МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЁЖНОЙ ПОЛИТИКИ КРАСНОДАРСКОГО КРАЯ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
КРАСНОДАРСКОГО КРАЯ
«НОВОРОССИЙСКИЙ КОЛЛЕДЖ РАДИОЭЛЕКТРОННОГО ПРИБОРОСТРОЕНИЯ»
(ГБПОУ КК НКРП)

Учебное пособие по основам робототехники на базе конструкторов LEGO MINDSTORMS

Новороссийск, 2019

Согласовано
Советом по методическим вопросам
протокол от 30.01 2019 г. № 5
Председатель
_____ Е.В. Заслонова

Утверждаю
Зам. директора по УР
_____ Т.В. Трусова
30 01 2019 г.

Рассмотрено
УМО общепрофессиональных и специальных дисциплин специальностей 11.02.02, 11.02.06, 11.02.10
Протокол от 09.01 2019 г. № 5
Председатель УМО
_____ А.А. Шмидберская

Организация-разработчик: ГБПОУ КК «Новороссийский колледж радиоэлектронного приборостроения» (далее ГБПОУ КК НКРП)

Разработчик:
преподаватель ГБПОУ КК НКРП
(должность, место работы)
_____ Горшков В.В. (ФИО)
(подпись)

Рецензенты:
В.И.Скрипников, _____ Индивидуальный предприниматель, сервисный центр Panasonic
(подпись)
(должность, место работы)
Квалификация по диплому:
Инженер

А.В. Борисов, _____ Преподаватель ГБПОУ КК НКРП
(подпись)
(должность, место работы)

Квалификация по диплому:
Инженер

РЕЦЕНЗИЯ

на учебное пособие по основам робототехники на базе конструкторов Lego Mindstorms

Учебное пособие подготовлено преподавателем ГБПОУ КК НКРП Горшковым В.В.

Платформа Lego является безусловным лидером образовательной робототехники. Наборами Lego Mindstorms оснащены кружки робототехники во многих странах мира. На лидирующих позициях Lego Mindstorms и в российских секциях робототехники. Из этого конструктора можно построить не только игрушечных роботов, но и прототипы таких серьезных конструкций как, например, принтер Брайля, космическая станция, космический самолет, солнечные панели и т.п. Причем придумывать и реализовывать все это могут дети.

Учебное пособие по основам робототехники на базе конструкторов Lego Mindstorms состоит из следующих разделов:

- раздел «Описание конструкторов компании LEGO»;
- раздел «Язык программирования RobotC»;
- раздел «Комплекс программ по основам робототехники на базе конструкторов LEGO MINDSTORMS » .

В учебном пособии было проведено описание конструкторов компании «Lego», таких как: Lego Mindstorms NXT, Lego Mindstorms EV3 и TETRIX, так как конструкторы этой компании применяются во многих робототехнических соревнованиях.

Во втором разделе проведено описание среды программирования RobotC. RobotC предназначен как для на новичков, так и для подготовленных программистов и имеет два режима работы – базовый и расширенный. В соответствии с этим среда программирования поддерживает два языка: собственно сам RobotC, являющийся особым диалектом C, и «естественный язык» Natural Language, позиционирующийся как переходный от графических форм (наподобие NXT-G) к текстовым блокам и использующий вместо низкоуровневых команд опроса датчиков и управления моторами процедуры с заранее определенными для робота действиями.

В третьем разделе описаны программы для управления мобильным роботом, движения робота по линии с помощью датчиков света и цвета и ориентации робота в пространстве. при помощи данных программ роботы могут не только перемещаться самостоятельно, но и переносить грузы, играть на музыкальных инструментах, подниматься по лестницам и принимать участие в спасении людей при чрезвычайных ситуациях.

Сейчас робототехника постепенно становится тем общим двигателем, который объединяет электротехнику, электронику, оптику, механику. Развитие данной науки дает возможность решать разного рода социальные проблемы, в частности, совершать уход за престарелыми людьми, снизить человеческие потери в военных конфликтах, ограничить миграцию низкоквалифицированной рабочей силы.

Рецензент:

Индивидуальный предприниматель,
сервисный центр Panasonic

_____ В.И.Скрипников
(подпись)

_____ 2019 г.

Рецензия

на учебное пособие по основам робототехники на базе конструкторов Lego Mindstorms

Наука не стоит на месте. Уже сейчас уровень развития робототехники достиг больших высот. Писатели-фантасты неоднократно пугали мир разнообразными вариациями на тему «бунта машин». Но ситуация с развитием робототехники в настоящее время складывается таким образом, что остановить это развитие в данной сфере невозможно. А все потому, что роботы уже заняли свою нишу в жизни общества. Они стали частью современной промышленной революции, характеризующейся широким внедрением адаптивных технологий и роботизацией производства. Ежегодно все больше предприятий автоматизируется, поэтому на данный момент завод, на котором работает всего несколько десятков человек, а всю основную работу выполняют роботы, уже никого не удивляет

Структура учебного пособия по основам робототехники состоит из описания конструкторов компании «LEGO», описания языка программирования RobotC и комплекса программ по основам робототехники.

Lego-робототехника подходит детям всех возрастов. Выпускаются наборы, ориентированные на разный уровень подготовки и возраст. К достоинствам относится:

- Безопасность: в процессе производства применяется безопасный сертифицированный и крепкий пластик. Его практически невозможно случайно сломать.
- Широкие возможности: наборы совместимы с платформами некоторых других фирм.
- Разнообразие серий: ассортимент, выпускаемый под этим брендом, представлен различными видами. Поэтому без проблем можно создать робота, который будет интересен ребенку.
- Нельзя забывать и о том, что lego-робототехника развивает мелкую моторику рук, активизирует желание узнавать мир, изучать точные дисциплины, заниматься самосовершенствованием навыков. В процессе образовательной деятельности дети становятся усидчивыми, могут концентрировать внимание, происходит развитие пространственного мышления.

Конкурсы мирового уровня — такие как WRO и FIRST, а также множество региональных мероприятий и фестивалей содержат в своих регламентах требование использовать Lego.

С помощью разработанного комплекса программ по основам робототехники обучающийся будет уметь устанавливать дополнительное навесное оборудование на базу мобильного робота, не снижая возможностей маневрирования робота, определять соответствующее аппаратное обеспечение (двигатели, датчики), необходимое для выполнения требований технического задания к функциональности своего дополнительного навесного оборудования, разрабатывать алгоритм выполнения конкурсных заданий для мобильного робота, включая: приемы ориентации и навигации, используя предложенные привода и датчики, загружать, устанавливать и производить все требуемые физические и программные настройки, необходимые для эффективного использования всего оборудования, поставляемого производителем мобильного робота и работать в команде.

Рецензент:

преподаватель

ГБПОУ КК НКРП

_____ А.В. Борисов

(подпись)

03 01 2019 г

ОГЛАВЛЕНИЕ

Введение

Развитие отечественной робототехники переживает большие трудности, отрасль фактически поставлена на грань выживания. Есть несколько причин такого положения. Во-первых, вся научно-техническая и исследовательская база робототехники долгие годы поддерживалась и финансировалась государством, основные приоритеты развития данной отрасли определялись на правительственном уровне. Во-вторых, развитие робототехники в Советском союзе стимулировалось и экономическим соперничеством между странами социалистического и капиталистического лагеря.

Развитие робототехники в России зависит от многих причин. Робототехника - важная и перспективная отрасль промышленности, поскольку при помощи роботов и их комплексов руководители предприятий могут создавать высокоэффективное производство с минимальными издержками и высоким качеством продукции. Для достижения этой задачи они готовы привлекать инвестиционный капитал и вкладывать в их развитие собственные средства предприятия с целью значительно увеличить чистую прибыль от продаж продукции в будущем. Таким образом, для многих развитых предприятий подобный подход стал основой стратегии работы на долгие годы.

Мировые лидеры в производстве робототехники уже конкурируют между собой на российском рынке, использую знания и опыт отечественных инженеров по робототехнике. Положение дел необходимо срочно менять: развивать отечественную робототехнику при помощи национальных проектов, которые должны приниматься и курироваться на федеральном уровне.

Вне сомнений, робототехника представляет собой естественное логическое продолжение техники как явления. Стремление автоматизировать

любой труд постепенно вытесняет человека из многих сфер его деятельности, предоставляя взамен все новые возможности для приложения усилий: просмотр кинофильмов, подводные погружения, компьютерные игры и т.д. Часть всеобщего труда, затрачиваемая человечеством на производство средств производства, а не конечного продукта потребления, постепенно увеличивается от 0%, очевидно стремясь к 100%. Уже сейчас усилия большинства наилучших современных роботов направлены на производство других машин: станков, автомобилей, компьютеров и т.д.

Будучи одной из самых интригующих вечных тем, робототехника с самых ранних времен привлекает к себе интерес философов и писателей. Прогресс в философском осмыслении вопросов создания искусственных думающих машин на текущий момент далеко опережает практические результаты в этой области.

В робототехнике наиболее часто используются следующие термины и определения: механизм, машина, робот, андроид.

Механизм – это непосредственное использование материалов для обеспечения некоторой механической функции; при этом все основано на взаимном сцеплении и сопротивлении тел.

Машина – это совокупность механизмов, заменяющих человека или животное в определенной области; преобразует энергию из одного вида в другие.

Робот – понятие неопределенное, к которому можно отнести любой вид машины; термин обычно используется для художественного эффекта или означает, что в машине используются манипуляторные механизмы, позволяющие машине манипулировать предметами. Важным свойством роботов является определенная степень автономности.

Наконец, Андроид - это робот-гуманоид, т.е. антропоморфная, имитирующая человека машина, стремящаяся заменить человека в любой его деятельности. Андроид обязан выглядеть и вести себя как человек.

Отдельную нишу занимают кибернетические организмы – живые системы, содержащие в себе искусственные компоненты для расширения своих возможностей.

Очевидно, не существует никакой реальной возможности затормозить развитие современной техники на пути к построению киборгов, андроидов и, в конечном итоге, искусственного разума. На фоне этого все чаще обсуждается возможность потери человечеством контроля над собственными созданиями.

Историю робототехники как прикладной науки о разработке и производстве автоматизированных технических систем можно условно разделить на две части :популярную и актуальную.

Популярная история робототехники ведет свое повествование от мифа о железных слугах Гефеста, «Франкенштейна» Мери Келли, через удивительные часовые механизмы в виде поющих бронзовых фазанов и целых движущихся городов к роботам на Марсе и гуманоидному роботу Asimo корпорации Honda, т.е. показывает развитие мечты о роботах.

Актуальная история робототехники включает в себя историю развития только тех идей и технологий, которые оказали наибольшее влияние на конструирование современных роботов, таких как сварочные линии автомобильных кузовов или автономные межпланетные исследовательские станции.

Основная часть

1 Описание конструкторов компании «LEGO»

Компания Lego стала производить свои конструкторы с 1940 года, начав с обычных деталей: балок, шестеренок и прочего, и продолжив высокотехнологичными устройствами в виде моторчиков, датчиков, пневматики и т.п. И хотя с годами меняется технология сборки, Lego старается поддерживать обратную совместимость с прошлыми версиями конструкторов.

Впервые коммерческий набор робототехники Mindstorms Lego выпустила осенью 1998 года. Всего существует три поколения роботов Lego Mindstorms RCX 1.0 (1998), NXT 2.0 (2006) и EV3 (2013 год).

