В. А. Векслер,
Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского

ПРОГРАММИРОВАНИЕ РОБОТОВ VEX IQ

Аннотация
Сегодня образовательная робототехника дает возможность уже на ранних этапах выявить технические наклонности учащихся. Робототехнику можно использовать в начальном, основном и среднем общем образовании в рамках урочной, кружковой и факультативной деятельности. В статье рассматриваются базовые примеры решения робототехнических задач проектирования движения конструкции, анимирования при помощи мозгового центра и регулирования точных поворотов робота, придания ему «интеллектуальности» и стабилизации при передвижении посредством набора датчиков.

Ключевые слова: робот, образовательная робототехника, программирование, кружковая деятельность, моделирование.

Контактная информация

Векслер Виталий Абрамович, канд. пед. наук, доцент кафедры информационных систем и технологий в обучении Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского; *адрес:* 410012, г. Саратов, ул. Астраханская, д. 83; *телефон:* (8452) 23-72-51; *e-mail:* vitalv74@mail.ru

V. A. Veksler,
Saratov State University

PROGRAMMING VEX IQ ROBOTS

Abstract
Today educational robotics provides an opportunity to identify the early technical inclinations of students and develop them in this direction. The article considers the basic examples of solving robotic problems of design of the structure motion, animation with the help of the brain center and the regulation of the exact turns of the training robot, giving it "intelligence" and stabilization when moving with the help of sensors.

Keywords: robot, educational robotics, programming, club activities, modeling.

Сегодня робототехнические кружки прочно вошли в жизнь современной школы и системы дополнительного образования. Дети развивают свое инженерное, наглядно-образное, пространственное и логическое мышление, мелкую моторику, приобретают знания по основам механики и программированию. Это совершенно новый вид занятий, реализующийся во время урочной и внеурочной деятельности, позволяющий по-настоящему вдохновить ребенка на образовательную деятельность путем конструирования и программирования. Повсеместное внедрение в современный образовательный процесс нового педагогического инструментария, основанного на технологиях образовательной робототехники, способствует формированию личностных, регулятивных, коммуникативных, познавательных универсальных учебных действий, являющихся важной составляющей ФГОС.

Существует большой ряд робототехнических конструкторов, созданных для работы в рамках образовательной деятельности. Выделим **набор VEX IQ**, который представляет собой серию конструкторов, созданных для самых маленьких учеников (от 8 до 14 лет). Датчики VEX IQ (датчики света, гироскоп, датчик расстояния, датчики прикосновения) помогают ребенку создать более продвинутых роботов, реагирующих на состояние внешней среды.

В качестве базовой основы возьмем **стандартную модель Clawbot IQ with Sensors и язык программирования RobotC**. Рассмотрим ряд примеров по программированию робота.

Приведем расположение в модели моторов и датчиков (схема подключения к портам).

Моторы:

порт 1: *leftMotor* (левый привод);
порт 6: *rightMotor* (правый привод);
порт 10: *armMotor* (рука);
порт 11: *clawMotor* (клешня).

Датчики:

порт 2: *touchLED* (Touch LED — сенсор касания);
порт 3: *colorDetector* (Color Sensor/Hue Mode — сенсор определения цвета);
порт 4: *gyroSensor* (Gyro — гироскоп);
порт 7: *distanceMM* (Distance — сенсор определения расстояния)
порт 8: *bumpSwitch* (Bumper Switch — переключатель).

Одной из интересных задач начального уровня становится **задача передвижения робота по комнате**.

Пример 1.

Робот передвигается вперед, пока не уткнется в препятствие. При обнаружении препятствия робот должен остановиться. Запуск движения — нажатие на датчик Touch LED (1 — нажат, 0 — не нажат). Остановка движения — срабатывание датчика Bumper Switch при упоре в препятствие (в этом случае значение переменной данного сенсора становится равным 1, в ненажатом состоянии оно равняется 0).

```
while (getTouchLEDValue(touchLED) == 0 {}
playSound(soundAirWrench);
while (getBumperValue(bumpSwitch) == 0)
 { setMotorSpeed(leftMotor, 127);
   setMotorSpeed(rightMotor, 127);
   sleep(2000); }
```

Пример 2.

