Сергей Александрович Филиппов

ОСНОВЫ РОБОТОТЕХНИКИ НА БАЗЕ КОНСТРУКТОРА LEGO MINDSTORMS NXT. ЗАНЯТИЕ 4. ТАНЕЦ В КРУГЕ: ИГРАЕМ В КЕГЕЛЬРИНГ

На предыдущем занятии мы построили двухмоторный мобильный робот, способный перемещаться по комнате под управлением программы. Среда обитания робота может быть любой. Поместим его внутрь круга, за пределы которого запрещено выходить. Оказывается, в таких ограниченных условиях можно сделать много интересного.

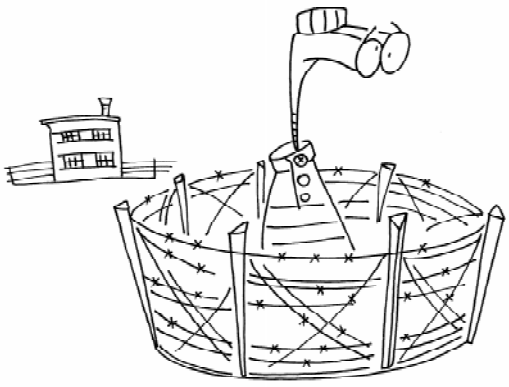
ТАНЕЦ В КРУГЕ

Для выполнения этой задачи надо собрать стандартную трехколесную тележку: два передних колеса ведущие, одно заднее подвижное на шарнире. Спереди по центру должен быть расположен датчик освещенности, направленный строго вниз и находящийся на расстоянии 5–10 мм от пола (рис. 1).

Рис. 1. Универсальный робот-«танцор»

Теперь приготовим ринг. Это может быть круг или его подобие диаметром около 100 см, очерченный двумя-тремя слоями черной изоленты (ширина черной линии около 50 мм). Вместо изоленты удобно использовать черную самоклеющуюся пленку. Цвет поверхности, на которой круг расположен, особого значения не имеет. Важно, чтобы она была светлой и однотонной. Главное условие успешности опыта состоит в том, что показания датчика на черной линии и внутри круга должны различаться не менее чем на 10–15 пунктов, а лучше на 20–25.

Робот ставится в центр и при старте должен двигаться внутри круга, не выходя за его пределы.



Поместим его внутрь круга, за пределы которого запрещено выходить.

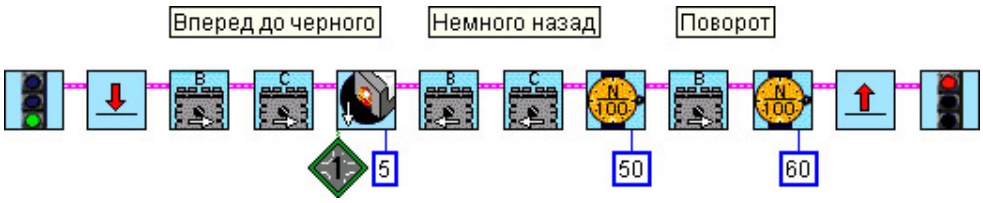


Рис. 2. Алгоритм «Танец в круге»

Последовательность действий такова:

1) ехать вперед, пока показания датчика не понизятся на 5 пунктов (лучше 10);

2) отъехать немного назад (полсекунды);

3) развернуться примерно на 120–150 градусов (тоже по времени);

4) повторять пункты 1–3 бесконечно.

Рассмотрите примеры программ на языках Robolab (рис. 2) и RobotC.

```
task main()
{
int white=SensorValue[S1];
        // Запомнить показания на белом
while (true){
  motor[motorB] = 100;
  motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
        // Ждать понижения на 5 пунктов
  motor[motorB] = -100;
  motor[motorC] = -100;
  wait1Msec(500);
  motor[motorB] = 100;
  wait1Msec(600);
}
}
```

В результате выполнения программы робот будет двигаться внутри круга, «вычерчивая» ломаную линию (рис. 3).

