TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatika instituut
Tarkvaratehnika õppetool

LEGO tüüpi roboti C keeles programmeerimise kursus

Bakalaurusetöö

Üliõpilane: Roman Trohhalev
Üliõpilaskood: 112170IAPB
Juhendaja: Kaarel Allik

Tallinn
2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Selle töö eesmärgiks on luua Lego Mindstorms EV3 roboti C keeles programmeerimise kursus. See kursus on mõeldud lastele alates 16. eluaastast, kellel on juba kogemus programmeerimises ja kes sooviks laiendada oma teadmisi programmeerimises ja robootika alusel.
Töö kirjeldab põhireeglid, kuidas kirjutada programme robotile Lego Mindstorms EV3 kasutades C keele, on programmide näited, kus selgitatakse kuidas töötavad tsüklid, mootorid ja andurid, on antud ülesandeid iseseisvaks tööks

Töö käigus kõik püstitatud eesmärgid oli saavutatud. Tulemuseks on loonud kursus, mis võib anda lastele alused programmeerimise ja robootika mõistmises.

Lõputöö on kirjutatud vene keeles ning sisaldab teksti 39 leheküljel, 9 peatükki, 10 joonist, 2 tabelit.

Abstract

The aim of this thesis is to create a course of programming robot Lego Mindstorms EV3 using C language. This course is designed for children from the age of 16 who already have experience in programming and would like to expand their knowledge in the field of programming and the basics of robotics.
The thesis describes the basic rules of writing programs for robot Lego Mindstorms EV3 using C language, are given examples of programs to work with loops, motors and sensors and tasks for independent work.

During this work, the author's goals were successfully achieved and as a result, a course was created, that can give children the basics of understanding programming and robotics.

The thesis is written in Russian and contains 39 pages of text, 9 chapters, 10 figures, 2 tables.

Аннотация

Целью данной дипломной работы является создание курса программирования робота Lego Mindstorms EV3 на языке программирования C. Данный курс предназначен для детей от 16-ти лет, которые уже имеют опыт в программировании и хотели бы расширить свои познания как в области программирования, так и в основах робототехники.
В работе описаны основные правила написания программ для робота Lego Mindstorms EV3 на языке C, приведены примеры программ по работе с циклами, моторами и датчиками, приведены задания для самостоятельной работы.

В ходе работы все поставленные цели были достигнуты. Как результат, был создан курс, который может дать детям основы в понимании программирования и робототехники.

Данная работа написана на русском языке и состоит из 39 страниц текста, 9 глав, 10 иллюстраций, 2 таблиц.

Словарь терминов и аббревиатур

Образ диска ***Image***

файл, содержащий в себе полную копию содержания и структуры файловой системы и данных, находящихся на диске, таком как компакт-диск, дискета, раздел жёсткого диска или весь жёсткий диск целиком. [1]

Прошивка ***Firmware***

образ ПЗУ, предназначенный для записи в память соответствующего устройства с целью обновления его микропрограммы, а также собственно процесс записи этого образа в энергонезависимую память устройства. [2]

USB ***Universal Serial Bus***

последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств в вычислительной технике. [3]

Энкодер ***Encoder***

это устройство преобразующее линейное или угловое перемещение в последовательность сигналов, позволяющих определить величину перемещения. [4]

ИК ***Infrared***

сокращение для слова «инфракрасный».

Список иллюстраций

Список таблиц

Содержание

1. Введение

Начиная с 1960-х годов, кубики LEGO использовали в школе для преподавания различных дисциплин. В 1980 году компанией LEGO было принято решение об организации отдельного департамента развития образовательных продуктов. В 1989 году департамент был реформирован и получил название LEGO Dacta. Сегодня образовательная продукция компании LEGO выпускается под брендом LEGO Education. Отличительной особенностью продукции LEGO Education от традиционных конструкторов LEGO является сфера использования продукта: детские сады, школы и другие учебные учреждения, которые подразумевают участие в образовательном процессе профессионального преподавателя.

LEGO Mindstorms — конструкторы для создания программируемых роботов и соответствующее программное обеспечение. Впервые был представлен в 1998 году. Всего существует три поколения роботов LEGO Mindstorms RCX 1.0 (1998), NXT 2.0 (2006) и EV3 (2013 год).

Роботы применяются на уроках физики, информатики, математики и программирования в средней школе.

Лего-роботы Mindstorms принимают участие в соревнованиях по робототехнике. Самые крупные международные соревнования: WRO (World Robot Olympiad), FIRST Robotics Competition. [5]

Целью данной дипломной работы является создание курса программирования робота Lego Mindstorms EV3 на языке программирования C. Данный курс предназначен для детей от 16-ти лет, которые уже имеют опыт в программировании и хотели бы расширить свои познания как в области программирования, так и в основах робототехники.

1.1 Предпосылки к написанию работы и описание проблемы

Автор является учителем робототехники в начальных и старших классах, именно поэтому появилась идея написания курса программирования робота Lego Mindstorms EV3 на языке программирования C, а также по просьбе учащихся, которые желают

получать навыки робототехники, параллельно изучая программирование. Поиск показал, что материалов по данной версии робота, на момент написания работы, очень мало, в силу того, что робот появился на рынке относительно недавно, в 2013-ом году. Поиск материалов также показал, что руководств, по написания программ на языке С для LEGO роботов, на русском языке нет. Это повлияло на выбор языка работы, так как преподавание ведётся на русском языке. Модель робота была выбрана в силу ограниченности ресурсов для преподавания.

1.2 Основные учебные цели

Основными целями данного курса являются:

- Знакомство с основами Lego-конструирования, программирования и работы с компьютером;
- Общее разностороннее развитие учащихся благодаря выполнению исследований, написания отчётов, а также общению в процессе работы;
- Развитие интереса к ключевым областям наук, технологии, инженерии, математики и другим предметам школьной программы;

1.3 Обзор работы

Курс условно поделён на 2 части. Первая часть – это изучение основ программирования и изучение работы датчиков. Вторая часть – это внеклассная деятельность, изучение подпрограмм-функций и подготовка к соревнованиям. В моей работе описана первая часть курса – изучение основ.

Работа поделена по главам следующим образом:

Глава 2 посвящена годовому плану и задачам, которые преследует данный курс.

Главы 3-8 посвящены изучению основ программирования на языке С и базовым программам в среде RobotC.

Глава 9 полностью посвящена изучению работы датчиков, с примерами работы датчиков и списком заданий для самостоятельного решения.

2. Годовой план курса

2.1 Задачи курса

- Развитие творческого мышления при создании действующих моделей;
- Создание условий для экспериментальных исследований;
- Проведение систематических наблюдений и измерений;
- Развитие логического мышления;
- Проектирование, строительство и демонстрация роботов, которые могут быть разработаны и протестированы для выполнения конкретных задач;
- Эксперименты со скоростью и расстоянием под воздействием силы трения;
- Развитие исследовательских навыков, навыков выдвижения гипотез и прогнозирования;
- Решение практических заданий и развитие навыков работы в команде.

Курс рассчитан на 35 недель. Каждую неделю изучается новая тема. Курс условно поделён на 2 части. Первая часть – это изучение основ, которая длится 15 недель. Вторая часть – это внеклассная деятельность и подготовка к соревнованиям. В моей работе описана первая часть – изучение основ.

Таблица 1. Годовой план курса.

Номер недели	Тема	Цели
1	Ведение	Знакомство с роботом Lego Mindstorms EV3. Базовый принципы конструирование. Создание модели базового робота.
2	Программирование. Моторы	Программирование движений по различным траекториям
3	Программирование. Вывод информации	Работа с экраном, работа с подсветкой кнопок на блоке
4	Программирование. Вывод информации	Работа со звуком
5	Программирование. Цикл	Цикл с постусловием
6	Программирование. Цикл	Цикл “переключатель”
7	Программирование. Работа с данными	Типы данных. Проводники
8	Программирование. Работа с данными	Переменные и константы
9	Программирование. Работа с данными	Математические операции с данными
10	Программирование. Работа с данными	Логические операции с данными

11	Работа с датчиками	Датчик касания
12	Работа с датчиками	Датчик цвета
13	Работа с датчиками	Гироскопический датчик
14	Работа с датчиками	Датчик ультразвука
15	Работа с датчиками	Инфракрасный датчик
16	Работа с датчиками	Датчики Virnier
17	Работа с файлами	Совместная работа нескольких роботов
18	Подпрограммы	Создание подпрограмм
19	Конструирование	Основы конструирования 1
20-21	Соревнования “Сумо”	Создание и программирование робота сумоиста
24	Соревнование “Сумо”	Участие в внутришкольных соревнованиях
25-26	Соревнование “Движение по линии”	Создание и программирование робота, двигающегося по линии
27	Соревнование “Движение по линии”	Участие в внутришкольных соревнованиях
28	Конструирование	Основы конструирования 2
29	Проект “Green City”	Работа над проектом. Изучение проблем экологии больших городов. Изучение видов потребляемых энергий в городе.
30-34	Проект “Green City”	Прохождение миссий города
35	Проект “Green City”	Соревнования и показ проектов

3. Среда для разработки RobotC

При выборе среды для разработки, с использованием робота LEGO Mindstorms EV3, мною были рассмотрены и опробованы различные варианты сред и языков, которые способны оперировать с данным роботом, таких как lejOS, brickOS, nXc, однако они не показали стабильности при работе с роботом LEGO Mindstorms EV3 и их функционал оказался весьма скудным. Мой выбор остановился на RobotC, так как он поддерживает один из самых популярных языков программирования C/C++. Также, большим плюсом является возможность использования в данной среде виртуального робота, что позволяет тестировать робота и изучать материал без использования реальной модели робота.

