Управление моторами тележки с контроллером Трик на JavaScript

О. М. Киселев, И. О. Киселев,
при участии консультанта от разработчиков ТРИК
Я.А. Кириленко

8 декабря 2017 г.

Содержание

1 Введение

Цель предлагаемого ниже текста – предоставить краткое введение для решения базовых задач учебной робототехники на платформе Трик. Текст подготовлен для читателей уже имеющих знания и умения по следующим разделам:

- энкодеры моторов;
- пропорциональный регулятор;
- ультразвуковой датчик расстояния;
- оптический датчик расстояния;
- гироскопический датчик;
- язык программирования JavaScript.

В тексте приводятся алгоритмы для прямолинейного движения и поворотов робота по показаниям энкодеров, по датчику гироскопа, алгоритм движения вдоль стены по датчикам расстояния. Эти алгоритмы основаны на использовании пропорционального регулятора. В программах коэффициент пропорциональности регулятора обозначается идентификатором k. Методика подбора значений коэффициента не обсуждается.

Замечание. В PDF-файл встроены короткие видео ролики. Для их просмотра нужна специальная настройка просмотрщика PDF документов. При подготовке текста для просмотра роликов в PDF использовалась свободно распространяемая программа Evince.

2 Равномерное, прямолинейное движение.

Тележка, собранная по дифференциальной схеме., как правило, движется непрямолинейно при одинаковой мощности, подаваемой на моторы. Причин несколько.

1. Моторы имеют различный стартовый крутящий момент.
2. Моторы выдают разное число оборотов при одной и той же подаваемой мощности.
3. Тележка нагружена несимметрично.
4. Шины колес под весом тележки деформируются по-разному из-за разной жесткости.

Существуют и другие причины.

Здесь рассмотрено несколько способов управления моторами для движения, близкого к прямолинейному.

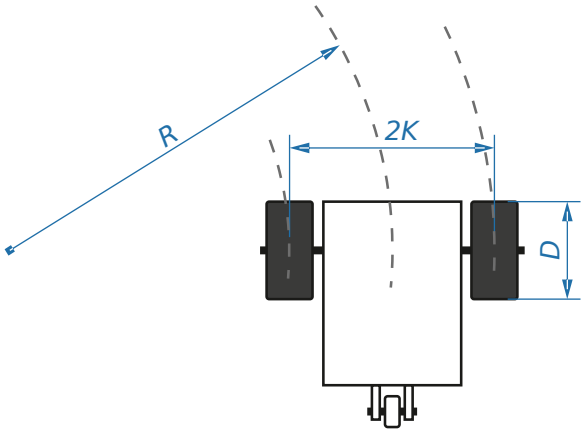

Рис. 1: Геометрические размеры робота и движение по дуге. (Рис. А. Колотова.)

2.1 Трудности

Основные трудности возникают из-за особенностей имеющейся в распоряжении тележки с контроллером Трик. Моторы имеют довольно большой стартовый крутящий момент. Вместе с входящими в набор колесами диаметром 85 мм это приводит к тому, что медленное равномерное движение затруднительно. При медленном движении пропорциональный регулятор близок к релейному:

1. Большое значение стартового момента моторов приводит к тому, что робот начинает движение с довольно высокой скоростью.

2. После старта, если сохранять мощность моторов равной стартовой, робот разгоняется, так как крутящий момент, необходимый для старта, выше, чем крутящий момент, необходимый для медленного движения.

3. При движении с малой скоростью снижение мощности при регулировании приводит к остановке одного из моторов. Это вызывает поворот робота в сторону остановившегося колеса.

4. Скорость движения тележки, запрограммированной на JavaScript с мощностью от 1 до 20 не меняется, что приводит к скачкам при плавной регулировке.

2.2 Тесты нерегулируемых моторов

Для проверки реакции моторов на выставленные в программе значения мощности были проведены тесты. Тележка запускалась на 2 секунды с различными значениями мощности. Значения энкодеров после пробега приведены в таблице.

Приведенные в таблице 1 данные показывают:

- моторы неодинаково реагируют на одинаковую мощность;
- изменение мощности от 1 до 20 не приводит к изменению скорости тележки.

мощность	левое колесо	правое колесо
1	543	463
1	535	486
1	539	474
10	501	472
10	546	541
10	628	428
20	465	450
20	533	464
20	529	495
30	682	805
30	745	692
30	791	671
40	980	872
40	957	810
40	933	960
50	1231	1164
50	1208	1070
50	1175	1093

мощность	левое колесо	правое колесо
60	1604	1328
60	1601	1328
60	1635	1326
70	1850	1511
70	1828	1566
70	1896	1552
80	2196	1978
80	2064	1964
80	2026	1842
90	2484	2234
90	2460	2102
90	2497	2235
100	2636	2653
100	2586	2601
100	2600	2580

Таблица 1: Показания энкодеров для нерегулируемых моторов при пробеге за 2 секунды. Напряжение питания 11,4В.

2.2.1 Количество импульсов энкодера за оборот

Для использования энкодеров необходимо определить изменение значения энкодера за один оборот. К сожалению, данные приведенные в издании Знакомство с средой программирования TRIK Studio, Широколобов И.Ю. не совпадают с имеющимся экземпляром.

Для определения числа импульсов энкодера на один оборот можно повернуть колесо на десяток оборотов и определить среднее значение. В результате получилось: изменение датчика энкодера за один оборот колеса $\sim 238,3$

2.3 Регулировка мощности мотора по показаниям энкодера ведущего мотора.

