РОССИЙСКАЯ АКАДЕМИЯ НАУК
Институт проблем машиноведения

Серия «Шаги в кибернетику»

С. А. Филиппов

Робототехника для детей и родителей

Под редакцией
д-ра техн. наук, проф.
А. Л. Фрадкова

Издание 3-е, дополненное и исправленное

Санкт-Петербург
«НАУКА»
2013

УДК 621.86/.87
ББК 32.816
Ф53

Филиппов С.А. Робототехника для детей и родителей. – СПб.: Наука, 2013. 319 с.

ISBN 978-5-02-038-200-8

Уже много лет мы читаем в книгах и газетах, слышим по радио и из телевизора, что скоро нас будут окружать умные, добрые и интересные роботы. Однако в реальной жизни роботов все нет и нет. Лишь несколько лет назад знаменитая датская компания Lego сделала роскошный подарок любителям мехатроники, роботов и других кибернетических игр и игрушек: выпустила робототехнический конструктор Lego Mindstorms NXT, который с успехом используется как дома, так и в учебе.

Эта книга одна из первых на русском языке поможет не только самому строить и программировать разнообразных роботов из Lego, но и научить этому других школьников, студентов. В ней рассматриваются основы конструирования, программирования на языках NXT-G, Robolab и RobotC, а также элементы теории автоматического управления.

В третьем издании добавлены описания усовершенствованых конструкций роботов, а также рассмотрены новые задачи: прохождение лабиринта, роботы-манипуляторы, инверсная линия и др. По-прежнему большое внимание уделено алгоритмам управления: от П- и ПД-регулятора для движения по линии до ПИД-регулятора для балансирующего робота-сигвея.

Предназначена для преподавателей кружков робототехники школ и вузов, для широкого круга читателей.

Рецензент
д-р техн. наук, проф. Б. Р. Андриевский

ISBN 978-5-02-038-200-8

ОГЛАВЛЕНИЕ

Предисловие

Уже много лет мы читаем в книгах и газетах, слышим по радио и из телевизора, что скоро нас будут окружать умные, добрые и интересные роботы. Однако в реальной жизни роботов все нет и нет. И так же часто в научно-технических журналах мы читаем о мехатронике — удивительной науке на стыке механики, электроники, компьютеров и теории управления (кибернетики). Однако и мехатронными устройствами ученые тоже что-то не торопятся нас окружить.

И вот несколько лет назад знаменитая датская компания Lego сделала роскошный подарок любителям мехатроники, роботов и других кибернетических игр и игрушек: выпустила робототехнический конструктор Lego Mindstorms. Из него можно собрать не только фантастические человекоподобные и другие роботы, не только разнообразные мехатронные устройства, но и приборы для измерения, связи, контроля и т.п. Главное же, этот конструктор позволяет легко и с удовольствием научиться самому строить такие штуковины и учить этому молодежь, начиная с возраста 8—10 лет. Следующее поколение киберконструктора, Lego Mindstorms NXT, обладает новыми возможностями: общение по протоколу Bluetooth, богатый набор бортовых датчиков, включая видеокамеры. Неужели скоро мы сами сможем окружить себя кибернетическими помощниками?

Проблема только в одном: нет пока на русском языке подходящих учебников для такого обучения. Однако предлагаемая вниманию читателя книга позволяет, кажется, решить и эту проблему. Из ее названия как раз и ясно, что она предназначена научить практической робототехнике детей и родителей. Причем учить этому, пользуясь советами опытного наставника, который сам прошел все этапы кибертворчества.

Сергей Александрович Филиппов имеет опыт руководства кружками робототехники в нескольких школах Санкт-Петербурга. Ведет семинары и мастер-классы для школьных учителей, методистов, для членов команд города на олимпиадах по роботам. Сам ездит на олимпиады и конференции со своими замечательными учениками[1]. Наверное, поэтому книга получилась и увлекательной, и поучительной, и доступной. Поучительной не только для детей и родителей, купивших конструктор, но и для учителей школ, руководителей кружков и преподавателей вузов, стремящихся помочь своим ученикам сделать первые шаги в мир техники будущего, в мир робототехники и мехатроники.

[1] В ноябре 2012 г. команда из Санкт-Петербурга Hand-Friend под руководством С. А. Филиппова в составе сборной России завоевала золотую медаль на Всемирной олимпиаде роботов в г. Куала-Лумпур, Малайзия, с проектом «Грета играет в ладушки». Вот имена чемпионов: Мария Муретова, Денис Никитин, Андрей Свечинский.

Удовольствие от чтения получат даже те, у кого еще пока нет киберконструктора: книга, как золотой ключик, откроет дверь в фантастическую страну кибернетических игр и игрушек, удивительно похожих на многие серьезные автоматические приборы и системы.

Мне особенно приятно, что часть описанных в книге идей и приемов родилась в ходе нашего совместного проекта «Киберфизическая лаборатория», начатого в 2008 г. физико-математическим лицеем №239 и кафедрой теоретической кибернетики математико-механического факультета СПбГУ под эгидой института проблем машиноведения Российской академии наук (ИПМаш РАН) и поддержанного программой президиума РАН «Поддержка молодых ученых» и федеральной целевой программой «Научные и научно-педагогические кадры инновационной России».

Среди других проектов отмечу Санкт-Петербургские олимпиады по кибернетике, проводимые с 1999 г. ведущими вузами города под эгидой ИПМаш РАН. Дальнейшую информацию об олимпиадах, книгах и других наших проектах можно найти на сайте *www.cyber-net.spb.ru*.

Желая книге С. А. Филиппова успеха у читателей, отмечу и то, что за ней должны последовать другие, поскольку она открывает серию научно-популярных книг и учебных пособий «Шаги в кибернетику», предназначенную как для школьников и студентов, так и для родителей и преподавателей. Книги серии помогут выбрать будущую профессию, а тем, кто уже сделал свой выбор, помогут сделать первые шаги на пути к профессионализму, познакомиться «изнутри» с современной кибернетикой: роботами и киборгами, оптимизацией и адаптацией, искусственным интеллектом и управлением хаосом. Девиз серии «Учись играя» означает, что книги будут нацелены не только на обучение, но и на развлечение, на воспитание новых поклонников и фанатов увлекательной науки кибернетики, которой так много предстоит сделать в XXI веке.

В серии «Шаги в кибернетику» в 2011—2013 гг. вышли следующие книги:

— В. Г. Быков «От маятника к роботу. Введение в компьютерное моделирование управляемых механических систем»,

— Р. М. Лучин «Программирование встроенных систем. От модели к роботу»,

— С. А. Филиппов «Робототехника для детей и родителей», 2-е и 3-е издания,

— «Санкт-Петербургские олимпиады по кибернетике 1999—2012».

Зав. лабораторией «Управление сложными системами»
Института проблем машиноведения РАН
доктор технических наук, профессор
А. Л. Фрадков

Предисловие автора к третьему изданию

Во третьем издании книги добавлено несколько тем, которые могут быть полезны начинающим робототехникам, улучшены иллюстрации, добавлены примеры на RobotC, а также исправлен ряд опечаток и ошибок.

Из нового отмечу следующие темы: улучшенная модель одномоторной тележки, робот для лабиринта, скоростной робот для движения по линии, робот-манипулятор, шестиногий шагающий робот, массивы и файлы в RobotC. Самые интересные алгоритмические примеры сосредоточены в главах «Алгоритмы управления» и «Задачи для робота».

Благодарю всех коллег и учеников, которые так или иначе приняли участие в работе над третьим изданием. Особенно рад выделить помощь Евгения Михайловича Сырова, благодаря содействию которого были выявлены многие неточности и опечатки.

Надеюсь, чтение этой книги будет интересным, а более всего принесут пользы практические опыты с роботами.

Пожелания и замечания прошу присылать по следующему адресу: *robobook@mail.ru*.

Глава 1. Знакомство с конструктором

Введение

В современном сознании, сформированном не одним поколением фантастов, робот представляет собой некоторый человекоподобный механизм, выполняющий полезную людям работу (или, наоборот, бунтующий и чрезвычайно опасный). Однако промышленные роботы редко похожи на людей или животных.

Само слово «робот» является существительным, обозначающим неодушевленный предмет, и мы говорим: «строим робот*ы*». Сравните: «строим мост*ы*» и «разводим слон*ов*». Но ребенку свойственно анимировать попадающую ему в руки игрушку, т. е. воображать ее подобной живому существу, одушевленной. А разве взрослым не хочется того же? Отчасти поэтому допустимы два варианта склонения.

Роботы очаровательны. Идея неживой материи, которая самостоятельно выполняет сложные задания, просто поразительна! С тех пор как роботы стали такими технологически сложными и современными, можно было бы подумать, что для их конструирования и программирования необходимы большие знания и навыки. Однако серия *кибернетических* конструкторов Lego Mindstorms делает робототехнику легкой и увлекательной как для взрослых, так и для детей.

Серия конструкторов Mindstorms началась еще в 1998 г. с робототехнической изобретательской системы (Robotics Invention System — RIS), созданной на базе контроллера RCX. Устройства вроде моторов, датчиков и микрокомпьютеров могли совмещаться с другими обычными деталями Lego для создания действующих роботов (рис. 1.1). RIS также была оснащена доступным языком программирования, который позволял самостоятельно запрограммировать действия самодельных роботов на базе RCX.

Рис. 1.1. Робот на базе RCX.

Начиная с 2006 г. с новым набором Lego Mindstorms NXT пользователи получили многочисленные усовершенствования по сравнению с RIS, делающие создание роботов еще проще и увлекательнее.

Однако конструктор NXT выходит за пределы простых усовершенствований «железа» и программного обеспечения. Новый набор открывает робототехнику для всех возрастов.

Как он может попасть к Вам в руки

Если за последнее десятилетие Вам не удалось познакомиться с RIS или другими наборами на базе RCX, не стоит огорчаться. Практически все их возможности и даже гораздо больше можно получить, используя новое поколение конструкторов — NXT. Гладкие детали от Lego Technic[1], усовершенствованные моторы с датчиками и принципиально новый контроллер — вот основные внешние отличия от коробкообразных роботов прошлого поколения.

Рис. 1.2. Наборы серии Lego Minstorms NXT с роботом Alpha Rex на обложке: слева 8527, справа 8547 NXT 2.0.

Практически в любом отделе Lego магазина игрушек есть набор Lego Mindstorms NXT с кодами 8527 или 8547 (рис. 1.2). На его обложке изображен робот, напоминающий андроида: сплющенная голова с круглыми глазками, руки без кистей, ноги с широченными ступнями и контроллер NXT вместо туловища. Забавно, но не стоит обольщаться: самое интересное будет не в этой модели Alpha Rex, которая служит в основном для привлечения внимания покупателей, а на деле не очень функциональна. Инструкцию по сборке вместе с соответствующим программным обеспечением можно найти на прилагающемся к набору компакт-диске. Но настоящее творчество начнется в тот момент, когда из тех же деталей счастливый обладатель конструктора соберет и запрограммирует совершенно нового робота, которого придумает сам.

[1] Если у Вас уже есть Lego Technic, будьте уверены: они с Lego Mindstorms дополнят друг друга.

Набор 8547 носит гордое имя NXT 2.0, хотя изменений в нем совсем немного: разработаны несколько новых деталей и конструкций, изменен состав датчиков и улучшена среда программирования для малышей. Неприятным открытием оказалось уменьшение числа шестеренок, которые так важны юному робототехнику. Недостающие детали теоретически можно приобрести у компании Lego, но в России это сделать трудно.

В Интернет-магазине робототехнические наборы будут стоить немного дешевле[1], чем в обычном. Это так называемое «коммерческое Lego», версия для дома.

Существует также версия «образовательного Lego», представленная компанией Lego Education. Такой набор найти в отделе игрушек в России нельзя. Поставки конструкторов Lego Mindstorms NXT Edu с кодом 9797 ведутся централизованно по школам через представительство Lego Education в России. Однако такой же конструктор, а также ресурсный набор к нему (с кодом 9695) можно приобрести через Интернет-магазины, которые не так давно появились в России, хотя за границей это будет существенно дешевле. К сожалению, из Интернет-магазина *http://www.legoeducation.us* доставка в Россию так называемого «образовательного Lego» не осуществляется (по крайней мере, на момент написания этой главы), поэтому, желая сэкономить, придется искать обходные пути, приобретать через посредников либо на Интернет-аукционах. Кроме того, к набору 9797 не прилагается программное обеспечение. Его можно приобрести отдельно.

Если наш читатель уже продвинутый робототехник и готов усовершенствовать конструктор, дополнив его новыми датчиками, то в этом помогут производители дополнительных устройств и расширений для Lego Mindstorms NXT: компании HiTechnic (*www.hitechnic.com*), MindSensors (*www.mindsensors.com*), Vernier (*www.vernier.com*) и др. Интернет-магазины, расположенные на сайтах этих компаний, как правило, осуществляют доставку в нашу страну. Дополнительные комплектующие теоретически можно приобрести и в Интернет-магазине Lego (*http://shop.lego.com*), но, как было сказано выше, с доставкой заказов в Россию у Lego не все гладко.

[1] В связи с прекращением выпуска набора 8527 в некоторых Интернет-магазинах остались раритетные экземпляры, цена на которые может быть завышена. Зато цена набора 8547 пока что держится стабильной.

Наборы для школы и дома

Итак, наборы Lego Mindstorms NXT продаются двух видов: для школы (9797) и дома (8527, 8547). Набор для школы (рис. 1.3) уложен в красивый белый пластиковый контейнер с двухуровневым хранилищем деталей внутри: сверху в оранжевых ячейках — основные строительные элементы; внизу — электронные элементы, колеса и некоторые другие крупные детали. На специальных карточках нарисовано, в какой ячейке сколько должно быть деталей определенного типа. Такой набор можно использовать для работы в нескольких различных группах и всякий раз в начале и в конце занятия проверять, все ли детали на месте.

Рис. 1.3. Образовательный набор Lego Mindstorms NXT 9797 (слева) и ресурсный набор 9648 (справа).

Детали набора для дома хранятся все вместе в красочной картонной коробке, и рассортировать их представляется непростой задачей. Находчивые робототехники приобретают недорого в строительных магазинах контейнеры для хранения мелких деталей, и конструктор переезжает на новое место жительства. Однако, несмотря на некоторый беспорядок, набор для дома содержит многие полезные элементы, отсутствующие в школьной версии. В связи с этим вместе с конструктором 9797 рекомендуется приобретать ресурсный набор 9695 (ранее 9648), который стоит недорого и содержит все необходимое (рис. 1.3).

Школьный набор укомплектован также некоторыми устройствами, отсутствующими в наборе для дома. И здесь тоже не все гладко. Во-первых, следует упомянуть аккумулятор Lego, который позволяет заменить шесть пальчиковых аккумуляторов или батареек, но без блока питания его использование не имеет смысла (а этот блок питания к набору

не прилагается). Во-вторых, провода-конвертеры для поддержки устройств RCX и три соответствующие лампочки. И наконец, дополнительный датчик касания, для которого, по необъяснимым причинам, не предусмотрено место в коробке с датчиками.

Ни к одному из наборов не прилагается Bluetooth-адаптер для соединения с компьютером, его надо покупать отдельно. А если решите использовать свой адаптер, будьте внимательны при установке драйверов: для соединения с NXT у Lego есть определенные требования[1]. Правда, для загрузки программ на NXT в этом нет необходимости: к каждому набору прилагается стандартный USB-кабель.

Основной состав набора: что мы купили?

Электроника

Компания Lego продает базовый набор, содержащий все основные детали системы NXT. Он включает в себя несколько электронных устройств, среди которых микрокомпьютер, датчики и моторы. Микрокомпьютер называется процессорным блоком (контроллером) NXT, и это разумный, управляемый компьютером блок, играющий роль «мозга» ваших робототехнических конструкций. Программы управляют им для получения входных данных с датчиков, для активации моторов, проигрывания звуков и многого другого. Сам по себе он является интеллектуальным компьютерным строительным блоком Lego, который дает возможность роботу Mindstorms становиться «живым» и выполнять различные операции.

Процессорный блок NXT (рис. 1.4) имеет семь основных портов, два из которых связаны с возможностью загружать на него программы. На одной стороне процессорного блока есть порт для подключения USB-кабеля. После того как кабель уже подключен, можно использовать его для закачки программ на NXT. У процессорного блока также есть встроенный Bluetooth, который делает возможной беспроводную загрузку программ и сообщение с другими процессорными блоками, мобильными телефонами, оборудованными Bluetooth, и другими BT-

[1] Поддерживаемое программное обеспечение для адаптера Bluetooth – Widcomm® Bluetooth для Windows не ниже версии v.1.4.2.10 SP5 и драйверы для поддержки технологии Bluetooth, включенные в Microsoft Windows XP с Service Pack 2 или Service Pack 3, Windows Vista или Vista Service Pack 1, Apple MacOS X 10.3.9, 10.4 и 10.5.

устройствами. Четвертый порт датчиков оснащен возможностью соединения двух контроллеров обычным проводом NXT по стандарту HS485.

LCD-дисплей на верхней панели процессорного блока может показывать тексты и рисунки, а динамик может проигрывать музыку (как мог и RCX), так же как и заранее записанные звуковые файлы. Например, вы можете запрограммировать вашего робота говорить фразы типа «Привет!» или «Как дела?» через динамик. Это свойство позволяет вывести роботов на новый уровень контакта с человеком и дает детям еще больше удовольствия от игры.

Рис. 1.4. Порты контроллера NXT.

Кнопки NXT выполняют следующие функции:

— оранжевая кнопка — включение/ввод/запуск;

— светло-серые стрелки — используются для перемещения вправо и влево в меню NXT;

— темно-серая кнопка — очистить/назад/выход.

Для управления моторами и получения входных данных от датчиков у блока NXT есть три выходных и четыре входных порта. Датчики могут быть подключены к входным портам, пронумерованным от одного до четырех, соединительными кабелями, которые также прилагаются в системе NXT. Как только датчики подсоединяются к устройству, они начинают посылать информацию об окружающей среде процессорному блоку, и эта информация впоследствии используется для воздействия на поведение робота. Моторы могут быть подключены к трем выходным портам — A, B и C — после этого они служат для того, чтобы робот ходил, поднимал предметы или проделывал многие другие движения.

Моторы NXT являются сервомоторами. Они более мощные, чем моторы RCX, поэтому позволяют создавать более сильных и быстрых роботов. У них также есть встроенные датчики вращения, которые измеряют обороты мотора (в градусах или в полных оборотах), — эта особенность позволяет делать движения робота очень точными.

Всего в стандартной системе NXT существует четыре вида датчиков: 1) касания (Touch Sensor), 2) звука (Sound Sensor), 3) освещенности (Light Sensor), 4) ультразвуковой датчик (Ultrasonic Sensor) (рис. 1.4). В версии 8547 появился новый датчик цвета (Color Sensor), который заменяет собой датчик освещенности и, кроме того, может определять цвета. Однако его быстродействие существенно ниже.

У датчиков касания есть кнопка, которая чувствует, когда на нее нажимают, отпускают или ударяют по ней. Этот датчик может быть полезен для роботов, которые должны обнаруживать препятствия или реагировать на прикосновение.

Датчик звука контролирует громкость звуков окружающей среды. Роботы могут использовать этот датчик для реагирования на голосовые команды.

Датчики освещенности выявляют интенсивность света вокруг них, и они также оборудованы красным светодиодом, так что ваш робот может определять интенсивность отраженного света. Эти датчики позволяют роботу делать множество вещей, например, оценивать уровень освещенности в помещении или двигаться по линии. В некоторых задачах могут быть использованы сразу три или четыре таких датчика.

Датчик цвета в наборе 8547 совмещен с датчиком освещенности и обладает широким спектром возможностей по определению цветовых оттенков. С помощью него можно, например, сортировать цветные кубики или шарики.

Ультразвуковой датчик измеряет время, которое требуется звуковой волне, чтобы отразиться от объекта и вернуться, для измерения расстояния между датчиком и объектом. У этого датчика много видов применения, таких как картографирование окружающей среды робота, выявление препятствий, предотвращение столкновений, выявление движения и др.

Технические параметры блока NXT

— 32-битовый микроконтроллер ARM7: тактовая частота 48МГц, оперативная память (RAM) 64 Кбайт, внешняя память (FLASH) 256 Кбайт;

— 8-битовый микроконтроллер AVR: тактовая частота 8МГц, оперативная память (RAM) 512 байт, внешняя память (FLASH) 4 Кбайт;

— беспроводной канал Bluetooth (устройство соответствует требованиям Bluetooth Class II V2.0);

— скоростной порт USB (12 Мбит/с);

— четыре порта входа, шестипроводной кабель для цифровой платформы (один из портов включает порт расширения, соответствующий требованиям IEC 61158 Type 4/EN 50 170 для использования в будущем);

— три порта выхода, шестипроводной кабель для цифровой платформы;

— графический ЖК-дисплей 100 × 64 пикселя;

— громкоговоритель — качество аудио 8 КГц, аудиоканал с 8-битовым квантованием и частотой семплирования 2—16 КГц;

— источник питания: шесть батарей типа AA или аккумулятор[1] Lego.

