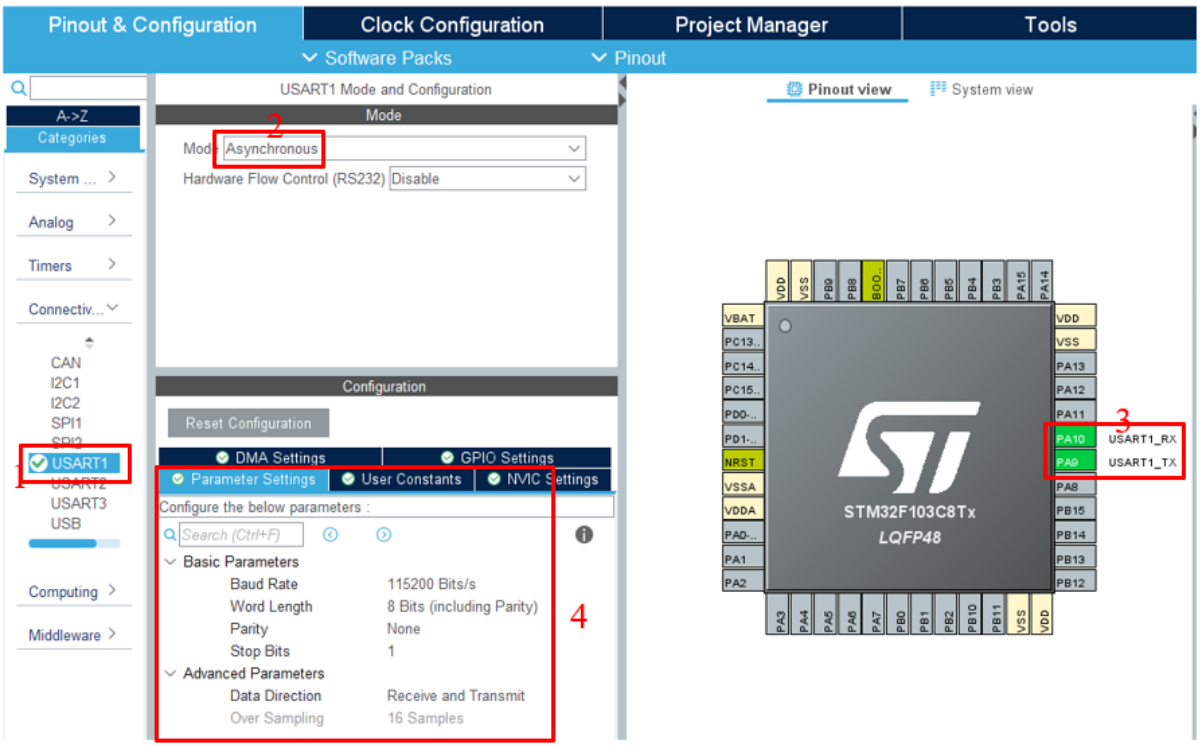


Рис. 2.4.23

Примечание. Обратите внимание, в тексте используется понятие `UART`, а у микроконтроллера используется `USART`. По этой ссылке можно прочитать про их отличие: https://www.rlocman.ru/review/article.html?di=16440. Однако USART в `STM32CubeIDE` настроен так, чтобы он стал `UART`.

В четвертом блоке можно увидеть настройки самого `UART`, о которых говорилось выше:

- `Baud Rate` — скорость обмена, в примере используется 115200 б/с;
- `Word Length` — структура бит, по умолчанию будет формироваться 8 бит информации, а девятый бит будет содержать в себе информацию о четности;
- `Stop Bits` — количество стоповых битов, по умолчанию задается 1.

Более подробно с UART можно ознакомиться здесь: http://mypractic.ru/urok-20-interfejs-uart-v-stm32-rabota-s-nim-cherez-registry-cms-is-ispolzovanie-preryvaniya-uart.html.

Снова сгенерируйте код. И после этого найдите в файле main.c новую функцию: static void MX_USART1_UART_Init(void).

C++

```
187 static void MX_USART1_UART_Init(void)
188 {
189     /* USER CODE BEGIN USART1_Init 0 */
190 
191     /* USER CODE END USART1_Init 0 */
192 
193     /* USER CODE BEGIN USART1_Init 1 */
194 
195     /* USER CODE END USART1_Init 1 */
196     huart1.Instance = USART1;
197     huart1.Init.BaudRate = 115200;
198     huart1.Init.WordLenght = UART_WORDLENGHT_9B;
199     huart1.Init.StopBits = UART_STOPBITS_1;
200     huart1.Init.Parity = UART_PARITY_EVEN;
201     huart1.Init.Mode = UART_MODE_TX_RX;
202     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
203     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
204     huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
205     huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
206     if (HAL_UART_Init(&huart1) != HAL_OK)
207     {
208         Error_Handler();
209     }
210     /* USER CODE BEGIN USART1_Init 2 */
211 
212     /* USER CODE END USART1_Init 2 */
213 }
```

Здесь можно увидеть все описанные выше настройки в виде параметров структуры huart1.Init.

Что такое структура, можно узнать здесь: http://www.c-cpp.ru/books/struktury.

Для передачи данных по UART используется функция HAL_UART_Transmit(). Допустим, требуется передать строку, которую храним в переменной str.

C++

```
1 /* USER CODE BEGIN Init */
2 char str[] = "Hello\n";
3 /* USER CODE END Init */
```

Тогда код будет выглядеть следующим образом.

C++

```
while(1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_UART_Transmit(&huart1, (uint8_t*)str, strlen(str), 10);
    HAL_Delay(1000);
}
```

В качестве параметров функции `HAL_UART_Transmit` используются следующие:

- `huart1` — указатель на тот `UART`, который используется (напомним, что у микроконтроллеров, как правило, несколько интерфейсов `UART`);
- `str` — указатель на массив данных (в данном случае переменную `str`), которую хотим передать по `UART`;
- `strlen(str)` — количество передаваемых по `UART` байт (функция `strlen` сама считает количество байт в `str`);
- 10 — максимальное время в микросекундах, выделяемое на операцию (по истечении этого времени функция `HAL_UART_Transmit` возвратит ошибку по таймауту).

Если написать эту функцию в файле `main.c` и кликнуть правой кнопкой мыши, то можно также открыть ее описание (`Open Declaration`).

В новом открывшемся файле также можно посмотреть типовые функции `HAL`, которые можно использовать для работы с интерфейсом `UART`.

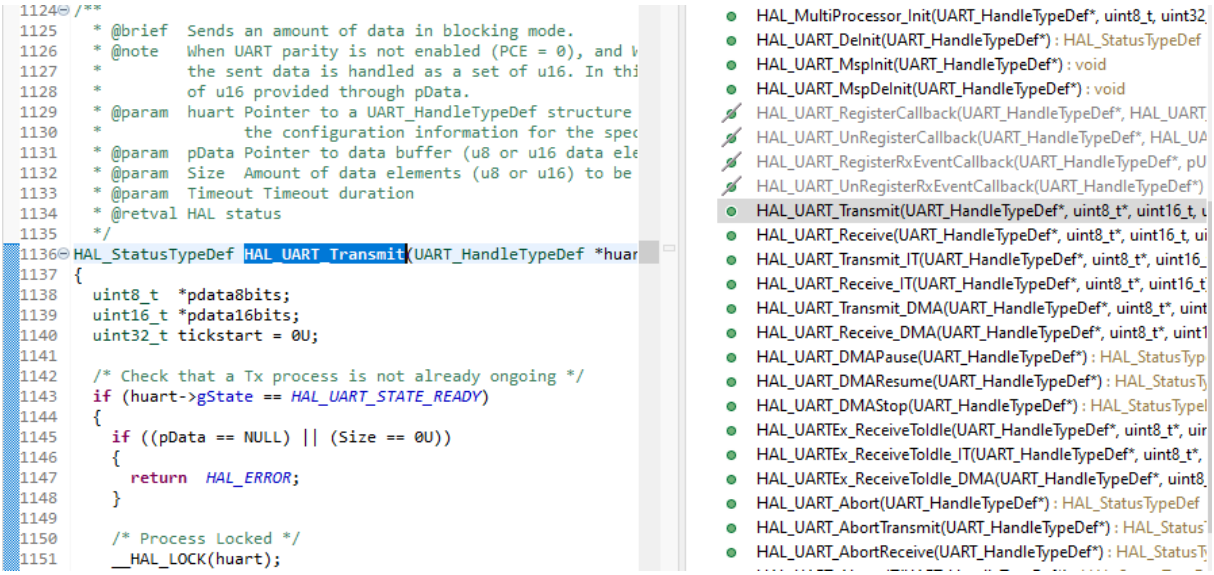


Рис. 2.4.24

Найдите в открывшемся файле функции:

- `HAL_UART_Receive()`
- `HAL_UART_Receive_IT()`

и ознакомьтесь с их описанием и тем, какие аргументы необходимо указать для их работы.

Условие

Космокот Кас решил написать код, который будет считывать данные по `UART1` и отправлять их по `UART2`. Помогите ему, выбрав среди вариантов ниже тот, который будет работать правильно.

1.
```C++
unsigned char USART_Buf;

HAL_UART_Receive(&huart1, &USART_Buf, 1, 1000);
HAL_UART_Transmit(&huart1, &USART_Buf, 1, 1000);
```

2.
```C++
unsigned char USART_Buf[2];

HAL_UART_Receive(&huart1, &USART_Buf, 1, 1000);
HAL_UART_Transmit(&huart2, &USART_Buf, 2, 1000);
```

3.
```C++
unsigned char USART_Buf[2];

HAL_UART_Receive(&huart1, USART_Buf, 1, 1000);
HAL_UART_Transmit(&huart2, USART_Buf, 1, 1000);
```

4.
```C++
unsigned char USART_Buf;

HAL_UART_Receive(&huart1, USART_Buf, 1, 1000);
HAL_UART_Transmit(&huart2, USART_Buf, 1, 1000);
```

Ответ: 3.

Задача 2.4.1.8. Нас много, но мы едины или как использовать I2C (2 балла)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

В сложных системах, содержащих множество устройств, осмысленно использование единой логики работы для всех устройств. Реализуется это с помощью применения различных интерфейсов и протоколов. `I2C` — один из самых простых и удобных интерфейсов, позволяющих с помощью одного микроконтроллера организовать работу множества устройств.

Шина `I2C` позволяет подключить одновременно до 127 устройств, которые смогут управляться от одного микроконтроллера. Управляющее устройство шины также принято называть `master`, а зависимые — `slave`. В нашем случае плата с микроконтроллером `STM32F103C8T6` — `master`, а остальные устройства — `slave`.

Шина I2C состоит из двух линий: SCL и SDA. Линия SCL является линией тактирования и отвечает за синхронизацию устройств между собой по частоте работы. Линия SDA является шиной данных и отвечает за их прием и передачу.

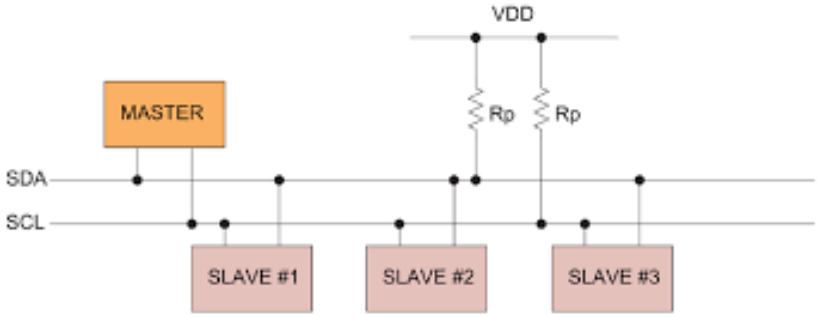


Рис. 2.4.25

Каждое устройство, подключенное к шине I2C, имеет свой I2C-адрес. Сообщение от микроконтроллера идет по шине, и каждое устройство проверяет, относится ли данное сообщение к нему, сравнивая свой адрес и адрес, указанный в сообщении. Обычно данный адрес задается производителем устройства и указывается в datasheet устройства.

Как правило, адрес является семибитным числом, что и накладывает ограничение в 127 устройств (т. к. 1111111 в двоичной системе — это 127).

Более подробно о шине I2C можно прочитать здесь (рекомендуем сюда заглянуть, чтобы посмотреть на схемы работы интерфейса): http://easyelectronics.ru/interface-bus-iic-i2c.html.

Снова вернемся к вкладке с распиновкой микроконтроллера. В пункте Connectivity выберите также I2C1. В настройках Mode выберите I2C → I2C.

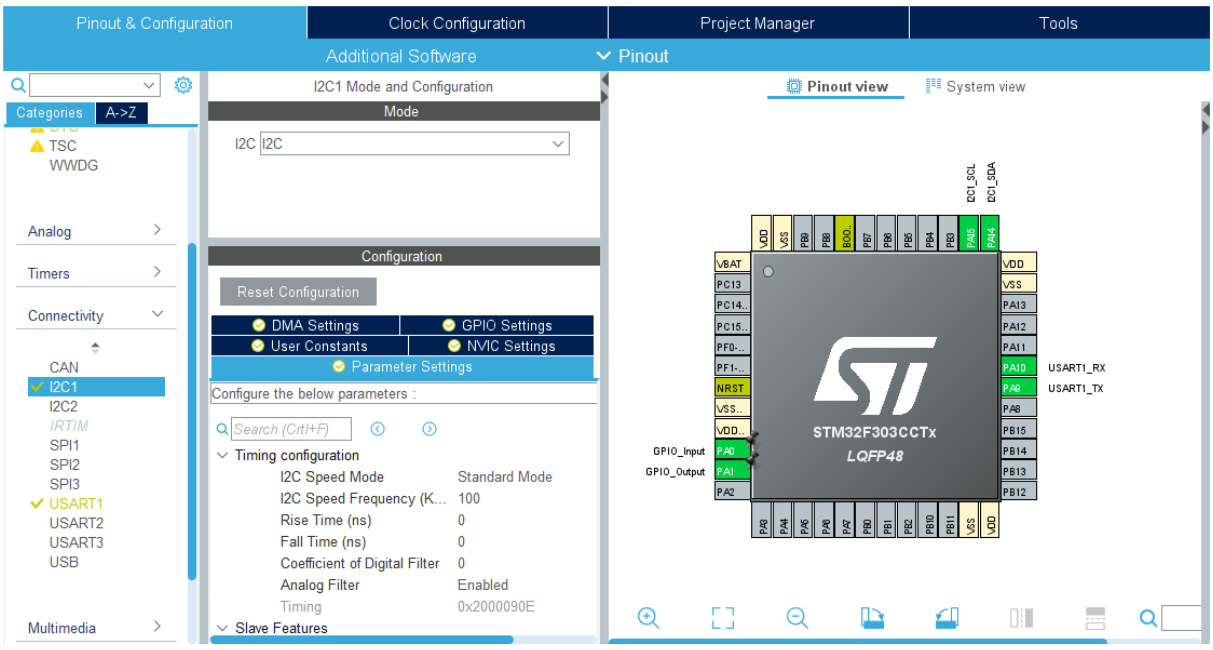


Рис. 2.4.26

Аналогично с USART, можно увидеть, что инициализируются два пина с подписями I2C1_SDA и I2C1_SCL. Обратите внимание, у нашего микроконтроллера есть две шины I2C: I2C1 и I2C2.

Сгенерируйте код. Найдите функцию static void MX_I2C1_Init(void). Эта функция отвечает за инициализацию I2C.

Так же как и для UART, для I2C существует библиотека HAL с типовыми функциями, позволяющими построить обмен данными между устройствами. Например, функция HAL_I2C_Master_Receive() позволяет считывать данные с подчиненного микроконтроллеру устройства.

Функция чтения данных с датчика освещенности выглядит следующим образом.

C++

```
unit16_t BH1750_readLightLevel(I2C_HandleTypeDef *I2C, unit8_t addr) {
    unit16_t level; // текущее измеренное значение
    unit8_t buf_in[2];

    while (HAL_I2C_GetState(I2C) != HAL_I2C_STATE_READY);
    // ждем готовность шины I2C
    HAL_I2C_Master_Receive(I2C, addr << 1, &buf_in, 2, 10);
    // читаем 2 байта по адресу датчика

    level = buf_in[0]; // загружаем старший байт результата
    level <<= 8; // сдвигаем его вверх
    level |= buf_in[1]; // добавляем младший байт результата

    return level;
}
```

Переменная addr хранит I2C-адрес, заданный шестнадцатеричным числом. Чтобы получить семибитный адрес, необходимо осуществить побитовый сдвиг на 1 бит, что задается выражением add << 1.

Переменная level имеет размерность в 16 бит. По I2C принимаем 2 байта информации, и в переменной, чтобы получить полное значение, объединяем два полученных байта в одно число. Делается это также с помощью побитового сдвига.

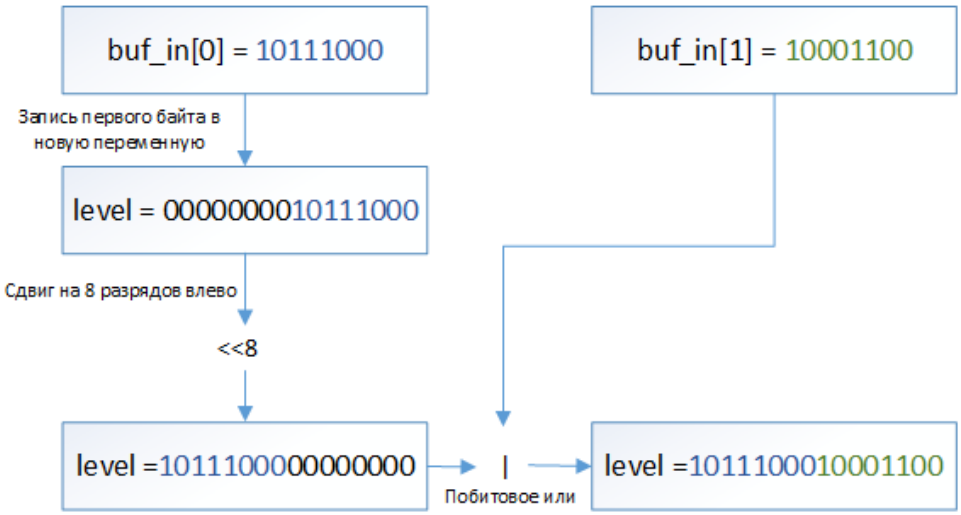


Рис. 2.4.27

Условие

Космокоту Касу необходимо считать один байт данных по адресу `0x5C`, используя протокол `I2C`, и получить `0x9C` (значение переданное со `slave` устройства), но свободных `I2C` интерфейсов не осталось, и ему пришлось реализовать программный `I2C`, но что-то пошло не так. Чтобы понять, что он сделал неправильно, он решил подключить осциллограф, и получил такую осциллограмму.

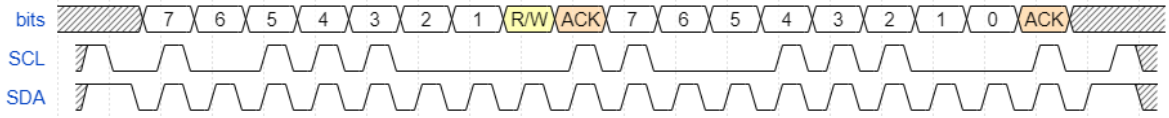


Рис. 2.4.28

Что он сделал неправильно? Выберите правильные варианты ответа:

1. Перепутал местами контакты `I2C`.
2. Некорректно установил комбинацию начала пакета.
3. Использовал неправильный адрес.
4. Некорректно указал режим работы `Slave` устройства.
5. Неправильно использовал бит подтверждения при отправке адреса устройства.
6. Неправильно использовал комбинацию смены направления передачи данных.
7. Некорректно начал считывать данные.
8. Получил неправильные данные.
9. Неправильно использовал бит подтверждения при получении данных с устройства.
10. Некорректно установил комбинацию конца пакета.

Ответ: 4, 9.

Правильный пакет.

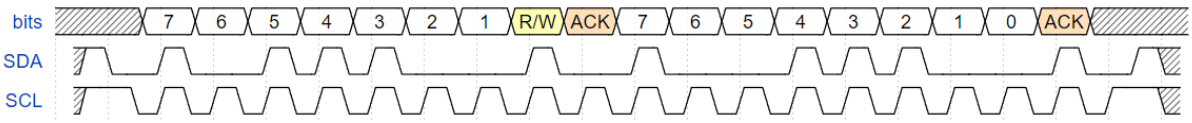


Рис. 2.4.29

Задача 2.4.1.9. Прерви меня, если сможешь (2 балла)

Темы: программирование, микроконтроллер, `STM32`.

Теоретическая часть

До этого момента ни разу не упоминалось такое понятие, как прерывание, однако оно весьма важно, когда речь идет о микроконтроллерах.

Как известно, при работе микроконтроллера код, написанный в цикле `while (1)`, будет выполняться циклично. Однако такой принцип работы не всегда удобен.

Допустим, что в проектируемой системе есть датчик, который реагирует на какое-то внешнее физическое изменение окружающей среды (свет, давление и пр.). Если такие изменения происходят достаточно редко, то проверять его наличие каждый раз в цикле `while(1)`, означает тратить вычислительные мощности впустую. К тому же, если это изменение требует быстрого реагирования, а код в цикле `while(1)` занимает какое-то время на выполнение, то можно вообще упустить нужный момент.

В таких случаях пользуются прерываниями. Прерывание — это действие, при котором микроконтроллер переключается с выполнения цикла `while(1)` (или иного процесса) на выполнение сторонней функции. Такое переключение всегда инициализируется неким событием, например, изменением показания с датчика.

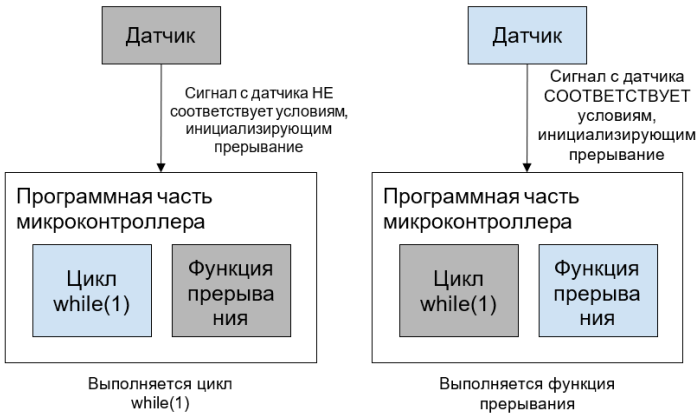


Рис. 2.4.30

Любой пин может быть настроен на работу с прерываниями, однако нельзя одновременно задействовать пины с одним и тем же номером. Так, например, нельзя одновременно настроить на работу с прерываниями пины `PA0` и `PC0`, но `PA0` и `PC1` — можно.

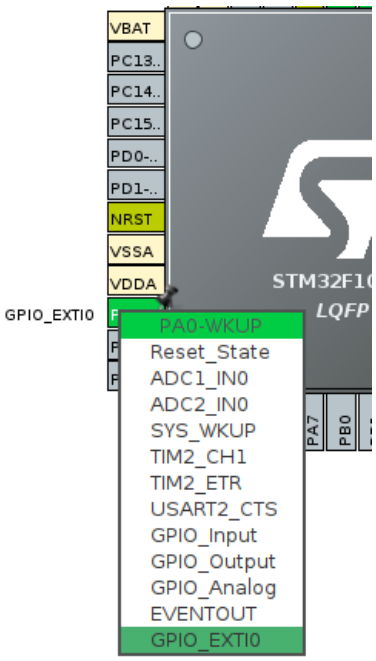


Рис. 2.4.31

Пример с фоторезистором и реле можно также рассмотреть с использованием прерывания. Вернитесь на вкладку с распиновкой микроконтроллера. Кликните по пину **PA0** и выберите **GPIO_EXTI** (рис. 2.4.31). Эта настройка определит работу пина в режиме внешнего (аппаратного) прерывания.

Прерывания также могут быть и внутренними (программными), однако сейчас в эту тему углубляться не будем.

Далее перейдите в **System Core** → **GPIO**, в настройках **PA0** переключитесь на вкладку **NVIC** и поставьте галочку, разрешающую прерывание.

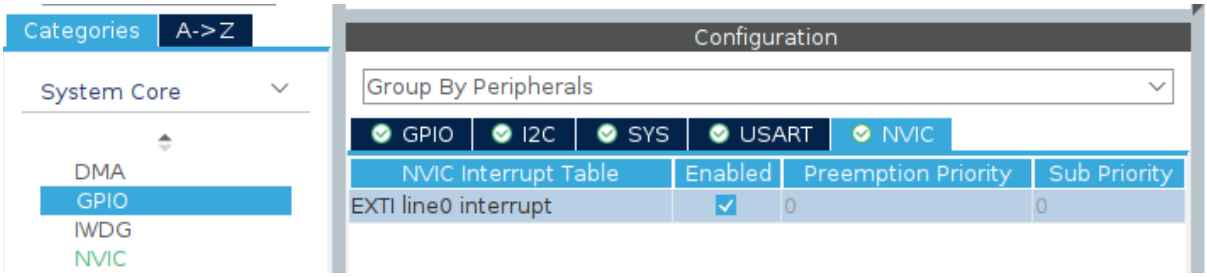


Рис. 2.4.32

NVIC — это контроллер прерываний, неотъемлемая часть микроконтроллера. Именно он отвечает за работу с прерываниями.

Сгенерируйте код. В коде в функции `MX_GPIO_Init()` должны появиться следующие строки.

C++

```
/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI0_IRQn);
```

Первая из них назначает приоритет прерывания, а вторая разрешает прерывание.

Если в работе микроконтроллера используется несколько прерываний, то необходимо будет задать им приоритет. При возникновении двух событий, инициализирующих разные прерывания, микроконтроллер переключится на выполнение той функции прерывания, приоритет у которой будет выше.

Для обработки прерывания нужно будет добавить в файл `main.c` (но не в `while(1)`, а в любую часть, выделенную как `USER CODE`) обработчик прерывания.