Алгоритм прямолинейного движения следующий. В качестве ведущего выбрать либо правый, либо левый мотор. Задать на нем желаемую мощность. Затем мощность второго мотора регулировать в зависимости от показаний энкодера ведущего мотора, например, с помощью пропорционального регулятора.

Пример реализации алгоритма "Движение робота при выравнивании по энкодеру левого мотора" доступен на видеоролике по ссылке.

Регулировка движения по энкодеру левого мотора.

Входные значения функции – мощность подаваемая на левый мотор и длина пробега в делениях энкодера. Мощность правого мотора подстраивается так, чтобы показания энкодеров моторов совпадали. Для этого используется простой пропорциональный регулятор.

```
function rightToLeftForward (power,length){
    left=brick.motor('M3');//левый мотор
    right=brick.motor('M4');//правый моторо
    el=brick.encoder('E3');//энкодер левого мотора
    er=brick.encoder('E4');//энкодер правого мотора
```

мощность	левое колесо	правое колесо
10	997	1009
10	995	1008
10	1003	1008
20	1000	1008
20	не стартовал	не стартовал
20	998	1003
30	1005	1005
30	1006	1010
30	997	1004
40	1010	1019
40	998	1006
40	1008	1014
50	1013	1021
50	1014	1019
50	1012	1017

Таблица 2: Показания энкодеров при регулировке правого колеса по энкодеру левого мотора. Длина пробега – 1000 делений энкодера, напряжение питания 11,4В. При низких значениях «мощности» генерируемый моторами крутящий момент низкий, иногда недостаточный для троганья с места.

```
    el.reset();//обнулить энкодер левого мотора
    er.reset();//обнулить энкодер правого мотора
    var l=0;//переменная для показаний левого энкодера
    var r=0;//переменная для показаний правого энкодера
    var k=3;//Коэффициент регулятора
    //задать требуемую мощность на левом моторе
    left.setPower(power);
    while(l<length){
        l=el.read();
        r=er.read();
        //сдвиг мощности на правом моторе
        // пропорционален разности показателей энкодеров
        right.setPower((l-r)*k+power);
        script.wait(10);
    }
}
```

Регулировка движения по энкодеру правого мотора может быть сделана так же. В функции нужно сделать ведущим правый мотор.

```
function leftToRightForward (power, length){
    left=brick.motor('M3');
    right=brick.motor('M4');
    el=brick.encoder('E3');
    er=brick.encoder('E4');
    el.reset();
    er.reset();
    var l=0;//переменная для показаний левого энкодера
```

```
    var r=0;//переменная для показаний правого энкодера
    var k=3;//Коэффициент регулятора
    right.setPower(power);
    while(r<length){
        l=el.read();
        r=er.read();
        left.setPower((r-l)*k+power);
        script.wait(10);
    }
}
```

Пример реализации алгоритма "Движение робота при выравнивании по энкодеру правого мотора" доступен на видеоролике по ссылке.

мощность	левое колесо	правое колесо
10	1002	1003
10	не стартовал	не стартовал
10	1005	1000
20	не стартовал	не стартовал
20	не стартовал	не стартовал
20	1003	1007
30	1002	1003
30	1008	1007
30	1005	1002
40	1012	1016
40	1004	1009
40	1010	1009
50	1007	1008
50	1004	1007
50	1001	1010

Таблица 3: Показания энкодеров при регулировке левого колеса по энкодеру правого мотора. Длина пробега – 1000 делений энкодера, напряжение питания 11,4В.

Эти способы регулировки моторов пригодны при работе моторов на мощностях не слишком малых и не слишком высоких.

При высоком значении мощности необходим запас мощности у ведомого мотора, чтоб при необходимости догнать ведущий мотор. То есть, ведомый мотор должен иметь возможность увеличить мощность и тем самым увеличить скорость. Наоборот, при низком значении мощности одни из моторов может застопориться.

2.4 Регулировка колеса к колесу

Комбинацией управления по ведущему колесу является регулировка движения «колесо к колесу». Такой режим управления моторами предполагает изменение мощности на обоих моторах в зависимости от разности значения энокдеров. При этом опережающий мотор получает меньшую мощность, отстающий – большую. Сдвигом мощности на моторах можно управлять, например, с помощью пропорционального регулятора.

Пример реализации алгоритма "Движение робота при выравнивании «колесо к колесу»" доступен на видеоролике по ссылке.

мощность	левое колесо	правое колесо
10	1002	1005
10	1008	1011
10	1001	1005
20	1006	1010
20	1003	1008
20	1007	1010
30	1007	1008
30	1006	1010
30	1009	1022
40	1014	1015
40	1008	1012
40	1010	1014
50	1011	1014
50	1017	1020
50	1013	1017

Таблица 4: Показания энкодеров при регулировке «колесо к колесу». Длина пробега – 1000 делений энкодера, напряжение питания 11,4В.

```
function wheelTowheelForward (power,length){
    left=brick.motor('M3');
    right=brick.motor('M4');
    el=brick.encoder('E3');
    er=brick.encoder('E4');
    el.reset();
    er.reset();
    var l=0;//переменная для показаний левого энкодера
    var r=0;//переменная для показаний правого энкодера
    var k=3;//Коэффициент регулятора
    while((r+l)/2<length){
        l=el.read();
        r=er.read();
        left.setPower((r-l)*k+power);
        right.setPower((l-r)*k+power);
        script.wait(10);
    }
}
```

Регулировка «колесо к колесу» позволяет адаптировать движение при большой мощности. Если один из моторов не может развить ту же скорость вращения, что и другой, то на быстрый мотор начинает подаваться меньшая мощность, и скорости вращения выравниваются.