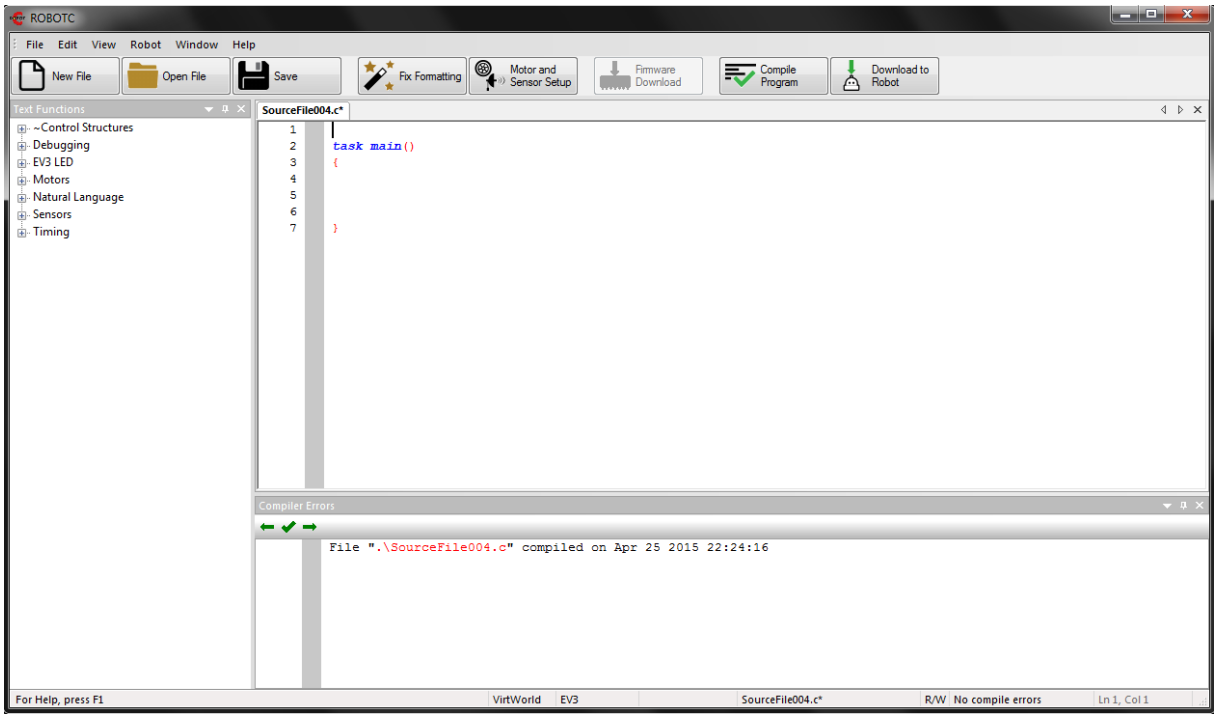

Рисунок 1. Среда разработки RobotC.

3.1 Выбор типа платформы

RobotC имеет поддержку как более новой версии роботов серии Lego Mindstorms – EV3, так и поддержку старого поколения роботов – NXT. Для того, чтобы выбрать тип платформы для работы, необходимо открыть меню “Robot”, перейти в меню “Platform Type” и выбрать LEGO Mindstorms EV3. [5]

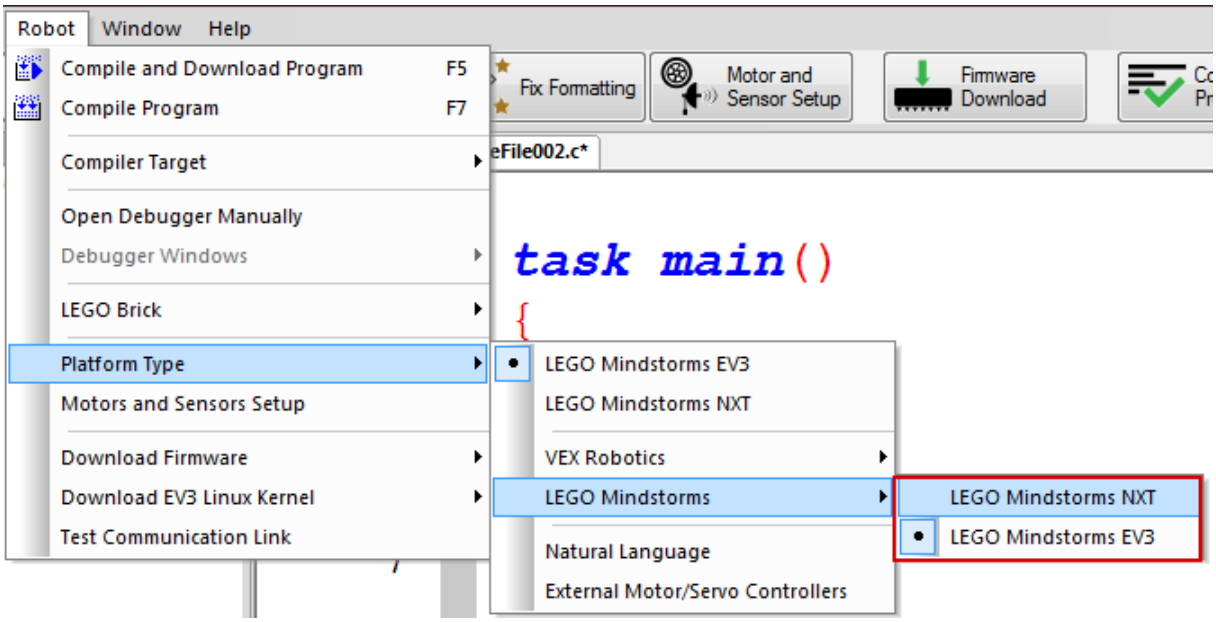

Рисунок 2. Выбор типа платформы.

3.2 Обновление ядра микрокомпьютера EV3

Прежде чем приступить к работе с роботом Lego Mindstorms EV3, необходимо загрузить на микрокомпьютер специальную прошивку от RobotC. Данная прошивка имеет совместимость с RobotC, LabVIEW и стандартной средой для разработки от LEGO. Загрузка прошивки занимает от 5-ти до 6-ти минут. [5]

Примечание: микрокомпьютер EV3 должен быть подключен к персональному компьютеру через USB порт.

1) В меню “Robot”, необходимо выбрать пункт Download EV3 Linux Kernel. Чтобы начать процесс обновления прошивки, необходимо выбрать пункт Standard File.