«Робот-автопилот». При обнаружении препятствия модель должна немного отъехать назад, развернуться и поехать в другую сторону.

```
while (getTouchLEDValue(touchLED) == 0) {}
playSound(soundAirWrench);
repeat (forever) {
 if(getBumperValue(bumpSwitch) == 0)
  { setMotorSpeed(leftMotor, 127);
    setMotorSpeed(rightMotor, 127);
    sleep(20); }
 else
  { setMotorSpeed(leftMotor, -127);
    setMotorSpeed(rightMotor, -127);
    sleep(500);
    setMotorSpeed(leftMotor, 127);
    setMotorSpeed(rightMotor, 0);
    sleep(1500); }}
```

Пример 3.

Доработайте модель «Робот-автопилот». Для упрощения обращения к моторам используйте возможности методов. Осуществите поворот случайным образом. Скорости моторов при каждом новом развороте должны зависеть от случайных чисел.

```
void move (long lm, long rm, long tm)
{ motor[leftMotor] = lm;
  motor[rightMotor]= rm;
  wait1Msec(tm); }
task main(){
repeat(forever) {
 if (SensorValue[bumpSwitch] == 0)
  { move(127, 127, 100); }
 else
  { move(-60, -60, 400);
    srand(nSysTime);
    move(rand() % 61 ,-1*rand() % 61,400); }}
```

«Мозговой центр» робота использует технологии с широкими функциональными возможностями. Кроме портов для подключения датчиков «мозговой центр» обладает LCD-дисплеем. Дисплей позволяет отображать не только информацию служебного характера, определенную разработчиком, но и информацию, программируемую пользователем. Дисплей работает в двух режимах:

- в текстовом режиме он отображает символы на одной из шести возможных текстовых строк (0 — верхняя строка, 5 — нижняя строка);
- в графическом режиме отображение объекта происходит по координатам: 48 вертикальных пикселей (*Y — yPos* от 0 до 47) и 128 горизонтальных пикселей (*X — xPos* от 0 до 127), начало отсчета — правый нижний угол.

Пример 4.

При приближении робота к объекту красного цвета необходимо вывести на экран сообщение «Hello Red».

```
setMultipleMotors(20, leftMotor, rightMotor);
waitUntil(getColorName(colorDetector) ==
  colorRed);
string printMe = "Hello";
displayBigStringAt(10, 21, "%s ", printMe);
```

Пример 5.

При движении робота выражение «I am go» двигается вертикально по строчкам монитора. Робот останавливается при приближении к объекту зеленого цвета.

```
string printMe = "I am go";
int NumberLine = 0;
while ((getColorValue(colorDetector) < 51) ||
(getColorValue(colorDetector) > 130))
 { setMultipleMotors(20, leftMotor, rightMotor);
   wait1Msec(1000);
   displayClearTextLine(NumberLine); // очистка
                                       строки
   NumberLine++;
   if (NumberLine == 7) NumberLine = 0;
   displayCenteredTextLine(NumberLine, "%s",
printMe); }
```

Переключается датчик цвета в режим «Hue». Цветовой датчик VEX IQ при этом возвращает оттенок как значение в диапазоне от 0 до 255. Становится возможным использовать следующие диапазоны спектра для определения цвета:

- красный — от 0 до 50 и от 220 до 255;
- зеленый — от 51 до 130;
- синий — от 131 до 219.

Пример 6.

Нарисуйте роботу глаза.

```
drawEllipse(1, 46, 63, 1);
fillCircle(31, 23, 10);
drawEllipse(64, 46, 126, 1);
fillCircle(95, 23, 10);
```

Пример 7.