2.5 Недостатки режимов управления по ведущему колесу и колесо-к-колесу

При низком значении мощности ведомый мотор должен иметь возможность так снизить мощность, чтоб при этом на нем все еще воспроизводился крутящий момент, необходимый для вращения встроенного редуктора мотора и колеса. Иначе мотор

мощность	скорость
1	не стартовал
5	0,06 м/с
10	0,21 м/с
15	0,33 м/с
20	0,43 м/с
25	0,5 м/с
30	0,5 м/с
35	0,6 м/с
40	0,6 м/с
50	0,75 м/с
100	1,5 м/с

Таблица 5: Средняя скорость движения тележки на расстоянии 3 м, напряжение питания 12,4В, регулировка «колесо к колесу»

при снижении мощности будет стопориться, и движение окажется неравномерным и дергающимся.

При регулировке по ведущему колесу не удаётся двигаться с произвольно малой скоростью. Причина – для троганья с места необходимо развить на моторах крутящий момент больше, чем стартовый момент. При достижении стартового момента робот начинает двигаться, но для равномерного движения на малой скорости необходим крутящий момент, меньший чем, стартовый. В результате, вместо очень медленного движения, робот при крутящем моменте на колесах, превышающем стартовый, будет разгоняться. Поэтому настроить движение робота с произвольно малой скоростью с помощью управления по ведущему колесу затруднительно.

2.6 Регулировка по виртуальному колесу

Регулировка по виртуальному колесу предполагает программную реализацию счетчика – «виртуального колеса», по которому синхронизируются энкодеры правого и левого колес.

Пример реализации алгоритма "Движение робота при выравнивании по «виртуальному колесу»" доступен на видеоролике по ссылке.

Счетчик – «виртуальное колесо» – можно настроить как на очень медленное увеличение – то есть медленное вращение, так и на очень быстрое– быстрое вращение. Поэтому синхронизация с виртуальным колесом позволяет устойчиво двигаться с малой скоростью, меньшей, чем это возможно при регулировке по ведущему колесу или регулировке колеса к колесу. При медленном вращении виртуального колеса подаваемая мощность на неподвижные моторы медленно нарастает, и при превышении порога начала движения робот начинает двигаться, при уменьшении мощности меньше порога робот останавливается. В результате, при малых скоростях движение оказывается с небольшими скачками, то есть неравномерным.

```
function wheelsToPoseForward(speed,length){
    left=brick.motor('M3');
    right=brick.motor('M4');
    el=brick.encoder('E3');
    er=brick.encoder('E4');
```

```
    el.reset();
    er.reset();
    var l=0;//переменная для показаний левого энкодера
    var r=0;//переменная для показаний правого энкодера
    var expected=0;//энкодер виртуального колеса
    var k=2;//Коэффициент регулятора
    while(expected<length){
        expected+=speed;//энкодер увеличивается
                        //на величину скорости
        l=el.read();
        r=er.read();
        //синхронизация с виртуальным колесом левого мотора
        left.setPower((expected-l)*k);
        //синхронизация с виртуальным колесом правого
        right.setPower((expected-r)*k); //мотора
        script.wait(10);
    }
}
```

делений энкодера/10мс.	левое колесо	правое колесо
1	1000	1000
1	996	1006
1	1000	1001
5	1002	1008
5	998	1024
5	1033	1003
10	1020	1014
10	998	1024
10	1033	1003

Таблица 6: Показания энкодеров при регулировке по «виртуальному колесу». Длина пробега – 1000, напряжение питания 11,4.

Основной недостаток регулировки по виртуальному колесу сказывается при движении с большой скоростью. Реальные колеса могут не догонять виртуальное, и произойдет рассогласование управления.

«скорость виртуального колеса»	скорость тележки
1	0,005 м/с
2	0,09 м/с
4	0,19 м/с
8	0,37 м/с

Таблица 7: Средняя скорость движения тележки на расстоянии 3 м, округленная до сотых. Напряжение питания 12,4В. Регулировка по «виртуальному колесу»

2.7 Замечания разработчика

Разработчик Яков Кириленко отметил,

- **Для ускорения обработки данных лучше не обращаться несколько раз подряд к энкодеру, а записывать показания в переменную и работать с полученным значением.** Энкодер мотора при повороте оси мотора генерирует импульс. Этот импульс в свою чередь генерирует прерывание на контроллере MSP430. Этот контроллер по шине i2c с частотой синхронизации 100кГц подключен к центральному процессору. Поэтому данные, которые считывает программа на JS, получаются не прямо с энкодера. Сначала процессор по шине i2c делает запрос к контроллеру, затем полученные по шине данные заносятся в кеш-память процессора. Только после этого поток JS получает доступ к данным.

- **В конце цикла управления рекомендуется сделать паузу script.wait(Длительность**
  Длительность паузы можно установить опытным путем. В разных циклах управления она может быть различной. Пауза особенно важна, если в Вашей программе используется несколько параллельно работающих ветвей, чтобы процессор имел возможность переключать работу ветвей JS, не обращаясь к вытесняющей многозадачности операционной системы.