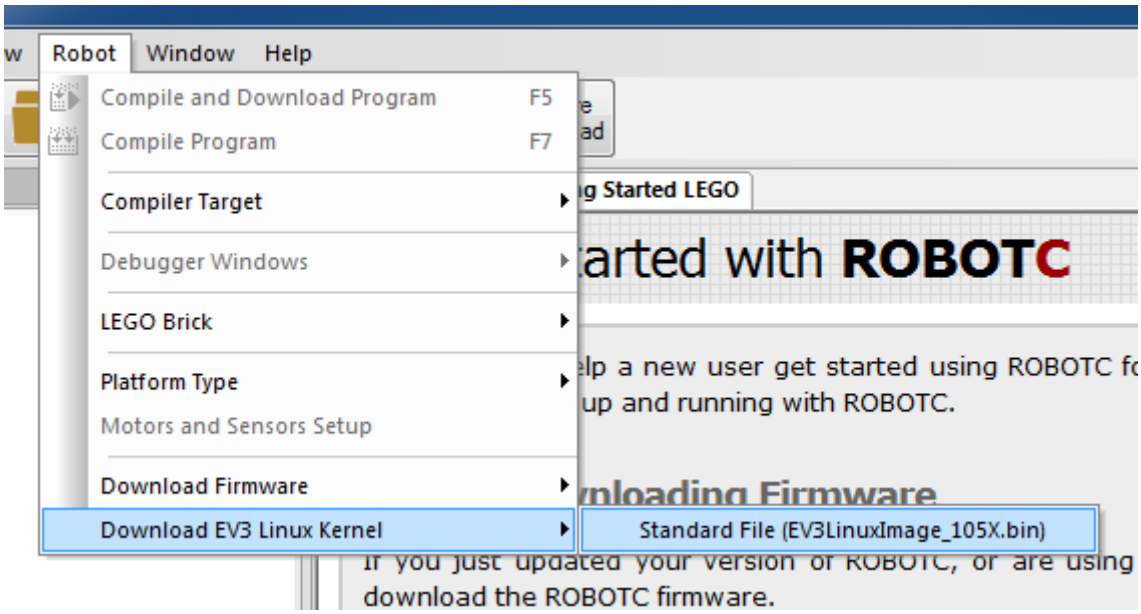

Рисунок 3. Выбор образа для обновления ядра микрокомпьютера EV3.

2) После завершения загрузки, если всё прошло успешно, появится следующее окно:

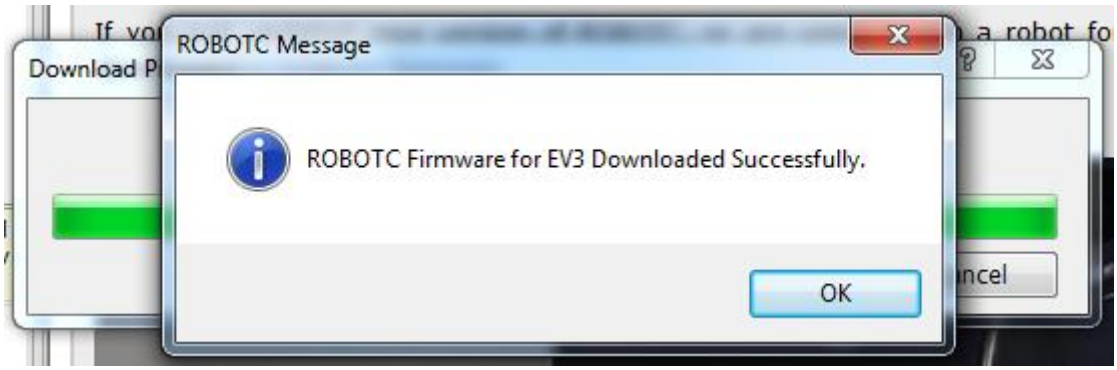

Рисунок 4. Окно успешного обновления ядра микрокомпьютера EV3.

3.2.1 Загрузка прошивки

После успешного обновления ядра микрокомпьютера EV3, необходимо установить ROBOTC Virtual Machine (VM), чтобы позволить программировать микрокомпьютер EV3 при помощи RobotC.

1) Чтобы установить ROBOTC VM, необходимо открыть меню “Robot” и выбрать Standard File. После этого, начнется процесс закачки прошивки.

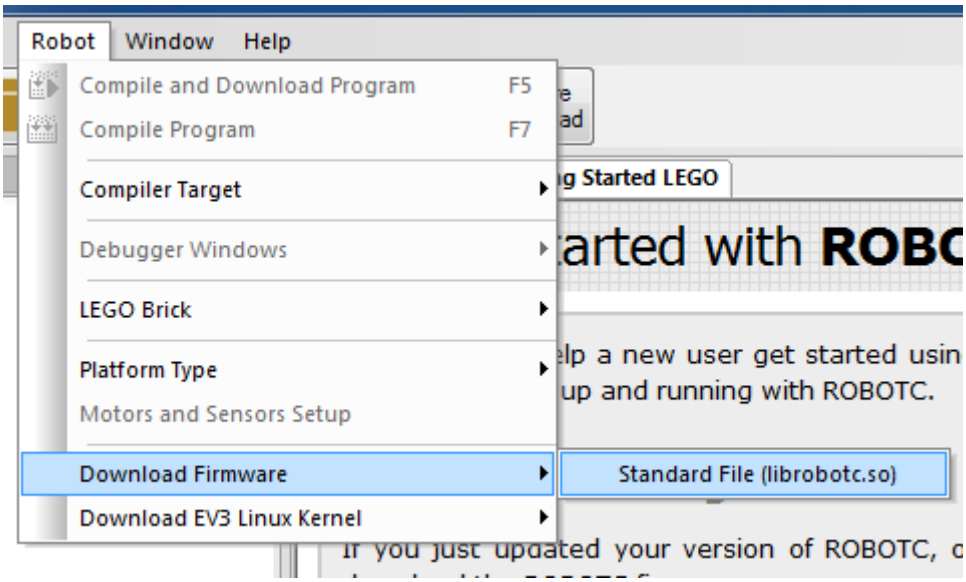

Рисунок 5. Выбор образа для обновления прошивки микрокомпьютера EV3.

2) В отличии от обновления ядра, установка ROBOTC VM занимает всего 5 секунд.

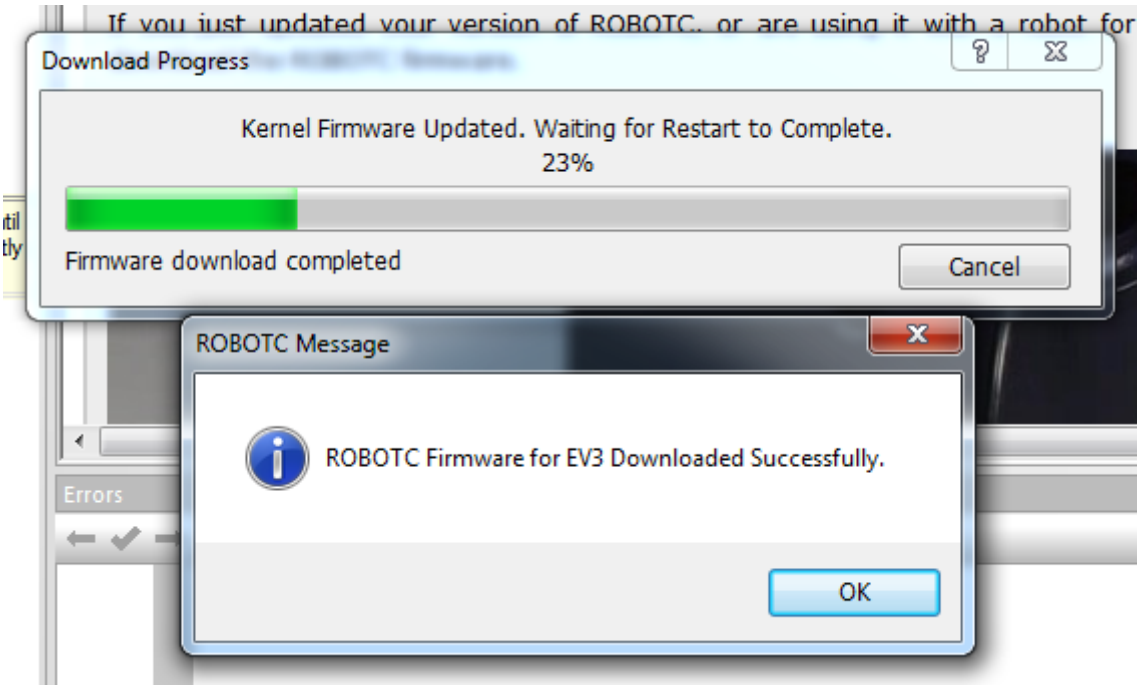

Рисунок 6. Окно успешного обновления прошивки микрокомпьютера EV3.

3.3 Спецификации микрокомпьютера EV3

Программируемый микрокомпьютер EV3 является сердцем и мозгом роботов, построенных на платформе LEGO® MINDSTORMS® Education EV3. Микрокомпьютер включает в себя шестикнопочный интерфейс управления с функцией изменения подсветки для индикации режима работы микрокомпьютера, монохромный дисплей с высоким разрешением, встроенный спикер, порт USB, слот для чтения карт памяти формата mini SD, 4 порта ввода и 4 порта вывода. Микрокомпьютер EV3 также поддерживает Bluetooth, WiFi (поддерживается WiFi адаптер NETGEAR WNA1100 Wireless-N 150) для связи с компьютерами имеет программный интерфейс, позволяющий создавать программы и настраивать регистрации данных непосредственно на микрокомпьютере EV3. Микрокомпьютер совместим с мобильными устройствами и питается батареями типа AA или аккумуляторной батареей EV3. [6]

Рисунок 7. Микрокомпьютер EV3.