C++

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(unit16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_0)
    {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
    }
}
/* USER CODE END 4 */
```

Такой код будет при высоком уровне на `PA0` переключать состояние `PA1`.

Помимо использования прерываний на GPIO, можно также использовать прерывания на таймерах и интерфейсах типа UART, I2C и пр.

С прерываниями, на примере таймера, можно ознакомиться по следующей ссылке: http://mypractic.ru/urok-18-sistema-preryvanij-stm32-organizaciya-i-upravlenie-preryvaniyami.html.

Условие

Космокот Кас настроил микроконтроллер на работу с прерываниями по UART, I2C и GPIO, выставив им следующие приоритеты:

- UART: 0;
- I2C: 10;
- GPIO: 9,

где 0 — максимальный приоритет. В ходе работы микроконтроллера события, инициализирующие прерывания, приходили в следующем порядке.

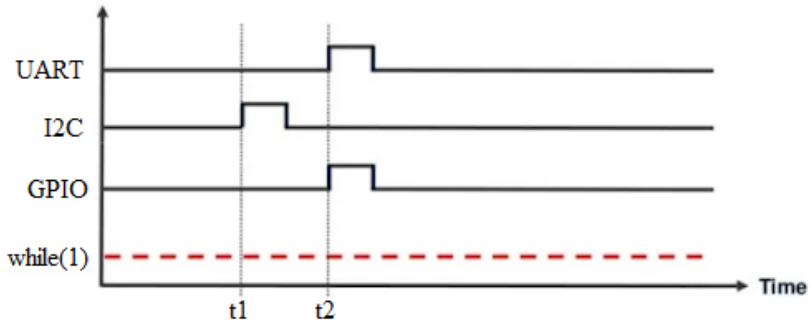


Рис. 2.4.33

Здесь:

- $t_1 = 120$ мкс от начала работы микроконтроллера;
- $t_2 = 140$ мкс от начала работы микроконтроллера.

Продолжительность выполнения обработчиков прерываний занимает следующее время:

- UART: 10 мкс;
- I2C: 25 мкс;
- GPIO: 15 мкс.

Укажите, что будет происходить в каждый интервал времени работы микроконтроллера.

От 0 до 120 мкс	Выполнение цикла while(1)	Выполнение обработчика I2C	Выполнение обработчика UART	Выполнение обработчика GPIO

От 120 до 125 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 125 до 130 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 130 до 135 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 135 до 140 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 140 до 145 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 145 до 150 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 150 до 155 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 155 до 160 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`
От 160 до 165 мкс	Выполнение цикла `while(1)`	Выполнение обработчика `I2C`	Выполнение обработчика `UART`	Выполнение обработчика `GPIO`

Решение

Согласно приоритетам прерываний, до первого события, инициализирующего прерывание, будет идти выполнение цикла `while(1)`. После первого события выполнение переключится на обработчик прерывания по `I2C`, которое занимает 25 мкс. Однако через 20 мкс придет более приоритетное событие на прерывание, и выполнение переключится на обработчик прерывания по `UART`, который будет выполнять в течение 10 мкс. Далее микроконтроллер переключится на прерывания `GPIO`, так как его приоритет выше, чем у `I2C`. Через 15 мкс микроконтроллер закончит обработку прерывания по `GPIO` и переключится на обработку прерывания по `I2C`. Далее через 5 мкс вернется в `while(1)`.

Ответ:

- от 0 до 120 мкс выполнение цикла `while(1)`;
- от 120 до 140 мкс выполнение обработчика `I2C`;
- от 140 до 150 мкс выполнение обработчика `UART`;
- от 150 до 165 мкс выполнение обработчика `GPIO`.

Задача 2.4.1.10. Тик — он и в микроконтроллере тик (4 балла)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

У микроконтроллера есть возможность использовать один из нескольких источников тактирования для работы системного таймера. Для того чтобы ознакомиться с ними более подробно, рекомендуем ознакомиться с материалами по ссылке: http://mypractic.ru/urok-5-sistema-taktirovaniya-stm32.html.

От источника тактирования зависит частота, с которой смогут работать таймеры и периферийные интерфейсы. Помимо системного таймера, существуют также базовые таймеры, таймеры общего назначения и продвинутые таймеры. Более подробно о них можно почитать по следующим ссылкам:

- https://robocraft.ru/arm/722;
- https://robocraft.ru/arm/739.

Одно из применений таймеров общего назначения — работа с сигналами прямоугольной формы. Получая на свой вход такой сигнал, таймер может определять длительность импульсов.

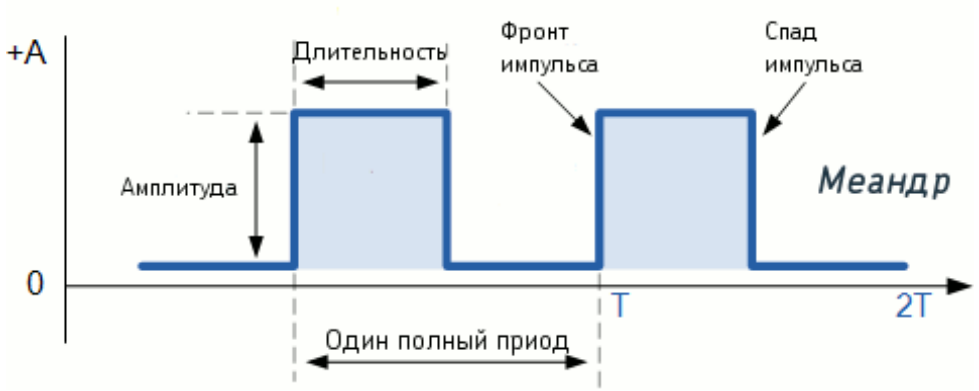


Рис. 2.4.34

Таймер может определять моменты наступления фронта и спада импульса, тем самым также определяя и его длительность. Подобного рода информация нередко используется в технических системах, например, при работе с оптическими датчиками, лежащими в основе датчиков оборотов двигателей.

Допустим, что к таймеру подключен такой оптический датчик. Если снова вернуться на вкладку с распиновкой микроконтроллера и выбрать, например, таймер 2 (TIM2), то можно увидеть, что у него в настройках есть четыре канала. Два из этих каналов можно настроить так, чтобы один из них фиксировал момент фронта импульса, а второй — момент спада импульса. Для этого необходимо в Channel1 выбрать Input Capture direct mode, а в Channel2 — Input Capture indirect mode.

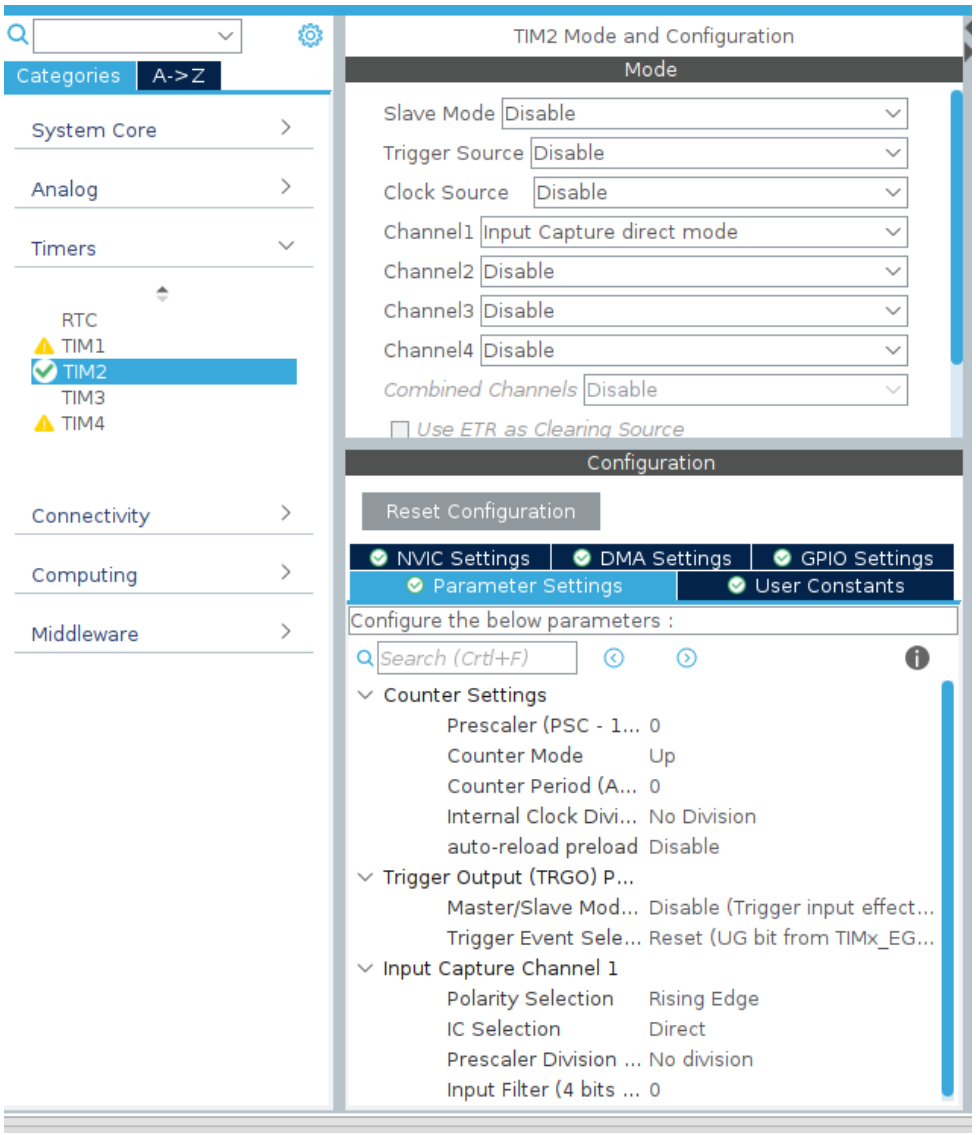


Рис. 2.4.35

Ниже есть общие настройки таймера и настройки его каналов. Посмотрим сначала на настройки каналов. В канале 1 в `Polarity Selection` должно стоять `Rising Edge`, в канале 2 — `Falling Edge`.

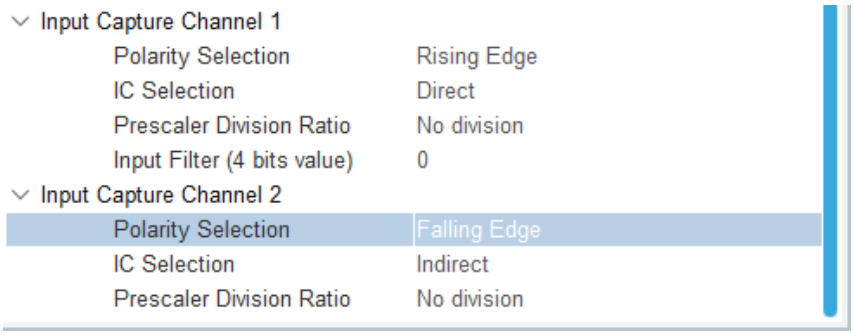


Рис. 2.4.36

При таких настройках таймер будет работать в описанном выше режиме.

Теперь вернемся к общим настройкам таймера.

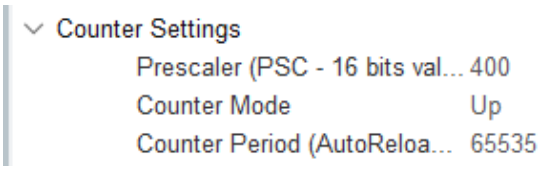


Рис. 2.4.37

- `Prescaler` — предделитель, который базовую частоту работы таймера делит на указанное число $+1$ (то есть если не требуется уменьшение частоты работы таймера, нужно поставить 0);
- `Counter Mode` — режим отчета тиков таймера: с убыванием от установленного значения или с возрастанием до него;
- `Counter Period` — как уже говорилось, значение количества тиков, по достижении которого таймер перезагружается.

Более подробно о настройках таймера можно узнать по ссылке: http://mypractic.ru/urok-16-tajmery-stm32-v-rezhime-schetchikov-generaciya-ciklicheskix-preryvanij-ot-tajmerov.html.

Условие

Космокот Кас решил сделать таймер, который будет обнуляться каждые 2,38547240277777777777777777777778 с. Кас решил подключить таймер к `HCLK` с частотой 72 МГц. Помогите Касу найти `Prescaler` и Counter period.

Примечание. Точность вычислений до 10^{-32}.

Решение

Количество тиков источника на один период таймера (i) равно:

$$72\,000\,000 \cdot 2{,}38547240277777777777777777777778 = 171\,754\,013.$$

Далее необходимо найти все делители числа 171 754 013:

$$1, 4\,129, 41\,597, 171\,754\,013.$$

Всего делителей четыре. Из них подходят 4 129 и 41 597. Для правильного ответа необходимо вычесть из полученных чисел единицу.

Ответ: 4128, 41596.

Задача 2.4.1.11. ШурШИМ двигателями (5 баллов)

Темы: программирование, микроконтроллер, STM32.

Теоретическая часть

Одним из режимов работы таймера является генерация ШИМ. ШИМ — это широтно-импульсная модуляция, которая представляет собой череду прямоугольных импульсов.

ШИМ позволяет с помощью цифрового сигнала варьировать значение в диапазоне от 0 до 3,3 В с помощью генерируемых импульсов. Импульс, как и цифровой сигнал, может принимать всего два значения: 0 или 3,3 В. У каждого импульса есть длительность, а у череды импульсов есть частота, с которой они генерируются. Меняя длительность импульса и частоту импульсов, можно получить среднее значение между двумя уровнями 0 и 3,3 В за счет того, что частота генерации импульсов высокая, и управляемое устройство воспринимает сигнал не как череду отдельных импульсов, а как их некоторое усредненное значение, так как не успевает среагировать на каждый импульс отдельно.

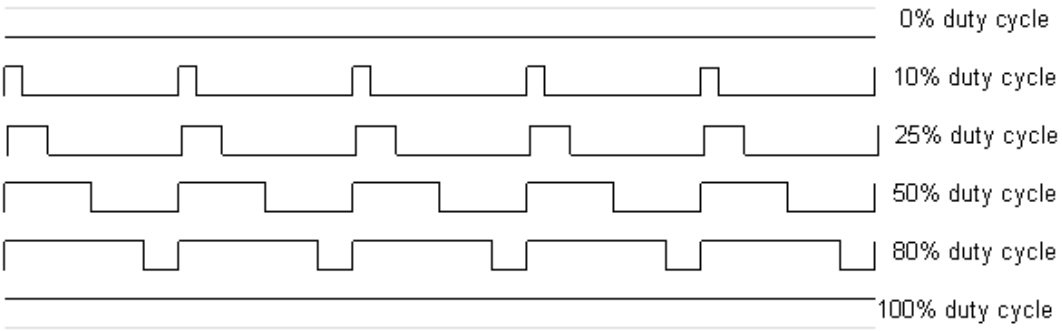


Рис. 2.4.38

На рисунке показано разное заполнение скважности ШИМ: от 0 до 100%. Генерируемые импульсы дают соответствующее напряжение на выходе: для 0% — 0 В, для 25% — 0,825 В, для 50% — 1,65 В и т. д., вплоть до 100%, которое соответствует максимальным 3,3 В.

Так, если на светодиод подать 3,3 В, то он будет гореть ярко. Однако ШИМ-сигнал, подаваемый на светодиод, позволит регулировать его от тусклого свечения до полной яркости.

Помимо управления яркостью светодиода, ШИМ также можно применять для управления двигателем постоянного тока. Разное заполнение скважности ШИМ повлияет на подаваемое напряжение на двигатель, что будет, в свою очередь, регулировать скорость вращения его вала.

В спутниковых системах двигатели используются в системах стабилизации и ориентации. На такой двигатель закрепляют маховик (металлический диск определенных формы и веса). Двигатель, раскручивая маховик, может управлять вращением аппарата.

Однако наличия двигателей-маховиков недостаточно для управления положением спутника в пространстве. Для корректной работы системы нужны еще датчики положения, которые могли бы определять положение аппарата относительно какого-либо ориентира.

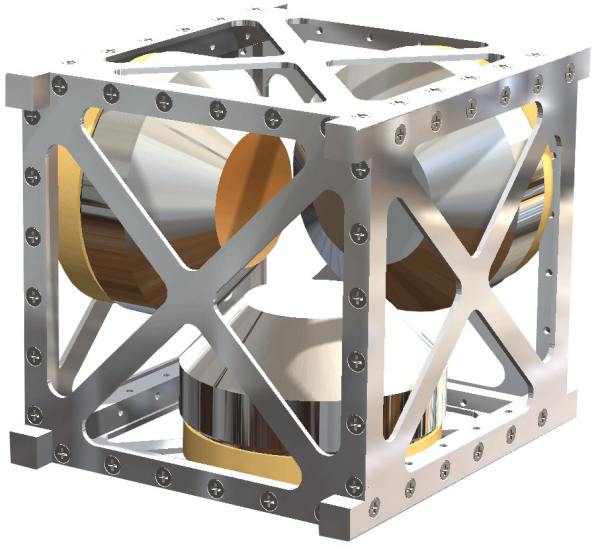


Рис. 2.4.39

Для того чтобы настроить таймер на работу с ШИМ, нужно выбрать настройку одного из его каналов, а именно, `PWM Generation`.

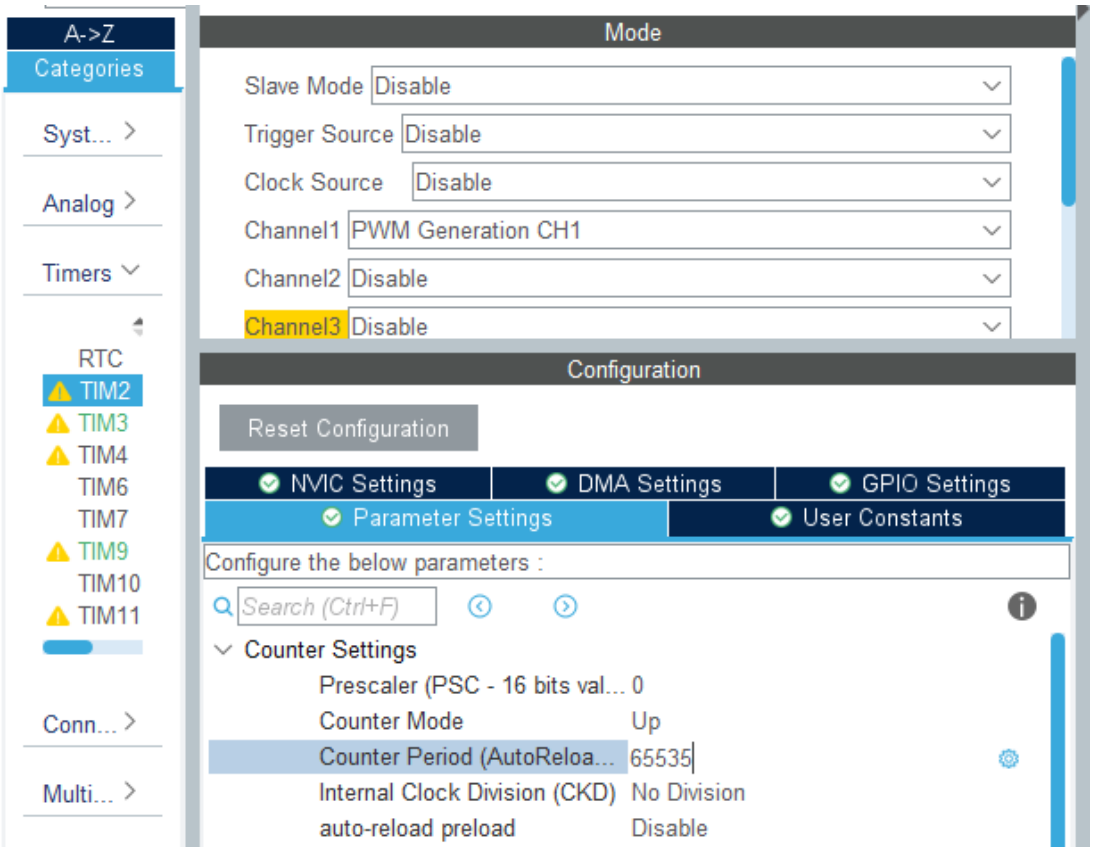


Рис. 2.4.40

В настройках в режиме ШИМ `Counter Period` будет определять скважность ШИМ, то есть максимальное количество отсчетов для полного заполнения ШИМ. В данном примере указано значение 65535, это означает, что для сигнала 3,3 В в соответствующий регистр таймера нужно будет записать 65535, для сигнала 1,67 В — 32767 и т. д.

Условие

Космокоту Касу понадобился сервопривод, он знает, что хочет управлять им с шагом 0,18°. Сервомашинка управляется импульсами. Импульс состоит из положительного фронта от 1 до 2 мс, с частотой этих импульсов 50 Гц.

Длина импульса, мс	Положение сервопривода, °
1	0
1,5	90
2	180

Примеры сигналов.

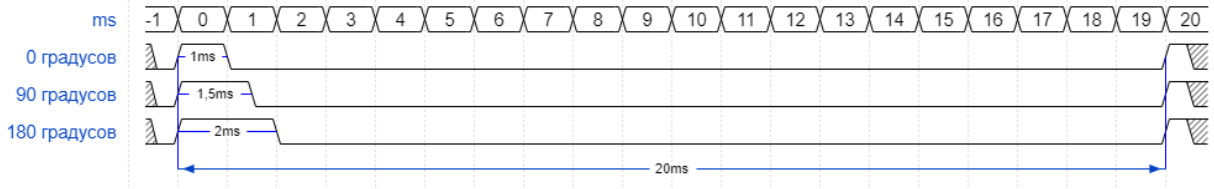


Рис. 2.4.41

Нужно помочь Касу настроить таймер (`Prescaler` и `Counter period`) для генерации такого сигнала. Таймер подключен к `HCLK` с частотой 72 МГц.

Решение

Из шага 0,18° можно сделать вывод, что количество шагов изменения положения должно равняться $180/0{,}18 = 1\,000$. Так как диапазон времени, внутри которого должны лежать эти шаги, должно быть 2-1 мс, то, следовательно, шаг изменения положения на 0,18 должен равняться 1 мкс. Тогда общее количество тиков таймера до переполнения должно равняться $20\,\text{мс}/1\,\text{мкс} = 20\,000$.

Зная частоту, которая приходит на вход таймера, и количество тиков до переполнения, можно высчитать предделитель для таймера (72 МГц / 50 Гц) / 20000 = 72. В ответ надо записать числа на единицу меньше из-за строения таймеров микроконтроллера.

Ответ:

- `Prescaler` — 71;
- `Counter period` — 19 999.

Задача 2.4.1.12. ПИД-регулирование — это почти магия (5 баллов)

Темы: программирование, микроконтроллер, `STM32`.

Теоретическая часть

В предыдущем задании уже затрагивалась тема законов управления. Все устройства, составляющие любую техническую систему, связаны единой логикой работы. Рассмотрим это на примере системы стабилизации космического аппарата, в которую может входить датчик угловой скорости и двигатель-маховик.

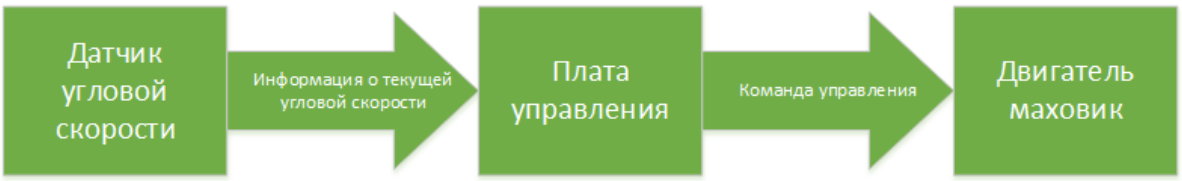


Рис. 2.4.42

Датчик угловой скорости предоставляет информацию об угловой скорости космического аппарата. Из этого можно получить различные данные после обработки, например, информацию об угле поворота за определенный момент времени.

Если задачей системы является сведение угловой скорости космического аппарата к нулю, то для достижения этой цели потребуется грамотное управление маховиком. Для организации такого управления нужно учесть следующее:

- актуальную информацию о том, что происходит со спутником: в данном случае за это отвечает датчик угловой скорости;
- алгоритм управления, позволяющий сформировать необходимую команду управления на маховик;
- постоянное обновление информации по угловой скорости после отработки маховика для учета корректировок управления.

Одним из популярных алгоритмов управления, который используют в подобных задачах, является алгоритм ПИД-регулирования. О ПИД-регулировании уже написано немало статей, поэтому не будем повторяться, а просто приведем здесь одну из них для ознакомления: https://alexgyver.ru/lessons/pid/.

Условие

Космокот Кас разбирается в телеметрии систем спутника и пытается понять, корректно ли происходит работа космического аппарата. Надо помочь Касу определить состояние спутника по графикам и тому, как эти системы себя вели при старте миссии.

1. Нагреватель колбы с бактериями. Время переходного процесса было 8 с. Перерегулирование достигало 5%, колебаний при регулировании не замечалось.

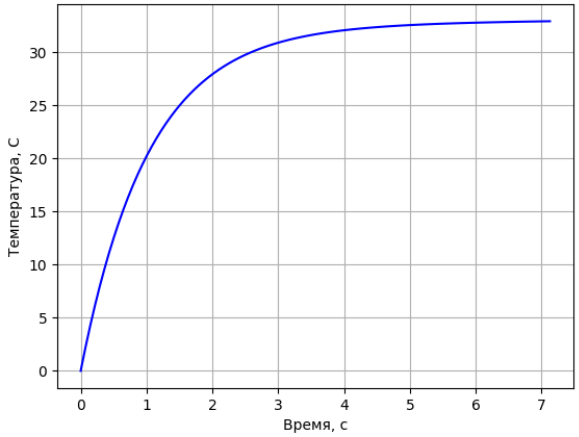


Рис. 2.4.43

2. Система поддержания температуры электроники спутника. Время переходного процесса было 100 с. Колебания были не больше 25% уставки.

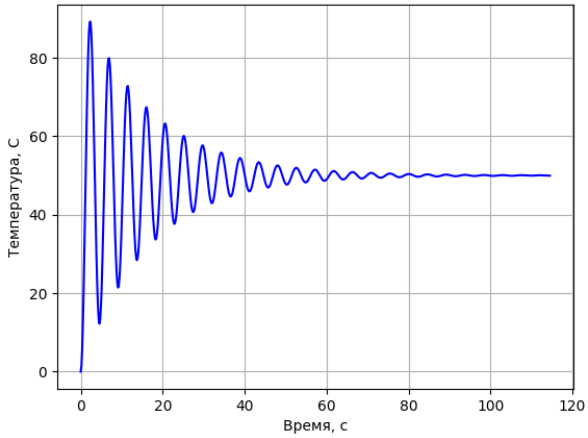


Рис. 2.4.44

3. Система ориентации спутника. Время переходного процесса было 40 с. Колебания были не больше 20% уставки.

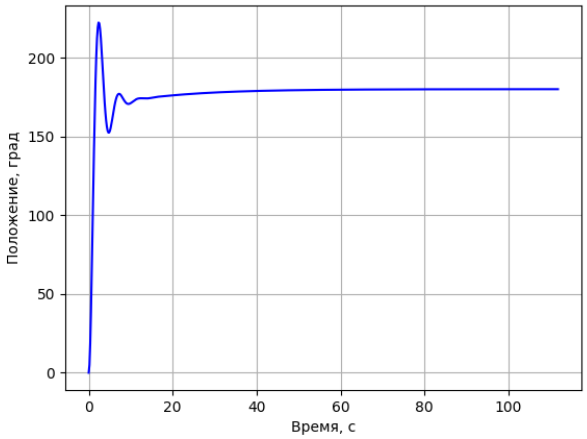


Рис. 2.4.45

Ответ: работает нормально 1, работают с ошибками — 2 и 3.

2.4.2. Задачи по компетенции Инженер-программист полезной нагрузки

За каждым великим программистом стоит не менее великий схемотехник.

Печатная плата с микроконтроллером, позволяющая управлять спутником, датчики, с которых можно получить данные об освещенности или позиционировании аппарата, система электропитания и другие устройства — все это является результатом труда инженеров-схемотехников и специалистов по полезной нагрузке. Именно они создают сложные электронные схемы и интегрируют сенсоры, обеспечивая точное и надежное функционирование спутника на орбите.

С помощью данного тестирования участники инженерного тура вспомнят, узнают или откроют для себя новые аспекты основ схемотехники и работы с сенсорами; научатся проектировать источники питания, без которых не может функционировать ни один спутник, и узнают о том, как устроены некоторые датчики.

Для успешного освоения представленного материала и решения проверочных заданий будет полезно вспомнить раздел физики, посвященный электричеству.

Задача 2.4.2.1. Сопротивление не бесполезно! (1 балл)

Тема: схемотехника.

Теоретическая часть

Самая простая электрическая цепь состоит из источника питания, нагрузки (компонента, потребляющего электроэнергию) и соединительных проводов (это также могут быть и проводящие дорожки на печатной плате).

Источник питания поддерживает разность потенциалов между выводами, из-за чего по цепи начинает протекать электрический ток: свободные электроны отталкиваются от области, где сконцентрирован отрицательный заряд, и притягиваются к области положительного заряда. Исторически сложилось, что направление тока считают обратным движению электронов. Разность потенциалов иначе называют напряжением, и его всегда измеряют между двумя точками схемы, а ток протекает через элемент схемы, поэтому для его измерения необходимо подключать амперметр в разрыв цепи.

Одним из наиболее распространенных в электронике элементов является резистор — элемент, который сопротивляется протеканию электрического тока, поглощая при этом часть электроэнергии и рассеивая ее в виде тепла.

Взаимосвязь между сопротивлением участка цепи, падением напряжения на нем и силой протекающего через него тока определяется законом Ома. Ток стремится протекать по более «простому» пути — там, где сопротивление меньше; а чем больше напряжение, тем больше энергии для перемещения зарядов и тем, соответственно, больше электрический ток:

$$I = \frac{U}{R}.$$

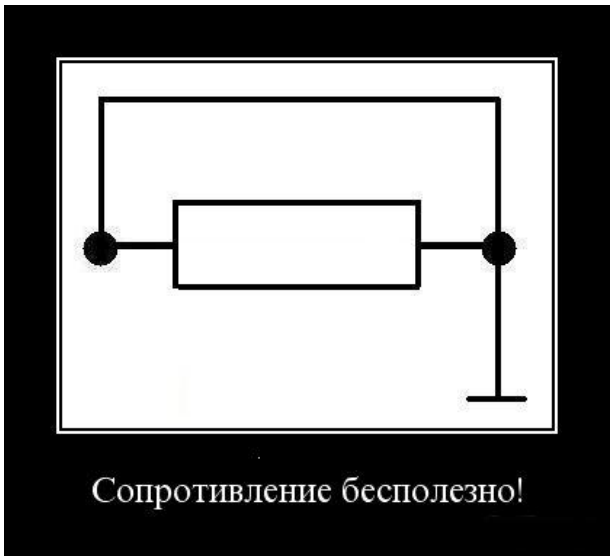


Рис. 2.4.46

Казалось бы, зачем использовать в схемах резисторы, если на них теряется часть энергии? Как раз для того чтобы ограничивать протекающий в цепи ток! Без резисторов электронные компоненты, обладающие малым внутренним сопротивлением, вышли бы из строя.

Например, желтый светодиод — полупроводниковый прибор, который светится желтым светом при пропускании через него тока определенной направленности (о диодах мы поговорим чуть позже), — требует напряжения питания 2 В, но ток, протекающий через него, необходимо ограничить значением 20 мА, чтобы он не сгорел. Представим, что единственный имеющийся источник питания — это аккумулятор с напряжением на выходе 9 В.

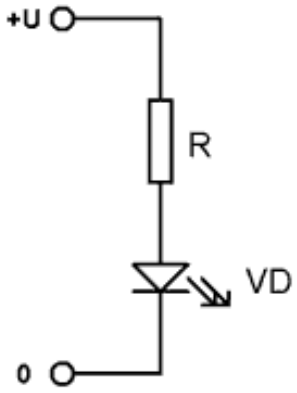


Рис. 2.4.47

Светодиод нужно подключить через резистор, который будет играть роль ограничителя тока. Рассчитать его сопротивление можно по закону Ома:

$$U_R = U_{\text{пит}} - U_{\text{д}} = 9\,\text{В} - 2\,\text{В} = 7\,\text{В};$$

$$R = \frac{U_R}{I} = \frac{7\,\text{В}}{0{,}02\,\text{А}} = 350\,\text{Ом}.$$

В этом вычислении руководствуемся правилом, что через последовательно подключенные элементы цепи протекает один и тот же ток, а падения напряжения на них в сумме дают значение напряжения источника.

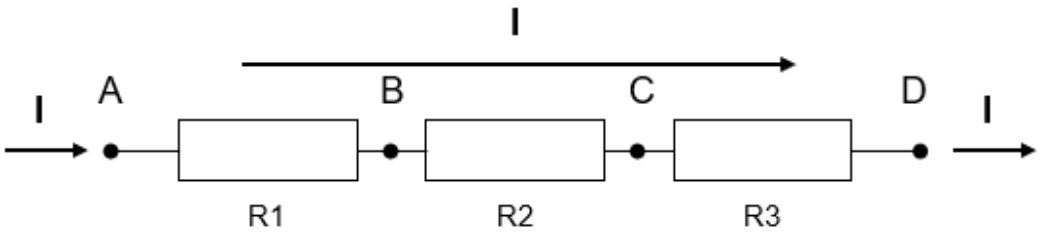


Рис. 2.4.48

$$U_{\text{общ}} = \varphi_A - \varphi_D = U_{R1} + U_{R2} + U_{R3},$$
$$I_{\text{общ}} = I_{R1} = I_{R2} = I_{R3}.$$

При параллельном соединении элементов, наоборот, падение напряжения на них остается неизменным (так как их выводы попарно объединены, и потенциалы в этих точках одинаковы), а ток разветвляется.

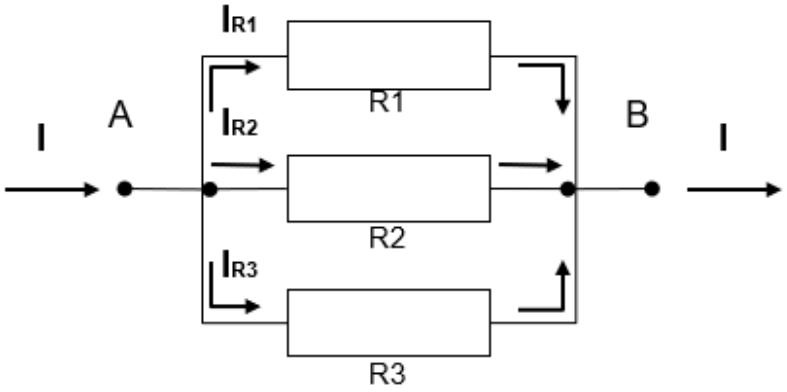


Рис. 2.4.49

$$U_{\text{общ}} = \varphi_A - \varphi_B = U_{R1} = U_{R2} = U_{R3},$$
$$I_{\text{общ}} = I_{R1} + I_{R2} + I_{R3}.$$

Нужно помнить, что любой проводник обладает такой характеристикой, как удельное сопротивление — способностью пропускать электрический ток. Например, удельное сопротивление меди составляет 0,017 Ом · мм^2/м при 20 °C, но с ростом температуры это значение будет увеличиваться, так как свободные заряды будут обладать большей энергией для хаотичного движения в межатомном пространстве. Так что провода, соединяющие элементы электрической цепи, имеют свое сопротивление, и на них тоже происходит падение напряжения.

Металлический провод можно представить как вытянутый цилиндр с известной площадью сечения и длиной. Чем меньше площадь сечения проводника, тем меньше электронов сможет протекать через него за одно и то же время, соответственно, сила тока уменьшится; а чем длиннее проводник, тем слабее электрический ток, так как электроны по пути будут «теряться», рассеиваться вследствие неидеальной структуры вещества.

При проектировании печатной платы (это диэлектрическая пластина с проводящими металлическими дорожками, соединяющими напаянные на нее электронные компоненты) выбор ширины проводящей дорожки определяется максимальной силой тока, протекающего через нее, и желаемым максимальным повышением температуры — как и на обычном резисторе, часть энергии на дорожке будет рассеиваться в виде тепла.

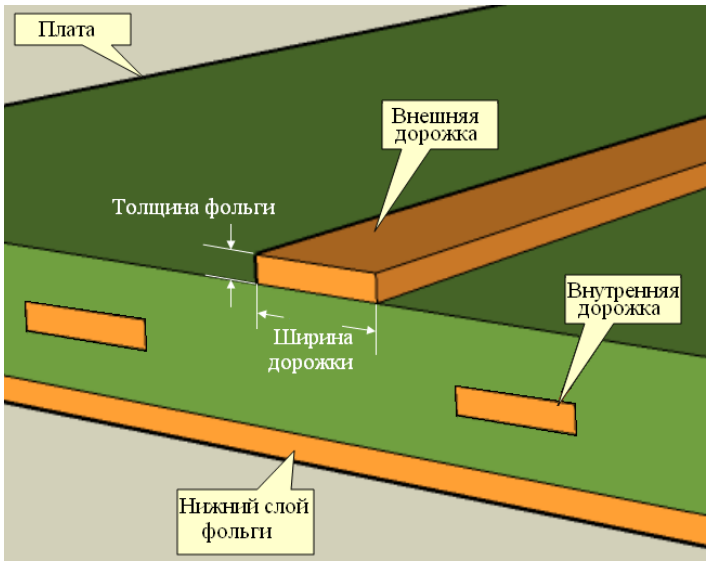


Рис. 2.4.50

Сопротивление дорожки увеличивается с ее длиной, а площадь сечения можно рассчитать как произведение толщины медного слоя на ширину дорожки. Чем больше площадь сечения, тем меньше потери энергии на дорожке. Для медных дорожек, расположенных на поверхности печатной платы, справедливо следующее равенство (исходя из международного стандарта проектирования печатных плат `IPC-2221`):

$$I = 0{,}048 \cdot \Delta T^{0{,}44} \cdot (W \cdot H)^{0{,}725}.$$

В данной формуле I — ток в А, ΔT — желаемое повышение температуры в градусах, W и H — соответственно ширина и толщина дорожки в мил — тысячных долях дюйма. Один мил равен 0,0254 мм.

Условие

Одной из важнейших задач при проектировании печатных плат является выбор правильной ширины дорожки. Кас-младший при разработке измерительного модуля столкнулся с проблемой — один из датчиков очень чувствителен к питающему напряжению. После размещения всех элементов на плате Кас-младший измерил длину дорожки, которая питает датчик, получилось 13,52 мм. Потребляемый датчиком ток равен 1,65 А, а отклонение напряжения не должно превышать 0,04 В. Помогите Касу-младшему подобрать минимальную ширину дорожки, чтобы датчик мог работать нормально. Толщина медной фольги 0,035 мм. Условия примите нормальными, температура 20 °C. Ответ дайте в миллиметрах, округлив до тысячных.

Решение

1. Из закона Ома находим максимально допустимое сопротивление:

$$I = \frac{U}{R},\ R = \frac{U}{I}.$$

2. По формуле сопротивления проводника находим требуемую ширину:

$$R = \rho\frac{l}{S},\ S = WH,\ R = \rho\frac{l}{WH},\ \frac{U}{I} = \rho\frac{l}{WH},\ W = \rho\frac{lI}{UH}.$$

3. $W = 0{,}270883$.

Ответ: 0,27 мм.

Задача 2.4.2.2. Зачем рассеивать, если можно накапливать? (2 балла)

Тема: схемотехника.

Теоретическая часть

Резистор поглощает электроэнергию и рассеивает часть мощности в виде тепла. Но существуют элементы, которые, напротив, могут запасать часть энергии источника и затем отдавать ее на нагрузку. Такие элементы называются реактивными. Первый из них, используемый практически во всех схемах, — это конденсатор, устройство, способное накапливать электрический заряд, когда к нему приложено напряжение. Конденсатор в простейшем случае представляет собой две проводящие пластинки, находящиеся на небольшом расстоянии друг от друга, с диэлектриком между ними.

Ток, проходящий через конденсатор, меняется пропорционально скорости изменения напряжения на нем. Способность конденсатора накапливать заряд характеризуется его емкостью, обозначаемой буквой С и измеряемой в Ф (фарадах). Емкость нескольких параллельно соединенных конденсаторов равна сумме их емкостей; для последовательно соединенных конденсаторов справедливо равенство:

$$\frac{1}{C_{\text{общ}}} = \frac{1}{C_1} + \frac{1}{C_2} + \cdots + \frac{1}{C_N}.$$

Конденсатор не рассеивает мощность, а сохраняет ее в виде внутренней энергии электрического поля, которая высвобождается при его разряде. Конденсатор можно представить как резистор, сопротивление которого обратно пропорционально его емкости и частоте протекающего через него сигнала. В цепи постоянного тока, когда конденсатор уже заряжен, между его обкладками не может протекать электрический ток, и он становится эквивалентен разрыву электрической цепи — резистору с бесконечным сопротивлением. В цепи переменного тока с увеличением частоты изменения входного сигнала эквивалентное сопротивление конденсатора будет уменьшаться.

Конденсаторы применяются в цепях переменного тока — то есть в таких цепях, где токи и напряжения меняются с течением времени. Они используются почти повсеместно: для фильтрации, генерации, шунтирования сигналов. Конденсаторы

также обязательно ставят между питающими выводами микросхем, чтобы сгладить пульсации питания. Низкочастотные колебания подавляет, как правило, танталовый конденсатор большей емкости (например, 10 мкФ), а с высокочастотными справляется керамический конденсатор меньшей емкости — 0,1 мкФ.

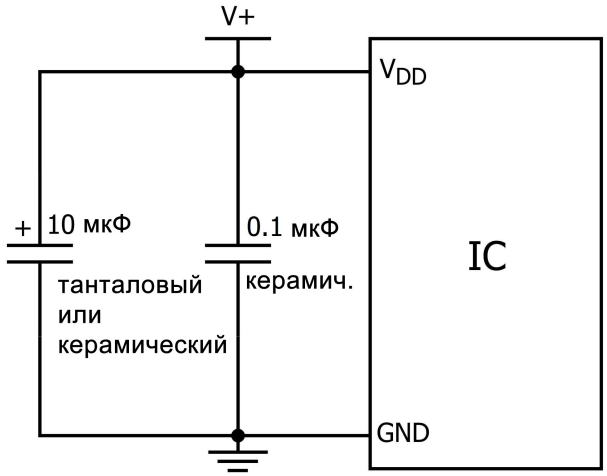


Рис. 2.4.51

Если заряженный конденсатор подключить к резистору, он будет постепенно разряжаться (напряжение на нем будет уменьшаться по экспоненте).

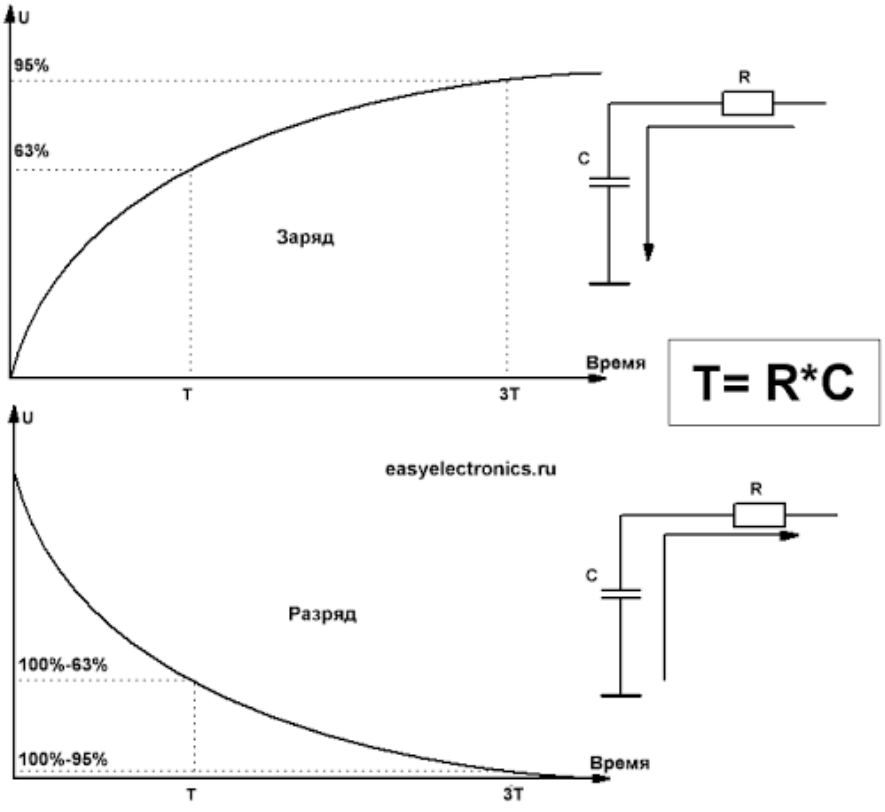


Рис. 2.4.52

Произведение сопротивления резистора в омах и емкости конденсатора в фарадах называют постоянной времени (τ); она измеряется в секундах. При подаче напряжения на вход RC-цепи на выходе оно установится не сразу, а время наблюдаемого переходного процесса (постепенного нарастания или убывания сигнала на выходе схемы при включении или выключении напряжения на входе) зависит от постоянной времени. Существует практическое правило для облегчения расчета RC-цепей: за время, равное 5τ, конденсатор заряжается или разряжается на 99%.

Условие

Кас приобрел на рынке электроники старый, но редкий цифровой датчик, который выдает прямоугольный сигнал с амплитудой от 0 до 3,3 В, периодом 1 мс и временем импульса 0,5 мс. На выходе датчика наблюдаются сильные высокочастотные помехи, которые могут вызвать неправильное распознавание логического уровня сигнала.

Для сглаживания помех Кас хочет добавить конденсатор С на выходе датчика. Логическая единица считается надежно распознанной, если напряжение на выходе датчика держится не ниже 99% от 3,3 В в течение как минимум 0,23 мс после перехода сигнала в состояние высокого уровня.

Внутреннее сопротивление датчика составляет $R = 1$ кОм. Помогите Касу определить максимальную емкость С, которую можно добавить для сглаживания помех, чтобы сигнал не терялся. Считайте, что логическая единица не может быть распознана после перехода сигнала в состояние низкого уровня. Ответ дайте в микрофарадах (мкФ) и округлите до сотых.

Решение

1. Зная время импульса и время, в течении которого должна держаться логическая единица найдем время зарядки конденсатора:

$$t = 0{,}5 - 0{,}23 = 0{,}27\,\text{мс}.$$

2. Для зарядки конденсатора до 99% требуется время равное 5τ, тогда:

$$\frac{t}{5} = RC,$$

отсюда

$$C = 54\,\text{мкФ}.$$

Ответ: 54 мкФ.

Задача 2.4.2.3. Индуктивность и паразиты (2 балла)

Тема: схемотехника.

Теоретическая часть

Электричество и магнетизм неразрывно связаны: любой движущийся заряд создает вокруг себя магнитное поле, поэтому и проводник с током всегда находится

в им же созданном магнитном поле. Но магнитное поле само влияет на заряды: на них действует сила Лоренца.

Сила Лоренца зависит от магнитной индукции (это силовая характеристика магнитного поля, показывающая, насколько сильно оно будет влиять на движение заряда), величины заряда, его скорости и угла, под которым заряд движется относительно магнитных линий.

Если перемещать в магнитном поле проводник, то свободные заряды в нем будут двигаться таким образом, что возникнет разность потенциалов, которую называют ЭДС электромагнитной индукции. Но не обязательно двигать проводник, чтобы наблюдать это явление: можно замкнуть его в контур и изменять величину магнитного потока, проходящего через него, наблюдая возникновение ЭДС электромагнитной индукции. Она будет тем больше, чем быстрее произошло изменение магнитного потока через контур. Направление индукционного тока, который начинает течь через контур, стремится компенсировать это изменение.

То же самое происходит в замкнутой электрической цепи, когда в ней меняется сила тока, например, при включении и выключении источника питания. Переменный электрический ток в цепи вызывает пропорциональные ему изменения магнитного потока, а это, в свою очередь, порождает в цепи ЭДС электромагнитной индукции, которая в данном случае называется просто ЭДС самоиндукции. ЭДС самоиндукции всегда препятствует изменению силы тока, протекающего по цепи.

Можно увеличить наведенную ЭДС самоиндукции в проводнике, свернув его в спираль, то есть, превратив его в катушку индуктивности. В каждом витке катушки при изменении электрического тока будет наводиться ЭДС самоиндукции, и все ЭДС отдельных витков будут складываться.

Способность катушки создавать магнитное поле, а значит, и величина возникающей ЭДС, характеризуется индуктивностью, которая обозначается буквой L и измеряется в генри (Гн). Катушки индуктивности широко применяются в радиочастотных и резонансных схемах, они также могут использоваться для фильтрации сигнала.

Если конденсаторы сохраняют мощность в виде энергии электрического поля, то катушки индуктивности — в виде энергии магнитного поля. Они способны отдать эту энергию, если прервать протекание тока через катушку. Пока ток через катушку постоянен, ее можно воспринимать как участок цепи, замкнутый накоротко — резистор с нулевым сопротивлением. С увеличением частоты сигнала, как и с увеличением индуктивности катушки, ее эквивалентное сопротивление растет — она противодействует изменению протекающего через нее тока.

Катушка индуктивности и конденсатор образуют колебательный контур. Если заряженный конденсатор подключить параллельно катушке индуктивности, то конденсатор будет разряжаться, а в цепи потечет ток, который вызовет в катушке ЭДС самоиндукции. Полярность сменится, и наведенная в катушке ЭДС самоиндукции перезарядит конденсатор, после чего в цепи снова потечет ток и так далее. Эти колебания напряжения и тока будут затухающими из-за того, что все элементы — проводники, конденсатор и катушка — имеют паразитное сопротивление, на котором теряется мощность, и часть энергии будет рассеиваться в виде тепла и радиоволн. Поэтому колебания необходимо поддерживать с помощью внешнего источника. Колебательные контуры широко используются в радиотехнике в качестве фильтров, например, чтобы настраивать приемник на нужную частоту.

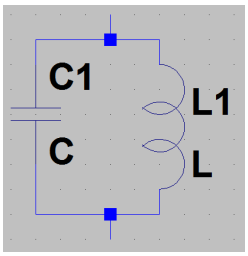


Рис. 2.4.53

На практике даже обыкновенный проводник обладает не только сопротивлением, но и емкостью (так как между его параллельно расположенными выводами есть разность потенциалов), и индуктивностью (так как при протекании тока через проводник создается магнитное поле, сопротивляющееся изменению этого тока).

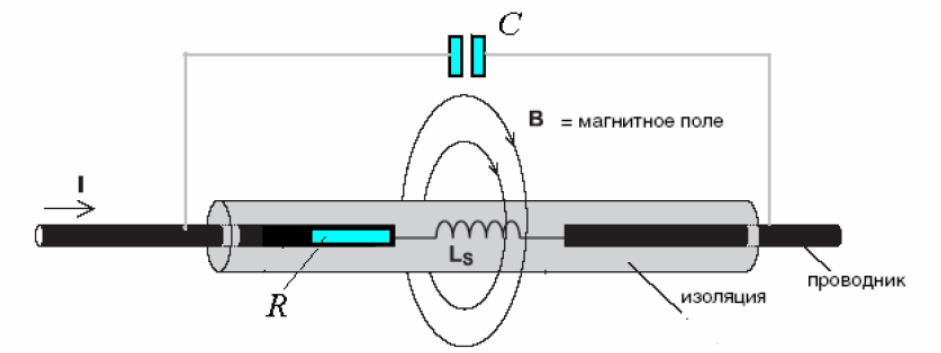


Рис. 2.4.54

Резисторы, конденсаторы и катушки индуктивности также обладают паразитными характеристиками — их можно представить как следующие эквивалентные схемы (схемы, в которых реальные компоненты замещены идеальными, но функционирующие так же, как исходные схемы).

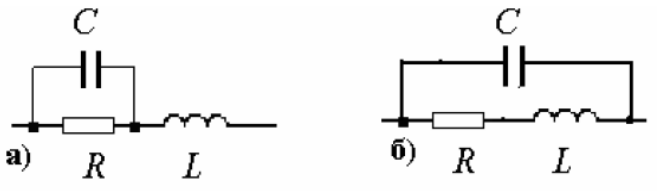


Рис. 2.4.55. Эквивалентные схемы резистора

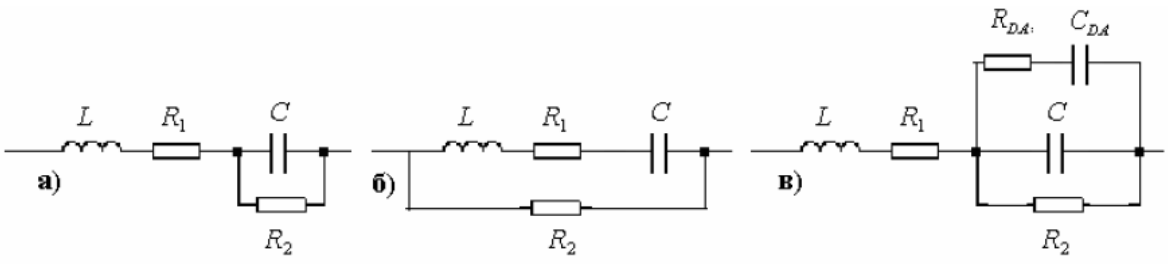


Рис. 2.4.56. Эквивалентные схемы конденсатора

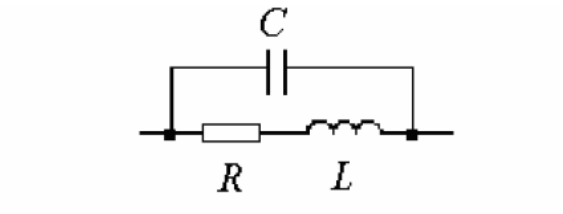


Рис. 2.4.57. Эквивалентная схема индуктивности

Проволочные и пленочные резисторы производятся в виде катушек, соответственно, с проводом из высокоомного металла и металлической пленки, но индуктивность пленочных резисторов меньше, что позволяет использовать их на высоких частотах. Выводы резисторов параллельны друг другу, поэтому между ними есть емкость. Выводы конденсатора представляют собой индуктивность на высоких частотах и имеют сопротивление, как и диэлектрик между обкладками конденсатора. Витки катушки индуктивности выполнены из провода, обладающего сопротивлением, а паразитная емкость возникает между любыми близко расположенными параллельными проводниками — в данном случае, витками катушки.

На самом деле эквивалентные схемы проводов, дорожек печатной платы, резисторов, конденсаторов и катушек индуктивности представляют собой фильтры и колебательные контуры. Из-за этого можно наблюдать, что схема, спроектированная без учета паразитных характеристик, не пропускает сигналы требуемой высокой частоты (фронты сигналов искажаются, их длительность значительно увеличивается по сравнению с длительностью импульсов) или самовозбуждается на выходе (генерирует ненужные колебания).

Длинные проводники ведут себя как антенны, а у изогнутых проводников увеличивается индуктивность, и они становятся петлевыми антеннами. Лучшим вариантом при разводке на рисунке ниже является вариант В, но проводники не должны быть расположены слишком близко друг к другу, чтобы уменьшить эффект создаваемой между ними паразитной емкости. Голубым цветом на рисунке показан полигон земли, который помогает в некоторой степени защитить схему от электромагнитных помех.

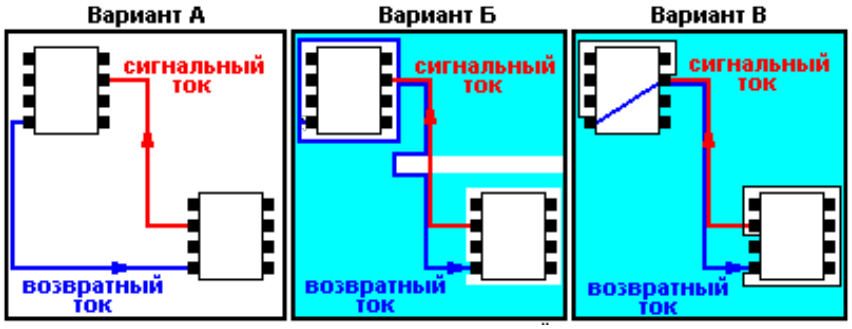


Рис. 2.4.58. Образование петли и щелевой антенны при некорректной разводке

Кроме того, между близко расположенными сигнальными проводниками образуется емкостная и индуктивная связь. Поэтому, если параллельное расположение проводников неизбежно, необходимо обеспечить между ними зазор минимум в три раза больше ширины проводников.

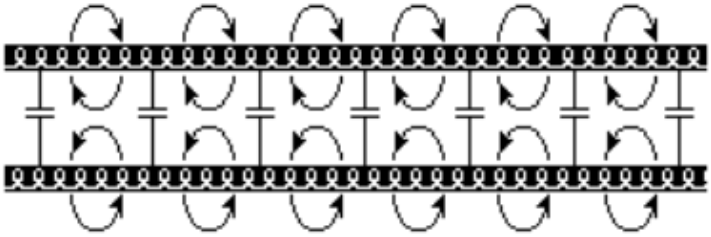


Рис. 2.4.59. Взаимосвязь между параллельными сигнальными проводниками