Детали для конструирования

Для создания корпуса робота в системе NXT имеются строительные части, какие можно было бы ожидать от набора Lego. Однако они не являются типичными деталями Lego: у большинства из них нет выступов. Как уже было упомянуто ранее, строительные детали системы NXT — серии Technic. И хотя может показаться, что придется потратить много времени, чтобы привыкнуть к конструированию с этими деталями без выступов, они дают больше гибкости и силы конструкциям.

Наряду с базовыми деталями серии Technic, такими как балки, штифты, оси, базовый набор NXT включает и другие, которых не было в RIS. Например, этот набор включает в себя два шарика Lego, поворотные диски и зубцы. Одни из этих деталей были добавлены для облегчения создания конструкций на основе серии Technic, а другие — просто для раскрытия бóльших возможностей. В наборе 8547, а также в новой версии набора 9797 v.95 добавлены дополнительные детали, которые оказались наиболее востребованы пользователями.

В общем и целом разнообразие составных частей, включенных в набор, обеспечивает вас почти бесконечным запасом конструкций роботов. Если не брать во внимание малое число крупных зубчатых колес, с 612 элементами вряд ли ощутится недостаток деталей (или идей!) для конструирования в ближайшем будущем.

Что потребуется еще?

Убедитесь, что Вы не забыли укомплектовать конструктор 6-ю (а лучше 12-ю) пальчиковыми аккумуляторами типа AA и зарядным устройством для них. Запасной комплект аккумуляторов иметь полезно, чтобы не терять время, если они сядут в самый неподходящий момент. Батарейки тоже подойдут, на них роботы будут двигаться несколько резвее, но все хорошее быстро кончается, и придется снова идти в магазин за элементами питания.

[1] Входит в комплект образовательного набора Lego Mindstorms NXT 9797.

Если говорить о выборе батареек для NXT, то по этой теме проведена масса исследований. Главный критерий в том, что приобретать стоит батарейки для высокотехнологичных устройств. По мнению автора, неплохим выбором являются: Varta High Energy (высокая длительность работы), Energizer Ultimate Lithium (наиболее стабильное напряжение, но стоят они дороже). Из самых доступных и разрекламированных вариантов можно назвать Energizer Maximum и Duracel Turbo, хотя они имеют средние показатели.

Еще потребуются гладкая светлая однотонная поверхность площадью не менее 1 м^2 (стол, щит или пол), черная изолента или самоклеющаяся пленка и разнообразные вспомогательные предметы: горки, коробки, пластиковые стаканчики, банки из-под лимонада и т.п. Кстати, картонные коробочки, в которые были упакованы детали конструктора, не рекомендуем выбрасывать — они тоже могут пригодиться.

Обзор дополнительных возможностей

В настоящий момент помимо датчиков, поставляемых в стандартном наборе, существуют также датчики «компас», датчики ускорения, гироскопические датчики, цветовые и температурные датчики, и пока вы читаете это, их выпускается еще больше. Компания Lego и компании-партнеры, такие как HiTechnic (*http://www.hitechnic.com*) или Mindsensors (*http://www.mindsensors.com*), посвящают много времени увеличению числа датчиков, работающих с NXT (рис. 1.5). С их помощью можно значительно расширить функциональность роботов.

Рис. 1.5. Датчик от Hitechnic.

Вас интересует новая электроника? Теперь с контроллером NXT могут работать почти любые сервомоторы благодаря разработке компании Mindsensors — сервоконтроллеру NXTServo. В январе 2013 г. была анонсирована новая серия конструкторов Lego Mindstorms EV3, которые совместимы с датчиками и моторами NXT, но обладают бóльшими возможностями. Однако можно быть уверенным, что с тем множеством расширений, которые были созданы для платформы NXT, еще долгие годы она будет использоваться и в учебе, и в науке, и для развлечений. Lego с партнерами осознает популярность Mindstorms и активно работает над усовершенствованием старых деталей, над новыми деталями и устройствами для пользователей, которые жаждут создавать еще более быстрых, умных и сложных роботов.

Программное обеспечение

Зарубежные разработки

Серия конструкторов Lego Mindstorms нашла своих поклонников как среди детей, увлеченных изобретательством, так и среди взрослых инженеров, занимающихся серьезными разработками. Поэтому и программное обеспечение для роботов NXT было выпущено с ориентацией на различный возраст и уровень подготовки пользователей.

Вместе с наборами «для дома» поставляется оригинальная графическая среда программирования Lego Mindstorms NXT. Версия Lego Mindstorms NXT Edu, предназначенная для школ, отличается от «домашней» только тремя буквами в названии и электронным руководством пользователя. Язык программирования системы NXT, именуемый NXT-G, — это графический, drag-and-drop язык, который является не только очень простым для освоения, но еще и мощным. Если вы использовали программное обеспечение ROBOLAB с RCX, возможно, вы обнаружите некоторую схожесть.

Однако в школах, по мнению автора, для изучения робототехники следует использовать именно ROBOLAB версии 2.9, которая поддерживает NXT. Это связано с ресурсоемкостью среды NXT-G: при достаточно широких возможностях в ней можно создавать только очень маленькие программы. Причем не на всех компьютерах NXT-G нормально заработает. Обе среды были разработаны как дополнения к высоко оцениваемому профессиональному языку программирования, называемому LabVIEW, и многим обязаны ему. LabVIEW, далеко не игрушка, используется в сложных системах сбора данных и системах управления по всему миру, служит гибким и мощным орудием для ученых и инженеров[1]. Robolab по своим возможностям существенно ближе к LabVIEW и менее требователен к ресурсам, чем NXT-G. Одним из его достоинств Robolab 2.9 можно назвать наглядность и схожесть с языком блок-схем. Приобрести его можно, например, в Интернет-магазине Lego Education по адресу *http://www.legoeducation.us*. Полноценная поддержка осуществляется на сайте *http://www.legoengineering.com*; там же следует скачивать патчи, расширяющие возможности Robolab, в том числе по работе с датчиками различных производителей.

Надо признать, что большим сюрпризом в NXT-G стало то, что его чрезвычайно просто освоить. Пользователи, у которых совсем нет опыта программирования, могут втянуться очень быстро. Lego мудро ре-

[1] В 2010 г. в России вышла книга «Программируем микрокомпьютер NXT в LabVIEW» [8], ориентированная на старших школьников.

шила включить множество инструкций и рекомендаций по программированию в программное обеспечение; они демонстрируют многие основные управляющие блоки, а также различные техники программирования, которые принесут пользу как начинающим, так и продвинутым пользователям. Графический пользовательский интерфейс так прост в обращении и интуитивно понятен, что многие, погружаясь в него, начинают экспериментировать с программным обеспечением, постигая его работу путем проб и ошибок. Поэтому, надеясь на увеличение мощностей компьютеров в будущем (объем памяти, частота процессора, размеры экрана), стоит не отвергать NXT-G и позиционировать его, как язык для начального самостоятельного освоения программирования роботов, тем более, что он поставляется вместе с конструкторами 8527 и 8547 «для дома».

Гибкость системы NXT допускает программирование и на других языках. Три наиболее общепринятых — это NBC, NXC и RobotC. NBC и NXC — свободные языки, созданные Джоном Хансеном. Оба они текстовые, а NXC похож на язык C (NXC расшифровывается как Not eXactly C — не совсем C). Их можно бесплатно скачать на сайте *http://bricxcc.sourceforge.net/nbc*. Надо признать, что эти языки не раскрывают всю мощь текстового программирования для NXT. RobotC — тоже текстовый язык, очень похожий на C, — обладает существенно большими возможностями. Продукт Carnegie Mellon University's Robotics Academy может быть скачан с *http://www.robotc.net*. Полнофункциональная 30-дневная демоверсия RobotC бесплатна, по прошествии этого срока можно приобрести лицензию за доступную сумму (80$ США).

Остановив свой выбор на трех языках — NXT-G, Robolab 2.9 и RobotC, — рассмотрим классификацию по возрасту и уровню подготовки пользователей, приведенную в табл. 1.1.

Таблица 1.1. Среды программирования роботов на базе NXT

Среда	**Язык**	**Возраст**	**Назначение**
Lego Mindstorms NXT Software	NXT-G	8—12 лет (дети и родители)	Самостоятельное изучение дома, основы
Robolab 2.9.4[1]	Robolab	8—16 лет, (дети, родители, учителя)	Изучение на уроках робототехники, использование на состязаниях роботов
RobotC for Mindstorms	RobotC	14—99 лет (преимущественно программисты)	Использование личного опыта программирования на языке C для создания роботов с широкими возможностями

[1] Robolab 2.9 с установленным патчем до версии 2.9.4.

У компании Lego Education свой взгляд на возраст пользователей конструкторов. Он выражен в диаграмме с сайта *http://www.legoengineering.com*, относящейся ко времени появления среды Lego Mindstorms NXT (рис. 1.6).

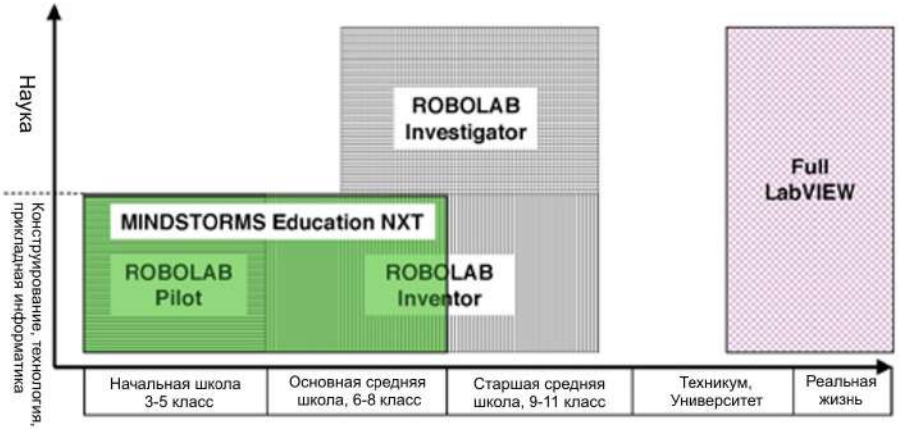


Рис. 1.6. Программное обеспечение для Mindstorms к августу 2006 г.

В 2010 году Lego совместно с National Instruments выпустила продукт LabView for Mindstorms для старшей школы, чтобы заполнить существующий на данный момент пробел между «игрушечной» средой графического программирования Lego Mindstorms NXT и «взрослой» средой LabView, которую используют инженеры (рис. 1.7).

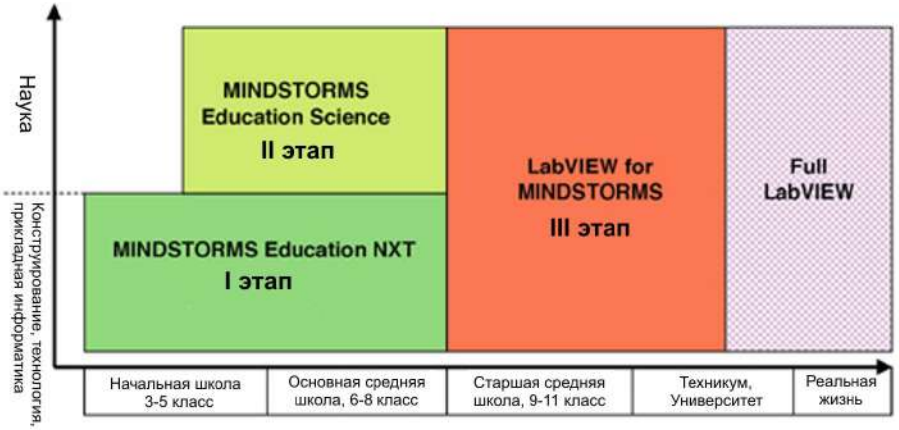


Рис. 1.7. Планы развития программного обеспечения к 2010 г.

До тех пор обновленная версия Robolab 2.9.4 была призвана временно заполнить пробел и обеспечить безболезненный переход к NXT-программированию. Однако, несмотря на появление новой версии

LabVIEW для школ, стандартом по-прежнему остается Robolab, полюбившийся пользователям за свою функциональность, простоту и наглядность. Последним подтверждением преимущества Robolab стало разработанное дополнение к образовательной версии LabView, которое полностью реализует его функционал и носит название Robolab 2.9.4d.

Обратите внимание на то, что в диаграммах отсутствует даже намек на RobotC или аналогичный язык. К сожалению, текстовые среды программирования в школах не распространены в силу всеобщей ориентации на более доступный графический интерфейс. Такая тенденция присутствует во всем. В итоге робототехникой может заниматься человек, который по сути не является программистом. В этом есть и плюсы, и минусы: с одной стороны, роботы входят в нашу жизнь, становятся реальностью, с которой необходимо считаться. Благодаря средам графического программирования можно существенно повысить общий уровень грамотности учащихся в этой сфере. С другой стороны, серьезными специалистами, скорее всего, станут только те, кто имеет глубокую алгоритмическую подготовку.

Отечественные разработки

Россия сильна своими математиками и программистами. И хотя нет пока отечественного робототехнического конструктора, но уже появилась серьезная альтернатива зарубежным средам программирования роботов, которая в ближайшем будущем сможет превзойти их по всем параметрам. Это разработка ГУП «Терком», базирующегося на математико-механическом факультете Санкт-Петербургского государственного университета. Программный продукт QReal:Robots — это среда графического проектирования, позволяющая не только быстро создавать программы, похожие на блок-схемы, но и сразу просматривать их текстовый аналог на языке Си.

Для генерации исполняемого кода используется свободнораспространяемая операционная система реального времени nxtOSEK, задача которой состоит в управлении контроллером NXT. Для специалистов nxtOSEK сама по себе интересна быстродействием и эффективным использованием ресурсов контроллера.

Для детей и преподавателей QReal:Robots интересна как многофунциональная среда, содержащая возможности графического и текстового программирования одновременно. Начиная с красочных пиктограмм, учащиеся постепенно переходят к строгому и функциональному коду на языке Си. Кроме того, QReal снабжена режимом моделирования поведения робота в виртуальной среде. Этим могут «похвастаться» разве что Microsoft Robotics Studio, RobotC Virtual Worlds и MatLab с соответствующими надстройками.

Глава 2. Конструирование

Способы крепления деталей

Если читатель уже имеет опыт конструирования на основе Lego Technic, то этот раздел можно смело пропустить и переходить сразу к разделу «Волчок». Если же подобный конструктор у вас в руках впервые, стоит изучить эту главу даже раньше, чем инструкцию Quick Start, предлагаемую Lego.

Различия принципов конструирования RIS и NXT

Хотя система NXT и была разработана, чтобы усовершенствовать RIS, многие приверженцы старой системы, как это часто случается, были сначала недовольны некоторыми изменениями. Одна из претензий была связана с новым типом строительных деталей в системе NXT (детали без выступов). В RIS большинство строительных элементов похожи на обычные строительные блоки Lego, у них есть выступы или шипы (такие маленькие круглые части, которое выпирают на верхушке деталей, придавая им классический вид Lego). Строительные детали системы NXT практически все относятся к серии Technic и не имеют выступов. Кстати говоря, вы можете услышать, как многие пользователи говорят о «гладком» конструировании, имея ввиду именно эти новые детали серии Technic! Однажды привыкнув к конструированию с использованием деталей с выступами, может быть, сложно перейти к «гладкому» конструированию. Однако, скорее всего, вы обнаружите, что на самом деле оно упрощает построение более сильных и гибких конструкций.

Конструирование с деталями Technic действительно улучшает модели; детали крепче соединяются друг с другом и благодаря разнообразию их форм NXT-наборы предлагают много новых модификаций в форме роботов.

Используя RIS, пользователи часто жаловались, что модели выглядят коробкообразно, и «квадратные» роботы были нормой, так как строились с помощью стандартных кирпичиков Lego (а они прямоугольные). Роботы NXT, напротив, могут иметь различные формы, и пользователи на самом деле получают удовольствие, создавая уникальный, никем не виданный дизайн. Кроме того, роботы NXT выглядят более похожими на реальных роботов.

Другой причиной жалоб была хрупкая природа роботов на базе RCX. Многих из них постигла неприятная судьба: в результате падения с высоты стола роботы разбивались вдребезги на дюжины или даже сотни кусочков. Этого больше не повторится! Детали NXT Technic более шершавы и крепко держатся вместе; они, конечно, разъединятся, если робот будет сброшен с достаточной высоты, но все равно скрепляются вместе намного лучше своих предшественников.

В общем и целом работа с деталями Technic достаточно легка для понимания, крепление деталей и создание новых моделей происходит довольно быстро.

Первая игра: фантастическое животное

Эта игра очень проста. Участвуют двое. Надо разделить часть имеющихся деталей на два одинаковых комплекта (чем младше участники, тем меньше деталей в комплекте). Один игрок втайне от второго строит из своих деталей некое фантастическое животное. Затем второй игрок берется за свой комплект. Оба отворачиваются друг от друга и второй игрок под диктовку первого строит копию этого фантастического животного, ни разу не взглянув на него. А первый тоже не должен видеть, что делает второй. К концу строительства обе конструкции сравниваются (рис. 2.1). Во втором раунде игроки меняются ролями.

Рис. 2.1. Не все пары получаются похожими, но что-то в этом есть.

Интереснее всего будет зрителю, который наблюдает со стороны. «Возьми серую загогулину, в которой по четыре дырочки. Прикрепи к ней оранжевый зуб с помощью синей палочки…» — это самое понятное из того, что можно будет услышать в ходе игры. Вывод напрашивается сам собой. Чтобы понимать друг друга, надо знать названия деталей. К сожалению, эти названия не прилагаются к конструкторам и доступны только в методических пособиях. Но часть из них мы все-таки публикуем в Приложении 1.

Высокая башня

Долго рассказывать о способах крепления деталей бессмысленно, надо сразу начинать конструировать. Первая самостоятельная задача, которую стоит поставить перед начинающим робототехником, — это строительство высокой башни из всех возможных деталей конструктора. Возможно, до потолка. Правда, было замечено, что при высоте более одного метра башня начинает терять равновесие, падать и разваливаться. В связи с этим стоит принять за правило не вешать микроконтроллер на саму верхушку. Это понятно: шпиль должен быть легким, а основная масса сосредоточена внизу. Для того чтобы создать устойчивую конструкцию, волей-неволей придется перепробовать многие способы крепления деталей (рис. 2.2).

Рис. 2.2. Лишь бы не упала!

Механический манипулятор

Эту игрушку можно назвать по-разному. Иногда на ее конце располагается голова клоуна, которая внезапно высовывается из потаенного ящичка на большое расстояние. Иногда она похожа на длинную свернутую трубочку — «тещин язык», которая выпрямляется с противным свистом. Иногда на конце располагается маленькая боксерская перчатка. Добавим игрушке немного функциональности и назовем ее

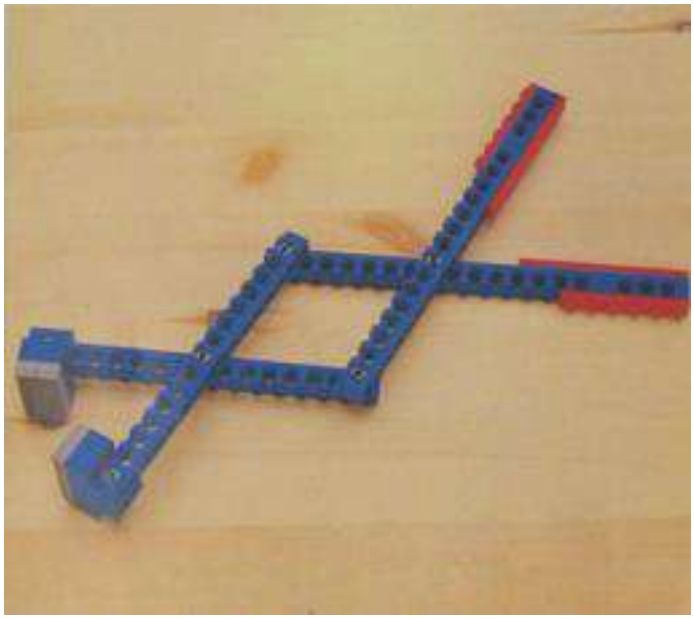


Рис. 2.3. «Хваталка», созданная из старинного набора «Простейшие машины и механизмы».

механическим манипулятором или просто «хваталкой». И постараемся сделать ее как можно длиннее (рис. 2.3).

Опишем требования к конструкции:

— хватательный механизм должен иметь минимальную длину в сложенном состоянии и максимальную в разложенном;

— у механизма должно быть две ручки, как у щипцов, и многоколенчатое соединение, ведущее к хватательной части;

— изобретатель должен суметь взять с помощью «хваталки» некоторый предмет (например, колесо из набора) и перенести его с места на место.

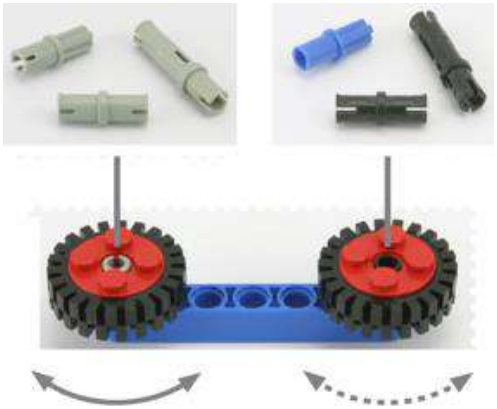


Рис. 2.4. Серые штифты и бежевые штифты-полуоси предназначены для вращения, а черные и синие – для фиксации [3].

