Работа призера заключительного этапа

командной инженерной олимпиады школьников

Олимпиада Национальной технологической инициативы

Профиль «Системы связи и дистанционного зондирования Земли»

Гирин Иван Андреевич

Класс: 11

Город: г. Москва

Школа: ГБОУ школа №1375

Регион: Москва

Уникальный номер участника: 637

Команда на заключительном этапе: ВИАС

Параллель: 10-11

Результаты заключительного этапа:

ИНДИВИДУАЛЬНАЯ ЧАСТЬ																	КОМАНДНАЯ ЧАСТЬ									Итог
МАТЕМАТИКА				ФИЗИКА				ИНФОРМАТИКА							Сумма	Макс. балл	0	1	2	3	4	5	6	Сумма	Макс. балл	
1	2	3	4	1	2	3	4	1/1	1/2	1/3	2/1	2/2	3/1	3/2												
0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	300	2	-	9	8	28	20	-	67	131	26,57

Индивидуальная часть

Персональный лист участника с номером 637:

Математика

Олимпиада НТИ

ФИО Гарин Иван Андреевич

Город Москва

Школа № 1575

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Космо

Предмет математика

Номер участника 637

1	2а	2б	3а	3б	3в	Σ
0	0	0	0	0	0	0

1)

	С	Т
0.	200 к	100 м
1)	100 к	100 м + 100 к
2)	150 к + 50 м	50 к + 50 м
3)	75 к + 25 м	125 к + 75 м
4)	137,5 к + 62,5 м	62,5 к + 37,5 м

к – кофе м – молоко

Далее я уже не буду обозначать объем молока

5)	68,75 к	131,25 к
6)	129,375 к	60,625 к
7)	64,6875 к	125,3125 к
8)	127,34375 к	62,65625 к

Перелив кофе 8 раз концентрация у Саши – $\frac{127,34375}{200} \approx \frac{63,6718}{100}$,

а у Рустама – $\frac{62,65625}{100}$;

Разница между ними 1,02 %

Поэтому надо сделать еще 2 переливания

9)	63,671875 к	126,328075 к
10)	126,8408775 к	63,16902775 к

$$\begin{array}{r} 63,67185 \\ +\ 63,1690275 \\ \hline 126,8408775 \end{array}$$

Получаем, что у Саши концентрация кофе $\frac{126,8408775}{200} \approx \frac{63,42043875}{100}$,

а у Рустама 63,169

Разница концентраций меньше 1%, начальный объем сохранен

Ответ: 10 переливаний

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Космо

Предмет математика

Номер участника 637

№ 2

а) Путь до центра - 10 м

$v_{робота} = 8$ м/мин $\quad t = \frac{10}{8} = \frac{5}{4}$ минуты ;

$T_{лазера} = 1$ минута ; $T_i < t$, значит робот при скорости 8 м/мин не успеет достигнуть центра невалеченным

б) Если $v_{робота} = 10$ м/мин, то $t = \frac{10}{10} = 1$ минута ;

$T_i = t$

Если считать робота материальной точкой, то он успеет достигнуть центра в момент, когда он будет обнаружен, но робот имеет некоторые размеры, значит будет обнаружен за несколько секунд до достижения центра.

Ответ не может. (Может ½, но не двигаясь по прямой!)

Физика

Олимпиада НТИ

ФИО Гурин Иван Андреевич

Город Москва

Школа № 1375

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Космо

Предмет физика

Номер участника 637

1	2	3	4	Σ
0	6	0	0	6

№2

Вычислим какую часть петли пройдет тело, обозначим ее K

$L = \pi R \cdot K$

$K = \frac{11,78}{3,14 \cdot 5} = 0,75$

рис 1 +6.

рис. 2

Тело сорвалось, значит у него закончилась энергия, не верно

На тело в точке L действует только сила тяжести, которая направлена вертикально вниз. На рисунке 2 показано куда упадет тело

Найдем расстояние от точки начала петли O, до точки падения L'

Поскольку ранее $L = \frac{3\pi}{4} \cdot R$, то нам падения $L' = \frac{\pi}{4} \cdot R$

$L' = \frac{3,14 \cdot 5}{4} = 3,925$ 6.

Ответ: Тело упадет в точку на дуге на расстоянии 3,925 м

№1

$S = 20 \cdot 10^3$ м

$v_p = 1,77 \cdot 10^3$ м/с

H

ΔS ?

Вокруг какой спутник

$V = \sqrt{\frac{GM}{R+H}} = 974,3$ м/с

Ракета движется равномерно и прямолинейно, значит

$t = \frac{S}{v_p} = \frac{20 \cdot 10^3}{1,77 \cdot 10^3} = 11,3$ с

$\Delta S = v_{отн} \Delta t$ $\Delta S = V_{отн} \Delta t = 11001$ м

Ответ: 11 км

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Ядерно

Предмет Физика

Номер участника 137

№4

можно считать, что сигналы приходят одновременно:

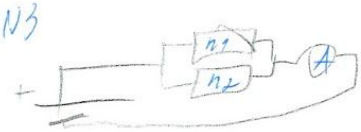
1) сигналы посылаются одновременно, и ~~путь~~ спутники не влияют на время передачи сигнала.

2) спутники движутся, образуя квадрат, значит расстояние между спутниками одинаковые ⇒ сигнал проходит одинаковый путь. не верно!

3) значит сигнал проходит одинаковый путь с одинаковыми скоростями, следовательно задержки между сигналами можно пренебречь.

Ответ: сигналы приходят одновременно ⊖

№3



$R_л$ – нагрузочное сопротивление

t_0: при n_1 или n_2 $I = I_р$ (I расчетный)

t_1 { n_1: $I = 0{,}18\, I_р$

{ n_2: $I = I_р$

$n_1 + n_2$: ~~бли~~

1) При параллельном соединении $U_1 = U_2 = U = IR$

$U_1 = I_1(R + R_л)$

$U_2 = I_2 R$

$I_1(R + R_л) = I_2 R$

$0{,}18\, I_р (R + R_л) = I_р R$

$0{,}18 R + 0{,}18 R_л = R$

$R_л = \frac{1 - 0{,}18}{0{,}18} = 45{,}6$ Ом

не учтено внутреннее сопротивление источника

ответ: $R_л = 45{,}6$ Ом

⊖

Информатика

Решение задач по информатике не было предложено.