Простейшая анимация: робот закрывает глаза, когда видит объект очень темного цвета. Дополнительно выведите значение датчика серого цвета.

```
int er=1;
while(true) {
 if(getColorGrayscale(colorDetector) < 10)
  { if (er == 0)
    { er = 1;
      fillEllipse(1, 46, 63, 8);
      fillEllipse(64, 46, 126, 8);}}
 else
  { if (er == 1)
    { eraseDisplay();
      er=0;
      drawEllipse(1, 46, 63, 8);
      fillCircle(31, 23, 10);
      drawEllipse(64, 46, 126, 8);
      fillCircle(95, 23, 10); }}
 drawUserText(1, 7, "ColorGrayscale : %d",
 getColorGrayscale(colorDetector)); }
```

Для определения черного цвета используем функцию *getColorGrayscale* (в режиме «grayscale mode»), которая возвращает значение датчика цвета, в режиме «оттенки серого». Значения, возвращаемые цветовым датчиком, обычно находятся в диапазоне от 0 до 400. Темные объекты возвращают более низкие значения, светлые объекты возвращают более высокие значения. Возвращенные значения будут различаться в зависимости от обнаруженной поверхности, внешнего освещения и других факторов окружающей среды (например, более яркий окружающий свет может привести к тому, что датчик вернется к более высоким средним значениям, чем обычно). В код вводится дополнительная переменная *er*, представляющая собой флаг для избежания повторных прорисовок.

Пример 8.

Простейшая анимация: если робот смотрит на менее насыщенный объект, его зрачки увеличиваются, иначе — уменьшаются.

```
float radius=10;
drawEllipse(1, 46, 63, 9);
```

```
drawEllipse(64, 46, 126, 9);
while (1) {
 if (getColorSaturation(colorDetector) < 60)
  { radius = radius + 0.001;
    if (radius > 16) radius = 16; }
 else
  { eraseCircle(31, 28, radius);
    eraseCircle(95, 28, radius);
    radius = radius - 0.001;
    if (radius < 3) radius = 3; }
 fillCircle(31, 28, radius);
 fillCircle(95, 28, radius);
 drawUserText(1, 7, "ColorSat : %d %d",
 getColorSaturation(colorDetector), radius); }
```

В коде используется функция сенсора для определения насыщенности *getColorSaturation* (в режимах «Hue» и «Color»). Возвращенные функцией значения варьируются от 0 (низкая насыщенность) до 255 (высокая насыщенность). Насыщенность при этом представляет собой воспринимаемую датчиком интенсивность цвета.

В сочетании с функциями датчиков дисплей может быть использован как часть механизма реакции робота на определенные события, протекающие вокруг него. Дисплей превращается в простейший монитор, с которым ребенок может проводить базовые манипуляции. На дисплее могут быть выведены также данные, рассчитанные по параметрам протекающей ситуации.

При программировании роботов для точных поворотов учащимся, только начавшим изучать язык RobotC, приходится подбирать экспериментально скорость и время. При этом зачастую данные параметры должны определяться индивидуально для каждого робота. Ситуацию может исправить специальный вид датчика, Gyro Sensor — гироскоп или контроллер угловой скорости. Гироскопы позволяют сделать повороты очень точными и независимыми от скорости, с которой поворачивается сам робот. Датчик гироскопа измеряет угол поворота робота. На поверхности датчика изображена стрелка, указывающая направление. Необходимо установить датчик плотно на роботе, сторона с рисунком должна находиться параллельно к измеряемой поверхности. Перед началом измерения датчик должен зарегистрировать текущее положение как нулевую точку. Если робот поворачивается против часовой стрелки, изменения регистрируются как положительные значения угла. Если же он поворачивается по часовой стрелке, датчик регистрирует отрицательное значение.

Задания, предлагаемые учащимся, выдаются по мере нарастания сложности решаемых проблем. В ходе работы ребята закрепляют на практике все изученные ранее алгоритмические конструкции языка программирования.

Пример 9.

Развернуть робота на 90 градусов влево.

```
resetGyro(gyroSensor);
repeatUntil(getGyroDegrees(gyroSensor) > 90)
 { setMotorSpeed(leftMotor, -50);
   setMotorSpeed(rightMotor, 50);}
setMotorSpeed(leftMotor, 0);
setMotorSpeed(rightMotor, 0);
```

Функция *getGyroDegrees* возвращает текущее вращательное значение датчика в градусах. Перед выполнением поворотов значения гироскопа сбрасываются при помощи функции *resetGyro*. Обратите внимание: робот может повернуться и более чем на 90 градусов из-за дрейфа (отклонения), вызванного импульсом.