Расположенные параллельно катушки индуктивности ведут себя как трансформатор. Ток, протекающий через одну из катушек, влияет на ток в другой катушке и наоборот, из-за чего помехи передаются от одного сигнала к другому. Если невозможно разнести катушки дальше друг от друга на плате, то можно уменьшить их взаимное влияние с помощью ориентации: магнитная связь между катушками максимальна при параллельном размещении и минимальна при перпендикулярном.

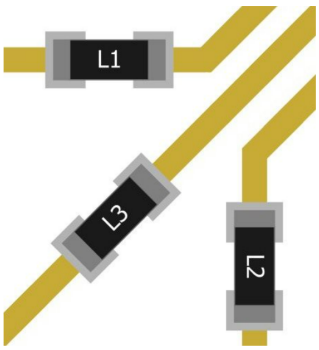


Рис. 2.4.60

Условие

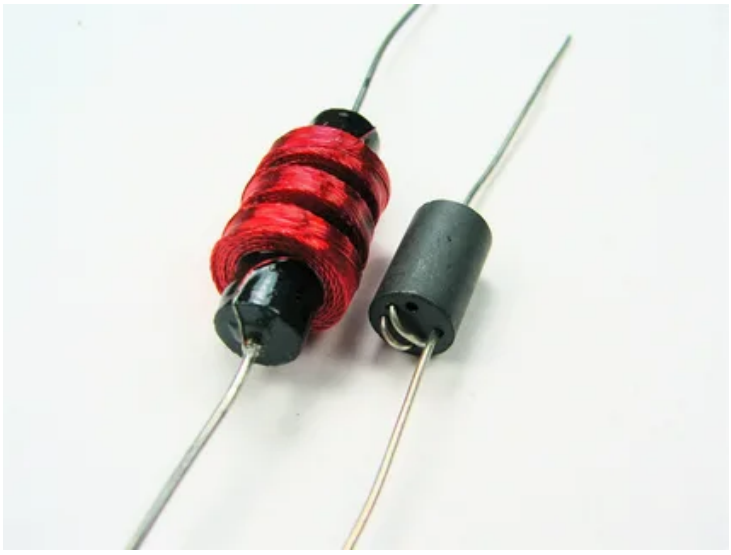


Рис. 2.4.61

Для чего устанавливают ферритовую бусину на линию аналогового питания микроконтроллеров?

Варианты ответа:

1. Для увеличения силы тока на аналоговом питании.
2. Для уменьшения тепловыделения микроконтроллера при работе с аналоговыми сигналами.
3. Для фильтрации высокочастотных помех, чтобы улучшить качество аналоговых сигналов.
4. Для стабилизации напряжения на аналоговом питании.

Решение

Для фильтрации высокочастотных помех, чтобы улучшить качество аналоговых сигналов.

Ответ: 3.

Задача 2.4.2.4. Полупроводники повсюду (2 балла)

Тема: схемотехника.

Теоретическая часть

Резисторы, конденсаторы и катушки индуктивности являются линейными элементами: изменение приложенного напряжения вызовет пропорциональное изменение тока. Также эти элементы относят к пассивным, так как у них нет встроенного источника энергии.

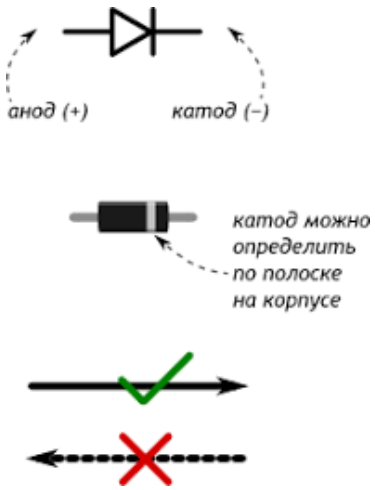


Рис. 2.4.62

Современная электроника строится на основе полупроводников: полупроводниковые элементы способны изменять свои электрические характеристики в зависимости от подаваемого на них напряжения и даже усиливать его.

Диод — самый простой полупроводниковый прибор, который проводит электрический ток в одном направлении и практически не пропускает его в обратном. На условном обозначении диода направление стрелки (анода) совпадает с направлением тока.

Диод является нелинейным пассивным элементом, поэтому нельзя описать зависимость тока от напряжения на нем простой формулой. Взаимосвязь тока и напряжения нелинейных элементов отражается в вольт-амперной характеристике (ВАХ).

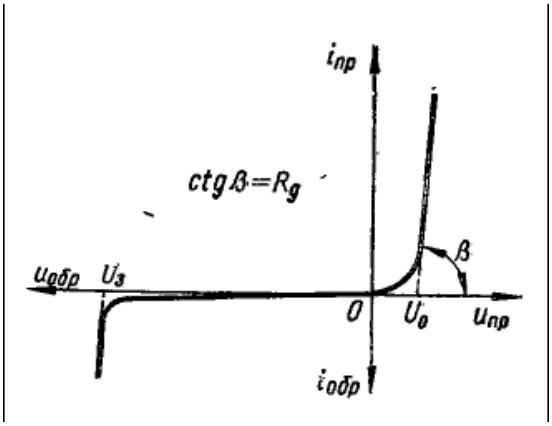


Рис. 2.4.63

ВАХ диода — это зависимость тока, протекающего через диод, от приложенного к нему напряжения. Чтобы открыть диод, обычно достаточно напряжения 0,3 В для германиевых диодов и 0,7 В для кремниевых, а для светодиодов это напряжение может достигать 2 В. Как видно из графика, при напряжении меньше, чем U_0, ток через диод мал и меняется слабо, а в открытом состоянии небольшое изменение напряжение приводит к резкому возрастанию тока. Этот ток не может достигать больше определенного значения, чтобы диод не сгорел. При подаче обратного напряжения ток также очень мал и почти не изменяется, пока не будет достигнуто напряжение пробоя. При пробое структура полупроводника разрушается, и диод выходит из строя, если это не стабилитрон (зенеровский диод), который работает в области пробоя.

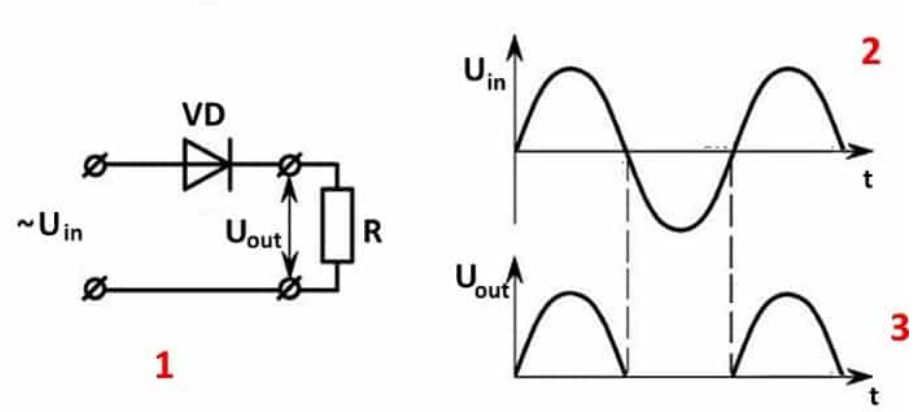


Рис. 2.4.64

Диоды часто используют как выпрямители — они помогают преобразовать переменный ток в постоянный. Кроме того, их ставят в качестве защиты, чтобы не

допустить разрядки накопительных элементов схемы или уберечь чувствительные приборы от подключения источника питания обратной полярности, а также для ограничения напряжения.

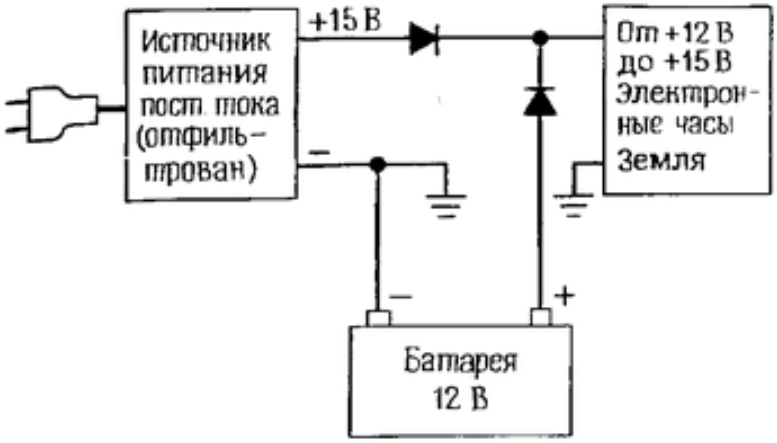


Рис. 2.4.65

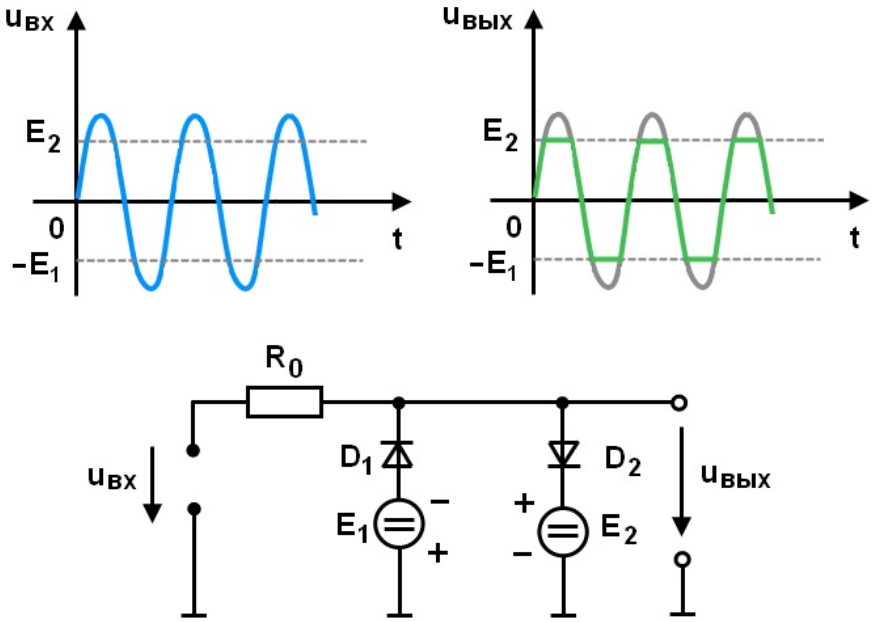


Рис. 2.4.66

Транзисторы, в отличие от диодов, являются активными компонентами — они способны усилить мощность входного сигнала. Они делятся на две крупные группы: биполярные и полевые. Биполярные транзисторы используются в основном в аналоговой технике, а полевые — в цифровой.

У биполярного транзистора три вывода: коллектор, база и эмиттер. Цепи база-эмиттер и база-коллектор работают как диоды. Рассмотрим работу npn-транзистора: если между коллектором и эмиттером есть разность потенциалов, то через коллектор ток потечет только в том случае, если на базу будет подано напряжение, открывающее диод база-эмиттер. Ток, протекающий через коллектор, пропорционален току базы:

$$I_к = \beta I_б,$$

где β — коэффициент усиления по току.

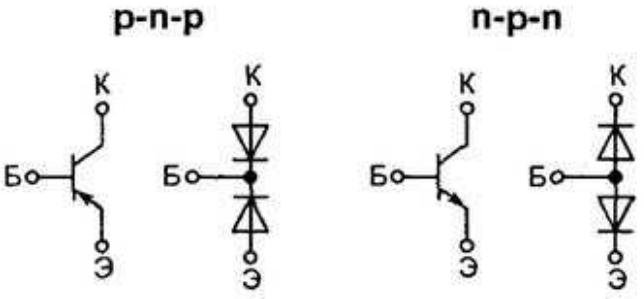


Рис. 2.4.67

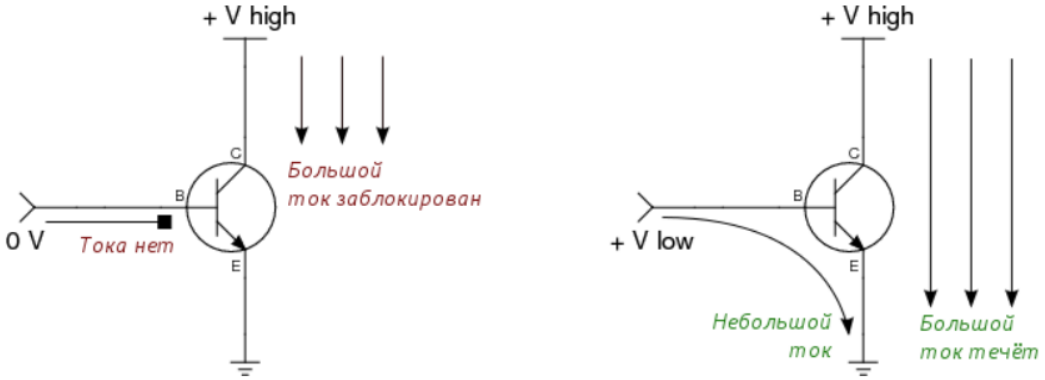


Рис. 2.4.68

Выходит, что небольшим током базы транзистора можно управлять большим током коллектора. При этом слишком большой ток базы создавать нельзя: необходимо сохранять напряжение между базой и эмиттером около 0,7 В, чтобы транзистор был открыт, и через эмиттер и коллектор протекал ток.

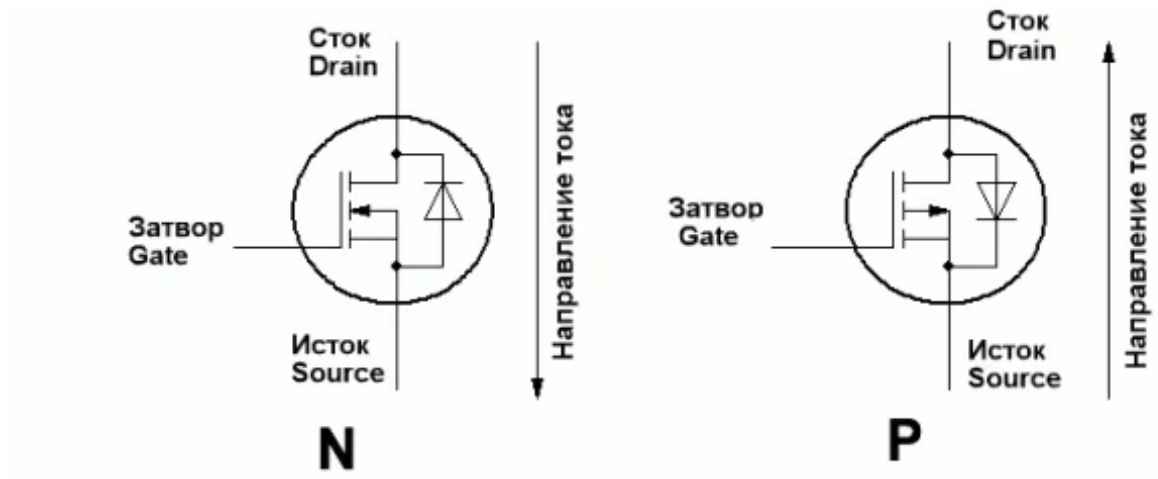


Рис. 2.4.69

У полевого транзистора также три вывода: затвор (*gate*), сток (*drain*) и исток (*source*). Если приложить к стоку транзистора с n-каналом положительный, относительно истока, потенциал и подать положительное напряжение затвор-исток, то транзистор откроется, и через сток и исток потечет ток.

Логические элементы сложных цифровых электронных устройств, в том числе микропроцессоров и микроконтроллеров, на самом деле состоят из транзисторов, но не дискретных (отдельных), а произведенных на одном полупроводниковом кристалле и очень маленьких. Микроконтроллер может пропускать через свои выводы только небольшие токи: например, через один вывод общего назначения микроконтроллера `STM32F103` может протекать ток не более 20 мА, а рекомендуемое значение — 8 мА. При этом микроконтроллеры часто должны управлять мощной нагрузкой. Если нагрузка включается при подаче на нее напряжения и требует большого тока, напрямую подключить ее к микроконтроллеру нельзя — он сгорит. Здесь и помогут транзисторы — управляющий вывод микроконтроллера соединяется с базой или затвором транзистора, а нагрузка включается между землей или питанием и одним из выводов транзистора, в зависимости от его типа.

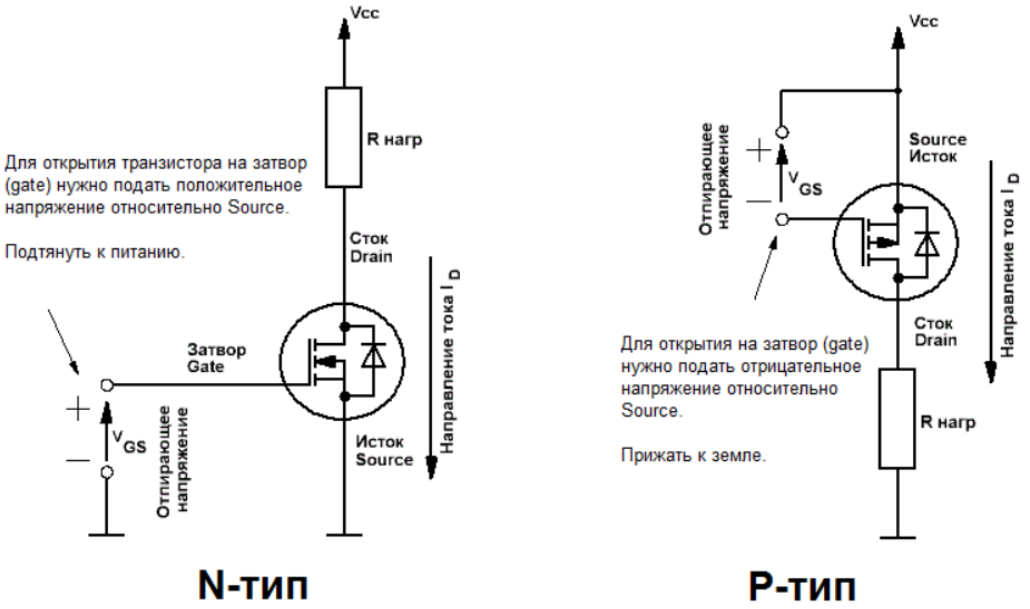


Рис. 2.4.70

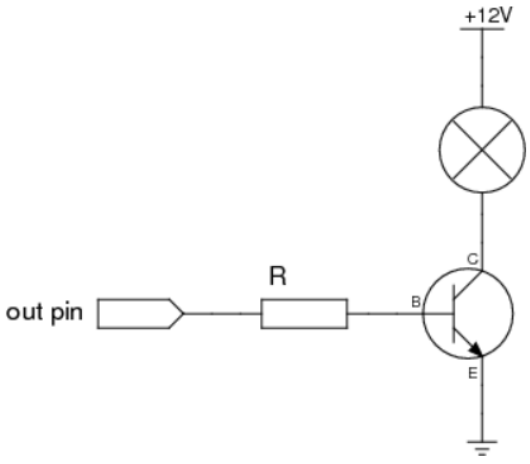


Рис. 2.4.71

Стоит помнить о том, что при протекании больших токов на транзисторе выделяется много тепла, и при проектировании электронной схемы необходимо также продумать теплоотвод.

Условие

Разрабатывая коммутацию электропитания своего спутника, космокот Кас решил заложить возможность отдельного включения питания 5 В для электромагнитных катушек, так как они не нужны все время. Кас решил, что для этой цели ему подойдут два транзистора: n-канальный транзистор `BSS138` и p-канальный транзистор `IRF7404`, способный пропускать довольно большой ток (а катушки потребляют как раз большой ток).

Вкратце задумка Каса выглядит следующим образом: когда приходит высокий уровень сигнала `PWR_ON` на затвор одного из транзисторов, он открывается и начинает пропускать ток через сток, соединенный с затвором следующего транзистора. Из-за этого следующий транзистор тоже открывается, таким образом обеспечивая коммутацию между источником напряжения 5 В и питанием катушек. Для защиты от ложного срабатывания и помех к транзисторам также подключены резисторы номиналом 100 кОм.

Помогите космокоту развести плату правильно! Выберите правильный вариант трассировки участка платы из четырех вариантов, представленных ниже.

У транзистора `BSS138` корпус `SOT-23`, а у `IRF7404` — `SOP-8`. Транзисторы на рисунке обозначены буквой Q, а резисторы — R.

На рисунках ниже входящие и выходящие из участка схемы сигналы обозначены следующим образом:

- `PWR_ON` — сигнал, разрешающий подачу напряжения 5 В на питание катушек;
- `PWR` — питание катушек;
- `+5V` — источник напряжения 5 В;
- `GND` — земля.

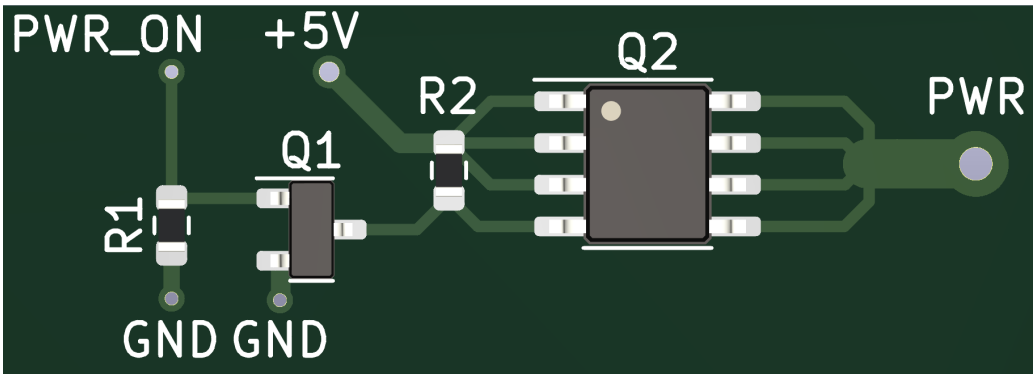


Рис. 2.4.72

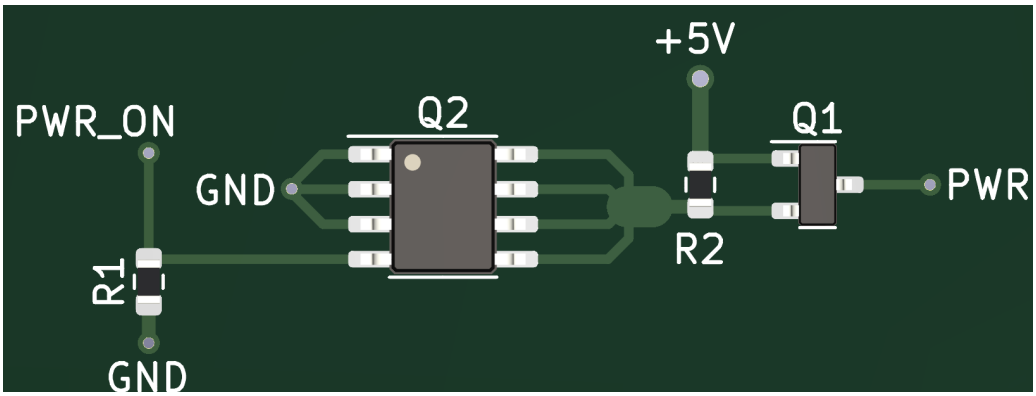


Рис. 2.4.73

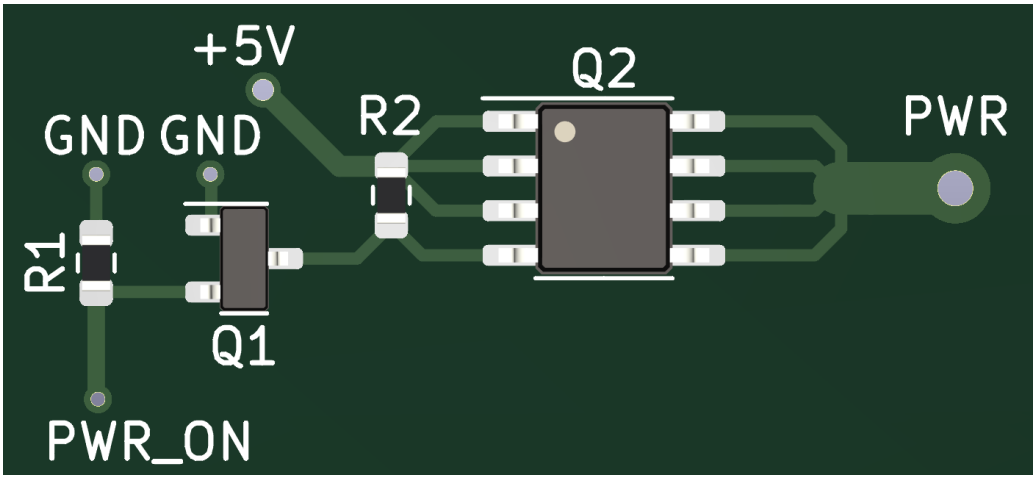


Рис. 2.4.74

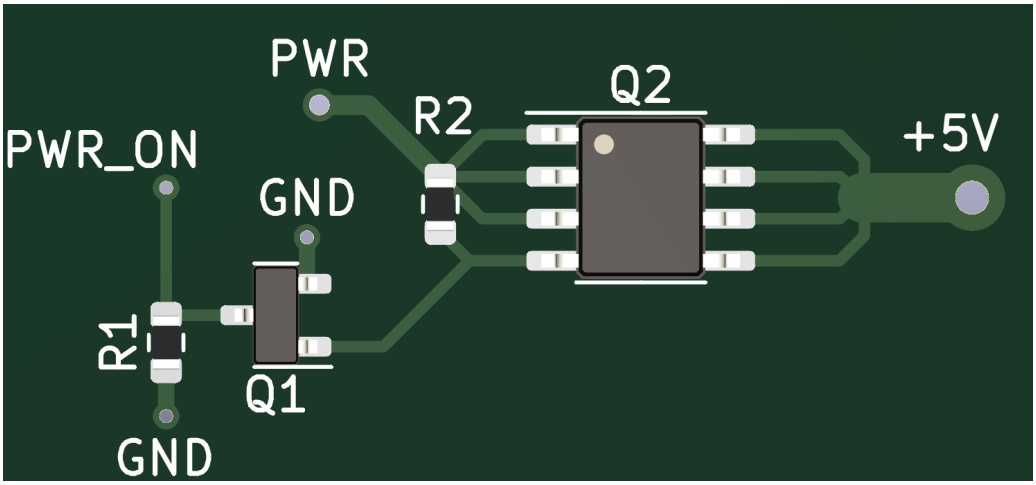


Рис. 2.4.75

Решение

Верная трассировка на рис. 2.4.72. Управляющий сигнал подается на затвор n-канального транзистора, сток которого подключен к затвору *p*-канального тран-

зистора. Питание подается на исток p-канального транзистора, который может пропустить большой ток, питающий катушки. Таким образом, питание катушек должно сниматься со стока p-канального транзистора. Для верного подключения также необходимо заземлить исток n-канального транзистора.

Правильная схема на рис. 2.4.76.

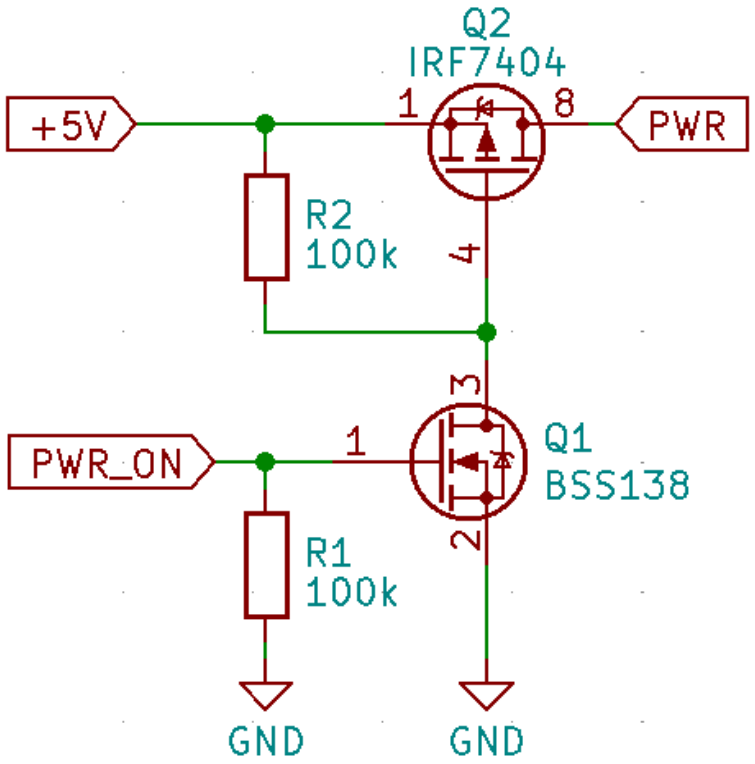


Рис. 2.4.76

Ответ: 1.

Задача 2.4.2.5. Источники питания (2 балла)

Тема: схемотехника.

Теоретическая часть

Система электропитания — важная часть любого спутника. Спутники малого формата, такие как кубсаты, в большинстве случаев используют в качестве первичного источника энергии солнечные батареи. Но полученную энергию необходимо преобразовать, чтобы установить некоторый постоянный уровень напряжения, не зависящий от внешних условий; накапливать часть энергии в аккумуляторных батареях, чтобы обеспечить работу спутника в тени, а также распределить электроэнергию между подсистемами, которые зачастую имеют различные требования по уровню напряжения и мощности. Существует несколько способов преобразовать энергию, полученную от солнечных батарей.

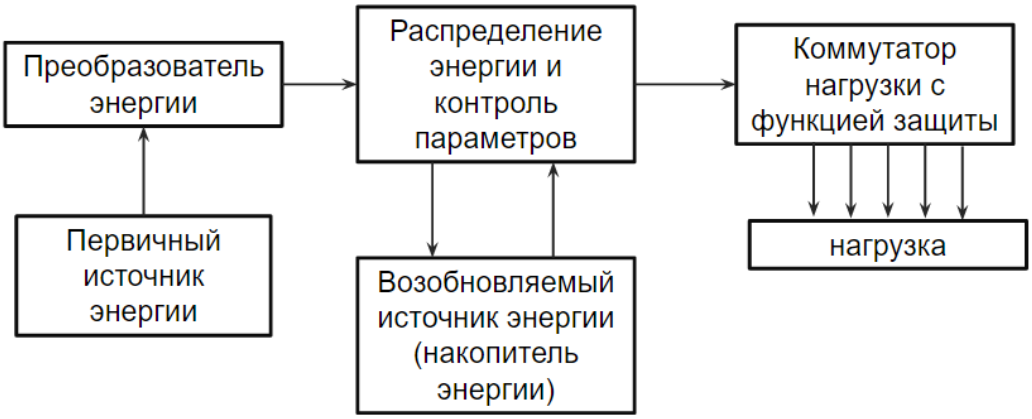


Рис. 2.4.77

Самый простой способ преобразовать напряжение — это использовать делитель напряжения.

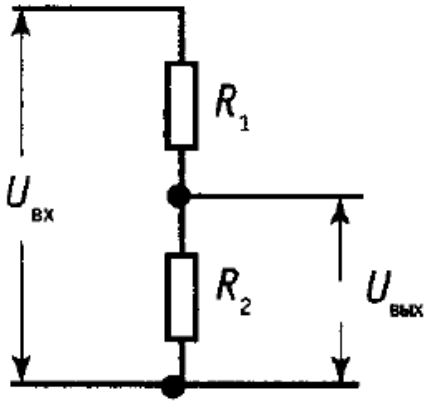


Рис. 2.4.78

Напряжение на выходе такой схемы будет составлять долю от входного напряжения, зависящую от значений сопротивления резисторов R_1 и R_2. Предположим, что в точке, где снимается выходное напряжение, нагрузки нет (так называемый режим холостого хода — ток туда течь не будет). Тогда по закону Ома мы легко сможем посчитать сначала силу тока, протекающего в этой цепи, а затем и выходное напряжение:

$$I = \frac{U_{\text{вх}}}{R_1 + R_2};$$

$$U_{\text{вых}} = I \cdot R_2 = \frac{U_{\text{вх}} \cdot R_2}{R_1 + R_2}.$$

Делитель напряжения полезен, когда нужно ограничить выходное напряжение, но оно всегда будет пропорционально напряжению на входе, то есть будет меняться, если первичный источник энергии нестабилен, что не подходит для питания других устройств.

Стабилитрон или зенеровский диод может обеспечить почти неизменное выходное напряжение при меняющемся питающем токе: для этого его нужно поставить в цепь в обратном включении.

При постоянном токе стабилитрон, как и любой диод, можно представить в виде резистора с сопротивлением, равным динамическому сопротивлению диода — то есть мгновенному значению отношения напряжения к току. Динамическое сопротивление зенеровского диода в режиме стабилизации изменяется обратно пропорционально току. Его выгодно использовать, когда схема питается именно от источника тока, не зависящего от напряжения. Недостатками является то, что нельзя установить выходное напряжение на заданное значение и недостаточное сглаживание пульсации входного напряжения.

Предположим, необходимо стабилизировать с помощью зенеровского диода напряжение, изменяющееся по синусоидальному закону. Напряжение на выходе, которое будет поддерживать стабилитрон, определяется его характеристикой, которую можно посмотреть в `datasheet` на него — обратным напряжением пробоя. Чтобы ввести стабилитрон в режим пробоя, необходимо обеспечить протекание через него обратного тока пробоя — его значение также можно узнать из спецификации на диод.

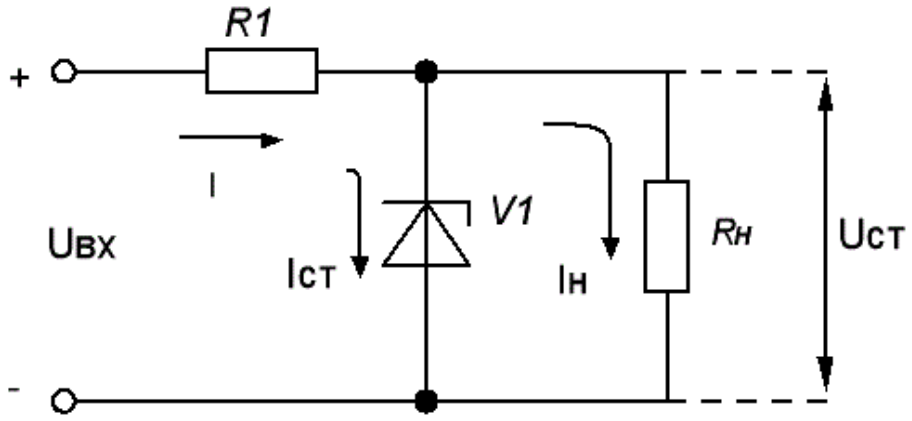


Рис. 2.4.79

Чтобы через стабилитрон протекал необходимый ток, нужно верно подобрать номинал резистора R_1. Это можно сделать, рассмотрев наихудший случай: представим, что напряжение на входе минимально. Тогда, если стабилитрон находится в режиме пробоя, на резисторе будет падать напряжение, равное разнице входного напряжения и напряжения пробоя. Сила тока, протекающего через стабилитрон, равна силе тока на резисторе, так как они соединены последовательно (рассматриваем режим холостого хода, когда нагрузка не подключена). Тогда сопротивление резистора можно рассчитать по следующей формуле:

$$R_1 = \frac{U_{\text{вх.мин.}} - U_{\text{пробоя}}}{I_{\text{пробоя}}}.$$

Для синусоидального сигнала, меняющегося от 9 до 11 В, и стабилитрона с обратным напряжением пробоя 6,2 В напряжение на выходе без нагрузки будет незначительно колебаться под значением 6,2 В.

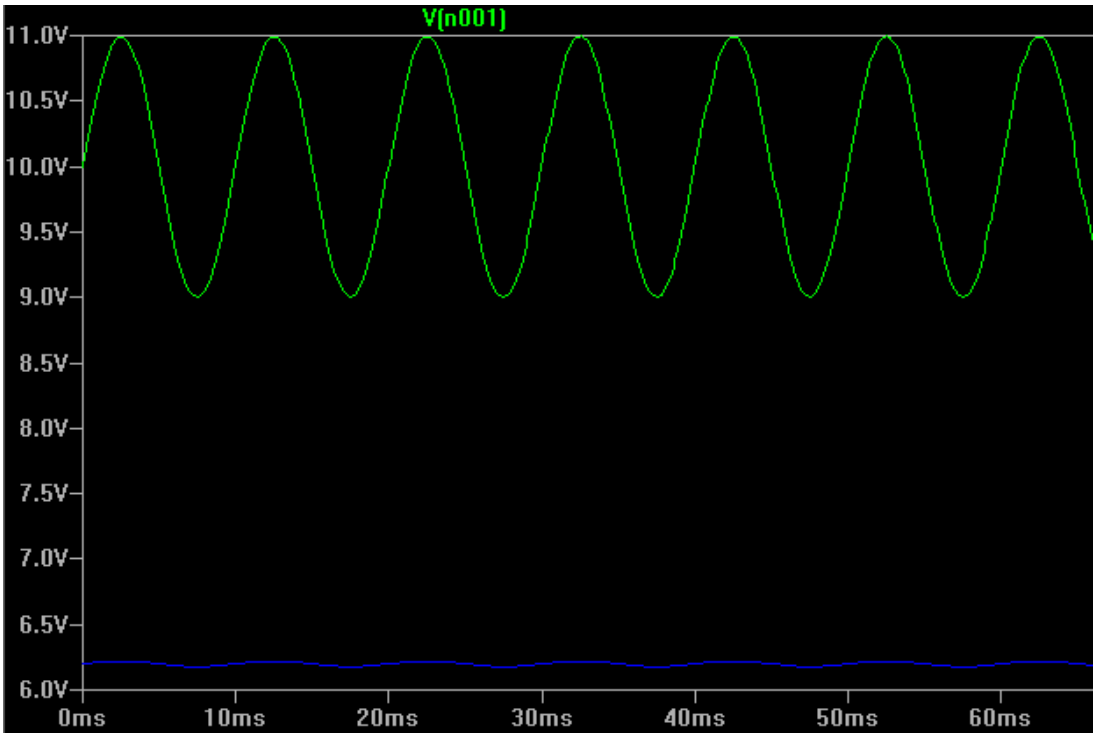


Рис. 2.4.80

Но если подключить низкоомную нагрузку, в которую потечет большая часть тока, напряжение на выходе стабилитрона будет сложно назвать стабильным, так как в какой-то момент он выйдет из режима пробоя.

Улучшить схему со стабилитроном можно, отделив зенеровский диод от нагрузки эмиттерным повторителем. Эмиттерный повторитель — схема, в которой выходной сигнал снимается с эмиттера. Напряжение на нем почти равно напряжению на базе. Входное сопротивление такой схемы значительно больше выходного, поэтому эмиттерный повторитель обеспечивает усиление по току и, так как напряжение практически неизменно, по мощности.

Ток через базу мал и, соответственно, значительно уменьшается мощность, рассеиваемая на стабилитроне. С целью снижения пульсаций тока в стабилитроне можно использовать дополнительный источник тока для его питания или добавить RC-цепочку в качестве фильтра низких частот.

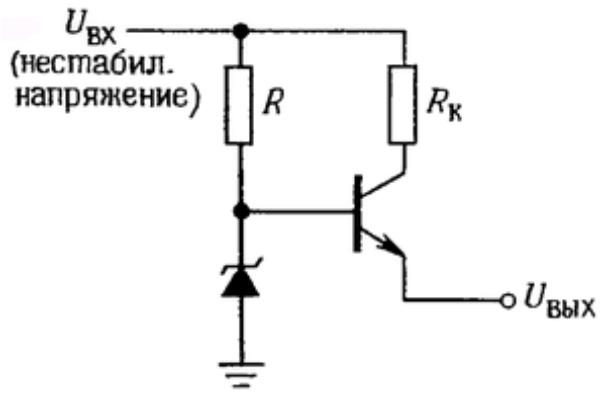


Рис. 2.4.81

Схему с эмиттерным повторителем также можно улучшить, добавив дифференциальный усилитель на транзисторах T_3 и T_4, который вместе со схемой Дарлингтона T_1 работает в качестве мощного операционного усилителя. Схема Дарлингтона — это два транзистора с соединенными коллекторами, позволяющая получить высокий коэффициент усиления. Делитель напряжения R_1, R_2 охватывает усилитель обратной связью и выдает на выход усиленное опорное напряжение.

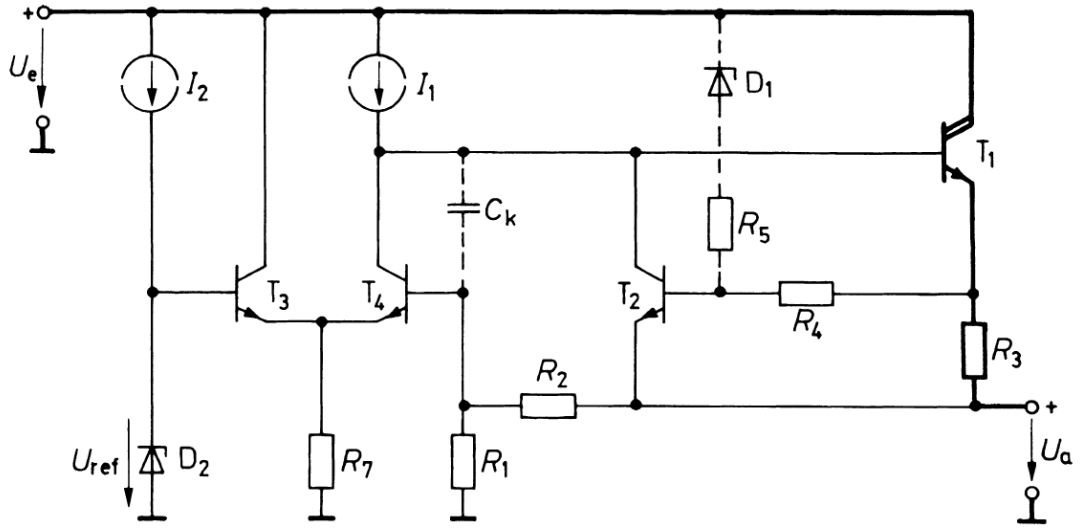


Рис. 2.4.82

Данную схему собирать из отдельных компонентов нет особого смысла, так как разброс характеристик транзисторов не позволит добиться хороших результатов. Чтобы получить стабильное напряжение заданной величины, используем готовую интегральную схему (выполненную на одной полупроводниковом кристалле) `LM317` — это трехвыводной регулируемый стабилизатор.

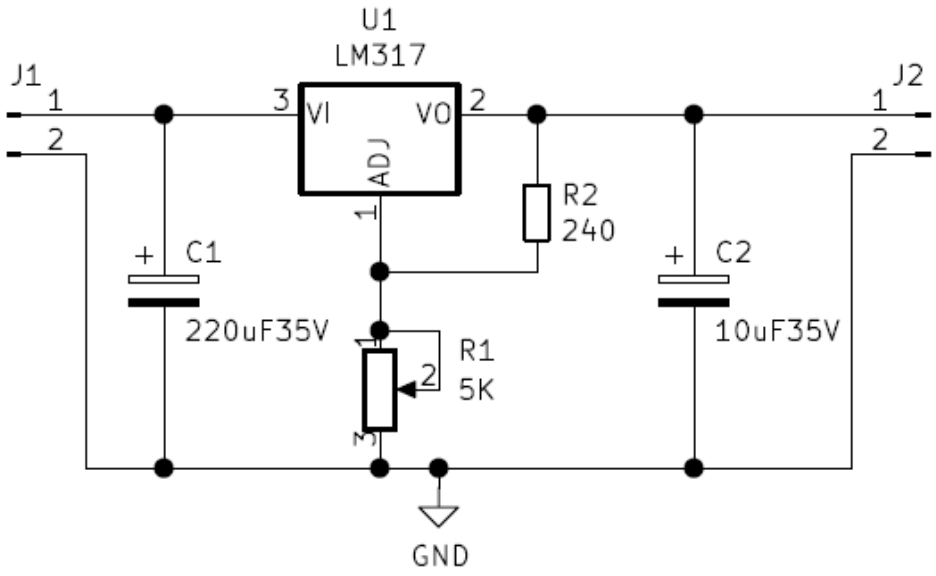


Рис. 2.4.83

Его особенность состоит в том, что он имеет три вывода: вход, выход и регулируемый вывод, причем между регулировочным выводом и выходом всегда поддерживается постоянное напряжение U_{REF}, обусловленное внутренней схемой стабилиза-

тора. Значение этого напряжения можно посмотреть в спецификации на конкретный стабилизатор; для LM317 оно равно 1,25 В. Ток, потребляемый регулировочным выводом, пренебрежимо мал. Регулировочный вывод и выход соединяют через делитель напряжения (резисторы R_1 и R_2), чтобы можно было настраивать значение напряжения на выходе. Рекомендации по значению сопротивления резистора R_2 обычно даются изготовителем, например, для LM317 подойдет резистор номиналом 240 Ом.

Если вместо резистора R_1 поставить потенциометр, то изменением его сопротивления можно добиться различных значений выходного напряжения. Рассчитать значение сопротивления R_1 для желаемого выходного напряжения можно следующим образом:

$$U_{OUT} = U_{REF}\left(1 + \frac{R_1}{R_2}\right).$$

Схемы линейных стабилизаторов сильно нагреваются, если использовать их постоянно (поэтому их используют при малой нагрузке и периодических кратковременных включениях), поэтому необходимо озаботиться отводом тепла, который можно обеспечить с помощью радиаторов.

Другой вид преобразователей напряжения — импульсные преобразователи. В них используется свойство катушки индуктивности накапливать и затем отдавать энергию. Для его работы также требуется тактовый генератор, который будет управлять транзистором в режиме ключа. Транзистор периодически прикладывает к катушке индуктивности часть напряжения, а катушка запасает энергию в магнитном поле и затем передает ее на конденсатор выходного сглаживающего фильтра. Управление выходом осуществляется за счет изменения длительности импульсов генератора.

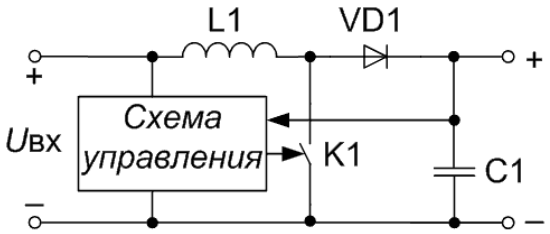


Рис. 2.4.84

Такие схемы могут не только стабилизировать напряжение, но и повышать его за счет магнитной энергии, которую накапливает катушка индуктивности.

Условие

Кас-младший проектирует систему питания спутника. Ему нужно согласовать напряжение солнечных батарей с бортовым напряжением 3,7 В. Для этого он решил использовать стабилизатор MIC29152, вот его типовая схема подключения.

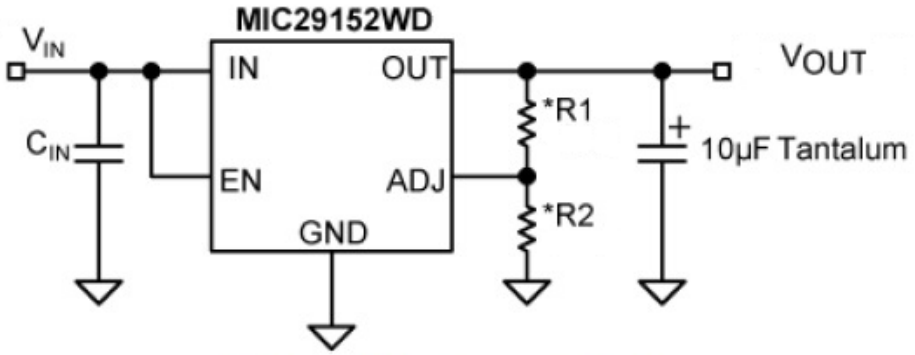


Рис. 2.4.85

Помогите Касу-младшему найти номинал резистора R_1, чтобы на выходе стабилизатора было 3,7 В. Известно, что $R_2 = 62$ Ом. Дайте ответ в омах, округлив до десятых. Током через пин `ADJ` можно пренебречь (подсказка: нужно поискать нужную для решения информацию в `datasheet` на стабилизатор, используйте типичные значения параметров).

Решение

1. Из `datasheet`'а на стабилизатор находим, что $V_{ADJ} = 1{,}24$ В. По формуле
$$R_1 = R_2 \left(\frac{V_{out}}{V_A DJ} - 1 \right)$$
находим сопротивление резистора.
2. $R_1 = 123$ Ом.

Ответ: 123 Ом.

Задача 2.4.2.6. Откуда мы знаем то, что мы знаем, или учимся читать `datasheet` (4 балла)

Тема: схемотехника.

Условие

Космокот Кас-младший выбирает магнитометр для своего спутника на базе микроконтроллера `STM32`. Главные критерии отбора следующие:

1. Точность измерений: магнитометр должен обеспечивать точность измерений не хуже 0,09 μT.
2. Диапазон измерений: магнитометр должен иметь диапазон измерений не менее ± 8 Гс (гаусс).
3. Рабочий температурный диапазон: магнитометр должен работать в условиях от -40 °C до $+85$ °C.
4. Интерфейсы: магнитометр должен поддерживать интерфейс `I2C`.
5. Энергопотребление: энергопотребление должно быть менее 100 мкА в режиме ожидания.

Помогите Касу-младшему выбрать наиболее подходящий магнитометр из следующего списка:

1. `HMC5883L`;
2. `LIS3MDL`;
3. `MAG3110`;
4. `MMC5883MA`.

Решение

Находим нужные данные в спецификациях на сенсоры и определяем, что заданным требованиям удовлетворяет только вариант 4.

Ответ: 4.

Задача 2.4.2.7. Как найти Солнце (6 баллов)

Тема: разработка полезной нагрузки.

Теоретическая часть

Используя различные схемотехнические элементы, можно управлять электрической энергией — осталось только ее найти. Находясь на Земле и нуждаясь в энергии, можно воспользоваться розеткой или, в крайнем случае, бензиновым генератором. В космосе такая роскошь непозволительна, потому спутники (за редким исключением), подобно цветам, вынуждены тянуться (солнечными батареями) к Солнцу. Но если растения справляются с этим самостоятельно, то спутникам необходима помощь инженеров.

Датчик Солнца — это устройство, используемое на спутниках и космических аппаратах для определения положения Солнца относительно ориентации самого аппарата. Они играют ключевую роль в системах ориентации и стабилизации спутников, позволяя им поддерживать правильное направление в пространстве. В зависимости от задач, поставленных перед космическим аппаратом, могут использоваться различные типы датчиков Солнца.

Грубые датчики применяются для приблизительного определения направления на Солнце. Они имеют простую конструкцию, состоящую из нескольких широких фотодиодов, каждый из которых покрывает определенный сектор неба. Если солнечный свет попадает на один из них, это означает, что Солнце находится в соответствующем секторе. Несмотря на простоту, такие датчики выполняют важную функцию, обеспечивая начальную ориентацию аппарата и помогая ему удерживать положение в пространстве. Они особенно полезны в ситуациях, когда необходима быстрая и грубая ориентация, например, сразу после выхода на орбиту.

Аналоговые датчики Солнца, также известные как дифференциальные, предлагают более высокую точность по сравнению с грубыми датчиками. Они состоят из ряда фоточувствительных элементов, например, фотодиодов, расположенных на плоскости или в виде полусферы. Когда солнечный свет попадает на датчик, он освещает различные фотодиоды с разной интенсивностью в зависимости от угла падения

солнечных лучей. Сигналы с фотодиодов анализируются, и на основе разности интенсивности света между ними определяется точное направление на Солнце. Такие датчики позволяют более точно ориентировать спутник, что особенно важно для задач, требующих стабильного удержания направления, например, для спутников связи или наблюдения Земли.

Рис. 2.4.86

Цифровые датчики Солнца представляют собой наиболее современные и точные устройства в этой категории. Они часто используют матричные фотосенсоры, такие как `CMOS` и `CCD` (`CCD` — `Charge-Coupled Device` и `CMOS` — `Complementary Metal-Oxide-Semiconductor`), на которые проецируется изображение Солнца. Компьютер анализирует положение светлого пятна на матрице и определяет угол падения солнечных лучей с высокой точностью. Эти датчики способны обеспечивать точную ориентацию даже при сложных условиях, когда необходима высокая стабильность и точность как в космических миссиях, так и для спутников, работающих на солнечных батареях.

Для спутников, выполняющих научные миссии, наблюдения Земли или другие задачи, стабильная ориентация относительно Солнца необходима для точного выполнения своих функций. В сочетании с другими датчиками, такими как звездные датчики, гироскопы и магнитометры, датчики Солнца обеспечивают комплексное решение для определения ориентации и стабилизации космического аппарата в пространстве. Они не только поддерживают корректное функционирование спутника, но и продлевают его жизненный цикл за счет эффективного управления энергопотреблением и ориентацией.

Условие

Кас, прочитав статьи о датчиках Солнца, решил усовершенствовать свой корабль, заменив в нем систему фотодиодов на новые матричные датчики, позволяющие определять направление на Солнце. Установив датчики, Кас понял, что они возвращают в его бортовой компьютер только два числа, а не три, как должно быть в векторе. Тогда он достал инструкцию и внимательно с ней ознакомился. Там было сказано, что датчик возвращает координаты пикселя матрицы, на который попало больше всего света. Также из описания устройства датчика Кас узнал, что размер матрицы $22{,}3 \times 14{,}9$ мм, размер пикселя $0{,}025$ мм. На расстоянии 5 мм от матрицы установлена металлическая пластина с небольшим отверстием ровно по центру.

Помогите Касу рассчитать вектор направления на Солнце в текущий момент

времени, если известно, что датчик вернул числа: 117, 351. В ответе необходимо дать единичный вектор в системе координат датчика.

Координаты пикселей считать от левого верхнего края:

- ось Y — длинная сторона матрицы (направлена вправо из левого верхнего края матрицы);
- ось X — короткая (направлена вниз из левого верхнего края матрицы);
- нумерация пикселей начинается с единицы.

Ноль системы координат датчика считать центром металлической пластины, оси X и Y сонаправлены с осями координат матрицы, ось Z дополняет систему до правой тройки.

При расчете считайте, что свет попадает ровно в центр пикселя. Ответ округлите до тысячных.

Решение

Чтобы учесть, что свет попадает ровно в центр пикселя, нужно скорректировать координаты пикселя, добавив половину его размера. Затем определим обратный вектор направления на Солнце.

1. Определение координат центра матрицы.
   Центр матрицы в пикселях:

$$X_{mid} = 298,\ Y_{mid} = 446.$$

2. Определение координат пикселя с учетом его центра.
   Координаты пикселя (117, 351) относительно центра матрицы:

$$\Delta x = 117 - 0{,}5 - 298 = -181{,}5 \text{ пикселей},$$

$$\Delta y = 351 - 0{,}5 - 446 = -95{,}5 \text{ пикселей}.$$

   Переводим координаты из пикселей в миллиметры:

$$X_{mm} = -181{,}5 \cdot 0{,}025 \text{ мм/пиксель} = -4{,}5375 \text{ мм},$$

$$Y_{mm} = -95{,}5 \cdot 0{,}025 \text{ мм/пиксель} = -2{,}3875 \text{ мм}.$$

3. Определение вектора направления на Солнце.
   Определим координаты вектора направления на Солнце, зная, что $z = -5$ мм:

$$R = (-4{,}5375 \text{ мм}, -2{,}3875 \text{ мм}, -5 \text{ мм}).$$

4. Нормировка вектора.
   Вычислим длину вектора:

$$|R| = \sqrt{(-4{,}5375)^2 + (-2{,}3875)^2 + (-5)^2} \approx 7{,}16164 \text{ мм}.$$

   Найдем единичный вектор:

$$R_{norm} = \left(-\frac{4{,}5375}{7{,}16164}, -\frac{2{,}3875}{7{,}16164}, -\frac{5}{7{,}16164}\right) \approx (-0{,}634, -0{,}333, -0{,}698).$$

5. Найдем обратный вектор.
   Чтобы получить обратный вектор, необходимо изменить знаки всех компонентов:
$$R_{reverse} = (0{,}634, 0{,}333, 0{,}698).$$

Ответ: единичный вектор направления на Солнце в системе координат датчика: $R_{ans} = R_{reverse} \approx (0{,}634, 0{,}333, 0{,}698)$.

Задача 2.4.2.8. Сейчас вылетит птичка! (6 баллов)

Тема: схемотехника.

Теоретическая часть

Спутниковая фотосъемка является одним из важнейших инструментов в современном мире, обеспечивающим беспрецедентные возможности для мониторинга и изучения Земли, космоса и атмосферы. Эта технология оказывает значительное влияние на самые разные области, от сельского хозяйства до обороны и научных исследований.

Одной из ключевых причин важности спутниковой фотосъемки является ее способность предоставлять данные в реальном времени или с минимальной задержкой, охватывая огромные территории, которые было бы невозможно или чрезвычайно дорого исследовать традиционными наземными методами. Спутники могут отслеживать изменения на поверхности планеты, наблюдать за погодными явлениями, оценивать состояния лесов, водоемов и почв, а также контролировать передвижения ледников и даже следить за активностью вулканов и землетрясениями. Эти данные жизненно важны для предотвращения и минимизации последствий природных катастроф, а также для долгосрочного планирования в области урбанизации и сельского хозяйства.

Спутниковые камеры — это сложные высокотехнологичные устройства, которые претерпели значительную эволюцию с момента первых запусков. Современные спутниковые камеры могут снимать в различных диапазонах электромагнитного спектра, от видимого света до инфракрасного и радиоволн. Это дает возможность получать изображения с разными уровнями детализации в зависимости от целей съемки. Например, инфракрасная съемка полезна для оценки состояния растительности или температуры поверхности океанов, в то время как радиоволны позволяют проникать сквозь облака и пыль, предоставляя изображения поверхности Земли даже в условиях плохой видимости.

Важной частью работы спутниковых камер является оптическая система, включающая линзы и зеркала, которые фокусируют свет на матрице детекторов. Как правило, она состоит из миллионов светочувствительных элементов, преобразующих свет в электрические сигналы. Эти сигналы затем обрабатываются и передаются на Землю для дальнейшего анализа.

Одной из самых сложных задач в спутниковой фотосъемке является обеспечение стабильности и точности. Спутники двигаются с большой скоростью, и малейшие колебания или вибрации могут исказить изображение. Для решения этой проблемы используются гироскопы и другие стабилизирующие устройства, а также алгоритмы цифровой стабилизации изображения.

Разрешение снимков также играет критическую роль. Чем выше разрешение, тем больше деталей можно увидеть на изображении. Современные спутники способны получать снимки с разрешением до нескольких десятков сантиметров на пиксель, что позволяет различать даже небольшие объекты на Земле. Однако для получения таких снимков требуется не только высококачественная оптика, но и сложные алгоритмы обработки данных, которые учитывают такие факторы, как атмосферные искажения и движение самого спутника.

Также важно отметить роль спутниковых камер в космических исследованиях. Они позволяют наблюдать за далекими объектами и изучать планеты, звезды и галактики. Съемка в разных диапазонах спектра дает возможность ученым анализировать состав и структуру космических тел, а также искать потенциально обитаемые экзопланеты.

Формирование изображения

Ключевую роль в процессе получения изображений в спутниковых камерах играют матрицы — массивы светочувствительных элементов (пикселей). Каждый пиксель — это фотодиод, который преобразует свет (попадающие на него фотоны) в электрические сигналы. Попадающий свет вызывает накопление электрического заряда. Накопленный заряд зависит от количества света, попавшего на пиксель.

Существуют два основных типа матриц, используемых в спутниковой съемке: `CCD — Charge-Coupled Device` и `CMOS — Complementary Metal-Oxide-Semiconductor`.

CCD-матрицы обладают высокой чувствительностью и точностью, что делает их особенно полезными в условиях низкой освещенности. В CCD-матрицах свет попадает на пиксели, которые собирают и накапливают электрический заряд. После завершения экспозиции заряд передается по цепочке пикселей на выходной регистр, где он считывается и преобразуется в аналоговый сигнал. Этот сигнал затем оцифровывается и превращается в цифровые данные, которые могут быть интерпретированы как изображение. Процесс требует точного синхронизированного управления, что делает CCD-матрицы более энергоемкими, но они обеспечивают высокую точность и низкий уровень шума.

CMOS-матрицы работают по-другому: каждый пиксель имеет свой собственный транзистор, который преобразует накопленный заряд в напряжение непосредственно в ячейке. Это напряжение затем оцифровывается и передается на выход для последующей обработки. CMOS-матрицы потребляют меньше энергии и обладают более высокой скоростью считывания, что делает их подходящими для съемки быстродвижущихся объектов. Хотя CMOS-матрицы могут уступать CCD-матрицам по чувствительности в условиях низкой освещенности, они позволяют интегрировать дополнительные функции, такие как аналого-цифровое преобразование на одном кристалле, что упрощает конструкцию и снижает стоимость камер.

Для получения цветного изображения матрица должна различать свет разных длин волн (красный, зеленый и синий). Это достигается с помощью цветных фильтров, наложенных на каждый пиксель. Самым распространенным способом является использование массива Байера, где каждый пиксель покрыт фильтром одного из трех цветов. Обычно на один красный пиксель приходится два зеленых и один синий, так как глаз человека более чувствителен к зеленому цвету. Такое соотношение пиксе-

лей используется в привычных нам камерах для получения изображения, наиболее близкого к тому, что видит человек, в научных же целях соотношение пикселей может быть другим — в зависимости от задач. Когда свет проходит через эти фильтры, каждый пиксель регистрирует интенсивность света определенного цвета, и затем специальный алгоритм, называемый демозаикацией, объединяет данные с различных пикселей для создания полноценного цветного изображения.

После того как свет преобразован в электрические сигналы, они проходят через аналого-цифровой преобразователь (АЦП), который превращает их в цифровые значения. Эти значения представляют собой уровни яркости для каждого пикселя. Затем цифровые данные подвергаются обработке, где применяются различные коррекции: баланс белого, коррекция экспозиции и шумоподавление. На этом этапе также может проводиться сжатие данных для уменьшения размера файла. После завершения всех этапов обработки данные собираются в единое цифровое изображение.

Важность выдержки в спутниковой съемке

Выдержка — это время, в течение которого матрица подвергается воздействию света. Правильный выбор выдержки имеет решающее значение для получения качественного изображения. При короткой выдержке матрица экспонируется на протяжении короткого времени, что позволяет заморозить движение объектов на снимке. Это особенно важно для спутников, движущихся с высокой скоростью относительно поверхности Земли, так как короткая выдержка помогает избежать размытости изображения. Однако слишком короткая выдержка может привести к недостаточной экспозиции в условиях низкой освещенности, делая изображение темным и шумным.

