Сергей Александрович Филиппов

ОСНОВЫ РОБОТОТЕХНИКИ НА БАЗЕ КОНСТРУКТОРА LEGO MINDSTORMS NXT. ЗАНЯТИЕ 5. ДВИЖЕНИЕ ПО ЛИНИИ

Одна из классических задач для мобильного робота – это движение по черной линии на белом поле с использованием датчиков освещенности. Существует множество алгоритмов, рассчитанных на различное число датчиков: от 1 до 10 и более. Сегодня мы познакомимся с простейшими из них. При этом придется затронуть элементы теории автоматического управления, которая лежит в основе программирования поведения многих роботов и автоматизированных систем.

РОБОТ С ОДНИМ ДАТЧИКОМ

В первом опыте используем робот, созданный для задания «Танец в круге», и то же поле – черную окружность на белом фоне. Единственная поправка: датчик освещенности следует выдвинуть немного вперед, чтобы он образовывал вместе с ведущими колесами равносторонний или хотя бы равнобедренный прямоугольный треугольник (рис. 1).

Конструкцию можно построить множеством способов. Рассмотрим один из них. Крепление датчика освещенности к трехколесной тележке (рис. 2). Короткая двухмодульная ось может быть красного или черного цвета (рис. 3). Высота датчика над поверхностью поля – от 5 до 10 мм (рис. 4).

РЕЛЕЙНЫЙ РЕГУЛЯТОР

Задача такова: двигаться по плоскому полю вдоль границы черного и белого (рис. 5). Решается элементарно применением релейного двухпозиционного регулятора[1]. В таком регуляторе рассматри-

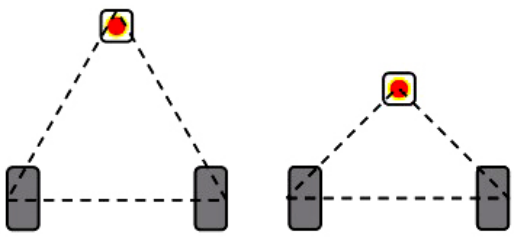


Рис. 1. Расположение датчика освещенности

[1] В нашем случае под регулятором понимается устройство, которое определяет отклонение объекта от заданного состояния и вырабатывает управляющее воздействие на моторы.

Рис. 2. Крепление датчика освещенности к трехколесной тележке

Рис. 3. Короткая двухмодульная ось может быть красного или черного цвета

ваются только два состояния датчика и, соответственно, два вида управляющего воздействия на моторы. Пока датчик на белом, робот двигается в сторону черного, пока датчик на черном, робот двигается в сторону белого. Благодаря тому, что поворот осуществляется по дуге с небольшим радиусом, в итоге происходит поступательное движение вперед.

Алгоритм будет записан с использованием блоков «Жди темнее» и «Жди светлее». Вот простейшее решение для начинающих (рис. 6).

В языке RobotC нет точного соответствия этим двум командам Robolab. Не усложняя программу, можно записать примерно следующим образом.