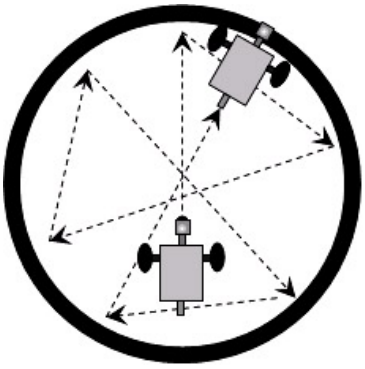


Рис. 3. Траектория движения робота в круге

Параметры, указанные в модификаторах, можно подобрать самостоятельно: степень понижения освещенности на черной линии, время отъезда назад и время поворота.

ВЫТОЛКНУТЬ ВСЕ БАНКИ

Вернемся к нашему кругу. Будем считать, что его диаметр – 1 м. Несколько пластиковых стаканчиков или пустых жестяных банок, расставленные внутри за черной линией (на расстоянии 12–15 см от нее), – это мусор, от которого необходимо очистить круг за кратчайшее время.

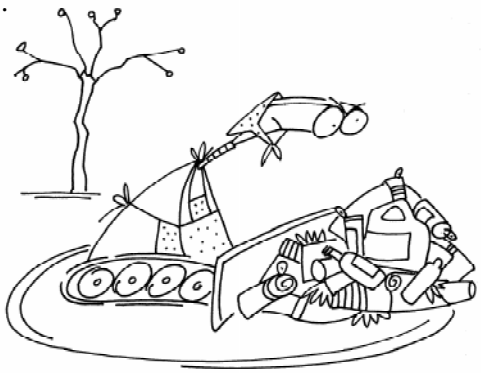
Первые попытки запуска робота покажут несколько недостатков:

1) стаканчики попадают под колеса, падают и плохо выталкиваются;

2) даже вытолкнутые стаканчики остаются частично внутри круга, поскольку, увидев край, робот сразу устремляется назад;

3) робот ведет себя как слон в посудной лавке;

4) робот делает много движений впустую.



Несколько пластиковых стаканчиков или пустых жестяных банок, расставленные внутри за черной линией... — это мусор, от которого необходимо очистить круг за кратчайшее время.

Избавимся от первого недостатка. Для этого следует построить бампер шириной 20–25 см рядом с датчиком освещенности (рис. 4). Подумайте, можно ли совсем спрятать датчик за бампером?

Второй и третий недостатки устраняются программно. Пусть, увидев край, тележка еще немного двигается вперед, выталкивая стаканчик, и только после этого отъезжает внутрь круга. Например, так (рис. 5).

Самый надежный способ заехать точно за пределы черной линии – это дождаться значения белого на датчике освещенности. Поэтому время можно заменить на «ожидание белого». Для экономии места стоит сгруппировать команды управления моторами, а также использовать «реверс» при смене направления на работающих моторах (рис. 6).

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  while(SensorValue[S1]<white-5);
  motor[motorB] = motor[motorC] = -100;
  wait1Msec(500);
  motor[motorB] = 100;
  wait1Msec(600);
}
}
```

Рис. 4. Бампер для защиты колес

Теперь стоит поработать над точностью движения, по возможности не теряя скорости.

В зависимости от конструкции робота, при резкой смене направления он может потерять равновесие или просто «встать на дыбы» на передние колеса. Поэтому последние несколько сантиметров можно проехать на торможении по инерции, то есть полностью освободив моторы.

И второе. Точность поворота будет зависеть от того, какие команды подаются на моторы и по какому принципу рассчитывается длительность поворота. К сожалению, таймер – ненадежный помощник. По инерции на малых промежутках времени робот может поворачиваться на различные углы.

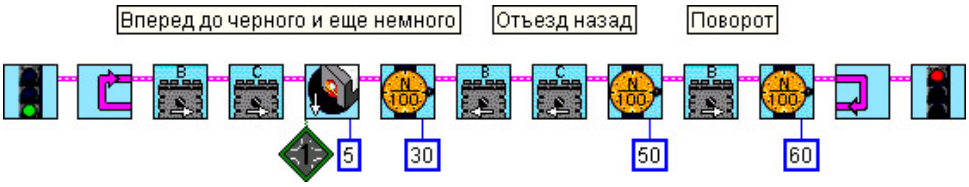


Рис. 5. Алгоритм «Танец в круге» с выталкиванием кеглей

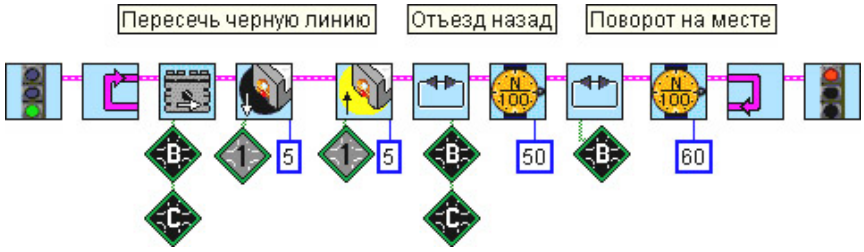


Рис. 6. Алгоритм «Танец в круге» с выездом точно за его пределы

Рис. 7. Алгоритм «Танец в круге» с плавным торможением

Мы можем пожертвовать реверсом в последней команде управления мотором B, для того чтобы достичь неторопливого движения обоими моторами. Длительность поворота при этом немного возрастет (рис. 7).

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  bFloatDuringInactiveMotorPWM = true;
  motor[motorB] = motor[motorC] = 0;
  while(SensorValue[S1]<white-5);
  bFloatDuringInactiveMotorPWM = false;
  motor[motorB] = motor[motorC] = -100;
  wait1Msec(500);
  motor[motorB] = -50;
  motor[motorC] = 50;
  wait1Msec(600);
}
}
```