- В конфигурационном файле JS есть некоторая модель управления моторами. Эту модель управления можно отключать, например, для мотора 'M3', используя параметр «false» в команде

  ```
  brick.motor('M3').setPower(power, false);
  ```

По результатам обсуждения с разработчиком длительности пауз в управлении моторами выяснено, что чем меньше такие паузы, тем ровнее движется робот. Паузы в 10-20 мс. вполне рабочие. Однако паузы длительностью 70-100 мс приводят к полной разбалансировке движения.

Пример реализации алгоритма "Зависимость движение робота от частоты управляющего воздействия" доступен на видеоролике по ссылке.

2.8 Заключение

- Для прямолинейного движения со скоростью менее чем 0.5 м/с наиболее удобным является алгоритм движения по виртуальному колесу.

- Если требуемая скорость движения более чем 0,5 м/с и менее, чем 1,5 м/с, можно пользоваться любым из рассмотренных алгоритмов движения по прямой линии.

3 Движение с использованием гироскопа

3.1 Спецификация датчика гироскопа

Датчик гироскопа установлен непосредственно внутри блока Трик. Основы работы с датчиком изложены на страница 30-34 документа Знакомство с средой программирования TRIK Studio, Широколобов И.Ю.

Со слов разработчика известно, что JS поток останавливается, пока не получит ответ на запрос датчика. Определить время такой остановки потока можно с помощью следующей функции:

```
function delay(){
    var dr=[0,0,0];//массив трех элементов
    var t=0;//переменная для вычисления
            //среднего времени обращения к датчику
    var t0=0;//предыдущее значение времени
    for (n=0; n<100; n++){
        t0=Date.now();
        dr=brick.gyroscope().read();
        t=t+(Date.now()-t0);
    }
    return t/100;
}
```

По результатам многократного запуска этой программы выяснено, что время выполнения команды

```
dr=brick.gyroscope().read();
```

примерно –5-7мс.

При запросе данных с датчика гирскопа

```
brick.gyroscope().read();
```

выдается массив из семи элементов. Обычно ожидается, что часть из этого массива – это значения скоростей поворота робота относительно трех пространственных осей. Однако это не совсем так. Известно, что датчики гироскопов имеют дрейф. То есть, даже в неподвижном состоянии датчик выдает ненулевые значения по всем трем осям. Поэтому для получения значения скоростей поворота необходимо знать скорость дрейфа гироскопа.

Разработчики Трик подготовили процедуру калибровки датчика и интегрирование массива мгновенных угловых скоростей

```
brick.gyroscope().read()[0],
brick.gyroscope().read()[1],
brick.gyroscope().read()[2]
```

в массив углов поворота:

```
brick.gyroscope().read()[4],
brick.gyroscope().read()[5],
brick.gyroscope().read()[6].
```

3.2 Алгоритм движения по прямой с помощью гироскопа

Для движения по прямой с помощью гироскопа необходимо подправлять скорости колес робота так, чтоб угол поворота по гироскопу был неизменным. Для прямолинейного движения опять удобно воспользоваться пропорциональным регулятором.

Алгоритм движения прост:

- если изменение угла <0, нужно подкручивать правое колесо робота;
- если изменение угла >0, нужно подкручивать левое колесо робота;

Пример реализации алгоритма "Прямолинейное движение робота по датчику гироскопа" доступен на видеоролике по ссылке.

Ниже приведена программа на JS:

```
var __interpretation_started_timestamp__;
var pi = 3.1415926535897931;
var angles=[0,0,0];//глобальный массив текущих углов поворота
var drifts=[0,0,0];//глобальный массив скоростей дрейфа гироскопа

var main = function()
{
    __interpretation_started_timestamp__ = Date.now();
    brick.gyroscope().calibrate(10000);
    script.wait(11000);
    var init_a=brick.gyroscope().read()[6]/1000;
    forward_gyro(1000,-25);
    print(init_a,"  ",brick.gyroscope().read()[6]/1000,"  ",
        brick.encoder(E3).read(),"   ",brick.encoder(E4).read());
    return 0;
}

function forward_gyro(length,power){
    var k=3;//коэффициент пропорционального регулятора
    //Начальное направление робота
    var z0=brick.gyroscope().read()[6]/1000;
    brick.encoder(E3).reset();
    brick.encoder(E4).reset();
    while(!((brick.encoder(E3).read()+brick.encoder(E4).read())/2 <-length)){
        z=z0-brick.gyroscope().read()[6]/1000;
        if(z>10) z=10;
        if(z<-10) z=-10;
        pl=power+k*z;
        pr=power-k*z;
        brick.motor(M4).setPower(pr);
        brick.motor(M3).setPower(pl);
        script.wait(10);
    }
}
```

начальный угол	конечный угол	левый энкодер	правый энкодер
-0.008	2.71	-996	-1010
0.021	4.155	-999	-1014
-0.011	-2.048	-1008	-1003

Таблица 8: Результаты трех заездов робота на расстояние 1000 делений полусуммы энкодеров колес. Средняя мощность, подаваемая на моторы $p = -15$. Управление – пропорциональный регулятор по датчику гироскопа. Напряжение питания 11,8В.

В приведенном примере скорости вращения колес при движении регулируются пропорциональным регулятором. Довольно трудно добиться устойчивого движения робота, регулируя напрямую мощность моторов. Приведем две основных причины.

- При изменении значений мощности от 0 до 20 Трик почти не меняет скорость вращения моторов.
- Каждый из моторов имеет свою чувствительность к изменению мощности.