Спецификации микрокомпьютера EV3:

• Процессор типа ARM 9 с Linux-образной операционной системой

• 4 порта ввода информации с частотой работы до 1 кГц

• 4 порта вывода для выполнения команд

• Встроенная память, включающая 16 МБ флеш-памяти и 64 МБ оперативной памяти

• Слот для чтения карт памяти формата Mini SDHC с поддержкой чтения карт объемом до 32 ГБ

• Шестикнопочный интерфейс управления с функцией изменения подсветки (3 цвета) для индикации режима работы микрокомпьютера

• Монохромный дисплей с разрешением 178 x 128 пикселей позволит осуществлять детальный просмотр графиков и чтение данных с датчиков

• Высококачественный встроенный динамик

• Возможность программирования и регистрации данных с помощью микрокомпьютера, созданные программы и полученные данные могут быть экспортированы в программное обеспечение EV3

• Поддержка связи с компьютерами через встроенный порт USB или подключаемые приемники WiFi или Bluetooth

• Режим USB 2.0 хостинга, позволяющий соединять микрокомпьютеры в последовательную цепь

• Поддержка WiFi и поддержка подключения USB флеш-карт

• Питание от 6 батарей типа AA или от аккумуляторной батареи постоянного тока EV3 емкостью 2050 мАч

4. Написание программы в среде RobotC

Программа в RobotC состоит из подпрограммы, которую называют task-ом. В каждой программе должна содержаться как минимум одна подпрограмма с названием main, именно это подпрограмма будет загружаться при работе с роботом. Каждая подпрограмма ограничена скобками. Каждая подпрограмма должна содержать действия и действие должно заканчиваться точкой с запятой.

4.1 Загрузка и запуск программы RobotC

Основные действия с роботом выполняются на клавиши:

F5 – Compile and Download Program, компилирует программу и загружает её в робота.

F7 – Compile Program, компиляция без загрузки программы в робота.

Во вкладке Robot -> Compiler Target необходимо выбрать конечную цель, куда будет загружаться программа, в физического робота или в виртуального.

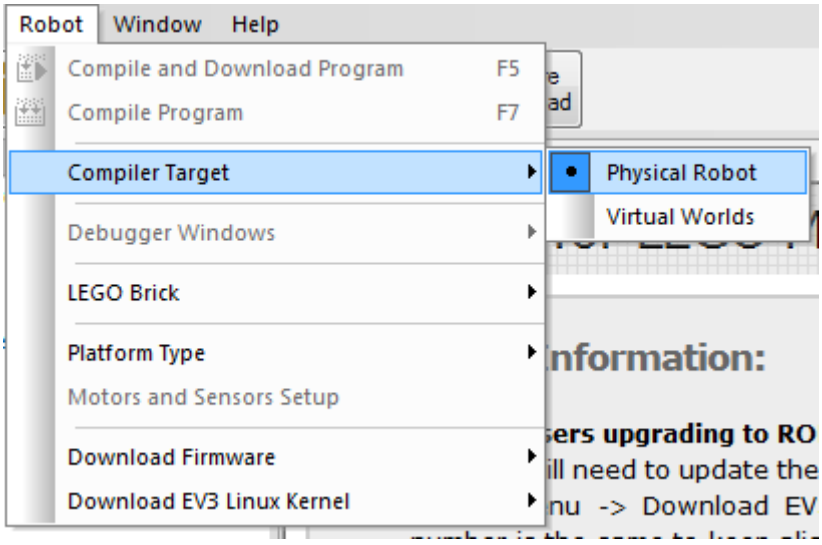

Рисунок 8. Выбор цели для загрузки программы.

4.2 Конфигурация моторов и сенсоров робота

Рисунок 9. Базовая модель робота Lego Mindstorms EV3.

Для работы с роботом Lego Mindstorms EV3 в среде RobotC необходимо указать конфигурацию портов робота. Так как в данной работе будет использоваться базовая модель робота, порты у робота будут распределены следующим образом:

Конфигурация портов для сенсоров:
S1 = Датчик касания
S2 = Гироскопический датчик
S3 = Датчик цвета
S4 = Ультразвуковой датчик

Конфигурация портов для моторов:

A = Средний мотор
B = Левый большой мотор
C = Правый большой мотор

Для использования данной конфигурации, в начале каждой программы необходимо прописывать сточку `#pragma config(StandardModel, "EV3_REMBOT")`. Во всех приведенных в этой работе программах эта сточка была удалена, однако для функционирования робота, должна быть прописана.

5. Программирование. Моторы

У робота EV3 есть 4 порта для моторов, которые маркированы буквами A, B, C, D.

Каждый мотор оснащен встроенным энкодером, который используется для определения позиции мотора. Мотор имеет 360 позиций (градусов) в одном обороте. То есть, чтобы мотору сделать пол оборота, необходимо задать 180 градусов.

5.1 Движение робота вперёд

В данной программе робот едет вперед на протяжении двух секунд при скорости 50, после этого, программа автоматически выключается.

```
task main()
{
      //Set the leftMotor (motor1) to half power (50)
    setMotorSpeed(leftMotor, 50);
      //Set the rightMotor (motor6) to half power (50)
    setMotorSpeed(rightMotor, 50);
      //Wait for 2 seconds before continuing on in the program.
    sleep(2000);
}
```

Первый параметр в команде `setMotorSpeed(leftMotor, 50);` указывает с каким мотором происходит действие, второй параметр указывает скорость мотора. Максимальная скорость робота 100, минимальная -100. В последнем случае робот будет ехать назад. Команда `sleep(2000);` говорит о том, что робот будет ехать 2 секунды в данном случае.

5.2 Движение робота назад

В данной программе робот едет назад на протяжении двух секунд при скорости -50, после этого, программа автоматически выключается.

```
task main()
{
    setMotorSpeed(leftMotor, -50);
    setMotorSpeed(rightMotor, -50);
    sleep(2000);
}
```

5.3 Реализация поворотов

В данной программе робот поворачивает налево, за счет движения правого колеса на скорости 100. После первого поворота, робот возвращается на место, с которого начал поворот, за счет движения в обратную сторону.

```
task main()
{
 //Moves right motor 1000 degrees (forward)
 //at full speed forward (+100)
 moveMotorTarget(rightMotor, 1000, 100);

 //Holds program flow until right motor comes to a complete stop.
 waitUntilMotorStop(rightMotor);

 //Moves right motor -1000 encoder counts (backwards)
 //at a speed level of 50 backwards (-50)
 moveMotorTarget(rightMotor, -1000, -50);

 //Holds program flow until right motor comes to a complete stop.
 waitUntilMotorStop(rightMotor);
}
```

Первый параметр в команде `moveMotorTarget(rightMotor, 1000, 100);` указывает с каким мотором происходит действие. Второй параметр указывает на количество градусов, на которое повернётся мотор. Третий параметр отвечает за скорость.

Команда `waitUntilMotorStop(rightMotor);` ожидает завершения работы, в данном случае, правого мотора и обеспечивает остановку робота.

5.4 Упражнения с использованием моторов

Задание 1

- Проехать вперёд на протяжении двух секунд со скоростью 80
- Остановиться
- Проехать назад со скоростью 50 с поворотом моторов на 600 градусов
- Остановиться

Задание 2

Проехать по траекториям:

- Круг
- Квадрат
- Треугольник
- Ромб

6. Программирование. Работа с данными

Компилятор RobotC поддерживает несколько различных типов данных. Некоторые из них являются стандартными для языка С, другие были специально созданы для RobotC. Перечень стандартных типов данных языка С вынесен на отдельный лист (см. Таблица 2. Перечень стандартных типов данных языка С).

7. Программирование. Цикл

Программы состоят из циклов, условий и команд. Цикл представляет собой повторяющееся действие, например, “каждый день я ложусь спать в 10 вечера”. Условия представляют собой предложения, которые содержат слова “если (if)…, тогда (then)…”, например, если сегодня рабочий день, тогда я встаю в 7 утра. Условия бывают и многоуровневые, их можно описать словами “если (if)…, тогда (then)…, иначе (else)…”, например, “если сегодня рабочий день, тогда я просыпаюсь в 7 утра, иначе я встаю в 9 утра”.

При программировании роботов, часто используют бесконечные циклы. При использовании таких циклов, робот будет выполнять действие до бесконечности, до тех пор, пока программа не будет выключена. [7]

7.1 WHILE цикл

Повторяющиеся действия можно реализовать при помощи while-цикла.