Первый набор Mindstorms Robotics Invention System состоял из двух моторов, двух датчиков касания и одного датчика света. Версия NXT содержала уже три серводвигателя и один световой, один звуковой и один сенсорный датчик, а также дальномер. NXT 2.0 - два датчика касания, датчик цвета и ультразвуковой дальномер, а также возможность поддерживать до 4-х датчиков без использования мультиплексора.

Сейчас Lego Mindstorms может быть использован для моделирования встраиваемых систем с компьютерным управлением электромеханических частей. Существуют различные реальные встраиваемые системы (начиная от контроллера лифта и заканчивая промышленными роботами) которые могут быть смоделированы с использованием Mindstorms.

1.1 Описание конструктора Lego mindstorms nxt

NXT является интеллектуальным, управляемым компьютером роботом на базе элементов Lego и системы mindstorms.

В конструкторе NXT применены новейшие технологии робототехники: новейший 32-битный программируемый микроконтроллер,

чувствительные датчики и интерактивные сервомоторы, разъемы для беспроводного Bluetooth и USB подключения, 256 Кбайт FLASH, 64 Кбайт RAM, графический ЖК - дисплей 100×64 пикселя.

Контроллер NXT с комплектующими представлен на рисунке 1.

Рисунок 1 - Контроллер NXT.

В конструктор NXT входит:

1. Датчик касания - позволяет произвести проверку его состояния на определенном этапе исполнения программы. Он посылает зарегистрированный сигнал в виде логической команды ("истина" или "ложь") через шину данных. Если датчик сработал, блок отправит команду "истина"; если датчик не сработал, блок отправит команду "ложь". Датчик касания можно использовать для подачи роботу команды.

2. Датчик цвета - позволяет различать цвета и может использоваться как датчик освещенности. Фактически, датчик цвета совмещает 3 функции. Датчик цвета позволяет роботу различать цвета и отличать свет от темноты. Он может различать 6 цветов, считывать интенсивность света в помещения, а так же измерять цветовую интенсивность окрашенных поверхностей. Датчик цвета так же может использоваться в качестве цветной подставки.

3. Ультразвуковой датчик - позволяет обнаруживать объекты на расстоянии до 100 дюймов (250 см). Датчик передает на выход текущее значение расстояния и логическую команду ("истина" или "ложь") в зависимости от того выше или ниже текущее расстояние, чем значение срабатывания.

Он работает по тому же принципу, что и локатор летучей мыши: он измеряет расстояние путем расчета времени, которое потребовалось звуковой волне для возвращения после отражения от объекта, подобно эху.

Значение срабатывания является специальным значением, лежащим в том диапазоне значений, в пределах которого происходит изменение условий.

4. Интерактивные сервомоторы - сервомотор со встроенным датчиком скорости вращения, который измеряет скорость и расстояние и передает данные на блок NXT. Этот датчик позволяет двигателю контролировать поворот вала с точностью до одного градуса. Можно настроить сервомоторы, чтобы они работали с одинаковой скоростью.

5. Датчик скорости вращения к микрокомпьютеру NXT

Датчик представляет собой одноосный гироскоп, связанный с кварцевым резонатором.

Он измеряет скорость и направление вращения в горизонтальной плоскости. На выход датчика выдается число градусов в секунду вращения и логические сигналы «истина/ложь", когда скорость вращения больше или меньше указанного значения.

Пример робота:

Робот-сигвей, в состоянии стоять на одном колесе и не падать.

6. Детектор инфракрасного излучения к микрокомпьютеру NXT

Датчик измеряет уровень и направление приема инфракрасного сигнала, такого как от инфракрасного мяча. Датчик имеет 5 инфракрасных

приемников ИК-излучения, направленных в разные стороны через каждые 60°.

Значения уровня сигнала устанавливаются в диапазоне от 0 до 9. Если значения уровня сигнала равно 0, это означает, что источник ИК-излучения не может быть определен. Если значение лежит в диапазоне от 1 до 9, то уровень сигнала может быть определен по диаграмме 1. Кроме числовых значений уровня сигнала и направления на выход датчика подаются логические сигналы: «истина», когда сигнал определен и не равен нулю и «ложь», когда сигнала нет.

Пример робота:

Робот-футболист который в состоянии находить инфракрасный мяч.

Микроконтроллер NXT имеет 4 входных порта для подключения датчиков и 3 порта выхода для подключения моторов.

Порт USB и беспроводной канал Bluetooth позволяют вести загрузку и обмен данными между компьютером и NXT. Если компьютер оснащен функцией Bluetooth, то загрузка программ в NXT возможна без использования кабеля USB.

Главное меню микроконтроллера NXT состоит из следующих подменю:

1. Мои файлы - позволяет осуществить обзор всех программ, заданных для NXT или загруженных с компьютера. Данное подменю состоит из папок файлы программ, файлы NXT, звуковые файлы, файлы каталогизатора. Файлы автоматически загружаются в соответствующих папках.

2. Программа NXT. С помощью данного подменю можно создать тысячи различных программ без помощи компьютера.

3. Попробуй (Try Me) . Данное подменю позволит провести тесты датчиков и моторов.

4. Обзор (View). Позволяет провести быстрое тестирование датчиков и моторов, а так же получить текущие данные с каждого устройства.

5. Установки (Setting) . Позволяет настраивать параметры установки NXT, включая громкость громкоговорителей или режим автоматического отключения. Это подменю позволяет также удалять программы, хранящиеся в памяти NXT.

6. Bluetooth. Подменю Bluetooth позволяет создать канал беспроводной связи между NXT и другими устройствами с поддержкой Bluetooth.

1.2 Описание конструктора Lego mindstorms EV3

Характеристики модуля EV3:

1. Операционная система — LINUX;
2. Контроллер ARM9 300 МГц;
3. Флэш-память — 16 МБ;
4. Оперативная память — 64 МБ;
5. Разрешение экрана модуля — 178x128/черно-белый;
6. Связь с главным ПК через шину USB 2.0 — до 480 Мбит/с;
7. Связь с главным ПК через шину USB 1.1 — до 12 Мбит/с;
8. Карта памяти Micro SD — поддерживает SDHC, версия 2.0, макс. 32 ГБ;
9. Порты мотора и датчика;
10. Коннекторы — RJ12;
11. Поддержка автоматической идентификации;
12. Питание — 6 батарей типа АА.

Рисунок контроллера EV3 представлен на рисунке 2.

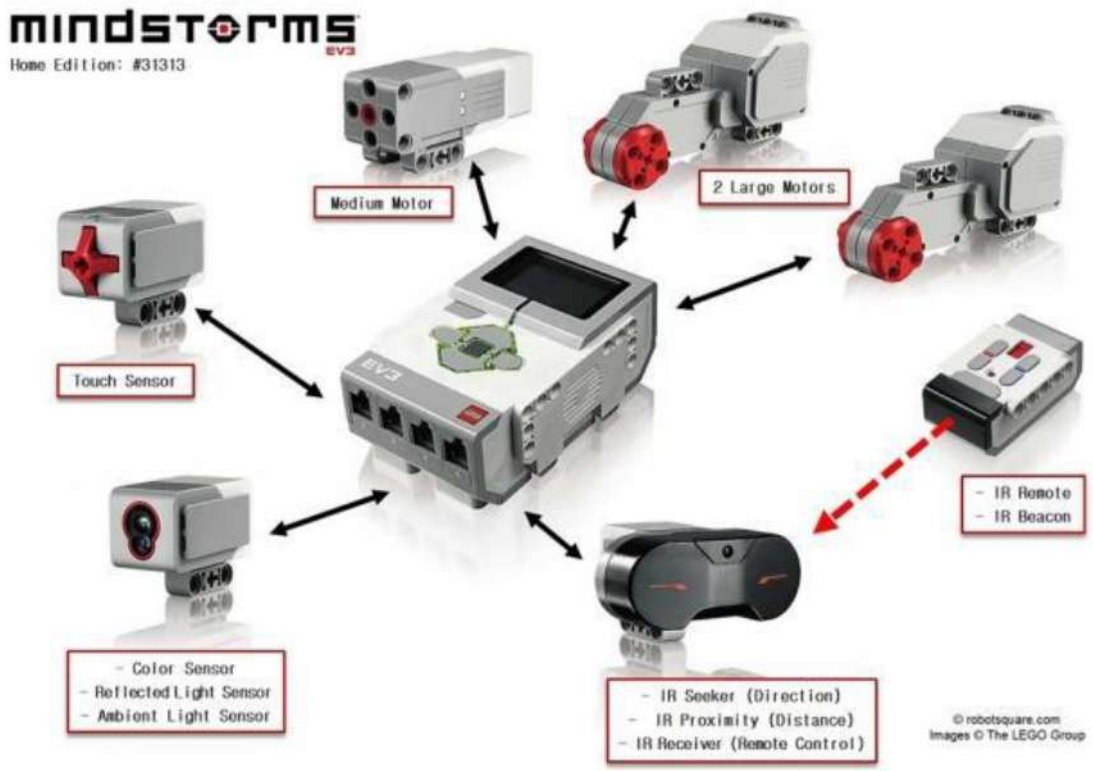

Рисунок 2 - Контроллер EV3.

В конструктор EV3 входит:

1. Большой мотор — это мощный «умный» мотор. В нем есть встроенный датчик вращения с разрешением 1 градус для точного контроля. Большой мотор оптимизирован для выполнения роли приводной платформы в ваших роботах. Используя программные блоки «Рулевое управление» или «Независимое управление моторами» в программном обеспечении EV3, можно координировать работу двух моторов одновременно. Большой мотор работает со скоростью 160–170 об/мин, с вращающим моментом при работающем моторе 20 Нсм и с пусковым моментом 40 Нсм (медленнее, но мощнее).

2. Средний мотор - имеет встроенный датчик вращения (с разрешением 1 градус), но он меньше и легче, чем большой мотор. Это означает, что он способен реагировать быстрее, чем большой мотор. Средний мотор можно запрограммировать таким образом, чтобы он включался или

выключался, контролировал уровень питания, работал в течение определенного времени или выполнял определенное число оборотов. Средний мотор работает со скоростью 240–250 об/мин, с вращающим моментом при работающем моторе 8 Нсм и с пусковым моментом 12 Нсм (быстрее, но с меньшей мощностью).

3. Датчик цвета - это цифровой датчик, который может определять цвет или яркость света, поступающего в небольшое окошко на лицевой стороне датчика. Этот датчик может работать в трех разных режимах: в режиме «Цвет», в режиме «Яркость отраженного света» и в режиме «Яркость внешнего освещения».

В **режиме «Цвет»** датчик цвета распознает семь цветов: черный, синий, зеленый, желтый, красный, белый и коричневый, а также отсутствие цвета. Эта способность различать цвета означает, что ваш робот может быть запрограммирован таким образом, чтобы он сортировал цветные мячи или кубики, произносил названия обнаруженных им цветов или прекращал действие, увидев красный цвет.

В **режиме «Яркость отраженного света»** датчик цвета определяет яркость света, отраженного от лампы, излучающей красный свет. Датчик использует шкалу от 0 (очень темный) до 100 (очень светлый). Это означает, что ваш робот может быть запрограммирован таким образом, чтобы он двигался по белой поверхности до тех пор, пока не будет обнаружена черная линия, или чтобы он интерпретировал идентификационную карточку с цветовым кодом.