Командная часть

Результаты были получены в рамках выступления команды: ВИАС

Личный состав команды:

- Ярошевич Александр, 11 класс, Москва, школа 1375
- Гирин Иван, 11 класс, Москва, школа 1375
- Буссе Александр, 11 класс, Звенигово, ЗСОШ №3
- Сизов Иван, 10 класс, Вологда, Лицей №32

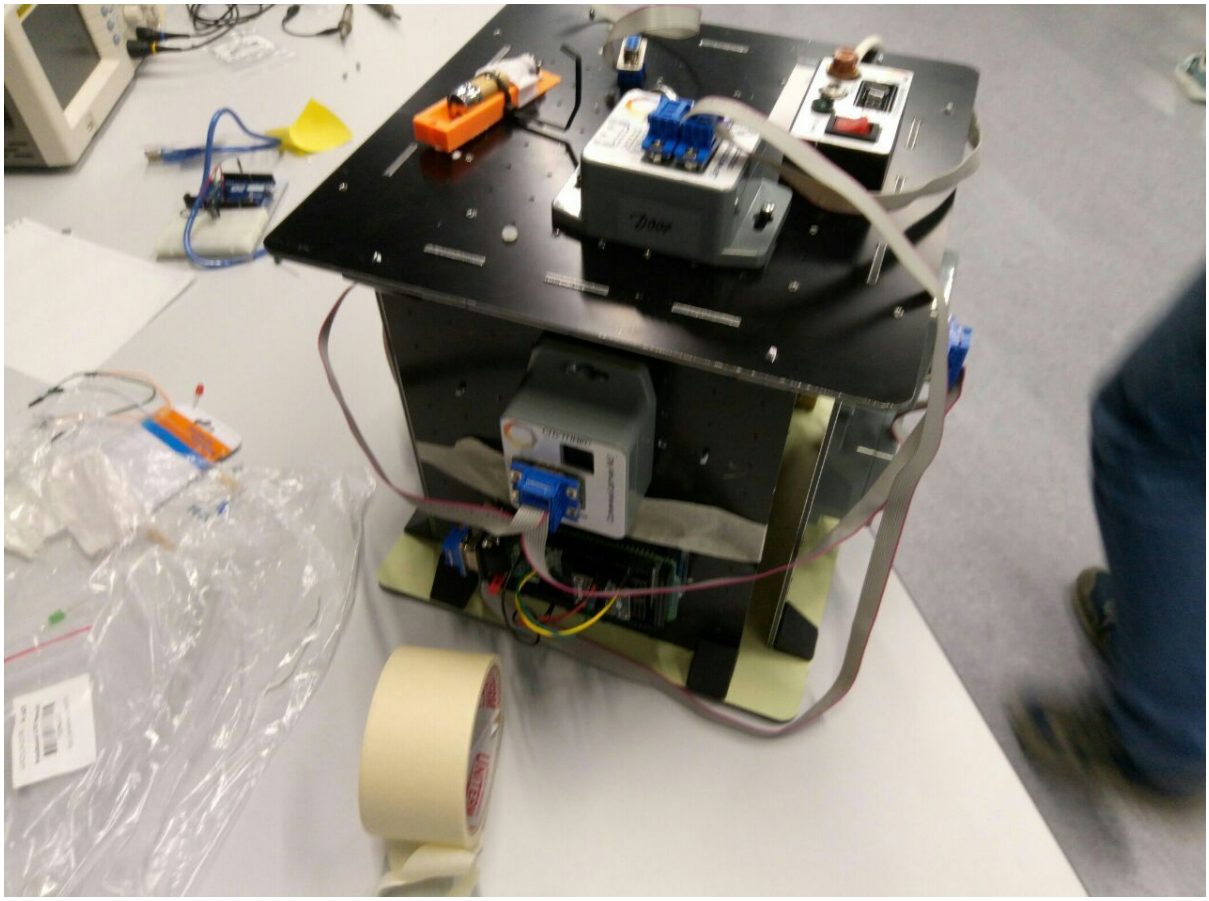
Команда «ВИАС» с финальной версией собранного устройства:

Протокол выполнения заданий

Задание	**Итог**
Задание 0. Подготовка к работе	Выполнено (2 балла)
Задание 1. Ориентация и стабилизация	Не выполнено (0 баллов) Максимальное достигнутое время удержания луча на мишени составило 8,5 секунд.
Задание 2. Оптический канал связи	Выполнена (9 баллов). Предложенный вариант решения не передавал символы 0x00. В итоге передано 6 из 9 сообщений.
Задание 3. Неизвестная планета	Выполнено (8 баллов)
Задание 4. Новый модуль спутника	Выполнено (28 баллов)
Задание 5. Канал телеметрии	Выполнено (20 баллов)
Задание 6. Миссия	Не выполнено (0 баллов)

Задание 0. Подготовка к работе

Собранный аппарат выглядел следующим образом:



Задание 1. Ориентация и стабилизация

Для выполнения задания был предложен следующий код:

```
#include "libschsat.h"
const int num = 1;
int16_t x,y,z,i,k,b,mSpeed,maxside;
uint16_t sun1[5],sun2[5];
//Стабилизация
//Можно поменять 7000 на 8000 чтобы быстрее стабилизироваться.
void stabil(void){
        int16_t SetSpeed;
   hyro_request_raw(num, &x, &y, &z);
        motor_request_speed(num, &mSpeed);
        while(abs(z)>250){
          if (abs(mSpeed)>8000){
                motor_set_speed(num, 0, &k);
          }
          SetSpeed=mSpeed-z;
          if (abs(z)>250){
                if (abs(SetSpeed)>8000){
                  motor_set_speed(num, 0, &k);
                }else{
                        motor_set_speed(num, SetSpeed, &k);
                }
          }
          printf("%d   %d\n",z,mSpeed);
          mSleep(200);
          motor_request_speed(num, &mSpeed);
          hyro_request_raw(num, &x, &y, &z);
        }
}
//Функция для поворота к солнцу
void sunturn(void){
        int16_t max,min,minside;
        max=0;
        min=30000;
        for (i=1;i<=4;i++){
                sun_sensor_request_raw(i, &sun1[i], &sun2[i]);
                printf("n%d  1: %d  2:%d\n",i,sun1[i],sun2[i]);
                /*if (i==2){
                        sun1[i]=sun1[i]*0.9;
                        sun2[i]=sun2[i]*0.9;
                }*/
                if (sun1[i]+sun2[i] > max){
                maxside=i;
                        max=sun1[i]+sun2[i];
                }
                if (sun1[i]+sun2[i] < min){
                  minside=i;
```

```
                min=sun1[i]+sun2[i];
            }
        }
        printf("maxside %d\n",maxside);
        motor_request_speed(num, &mSpeed);
        if (maxside=1){
            if ((sun1[4]+sun2[4]-sun1[2]-sun2[2])>15){
                motor_set_speed(num, mSpeed+100,&k);
           }else if((sun1[4]+sun2[4]-sun1[2]-sun2[2])<-15){
                motor_set_speed(num, mSpeed-100,&k);
            }
        }else{
            hyro_request_raw(num, &x, &y, &z);
            if (maxside=2){
                motor_set_speed(num, mSpeed-300,&k);
            }
            if (maxside=4){
                motor_set_speed(num, mSpeed+300,&k);
            }
            if (maxside=3){
                if ((sun1[4]+sun2[4])>(sun1[2]-sun1[2])){
                   motor_set_speed(num, mSpeed+600,&k);
                }else{
                   motor_set_speed(num, mSpeed-600,&k);
                }
            }
        }
}
void control(void){
        hyro_turn_on(num);
        mSleep(500);
    motor_turn_on(num);
        mSleep(500);
        for (i=1;i<=4;i++){
            sun_sensor_turn_on(i);
            mSleep(100);
        }
        if (LSS_OK == sun_sensor_request_raw(num, &sun1[1], &sun2[1])) {
            printf(" raw=%d\n", sun1[1]);
        } else {
            puts("Fail!");
        }
        if (LSS_OK == hyro_request_raw(num, &x, &y, &z)) {
            printf("%d: x=%d y=%d z=%d\n", num, x, y, z);
        } else {
            puts("Fail!");
        }
        if (LSS_OK == motor_request_speed(num, &mSpeed)) {
```

```
                printf("Got speed %d >>>\n", mSpeed);
        } else {
                puts("Fail! >>>");
        }
        motor_set_speed(num,-7000,&k);
        Sleep(15);
        motor_set_speed(num,0,&k);
        stabil();
        b=1;
        while(b>0){
          sunturn();
                if (maxside==1){
                if (abs((sun1[1]-sun2[1]))<2000){
                  mSleep(200);
                }else mSleep(500);
                }else
          mSleep(600);//Если mSleep поставить больше, то спутник будет поворачиваться к
солнцу быстрее, но не очень точно.
          stabil();
        }
}
```