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = 100;
  motor[motorC] = 0;
  while(SensorValue[S1]>white-5);
  motor[motorB] = 0;
  motor[motorC] = 100;
  while(SensorValue[S1]<white-5);
}
}
```

Без дополнительных уточнений предполагается, что датчик освещенности под-

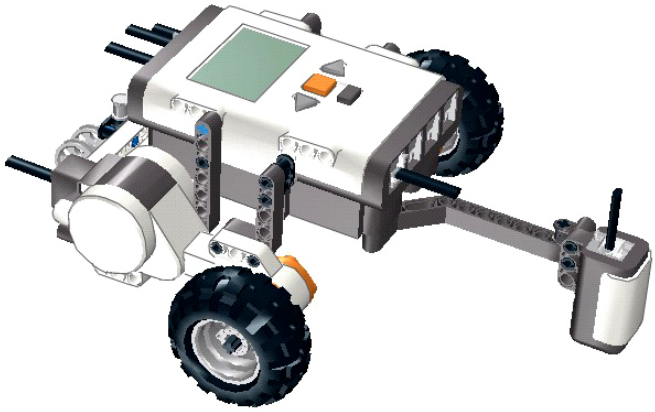


Рис. 4. Высота датчика над поверхностью поля – от 5 до 10 мм

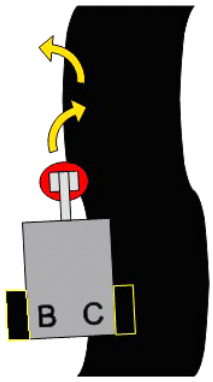


Рис. 5. Движение вдоль границы черного и белого

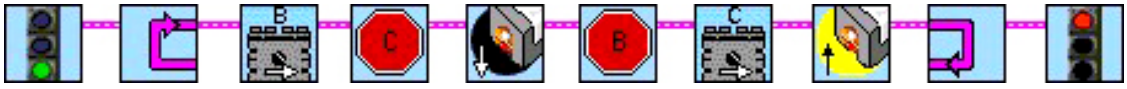


Рис. 6. Алгоритм движения по линии с одним датчиком освещенности

ключен к первому порту, а на моторы подается максимальная мощность (рис. 6). Перед стартом ставим робот на линию так, чтобы датчик был над белым полем на расстоянии 2–3 см слева от черного. По алгоритму робот плавно поворачивает направо, пока освещенность не понизится на 5 пунктов (по умолчанию). Затем поворачивает налево, пока освещенность не повысится на 5 пунктов. Движение получается похожим на «змейку» (рис. 5).

ВОЗМОЖНЫЕ ПРОБЛЕМЫ.

1. Робот крутится на месте, не заезжая на линию. В этом случае следует либо стартовать с другой стороны линии, либо поменять подключения моторов к контроллеру местами.

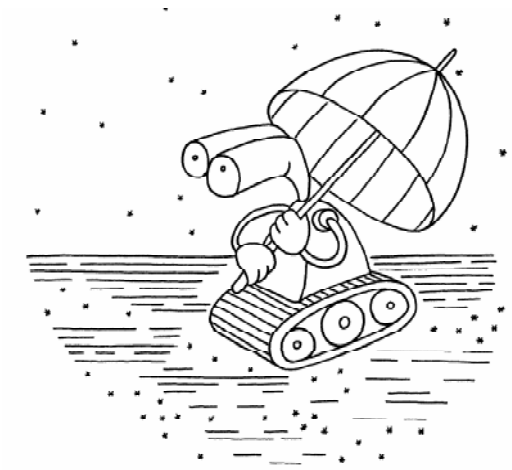
2. Робот проскакивает линию, не успевая среагировать. Следует понизить мощность моторов.

3. Робот реагирует на мелкие помехи на белом, не доезжая до черного. Надо увеличить порог чувствительности датчика (например, не на 5, а на 8 пунктов). Вообще говоря, это число можно рассчитать: снять показания датчика на белом, затем на черном, вычесть одно из другого и поделить пополам. Например, (53 – 37)/2 = 8. На контроллере «NXT» снятие показаний производится через пункт **View** главного меню.

Так выглядит усовершенствованная программа (рис. 7):

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = 50;
  motor[motorC] = 0;
  while(SensorValue[S1]>white-8);
  motor[motorB] = 0;
  motor[motorC] = 50;
  while(SensorValue[S1]<white-8);
}
}
```

Особенность приведенных алгоритмов в том, что во всех случаях роботу требуется стартовать на белом поле и исполь-



Робот реагирует на мелкие помехи на белом, не доезжая до черного.