Поэтому наиболее надежным методом управления моторами для медленного оказывается управление с помощью виртуального колеса. Здесь будет использовано два виртуальных энкодера для каждого из ведущих колес.

Пример реализации алгоритма "Прямолинейное движение с использованием датчика гироскопа регуляции скорости виртуального колеса" доступен на видеоролике по ссылке.

```
var __interpretation_started_timestamp__;
var pi = 3.1415926535897931;
var main = function(){
    __interpretation_started_timestamp__ = Date.now();
    brick.gyroscope().calibrate(10000);
    script.wait(11000);
    var init_a=brick.gyroscope().read()[6]/1000;
    forward_virt_gyro(1000,1);
    print(init_a,"  ",brick.gyroscope().read()[6]/1000,"  " , brick.encoder(E3).read()
    return 0;
}
//прямолинейное движение по виртуальным колесам
//входные параметры:
//length --средний пробег виртуальных колес
//v --средняя скорость виртуальных колес
function forward_virt_gyro(length,v){
    var k=10;//коэффициент регулятора виртуальных колес
    var k1=10;//коэффициент мощности
    //Начальное направление робота
    var z0=brick.gyroscope().read()[6]/1000;
    var p0=0;//энкодер левого виртуального колеса
    var p1=0;//энкодер правого виртуального колеса
    brick.encoder(E3).reset();
    brick.encoder(E4).reset();
    e3=brick.encoder(E3).read();
    e4=brick.encoder(E4).read();
    while(!((e3+e4)/2 <-length)){
        z=z0-brick.gyroscope().read()[6]/1000;
        if(z>10) z=10;
        if(z<-10) z=-10;
        p0+=-v-z/k;//энкодер левого виртуального колеса
        p1+=-v+z/k;//энкодер правого виртуального колеса
        e3=brick.encoder(E3).read();
        e4=brick.encoder(E4).read();
        brick.motor(M4).setPower((p0-e4)/k1,false);
        brick.motor(M3).setPower((p1-e3)/k1,false);
        script.wait(10);
    }
}
```

начальный угол	конечный угол	левый энкодер	правый энкодер
-0.017	0.907	-1015	-989
0.021	-0.234	-994	-1011
-0.036	-1.074	-988	-1015

Таблица 9: Результаты трех заездов робота на расстояние 1000 делений полусуммы энкодеров колес. Средняя скорость виртуального колеса $v = 1$. Управление – пропорциональный регулятор по датчику гироскопа с виртуальными колесами. Напряжение питания 11,7 В.

4 Алгоритмы поворота

Для простоты здесь и ниже рассмотрены повороты на величину прямого угла. Наиболее распространены два вида поворота.

- Поворот вокруг центра оси ведущих колес. Для такого поворота почти не требуется дополнительного пространства вокруг робота. Если центр оси ведущих колес совпадает с геометрическим центром робота, тогда поворот делается практически на месте, без дополнительного перемещения центра робота.
- Поворот вокруг неподвижного колеса. Такой поворот требует достаточно много пространства вокруг робота. Это поворот по дуге окружности с центром в точке опоры неподвижного колеса. Радиус окружности поворота, которую описывает центр робота, равен половине колеи робота.

4.1 Поворот по энкодерам

Для управления поворотом можно использовать показания энкодеров моторов.

4.1.1 Геометрическая размерность робота.

Примем обозначения

- Диаметр колес робота $D = 85$мм. На самом деле колеса робота с шипами, поверхность шины слегка проминается под весом робота. Поэтому приведенная величина диаметра должна рассматриваться как приблизительная.
- Колея робота $K = 185$мм. Колея робота измерена от середины левой шины, до середины правой. Шины робота существенно проминаются под его весом, центр пятна контакта колеса с опорной поверхностью может оказываться не обязательно посередине шины. Это означает, что измеренное геометрическое значение колеи робота может отличаться от реального,значения, возникающего при движении.

4.1.2 Поворот вокруг центра оси ведущих колес.

При повороте вокруг центра ведущих колес левое и правое колеса робота вращаются в разные стороны с одинаковыми скоростями. При этом каждое из колес пробегает четверть окружности радиуса $K/2$. Длина дуги четверти такой окружности:

$$L = \frac{\pi K}{4}.$$

При этом каждое из колес должно повернуться на угол, абсолютная величина которого:

$$\alpha = 2\pi\frac{L}{\pi D} = 2\pi\frac{K}{4D} = \pi\frac{185}{2\times 85} \sim 3,42(\text{рад.})$$

или

$$\alpha = \frac{3,42}{2\pi}360 \sim 196^\circ.$$

За один оборот энкодер мотора дает 238 импульсов (см. 6.3). Тогда изменение показаний энкодера колеса при повороте:

$$n = \frac{196^\circ}{360^\circ}238 \sim 130.$$

Для определенности примем, что это поворот направо. Тогда изменение показаний энкодера левого колеса: -130, правого колеса: 130

Пример реализации алгоритма "Поворот по энкодерам вокруг центра оси ведущих колес" доступен на видеоролике по ссылке.