Задание 2. Оптический канал связи

Код для передатчика:

```
#define tranPin 11

uint8_t buf[254], cnt = 0;

void UART_sleep(int mill)
{
 TCCR1A &= ~(1 << COM1A0);
 delay(mill);
 TCCR1A |= (1 << COM1A0);
}

int COBS_encode(uint8_t * buf, size_t sz = 254, bool ir=false)
{
 if (sz > 254)
  return 1;
 size_t last_null = 0;
 //Serial1.write(sz);
 //Serial.write(0xCC);
 //Serial.write(buf, sz);
 //Serial.write(0xDD);
 if (ir)
```

```
   UART_sleep(120);
  for (size_t i = 0; i <= sz; ++i)
  {
   if (i == sz || !buf[i])
   {
    size_t cnt = i - last_null + 1;
    if (ir)
    {Serial.write(cnt);
    Serial1.write(cnt);UART_sleep(120);}
    if (last_null != i)
    {
     //Serial.write((buf + last_null), cnt);
     for (size_t j = last_null; j < i; ++j)
      if(ir){Serial.write(buf[j]);
      Serial1.write(buf[j]);UART_sleep(120);}
    }
    last_null = i + 1;
   }
   //for (size_t j = i
  }
  if (!ir)
  {
  Serial.write(0);
  Serial1.write(0);
  }
  return 0;
};

uint8_t COBS_decode(uint8_t * buf)
{
  uint8_t i = 0xFF, next_null = 0;
  //while (Serial.available())
  while (true)
  {
   if (Serial.available())
   {
    uint8_t in_byte = Serial.read();
    //Serial.write(in_byte);
    if (!in_byte || i == 0xFE)
     break;
    if (i == 0xFF)
     next_null = in_byte - 1;
    else if (i == next_null)
    {
     next_null = i + in_byte;
     buf[i] = 0;
    }
    else
```

```
      buf[i] = in_byte;
     ++i;
    }
  }
  //Serial.write(0xAA);
  return (i < 0xFE) ? i : 0xFF;
}

void PWM_init()
{
  pinMode(tranPin, OUTPUT);

  // initialize timer1
  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;

  OCR1A = 141;            // compare match register 16MHz/1/140KHz
  TCCR1A |= (1 << COM1A0);
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS10);    // 1 prescaler
  //TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}

/*ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  digitalWrite(tranPin, digitalRead(tranPin) ^ 1);   // toggle LED pin
}*/

ISR(PCINT0_vect)
{
   if (TCCR1A & (1 << COM1A0))
    TCCR1A &= ~(1 << COM1A0);
   else
    TCCR1A |= 1 << COM1A0;
   /*if (!(TCCR1A & (1 << COM1A0)))
    TCCR1A |= 1 << COM1A0;
    delay(2);
    TCCR1A &= ~(1 << COM1A0);*/
}

/*void tran_init()
{
  // initialize timer3
  noInterrupts();           // disable all interrupts
  TCCRA = 0;
```

```
  TCCR3B = 0;
  TCNT3  = 0;

  OCR2A = 833;            // compare match register 16MHz/64/300Hz
  TCCR3B |= (1 << WGM12);   // CTC mode
  TCCR3B |= (1 << CS10) | (1 << CS11);    // 64 prescaler
  TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}

ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  digitalWrite(tranPin, digitalRead(tranPin) ^ 1);   // toggle LED pin
}*/

int Hamming_encode(uint16_t * ibuf, uint32_t * obuf, size_t sz=16)
{
    if (sz > 254)
      return 1;
}

void setup() {
  // put your setup code here, to run once:
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT4);

  Serial.begin(115200);
  Serial1.begin(100);
  //uint8_t buf[254];
  for (size_t i = 0; i < 254; ++i)
    buf[i] = i + 0xAA; //i + 1;// i % 3;
  //Serial.write(buf, 254);
  //Serial.write(0xFF);
  //COBS_encode(buf);
  PWM_init();
  Serial.write(buf, 7);
}


void loop() {
  // put your main code here, to run repeatedly:
  /*if (Serial1.available())
  {
    in_byte = Serial1.read();
    Serial.println(in_byte, HEX);
  }*/

  //if (Serial.available())
```

```
//{
   //uint8_t in_byte = Serial.read();
   //if (in_byte == 0xEE)
   //{
     //Serial.println(cnt);
     //Serial.write(buf,cnt);
     //Serial.println();
     //COBS_encode(buf, 1);
     //Serial.write(0xAA);
     //COBS_encode();
     //cnt = 0;
   //}
   //else
    //buf[cnt++] = in_byte;
//}
/*if (Serial.available())
{
  uint8_t cnt = COBS_decode(buf);
  Serial.write(cnt);
  Serial.write(0xBB);
  Serial.write(buf, cnt);
}*/

//if (Serial.available())
//{
  //cnt = 0;
  //while (Serial.available())
  //{
    //uint8_t in_byte = Serial.read();
    //buf[cnt++] = in_byte;
  //}
  //Serial.write(buf[0]);
  //Serial1.write(buf[0]);
  /*UART_sleep(120);
  for (size_t i = 0; i < cnt; ++i)
  {
    Serial.write(buf[i]);
    Serial1.write(buf[i]);
    UART_sleep(120);
  }*/
  COBS_encode(buf,32,true);
//}
//Serial1.write(0x01);
//Serial1.write(buf[cnt++]);
//for (size_t i = 0; i < 5; ++i);
  //Serial1.write(0x00);
//COBS_encode(buf, 7);
//delay(2);
```

```
  //UART_sleep(120);
  //if (cnt == 10)
    //cnt = 0;
}
```