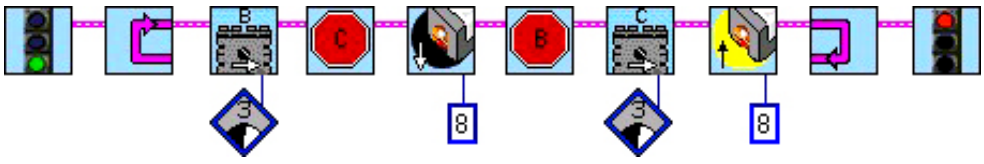


Рис. 7. Алгоритм движения по линии с одним датчиком освещенности: понижена скорость, увеличена разность между черным и белым

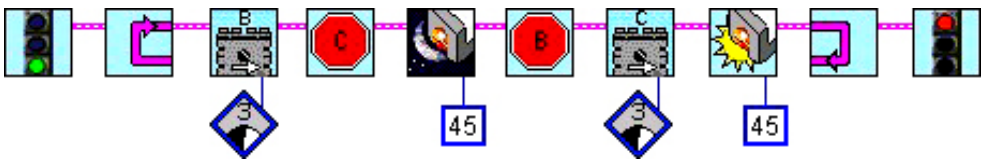


Рис. 8. Алгоритм движения по линии с абсолютным значением серого на датчике освещенности

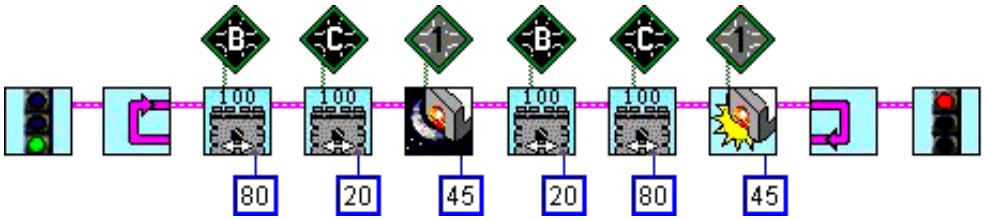


Рис. 9. Алгоритм движения по линии с одним датчиком освещенности: улучшено управление моторами

зуется относительное понижение освещенности. Однако есть возможность заранее определить уровень освещенности на данном поле и использовать его абсолютное значение. Воспользовавшись показаниями датчика на белом и черном, полученными через меню **View**, рассчитаем их среднее арифметическое (53 + 37)/2 = 45, которое условно назовем «значением серого». Пересекая датчиком значение 45, робот будет менять направление движения. Очевидно, что по левую сторону от «серого» все показания датчика будут «белыми», а по правую «черными». В алгоритме заменим блоки ожидания показаний датчиков на «Жди черного» и «Жди белого» (рис. 8).

```
task main()
{
while (true){
  motor[motorB] = 50;
  motor[motorC] = 0;
  while(SensorValue[S1]>45);
  motor[motorB] = 0;
  motor[motorC] = 50;
  while(SensorValue[S1]<45);
}
}
```

Более устойчиво алгоритм работает, если в Robolab использовать моторы с управлением мощностью от –100 до 100. В этом случае есть возможность отрегулировать плавность поворота в соответствии с кривизной линии (рис. 9).

```
task main()
{
while (true){
  motor[motorB] = 80;
  motor[motorC] = 20;
  while(SensorValue[S1]>45);
  motor[motorB] = 20;
  motor[motorC] = 80;
  while(SensorValue[S1]<45);
}
}
```

В этом алгоритме притормаживающие моторы на повороте не останавливаются полностью, а лишь понижают мощность до 20 пунктов. Это делает поворот более плавным, но может привести и к потере линии на резком повороте. Поэтому числа 80 и 20 поставлены условно, их стоит подобрать самостоятельно.

Следующий пример движения вдоль границы черного и белого с использованием ветвления (рис. 10). В качестве значения перехода границы черного и белого взято то же число 45, которое было рассчитано для предыдущих алгоритмов.