В **режиме «Яркость внешнего освещения»** датчик цвета определяет силу света, входящего в окошко из окружающей среды, например солнечного света или луча фонарика. Датчик использует шкалу от 0 (очень темный) до 100 (очень светлый). Это означает, что ваш робот может быть запрограммирован таким образом, чтобы он подавал сигнал утром, когда

восходит солнце, или чтобы он прекращал действие, если свет гаснет. Частота выборки датчика цвета составляет 1 кГц/с.

4. Датчик касания - это аналоговый датчик, который может определять, когда красная кнопка датчика нажата, а когда отпущена. Это означает, что датчик касания можно запрограммировать для действия в зависимости от трех условий: нажатие, отпускание и щелчок (нажатие и отпускание). Используя вводы датчика касания, робота можно запрограммировать таким образом, чтобы он воспринимал мир, как его может воспринимать слепой человек, когда он протягивает руку и реагирует при соприкосновении с чем-либо (нажатие).

Вы можете построить робота с датчиком касания, который прижат к поверхности под ним. Вы можете запрограммировать робота так, чтобы он реагировал (Стоп!), когда он вот-вот скатится с края стола (когда датчик отпущен).

5. Инфракрасный датчик и удаленный инфракрасный маяк.

Инфракрасный датчик — это цифровой датчик, который может обнаруживать инфракрасный цвет, отраженный от сплошных объектов. Он также может обнаруживать инфракрасные световые сигналы, посланные с удаленного инфракрасного маяка.

Инфракрасный датчик может использоваться в трех разных режимах: в режиме приближения, в режиме маяка и в дистанционном режиме.

В режиме приближения инфракрасный датчик использует световые волны, отраженные назад от объекта, для определения расстояния между датчиком и этим объектом. Он сообщает расстояние, используя значения от 0 (очень близко) до 100 (далеко), а не конкретное число сантиметров или дюймов. Датчик может обнаруживать объекты на удалении до 70 см, в зависимости от размера и формы объекта.

Можно выбрать одни из четырех каналов удаленного инфракрасного маяка с помощью красного переключателя каналов. Инфракрасный датчик

обнаружит сигнал маяка, соответствующий каналу, который вы укажете в своей программе, на удалении примерно до 200 см в направлении перед ним. После обнаружения датчик может оценить общее направление (курс) и расстояние (приближение) до маяка. Используя эту информацию, можно запрограммировать робота так, чтобы он играл в прятки, используя удаленный инфракрасный маяк в качестве искомой цели. Направление будет выражено величиной от -25 до 25, при этом 0 указывает, что маяк находится прямо перед инфракрасным датчиком. Приближение будет выражено величинами от 0 до 100.

Удаленный инфракрасный маяк — это отдельное устройство, которое можно держать в руке или которое может быть встроено в другую модель LEGO. Для него необходимы две щелочные батареи типа AAA. Для включения удаленного инфракрасного маяка нажмите большую кнопку «Режим маяка», расположенную сверху на устройстве. Загорится зеленый светодиодный индикатор, указывая, что устройство активно и постоянно передает сигналы.

При повторном нажатии кнопки «Режим маяка» он выключится (после бездействия маяк выключается автоматически).

Также можно использовать удаленный инфракрасный маяк для дистанционного управления своим роботом. Работая в дистанционном режиме, инфракрасный датчик может определять, какая кнопка (или комбинация кнопок) на маяке нажата.

Подключить модуль EV3 к компьютеру можно с помощью USB-кабеля или посредством беспроводной связи, используя либо Bluetooth, либо Wi-Fi.

1.3 Описание конструктора TETRIX

TETRIX компании PITSCO – робототехнический конструктор нового поколения, который позволяет перевести процесс создания робота на новый качественный уровень.

На его основе можно построить робота с дистанционным управлением или, используя микрокомпьютер и датчики, создать автономного робота.

TETRIX - основной конструктор международных соревнований FIRST Tech Challenge. Для конструкций из TETRIX установлены специальные номинации на во многих соревнованийх на всероссийском фестивале «РобоФест». Детали TETRIX широко используют участники соревнований ABU ROBOCON, ELROB и др.

Конструктор TETRIX совместим с конструктором LEGO, позволяет использовать контроллер LEGO NXT, любые совместимые датчики и исполнительные устройства.

Конструктор TETRIX содержит все необходимое для создания металлического робота , которые управляются с помощью контроллера NXT или EV3.

Конструктор TETRIX состоит из:

1. Двигатели постоянного тока и контроллер к двигателям постоянного тока HiTechnic;

2. Сервомоторы и контроллер к серводвигателей HiTechnic;

3. Аккумулятор 12 вольт;

4. Металлические балки, пластины, уголки, скобы и трубки из сплава алюминия и дюрали.

2 Язык программирования RobotC

ROBOTC - это текстовый язык программирования, основанный на стандартном языке программирования C. В настоящий момент это единственный язык программирования для роботов, который предоставляет развитый режим отладки во время выполнения программ. ROBOTC является кросс-платформенным решением, которое позволит студентам и ученикам изучить C-подобный язык, используемый в большинстве образовательных и профессиональных приложений.

Команды для робота записываются в виде текста на экране, после чего компилятор ROBOTC обрабатывает файл и переводит текст на машинный язык. Затем файл загружается на робота, где он может быть запущен.

ПО RobotC позволяет разрабатывать приложения для работы со следующими платформами: TETRIX, NXT, Cortex, RCX, PIC, VEX PIC, Arduino Diecimila, Duemilanove, Mega 2560, Mega 1280, Uno. Программное обеспечение имеет схожую с Visual Studio среду и включает в себя мощный интерактивный отладчик, способный функционировать в режиме реального времени, тем самым существенно сокращая время отладки кода. Данная среда обладает развитыми возможностями для работы с математическими выражениями, с помощью которых можно составлять весьма эффективные и сложные программы. В RobotC существует опция предоставления данных с датчиков в «сыром» виде в формате RAW. Среда может поддерживать связь с устройствами посредством инфракрасного канала или Wi-Fi.

Интерфейс RobotC показан на следующем рисунке:

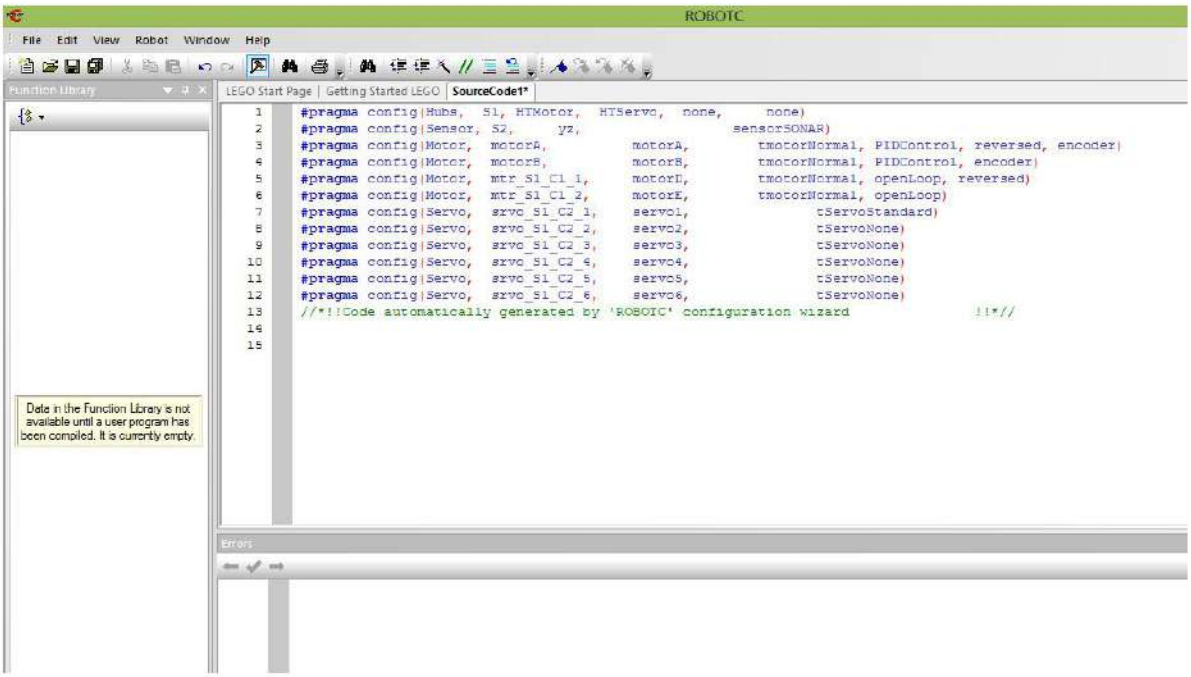

Рисунок 3 - Интерфейс языка программирования RobotC.

Интерфейс RobotC состоит из следующих меню:

1. Меню Файл используется для запуска новых программ , открытие существующих программ, программ сохранения, печати и доступа к недавно открытых программ.

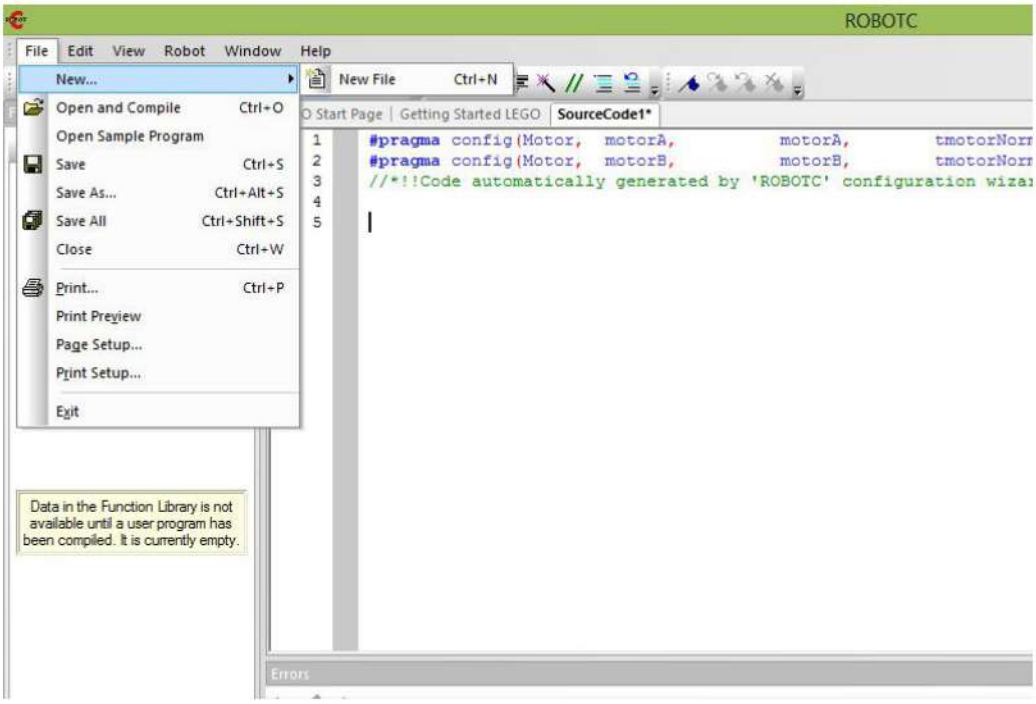

Рисунок 4 - Меню «Файл».

2. Меню Правка содержит полезные инструменты для написания программ. Функции такие как отменить, повторить, копировать, вставить, найти и закладки можно найти в меню Правка.