В этой программе робот будет ехать вперёд до тех пор, пока его программа не будет выключена вручную.

```
task main()
{
    while(true)
    {
      //Set the leftMotor (motor1) to half power (50)
    setMotorSpeed(leftMotor, 50);
      //Set the rightMotor (motor6) to half power (50)
    setMotorSpeed(rightMotor, 50);
}
}
```

Внутри while-цикла находится параметр true. Это условие, которое проверяет ход роботы цикла. Если параметр равен true, то цикл выполняется, если параметр равен false, то программа выходит из цикла. В программе, представленной выше, установлен параметр true, это значит, что робот будет выполнять программу, находящуюся в цикле до бесконечности.

Иногда необходимо написать такой цикл, который будет выполняться до определенных условий. Например, робот, который будет ехать вперёд до тех пор, пока не увидит перед собой препятствие на расстоянии 45 см.

```
//While the IR Sensor (plugged into port 4) does not return a value less
than 45
while(getIRDistance(S4) < 45)
{
    //Keep moving forward
    setMotorSpeed(motorC, 50);
    setMotorSpeed(motorB, 50);
}

//Stop the robot
setMotorSpeed(motorC, 0);
setMotorSpeed(motorB, 0);
```

В данной программе прописано условие `while(getIRDistance(S4) < 45)`, которое означает, что робот будет выполнять действие находящееся внутри цикла до тех пор, пока расстояние до препятствия не станет меньше 45 см. Если расстояние от робота до препятствия становится меньше 45 см, то робот выходит из цикла и, в данном случае, останавливается.

7.2 IF ELSE

Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. В случае, если условие не является верным, выполняется подпрограмма оператора else.
Пример программы со структурой if else:

```
task main()
{
    while (true)
    {
        // Turn the LED red if the touch sensor is pressed
        if (SensorValue[Touch])
        {
            setLEDColor(ledRed);
        }
        // If it's in a released state, turn the LED green
        else
        {
            setLEDColor(ledGreen);
        }
        //Loop to monitor value in Sensor debugger window
        sleep(50);
    }
}
```

В данной программе происходит проверка `if (SensorValue[Touch])` и если условие является верным, то подсветка кнопок на блоке загорается красным цветом. Если же условие не является верным, то подсветка кнопок на блоке загорается зелёным цветом.

7.3 DO WHILE цикл

Цикл do while отличается от цикла while тем, что в do while сначала выполняется тело цикла, а затем проверяется условие продолжения цикла. Из-за такой особенности do while называют циклом с постусловием. Таким образом, если условие do while заведомо ложное, то хотя бы один раз блок операторов в теле цикла do while выполнится. В итоге do while отличается от цикла while структурой. Если в while сначала выполняется проверка условия продолжения цикла, и если условие истинно, то только тогда выполняется тело цикла. Цикл do while работает с точностью да наоборот, сначала выполняется тело цикла, а потом проверяется условие, вот почему тело цикла do while, хотя бы раз, выполнится. [8]
Пример работы do while цикла:

```
task main()
{
// Create variable 'bursts' of type int and set it to '0'.
int bursts = 0;
do
 {
      //Set the leftMotor (motor1) to half power (50)
    setMotorSpeed(leftMotor, 50);
      //Set the rightMotor (motor6) to half power (50)
    setMotorSpeed(rightMotor, 50);
      // increment 'bursts' by 1
    bursts = bursts + 1;
      //Wait for 2 seconds before continuing on in the program.
  sleep(2000);  }
      //while 'bursts' is less than 3
 while(bursts < 3)
      //Stop leftMotor
    setMotorSpeed(leftMotor, 0);
      //Stop rightMotor
    setMotorSpeed(rightMotor, 0);
}
```

В начале данной программы создается переменная `int bursts = 0;`, как только переменная `bursts` станет больше 3, запустится цикл while и робот остановится.

7.4 FOR цикл

For цикл способен выполнять цикл определённое количество раз. В начале цикла, одной из переменных присваивается начальное значение, после этого начальное значение сравнивается с заданным числом, если сравнение удовлетворительно, то обычно к начальному числу добавляется единица и всё происходит заново. [5]

```
task main()
{
      /* initialize int 'i' to 0, and run the loop as long as 'i' is less
      than 3, incrementing 'i' by 1 after each iteration of the loop */
    for(int i = 0; i <= 3; i++)
    {
      //Set the leftMotor (motor1) to half power (50)
    setMotorSpeed(leftMotor, 50);
      //Set the rightMotor (motor6) to half power (50)
    setMotorSpeed(rightMotor, 50);
      //Wait for 2 seconds before continuing on in the program.
    sleep(2000);
      //Set the leftMotor (motor1) to zero power (0)
    setMotorSpeed(leftMotor, 0);
      //Set the rightMotor (motor6) to half power (50)
    setMotorSpeed(rightMotor, 50);
      //Wait for 750 ms before continuing on in the program.
    sleep(750);
}
}
```

В данной программе робот проедет вперёд на скорости 50 на протяжении двух секунд, после этого, робот развернется налево. Благодаря циклу `for(int i = 0; i <= 3;`

`i++)` он это выполнит эти действия суммарно 4 раза, в результате чего, получится квадрат.

7.5 SWITCH CASE

Оператор выбора switch является очень удобной заменой множественного использования операторов if. Оператор switch сравнивает значение одной переменной с несколькими константами. Основной формат для использования оператора множественного выбора switch case показан — ниже. Значение переменной, указанной в условии switch, сравнивается со значениями, которые следуют за ключевым словом case. Когда значение в переменной, соответствует значению в строке с оператором case, компьютер продолжит выполнение программы с этого места.

```
switch ( /*variable*/ ) {
case const1:
  /*Code that will be done, if variable equals const1*/
  break;
case const2:
  /*This code will be done, if variable equals const2*/
  break;
/*...*/
default:
  /*This code will be done, if none of constants will equal variable*/
  break;
}
```

Когда сравниваемое значение в переменной **variable** совпадет с первым значением оператора case, программа начнет выполнять код, который находится между текущим оператором case и оператором break. Оператор break используется для того, чтобы прерывать ход программы в операторе switch и передавать управление следующему оператору, после switch. Если не использовать оператор break, то, сразу после того, как выполнится один блок кода, программа переключится на выполнения следующего case, даже, если константное значение не будет равно значению в переменной variable. Поэтому, в операторе выбора switch, блоки кода после case всегда должны предваряться оператором break.

Также стоит обратить внимание на ключевое слово default, оно не является обязательным, но в то же время оно необходимо для обработки неожиданных ситуаций. Например, когда значение переменной не совпадает ни с одним из значений case, в таком случае выполнится код, который находится в ветке default. Это может быть полезно, в случае, если мы не ожидаем, что ни одно из значений case не совпало со значением переменной в условии switch. В таком случае, мы увидим, что сработал код в ветке default. [9]

8. Программирование. Вывод информации

RobotC имеет широкий ассортимент функций для вывода текста и фигур на LCD экране. Робот EV3 оснащён дисплеем с разрешением 100 на 64 пикселя.

- Нижний левый угол имеет координаты (0, 0), в то время как правый угол имеет координаты (177, 127).
- Дисплей имеет восемь линий для текса. 0 является самой верхней линией и 7 является нижней линией.

8.1 Вывод “Hello World!”

Данная программа выводит на экран робота традиционную фразу “Hello world”.

```
task main()
{
// infinite loop
while(true)
 {
      // create the string named s1, "Hello World"
   string s1 = "Hello World!";
      // display the string, 's1' on line 3
   displayTextLine(3, "%s", s1);
      // wait 50 milliseconds (helps refresh rate of LCD screen)
   wait1Msec(50);
}
}
```

Команда `displayTextLine(3, "%s", s1);` выводит на третью строку экрана EV3 фразу “Hello World!”. Благодаря циклу while(true), фраза будет выводиться до бесконечности.

8.2 Работа с подсветкой на блоке

Всего у блока EV3 есть 3 цвета подсветки: зелёный, красным и оранжевый. Данная программа поочередно меняет цвет подсветки на мигающий зелёный, пульсирующий красный и постоянный оранжевый.

```
task main()
{
//Infinite loop
    while(true)
    {
        //Flash the LED Green for two seconds
```

```
            setLEDColor(ledGreenFlash);
            sleep(2000);

            //Pulse the LED Red for two seconds
            setLEDColor(ledRedPulse);
            sleep(2000);

            //Turns the LED solid Orange for two seconds
            setLEDColor(ledOrange);
            sleep(2000);
        }
}
```

У команды `setLEDColor(ledGreenFlash);` есть только один параметр, который может принимать следующие значения:

- **LED_BLACK**: Выключает LED подсветку.
- **LED_GREEN**: LED подсветка загорается постоянным зелёным цветом.
- **LED_RED**: LED подсветка загорается постоянным красным цветом.
- **LED_ORANGE**: LED подсветка загорается постоянным оранжевым цветом.
- **LED_GREEN_FLASH**: LED подсветка загорается мигающим зелёным цветом.
- **LED_RED_FLASH**: LED подсветка загорается мигающим красным цветом.
- **LED_ORANGE_FLASH**: LED подсветка загорается мигающим оранжевым цветом.
- **LED_GREEN_PULSE**: LED подсветка загорается пульсирующим зелёным цветом.
- **LED_RED_PULSE**: LED подсветка загорается пульсирующим красным цветом.
- **LED_ORANGE_PULSE**: LED подсветка загорается пульсирующим оранжевым цветом.