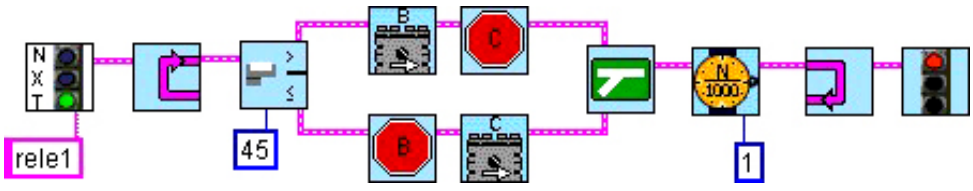


Рис. 10. Алгоритм движения вдоль границы черного и белого на релейном регуляторе с использованием ветвления

```
task main()
{
while (true){
  if (SensorValue[S1]>45) {
    motor[motorB] = 100;
    motor[motorC] = 0;
  }
  else {
    motor[motorB] = 0;
    motor[motorC] = 100;
  }
  wait1Msec(1);
}
}
```

Задержка в одну миллисекунду предназначена для того, чтобы немного разгрузить микроконтроллер: на небольших скоростях за это время ничего существенного не происходит.

В алгоритме использован модифицированный зеленый светофор «begin NXT», который позволяет передавать программы на контроллер не только с именем *rbl*, но и с любым другим длиной до 6 символов. Пиктограмма этого светофора находится в палитре «NXT». Используя различные имена, можно хранить в памяти контроллера сразу несколько программ. Имя указывается в прямоугольном модификаторе, присоединяемом к светофору (на рис. 10 имя *rele1*). В среде RobotC эта проблема решена иначе: программа загружается на «NXT» с тем именем, с которым сохранена на компьютере.

П-РЕГУЛЯТОР

Пропорциональный регулятор – это устройство, оказывающее управляющее воздействие $u(t)$ на объект пропорционально его линейному отклонению $e(t)$ от заданного состояния $x_0(t)$;

$$e(t) = x_0(t) - x(t),$$

где $x(t)$ – состояние в данный момент времени;

$$u(t) = ke(t),$$

где k – усиливающий коэффициент.

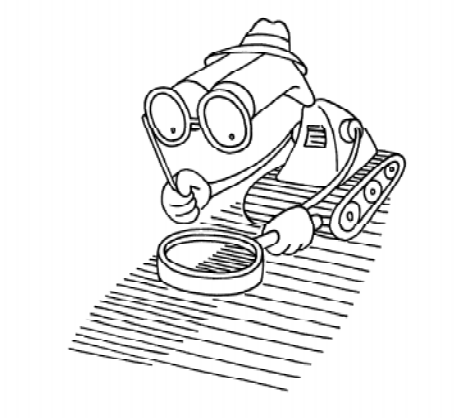
То есть, чем дальше робот отклоняется от заданного курса, тем активнее должны работать моторы, выравнивая его.

Движение по границе черного и белого тоже можно построить на П-регуляторе. Хотя внешне задача представляется решаемой только с помощью релейного регулятора, поскольку в системе присутствует всего два видимых человеческому глазу состояния: черное и белое. Но робот все видит иначе, для него отсутствует резкая граница между этими цветами. Можно сказать, он близорук и видит градиентный переход оттенков серого (рис. 11). Вот это нам и поможет построить П-регулятор.

Определяя состояние робота как показания датчика освещенности, научимся оказывать пропорциональное управляющее воздействие на моторы по следующему закону:

$$e = s_1 - grey,$$

где s_1 – текущие показания датчика, а $grey$ – заданное значение,



Можно сказать, он близорук и видит градиентный переход оттенков серого...

Рис. 11. Отличие восприятия робота и человека

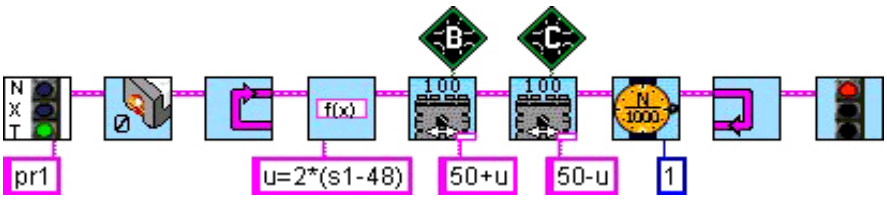


Рис. 12. Алгоритм движения по линии с одним датчиком освещенности на П-регуляторе

$$u = k*e,$$
$$MotorB = N + u,$$
$$MotorC = N - u,$$

где N – базовая мощность двигателей.