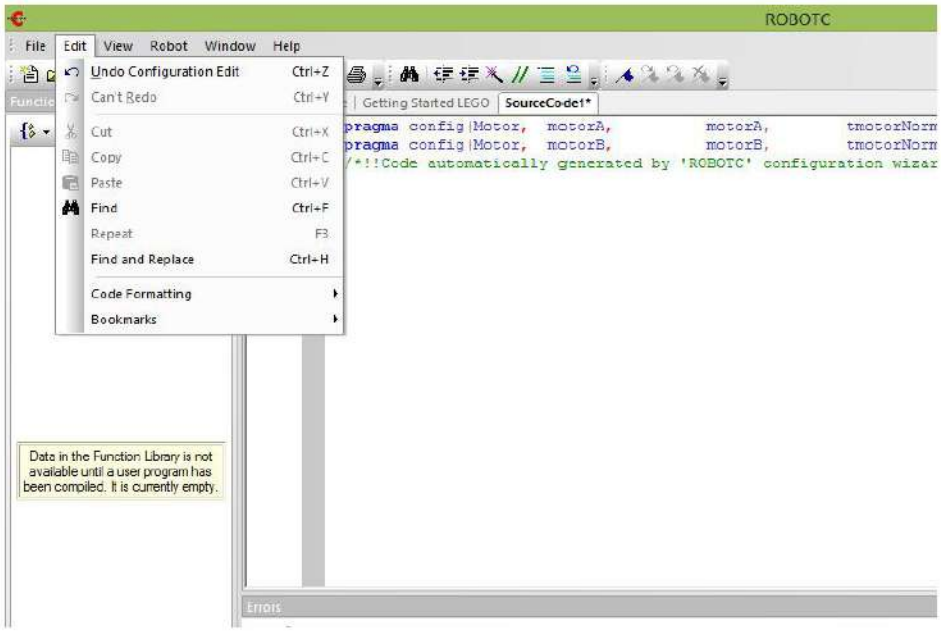

Рисунок 5 - Меню «Правка».

3. Меню Вид позволяет настроить способ отображения интерфейса ROBOTC. В зависимости от положения режима "Basic" или "Expert" существует два основных уровня меню Вид.

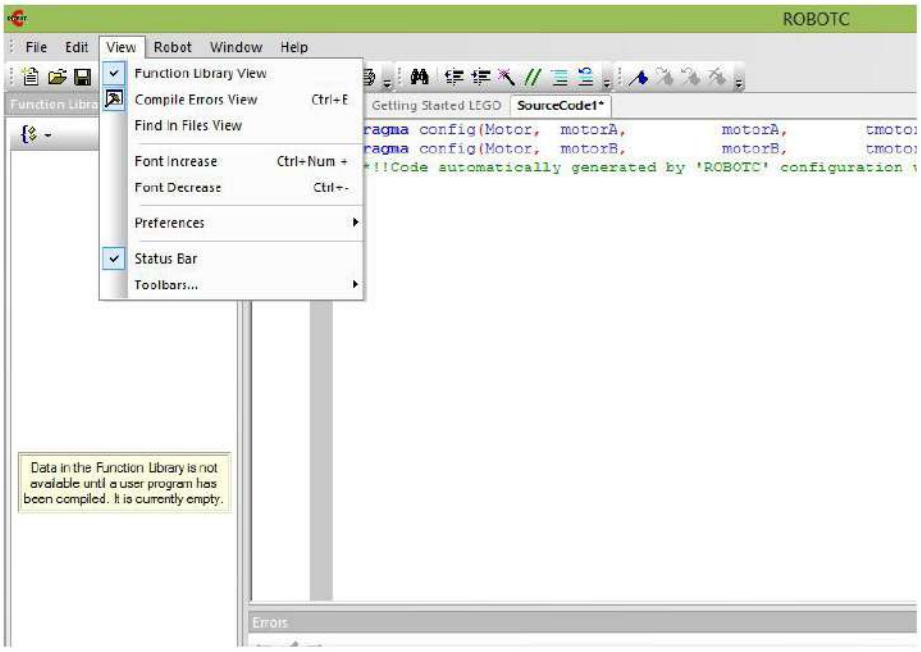

Рисунок 6 - Меню «Вид».

4. Меню Robot обеспечивает доступ ко всем инструментам, доступных для взаимодействия с вашим контроллером робота. Загрузка программ, встроенного программного обеспечения и запуск различных отладочных окон сделаны через это меню.

Можно также изменить платформу ROBOTC, установки и конфигурацию датчиков и двигателей для данной конкретной платформы.

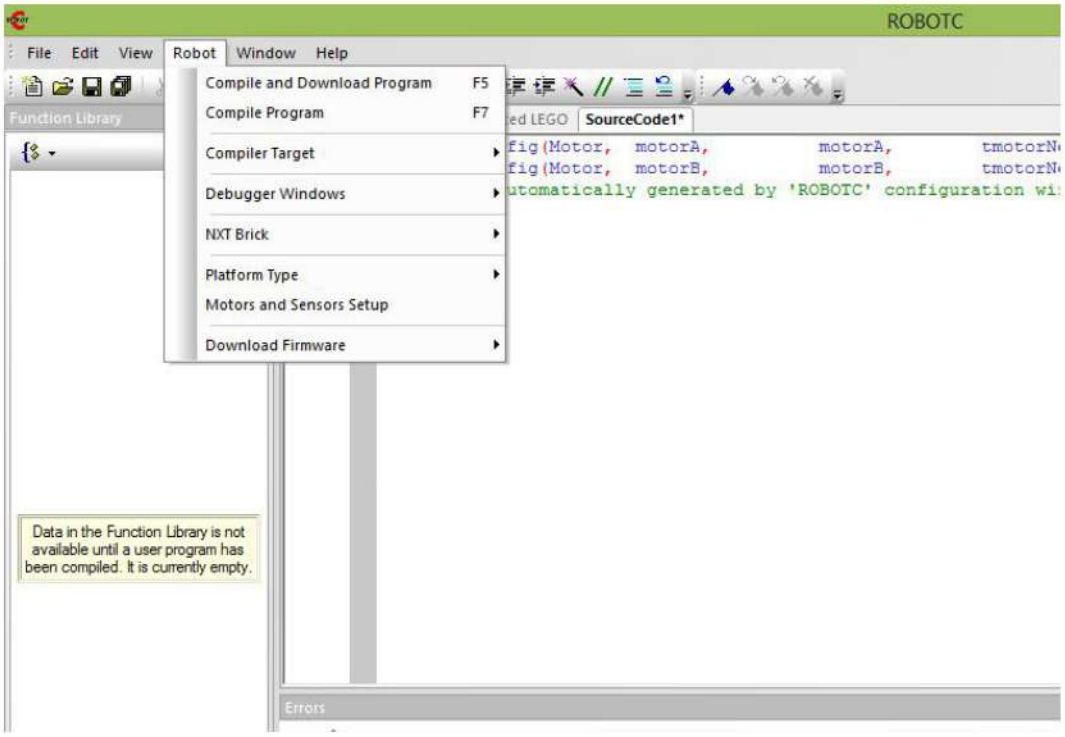

Рисунок 7 - Меню «Robot».

5. Меню Окно используется для переключения между режимами меню в ROBOTC:

- Основной режим (Basic) скрывает больше дополнительных настроек конфигурации, делая интерфейс легче ориентироваться для новых пользователей .
- Экспертный режим (Expert) показывает большинство дополнительных настроек ROBOTC, предоставляя больше инструментов для опытных пользователей .

- Супер Пользовательский режим (Super User) показывает все расширенные настройки ROBOTC , позволяя пользователю в полной мере воспользоваться возможностями ROBOTC.

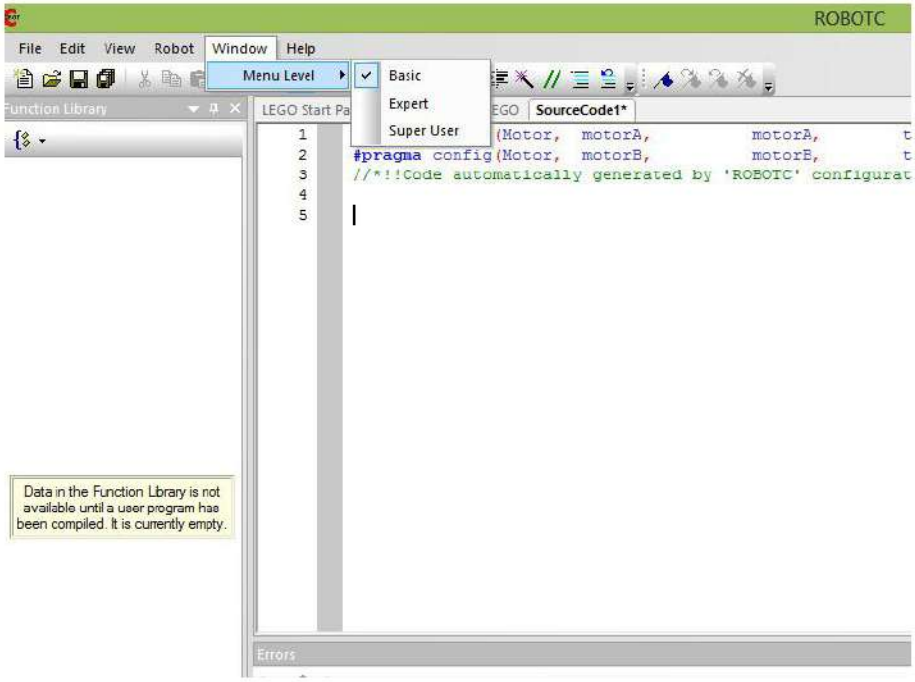

Рисунок 8 - Меню «Окно».

6. Меню Справка позволяет получить доступ к ресурсам программирования ROBOTC .

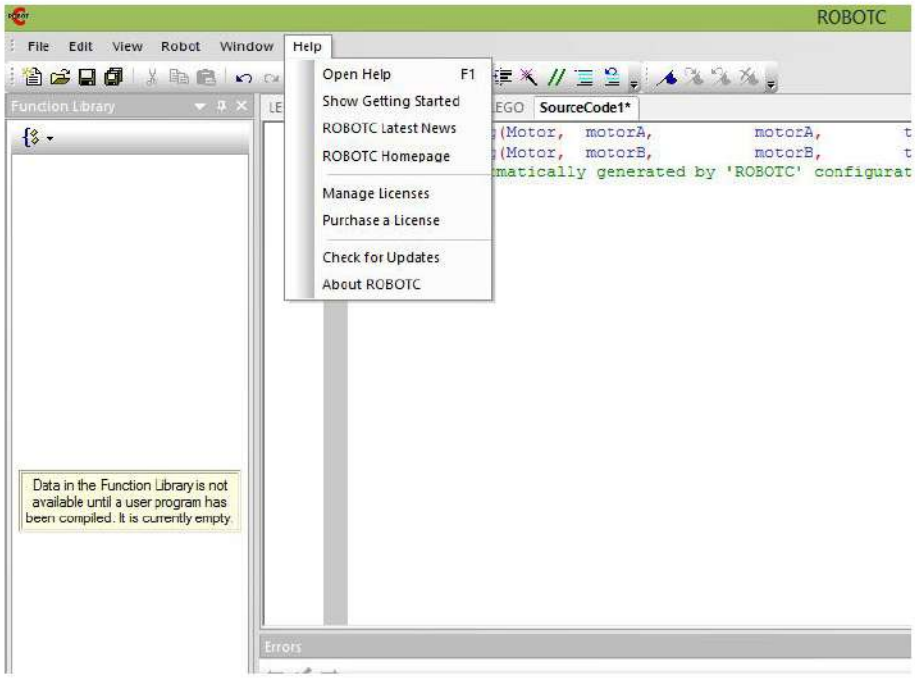

Рисунок 9 - Меню «Справка».