Начальный этап создания конструкции прост: шарнирные соединения с помощью штифтов в трех

точках на каждой из используемых балок: посередине и по краям. Здесь обратим внимание на типы используемых штифтов. Часть из них гладкие, а другие имеют небольшие ребра для фиксации в отверстиях (рис. 2.4).

Конечно, в нашей конструкции лучше использовать гладкие, пусть даже трехмодульные детали, поскольку штифты с фиксатором затрудняют вращение вокруг них. Но они не блокируют движение полностью, поэтому при отсутствии достаточного числа гладких фиксирующие штифты тоже подойдут.

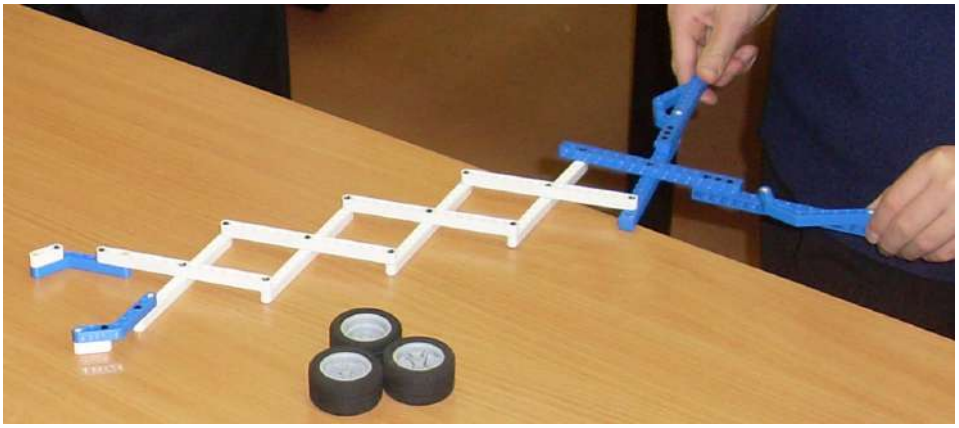


Рис. 2.5. Задача для механического манипулятора — сложить пирамидку из колес и переместить ее с места на место.

На втором этапе необходимо соорудить хватательную часть, которой будут удерживаться предметы (рис. 2.5). А третий этап — научиться пользоваться манипулятором только одной рукой. Оставляем это для фантазии читателя.

Механическая передача

Важнейшей частью почти каждого робота является механическая передача. В разных конструкторах предлагается несколько ее видов: зубчатая, ременная, цепная и др. Передача бывает необходима, для того чтобы передать крутящий момент с вала двигателя на колеса или другие движущиеся части робота. Довольно часто требуется передать вращение на некоторое расстояние или изменить его направление, например на 180 или 90 градусов.

Передаточное отношение

При всякой передаче существенную роль играет особая величина — передаточное отношение (а также передаточное число), которое надо научиться рассчитывать. Для этого необходимо знать число зубчиков на шестеренках при зубчатой или цепной передаче и диаметр шкивов при ременной передаче. На крупных шестеренках число зубцов указано: например, «Z40» на самой большой. На мелких нетрудно сосчитать их самостоятельно.

Теперь посмотрим, что происходит при зубчатой передаче. Во-первых, направление вращения ведомой оси противоположно направлению вращения ведущей. Во-вторых, можно заметить, что разница в размере шестеренок влияет на угловую скорость вращения ведомой оси. Каким образом?

Ведущая меньше ведомой — скорость уменьшается. Ведущая больше ведомой — скорость увеличивается.

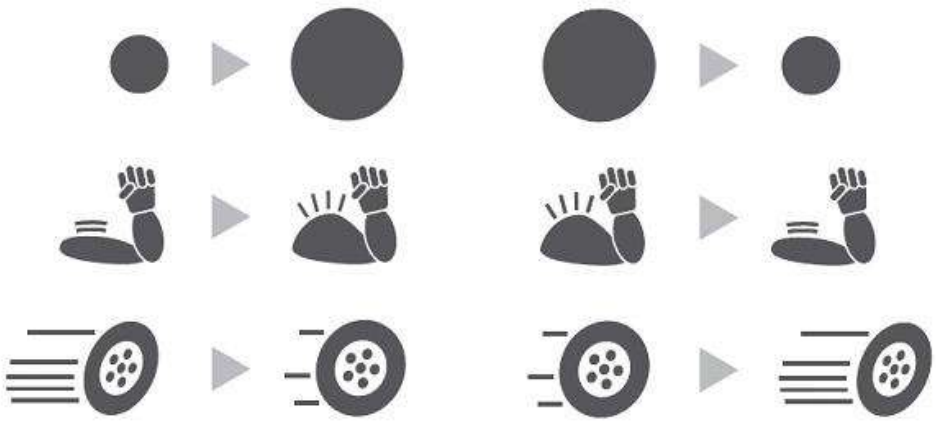


Рис. 2.6. При передаче с малого колеса на большое выигрываем в силе, но теряем в скорости. При передаче с большого на малое — все наоборот [3].

Однако надо понимать: выигрыш в скорости должен обернуться проигрышем в чем-то ином. И наоборот. Что же мы теряем при увеличении скорости? Очевидно, тяговую силу. А при понижении скорости выигрываем в силе (рис. 2.6). Это замечательное свойство зубчатой передачи используется во множестве механизмов, созданных человеком, — от будильника до автомобиля.

Как точно узнать, во сколько раз увеличилась тяговая сила? За это отвечает специальная величина, именуемая «передаточное отношение». Для нашего конструктора мы определим ее следующим образом:

$$i = \frac{z_2}{z_1},$$

где i — передаточное отношение, z_2 — количество зубцов на ведомой шестерне, z_1 — число зубцов на ведущей шестерне.

Таким образом, при $i < 1$ тяговая сила уменьшается, а угловая скорость возрастает (рис. 2.7); при $i > 1$ сила увеличивается, а скорость падает. Очевидно, что при $i = 1$ и сила, и скорость остаются прежними. В этом случае мы можем ощутить изменения только за счет потерь при трении.

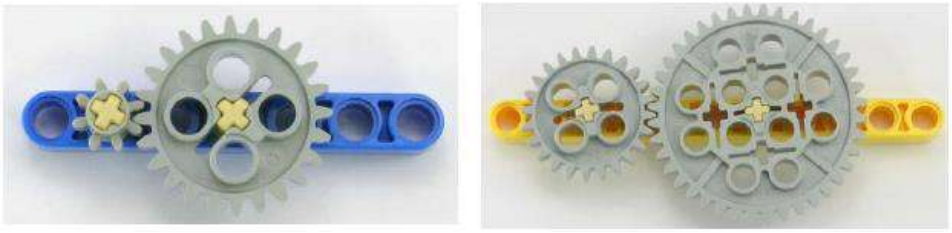


Рис. 2.7. Передача с понижением скорости: слева $i = 3 : 1$, справа $i = 5 : 3$ [3].

Если в передаче используется несколько подряд установленных зубчатых колес, то при расчете передаточного отношения учитывается только первое и последнее из них, а остальные называются «паразитными» (рис. 2.8). Паразитные шестерни исполняют полезную функцию только при необходимости передачи вращения на некоторое расстояние. В остальных случаях они лишь увеличивают потери на трение.

Рис. 2.8. Две промежуточные шестерни — паразитные [3].

Однако зубчатую передачу можно построить таким образом, чтобы каждая шестерня выполняла полезную функцию и служила либо для увеличения, либо для уменьшения передаточного отношения.

В этом случае каждая вторая пара соседних шестеренок должна находиться на одной оси. А общее передаточное отношение рассчитывается как произведение всех передаточных отношений соприкасающихся шестеренок.

$$i = i_{12} \cdot i_{34} \cdot i_{56} \cdots, \text{ где } i_{12} = \frac{z_2}{z_1},\ i_{34} = \frac{z_4}{z_3},\ i_{56} = \frac{z_6}{z_5} \ldots$$

Нетрудно догадаться, что шестеренки, находящиеся на одной оси, вращаются абсолютно одинаково и их передаточное отношение равно единице. Следовательно, эти значения в произведении могут не участвовать (рис. 2.9).

Рис. 2.9. Двухступенчатая передача [3].

И, наконец, определим понятие «передаточное число». Его используют, когда необходимо вычислить коэффициент изменения скорости или силы вне зависимости от направления возрастания. Таким образом, передаточное число можно определить как наибольшее из отношений $u = i / 1$ или $u = 1 / i$. Следовательно, передаточное число всегда не меньше единицы: $i \geq 1$. Для примера, при передаточном отношении $i = 1 : 15$, как и при $i = 15 : 1$, передаточное число $u = 15$ (рис. 2.10).

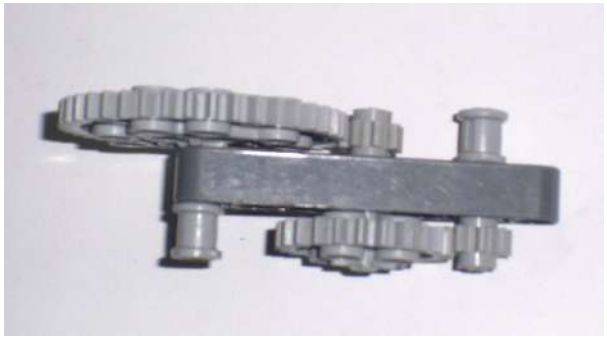


Рис. 2.10. Передаточное число 15.

Червячная передача — это частный случай зубчатой (рис. 2.11). В нашем конструкторе она обладает определенными свойствами. Во-первых, один оборот червяка соответствует одному зубцу любой шестерни. Значит, при расчете передаточного отношения число зубцов червяка можно считать равным единице: $z_ч = 1$. Во-вторых, червячная пере-

дача работает только в одном направлении от червяка к шестерне и блокирует движение в обратном направлении.

Задача. Постройте механическую передачу с максимальным передаточным отношением. По приблизительным подсчетам, из всех шестеренок конструктора 8527 можно построить передачу, увеличивающую силу вращения (и понижающую скорость) примерно в 2 млн раз. Это, конечно, теоретически. Но по сути это означает, что для одного полного оборота ведомой оси потребуется около 2 млн оборотов ведущей. Многовато.

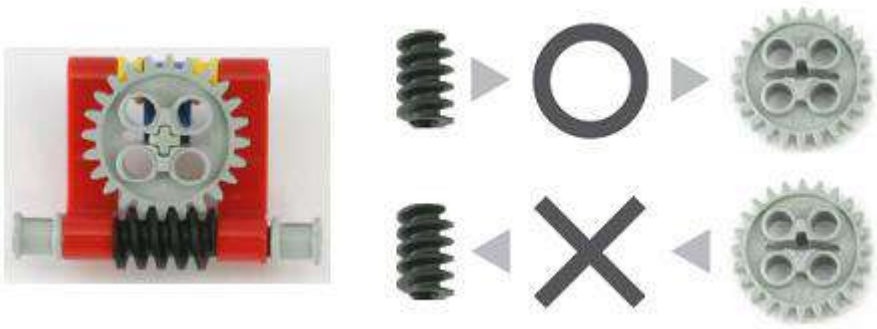


Рис. 2.11. Червячная передача работает только в одну сторону: от червяка к шестерне [3].

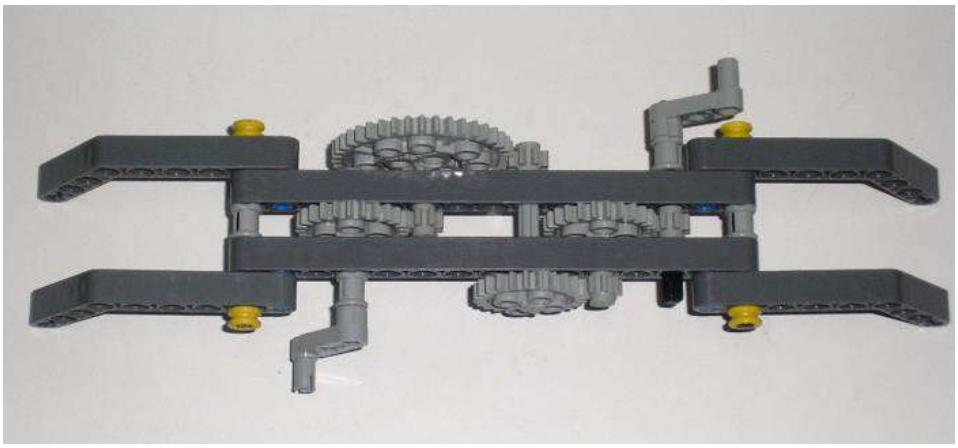


Рис. 2.12. Механическая передача с передаточным числом 135.

Для начала попробуйте построить передачи с передаточным числом 9, 27, 45, 135 (рис. 2.12). А если не получится, то поможет следующий параграф. Только в нем мы будем не замедлять, а ускорять движение.

Глава 3. Первые модели

Моторы вперед!

Следующие несколько проектов можно выполнить, не вдаваясь особо в программирование. В каждом из них достаточно будет включить один из моторов. Это делается несколькими различными способами. Составим программу, включающую мотор в каждой из трех сред, с которыми мы познакомимся подробнее в главе «Программирование», а также с помощью встроенной оболочки самого NXT.

NXT Program

Во встроенной оболочке NXT есть возможность включить моторы B и C с мощностью около 75 %, не прибегая к компьютеру (рис. 3.1). При этом в некоторых версиях оболочки (Firmware) по умолчанию требуется, чтобы были подсоединены обязательно оба мотора. В случае если хотя бы одного из них не хватает, алгоритм пробуксовывает. Однако вращение так или иначе происходит. В частности, при оригинальной прошивке Lego Mindstorms NXT такое движение будет прерывистым. Избавиться от этого можно подсоединением второго мотора.

Итак, в квадратных ячейках требуется разместить всего пять команд (см. рис. 3.1):

```
1) Forward (Backward),
2) Empty,
3) Forward (Backward),
4) Empty,
5) Loop.
```

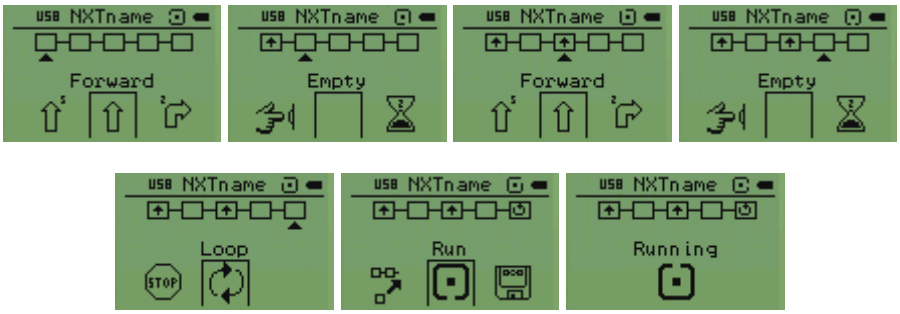


Рис. 3.1. Составление и запуск программы для включения мотора B или C.

Бывает так, что конструктивно лучше дать моторам команду «Назад». Для этого команду Forward следует заменить на Backward в обоих ячейках.

Созданную программу можно сохранить, и она появится в меню NXT Files, расположенном в разделе My Files.

Обратите внимание, что мотор A в этом случае останется неподвижным.

У этих команд управления моторами есть особенность, из-за которой моторы продолжают вращаться и после принудительного завершения программы. Остановить их можно или выключением NXT, или выполнением программы с командой Stop в последнем блоке (рис. 3.2).

Рис. 3.2. Остановка моторов с помощью команды Stop в конце программы.

Чтобы заменить команду Loop на Stop, достаточно нажать пару раз темно-серую кнопку, откатившись в режим редактирования программы, и выбрать Stop. Затем остается снова выбрать Run. Сама по себе возможность запуска одного мотора не предусмотрена разработчиками NXT Program, а найденное решение не идеально, но позволяет сэкономить немного времени.

Подробнее о встроенной оболочке NXT Program читатель узнает после конструирования двухмоторной тележки.

NXT-G

В этой среде программа, запускающая мотор A вперед, выглядит так, как показано на рис.3.3.

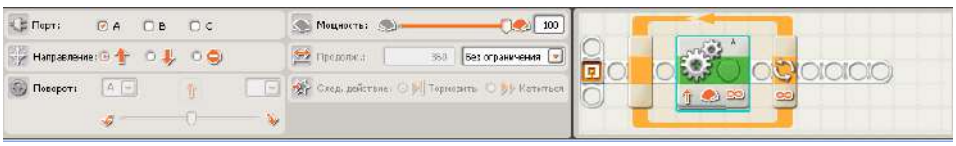


Рис. 3.3. Мотор A вперед на языке NXT-G.

Создайте цикл, который будет выполняться постоянно, а в него поместите пиктограмму «Движение». В окне свойств установите галочку напротив мотора A, задайте максимальную мощность и установите продолжительность «Без ограничения».

Загрузить программу в NXT можно, щелкнув мышкой кнопку «Загрузка» на командном центре (рис. 3.4), предварительно соединив NXT с компьютером и включив его.

Программа появится в памяти NXT в меню My Files → Software Files с именем, которое вы ей дадите в среде при сохранении файла. По умолчанию, это Untitled.

Рис. 3.4. Загрузка программы на NXT.

Robolab 2.9

В среде Robolab включить мотор A можно аналогичным способом.

В разделе «Программист» кликните дважды пункт Inventor 4 и на белом поле создайте программу, изображенную на рис. 3.5.

Рис. 3.5. Мотор A вперед на языке Robolab.

Для загрузки программы в NXT необходимо кликнуть по белой стрелочке в левом верхнем углу экрана. Если NXT ответил звуковым сигналом, значит все прошло успешно. Программа появится в меню My Files → Software Files с именем «rbl».

RobotC

Вот почти самая короткая программа на этом замечательном языке:

```
task main()
{
  while(1)
    motor[motorA] = 100;
}
```

Для загрузки программы нажмите F5, после чего, не дожидаясь звукового сигнала, ищите ее в меню NXT My Files → Software Files.

Если у вас возникнет желание работать сразу с несколькими средами, имейте ввиду, что между ними нет совместимости на уровне прошивки микроконтроллера. Поэтому при каждом переходе придется заново загружать прилагающуюся версию операционной системы (Firmware) NXT.

Тележки

Роботы призваны заменить человека во множестве жизненных ситуаций. Для этого они должны обладать различными качествами: силой, мобильностью, прочностью, интеллектом, памятью и т. д. Для каждой задачи требуются свои характеристики, поэтому никого не удивит, если одни роботы вообще не будут иметь конечностей, а другие, например, будут ездить на колесах, как автомобили. До искусственного интеллекта нам пока далеко, а первую мобильную конструкцию назовем «Тележка». Рассмотрим ее простейший вариант.

Какие требования предъявим к этому механизму:

— четыре колеса, плото сцепленные с поверхностью земли, попарно соединенные осями; колеса должны свободно вращаться;

— отсутствие других частей конструкции, соприкасающихся с поверхностью;

— наличие электромотора, приводящего в движение передние и / или задние колеса;

— наличие источника питания для электромотора и программы управления им;

— по команде оператора тележка должна двигаться вперед или назад и транспортировать полезный груз.

Одномоторная тележка

Тележка в первой версии не может поворачивать, поэтому ограничимся одним мотором с передним приводом (рис. 3.6—3.11).

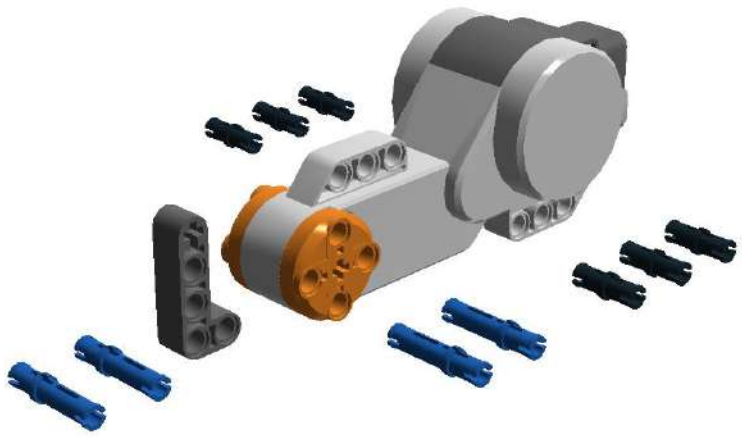


Рис. 3.6. Одномоторная тележка. Синие трехмодульные штифты вставляются в угловую балку и верхнюю дужку мотора симметрично с краю.

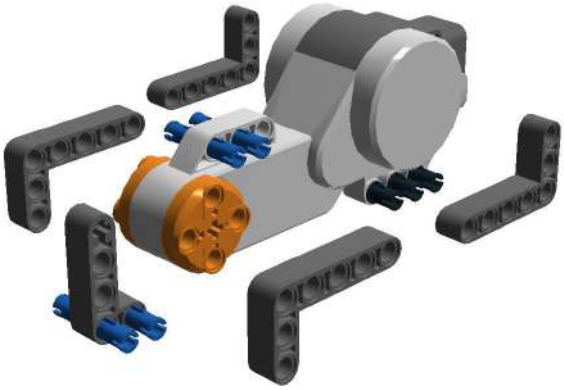


Рис. 3.7. Уголки 3 × 5 крепятся на все выступающие части штифтов.