Так выглядит алгоритм пропорционального регулятора для одного датчика освещенности (рис. 12).

Величина s_1 – это показания датчика освещенности после его инициализации на первом порту (по умолчанию). В качестве заданного среднего значения датчика освещенности взято число 48. На моторы подается базовая мощность 50%, которая сохраняется при нулевом управляющем воздействии u. В данном случае отклонение $(s_1 - 48)$ умножается на усиливающий коэффициент 2, который на высоких скоростях может быть и больше.

Тот же алгоритм на RobotC.

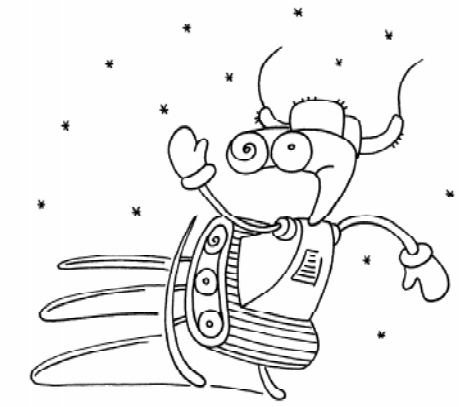
```
task main()
{
  int u;
  float k=2;
while (true) {
  u=k*(SensorValue[S1]-48);
  motor[motorB]=50+u;
  motor[motorC]=50-u;
  wait1Msec(1);
}
}
```

При слишком высоком усиливающем коэффициенте поведение робота напоминает движение на релейном регуляторе: его постоянно заносит из стороны в сторону. При низком коэффициенте робот может потерять линию. Для достижения наибольшего эффекта от П-регулятора следует найти оптимальное соотношение конструкции робота, базовой мощности (которая может достигать 100%) и усиливающего коэффициента.

КАЛИБРОВКА ДАТЧИКА

Обратимся к числу 48, использованному в формуле управления u. Это среднее арифметическое показаний датчика освещенности на черном и на белом, например $(40 + 56)/2 = 48$. Однако показания датчиков часто меняются по разным причинам: другая поверхность, изменение общей освещенности в помещении, небольшая модификация конструкции и т. п. (в предыдущих примерах было число 45, допустимое в комнатных условиях, а где-нибудь на стадионе это значение может вырасти на 20–30 пунктов). Поэтому имеет смысл научить робота самостоятельно вычислять среднее арифметическое, то есть значение границы белого и черного.

Есть несколько способов выполнить калибровку датчика. В простейшем случае вместо вычисления среднего арифметического просто понижается значение белого. Смысл способа в том, что робот снимает показания на белом, вычитает из него некоторое предполагаемое значение и полученное число считает границей белого и чер-



...его постоянно заносит из стороны в сторону.

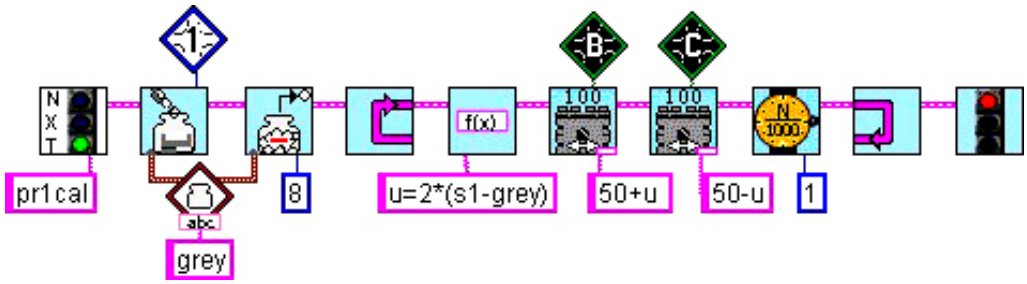


Рис. 13. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе с предварительной калибровкой (определением значения серого).