ROBOTC for MINDSTORMS имеет поддержку четырех различных платформ :

- LEGO Mindstorms RCX;
- LEGO Mindstorms NXT;
- LEGO Mindstorms NXT + TETRIX;
- Робот Алгебра (NXT).

Перед тем, как использовать ROBOTC, нужно убедиться в правильности выбора платформы. Изменить тип платформы можно с помощью меню Robot (рисунок 10).

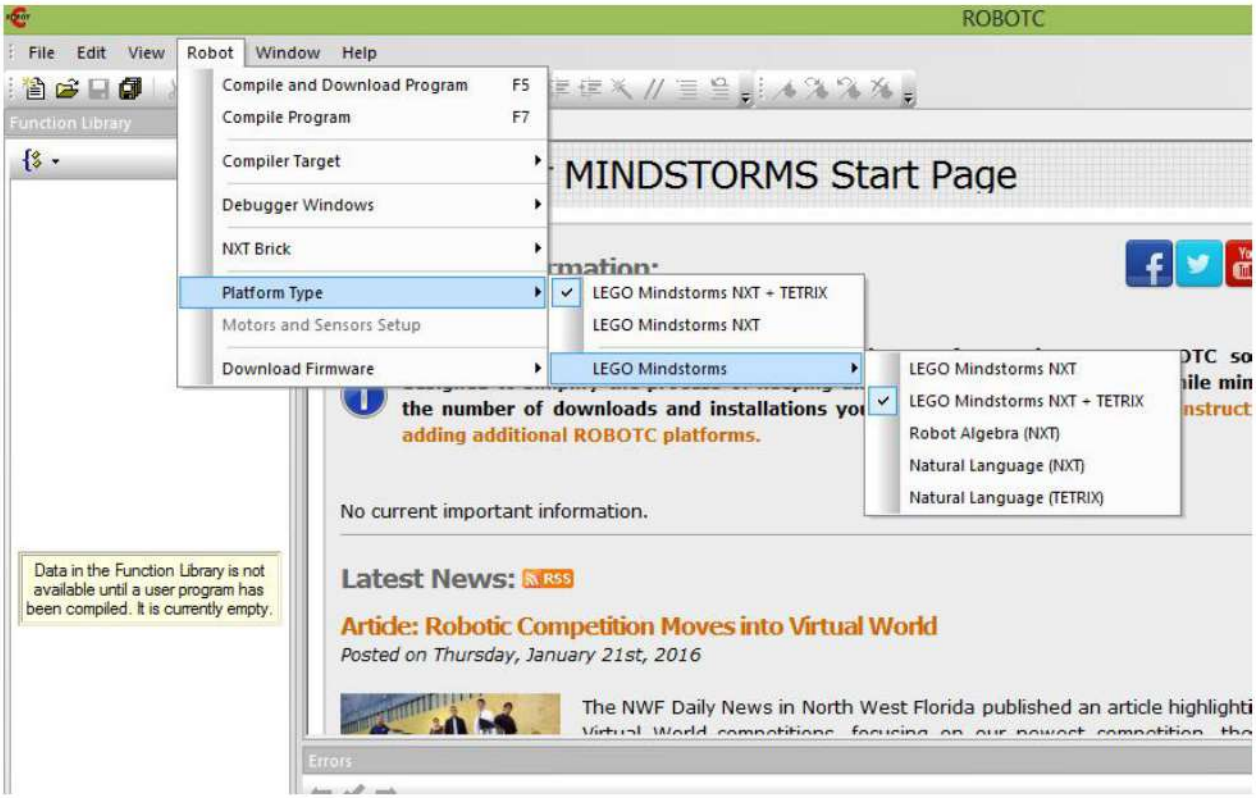

Рисунок 10- Выбор типа платформы.

Один из лучших способов начать работать с ROBOTC - это просмотр уже разработанных программ. ROBOTC for MINDSTORMS предоставляет более чем 150 примеров программ, чтобы помочь начинающим программистам научиться программировать своих роботов .

NXT Brick содержит функции управления и соединения NXT. "Joystick Control - Basic" и "Joystick Control - FTC" это окна отладчика для управления NXT через USB- пульт дистанционного управления Logitech (рисунок 11). Окно «Communication Link Setup» позволяет выполнять соединение с компьютером контроллера NXT с помощью USB - кабеля, Bluetooth или Wi-Fi (рисунок 12).

Рисунок 11 - Подключение джойстика.

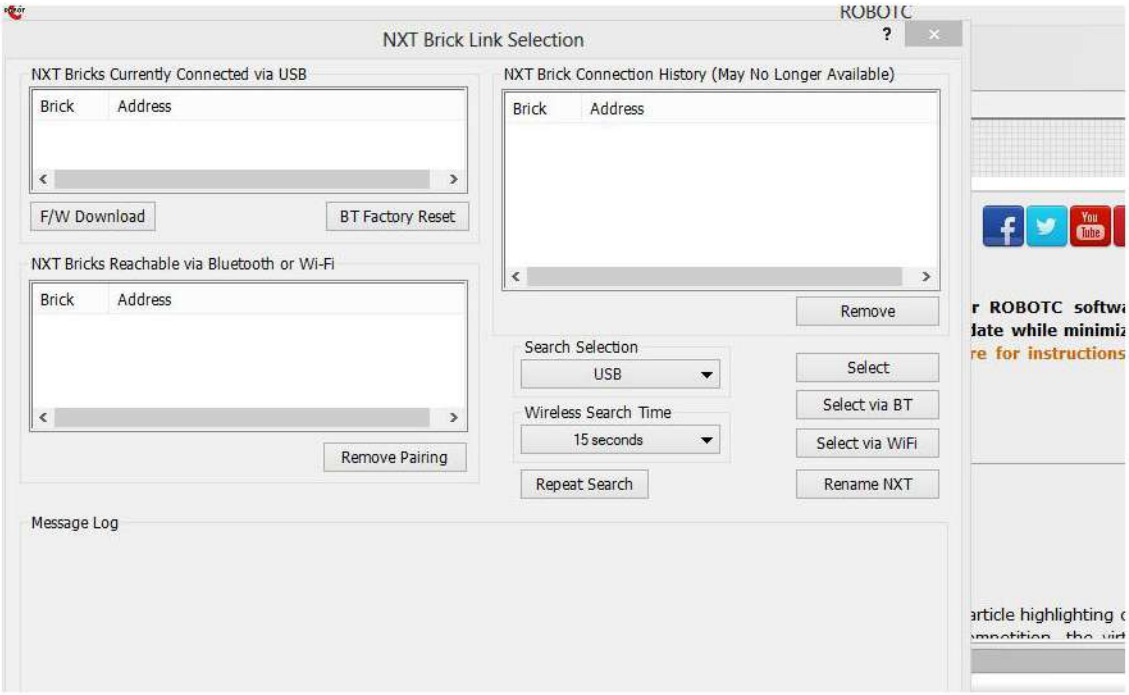

Рисунок 12 - Соединение контроллера NXT с компьютером.

3 Комплекс программ по основам робототехники на базе конструкторов LEGO MINDSTORMS

3.1 Управление мобильным роботом

Главная особенность мобильного работа - способность к маневрированию. В отличие от привычного автомобиля, оснащенного рулевым управлением и одним двигателем, робот может быть оснащен несколькими моторами, два из которых используются для движения и маневров, соединенные по отдельности с правыми и левыми колесами.

Для движения вперед используются команды управления моторами. Эти команды просто включают моторы.

```
task main() // Пример включения
            // моторов на RobotC
{
motor[motorB] = 100; // моторы вперед
motor[motorC] = 100;// с максимальной
                    // мощностью
}
```

Обе команды включаются практически мгновенно. Если сразу за ними выключить моторы, то тележка просто дернется и будет стоять на месте.

```
task main()
{
motor[motorB] = 100;
motor[motorC] = 100;
motor[motorB] = 0; // стоп мотор
motor[motorC] = 0;
}
```

Таким образом, для осуществления движения требуется некоторая задержка перед выключением моторов. Команды ожидания не производят никаких конкретных действий , зато дают возможность моторам выполнить свою часть работы.

```
task main()
{
motor[motorB] = 100;
motor[motorC] = 100;
wait1Msec(1000);   // Ждать 1000 мс
motor[motorB] = 0;
motor[motorC] = 0;
}
```

Движение вперед и назад определяется направлением вращения моторов. Для смены направления не требуется остановка.

```
task main()
{
motor[motorB] = 100;
motor[motorC] = 100;
wait1Msec(1000);
motor[motorB] = -100;
                 // «Полный назад»
motor[motorC] = -100;
wait1Msec(1000);
motor[motorB] = 0;
motor[motorC] = 0;
}
```

В момент смены направления на высокой скорости возможен занос. Плавное торможение возможно. Для этого перед командой подачи назад с моторов снимается напряжение и робот некоторое время едет по инерции.

Более краткий промежуток , чем одна секунда задается , задается с помощью команды N/100 и модификатора.

```
task main()
{
motor[motorB] = 100;
motor[motorC] = 100;
wait1Msec(1000);
// Включить плавающий режим
// управления моторами
bFloatDuringInactiveMotorPWM = true;
motor[motorB] = 0;
motor[motorC] = 0;
wait1Msec(500);
motor[motorB] = -100;
motor[motorC] = -100;
wait1Msec(1000);
// Включить режим «торможения»
bFloatDuringInactiveMotorPWM = false;
motor[motorB] = 0;
motor[motorC] = 0;
}
```

Для выполнения поворота на месте достаточно включить моторы в разные стороны. Тогда робот будет вращаться приблизительно вокруг центра оси ведущих колес со смещением в сторону центра тяжести.

```
task main()
{
motor[motorB] = 100;  // Моторы в
motor[motorC] = -100; // разные стороны
wait1Msec(300);
motor[motorB] = 0;
motor[motorC] = 0;
}
```

Используя полученные знания управления моторами, можно запрограммировать движение по квадрату или другому многоугольнику с помощью цикла или безусловного перехода.

```
task main()
{
while (true){
  motor[motorB] = 100;
  motor[motorC] = 100;
  wait1Msec(1000);
  motor[motorC] = 0;
  wait1Msec(1000);
  motor[motorB] = 0;
}
}
```

Уточнив длительность переходов и количество повторений, научим тележку объезжать квадрат по периметру один раз. Для точности поворотов снизим мощность моторов вдвое. Задержки придется подбирать самостоятельно.

```
task main()
{
for(int i=0;i<4;i++){
        // Цикл выполняется 4 раза
  motor[motorB] = 50;
  motor[motorC] = 50;
  wait1Msec(1000);
  motor[motorC] = -50;
  wait1Msec(400);
  motor[motorB] = 0;
}
}
```

Что бы поворот не зависел от заряда батареек, можно воспользоваться встроенным в двигатель датчиком оборотов, который позволяет делать измерения с точностью до 1 градуса.

```
task main()
{
nMotorEncoder[motorB]=0;
          // Инициализация энкодера
motor[motorB] = 100;
motor[motorC] = -100;
// Пустой цикл ожидания
//показаний энкодера
while(nMotorEncoder[motorB]<250);
motor[motorB] = 0;
motor[motorC] = 0;
```

Нормальная среда обитания для робота, построенного из школьного или домашнего конструктора - это комната с мебелью. И нужно научиться путешествовать по ней, не натыкаясь на предметы и не застревая.