8.2.1 Упражнения с использованием подсветки

Задание 1

Выполнить следующую задачу: Робот загорается красной пульсирующей подсветкой в течении трёх секунд, проезжает вперёд две секунды, останавливается и загорается последовательно: зелёной, оранжевой и красной подсветкой

8.3 Работа со звуком

RobotC предоставляет полный набор функций для работы со звуком. Прошивка от RobotC позволяет держать в очереди для воспроизведения до 10-ти звуковых файлов. Это позволяет роботу параллельно выводить звук и продолжать действия, без необходимости ожидания произведения звука.

Блок EV3 может проигрывать звуковые файлы в формате RSF. Среда для разработок RobotC имеет встроенную утилиту для закачки RSF файлов в блок EV3. [5]

8.3.1 Проигрывание частотного звука

```
task main()
{
    // First argument is the frequency or tone of the sound in Hertz (Hz)
    // The second is the duration in 10 ms, i.e.
    // 20 means 200 ms and 200 means 2000 ms or 2 seconds.
    playTone(400, 20);

    // Wait while the sound is playing in the background.
    // The bSoundActive variable will be "true" until the
    // EV3 is done playing the tone.
    while(bSoundActive)
        sleep(1);

    // Play a sound at 600 Hz for 200 ms.
    playTone(600, 20);
    while(bSoundActive)
        sleep(1);

    return;
}
```

Первый аргумент команды `playTone(400, 20)` является звуком определённой высоты в Герцах (Hz). Второй аргумент отвечает за продолжительность в 10 ms, например, 20 в данном случае будет означать 200 ms. Аргумент 200 будет означать 2000 ms, что равно двум секундам.

Команда `while(bSoundActive)` будет ждать завершения проигрывания звука, во время проигрывания звука, переменная `bSoundActive` имеет значение истина (“true”).

8.3.2 Проигрывание звукового файла

Примечание: *Полный перечень предустановленных звуков можно найти в директории установки RobotC, директорией по умолчанию является C:\Program Files (x86)\Robomatter Inc\ROBOTC Development Environment 4.X\EV3 System Files\Sounds*

Проигрывание звука из файла

```
setSoundVolume(75);

// Starts playing a soundfile, 'Bravo.rsf' on the EV3
playSoundFile("Bravo");

// Gives the file 2 seconds to play
sleep(2000);
```

Команда `setSoundVolume(75)` устанавливает громкость воспроизводимого блоком EV3 звука на 75 процентов от максимума. Команда `playSoundFile("Bravo")` проигрывает звук с названием "Bravo".

8.4 Упражнения по работе со звуком

Задание 1

Написать программу для робота, которая будет комментировать все действия робота (например, проигрывать файлы “speed up”, “speed down”, “stop”).

Задание 2

Написать программу для робота, которая будет последовательно менять подсветку кнопок робота и параллельно выводить любую ноту.

9. Работа с датчиками

Специально созданные для работы с микрокомпьютером EV3 датчики позволяют создавать сложных роботов, умеющих двигаться в разнообразных направлениях и по сложным траекториям, выполнять сложные действия, моделировать реальные технологии, используемые в технической аппаратуре и в производственных процессах. [10]

9.1 Датчик касания

Аналоговый датчик касаний EV3 простой, но высокоточный инструмент, который определяет нажата ли его кнопка или нет, а также способен определить количество нажатий, как одиночных, так и множественных. Ученики могут использовать его для построения систем контроля запуска/остановки или для создания роботов, способных выйти из лабиринта. Все это дает глубокое понимание технологии, применяющейся в цифровых музыкальных инструментах, компьютерных клавиатурах и в кухонных устройствах. [11]

9.1.1 Пример работы датчика касания

Данная программа меняет цвет подсветки в зависимости от того, нажат ли датчик касания. Если он нажат – подсветка имеет красный цвет, а на экране будет выведено “Pressed!”. Если датчик отпущен – то подсветка будет гореть зелёным цветом, а на экране будет выведено сообщение “Not Pressed!”.

```
task main()
{
    while (true)
    {
        // Turn the LED red if the touch sensor is pressed
        if (SensorValue[Touch])
        {
            displayCenteredBigTextLine(4, "Pressed!");
            setLEDColor(ledRed);
        }
        // If it's in a released state, turn the LED green
        else
        {
            displayCenteredBigTextLine(4, "Not Pressed!");
            setLEDColor(ledGreen);
        }
        // Loop to monitor value in Sensor debugger window
        sleep(50);
    }
}
```

9.1.2 Программа, подсчитывающая количество нажатий на кнопку датчика касания

```
task main()
{
    long bumpCount = 0;

    while (true)
    {
        bumpCount = getBumpedValue(touchSensor);

      writeDebugStreamLine("Bumps detected: %d", bumpCount);
        sleep(100);
    }
    return;
}
```

Команда `getBumpedValue(touchSensor)` считывает количество нажатий на кнопку датчика касания.

9.2 Упражнения по датчику касания

Задание 1

Написать программу, в которой при нажатой кнопке датчика касания, робот будет стоять, при отпущенной – будет ехать вперёд.

Задание 2

Написать программу, в которой при первом нажатии кнопки датчика касания робот начнет ехать вперёд, при повторном нажатии кнопки датчика касания – начнет ехать назад.

9.3 Датчик цвета

Цифровой датчик цвета EV3 различает восемь разных цветов. Он также используется в качестве светового датчика для измерения интенсивности свечения. Ученики могут создавать роботов, которые сортируют предметы по цвету, проводить опыты с отраженным света различных цветов. [12]
В RobotC, датчик цвета может измерять показания несколькими способами. Он может различать простые цвета (Красный, Синий, Черный, Белый, Оранжевый, Зелёный, Коричневый, а также отсутствие цвета), окружающее освещение (например, яркость освещения в комнате) и отраженный свет (используя встроенную в датчик ***LED*** подсветку).

Режимы работы датчика цвета в RobotC:

- **Reflected Mode** (Режим отраженного света)

Включает встроенную в датчик подсветку красного цвета и возвращает данные об интенсивности отражённого света в ресивер датчика. Данные варьируются от 0 до 100, где 0 означает, что никакой свет не отразился, а 100 означает, что свет полностью отразился.

- **Ambient Mode** (Режим окружающий среды)
  Выключает встроенную в датчик подсветку красного цвета и возвращает данные об интенсивности окружающего света в ресивер датчика. Данные варьируются от 0 до 100, где 0 означает, что никакой свет не отразился (полная темнота), а 100 означает, что свет полностью отразился (например, если датчик навести на солнце или лампу).

- **Color Mode** (Режим определения цвета)
  В данном режиме, датчик замеряет показания Красного, Синего и Зелёных цветов, а также уровень освещенности. На основе этих данных, датчик определяет цвет, который перед ним находится. Значения, которые может принять датчик цвета:
  - **colorNone**: Датчик цвета не обнаружил никаких объектов
  - **colorBlack**: Датчик цвета обнаружил объект черного цвета
  - **colorBlue**: Датчик цвета обнаружил объект синего цвета
  - **colorGreen**: Датчик цвета обнаружил объект зелёного цвета
  - **colorYellow**: Датчик цвета обнаружил объект желтого цвета
  - **colorRed**: Датчик цвета обнаружил объект красного цвета
  - **colorWhite**: Датчик цвета обнаружил объект белого цвета
  - **colorBrown**: Датчик цвета обнаружил объект коричневого цвета