ного. Например, 56 – 8 = 48 можно считать показаниями датчика на сером (рис. 13).

Значение освещенности с датчика на порту 1 считывается в именованный контейнер *grey*, после чего оно уменьшается на число 8, и в формуле управления *u* переменная *grey* используется как заданное значение серого. Инициализация датчика происходит при вызове команды считывания в контейнер, поэтому отдельная команда не требуется. Другим способом аналогичную калибровку можно выполнить так (рис. 14).

Вообще говоря, в RobotC тоже требуется инициализация датчика. Произвести ее можно через меню **Robot → Motors and Sensors Setup**. В качестве типа датчика устанавливается **Light Active**. Также рекомендуется задать имя датчика и в программе использовать имя вместо привязанного к конкретному порту значения S_1. Однако в данной статье для краткости оставим прежний стиль.

```
task main()
{
  int u;
float k=2;
int grey=SensorValue[S1]-8;
while (true) {
  u=k*(SensorValue[S1]-grey);
  motor[motorB]=50+u;
  motor[motorC]=50-u;
  wait1Msec(1);
}
}
```

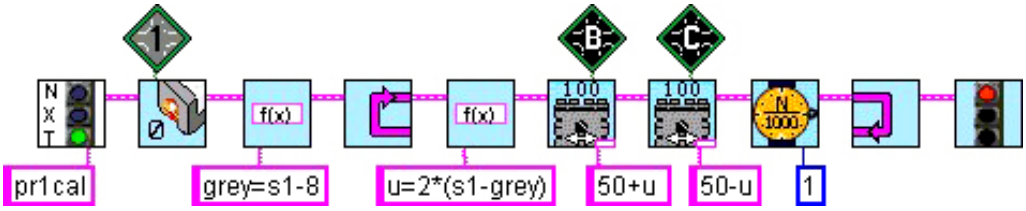


Рис. 14. Алгоритм движения по линии на пропорциональном регуляторе с инициализацией датчика и калибровкой

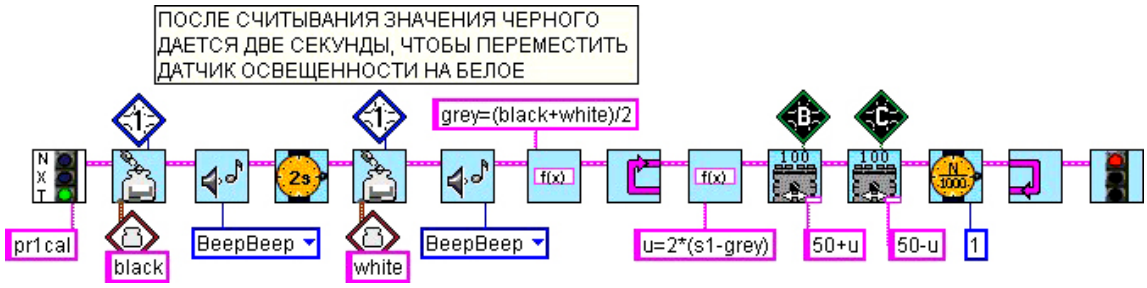
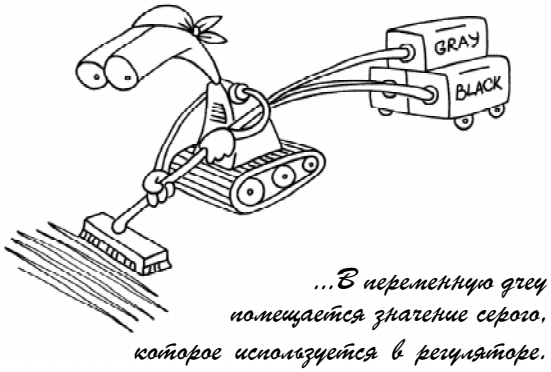


Рис. 15. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе с расчетом значения серого