Подходящая конструкция для такого робота - это тележка с установленным ультразвуковым датчиком наверху. Этот датчик следует расположить строго горизонтально относительно пола, иначе любая соринка будет воспринята как препятствие.

Программа похожа на движение по квадрату с небольшим добавлением: встречая предмет, робот немного отъезжает назад, прежде чем приступить к повороту.

```
task main()
{
while (true){
  motor[motorB] = 100;
  motor[motorC] = 100;
  while(SensorValue[S4]>25);
// пустой цикл ожидания препятствия
  motor[motorB] = -100;
  motor[motorC] = -100;
  wait1Msec(500);
  motor[motorB] = 100;
  wait1Msec(600);
}
}
```

Можно сделать намного короче, если заменить отъезд назад с поворотом на месте одним действием: плавным поворотом задним ходом.

```
task main()
{
while (true){
  motor[motorB] = 100;
  motor[motorC] = 100;
  while(SensorValue[S4]>25);
  motor[motorB] = -100;
  motor[motorC] = 0;
  wait1Msec(700);
}
}
```

3.2 Движение по линии

Одна из классических задач мобильного робота - это движение по черной линии на белом поле с использованием датчиков освещенности.

Высота датчика над поверхностью поля от 5 до 10 мм.

Задача: двигаться по плоскому полю вдоль границ черного и белого. Решается применением релейного двухпозиционного регулятора. В таком регуляторе рассматриваются только 2 состояния датчика и 2 вида управляющего воздействия на моторы. Пока датчик на белом, робот двигается в сторону черного, пока датчик на черном, робот двигается в сторону белого.

Данную программу можно записать следующим образом:

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

Перед стартом робот ставится на линию так, что бы датчик был над белым полем на расстоянии 2-3 см слева от черного. По алгоритму робот плавно поворачивает направо, пока освещенность не понизится на 5 пунктов. Затем поворачивает налево, пока освещенность не повысится на 5 пунктов.

Возможные проблемы:

1. Робот крутится на месте, не заезжая на линию.

В этом случае следует стартовать с другой стороны линии, либо поменять подключение моторчиков к контроллеру местами.

2. Робот проскакивает линию, не успевая среагировать.

Следует понизить мощность моторов.

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

Надо увеличить порог чувствительности датчика.

Усовершенствованная программа выглядит так:

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

Особенность приведенных алгоритмов в том, что во всех случаях роботу требуется стартовать на белом поле и используется относительное понижение освещенности. Однако есть возможность заранее определить уровень освещенности на данном поле и использовать его абсолютное значение. Воспользовавшись показаниями датчика на белом и на черном, можно рассчитать их среднее арифметическое, которое можно условно назвать «значением серого». Пересекая датчиком значение 45, робот будет менять направление движения. По левую сторону от «серого» все показания будут «белыми», а по правую «черными». В алгоритме заменятся блоки ожидания показаний датчиков на «жди черного» и «жди белого».

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

Более устойчиво алгоритм работает, если использовать моторы с управлением мощностью от -100 до 100.

В этом случае есть возможность отрегулировать плавность поворота в соответствии с кривизной линии.

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

В этом алгоритме притормаживающие моторы на повороте не останавливаются полностью, а лишь понижают мощность до 20 пунктов. Это делает поворот плавным , но может привести и к потере линии на резком повороте.

Следующий пример движения вдоль границы черного и белого с использованием ветвления. В качестве значения перехода границы черного и белого взято значение 45, которое было рассчитано для предыдущих алгоритмов.

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

Задержка в одну миллисекунду предназначена для того, чтобы немного разгрузить микроконтроллер.

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

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

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

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

где k - усиливающий коэффициент.

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

Движение по границе черного и белого можно настроить с помощью П- регулятора.

Определяя состояние робота как показания датчика освещенности , можно оказывать пропорциональное управляющее воздействие на моторы последующему закону:

$$e = s_1 - grey,$$

где s_1 - текущие показания датчика, а grey - заданное значение,

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

где N - базовая мощность двигателей.

Так выглядит алгоритм пропорционального регулятора для одного датчика освещенности:

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

Калибровка датчика.

Обратимся к числу 48, использованному в формуле управления u. Это среднее арифметическое показаний датчика освещенности на черном и на белом. Однако показания датчиков часто меняются по различным причинам: другая поверхность, изменение общей освещенности в помещении, небольшая модификация конструкции. Поэтому имеет смысл научит работа самостоятельно вычислять среднее арифметическое , то есть значение границы белого и черного.

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

Значение освещенности с датчика на порту 1 считывается в именованный контейнер grey , после чего оно уменьшается на число 8, и в формуле управления u переменная grey используется как заданное значение серого. Инициализация датчика происходит при вызове команды считывания в контейнер, поэтому отдельная команда не требуется.

В RobotC инициализация датчика происходит через меню Robot → Motors and Sensors Setub. В качестве типа датчика устанавливается Light Active. Так же рекомендуется задать имя датчика и в программе использовать имя вместо привязанного к конкретному порту значения S_1 .

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

Надо иметь в виду, что такой способ калибровки не учитывает все возможные варианты, а только экономит время на программирование и отладку.

Если же времени достаточно, то есть другой способ, при котором действительно производится расчет среднего арифметического.

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

Предложенный алгоритм обладает некоторым неудобством: при запуске потребуется быть внимательным и не пропустить звукового сигнала , после которого робота нужно переместить так, что бы датчик освещенности оказался над белым полем. В контейнере Black будет сохранено значение черного, в контейнере white - значение белого. В переменную grey помещается значение серого , которое используется в регуляторе. Сразу после второго сигнала робот начнет движение.

Процесс калибровки можно сделать управляемым. Для этого после каждого считывания данных необходимо вставить ожидание какого-либо внешнего события, например нажатия на датчик касания, уменьшения расстояния на ультразвуковом датчике или просто нажатия на кнопку NXT.

В двух последних примерах использованы именованные контейнеры, которые являются переменными, как в обычном языке программирования. Звуковой сигнал перед стартом способствует тому, что бы робот не среагировал дважды подряд на одно и тоже нажатие, что не исключено. Эта проблема решается ожиданием отпускания датчика.

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

После первого звукового сигнала нужно поставить тележку так, чтобы датчик освещенности оказался над белым. После второго - подготовиться к старту и по повторному нажатию кнопки робот начнет движение.

3.3 Танец в круге

Для выполнения этой задачи необходимо 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.

Теперь нужно подготовить ринг. Это может быть круг или его подобие диаметром 100 см, очерченный двумя- тремя слоями черной

изоленты. Главное условие опыта состоит в том, что показания датчика на черной линии и внутри круга должны различаться не менее чем на 15 пунктов.

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

Последовательность действий такова:

1. Ехать вперед, пока показания датчиков не понизятся на 5 пунктов;
2. Отъехать немного назад;
3. Развернуться примерно на 120-150 градусов;
4. Повторять пункты 1-3 бесконечно.

```
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.

Вытолкнуть все банки.

Диаметр круга - 1 м. Несколько пластиковых стаканчиков или жестяных банок , расставленные внутри за черной линией - это мусор, от которого необходимо избавиться за кротчайшее время.

Первые попытки запуска робота покажут следующие недостатки:

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 делает много движений впустую.

Первый недостаток устраняется конструктивно, а второй - программно.

Пусть, увидев край, тележка еще немного движется вперед , выталкивая стаканчик, и только после этого отъезжает внутрь круга. 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 так же использовать реверс при смене направления работающих моторах.

```
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);
}
}
```

Теперь стоит поработать над точностью движения, при этом не теряя скорости.

В зависимости от конструкции робота, при резкой смене направления он может потерять равновесие или просто встать на передние колеса.

Поэтому последние несколько сантиметров можно проехать на торможении по инерции, то есть полностью освободив моторы.

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.

Можно пожертвовать реверсом в последней команде управления мотором B, для того чтобы достичь неторопливого движения обоими моторами. Длительность поворота при этом немного возрастет.

```
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 внести 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.

```
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);
}
}
```

Не делать лишних движений.

Желание контролировать положение робота приводит к необходимости изменить траекторию движения таким образом, что бы каждый раз, доехав до края, он возвращался в центр круга. Если бы не было встроенных датчиков оборотов, пришлось бы засекать время, которое робот проехал вперед и давать команду столько же ехать назад. Вот соответствующий пример:

```
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.

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

```
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. Датчик можно расположить вертикально, любой стороной, и он будет работать вполне сносно.

Вращение осуществляется до тех пор, пока на датчик расстояния не поступит сигнал «ближе 45». То есть обнаружен объект на расстоянии ближе 45 см. Надо иметь ввиду, что на стаканчики, которые робот уже вытолкнул за линию он не должен обращать внимания. По этому указанное расстояние не следует делать больше радиуса круга.

```
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, во-первых, нужно их выталкивать дальше за пределы круга, во-вторых, начинать поворот в центре вслепую, а в-третьих, все действия 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);
}
}
```

Состязания «Кегельринг» уже несколько лет проводятся в России и находят все больше поклонников, поскольку отличаются простотой и доступностью.

3.4 Ориентация на местности: объезд стены.

Когда-нибудь в недалеком будущем роботы выйдут в коридоры, на улицы и станут нашими соседями не только по лабораториям. Поэтому нужно научить робота объезжать стены.

Движение вдоль стенки.

Решим такую задачу. Робот должен двигаться вдоль стенки на заданном расстоянии L (рисунок 13), которое определяется в момент старта. Предположим, что левое колесо робота управляется мотором B, правое - мотором C, а датчик расстояния, подключенный к порту 1, закреплен несколько впереди корпуса тележки и направлен на стенку справа по ходу движения.

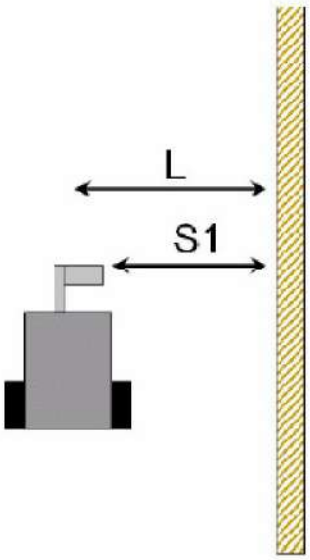

Рисунок 13 - Движение робота вдоль стенки.

Расстояние до стенки в настоящий момент времени, которое показывает датчик, обозначим S1. Измеряется оно в сантиметрах.