С другой стороны, длинная выдержка позволяет матрице улавливать больше света, что полезно в условиях слабого освещения. Однако при длительной выдержке возникает риск размытости изображения из-за движения спутника, что особенно критично для съемки с высоким разрешением. Чтобы компенсировать эти факторы, в спутниковых камерах применяются технологии стабилизации изображения, такие как гироскопы и сенсоры, корректирующие движение камеры в реальном времени. Также используются алгоритмы цифровой обработки, которые анализируют серию снимков и выбирают наиболее четкие кадры.

Условие

Спутник, находящийся на орбите, должен сделать фотографию объекта на поверхности Земли. Известно, что камера спутника имеет фокусное расстояние $f = 1$ м и разрешение матрицы $R = 10^6$ пикселей. Сама матрица — квадратная, размер пикселя — 1 мм. Угловой размер объекта относительно спутника составляет $\theta = 0{,}05$ рад. Камера обладает выдержкой $t = 0{,}01$ с. Спутник движется по орбите со скоростью $V_{orb} = 7{,}5$ км/с на высоте $h = 500$ км над поверхностью Земли.

1. Определить линейный размер объекта на поверхности Земли.
2. Найти линейное разрешение камеры на данной высоте.
3. Определить максимальную допустимую угловую скорость вращения спутника ω_{max}, при которой объект не сместится на изображении более, чем на 3 пикселя.

Решение

1. Линейный размер объекта на поверхности Земли:

$$L = h \cdot \theta = 500 \text{ км} \cdot 0{,}05 \text{ рад} = 25 \text{ км}.$$

2. Линейное разрешение камеры на поверхности Земли можно найти как:

$$\delta_x = \frac{h \cdot \delta_p}{f},$$

где δ_p — размер 1 px (пикселя) на матрице.
Тогда линейное разрешение на поверхности Земли:

$$\delta_x = 500 \text{ м/пиксель}.$$

3. Максимальная допустимая угловая скорость вращения спутника. Максимальное смещение объекта на изображении, допустимое при выдержке, — это 3 пикселя. На поверхности Земли это смещение составит:

$$\delta_L = 3 \cdot \delta_x = 1500 \text{ м}.$$

Максимальная допустимая угловая скорость вращения спутника:

$$\omega_{max} = \frac{\delta_L}{h \cdot t} = 0{,}003 \text{ рад/с}.$$

Ответ:

1. 25 км;
2. 500 м/пиксель;
3. 0,003 рад/с.

2.4.3. Задачи по компетенции Радиосвязь

После вывода спутника на орбиту единственным инструментом для взаимодействия с ним остается радиосвязь. Радиоволны применяются как для управления аппаратом и мониторинга его состояния, так и для реализации функционала полезной нагрузки, работающей в соответствии с общепринятыми стандартами связи, определяющими то, как сигнал формируется кодируется, а также принципы ведения радиообщения с учетом движения спутника, состояния радиоэфира, целей и задач системы и иных факторов.

С помощью данного тестирования можно познакомиться с необходимой теорией, а также начать работать с инструментом обработки сигналов `GNU Radio`.

Задача 2.4.3.1. Что такое связь и почему она радио (1 балл)

Тема: радиосвязь.

Теоретическая часть

Для передачи текстового, звукового или видеосообщения из точки А в точку В его необходимо подвергнуть ряду преобразований и отправить по линии связи. Важными компонентами такой линии являются:

- передатчик;
- канал связи;
- приемник.

Передатчик — устройство, формирующее сигнал для передачи и отправляющее сигнал по линии связи. Его задачей является преобразование сообщения в форму, пригодную для передачи, то есть в сигнал.

Канал связи — проводное или беспроводное соединение между передатчиком и приемником. Именно по нему происходит передача сообщения.

Рис. 2.4.87

Приемник — устройство, осуществляющее прием сигнала и преобразование его обратно в сообщение.

Если говорить о беспроводном соединении, то в качестве основы линий связи используются радиоволны, а также инфракрасное, оптическое или лазерное излучение.

Более подробно остановимся на радиоволнах.

Радиоволны — это электромагнитные волны (распространяющееся в пространстве возмущение/изменение состояния электромагнитного поля). Определяющими характеристиками радиосвязи являются частота колебаний волн (число полных колебаний или циклов волны, совершенных в единицу времени) и длина волны (пространство, которое проходит электромагнитная энергия за время одного периода, распространяясь со скоростью света). Радиоволны в электромагнитном спектре располагаются от крайне низких частот до инфракрасного диапазона.

С учетом классификации Международным союзом электросвязи радиоволн по диапазонам, к радиоволнам относят электромагнитные волны с частотами от 0,03 Гц до 3 ТГц, что соответствует длине волны от 10 км до 0,1 мм.

Различные диапазоны частот обладают различными физическими характеристиками, определяющие такие параметры, как помехоустойчивость, максимальное расстояние передачи, проникающая способность и многое другое. Служебные каналы связи спутников чаще всего управляются радиоволнами в УКВ (ультракоротковолновом) диапазоне частот. УКВ — традиционное в СССР название диапазона радиоволн, объединяющего метровые, дециметровые, сантиметровые и миллиметровые волны (или диапазоны очень высоких частот — ОВЧ, ультравысоких частот — УВЧ, сверхвысоких частот — СВЧ и крайне высоких частот — КВЧ). То есть это все радиоволны, длина которых менее 10 м.

Любой радиосигнал — это электромагнитная волна, которая несет в себе заложенную определенным образом информацию.

Скорость, с которой распространяются радиоволны в вакууме, равна скорости света (в атмосфере эта скорость немногим меньше).

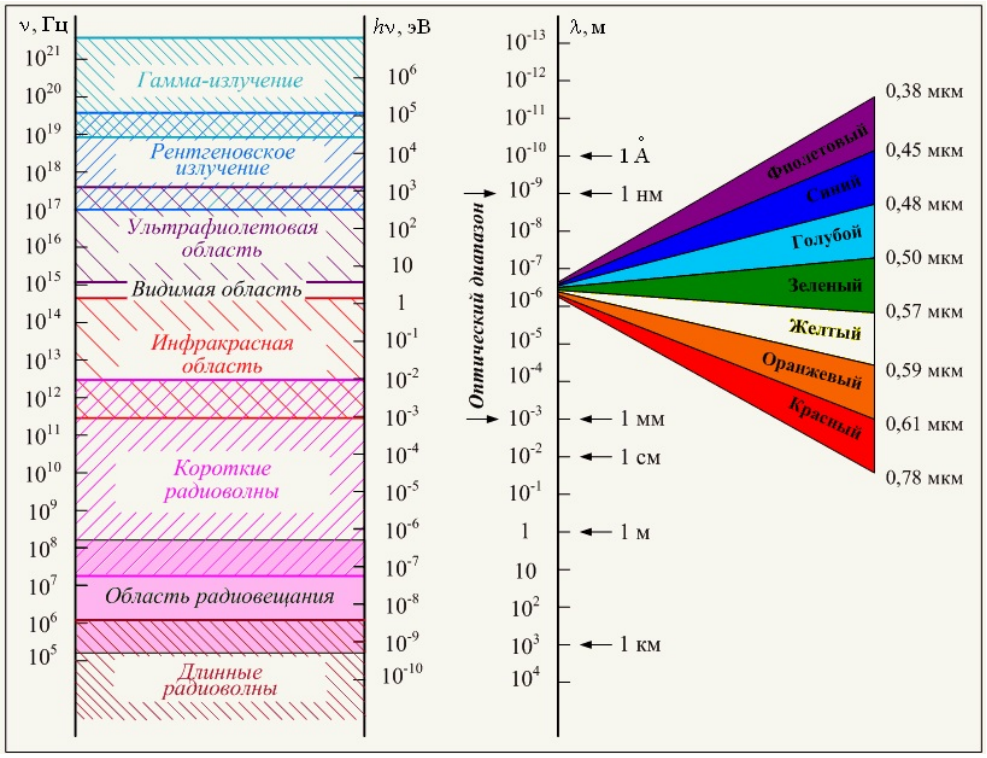


Рис. 2.4.88

Радиосигнал имеет несколько характеристик, определяющих его свойства:

- длина волны — это расстояние между двумя повторяющимися пиками;
- частота — количество повторений в секунду ($1\,\text{Гц} = 1/\text{с}$);
- амплитуда — максимальное значение.

У электромагнитных волн также имеется две составляющих — электрическая и магнитная. Векторы напряженности электрического поля и магнитной индукции перпендикулярны друг другу и перпендикулярны направлению распространения волны.

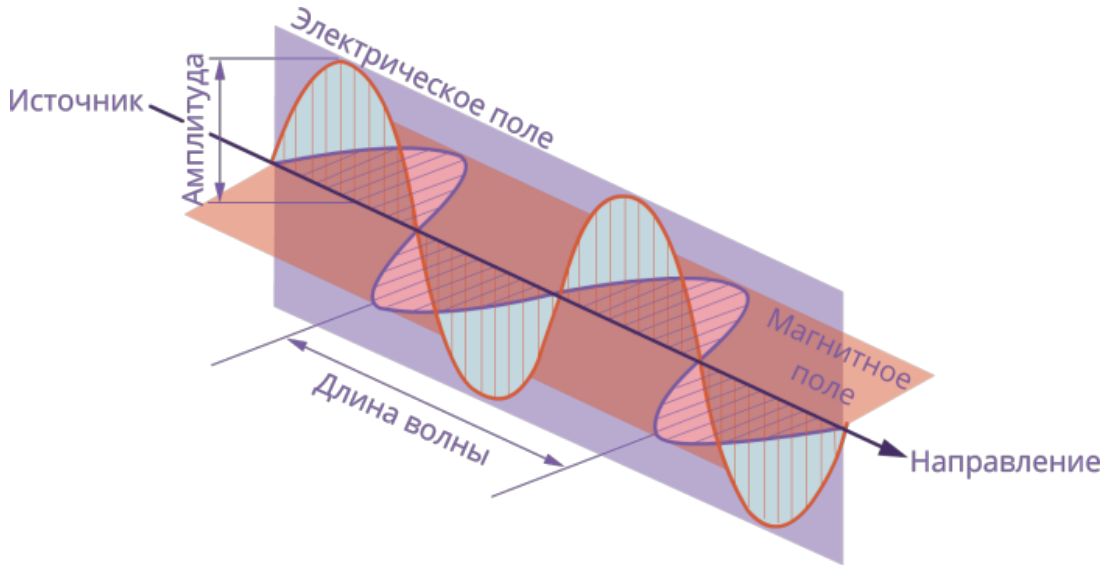
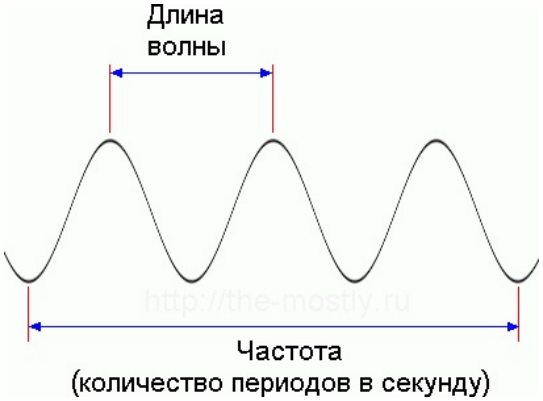
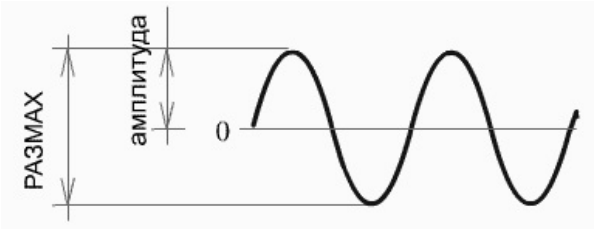


Рис. 2.4.89





Длина волны определяет ее способность огибать объекты в пространстве: чем больше длина волны, тем сильнее волна будет огибать препятствующие ее прямому распространению объекты.

Верхняя часть атмосферы, лежащая на границе с космосом, называется **ионосферой** Земли. Из-за облучения солнечной радиацией в этом месте большое содержание свободных электронов и ионов, что способствует образованию полярных сияний, молний, а также отражению низкочастотных радиосигналов и преломлению всех остальных. Это делает низкие частоты непригодными для связи с космосом с поверхности Земли.

Таблица 2.4.5

Диапазон	**Длина волны**	**Частота**	**Взаимодействие с ионосферой**
Сверхдлинные волны (СДВ)	от 10 до 100 км	3–30 кГц	Поглощаются ионосферой, распространяются огибающей поверхностной волной вдоль Земли
Длинные волны (ДВ)	от 1 до 10 км	30–300 кГц	

Диапазон	Длина волны	Частота	Взаимодействие с ионосферой
Средние волны (СВ)	от 100 до 1000 м	от 300 кГц до 3 МГц	Днем поглощаются ионосферой, ночью отражаются
Короткие волны (КВ)	от 10 до 100 м	от 3 до 30 МГц	Отражаются ионосферой
Ультракороткие волны (УКВ)	от 0,1 мм до 10 м	от 30 МГц до 3 ТГц	Проходят сквозь ионосферу

По способу распространения выделяют следующие виды волн:

- прямые — волны, распространяющиеся прямолинейно, аналогично свету, исходящему от источника света, не способны огибать поверхность земного шара и проходят ионосферу насквозь;
- поверхностные — волны, распространяющиеся вдоль поверхности Земли и огибающие ее выпуклость, не проходят через ионный слой;
- пространственные — волны, распространяющиеся по поверхности Земли за счет отражения от ионного слоя и поверхности Земли, за счет многочисленного отражения со временем затухают.

На рис. 2.4.90 цифрой 1 обозначена прямая волна, 2 и 3 — поверхностные волны, 4 — пространственная волна.

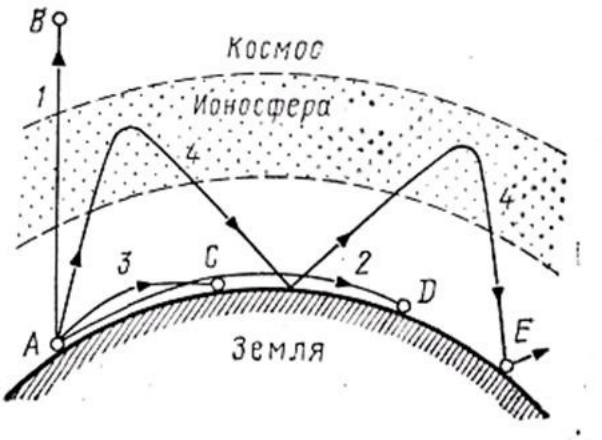


Рис. 2.4.90

Таблица 2.4.6

Вид волны	Длина волны	Диапазон частот	Распространение	Применение
Длинные	От 1 до 10 км	300–30 кГц	Поверхностные	Радиосвязь, радиовещание
Средние	От 100 до 1000 м	3–0,3 МГц	Поверхностные	Радиовещание, радиосвязь

Вид волны	**Длина волны**	**Диапазон частот**	**Распространение**	**Применение**
Короткие	От 10 до 100 м	30–3 МГц	Пространственные	Радиосвязь, радиовещание, рации
Метровые	От 1 до 10 м	300–30 МГц	Прямые	Радиосвязь, телевидение
Дециметровые	От 1 до 10 дм	3–0,3 ГГц	Прямые	Радиолокация, радиорелейная связь, радионавигация, телевидение, мобильные телефоны, рации, микроволновые печи, спутниковая навигация
Сантиметровые	От 1 до 10 см	30–3 ГГц	Прямые	Радиолокация, радиорелейная связь, интернет, спутниковое телевещание, спутниковая связь, беспроводные компьютерные сети
Миллиметровые	От 1 до 10 мм	300–30 ГГц	Прямые	Радиоастрономия, высокоскоростная радиорелейная связь, радиолокация (метеорологическая, управление вооружением), медицина, спутниковая радиосвязь
Субмиллиметровые	От 0,1 до 1 мм	3000–300 ГГц	Прямые	Освоение для специальных применений

Условие

Космокот Кас поехал в далекую-далекую глубинку, чтобы навестить свою бабушку. Рассказывая ей о своих подвигах на олимпиаде, Кас обратил внимание на высокую радиомачту прямо во дворе дома. Оказалось, что она принадлежала дедушке Каса, и в периоды отсутствия почтового бота эта станция остается единственным способом связаться с миром. Но прямо перед приездом Каса старенький бабушкин дневник с инструкцией от дедушки рассыпался на страницы! Помогите по обрывкам записей из дневника восстановить контакты и используемые ими диапазоны для связи.

1. **Гена**, *27,135 МГц, развозит хлеб на своем синем грузовике.*
2. **Маня**, *446 МГц, почтовый бот, не забыть передать талончик.*
3. **Жучка**, **** *кГц, подружка из Котяткино, почему-то принимает и пере-*

дает только ночью...

4. **Гришка**, *1** МГц, летает тут на своем «авиа», кур распугивает.*
5. **Старлинк**, *10,7 ГГц, Кас мне все в письмах рассказывал, но только не говорит там никто.*

Ответ:

1. КВ;
2. дециметровые;
3. СВ;
4. метровые;
5. сантиметровые.

Задача 2.4.3.2. Изучаем волны со стилем и вкусом (1 балл)

Тема: радиосвязь.

Теоретическая часть

Любой сигнал имеет свою форму и свои характеристики. Обычно мы говорим о сигнале как о некоторой функции синуса (или косинуса) от времени, в частности, гармонический сигнал описывается уравнением:

$$s(t) = A \cdot \cos(2\pi f t + \varphi).$$

Однако реальные сигналы очень редко бывают гармоническими и представляют собой сложную форму, сочетающую в себе сразу несколько различных тонов. А поэтому гораздо чаще в обработке сигналов используют другое представление сигнала, называемое **спектром**.

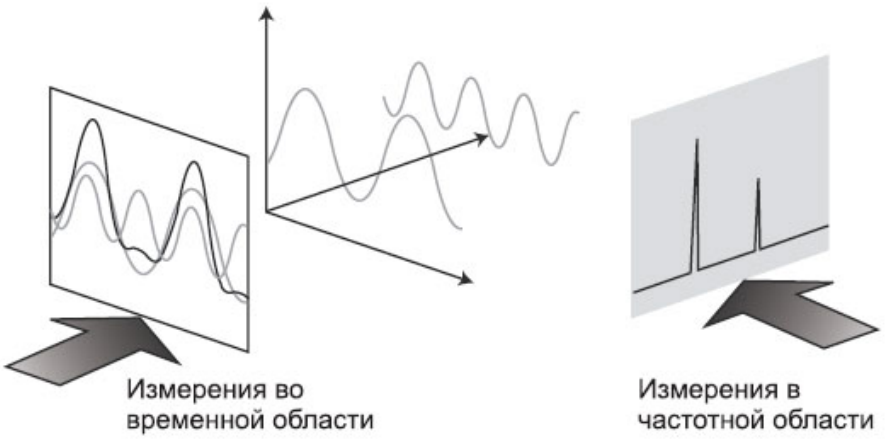


Рис. 2.4.91

Спектр сигнала — это его математическое представление, функция, зависящая не от времени, а от **частоты**, и представляющая собой сумму гармоник сигнала.

Гармониками сигнала называют выражения типа:

$$\sin(k \cdot \omega \cdot t) \text{ и } \cos(k \cdot \omega \cdot t),$$

где k — номер гармоники, ω — частота, t — момент времени.

При суммировании достаточно большого количества таких составляющих можно математически описать принимаемый сигнал. Такое математическое представление сигнала позволяет осуществлять его дальнейшую обработку с помощью цифровой аппаратуры. На рис. 2.4.92 представлено спектральное представление в диапазоне 433 МГц.

Здесь по оси абсцисс идут частоты, а по оси ординат — интенсивность сигнала в данный момент.

Разложение сигнала в гармоники реализуют с помощью так называемого преобразования Фурье, суть которого в данном курсе подробно не излагается, однако его упоминание еще не раз встретится при работе с сигналами.

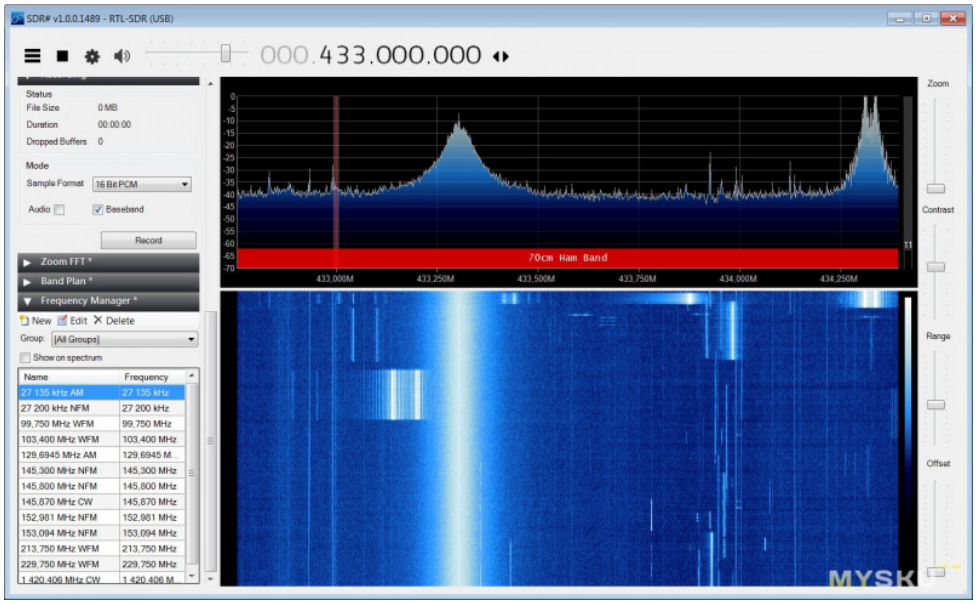


Рис. 2.4.92

Чтобы лучше понять, как работает спектр и увидеть его для разных примеров, рекомендуем обратить внимание на это приложение: https://academo.org/demos/spectrum-analyzer/.

Условие

Космокот Кас столкнулся с проблемами с микрофоном: его друзья жалуются, что у него противный посторонний шум на фоне. Кас решил для начала провести диагностику и определить, на какой частоте вмешивается посторонний шум.

Определите частоту, на которой слышен шум на пробной записи: https://drive.google.com/file/d/1yWLi9-bz8qyzvIGwfXAEWFpbxXPaXorg/view?usp=sharing. Ответ дайте в герцах.

Ответ: достаточно открыть любой спектроанализатор, чтобы увидеть частоту шума — 6 000 Гц.

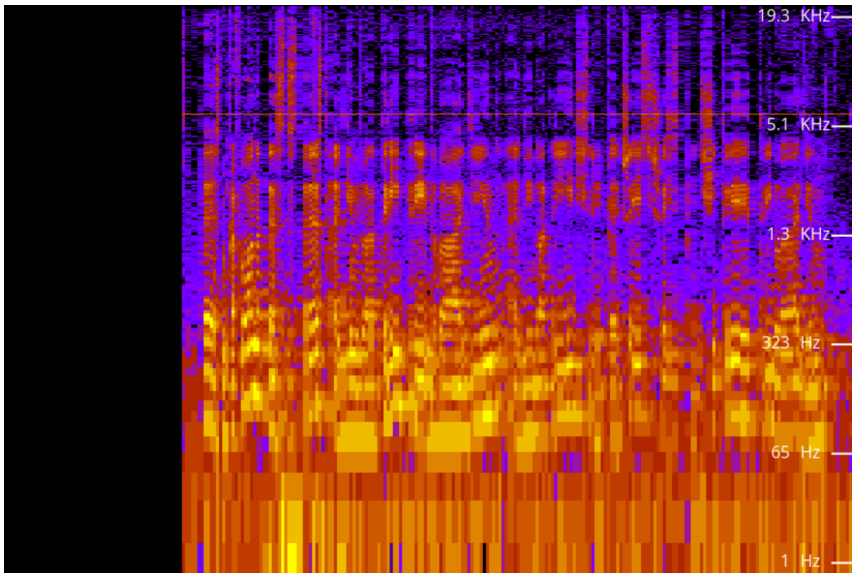


Рис. 2.4.93

Задача 2.4.3.3. Наполнение волн смыслом (2 балла)

Темы: радиосвязь, спектр.

Теоретическая часть

Каждый передаваемый сигнал имеет свою частоту. Обычно те сигналы, которые необходимо передать, являются низкочастотными (до нескольких МГц). Однако в своем первоначальном виде передавать их не всегда удобно, поэтому применяют способ передачи с помощью опорной волны, которая по частоте гораздо выше исходного сигнала. Она называется несущим колебанием (ее частота называется несущей частотой), и у него все те же характеристики, что и у любой волны: длина волны, частота и амплитуда сигнала.

Однако само по себе несущее колебание не несет никакой информации. Информацию несет в себе сигнал. И чтобы данную информацию передать из одной точки в другую, необходимо «наложить» форму сигнала на несущую частоту. Такое «наложение» осуществляется путем изменения одной или нескольких характеристик сигнала: амплитуды, частоты или фазы. Этот процесс называется модуляцией.

Модуляция (лат. *modulatio* — размерность) — процесс изменения одного или нескольких параметров высокочастотного несущего колебания по закону низкочастотного информационного сигнала.

Эта техника дает несколько важных преимуществ, частности, позволяет:

- сформировать радиосигнал, который будет обладать свойствами, соответствующими свойствам несущей частоты;
- использовать антенны малого размера, поскольку размер антенны должен быть пропорционален длине волны;
- избежать интерференции с другими радиосигналами

Существует несколько видов модуляции: изменение амплитуды несущего колебания для передачи сигнала называется амплитудной модуляцией (АМ), изменение

частоты — частотной (FM), а изменение фазы — фазовой (PM).

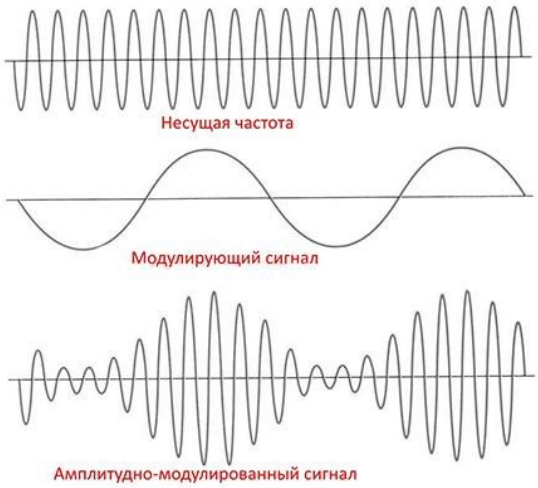


Рис. 2.4.94

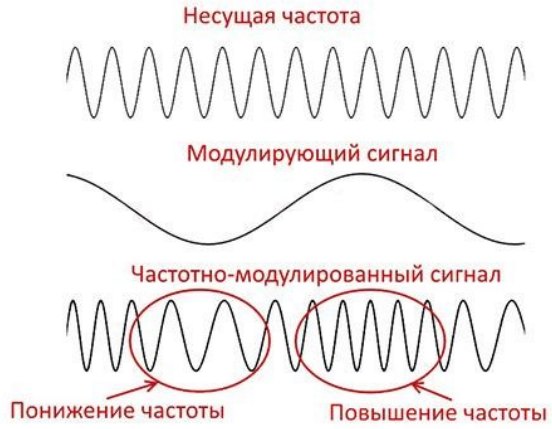


Рис. 2.4.95

Сам сигнал передается не строго на частоте несущего колебания, но и захватывает некоторые соседние частоты в некотором диапазоне. Диапазон частот, в котором передается сигнал, называется **полосой частот**.

Спектр АМ-сигнала состоит из трех компонент: центральная — несущая — и две боковые — модулирующие. При изменении несущей частоты происходит смещение всего АМ-сигнала по частоте.

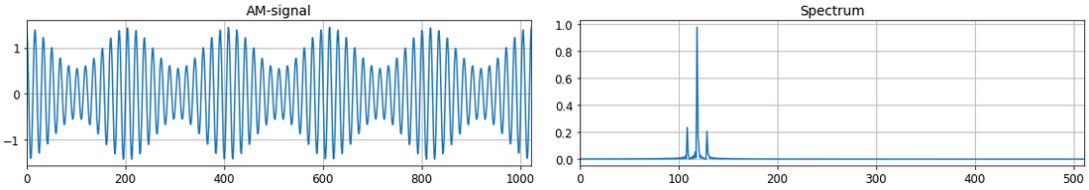


Рис. 2.4.96

Для ЧМ-сигналов появляется новый термин — девиация. Это отклонение частоты от несущей в процессе модуляции. Иначе говоря, чем больше девиация, тем больше ширина спектра ЧМ-сигнала.

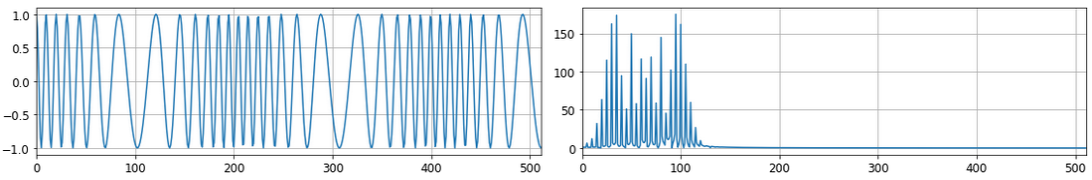


Рис. 2.4.97

Модуляция цифровых сигналов называется манипуляцией. **Амплитудная манипуляция** (*Amplitude-shift keying*, ASK) — преобразование сигнала, при котором скачкообразно меняется амплитуда несущего колебания.

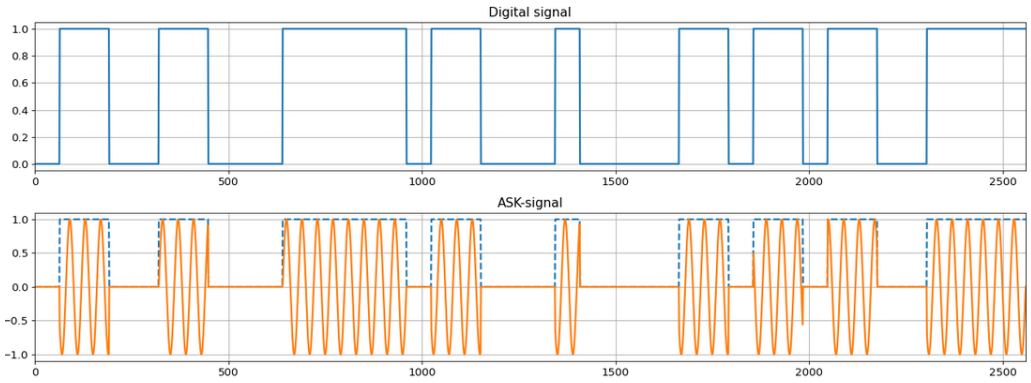


Рис. 2.4.98

Частотная манипуляция (*Frequency-shift keying, FSK*) — преобразование сигнала, при котором скачкообразно меняется частота несущего сигнала в зависимости от значения цифрового сообщения.

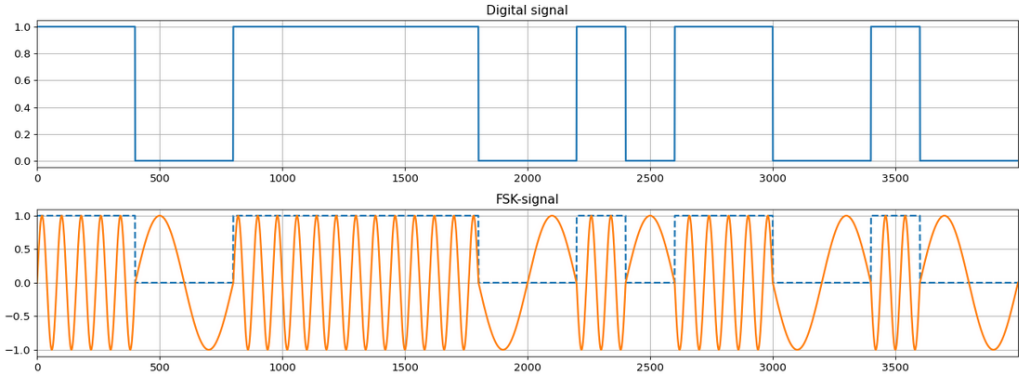


Рис. 2.4.99

Фазовая манипуляция (*Phase-shift keying, PSK*) — процесс преобразования сигнала, при котором скачкообразно изменяется фаза несущего колебания. Существует большой класс сигналов с фазовой манипуляцией (BPSK, QPSK, 8-PSK и т. д.).

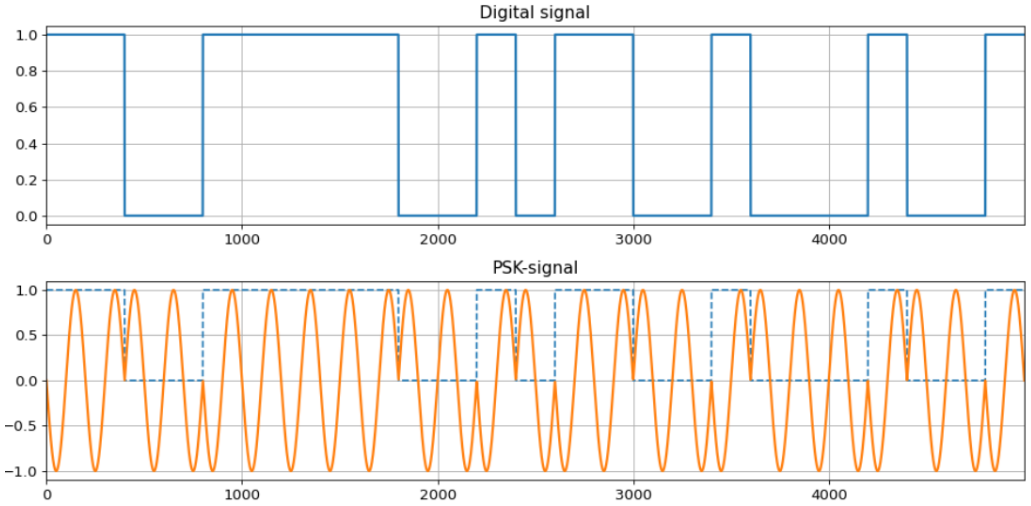


Рис. 2.4.100

Для каждого конкретного вида модуляции спектр будет иметь свою форму.

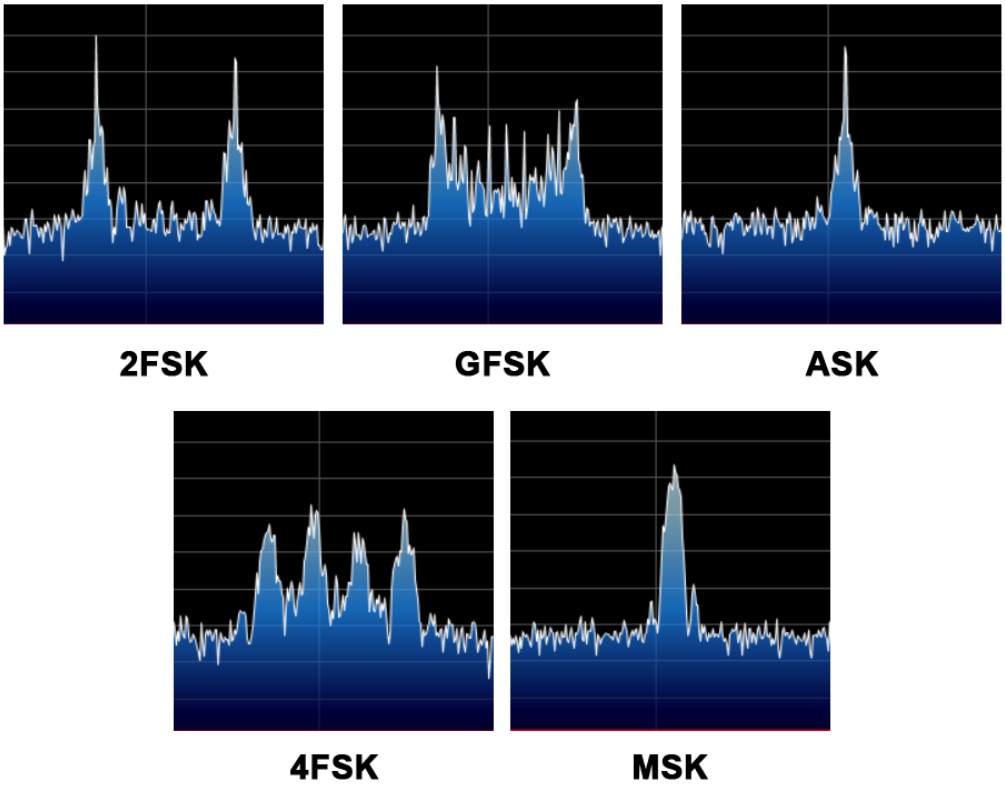
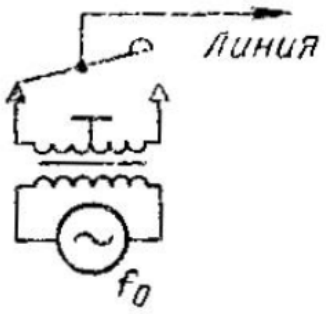
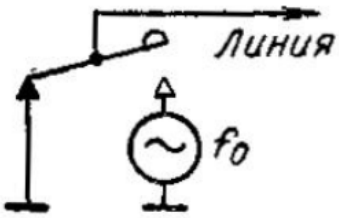
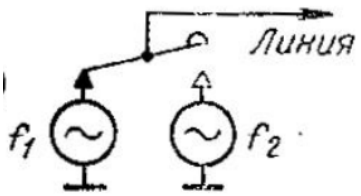


Рис. 2.4.101

Условие

Космокот Кас посетил экскурсию в своем городском музее. Темой экскурсии была «История связи». Благодаря бабушке Кас уже хорошо ориентировался в радиосвязи, но тем не менее с интересом рассматривал стоящий в музее телеграф. Рядом на столе лежал рисунок.

А	Б	В
		

Помогите Касу определить, какая схема посылает по телеграфному проводу АМ-сигнал, ЧМ-сигнал и ФМ-сигнал?

Ответ: А — ФМ, Б — АМ, В — ЧМ.

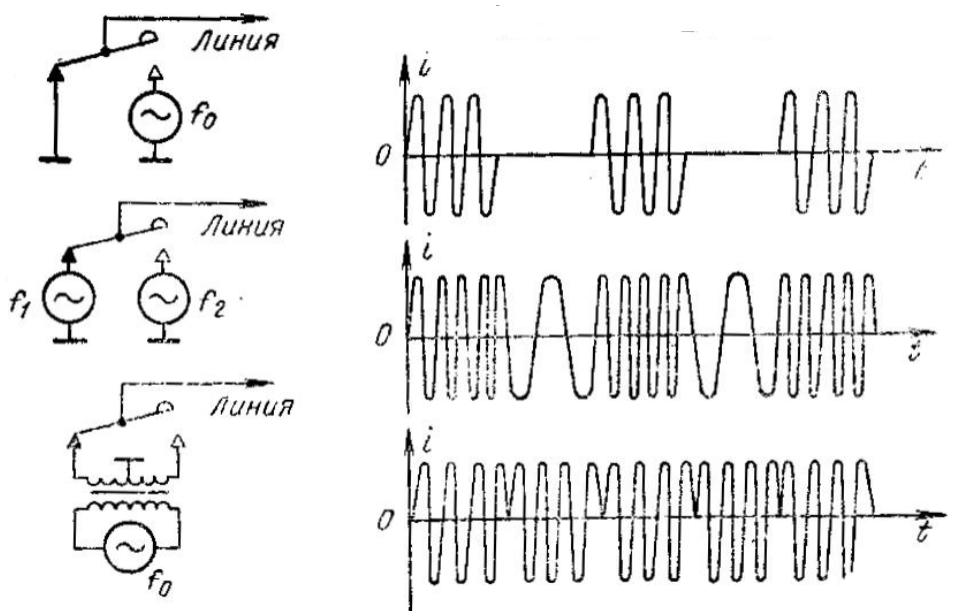


Рис. 2.4.102

Задача 2.4.3.4. Как поместить сигнал в компьютер (2 балла)

Тема: радиосвязь.

Теоретическая часть

Радиосигнал изначально имеет аналоговую природу, но чтобы было возможным его обработать с помощью компьютера, необходимо его оцифровать. Сделать это можно, измеряя с постоянной периодичностью получаемый уровень сигнала. Частота, с которой происходят измерения, называется частотой дискретизации, сами измерения — сэмплами, а устройство производящие замеры — АЦП (аналого-цифровой преобразователь).

Тип радиоприемников, которые оцифровывают радиосигнал и передают его на компьютер, называется SDR (*Software-defined radio*). Их отличие от аналоговых радиоприемников в том, что они позволяют принимать не только центральную частоту, но и часть спектра возле нее. При этом вся обработка сигнала происходит программно, поэтому возможна только в связке с компьютером. Долгое время у человечества не было достаточных вычислительных возможностей, и только в последнее время эта технология стала по-настоящему массовой.

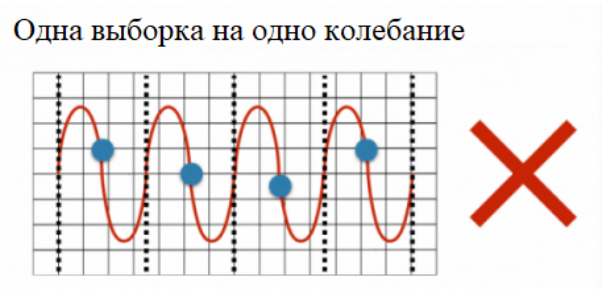


Рис. 2.4.103

Рис. 2.4.104

Например, частота дискретизации микрофона, подключенного к компьютеру или смартфону, чаще всего > 40 КГц, потому что оцифрованный сигнал должен содержать в себе весь акустический диапазон (от 20 Гц до 20 КГц).

Теорема Котельникова задает достаточно строгие ограничения, которые могут доставить проблемы при работе с высокочастотными диапазонами. Однако чаще всего в приемниках не оцифровывают волну напрямую. Вместо этого задается центральная частота, после чего ее снижают на некоторую величину за счет смесителя. Так, центральная частота переносится на частоту заданного отступа, что и является промежуточной частотой для работы АЦП. Такое снижение позволяет принимать частоты сильно больше частоты дискретизации, но, как и в случае с микрофоном, теорема Котельникова ограничивает размер куска спектра, который можно оцифровать.

Помимо прочего, частота дискретизации ограничивает максимальную скорость передачи данных. Это легко увидеть на примере: допустим, передатчик модулирует сигнал на частоте 433 МГц, а приемник принимает этот сигнал и оцифровывает с частотой дискретизации 2 МГц. В этом случае передатчик может менять параметры волны 866 млн раз в секунду (согласно критерию Найквиста), а приемник может зафиксировать лишь 2 млн изменений в секунду. Получается, в данном случае узким местом в передаче данных будет именно частота дискретизации приемника.

Также у оцифровки есть еще один параметр: глубина — это количество информации, используемое под одно измерение. Чем больше информации мы используем, тем точнее будет зафиксировано измерение. Этот параметр определяется используемым АЦП. Чаще всего в SDR-приемниках используется 16 бит под одно измерение, но возможны и другие варианты. Повышение точности полезно, когда нужно различать очень малые изменения в сигнале, например, в случае приема слабого источника.

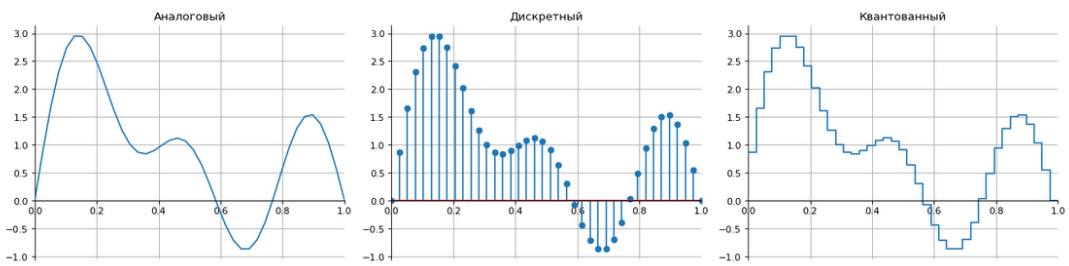


Рис. 2.4.105

Таким образом, сигнал можно описать длинной последовательностью чисел. Именно так хранят звук wav-файлы и с таким видом данных работают приложения, например, для звонков.

Но радиоприемникам этого мало, и они эту последовательность чисел дополнительно с помощью несложных математических преобразований переводят в IQ-сигнал.

Более подробно об IQ-сигнале можно прочитать тут: https://disk.yandex.ru/i/B8-yLta-YE6gSg.

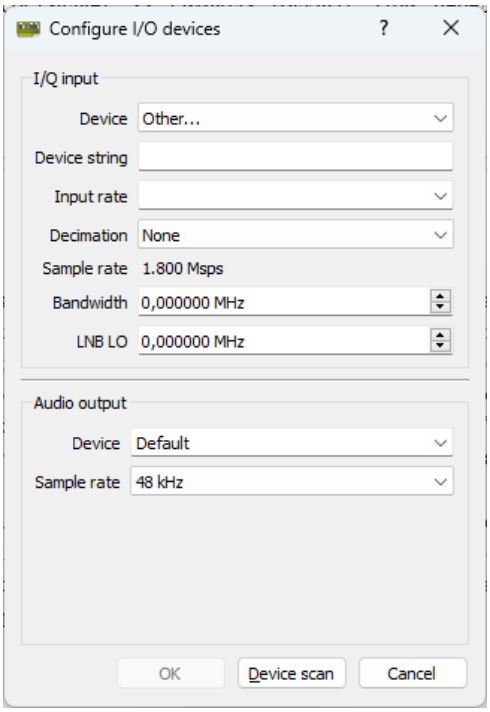


Рис. 2.4.106

Для работы с IQ-сигналом существует множество программ, таких как SDR Sharp, SDR++, GNU Radio и входящий в его состав GQRX. На примере GQRX рассмотрим простейшую работу с IQ-сигналом, а более сложные преобразования будем совершать с GNU Radio. Рекомендуется использовать дистрибутив GNU Radio под названием Radioconda (https://github.com/ryanvolz/radioconda/releases/tag/2023.07.26). Сам GQRX, после установки Radioconda, также как и GNU Radio, будет доступен в меню **Пуск** или через поиск. Начнем работу с запуска GQRX. После запуска, помимо командной строки, можно увидеть окно, изображенное на рис. 2.4.106.

Это окно настройки источника IQ-сигнала. Нужно выбрать **Device** — **Complex Sampled (IQ) File**. После этого необходимо будет настроить параметры в строке **Device String**. Самый важный из них **file**: после равно необходимо указать полный путь до файла IQ-сигнала, заменив в нем все обратные слеши (\ ) на прямые (/). Вот пример правильного параметра:

`file=C:/Users/user/Downloads/CW_IQ_samprate-2016k.iq`.

Для того чтобы скопировать путь до файла, выделите файл в проводнике, зажмите SHIFT и нажмите правой кнопкой мыши по нему, в меню выберите **Копировать как путь**. Не забудьте заменить слеши и убрать кавычки, которые копируются вместе с путем.

Примечание. Нежелательно, чтобы путь содержал кавычки, пробелы, русские символы. В этом случае программа может не найти нужный файл. Если у вас такой

случай, переместите файл в корень диска, то есть чтобы путь до него был, например, `C:/file.iq`.

Далее обязательно необходимо указать `rate`, который задает `sample rate` записи, он обычно указан в том же месте, где загружается файл, а чаще всего и в названии самого файла. Указывать удобнее всего в научном формате, то есть, например, число 2 016 000 можно указать как 2,016e6, что значит $2{,}016 \cdot 10^6$. Остальные параметры необязательные, но на всякий случай тоже их опишем.

1. `freq` — чисто косметический параметр, отображаемая в программе частота. Может быть любой.
2. `repeat` — повторять ли файл по кругу? `true` — да, `false` — нет.
3. `throttle` — ограничивать ли скорость воспроизведения файла реальной скоростью. За исключением очень редких случаев должен стоять true, иначе запись может воспроизвестись ускоренно.

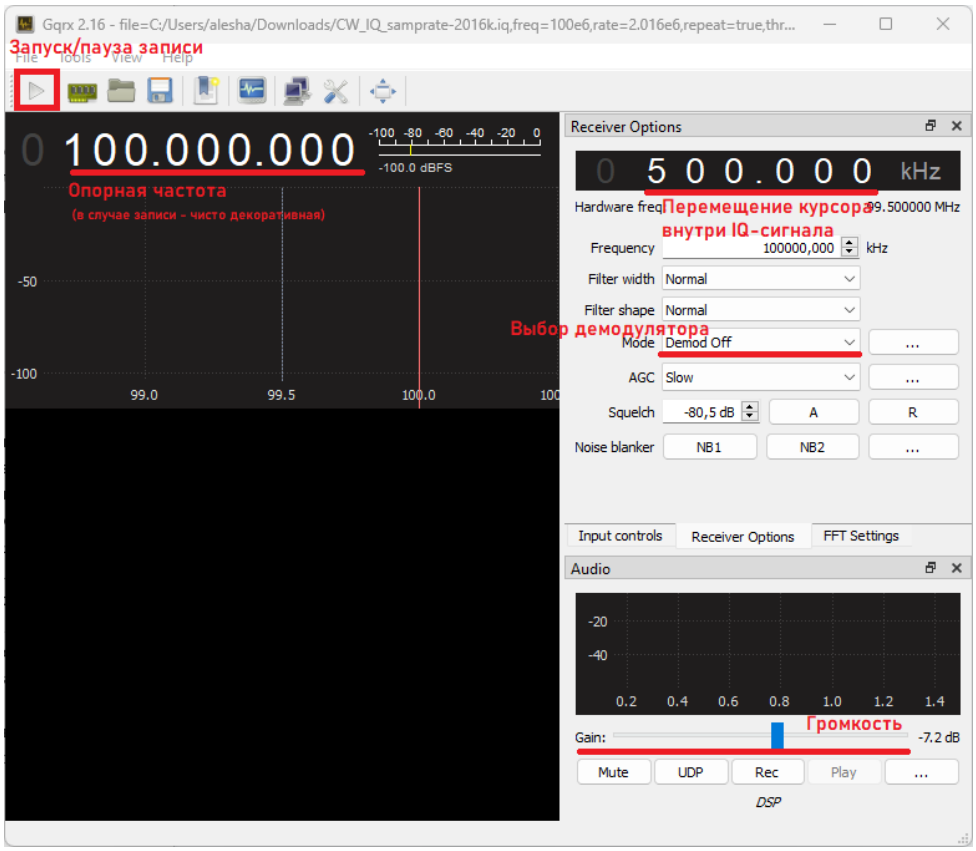


Рис. 2.4.107

После завершения настройки `Device String` можно нажимать `OK`, остальные параметры настраивать необязательно. Если появилась ошибка, перепроверьте правильность пути в параметре `file`. Если же все хорошо, откроется новое окно (рис. 2.4.107). Рассмотрим его основные органы управления.

Также могут пригодиться настройки в разделе `FFT Settings`, они влияют на внешний вид сигнала в программе.

Условие

Космокот Кас как-то раз разбирал свои вещи и нашел неизвестный и очень старый прибор. Оказалось, это был дедушкин радиопередатчик, в который можно было вставить перфокарту. Как раз рядом с передатчиком лежала одна из таких. Касу так и не удалось выяснить, что было записано на перфокарте. Как бы пристально он в нее ни всматривался, понимание не приходило. Поэтому он решил вставить перфокарту в передатчик и записать сигнал на SDR приемник, который удачно оказался под рукой.

Однако, записав `IQ`-сигнал, Кас понятия не имел, что с ним дальше делать. Помогите Касу получить послание. Напишите дословное послание в ответ строчными буквами.

*`IQ` запись сделана с `sample rate` = 512 000 сэмплов/с, ссылка на файл: https://drive.google.com/file/d/1Bt08W-6zNS0o8aTTd14uXMlVOky_uGos/view?usp=sharing.

Решение

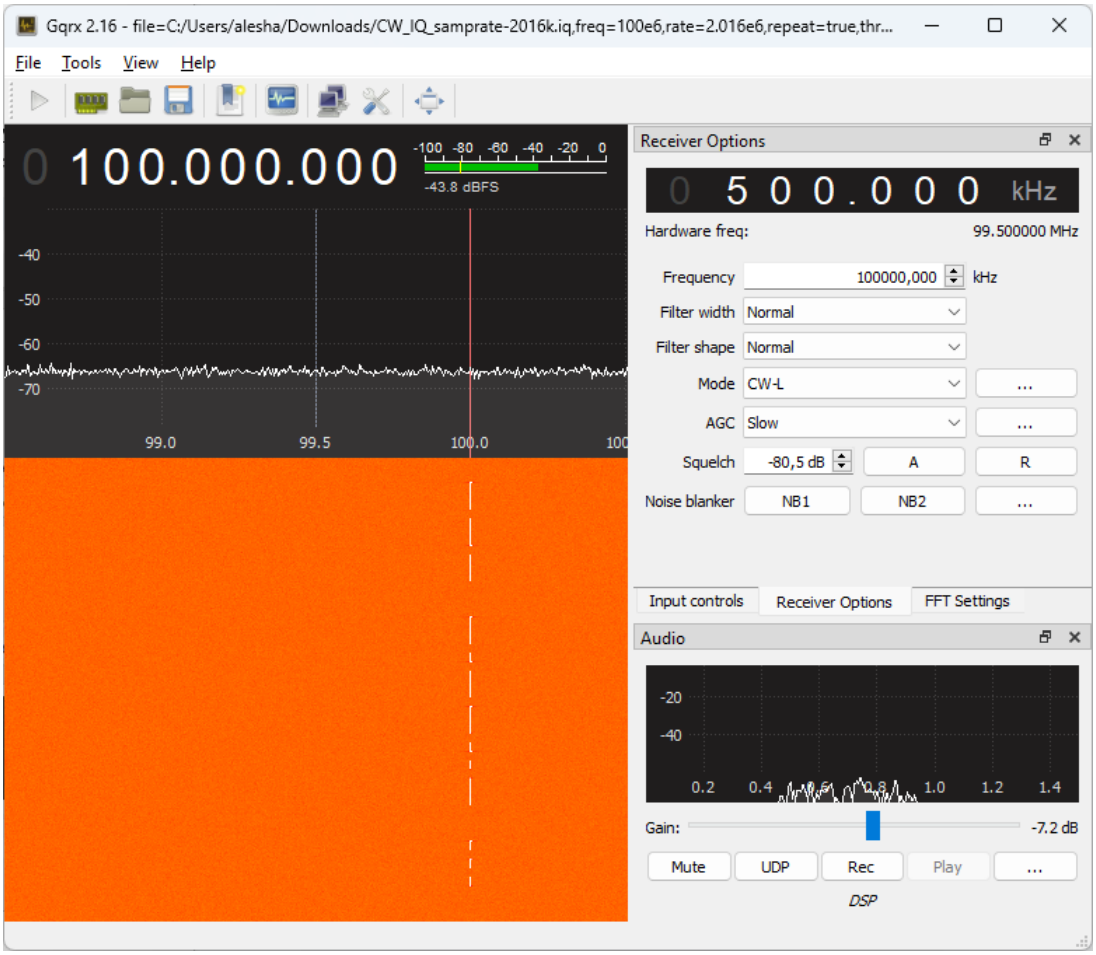


Рис. 2.4.108

Самое простое решение — посмотреть на спектр и увидеть на частоте 500 кГц идущий телеграфный сигнал, далее его записать и декодировать с помощью азбуки Морзе.

Ответ: «Слава труженикам космокотовой науки!».

Задача 2.4.3.5. Автоматизируем, оптимизируем (3 балла)

Темы: радиосвязь, демодуляция.

Теоретическая часть

Продолжим работу с сигналами уже с более мощным инструментом — `GNU Radio`. После запуска `GNU Radio` автоматически должен создаться новый проект, но если этого не произошло, то создайте его сами. Для этого необходимо в меню выбрать **File** →**New** →**QT GUI**.

Сразу же можно увидеть два блока, созданных по умолчанию: **Options** и **Variable** (рис. 2.4.109).

Блок **Variable** содержит переменную с названием **samp_rate**. Эта переменная задает частоту дискретизации входного сигнала. Когда `GNU Radio` начинает работать с сигналом, то первое, что делает данное ПО — преобразует сигнал с приемника в приемлемый оцифрованный вид. Глобальная переменная **samp_rate** задает частоту дискретизации оцифрованного сигнала.

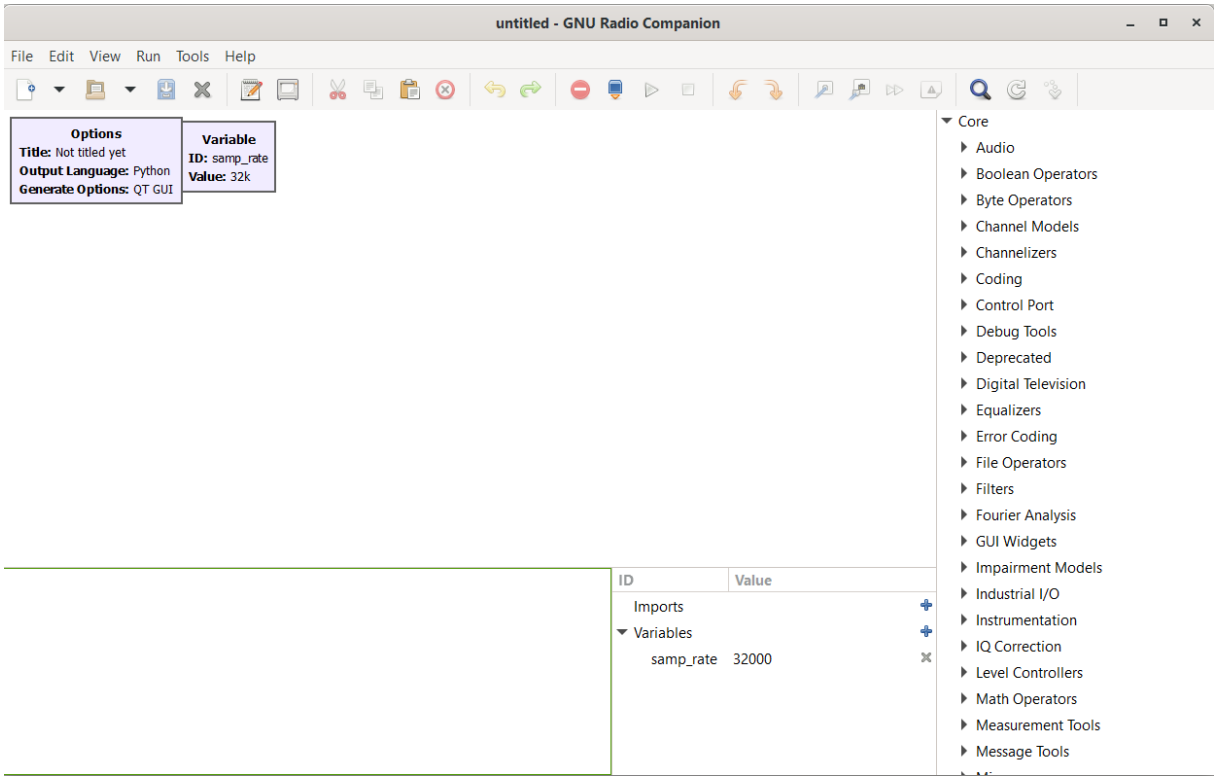


Рис. 2.4.109. Создание нового проекта в `GNU Radio`

Поставим значение **samp_rate** для рабочей частоты 433 МГц, равное 1 000 000. Чтобы задать частоту выборок кликните на блок **Variable**, после чего откроется новое окно с настройками блока, введите значение 1 000 000 или $1e6$, и затем нажмите **Apply**.

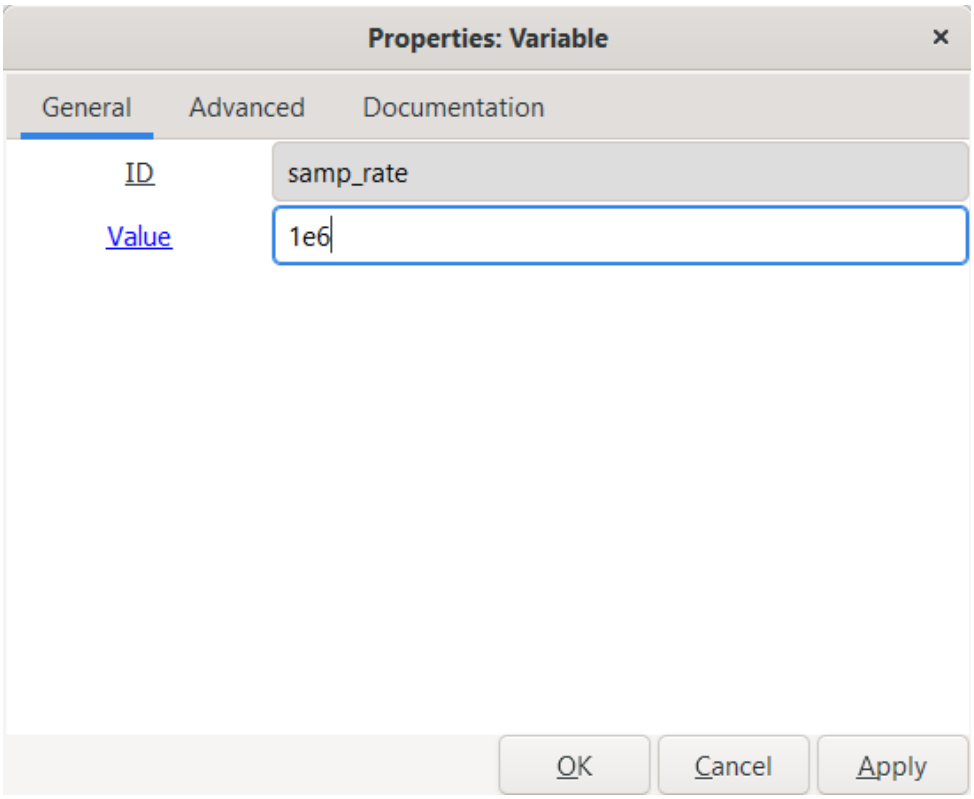


Рис. 2.4.110

Чтобы начать обработку сигнала, необходимо для начала выбрать источник, откуда этот сигнал будет приходить. Блоки источников сигнала имеют в своем названии слово **Source**. Если ввести в поиске слово **Source**, можно увидеть, какие блоки можно использовать.

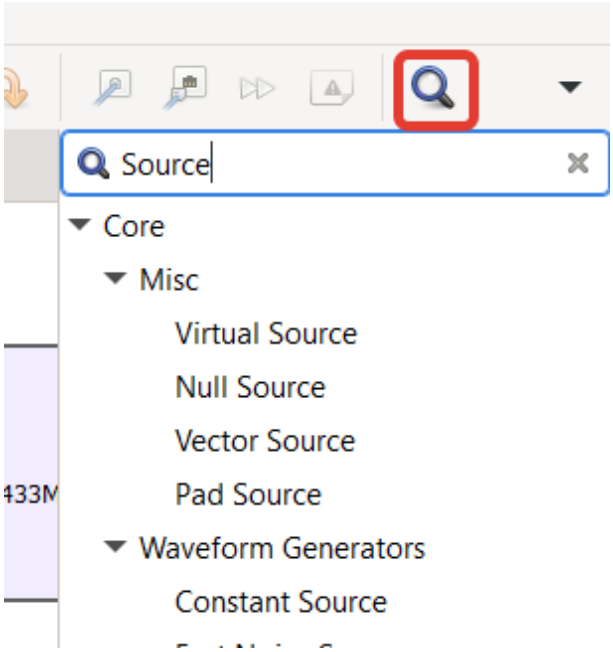


Рис. 2.4.111

В качестве источника может использоваться файл с записью (**File Source**), выход с приемника (**osmocom Source**, **RTL-SDR Source**) или генерируемый самой программой простой сигнал (**Signal Source**).

Примечание. Более подробно о блоках `GNU Radio` можно узнать здесь: http://microsin.net/adminstuff/others/gnuradio-short-reference.html.

Теперь выберем выходной блок. Выходные блоки имеют в своем названии слово **Sink**. Из перечня выберем **QT GUI Sink**. В настройках блока можно установить параметр `Central Frequency` 433e6 и выбрать **yes** напротив **Show RF Freq**, чтобы на оси частот центральной частотой видеть не 0, а 433 МГц.

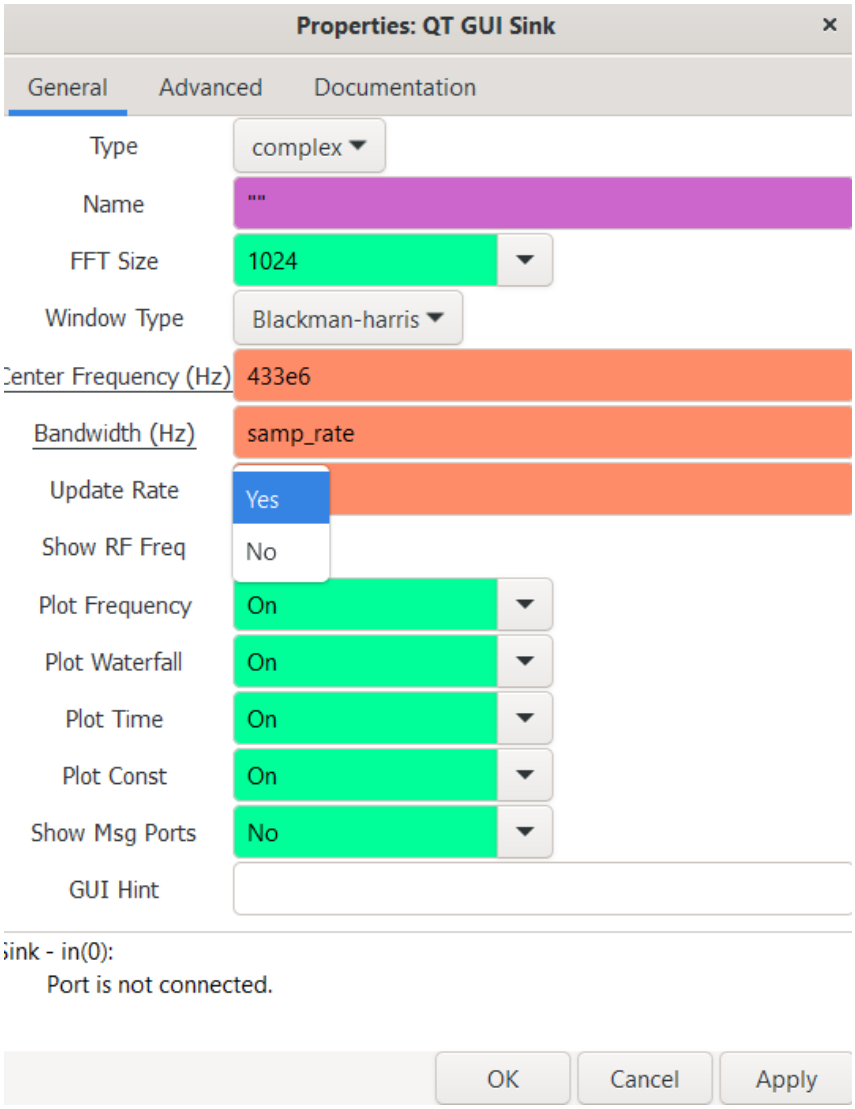


Рис. 2.4.112

Соединив два данных блока, получим самый простой граф. Все дальнейшие графы будут собираться в длинный «конвейер». Чтобы запустить работу графа, нажмите на панели инструментов значок треугольника.

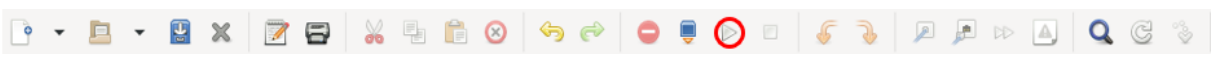


Рис. 2.4.113

Через какое-то время появится всплывающее окно со спектром сигнала. Чтобы запомнить пик сигнала, можно в открывшемся окне поставить галочку на опции **Max Hold**.

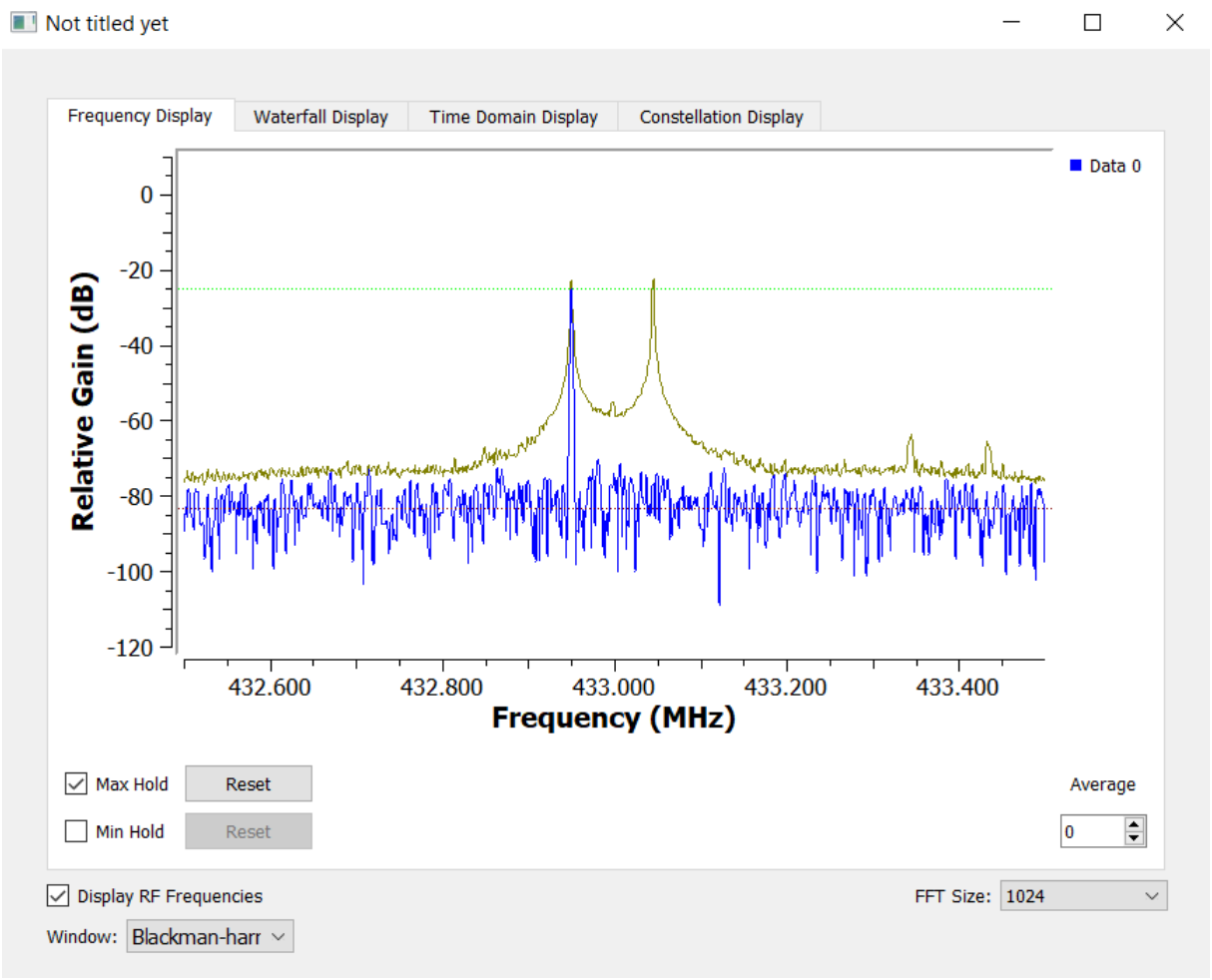


Рис. 2.4.114

Полученный сигнал может быть необходимо демодулировать. Сделать это можно, зная модуляцию исходного сигнала, с помощью соответствующих блоков. Полный список доступных блоков можно увидеть с помощью поиска **Demod**.

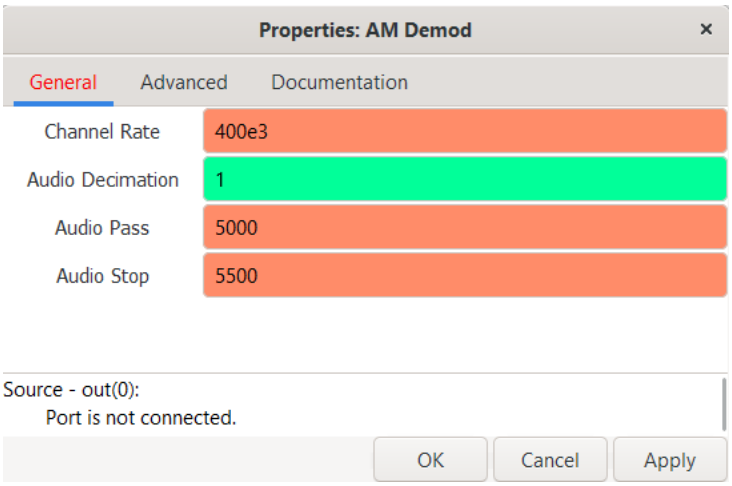


Рис. 2.4.115

Условие

Космокот Кас вместе с другом решили купить рации и захотели их протестировать. Для этого они согласовали частоту сигнала и время сеанса связи. Чтобы проверить, что они слышат именно друг друга, собеседник Каса должен во время сеанса передать известный только ему код, и потом они его вместе проверят. К сожалению, Кас накануне много играл с рацией, и в самый ответственный момент рация разрядилась. Однако под рукой у него оказался SDR-приемник, и он успел запустить запись IQ-сигнала. Помогите Касу получить код, который передал его друг. IQ-сигнал https://disk.yandex.ru/d/zhwq8F-6Stv53Q.

`sample rate`: 336 000 сэмплов/с.

Решение

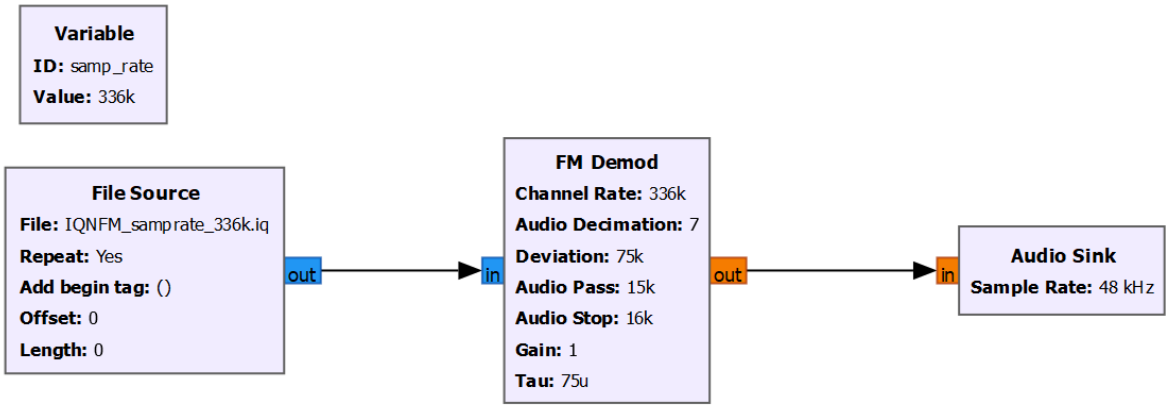


Рис. 2.4.116

Ответ: 27015.

Задача 2.4.3.6. Теряем пакеты (4 балла)

Темы: радиосвязь, кодирование.

Теоретическая часть

Как уже было описано, для передачи информации с помощью радиоволн используют модуляцию сигнала. С помощью модуляции преобразуются характеристики волны в соответствии с определенными схемами преобразования.

Изначальная информация представляет собой массив битов (нулей и единиц), представляемых в виде разных уровней цифрового сигнала (зачастую высокий уровень сигнал — 1 и низкий уровень сигнала — 0).

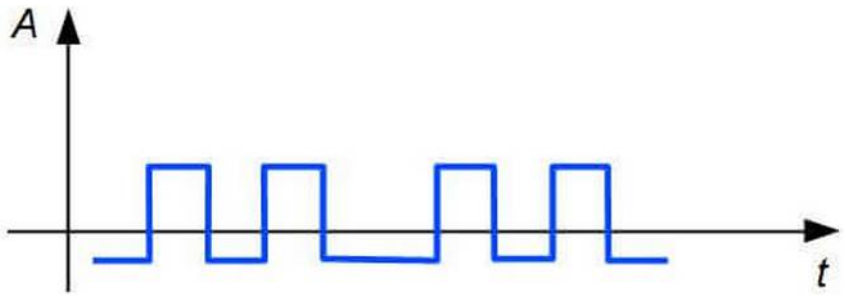


Рис. 2.4.117

Однако при передаче нерационально просто отправлять биты напрямую, поскольку это может привести к потерям данных и затруднит синхронизацию работы приемника и передатчика. А без синхронизации практически невозможно в массиве 0 и 1 структурировать начальный вид данных.

Для решения этой проблемы передачи данных используют пакетную связь — систему связи, в которой передаваемые (например, посредством электрических сигналов) сообщения разбиваются на пакеты (небольшие блоки определенного формата), каждый из которых снабжается начальной и конечной служебной информацией, помогающей определять начало и конец пакета данных, а также проверять его целостность и корректность.

Рис. 2.4.118

Существует множество протоколов, которые формируют пакеты данных согласно определенным правилам. Задокументированный протокол помогает согласовать работу передающей и принимающей сторон и передавать данные более структурированно.

Условие

Как-то раз Космокот Кас нашел сайт одного радиолюбителя. До этого Кас не очень интересовался радиосвязью, но после нескольких прочитанных статей, еще большего количества прочитанных историй и несчетного количества увиденных снимков понял, как много он упустил, и твердо решил к этому приобщиться.

На сайте он обнаружил, что радиолюбитель часто проводит сеансы связи на одном из спутников ретрансляторов, передавая и принимая через него короткие текстовые сообщения. Также радиолюбитель сообщает, что его сообщения всегда будут начинаться с символов `AL`.

Кас обрадовался, написал простенькую программу для приема сообщений, дождался спутника и, услышав сигнал, передал его в программу. Однако в программе

он не увидел ожидаемых сообщений, вместо этого там повторялся непонятный набор байт: `29 89 0D 24 24 09 24 ED A4 0D 0C AE 4C A4 24 09 24 ED A4 0E EC 2D 2E 8D 2D CC E4 0C CD EE 44 0F 2D EE AE 44 0D AC AE 6E 6C 2C EC AE 68`.

Помогите Космокоту Касу получить оригинальное сообщение, отправляемое радиолюбителем. Напишите в ответе оригинальное сообщение без известных начальных символов.

Решение

Задача напрямую связана с понятием преамбулы из теории.

1. Нужно перевести начальные символы в шестнадцатиричный вид по таблице ASCII: `41 4C`.
2. Заявленные начальные байты не содержатся в полученных, следовательно, байты начали собираться не с того места, вследствие чего собираются со сдвигом по битам.
3. В задании сказано, что набор байт повторяется, следовательно, сдвиг и сообщение постоянны. Значит, если последний байт сообщения захватывает начальные биты следующего, можно использовать эти биты как начало текущего сообщения, потому что они идентичны. Аналогичные рассуждения в случае, когда начальный байт сообщения захватывает конец предыдущего.
4. Теперь можно найти исходный текст, для этого нужно сдвигать биты с конца в начало или наоборот, до тех пор пока биты, собранные в байты, не окажутся осмысленным текстом.

Пример программы-решения

Ниже представлено решение на языке Python.

Python