Есть еще одна тонкость. На разных поверхностях соотношение времени вращения колес и реального перемещения тележки будет различным. Поэтому для надежности при возврате назад и повороте можно ввести ожидание оборотов моторов.

Для точности управления моторами необходимо использовать другой тип команд: с контролируемым вращением. Эти команды находятся в разделе «Advanced Output Control» и позволяют задавать мощность моторов от –100 до 100. В новом примере для компактности разместим все числовые параметры сверху, а модификаторы портов снизу (рис. 8).

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  bFloatDuringInactiveMotorPWM = true;
  motor[motorB] = motor[motorC] = 0;
  while(SensorValue[S1]<white-5);
  bFloatDuringInactiveMotorPWM = false;
  nMotorEncoder[motorC]=0;
        // Инициализация энкодера
  motor[motorB] = motor[motorC] = -100;
  while(nMotorEncoder[motorC]>-180);
  nMotorEncoder[motorC]=0;
  motor[motorB] = -50;
  motor[motorC] = 50;
  while(nMotorEncoder[motorC]<120);
}
}
```

Рис. 8. Алгоритм «Танец в круге» с движением по датчикам оборотов

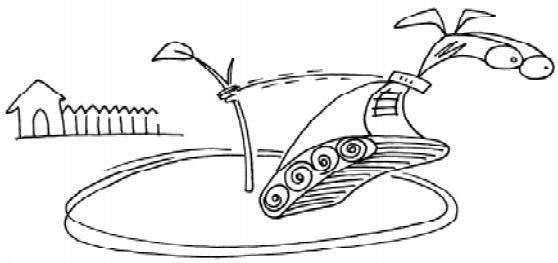
Четвертый недостаток, связанный с избыточными движениями, будет устранен далее.

НЕ ДЕЛАТЬ ЛИШНИХ ДВИЖЕНИЙ

Желание проконтролировать положение робота приводит к необходимости изменить траекторию движения таким образом, чтобы каждый раз, доехав до края, он возвращался в центр круга. Если бы не было встроенных датчиков оборотов, пришлось бы засекать время, которое робот ехал вперед, и давать команду столько же ехать назад. Точность такого решения оставляет желать лучшего, хотя и оно имеет право на существование. Вот соответствующий пример (рис. 9).

```
task main()
{
int white=SensorValue[S1];
while (true){
  ClearTimer[T1];  // Сбросить таймер
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  int t = time1[T1];
        // Запомнить прошедшее время
  ClearTimer[T1];
  motor[motorB] = motor[motorC] = -100;
  while(time1[T1]<t);
        // Двигаться назад то же время
}
}
```

Эта программа гоняет робота вперед–назад: до линии и на исходную позицию (пока без поворотов).



...чтобы каждый раз, доехав до края, он возвращался в центр круга.

К счастью, сервомоторы NXT имеют встроенный датчик оборотов, этим непременно надо воспользоваться. Но как определить, сколько оборотов надо сделать, чтобы вернуться в центр? Так же, как с таймером, запоминать результат в контейнер? Гораздо проще. Для этого нужно всего-навсего каждый раз обнулять показания датчика оборотов, когда робот оказывается в центре. Отчет времени невозможно повернуть вспять, чтобы снова придти в нулевую точку, а моторы можно. По замыслу робот проезжает некоторое количество оборотов вперед, после чего следует назад, пока на датчике оборотов снова не будет ноль.

В Robolab для этого следует использовать специфический блок, который не обнуляет показания датчика оборотов при вызове (с буквой A на пиктограмме). Кроме того, в примере добавлены модификаторы «Encoder C» из палитры «NXT Commands» для ясности различия между командами управления мотором и датчиками (рис. 10).

Рис. 9. Алгоритм «Танец в круге» с возвратом по времени

Рис. 10. Алгоритм «Танец в круге» с датчиками оборотов