Пример 10.

Устраните возможный дрейф из решения примера 9.

```
while (getGyroDegrees(gyroSensor) < 90) {
 if (getGyroDegrees(gyroSensor) < 70)
  { setMotor(leftMotor, -50);
    setMotor(rightMotor, 50); }
 else
  { setMotor(leftMotor, -10);
    setMotor(rightMotor, 10); }}
setMotor(rightMotor, 0);
setMotor(rightMotor, 0);
```

Пример 11.

Развернуть робота на 90 градусов вправо.

```
resetGyro(gyroSensor);
repeatUntil(getGyroDegrees(gyroSensor) < -90)
 { setMotorSpeed(leftMotor, 50);
   setMotorSpeed(rightMotor, -50);}
setMotorSpeed(leftMotor, 0);
setMotorSpeed(rightMotor, 0);
```

Пример 12.

Определите, на сколько градусов в секунду разворачивается робот при развороте на 180 градусов с мощностью моторов 70 единиц.

```
resetGyro(gyroSensor);
repeatUntil(getGyroDegrees(gyroSensor) > 180)
 { setMotorSpeed(leftMotor, -70);
   setMotorSpeed(rightMotor, 70);
   displayTextLine(1, "   Rate: %d",
getGyroRate(gyroSensor)); }
setMotorSpeed(leftMotor, 0);
setMotorSpeed(rightMotor, 0);
```

Следующая задача придает нашему роботу задатки «псевдоинтеллектуальности». Также в рамках данной задачи учащиеся знакомятся с возможностями упрощения кода путем создания отдельных функций, вызываемых из основной программы.

Пример 13.

После ручного изменения ориентации должен произойти возврат робота в исходную градусную ориентацию. Отобразите угол поворота на экране.

```
void move (int Vl, int Vr, int Tm)
{ motor[leftMotor]=Vl;
  motor[rightMotor]=Vr;
  sleep(Tm); }
task main() {
 resetGyro(gyroSensor);
 repeat (forever)
 { displayTextLine(1, "  degrees : %d",
getGyroDegrees(gyroSensor));
   while(getGyroDegrees(gyroSensor) != 0) {
    if (getGyroDegrees(gyroSensor) < 0)
      move(-10, 10, 2);
    else move(10, -10, 2);
    displayTextLine(1, "  degrees : %d",
getGyroDegrees(gyroSensor)); }
 move(0, 0, 5); }}
```

Пример 14.

Робот управляется джойстиком (одним стиком). После нажатия датчика Touch LED робот возвращается в исходную градусную ориентацию.

```
resetGyro(gyroSensor);
repeat (forever) {
 move(vexRT[ChA], vexRT[ChB], 2);
 displayTextLine(1, "  degrees : %d",
getGyroDegrees(gyroSensor));
 if (SensorValue[touchLED] == 1)
 { while(getGyroDegrees(gyroSensor) != 0)
  { if (getGyroDegrees(gyroSensor) < 0)
      move(-10, 10, 2)
    else move(10, -10, 2);
    displayTextLine(1, "  degrees : %d",
      getGyroDegrees(gyroSensor)); }
  move(0, 0, 5); }}
```

Пример 15.

Робот управляется джойстиком (двумя стиками). После того как оба стика приведены в нулевое положение, робот возвращается в исходную градусную ориентацию.

```
resetGyro(gyroSensor);
repeat (forever)
 { move(vexRT[ChA], vexRT[ChD], 2);
   displayTextLine(1, "  degrees : %d",
getGyroDegrees(gyroSensor));
   if (vexRT[ChA] == 0 && vexRT[ChD] == 0)
    { while(getGyroDegrees(gyroSensor) != 0)
      { if (getGyroDegrees(gyroSensor) < 0)
          move(-10, 10, 2);
        else move(10, -10, 2);
      displayTextLine(1, "  degrees : %d",
        getGyroDegrees(gyroSensor)); }
   move(0, 0, 5); }}
```

Рассмотрев с учащимися приведенные задачи, можно предложить им придумать самостоятельно алгоритмы управления роботом, в которых можно использовать изученные датчики. В частности, предлагается задача о путях совершенствования и стабилизации передвижения роботов при помощи гироскопа.