Код для приемника:

```
#define rxPin 7
uint8_t buf[254], cnt = 0;
uint8_t COBS_decode(uint8_t * buf)
{
  uint8_t i = 0xFF, next_null = 0;
  //while (Serial.available())
  while (true)
  {
    if (Serial1.available())
    {
      uint8_t in_byte = Serial1.read();
      //Serial.write(in_byte);
      if (!in_byte || i == 0xFE)
        break;
      if (i == 0xFF)
        next_null = in_byte - 1;
      else if (i == next_null)
      {
        next_null = i + in_byte;
        buf[i] = 0;
      }
      else
        buf[i] = in_byte;
      ++i;
    }
  }
  //Serial.write(0xAA);
  return (i < 0xFE) ? i : 0xFF;
}

void setup() {
  // put your setup code here, to run once:
  //PCICR |= (1 << PCIE0);
  //PCMSK0 |= (1 << PCINT4);

  Serial.begin(115200);
  Serial1.begin(100);
  //uint8_t buf[254];
  //for (size_t i = 0; i < 254; ++i)
  //  buf[i] = 0xAA; //i + 1;// i % 3;
  //Serial.write(buf, 254);
```

```
  //Serial.write(0xFF);
  //COBS_encode(buf);
  //PWM_init();
  //Serial.write(buf, 7);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial1.available())
  {
    //uint8_t in_byte = Serial1.read();
    //Serial.write(in_byte);
    cnt = COBS_decode(buf);
    Serial.write(buf[0]);
    Serial.write(buf + 2,cnt);
  }
}
```

Задание 3. Неизвестная планета

Для выполнения задания был предложен следующий код:

```
#include "libschsat.h"
#include "time.h"

void control()
{
	transceiver_turn_on(2);
	sleep(1);
	char a[]="                                                                                ";
	int start,tm,data;
	time(&start);
	bus_setup();
	//transceiver_request_reset(2);
	//[10];for(int i=0;i<10;i++){a[i]='f';}
	//transceiver_send(2,1,&a,1);
	while(1){

	//const char hello[] = "hello, world!";
	/*if (LSS_OK != transceiver_send(2, 1, (uint8_t *) hello, sizeof(hello)))
		puts("Fail!");*/
	//transceiver_request_buff(2,a);
	//transceiver_request_buff(2,&data);
	/*for (int i=0;i<10;i++)
	{*/
		transceiver_request_buff(2,&a);
		printf("%s",a);
	//}


	/*for(int i=0;i<10;i++)
	{
		printf("%s",a[i]);
	}*/
	sleep(1);

	}

}
```