```python
#Преобразование в 2-ичное число с фиксированным размером
def bbin(a):
    b = bin(a)[2:]
    return (8 - len(b))*"0" + b

# Преобразование в строку бит
def convert(msg):
    rez = ""
    for i in msg:
        rez += bbin(i)
    return rez

# Сдвиг строки бит
def swap(msg, step):
    return msg[step:] + msg[:step]

# Преобразование из строки бит в байты
def deconvert(msg):
    rez = bytearray()
    for i in range(0, len(msg), 8):
        rez.append(int(msg[i:i+8], 2))
```

```
    return rez

bytesStr = "29 89 0D 24 24 09 24 ED A4 0D 0C AE 4C A4 24 09 24 ED A4
    ↪ 0E EC 2D 2E 8D 2D CC E4 0C CD EE 44 0F 2D EE AE 44 0D AC AE 6E 6C
    ↪ 2C EC AE 68"
bytesArr = bytesStr.split(" ")
bytesInt = list(map(lambda a: int(a, 16), bytesArr)) # Преобразование
    ↪ 16-ричных чисел в int

bitsArr = convert(bytesInt)
for i in range(1, 8):
    testArr = swap(bitsArr, i)
    test = deconvert(testArr)
    print(test)
```

Полученный результат

```
bytearray(b'S\x12\x1aHH\x12I\xdbH\x1a\x19\\\x99HH\x12I\xd⌋
    ↪ bH\x1d\xd8Z]\x1a[\x99\xc8\x19\x9b\xdc\x88\x1e[\xdd\\\x88\x1⌋
    ↪ bY\\\xdc\xd8Y\xd9\\\xd0')
bytearray(b'\xa6$4\x90\x90$\x93\xb6\x9042\xb92\x90\x90$\x93\xb6\x90;\x⌋
    ↪ b0\xb4\xba4\xb73\x9037\xb9\x10<\xb7\xba\xb9\x106\xb2\xb9\xb9\x⌋
    ↪ b0\xb3\xb2\xb9\xa0')
bytearray(b"LHi! I\'m here! I\'m waiting for your messagesA")
bytearray(b'\x98\x90\xd2B@\x92N\xda@\xd0\xca\xe4\xcaB@\x92N\xda@\x⌋
    ↪ ee\xc2\xd2\xe8\xd2\xdc\xce@\xcc\xde\xe4@\xf2\xde\xea\xe4@\xda\xc⌋
    ↪ a\xe6\xe6\xc2\xce\xca\xe6\x82')
bytearray(b'1!\xa4\x84\x81$\x9d\xb4\x81\xa1\x95\xc9\x94\x84\x81$\x9d\x⌋
    ↪ b4\x81\xdd\x85\xa5\xd1\xa5\xb9\x9c\x81\x99\xbd\xc8\x81\xe5\x⌋
    ↪ bd\xd5\xc8\x81\xb5\x95\xcd\xcd\x85\x9d\x95\xcd\x05')
bytearray(b'bCI\t\x02I;i\x03C+\x93)\t\x02I;i\x03\xbb\x0bK\x⌋
    ↪ a3Ks9\x033{\x91\x03\xcb{\xab\x91\x03k+\x9b\x9b\x0b;+\x9a\n')
bytearray(b'\xc4\x86\x92\x12\x04\x92v\xd2\x06\x86W&R\x12\x04\x92v\xd2
    ↪ \x07v\x16\x97F\x96\xe6r\x06f\xf7"\x07\x96\x⌋
    ↪ f7W"\x06\xd6W76\x16vW4\x14')
```

В выводе программы видим строчку с читаемым текстом, но она по-прежнему не начинается с AL, это связано с тем, что к имеющемуся сдвигу 3 добавили 5, в результате чего весь первый символ переместился в конец. Если использовать отрицательные сдвиги — сразу получится искомая строка.

Ответ: `Hi! I'm here! I'm waiting for your messages`

Задача 2.4.3.7. Изобретаем свой мессенджер (7 баллов)

Темы: радиосвязь, демодуляция, декодирование.

Теоретическая часть

Обработка цифровых сигналов требует некоторых нововведений. Чтобы получить информацию в байтах, необходимо демодулировать сигнал, нормализовать его, привести к битам и объединить биты в байты.

Обратите внимание на доступные блоки демодуляции, они могут упростить работу. Но в общем случае для цифровых сигналов потребуется блок квадратурной

демодуляции, который принимает комплексный сигнал и выдает последовательность чисел, указывающих на уровень сигнала.

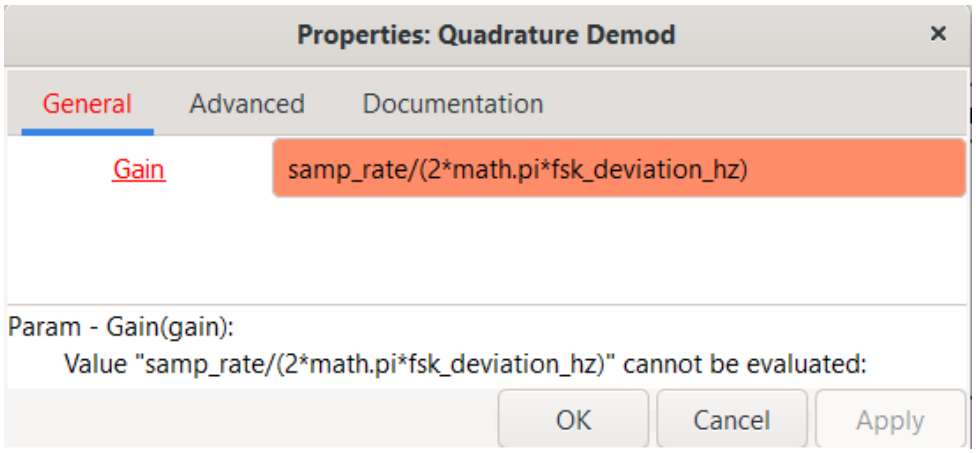


Рис. 2.4.119

Примечание. Значение `Gain` можно установить в единицу, но в таком случае позже придется нормировать сигнал по амплитуде. Рекомендуем воспользоваться формулой по умолчанию и задать необходимые переменные.

Теперь, зная уровень, нужно найти и объединить символы. Символ может быть записан сразу несколькими семплами, поэтому зная количество таких сэмплов, приходящихся на один символ, можно усреднить их в «мягкий» символ. Мягким он называется, так как еще не является 0 или 1. Для этого понадобится блок `Symbol Sync` (рис. 2.4.120).

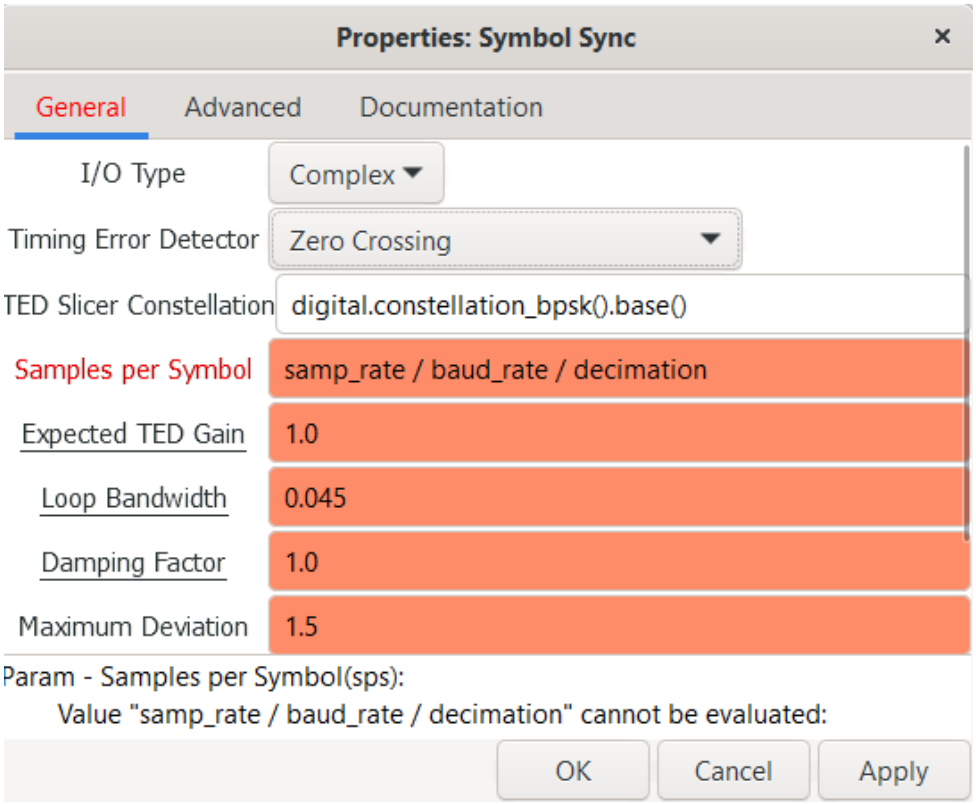


Рис. 2.4.120

Наконец, привести «мягкий» символ к 0 или 1 поможет `Binary Slicer`. Однако в последовательности обычных битов трудно найти байт информации. В этом помогает пакетная структура сообщения. Если известно, что сообщение начинается и/или заканчивается некоторой последовательностью бит, то можно попытаться найти ее с помощью семейства блоков `Correlate Access Code`. Они обнаруживают заранее известные последовательности и вешают на них тэг (о начале, окончании или размере пакета), который можно отслеживать в других блоках. Примечательно, что они будут синхронизировать поток бит, поэтому, разместив позади такого блока другой блок под названием `Repack Bits`, можно получить правильные байты информации и не допустить ошибки Каса из прошлого задания. Рекомендуем подробно ознакомиться с работой с тегами и примерами на официальной вики по `GNU Radio`.

Условие

Уже давно знакомый Космокот Кас стал регулярно общаться с другом-радиолюбителем. Чем больше он проводил сеансов связи, тем меньше ему хотелось делать кучу дополнительных действий, чтобы получить сообщения. Как известно, коты — ленивые создания, поэтому он решил переписать программу для приема сообщений в `GNU Radio`.

Для того чтобы ее тестировать, он записал `IQ`-сигнал. Ему известно, что спутник использует модуляцию `2-FSK` с девиацией 50 кГц, и скорость передачи — 100 бод/с. Радиолюбитель передает сообщения с небольшим интервалом между ними. После передачи сообщения до начала следующего сигнал со спутника отсутствует. Сообщения начинаются с символов `AL` и заканчиваются символами `EL`.

Помогите Касу составить граф для декодирования сообщений и расшифруйте сообщения на записи. Запишите в ответ переданный Касу код. `IQ`-сигнал записан с `sample rate` = 512 000 сэмпл/с.

`IQ`-файл: https://disk.yandex.ru/d/8EzKrhDTT27LNQ.

Для решения задачи можно воспользоваться справкой по основным блокам: https://disk.yandex.ru/i/DniDa-SiPmLDow.

Решение

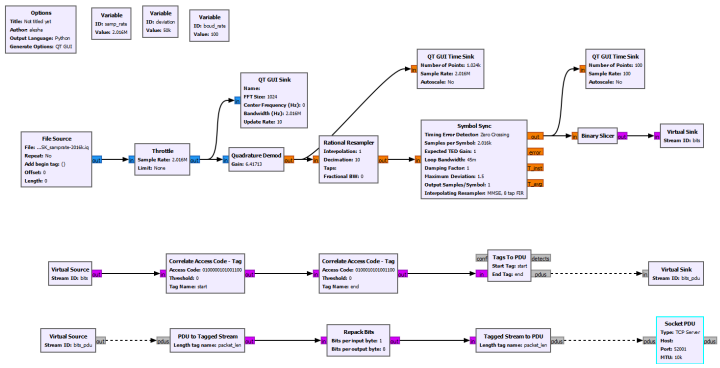


Рис. 2.4.121

Ссылка на картинку в большем размере: https://disk.360.yandex.ru/d/XKgrIOCaSIS9kw/ss_n11.png.

Ответ: 41428731.

Задача 2.4.3.8. Да кто это там шумит?! (5 баллов)

Темы: радиосвязь, демодуляция, декодирование, фильтры.

Теоретическая часть

Из всех этапов, сопровождающих обработку сигнала, еще ни разу не поднималась тема фильтрации. Прежде всего стоит отметить, что существует несколько видов фильтров:

- фильтр нижних частот — пропускает сигналы ниже указанной частоты и не пропускает сигналы выше указанной частоты (т. н. частоты среза);
- фильтр верхних частот — пропускает сигналы выше частоты среза и не пропускает сигналы ниже частоты среза;
- полосовые фильтры — пропускают заданную полосу частот;
- режекторные фильтры — не пропускают заданную полосу частот.

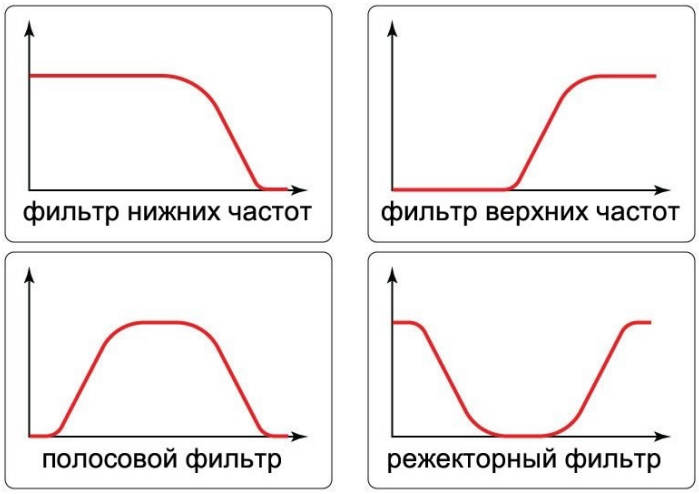


Рис. 2.4.122

Как уже упоминалось, важной характеристикой фильтра является частота среза — та частота, до или после которой фильтр будет стремиться убрать амплитуду сигнала.

Однако на практике фильтр не обрезает сигнал сразу на частоте среза, а делает это постепенно. Поэтому, помимо частоты среза, существует еще понятие переходной полосы — той части полосы частот, на протяжении которой фильтр гасит амплитуду сигнала до минимума.

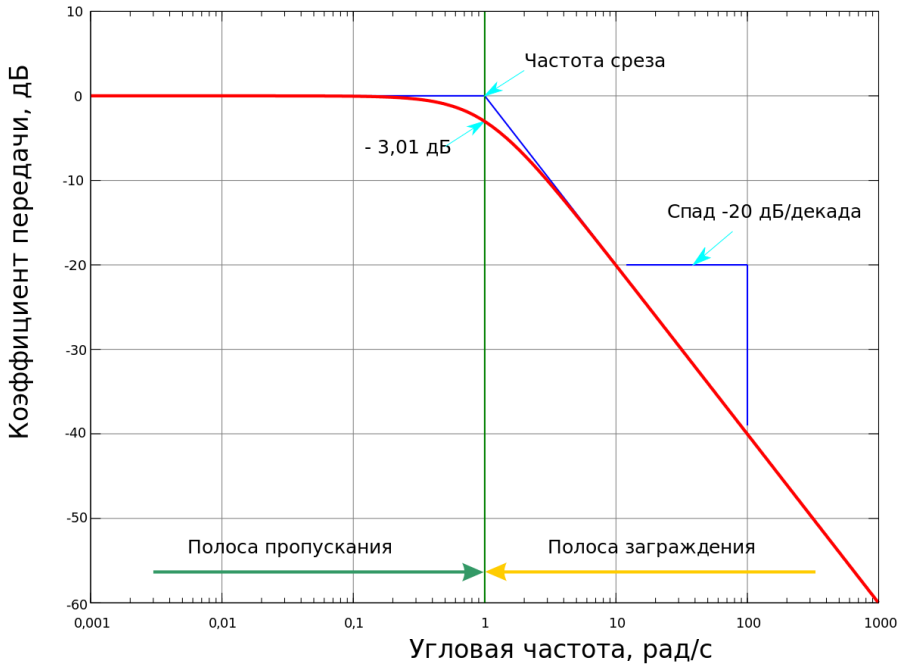


Рис. 2.4.123

Важно помнить, что фильтрация в цифровой обработке сигналов сводится к свертке (циклическому перемножению) некоторой ограниченной во временной области части сигнала на особую функцию — огибающую. Люди придумали различные такие функции и по их характерной форме называют **окнами**. В задачах ЦОС достаточно давно придуманы окна различной формы, которые при наложении на сигнал во временной области позволяют улучшить его спектральные характеристики.

Большое количество всевозможных окон обусловлено, в первую очередь, одной из главных особенностей любого оконного наложения, которая отражает взаимосвязь уровня боковых лепестков и ширины центрального лепестка — *чем сильнее подавление боковых лепестков, тем шире главный лепесток, и наоборот*.

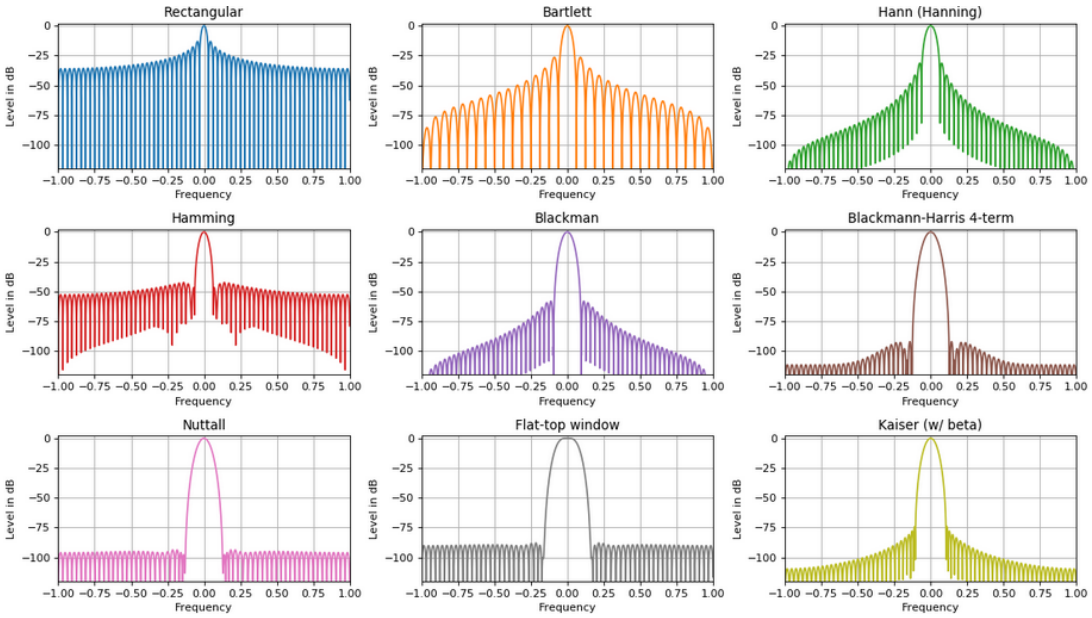


Рис. 2.4.124

Боковые лепестки — неприятная особенность оконного преобразования Фурье, которая приводит к растеканию полезного сигнала на спектре, но об этом в другой раз. Лепестки хорошо видно на иллюстрации выше с примерами различных окон.

Фильтры в `GNU Radio` представлены большим множеством блоков, позволяющих быстро и просто настроить фильтрацию. Некоторые из них отличаются высокой точностью, как `Decimating FIR Filter`, другие — высокой производительностью, как `FFT Filter`, остальные проще в настройке. Первые два упомянутых требуют `Taps` — строку, которая описывает параметры фильтра. Ее можно составить самостоятельно, пользуясь документацией, либо воспользоваться любым блоком фильтра, который оканчивается на `taps`. Такие блоки представляют собой конструктор такой строки.

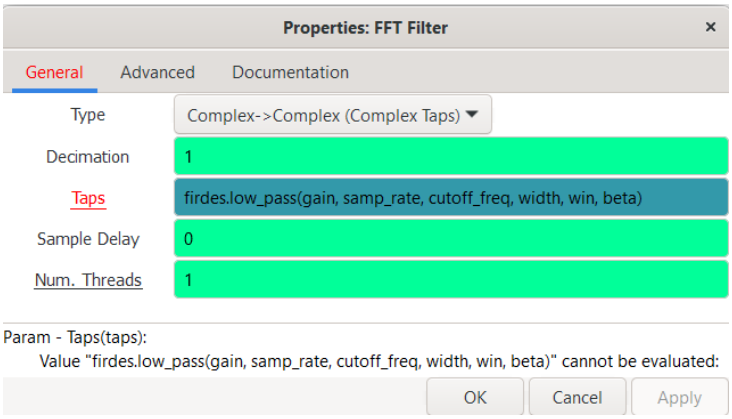


Рис. 2.4.125

Условие

Космокот Кас продолжает участвовать в сеансах связи с радиолюбителем, но на этот раз спутник пролетал в совсем неудачное время. Кас в этот момент находился в центре города и не имел с собой хорошей антенны, да и радиошума было значительно больше. Тем не менее он решил попробовать принять сигнал. Он уже начал принимать сигнал, как вдруг кто-то еще начал вещать на частоте спутника. Будем надеяться, что кто-то не специально мешает спутнику. Ну, а теперь нужно помочь Касу получить сообщения. Хорошо, что у Каса появилась привычка параллельно с декодированием включать и запись `IQ`-сигнала.

Так же как и в прошлом задании — спутник использует модуляцию `2-FSK` с девиацией 50 КГц и скорость передачи — 100 бод/с. Радиолюбитель передает сообщения с небольшим интервалом между ними. После передачи сообщения до начала следующего сигнал со спутника отсутствует. Сообщения начинаются с символов `AL` и заканчиваются символами `EL`.

`Sample rate` в записи `IQ`-сигнала $= 512\,000$ сэмплов/с.

`IQ`-сигнал: https://disk.yandex.ru/d/zF5c9U2GxM6jTQ.

Для решения задачи можно воспользоваться справкой по основным блокам: https://disk.yandex.ru/i/DniDa-SiPmLDow.

Решение

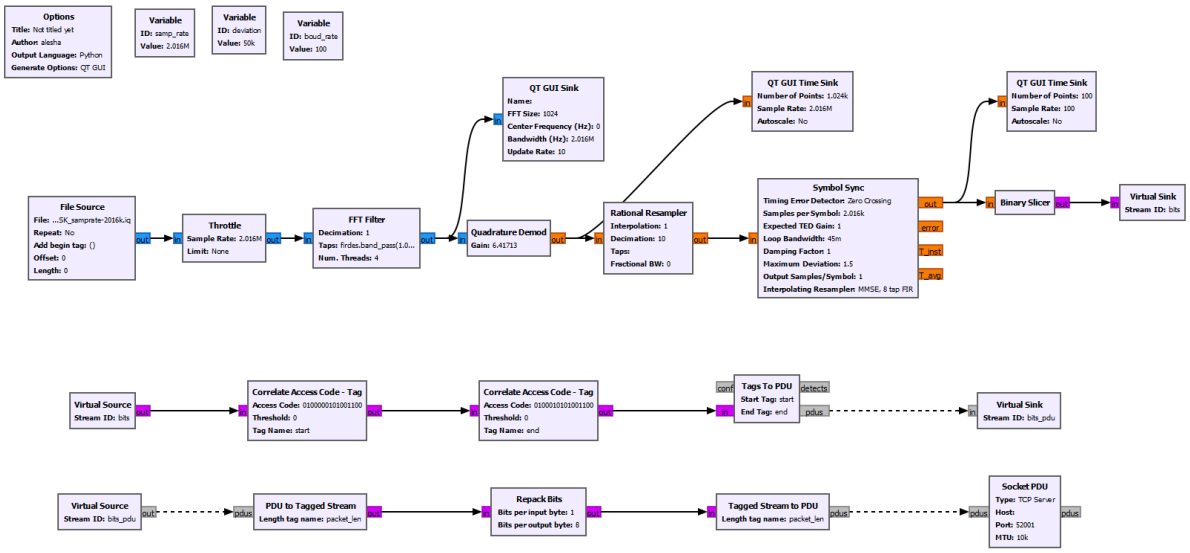


Рис. 2.4.126

Ответ: Kas, privet! Ya nauchilsya stroit' fil'try v GNU Radio! Hochu podelit'sya s toboj, vyhodi na svyaz'! Do vstrechi v efire! Konets svyazi.

Альтернативные варианты ответа на случай невозможности текстового ввода:

- Kas, privet! U menya pochemu-to ne poluchaetsa naladit' svyaz. U vas cho-to sluchilos'? Mozhet, u vas svet propal, ili chto-to proiskhodit v efire? Priyom!
- Kas, privet! U menya pochemu-to ne poluchaetsa naladit' svyaz. Neuzheli u menya slomalos' oborudovanie? Otprav', pozhalujsta, otvet, esli slyshish' menya. Do svyazi!
- Na svyaz', Kas! U menya pochemu-to ne poluchaetsa naladit' svyaz. Kazhetsya, budto by u vas v gorode glushat, ili ya oshibayus'? Zhdu tebya v efire, konets svyazi.
- Na svyaz', Kas! Ty poluchil moyo proshloe soobshchenie? Nikak ne mogu svyazat'sya ni s kem iz vashego goroda. U vas chto-to sluchilos'? Korporaty nynche pytayutsya vstavit' reklamu povsyudu. Nadeyus', ona tebe ne meshaet? Do svyazi!
- Privet, Kas! U menya pochemu-to ne poluchaetsa naladit' svyaz. Ty ne znaesh', vsyo li v poryadke? Vam nuzhna kakaya-nibud' pomoshch'? Do svyazi v sleduyushchem efire.
- Na svyaz', Kas! U menya pochemu-to ne poluchaetsa naladit' svyaz.Kosmicheskoe izluchenie seychas ochen' silnoe, mozhet v etom delo?Yasnogo tebe neba!
- Kas, privet! U menya pochemu-to ne poluchaetsa naladit' svyaz. Davai vospolzuemsya drugim kanalov svyazi? Mozhet laser?Zhdu ot tebya otveta! Konets
- Kas, privet! Davno ti ne vihodil na svyaz. Mne est' chto tebe rasskazat', eto ideya dlya zadachi na olimpiadu!Zhdu tebya v efire!Priyom!
- Kas, privet! U menya pochemu-to ne poluchayetsa svyazatsa ni s kem iz

vashego gorodaU vas cho-to sluchilos'?Tu nichego strannogo v efire ne zamechal, naprimer, v predidushie nashi seansu?Ladno, konets svyazi

2.4.4. Задачи по компетенции Баллистика

Даже идеально спроектированный космический аппарат может провалить миссию, если его орбита будет плохо подобрана под его целевую задачу. Поэтому проектирование космической миссии не обходится без определения возможных орбит для выведения спутника. Ниже предлагается познакомиться с тем, как положение орбит задается в пространстве, какие орбиты лучше подходят для определенных миссий. Не обойдется и без практической части: в ней будет рассмотрена работа в ПО баллистического моделирования `GMAT` (*General Mission Analysis Tool*), а также для тренировки будет предложено решить несколько задач в симуляторе «Орбита.Челлендж». Полученные навыки пригодятся как на втором этапе, так и на самом заключительном этапе.

Задача 2.4.4.1. Ньютон и Кеплер — герои нашего пространства и времени (2 балла)

Тема: баллистика.

Теоретическая часть

Космические искусственные спутники Земли находят очень широкое применение. Причем, многие из этих задач можно технически решать и без применения космических аппаратов: например, для фотосъемки привлекать авиацию, для радиосвязи — наземные ретрансляторы. Именно космические аппараты выгодно использовать для подобных целей, благодаря особенностям их движения. Способность длительное время двигаться на нужном расстоянии от Земли с нужной скоростью почти без затрат топлива — именно та причина, по которой космические аппараты «отвоевали» многие области применения у наземных средств и авиации.

Рис. 2.4.127

Что же позволяет космическому аппарату длительное время двигаться вокруг Земли вопреки ее притяжению? Мысленно проведем эксперимент: поднимем спутник на высоту его орбиты (скажем, 600 км над Землей) и отпустим. Спутник упадет на Землю, двигаясь по прямой линии. Если спутнику, перед тем как сбросить, придать некоторую начальную скорость по касательной к земному шару, он упадет уже по криволинейной траектории. Если скорость увеличивать, точка падения будет удаляться все дальше и дальше, пока кривая не замкнется в круг.

Скорость, при которой это происходит, называется круговой $V_{\text{кр}}(r)$. Она зависит от массы планеты, вокруг которой запускается спутник, и от высоты круговой орбиты. Круговая скорость, рассчитанная для точки на поверхности планеты (нулевая высота), называется первой космической $V_{\text{к1}}$. К примеру, первая космическая скорость для Земли равна 7 910 м/с, а круговая скорость для орбиты Марса высотой, скажем, 1 000 км равна 3 546 м/с.

Иными словами, тело, вращающееся по орбите, «падает» так же, как и все остальные тела, но благодаря касательной скорости и кривизне земной поверхности это «падение» никогда не прекращается.

Как известно из школьного курса физики, тело совершает круговое движение под действием центростремительного ускорения. Оно направлено в сторону центра вращения. Для раскрученного тела, привязанного нитью к некоторой точке, это ускорение создается силой натяжения нити.

В случае кругового движения спутника, естественно, никаких материальных нитей не существует. Центростремительное ускорение создается силой гравитации небесного тела.

Гравитация — явление, при котором любые два тела, обладающие массой, притягиваются друг к другу с равной для каждого тела силой, пропорциональной произведению их масс и обратно пропорциональной квадрату ускорения.

$$F(r) = G\frac{Mm}{r^2},$$

где M, кг — масса первого объекта (Земли или другого небесного тела),
m, кг — масса второго тела (спутника),
r, м — расстояние от центра масс Земли до объекта,
$G = 6{,}67408 \cdot 10^{-11}$, Н·м^2/кг^2 — коэффициент, называемый гравитационной постоянной Ньютона.

Напрашивается вывод о том, что орбита должна иметь форму окружности. Однако это не совсем так. Немецкий ученый Иоганн Кеплер установил законы движения небесных тел, которые справедливы и для искусственных космических аппаратов: траектория движения объектов в гравитационном поле небесных тел имеет форму эллипса.

Эллипс — замкнутая кривая на плоскости, которая характеризуется большой и малой полуосями. Помимо этого, эллипс имеет две особые точки, называемыми фокусами: F_1 и F_2. Если в точках эллипса закрепить концы нити, то при ее натяжении карандашом можно будет нарисовать правильный формы эллипс.

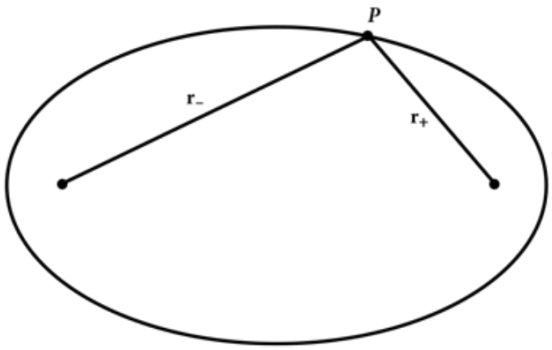


Рис. 2.4.128

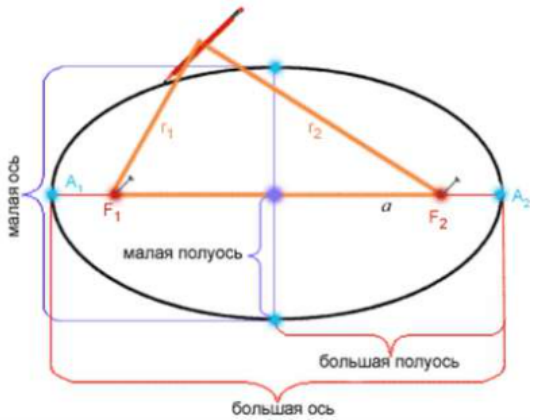


Рис. 2.4.129

Именно это свойство определяет эллипс как фигуру: сумма расстояний от любой точки на эллипсе до его фокусов постоянна и равна большой оси эллипса.

Другим важным параметром эллипса является эксцентриситет – числовая характеристика конического сечения, показывающая степень его отклонения от окружности. Вычисляется эксцентриситет по следующей формуле:

$$e = \sqrt{1 - \frac{b^2}{a^2}},$$

где a и b — большая и малая полуоси эллипса.

Эксцентриситет определяет вытянутость эллипса: чем больше его значение, тем более вытянутым будет сам эллипс.

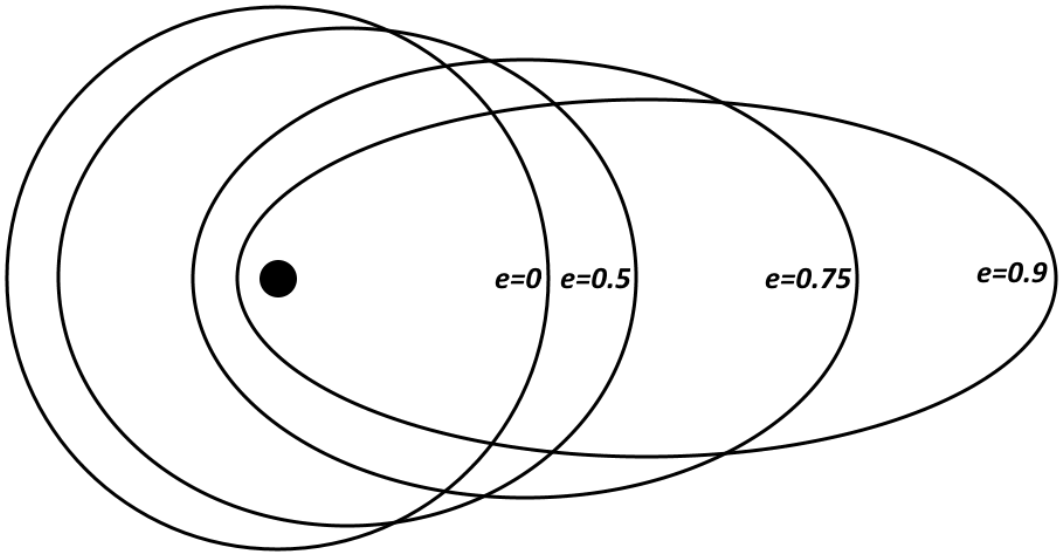


Рис. 2.4.130

При эксцентриситете, равном нулю, эллипс превращается в окружность, при эксцентриситете, равном единице, — в параболу, при эксцентриситете, большем, чем единица, — в гиперболу.

Условие

Космический аппарат вращается по круговой орбите вокруг неизвестной планеты, названной Планета ОБУ, на высоте 400 км от ее поверхности. С помощью научного оборудования на борту удалось определить, что радиус планеты составляет 14 500 км. Помогите космокоту Касу рассчитать массу планеты ОБУ, если известно, что аппарат делает полный виток вокруг планеты за 4 ч.

Решение

Для начала рассчитаем круговую скорость аппарата, для этого посчитаем длину окружности, по которой он вращается по формуле $2\pi(r+h)$, где r — радиус планеты, h — высота полета спутника над поверхность планеты, и поделим эту величину на время обращения вокруг планеты, получается 6501,35146 м/с.

Далее воспользуемся формулой для первой космической скорости при движении по окружности с постоянной скоростью, где $V_1 = 6501{,}35146$ м/с — первая космическая скорость для данной планеты на заданной высоте, $G = 6{,}67 \cdot 10^{-11}$ м$^3 \cdot$кг$^{-1} \cdot$с$^{-2}$ — гравитационная постоянная, $R_0 = 14\,500$ км, $h = 400$ км — высота спутника над поверхностью планеты, M — масса планеты (искомое). Выражаем M, считаем, получаем ответ.

Ответ: $9{,}44208 \cdot 10^{24}$ кг.

Задача 2.4.4.2. Эллиптические орбиты (2 балла)

Тема: баллистика.

Условие

Космический аппарат, находящийся над планетой ОБУ, которая, напомним, имеет радиус 14 500 км, совершил маневр перехода с круговой на эллиптическую орбиту, однако двигатель выдал не тот импульс, на который рассчитывала команда на Земле, и теперь параметры орбиты неизвестны.

Помогите Касу узнать большую полуось и эксцентриситет орбиты, если известно, что аппарат в перигее проходит на высоте 500 км над поверхностью планеты, а в апогее — на высоте 20 500 км над поверхностью планеты.

Решение

Большая полуось орбиты (a) есть сумма высоты перигея, радиуса орбиты и расстояния из центра эллипса до одного из его фокусов. Последнее рассчитывается по формуле $e \cdot a$. Высота апогея — это два фокусных расстояния в сумме с радиусом планеты и высотой перигея. Но так как эта высота над поверхностью планеты, то радиус планеты надо вычесть. Тогда можно составить систему:

$$\begin{cases} e \cdot a + 14\,500 + 500 = a, \\ 2e \cdot a + 500 = 20\,500. \end{cases}$$

Решив эту систему, получаем, что $a = 20\,500, e = 0{,}4$.

Ответ: 20 500 км; 0,4.

Задача 2.4.4.3. Кеплер продолжает быть с нами (2 балла)

Тема: баллистика.

Теоретическая часть

Как известно, орбита аппарата имеет форму эллипса. Однако важна не только форма орбиты, но и ее расположение в пространстве, и чтобы описать положение орбиты в пространстве, нужно воспользоваться определенными параметрами.

Одним из способов задания положения орбиты в пространстве являются кеплеровы элементы. Некоторые из них уже упоминались как характеристики формы эллипса: эксцентриситет и большая полуось. Однако, помимо этих двух элементов, используются еще четыре. Ознакомиться с ними можно по видео: https://youtu.be/cAnaiBx-kss.

Условие

Космокот Кас вместе с учеными планирует запуск спутника и рассчитывает Кеплеровы элементы орбиты для будущего спутника. Помогите Касу рассчитать наклонение орбиты, если известно, что наклонение для орбиты надо выбрать таким образом, чтобы оно было минимальным и позволяло космическому аппарату пройти ровно над станцией, расположенной в городе Новосибирске $(55{,}0415, 82{,}9346)$. Радиус Земли примите 6 371 км.

Наклонение округлить до сотых долей.

Решение

Для того чтобы спутник мог пройти ровно над городом, наклонение орбиты должно быть не меньше широты, на которой расположен город. Так как по условию требуется взять минимальное значение, то для ответа стоит выбрать саму широту города Новосибирска.

Ответ: 55,04.

Задача 2.4.4.4. Элементы орбиты в GMAT (5 баллов)

Тема: баллистика.

Теоретическая часть

По сухой теории не всегда можно в достаточной степени понять, что такое элементы орбиты, и на что каждый из них будет влиять. Поэтому чтобы лучше

изучить данную тему, предлагаем поработать в программном обеспечении General Mission Analysis Tool (GMAT). Это ПО предназначено как раз для построения орбит и моделирования космических миссий.

Скачать данное ПО можно по ссылке: https://sourceforge.net/projects/gmat/.

Для ряда задач используются достаточно определенные орбиты. В следующем видео можно ознакомиться с работой в ПО GMAT, а также с некоторыми типовыми орбитами. Рекомендуем построить каждую орбиту в ПО GMAT, чтобы лучше понять их специфику.

Условие

1 марта 2025 года в 00:00 по UTC на орбиту Земли выводится спутник. Элементы его орбиты:

- большая полуось: 8 000 км;
- эксцентриситет: 0;
- наклонение: 45°;
- долгота восходящего узла: 0°;
- аргумент перицентра: 0°;
- истинная аномалия на момент начала моделирования: 0°.

Космокот Кас находится в ЦУП, расположенном в следующем месте:

- широта: 42,8816;
- долгота: 47,6392;
- высота над уровнем моря: −16 м.

Помогите Касу узнать, во сколько начнется первый сеанс связи со спутником 3 марта, и сколько в сумме будет держаться связь со спутником за этот день, если антенный комплекс ЦУП способен стабильно принимать сигнал только тогда, когда спутник находится на высоте, равной или превышающей 60° над горизонтом.

Рекомендуется выполнять задание с помощью ПО GMAT.

В качестве ответа приведите количество секунд, прошедшее от начала суток 3 марта до начала первого сеанса связи (округлите до целого числа) и продолжительность сеансов связи за сутки с 00:00 UTC 3 марта по 00:00 UTC 4 марта в секундах (округлите до десятых долей)..

Решение

Задаем все известные данные по орбите и наземной станции в GMAT, создаем контакт локатор, запускаем скрипт и смотрим в output'е первый сеанс 3 марта, а также по выводу считаем общее количество секунд всех контактов 3 марта.

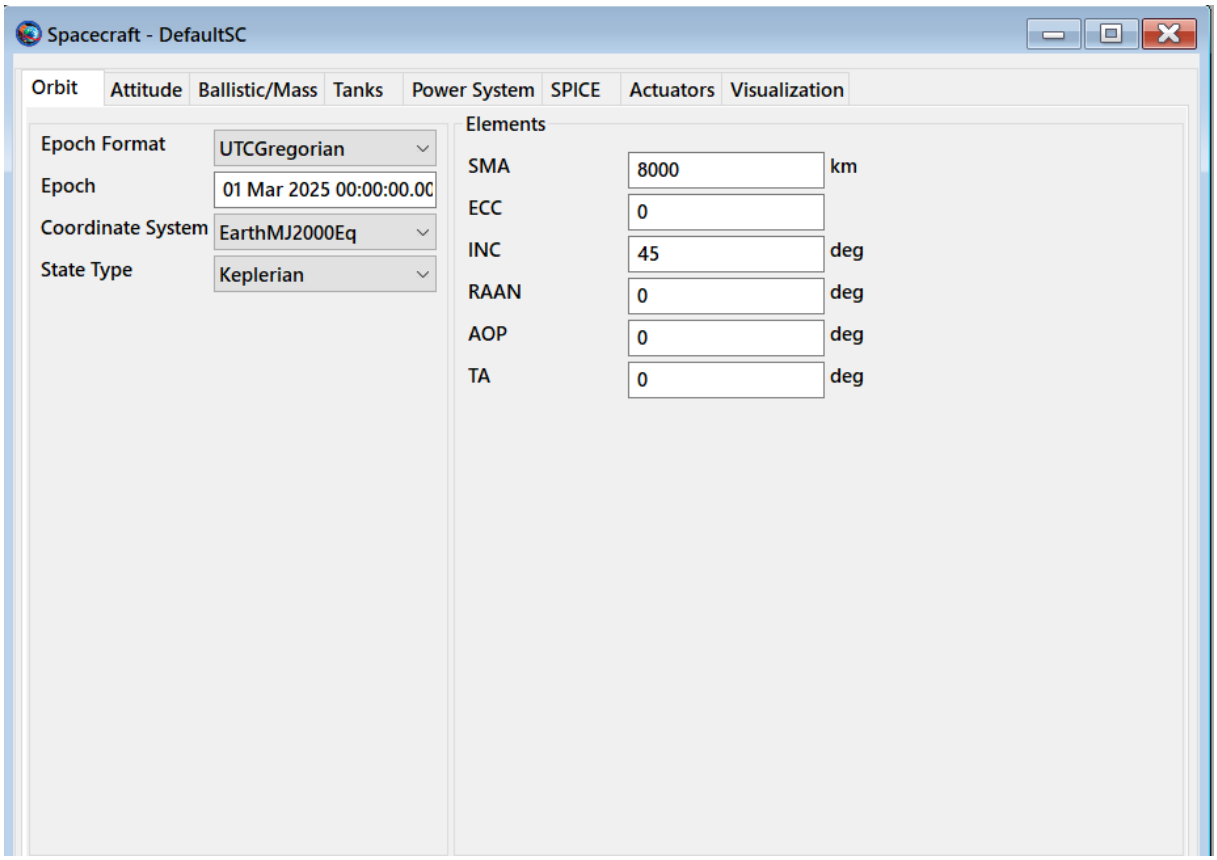


Рис. 2.4.131

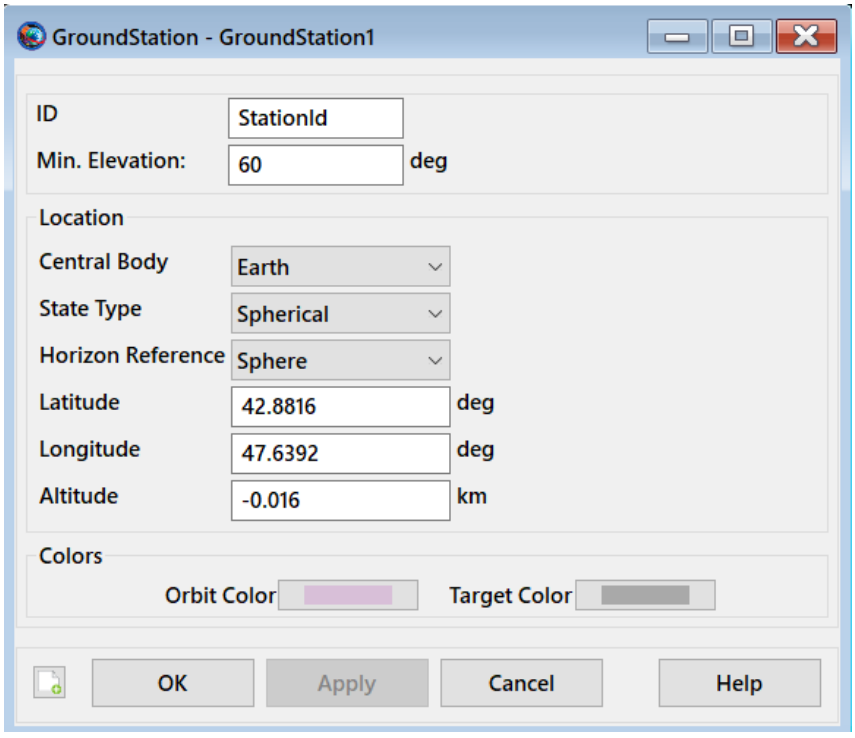


Рис. 2.4.132

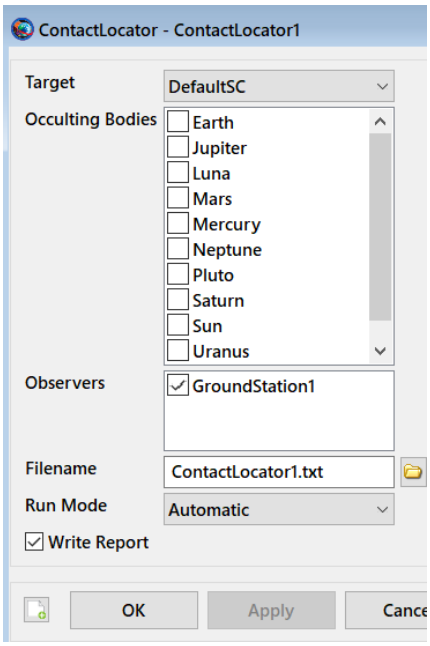


Рис. 2.4.133

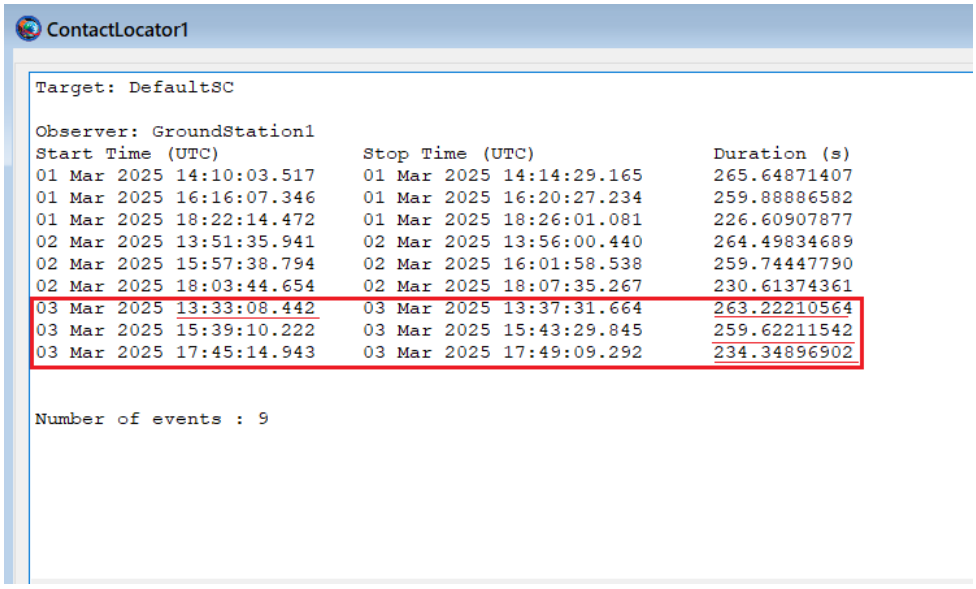


Рис. 2.4.134

Ответ: начало первого сеанса: 48 788 с, продолжительность: 757,2 с.

Задача 2.4.4.5. Аппарат над заданной точкой (6 баллов)

Тема: баллистика.

Условие

Спутник может быть размещен на произвольной орбите 20 марта 2025 года в 00:00 по UTC. Задайте параметры орбиты через Кеплеровы элементы так, чтобы 21 и 22 марта в 09:00 по UTC каждого дня аппарат прошел над Находкой.

Координатами Находки считать GPS — координаты: 41,8138 по широте, 132,873 по долготе.

Начальные данные:

- радиус Земли, м: 6371008,8;
- гравитационный параметр, $\mu = G \cdot M$, м^3/с^2: $3{,}986004418 \cdot 10^{14}$;
- средняя скорость вращения Земли, об/сут: 1,00273781191135448;
- начальный угол вращения Земли: 108,9837°.

Правила начисления баллов:

- аппарат в любой из дней проходит с отклонением до 100 км от цели и до 60 мин от нужного времени — 0,5 балла;
- аппарат в любой из дней проходит с отклонением до 100 км от цели и до 15 мин от нужного времени — 1 балл;
- аппарат в любой из дней проходит с отклонением до 50 км от цели и до 60 мин от нужного времени — 0,5 балла;
- аппарат в любой из дней проходит с отклонением до 50 км от цели и до 15 мин от нужного времени — 1 балл.

Решение

Решение (задание в симуляторе «Орбита») по образу и подобию https://drive.google.com/file/d/1bRrARogVYMs1xu0mcfQoid6SaviyJKDK/view.

Ответ.

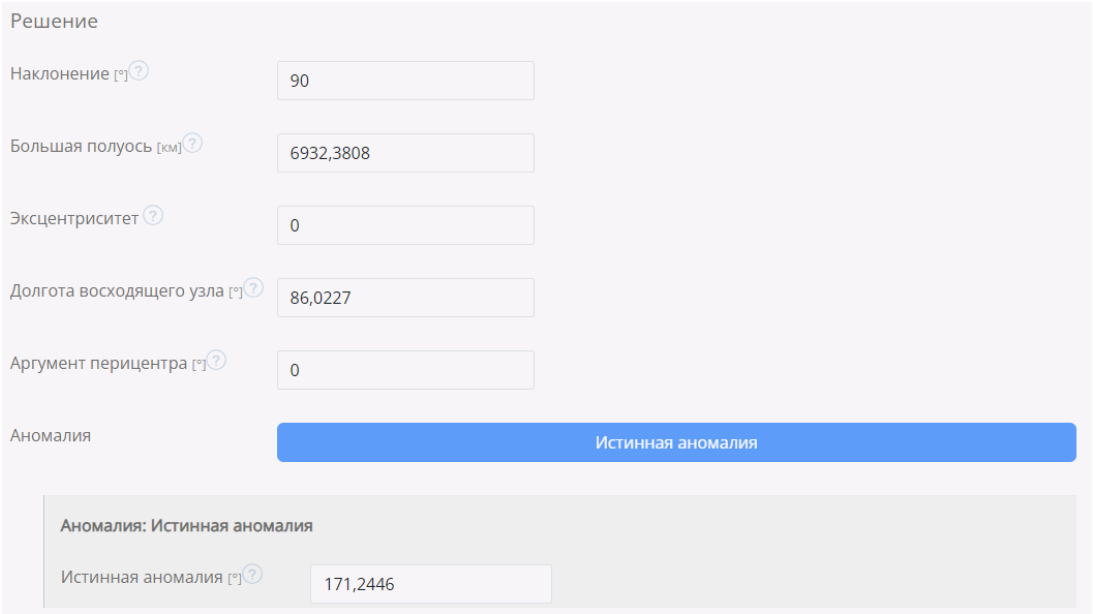


Рис. 2.4.135

Задача 2.4.4.6. Обеспечение энергией (8 баллов)

Тема: баллистика.

Условие

Космокот Кас решил запустить на орбиту спутник для съемки поверхности Земли. Оптика на спутнике позволяет проводить съемку, если апоцентр не превышает 750 км, а перицентр находится не ниже, чем 350 км. Для эффективного использования ресурса спутника Кас хочет подобрать его орбиту таким образом, чтобы он находился на солнце максимальное возможное время. Однако, из-за долгого нахождения спутника на солнце, аппаратура рискует перегреться, поэтому аппарату необходимо хотя бы 15 мин за виток вокруг Земли находиться в тени.

Помогите Касу подобрать элементы орбиты, отвечающие приведенным требованиям. В качестве ответа задайте параметры орбиты через кеплеровы элементы.

Время начала моделирования: 1 марта 2025 года 00:00 по UTC. Длительность моделирования: 48 ч.

Начальные данные:

- радиус Земли, м: 6 371 008,8;
- гравитационный параметр, $\mu = G \cdot M$, м^3/с^2: $3{,}986004418 \cdot 10^{14}$;

Правила начисления баллов:

- если время нахождения в тени за виток составляет менее 15 мин (900 с), ставится 0 баллов;
- иначе балл рассчитывается по формуле: $8 \cdot 2^{-1/500 \cdot (time - 15 \cdot 60)}$, где $time$ — время нахождения спутника на текущем витке в тени.

Решение

Для удобства работы с орбитой выберем полярную круговую орбиту, наклонение 90, эксцентриситет. Так как орбита круговая, можно задать аргумент перицентра, также равный нулю, чтобы перицентр располагался на экваторе. Аномалия также неважна в этой задаче, возьмем ее 0.

Высоту орбиты можно принять любой, удовлетворяющей условиям задачи, но стоит взять выше среднего значения, для удобства дальнейшего расчета большая полуось 7 000 км.

Так как время симуляции невелико, главным параметром, определяющим положение спутника относительно света и тени от Земли, будет долгота восходящего узла. Воспользовавшись инструментами `GMAT`, а конкретнее `Eclipse locator`, можно отслеживать время, в течение которого спутник находится в тени за 1 виток.

Так как это единственный оставшийся параметр системы, используя данные `Eclipse locator`, можно за несколько итераций подобрать его значение таким образом, чтобы минимальное время в тени было чуть больше либо равным 900 с (15 мин из условия задачи) для того, чтобы набрать максимальный балл за задачу. Достигается это условие при значении ДВУ $= 280{,}75$.

Ответ.

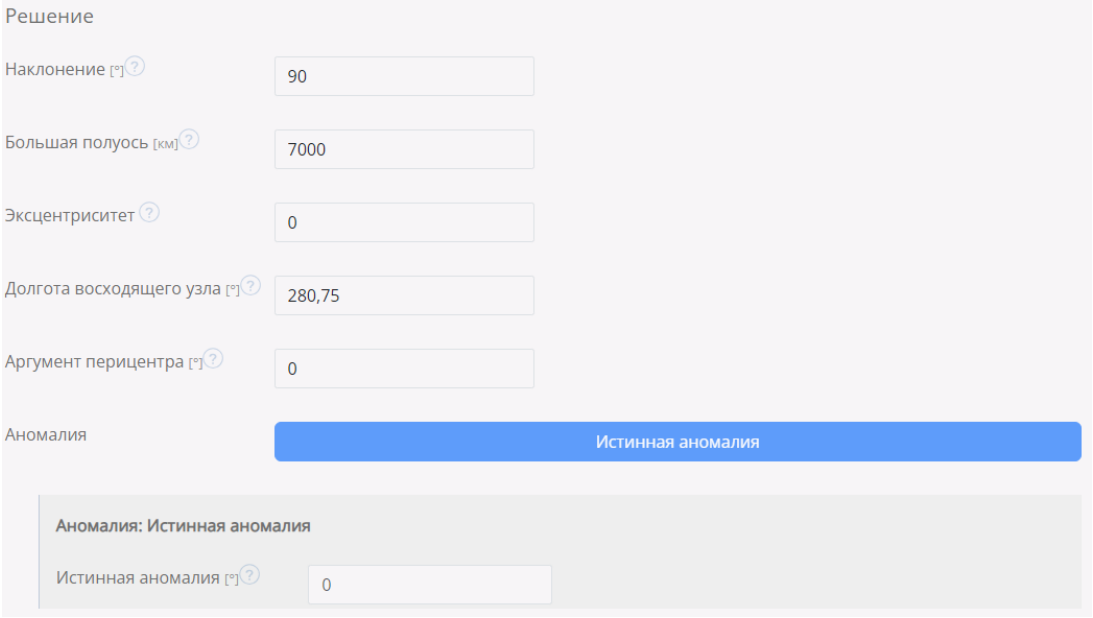

Решение
Наклонение [°]
90
Большая полуось [км]
7000
Эксцентриситет
0
Долгота восходящего узла [°]
280,75
Аргумент перицентра [°]
0
Аномалия
Истинная аномалия
Аномалия: Истинная аномалия
Истинная аномалия [°]
0

Рис. 2.4.136

3. Второй отборочный этап

3.1. Работа наставника НТО на этапе

На втором отборочном этапе НТО участникам предстоит решать как индивидуальные, так и командные задачи в рамках выбранного профиля. Подготовка к этому этапу требует от них не только глубокого понимания предметной области, но и умения работать в команде, эффективно распределять роли и применять полученные знания на практике. Наставник играет здесь важную роль — он помогает участникам выстроить осмысленную и целенаправленную траекторию подготовки.

Вот основные направления, в которых наставник может поддержать участника:

- **Подготовка по образовательным программам НТО.** Наставник может готовить участников, используя готовые образовательные программы по технологическим направлениям, рекомендованные организаторами, а также адаптировать их под уровень подготовки школьников.
- **Разбор заданий прошлых лет.** Изучение задач второго отборочного этапа прошлых лет помогает участникам понять формат заданий, определить типовые ошибки и выработать стратегии решения.
- **Онлайн-курсы.** Участники могут пройти курсы по разбору задач прошлых лет или курсы, рекомендованные разработчиками отдельных профилей. Наставник может включить эти курсы в план подготовки, а также сопровождать процесс изучения и помогать с возникшими вопросами.
- **Анализ материалов профиля.** Совместный разбор методических материалов, размещенных на страницах профилей, помогает уточнить требования к участникам и направить подготовку на ключевые темы.
- **Практикумы.** Это важный элемент подготовки, позволяющий применять знания на практике. Наставник может:
  - ◇ организовать практикумы по методическим материалам с сайта профиля;
  - ◇ декомпозировать задачи заключительного этапа прошлых лет на отдельные элементы и проработать их с участниками;
  - ◇ провести анализ требуемых профессиональных компетенций и спланировать занятия для развития наиболее значимых из них;
  - ◇ направить участников на практикумы и мероприятия от организаторов, которые анонсируются в официальных сообществах НТО, например, в телеграм-канале для наставников: https://t.me/kruzhok_association.
- **Командная работа.** Одной из ключевых задач наставника на втором этапе является помощь в формировании команды или в поиске подходящей. Наставник может помочь участникам определить их сильные стороны, выбрать роль в команде и сориентироваться в процессе командообразования, включая участие в бирже команд в рамках конкретного профиля.

Если участники не прошли отборочный этап

Случается, что несмотря на усилия и серьезную подготовку, участники не проходят во второй или заключительный этап Олимпиады. В такой ситуации особенно важна поддержка наставника.

- **Поддержка и признание усилий.** Наставнику важно подчеркнуть ценность пройденного пути: полученные знания, навыки, преодоленные трудности и личностный рост. Это помогает участникам сохранить мотивацию и не воспринимать результат как окончательное поражение.
- **Рефлексия.** Полезно организовать встречу для обсуждения впечатления от участия, трудности, с которыми столкнулись школьники и то, что они узнали о себе и команде. Наставник может направить разговор в конструктивное русло: какие выводы можно сделать? Что сработало хорошо? Что можно улучшить?
- **Анализ ошибок и пробелов.** Наставник вместе с участниками анализирует, какие темы вызвали наибольшие затруднения, чего не хватило в подготовке — теоретических знаний, практических навыков, командного взаимодействия. Это позволяет выстроить более эффективную стратегию на будущее.
- **Планирование дальнейшего пути.** Участникам можно предложить:
  - ◇ продолжить углубленное изучение профиля или смежных направлений;
  - ◇ заняться проектной деятельностью, которая укрепит знания и навыки;
  - ◇ сформировать план по подготовке к следующему циклу НТО, начиная с работы над типовыми заданиями и курсами.
- **Создание устойчивой мотивации.** Важно показать школьникам, что участие в НТО — это не просто соревнование, а часть большого образовательного маршрута. Даже неудачный результат может стать толчком к профессиональному росту, если воспринимать его как точку развития, а не как конец пути.

Таким образом, наставник помогает участникам не только готовиться к этапам НТО, но и справляться с неудачами, выстраивать долгосрочную стратегию и сохранять интерес к инженерному и технологическому творчеству.

3.2. Инженерный тур

Задачи второго этапа делятся на командную и индивидуальную часть. Второй этап профиля проходит на платформе «Орбита.Челлендж» (https://orbita.education/ru/).

Уже начиная с первого этапа, идет подготовка к задаче заключительного этапа в рамках инженерного тура. Индивидуальная часть является продолжением задач инженерного тура. Помимо этого, решения индивидуальной части входят в состав первой командной части.

Индивидуальная часть может дать команде до 40 баллов: по 10 баллов в каждом направлении. В таблице 3.2.1 показаны компетенции по направлениям, развиваемые у участников по ходу участия в профиле. Можно увидеть, что они частично перекрывают компетенции, необходимые на финале.

Таблица 3.2.1. Компетенции по направлениям

Программист МК	**Инженер связи**	**Инженер-программист ПН**	**Баллистик**
Написание алгоритмов управления КА	Общие принципы реализации протоколов связи	Понимание принципов работы датчиков; навыки программирования	Основы орбитальной механики; работа в ПО «Орбита»

В рамках командного этапа, помимо отборочных испытаний, предусмотрена также тренировочная задача. Отборочные испытания предполагают проверку сразу нескольких компетенций, связанных с направлениями деятельности. Командный этап может принести команде до 60 баллов.

В таблице 3.2.2 представлены задачи для командного этапа.

Таблица 3.2.2. Командные задачи

Первая командная задача	**Вторая командная задача**
• Подбор орбиты. • Работа с протоколом связи. • Написание алгоритмов управления КА	• Подбор орбиты с учетом зон интереса. • Работа с протоколом связи, помехоустойчивость. • Написание алгоритмов управления КА. • Обработка данных матричных датчиков, примитивная идентификация объектов

3.2.1. Индивидуальные задачи

Задача 3.2.1.1. Задача инженера-программиста полезной нагрузки (10 баллов)

Тема: программирование.

Условие

Облачность является одной из самых значимых проблем для спутниковой съемки поверхности Земли. Облака могут затруднять или полностью скрывать наблюдения, особенно если миссия нацелена на получение детализированных изображений для картографии, мониторинга сельскохозяйственных угодий или оценки природных ресурсов. Компании и космические миссии используют несколько стратегий для детектирования облаков и минимизации влияния облачности на результаты съемки.

Одним из ведущих методов является мультиспектральная съемка. Спутники, такие как Landsat, Sentinel и другие, оснащены камерами, которые работают в разных диапазонах электромагнитного спектра, включая видимый свет, ближний и средний инфракрасный диапазоны. В инфракрасном диапазоне облака хорошо различимы, поскольку их температура ниже температуры земной поверхности. Это позволяет отделить облака от объектов на земле, особенно если используется комбинация нескольких каналов спектра.

В реальных миссиях для детектирования облаков часто применяется метод анализа рефлектанса — оценки того, как свет отражается от объектов. Белые облака обладают высоким отражением в видимом диапазоне, в то время как для поверхности Земли это значение ниже. Например, для картографических миссий, таких как Sentinel-2, используются комбинации видимого и инфракрасного диапазонов для создания индексов облачности (в частности, NDSI — Normalized Difference Snow Index). Эти индексы помогают точно выделить облака на снимках. Также активно применяются алгоритмы машинного обучения и искусственного интеллекта для автоматической классификации изображений. Данные с низкого разрешения анализируются нейросетями, которые обучены выявлять облака на основании множества различных признаков: форма, яркость, спектральные характеристики и движение. Эти системы активно используются компаниями типа Planet Labs и в миссиях NASA, где задача состоит в том, чтобы эффективно фильтровать кадры с высоким уровнем облачности.

Существуют и другие подходы к обработке облаков. Например, миссии часто используют временные ряды изображений. В этом случае несколько снимков одного и того же участка делаются через короткие промежутки времени, и на их основе объединяются данные с минимальным уровнем облачности. Этот метод широко применяется в программах типа Google Earth Engine для составления карт Земли с использованием чистых участков снимков за длительный период времени.

Отдельно стоит упомянуть о проблемах, связанных с тонкими облаками или дымкой, которые могут не перекрывать поверхность полностью, но снижать качество снимков. Для таких ситуаций также применяются специализированные фильтры и алгоритмы, которые могут на основе спектрального анализа отличать полупрозрачные облака от чистой атмосферы. Системы картографической съемки с применением облачных фильтров представляют собой сложные решения, интегрирующие оптические технологии, алгоритмы обработки данных и умные системы принятия решений. Эти решения позволяют значительно повысить качество снимков, избегая съемок участков с высокой степенью облачности.

Задача

Космический аппарат (КА), предназначенный для картографической съемки поверхности Земли, оснащен двумя сенсорами. Основная камера обладает высоким разрешением, но ее использование требует значительного потребления энергии, поэтому для предварительной оценки условий съемки используется небольшой сенсор с матрицей 100×100 пикселей. Этот сенсор постоянно делает снимки местности.

Каждый снимок с маленького сенсора содержит:

- время снимка UTC,
- географические координаты спутника на момент съемки (`lat_lon`),
- массив данных 100×100 пикселей (каждый пиксель представлен ASCII символом).

Пример пакета:

```
2018-06-14_10:12:00.000_55.7522_37.6156_dlkfgvmih...vogrhgvo
```

Пример строки, задающей интересующий квадрат:

```
"55.7522_37.6156_55.7522_37.6156"
```

Формат:

```
"(lat min)_(lat max)_(lon min)_(lon max)"
```

Необходимо разработать систему фильтрации, которая будет анализировать снимки с маленького сенсора и принимать решение о том, когда включать основную камеру. Условия для включения следующие:

- снимок сделан в пределах заданных координат местности;
- на снимке не более 40% пикселей занято облаками.

Облака детектируются с помощью анализа яркости пикселей (пороговое значение яркости для облаков заранее известно).

Дополнительные указания:

- Квадрат местности задан географическими координатами его углов.
- Яркость пикселей, указывающая на облачность, варьируется от 0 до 127.
- Порог яркости для облаков: 100.
- Программа должна возвращать следующую строку:

  ```
  (процент облаков на снимке)_(флаг включения основной камеры)
  ```

  Процент облаков на снимке — целая часть от рассчитанной облачности без округлений ($31{,}3 \to 31, 31{,}8 \to 31$).

Пример выходной строки:

- `80_0` (облачность: 80%, статус: 0 — выкл);
- `15_1` (облачность: 15%, статус: 1 — вкл);
- `0_0` (снимок сделан за пределами области интереса).

Примеры

Пример №1

Стандартный ввод
2018-06-14_10:12:00.000_37.6156_55.7522_bMLfgQEjkbX9UHmWngXxJ6F ZQWGvd79z1XWObV51oAvJTY2rqaj4XsRO3AClbsH8Tmh9i7KhxpPC95lHiSemNl GJZNGSrsgUvFkj 36.7000_37.6156_55.7522_59.6382

Стандартный вывод
36_1

Критерии оценивания

Максимально возможный балл — 10. Первые две попытки идут без штрафа. За каждую следующую попытку максимально возможный балл уменьшается на 10% (т. е. на третьей попытке до 9 баллов, на четвертой — до 8 и т. д.) до 1 балла и далее не уменьшается.

Решение

Ниже представлено решение на языке Python.

Python

```python
import sys
import math

image_packet = sys.stdin.readline().rstrip() # Чтение пакета
area = sys.stdin.readline() # Чтение области интереса

CLOUD_BRIGHTNESS_THRESHOLD = 100
MAX_CLOUD_COVERAGE = 0.4

def is_within_area(latitude, longitude, area_bounds):
    lat_min, lat_max, lon_min, lon_max = area_bounds
    return lat_min <= latitude <= lat_max and lon_min <= longitude <=
    ↪ lon_max

def calculate_cloud_coverage(pixel_data):
    cloud_pixels = sum(1 for pixel in pixel_data if pixel >=
    ↪ CLOUD_BRIGHTNESS_THRESHOLD)
    total_pixels = len(pixel_data)
    return cloud_pixels / total_pixels * 100

def process_image_packet(packet, area_str):
    date_str, timestamp_str, lat_str, lon_str, pixel_data_str =
    ↪ packet.split('_', 4)
    latitude = float(lat_str)
    longitude = float(lon_str)

    pixel_data = [ord(char) for char in pixel_data_str]

```

```
26      lat_min, lat_max, lon_min, lon_max = map(float,
        ↪  area_str.split('_'))
27      area_bounds = (lat_min, lat_max, lon_min, lon_max)
28
29      if not is_within_area(latitude, longitude, area_bounds):
30          return "0_0"
31
32      cloud_coverage = calculate_cloud_coverage(pixel_data)
33      main_camera_status = 1 if cloud_coverage <= MAX_CLOUD_COVERAGE *
        ↪  100 else 0
34
35      return f"{int(math.trunc(cloud_coverage))}_{main_camera_status}"
36
37  output = process_image_packet(image_packet, area)
38
39  sys.stdout.write(output) # Вывод ответа
```

Ответ: 36_1.

Задача 3.2.1.2. Задача инженера связи (10 баллов)

Темы: радиосвязь, программирование.

Условие

Всем уже знакомый космокот Кас планирует запустить научный спутник с важным научным оборудованием и передатчиком для передачи данных на Землю. Это оборудование формирует текстовую строку с данными. Помогите Касу разработать программы для передатчика и наземного приемника, которые бы обеспечили отправку ценных научных данных на Землю. Известно, что данные в процессе передачи могут искажаться из-за сторонних помех случайным образом.

Программа передатчика

На вход функции `proceed` программы передатчика поступают раз в $0{,}015n \pm \pm 5\%$ секунд отдельные текстовые сообщения с научными данными переменной длины. Длина строки научных данных гарантированно n символов. Вывод функции отправляется на передатчик спутника. Скорость передатчика составляет 100 байт/с. Если передатчик не успел передать данные до прихода следующих данных, текущая отправка данных прерывается и запускается следующая. Программа передатчика не может и не должна иметь сквозного буфера между ее исполнениями в силу ограниченности памяти на спутнике. Временем выполнения программы передатчика пренебречь.

Программа приемника

На вход функции `proceed` поступают данные с приемника. Скорость приема приемника соответствует скорости передатчика. Входные данные подаются частями по $16n$ бит. В связи с ограничениями памяти на наземной станции приема, разрешается хранить в сквозном буфере не более $80n$ бит. Если сообщение не было

найдено или не было декодировано, функция должна вернуть пустую строку. Также в функцию передается признак окончания работы приемника, его истинность означает, что текущая часть данных — последняя. Учтите, что при истинности этого признака размер части данных может быть $\leqslant 16n$. При истинности этого признака функция продолжит вызываться до тех пор, пока не вернет пустую строку, чтобы была возможность дообработать сквозной буфер. Функция должна вернуть строку типа `str`.

Примечания

Значение переменной n определяется в процессе выполнения программы. Известно, что ее значение находится в диапазоне $[130, 170]$.

Формат входных данных

Программа передатчика

Программа передатчика должна иметь функцию `proceed`, которой на вход будет подаваться строка (`str`) с научными данными. Функция должна возвращать объект типа `bytes`, содержимое которого передается на передатчик.

Пример объявления функции:

```
1 def proceed(data: str) -> bytes:
```

Программа приемника

Пример объявления функции:

```
1 def proceed(inp, buf, end):
2     return ""
```

где:

- `inp` — часть данных с приемника — `numpy`-массив с типом данных `bool` длиной $16n$;
- `buf` — сквозной буфер — `numpy`-массив с типом данных `bool` длиной $80n$;
- `end` — признак окончания работы типа `bool`.

Критерии оценивания

Итоговый балл вычисляется как максимальный балл за задачу, умноженный на отношение корректно принятых сообщений к общему числу отправленных сообщений.

Сообщение является корректно принятым, если совпадает с любым сообщением, которое было подано в функцию программы передатчика, кроме тех, что уже были корректно приняты.

Первые четыре попытки без штрафа. С каждой следующей попыткой балл убывает на 10% от максимального балла (на 1 балл) до 1 балла, далее не убывает.

Решение

В решении задачи применяется opensource библиотека `reedsolo`, которая реализует код Рида-Соломона. Альтернативно, участники могли использовать код Хэмминга либо также в виде библиотеки, либо самостоятельно реализованный. Для применения библиотеки ее исходный код необходимо скопировать в начало программы.

Программа спутникового передатчика

Ниже представлено решение на языке Python.

Python

```python
#Содержимое https://github.com/tomerfiliba-org/reedsolomon/blob/master/src/reedsolo/reedsolo.py

def proceed(msg):
    n = len(msg)
    l = 1.5*n
    l -= 0.05*l
    l = int(l)
    print("n: ", n)
    print("l: ", l)
    codec = RSCodec(l - n - 4)
    print("msg:", msg)
    ret = bytes([3, 3, l-4, l-4]) + codec.encode(bytes(msg, "ASCII"))
    print("ret:", ret)
    return ret
```

Программа наземного приемника

Ниже представлено решение на языке Python.

Python

```python
#Содержимое https://github.com/tomerfiliba-org/reedsolomon/blob/master/src/reedsolo/reedsolo.py


preambul = np.array([0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1], dtype=bool)


def check(len_chunk):
    arr = np.packbits(len_chunk)
    if arr[0] == arr[1]:
        return int(arr[0])
    return 0


def getMessage(chunk, c):
    rsc = RSCodec(c)
    arr = bytes(np.packbits(chunk))
    try:
        return rsc.decode(arr)[0]
    except:
        return None


def proceed(inp, buf, end):
```

```
    li_orig = len(inp)
    n = int(len(buf) / 10 / 8)
    print("n recv:", n, "len buf", len(buf))
    l = n * 1.5
    l -= 0.05*l
    l = int(l)
    print("l recv:", l)
    inp = np.concatenate([buf[-li_orig:], inp])

    i = 16
    li = len(inp)
    while i < li:
        if np.count_nonzero(inp[i-16:i] != preambul) > 2:
            inp[i-16] = 0
            i += 1
            continue
        if li - i < 16:
            break
        c = check(inp[i:i+16])
        if c == 0:
            inp[i-16] = 0
            i += 1
            continue
        if li - i < c * 8 + 16:
            break
        inp[i:i+16] = np.zeros(16)
        i += 16
        msg = getMessage(inp[i:i+c*8], l - n - 4)
        inp[i:i + c*8] = np.zeros(c*8)
        i += c*8
        if not msg:
            continue
        tl = max(-(li-i), -li_orig)
        if tl != 0:
            buf[tl:] = inp[tl:]
        return str(msg, "ASCII")
    buf[-li_orig:] = inp[-li_orig:]
    return ""
```

Задача 3.2.1.3. Задача по баллистике (10 баллов)

Тема: орбитальная механика.

Условие

Задача — обеспечить обзорную съемку с помощью малых космических аппаратов с камерой низкого разрешения части сибирской тайги с целью наблюдения за индексом вегетации. Используя два космических аппарата, обеспечьте полное покрытие указанной зоны съемкой аппаратами с мультиспектральными камерами в ИК-диапазоне с ПР (пространственным разрешением) не более 30 м. Такие условия можно обеспечить, если высота аппарата над зоной съемки не будет превышать 900 км и не будет находиться ниже 300 км. Ширина полосы съемки 30 км. Зона съемки: область, ограниченная 59° и 61° северной широты, 93° и 95° восточной долготы.

Определение узлов зоны

Зона разбивается по широте с севера на юг на полосы шириной 4 км, остаток отбрасывается. Дальше каждая полоса разбивается по долготе с запада на восток на ячейки 4×4 км, остаток отбрасывается. Назовем узлом центр каждой такой ячейки. Узел считается снятым, если след аппарата в надире прошел от него не далее, чем на расстоянии половины ширины полосы съемки. «Вес» узла $= 1/N$, где N — количество узлов в зоне интереса.

Старт работы КА на орбите начинается 01 июля 2025 года с 00:00 по UTC. Длительность симуляции — 24 часа.

Начальные данные:

- Радиус Земли, м: 6 371 008,8.
- Гравитационный параметр, $\mu = G \cdot M$, м^3/с^2: $3{,}986\,004\,418 \cdot 10^{14}$.
- Влияние атмосферы отключено. Прецессия орбиты отключена.

Критерии оценивания

Начисление баллов:

- Начисляется 10/`grid.length` балла за прохождение над непройденным ранее узлом, где `grid.length` — количество узлов в зоне съемки.
- При снижении аппарата на высоту 300 км и менее баллы перестают начисляться.
- При подъеме аппарата на высоту 900 км и более баллы перестают начисляться.

Максимально возможный балл — 10. Первые четыре попытки идут без штрафа. За каждую следующую попытку максимально возможный балл уменьшается на 10% (т. е. на пятой попытке до 9 баллов, на шестой — до 8 и т. д.) до 1 балла и далее не уменьшается.

Решение

Первый аппарат:

- наклонение 90°;
- большая полуось 6775 км;
- эксцентриситет 0;
- долгота восходящего узла 201,6°;
- аргумент перицентра 0°;
- истинная аномалия 0°.

Второй аппарат:

- наклонение 90°;
- большая полуось 6775 км;
- эксцентриситет 0;
- долгота восходящего узла 200,6°;
- аргумент перицентра 0°;
- истинная аномалия 0°.

Задача 3.2.1.4. Задача по программированию микроконтроллеров (10 баллов)

Тема: программирование.

Условие

Перед тем как получить данные температуры с датчика, надо сначала изучить его документацию, узнать, какой интерфейс он использует, и какие данные можно получить.

Часть 1. Тест

Космический аппарат (КА) использует для измерения температуры датчик AMG8833. Нужно понять, по какому интерфейсу он общается с КА:

- Какие данные и по какому адресу необходимо записать в регистр управления питанием датчика, чтобы датчик был в режиме работы (`normal`)?
- Какие данные и по какому адресу необходимо записать в регистр сброса датчика, чтобы датчик перешел в первоначальное состояние?
- Какие данные и по какому адресу необходимо записать в регистр скорости работы датчика, чтобы датчик обновлял матрицу раз в секунду?
- С какого по какого адреса в памяти датчика надо считывать данные регистров, чтобы получить температуру с матрицы?

Часть 2. Программа

Как преобразовать данные для 1 пикселя, зная младшую и старшую часть значений пикселя?

Написать программу, которая преобразует младшую и старшую часть числа в температуру в градусах Цельсия, и при выходе за диапазон измерений надо возвращать `NaN`. В старшей части необходимо брать не больше 4 младших бит данных, а в младшей — не больше 8 младших бит данных, остальное необходимо обрезать и игнорировать.

Дополнительная информация:

- https://industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ADI8000C66.pdf;
- https://www.robot-electronics.co.uk/files/grideyeappnote.pdf.

Формат входных данных

Входные данные для программы — два целых положительных числа от 0 до 512, разделенные пробелом. Первое — старшая часть, второе — младшая часть.

Примеры:

- 20 410;
- 0 46;
- 120 0.

Формат выходных данных

Выходные данные — дробное число с точностью два знака после точки или `NaN`.

Примеры:

- 10.25;
- 30.00;
- -20.50;
- `NaN`.

Критерии оценивания

- Адреса регистров должны быть указаны в 16-ричном формате, начиная с `0x`.
- 16-ричное число должно быть двухзначным. Например: `0xFF, 0x33, 0x6C`.
- Значения регистров должны быть указаны в 16-ричном формате, начиная с `0x`.
- 16-ричное число должно быть двухзначным. Например: `0xFF, 0x33, 0x6C`.

Правила начисления баллов за тест

- Баллы за правильный интерфейс — 1.
- Баллы за правильный адрес и значения регистра управления питанием — 1.
- Баллы за правильный адрес и значения регистра сброса — 1.
- Баллы за правильный адрес и значения регистра скорости работы — 1.
- Баллы за правильный первый и последний адрес регистров для считывания температуры матрицы с пикселей — 1.
- Баллы за преобразование данных начисляются в процентном соотношении пройденных тестов.

Всего баллов за тест — 5.

Правила начисления баллов за программу

- Полный балл начисляется при прохождении 100% тестов.
- При прохождении меньше чем 100% теста, баллы начисляются как пятикратный процент пройденных тестов.

Всего баллов за программу — 5.

Всего баллов за задачу — 10.

Решение

Интерфейс: SPI, I2C, UART, OneWire.

Адрес регистра управления питанием: `0x00`.

Значение регистра управления питанием: `0x00`.

Адрес регистра сброса: `0x01`.

Значение регистра сброса: `0x3F`.

Адрес регистра скорости работы: `0x02`.

Значение регистра скорости работы: `0x01`.

Первый адрес регистра для считывания температуры с матрицы пикселей: `0x80`.

Последний адрес регистра для считывания температуры матрицы с пикселей: `0xFF`.

Формула преобразования (Python):

Python