Моторы двигаются со средней скоростью 50% от максимума, но при отклонении от заданного курса на них осуществляется управляющее воздействие u. Обозначим это следующим образом:

```
Motor[MotorB]=50+u;
Motor[MotorC]=50-u;
```

Осталось определить, чему будет равно управляющее воздействие. Это нетрудно сделать с помощью пропорционального регулятора:

$$u = k*(Sl-L).$$

Таким образом, при S1 = L робот не меняет курса и едет прямо. В случае отклонения его курс корректируется. Здесь k - это некоторый усиливающий коэффициент, определяющий воздействие регулятора на систему. Для робота NXT средних размеров коэффициент k может колебаться от 1 до 10, в зависимости от многих факторов.

```
task main()
{
  float u, k=3;
  int L=SensorValue[S1];
  while(true)
  {
     u=k*(SensorValue[S1]-L);
     motor[motorB]=50+u;
     motor[motorC]=50-u;
     wait1Msec(1);
  }
}
```

В данном случае П-регулятор будет эффективно работать только при малых углах отклонения. Кроме того, движение практически всегда будет происходить по волнообразной траектории. Сделать регулирование более точным позволит введение новых принципов, учитывающих отклонение робота от курса.

Пропорционально-дифференциальный регулятор.

В некоторых ситуациях П-регулятор может вывести систему из устойчивого состояния (рисунок 14, а). Например, если робот направлен от стенки, но находится по отношению к ней ближе заданного расстояния, на моторы поступит команда еще больше повернуть от стенки, в результате чего с ней может быть утерян контакт (датчик расстояния получает отраженный сигнал практически только от перпендикулярной поверхности).

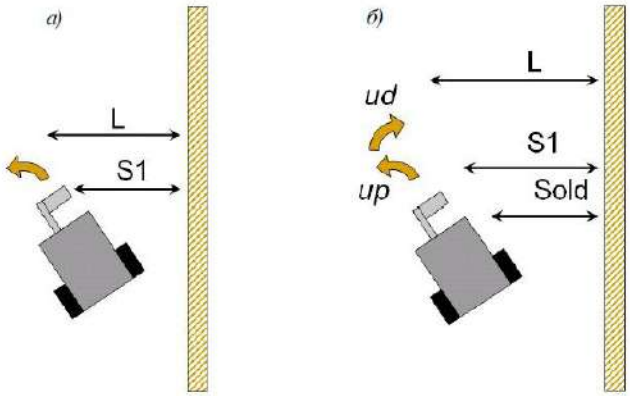

Рисунок 14 - Проблема пропорционального регулятора - потеря контакта со стенкой (а). Необходима дифференциальная составляющая (б).

Для защиты от подобных ситуаций добавим в регулятор дифференциальную составляющую, которая будет следить за направлением движения робота (рисунок 14, б). Иными словами, вектор скорости будет влиять на управляющее воздействие.

Известно, что скорость находится по следующей формуле:

$$v = \Delta s / \Delta t,$$

где Δs - это изменение расстояния за промежуток времени Δt. Определим дифференциальную составляющую через скорость отклонения робота от заданного положения:

$$ud = k*(S1 - Sold)/\Delta t,$$

где S1 - текущее расстояние до стенки, Sold - расстояние на предыдущем шаге.

Поскольку замеры производятся через равные промежутки времени, то Δt можно принять за константу, взяв k2=k* Δt.

$$ud = k2*(S1 - Sold)$$

Таким образом, ПД-регулятор описывается формулой из двух слагаемых:

$$u = up+ud = k1*(S1 - L) + k2*(S1 - Sold).$$

Можно доказать математически, что для устойчивого достижения цели коэффициент k2 при дифференциальной составляющей должен превышать k1.

Алгоритм движения вдоль стенки на ПД-регуляторе в целом будет выглядеть так:

```
task main()
{
  float u, k1=2, k2=10;
  int Sold, L;
  Sold=L=SensorValue[S1];
  while(true)
  {
    u= k1*(SensorValue[S1]-L) +
        k2*(SensorValue[S1]-Sold);
    motor[motorB]=50+u;
    motor[motorC]=50-u;
    Sold=SensorValue[S1];
    wait1Msec(1);
  }
}
```

Датчик под углом.

Описанный выше робот сможет объезжать стены только при малых отклонениях от прямой линии. Рассмотрим вариант, при котором на пути движения будут возникать серьезные повороты, вплоть до прямых углов. Потребуется внести модификации и в конструкцию и в программу (рисунок 15).

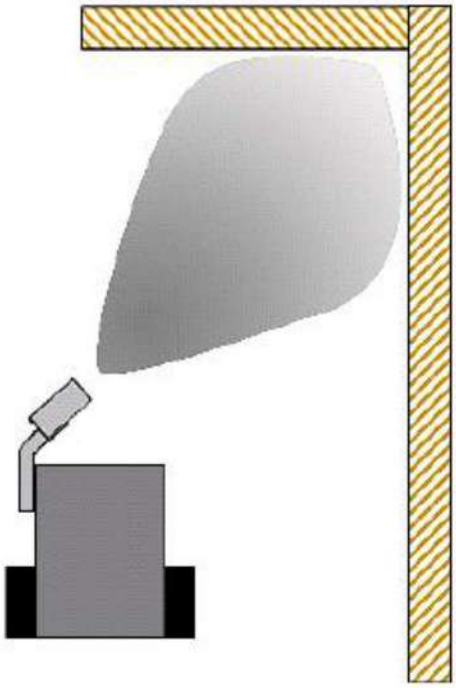

Рисунок 15 - Датчик расстояния устанавливается под острым углом к направлению движения

Во-первых, робот должен будет смотреть не только направо, но и вперед. Ставить второй дальномер довольно затратно. Однако можно воспользоваться эффектом того, что ультразвуковой датчик имеет расширяющуюся область видимости (рисунок 15). Это напоминает угловое зрение человека: кое-что он может увидеть краем глаза. Стоит воспользоваться таким свойством и разместить датчик расстояния не перпендикулярно курсу движения, а под острым углом. Так можно «убить сразу двух зайцев». Во-первых, робот будет видеть препятствия спереди, во-вторых, более стабильно будет придерживаться курса вдоль стены, постоянно находясь на грани видимости. Таким образом, без добавления новых устройств будет получено более эффективное использование возможностей дальномера.

Важное замечание. При старте робота его надо будет направлять датчиком строго на стену, чтобы процесс считывания начального значения прошел без помех.

Очевидно, что изменение конструкции влечет изменение коэффициентов регулятора k1 и k2. Обычно подбор начинается с пропорционального коэффициента при нулевом дифференциальном. Когда достигнута некоторая стабильность на небольших отклонениях, добавляется дифференциальная составляющая.

Поворот за угол.

Следующим шагом необходимо ограничить реакцию робота на «бесконечность». Как известно, когда в поле видимости нет объекта, показания датчика расстояния NXT равны 250 или 255 см. Если это число попадает на пропорциональный регулятор, робот начинает крутиться на месте. А в ситуации, когда роботу следует завернуть за угол, именно это и произойдет.

Для объезда предметов потребуется ввести контроль показаний датчика расстояния: при резком изменении робот должен делать вывод о возможном повороте, который надо будет производить с другими коэффициентами или просто с постоянным значением управляющего воздействия.

Рассмотрим пример поворота направо «за угол». Если робот движется на расстоянии L от стены, то и поворот, очевидно, он будет выполнять по окружности с радиусом L (рисунок 16).

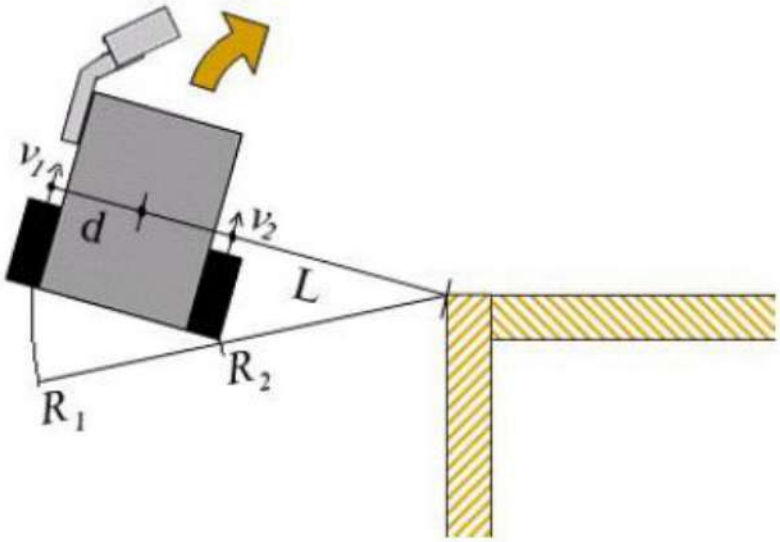

Рисунок 16 - Выполнение поворота при потере контакта со стенкой.

Нетрудно рассчитать, каким должно быть отношение скоростей колес, чтобы радиус поворота оказался равен L. Для этого достаточно измерить расстояние между передними колесами. Пусть в нашем роботе оно будет равно k =16 см, а его половина d= 16/2 = 8 см. Тогда левое и правое колеса движутся по окружностям радиусов, соответственно, Rx = L + d и R2 = L - d. Пройденные ими пути за единицу времени должны быть пропорциональны радиусам, следовательно, скорости точек крепления колес υ1 и υ2 связаны следующим отношением:

$$\frac{v_1}{v_2} = \frac{R_1}{R_2}.$$

Выражая скорости перемещения колес через базовую скорость υ и неизвестную x, а радиусы через L, получаем:

$$\frac{v+x}{v-x}=\frac{L+d}{L-d},$$

$$vL+xL-vd-xd=vL+vd-xL-xd,$$

$$2xL=2vd,\ 2xL=2vd,\ 2xL=2vd,$$

$$v_2=v-\frac{vd}{L}=v(1-\frac{d}{L}).$$

Линейная скорость υ пропорциональна угловой скорости колеса ω, которая, в свою очередь, пропорциональна мощности, подаваемой на моторы (в режиме торможения). Мы привели закон управления к стандартному виду, что позволяет задать управляющее воздействие на время поворота за угол. Таким образом, получаем расчет для управления моторами нашего робота:

```
u=50*8/L;
motor[motorB]=50+u;
motor[motorC]=50-u;
```

Когда расстояние до стены становится больше 2L (используем такой порог видимости), то есть открывается поворот за угол, управляющее воздействие начинает вычисляться по приведенным формулам.

В программе на RobotC добавим переменных, чтобы сделать ее более строгой и наглядной. При этом не требуется вводить переменную L2, поскольку здесь имеется возможность писать в условиях формулы:

```
task main()
{
  float u, k1=2, k2=10;
  int v=50, d=8, Sold, L;
  Sold=L=SensorValue[S1];
  while(true)
  {
    if (SensorValue[S1]>L*2) {
      u=v*d/L;
      Sold=L*2;
    }
```

```
        else {
          u = k1*(SensorValue[S1]-L) +
            k2*(SensorValue[S1]-Sold);
          Sold=SensorValue[S1];
        }
        motor[motorB]=v+u;
        motor[motorC]=v-u;
        wait1Msec(1);
    }
}
```

Надо заметить, что мы слегка обманываем робота, подставляя ему ограниченные сверху значения. Такой алгоритм будет стабильно работать на расстоянии L от 25 до 125 см.

Фильтрация данных.