Задание 4. Новый модуль спутника

Для выполнения задания был предложен следующий код:

```
#define tranPin 11

uint8_t buf[254], cnt = 0;

bool transmit = 0, strtflag = 0;

char to_addr_str[100], from_addr_str[100], msg_type_str[100];

/*int COBS_encode(uint8_t * buf, size_t sz = 254)
{
 if (sz > 254)
  return 1;
 size_t last_null = 0;
 //Serial1.write(sz);
 Serial.write(0xCC);
 Serial.write(buf, sz);
 Serial.write(0xDD);
 for (size_t i = 0; i <= sz; ++i)
 {
  if (i == sz || !buf[i])
  {
   size_t cnt = i - last_null + 1;
   Serial.write(cnt);
   Serial1.write(cnt);
   if (last_null != i)
   {
    //Serial.write((buf + last_null), cnt);
    for (size_t j = last_null; j < i; ++j)
     {Serial.write(buf[j]);
     Serial1.write(buf[j]);}
   }
   last_null = i + 1;
  }
  //for (size_t j = i
 }
 Serial.write(0);
 Serial1.write(0);

 return 0;
};*/

uint8_t COBS_decode(uint8_t * buf)
{
 uint8_t i = 0xFF, next_null = 0;
 //while (Serial.available())
```

```
  while (true)
  {
   if (Serial1.available())
   {
    uint8_t in_byte = Serial1.read();
    //Serial.write(in_byte);
    if (!in_byte || i == 0xFE)
     break;
    if (i == 0xFF)
     next_null = in_byte - 1;
    else if (i == next_null)
    {
     next_null = i + in_byte;
     buf[i] = 0;
    }
    else
     buf[i] = in_byte;
    ++i;
   }
  }
  //Serial.write(0xAA);
  return (i < 0xFE) ? i : 0xFF;
}

/*void PWM_init()
{
  pinMode(tranPin, OUTPUT);

  // initialize timer1
  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;

  OCR1A = 141;            // compare match register 16MHz/1/140KHz
  TCCR1A |= (1 << COM1A0);
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS10);    // 1 prescaler
  //TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}*/

/*ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  digitalWrite(tranPin, digitalRead(tranPin) ^ 1);   // toggle LED pin
}*/

ISR(PCINT0_vect)
```

```
{
  transmit = (transmit) ? 0 : 1;
}

int msg_type_decode(char * res_str, uint16_t msg_type)
{
 char *msg_type_str;
 //Serial.println("checking ");
 //Serial.println(msg_type, HEX);
 switch (msg_type)
 {
  case (0x0000): msg_type_str="RPI_OBC_BASE"; break;
  case (0x0200): msg_type_str="SENS_LIGHT_REQ_RAW"; break;
  case (0x0208): msg_type_str="SENS_LIGHT_CMD_RESET"; break;
  case (0x0210): msg_type_str="SENS_LIGHT_REQ_MAXRAW"; break;
  case (0x0220): msg_type_str="SENS_LIGHT_SET_MINVALUE"; break;
  case (0x0228): msg_type_str="SENS_LIGHT_SET_CALIBRATE"; break;
  case (0x0280): msg_type_str="SENS_LIGHT_RESP_RAW"; break;
  case (0x0288): msg_type_str="SENS_LIGHT_RESP_MAXRAW"; break;
  case (0x0300): msg_type_str="SENS_MAG_REQ_RAW"; break;
  case (0x0304): msg_type_str="SENS_MAG_CMD_RESET"; break;
  case (0x0380): msg_type_str="SENS_MAG_RESP_RAW"; break;
  case (0x0400): msg_type_str="SENS_HYRO_REQ_RAW"; break;
  case (0x0404): msg_type_str="SENS_HYRO_CMD_RESET"; break;
  case (0x0480): msg_type_str="SENS_HYRO_RESP_RAW"; break;
  case (0x0500): msg_type_str="SENS_SUN_REQ_RAW"; break;
  case (0x0508): msg_type_str="SENS_SUN_CMD_RESET"; break;
  case (0x0510): msg_type_str="SENS_SUN_REQ_MAXRAW"; break;
  case (0x0518): msg_type_str="SENS_SUN_SET_MINVALUE"; break;
  case (0x0580): msg_type_str="SENS_SUN_RESP_RAW"; break;
  case (0x0588): msg_type_str="SENS_SUN_RESP_MAXRAW"; break;
  case (0x0600): msg_type_str="SENS_ACCEL_REQ_RAW"; break;
  case (0x0604): msg_type_str="SENS_ACCEL_CMD_RESET"; break;
  case (0x0680): msg_type_str="SENS_ACCEL_RESP_RAW"; break;
  case (0x0D00): msg_type_str="CONTROL_WHEEL0_SET_SPEED"; break;
  case (0x0D04): msg_type_str="CONTROL_WHEEL0_CMD_RESET"; break;
  case (0x0D08): msg_type_str="CONTROL_WHEEL0_REQ_SPEED"; break;
  case (0x0D10): msg_type_str="CONTROL_WHEEL0_RESP_SPEED_CONFIRM"; break;
  case (0x0D0C): msg_type_str="CONTROL_WHEEL0_RESP_SPEED"; break;
  case (0x0E00): msg_type_str="CONTROL_FAN0_SET_SPEED"; break;
  case (0x0E04): msg_type_str="CONTROL_FAN0_CMD_RESET"; break;
  case (0x0E08): msg_type_str="CONTROL_FAN0_REQ_SPEED"; break;
  case (0x0E10): msg_type_str="CONTROL_FAN0_RESP_SPEED_CONFIRM"; break;
  case (0x0E0C): msg_type_str="CONTROL_FAN0_RESP_SPEED"; break;
  case (0x0F00): msg_type_str="CONTROL_COIL_SET_VAL"; break;
  case (0x0F10): msg_type_str="CONTROL_COIL_CMD_RESET"; break;
  case (0x0F20): msg_type_str="CONTROL_COIL_RESP_VAL_CONFIRM"; break;
  case (0x1000): msg_type_str="CONTROL_ENGINE_SET_VAL"; break;
```

```
    case (0x1010): msg_type_str="CONTROL_ENGINE_CMD_RESET"; break;
    case (0x1020): msg_type_str="CONTROL_ENGINE_RESP_VAL_CONFIRM"; break;
    case (0x2000): msg_type_str="MISC_LASER_CMD_ON"; break;
    case (0x2004): msg_type_str="MISC_LASER_CMD_OFF"; break;
    case (0x2008): msg_type_str="MISC_LASER_CMD_RESET"; break;
    case (0x200C): msg_type_str="MISC_LASER_RESP_ON_CONFIRM"; break;
    case (0x2010): msg_type_str="MISC_LASER_RESP_OFF_CONFIRM"; break;
    case (0x2100): msg_type_str="MISC_HF_REQ_STATE"; break;
    case (0x2110): msg_type_str="MISC_HF_CMD_RESET"; break;
    case (0x2120): msg_type_str="MISC_HF_RESP_STATE"; break;
    case (0x2200): msg_type_str="MISC_UHF_CMD_SEND"; break;
    case (0x2210): msg_type_str="MISC_UHF_CMD_RESET"; break;
    case (0x2220): msg_type_str="MISC_UHF_REQ_BUFFER"; break;
    case (0x2230): msg_type_str="MISC_UHF_RESP_BUFFER"; break;
    case (0x2300): msg_type_str="MISC_SUN_REQ_RAW"; break;
    case (0x2308): msg_type_str="MISC_SUN_CMD_RESET"; break;
    case (0x2310): msg_type_str="MISC_SUN_REQ_MAXRAW"; break;
    case (0x2380): msg_type_str="MISC_SUN_RESP_RAW"; break;
    case (0x2388): msg_type_str="MISC_SUN_RESP_MAXRAW"; break;
    case (0x2400): msg_type_str="MISC_EARTHFIELD_CMD_SET"; break;
    case (0x2408): msg_type_str="MISC_EARTHFIELD_CMD_AMPRESET"; break;
    case (0x2410): msg_type_str="MISC_EARTHFIELD_CMD_RESET"; break;
    case (0x2500): msg_type_str="MISC_GROUNDLEDS_CMD_SET"; break;