9.3.1 Программа, различающая цвета

```
#pragma config(Sensor, S3,     Colour,         sensorEV3_Color,
modeEV3Color_Color)

/*
Colour sensor modes
0 - modeEV3Color_Reflected
1 - modeEV3Color_Ambient
2 - modeEV3Color_Color
3 - modeEV3Color_Reflected_Raw
4 - modeEV3Color_RGB_Raw
5 - modeEV3Color_Calibration - Not utilized
*/

task main()
{
    short currentColour;
    while (true)
    {
      // Colours range from 0 to 7
      // None    = 0
      // Black   = 1
      // Blue    = 2
      // Green   = 3
      // Yellow  = 4
      // Red     = 5
```

```
        // White   = 6
        // Brown   = 7
          currentColour = SensorValue[Colour];

          switch(currentColour)
          {
              case 0:  displayCenteredBigTextLine(4, "none");
              break;
              case 1:  displayCenteredBigTextLine(4, "black");
              break;
              case 2:  displayCenteredBigTextLine(4, "blue");
              break;
              case 3:  displayCenteredBigTextLine(4, "green");
              break;
              case 4:  displayCenteredBigTextLine(4, "yellow");
              break;
              case 5:  displayCenteredBigTextLine(4, "red");
              break;
              case 6:  displayCenteredBigTextLine(4, "white");
              break;
              case 7:  displayCenteredBigTextLine(4, "brown");
              break;
              default: displayCenteredBigTextLine(4, "unknown");
          }

          // Wait 20 ms to get 50 readings per second
          sleep(20);
    }
}
```

Команда `SensorValue[Colour]` считывает значения датчика цвета, где каждый цвет имеет значение в виде порядкового номера. В соответствии с номером, структура `switch(currentColour)` выводит по центру экрана название цвета объекта, который находится перед датчиком цвета.

9.4 Упражнения по датчику цвета

Задание 1

Написать программу для робота, в которой робот будет ехать вперёд до тех пор, пока не увидит перед собой черную линию.

Задание 2

Написать программу для робота, который будет комментировать при помощи звуков цвета всех объектов перед собой.

9.5 Гироскопический датчик

Цифровой гироскопический датчик EV3 позволяет измерять движение вращения робота, а также улавливать изменения в его движении и положении. С помощью этого датчика легко можно измерить углы, создать балансирующего робота и исследовать технологии, которые используются в настоящих навигационных системах и игровых контроллерах. [13]
Задачей гироскопического датчика является измерение угла поворота робота. Гироскопический датчик постоянно отслеживает угловую позицию робота и его направление, однако может отслеживать вращения только в одном направлении. Это направление отмечено стрелкой на корпусе датчика. (см. Рисунок 10. Направления вращения гироскопического датчика Lego Mindstorms EV3).

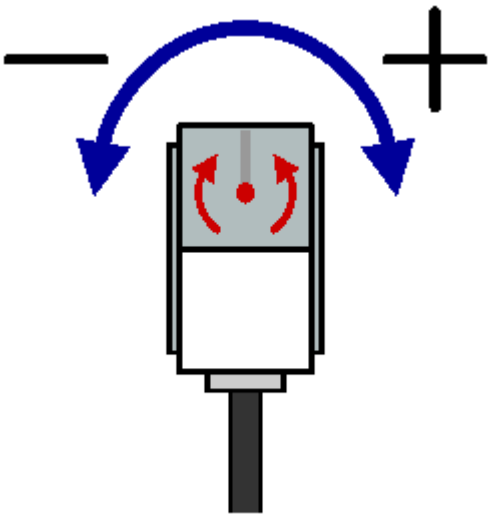

Рисунок 10. Направления вращения гироскопического датчика Lego Mindstorms EV3.

В RobotC, при повороте гироскопического датчика против часовой стрелки, увеличивается значение датчика, которое измеряется в градусах. Повороты по часовой стрелки наоборот, уменьшат значение датчика. Чтобы сбросить текущее значение гироскопического датчика до нуля, необходимо использовать команду resetGyro. Точность измерения датчика равна +/-3°.

Программа, устанавливающая зависимость между скоростью мотора и углом поворота

```
#pragma config(Sensor, S4,     Gyro,           sensorEV3_Gyro,
modeEV3Gyro_Rate)
#pragma config(Motor,  motorB,          motorLeft,     tmotorEV3_Large,
PIDControl, encoder)

/*
Gyro modes
0 - modeEV3Gyro_Angle
1 - modeEV3Gyro_Rate
2 - modeEV3Gyro_Fast
3 - modeEV3Gyro_RateAndAngle
4 - modeEV3Gyro_Calibration - Not utilized
*/
```

```
task main()
{
 int gyroRate = 0;

 // The code below will turn the motor proportionally to the gyro rate
 while (true)
 {
  // Read the current rate value in degrees/second
  gyroRate = SensorValue[Gyro];

  displayCenteredBigTextLine(4, "Gyro: %d", gyroRate);
  // Clip the value of the gyro to -100 to +100 (min and max motor speeds)
  if (gyroRate > 100)
   gyroRate = 100;
  else if (gyroRate < -100)
   gyroRate = -100;

  // Set motor speeds to the rate of the gyro
  motor[motorLeft] = gyroRate;
  sleep(50);
 }
}
```

Команда `gyroRate = SensorValue[Gyro];` считывает показания с гироскопического датчика, команда `motor[motorLeft] = gyroRate;` устанавливает скорость левого мотора равной показаниям гироскопического датчика. Структура `if else` не позволяет превысить показаниям датчика диапазон -100 до +100.

9.6 Упражнения по гироскопическому датчику

Задание 1

Реализовать программу, в которой робот будет точно поворачиваться на следующие градусы:

- 45°
- 90°
- 180°
- -45°

9.7 Ультразвуковой датчик

Цифровой ультразвуковой датчик EV3 генерирует звуковые волны и фиксирующий их отражения от объектов, тем самым измеряя расстояние до объектов. Он также может использоваться в режиме сонара, испуская одиночные волны. Кроме того, датчик может улавливать звуковые волны, которые будут являться триггерами для запуска программ. К примеру, ученики могут использовать датчик для построения системы мониторинга трафика, измерения расстояния между автомобилями. Благодаря этому датчику принципы работы ультразвуковой технологии и способы ее применения в автоматических дверях, машинах и заводских системах становятся понятны и захватывающи. [14]

9.7.1 Программа, которая держит дистанцию до объекта впереди

```
#pragma config(Sensor, S4,     sonar4,          sensorEV3 Ultrasonic)
#pragma config(Motor,  motorB,          motorLeft,     tmotorEV3_Large,
PIDControl, encoder)
#pragma config(Motor,  motorC,          motorRight,    tmotorEV3_Large,
PIDControl, encoder)

task main()
{
    // Distance to maintain to the target (30 cm)
    const int distanceToMaintain = 30;

    int currentDistance = 0;

    while(true)
    {
        // Read the sensor
        currentDistance = SensorValue[sonar4];
        displayCenteredBigTextLine(4, "Dist: %3d cm", currentDistance);

        // We're too far away, move forward
        if ((distanceToMaintain - currentDistance) < -2)
        {
            motor[motorLeft] = 30;
            motor[motorRight] = 30;
        }
        // We're too close, move backwards
        else if ((distanceToMaintain - currentDistance) > 2)
        {
            motor[motorLeft] = -30;
            motor[motorRight] = -30;
        }
        // We're good, don't go anywhere
        else
        {
            motor[motorLeft] = 0;
            motor[motorRight] = 0;
        }

        //Loop to monitor value in Sensor debugger window
        sleep(50);
```

```
    }
}
```

В данной программе, команда `SensorValue[sonar4];` измеряет показания ультразвукового датчика в сантиметрах, с точностью до десятых. Максимальное расстояние до объекта, которое датчик может измерить, равно 250 см. Команда `displayCenteredBigTextLine(4, "Dist: %3d cm", currentDistance);` выводит на экран текущее расстояние от ультразвукового датчика до ближайшего объекта.

9.8 Упражнения по ультразвуковому датчику

Задание 1

Написать программу для робота, в соответствии с которой робот будет ехать вперёд пока не обнаружит перед собой на расстоянии 15 см объект. После этого он должен остановиться и издать звук “Stop”.

9.9 Инфракрасный датчик

Цифровой ИК-датчик EV3 способен определять приближение робота. Он также способен улавливать ИК-сигналы, излучаемые ИК-маяком, позволяя создавать дистанционно управляемых роботов, навигационные системы для преодоления препятствий. [15]
ИК-датчик может работать с четырьмя различными каналами, каждый из которых имеет свою частоту.
ИК-датчик может работать в трёх различных режимах, значения, которые получит датчик, будут изменяться в соответствии с выбранным режимом:

- **Proximity Mode** (Режим определения расстояния)
  В этом режиме используется встроенный в датчик LED, чтобы определить расстояние до объекта. Диапазон значений варьируется от 0 до 100. Полученное значение зависит от того, сколько света от LED датчика отразилось назад, в ресиверы ИК-датчика. Как правило, чем ближе объект, тем ниже возвращаемое значение.
- **Seeker Mode** (Режим поиска)
  В этом режиме используется встроенный в датчик LED, чтобы определить расстояние до объекта. Диапазон значений варьируется от 0 до 100. Полученное значение зависит от того, сколько света от LED датчика отразилось назад, в ресиверы ИК-датчика. Как правило, чем ближе объект, тем ниже возвращаемое значение. Если вернулось значение 100, то это означает, что никаких объектов обнаружено не было.
- **Remote Mode** (Режим дистанционного управления)
  Настраивает ИК-датчик на работу с ИК-пультом. ИК-пульт и ИК-датчик должны быть на линий видимости для корректной работы.