```python
def convert(high: int, low: int) -> float:
    pixel: int = (high & 0x0F) << 8 | low
    temp: float
    if pixel & 0x800 > 0:
        temp = ((~pixel & 0x7FF) + 1) * -0.25
    else:
        temp = pixel * 0.25
    if temp <= 80 and temp >= -0:
        return temp
    return float('NaN')

```

3.2.2. Командные задачи

Задача 3.2.2.1. Первая командная задача (25 баллов)

Темы: программирование, радиосвязь, орбитальная механика.

Условие

В рамках испытаний модуля приема-передачи данных для космических аппаратов две группы исследователей решили организовать спутниковую связь между научными центрами, расположенными в Ла-Кьяке $(-22{,}102\,3, 294{,}407\,01)$ и Гонконге $(22{,}285\,52, 114{,}157\,69)$. При этом Ла-Кьяка работает только на передачу, а Гонконг — только на прием.

Известно, что передаваемые данные являются случайным массивом байт типа `Uint8Array` размером 20 элементов. Также известно, что в процессе передачи сообщения между станцией в Ла-Кьяка и спутником пакет может быть поврежден сторонними помехами случайным образом. Приемопередающая аппаратура спутника не может функционировать нормально, если угловая скорость по оси X превышает $0{,}000\,3$ рад/с.

Для корректной работы модуля приема-передачи данных необходимо свести начальную закрутку спутника до нуля, используя установленные на нем маховики.

Для успешного решения задачи необходимо:

- определить орбиту спутника связи (высота аппарата относительно центра Земли не может превышать 15 000 км);
- стабилизировать его после начала работы на орбите (свести начальную угловую скорость по оси X к нулю);
- реализовать алгоритм работы спутника ретранслятора;
- написать программу для наземной станции в Ла-Кьяке по передаче сообщений на спутник;
- совершить успешную передачу 50 000 пакетов.

Время начала моделирования: 1 декабря 2025 года 00:00 по UTC. Длительность моделирования: 24 ч.

Примечания

- Модуль ориентации выдает скорость вращения спутника в градусах в секунду в двух байтах информации.

Таблица 3.2.3

Знак	q_{10}	q_9	q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	q_0	q_{-1}	q_{-2}	q_{-3}	q_{-4}
Знак	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

  Первый байт — старшая часть, второй байт — младшая часть, данные записываются в дополнительном коде.
- Первые 320 с после вывода на орбиту отводятся на переходный процесс, в это время передача данных не происходит.
- На вход в программу для передающей наземной станции подается массив байт типа `Uint8Array`, состоящий из 20 байт. Вывод программы передающей наземной станции направляется на передатчик. Спутник может принимать данные с помощью установленного на нем приемника. Во время работы спутник может отправить данные в виде массива байт типа `Uint8Array` на передатчик. Эти данные принимает наземная станция в Гонконге.
- Вероятность возникновения ошибки в пакете: 50%.
- Область памяти спутника, отведенная для хранения сообщений, ограничена 1 Мб.
- Область памяти наземных станций, отведенная для хранения данных, ограничена 1 Мб.
- Сообщение считается успешно принятым, только если оно полностью совпадает с сообщением, введенным в программу наземного передатчика.
- Все передатчики принимают данные на вход в виде массива байт типа `Uint8Array`.
- Скорость передачи между спутником и передающей станцией — 36 байт в 0,01 с.
- Скорость передачи на наземную станцию приема со спутника — 20 байт в 0,01 с.

Ссылка на API: https://disk.yandex.ru/i/lK8h9GH7OXc2bw.

Начальные данные:

- Координаты Ла-Кьяки: $-22{,}102\,3, 294{,}407\,01$.
- Координаты Гонконга: $22{,}285\,52, 114{,}157\,69$.
- Скорость передачи между спутником и наземными станциями: 20 байт в 0,1 с.
- Радиус Земли, м: 6 371 008,8.
- Гравитационный параметр, $\mu = G \cdot M$, м^3/с^3: $3{,}986004418 \cdot 10^{14}$.
- Спутник считается прошедшим над указанной точкой, если он был виден из указанной точки в 60° над горизонтом или выше.

Критерии оценивания

Правила начисления баллов:

- за каждый пакет без ошибок, переданный на наземную станцию, начисляется 0,000 5 балла;
- за каждый пакет, содержащий ошибки, снимается 0,000 5 балла, если он передан на наземную станцию.

Максимальный балл за задачу: 25.

Правила дисконтирования:

- Количество решений без штрафа — 3 на участника команды.
- За каждое последующее решение будет накладываться штраф 10% до убывания в 20%.
- Максимальная разница попыток между участниками команды — 3.
- При наличии ошибки в синтаксисе попытка не считается потраченной.

Решение

Орбита аппарата:

- наклонение: 22,28°;
- большая полуось: 14 990 км;
- эксцентриситет: 0;
- долгота восходящего узла: 0°;
- аргумент перицентра: 0°;
- истинная аномалия: 0°.

Программа управления спутником

JavaScript

```
let gyroGetter;
let transmitter;
let receiver;

var received_packet = Array.from([]);

function crc16_ccitt_false(data) {
    let crc = 0xFFFF;
```

```
    for (let i = 0; i < data.length; i++) {
        crc ^= data[i] << 8;
        for (let j = 0; j < 8; j++) {
            if (crc & 0x8000) {
                crc = (crc << 1) ^ 0x1021;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc & 0xFFFF;
}

function check_packet(data) {
    let crc = (data[data.length - 2] << 8) | data[data.length - 1];
    let calculated_crc = crc16_ccitt_false(data.slice(0, -2));
    return crc === calculated_crc;
}

function setup() {
    gyroGetter = createAngularVelocityGetter(5);
    transmitter = spacecraft.devices[0].functions[0];
    receiver = spacecraft.devices[1].functions[0];
}
function loop() {
    const motor = spacecraft.devices[4].functions;
    const velocity = gyroGetter(spacecraft);
    const velocityX = velocity[0];
    motor[0].motor_torque += velocityX;

    if (spacecraft.flight_time > 46350 && spacecraft.flight_time <
    ↪  47949) {
        let r = Array.from(receiver.receive(22));
        if (r.length > 0){
            if (check_packet(r)) {
                received_packet.push(...r);
            }
        }
    }
    if (spacecraft.flight_time > 57240) {
        if (received_packet.length > 0) {
            let received_packet_n = received_packet.slice(0, 20);
            received_packet.splice(0,22);
            transmitter.transmit(new Uint8Array(received_packet_n));
        }
    }
}

function createAngularVelocityGetter(gyroId)
{

    let lastResults = [0, 0, 0];

    const convertFunc = (array) => {
        if (array.length !== 2) { return undefined; }
        const preResult = array[0] << 8 | array[1];
        const result = preResult < 0x8000 ? preResult : -0x8000 +
        ↪  (preResult & 0x7FFF);
        return result / 16;
    };
```

```
    return (spacecraft) => {
        const gyro = spacecraft.devices[gyroId];
        const functions = gyro.functions;
        for (let i = 0; i < 3; i++)
        {
            const converted = convertFunc(functions[i].read(2));
            if (converted !== undefined) { lastResults[i] = converted;
             ↪  }
        }
        return lastResults;
    };
}
```

Программа передающей станции

JavaScript

```
let transmitter;
let receiver;

var crcTable = [0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5,
0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b,
0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210,
0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c,
0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401,
0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a, 0xb54b,
0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6,
0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,
0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5,
0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969,
0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96,
0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc,
0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03,
0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd,
0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97, 0x6eb6,
0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a,
0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca, 0xa1eb,
0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1,
0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c,
0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2,
0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea, 0xa5cb,
0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447,
0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,
0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2,
0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9,
0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806, 0x6827,
0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c,
0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0,
0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d,
0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26, 0x6c07,
0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
```

```
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba,
0x8fd9, 0x9ff8, 0x6e17, 0x7e36, 0x4e55, 0x5e74,
0x2e93, 0x3eb2, 0x0ed1, 0x1ef0];


function crc16(s) {
    var crc = 0xFFFF;
    var j, i;


    for (i = 0; i < s.length; i++) {

        if (s[i] > 255) {
            throw new RangeError();
        }
        j = (s[i] ^ (crc >> 8)) & 0xFF;
        crc = crcTable[j] ^ (crc << 8);
    }

    let crcNum =  ((crc ^ 0) & 0xFFFF);
    return (new Uint8Array((new
     ↪  Uint16Array([crcNum])).buffer)).reverse();

}

function setup() {
    transmitter = spacecraft.devices[0].functions[0];
    receiver = spacecraft.devices[1].functions[0];
}
function loop() {
    if (spacecraft.flight_time > 46350 && spacecraft.flight_time <
     ↪  47949) {
        let rx = receiver.receive(20);
        let crcpart = crc16(rx);
        let packet = new Uint8Array([...rx, ...crcpart]);
        if(packet.length === 22){
            transmitter.transmit(packet);
        }
    }
}
```

Задача 3.2.2.2. Вторая командная задача (35 баллов)

Темы: программирование, радиосвязь, орбитальная механика.

Условие

Целью группировки, состоящей из двух спутников, является сбор научных данных и их отправка на Землю. Один из спутников оснащен полезной нагрузкой, а второй — более развитой радиопередающей аппаратурой. Спутник, оснащенный мощной передающей аппаратурой, принадлежит компании товарища Каса и уже выведен на орбиту. Миссия научного спутника должна быть долгосрочной, а орбитальная скорость должна быть достаточно низкой для получения четких снимков Солнца, поэтому размещение его на низкой околоземной орбите исключено. Исходя из этого, для орбиты научного спутника выдвинуты следующие ограничения:

- снимок Солнца производится на высоте не менее 2 000 км от поверхности Земли;
- научный спутник может передавать данные на расстояние, не превышающее 1 500 км.

Орбита спутника-ретранслятора:

- большая полуось: 6771 км;
- эксцентриситет: 0;
- наклонение: 61,3°;
- ДВУ: 48°;
- аргумент перицентра: 0°;
- истинная аномалия на начало симуляции: 0°.

Для сбора научных данных используется специальная камера со светофильтром, которая может делать снимки Солнца. На одной грани вместе с камерой расположен матричный датчик Солнца, который работает постоянно и обновляет данные раз в 0,1 с.

Необходимо, анализируя данные датчика Солнца, определять, находится ли Солнце в поле видимости камеры или нет. Когда Солнце находится в поле видимости камеры, с нее необходимо запрашивать снимок, обрабатывать его и передавать по радиоканалу.

Требования к передаче данных:

- данные можно отправлять на наземные станции, расположенные в городах Анкоридж (61,2181, −149,9), Мельбурн (−37,814, 144,963) и Лиссабон (38,716 7, −9,133 33);
- скорость передачи между спутниками и между спутником-ретранслятором и наземной станцией составляет 100 байт/с.

Время выполнения итерации функции `loop` соответствует 0,1 с симуляции.

Справка по радиосвязи

- Данные на передатчик всегда отправляются в байтах.
- Данные на приемник всегда передаются в битах.
- Все приемники и передатчики синхронизированы по всем параметрам, кроме оговоренных далее в условии.
- Приемники и передатчики могут быть рассинхронизированы по времени между отправкой и получением бита данных. Максимальная рассинхронизация может составить до 7 бит включительно.
- Данные между спутниками передаются без помех.
- Данные между спутником-ретранслятором и наземной станцией подвержены помехам.

Справка по датчику Солнца

Строение датчика соответствует датчику в индивидуальной задаче программиста ПН. А именно, датчик возвращает координаты пикселя матрицы, на который попало больше всего света.

- Размер матрицы: $22{,}3 \times 14{,}9$ мм.
- Размер пикселя: 0,025 мм.

На расстоянии 5 мм от матрицы установлена металлическая пластина с небольшим отверстием точно по центру.

В качестве данных датчик возвращает два числа $[x; y]$ — в системе координат матрицы. Если все пиксели освещены одинаково (Солнце не в поле зрения), датчик вернет $[-1; -1]$.

Координаты пикселей считать от левого верхнего края: ось Y — длинная сторона матрицы (направлена вправо из левого верхнего края матрицы); ось X — короткая (направлена вниз из левого верхнего края матрицы).

Ноль системы координат датчика считать центром металлической пластины, оси X и Y сонаправлены с осями координат матрицы, ось Z дополняет систему до правой тройки. При расчете считать, что свет попадает точно в центр пикселя.

Справка по камере научного спутника

Угол обзора камеры — 30°, смещением датчика Солнца относительно оси камеры пренебречь.

Камера выдает сырые данные, представленные массивом из 1 600 байт (каждый байт соответствует пикселю матрицы 40×40). Чтобы получить сжатый снимок, необходимо произвести биннинг пикселей — разбить матрицу на квадраты, содержащие по 4 пикселя, и объединить пиксели внутри каждого квадрата в один, присвоив ему среднее арифметическое значение. На выходе получится массив из 400 байт, где каждый байт — среднее арифметическое по соответствующему квадрату.

Формат передачи снимка с камеры представлен следующей структурой: `array_data[1600]`.

Данные представлены типом `Uint8Array`.

Формат входных данных

- Данные датчика Солнца генерируются по запросу.
- Снимок с камеры научного спутника генерируется по запросу.
- Время начала моделирования: 1 декабря 2025 года 00:00 по UTC.
- Длительность моделирования: 24 ч.
- Спутник считается прошедшим над указанной точкой, если он был виден из указанной точки в 10° над горизонтом или выше.

Ссылка на API: https://disk.yandex.ru/i/muLu7QyEAd7uLQ.

Критерии оценивания

Правила начисления баллов:

- За каждую корректно обработанную фотографию и рассчитанный для нее угол (в градусах) отклонения Солнца от главной оптической оси датчика начисляется 0,0036 балла, если они помещены в хранилище на научном спутнике.
- Максимум за обработку показаний датчика Солнца и изображений с камеры начисляется 15 баллов.
- За каждый правильно переданный на наземную станцию пакет начисляется 0,096 балла. Для начисления баллов пакеты необходимо передавать по одному

в исходном виде (сжатый снимок с камеры в формате `Uint8Array[400]`).

- Максимум за передачу пакетов начисляется 20 баллов.

Максимальный балл за задачу: 35.

Правила дисконтирования:

- Количество решений без штрафа — 3 на участника команды.
- За каждое последующее решение будет накладываться штраф 10% до убывания в 20%.
- Максимальная разница попыток между участниками команды — 3.
- При наличии ошибки в синтаксисе попытка не считается потраченной.

Решение

Орбита аппарата:

- наклонение: 61,3°;
- большая полуось: 10 000 км;
- эксцентриситет: 0,33;
- долгота восходящего узла: 48°;
- аргумент перицентра: 0°;
- истинная аномалия: 0°.

Программа управления спутником ретранслятором

JavaScript

```
let transmitter;
let receiver;

function setup() {
    transmitter = spacecraft.devices[0].functions[0];
    receiver = spacecraft.devices[1].functions[0];
}

function loop() {
    const bits = receiver.receive(80);
    let bytes = new Uint8Array(10);
    for (let i = 0; i < 10; i++) {
        bytes[i] = parseInt(Array.from(bits.slice(i*8,
        ↪  (i+1)*8)).map(el => el ? '1' : '0').join(''), 2);
    }
    transmitter.transmit(bytes);
}
```

Программа управления научным спутником

JavaScript

```
const setConfig = ({ pow } = { pow: 4 }) => {
  if (isNaN(pow)) {
    throw new Error('please pass number');
  }
  if (pow < 4) {
    throw new Error(`argument need > 4, now is ${pow}`);
```

```
  }
  const BlockLength = Math.pow(2, pow);
  const ContainerLength = BlockLength - pow - 1;
  const position = [0];
  for (let i = 0; i < pow; i++) {
    position.push(1 << i);
  }

  const xor = (array) => (array.length ? array.reduce((one, two) =>
  ↪  one ^ two) : 0);
  const handlePosition = (index) => position.includes(index);

  const hammingList = (str) => {
    if (str.length > ContainerLength) {
      const message = `${str} length is too long, max
      ↪  ${ContainerLength}`;
      throw new Error(message);
    }
    const hammingBlock = [];
    const trueIndexList = [];
    const strLength = str.length;
    for (let i = 0, j = 0; j < strLength; i++) {
      if (handlePosition(i)) {
        hammingBlock[i] = 0;
      } else {
        if (str[j] && +str[j] !== 0) {
          trueIndexList.push(i);
        }
        hammingBlock[i] = str[j] || 0;
        j++;
      }
    }

    const xorData = `${xor(trueIndexList).toString(2)}`.padStart(pow,
    ↪  0);

    for (let i = 0, xorLength = xorData.length; i < xorLength; i++) {
      const position = Math.pow(2, i);
      if (+xorData[xorLength - i - 1]) {
        hammingBlock[position] = xorData[xorLength - i - 1];
      }
    }
    hammingBlock[0] = xor(hammingBlock);
    return hammingBlock;
  };

  const decodeHammingList = (str) => {
    const numList = str.split('');
    const totalCorrect = !xor(numList);

    const trueList = [];
    for (let i = 1; i < numList.length; i++) {
      if (numList[i] && +numList[i]) {
        trueList.push(i);
      }
    }

    const xorData = xor(trueList);

    let correct = false;
```

```
    if (xorData === 0 && totalCorrect) {
      correct = true;
    }
    if (xorData !== 0 && totalCorrect) {
      correct = false;
    }
    if (xorData !== 0 && !totalCorrect) {
      correct = true;
      numList[xorData] = Number(!+numList[xorData]);
    }
    const code = [];
    for (let i = 0; i < numList.length; i++) {
      if (!handlePosition(i)) {
        code.push(numList[i]);
      }
    }
    return {
      correct,
      code: code.join(''),
      originCode: str,
    };
  };

  const sliceContainer = (data, lengh) => {
    const result = [];
    for (let i = 0, len = data.length; i < len; i += lengh) {
      result.push(data.slice(i, i + lengh));
    }
    return result;
  };

  const encode = (str) => {
    const sliceArray = sliceContainer(str, ContainerLength);
    return sliceArray
      .map((item) => hammingList(item))
      .flat()
      .join('');
  };

  const decode = (str) => {
    const sliceArray = sliceContainer(str, BlockLength);
    const decodeResult = sliceArray.map((item) =>
    ↪  decodeHammingList(item));
    const res = decodeResult.reduce((one, two) => {
      return {
        correct: one.correct && two.correct,
        code: one.code + two.code,
        originCode: one.originCode + two.originCode,
      };
    });
    return {
      ...res,
      decodeResult,
    };
  };

  return {
    encode,
    decode,
  };
```

```
}


const hamming = setConfig();


function bytes2bits(bytes) {
    function bin(num) {
        let payload = num.toString(2);
        return "0".repeat(8 - payload.length) + payload;
    }
    return Array.from(bytes).map(bin).join("");
}

function bits2bytes(bits) {
    let aligned = bits + "0".repeat(bits.length % 8);
    let rezBuf = new Uint8Array(aligned.length / 8);

    for (let i = 0; i < rezBuf.length; i++) {
      rezBuf[i] = parseInt(aligned.slice(i*8, (i+1)*8), 2);
    }
    return rezBuf;
}

function proceed_message(msg) {
    const bits_msg = bytes2bits(msg);
    const encoded = hamming.encode(bits_msg);
    const encoded_bytes = bits2bytes(encoded);
    return encoded_bytes;
}

function decode_message(msg, bitslen) {
    const bits_msg = bytes2bits(msg).slice(0, bitslen);
    const decoded = hamming.decode(bits_msg);
    const decoded_bytes = bits2bytes(decoded.code);
    return decoded_bytes;
}

function encode(str) {
    return new Uint8Array(Array.from(str).map(el =>
    ↪  el.charCodeAt(0)));
}

let camera;
let sun_sensor;
let storage;
let sun_angle;
let transmitter;
let receiver;

let pixelX;
let pixelY;

function uint8ArrayToNums(arr) {
    pixelX = 0;
    pixelY = 0;
    for (let i = 7; i >= 0; i--) {
      pixelX = pixelX * 256 + arr[i];
      pixelY = pixelY * 256 + arr[i+8];
    }
```

```
}

function setup() {
    transmitter = spacecraft.devices[0].functions[0];
    receiver = spacecraft.devices[1].functions[0];
    camera = spacecraft.devices[2].functions[0];
    sun_sensor = spacecraft.devices[3].functions[0];
    storage = spacecraft.devices[4].functions[0];
}
function loop() {
    let pixels = sun_sensor.read(16);
    let image;
    let clear_image;
    uint8ArrayToNums(pixels);
    if(calculateSunDirection(pixelX, pixelY)){
        image = camera.read(1600);
        clear_image = removeNoise(image);
        storage.write(clear_image);
        let float = new Uint8Array(new
        ↪  Float32Array([sun_angle]).buffer);
        storage.write(float);
        const payload = proceed_message(clear_image);
        let msg = new Uint8Array(2 + payload.length);
        msg[0] = 3;
        msg[1] = 3;
        msg.set(payload, 2);
        transmitter.transmit(msg);
    }
}

function calculateSunDirection(pixelX, pixelY) {
    if( pixelX < 0 || pixelY < 0 || pixelX > 596 || pixelY > 892){
        return false;
    }
    const pixelSize = 0.025;
    const zDistance = -5;

    const xMid = (14.9 / 0.025) / 2;
    const yMid = (22.3 / 0.025) / 2;

    const deltaX = (pixelX - 0.5 - xMid) * pixelSize;
    const deltaY = (pixelY - 0.5 - yMid) * pixelSize;

    const vector = {
        x: -deltaX,
        y: -deltaY,
        z: -zDistance
    };
    const vectorLength = Math.sqrt(vector.x ** 2 + vector.y ** 2 +
    ↪  vector.z ** 2);

    const unitVector = {
        x: vector.x / vectorLength,
        y: vector.y / vectorLength,
        z: vector.z / vectorLength
    };
    const referenceVector = {
        x: 0,
        y: 0,
        z: 1
```

```
    };

    const dotProduct = unitVector.x * referenceVector.x + unitVector.y
    ↪  * referenceVector.y + unitVector.z * referenceVector.z;
    sun_angle = Math.acos(dotProduct) * (180 / Math.PI);

    if (sun_angle > 30) {
        return false;
    } else {
        return true;
    }
}

function removeNoise(array){
    const width = 20;
    const heigth = 20;

    const noisedWidthPixelOnReal = 2;
    const noisedHeigthPixelOnReal = 2;
    const noisedWidth = width * noisedWidthPixelOnReal;
    const noisedHeigth = heigth * noisedHeigthPixelOnReal;

    const realImageSize = width * heigth;

    const result = new Uint8Array(realImageSize);

    for (let i = 0; i < realImageSize; i++)
    {
        const row = i % width;
        const column = Math.floor(i / width);

        const newRow = row * noisedWidthPixelOnReal;
        const newColumn = column * noisedHeigthPixelOnReal;

        const pixels = [array[newRow + (newColumn + 0) * noisedWidth +
        ↪  0],
                        array[newRow + (newColumn + 0) * noisedWidth +
                        ↪  1],
                        array[newRow + (newColumn + 1) * noisedWidth +
                        ↪  0],
                        array[newRow + (newColumn + 1) * noisedWidth +
                        ↪  1]];
        result[i] = pixels.reduce((partialSum, a) => partialSum + a,
        ↪  0) / 4;
    }
    return result;
}
```

Программа принимающей станции

JavaScript

```
const setConfig = ({ pow } = { pow: 4 }) => {
  if (isNaN(pow)) {
    throw new Error('please pass number');
  }
  if (pow < 4) {
    throw new Error(`argument need > 4, now is ${pow}`);
  }
  const BlockLength = Math.pow(2, pow);
```

```
  const ContainerLength = BlockLength - pow - 1;
  const position = [0];
  for (let i = 0; i < pow; i++) {
    position.push(1 << i);
  }

  const xor = (array) => (array.length ? array.reduce((one, two) =>
  ↪  one ^ two) : 0);
  const handlePosition = (index) => position.includes(index);

  const hammingList = (str) => {
    if (str.length > ContainerLength) {
      const message = `${str} length is too long, max
      ↪  ${ContainerLength}`;
      throw new Error(message);
    }
    const hammingBlock = [];
    const trueIndexList = [];
    const strLength = str.length;
    for (let i = 0, j = 0; j < strLength; i++) {
      if (handlePosition(i)) {
        hammingBlock[i] = 0;
      } else {
        if (str[j] && +str[j] !== 0) {
          trueIndexList.push(i);
        }
        hammingBlock[i] = str[j] || 0;
        j++;
      }
    }

    const xorData = `${xor(trueIndexList).toString(2)}`.padStart(pow,
    ↪  0);

    for (let i = 0, xorLength = xorData.length; i < xorLength; i++) {
      const position = Math.pow(2, i);
      if (+xorData[xorLength - i - 1]) {
        hammingBlock[position] = xorData[xorLength - i - 1];
      }
    }
    hammingBlock[0] = xor(hammingBlock);
    return hammingBlock;
  };

  const decodeHammingList = (str) => {
    const numList = str.split('');
    const totalCorrect = !xor(numList);

    const trueList = [];
    for (let i = 1; i < numList.length; i++) {
      if (numList[i] && +numList[i]) {
        trueList.push(i);
      }
    }

    const xorData = xor(trueList);

    let correct = false;
    if (xorData === 0 && totalCorrect) {
      correct = true;
```

```
    }
    if (xorData !== 0 && totalCorrect) {
      correct = false;
    }
    if (xorData !== 0 && !totalCorrect) {
      correct = true;
      numList[xorData] = Number(!+numList[xorData]);
    }
    const code = [];
    for (let i = 0; i < numList.length; i++) {
      if (!handlePosition(i)) {
        code.push(numList[i]);
      }
    }
    return {
      correct,
      code: code.join(''),
      originCode: str,
    };
  };

  const sliceContainer = (data, lengh) => {
    const result = [];
    for (let i = 0, len = data.length; i < len; i += lengh) {
      result.push(data.slice(i, i + lengh));
    }
    return result;
  };

  const encode = (str) => {
    const sliceArray = sliceContainer(str, ContainerLength);
    return sliceArray
      .map((item) => hammingList(item))
      .flat()
      .join('');
  };

  const decode = (str) => {
    const sliceArray = sliceContainer(str, BlockLength);
    const decodeResult = sliceArray.map((item) =>
    ↪  decodeHammingList(item));
    const res = decodeResult.reduce((one, two) => {
      return {
        correct: one.correct && two.correct,
        code: one.code + two.code,
        originCode: one.originCode + two.originCode,
      };
    });
    return {
      ...res,
      decodeResult,
    };
  };

  return {
    encode,
    decode,
  };
};

```

```
125
126 const hamming = setConfig();
127
128
129 function bytes2bits(bytes) {
130     function bin(num) {
131         let payload = num.toString(2);
132         return "0".repeat(8 - payload.length) + payload;
133     }
134     return Array.from(bytes).map(bin).join("");
135 }
136
137 function bits2bytes(bits) {
138     let aligned = bits + "0".repeat(bits.length % 8);
139     let rezBuf = new Uint8Array(aligned.length / 8);
140
141     for (let i = 0; i < rezBuf.length; i++) {
142       rezBuf[i] = parseInt(aligned.slice(i*8, (i+1)*8), 2);
143     }
144     return rezBuf;
145 }
146
147 function decode_message(msg, bitslen) {
148     const bits_msg = bytes2bits(msg).slice(0, bitslen);
149     const decoded = hamming.decode(bits_msg);
150     const decoded_bytes = bits2bytes(decoded.code);
151     return decoded_bytes;
152 }
153
154 var receiver;
155 var consumer;
156
157 let buffer = new Uint8Array(80+16);
158 let offset = -1;
159
160 function setup() {
161     receiver = spacecraft.devices[0].functions[0];
162     consumer = spacecraft.devices[1].functions[0];
163 }
164
165 const preambul = new Uint8Array([0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1]);
166 const bytelen = 582;
167 const bitslen = 4655;
168 let debugi = 0;
169
170 function capture(off) {
171     let index = 0;
172     let buff = new Uint8Array(bytelen);
173     for (let i = off; i < 80; i+=8) {
174         const bytearr = buffer.slice(i, i+8);
175         buff[index] = parseInt(Array.from(bytearr).map(el => el ? '1' :
            ↪  '0').join(""), 2);
176         index++;
177     }
178     return () => {
179         for (let i = offset; i < 80; i+=8) {
180             const bytearr = buffer.slice(i, i+8);
181             buff[index] = parseInt(Array.from(bytearr).map(el => el ?
                ↪  '1' : '0').join(""), 2);
182             index++;
```

```
            if (index === bytelen) {
                proceed = findPreambul;
                const msg = decode_message(buff, bitslen);
                consumer.transmit(msg);
                break;
            }
        }
    };
}

function check(input) {
    let rez = 0;
    for (let i = 0; i < 16; i++) {
        if (input[i] !== preambul[i]) {
            rez++;
        }
    }
    return rez;
}

function findPreambul() {
    let minrez = [-1,0];
    for (let off = 0; off < 80; off++) {
        let rez = check(buffer.slice(off,16+off));
        if (rez <= minrez[1]) {
            minrez = [off, rez];
        }
    }
    if (minrez[0] !== -1) {
        offset = minrez[0] % 8;
        proceed = capture(minrez[0] + 16);
    }
}

let proceed = findPreambul;

function loop() {
    const bits = receiver.receive(80);
    buffer.set(buffer.slice(-16));
    buffer.set(bits, 16);
    proceed();
    debugi++;
}
```

4. Заключительный этап

4.1. Работа наставника НТО при подготовке к этапу

На этапе подготовки к заключительному этапу НТО наставник решает две важные задачи: помощь участникам в подготовке к предстоящим соревнованиям и формирование устойчивой и слаженной команды. Заключительный этап требует высокой слаженности, уверенности и глубоких знаний, и наставник становится тем, кто объединяет усилия участников и направляет их в нужное русло.

Наставник помогает участникам:

- разобрать задания прошлых лет, используя официальные сборники, чтобы понять структуру финальных испытаний, типы задач и ожидаемый уровень сложности;
- изучить организационные особенности заключительного этапа, включая формат проведения, регламент, продолжительность и технические нюансы;
- спланировать подготовку — на основе даты начала финала составляется четкий график занятий, в котором распределены темы, практикумы и командные тренировки;
- обратиться (при необходимости) за консультацией к разработчикам заданий по профилю, уточнить, на какие аспекты подготовки следует обратить особое внимание, и получить дополнительные материалы.

Также рекомендуется участие в мероприятиях от организаторов, таких как:

- установочные вебинары и открытые разборы задач;
- хакатоны, практикумы и мастер-классы для финалистов;
- встречи в онлайн-формате, информация о которых публикуется в группе НТО во «ВКонтакте» и в телеграм-чатах профилей.

Наставнику необходимо уделить внимание работе на формированием устойчивой, продуктивной и мотивированной команды:

- **Сплочение команды.** Это особенно актуально, если участники живут в разных городах. Регулярные онлайн-встречи, совместная работа над задачами и неформальное общение помогают наладить доверие и улучшить командную динамику.
- **Анализ ролей.** Наставник вместе с командой определяет, кто за что отвечает, какие задачи входят в зону ответственности каждого участника. Также обсуждаются возможности взаимозаменяемости на случай непредвиденных ситуаций.
- **Оценка компетенций.** Важно определить, какими знаниями и навыками уже обладают участники, а какие необходимо развить. На основе этого формируется индивидуальный и командный план подготовки.
- **Участие в подготовительных мероприятиях от разработчиков профилей.**

Перед заключительным этапом проводятся установочные вебинары, разборы задач прошлых лет, практикумы, мастер-классы для финалистов. Информация о таких мероприятиях публикуется в группе НТО в VK и в чатах профилей в Telegram.

- **Практика в формате хакатонов.** Наставник может организовать дистанционные хакатоны или практикумы с использованием заданий прошлых лет и методических рекомендаций из официальных сборников.

Таким образом, наставник становится координатором и моральной опорой команды, помогая пройти заключительный этап НТО с максимальной уверенностью и результатом.

4.2. Предметный тур

Задачи третьего этапа предметного тура профиля по информатике открыты для решения. Участие в соревновании доступно на платформе Яндекс.Контест: https://contest.yandex.ru/contest/72672/enter/.

4.2.1. Информатика. 8–11 классы

Задача 4.2.1.1. Тренировка многоборца (10 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

В качестве тренировки один спортсмен по многоборью дважды проводит забег-заезд из пункта A в пункт C.

В первый раз он пробегает часть этой дистанции от пункта A до некоторого промежуточного пункта B на маршруте бегом, в пункте B садится на велосипед и проезжает оставшуюся часть пути до пункта C на нем. В итоге он тратит на это X часов.

В следующий раз он, наоборот, из пункта A до пункта B проезжает на велосипеде, а из пункта B до пункта C бежит бегом. В этот раз он тратит на весь маршрут Y часов.

Требуется узнать, за какое время он может пробежать всю дистанцию от A до C, если его скорость на велосипеде в k раз выше скорости его бега. Можно считать, что скорость его бега и скорость его езды на велосипеде всегда постоянные.

Формат входных данных

В одной строке через пробел заданы три целых числа X, Y и k — время первого прохождения маршрута, время второго прохождения маршрута и величина, показывающая, во сколько раз быстрее он передвигается на велосипеде, чем бегом. $1 \leqslant X, Y, k \leqslant 1\,000$.

Формат выходных данных

Вывести одно число — время, за которое спортсмен пробежит весь маршрут от A до C.

Тесты, на которых будет проверяться ваше решение, сгенерированы таким образом, что ответ всегда является целым числом.

Примеры

Пример №1

Стандартный ввод
3 5 3

Стандартный вывод
6

Примечания

В примере из условия для объяснения теста приведем полное рассуждение с построением расстояний AB и BC, хотя эти расстояния находить не обязательно. Дано, что в первый раз спортсмен затратил три часа, а во второй — пять часов, при этом на велосипеде он едет в три раза быстрее, чем бежит бегом.

После некоторых рассуждений можно получить, что если взять длину AB равной 15 км, длину BC равной 45 км, скорость бега равной 10 км/ч, а скорость на велосипеде равной 30 км/ч, то получим требуемые результаты: $\frac{15}{10} + \frac{45}{30} = 3$, $\frac{15}{30} + \frac{45}{10} = 5$. Общее расстояние AB равно $15 + 45$ и равно 60 км. Тогда спортсмен, имея скорость бега равную 10 км/ч, пробежит 60 км за шесть часов.

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
using namespace std;
signed main()
{
    int x, y, k;
    cin >> x >> y >> k;
    int v = (x + y) / (k + 1);
    int b = x + y - v;
    cout << b << endl;
}
```

Задача 4.2.1.2. Пространственная решетка (15 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

После открытия так называемых точек сингулярности пространства и формирования теории пространственной решетки всей Вселенной пришла пора практического освоения космоса. Если кратко, то у каждой точки сингулярности A есть своя уникальная характеристика $P(A)$ — натуральное число от 1 до 10^{12}. Между некоторыми точками есть силовые линии, вдоль которых возможно движение со сверхсветовыми скоростями. Из одной точки A можно попасть таким образом в другую точку B, если характеристика $P(A)$ получается из $P(B)$ либо делением на простое число, либо умножением на простое число. Независимо от размеров характеристик, движение вдоль одной такой линии занимает одинаковое время, называемое константой Диогена. По этой причине все космические межзвездные перелеты измеряются этими константами.

Предлагаемая задача входит в базовый курс звездной навигации, изучаемой курсантами на первом цикле. Даны две точки сингулярности A и B, причем $P(A)$ делит $P(B)$. Необходимо перечислить все точки сингулярности, которые находятся на кратчайших путях из A в B.

Формат входных данных

Заданы два натуральных числа $P(A)$ и $P(B)$ — характеристики точек A и B. Гарантируется, что $P(B)$ нацело делится на $P(A)$. Обе характеристики находятся в пределах от 1 до 10^{12}.

Формат выходных данных

В первой строке вывести количество искомых точек. Во второй строке вывести через пробел в порядке возрастания все характеристики точек сингулярности, которые находятся на всех кратчайших расстояниях из точки A в точку B.

Примеры

Пример №1

Стандартный ввод
`6 120`

Стандартный вывод
`6` `6 12 24 30 60 120`

Примечания

Приведем несколько примеров кратчайших путей из точки 6 в точку 120 (точки представлены своими характеристиками):

6 – 12 – 24 – 120;

6 – 30 – 60 – 120;

6 – 12 – 60 – 120.

Можно видеть, что других чисел, кроме представленных в ответе на этих и других кратчайших путях из 6 в 120, также нет.

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define sz(a) (int)a.size()
#define int long long
using namespace std;
signed main(){
    int a, b;
    cin >> a >> b;
    int c = b / a;
    set<int> S;
    for(int i = 1; i * i <= c; i++){
        if(c % i == 0){
            S.insert(i);
            S.insert(c / i);
        }
    }
    cout << sz(S)<< endl;
    for(auto el : S){
        cout << el * a <<' ';
    }
    cout << endl;
}
```

Задача 4.2.1.3. Ключ и замок (20 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Замок представляет собой конструкцию размера h единиц в высоту и n единиц в длину. Внутри замка содержатся n вертикальных цилиндров, каждый имеет

ширину 1 и высоту a_i. Эти цилиндры расположены внутри замка в порядке неубывания, то есть $a_{i-1} \geqslant a_i$ для всех i от 2 до n. Все a_i строго меньше высоты замка h. Цилиндры внутри замка закреплены неподвижно.

Ключ представляет собой также объект длиной n единиц и состоит из n прямоугольников, каждый имеет ширину 1 и высоту b_i. В ключе, наоборот, высоты прямоугольников не возрастают, то есть $b_{i-1} \leqslant b_i$ для всех i от 2 до n. Все b_i также строго меньше высоты замка h.

Ключ и замок не обязательно подходят друг к другу. Ключ пытаются вставить как можно дальше в замок и фиксируют длину, на которую это получится сделать. Изначально самый маленький (допустим, левый) прямоугольник ключа находится на позиции $(n + 1)$, и все остальные его прямоугольники — на последующих еще больших позициях. Далее, до тех пор пока можно сдвинуть ключ на одну позицию левее, ключ сдвигают. В какой-то момент либо прямоугольники ключа и цилиндры замка начинают мешать дальнейшему движению, либо ключ вставляется до конца. В этот момент процесс прекращается и измеряется количество позиций, на которые был сдвинут ключ влево. Это и является ответом.

Для понимания условия рассмотрим первый тест. Схематичное его изображение приведено на рис.4.2.1. Замок обозначен серым, ключ — желтым. Можно видеть, что ключ можно вставить внутрь замка на 13 позиций, после чего прямоугольник ключа высотой 4 упрется в цилиндр замка высотой 7, и дальнейшее движение влево станет невозможным.

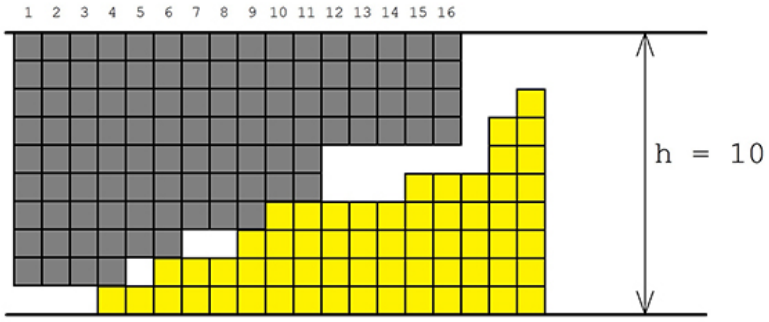


Рис. 4.2.1

Формат входных данных

В первой строке содержится два числа n и h через пробел — длина замка и ключа и высота замка, $1 \leqslant n, h \leqslant 3 \cdot 10^5$.

Во второй строке находится набор чисел a_i через пробел — высоты цилиндров замка в порядке их расположения внутри замка. $a_{i-1} \geqslant a_i$ для всех i от 2 до n, $1 \leqslant a_i < h$.

В третьей строке находится набор чисел b_i через пробел — высоты прямоугольников ключа в порядке их расположения внутри ключа. $b_{i-1} \leqslant b_i$ для всех i от 2 до n, $1 \leqslant b_i < h$.

Формат выходных данных

Вывести одно число — наибольшее число позиций, на которое можно вставить ключ внутрь замка.

Примеры

Пример №1

Стандартный ввод
16 10 9 9 9 9 8 8 7 7 7 6 6 4 4 4 4 4 1 1 2 2 2 3 4 4 4 4 4 5 5 5 7 8
Стандартный вывод
13

Пример №2

Стандартный ввод
16 10 9 9 9 9 8 8 7 7 7 6 6 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Стандартный вывод
16

Пример №3

Стандартный ввод
16 10 9 9 9 9 8 8 7 7 7 6 6 4 4 4 4 4 7 7 7 8 8 8 9 9 9 9 9 9 9 9 9 9
Стандартный вывод
0

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```
#include<bits/stdc++.h>
#define for0(i, n) for(int i = 0; i < n; i++)
#define int long long
using namespace std;
typedef vector<int> vi;
const int INF = 1e18;
```

```
signed main(){
    int n, h;
    cin >> n >> h;
    vi a(n), b(n);
    for0(i, n){
        cin >> a[i];
    }
    for0(i, n){
        cin >> b[i];
    }
    int tb = 0;
    int mx = -INF;
    for0(i, n){
        while(tb < n && b[tb] + a[i] <= h){
            tb++;
        }
        mx = max(mx, i + n - tb + 1);
    }
    cout << 2 * n - mx << endl;
}
```

Задача 4.2.1.4. Фрактальная федерация (25 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 64 Мбайт.

Условие

Фрактальная федерация имеет форму клетчатого прямоугольника. При ее основании было выбрано натуральное число z, большее 1, и основание k, большее 0. Далее на местности выделили клетчатый прямоугольник $z^k \times z^{k-1}$.

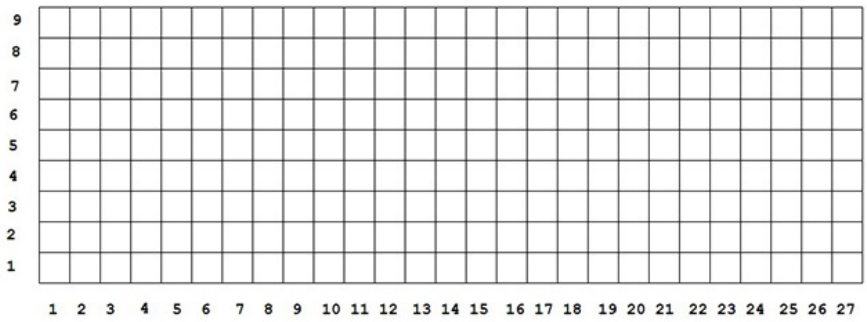


Рис. 4.2.2

На рис. 4.2.2 приведен пример для $z = 3$ и $k = 3$, то есть федерация в этом примере имеет вид прямоугольника 27×9.

Далее, если $k > 1$, федерацию разделили на z^2 федеральных территорий. Каждая такая территория подобна исходной федерации и имеет также форму прямоуголь-

ника, но размера $z^{k-1} \times z^{k-2}$. Федеральные территории расположены в один ряд, границы между ними имеют уровень $k-1$.

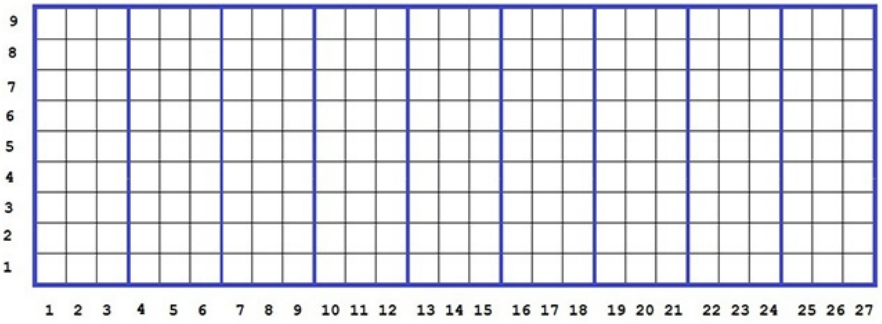


Рис. 4.2.3

9 синих прямоугольников обозначают 9 федеральных территорий 9×3 каждая, границы между ними имеют уровень 2 (рис. 4.2.3).

Далее, если $k > 2$, каждую федеральную территорию снова разделили на z^2 регионов. Каждый такой регион снова подобен исходной федерации и имеет также форму прямоугольника, но размера $z^{k-2} \times z^{k-3}$. Регионы снова расположены внутри федеральных территорий в один ряд, границы между ними имеют уровень $k-2$ (рис. 4.2.4).

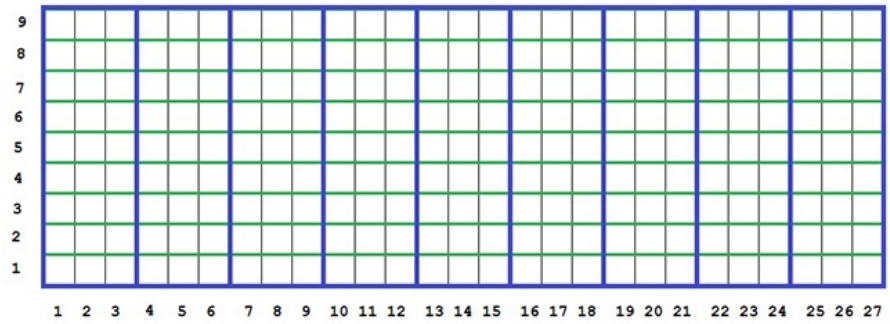


Рис. 4.2.4

Внутри каждой из 9 федеральных территорий зеленым выделены по 9 регионов 3×1. Зеленые границы имеют уровень 1.

Такое фрактальное деление производилось, и границы соответствующего уровня проводились и далее до тех пор, пока очередная область разбиения не стала иметь размеры $z \times 1$. На этом этапе решено было остановиться. Оставшиеся никак не затронутыми границы между единичными клетками получили уровень 0.

Клетки, из которых состоит федерация, пронумеровали от 1 до z^k по первой координате и от 1 до z^{k-1} по второй координате. Внутри федерации можно перемещаться из клетки в любую соседнюю с ней по стороне. При этом если между этими клетками проходит граница уровня t, то за это перемещение требуется заплатить z^t единиц.

На территории федерации находится n городов, каждый занимает ровно одну клетку. По заданным координатам этих городов требуется составить матрицу минимальных стоимостей для перемещения между двумя любыми городами. Более точно,

на пересечении строки номер i и столбца номер j в матрице должна содержаться минимальная стоимость, за которую можно добраться из города номер i в город номер j с учетом пересечения всех границ между ними. На главной диагонали матрицы должны содержаться нули.

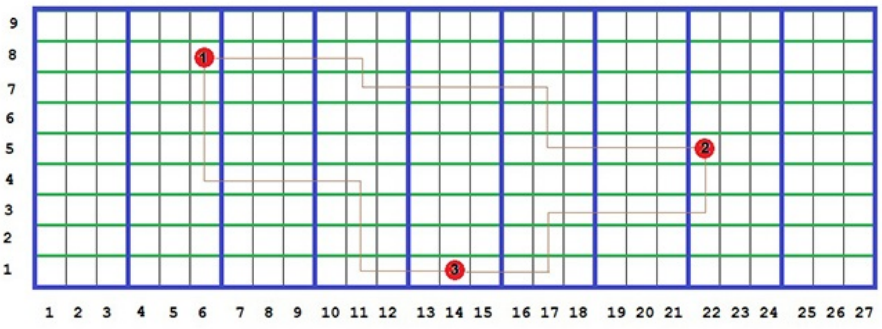


Рис. 4.2.5

В предложенном примере за пересечение любой черной границы требуется заплатить 1 единицу, за пересечение любой зеленой границы — 3 единицы, за пересечение любой синей границы — 9 единиц. На рис. 4.2.5 представлены три города и некоторые кратчайшие по стоимости пути между ними. Например, при движении от города № 1 до города № 2 по указанному на рисунке пути между ними, затраты составят: $9+1+1+9+1+3+1+9+1+1+9+1+3+3+1+9+1+1+9=73$.

Формат входных данных

В первой строке через пробел приведены два целых числа z и k, лежащие в основе федерации, $2 \leqslant z \leqslant 10$. Число $k > 0$ и таково, что $z^k \leqslant 10^{17}$.

Во второй строке содержится число городов n, $2 \leqslant n \leqslant min(z^{2k-1}, 1\,000)$.

В следующих n строках содержится по два целых числа x_i и y_i через пробел — координаты очередного города. $1 \leqslant x_i \leqslant z^k$, $1 \leqslant y_i \leqslant z^{k-1}$, все города находятся в попарно различных клетках.

Формат выходных данных

В ответ вывести матрицу $n \times n$. На пересечении строки номер i и столбца номер j в этой матрице должна содержаться минимальная стоимость, за которую можно добраться из города номер i в город номер j с учетом пересечения всех границ между ними. Если на пути между двумя городами находится третий город, это никак не влияет на стоимость перемещения. Числа внутри каждой строки матрицы разделять одним пробелом.

Примеры

Пример №1

Стандартный ввод
3 3 3 6 8 22 5 14 1

Стандартный вывод
0 73 53 73 0 44 53 44 0

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define sz(a) (int)a.size()
#define pb push_back
#define for0(i, n) for(int i = 0; i < n; i++)
#define x first
#define y second
#define int long long
using namespace std;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<vector<int> > vvi;
signed main(){
    int z, k;
    cin >> z >> k;
    int n;
    cin >> n;
    vector<pii> v(n);
    for0(i, n){
        cin >> v[i].x >> v[i].y;
        v[i].x--;
        v[i].y--;
    }
    vector<pii> V, H;
    V.pb({1, 1});
    int d =  1;
    for0(i, k - 1){
        if(i % 2 == 0){
            H.pb({d, d * z});
        }
        else{
            V.pb({d, d * z});
        }
        d *= z;
    }
    if(k % 2 == 0){
```

```
36          swap(V, H);
37      }
38      vvi ans(n, vi(n, 0));
39      for0(i, n){
40          for(int j = i + 1; j < n; j++){
41              int dxans = 0;
42              int dxk = 0;
43              for(int t = sz(V) - 1; t >= 0; t--){
44                  int di = v[i].x / V[t].x;
45                  int dj = v[j].x / V[t].x;
46                  dxans += (abs(di - dj) - dxk) * V[t].y;
47                  dxk += (abs(di - dj) - dxk);
48              }
49              int dyans = 0;
50              int dyk = 0;
51              for(int t = sz(H) - 1; t >= 0; t--){
52                  int di = v[i].y / H[t].x;
53                  int dj = v[j].y / H[t].x;
54                  dyans += (abs(di - dj) - dyk) * H[t].y;
55                  dyk += (abs(di - dj) - dyk);
56              }
57              ans[i][j] = dxans + dyans;
58              ans[j][i] = ans[i][j];
59          }
60      }
61  for0(i, n){
62      for0(j, n){
63          cout << ans[i][j]<<' ';
64      }
65      cout << endl;
66  }
67  }
```

Задача 4.2.1.5. Рейтинг проекта (30 баллов)

Имя входного файла: стандартный ввод или `input.txt`.

Имя выходного файла: стандартный вывод или `output.txt`.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 128 Мбайт.

Условие

Директор института Эдуард Леонидович составляет итоговый отчет о проделанной работе. Каждый день в подведомственном ему заведении либо стартует ровно один новый проект, либо завершается ровно один из уже начатых. Каждый проект имеет свой уникальный номер, но порядок, в котором они стартуют, никак не связан с их нумерацией.

Перед Эдуардом Леонидовичем лежит список, в котором содержится очередность произведенных институтом действий в хронологическом порядке. В нем для каждого дня указан номер начатого или законченного в этот день проекта. Например, если порядок такой: 2, 4, 5, 6, 6, 5, 3, 2, 1, 4, 1, 3, то это значит, что проект № 2 стартовал в первый день, а завершился в восьмой, проект № 3 стартовал в седьмой

день, а завершился в двенадцатый, и т. д.

Нужно заметить, что в министерстве, к которому принадлежит институт, важное значение имеет преемственность. Считается, что проект B является продолжением проекта A, если проект B начался между датами начала и окончания проекта A и закончился строго после окончания проекта A. Если проследить это в хронологическом списке, то указанные проекты должны в нем располагаться так: $\ldots A \ldots B \ldots A \ldots B \ldots$

Исходя из этого вычисляется рейтинг для каждого проекта. Для некоторого проекта A его рейтинг равен числу всех проектов, которые либо являются его продолжением, либо для которых проект A является продолжением.

В рассмотренном примере у проекта № 5 рейтинг равен 0, у проекта № 1 рейтинг равен 1 (он продолжает проект № 4), а у проекта № 4 рейтинг уже равен 3, так как он продолжает проект № 2, и его в свою очередь продолжают проекты 1 и 3.

Теперь нужно помочь Эдуарду Леонидовичу с отчетностью и для каждого проекта найти его рейтинг.

Формат входных данных

В первой строке задается количество проектов n, выполненных институтом за отчетный период, $1 \leqslant n \leqslant 3 \cdot 10^5$.

Во второй строке задана последовательность выполнения проектов. Она состоит из $2 \cdot n$ чисел через пробел. Каждое из этих чисел от 1 до n, и каждое встречается в последовательности ровно два раза. Первое вхождение соответствует началу проекта с этим номером, второе вхождение — окончанию.

Формат выходных данных

Вывести в одну строку через пробел рейтинги всех проектов. На первом месте должен быть рейтинг проекта № 1, но втором — рейтинг проекта № 2 и т. д. до рейтинга проекта № n.

Примеры

Пример №1

Стандартный ввод
6 2 4 5 6 6 5 3 2 1 4 1 3

Стандартный вывод
1 2 2 3 0 0

Пример программы-решения

Ниже представлено решение на языке C++.

C++

```cpp
#include<bits/stdc++.h>
#define for0(i, n) for(int i = 0; i < n; i++)
#define for1(i, n) for(int i = 1; i <= n; i++)
#define int long long
using namespace std;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<vector<int> > vvi;
const int INF = 1e18;
const int MOD = 1e9 + 7;
const int LG = 20;
const int N = (1LL << LG);
vi tr(2 * N, 0);
void upd(int pos, int d){
    pos += N;

    tr[pos] += d;
    pos /= 2;
    while(pos){
        tr[pos] = tr[2 * pos] + tr[2 * pos + 1];
        pos /= 2;
    }
}
int get(int l, int r){
    l += N;
    r += N;
    int res = 0;
    while(l <= r){
        if(l % 2 == 1){
            res += tr[l];
        }
        if(r % 2 == 0){
            res += tr[r];
        }
        l = (l + 1) / 2;
        r = (r - 1) / 2;
    }
    return res;
}
signed main(){
    int n;
    cin >> n;
    vi frst(n + 1, -1);
    vi ans(n + 1, -1);
    for0(i, 2 * n){
        int a;
        cin >> a;
        if(frst[a] == -1){
            frst[a] = i;
            upd(i, 1);
            continue;
        }
        int diff = get(frst[a] + 1, i - 1);
        int all = i - frst[a] - 1;
        int rep = all - diff;
        ans[a] = all - 2 * rep;
        upd(frst[a], -1);
        upd(i, 1);
    }
```