    case (0x2510): msg_type_str="MISC_GROUNDLEDS_CMD_RESET"; break;
    case (0x2600): msg_type_str="MISC_MOTOGLOBE_CMD_GO_SLOW"; break;
    case (0x2610): msg_type_str="MISC_MOTOGLOBE_CMD_GO_FAST"; break;
    case (0x2620): msg_type_str="MISC_MOTOGLOBE_CMD_START"; break;
    case (0x2630): msg_type_str="MISC_MOTOGLOBE_CMD_STOP"; break;
    case (0x2640): msg_type_str="MISC_MOTOGLOBE_CMD_ENABLEINT"; break;
    case (0x2650): msg_type_str="MISC_MOTOGLOBE_CMD_DISABLEINT"; break;
    case (0x2660): msg_type_str="MISC_MOTOGLOBE_CMD_RESET"; break;
    case (0x2700): msg_type_str="MISC_ARM_CMD_SET_STATE"; break;
    case (0x2710): msg_type_str="MISC_ARM_CMD_RESET"; break;
    case (0x2720): msg_type_str="MISC_ARM_RESP_STATE"; break;
    default: return 1; break;
  }
  strcpy(res_str, msg_type_str);
  return 0;
}

int addr_decode(char * res_str, uint16_t addr)
{
  char *addr_str;
  //Serial.println("checking ");
  //Serial.println(addr, HEX);
  switch (addr)
  {
    case (0x0001): addr_str="OBC_ADDRESS"; break;
```

```
    case (0x0010): addr_str="SENS_LIGHT_ADDRESS"; break;
    case (0x0018): addr_str="SENS_MAG_ADDRESS"; break;
    case (0x001C): addr_str="SENS_HYRO_ADDRESS"; break;
    case (0x0020): addr_str="SENS_SUN_ADDRESS"; break;
    case (0x0028): addr_str="SENS_ACCEL_ADDRESS"; break;
    case (0x0100): addr_str="CONTROL_WHEEL0_ADDRESS"; break;
    case (0x0110): addr_str="CONTROL_FAN0_ADDRESS"; break;
    case (0x0120): addr_str="CONTROL_COIL_ADDRESS"; break;
    case (0x0130): addr_str="CONTROL_ENGINE_ADDRESS"; break;
    case (0x0200): addr_str="MISC_LASER_ADDRESS"; break;
    case (0x0210): addr_str="MISC_HF_ADDRESS"; break;
    case (0x0220): addr_str="MISC_UHF_ADDRESS"; break;
    case (0x0230): addr_str="MISC_SUN_ADDRESS"; break;
    case (0x0240): addr_str="MISC_EARTHFIELD_ADDRESS"; break;
    case (0x0250): addr_str="MISC_GROUNDLEDS_ADDRESS"; break;
    case (0x0260): addr_str="MISC_MOTOGLOBE_ADDRESS"; break;
    case (0x0270): addr_str="MISC_ARM_ADDRESS"; break;
    case (0xFFFF): addr_str="MISC_ADDRESS_BROADCAST"; break;
    default: return 1; break;
  }
  strcpy(res_str, addr_str);
  return 0;
}

void setup() {
  // put your setup code here, to run once:
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT4);

  Serial.begin(115200);
  Serial1.begin(115200);
}


void loop() {
  // put your main code here, to run repeatedly:
  /*if (Serial1.available())
  {
    in_byte = Serial1.read();
    Serial.println(in_byte, HEX);
  }*/


  /*if (transmit)
  {
    if (Serial.available())
    {
      uint8_t out_byte = Serial.read();
```

```
    Serial1.write(out_byte);
    //COBS_encode(buf
  }
}
else
{
  if (Serial1.available())
  {
    uint8_t in_byte = Serial.read();
    Serial.write(in_byte);
  }
}*/
//if (Serial1.available())
if (Serial1.available())
{
  //Serial.println("ololo");

  //uncomment this!!!!
  uint8_t cnt = COBS_decode(buf);
  /*Serial.write(0xAA);
  Serial.write(cnt);
  Serial.write(0xBB);
  Serial.write(buf, cnt);
  Serial.write(0xCC);*/

  /*for (size_t i = 0; i < 6; ++i)
  {
    while (!Serial.available()) {};
    buf[i] = Serial.read();
  }*/


  uint16_t msg_type = ((uint16_t*)buf)[0],
  to_addr = ((uint16_t*)buf)[1],
  from_addr = ((uint16_t*)buf)[2];

  if (msg_type == 0x2308)
    strtflag = 1;
  if (!strtflag)
    return;

  /*Serial.write(msg_type);
  Serial.write(msg_type >> 8);
  Serial.write(0xDD);
  Serial.write(to_addr);
  Serial.write(to_addr >> 8);
  Serial.write(0xEE);
  Serial.write(from_addr);
```

```
    Serial.write(from_addr >> 8);
    Serial.write(0xFF);*/


    /*Serial.println(msg_type, HEX);
    //Serial.write(msg_type >> 8);
    //Serial.write(0xDD);
    Serial.println(to_addr, HEX);
    //Serial.write(to_addr >> 8);
    //Serial.write(0xEE);
    Serial.println(from_addr, HEX);
    //Serial.write(from_addr >> 8);
    //Serial.write(0xFF);*/

    if (to_addr == 0x0222)
    {
    Serial.println();
    uint16_t c_msg_type = msg_type, c_to_addr = to_addr, c_from_addr = from_addr;
    bool aflag = true;

    while (msg_type - c_msg_type < 5 && msg_type_decode(msg_type_str, c_msg_type) || (aflag =
false) == true)
    {
      --c_msg_type;
      //aflag = false;
    }
    if (!aflag)
    {
      Serial.print("type ");
      Serial.print(msg_type_str);
      Serial.print(' ');
      Serial.print(msg_type - c_msg_type);
      Serial.println();
    }
    else
      Serial.println("msg_type_fail");


    aflag = true;
    while (to_addr - c_to_addr < 5 && addr_decode(to_addr_str, c_to_addr) || (aflag = false) == true)
    {
      --c_to_addr;
      //aflag = false;
    }
    if (!aflag)
    {
      Serial.print("to ");
      Serial.print(to_addr_str);
```

```
    Serial.print(' ');
    Serial.print(to_addr - c_to_addr);
    Serial.println();
  }
  else
    Serial.println("to_addr_fail");

  aflag = true;
  while (from_addr - c_from_addr < 15 && addr_decode(from_addr_str, c_from_addr) || (aflag =
false) == true)
  {
    --c_from_addr;
    //aflag = false;
  }
  if (!aflag)
  {
    Serial.print("from ");
    Serial.print(from_addr_str);
    Serial.print(' ');
    Serial.print(from_addr - c_from_addr);
    Serial.println();
  }
  else
    Serial.println("from_addr_fail");



  //if (to_addr == 0x0222)
  //{
    //Serial.write(buf, cnt);
    for (size_t i = 6; i < cnt; ++i)
    {
      char nice_out[10];
      //sprintf(nice_out, "%x ", buf[i]);
      Serial.print(buf[i], HEX);
      Serial.print(' ');
      //Serial.write(nice_out, 2);
    }

    Serial.println();
  //}

  /*uint8_t in_byte = Serial1.read();

  if (!in_byte and cnt++)
  {
    Serial.write(in_byte);
    Serial.println();
```

```
  }
  else if (in_byte)
  {
   Serial.write(in_byte);
   cnt = 0;
  }*/
  }
 }

 /*if (Serial.available())
 {
  uint8_t cnt = COBS_decode(buf);
  Serial.write(cnt);
  Serial.write(0xBB);
  Serial.write(buf, cnt);
 }*/
 //Serial.write(buf, 7);
 //COBS_encode(buf, 7);
 //delay(100);
}
```