Ниже приведен код функции для поворота робота

```
var main = function(){
    turn(1,130);
    return;
}
//speed- скорость поворота
//deg -- изменение показаний энкодера
function turn(speed,deg){
    left=brick.motor('M3');
    right=brick.motor('M4');
    el=brick.encoder('E3');
    er=brick.encoder('E4');
    el.reset();//обнулить левый энкодер
    er.reset();//обнулить правый энкодер
    var expected=0;
    var ld=el.read();
    var rd=er.read();
    while((rd-ld)/2<deg){
        ld=el.read();
        rd=er.read();
        expected+=speed;//изменение виртуально энкодера
        left.setPower((-expected-ld)*3);
        right.setPower((expected-rd)*3);
        script.wait(10);
    }
}
```

4.1.3 Поворот вокруг неподвижного колеса

При повороте вокруг неподвижного колеса вращающееся колесо робота пробегает четверть окружности, радиус которой равен колее робота. Длина дуги четверти такой окружности:

$$L = \frac{2\pi K}{4} = \frac{\pi K}{2}.$$

Угол поворота колеса при этом:

$$\alpha = \frac{L}{\pi D} 2\pi = \frac{\pi K}{D} = \frac{185\pi}{85} \sim 6,83$$

или

$$\alpha = \frac{\pi K}{D} \frac{360}{2\pi} = 180 \frac{K}{D} \sim 391^\circ.$$

Изменение показаний энкодера колеса при повороте:

$$n = \frac{391^\circ}{360^\circ} 238 \sim 259.$$

Пример реализации алгоритма "Поворот по энкодерам вокруг неподвижного колеса" доступен на видеоролике по ссылке.

Ниже приведен код функции для поворота робота налево

```
//speed- скорость поворота
//deg -- изменение показаний энкодера
function turnRightWheel(speed,deg){
    right=brick.motor('M4');
    er=brick.encoder('E4');
    er.reset();
    var expected=0;
    var rd=er.read();
    while(rd<deg){
        rd=er.read();
        expected+=speed;
        right.setPower((expected-rd)*3);
        script.wait(10);
    }
}
```

4.2 Алгоритм поворота с помощью гироскопа

Для управления поворотом с помощью датчика гироскопа нужно перед поворотом откалибровать датчик. Затем делать либо поворот на месте, либо поворот вокруг неподвижного колеса до тех пор, пока датчик поворота не покажет угол в 90°.

Пример реализации алгоритма "Поворот по гироскопу вокруг центра оси ведущих колес" доступен на видеоролике по ссылке.

Ниже приведен код функции для поворота робота налево вокруг оси робота

```
var __interpretation_started_timestamp__;
var pi = 3.1415926535897931;
var main = function(){
    __interpretation_started_timestamp__ = Date.now();
    brick.gyroscope().calibrate(10000);
    script.wait(11000);
    var init_a=brick.gyroscope().read()[6]/1000;
    turn(1,90);
    print(init_a,"  ",brick.gyroscope().read()[6]/1000,"  "
        ,brick.encoder(E3).read(),"   ",brick.encoder(E4).read());
    return 0;
```

```
}
function turn(speed,deg){
    left=brick.motor('M3');
    right=brick.motor('M4');
    el=brick.encoder('E3');
    er=brick.encoder('E4');
    el.reset();
    er.reset();
    var expected=0;
    var ld=el.read();
    var rd=er.read();
    var a=brick.gyroscope().read()[6]/1000;
    while((a-brick.gyroscope().read()[6]/1000)<deg){
        ld=el.read();
        rd=er.read();
        expected+=speed;
        left.setPower((-expected-ld)*3);
        right.setPower((expected-rd)*3);
        script.wait(10);
    }
}
```

Пример реализации алгоритма "Поворот по гироскопу вокруг неподвижного колеса" доступен на видеоролике по ссылке.

Ниже приведен код функции для поворота робота налево вокруг неподвижного левого колеса.

```
function turnRightWheel(speed,deg){
    right=brick.motor('M4');
    er=brick.encoder('E4');
    er.reset();
    var expected=0;
    var rd=er.read();
    var a=brick.gyroscope().read()[6]/1000;
    while((a-brick.gyroscope().read()[6]/1000)<deg){
        rd=er.read();
        expected+=speed;
        right.setPower((expected-rd)*3);
        script.wait(10);
    }
}
```

5 Движение вдоль стены

Для движения вдоль стены можно использовать два типа датчиков из набора Трик – ультразвуковой датчик расстояния и инфракрасный датчик расстояния.

5.1 Спецификация ультразвукового датчика

В наборе Трик используется датчик расстояния HC SR04. Этот датчик широко распространен и его характеристики легкодоступны. Ниже приведены основные харак-

теристики датчика, взятые из документации к датчику от Cytron Technologies:

- **Рабочая частота** – 40 кГц.
- **Рабочий диапазон расстояний** – 2см-400см.
- **Точность измерений** – 0.3см.
- **Рабочий диапазон углов** ±15 градусов.

Датчик излучает 8 импульсов частотой 40 кГЦ, то есть время излучения этих импульсов – 2×10^{-4} сек. Для измерения расстояния около 4 м импульс должен в сумме преодолеть 8 метров. Скорость звука примерно 330 м/с, следовательно эхо от импульса вернется примерно через 25 мс. Общее время, необходимое для формирования импульса, ожидания эхо и приема импульса оценивается в 30 мс.

5.2 Замечания разработчика. Обработка Триком данных с датчика.