```
task main()
{
int white=SensorValue[S1];
while (true){
  nMotorEncoder[motorC]=0;
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  bFloatDuringInactiveMotorPWM = true;
  motor[motorB] = motor[motorC] = 0;
  while(SensorValue[S1]<white-5);
  bFloatDuringInactiveMotorPWM = false;
  motor[motorB] = motor[motorC] = -100;
  while(nMotorEncoder[motorC]>0);
  nMotorEncoder[motorC]=0;
  motor[motorB] = -50;
  motor[motorC] = 50;
  while(nMotorEncoder[motorC]<60);
}
}
```

Результат уже значительно лучше, но робот все равно иногда промахивается мимо кеглей. Конечно, ведь он поворачивает вслепую. Надо бы оснастить его зрением. Для этого подойдет датчик расстояния. Прежде чем начать использовать его, попробуем разобраться, как «сонар» работает.

Два глазка, которые делают робот похожим на живое существо, служат для разных целей. Один из них передает ультразвуковой сигнал, другой принимает. По наблюдениям автора, передающим является правый, если смотреть на датчик со стороны разъема подключения, то есть сзади. Учитывая, что скорость ультразвука в воздухе относительно невелика (330–350 м/с), расположим датчик горизонтально, так чтобы передающий глазок был первым по ходу вращения робота. Тогда в процессе вращения больше вероятность, что отраженный сигнал будет уловлен принимающим глазком, идущим следом. В обратном случае часть стаканчиков робот будет пропускать.

Кроме того, датчик можно расположить вертикально, любой стороной, и он будет работать вполне сносно. Заодно усовершенствуем конструкцию, спрятав датчик освещенности за бампером (рис. 11).

Рис. 11. Ультразвуковой датчик для поиска кеглей и модифицированный бампер

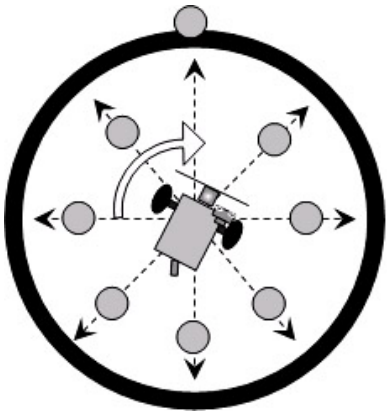


Рис. 12. Поиск и выталкивание кеглей с возвратом к центру

Рис. 13. Алгоритм «Танец в круге» с поиском кеглей

Итак, вращение осуществляется до тех пор, пока на датчик расстояния не поступит сигнал, например, «ближе 45». То есть обнаружен объект на расстоянии ближе 45 см. Надо иметь в виду, что на стаканчики, которые робот уже вытолкнул за линию, он не должен обращать внимания. Поэтому указанное расстояние не следует делать больше радиуса круга (рис. 12, 13).

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = 50;
  motor[motorC] = -50;
  while(SensorValue[S4]>45);
        // Ждем появление объекта
  nMotorEncoder[motorB]=0;
  motor[motorB] = motor[motorC] = 100;
  while(SensorValue[S1]>white-5);
  bFloatDuringInactiveMotorPWM = true;
  motor[motorB] = motor[motorC] = 0;
  while(SensorValue[S1]<white-5);
  bFloatDuringInactiveMotorPWM = false;
  motor[motorB] = motor[motorC] = -100;
  while(nMotorEncoder[motorB]>0);
}
}
```