Задание 5. Канал телеметрии

```
#include "libschsat.h"
#include "time.h"

void control()
{
        transceiver_turn_on(2);
        sleep(1);
        char a[]="                                                                                ";
        int start,tm,data;
        time(&start);
        bus_setup();
        //transceiver_request_reset(2);
        //[10];for(int i=0;i<10;i++){a[i]='f';}
        //transceiver_send(2,1,&a,1);
        while(1){
        time(&tm);
        //tm=tm-start;
        //const char hello[] = "hello, world!";
        /*if (LSS_OK != transceiver_send(2, 1, (uint8_t *) hello, sizeof(hello)))
                puts("Fail!");*/
        //transceiver_request_buff(2,a);
        //transceiver_request_buff(2,&data);
        /*for (int i=0;i<10;i++)
        {*/
                transceiver_request_buff(2,&a);
```

```
                printf("%s",a);
        //}
        tm=tm-start;
        printf("\nВИАС:time %d start %d;\n",tm);

        /*for(int i=0;i<10;i++)
        {
                printf("%s",a[i]);
        }*/
        sleep(1);

        }

}
```

Задание 6. Миссия

Для выполнения задания был предложен следующий код:

```
#define tranPin 11

uint8_t buf[254], cnt = 0;

bool transmit = 0, strtflag = 0;

char to_addr_str[100], from_addr_str[100], msg_type_str[100];

/*int COBS_encode(uint8_t * buf, size_t sz = 254)
{
 if (sz > 254)
   return 1;
 size_t last_null = 0;
 //Serial1.write(sz);
 Serial.write(0xCC);
 Serial.write(buf, sz);
 Serial.write(0xDD);
 for (size_t i = 0; i <= sz; ++i)
 {
  if (i == sz || !buf[i])
  {
   size_t cnt = i - last_null + 1;
   Serial.write(cnt);
   Serial1.write(cnt);
   if (last_null != i)
   {
    //Serial.write((buf + last_null), cnt);
    for (size_t j = last_null; j < i; ++j)
```

```
        {Serial.write(buf[j]);
        Serial1.write(buf[j]);}
      }
      last_null = i + 1;
    }
    //for (size_t j = i
  }
  Serial.write(0);
  Serial1.write(0);

  return 0;
};*/

uint8_t COBS_decode(uint8_t * buf)
{
  uint8_t i = 0xFF, next_null = 0;
  //while (Serial.available())
  while (true)
  {
    if (Serial1.available())
    {
      uint8_t in_byte = Serial1.read();
      //Serial.write(in_byte);
      if (!in_byte || i == 0xFE)
        break;
      if (i == 0xFF)
        next_null = in_byte - 1;
      else if (i == next_null)
      {
        next_null = i + in_byte;
        buf[i] = 0;
      }
      else
        buf[i] = in_byte;
      ++i;
    }
  }
  //Serial.write(0xAA);
  return (i < 0xFE) ? i : 0xFF;
}

/*void PWM_init()
{
  pinMode(tranPin, OUTPUT);

  // initialize timer1
  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
```

```
  TCCR1B = 0;
  TCNT1  = 0;

  OCR1A = 141;           // compare match register 16MHz/1/140KHz
  TCCR1A |= (1 << COM1A0);
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS10);    // 1 prescaler
  //TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}*/

/*ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  digitalWrite(tranPin, digitalRead(tranPin) ^ 1);   // toggle LED pin
}*/

ISR(PCINT0_vect)
{
   transmit = (transmit) ? 0 : 1;
}

int msg_type_decode(char * res_str, uint16_t msg_type)
{
  char *msg_type_str;
  //Serial.println("checking ");
  //Serial.println(msg_type, HEX);
  switch (msg_type)
  {
    case (0x0000): msg_type_str="RPI_OBC_BASE"; break;
    case (0x0200): msg_type_str="SENS_LIGHT_REQ_RAW"; break;
    case (0x0208): msg_type_str="SENS_LIGHT_CMD_RESET"; break;
    case (0x0210): msg_type_str="SENS_LIGHT_REQ_MAXRAW"; break;
    case (0x0220): msg_type_str="SENS_LIGHT_SET_MINVALUE"; break;
    case (0x0228): msg_type_str="SENS_LIGHT_SET_CALIBRATE"; break;
    case (0x0280): msg_type_str="SENS_LIGHT_RESP_RAW"; break;
    case (0x0288): msg_type_str="SENS_LIGHT_RESP_MAXRAW"; break;
    case (0x0300): msg_type_str="SENS_MAG_REQ_RAW"; break;
    case (0x0304): msg_type_str="SENS_MAG_CMD_RESET"; break;
    case (0x0380): msg_type_str="SENS_MAG_RESP_RAW"; break;
    case (0x0400): msg_type_str="SENS_HYRO_REQ_RAW"; break;
    case (0x0404): msg_type_str="SENS_HYRO_CMD_RESET"; break;
    case (0x0480): msg_type_str="SENS_HYRO_RESP_RAW"; break;
    case (0x0500): msg_type_str="SENS_SUN_REQ_RAW"; break;
    case (0x0508): msg_type_str="SENS_SUN_CMD_RESET"; break;
    case (0x0510): msg_type_str="SENS_SUN_REQ_MAXRAW"; break;
    case (0x0518): msg_type_str="SENS_SUN_SET_MINVALUE"; break;
    case (0x0580): msg_type_str="SENS_SUN_RESP_RAW"; break;
    case (0x0588): msg_type_str="SENS_SUN_RESP_MAXRAW"; break;
```

```
case (0x0600): msg_type_str="SENS_ACCEL_REQ_RAW"; break;
case (0x0604): msg_type_str="SENS_ACCEL_CMD_RESET"; break;
case (0x0680): msg_type_str="SENS_ACCEL_RESP_RAW"; break;
case (0x0D00): msg_type_str="CONTROL_WHEEL0_SET_SPEED"; break;
case (0x0D04): msg_type_str="CONTROL_WHEEL0_CMD_RESET"; break;
case (0x0D08): msg_type_str="CONTROL_WHEEL0_REQ_SPEED"; break;
case (0x0D10): msg_type_str="CONTROL_WHEEL0_RESP_SPEED_CONFIRM"; break;
case (0x0D0C): msg_type_str="CONTROL_WHEEL0_RESP_SPEED"; break;
case (0x0E00): msg_type_str="CONTROL_FAN0_SET_SPEED"; break;
case (0x0E04): msg_type_str="CONTROL_FAN0_CMD_RESET"; break;
case (0x0E08): msg_type_str="CONTROL_FAN0_REQ_SPEED"; break;
case (0x0E10): msg_type_str="CONTROL_FAN0_RESP_SPEED_CONFIRM"; break;
case (0x0E0C): msg_type_str="CONTROL_FAN0_RESP_SPEED"; break;
case (0x0F00): msg_type_str="CONTROL_COIL_SET_VAL"; break;
case (0x0F10): msg_type_str="CONTROL_COIL_CMD_RESET"; break;
case (0x0F20): msg_type_str="CONTROL_COIL_RESP_VAL_CONFIRM"; break;
case (0x1000): msg_type_str="CONTROL_ENGINE_SET_VAL"; break;
case (0x1010): msg_type_str="CONTROL_ENGINE_CMD_RESET"; break;
case (0x1020): msg_type_str="CONTROL_ENGINE_RESP_VAL_CONFIRM"; break;
case (0x2000): msg_type_str="MISC_LASER_CMD_ON"; break;
case (0x2004): msg_type_str="MISC_LASER_CMD_OFF"; break;
case (0x2008): msg_type_str="MISC_LASER_CMD_RESET"; break;
case (0x200C): msg_type_str="MISC_LASER_RESP_ON_CONFIRM"; break;
case (0x2010): msg_type_str="MISC_LASER_RESP_OFF_CONFIRM"; break;
case (0x2100): msg_type_str="MISC_HF_REQ_STATE"; break;
case (0x2110): msg_type_str="MISC_HF_CMD_RESET"; break;
case (0x2120): msg_type_str="MISC_HF_RESP_STATE"; break;
case (0x2200): msg_type_str="MISC_UHF_CMD_SEND"; break;
case (0x2210): msg_type_str="MISC_UHF_CMD_RESET"; break;
case (0x2220): msg_type_str="MISC_UHF_REQ_BUFFER"; break;
case (0x2230): msg_type_str="MISC_UHF_RESP_BUFFER"; break;