Разработчик Яков Кириленко сообщил:

- Трик на обработку сигнала УЗ датчика тратит 60 мс. Данные с датчика первоначально считываются микросхемой MSP 430. Эта микросхема подключена к основному процессору по i2c шине. Частота работы шины 100кГц. Сама микросхема опрашивается центральным процессором раз в 100мс. Данные опроса – расстояние до препятствия – заносятся в системную память процессора. Именно к этой ячейке памяти и обращается программа на JS при запросе данных с УЗ датчика.
- У разработчика нет уверенности, что по запросу поток JS получит результаты последнего измерения. вполне возможно, что будет получен результат предыдущих измерений из кеша данных.
- Обращение к шине i2C может потребовать **неожиданно и непредсказуемо** 10-20 мс дополнительно. При обращении к датчику поток JS блокируется до момента готовности данных.

Данные с ультразвукового датчика расстояния оказываются доступны программе примерно через 100 мс. после реального измерения.

5.3 Подключение датчика ультразвукового датчика

Рабочее напряжение датчика 5 В. При подключении датчика к Трику важно, чтоб жила питания, отмеченная красным, подключалась к левому контакту, если повернуть датчик излучателем и приемником звука к себе. При неправильном подключении датчик может выйти из строя.

5.4 Результаты измерений

Ультразвуковой датчик позволяет измерять расстояния до объектов, находящихся почти прямо перед датчиком – рабочий растр ±15 градусов. Расстояние до объекта, находящегося под нулевым углом к датчику l, тогда расстояние до объекта, находящегося под углом 15 градусов: $l/\cos(15) \sim l/0.96 \sim 1.035l$.

Это означает, что если объект находится на расстоянии $l = 10$, то различие между расстоянием до объекта, замеренным для объекта по углом 0 градусов и под углом 15 градусов $\sim$ 0.3см, то есть в пределах погрешности измерений. Поэтому при измерении расстояний до 30 см, различием между измерениями под разными углами можно пренебречь.

Данные измерений УЗ датчика часто сильно зашумлены – содержат ошибки. Поэтому рекомендуется при обработке измерений пользоваться фильтрами. Простейший способ фильтрации – использовать доверенный диапазон, то есть не реагировать на неожиданно малые и неожиданно большие данные измерений.

5.5 Спецификация инфракрасного датчика

В наборе Трик поставляется ИК датчик расстояния SHARP 2Y0A21. Документацию к этому датчику можно найти по ссылке.

В датчике в качестве источника ИК излучения используется светодиод. В качестве приемника – линейная последовательность CCD сенсоров, таких же, как в матрицах фотоаппаратов. Источник излучает луч через линзу, луч отражается от препятствия, попадает в собирающую линзу приемника излучения и фокусируется на сенсор.

Исходящий луч, отраженный луч и отрезок от излучателя до сенсора, принимающего отраженный луч, образуют почти равнобедренный треугольник. Зная номер принимающего сенсора в линейке CCD, можно определить угол у основания треугольника. Высота треугольника равна расстоянию до точки поверхности, отражающей луч.

Рабочие характеристики датчика:

- минимальное расстояние 10 см;
- максимальное расстояние 80 см;
- время одного измерения 42 ± 10 мс;
- длина волны излучения 870 нм.

5.6 Установка на робота

Есть два способа установки датчиков расстояния на робота.

- **Стандартный.** Датчик устанавливается вперед по отношению к оси колес. Датчик направлен перпендикулярно направлению движения робота.
- **Правильный.** Датчик установлен под углом 80-70 градусов (ультразвуковой датчик) или 60-45 градусов (инфракрасный датчик) по отношению к направлению движения.
  - Точка установки ИК датчика на роботе такова, чтобы точка отражения луча всегда оказывалась впереди оси колес.
  - Точка установки УЗ датчика должна быть вынесена вперед по отношению к оси ведущих колес робота.

При установке датчика и программировании робота нужно учитывать различия, вызванные разным характером распространения звука и ИК излучения. Звук в воздухе распространяется как волна сжатия с плохо выраженной направленностью – растр 30 градусов. ИК излучение, формируемое датчиком, распространяется как луч со сравнительно небольшим расхождением.

- УЗ датчик определяет кратчайшее расстояние до стены в рабочей зоне растра.
- ИК датчик показывает расстояние до точки отражения.

Опасны крутые повороты робота – например, для УЗ датчика – на угол более 15 градусов – к стене или от стены. При этом УЗ датчик может потерять контакт со стеной, и, как правило, выдает большие значения расстояния. ИК датчик будет давать измерения расстояния до точки отражения луча, которое, если эта точка далеко впереди или позади робота, оказывается существенно больше, чем расстояние от робота до стены. В результате робот, управляемый пропорциональным регулятором, либо упирается в стену, либо начинает ездить по кругу или вращаться вокруг собственной оси.

5.7 Алгоритм медленного движения вдоль стены

В приведенном примере скорости вращения колес при движении вдоль стены регулируются пропорциональным регулятором. Довольно трудно добиться устойчивого движения робота вдоль стены на заданном расстоянии, регулируя напрямую мощность моторов. Приведем две основных причины.

- При изменении значений мощности от 0 до 20 Трик почти не меняет скорость вращения моторов.
- Каждый из моторов имеет свою чувствительность к изменению мощности.

Поэтому наиболее надежным методом управления моторами для движения на заданном расстоянии вдоль стены оказывается управление с помощью виртуальных колес. Если для прямолинейного движения использовалось одно виртуальное колесо, то здесь их будет два – левое и правое.