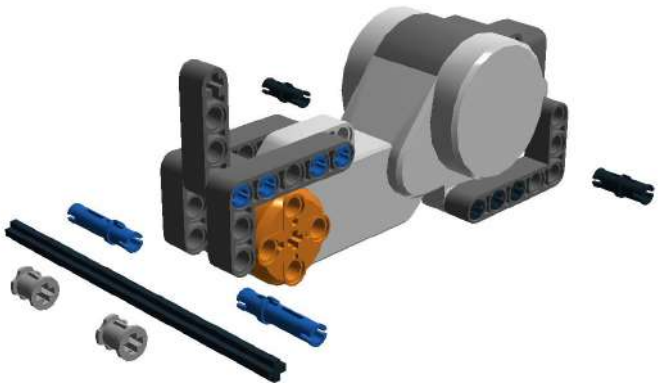


Рис. 3.8. В оранжевый диск мотора вставляется 12-модульная ось, которая будет ведущей.

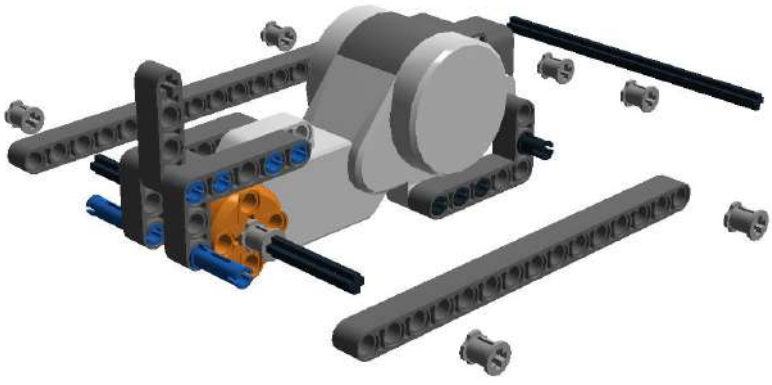


Рис. 3.9. Такая же ось сзади крепится в крайние отверстия несущих балок.

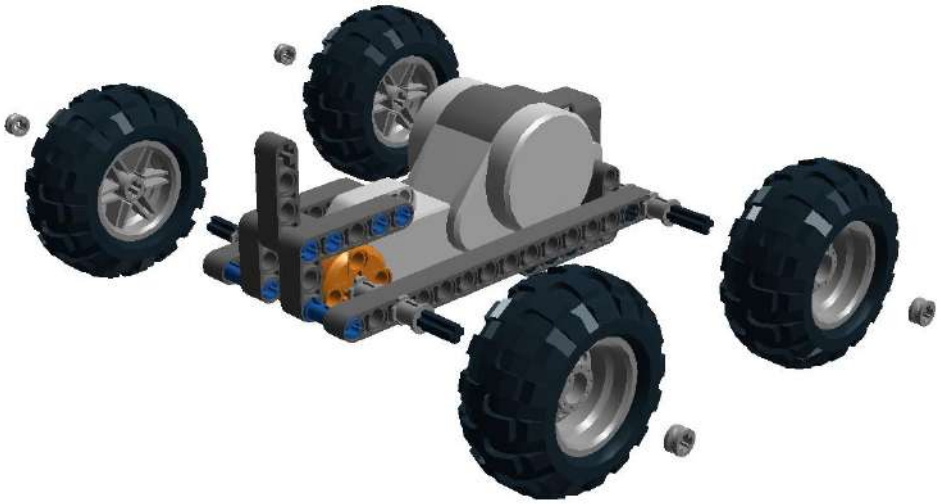


Рис. 3.10. Колеса крепятся так, чтобы не было трения с балками.

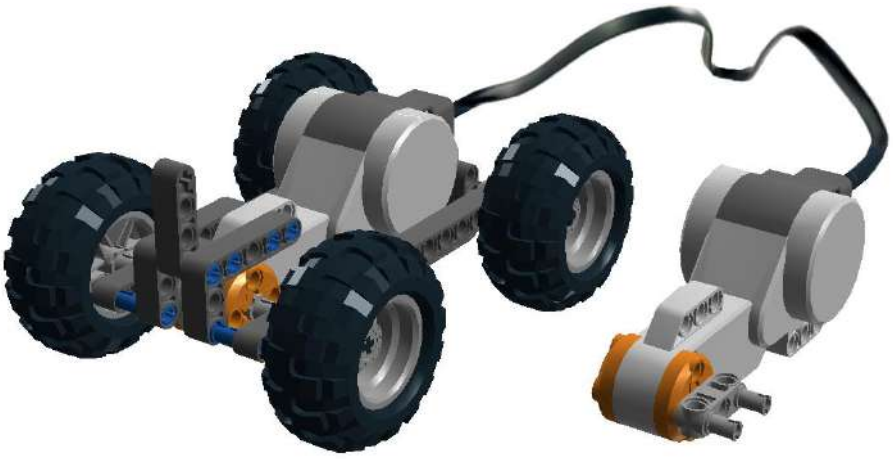


Рис. 3.11. Вращая управляющий мотор, добьемся движения тележки.

Управлять такой тележкой нетрудно, но, к сожалению, она связана с нами кабелем. Тем не менее стоит проехаться по столу, преодолеть препятствия и убедиться, что это возможно. Однако гораздо эффективнее по пересеченной местности движется тележка с полным приводом.

Полноприводная тележка

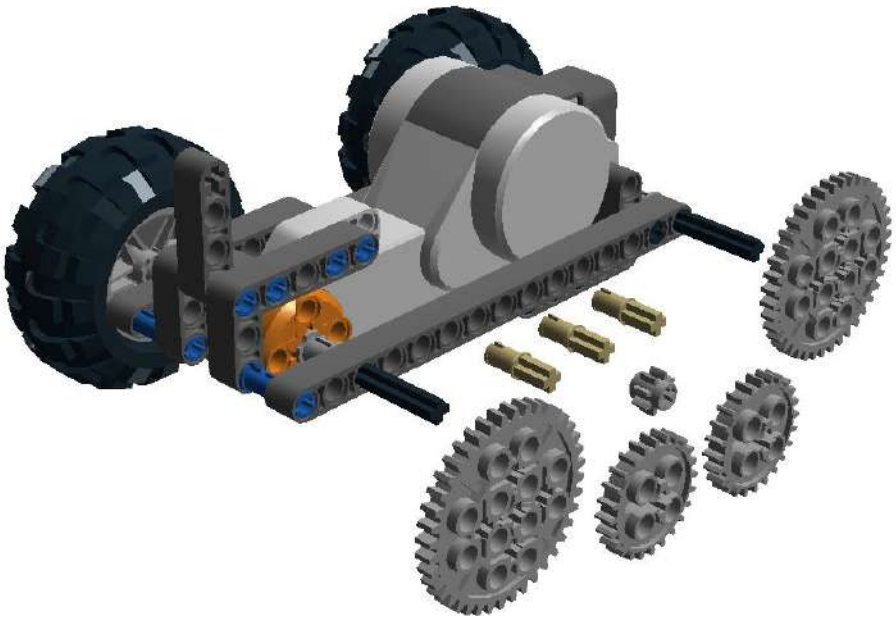


Рис. 3.12. Наиболее эффективное расположение шестеренок для полноприводной тележки.

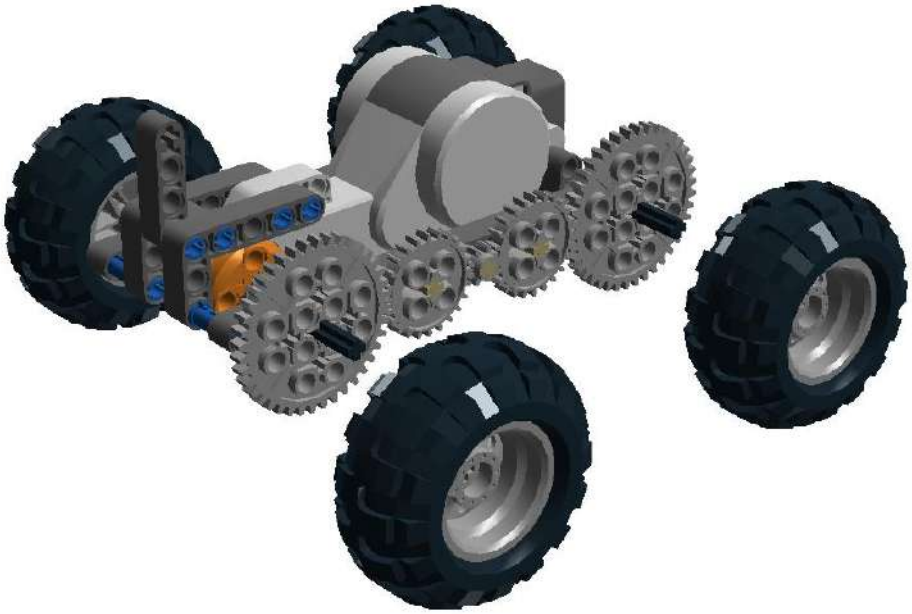


Рис. 3.13. Три из пяти шестеренок паразитные, но польза от них есть.

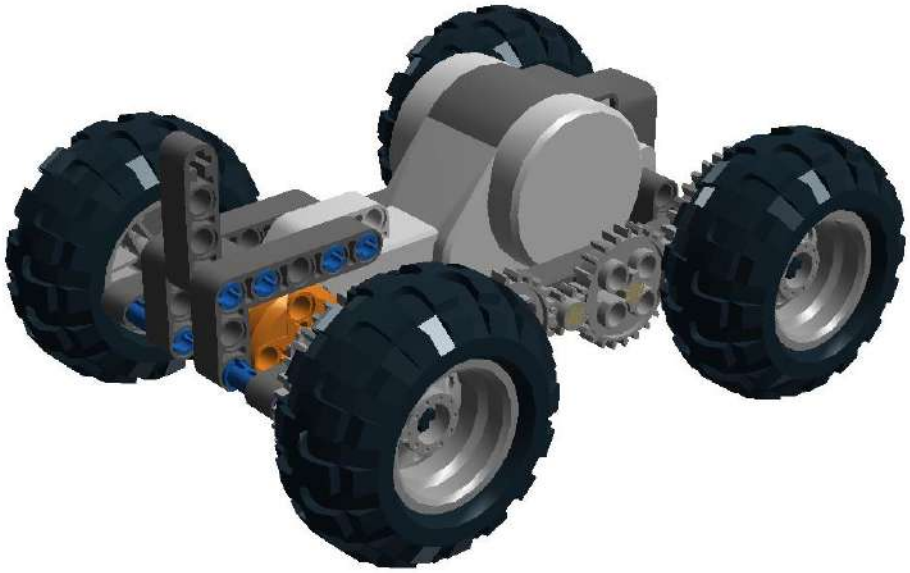


Рис. 3.14. При таком расположении колес возникает трение с соседними шестеренками.

Передаточное отношение между осями будет 1 : 1, поскольку три промежуточных шестеренки паразитные и влияют только на изменение направления вращения (рис. 3.13). Нечетное число паразитных шестеренок позволяет сохранить направление. Малая шестерня по центру не занимает пространство и не мешает преодолению бугристых препятствий.

Если колеса насажены слишком глубоко (рис. 3.14), может возникнуть нежелательное трение с соседними шестеренками, которые вращаются в противоположную сторону. Этого можно избежать: заменить 12-модульную ось двумя 8-модульными, состыковав их в оранжевом моторном цилиндре, или просто удлинить оси специальными втулками. Но об этом позже.

Итак, прототип «вездеходика» с ручным управлением готов.

Тележка с автономным управлением

Теперь давайте разберемся с полезным грузом. Для начала поставим задачу, чтобы тележка смогла вывезти сама себя. Микроконтроллер NXT с шестью аккумуляторами весит немало. Разместим его на корпусе тележки (рис. 3.15—3.20). В следующей версии массу груза можно будет довести до килограмма или двух.

Установка шестеренок и колес для полноприводной тележки уже знакома по предыдущей модели, поэтому вдаваться в детали не станем (рис. 3.21-3.23).

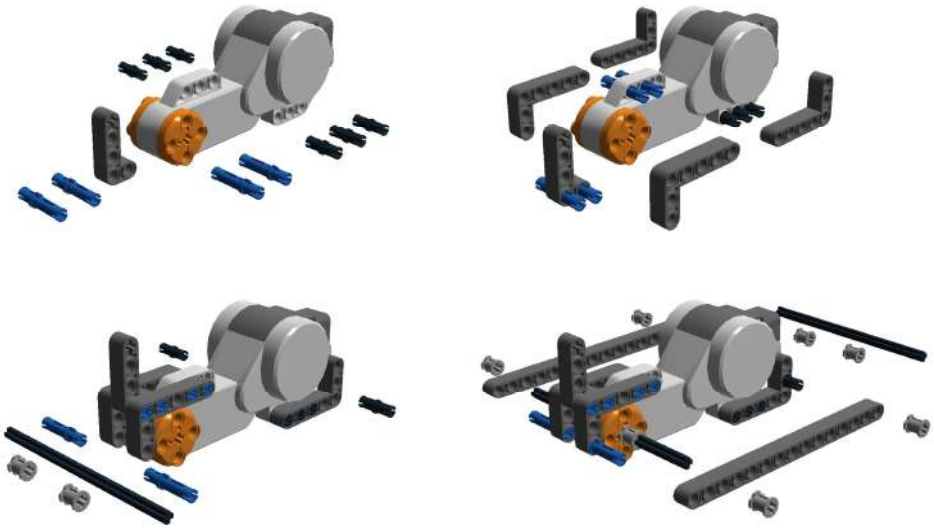


Рис. 3.15. Повторение основы тележки.

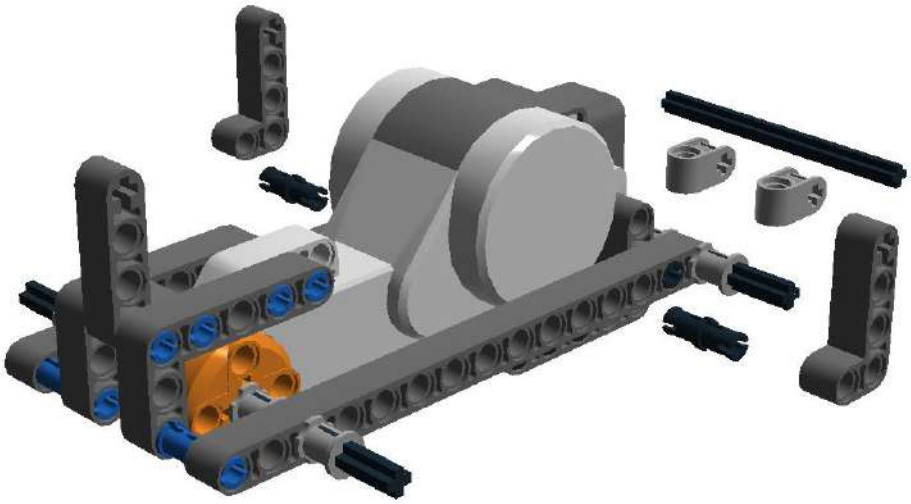


Рис. 3.16. Заднее крепление для NXT на базе одномоторной тележки.

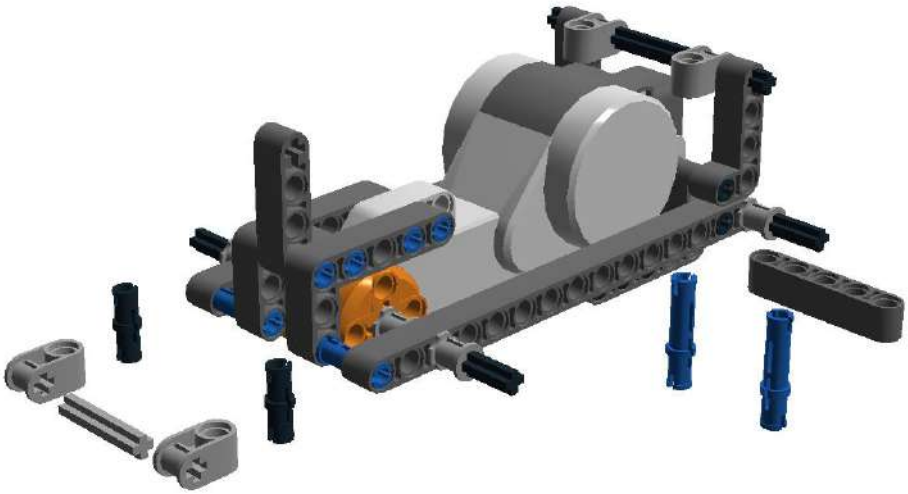


Рис. 3.17. Установка вертикальных штифтов для крепления к NXT снизу.

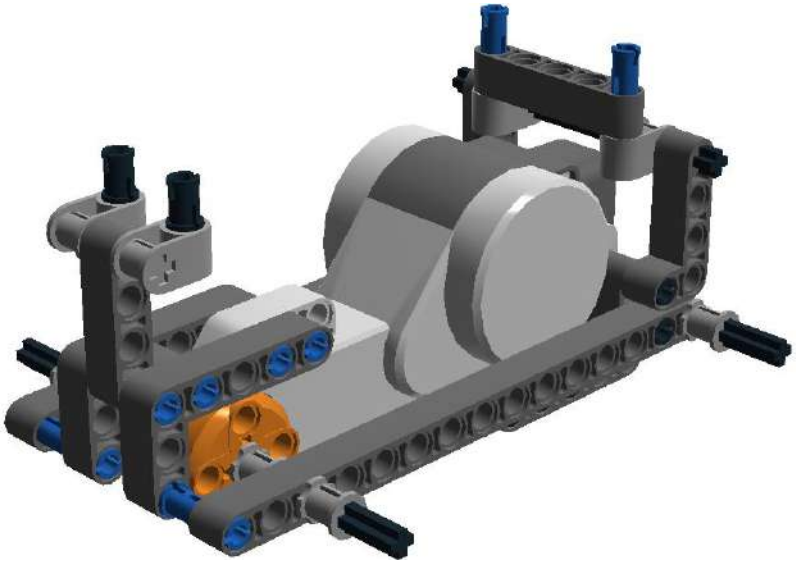


Рис. 3.18. Можно ставить контроллер.

Рис. 3.19. Для надежности колеса закрепляются полувтулками.

Рис. 3.20. Мотор подключается на порт B.

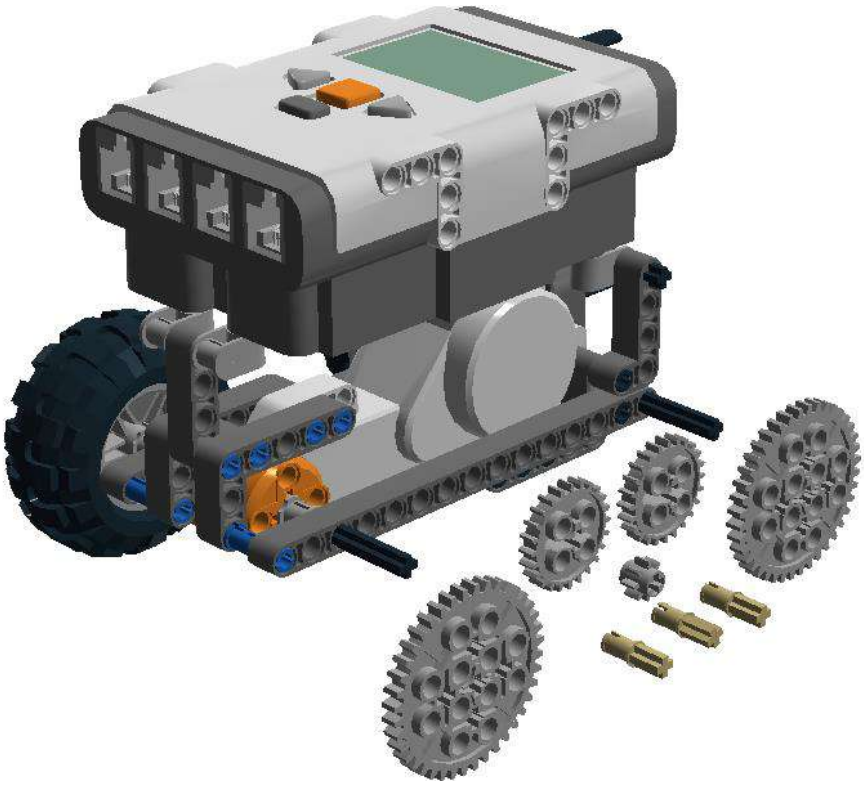


Рис. 3.21. Установка полного привода.