Наконец, для окончательной стабилизации робота следует ввести защиту от помех. В силу особенностей работы ультразвукового датчика, сигнал время от времени не попадает на глазок-приемник, и в результате не всегда поступают адекватные значения, что может внести серьезные возмущения в наш алгоритм. Например, небольшая щель в стене для робота выглядит целым проемом. Поскольку требования к скорости пока что невысоки, можно несколько замедлить реакцию датчика на изменения расстояния, установив программный фильтр его значений. Простейшая реализация такого фильтра состоит в усреднении очередного показания и предыдущего отфильтрованного значения:

$$Snew = (Snew + S1)/2.$$

Среднее арифметическое несколько сглаживает резкие скачки показаний датчика. Во всех формулах вместо S1 будет использоваться отфильтрованное значение Snew. Однако и этого может не хватить при слишком резких изменениях. Тогда следует ввести весовые коэффициенты c1 и c2 для усредненной и новой величины.

$$Snew = (c1*Snew + c2*S1)/(c1 + c2).$$

Раскрываем первые скобки:

$$Snew = \frac{c1}{c1+c2} \cdot Snew + \frac{c2}{c1+c2} \cdot S1$$

Легко заметить, что сумма полученных коэффициентов равна 1. Поэтому, заменив их, соответственно, на 1 - а и а, получим:

$$Snew = (1-a)*Snew + a*S1, \text{ где } 0 \le a \le 1$$

Подбирая значение а, можно регулировать степень фильтрации. Для начала примем а = 0,2. Сравнение результатов фильтрации показаний датчика ультразвука в течение 1 с приведено на рисунке 17. Синий график показывает показания датчика, пурпурный - фильтр по среднему арифметическому, желтый - фильтр с параметром а = 0,2.

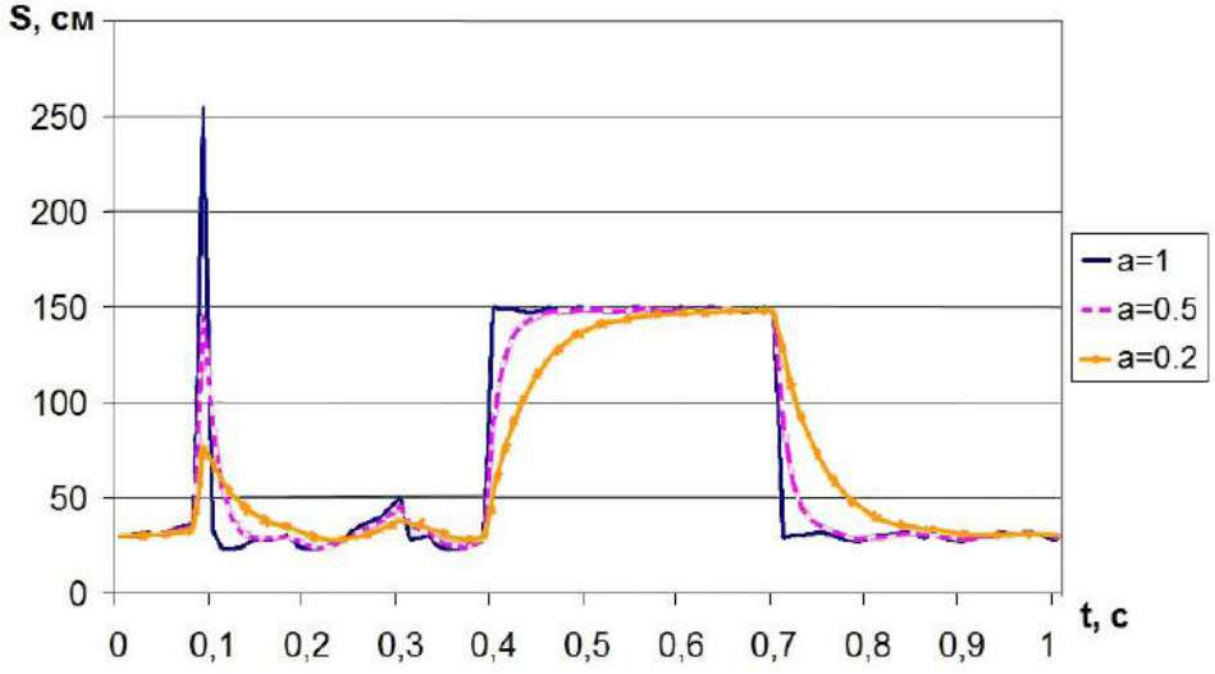

Рисунок 17 - Результаты фильтрации показаний датчика ультразвука с различными коэффициентами.

По графикам видно, что при малых значениях параметра фильтрация происходит эффективнее, и случайные скачки показаний практически нивелируются.

Перепишем программы под новые отфильтрованные показания датчика. Поначалу результат может оказаться неожиданным, однако подбор коэффициентов все расставит по своим местам:

```
task main()
{
  float u, k1=2, k2=10, a=0.2, Snew;
   int v=50, d=8, Sold, L;
   Snew=Sold=L=SensorValue[S1];
   while(true)
   {
    Snew=(1-a)*Snew+a*SensorValue[S1];
     if (Snew>L*2) {
       u=v*d/L;
       Sold=L*2;
     }

     else {
      u = k1*(Snew-L) + k2*(Snew-Sold);
       Sold=Snew;
     }
     motor[motorB]=v+u;
     motor[motorC]=v-u;
     wait1Msec(1);
   }
}
```

Фильтрация данных становится особенно актуальной, если на их основе требуется принимать решение о дальнейших действиях в долгосрочной перспективе. Например, увидев проем, остановиться или повернуть назад. Достаточно одной помехи, чтобы робот остановился не в том месте. Поэтому фильтры, хоть и затормаживают реакцию робота, но делают ее более стабильной и предсказуемой.

Заключение

В данном учебном пособии было проведено описание конструкторов компании «LEGO», таких как: LEGO MINDSTORMS NXT, LEGO MINDSTORMS EV3 и TETRIX, так как конструкторы этой компании применяются во многих робототехнических соревнованиях .

NXT является интеллектуальным, управляемым компьютером роботом на базе элементов Lego и системы mindstorms. В конструкторе NXT применены новейшие технологии робототехники: новейший 32-битный программируемый микроконтроллер, чувствительные датчики и интерактивные сервомоторы, разъемы для беспроводного Bluetooth и USB подключения, 256 Кбайт FLASH, 64 Кбайт RAM, графический ЖК - дисплей 100×64 пикселя.

Модуль EV3 — это программируемый интеллектуальный модуль, который, будучи мозгом робота, управляет моторами и датчиками, чтобы заставить его двигаться, ходить, говорить, а также обеспечивает беспроводную связь через Wi-Fi и Bluetooth.

TETRIX компании PITSCO – робототехнический конструктор нового поколения, который позволяет перевести процесс создания робота на новый качественный уровень. На его основе можно построить робота с дистанционным управлением или, используя микрокомпьютер и датчики, создать автономного робота. TETRIX является основным конструктором международных соревнований FIRST Tech Challenge.

Во втором разделе проведено описание язык программирования RobotC, так он является одним из ведущих языков программирования контроллеров компании «LEGO».

ROBOTC - это текстовый язык программирования, основанный на стандартном языке программирования C. В настоящий момент это

единственный язык программирования для роботов, который предоставляет развитый режим отладки во время выполнения программ.

ПО RobotC позволяет разрабатывать приложения для работы со следующими платформами: TETRIX, NXT, Cortex, RCX, PIC, VEX PIC, Arduino Diecimila, Duemilanove, Mega 2560, Mega 1280, Uno.

Во третьем разделе описан комплекс программ по основам робототехники на базе конструкторов LEGO MINDSTORMS.

В данный комплекс программ входит описание программ для управления мобильным роботом, движения робота по линии, «танца» робота в круге и ориентирования робота на местности: объезд стены.

Список источников и литературы

Научная литература.

1. Клаузен, Петер. Компьютеры и роботы. – М.: Мир книги, 2006.

2 . Накано Э. Введение в робототехнику.: Пер. с япон.- М.: Мир, 1988 - 334 с.

3. Предко М. Устройства управления роботами: Схемотехника и программирование / Control devices Robots: Circuitry and Programming. - ДМК Пресс, 2005. 416 с.

Учебная и иная литература

4.А. Федулеев. Руководство преподавателя по ROBOTC® для LEGO® MINDSTORMS®. - Москва, 2013. - 175 с.

5. Белиовская Л.Г. Узнайте, как программировать на LabVIEW. – Изд-во ДМК, 2013. – 140 с.

6. В.Л. Конюх. Основы робототехники: учеб. пособие для вузов по направлениям подготовки 220300 "Автоматизация технол. процессов и пр-в" и 220400 "Мехатроника и робототехника" [Текст] / В.Л. Конюх - Ростов н/Д: Феникс, 2008. - 282 с.

7. Журнал «Автоматизация и робототехника».

8. Журнал «Робототехника и техническая кибернетика».

9. Журнал «Шелезяка».

10. Журнал «Robome.ru - Робототехника и бизнес».

11. Лоуренс Вок. Книга открытий LEGO MINDSTORMS NXT 2.0. 2010. 204 с.

12. Овсяницкая, Л.Ю. Пропорциональное управление роботом Lego Mindstorms EV3 / Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий. – М.: Издательство «Перо», 2015. – 188 с.

13. Овсяницкая, Л.Ю. Алгоритмы и программы движения робота Lego Mindstorms EV3 по линии / Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий. – М.: Издательство «Перо», 2015. – 168 с.

14. Овсяницкая, Л.Ю. Курс программирования робота Lego Mindstorms EV3 в среде EV3: основные подходы, практические примеры, секреты мастерства / Д.Н. Овсяницкий, А.Д. Овсяницкий. – Челябинск: ИП Мякотин И.В., 2014. – 204 с.

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

16. Теоретический и прикладной научно-технический журнал «Мехатроника, Автоматизация, Управление».

17. Филиппов С.А. Робототехника для детей и родителей. – 3-е издание. - СПб.: Наука,2013. 319 с.

18. Юревич Е.И. Основы робототехники: Учебник для вузов. - Л.: Машиностроение , Ленингр. 1985. - 271 с.

Интернет-ресурсы

19. Группа «Робототехника с Lego Mindstorms»: https://vk.com/club19253040. (дата обращения 25.02.2016).

20. Клуб робототехники ФАПП (LEGO NXT, EV3, Arduino): https://vk.com/roboklub. (дата обращения 01.03.2016).

21. Сайт подразделения Lego Education: http://www.lego.com/education/. (дата обращения 01.02.2016).

22. Среда программирования RobotC: http://www.robotc.net/. (дата обращения 15.12.2015).

23. Сайт о роботах, робототехнике и микроконтроллерах: http://www.myrobot.ru/. (дата обращения 15.01.2016).

24. Сайт о роботах, робототехнике и технической кибернетике: http://rusrobotics.ru/ (дата обращения 27.01.2016).

25. Сайт Всероссийского робототехнического фестиваля «Робофест» http://www.russianrobofest.ru/programma-robototekhnika/ (дата обращения 02.02.2016).

26. Сайт компании PITSCO: https://www.pitsco.com/ (дата обращения 15.03.2016).

27.Сайт городского методического центра: http://mosmetod.ru/ metodicheskoe-prostranstvo/robototekhnika/uchebno-metodicheskiematerialy/ lego-konstruirovanie-i-robototekhnika/knizhnaya-polka-robototeh.html. (дата обращения 22.03.2016).

28. Сайт Большакова Александра: http://a-bolshakov.ru/index/0-125. (дата обращения 20.02.2016).

29. Форум обсуждение Lego Mindstorms (RCX/NXT/EV3): http://4pda.ru/forum/index.php?showtopic=502272&st=700. (дата обращения 25.12.2015).

30. Сайт Оренбургского клуба фанатов Lego Mindstorms, а так же Lego Technic и других конструкторов Lego: http://lego56.ru/ (дата обращения 16.02.2016).