В зависимости от конструкции робота, в приведенном алгоритме может быть один недостаток. Вернувшись в центр круга, робот начинает вращение до появления кегли. Однако, в силу неточности ультразвукового датчика, он может среагировать сразу на уже вытолкнутую кеглю. Поэтому имеет смысл, во-первых, выталкивать их подальше за пределы круга, во-вторых, начинать поворот в центре «вслепую», а в-третьих, все действия поначалу выполнять в замедленном темпе.

```
task main()
{
int white=SensorValue[S1];
while (true){
  motor[motorB] = 20;
  motor[motorC] = -20;
  wait1Msec(200);
  while(SensorValue[S4]>45);
  nMotorEncoder[motorB]=0;
  motor[motorB] = motor[motorC] = 50;
  while(SensorValue[S1]>white-5);
  bFloatDuringInactiveMotorPWM = true;
  motor[motorB] = motor[motorC] = 0;
  wait1Msec(500);
  bFloatDuringInactiveMotorPWM = false;
  motor[motorB] = motor[motorC] = -50;
  while(nMotorEncoder[motorB]>0);
}
}
```

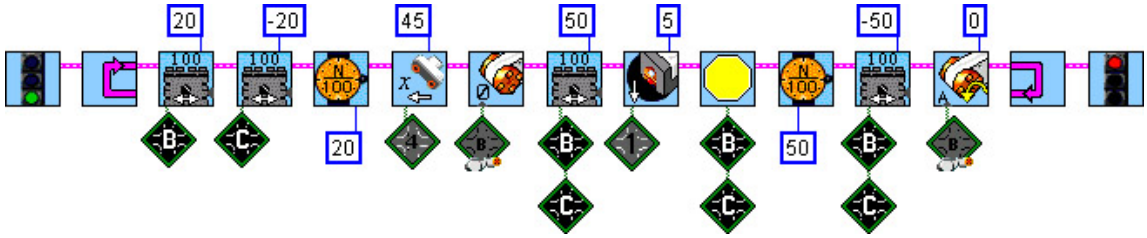
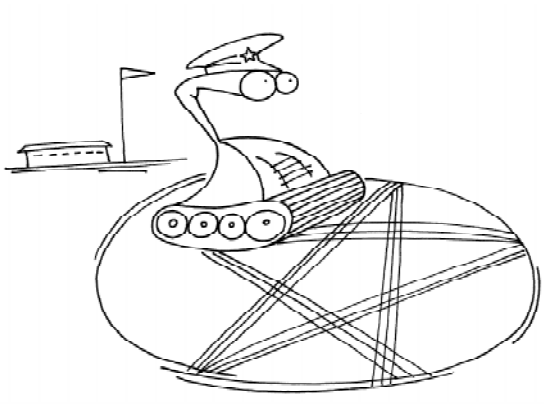


Рис. 14. Усовершенствованный поиск кеглей

Если все стаканчики оказались за пределами круга, надо попробовать еще раз, засечь время и подумать о том, как это сделать вдвое, втрое быстрее. Очевидно, сэкономить можно на остановках и поворотах, которые отнимают драгоценные секунды. Стоит подумать и над траекторией движения: возврат в центр совсем не является обязательным; вытолкнув одну кеглю, можно сразу браться за вторую. Робот способен вычерчивать звезды, восьмерки, спирали и другие фигуры, оптимальным образом очищая круг. Существует целая наука игры в Кегельринг, ее основы изложены на сайте http://www.myrobot.ru/ [9].

Состязания «Кегельринг» уже несколько лет проводятся в России и находят все больше поклонников, поскольку отличаются простотой и доступностью. Разнообразные роботы (не только из «Лего») выталкивают кегли из круга, возраст участников колеблется от 8 до 40 лет. Быть может, у того, кто прочел эту статью и попробовал в деле, найдется достаточно решимости подготовить своего робота и принять участие в подобных состязаниях. Оригинальные правила можно найти в Интернете [10].



Робот способен вычерчивать звезды, восьмерки, спирали и другие фигуры...

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. LEGO Technic Tora no Maki, ISOGAWA Yoshihito, Version 1.00 Isogawa Studio, Inc., 2007, http://www.isogawastudio.co.jp/legostudio/toranomaki/en/.

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

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

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

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

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

9. Кегельринг: как сделать робота и участвовать в соревнованиях http://myrobot.ru/articles/sport_kegelring.php.

10. Регламент состязаний роботов «Кегельринг»: http://myrobot.ru/sport/index.php?n=Reglaments.Kegelring.

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