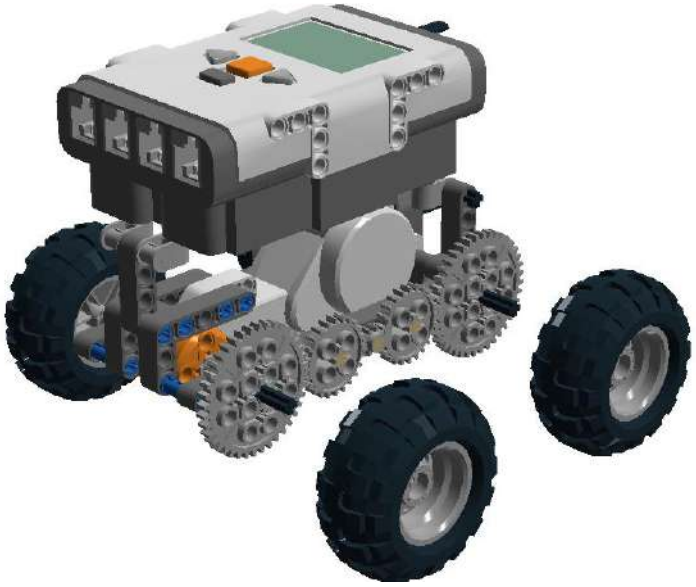


Рис. 3.22. Установка колес возможна с удлинителями осей.

Если вы до сих пор еще не начали программировать на NXT, то простую инструкцию по движению одно- или двухмоторной тележки найдете в начале данной главы. При запуске берегите пальцы – зубцы шестеренок соприкасаются с большой силой!

Рис. 3.23. Вид сбоку.

Тележка с изменением передаточного отношения

Итак, первая задача выполнена, тележка тронулась с места. Теперь попытаемся сделать из нее гоночный автомобиль. Понятно, что нагружать ее при этом пока не будем. Для увеличения скорости достаточно увеличить передаточное отношение (рис. 3.26).

Приступим к строительству. Необходимо снять с предыдущей модели колеса, шестеренки и несущие балки (рис. 3.24), модифицировав конструкцию (рис. 3.25—3.26).

Попробуйте поэкспериментировать. Можно заметить, что при достаточно высоком передаточном числе тележка просто не тронется с места: ее придется самим разгонять и подталкивать. А в приподнятом состоянии колеса будут крутиться быстро-быстро. Чего же не хватает? Очевидно, тяговой силы. Выиграв в скорости, мы потеряли в силе — моторам уже не хватает мощности для старта. Автомобилисты сталкиваются с этим при переключении коробки передач. Самое большое усилие развивается на низких передачах. Воспользуемся ими.

Двухмоторная тележка

Трехточечная схема

Это самая распространенная разновидность роботов. Тележка может быть с тремя точками опоры, две из которых — ведущие колеса, а третья — волокуша, или свободно вращающееся колесико (рис. 3.71). Такие модели являются базовыми для наборов 8527 и 9797. Инструкции по сборке прилагаеются. Если попытаетесь построить такую тележку самостоятельно, помните, что центр масс должен находиться не над волокушей, а ближе к ведущим колесам.

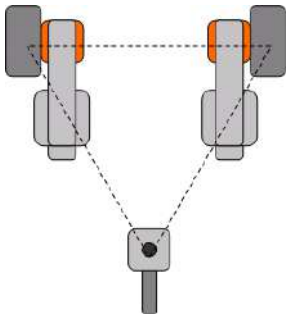


Рис. 3.71. Схема трехколесной тележки с подвижным третьим колесом.

Именно по этой схеме построена стандартная тележка из наборов 8527 и 9797 (рис. 3.72). В инструкциях этих наборов есть небольшие различия, но суть одна.

Рис. 3.72. Стандартная основа для робота из набора 9797.

Для тех, кто не хочет ограничиваться базовыми конструкциями, рассмотрим несколько примеров крепления моторов к NXT. От них можно отталкиваться при создании собственных роботов. Второй пример любезно предоставлен Центром инженерной поддержки образования на сайте *http://www.legoengineering.com* [4].

Простейшая тележка

Для придания устойчивости роботу имеет смысл поставить моторы по двум сторонам от NXT. Это несколько расширит корпус тележки (рис. 3.73).

Рис. 3.73. Широкая тележка — простейший вариант.

Рис. 3.74. Изогнутые балки для крепления моторов.

Предлагаемую конструкцию (рис. 3.74—3.81) можно делать вдвоем — бóльшая часть деталей устанавливается симметрично. А вот на подключение моторов следует обратить внимание. Для совместимости с алгоритмами, изложенными в этой книге, договоримся, что мотор B — слева, а мотор C — справа по курсу движения. На нашей тележке провода придется подсоединить накрест.

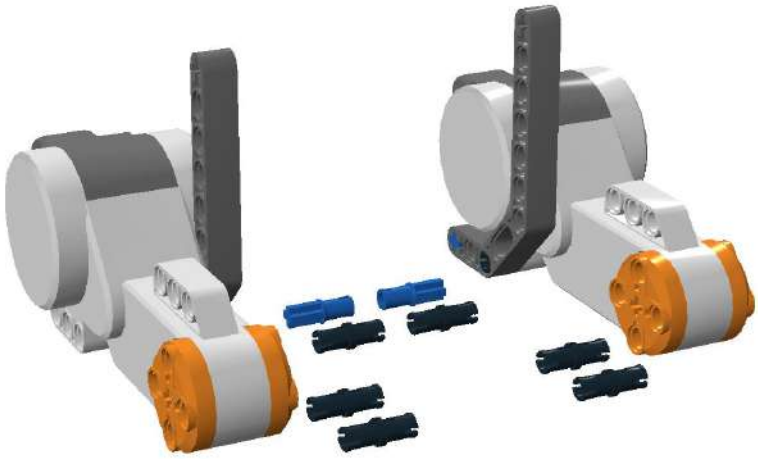


Рис. 3.75. В зависимости от расположения балок может быть смещен центр тяжести тележки.

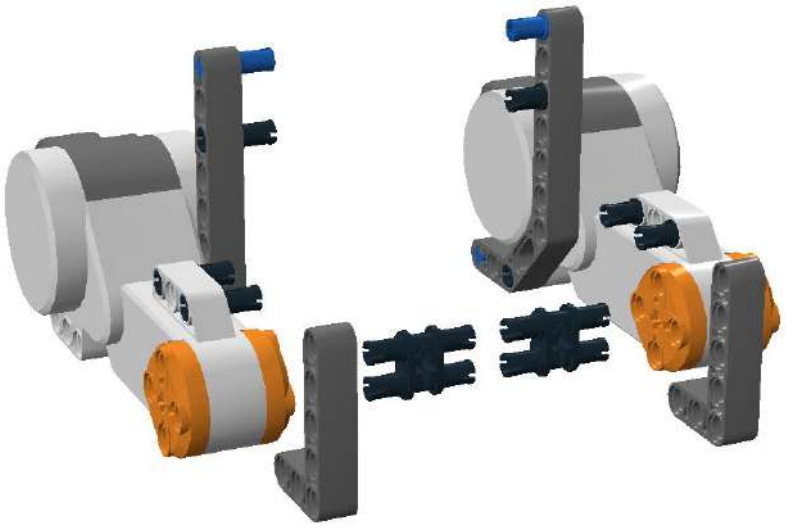


Рис. 3.76. Дополнительные крепления для придания устойчивости.

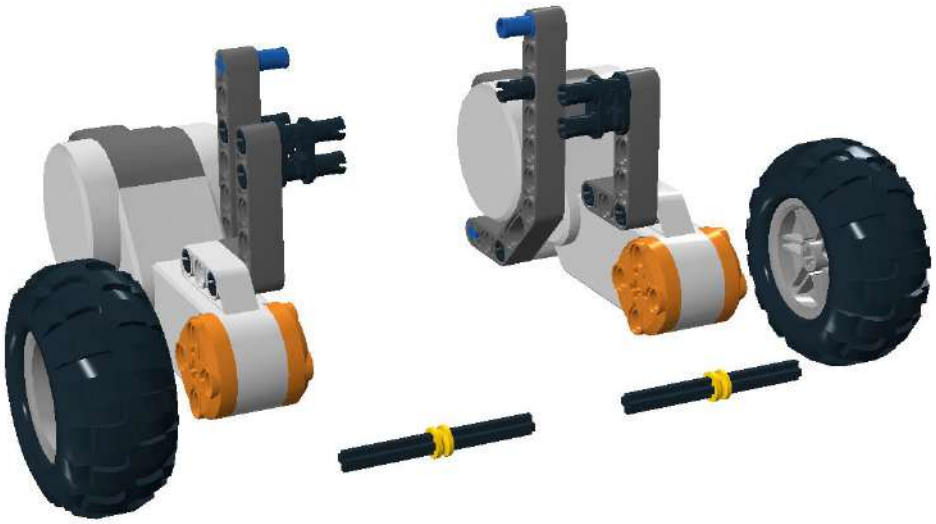


Рис. 3.77. Колеса устанавливаются на 6-модульные оси, втулки предохраняют от нежелательного трения шин о корпуса двигателей.

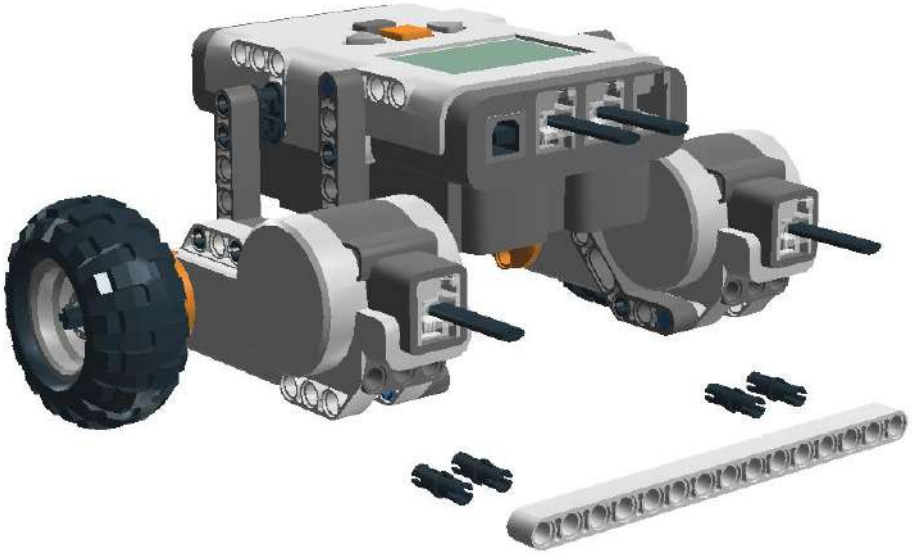


Рис. 3.78. Две половинки соединяются контроллером NXT и 15-модульной балкой, которая крепится к каждому мотору на 2 штифта.

Рис. 3.79. Элементы подвижного колеса. Длины осей — 3 и 5 модулей.

Рис. 3.80. Сборка заднего подвижного колеса. Обе оси должны вращаться свободно.

Простейшая конструкция тележки показана на рис. 3.81, однако в ней есть пара недостатков. Корпус тележки расположен с небольшим наклоном вперед. Если убрать одну втулку из вертикальной оси подвижного колеса, корпус выровняется, но тогда тележка потеряет возможность двигаться назад: колесико начнет цепляться за балку. Замена втулки на полувтулку решит проблему лишь отчасти.

Рис. 3.81. Тележка готова. Она будет слегка наклонена вперед.

Для построения строго горизонтальной тележки воспользуйтесь инструкцией на рис. 3.82—3.85. Задняя 16-модульная балка будет заменена на конструкцию из угловых балок, которая обеспечит более высокий подъем подвижного третьего колеса.

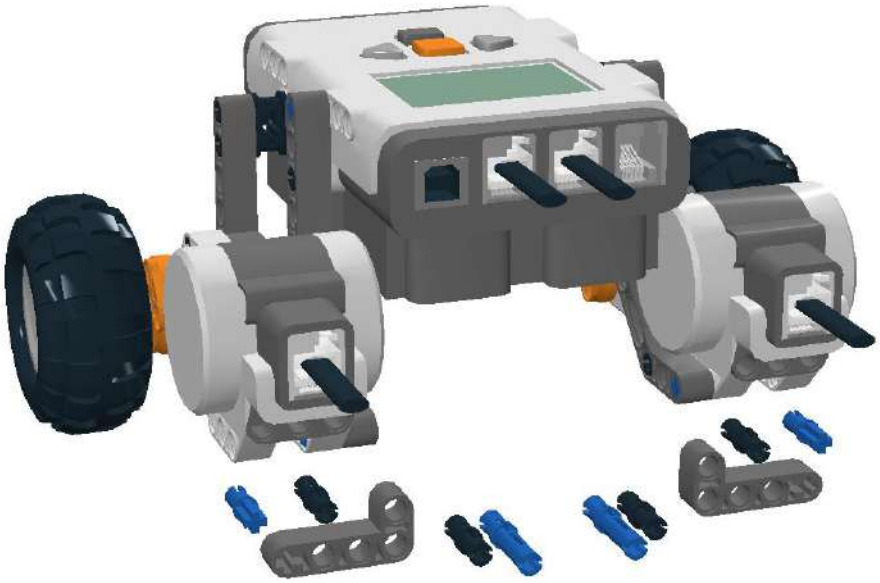


Рис. 3.82. Немного усложним конструкцию, подняв заднее крепление подвижного колеса.

Рис. 3.83. Еще пара уголков, к которым крепится задняя балка.

Рис. 3.84. Шины на диски подвижного колеса можно не надевать.

Собрав конструкцию обвязки из балок (рис. 3.92), отложите ее и соедините остальные блоки (рис. 3.93).

Далее остается прикрепить отложенную обвязку с противоположной стороны (рис. 3.94).

Компактная основа для тележки готова. Надо учесть, что при штатном Lego-аккумуляторе придется увеличить высоту вертикальных скрепляющих балок, поскольку он выступает на высоту одного модуля из корпуса NXT.

Если в наборе найдутся гусеницы, то лучшего варианта для гусеничной тележки не придумать.

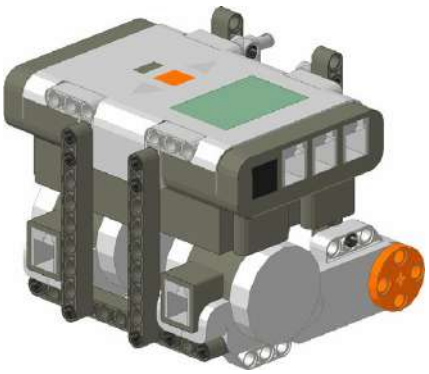


Рис. 3.94. Установка обвязки [4].

Полный привод

Можно сделать двухмоторную тележку, опирающуюся на четыре колеса, попарно соединенных с моторами. На улицах небольшого города встречаются подобные автомобили-погрузчики или электрокары. От своих собратьев с рулевым управлением они отличаются высокой маневренностью: способны выполнять поворот на месте. А от трехколесной тележки — высокой проходимостью.

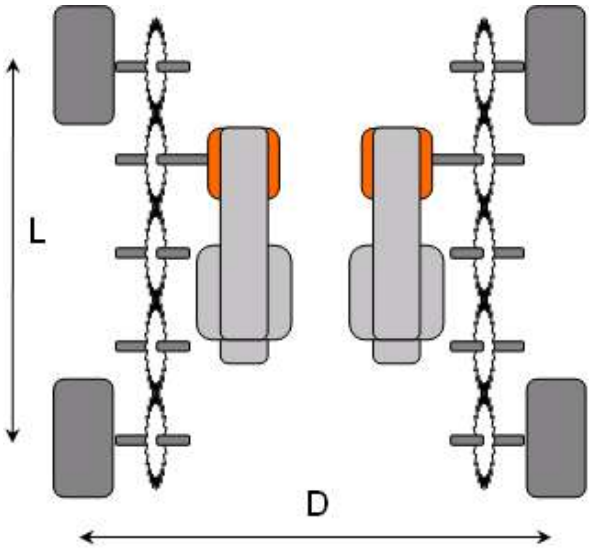


Рис. 3.95. Схема четырехколесной тележки [4].

Чтобы построить двухмоторную четырехколесную тележку с полным приводом, необходимо сконструировать механическую передачу с каждого мотора на оба боковых колеса (рис. 3.95). Если привод будет только на одно из колес с каждой стороны, то поворота, скорее всего, не будет из-за бокового трения второго колеса. Кроме того, расстояние между колесами по одной стороне L не дожно превышать расстояния между сторонами D: $L \leq D$.

Эта задача интересна, и читатель может попробовать решить ее самостоятельно, тем более что пример одномоторной тележки детально разобран. Остается расширить корпус добавлением второго мотора. Не забывайте только соблюдать четность и размеры шестеренок, чтобы колеса крутились в одну сторону с одинаковой скоростью.

Испробовав на готовой тележке небогатый арсенал NXT Program, читатель может приступить к серьезному программированию и выбрать задачу себе по вкусу: либо изучить какую-нибудь из предложенных сред (NXT-G в главе 4, Robolab в главе 5, RobotC в главе 6), либо, уже имея навыки программирования, освоить алгоритмы управления из главы 7, или сразу начать решать задачи для робота из главы 8.

Глава 4. Программирование в NXT-G

Введение

Способность NXT-робота выполнять любое задание, в чем бы оно ни заключалось, — следовать линии, бросить мяч или подметать пол, — не является интуитивной. Необходимо снабдить робота специальными инструкциями, которые будут диктовать ему, что делать; нужно запрограммировать робота. Программирование NXT включает в себя написание программы на компьютере и затем перенос в микроконтроллер, «мозг» робота, который запускает и выполняет программу. Программы должны сообщать NXT, как моторам работать, как датчикам получать информацию, как динамику воспроизводить звук и т. д.

Подходя к программированию NXT, первым делом обратим внимание на официальный язык NXT-G, включенный в пакет Lego Mindstorms NXT, поставляемый вместе с конструктором. NXT-G — это *графический язык программирования*, в котором программы можно создавать с помощью нажатия клавишей мыши и перетаскивания блоков кода на экране (G — graphical, графический). NXT-G довольно прост в использовании, но требует больших ресурсов компьютера и занимает много памяти. Далее рассмотрим еще две среды программирования NXT, которые работают быстрее, но могут оказаться менее доступны читателю.

В этой главе, во-первых, исследуем интерфейс NXT-G, который играет важную роль при создании программы. Во-вторых, обсудим некоторые базовые понятия языка NXT-G, которые необходимы для успешного программирования. В-третьих, рассмотрим готовые примеры из Robo Center.

Знакомство с NXT-G

При запуске Lego Mindstorms NXT появляется основной экран (рис. 4.1), откуда можно перемещаться к другим разделам. Начинающим настоятельно рекомендуем просмотреть краткие интерактивные руководства — Getting Started и Software Overview.

Чтобы увидеть интерфейс NXT-G и начать с ним работу, необходимо создать новую программу или открыть существующую.

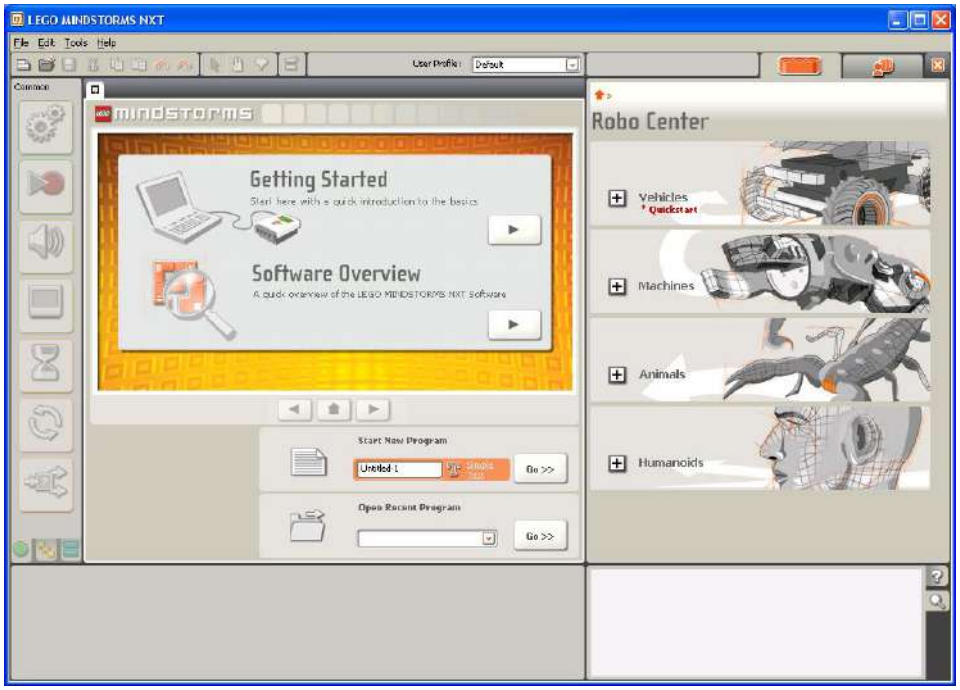


Рис. 4.1. Стартовое окно Lego Mindstorms NXT

Замечание. Предполагаем, что у читателя уже установлено программное обеспечение, поставляемое вместе с NXT-набором, и успешно проведено USB- или Bluetooth-соединение между компьютером и NXT. В случае несоответствия или сбоев операционной системы микроконтроллера следует воспользоваться разделом главного меню Tools → Update NXT Firmware.