9.9.1 Программа, определяющая расстояние до объекта

```
//While the IR Sensor (plugged into port 4) does not
//return a value less than 45
while(getIRDistance(S4) < 45)
{
    //Keep moving forward
    setMotorSpeed(motorC, 50);
    setMotorSpeed(motorB, 50);
}

//Stop the robot
setMotorSpeed(motorC, 0);
setMotorSpeed(motorB, 0);
```

9.9.2 Управление роботом при помощи ИК-пульта.

```
//Infinite loop
while(true)
{
      //If the IR sensor (port 4) sees a beacon
      //to the right (positive)
    if(getIRBeaconDirection(S4) > 5)
    {
        //Turn Right
        setMotorSpeed(motorC, 50);
        setMotorSpeed(motorB, -50);
    }
      //If the IR sensor (port 4) sees a beacon
      //to the left (negative)
    else if(getIRBeaconDirection(S4) < -5)
    {
      //Turn Left
        setMotorSpeed(motorC, -50);
        setMotorSpeed(motorB, 50);
    }
      //Otherwise, move forward
    else
    {
      //Move Forward
        setMotorSpeed(motorC, 50);
        setMotorSpeed(motorB, 50);
    }
}
```

Команда `getIRBeaconDirection(S4)` считывает значения ИК-датчика, который получает значения от ИК-пульта.

9.10 Упражнения по инфракрасному датчику

Задание 1

Написать программу, которая позволит управлять роботом при помощи ИК-пульта. Каждое своё действие робот должен комментировать при помощи звуков. Робот должен уметь ездить в 4-ех направлениях, издавать соответствующий звук и менять подсветку кнопок в зависимости от направления:

- Робот едет вперед – подсветка кнопок горит пульсирующим зелёным цветом.
- Робот едет назад – подсветка кнопок горит зелёным цветом.
- Робот едет налево – подсветка кнопок горит красным цветом.
- Робот едет направо – подсветка кнопок горит оранжевым цветом.

Заключение

В ходе работы все поставленные цели были достигнуты, был составлен курс, который даёт базовое понимание робототехники и программирования. Были предоставлены задания для самостоятельной работы. При необходимости, последовательность изучения материала в данном курсе может быть изменена. В будущем, планируется более детально расписать работу всех датчиков и расширить ассортимент команд для каждого датчика. Также планируется написание второй части курса, посвященной внеклассной работе, изучению подпрограмм-функций и подготовке к соревнованиям.

Разработка данного курса была крайне полезна для автора, поскольку при написании работы были задействованы знания, полученные не только в процессе обучения в университете, но также знания и опыт, полученные непосредственно при преподавании робототехники.

Kokkuvõte

Töö käigus olid kõik püstitatud eesmärgid saavutatud, oli koostatud kursus, mis annab alused programmeerimise ja robootika mõistmises. Olid antud ülesanded iseseisvaks tööks. Vajadusel on võimalik muuta õpitava materjali järjestust. Tulevikus on plaanis täpsemalt kirjeldada kõikide andurite töö ja laiendada käsude valikut iga sensoriga. Plaanis on ka kirjutada kursuse teine osa, mille eesmärgiks oleks kirjeldada koolivälist tegevust, alamprogrammide uurimist ja ettevalmistust võistlusteks.

Kursuse arendamine oli autori jaoks äärmuslikult kasulik, sest töö käigus sai rakendatud ülikoolis varem õpitud teadmisi ning ka teadmisi ja kogemusi mis autor sai robootika õpetamisel.

Summary

During this work, the author's goals were successfully achieved. A course was created, that gives a basic understanding of robotics and programming. Tasks for independent work were given. If necessary, the order of learning the material in this course can be changed. In the future, it is planned to describe in details the operation of all sensors, and expand the range of commands for each sensor. Also, it is planned to write second part of course, which is devoted to extracurricular activities, studying of subprograms-functions and preparation for competitions.

The development of this course was extremely useful for the author, since the writing of this work involved knowledge gained not only in the learning process at the university, but also the knowledge and experience gained in teaching robotics.

Список используемой литературы

[1] Wikipedia, «Disk image,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Disk_image. [Дата обращения: 18 05 2015].

[2] Wikipedia, «Firmware,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Firmware. [Дата обращения: 18 05 2015].

[3] Wikipedia, «USB,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/USB. [Дата обращения: 18 05 2015].

[4] «RoboCraft - сообщество любителей робототехники, электроники и программирования,» [В Интернете]. Available: http://robocraft.ru/blog/technology/734.html. [Дата обращения: 18 05 2015].

[5] Wikipedia, «LEGO Education (Dacta),» [В Интернете]. Available: https://ru.wikipedia.org/wiki/LEGO_Education_(Dacta). [Дата обращения: 19 05 2015].

[6] «ROBOTC for LEGO Mindstorms 4.0 - Users Manual,» [В Интернете]. Available: http://help.robotc.net/WebHelpMindstorms/index.htm. [Дата обращения: 18 05 2015].

[7] LEGO.com, «Микрокомпьютер EV3,» [В Интернете]. Available: https://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45500-intelligent-brick. [Дата обращения: 18 05 2015].

[8] L. Sepp, Robootika?... See on imelihtne!, MTÜ Robootika.COM, 11.02.2013.

[9] «Основы программирования на языках Си и C++ для начинающих. Цикл do while в C++,» [В Интернете]. Available: http://cppstudio.com/post/361/. [Дата обращения: 18 05 2015].

[10] «Основы программирования на языках Си и C++ для начинающих. Оператор выбора switch в C(Си),» [В Интернете]. Available: http://cppstudio.com/post/6691/. [Дата обращения: 18 05 2015].

[11] «Институт новых технологий. Датчики к микрокомпьютеру EV3,» [В Интернете]. Available: http://www.int-edu.ru/object.php?m1=445&m2=2&id=1560. [Дата обращения: 18 05 2015].

[12] LEGO.com, «Датчик касания EV3,» [В Интернете]. Available: https://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45507-touch-sensor. [Дата обращения: 18 05 2015].

[13] LEGO.com, «Датчик цвета EV3,» [В Интернете]. Available: http://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45506-color-sensor. [Дата обращения: 18 05 2015].

[14] LEGO.com, «Гироскопический датчик EV3,» [В Интернете]. Available: https://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45505-gyro-sensor. [Дата обращения: 18 05 2015].

[15] LEGO.com, «Ультразвуковой датчик EV3,» [В Интернете]. Available: http://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45504-ultrasonic-sensor. [Дата обращения: 18 05 2015].

[16] LEGO.com, «ИК-датчик EV3,» [В Интернете]. Available: https://education.lego.com/ru-ru/lego-education-product-database/mindstorms-ev3/45509-ir-sensor. [Дата обращения: 18 05 2015].

Дополнение 1

Таблица 2. Перечень стандартных типов данных языка С.

Тип данных	**Пояснение**	**Диапазон значений**	**Количество занимаемой памяти**
Int	Целые числа, может иметь положительные значения, отрицательные и 0.	-32768 до +32767	16 bits/2bytes
Long	Целые числа, может иметь положительные значения, отрицательные и 0. Диапазон значений шире, однако занимает больше места в памяти.	-2147483648 до +2147483647	32 bits/4bytes
Float	Целые числа с плавающей запятой, может иметь положительные значения, отрицательные и 0.	-3.4 × 1038 до +3.4 × 1038	32 bits/4bytes
Byte	Целые числа, может иметь положительные значения, отрицательные и 0. Диапазон значений ниже чем у Int, однако занимает меньше места в памяти.	-128 до +127	8 bits/1 byte
Ubyte	Целые числа, может иметь положительные значения и 0.	0 до +255	8 bits/1 byte
Bool	Может принимать значения True (1) и False (0).	0 или 1	4 bits/0.5 bytes

Char	Простой тип данных, предназначенный для хранения одного символа (буква, число или знак пунктуации) в ASCII кодировке.	Вся таблица ASCII	8 bits/1 byte
String	Несколько символов, соединённых воедино (например, слово или предложение).	До 20-ти символов	160 bits/20 bytes