Пример реализации алгоритма "Медленное движение вдоль стены с неравномерным движением из-за большого стартового момента моторов" доступен на видеоролике по ссылке.

```
var main = function()
{
    var p0=0; //энкодер левого виртуального колеса
    var p1=0; //энкодер правого виртуального колеса
    var p=0; //сдвиг виртуального колеса
    var e3=0; // переменная для хранения показания правого энкодера
    var e4=0; // переменная для хранения показания левого энкодера
    var d=0; //переменная для текущего расстояния до стены
    var d0=0;//переменная для хранения ответа датчика,
              //используется в фильтре
    var L=20; //требуемое расстояние до стены
    var k=10;//Коэффициент (обратная величина) регулятора скорости
    var k1=2;//Коэффициент регулятора мощности
    brick.encoder(E3).reset();
```

```
    brick.encoder(E4).reset();
    // здесь движение в обратную сторону.
    //счетчики энкодеров уменьшаются со временем
    while(e3+e4>-2000){
        e3=brick.encoder(E3).read();
        e4=brick.encoder(E4).read();
        d0=brick.sensor(D1).read();//Сенсор УЗ датчика
        //d0=brick.sensor(A1).read();//сенсор ИК датчика
        //простейший фильтр диапазона значений датчика расстояния
        if(2<d0 &&d0<100){
            d=d0;
        }
        p=(d-L)/k;
        //здесь энкодеры виртуальных колес
        //уменьшаются на -5 (<<скорость>>)
        //и сдвиг, зависящий от текущего расстояния до стены
        p0+=-5-p;//энкодер левого виртуального колеса
        p1+=-5+p;//енкодер правого виртуального колеса
        // мощность на моторах устанавливается без влияния
        //заданной в конфигурационном файле
        brick.motor(M4).setPower((p0-e4)*k1,false);
        brick.motor(M3).setPower((p1-e3)*k1,false);
        script.wait(10);
    }
}
```

5.8 Заключение

Для движения вдоль стены по датчику расстояния рекомендуется:

- двигаться на малой скорости,так как датчики расстояния «медленные»;
- использовать алгоритм управления виртуальными колесами;
- фильтровать полученные показания датчика УЗ от шума;
- контролировать угол между направлением движения робота и стеной. Особенно это важно для УЗ датчика, так как отклонения более 15 градусов приводят к потере контакта датчика со стеной.

6 Технические сведения.

6.1 Среднее время выполнения команды чтения данных из программы на JS

- Обращение к энкодеру $\sim 4 - 7$мс.
- Время обращения к датчику гироскопа $\sim 5 - 7$ мс.
- Время обращения к ультразвуковому датчику датчику расстояния $\sim 2 - 3$мс.
- Время обращения к инфракрасному датчику расстояния $\sim 4 - 7$мс.

- Время прохождения команды установки мощности на моторе $\sim 5 - 7$мс.

Методика тестирования – суммарное время ста обращений к датчику в цикле. Программа для оценки времени обращения к датчику гироскопа приведена в разделе о движении по гироскопу 3.1. Тесты для остальных датчиков проводились по той же методике.

6.2 Среднее время обновления данных при обращении из программы на JS

- Обновление данных энкодера $\sim 4 - 7$мс. В среднем происходит за 1 обращение к датчику.
- Обновление данных гироскопа ~ 10 мс. В среднем за 2-3 обращения к датчику.
- Обновление данных ультразвукового датчика расстояния $\sim 80 - 100$мс. В среднем за 10-20 обращений к датчику.
- Обновление данных инфракрасного датчика расстояния $\sim$ 10мс. За 2 обращения к датчику. **Этот результат противоречит техническим характеристикам датчика SHARP 2Y0A21, где указано, что среднее время измерения $\sim$ 42мс. Видимо программа JS обращается к кешу данных, однако и в этом случае результат требует дополнительных разъяснений со стороны разработчиков.**

Методика тестирования – суммарное время ста изменений данных при последовательных запросах датчика в цикле. Программа для оценки времени обращения к УЗ датчику приведена ниже.Тесты для остальных датчиков проводились по той же методике.

```
function timeCountUntilChanges(){
    var sum=0;
    var t1=0;
    var t2=0;
    brick.motor("M3").setPower(100);
    brick.motor("M4").setPower(-100);
    script.wait(500);
    var n=brick.sensor("D1").read();
    var o=n;
    var x=0;
    while(true){
        sum=0;
        x=0;
        for(var i=0;i<100;i++){
            n=brick.sensor("D1").read();
            o=n;
            t1=Date.now();
            while(n==o){
                x++;
                n=brick.sensor("D1").read();
            }
            t2=Date.now();
```

```
            sum+=t2-t1;
            script.wait(10);
        }
        script.wait(1000);
        print(sum/100+" "+x/100+" "+sum/x);
    }
    return t2-t1;
}
```

6.3 Показания датчиков на Трике

- Изменение датчика энкодера за оборот 238 импульсов. Среднее значение за 10 оборотов. По показаниям из меню Настройки->Тестирование->Энкодеры.
- Показания датчика гироскопа скорость поворота – милиградус в сек, угол поворота – милиградус. от ~ -180000 до ~ 180000. По данным тестовой программы измерялось только изменение угла рысканья – угла поворота относительно вертикальной оси робота.
- Диапазон показаний УЗ датчика расстояния $2-403$см. По показаниям из меню Настройки->Тестирование->Цифровые датчики.
- Диапазон показания ИК датчика расстояния $7-150$см. По показаниям из меню Настройки->Тестирование->Аналоговые датчики.
- Диапазон выставляемых значений мощности моторов ± 128 единиц.