Новая программа

Чтобы создать новую программу, напишите её название в рамке под словами *Start New Program* и затем нажмите кнопку **Go>>**. Чтобы открыть существующую программу, к которой вы недавно обращались, просто выберите программу из ниспадающего меню под словами *Open Recent Program* и затем нажмите кнопку **Go>>**.

Поскольку изначально не существует программы, напишите **First Program** в поле под словами *Start New Program* и нажмите кнопку **Go>>**, чтобы создать новую программу с именем *Test Program*.

Замечание. Можно создать новую программу, выбрав **File → New**, нажимая при этом сочетание клавиш Ctrl—N (для Windows) или CMD—N (для Mac).

Интерфейс NXT-G

Интерфейс NXT-G появляется, как только создана или открыта программа (Robo Center уменьшает обзор интерфейса, можно временно скрыть его). В результате получаем окно программы, состоящее из четырех основных прямоугольников: 1) рабочее поле программы, на котором расположен командный центр; 2) палитра команд; 3) окно настройки параметров команд; 4) окно просмотра общего вида программы (рис. 4.2). Кроме того, сверху находится главное меню и пикторамы переключения интерфейса среды, в частности пиктограмма оранжевой трехмодульной балки, по которой можно вернуться в Robo Center. Стрелкой на рис. 4.2 показана пиктограмма перехода к полной палитре команд.

Перетаскивая блоки из палитры команд на рабочее поле, можно создавать программу. При этом близлежащие блоки автоматически связываются проводами, похожими на гладкие балки серии Technic, создавая последовательность выполнения команд. Этими же балками, придерживая клавишу Shift, можно строить ответвления для параллельных задач.

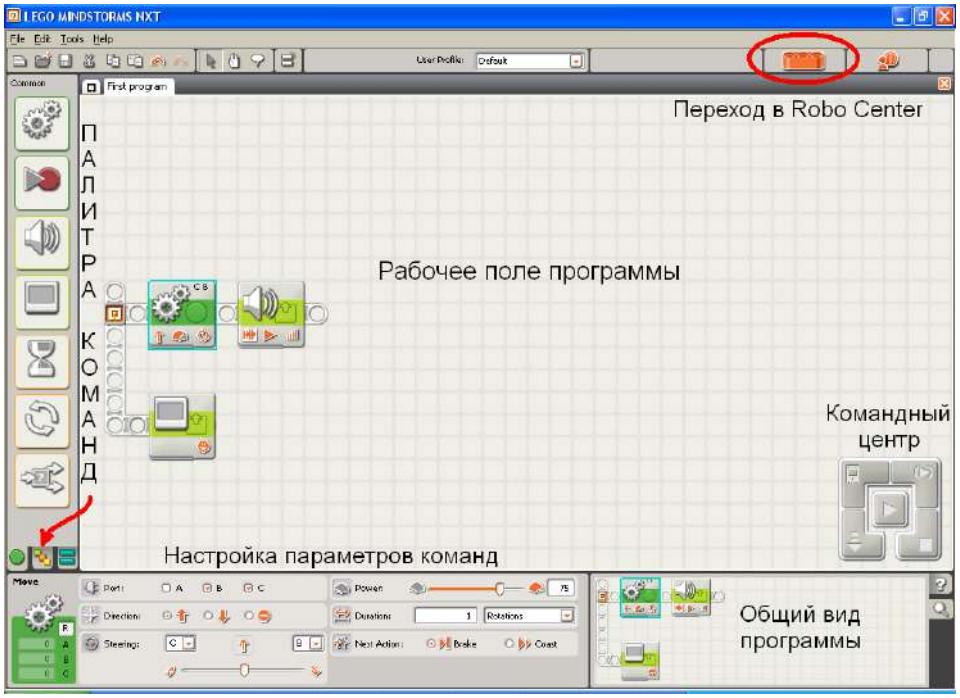


Рис. 4.2. Интерфейс NXT-G.

В окне настройки параметров команд указывается, к какому порту подсоединено устройство, а также в каком диапазоне значений и в ка-

ком режиме оно работает (для всех устройств по-разному). Внесенные параметры в виде миниатюрных пиктограмм отображаются на блоках команд в программе.

Рис. 4.3. Пиктограмма блока управления моторами.

Рассмотрим пиктограмму блока управления моторами «Move block» (рис. 4.3), параметры которого приведены в окне настройки (рис. 4.2).

Буквы в правом верхнем углу блока (1) показывают, какие из портов устройства NXT будут контролироваться. Пиктограмма «стрелка» (2) дает направление движения робота. Пиктограмма «индикатор мощности» (3) показывает уровень мощности. Скорость робота может также зависеть от прочих условий, например поверхности, по которой он движется, а также движения в гору или под гору. Пиктограмма (4) определяет значение, установленое для характеристики «Продолжительность» вращения: без ограничений, градусы, обороты или секунды.

У большинства блоков имеются концентраторы данных, которые по умолчанию спрятаны. Для извлечения концентратора достаточно щелкнуть мышкой по кнопке в левой нижней части блока, помещенного в рабочую область (рис. 4.4).

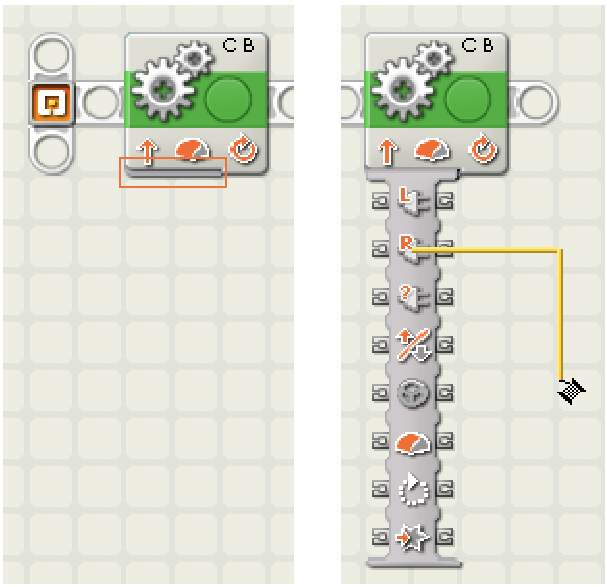


Рис. 4.4. Извлечение концентратора данных.

Как правило, шина данных создается между двумя концентраторами. На подключаемых разъемах один из них должен возвращать данные, а другой — принимать их. Если шину подключить неправильно, на несовместимые разъемы, то она будет выглядеть поврежденной, в виде пунктирной линии. Вот примеры подключении шин данных (рис. 4.5):
(A) — входной разъем,
(B) — выходной разъем,
(C) — числовая шина данных (желтая),
(D) — логическая шина данных (зеленая),
(E) — текстовая шина данных (оранжевая),
(F) — поврежденная шина данных (серая).

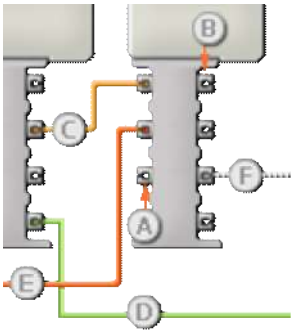


Рис. 4.5. Типы шин данных между концентраторами.

Ветвления

Есть еще несколько специфических блоков, которые стоит упомянуть. Первый из них «Блок принятия решений», или «Ветвление».

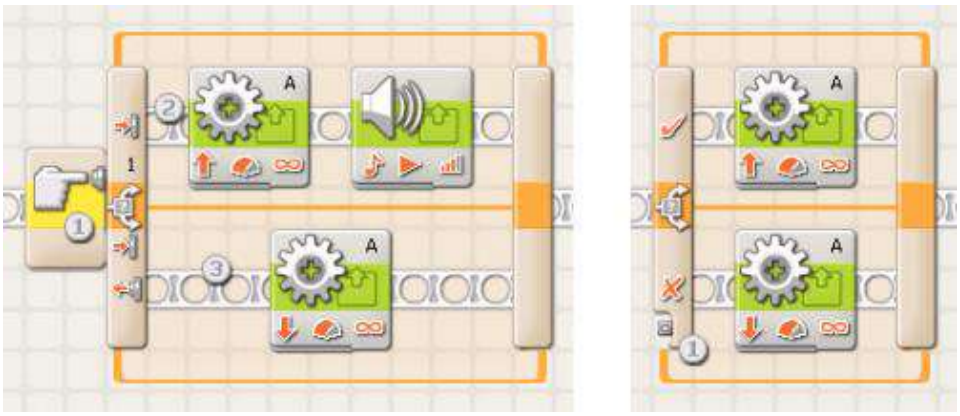


Рис. 4.6. Ветвления: по датчику нажатия (слева), по логическому или числовому значению (справа).

В приведенном примере (рис. 4.6, слева) в зависимости от состояния датчика нажатия (1) выполняется либо верхняя ветвь (2 — если нажат), либо нижняя (3 — если отпущен).

Любое ветвление с помощью окна настройки параметров можно перенастроить на другое условие. Например, можно принимать решение на основании некоторого значения (рис. 4.6, справа), подаваемого на специальный разъем. В этом случае значение должно поступить по логической или числовой шине данных от какого-то другого блока.

Ветвление может отображаться в двух режимах: 1) когда видны обе ветви (и «Да», и «Нет»), как на рис. 4.6; 2) с отображением только одной из выбранных ветвей, как на рис. 4.7. Это может быть полезно для экономии места на экране, которого в NXT-G катастрофически не хватает. Режим определяется установкой флажка «Flat view» в левой нижней части окна настройки параметров.

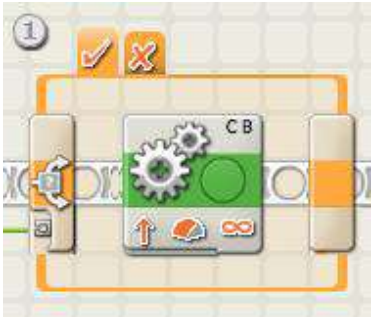


Рис. 4.7. Сокращенное отображение ветвления.

Циклы

Следующий блок, с которым стоит познакомиться, это цикл. Как правило, в NXT-G используются циклы либо с параметром, либо с постусловием.

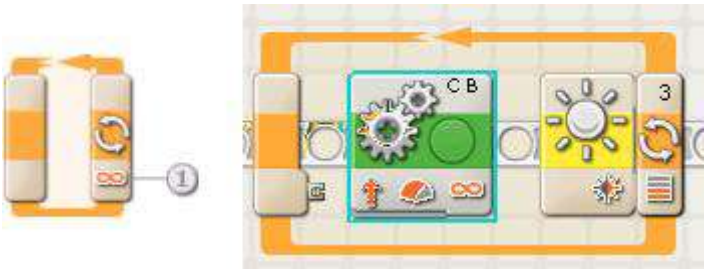


Рис. 4.8. Пиктограмма бесконечного цикла (слева), цикл по датчику освещенности (справа).

В качестве условия в правом нижнем углу пиктограммы (1) указывается режим повторений, в примере слева цикл работает бесконечно (рис. 4.8). Кроме того, циклы бывают с фиксированным числом повторений, по значению таймера, значению датчика, значению переменной.

В приведенном примере (рис. 4.8, справа) моторы B и C работают до тех пор, пока на датчике освещенности на 3-м порту не будет показано значение больше 50. При этом включен режим вывода значения счетчика повторений, и программист может подсоединить шину данных к разъему в левом нижнем углу пиктограммы цикла.

Переменные

Для использования переменных в NXT-G предусмотрен блок Variable, находящийся в палитре Data (рис. 4.9).

К единственному разъему (1) у него можно подключить шину данных, по которой будет передаваться значение (рис. 4.10).

Направление передачи на чтение или запись (2) и значение переменной (3) настраиваются в параметрах блока. По умолчанию присутствуют всего три переменных трех типов (1): логика, число и текст.

Программист может создать свою переменную, воспользовавшись пунктом главного меню Edit → Define Variables.

Рис. 4.9. Пиктограмма переменной.

Рис. 4.10. Настройка типа переменной.

Для того чтобы самостоятельно начать программировать, этого уже достаточно. Если же читатель заинтересуется глубже, он может заглянуть в полную палитру, добавить дополнительные блоки, создать свои блоки и многое другое. Среда языка NXT-G обладает массой интересных возможностей и представляется весьма полезным инструментом для изучения начинающими программирования NXT-роботов. Если же еще не появилась уверенность в своих силах, милости просим в Robo Center!

Robo Center

Раздел Robo Center содержит четыре модели, разработанных компанией Lego специально для пользователей NXT. Перейдя к любой из них, можно получить исчерпывающее руководство по сборке и составлению простейших программ для этих моделей. Эти разделы стоит пройти любому счастливому обладателю конструктора, поскольку в них не только можно найти неплохой самоучитель, но понять некоторые принципы, заложенные создателями Lego Mindstorms NXT в свое детище.

Собирая модель по инструкции, не забудьте раскрыть окно Robo Center на весь экран, чтобы лучше видеть мелкие детали и соединения.

Глава 5. Программирование в Robolab

Введение

Robolab 2.9 — это многофункциональная графическая среда программирования, созданная на основе LabView 7.0 и ориентированная на самые разные возрасты — от дошкольников до студентов. Текущая версия Robolab позволяет программировать несколько типов микроконтроллеров — Control Lab, RCX, NXT, также проводить независимые расчеты на компьютере. Следует заметить для тех, кто работал с RCX, что новая прошивка (Firmware) для него работает в 100 раз быстрее, правда загружается также долго (4—5 мин). Загрузка прошивки в NXT длится не более минуты. Следует обратить внимание на дань прошлому со стороны разработчиков: некоторые пункты меню содержат название RCX, но подразумевают также и работу с NXT.

При запуске Robolab предлагает три уровня работы: «Администратор», «Программист» и «Исследователь» (рис. 5.1).

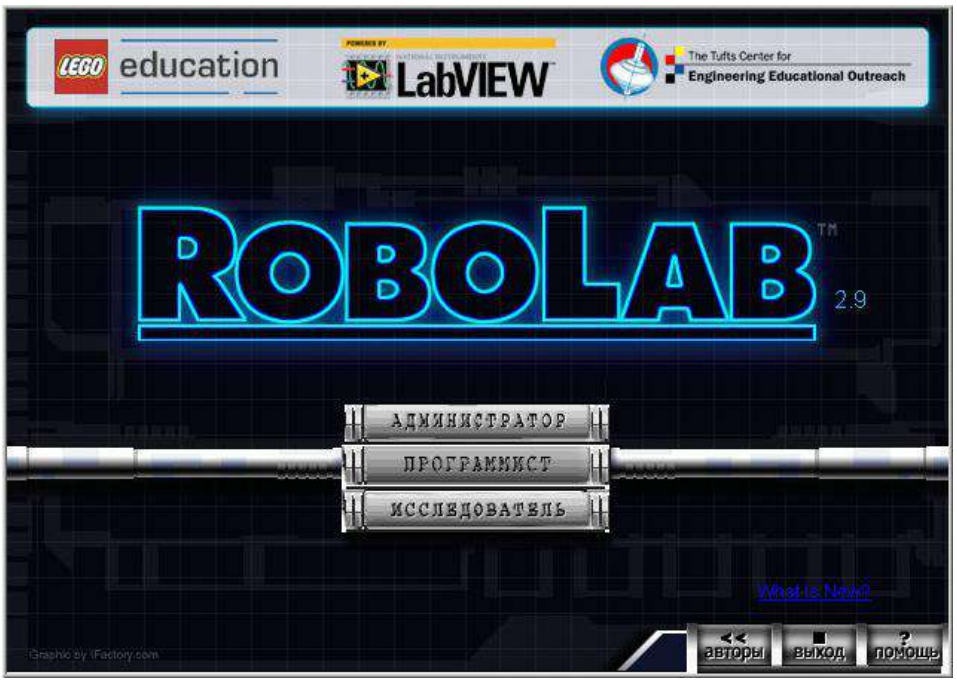


Рис. 5.1. Окно запуска Robolab 2.9.

Режим администратора позволяет настраивать контроллер на работу со средой.

Режим программиста позволяет непосредственно создавать программы и загружать их в микроконтроллер.

Режим исследователя позволяет осуществлять запись данных, поступающих с датчиков микроконтроллера, с их последующим анализом. Рассмотрим два предыдущих режима подробнее.

Режим «Администратор»

Первое, что следует сделать администратору — выполнить «Проверку связи с RCX» (подразумевается NXT). Есть три варианта исхода: 1) успешное завершение со звуковым сигналом на микроконтроллере; 2) сообщение о необходимости сменить операционную систему (ОС NXT или, иначе говоря, Firmware); 3) сообщение об ошибке связи.

Рис. 5.2. Соединение NXT с компьютером через USB.

Если связь компьютера и NXT не появилась сразу, следует воспользоваться меню «Select COM Port» для выбора USB-порта, выбрать автоопределение, выключить NXT, убедиться, что он хорошо соединен, и снова включить (рис. 5.2).

Рис. 5.3. Окно выбора устройства, с которым будет установлена связь.

Для поддержки новых возможностей NXT дополнительные опции добавлены в таблицу «Установка RCX/NXT». Здесь можно выбрать имя NXT, имя файла загружаемой в NXT программы. По умолчанию используется имя rbl. Допукается использовать до шести символов или можно задать его из программы с использованием расширенного NXT-светофора (блок NXT begin).

Поскольку Robolab поддерживает несколько устройств (RCX, NXT, Control Lab), диалоговое окно «Choose Hardware» добавлено в установки автоопределения. Оно может появляться в разных ситуациях при потере связи с NXT, и этого не надо бояться (рис. 5.3). Надеюсь, выбор будет очевиден.

Режим «Программист»

Раздел программиста делится на два: Pilot и Inventor, что не совсем точно переведено как «Управление» и «Конструирование» (рис. 5.4).

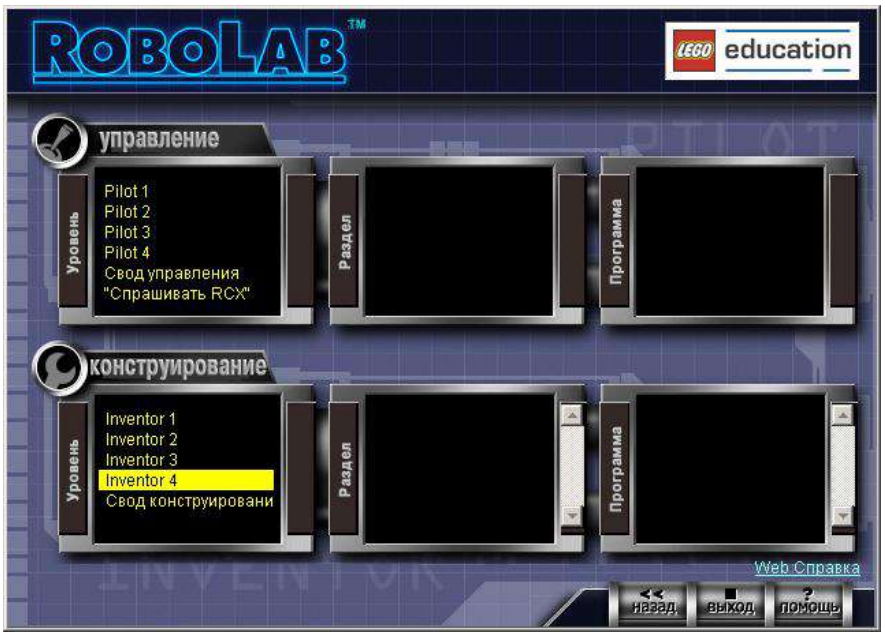


Рис. 5.4. Выбор режима программирования: двойной щелчок по Inventor 4.

Следовало бы назвать эти разделы «Новичок» и «Изобретатель». Можно было бы пройти их последовательно, постепенно окунаясь в среду графического программирования. Однако нет препятствий для того, чтобы начать сразу с последнего уровня Inventor 4, в котором

представлены все основные возможности программирования среды Robolab. Для перехода на этот уровень необходимо дважды щелкнуть по надписи Inventor 4, не касаясь подразделов.

Основные окна

Итак, мы зашли в среду программирования (рис. 5.5). На экране два основных окна, относящихся к одному проекту: Front Panel и Block Diagram. Первое (передняя панель) для программирования не пригодится, хотя его можно использовать в режиме исследователя. Второе, в котором уже расположены две пиктограммы светофоров (рабочее поле программы), предназначено для составления программы. Его стоит распахнуть на весь экран и приступить к работе.