```
60      for1(i, n){
61          cout << ans[i] <<' ';
62      }
63      cout << endl;
64  }
```

4.2.2. Физика. 8–9 классы

Задача 4.2.2.1. Два бака (15 баллов)

Условие

Двигатель новаторского суборбитального летательного аппарата может работать в двух режимах: самолетном и ракетном. В первом он сжигает топливо, используя в качестве окислителя забортный воздух, при этом удельная теплота сгорания топлива в этой реакции составляет $q_1 = 40$ МДж/кг. Второй режим активируется, когда аппарат поднимается на высоту, плотность воздуха на которой недостаточна для поддержания стабильного сгорания топлива. В этом режиме в качестве окислителя начинает использоваться жидкий реактив, хранящийся в баках на борту самолета. Удельная теплота сгорания (в расчете на массу топлива) в этой реакции составляет $q_2 = 60$ МДж/кг, но для ее поддержания приходится расходовать окислитель, массой в $n = 1{,}5$ превышающий массу окисляемого им топлива.

Для выполнения полетного задания двигателям аппарата придется произвести одинаковое количество теплоты в самолетном и ракетном режимах. Определите, в какой объемной пропорции необходимо разделить общее доступное для хранения реагентов пространство на борту аппарата между топливом и окислителем, чтобы добиться максимального запаса хода при таких требованиях. Плотность топлива $\rho_т = 900$ кг/м^3, плотность окислителя $\rho_о = 1\,200$ кг/м^3. Считайте, что зависимость расхода аппаратом топлива от массы реагентов в его баках пренебрежимо мала.

Решение

Пускай количество теплоты, производимое двигателем в каждом режиме, равно Q. Для самолетного режима эта величина может быть выражена как

$$Q = q_1 m_{т1} = q_1 \rho_т V_{т1} \Rightarrow V_{т1} = \frac{Q}{q_1 \rho_т},$$

где $m_{т1}$ и $V_{т1}$ — масса и объем сжигаемого в этом режиме топлива соответственно.

Для ракетного режима объем потребляемого топлива равен аналогично

$$V_{т2} = \frac{Q}{q_2 \rho_т}.$$

Таким образом, общий объем $V_т$, который необходимо выделить под топливо, равен

$$V_т = V_{т1} + V_{т2} = V_{т2}\left(\frac{q_2}{q_1} + 1\right).$$

Объем, требуемый для хранения окислителя, легко вычислить из известной пропорции масс. Обозначим $m_о$ и $V_о$ массу и объем окислителя соответственно. Нам известно, что $m_о = n m_{т2}$. Перепишем это равенство через объемы:

$$\rho_о V_о = n \rho_т V_{т2} \Rightarrow V_о = V_{т2} \frac{n \rho_т}{\rho_о}.$$

Осталось найти отношение двух полученных объемов:

$$\frac{V_т}{V_о} = \frac{\cancel{V_{т2}}(1 + q_2/q_1)}{\cancel{V_{т2}} n\rho_т/\rho_о} = \frac{20}{9}.$$

Ответ:

$$V_т : V_о = \frac{1 + q_2/q_1}{n\rho_т/\rho_о} = 20 : 9 \approx 2{,}22.$$

Критерии оценивания

Продемонстрировано понимание понятия «удельная теплота сгорания» (верно записана связь теплоты и массы топлива)	5 баллов
Верно записана связь масс топлива и окислителя с их объемами	3 балла
Точно: в общем виде или простой дробью получен правильный ответ	7 баллов
Правильный ответ получен только численно с погрешностью не более 0,05	4 балла
Всего	**15 баллов**

Если соответствующие шаги успешно проделаны только для части режимов работы двигателя — за них ставится пропорциональная часть максимального балла на усмотрение проверяющего.

Задача 4.2.2.2. Полюс и экватор (18 баллов)

Условие

Ракета-носитель выводит спутник на круговую орбиту радиуса $r = 7\,000$ км таким образом, что до достижения заданной высоты орбиты двигатели ракеты создают силу, направленную строго вертикально (от центра Земли). Уже достигнув заданной высоты с нулевой вертикальной скоростью, последняя ступень ракеты включает свои двигатели, за короткое время увеличивающие импульс спутника на некую постоянную величину Δp в строго горизонтальном направлении. Инженерами было установлено, что, действуя по данному алгоритму, ракета способна вывести на указанную орбиту спутник максимальной массы $m = 600$ кг, если запускать его с космодрома, находящегося на полюсе планеты. Спутник какой максимальной массы сможет запустить по такому же алгоритму ракета, стартующая с космодрома, находящегося на экваторе? Экваториальный радиус Земли $R = 6\,380$ км, ее масса $M = 6 \cdot 10^{24}$ кг, гравитационная постоянная $G = 6{,}67 \cdot 10^{-11}$ Н·м^2/кг^2.

Решение

Найдем сначала орбитальную скорость v спутника на орбите радиуса r. Для этого приравняем центростремительное ускорение этого спутника $a = v^2/r$ к ускорению свободного падения на соответствующей высоте:

$$\frac{v^2}{r} = G\frac{M}{r^2} \Rightarrow v = \sqrt{\frac{GM}{r}}.$$

В случае старта с полюса спутник не имеет изначальной горизонтальной компоненты скорости, а значит, всю эту скорость спутник получает благодаря горизонтальному импульсу последней ступени:

$$\Delta p = mv = m\sqrt{\frac{GM}{r}}.$$

При старте с экватора спутник изначально имеет горизонтальную составляющую скорости v_0, обусловленную суточным вращением космодрома относительно центра планеты:

$$v_0 = \frac{2\pi R}{T},$$

где $T = 24$ ч. Следовательно, импульс Δp должен обеспечить только повышение скорости от v_0 до v:

$$\Delta p = m_1(v - v_0) \Rightarrow m_1 = \frac{\Delta p}{v - v_0} = m\sqrt{\frac{GM}{r}}\left(\sqrt{\frac{GM}{r}} - \frac{2\pi R}{T}\right)^{-1}.$$

Ответ:

$$m_1 = m\sqrt{\frac{GM}{r}}\left(\sqrt{\frac{GM}{r}} - \frac{2\pi R}{T}\right)^{-1} \approx 639 \text{ кг}.$$

Критерии оценивания

Продемонстрировано понимание причин различных масс спутника при старте с экватора и полюса	2 балла
Верно найдена связь орбитальной скорости с высотой	6 баллов
Верно найдена скорость спутника, обусловленная его суточным вращением	3 балла
Верно записана связь всех скоростей с импульсом в задаче	3 балла
Получен правильный ответ	6 баллов
Всего	**20 баллов**

Частичный успех на каждом из этих шагов может оцениваться в долю от максимального балла за соответствующий шаг на усмотрение проверяющего. Например, верно записан закон всемирного тяготения, но орбитальную скорость выразить не удалось, получен верный ответ в общем виде, но при подстановке чисел допущена арифметическая ошибка и т. п. Аналогично в последующих задачах.

Задача 4.2.2.3. Транспирация (22 балла)

Условие

В некоторых ракетных двигателях используется метод транспирационного охлаждения. Он состоит в том, что жидкий охладитель продавливается сквозь пористую стенку охлаждаемого прибора, образуя на ее поверхности равномерную тонкую пленку, которая, испаряясь, уносит избыточное тепло агрегата равномернее и эффективнее, чем большинство других методов.

Инженерами в качестве такого охладителя был выбран жидкий аммиак, имеющий удельную теплоту парообразования и конденсации $L = 1{,}37\,\text{МДж/кг}$ и плотность $\rho = 683$ кг/м^3. Аммиак движется сквозь поры специального керамического покрытия с постоянной скоростью $v = 1$ мм/с. Определите, каким должен быть средний диаметр круглых сквозных пор в этом покрытии, чтобы при их поверхностной плотности $\sigma = 10^6$ м$^{-2}$ система охлаждения могла обеспечить плотность теплового потока $f = 10$ кВт/м^2. Теплоту, идущую на изменение температуры аммиака, считать пренебрежимо малой по сравнению с теплотой его испарения.

Решение

Круглая пора с диаметром d имеет площадь $S_0 = \pi d^2/4$. За некоторое время t сквозь каждую такую пору проходит объем аммиака

$$V = vtS_0 = \frac{\pi d^2}{4} vt.$$

Этот объем содержит в себе массу $m = \rho V$ вещества, а значит, при испарении отводит количество теплоты

$$Q_0 = L\rho V = \frac{\pi d^2}{4} L\rho vt.$$

Поверхностная плотность σ — есть отношение числа N пор на элементе поверхности к площади S этой поверхности:

$$\sigma = \frac{N}{S}.$$

Следовательно, с площади S поверхности за время t отводится тепло

$$Q = NQ_0 = \frac{\pi d^2}{4} L\rho v\sigma St.$$

Плотность потока тепла f (как можно судить из названия и единиц измерения) представляет собой отношение количества теплоты, уходящего с поверхности двигателя, ко времени и площади соответствующей поверхности:

$$f = \frac{Q}{St} = \frac{\pi d^2}{4} L\rho v\sigma,$$

откуда окончательно

$$d = \sqrt{\frac{4f}{\pi L\rho v\sigma}} \approx 0{,}12\,\text{мм}.$$

Примечание: поры такого диаметра и плотности расположения покрывают приблизительно 1% общей площади поверхности керамики.

Ответ:

$$d = \sqrt{\frac{4f}{\pi L \rho v \sigma}} \approx 0{,}12\,\text{мм}.$$

Критерии оценивания

Верно записана связь между удельной теплотой парообразования и количеством отводимой теплоты	3 балла
Верно записана связь диаметра пор, скорости течения жидкости и ее объемного расхода	6 баллов
Продемонстрировано понимание терминов «поверхностная плотность» и «плотность теплового потока»	по 3 балла
Получен правильный ответ	7 баллов
Всего	**22 балла**

Задача 4.2.2.4. Группировка (20 баллов)

Условие

Спутниковая группировка состоит из $n = 12$ одинаковых спутников, движущихся по одной и той же круговой орбите над экватором Земли. Согласно техническим требованиям, спутники группировки должны быть разделены равными расстояниями друг от друга, и каждая пара ближайших друг к другу спутников должна непрерывно обмениваться исследовательскими данными по радиоканалу. При этом используемые частоты радиосвязи отражаются от слоя ионосферы Земли, расположенного на постоянной высоте $h = 80\,\text{км}$ от поверхности планеты, но проходят через более высокие слои. Спутники выводят на самую низкую орбиту, удовлетворяющую этим требованиям. В некоторый момент $m = 4$ спутника группировки потребовалось вывести из эксплуатации. Определите, на какую величину Δv и в какую сторону необходимо изменить орбитальные скорости остальных спутников, чтобы вновь удовлетворить техническим требованиям на самой низкой возможной орбите. Экваториальный радиус Земли $R = 6\,380\,\text{км}$, ее масса $M = 6 \cdot 10^{24}\,\text{кг}$, гравитационная постоянная $G = 6{,}67 \cdot 10^{-11}\ \text{Н} \cdot \text{м}^2/\text{кг}^2$.

Решение

Поскольку спутники находятся на одной круговой орбите и разделены равными расстояниями, можно говорить, что они находятся в вершинах правильного $n-$угольника. Чтобы обеспечить нахождение ближайших друг к другу спутников в зоне взаимной видимости, прямая, соединяющая их (сторона $n-$угольника), не должна пересекать сферу с радиусом $R + h$ и центром в центре планеты, от которой отражаются радиосигналы. Наиболее низкое расположение спутников, при котором

это возможно, такое, что упомянутый $n-$угольник описан около соответствующей сферы. На рис. 4.2.6 для наглядности приведен пример для $n = 6$, а вид Земли и масштаб не соблюдены.

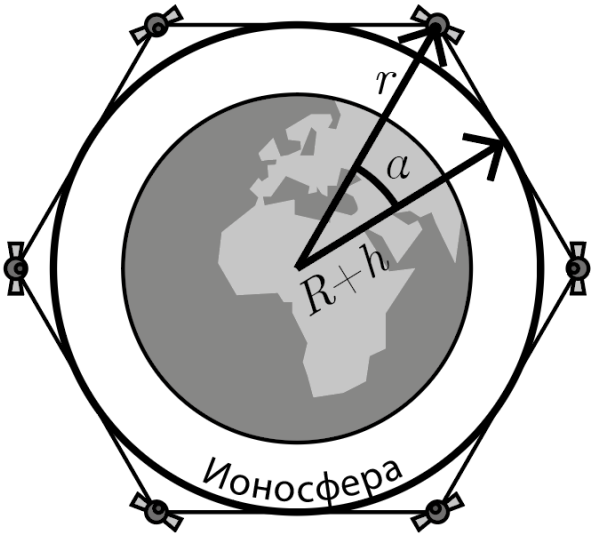


Рис. 4.2.6

Угол α образуется между двумя векторами (см. рис. 4.2.6). Один из них направлен из центра планеты на точку касания отрезка между соседними спутниками с ионосферой. Второй — на один из спутников. Этот угол равен половине угловой меры дуги орбиты, разделяющей два соседних спутника:

$$\alpha = \frac{\pi}{n}.$$

Через этот угол легко выразить связь между радиусом $R+h$ ионосферы и радиусом r орбиты спутников:

$$R + h = r\cos\alpha \Rightarrow r = \frac{R+h}{\cos(\pi/n)}.$$

Зная радиус орбиты, легко определить и соответствующую ему орбитальную скорость v (как в задаче 2) в зависимости от числа спутников n:

$$v(n) = \sqrt{\frac{GM}{r(n)}} = \sqrt{\frac{GM\cos(\pi/n)}{R+h}},$$

откуда получим окончательно

$$\Delta v = v(n-m) - v(n) = \sqrt{\frac{GM}{R+h}}\left(\cos^{1/2}\left(\frac{\pi}{n-m}\right) - \cos^{1/2}\left(\frac{\pi}{n}\right)\right).$$

Ответ:

$$\Delta v = \sqrt{\frac{GM}{R+h}}\left(\cos^{1/2}\left(\frac{\pi}{n-m}\right) - \cos^{1/2}\left(\frac{\pi}{n}\right)\right) \approx -170\,\text{м/с}.$$

То есть орбитальную скорость необходимо уменьшить на 170 м/с.

Критерии оценивания

Продемонстрировано понимание геометрии расположения спутников на орбите относительно ионосферы	4 балла
Верно найдена связь орбитальной скорости с радиусом орбиты	5 баллов
Верно найдена связь орбитальной скорости с радиусом ионосферы и числом спутников в группировке	4 балла
Получен правильный ответ	7 баллов
Всего	**20 балла**

Задача 4.2.2.5. Стратостат (25 баллов)

Условие

Стратостат, предназначенный для изучения атмосферного электричества, представляет собой шар объемом $V = 400$ м^3, заполненный гелием. Стратостат соединен с землей длинным проводом, электрическое сопротивление которого, согласно требованиям эксперимента, должно быть равно $R = 600\,\text{Ом}$. Изначально для этого использовался медный провод, в результате чего максимальная высота подъема аппарата оказывалась ограничена значением $H_1 = 16\,\text{км}$. До какой высоты сможет подняться тот же аппарат, если изготовить провод из алюминия? Плотность меди $\rho_\text{м} = 8\,960$ кг/м^3, плотность алюминия $\rho_\text{а} = 2\,700$ кг/м^3, удельное сопротивление меди $r_\text{м} = 1{,}7 \cdot 10^{-8}$ Ом·м, удельное сопротивление алюминия $r_\text{а} = 2{,}65 \cdot 10^{-8}$ Ом·м. На высотах полета аппарата плотность воздуха ρ снижается с высотой на $\Delta\rho = 0{,}02$ кг/м^3 на каждый километр высоты.

Решение

Высота, до которой может подняться стратостат, определяется балансом между его подъемной силой $F = \rho(h)gV$ и общим поднимаемым им весом P. Здесь $\rho(h)$ — плотность воздуха на высоте h. Этот вес включает в себя вес mg самого стратостата (включая оболочку, оборудование и наполняющий оболочку газ) и вес $\rho_\text{п}ghS$ провода, где $\rho_\text{п}$ — плотность металла, из которого он изготовлен, S — площадь его поперечного сечения, а h — высота подъема стратостата, совпадающая с минимальной длиной провода, позволяющего соединить аппарат с Землей:

$$mg + \rho_\text{п}ghS = \rho(h)gV.$$

Площадь поперечного сечения провода можно найти из требований на его сопротивление:

$$R = \frac{rh}{S} \Rightarrow S = \frac{rh}{R},$$

где r — удельное сопротивление. Отсюда

$$mg = \rho(h)gV - \frac{\rho_\text{п}grh^2}{R}.$$

Масса m в левой части этого уравнения не зависит от материала провода и высоты подъема стратостата. Поэтому, обозначив H_2 максимальную высоту, доступную стратостату с алюминиевым проводом, приравняем массы, выраженные из этого уравнения для двух описанных случаев, и получим

$$\rho(H_1)V - \frac{\rho_{\text{м}} r_{\text{м}} H_1^2}{R} = \rho(H_2)V - \frac{\rho_{\text{а}} r_{\text{а}} H_2^2}{R}.$$

Перепишем это уравнение в более удобной форме:

$$\frac{1}{R}(r_{\text{м}}\rho_{\text{м}}H_1^2 - r_{\text{а}}\rho_{\text{а}}H_2^2) + V(H_1 - H_2)\frac{\Delta\rho}{\Delta h} = 0,$$

где $\Delta h = 1\,\text{км}$, воспользовавшись тем, что разницу в плотностях на высотах H_1 и H_2 удобно записать напрямую через известную нам величину $\Delta\rho$:

$$\rho(H_1) - \rho(H_2) = (H_2 - H_1)\frac{\Delta\rho}{\Delta h}.$$

Обратите внимание: увеличение высоты приводит к уменьшению плотности воздуха, поэтому порядок следования H_1 и H_2 в правой и левой частях уравнения различен.

Полученное уравнение является квадратным относительно H_2 и имеет единственный положительный корень

$$H_2 = \frac{-\Delta p/\Delta h RV + \sqrt{\Delta p^2/\Delta h^2 R^2 V^2 + 4\Delta p/\Delta h H_1 RV r_{\text{а}}\rho_{\text{а}} + 4H_1^2 r_{\text{м}}\rho_{\text{м}} r_{\text{а}}\rho_{\text{а}}}}{2r_{\text{а}}\rho_{\text{а}}} \approx 18{,}8\,\text{км}.$$

Его, разумеется, проще получить, подставив промежуточные значения непосредственно в квадратное уравнение.

Ответ: $H_2 \approx 18{,}8\,\text{км}$.

Критерии оценивания

Продемонстрировано понимание природы ограничений на максимальную высоту подъема стратостата	3 балла
Верно записано условие равновесия стратостата	3 балла
Верно записана связь между полным и удельным сопротивлениями провода	4 балла
Хотя бы в одном уравнении верно учтено изменение плотности воздуха с высотой	5 баллов
Получен правильный ответ	10 баллов
Всего	**25 баллов**

4.2.3. Физика. 10–11 классы

Задача 4.2.3.1. Зонд (13 баллов)

Условие

Научный аэростат исследует высокие слои атмосферы планеты на предмет концентрации в них ионов. Для этого на его корпусе установлен зонд в виде двух параллельных друг другу металлических пластин, разделенных небольшим воздушным зазором. Зонд подключен к высокоточным вольтметру и амперметру, непрерывно измеряющим напряжение между пластинами и силу протекающего между ними тока. На малых высотах, на которых воздух практически не ионизирован, зонд является конденсатором с емкостью $C = 50\,\text{пФ}$, не пропускающим ток. На рабочей высоте измерения показали, что зонд является резистором с эффективным сопротивлением $R = 9\,\text{МОм}$. Найдите удельное сопротивление воздуха на этой высоте. Электрическая постоянная $\varepsilon_0 = 8{,}85 \cdot 10^{-12}\,\text{Ф/м}$.

Решение

Емкость плоского воздушного конденсатора задается выражением

$$C = \frac{S\varepsilon_0}{d},$$

где S — площадь его обкладок, а d — ширина зазора между ними.

Удельное сопротивление частично ионизированного разреженного газа заведомо значительно выше, чем металла, поэтому потенциал в пределах одной обкладки зонда можно считать постоянным с высокой точностью. В этом случае сопротивление резистора, образованного ионизированной воздушной прослойкой между ними, задается выражением

$$R = \frac{\rho d}{S} = \frac{\rho\varepsilon_0}{C},$$

где ρ — искомое удельное сопротивление. Отсюда получим окончательно

$$\rho = \frac{RC}{\varepsilon_0} \approx 5{,}08 \cdot 10^7\,\text{Ом} \cdot \text{м}.$$

Ответ:

$$\rho = \frac{RC}{\varepsilon_0} \approx 5{,}08 \cdot 10^7\,\text{Ом} \cdot \text{м}.$$

Примечательно, что этот результат на самом деле не зависит от формы обкладок зонда, но доказательство этого факта выходит за рамки школьной программы.

Критерии оценивания

Верно записана формула для емкости плоского конденсатора	3 балла
Верно записана формула сопротивления резистора	3 балла

Обоснована возможность применения предыдущей формулы	1 балл
Получен правильный ответ	6 баллов
Всего	**13 баллов**

Указание: существует более общее решение, основывающееся на применении теоремы Гаусса, закона Ома в дифференциальной форме и определений емкости и сопротивления. Если оно применено полностью корректно — за задачу следует ставить полный балл. Если часть его обоснована — балл ставится на усмотрение проверяющего из той же общей логики: 6 баллов за правильность ответа и до 7 баллов за обоснованность решения.

Задача 4.2.3.2. Цикл Брайтона (17 баллов)

Условие

Многие типы двигателей, широко применяемые в авиации — газотурбинные, турбореактивные и прямоточные воздушно-реактивные — используют в своей основе термодинамический цикл Брайтона, состоящий из двух изобар и двух адиабат. В цикле 12341 этого рода провели дополнительную изобару 56, «разделившую» его на два меньших цикла 15641 и 52365, КПД которых оказались одинаковы. Найдите КПД этих циклов если известно, что КПД исходного цикла 12341 равняется η.

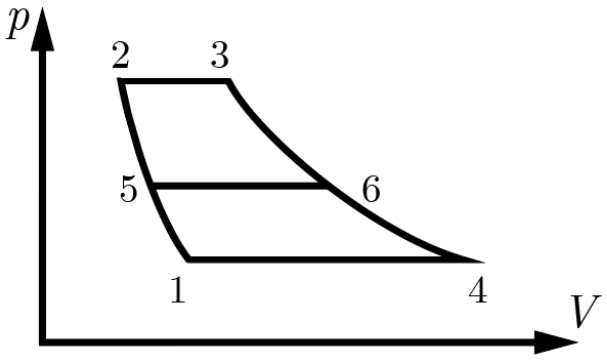


Рис. 4.2.7

Решение

КПД термодинамического цикла удобно найти по формуле

$$\eta = 1 - \frac{Q_{\text{о}}}{Q_{\text{п}}},$$

где $Q_{\text{о, п}}$ — количества теплоты, отданные газом холодильнику и полученное им от нагревателя соответственно.

Рассмотрим сначала исходный цикл 1234. На адиабатических участках цикла газ не получает и не теряет теплоту, соответственно, весь теплообмен рабочего тела с окружающей средой сосредоточен на изобарных участках 2–3 и 4–1. Тогда перепишем КПД цикла в виде

$$\eta = 1 - \frac{Q_{41}}{Q_{23}},$$

где Q_{41}, Q_{23} — модули количества полученной или отданной газом теплоты на соответствующих участках цикла.

Аналогично КПД циклов 15641 и 52365 равны

$$\eta' = 1 - \frac{Q_{41}}{Q_{56}} = 1 - \frac{Q_{56}}{Q_{23}}.$$

Следовательно,

$$\frac{Q_{41}}{Q_{56}} = \frac{Q_{56}}{Q_{23}} \Rightarrow Q_{56} = \sqrt{Q_{41}Q_{23}}.$$

Выражая Q_{41} через КПД исходного цикла: $Q_{41} = Q_{23}(1-\eta)$, получим

$$Q_{56} = Q_{23}\sqrt{1-\eta} \Rightarrow \eta' = 1 - \frac{\cancel{Q_{23}}\sqrt{1-\eta}}{\cancel{Q_{23}}} = 1 - \sqrt{1-\eta}.$$

Ответ: $\eta' = 1 - \sqrt{1-\eta}$.

Критерии оценивания

Верно записано любое определение КПД теплового двигателя	3 балла
Верно записана формула для КПД именно цикла Брайтона через характеристики пронумерованных процессов или состояний	3 балла
Найдена пропорция между количествами теплоты на трех изобарах или эквивалентная ей	4 балла
Получен правильный ответ	7 баллов
Всего	**17 баллов**

Задача 4.2.3.3. Непростой маршрут (20 баллов)

Условие

Исследовательский беспилотный планер начинает свое движение над южным полюсом Земли и по кратчайшей траектории достигает научной станции, расположенной строго на экваторе, вблизи бразильского города Макапа. Быстро проделав необходимые измерения и повернув по малому радиусу, он затем направляется, также по кратчайшей траектории, в следующую станцию, расположенную в Кении, тоже на экваторе; после этого вновь быстро проводит измерения и возвращается по кратчайшей траектории на Южный полюс. При этом линия возвращения беспилотника пересеклась на полюсе с линией его старта под прямым углом.

Определите общее время движения беспилотника, если известно, что все время в пути он двигался равномерно на пренебрежимо малых высотах, и в момент обоих поворотов беспилотника Солнце находилось над ним в зените, но не оказывалось над ним в зените в других точках отрезка Бразилия – Кения. Длина экватора Земли $L = 40\,076$ км, длина меридиана $l = 20\,004$ км.

Решение

Точки поворота беспилотника находятся на экваторе, поэтому траектория беспилотника состояла из двух меридианальных дуг, соединяющих эти точки с полюсом, и одной экваториальной дуги, соединяющей их друг с другом. Суммарная длина меридианальных дуг равна l, так как каждая из них составляет ровно половину меридиана.

Важно, что длина экваториальной дуги x не может быть найдена при помощи теоремы Пифагора или других соотношений евклидовой планиметрии, поскольку движение беспилотника происходит на сфере, и характерные расстояния этого движения сопоставимы с радиусом этой сферы. Вместо этого можно заметить, что длина этой дуги относится к общей длине экватора, так же как разница в широтах между двумя точками поворота относится к 2π, а эта разница в широтах есть не что иное, как угол между меридианами, вдоль которых взлетает и садится беспилотник:

$$\frac{x}{L} = \frac{\pi/2}{2\pi} \Rightarrow x = \frac{L}{4}.$$

Согласно условию, Солнце оказалось в зените в моменты обоих поворотов, но не между ними, при этом на участке Бразилия – Кения беспилотник двигался с запада на восток (навстречу Солнцу). Это значит, что угловая скорость видимого движения Солнца над беспилотником складывалась из угловой скорости его видимого движения, обусловленного вращением Земли и угловой скорости движения беспилотника относительно поверхности планеты, а за время t этого перелета относительно беспилотника Солнце совершило один полный оборот:

$$\frac{2\pi}{t} = \frac{\pi}{2t} + \frac{2\pi}{T},$$

где $T = 1$ сут.

Отсюда

$$t = T\left(1 - \frac{1}{4}\right) \Rightarrow v = \frac{x}{t} = \frac{L}{3T},$$

где v — линейная скорость беспилотника относительно поверхности Земли.

После того как была найдена скорость и длина траектории, легко дать ответ на вопрос задачи:

$$t_\text{о} = \frac{l + x}{v} = 3T\frac{l + L/4}{L} \approx 54\,\text{ч} \approx 1{,}94 \cdot 10^5\,\text{с}.$$

Ответ:

$$t_\text{о} = 3T\frac{l + L/4}{L} \approx 54\,\text{ч} \approx 1{,}94 \cdot 10^5\,\text{с}.$$

Критерии оценивания

Продемонстрировано понимание траектории беспилотника (она изображена или описана) и невозможности использования планиметрии для нахождения ее длины	4 баллов
Верно найдена длина всех участков траектории	3 балла

Верно записан закон сложения линейных или угловых скоростей для экваториального участка маршрута	3 балла
Верно найдено время в пути или скорость на этом участке	4 балла
Получен правильный ответ	6 баллов
Всего	**20 баллов**

Если участник продемонстрировал явное понимание необходимости сложения угловых скоростей для нахождения скорости планера на экваториальном участке, но допустил незначительную алгебраическую ошибку, например, перепутал знак, на этом этапе — за критерии 3 и 4 он может получить до 4 баллов суммарно.

Задача 4.2.3.4. Законы Кеплера (23 балла)

Условие

Первые два закона Кеплера (сформулированные для спутника Земли) гласят:

1. Невозмущенная орбита спутника Земли является эллипсом, в одном из фокусов которого находится центр планеты.
2. За равные промежутки времени радиус-вектор космического аппарата (с началом в центре планеты) заметает равные площади.

Космический аппарат, находящийся на круговой земной орбите радиусом $R = 10\,000$ км, провел маневр расстыковки, разделившись на два независимых отсека, каждый из которых получил небольшое приращение скорости. Для первого отсека это приращение оказалось направлено строго по ходу его орбитального движения, а для второго — строго против этого хода. При этом орбиты спутников изменились так, что скорость каждого из спутников в перигее оказалась на $k = 1\%$ раз больше, чем в апогее. Найдите расстояние между апогеем первого спутника и перигеем второго.

Решение

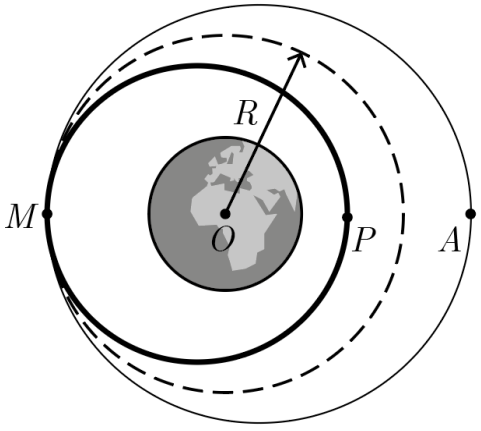


Рис. 4.2.8

На рис. 4.2.8 (не в масштабе) тонкой линией изображена орбита первого отсека, толстой — второго, пунктиром средней толщины — их общая изначальная круговая орбита. M — точка расстыковки спутников, A — апогей орбиты первого спутника, P — перигей орбиты второго. Расстояние AP является искомой величиной задачи. Это расстояние сильно преувеличено для наглядности.

При движении по круговой орбите вектор скорости спутника всегда ортогонален его радиус-вектору, проведенному из центра планеты. Векторы $\Delta\vec{v}$ приращения скоростей обоих отсеков коллинеарны вектору их орбитальной скорости в момент расстыковки, это значит, что и сразу после маневра, когда орбиты спутников уже изменились, их мгновенные скорости направлены перпендикулярно радиус-векторам и, следовательно, векторам действующих на них сил тяжести.

Из этого вытекает, что в момент расстыковки сила тяжести не совершает над отсеками работы, поэтому и не изменяются кинетические энергии отсеков, а это возможно только в моменты максимального сближения (перигей) и максимального отдаления (апогей) спутника от планеты. При этом для первого спутника скорость оказалась немного выше скорости его движения по круговой орбите и, следовательно, при постоянном центростремительном ускорении центр кривизны его траектории сместился немного дальше центра планеты, а значит, точка расстыковки M является для него перигеем. Для второго, наоборот, скорость оказалась несколько меньше скорости его движения по круговой орбите, поэтому точка M оказалась для него апогеем.

Рассмотрим малый отрезок времени t вблизи перигея или апогея орбиты спутника. В этот момент мгновенная скорость спутников, как было установлено, ортогональна радиус-вектору, поэтому площадь S заметаемого им сектора может быть найдена по формуле

$$S = \frac{vtr}{2},$$

где r — расстояние от спутника до центра планеты. Поэтому по второму закону Кеплера, скорости v_A и v_P, которые имеет любой спутник в апогее и перигее соответственно связаны с расстояниями r_A, r_P от центра планеты до этих точек его орбиты соотношениями

$$v_A r_A = v_P r_P.$$

Следовательно, расстояния r_P и r_A для двух отсеков находятся в пропорции

$$\frac{r_{A1}}{r_{P1}} = \frac{v_{P1}}{v_{A1}} = 1 + k = \frac{v_{P2}}{v_{A2}} = \frac{r_{A2}}{r_{P2}},$$

где индексы $1, 2$ указывают на принадлежность величины к орбите первого или второго спутников соответственно. Но $r_{P1} = r_{A2} = R$ и, следовательно,

$$r_{A1} = R(1 + k); \quad r_{P2} = R/(1 + k).$$

Тогда окончательно

$$l = r_{A1} - r_{P2} = R\left(1 + k - \frac{1}{1 + k}\right) \approx 199\,\text{км}.$$

Ответ:

$$l = R\left(1 + k - \frac{1}{1 + k}\right) \approx 199\,\text{км}.$$

Критерии оценивания

Верно изображены или описаны словами с указанием ключевых для задачи точек орбиты двух отсеков после расстыковки	5 баллов
Положение ключевых точек на этих орбитах физически обосновано	5 баллов
Из второго закона Кеплера или закона сохранения момента импульса найдена связь между модулем скорости и радиус-вектором в разных точках эллиптической орбиты	6 баллов
То же утверждение записано без доказательства	3 балла
Получен правильный ответ	7 баллов
Всего	**23 балла**

Частичный успех на каждом из шагов может оцениваться в долю от максимального балла за соответствующий шаг на усмотрение проверяющего. Например, верно найдена орбита только одного из сегментов, получен верный ответ в общем виде, но при подстановке чисел допущена арифметическая ошибка и т. п. Аналогично в остальных задачах.

Задача 4.2.3.5. Солнечный ожог (27 баллов)

Условие

Космический телескоп находится на околоземной орбите и изготовлен по схеме Ричи – Кретьена (см. рис. 4.2.9). Она состоит из кольцевого главного зеркала с оптической силой $D_1 = 0{,}2\,\text{дптр}$ и внешним диаметром $d_1 = 1{,}2\,\text{м}$ и вторичного зеркала с оптической силой $D_2 = -0{,}1\,\text{дптр}$ и диаметром $d_2 = 0{,}4\,\text{м}$, расположенном на расстоянии $l = 3\,\text{м}$ от главного. Внутренний диаметр главного зеркала совпадает с диаметром вторичного.

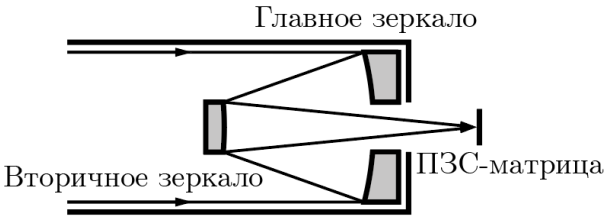


Рис. 4.2.9

Оптическая система отъюстирована таким образом, что пучок световых лучей, приходящих параллельно главной оптической оси телескопа фокусируется ею точно в центр светочувствительной ПЗС-матрицы, расположенной за главным зеркалом. Орбита телескопа оказалась ориентирована таким образом, что через некоторое время телескоп окажется сориентирован строго на Солнце, и если ничего не предпринять — фокусировка столь мощного источника света выведет ПЗС-матрицу из строя. К сожалению, за годы эксплуатации жесткое космическое излучение нарушило работу маневровых двигателей и электромотора, контролирующего крышку объектива,

поэтому единственной возможностью спасти матрицу оказались юстировочные сервоприводы, способные смещать ПЗС-матрицу вдоль оптической оси телескопа.

На какое минимальное расстояние необходимо сместить матрицу вдоль этой оси, если общая плотность потока мощности, попадающего на телескоп от Солнца и фокусируемой его оптикой, равна $I = 1200$ Вт/м^2, а предельная плотность мощности, попадание которой на ПЗС-матрицу не приведет к ее выгоранию равна $I' = 3$ Вт/мм^2?

Решение

Общий световой поток, попадающий на матрицу, равен произведению плотности светового потока на площадь его поперечного сечения. В частности, этот поток можно выразить через площадь главного зеркала и начальную плотность потока I, либо через площадь пятна, в которое фокусируется кольцевой световой пучок внешнего диаметра d_1 и внутреннего диаметра d_2, отраженный главным зеркалом телескопа и критическую допустимую плотность потока I' соответственно. Обозначив диаметр этого пятна d, запишем условие «выживания» матрицы в виде

$$\frac{\pi}{4}d^2 I' = \frac{\pi}{4}d_1^2 I \Rightarrow d = d_1\sqrt{\frac{I}{I'}}.$$

Оба эти пятна имеют форму геометрически подобных колец, поэтому нет необходимости учитывать их внутренние радиусы.

Чтобы выразить диаметр d, рассмотрим подробнее ход лучей в телескопе. Главным зеркалом, оптический центр которого обозначен на рис. 4.2.10 как O_1, они собираются в точке F_1, такой, что $O_1F_1 = 1/D_1$.

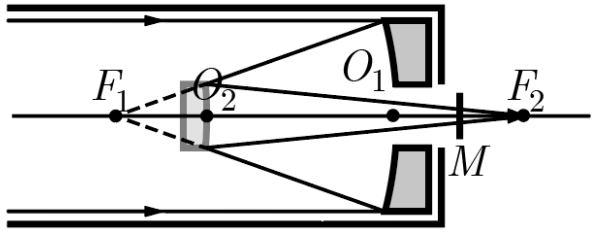


Рис. 4.2.10

Эта точка служит мнимым источником для вторичного зеркала (с оптическим центром в точке O_2), поэтому расстояние O_2F_2 может быть найдено по формуле тонкой линзы (которая для выпуклых и вогнутых зеркал выполняется с точностью до замены области пропускания лучей на область их отражения):

$$D_2 = -\frac{1}{O_2F_1} + \frac{1}{O_2F_2} \Rightarrow O_2F_2 = \left(D_2 + \frac{1}{O_2F_1}\right)^{-1} = \left(D_2 + \frac{1}{1/D_1 - l}\right)^{-1}.$$

Из подобия треугольников, образованных лучами, отраженными от вторичного зеркала, плоскостью этого зеркала и ПЗС-матрицей, следует

$$\frac{d_2}{d} = \frac{O_2F_2}{MF_2},$$

где M — точка расположения центра матрицы.

Таким образом, искомый отрезок F_2M равен

$$MF_2 = O_2F_2\frac{d}{d_2} = \left(D_2 + \frac{1}{1/D_1 - l}\right)^{-1} \cdot \frac{d_1}{d_2}\sqrt{\frac{I}{I'}} = 15\,\text{см}.$$

Ответ:

$$x = \left(D_2 + \frac{1}{1/D_1 - l}\right)^{-1} \cdot \frac{d_1}{d_2}\sqrt{\frac{I}{I'}} = 15\,\text{см}.$$

Критерии оценивания

Продемонстрировано понимание связи между плотностью потока и площадью светового пятна	5 баллов
Верно найден фокус главного зеркала	3 балла
Верно записана формула тонкой линзы с учетом мнимости источника вторичного зеркала	6 баллов
Формула тонкой линзы записана, но с ошибкой в знаках	3 баллов
Верно использовано подобие необходимых для решения задачи треугольников	4 балла
Получен правильный ответ	9 баллов
Всего	**27 баллов**

4.3. Инженерный тур

4.3.1. Общая информация

Современная космическая отрасль активно развивает технологии малых спутников, которые стали ключевым инструментом для оперативного получения данных о состоянии Земли. Эти аппараты используются в широком спектре задач: от мониторинга климатических изменений и состояния экосистем до быстрого реагирования на чрезвычайные ситуации, таких как природные катастрофы.

В этом году участникам Олимпиады предлагается стать частью команды разработчиков нового поколения малых спутников для исследования Земли и создать свой аппарат — перспективную платформу для получения оперативной информации о климате, экосистемах и урбанистических процессах. Его основная миссия включает три ключевых этапа:

1. ориентация в надир,
2. съемка данных высокого качества,
3. передача их на наземную станцию.

4.3.2. Легенда задачи

Участники Олимпиады — это команда инженеров, работающих над созданием малого спутника для дистанционного зондирования Земли. Аппарат должен быть компактным, энергоэффективным и способным выполнять задачи в условиях ограниченных ресурсов. Он разрабатывается на базе конструктора Introsat, поэтому для реализации проекта задействуются современные технологии и компоненты.

Для обеспечения точной ориентации спутника в надир используются инфракрасные датчики, которые позволяют определять положение Земли относительно аппарата. Это гарантирует стабильное наведение и минимизирует искажения при съемке. Главный инструмент для сбора данных — камера высокого разрешения, способная фиксировать детализированные изображения поверхности Земли.

Основные задачи, которые решают разработчики:

- создание системы ориентации,
- настройка камеры для съемки данных высокого качества,
- проектирование системы передачи данных на наземную станцию,
- расчет оптимальной орбиты для выполнения миссии.

4.3.3. Требования к команде и компетенциям участников

Команда заключительного этапа должна состоять из трех или четырех человек. Если в заключительный этап по итогам второго отборочного этапа проходит команда менее, чем из трех участников, и если нет со стороны команды возражений, то она будет расформирована, а ее члены должны вступить в другие команды и/или образовать новые, численностью не менее, чем из трех человек.

Роли в команде могут быть любыми, но для выполнения заданий заключительного этапа выделяют четыре сбалансированных направления (рекомендуется, чтобы каждым из них занимался выделенный участник):

- **Радиотехника**: сформированное представление об основах радиосвязи, умение работать с радиомодулями и организовывать каналы связи с использованием различных технических решений. Рекомендовано знакомство с ПО SDR#, GNU Radio.
- **Программирование (полезной нагрузки)**: разработка и создание ПО для обработки данных матричных датчиков и взаимодействие с другими системами КА.
- **Программирование микроконтроллеров/разработка спутниковых систем**: написание управляющих программ ко всем прототипам; решение задач потребует знакомства с STM32CubeIDE, умения работать с кодом на языке C и базовых понятий об алгоритмах управления.
- **Физика (баллистика):** решение задач по проектированию миссий на орбите, для чего в первую очередь необходимо знать основы небесной механики. Рекомендовано знакомство с синтаксисом языка JavaScript для написания алгоритмов в симуляторе.

4.3.4. Оборудование и программное обеспечение

Стенд

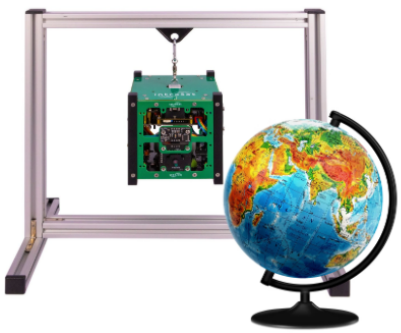


Рис. 4.3.1

В качестве небольшого настольного стенда командам предлагается использовать следующие компоненты (см. рис. 4.3.1):

- стенд с установленным на подвесе спутником;
- макет, имитирующий земной шар.

На испытаниях заключительного этапа при установке на подвес спутнику будет придаваться начальная закрутка. Он должен:

1. стабилизироваться;
2. сориентироваться на Землю;
3. сделать снимок;
4. передать снимок на наземную станцию.

Первый день

Программист (платформа спутника)

Первый день посвящен сборке оборудования и отработке материала удаленного практикума, а также написанию алгоритма стабилизации.

В работе можно использовать проект STM32CubeIDE, который получился у команды в результате отработки удаленного практикума, или запросить у организаторов сборку стартового проекта.

День делится на два этапа (см. таблицу 4.3.1).

Таблица 4.3.1

	Баллы, которые можно получить за сдачу задачи	**Первый день: 9:00–13:00**	**Первый день: 14:00–18:00**	**Второй день: 9:00–13:00**
Первая половина дня: 9:00–13:00	Настроить беспроводную передачу информации по UART и беспроводной способ прошивки микроконтроллера	2	1	0
	Задать управление двигателем, продемонстрировать изменение направления и скорости вращения	2	1	0
	Считать данные с датчика позиционирования (гироскоп) так, как это делалось на удаленной работе	2	1	0
Вторая половина дня: 14:00–18:00	Реализовать алгоритм стабилизации спутника	4	4	2
	Итого баллов	10		

Для выполнения задач первой половины дня советуем использовать методические указания:

- первое видео по удаленной работе с конструктором IntroSat: https://rutube.ru/video/1ac912e42f71b7edb7a1eb4c3c41e7b9/?r=wd;
- презентация по удаленной работе: https://disk.360.yandex.ru/i/YlAlXJvmbRAjSA.

Алгоритм стабилизации работает с рассогласованием, т. е. с ошибкой между желаемым значением параметра и текущим. Обычно рассогласование обозначают буквой e:

$$e = x_{\text{текущее}} - x_{\text{идеальное}},$$

где x — угловая скорость; в данном случае: $x_{\text{текущее}}$ — угловая скорость спутника; $x_{\text{идеальное}}$ — необходимая угловая скорость спутника (0).

Для того чтобы управлять изменением скорости вращения спутника, необходимо изменить скорость маховика.

Если воспользоваться законом сохранения момента импульса, то получим, что изменение скорости маховика пропорционально изменению скорости спутника, в данном случае для реализации алгоритма нужно найти коэффициент пропорциональности:

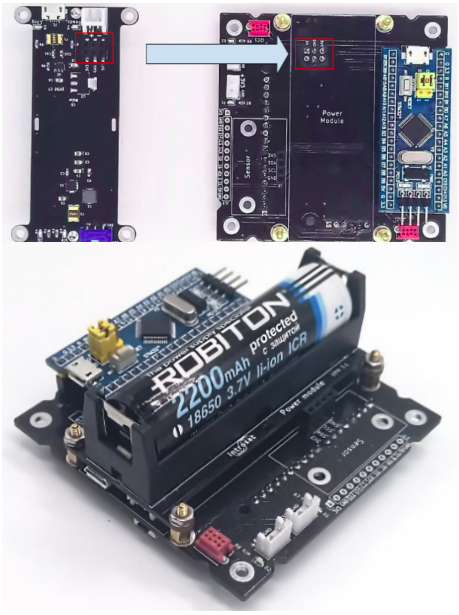
$$w_{wheel} += Kp \cdot e.$$

Сборка

Прежде чем приступить к работе с микроконтроллером, необходимо осуществить сборку основных плат конструктора IntroSat.

Таблица 4.3.2

№	Сборка плат	
1	Установите плату Blue Pill на материнскую плату. Важно! Разъем microusb на плате BluePill должен находиться над надписью USB SIDE.	

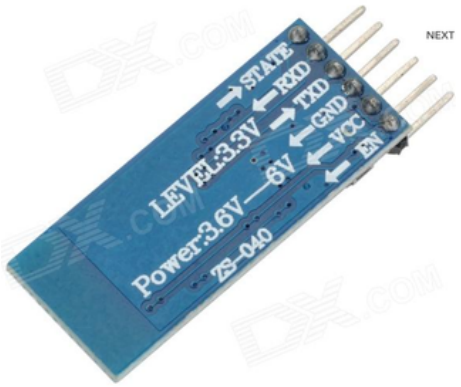
2	Установите модуль Bluetooth в соответствующий разъем.	
3	Закрепите 4 стойки PCHSN5 длиной по 5 мм на материнской плате при помощи винтов M2.5x4 с полукруглой головкой.	
4	Установите аккумулятор в модуль питания, при необходимости закрепите его пластиковой стяжкой через ушки в плате. Модуль питания установите на материнскую плату так, чтобы шестиконтактные разъемы сопрягались друг с другом. Закрепите гайками модуль питания.	

После этого можно проверить, что все подключено верно, сдвинув выключатель (белый ползунок) на модуле питания. Должны загореться светодиоды на плате Blue Pill и bluetooth-модуле.

Обмен данными по UART и беспроводная прошивка

К материнской плате подключается ведомый bluetooth-модуль, а ведущий соединяется с USB-UART конвертером по схеме, приведенной в таблице 4.3.3.

Таблица 4.3.3

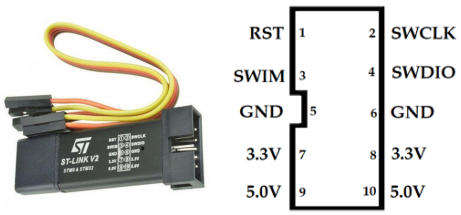
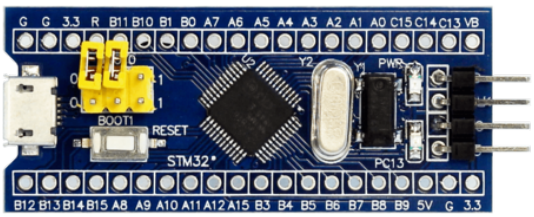
Bluetooth-модуль	**Usb-uart-конвертер**
	
RX (RXD) >--- TX (TXD) >--- GND >--- 5V >---	---> TX (TXD) ---> RX (RXD) ---> GND ---> VCC

Необходимо запрограммировать микроконтроллер на передачу текстового сообщения через UART и считать его из COM-порта компьютера, к которому подключен USB-UART-конвертер с ведущим bluetooth-модулем.

После этого нужно добавить код для беспроводной прошивки микроконтроллера через UART. Код загружается через ST-LINK, после чего появится возможность загружать последующие программы по беспроводному соединению. Убедитесь в работоспособности такого метода загрузки.

В таблице 4.3.4 условно показан способ коммутации программатора ST-Link V2 и платы с микроконтроллером Blue Pill.

Таблица 4.3.4

ST-Link V2*	**Blue Pill**
	
(2) SWCLK >--- (4) SWDIO >--- (5) или (6) GND >--- (7) или (8) 3.3V >---	---> SCK ---> DIO ---> GND ---> 3.3V

**Примечание: распиновка может отличаться от приведенной, однако важно соединить правильно соответствующие надписи.*

После подключения программатора к микроконтроллеру и затем к usb-порту ПК станет возможна загрузка программы в плату. Для загрузки программы в плату нажмите на значок  в верхней панели меню.

После этого, возможно, ST-Link попросит обновления. Для этого во вкладке меню **Help** есть соответствующая опция. Если ST-Link не просит обновления, просто пропустите этот шаг.

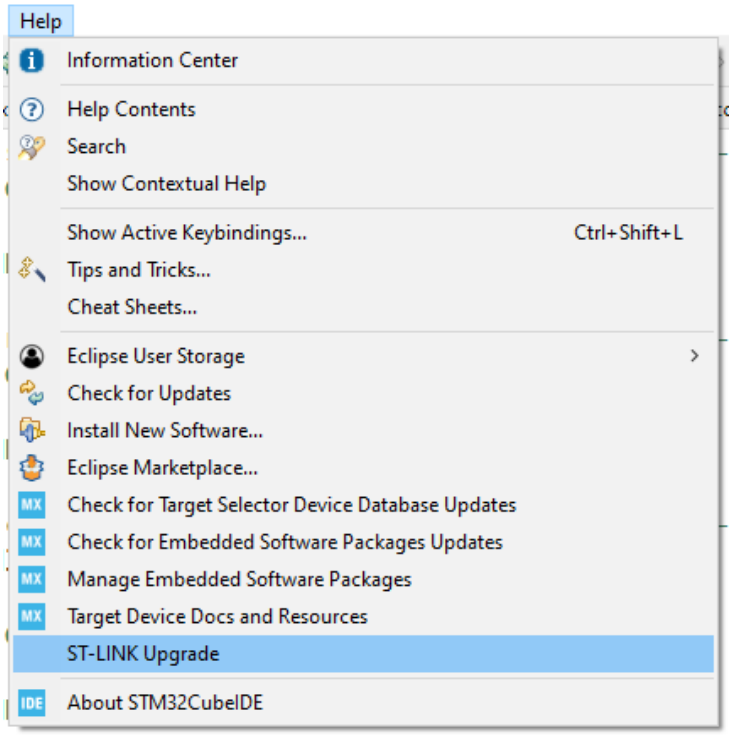


Рис. 4.3.2

Выберите ST-LINK Upgrade. Откроется новое окно. В нем выберите среди устройств ST-LINK/V2, а затем — **Open in update mode**. После этого нажмите **Upgrade**.

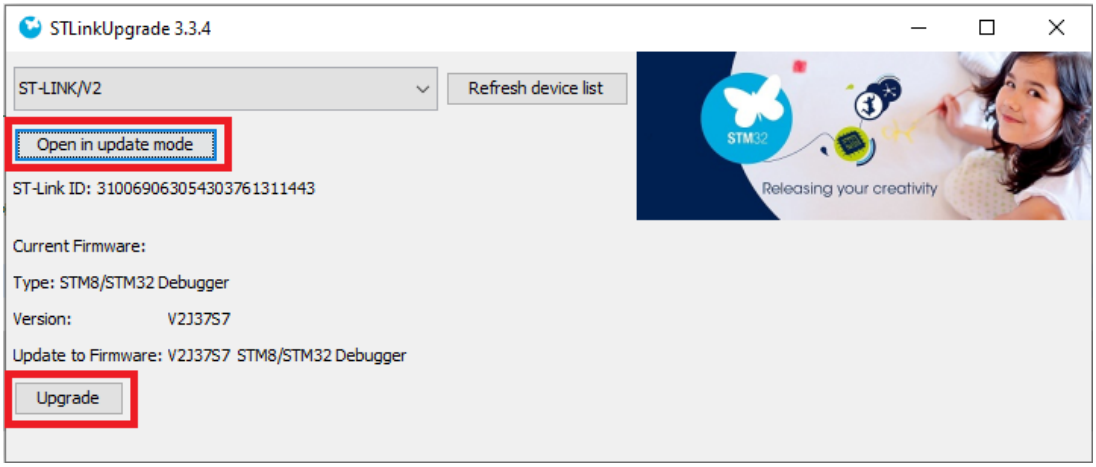


Рис. 4.3.3

Иногда требуется переподключить программатор, чтобы утилита по его обновлению идентифицировала его. Для загрузки программы в плату нажмите на значок

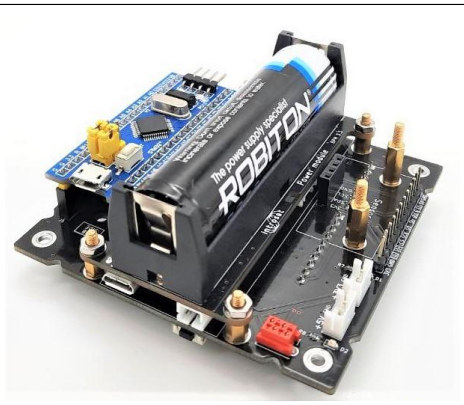
 в верхней панели меню.

Примечание. Помимо режима загрузки существует также режим отладки. Он бывает крайне полезен, если где-то в коде закралась ошибка, но ее не удается быстро найти. Для перехода в режим отладки (debug) можно нажать на значок жучка  в верхней панели меню. О том, как пользоваться режимом отладки можно посмотреть, например, здесь: https://youtu.be/BVC7KaUNCS8. На очном финале весьма полезно будет уметь пользоваться данным режимом.

Работа с датчиком позиционирования

Закрепите датчик позиционирования на материнской плате конструктора Intro-Sat.

Таблица 4.3.5

1	Закрепите две стойки PCHSN11 (длиной 11 мм) на материнской плате при помощи винтов M2.5x4 с полукруглой головкой	
2	Установите датчик позиционирования в соответствующий разъем и зафиксируйте его гайками	

Работа с датчиком позиционирования происходит по интерфейсу I2C. Необходимо считать значения кватернионов с гироскопа и магнитометра, как рассказывалось в видео по удаленной работе, и вывести их в COM-порт компьютера по UART.

Работа с двигателем

Подготовьте к работе плату маховика (см. таблицу 4.3.6).

Таблица 4.3.6

1	Установите маховик на адаптер на оси двигателя с помощью пары винтов M2.5x6 с выпуклой головкой	
2	Прикрутите плату маховика к нижней грани куба винтами с потайной головкой M2.5x12 и зафиксируйте гайками	

Подключите модуль маховика к материнской плате с помощью шлейфа I2C и провода питания. Для этого можно использовать любые из доступных разъемов MicroMatch на материнской плате и модуле маховика. Питание на двигатель маховика подается от 5-вольтового вывода материнской платы.

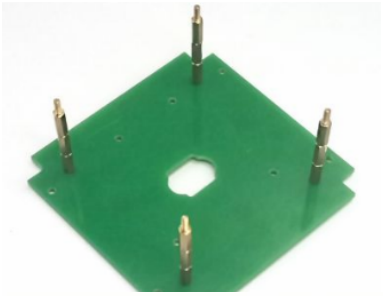
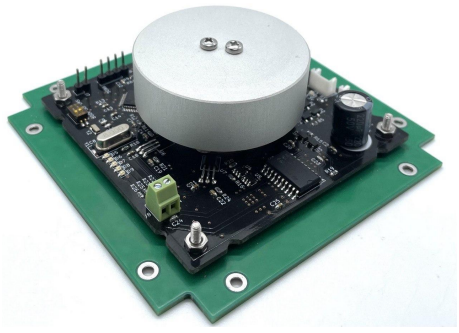
Финальная сборка

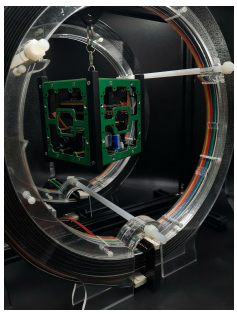
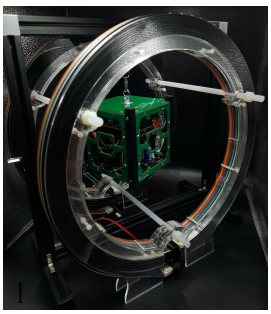
Перед финальной сборкой рекомендуется проверить правильность всего подключения, чтобы избежать необходимости пересборки. При этом удобным будет оставить ST-Link подключенным к Blue Pill, оставив его снаружи: вероятно, понадобится перепрошивать или отлаживать микроконтроллер через него.

Таблица 4.3.7

Сборка корпуса		
1	Возьмите два рельса и установите в выемки перекладины с пазом.	

2	В получившиеся углубления на стыке рельса и перекладины установите перекладины с торцевым отверстием так, чтобы поперечные отверстия в этих перекладинах были ориентированы перпендикулярно рельсу (они используются для установки боковых панелей). Закрепите угловое соединение винтами М2.5х12 с потайной головкой.	
3	Установите две оставшиеся перекладины и два оставшихся рельса аналогичным образом и окончательно соберите конструкцию при помощи винтов М2.5х12.	

4	Установите собранную материнскую плату на верхнюю панель с помощью межплатных стоек PCHSN-11 (для этого нужно будет соединить стойки между собой по 3 шт.) и гаек.	 
Сборка и подключение		
5	Закрепите верхнюю панель на раме с помощью четырех винтов M2.5x6 с потайной головкой.	
6	Закрепите плату на нижней панели с помощью винтов M2.5x12 с потайной головкой, используя дополнительные гайки в качестве прокладок между панелью и платой.	

7	Прикрепите нижнюю панель к раме с помощью винтов M2.5x6 с потайной головкой.	
8	Подключите модуль маховика к материнской плате с помощью шлейфа I2C и провода питания. Для этого можно использовать любые из доступных разъемов MicroMatch на материнской плате и модуле маховика. Питание на двигатель маховика подается от 5-вольтового вывода материнской платы. Боковые грани можно установить по желанию.	
Конечная сборка		
9	Подвесьте спутник на подвес.	 

Решение

Ниже представлено решение первой половины дня.

C++

```
/* USER CODE BEGIN Includes */
// Подключение необходимых библиотек
#include "string.h"
#include "stdio.h"
#include "IS_Bluetooth.h"
#include <GyroscopeV2.h>
```

```
#include <MagnetometerV2.h>
#include <MotorFlyWheel.h>
#include <Quaternion/Quaternion.h>
/* USER CODE BEGIN PM */
// Использование пространства имён интросата
using namespace IntroSatLib;
/* USER CODE BEGIN 0 */
// Необходимо создать устройства
GyroscopeV2 mpu(&hi2c1, 0x6B);
MotorFlyWheel flyWheel(&hi2c1, 0x38);
  /* USER CODE BEGIN 2 */
  // Строка для вывода информации при отладке
  char result[100] = {0};
  char in = 0;
  // Инициализация гироскопа и установка нулевой скорости маховика
  mpu.Init();
  flyWheel.NeedSpeed(0);
    /* USER CODE BEGIN 3 */
    HAL_UART_Receive(&huart1, &in, 1, 1000);
    if(in == 'b'){ in= 0; enter_bootloader(); }
    float x = mpu.X();
    float y = mpu.Y();
    float z = mpu.Z();
    sprintf(result, "Angle speed\t\tx:%f\ty:%f\tz:%f\r\n", x, y, z);
    HAL_UART_Transmit(&huart1, (uint8_t*)result, strlen(result),
     ↪  1000);
    HAL_Delay(250);
    motor.NeedSpeed(2000);
    HAL_Delay(1000);
    motor.NeedSpeed(-2000);
    HAL_Delay(1000);
   }
  /* USER CODE END 3 */
```

Ниже представлено решение второй половины дня.

C++

```
float last_err = 0;
float Integrator = 0;
float Kp = 400;
float Ki = 0.3;
float Kd = 0;
/* USER CODE END 0 */
/* USER CODE BEGIN 3 */
/* предыдущий код, тут только изменения */
/* sprintf(result, "Angle speed\t\tx:%f\ty:%f\tz:%f\r\n", x, y, z);
   HAL_UART_Transmit(&huart1, (uint8_t*)result, strlen(result), 1000);
   HAL_Delay(250);
   motor.NeedSpeed(2000);
   HAL_Delay(1000);
   motor.NeedSpeed(-2000);
   HAL_Delay(1000); */

   float needSpeed = 0;
   float err = z - needSpeed;
   last_err = err;
   Integrator = Integrator + err * 0.1;
   if (Integrator > 4.0f) { Integrator = 4.0f; }
   if (Integrator < -4.0f) { Integrator = -4.0f; }
```

```
23      float P = Kp * err;
24      float D = Kd * (err - last_err) / T;
25      float I = Integrator * Ki;
26      float PID_rez = P + I + D;
27      float Raw_Motor_Speed = motor.CurrentSpeed();
28      float Motor_Speed = (Raw_Motor_Speed + PID_rez);
29
30      if (Motor_Speed > 3000) { Motor_Speed = 3000; }
31      if (Motor_Speed < -3000) { Motor_Speed = -3000; }
32      motor.NeedSpeed(Motor_Speed);
33      HAL_Delay(1000);
```

Программист (полезная нагрузка)

Этот день направлен на подготовку сенсорики, необходимой для ориентации аппарата. Участникам нужно научиться получать информацию с матричного ИК-датчика AMG8833 и реализовать протокол для формирования управляющих сигналов на маховик. Задачи программиста ПН и программиста платформы спутника плотно связаны.

Начало работы

Перед началом работы необходимо спаять схему для отказоустойчивой работы датчика (см. рис. 4.3.4).

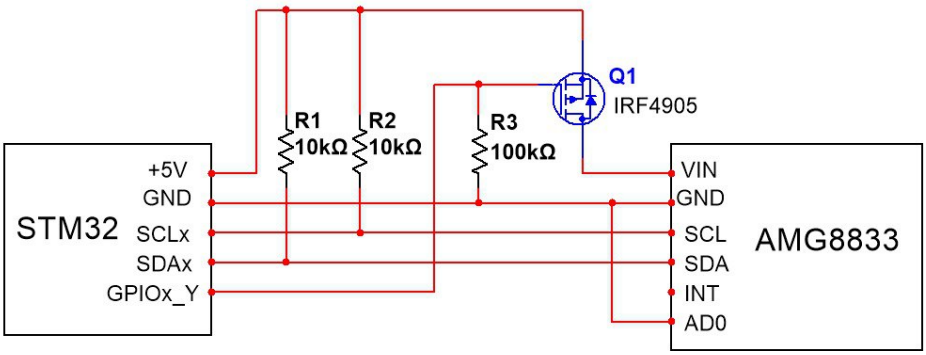


Рис. 4.3.4. Схема для отказоустойчивой работы датчика: SCLx и SDAx — контакты SCL и SDA I2C под номером x, GPIOx_Y — любой свободный контакт, настроенный на выход

Будьте внимательны: AMG8833 подключается к МК, отвечающему за ориентацию (тот, что установлен на материнской плате). Кроме того, на плате расширения устанавливается еще один МК, который отвечает за работу с камерой, поэтому для него нужно подготовить посадочное место.

Чтение данных

Для чтения данных можно воспользоваться документацией на датчик из открытых источников или использовать способ, реализованный в библиотеке IntrosatLib.

Поддерживаемые инструкции см. ниже.