case (0x2300): msg_type_str="MISC_SUN_REQ_RAW"; break;
case (0x2308): msg_type_str="MISC_SUN_CMD_RESET"; break;
case (0x2310): msg_type_str="MISC_SUN_REQ_MAXRAW"; break;
case (0x2380): msg_type_str="MISC_SUN_RESP_RAW"; break;
case (0x2388): msg_type_str="MISC_SUN_RESP_MAXRAW"; break;
case (0x2400): msg_type_str="MISC_EARTHFIELD_CMD_SET"; break;
case (0x2408): msg_type_str="MISC_EARTHFIELD_CMD_AMPRESET"; break;
case (0x2410): msg_type_str="MISC_EARTHFIELD_CMD_RESET"; break;
case (0x2500): msg_type_str="MISC_GROUNDLEDS_CMD_SET"; break;
case (0x2510): msg_type_str="MISC_GROUNDLEDS_CMD_RESET"; break;
case (0x2600): msg_type_str="MISC_MOTOGLOBE_CMD_GO_SLOW"; break;
case (0x2610): msg_type_str="MISC_MOTOGLOBE_CMD_GO_FAST"; break;
case (0x2620): msg_type_str="MISC_MOTOGLOBE_CMD_START"; break;
case (0x2630): msg_type_str="MISC_MOTOGLOBE_CMD_STOP"; break;
case (0x2640): msg_type_str="MISC_MOTOGLOBE_CMD_ENABLEINT"; break;
case (0x2650): msg_type_str="MISC_MOTOGLOBE_CMD_DISABLEINT"; break;
case (0x2660): msg_type_str="MISC_MOTOGLOBE_CMD_RESET"; break;
```

```
    case (0x2700): msg_type_str="MISC_ARM_CMD_SET_STATE"; break;
    case (0x2710): msg_type_str="MISC_ARM_CMD_RESET"; break;
    case (0x2720): msg_type_str="MISC_ARM_RESP_STATE"; break;
    default: return 1; break;
  }
  strcpy(res_str, msg_type_str);
  return 0;
}

int addr_decode(char * res_str, uint16_t addr)
{
  char *addr_str;
  //Serial.println("checking ");
  //Serial.println(addr, HEX);
  switch (addr)
  {
    case (0x0001): addr_str="OBC_ADDRESS"; break;
    case (0x0010): addr_str="SENS_LIGHT_ADDRESS"; break;
    case (0x0018): addr_str="SENS_MAG_ADDRESS"; break;
    case (0x001C): addr_str="SENS_HYRO_ADDRESS"; break;
    case (0x0020): addr_str="SENS_SUN_ADDRESS"; break;
    case (0x0028): addr_str="SENS_ACCEL_ADDRESS"; break;
    case (0x0100): addr_str="CONTROL_WHEEL0_ADDRESS"; break;
    case (0x0110): addr_str="CONTROL_FAN0_ADDRESS"; break;
    case (0x0120): addr_str="CONTROL_COIL_ADDRESS"; break;
    case (0x0130): addr_str="CONTROL_ENGINE_ADDRESS"; break;
    case (0x0200): addr_str="MISC_LASER_ADDRESS"; break;
    case (0x0210): addr_str="MISC_HF_ADDRESS"; break;
    case (0x0220): addr_str="MISC_UHF_ADDRESS"; break;
    case (0x0230): addr_str="MISC_SUN_ADDRESS"; break;
    case (0x0240): addr_str="MISC_EARTHFIELD_ADDRESS"; break;
    case (0x0250): addr_str="MISC_GROUNDLEDS_ADDRESS"; break;
    case (0x0260): addr_str="MISC_MOTOGLOBE_ADDRESS"; break;
    case (0x0270): addr_str="MISC_ARM_ADDRESS"; break;
    case (0xFFFF): addr_str="MISC_ADDRESS_BROADCAST"; break;
    default: return 1; break;
  }
  strcpy(res_str, addr_str);
  return 0;
}

void setup() {
  // put your setup code here, to run once:
  PCICR |= (1 << PCIE0);
  PCMSK0 |= (1 << PCINT4);

  Serial.begin(115200);
  Serial1.begin(115200);
```

```
}


void loop() {
  // put your main code here, to run repeatedly:
  /*if (Serial1.available())
  {
    in_byte = Serial1.read();
    Serial.println(in_byte, HEX);
  }*/


  /*if (transmit)
  {
   if (Serial.available())
   {
     uint8_t out_byte = Serial.read();
     Serial1.write(out_byte);
     //COBS_encode(buf
   }
  }
  else
  {
   if (Serial1.available())
   {
     uint8_t in_byte = Serial.read();
     Serial.write(in_byte);
   }
  }*/
  //if (Serial1.available())
  if (Serial1.available())
  {
   //Serial.println("ololo");

   //uncomment this!!!!
   uint8_t cnt = COBS_decode(buf);
   /*Serial.write(0xAA);
   Serial.write(cnt);
   Serial.write(0xBB);
   Serial.write(buf, cnt);
   Serial.write(0xCC);*/

   /*for (size_t i = 0; i < 6; ++i)
   {
    while (!Serial.available()) {};
    buf[i] = Serial.read();
   }*/
```

```
  uint16_t msg_type = ((uint16_t*)buf)[0],
  to_addr = ((uint16_t*)buf)[1],
  from_addr = ((uint16_t*)buf)[2];

  if (msg_type == 0x2308)
    strtflag = 1;
  if (!strtflag)
    return;

  /*Serial.write(msg_type);
  Serial.write(msg_type >> 8);
  Serial.write(0xDD);
  Serial.write(to_addr);
  Serial.write(to_addr >> 8);
  Serial.write(0xEE);
  Serial.write(from_addr);
  Serial.write(from_addr >> 8);
  Serial.write(0xFF);*/


  /*Serial.println(msg_type, HEX);
  //Serial.write(msg_type >> 8);
  //Serial.write(0xDD);
  Serial.println(to_addr, HEX);
  //Serial.write(to_addr >> 8);
  //Serial.write(0xEE);
  Serial.println(from_addr, HEX);
  //Serial.write(from_addr >> 8);
  //Serial.write(0xFF);*/

  if (to_addr == 0x0222)
  {
  Serial.println();
  uint16_t c_msg_type = msg_type, c_to_addr = to_addr, c_from_addr = from_addr;
  bool aflag = true;

  while (msg_type - c_msg_type < 5 && msg_type_decode(msg_type_str, c_msg_type) || (aflag =
false) == true)
  {
    --c_msg_type;
    //aflag = false;
  }
  if (!aflag)
  {
    Serial.print("type ");
    Serial.print(msg_type_str);
    Serial.print(' ');
```

```
  Serial.print(msg_type - c_msg_type);
  Serial.println();
 }
 else
  Serial.println("msg_type_fail");


 aflag = true;
 while (to_addr - c_to_addr < 5 && addr_decode(to_addr_str, c_to_addr) || (aflag = false) == true)
 {
  --c_to_addr;
  //aflag = false;
 }
 if (!aflag)
 {
  Serial.print("to ");
  Serial.print(to_addr_str);
  Serial.print(' ');
  Serial.print(to_addr - c_to_addr);
  Serial.println();
 }
 else
  Serial.println("to_addr_fail");

 aflag = true;
 while (from_addr - c_from_addr < 15 && addr_decode(from_addr_str, c_from_addr) || (aflag =
false) == true)
 {
  --c_from_addr;
  //aflag = false;
 }
 if (!aflag)
 {
  Serial.print("from ");
  Serial.print(from_addr_str);
  Serial.print(' ');
  Serial.print(from_addr - c_from_addr);
  Serial.println();
 }
 else
  Serial.println("from_addr_fail");



 //if (to_addr == 0x0222)
 //{
  //Serial.write(buf, cnt);
  for (size_t i = 6; i < cnt; ++i)
```

```
      {
        char nice_out[10];
        //sprintf(nice_out, "%x ", buf[i]);
        Serial.print(buf[i], HEX);
        Serial.print(' ');
        //Serial.write(nice_out, 2);
      }

      Serial.println();
    //}

    /*uint8_t in_byte = Serial1.read();

    if (!in_byte and cnt++)
    {
      Serial.write(in_byte);
      Serial.println();
    }
    else if (in_byte)
    {
      Serial.write(in_byte);
      cnt = 0;
    }*/
    }
  }

  /*if (Serial.available())
  {
    uint8_t cnt = COBS_decode(buf);
    Serial.write(cnt);
    Serial.write(0xBB);
    Serial.write(buf, cnt);
  }*/
  //Serial.write(buf, 7);
  //COBS_encode(buf, 7);
  //delay(100);
}
```