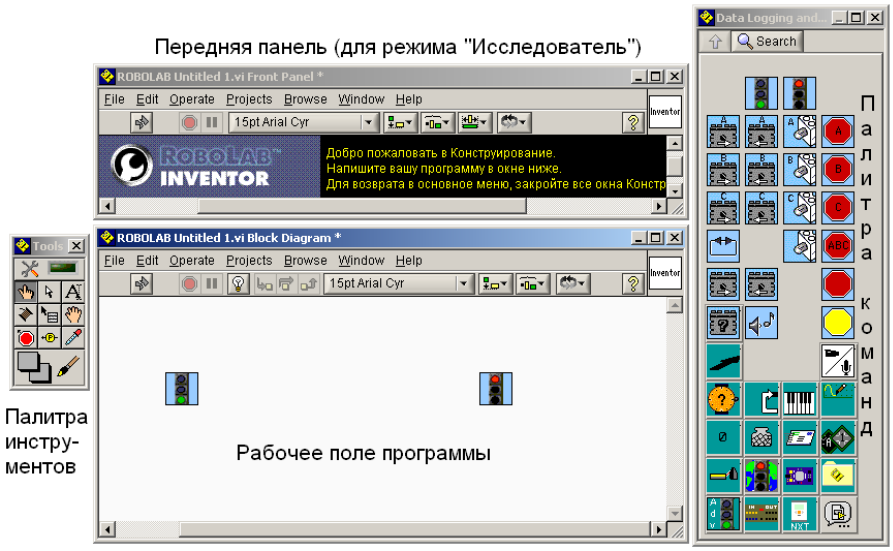


Рис. 5.5. Окна среды программирования Robolab.

Два вспомогательных окна — Tools Palette и Functions Palette — содержат все необходимое для составления программы. В случае закрытия их снова можно вывести на экран через пункт Windows верхнего меню.

Программа в Robolab похожа на блок-схему, положенную на левый бок. Она читается слева направо, хотя блоки располагать можно как угодно. Блоки команд находятся в окне Functions Palette (палитра команд). Они связываются между собой проводами (рис. 5.6), а также управляются инструментами, находящимися в меню Tools Palette (палитра инструментов).

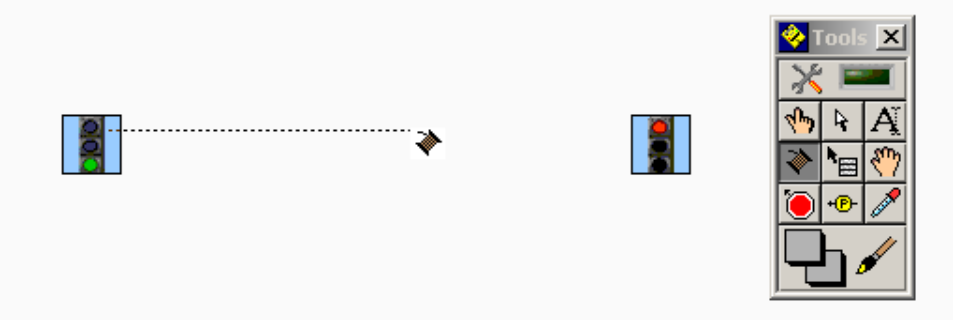


Рис. 5.6. Соединение блоков в окне программы.

Название любого блока в Functions Palette можно прочитать в верхней части окна, подведя к нему курсор. Кроме того, под заголовком окна находится кнопка Search (поиск), с помощью которой можно найти пиктограмму по названию.

Некоторые из пиктограмм сами являются палитрами, и при переходе открывается новое окно (рис. 5.7). Вернуться в предыдущее можно по стрелочке, расположенной в левом верхнем углу палитры рядом с кнопкой Search.

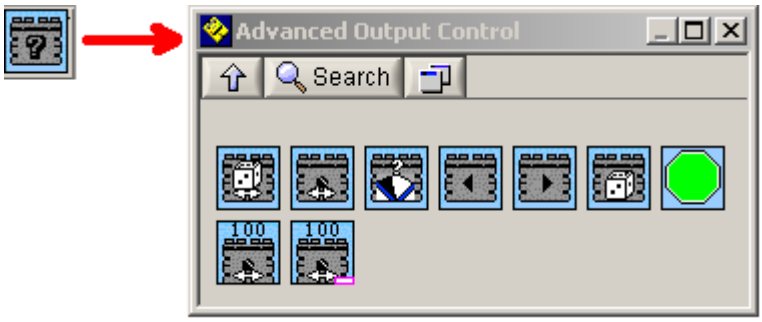


Рис. 5.7. Дополнительная палитра команд.

Готовые примеры программ

Одним из новшеств Robolab 2.9 является палитра примеров Behaviors (рис. 5.8). Это блоки, содержащие в себе готовые части программ под определенные задачи. Например, блок Go Straight (двигаться прямо) запускает моторы A и C вперед и через 1 секунду выключает их (рис. 5.9, слева). Однако для запуска этого фрагмента программы его надо обязательно поместить между светофорами, соединив их розовыми проводами (рис. 5.9, справа).

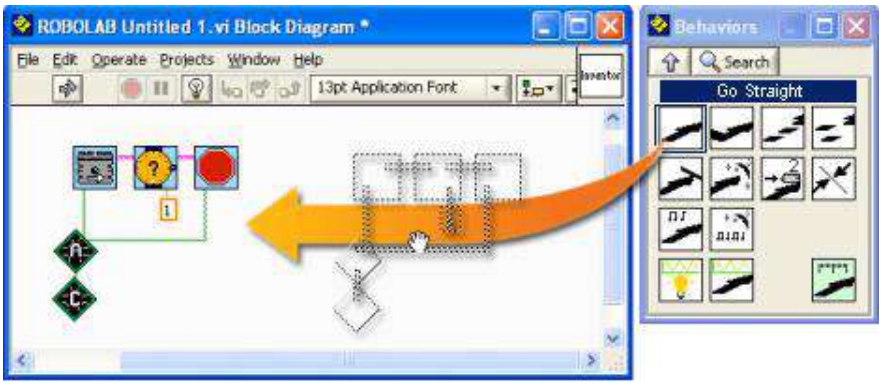


Рис. 5.8. Перетаскивание примера в поле программы.

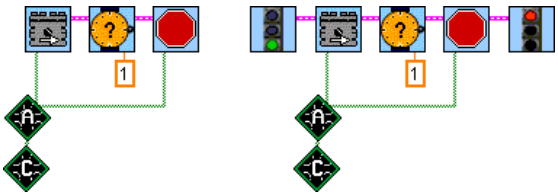


Рис. 5.9. Содержание примера и размещение его между светофорами.

Палитра Behaviors содержит ряд серьезных примеров программирования в Robolab, по которым можно самостоятельно многому научиться. Некоторые из таких примеров рассмотрены ниже.

Взаимодействие с NXT

В режиме программиста могут потребоваться настройка USB-подключения NXT (соединение с компьютером через Bluetooth не поддерживается) и загрузка операционной системы. Это реализуется через меню Projects, которое для опытного робототехника содержит много интересного.

Рис. 5.10. Загрузка ОС NXT через меню Project → Detective.

Рассмотрим некоторые возможности настройки.

Пункт меню Project → Select COM Port позволяет выбрать порт USB-подключения NXT с указанием имени устройства.

Пункт меню Project → Detective позволяет не только выбрать соответствующий порт подключения NXT, но и загрузить Firmware, если это не было сделано в режиме администратора (рис. 5.10). Во время за-

грузки операционной системы у NXT пропадает изображение на экране, но в течение минуты категорически нельзя предпринимать никаких действий над контроллером[1].

Когда операционная система загружена, стандартная программа, попавшая в память NXT, размещается в разделе My Files → Software Files и получает имя rbl. Для того чтобы дать другое имя, следует в программе Robolab заменить начальный блок — зеленый светофор — на аналогичный с надписью NXT из палитры NXT. На таком блоке в специальном розовом прямоугольнике можно задать латинскими буквами и цифрами любое имя программы, причем значимыми будут не более шести первых символов (рис. 5.11).

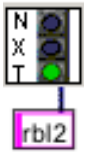


Рис. 5.11. Уникальное имя программы.

Не забывайте сохранять свои программы на диске компьютера, давая им запоминающиеся имена. Когда в другой раз попытаетесь открыть сохраненную программу и увидите черное окно, не пугайтесь: чтобы перейти к алгоритму, необходимо выполнить пункт меню Window → Show Block Diagram.

Типы команд

Блоки Functions Palette можно классифицировать следующим образом:

— команды действия,
— команды ожидания,
— управляющие структуры,
— модификаторы.

Начнем с простейших команд. Их можно разделить на два типа: 1) «Жди» и 2) «Делай».

Команды типа «Делай» посылают управляющий сигнал на одно из устройств управления микроконтроллера, например, «включить моторы», «остановить моторы», «издать звуковой сигнал», «отправить сообщение», «обнулить таймер» и т.п. Это действие, как правило, выполняется практически мгновенно (за исключением звуковых сигналов), после чего программа переходит к следующему блоку. Заметим, что

[1] NXT, вышедший из строя при несанкционированном отключении во время загрузки операционной системы, можно восстановить из другой системы. Например, временно загрузить Firmware из RobotC.

Глава 6. Программирование в RobotC

Введение

Язык программирования RobotC отличается от стандартного C расширенным набором команд по работе с устройствами микроконтроллера. Опытный программист найдет эту среду гораздо более удобной, чем пакеты графического программирования. Для тех же, кто не знаком с языком C, текстовое программирование микроконтроллеров может показаться недостаточно наглядным. Однако именно в текстовом режиме можно составлять наиболее сложные и эффективные программы.

Firmware

Операционная система, предназначенная для исполнения на NXT программ, написанных на RobotC, отличается от аналогичных прошивок микроконтроллера для Robolab или NXT-G. Хотя внешнее сходство присутствует. Однако при попытке исполнения программы, загруженной из другой среды, будет выдаваться сообщение об ошибке. Поэтому перед началом работы необходимо обеспечить загрузку соответствующего Firmware (рис. 6.1).

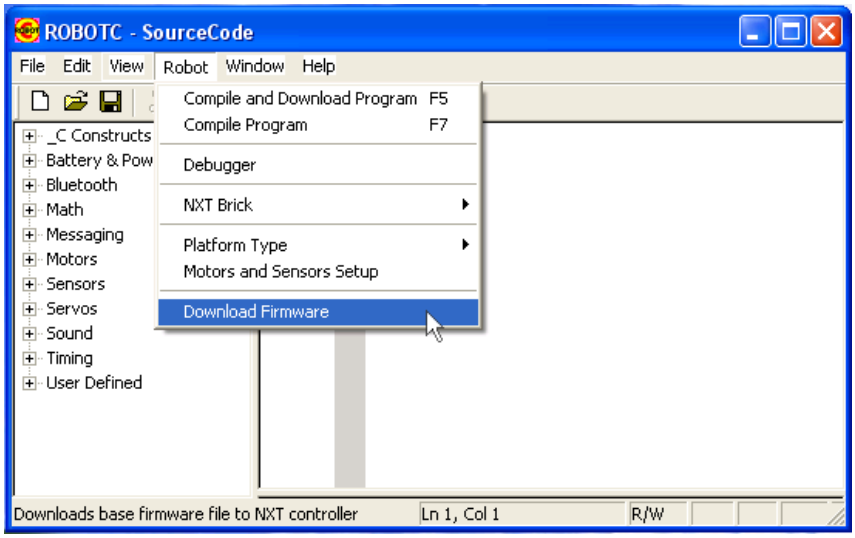


Рис. 6.1. Загрузка операционной системы на NXT.

Hello, world!

Традиционно первая программа на языке Си должна вывести на экран приветствие миру: «Hello, world!». Воспользовавшись возможностями NXT, исполним традицию:

```
task main()
{
  nxtDisplayTextLine(0, "Hello, world!");
  wait1Msec(5000);
}
```

Команда nxtDisplayTextLine(0, "Hello world!"); выведет в строку экрана NXT с номером ноль сообщение «Hello world!». По прошествии 5 с программа закончит выполнение и текст исчезнет. Сохраните программу на жесткий диск с именем "hello.c".

Для загрузки в микроконтроллер подключите его к компьютеру, включите NXT и нажмите на клавиатуре компьютера клавишу F5. Если программа загрузилась успешно, на экране компьютера появится окно отладки Program Debug (рис. 6.2).

Рис. 6.2. Окно отладки программы.

Если нажать в нем кнопку Start, программа запустится на NXT. Можно игнорировать это окно, найти в меню микроконтроллера файл hello и запустить его оранжевой кнопкой.

Усовершенствуем программу. То же сообщение можно вывести по центру экрана:

```
task main()
{
  nxtDisplayCenteredTextLine(3, "Hello world!");
  wait1Msec(5000);
}
```

На этот раз сообщение будет выведено в третьей строке дисплея NXT с выравниванием посередине.

Используя набор команд по работе с экраном NXT, в том числе и графических, можно написать собственные миниигры для микроконтроллера. Но это тема отдельной главы или даже книжки.

Структура программы

Теперь обратим внимание на структуру исходного кода. Главная задача, с которой начинается выполнение программы в RobotC, называется task main(). Тело задачи располагается между двумя фигурными скобками: открывающей и закрывающей. Все команды должны быть размещены между ними.

Более подробно о задачах написано в разделе «Параллельные задачи».

Управление моторами

Состояние моторов

Простейшая задача: вращать мотор C вперед 3 с:

```
task main()
{
  motor[motorC] = 100;
  wait1Msec(3000);
}
```

Команда motor[] представляет собой массив из трех элементов, каждый из которых определяет текущее состояние вращения моторов A, B и C. Действие «полный вперед» задается числом 100, «полный назад» — числом –100, остановка — числом 0. Действие этой команды можно считать мгновенным. После ее выполнения мотор включается и продолжает работать до тех пор, пока не будет остановлен другой аналогичной командой.

Команда wait1Msec() задает время ожидания в миллисекундах (1 мс = 1/1000 с). Таким образом, wait1Msec(3000) означает «ждать 3 секунды».

Теперь сформулируем задачу для робота более четко: проехать вперед 2 секунды со средней скоростью и остановиться. В программе последовательно включаются моторы C и B с мощностью 50, работают в течение 2 с, после чего последовательно выключаются. По сути вклю-

чение обоих моторов произойдет практически одновременно, как и их выключение:

```
task main()
{
  motor[motorC] = 50;
  motor[motorB] = 50;
  wait1Msec(2000);
  motor[motorC] = 0;
  motor[motorB] = 0;
}
```

Следующий пример демонстрирует возожность бесконечного вращения мотора с помощью цикла while. Остановить его можно, прервав выполнение программы на NXT:

```
task main()
{
  while (true)
    motor[motorA]=100;
}
```

Следующий пример работает аналогично:

```
task main()
{
  motor[motorA]=100;
  while (true)
    wait1Msec(1);
}
```

Миллисекундная задержка полезна для разгрузки контроллера при работе бесконечного цикла.

Встроенный датчик оборотов

Команда nMotorEncoder[] — это обращение к датчику оборотов мотора, который возвращает значение в градусах. Каждый из трех элементов этого массива имеет 16-битное значение, что дает возможность хранить число в диапазоне –32768...32767, это соответствует 95 полным (360 градусов) оборотам моторов. В длительно работающих программах следует время от времени обнулять значения массива во избежание переполнения.

```
nMotorEncoder[motorA] = 0;
```

Этим же действием датчик оборотов инициализируется для начала измерений.

Следующая программа обеспечит вращение мотора A на 1000 градусов.

```
task main()
{
  motor[motorA]=50;
  while (nMotorEncoder[motorA] < 1000)
  {
  // никаких действий можно не производить
  }
}
```

Команда nMotorEncoderTarget[] обеспечивает остановку мотора при повороте на заданное число градусов. Команда обладает собственным «интеллектом» и, учитывая инерцию движения, может откорректировать окончательную позицию мотора.

```
task main()
{
  nMotorEncoderTarget[motorA] = 1000;
  motor[motorA] = 50;
  wait1Msec(10000);
}
```

Синхронизация моторов

На практике часто можно заметить, что движение колесного робота непрямолинейно. Это может быть связано с различным трением приводов, разницей нагрузок на колеса и т. п. Команда nSyncedMotors позволяет синхронизовать моторы, объявив один из них главным, второй — подчиненным (один ведущим, второй — ведомым):

```
nSyncedMotors = synchNone; // Отключить синхронизацию
nSyncedMotors = synchAC; // Мотор C подчинен мотору A
```

В режиме синхронизации достаточно управлять только одним мотором: второй будет в точности повторять его действия.

Переменная nSyncedTurnRatio позволяет изменить соотношение движения синхронизированных моторов. Ее значения изменяются от –100 до 100 %. Отрицательный знак указывает на противоположное направление движения подчиненного мотора. Абсолютное значение показывает отношение скорости ведомого мотора к скорости ведущего. Если оно меньше 100, робот поворачивает.

В следующем фрагменте программы робот проедет по прямой, а затем повернется:

```
nSyncedMotors = synchAC; // Мотор C подчинен мотору A

// Движение по прямой
nSyncedTurnRatio = +100; // Режим движения прямо
nMotorEncoder[motorA] = 0;
// остановиться через 1000 градусов
nMotorEncoderTarget[motorA] = 1000;
motor[motorA] = 100;

while (nMotorEncoder[motorA] < 1000) // дождаться остановки
{}

// Повернуться на месте

nSyncedTurnRatio = -100; // Режим поворота
// остановиться через 200 градусов
nMotorEncoderTarget[motorA] = 200;
motor[motorA] = 50;
wait1Msec(3000);
```

Режим импульсной модуляции

Скорость моторов контролируется технологией широтно-импульсной модуляции (ШИМ, pulse width modulation — PWM). Импульсы тока подаются на моторы тысячи раз в секунду. Каждый импульс представляет из себя волну, содержащую период наличия напряжения (on-time) и период отсутствия напряжения (off-time). Отношение между этими двумя периодами определяет мощность, подаваемую на мотор. В периоды on-time используется полный ресурс батареи, и эта технология более эффективна, чем регуляция скорости с помощью изменения напряжения.

В периоды off-time, когда напряжение на моторы не подается, цепь может находиться в двух состояниях: разомкнутом и замкнутом. Разомкнутая цепь получается в режиме плавающего напряжения, замкнутая — в режиме торможения. RobotC позволяет выбрать режим как в установках среды, так и с помощью специальной переменной bFloatDuringInactiveMotorPWM. При компиляции программы будет установлен выбранный режим:

```
bFloatDuringInactiveMotorPWM = false; // режим торможения
bFloatDuringInactiveMotorPWM = true; // плавающий режим
```

Для моторов NXT предпочтительным является режим торможения, он и выставляется по умолчанию.

Глава 7. Алгоритмы управления

Релейный регулятор

Одной из главных задач теории автоматического управления является управление с помощью обратной связи. В таких задачах можно выделить четыре основных компонента [1]:

— управляемую систему (или как говорят специалисты, объект управления) — то, чем мы хотим управлять;

— цель управления — то, чего мы хотим достичь при помощи управления, т.е. желаемое поведение объекта управления;

— список измеряемых переменных (или выходов) — то, что мы можем измерять;

— список управляющих переменных (или входов) — то, что мы можем менять для того, чтобы воздействовать на объект управления.

Еще один важный компонент — регулятор — устройство, вырабатывающее входные величины, необходимые для достижения заданной цели. Этот пятый элемент обычно появляется после того, как теоретическое решение задачи найдено. Под решением проблемы управления будем понимать нахождение закона управления (алгоритма управления), обеспечивающего достижение цели. Как только искомый закон найден, он может быть использован для вычисления управляющих входов по измеренным значениям выходов объекта управления. Полученные значения входов в виде некоторых сигналов подаются на исполнительные устройства.

В формировании этих сигналов может принимать участие микропроцессор, производящий достаточно сложные вычисления в соответствии с заданным алгоритмом.

Будем рассматривать простейший случай, когда задача состоит в поддержании системы в определенном состоянии. Тогда можно говорить о задаче регулирования как частном случае задачи автоматического управления. Для осуществления автоматического регулирования к объекту подключается регулятор. Под регулятором будем понимать устройство, которое с помощью чувствительного элемента (датчика) измеряет регулируемую величину и в соответствии с законом регулирования вырабатывает воздействие на регулирующий орган объекта. Система, состоящая из объекта и регулятора, называется системой управления [1].

Исполнительное устройство, осуществляющее механическое перемещение регулирующего органа, обычно называется сервоприводом.

Релейным двухпозиционным регулятором называется регулятор, у которого регулирующий орган под действием сигнала от датчика может принимать одно из двух крайних положений: «открыт» или «закрыт». При этом управляющее воздействие на регулируемый объект может быть только максимальным или минимальным.

Управление мотором

«Робот поднимает меч и бросается на противника!» Здорово. Как в сказке. А что если меч наткнется на что-нибудь и потеряет свое заданное положение? Нехорошо бросаться на противника с опущенным мечом.

Для примера прикрепите к мотору длинную балку с помощью сдвоенного трехмодульного штифта. Это будет меч. По сценарию, в момент запуска его надо поднять на 45 градусов и удерживать в этом положении, что бы ни случилось (рис. 7.1).

Рис. 7.1. Стабилизация мотора в положении 45 градусов.

Регулируемой величиной будет угол поворота мотора, определяемый через показания энкодера, регулирующим органом — сам мотор.

Алгоритм таков: обнулить показания датчика оборотов, задать желаемое положение в 45 градусов и в цикле поддерживать это положение, учитывая возможные отклонения. Если показания энкодера превышают 45, мотор вращается назад. В противном случае вращается вперед. Задержка в 1 мс предназначена для разгрузки контроллера.