```
IRCamera amg(&hi2c1); // создание объекта для работы с
amg.useForceReset(GPIOx, GPIO_PIN_Y); // настройка контакта для сброса
amg.useMirrored(); // использовать если модуль расположен так:
```

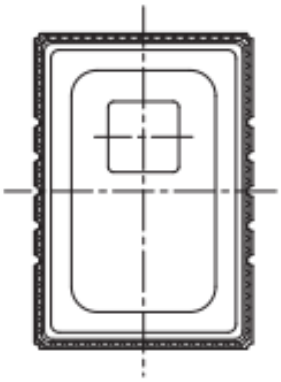


Рис. 4.3.5

```
amg.Init() // инициализация модуля
uint8_t res = amg.Read(); // возвращает 0, если данные правильно прочитались.
float value = amg.getPixel(x, y); // получение градусов по цельсию, x - 0 самая левая часть снимка, x - 7 самая правая часть снимка, y - 0 самая верхняя часть снимка, y - 7 самая нижняя часть снимка.
int16_t rawValue = amg.getPixelRaw(x, y); //получение сырых данных.
```

Пример кода

C++

```
amg.useForceReset(GPIOX, GPIO_PIN_Y);
amg.useMirrored();
amg.Init();
/*USER CODE END 2*/

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if (amg.Read()) { continue; }
    for (uint8_t y = 0; y < 8; y++)
    {
        for (uint8_t x = 0; x < 8; x++)
        {
            char data[10];
            uint32_t len = sprintf(data, "%0.2f ", amg.getPixel(x, y));
            HAL_UART_Transmit(&huart2, (uint8_t*)data, len, 100);
        }
        char next = '\n';
         HAL_UART_Transmit(&huart2, (uint8_t*)&next, 1, 100);
        }
```

```
    }
}
/* USER CODE END WHILE */
```

На выходе необходимо получать массив температур 8×8. Для проверки полученных данных можно реализовать скрипт на языке Python, визуализирующий снимок.

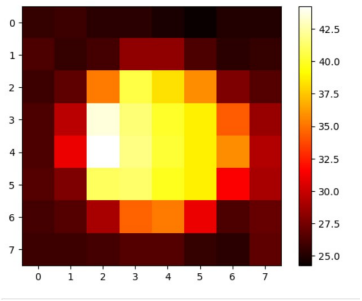


Рис. 4.3.6

Обработка данных

Для наведения с использованием показаний ИК-датчика необходимо создать протокол, сообщающий о том, в каком месте кадра находится Земля (теплый шар, имитирующий ее), чтобы затем отдавать корректировки в программу управления маховиками. С этой целью рекомендуется реализовать следующую логику, а именно: функция оценки положения должна обрабатывать полученный снимок и в зависимости от его содержания возвращать:

- 0 — если корректировки не требуются;
- положительное число (цифру) — если для корректировки необходимо вращать спутник по часовой стрелке;
- отрицательное число (цифру) — если для корректировки необходимо вращать спутник против часовой стрелки;
- большую по модулю константу (например, 127) — если Земли нет в кадре.

Более конкретно протокол общения лучше обсудить с программистом платформы спутника, так как в будущем он будет работать с этой функцией.

Решение

C++

```
//часть кода цикла while(1)
    using namespace IntroSatLib;
    IRCamera amg(&hi2c2);
    amg.Init();
    amg.useForceReset(GPIOA, GPIO_PIN_11);
    float IRData[8][8];
    int command = 0;
    while (1) {
        if (amg.Read() == 0) { continue }
```

```
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                IRData[i][j] = amg.getPixel(j, i);
            }
        }
        command = findCenter();
    }
    // конец части кода цикла while(1)
int findCenter() {
    int edgeCount = 0;
    int leftEdgePosition = 0;
    int rightEdgePosition = 0;
    for (int j = 0; j < 6; j++) {
        if (IRData[5][j + 1] - IRData[5][j] >= 5){
            edgeCount++;
            rightEdgePosition = j;
        } else if (IRData[5][j + 1] - IRData[5][j] >= -5){
            edgeCount++;
            leftEdgePosition = j + 1;
        }
    }
    if (edgeCount == 0) {
        return -127;
    } else if (edgeCount == 1 && leftEdgePosition != 0){
        return 1;
    } else if (edgeCount == 1 && rightEdgePosition != 0){
        return -1;
    } else if (edgeCount == 2){
        if ((leftEdgePosition) == (7 - rightEdgePosition)){
            return 0;
        } else if ((leftEdgePosition) > (7 - rightEdgePosition)){
            return 1;
        } else if ((leftEdgePosition) < (7 - rightEdgePosition)){
            return -1;
        }
    }
}
```

Инженер связи

В течение первого дня необходимо научиться передавать данные по радиоканалу. Для этого следует:

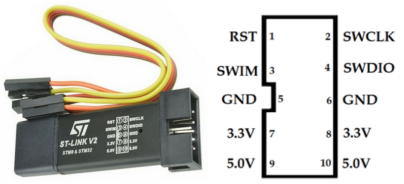
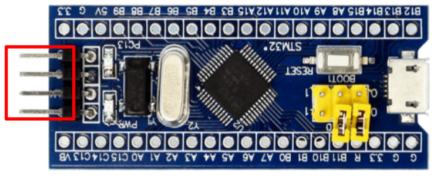
1. подключить передатчик CC1101 к STM32;
2. научиться передавать через него данные;
3. собрать минимальный граф по обработке радиосигнала для получения данных на Земле.

1. **Подключение программатора**

   Код прошивается при помощи программатора ST-Link V2.

   В таблице 4.3.8 условно показан способ коммутации программатора ST-Link V2 и платы с микроконтроллером Blue Pill.

Таблица 4.3.8

ST-Link V2*	**Blue Pill**
	
(2) SWCLK >---	---> SCK
(4) SWDIO >---	---> DIO
(5) или (6) GND >---	---> GND
(7) или (8) 3.3V >---	---> 3.3V

**Примечание: распиновка может отличаться от приведенной, однако важно соединить правильно соответствующие надписи.*

2. **Подключение передатчика CC1101**

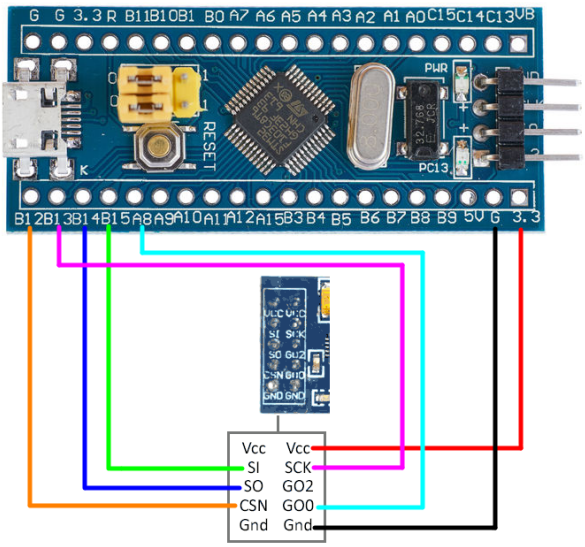


Рис. 4.3.7

Подключение передатчика осуществляется по следующей схеме (см. рис. 4.3.7, таблицу 4.3.9).

Таблица 4.3.9

CC1101	**BluePill**
VCC	3.3
GND	G
SI	PB15

CC1101	BluePill
SO	PB14
CSN	PB12
SCK	PB13
GD0	PA8

3. Подготовка прошивки

Рекомендуем использовать единый начальный проект, познакомиться с которым можно по ссылке: https://disk.360.yandex.ru/d/qDrmWDJBvF7ZaA/SS2025Final-master.zip.

3.1. Создание объекта радиопередатчика

В стартовом объекте уже есть строки, связанные с созданием объекта передатчика, теперь их необходимо раскомментировать.

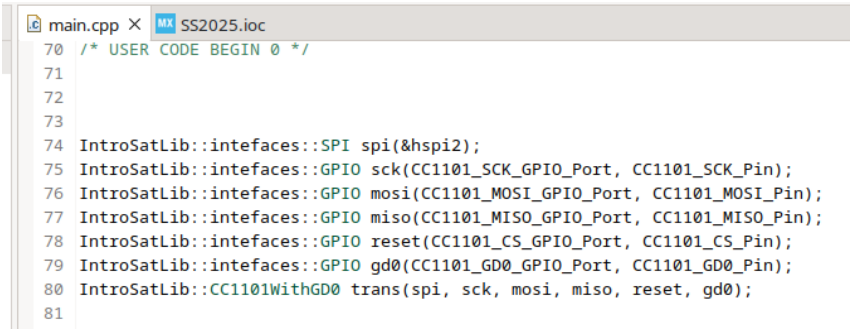


Рис. 4.3.8

3.2. Инициализация передатчика

Далее следует инициализировать созданный объект передатчика, задав ему настройки. Это можно сделать в секции `User Code 2`, которая находится перед бесконечным циклом в функции `main`. Ниже приведен пример конфигурации передатчика.

C++

```
trans.Init();                   // must be set to initialize the cc1101!
trans.setCCMode(1);             // set config for internal transmission mode.
trans.setModulation(0);         // set modulation mode. 0 = 2-FSK, 1 = GFSK, 2 = ASK/OOK, 3 = 4-FSK, 4 = MSK.
trans.setMHZ(<частота в МГц>);          // Here you can set your basic frequency. The lib calculates the frequency automatically (default = 433.92).The cc1101 can: 300-348 MHZ, 387-464MHZ and 779-928MHZ. Read More info from datasheet.
trans.setDeviation(47.60);      // Set the Frequency deviation in kHz. Value from 1.58 to 380.85. Default is 47.60 kHz.
trans.setChannel(0);            // Set the Channelnumber from 0 to 255. Default is cahnnel 0.
```

```
7  trans.setChsp(199.95);       // The channel spacing is
   ↪ multiplied by the channel number CHAN and added to the
   ↪ base frequency in kHz. Value from 25.39 to 405.45.
   ↪ Default is 199.95 kHz.
8  trans.setRxBW(812.50);       // Set the Receive Bandwidth
   ↪ in kHz. Value from 58.03 to 812.50. Default is 812.50
   ↪ kHz.
9  trans.setDRate(0.1);         // Set the Data Rate in
   ↪ kBaud. Value from 0.02 to 1621.83. Default is 99.97
   ↪ kBaud!
10 trans.setPA(10);             // Set TxPower. The following
   ↪ settings are possible depending on the frequency band.
   ↪ (-30  -20  -15  -10  -6    0    5    7    10   11
   ↪ 12) Default is max!
11 trans.setSyncMode(2);        // Combined sync-word
   ↪ qualifier mode. 0 = No preamble/sync. 1 = 16 sync word
   ↪ bits detected. 2 = 16/16 sync word bits detected. 3 =
   ↪ 30/32 sync word bits detected. 4 = No preamble/sync,
   ↪ carrier-sense above threshold. 5 = 15/16 +
   ↪ carrier-sense above threshold. 6 = 16/16 +
   ↪ carrier-sense above threshold. 7 = 30/32 +
   ↪ carrier-sense above threshold.
12 trans.setSyncWord(211, 145); // Set sync word. Must be the
   ↪ same for the transmitter and receiver. (Syncword high,
   ↪ Syncword low)
13 trans.setAdrChk(0);          // Controls address check
   ↪ configuration of received packages. 0 = No address
   ↪ check. 1 = Address check, no broadcast. 2 = Address
   ↪ check and 0 (0x00) broadcast. 3 = Address check and 0
   ↪ (0x00) and 255 (0xFF) broadcast.
14 trans.setAddr(0);            // Address used for packet
   ↪ filtration. Optional broadcast addresses are 0 (0x00)
   ↪ and 255 (0xFF).
15 trans.setWhiteData(0);       // Turn data whitening on /
   ↪ off. 0 = Whitening off. 1 = Whitening on.
16 trans.setPktFormat(0);       // Format of RX and TX data.
   ↪ 0 = Normal mode, use FIFOs for RX and TX. 1 =
   ↪ Synchronous serial mode, Data in on GDO0 and data out
   ↪ on either of the GDOx pins. 2 = Random TX mode; sends
   ↪ random data using PN9 generator. Used for test. Works
   ↪ as normal mode, setting 0 (00), in RX. 3 =
   ↪ Asynchronous serial mode, Data in on GDO0 and data out
   ↪ on either of the GDOx pins.
17 trans.setLengthConfig(1);    // 0 = Fixed packet length
   ↪ mode. 1 = Variable packet length mode. 2 = Infinite
   ↪ packet length mode. 3 = Reserved
18 trans.setPacketLength(0);    // Indicates the packet
   ↪ length when fixed packet length mode is enabled. If
   ↪ variable packet length mode is used, this value
   ↪ indicates the maximum packet length allowed.
19 trans.setCrc(0);             // 1 = CRC calculation in TX
   ↪ and CRC check in RX enabled. 0 = CRC disabled for TX
   ↪ and RX.
20 trans.setCRC_AF(0);          // Enable automatic flush of
   ↪ RX FIFO when CRC is not OK. This requires that only
   ↪ one packet is in the RXIFIFO and that packet length is
   ↪ limited to the RX FIFO size.
```

```
21 trans.setDcFilterOff(0);     // Disable digital DC
   ↪ blocking filter before demodulator. Only for data
   ↪ rates <= 250 kBaud The recommended IF frequency
   ↪ changes when the DC blocking is disabled. 1 = Disable
   ↪ (current optimized). 0 = Enable (better sensitivity).
22 trans.setManchester(0);      // Enables Manchester
   ↪ encoding/decoding. 0 = Disable. 1 = Enable.
23 trans.setFEC(0);             // Enable Forward Error
   ↪ Correction (FEC) with interleaving for packet payload
   ↪ (Only supported for fixed packet length mode. 0 =
   ↪ Disable. 1 = Enable.
24 trans.setPRE(0);             // Sets the minimum number of
   ↪ preamble bytes to be transmitted. Values: 0 : 2, 1 :
   ↪ 3, 2 : 4, 3 : 6, 4 : 8, 5 : 12, 6 : 16, 7 : 24
25 trans.setPQT(0);             // Preamble quality estimator
   ↪ threshold. The preamble quality estimator increases an
   ↪ internal counter by one each time a bit is received
   ↪ that is different from the previous bit, and decreases
   ↪ the counter by 8 each time a bit is received that is
   ↪ the same as the last bit. A threshold of 4xPQT for
   ↪ this counter is used to gate sync word detection. When
   ↪ PQT=0 a sync word is always accepted.
26 trans.setAppendStatus(0);    // When enabled, two status
   ↪ bytes will be appended to the payload of the packet.
   ↪ The status bytes contain RSSI and LQI values, as well
   ↪ as CRC OK.
```

Обратите внимание на ключевые настройки, приведенные в таблице 4.3.10.

Таблица 4.3.10

Настройка	**Описание**
`setMHz`	Установка центральной частоты (в МГЦ)
`setDeviation`	Установка девиации сигнала (в кГц)
`setDRate`	Установка скорости передачи (в Кбод/с)
`setModulation`	Выбор модуляции

Для первого тестирования рекомендуется установить минимальную скорость передачи — 100 бод/с, чтобы было проще наблюдать сообщение на радиоприемнике.

3.3. Передача данных

После инициализации передатчика доступна возможность отправлять через него данные с помощью метода `.SendData`. Для того чтобы код отправлял сообщение постоянно, нужно поместить вызов этого метода в раздел `USER CODE WHILE`, который находится внутри бесконечного цикла.

Например, сообщения можно отправлять так (см. ниже).

```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    trans.SendData("Hello world!");
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

У метода `.SendData` есть два вида реализации, которые отличаются принимаемыми аргументами (см. таблицу 4.3.11).

Таблица 4.3.11

Сигнатура	Описание
`.SendData(const char* string)`	Реализация, которая подходит только для передачи текста.
`.SendData(uint8_t* txBuffer, uint8_t size)`	Универсальная реализация, подходящая под любой вид данных, включая текст. На нее подается указатель на начало данных и их длина.

4. **Процесс прошивки**

   После подключения программатора к микроконтроллеру, а затем — к USB-порту ПК станет возможна загрузка программы в плату.

   Для загрузки программы в плату нажмите на значок  в верхней панели меню.

   Примечание. Помимо режима загрузки существует также режим отладки, который бывает крайне полезен, если где-то в коде закралась ошибка, но ее не удается найти быстро. Для перехода в режим отладки (debug) можно нажать на значок жучка  в верхней панели меню. О том, как пользоваться режимом отладки можно посмотреть, например, здесь: https://youtu.be/BVC7KaUNCS8.

5. **Проверка приемника**

   Для того чтобы убедиться, что приемник на базе RTL-SDR принимает сигнал, можно воспользоваться gqrx. В ПО должен отчетливо быть виден пик полезного сигнала.

6. **Проверка работы GNU Radio**

   Убедившись, что в gqrx есть сигнал, постройте простейший граф в GNU Radio, в котором используйте лишь источник сигнала (блок osmocom source) и его вывод (блок QT GUI Sink). Не забудьте установить корректную частоту дискретизации (sample rate), например, 1 000 000 сэмплов/с.

   Ссылка на вебинар: https://orbita.education/ru/materials/298/302/4867.

Предупреждение

Во избежание проблем пересечения каналов разных команд, за каждой командой закрепляется ее центральная частота. Для получения своей частоты обратитесь к организаторам.

Внимание: за преднамеренную работу на других частотах, с целью помешать другим командам, будут начисляться штрафы по задачам инженера связи.

Решение

В соответствии с заданием, граф состоит из двух блоков. Все параметры остаются по умолчанию за исключением частоты, которая будет отличаться у разных команд.

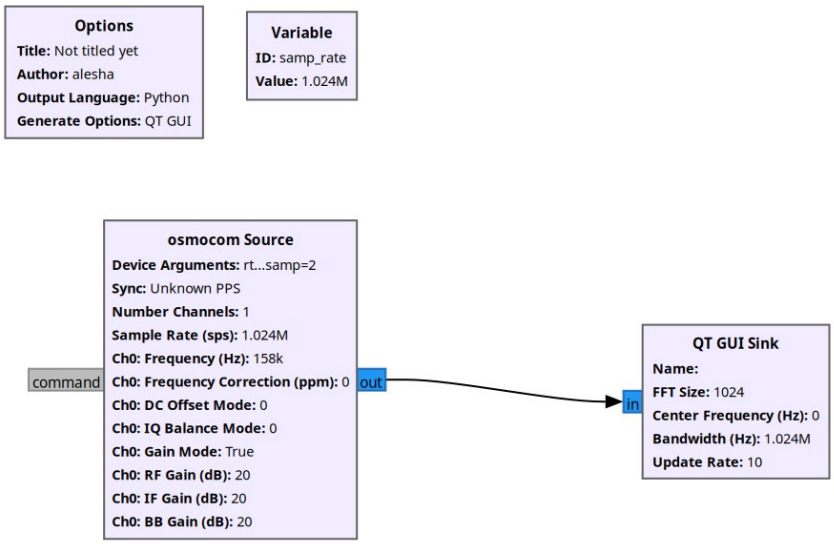


Рис. 4.3.9

Код для передатчика см. ниже.

C++

```
/* Private user code
   ↪ ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
IntroSatLib::intefaces::SPI spi(&hspi2);
IntroSatLib::intefaces::GPIO sck(CC1101_SCK_GPIO_Port,
   ↪ CC1101_SCK_Pin);
IntroSatLib::intefaces::GPIO mosi(CC1101_MOSI_GPIO_Port,
   ↪ CC1101_MOSI_Pin);
IntroSatLib::intefaces::GPIO miso(CC1101_MISO_GPIO_Port,
   ↪ CC1101_MISO_Pin);
IntroSatLib::intefaces::GPIO reset(CC1101_CS_GPIO_Port,
   ↪ CC1101_CS_Pin);
IntroSatLib::intefaces::GPIO gd0(CC1101_GD0_GPIO_Port,
   ↪ CC1101_GD0_Pin);
IntroSatLib::CC1101WithGD0 trans(spi, sck, mosi, miso, reset, gd0);
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
```

```
14    */
15  int main(void)
16  {
17  //.....
18    /* Initialize all configured peripherals */
19    MX_GPIO_Init();
20    MX_DMA_Init();
21    MX_I2C1_Init();
22    MX_I2C2_Init();
23    MX_SPI1_Init();
24    MX_SPI2_Init();
25    MX_USART1_UART_Init();
26    MX_USART2_UART_Init();
27    MX_FATFS_Init();
28    /* USER CODE BEGIN 2 */
29    trans.Init();                 // must be set to initialize the
      ↪  cc1101!
30    trans.setCCMode(1);           // set config for internal transmission
      ↪  mode.
31    trans.setModulation(0);
32    trans.setMHZ(444);
33    trans.setDeviation(47.60);
34    trans.setChannel(0);
35    trans.setChsp(199.95);
36    trans.setRxBW(812.50);
37    trans.setDRate(0.1);
38    trans.setPA(10);
39    trans.setSyncMode(2);
40    trans.setSyncWord(211, 145);
41    trans.setAdrChk(0);
42    trans.setAddr(0);
43    trans.setWhiteData(0);
44    trans.setPktFormat(0);
45    trans.setLengthConfig(1);
46    trans.setPacketLength(0);
47    trans.setCrc(0);
48    trans.setCRC_AF(0);
49    trans.setDcFilterOff(0);
50    trans.setManchester(0);
51    trans.setFEC(0);
52    trans.setPRE(0);
53    trans.setPQT(0);
54    trans.setAppendStatus(0);
55      /* USER CODE END 2 */
56    /* Infinite loop */
57    /* USER CODE BEGIN WHILE */
58    while (1)
59    {
60            trans.SendData("Hello world");
61    }
62    /* USER CODE END 3 */
63  }
```

Баллистик

Аппарат ДЗЗ был развернут на орбите, однако по расчетному времени пролетов над ЦУП не выходит на связь. Через некоторое время удалось поймать сигнал от аппарата, однако не в заданное время и не в заданном месте. Это может свиде-

тельствовать о том, что аппарат находится не на той орбите. Необходимо уточнить параметры текущей орбиты аппарата.

Известно, что:

- проход над Канберра (–35,2931, 149,1269) был в момент времени с 03:41 по 03:47;
- проход над Синьюй (27,8043, 114,9333) был в момент времени с 16:24 по 16:29;
- проходы над Эль-Минья (28,1194, 30,7444) были в моменты времени с 08:26 по 08:32 и с 23:16 по 23:21.

Все проходы даны за 01 апреля 2025 года.

Параметры планируемой орбиты миссии:

- Большая полуось: 7 195 км.
- Наклонение: 60°.
- Эксцентриситет: 0,003.
- Аргумент перицентра: 0.
- Долгота восходящего узла: 185°.
- Истинная аномалия: 0°.

Время вывода аппарата на орбиту: 01 апреля 2025 года 00:00 по UTC. Время моделирования — 24 ч.

Известно, что наклонение и долгота восходящего узла остались неизменными.

Радиус Земли: 6 371 008,8 м.

Угол над горизонтом, при котором возможен прием данных: 30°.

За каждый проход над станцией в установленный промежуток времени начисляется 2,25 балла. Максимальный балл за задачу — 9. Начиная с пятой попытки начинается дисконт за каждую попытку, равный 10% максимального балла. Оценка снижается вплоть до 0,9 балла, далее не уменьшается с числом попыток.

Решение

Параметры установившейся орбиты миссии:

- Большая полуось: 7 250 км.
- Наклонение: 60°.
- Эксцентриситет: 0.
- Аргумент перицентра: 0.
- Долгота восходящего узла: 185°.
- Истинная аномалия: 155°.

Критерии оценки первого дня

Таблица 4.3.12

Задача	Максимальный балл
Программист (платформа спутника)	
Настроена беспроводная передача информации по UART и беспроводной способ прошивки микроконтроллера	2
Задано управление двигателем, продемонстрировано изменение направления и скорости вращения	2
Считаны данные с датчика позиционирования (как это делалось на удаленной работе)	2
Реализован алгоритм стабилизации спутника	4
Программист (полезная нагрузка)	
Реализовано чтение данных с датчика AMG8833	3
Спаяна схема для восстановления работы датчика, реализована соответствующая управляющая функция	2
Реализована функция оценки положения по данным датчика AMG8833	4
Инженер связи	
Подключен передатчик, написана и загружена прошивка для передачи данных	3
Подключен и проверен приемник	1
С помощью приемника подтверждена работа передатчика; проверена правильность его центральной частоты и девиации	4
Проверена работа GNU Radio	1
Баллистик	
Решена первая задача в симуляторе	9
Итого	**37**

Второй день

Программист (платформа спутника)

Второй день посвящен алгоритму ориентации.

Основная задача второго дня: написание алгоритма ориентации спутника на землю, используя показания с матричного ИК-датчика, которые предобработал программист полезной нагрузки, написав алгоритм ориентации на Землю.

Для решения задачи можно принять, что начальная скорость спутника равна 0. Код опроса датчика программиста полезной нагрузки необходимо встроить в прошивку. Ориентация функционирует, и опрос датчика работает на одном контроллере.

Цитата из задачи программиста полезной нагрузки прошлого дня:

«Функция оценки положения должна обрабатывать полученный снимок

и в зависимости от его содержания возвращать:

- *0 — если корректировки не требуются;*
- *положительное число (цифру) — если для корректировки необходимо вращать спутник по часовой стрелке;*
- *отрицательное число (цифру) — если для корректировки необходимо вращать спутник против часовой стрелки;*
- *большую по модулю константу (например, 127) — если Земли нет в кадре».*

Решение

C++

```
/* USER CODE BEGIN 3 */
    /* предыдущий код, тут только изменения */
/* sprintf(result, "Angle speed\t\tx:%f\ty:%f\tz:%f\r\n", x, y, z);
HAL_UART_Transmit(&huart1, (uint8_t*)result, strlen(result), 1000);
HAL_Delay(250);
motor.NeedSpeed(2000);
HAL_Delay(1000);
motor.NeedSpeed(-2000);
HAL_Delay(1000); */
float center = findCenter();
float needSpeed = (center < -1 ? -1 : center) * 0.2;
```

Программист (полезная нагрузка)

Этот день посвящен:

1. работе с камерой и формированию пакета данных для передачи его на наземную станцию;
2. созданию скрипта для визуализации снимка.

Работа с камерой и дальнейшая передача снимка на наземную станцию для удобства реализации вынесена на отдельный МК. Поэтому задачи этого и следующего дня будут тесно связаны с задачами инженера связи.

Взаимодействие с камерой

Камера способна делать черно-белые снимки с максимальным разрешением 640× × 480 пикселей. Общение с камерой происходит по протоколу UART на скорости 230 400 бод без бита четности.

Поддерживаемые команды:

- Команда: `0x74` (или символ `t`) — сделать снимок (делает новый снимок и сохраняет его в память платы камеры).
- Команда: `0x70` (или символ `p`) — запрос данных снимка (запрашивает данные о сохраненном снимке; возвращает структуру `ImageProperties` побайтово, `Little-endian`).

```
typedef struct {
  uint16_t height, width;
  uint16_t vStart, hStart;
  Colorspace colorspace = EMPTY;
  uint16_t exposure;
  uint32_t length;
  uint16_t numberOfChunks;
} ImageProperties;
```

Рис. 4.3.10

Поля:

- ◇ `height` — 2 байта. Хранит высоту снимка в пикселях.
- ◇ `width` — 2 байта. Хранит ширину снимка в пикселях.
- ◇ `vStart` — 2 байта. Хранит индекс первой строки матрицы, включенной в передаваемый снимок.
- ◇ `hStart` — 2 байта. Хранит индекс первого столбца матрицы, включенного в передаваемый снимок.
- ◇ `colorspace` — 2 байта. Не используется.
- ◇ `exposure` — 2 байта. Хранит значение экспозиции снимка. Чем выше это значение, тем сильнее камера усиливала приходящий на сенсор сигнал. Имеет диапазон [0–509]. Верхняя граница в 509 — искусственная (?). Вручную можно установить больше, это никак не изменяет фактическую картинку по сравнению со значением 509.
- ◇ `length` — 4 байта. Хранит длину снимка в байтах.
- ◇ `numberOfChunks` — 2 байта. Хранит количество пакетов данных, на которые разбито изображение.

- Команда: `0x6e` (или символ `n`) — запрос следующего пакеты данных. В ответ на эту команду камера присылает побайтово структуру с данными снимка, `Littleendian`.

```
typedef struct {
  uint16_t chunkID, payloadLength;
  bool isLastChunk;
  uint8_t payload[240];
  uint8_t checksum;
} Chunk;
```

Рис. 4.3.11

Поля:

- ◇ `chunkID` — 2 байта. Порядковый номер пакета данных в кадре.

- ◇ `payloadLength` — 2 байта. Количество полезных данных в поле `payload`. Общая длина пакета всегда остается одинаковой — в поле `payload` всегда 240 байт, но лишь часть из них может быть полезной (в последнем пакете снимка, количество байт в котором некратно 240).
- ◇ `isLastChunk` — 2 байта. `True`, если пакет является последним. `False`, если пакет не является последним.
- ◇ `payload` — 240 байт. Байты с 0 до `payloadLength-1` — пиксели картинки. 1 байт = 1 пиксель.
- ◇ `checksum` — 2 байта. Не используется.

- Команда: `0x72` (или символ `r`) — сброс `chunkID`. Сбрасывает счетчик чанков в памяти камеры. После сброса по команде `0x6e` будет отправлен первый пакет данных кадра.
- Команда: `0x73` (или символ `s`) + `uint16_t` (ширина) + `uint16_t` (высота) — изменение размера снимка. Байты ширины и высоты отправляются `little-endian`.
- Команда: `0x65` (или символ `e`) + `uint16_t` (экспозиция) — изменение экспозиции.

  Байты значения экспозиции отправляются `little-endian`. Значение экспозиции может быть в диапазон [0–509]. Верхняя граница в 509. При отправке значения, отличного от нуля, камера начинает снимать с установленной экспозицией. При отправке нуля камера переходит в режим автоэкспозиции. После этого может потребоваться некоторое время, чтобы алгоритм автоэкспозиции подобрал идеальное значение.

Формирование пакета данных

После того как участники научились получать снимки с камеры, нужно сформировать пакет для отправки его на наземную станцию. Формат пакета и его структура могут быть выбраны на усмотрение команды, но каждый отправляемый на вход передатчика пакет обязательно должен содержать:

- преамбулу;
- время формирования пакета в бортовой системе (от начала миссии);
- идентификатор пакета внутри снимка (так как один снимок передается несколькими пакетами по частям);
- длину полезной нагрузки (будет полезно для задачи следующего дня);
- 480 байт данных снимка (два чанка);
- постамбулу.

Создания скрипта для визуализации изображения

Чтобы проверить правильность передачи данных, следует создать python-скрипт для обработки пакетов и визуализации полученного снимка. Для взаимодействия с GNURadio можно использовать клиент-серверную архитектуру. Скрипт обработки сигнала, реализуемый в GNURadio, в данном случае будет являться сервером, а python-скрипт для визуализации сообщения — клиентом.

Шаблон серверной части см. ниже.

Python

```python
import socket  # Импорт модуля для работы с сокетами (сетевое взаимодействие)
import cv2  # Импорт OpenCV для работы с изображениями
import numpy as np  # Импорт NumPy для работы с массивами данных
# Создание сокета для подключения к серверу
sock = socket.socket()
# Подключение к серверу по адресу 127.0.0.1 (локальный хост) и порту 52001
sock.connect(("127.0.0.1", 52001))
# Флаг для управления циклом приема данных
rec_flag = True
# Переменная для накопления данных изображения
image_data = b''
# Функция для отображения изображения
def show_image(image_data, width, height):
    # Преобразование байтовых данных в массив NumPy
    img_array = np.frombuffer(image_data, dtype=np.uint8)
    # Изменение формы массива в соответствии с размерами изображения
    img_array = img_array.reshape((height, width))
    # Отображение изображения в окне с названием "Captured Image"
    cv2.imshow("Captured Image", img_array)
    # Ожидание нажатия любой клавиши для закрытия окна
    cv2.waitKey(0)
    # Закрытие всех окон OpenCV
    cv2.destroyAllWindows()
# Основной блок программы
if __name__ == "__main__":
    try:
        # Цикл приема данных от сервера
        while(rec_flag):
            # Получение данных от сервера (bufsize байт за раз)
            data_raw = sock.recv(bufsize)
            # Здесь нужно обработать пакет и выделить полезные данные (payload)
            # Например, извлечь байты изображения из data_raw
            # Добавление полученных данных к общему массиву image_data
            image_data += payload
            # Условие окончания приема изображения (нужно доработать)
            # Например, если получен специальный флаг окончания передачи
            if _____ :  # Здесь нужно добавить условие завершения
                rec_flag = False  # Остановка цикла приема данных
        # Отображение полученного изображения
        show_image(image_data, 640, 480)

    # Обработка прерывания с клавиатуры (Ctrl+C)
    except KeyboardInterrupt:
        print("Завершение работы")
    # Блок finally выполняется всегда, даже если произошла ошибка
    finally:
        # Закрытие соединения с сервером
        sock.close()
        print("Соединение закрыто.")
```

Решение

C

```c
typedef struct {
    uint16_t height, width;
    uint16_t vStart, hStart;
    uint16_t colorspace;
    uint16_t exposure;
    uint32_t length;
    uint16_t numberOfChunks;
} ImageProperties;
typedef struct {
    uint16_t chunkID, payloadLength;
    bool isLastChunk;
    uint8_t payload[240];
    uint8_t checksum;
} Chunk;
//часть кода цикла while(1)
uint8_t buffer[3 + 0 + 3] = {0};
uint8_t bufferProperties[3 + sizeof(ImageProperties) + 3] = {0};
uint8_t bufferData[3 + sizeof(Chunk) + 3] = {0};
HAL_UART_Transmit(&huart2, (uint8_t *)"t", 1, 100);
HAL_UART_Receive(&huart2, buffer, 3 + 0 + 3, 2000);
HAL_UART_Transmit(&huart2, (uint8_t *)"p", 1, 100);
HAL_UART_Receive(&huart2, bufferProperties, 3 +
↪  sizeof(ImageProperties) + 3, 2000);
ImageProperties* properties = (ImageProperties*)(bufferProperties +
↪  3);
for (int i = 0; i < properties->numberOfChunks; i++){
    HAL_UART_Transmit(&huart2, (uint8_t *)"n", 1, 100);
    HAL_UART_Receive(&huart2, bufferData, 3 + sizeof(Chunk) + 3,
    ↪  2000);
    Chunk* data = (Chunk*)(bufferData + 3);
    for(int j = 0; j < data->payloadLength; j + 24){
        uint8_t msg[32] = {0};
        uint32_t time = HAL_GetTick();
        uint16_t id = data->chunkID * 10 + j;
        msg[0] = 0xFF;
        msg[1] = 0xFF;
        msg[2] = (uint8_t)(time >> 0);
        msg[3] = (uint8_t)(time >> 8);
        msg[4] = (uint8_t)(time >> 16);
        msg[5] = (uint8_t)(time >> 24);
        msg[6] = (uint8_t)(id >> 0);
        msg[7] = (uint8_t)(id >> 16);
        for (int k = 0; k < 24; ++k) {
            msg[k + 8] = (data->payload)[j * 24 + k];
        }

        HAL_UART_Transmit(&huart1, msg, 32, 1000);
    }
}
//конец части кода цикла while(1)
```

Инженер связи

В течение дня необходимо наладить канал для получения данных полезной нагрузки МК, протестировать их передачу и прием, а также начать реализовы-

вать помехозащищенное кодирование для более надежной передачи данных (см. рис. 4.3.12–4.3.13).

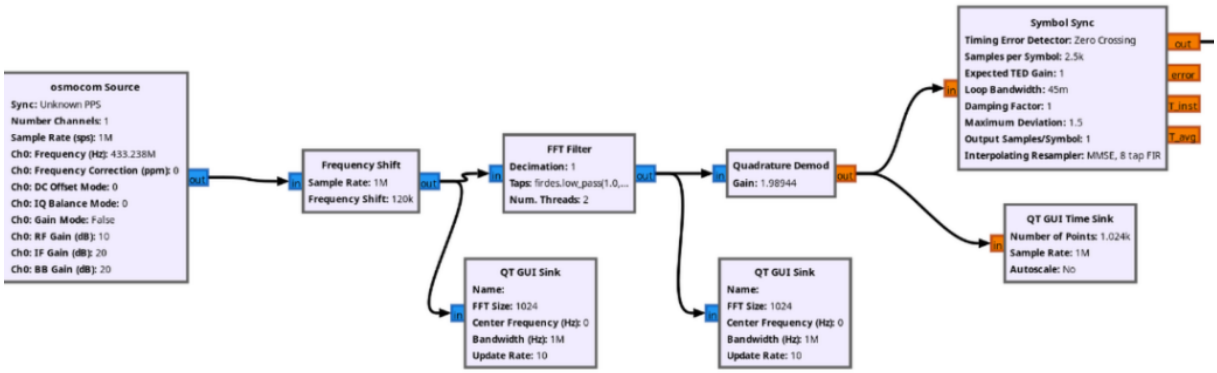


Рис. 4.3.12

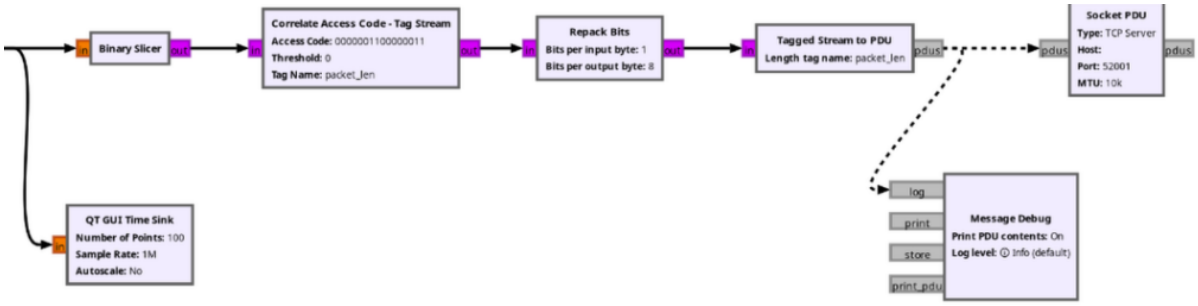


Рис. 4.3.13

Решение

Реализация канала связи между МК и помехозащищенного кодирования остается на выбор участников.

Пример реализации графа для обработки сигнала, где пакет состоит из преамбулы, двух 16-битных длин полезной нагрузки и самой полезной нагрузки.

Баллистик

Ученые вывели на орбиту спутник для наблюдения за поверхностью Земли. Целевая зона съемки: область, ограниченная 59° и 61° северной широты, 93° и 95° восточной долготы. Аппарат должен снять часть указанной зоны, а после передать данные на наземную станцию в Бразилиа (–15,7797, 312,0703).

Однако спутник имеет конструкционную особенность: камера, которая производит съемку местности, расположена на оси его ориентации в положительном направлении. Антенна, позволяющая передавать снимки на Землю, расположена на той же оси, но в противоположном направлении.

Поэтому для того, чтобы снять зону и передать снимок на Землю, аппарату необходимо при пролете над зоной съемки ориентироваться в надир (по отношению к Земле) камерой, а затем ориентироваться в надир антенной (совершить поворот на 180° и поддерживать направление в надир далее), чтобы передать данные на Землю при пролете над городом Бразилиа.

В качестве решения нужно рассчитать орбиту спутника и написать программу управления спутником.

Время вывода аппарата на орбиту: 01 апреля 2025 года 00:00 по UTC. Время моделирования — 24 ч.

На момент начала симуляции считать спутник стабилизированным.

По умолчанию спутник ориентирован в положительном направлении оси oZ.

Справка по начальным параметрам и синтаксису: https://disk.360.yandex.ru/d/HyvQpPND51Y1Ew.

Радиус Земли: 6 371 008,8 м.

Угол над горизонтом, при котором возможен прием данных, при условии, что антенна спутника смотрит в надир: 60°.

Начисляется 8 баллов, если спутник пролетел над зоной интереса и в этот момент был ориентирован камерой в надир. Еще 8 баллов начисляется, если после описанного пролета над зоной интереса спутник пролетел над наземной станцией и был ориентирован антенной в надир.

Решение

Параметры орбиты:

- Большая полуось: 7 000 км.
- Наклонение: 90°.
- Эксцентриситет: 0.
- Аргумент перицентра: 0.
- Долгота восходящего узла: 0°.
- Истинная аномалия: 0°.

Код управления спутником см. ниже.

JavaScript

```
'use strict';
var wheels;
var gyros;
function setup() {
    wheels = spacecraft.devices[0];
    gyros = spacecraft.devices[1];
 }
var old_a_cam = 0;
var old_a_ant = 0;
 function loop() {
     if (spacecraft.flight_time > 0 && spacecraft.flight_time < 800) {
        wheels.functions[1].motor_torque = 0;
        let a = orientation_angle_cam();
```

```
        let m = 90 + spacecraft.devices[2].functions[0].location[0];
        let s = 0.000005*(a-m) + 0.00007*(a-old_a_cam);
        wheels.functions[1].motor_torque = -s;
        old_a_cam = a;
     }

    else if (spacecraft.flight_time > 50000 && spacecraft.flight_time < 50800) {
        wheels.functions[1].motor_torque = 0;
        let a = orientation_angle_ant();
        let m = 90 + spacecraft.devices[2].functions[0].location[0];
        let s = 0.000005*(a-m) + 0.00007*(a-old_a_ant);
        wheels.functions[1].motor_torque = s;
        old_a_ant = a;
    }
    else {
        wheels.functions[1].motor_torque = 0;
    }
 }

function rotateVectorByQuaternion(vector, quaternion) {
    const [qx, qy, qz, qw] = quaternion;
    const [vx, vy, vz] = vector;
    const rotatedVector = [
        (1 - 2 * qy * qy - 2 * qz * qz) * vx + (2 * qx * qy - 2 * qz * qw) * vy + (2 * qx * qz + 2 * qy * qw) * vz,
        (2 * qx * qy + 2 * qz * qw) * vx + (1 - 2 * qx * qx - 2 * qz * qz) * vy + (2 * qy * qz - 2 * qx * qw) * vz,
        (2 * qx * qz - 2 * qy * qw) * vx + (2 * qy * qz + 2 * qx * qw) * vy + (1 - 2 * qx * qx - 2 * qy * qy) * vz
    ];
    return rotatedVector;
}
function angleBetweenVectors(vector1, vector2) {
    const dotProduct = vector1[0] * vector2[0] + vector1[1] * vector2[1] + vector1[2] * vector2[2];
    const magnitude1 = Math.sqrt(vector1[0] * vector1[0] + vector1[1] * vector1[1] + vector1[2] * vector1[2]);
    const magnitude2 = Math.sqrt(vector2[0] * vector2[0] + vector2[1] * vector2[1] + vector2[2] * vector2[2]);
    const angleRadians = Math.acos(dotProduct / (magnitude1 * magnitude2));
    const angleDegrees = angleRadians * (180 / Math.PI);
    return angleDegrees;
}
var cam_vector = [0,0,1];
var ant_vector = [0,0,-1];
function my_angle_cam(orientation) {
    return angleBetweenVectors(cam_vector, rotateVectorByQuaternion(cam_vector, orientation));
}
function my_angle_ant(orientation) {
    return angleBetweenVectors(cam_vector, rotateVectorByQuaternion(ant_vector, orientation));
}
function orientation_angle_cam() {
     let star_tracker = spacecraft.devices[3].functions[0];
     let orientation = star_tracker.orientation;
     return my_angle_cam(orientation);
}
```

```
function orientation_angle_ant() {
    let star_tracker = spacecraft.devices[3].functions[0];
    let orientation = star_tracker.orientation;
    return my_angle_ant(orientation);
}
```

Критерии оценки второго дня

Таблица 4.3.13

Задача	**Максимальный балл**
Программист (платформа спутника)	
Аппарат сориентирован на Землю	15
Программист (полезная нагрузка)	
Создана функция получения снимка с камеры	4
Создана функция формирования пакета данных	5
Доработан скрипт визуализации, получена картинка	7
Инженер связи	
Реализована отправка, прием и корректное декодирование тестовых данных	7
Реализована на МК отправка пакета данных от программиста полезной нагрузки	1
Реализован прием пакета данных от программиста полезной нагрузки	7
Реализована передача принятого пакета данных программисту полезной нагрузки посредством клиент-серверного взаимодействия	1
Баллистик	
Решена вторая задача в симуляторе	16
Итого	**63**

Третий день

Программист (платформа спутника)

Необходимо выполнить несколько задач:

- собрать единую программу управления, в которой реализована стабилизация аппарата после начальной закрутки, ориентации на Землю и отправке сигнала модулю с камерой о необходимости сделать снимок;
- доработать точность алгоритма ориентации, чтобы Солнце было строго по центру.

Таблица 4.3.14. Циклограмма работы спутника

Шаг	Описание шага
1	Стабилизация спутника после начальной закрутки
2	Ориентация на землю
3	Передача сигнала о необходимости сделать кадр

Примечание:

- сигналом о необходимости сделать кадр является подача 1 на ножку микроконтроллера, подключенную к контроллеру передатчику;
- точность наведения оценивается по среднему сдвигу Земли относительно центра кадра за три запуска.

Решение

C++

```cpp
/* предыдущий код, тут только изменения */
int counter = 0;
/* USER CODE END 0 */
    /* USER CODE BEGIN 3 */
    /* предыдущий код, тут только изменения */

    float center = findCenter();
    if (center == 0) { counter++; }
    else { counter = 0; }
    HAL_GPIO_WritePin(TAKE_PHOTO_GPIO_Port, TAKE_PHOTO_Pin, counter > 30 ? GPIO_PIN_SET : GPIO_PIN_RESET);
    float needSpeed = (center < -1 ? -1 : center) * 0.2;
```

Программист (полезная нагрузка)

В течение дня необходимо подготовить все для финальной защиты, а именно:

- Завершить задачи прошлых дней, если они остались.
- Наладить взаимодействие созданных ранее программ работы с камерой и формирования пакетов с программой передачи данных инженера связи.
- Наладить взаимодействие с МК, отвечающим за ориентацию (описано ниже).
- Интегрировать приемопередающий комплекс в состав спутника.
- Проверить полный цикл работы спутника.

Основная задача — подготовка к защите, именно ее рекомендуется сделать приоритетной. Если члены команды уверены, что они располагают временем, то им предлагается решить задачу сжатия данных.

Взаимодействие с МК, отвечающим за ориентацию

В качестве взаимодействия со вторым микроконтроллером требуется реализовать функцию, которая запускает процесс получения снимка с камеры и его отправки на наземную станцию, если на определенный (на усмотрение участников) GPIO-пин приходит высокий логический сигнал.

Реализация алгоритма сжатия пакетов

В рамках задачи следует реализовать описанный ниже алгоритм сжатия данных. Учтите, что обычно эффективность сжатия зависит от размера сжимаемого блока данных — чем больше блок данных, тем выше коэффициент сжатия. Не забудьте реализовать алгоритм распаковки в скрипте для визуализации снимка.

Алгоритм RLE-сжатия

Данный алгоритм использует метод **Run-Length Encoding (RLE)** для эффективного сжатия последовательности байтов. **RLE** — алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов.

Серией называется последовательность, состоящая из нескольких одинаковых символов. При кодировании (упаковке, сжатии) строка одинаковых символов, составляющих серию, заменяется строкой, содержащей сам повторяющийся символ и количество его повторов.

Правила кодирования

1. **Флаговый бит**

   Старший бит первого байта группы используется как флаг:

   - 0 — последовательность уникальных байтов.
   - 1 — последовательность повторяющихся байтов.

2. **Повторяющиеся байты (RLE-группы)**

   Если байт повторяется два и более раз подряд, он кодируется как:

   ```
   (0x80 | количество), значение
   ```

   - где `(0x80 | количество)` указывает, что это RLE-группа;
   - максимальное количество повторений в одной группе — **127** (из-за 7-битного счетчика);
   - если количество повторений превышает 127, последовательность разбивается на несколько групп.

   Пример: `AA AA AA AA BB BB BB BB BB`.

   Кодируется как: `0x83 AA 0x85 BB`.

   - `0x84 = 0x80 | 4` — 4 повторения AA.
   - `0x85 = 0x80 | 5` — 5 повторений BB.

3. **Уникальные байты (обычные последовательности)**

   Если несколько байтов **не повторяются подряд**, они записываются в обычном виде:

   ```
   количество, данные
   ```

   - Максимальная длина такой последовательности — 127.

   Пример: `12 34 56 78 AA AA AA 99 88`.

   Кодируется как: `0x04 12 34 56 78 0x83 AA 0x02 99 88`.

   - `0x04` — 4 байта без изменений.
   - `0x83 AA` — 3 повторения AA.
   - `0x02 99 88` — 2 байта без изменений.

Решение

C++

```cpp
// Взаимодействие с МК отвечающим за ориентацию
HAL_GPIO_WritePin(TAKE_PHOTO_GPIO_Port, TAKE_PHOTO_Pin, GPIO_PIN_SET);
```

C

```c
// Функция RLE сжатия для МК
int rle_encode(const uint8_t *input, int input_size, uint8_t *output) {
    int i = 0, out_index = 0;
    while (i < input_size) {
        int run_length = 1;
        while (i + run_length < input_size && input[i] == input[i + run_length] && run_length < 127) {
            run_length++;
        }
        if (run_length > 1) {
            output[out_index++] = 0x80 | run_length;
            output[out_index++] = input[i];
            i += run_length;
        } else {
            int unique_length = 1;
            while (i + unique_length < input_size &&
                   (unique_length == 1 || input[i + unique_length] != input[i + unique_length - 1]) &&
                   unique_length < 127) {
                unique_length++;
            }
            output[out_index++] = unique_length;
            for (int j = 0; j < unique_length; j++) {
                output[out_index++] = input[i + j];
            }
            i += unique_length;
        }
    }
    return out_index;
}
```

Python

```python
# Функция для декодирования на python
def rle_decode(encoded):
    decoded = []
    i = 0
    while i < len(encoded):
        header = encoded[i]
        i += 1
        if header & 0x80:
            count = header & 0x7F
            value = encoded[i]
            decoded.extend([value] * count)
            i += 1
        else:
            count = header
            decoded.extend(encoded[i:i + count])
            i += count
    return decoded
```

Инженер связи

В течение дня радист продолжает работу над помехозащищенным кодированием.

Решение

Выбор помехозащищенного кодирования и место его декодирования остаются за участниками. В качестве примера рассмотрим решение, в котором в качестве помехозащищенного кодирования используется код Хэмминга в рамках графа GNU Radio. Для этого можно использовать блок Python.

Блок, в котором задать следующий код, см. ниже.

Python

```python
import numpy as np
from gnuradio import gr
import pmt
class hamming_decoder_pdu(gr.basic_block):
    def __init__(self, n, k):
        """
        Конструктор блока.
        Параметры:
        n (int): Длина кодового слова (n >= k).
        k (int): Длина информационного слова (k < n).
        """
        gr.basic_block.__init__(self,
            name="hamming_decoder_pdu",
            in_sig=None,
            out_sig=None)
        # Сохраняем параметры кода Хэмминга
        self.n = n
        self.k = k
        # Вычисляем количество проверочных битов
        self.r = n - k
        # Регистрируем порты для PDU
        self.message_port_register_in(pmt.intern("in"))
```

```
23          self.message_port_register_out(pmt.intern("out"))
24          # Устанавливаем обработчик входящих сообщений
25          self.set_msg_handler(pmt.intern("in"), self.handle_msg)
26      def handle_msg(self, msg):
27          """
28          Обработчик входящих PDU сообщений.
29          """
30          # Получаем метаданные и данные из PDU
31          meta = pmt.car(msg)
32          data = pmt.cdr(msg)
33          # Преобразуем данные в массив питру
34          data = pmt.u8vector_elements(data)
35          # Декодируем данные с использованием кода Хэмминга
36          decoded_data = self.hamming_decode(data)
37          # Преобразуем декодированные данные обратно в PDU
38          decoded_pdu = pmt.init_u8vector(len(decoded_data),
            ↪  decoded_data)
39          self.message_port_pub(pmt.intern("out"), pmt.cons(meta,
            ↪  decoded_pdu))
40      def hamming_decode(self, data):
41          """
42          Декодирует данные с использованием кода Хэмминга (n, k).
43          Параметры:
44          data (list): Входные данные в виде списка битов.
45          Возвращает:
46          list: Декодированные данные.
47          """
48          decoded_data = []
49          # Проходим по данным с шагом n
50          for i in range(0, len(data), self.n):
51              chunk = data[i:i + self.n]
52              # Если длина чанка меньше n, пропускаем его
53              if len(chunk) < self.n:
54                  continue
55              # Вычисляем синдром
56              syndrome = self.calculate_syndrome(chunk)
57              # Если синдром не равен нулю, исправляем ошибку
58              if syndrome != 0:
59                  if syndrome - 1 < len(chunk):
60                      chunk[syndrome - 1] ^= 1  # Инвертируем ошибочный
                        ↪  бит
61              # Извлекаем информационные биты
62              info_bits = self.extract_info_bits(chunk)
63              decoded_data.extend(info_bits)
64          return decoded_data
65      def calculate_syndrome(self, chunk):
66          """
67          Вычисляет синдром для чанка данных.
68          Параметры:
69          chunk (list): Чанк данных длиной n.
70          Возвращает:
71          int: Значение синдрома.
72          """
73          syndrome = 0
74          for i in range(self.r):
75              mask = 1 << i
76              parity = 0
77              for j in range(self.n):
78                  if (j + 1) & mask:
79                      parity ^= chunk[j]
```

```
            syndrome |= parity << i
        return syndrome
    def extract_info_bits(self, chunk):
        """
        Извлекает информационные биты из чанка данных.
        Параметры:
        chunk (list): Чанк данных длиной n.
        Возвращает:
        list: Информационные биты длиной k.
        """
        info_bits = []
        info_positions = self.get_info_positions()
        for pos in info_positions:
            info_bits.append(chunk[pos])
        return info_bits
    def get_info_positions(self):
        """
        Возвращает позиции информационных битов в кодовом слове.
        Возвращает:
        list: Список позиций информационных битов.
        """
        positions = []
        for i in range(1, self.n + 1):
            if not (i & (i - 1)) == 0:  # Проверяем, не является ли i
            ↪  степенью двойки
                positions.append(i - 1)
        return positions[:self.k]  # Возвращаем только первые k
        ↪  позиций
```

Баллистик

После проведения успешных испытаний с опытными образцами космических аппаратов для дистанционного зондирования Земли было принято решение развернуть группировку таких аппаратов на орбите.

Цель группировки: обеспечить полное покрытие указанной зоны съемкой аппаратами с мультиспектральными камерами в ИК-диапазон.

Орбита аппаратов должна удовлетворять следующим условиям:

- высота аппарата над зоной съемки не должна превышать 1 000 км и не должна находиться ниже 300 км;
- ширина полосы съемки — 1 000 км;
- зона съемки — территория РФ (считать ее прямоугольником, который ограничен 41,16° и 77,72° северной широты, 27,33° и 190,35° восточной долготы).

Порядок определения узлов зоны:

1. зона разбивается по широте с севера на юг на полосы шириной 100 км, остаток отбрасывается;
2. каждая полоса разбивается по долготе с запада на восток на ячейки 100×100 км, остаток отбрасывается.

Назовем узлом центр каждой такой ячейки. Узел считается снятым, если след аппарата в надире прошел от него не далее, чем на расстоянии половины ширины полосы съемки.

Задача — используя минимальное количество аппаратов, обеспечить максимальное покрытие за сутки симуляции (то есть за 24 ч должна быть снята максимальная площадь зоны съемки).

Количество спутников, доступных для миссии — не более 100.

Внимание: учитывать теплобаланс, электробаланс и создавать программу управления спутником не нужно!

Дата и время начала симуляции: 01.04.2025 00:00 UTC.

Время симуляции: 24 ч.

Решение

Параметры орбиты «Аппарат 1»:

- Большая полуось: 7 300 км.
- Наклонение: 90°.
- Эксцентриситет: 0.
- Аргумент перицентра: 40.
- Долгота восходящего узла: 27,33°.
- Истинная аномалия: 0°.

Параметры орбиты «Аппарат 2»:

- Большая полуось: 7 300 км.
- Наклонение: 90°.
- Эксцентриситет: 0.
- Аргумент перицентра: 40.
- Долгота восходящего узла: 60,73°.
- Истинная аномалия: 0°.

Параметры орбиты «Аппарат 3»:

- Большая полуось: 7 300 км
- Наклонение: 90°.
- Эксцентриситет: 0.
- Аргумент перицентра: 40.
- Долгота восходящего узла: 93,34°.
- Истинная аномалия: 0°.

Параметры орбиты «Аппарат 4»:

- Большая полуось: 7 300 км.
- Наклонение: 90°.
- Эксцентриситет: 0.
- Аргумент перицентра: 40.
- Долгота восходящего узла: 125,94°.
- Истинная аномалия: 0°.

Параметры орбиты «Аппарат 5»:

- Большая полуось: 7 300 км.
- Наклонение: 90°
- Эксцентриситет: 0.
- Аргумент перицентра: 40.
- Долгота восходящего узла: 189,35°.
- Истинная аномалия: 0°.

Критерии оценки третьего дня

Таблица 4.3.15

Задача	**Максимальный балл**
Программист (платформа спутника)	
Точность алгоритма ориентации. Оценки за попытку: • отсутствие Земли в кадре — 0% отклонения; • Земля в кадре — разница координат центра Земли и центра кадра, деленное на половину ширины кадра, умноженное на 100%; • общая оценка — среднее арифметическое трех попыток.	10
Исполнение циклограммы (каждый шаг — 5 баллов).	15
Программист (полезная нагрузка)	
Налажено взаимодействие полезной нагрузки с МК, отвечающим за ориентацию аппарата.	3
Налажено взаимодействие полезной нагрузки с МК, отвечающим за передачу данных.	4
Полезная нагрузка интегрирована в корпус аппарата.	4
Реализованы функции кодирования и декодирования согласно алгоритму RLE.	14
Инженер связи	
Реализовано помехозащищенное кодирование.	5
Повреждено не более 10% изображения.	8
Повреждено не более 3% изображения.	13
Баллистик	
Решена третья задача в симуляторе.	25
Итого	**100**

Критерии оценки защиты

Таблица 4.3.16

Задача		**Максимальный балл**
Спутник собран, все элементы надежно закреплены, при этом не используются кустарные методы крепления (клей, изолента и пр.).		5
Циклограмма работы спутника (баллы начисляются за корректную работу в рамках каждого этапа отдельно).	Стабилизация	4
	Ориентация	8
	Съемка	3
	Передача данных	6
	Прием данных	9
Соблюдена последовательность выполнения этапов (по 0,6 балла за каждый выполненный корректно в свою очередь этап).		3
Положение Земли относительно спутника при съемке (оценивается отклонение от оптической оси камеры) Расчет по формуле: $(30° - \Delta)/30° \cdot 6$.		6
Качество переданного снимка — оценивается с помощью сравнения снимка, принятого с помощью эталонного решения и снимка, принятого участниками.		6
Итого		**50**

5. Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи предметного тура по двум предметам: информатике и физике и инженерного тура. В каждом предмете максимально можно было набрать 100 баллов, в инженерном туре 100 баллов. Для того чтобы пройти во второй этап, участники должны были набрать в сумме по обоим предметам и инженерному туру не менее 25,0 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется. Победители второго отборочного этапа должны были набрать не менее 30,0 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- информатика — максимально возможный балл за все задачи — 100 баллов;
- физика — максимально возможный балл за все задачи — 100 баллов.

Командный инженерный тур

Команды заключительного этапа получали за командный инженерный тур от 0 до 250,00 баллов: команда, набравшая наибольшее число баллов среди других команд, становилась командой-победителем.

Все результаты команд нормировались по формуле:

$$\frac{100 \times x}{MAX},$$

где x — число баллов, набранных командой,

MAX — число баллов, максимально возможное за инженерный тур.

В заключительном этапе олимпиады индивидуальные баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач по предмету 1 (информатика) с весом $K_1 = 0{,}15$,

по предмету 2 (физика) с весом $K_2 = 0{,}15$, баллы за командное решение задач инженерного тура с весом $K_3 = 0{,}7$.

Итоговый балл определяется по формуле:

$$S = K_1 \cdot S_1 + K_2 \cdot S_2 + K_3 \cdot S_3,$$

где S_1 — балл первой части заключительного этапа по информатике (предметный тур) ($S_{1\ \text{макс}} = 100$);

S_2 — балл первой части заключительного этапа по физике (предметный тур) ($S_{2\ \text{макс}} = 100$);

S_3 — итоговый балл инженерного командного тура ($S_{3\ \text{макс}} = 100$).

Итого максимально возможный индивидуальный балл участника заключительного этапа — 100 баллов.

Критерий определения победителей и призеров

Чтобы определить победителей и призеров (независимо от класса) на основе индивидуальных результатов участников, был сформирован общий рейтинг всех участников заключительного этапа. С начала рейтинга были выбраны 2 победителя и 6 призеров (первые 25% участников рейтинга становятся победителями или призерами, из них первые 8% становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня)

Категория	**Количество баллов**
Победители	57,93 и выше
Призеры	От 46,53 до 52,88

6. Работа наставника после НТО

Участие школьника в Олимпиаде может завершиться после любого из этапов: первого или второго отборочных, либо после заключительного этапа. В каждом случае после завершения участия наставнику необходимо провести с учениками рефлексию — обсудить полученный опыт и проанализировать, что позволило достичь успеха, а что привело к неудаче. Подробные материалы о проведении рефлексии представлены в курсе «Наставник НТО»: https://academy.sk.ru/events/310.

Наставнику важно проинформировать руководство образовательного учреждения, если его учащиеся стали финалистами, призерами и победителями. Публичное признание высоких результатов дополнительно повышает мотивацию.

В процессе рефлексии с учениками, не ставшими призерами или победителями, рекомендуется уделить особое внимание особенностям командной работы: распределению ролей, планированию работы, возникающим проблемам. Для этого могут использоваться опросники для самооценки собственной работы и взаимной оценки участниками других членов команды (P2P). Они могут выявить внутренние проблемы команды, для решения которых в план подготовки можно добавить мероприятия, направленные на ее сплочение.

Стоит рассказать, что в истории НТО было много примеров, когда не победив в первый раз, на следующий год участники показывали впечатляющие результаты, одержав победу сразу в нескольких профилях. Конечно, важно отметить, что так происходит только при учете прошлых ошибок и подготовке к Олимпиаде в течение года.

Важным фактором успешного участия в следующих сезонах НТО может стать поддержка родителей учеников. Знакомство с ними помогает наставнику продемонстрировать важность компетенций, развиваемых в процессе участия в НТО, для будущего образования и карьеры школьников. Поддержка родителей помогает мотивировать участников и позволяет выделить необходимое время на занятия в кружке.

С участниками-выпускниками наставнику рекомендуется обсудить их дальнейшее профессиональное развитие и его связь с выбранными профилями НТО. Отдельно можно обратить внимание на льготы для победителей и призеров, предлагаемые в вузах с интересующими ученика направлениями. Кроме того, ряд вузов предлагает льготы для всех финалистов НТО, а также учитывает результаты Конкурса цифровых портфолио «Талант НТО».