Надо иметь в виду, что такой способ калибровки не учитывает все возможные варианты, а только экономит время на программирование и отладку. Если же времени достаточно, есть другой способ, при котором действительно производится расчет среднего арифметического (рис. 15).

```
task main()
{
  int u, white, black;
float grey, k=2;
white = SensorValue[S1];
PlaySound(SoundBeepBeep);
wait1Msec(2000);
black = SensorValue[S1];
PlaySound(SoundBeepBeep);
grey=(white+black)/2;
while (true) {
  u=k*(SensorValue[S1]-grey);
  motor[motorB]=50+u;
  motor[motorC]=50-u;
  wait1Msec(1);
}
}
```

Предложенный алгоритм обладает некоторым неудобством: при запуске потребуется быть внимательным и не пропустить звукового сигнала, после которого робот надо переместить так, чтобы датчик освещенности оказался над белым полем. Понятно, что вначале следовало поместить робот точно над черной линией. В контейнере ***black*** будет сохранено значение черного, в контейнере *white* – значение белого. В переменную *grey* помещается значение серого, которое используется в регуляторе. Сразу после второго звукового сигнала робот начнет движение.



Процесс калибровки можно сделать управляемым. Для этого после каждого считывания данных необходимо вставить ожидание какого-либо внешнего события, например нажатия на датчик касания, уменьшения расстояния на ультразвуковом датчике или просто нажатия на кнопку «NXT». Рассмотрим простейший пример с дополнительным датчиком касания, подсоединенным ко второму порту. Запустить программу имеет смысл, аккуратно установив тележку с датчиком освещенности над черной линией (рис. 16).

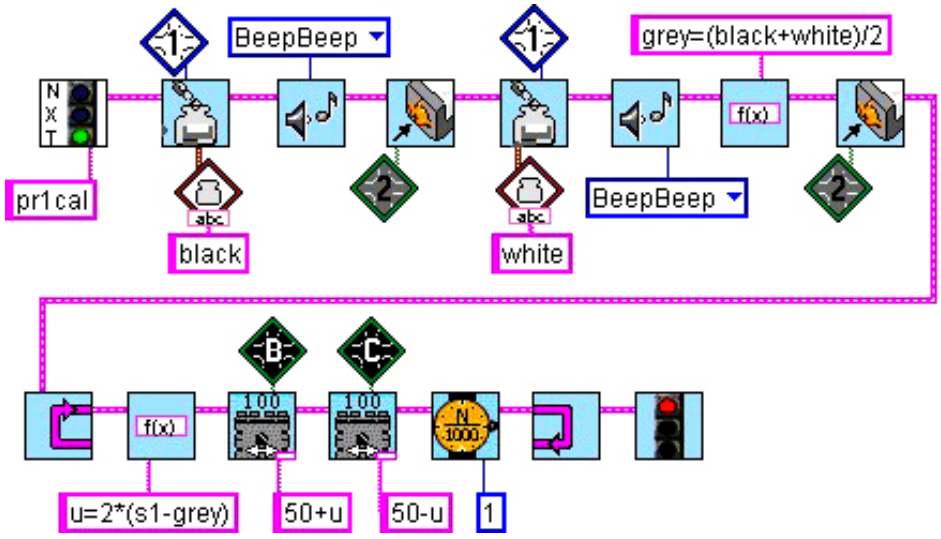


Рис. 16. Калибровка датчика освещенности с ожиданием касания

В двух последних примерах использованы именованные контейнеры (*black* и *white*), которые по сути являются переменными, как в обычном языке программирования. Обратите внимание, что звуковой сигнал перед стартом способствует тому, чтобы робот не среагировал дважды подряд на одно нажатие, что не исключено. В примере на языке RobotC эта проблема решается ожиданием отпускания датчика.

```
task main()
{
  int u, white, black;
float grey, k=2;
white = SensorValue[S1];
PlaySound(SoundBeepBeep);
while (SensorValue[S2]==0);
                //Ждать нажатия
while (SensorValue[S2]==1);
                //Ждать отпускания
black = SensorValue[S1];
PlaySound(SoundBeepBeep);
grey=(white+black)/2;
while (SensorValue[S2]==0);
while (SensorValue[S2]==1);
while (true) { // Начало движения
  u=k*(SensorValue[S1]-grey);
  motor[motorB]=50+u;
  motor[motorC]=50-u;
  wait1Msec(1);
}
}
```

После первого звукового сигнала нужно переставить тележку так, чтобы датчик освещенности оказался над белым. После второго – подготовиться к старту (датчик освещенности на границе между черным и белым), и по повторному нажатию кнопки робот начнет движение.