* * *

Образовательная робототехника сегодня становится полноценным педагогическим инструментом, закладывающим у учащихся базис системного, логического и практического мышления, интеграции информатики с другими дисциплинами. Занятия робототехникой дают хороший задел на будущее, вызывают у ребят интерес к научно-техническому творчеству и программированию, заметно способствуют целенаправленному выбору профессий инженерной направленности. Программируя роботов, школьник изучает основные алгоритмические конструкции, сразу понимая их практическое назначение. Именно «сиюминутная практическая реализация» позволяет повысить как мотивацию, так и интерес ребенка к получению более высоких результатов. Доступный и понятный робототехнический конструктор VEX IQ позволяет поднять изучение робототехники уже в младших классах на совершенно новый уровень.

Список использованных источников

1. *Горнов О. А.* Основы робототехники и программирования с Vex EDR. М.: Экзамен, 2016.

2. *Филиппов С. А.* Основы робототехники на базе конструктора LEGO MINDSTORM NXT. Занятие 6. Ориентация на местности: объезжаем стены // Компьютерные инструменты в школе. 2010. № 6.

НОВОСТИ

Робота Autodesk научили собирать LEGO с помощью машинного обучения

Команда исследователей из Autodesk AI Lab в Сан-Франциско разработала проект BrickBot — роботизированную систему, которая учится анализировать окружающие условия и адаптироваться к ним для того, чтобы правильно выполнять поставленную перед ней задачу. Робот BrickBot оснащен камерами, сенсорами и нейронными сетями, которые позволяют ему обрабатывать поступающую информацию и реагировать на нее.

Сегодня промышленные роботы на фабриках запрограммированы на выполнение одной и той же задачи — например, точечной сварки в строго определенном месте на автоматическом конвейере для сборки. Они абсолютно не способны приспосабливаться к изменяющимся обстоятельствам реального мира. При этом на их обучение требуются многие месяцы или даже годы. Цель проекта BrickBot — изменить этот подход и предложить строительным и промышленным предприятиям умные машины, которые смогут не только адаптироваться к окружающей среде, но и понимать, какой цели им нужно достичь.

Обучение BrickBot происходит с помощью кирпичиков LEGO. Задача робота — разобраться в горстках игрушечных кубиков, используя машинное обучение, выбрать соответствующие детали для заданной конструкции, а затем собрать ее, хватая и складывая кирпичики вместе в правильной последовательности и в нужном месте. Система учится работать с кирпичиками всех возможных конфигураций — прямоугольниками, квадратами, миниатюрными человечками и т. д., поскольку в промышленности компоненты могут быть абсолютно любой формы.

«Использование пластиковых кубиков позволило нам сохранять контроль над проектом и в то же время иметь свободу для экспериментов на всех этапах от стадии проектирования до конечного продукта, — сказал Йотто Кога (Yotto Koga), архитектор ПО Autodesk с ученой степенью в области робототехники. — Теперь мы близки к тому, чтобы предпринять следующий шаг. Мы планируем выбрать по одному клиенту из промышленной и строительной отраслей для того, чтобы вместе с ними посмотреть, как технология BrickBot может применяться в реальном мире».

На ранних этапах BrickBot использовал машинное обучение для получения данных с сенсоров и пытался с помощью изображений с камер выбрать правильные кирпичики LEGO. Это не принесло нужного результата, поэтому Кога попробовал использовать стереозрение для получения роботом более подробной информации. Наконец ему были добавлены сенсоры глубины, которые могли воспринимать цвет. Постепенно BrickBot научился ориентироваться между разными цветами и брать правильный кирпичик из ящика.

Следующий шаг — позволить системе выйти за пределы получаемых инструкций и пошагового создания модели. Предполагается, что робот может при этом выбрать последовательность шагов, совершенно отличающуюся от человеческой. И таким образом от сегодняшней работы с маленькими кирпичиками LEGO BrickBot сможет со временем перейти к более масштабным промышленным проектам.

(По материалам CNews)