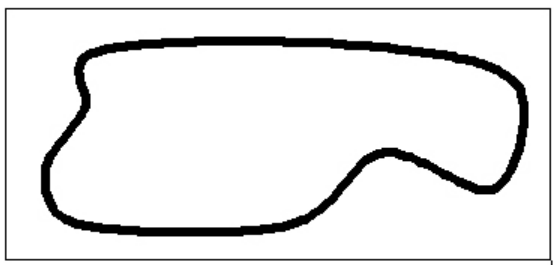


Рис. 17. Пример траектории для состязаний «Движение по линии»

Аналогичный опыт можно провести, используя датчик расстояния вместо датчика нажатия. Преимущество здесь в том, что старт робота будет осуществляться бесконтактно. Это поможет стартовать в точно выбранном положении. Только надо быть внимательным и нечаянно не провести рукой возле датчика расстояния при перемещении робота. Модифицируйте алгоритм самостоятельно. Команда ожидания объекта уже встречалась в предыдущей статье в алгоритме с поиском кеглей.

ПОЛЕ

Более интересную траекторию, чем окружность, стоит сделать самому на светлой поверхности достаточно большой площади с помощью той же черной изоленты. В качестве поверхности подойдет лист фанеры, оргалита или ПВХ, обратная сторона листа линолеума, баннерная ткань, белая клеенка и многое другое. Размеры поля желательно делать не меньше, чем 100×150 см. При разметке траектории следует учесть отступ от линии до края поля не менее 20 см, чтобы колеса робота не съезжали с трассы во время движения.

Имея определенный навык, можно наклеить изоленту так, что получится замкнутая кривая. Если не выходит одним куском, смело пользуйтесь ножницами, чтобы изгибы с малым радиусом кривизны составить из нескольких частей. Для начала не стоит рисовать слишком резких поворотов. Вот пример траектории (рис. 17).

Линию можно составить как из одной, так и из двух-трех полос изоленты или самоклеющейся пленки. Тогда роботу будет легче ориентироваться и не съехать с курса.

Наиболее интересные соревнования движения по черной линии в России проводятся в Политехническом музее в Москве [8]. В турнире на кубок Политехнического музея участвуют как Лего-роботы, так и самодельные творения молодых и взрослых робототехников. Ширина линии на этих состязаниях составляет 5 см, а длина превышает 10 м.

Литература

1. *С.А. Филиппов.* Робототехника для детей и родителей. Под ред. *А.Л. Фрадкова.* СПб.: Наука, 2010.

2. *М.С. Ананьевский, Г.И. Болтунов, Ю.Е. Зайцев, А.С. Матвеев, А.Л. Фрадков, В.В. Шиегин.* Санкт-Петербургские олимпиады по кибернетике. Под ред. *А.Л. Фрадкова, М.С. Ананьевского.* СПб.: Наука, 2006.

3. Сайт подразделения Lego Education: http://www.lego.com/education/.

4. Среда трехмерного моделирования Lego Digital Designer: http://ldd.lego.com/.

5. Среда программирования RobotC: http://www.robotc.net/.

6. Сайт поддержки пользователей Lego Mindstorms, Robolab 2.9.4 и пр.: http://www.legoengineering.com/.

7. Сайт о роботах, робототехнике и микроконтроллерах: http://www.myrobot.ru/.

8. Регламент состязаний роботов «Следование по линии»: http://railab.ru/kanikul/2011/line_v2_2.html.

Филиппов Сергей Александрович,
учитель информатики
физико-математического лицея
№ 239, методист.