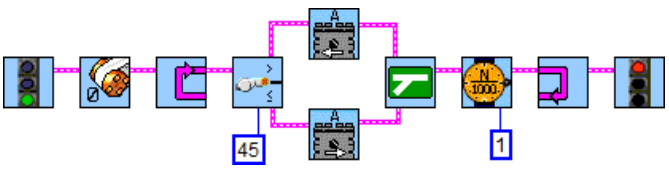


Рис. 7.2. Релейное управление одним мотором.

```
task main()
{
  int alpha=45;
  nMotorEncoder[motorA]=0;
  while(true)
  {
    if(nMotorEncoder[motorA]>alpha)
      motor[motorA]=-100;
    else
      motor[motorA]=100;
    wait1Msec(1);
  }
}
```

Что же мы увидим? Робот ведет себя как новобранец с мечом. Мотор удерживает балку, но как-то неуверенно: происходят постоянные колебания. Уменьшить их можно, разве что понизив мощность мотора. Попробуйте сделать это.

Особенность релейного регулятора в том, что он в принципе не может стабилизироваться в нужном положении и вызывает колебания с той или иной амплитудой. Как этого избежать, описано чуть дальше.

Движение с одним датчиком освещенности

Рассмотрим пример трехколесного Lego-робота с одним датчиком освещенности, который должен двигаться по плоской поверхности вдоль границы черного и белого (рис. 7.3).

Рис. 7.3. Движение вдоль границы черного и белого.

Исполнительным органом объекта в данном случае будут два колеса, подключенные к сервоприводам. Регулируемая величина — положение датчика света на границе черного и белого, зависящее от уровня освещенности под ним. Возмущающее воздействие — это движение робота вперед, которое приводит к отклонению от границы. Поскольку линия может быть кривой, а также вследствие других факторов при отсутствии регулирующего воздействия робот непременно съедет с линии.

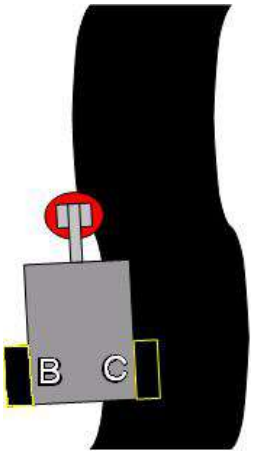


Рис. 7.4. Расположение моторов и датчика освещенности.

Подключим левый мотор на порт B, правый — на порт C (рис. 7.4). Стартовая позиция робота — датчик на белом. Построим программный регулятор, который обеспечит движение по дуге в сторону черного, пока робот на белом, и движение по дуге в сторону белого, пока робот на черном. Для этого выключается или резко понижается мощность одного из моторов. Вот реализация на RobotC:

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

Значение «серого» (grey) может быть константой для данной системы или вырабатываться в результате предварительной калибровки как среднее арифметическое черного и белого.

Под управлением такого регулятора робот движется вдоль линии границы по ломаной кривой, периодически наезжая то на черное, то на белое. Скорость его невысока, стабильность тоже, а траектория движе-

ния оставляет желать лучшего. Все это издержки использования двухпозиционного релейного регулятора.

Программа для движения по линии с помощью релейного регулятора в Robolab показана на рис. 7.5.

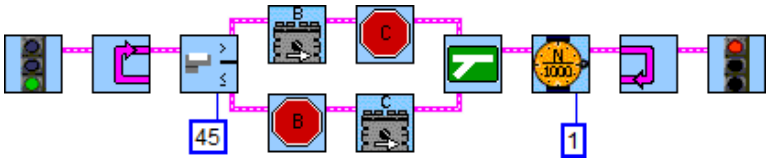


Рис. 7.5. Алгоритм движения вдоль границы черного и белого на релейном регуляторе.

Движение с двумя датчиками освещенности

Правильно проехать перекресток с одним датчиком освещенности довольно сложно. Если требуется сделать это с достаточно высокой скоростью, нужны хотя бы два датчика, поставленные на расстоянии в две-три ширины линии (или шире). Для начала используем релейный регулятор, с помощью которого можно обработать четыре возможных состояния датчиков (рис. 7.6):

— оба на белом — движение прямо;

— левый (S1) на черном, правый (S2) на белом — движение налево;

— левый на белом, правый на черном — движение направо;

— оба на черном — движение прямо.

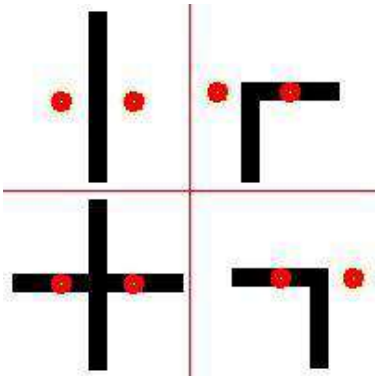


Рис. 7.6. Варианты расположения двух датчиков освещенности над черной линией.

Этот регулятор реализуется программно с помощью вложенных ветвлений:

```
task main()
{
  int grey1=45, grey2=45;
  while(true) {
    if(SensorValue[S1]>grey1) {
      if(SensorValue[S2]>grey2) { // Оба на белом
        motor[motorB]=100;
        motor[motorC]=100;
```

```
      }
      else {  // Правый на черном
        motor[motorB]=100;
        motor[motorC]=-100;
      }
    }
    else {
      if(SensorValue[S2]>grey2) {    // Левый на черном
        motor[motorB]=-100;
        motor[motorC]=100;
      }
      else {  // Оба на черном
        motor[motorB]=100;
        motor[motorC]=100;
      }
    }
  wait1Msec(1);
  }
}
```

То же самое в Robolab (рис. 7.7):

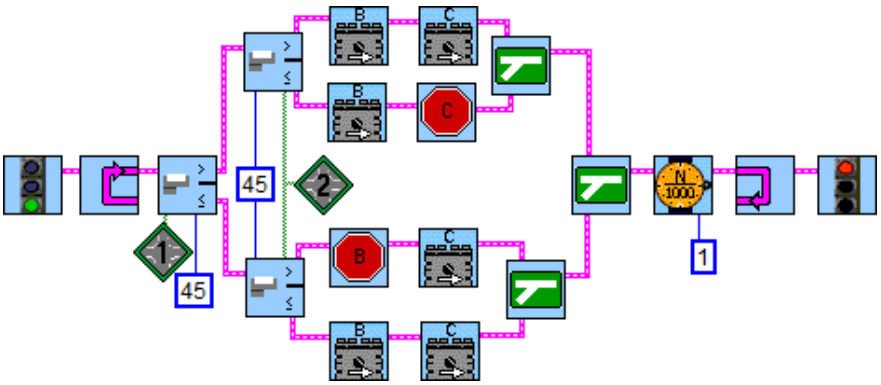


Рис. 7.7. Алгоритм движения вдоль черной линии с двумя датчиками на релейном регуляторе.

Вот такой, довольно длинный и малоэффективный алгоритм автор и многие его коллеги по состязаниям Lego-роботов использовали для решения классической задачи движения по траектории. Чтобы не сбиться с линии, приходилось значительно усложнять его, вводить дополнительные проверки и, конечно, калибровку датчиков под конкретную освещенность. Однако достаточно было в окнах появиться солнышку, как робот терял линию, поскольку значения были жестко фиксированными. Так продолжалось до тех пор, пока мы не познакомились с базовыми принципами автоматического регулирования.

Пропорциональный регулятор

Описание

При автоматическом регулировании управляющее воздействие *u(t)* обычно является функцией динамической ошибки — отклонения *e(t)* регулируемой величины *x(t)* от ее заданного значения *x0(t)*:

$$e(t) = x0(t) - x(t).$$

Это принцип Ползунова-Уатта регулирования по отклонению, или принцип обратной связи. Математическое выражение функциональной зависимости желаемого управляющего воздействия *u0(t)* от измеряемых регулятором величин называется законом или алгоритмом регулирования, о котором говорилось выше.

Пропорциональный регулятор — это устройство, оказывающее управляющее воздействие на объект пропорционально его отклонению от заданного состояния:

$$u0(t) = ke.$$

Здесь *k* — это коэффициент усиления регулятора.

Заданное состояние *x0* принято называть *уставкой*, а отклонение от него *e* — *невязкой.* Далее для определенности будем обозначать невязку сокращением *err* (от английского слова «error» — ошибка).

Управление мотором

Опытный воин не станет размахивать мечом, как это делает робот на релейном регуляторе. Надо придумать алгоритм, который задержит мотор, удерживающий меч, в строго фиксированном положении (рис. 7.1). В этом поможет П-регулятор.

Пусть *e*1 — показания датчика оборотов[1] на моторе A — является регулируемой величиной. Уставка *x0* = 45, а невязка *e* = 45 – *e*1. Тогда управляющее воздействие на мотор задается формулой

$$u = k \cdot (45 - e1).$$

Здесь k — коэффициент усиления, например 5, который позволит усилить реакцию мотора даже при небольших отклонениях от уставки.

[1] Не стоит путать математическое обозначение невязки *e* (от error) с показаниями энкодера *e*1 (от encoder), предопределенной переменной среды Robolab.

При отклонении в положительную сторону на мотор будет подаваться отрицательное управляющее воздействие, и наоборот. Это управление можно применять к мотору в цикле с небольшой задержкой в 1-10 мс, чтобы разгрузить контроллер (рис. 7.8).

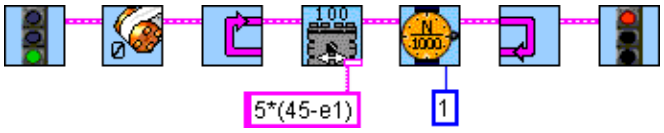


Рис. 7.8. Алгоритм управления мотором на пропорциональном регуляторе.

Если усиливающий коэффициент увеличить с 5 до 100, наш пропорциональный регулятор начнет работать как релейный, вызывая сильные колебания из-за возникновения эффекта перерегулирования.

В языке RobotC нет такого удобного обозначения показаний энкодера как в Robolab, поэтому программа выглядит немного длиннее:

```
task main()
{
  int k=5, u;
  nMotorEncoder[motorA]=0;
  while(true)
  {
    u=k*(45-nMotorEncoder[motorA]);
    motor[motorA]=u;
    wait1Msec(1);
  }
}
```

Далее, чтобы нанести «удар мечом», достаточно, имея вместо числа 45 переменную, изменить ее значение извне, например, из параллельной задачи. Об этом написано в разделе, посвященном роботам-барабанщикам в главе 8.

А сейчас построим регулятор, управляющий не только статическим положением мотора, но и скоростью его движения. Следуя логике алгоритма, уставка, которая до сих пор была константой и не менялась, должна начать движение в сторону увеличения или уменьшения. Повинуясь регулятору, мотор неизбежно будет следовать за ней. Самый простой инструмент для постоянного приращения значения уставки — это таймер.

В контроллере NXT есть четыре встроенных таймера, каждый из которых может отмерять время в десятых, сотых и тысячных долях секунды. Освоим первый таймер, который за секунду совершает 10 «тиков». В Robolab он обозначается T1 или Timer100ms1, а в RobotC — timer100[T1].

Угол *alpha* отклонения мотора, заданный в предыдущем примере значением 45, поставим в зависимость от показаний таймера с ускоряющим коэффициентом *k*2:

$$alpha = k2 \cdot T1.$$

Управляющее воздействие останется прежним с усиливающим коэффициентом *k*1:

$$u = k1 \cdot (alpha - e1).$$

Кратко в программе на языке Robolab управляющее воздействие подадим сразу на мотор, предварительно инициализировав таймер (рис. 7.9).

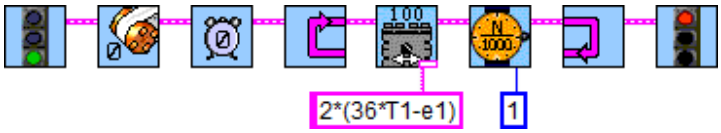


Рис. 7.9. Управление скоростью мотора — один оборот в секунду.

Коэффициент $k2 = 36$ определяет, что за секунду значение *alpha* набегает до 360, что соответствует одному полному обороту двигателя:

```
task main()
{
  int k1=2, k2=36, u, alpha;
  nMotorEncoder[motorA]=0;
  ClearTimer(T1);
  while(true)
  {
    alpha=timer100[T1]*k2;
    u=k1*(alpha-nMotorEncoder[motorA]);
    motor[motorA]=u;
    wait1Msec(1);
  }
}
```

Используя целочисленное деление, принятое в языке C (и в Robolab) для переменных целого типа, можно достичь дискретного изменения угла, т.е. приращения его раз в секунду:

$$alpha = T1 / 10 \cdot k2.$$

При коэффициенте $k2 = 60$ перемещения балки будут соответствовать движению секундной стрелки на циферблате часов. Но это мало

заметно. Для наглядности можно задать *k2* = 30, тогда стрелка сделает полный оборот за 12 «тиков» по 30 градусов каждый. Будьте внимательны с последовательностью операций целочисленного деления и умножения, при изменении их порядка или «сокращении» непременно изменится результат (рис. 7.10).

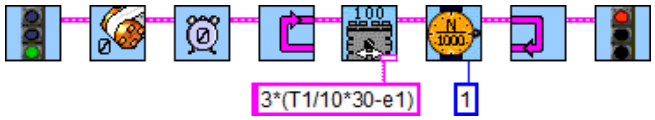


Рис. 7.10. Ускоренная имитация движения стрелки часов.

И, наконец, пример математического барабанщика. Вместо постоянного движения вперед стрелка будет совершать колебания вперед-назад под управлением П-регулятора. В этом поможет операция деления с остатком, которая в языке С обозначается знаком %. Остаток от деления неотрицательного целого числа на 2 всегда будет 0 или 1:

$$alpha = T1 \,\%\, 2 \cdot k2.$$

Усилив отклонение в *k2*=15 раз, получим колеблющуюся уставку *alpha*, что вынудит регулятор 5 раз в секунду перемещать мотор то в 0º, то в 15 градусов. Изменения в программе невелики. Рассмотрим пример на RobotC:

```
task main()
{
  int k1=3, k2=15, u, alpha;
  nMotorEncoder[motorA]=0;
  ClearTimer(T1);
  while(true)
  {
    alpha=timer100[T1]%2*k2;
    u=k1*(alpha-nMotorEncoder[motorA]);
    motor[motorA]=u;
    wait1Msec(1);
  }
}
```

Этот прототип барабанщика наносит удары по столу через одинаковые промежутки времени. Главное, стартовать в нужной позиции. Используя целочисленную математику, можно задать и более сложный ритмический рисунок, например (табл. 7.1):

$$alpha = T1 \,\%\, 5 \,\%\, 2 \cdot k2.$$

center = *S*3.

Коэффициент определяется в цикле:

*k*1 = *c* + (*S*3 - *center*) / *k*2.

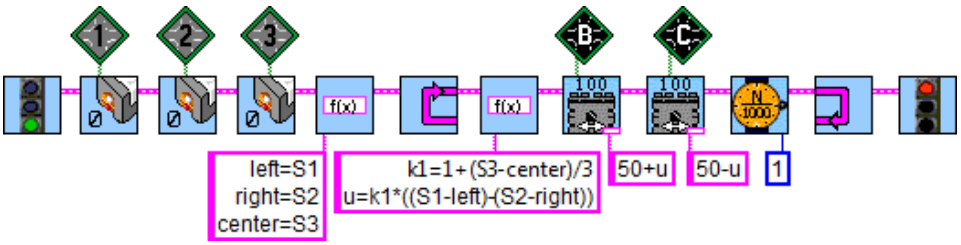


Рис. 7.36. Движение по линии на пропорциональном регуляторе с плавающим коэффициентом.

Полученный закон управления коэффициентами усиления можно применить не только к пропорциональной составляющей, но и к любой другой, а также к управляющему воздействию в целом (рис. 7.36).

ПИД-регулятор

Пропорционально-интегрально-дифференциальный (ПИД) регулятор является одним из наиболее популярных и используется в огромном количестве устройств самых разных типов, в которых требуются быстрота реакции и точность позиционирования системы. Как следует из названия, этот регулятор состоит из суммы трех компонент и графически изображен на рис. 7.37.

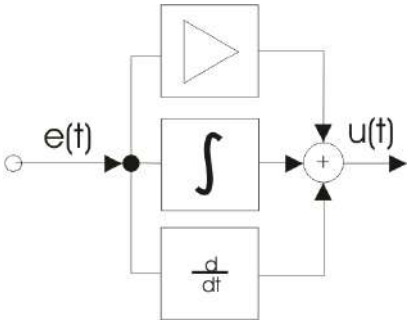


Рис. 7.37. Схема ПИД-регулятора.

Это упрощенная схема. На вход регулятора подается значение динамической ошибки $e(t)$, а на выходе вырабатывается управляющее воздействие $u(t)$:

$$u(t) = p + i + d = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau)d\tau + k_d \cdot \frac{de}{dt}.$$

Пропорциональная составляющая, изображенная на схеме треугольником, отвечает за позиционирование системы в заданном состоянии. В некоторых случаях может вызвать перерегулирование с последующими автоколебаниями. То есть П-регулятор может «перестараться» и робота начнет заносить из стороны в сторону.

Интегральная составляющая накапливает отрицательный опыт (суммирует ошибки) и вырабатывает компенсирующее воздействие. При минимальных отклонениях пропорциональная составляющая «слабеет» и интегральная, за счет своего быстрого увеличения суммированием, помогает «дотянуть» регулируемую величину до уставки.

Дифференциальная составляющая (Д-составляющая) следит за скоростью изменения состояния системы и препятствует возможному перерегулированию. В некоторых случаях Д-составляющая противоположна пропорциональной по знаку, а в некоторых совпадает.

С пропорциональной составляющей мы уже знакомы, дифференциальная описана в предыдущей главе 6. Возьмемся за интегральную. Эта составляющая определяется динамически, суммируясь с предыдущим значением:

$$i = i + k_i \cdot e(t) \cdot dt.$$

Физический смысл величины $e(t) \cdot dt$ состоит в том, что она пропорциональная длительности нахождения системы в состоянии ошибки. Поскольку коэффициент k_i выносится за скобки, можно говорить о величине i как сумме длительностей ошибок. Таким образом, мы находим интеграл путем суммирования.

Рассмотрим применение ПИД-регулятора на примере робота, балансирующего на двух колесах. Эта классическая задача может быть решена с помощью различных датчиков множеством способов. В предложенном примере использован датчик освещенности и простейшая форма ПИД-регулятора. Однако для достижения стабилизации робота потребуется использовать более точные показания датчика.

Формат RAW

Данные с датчиков поступают в контроллер NXT в необработанном «сыром» виде. Все датчики передают операционной системе цифровое значение от 0 до 1023, которое затем обрабатывается соответствующим драйвером и приводится к более понятному виду (расстояние 0...255, освещенность 0...100, касание 0 или 1 и т. д.). Но данные можно получать и, минуя драйвер, напрямую. Такой необработанный формат принято называть RAW (от англ. «сырой»). В некоторых случаях с помощью него можно получить бóльшую точность. Так, например, диапазон значений датчика освещенности может увеличиться примерно в 10 раз. Именно эта возможность использована далее.

Получать данные в формате RAW можно и в Robolab, и в RobotC. Для этого датчик инициализируется соответствующим образом, а данные с него считываются с использованием специальной предопределенной переменной.

Балансирующий робот

Конструкция робота-сигвея изображена на рис. 7.38: вертикально расположенный контроллер, близко поставленные колеса и датчик освещенности, направленный вниз. Алгоритм будет несколько сложнее.

Принцип стабилизации сигвея в положении равновесия состоит в следующем. Если робот наклоняется вперед, показания на датчике освещенности повышаются за счет отраженного света. В ответ на это вырабатывается управляющее воздействие, заставляющее робота ехать вперед и тем самым снова принимать вертикальное положение.

При отклонении назад показания датчика понижаются и робот начнет движение назад. За все это отвечает пропорциональная составляющая. На роль интегральной и дифференциальной составляющих отводится страховка от перерегулирования.

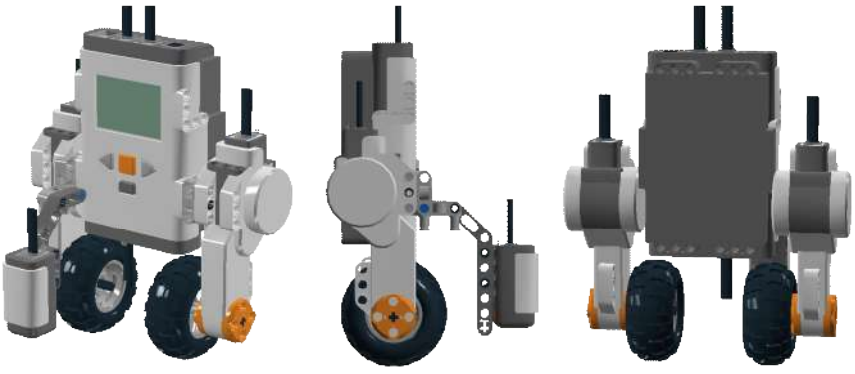


Рис. 7.38. Балансирующий робот-сигвей.

На рис. 7.39 представлен алгоритм в Robolab. Бóльшую его часть занимает инициализация переменных. Для повышения точности не только данные с датчика считываются в формате RAW, но большинство переменных объявляется в вещественном формате float. Собственно ПИД-алгоритм находится в цикле.

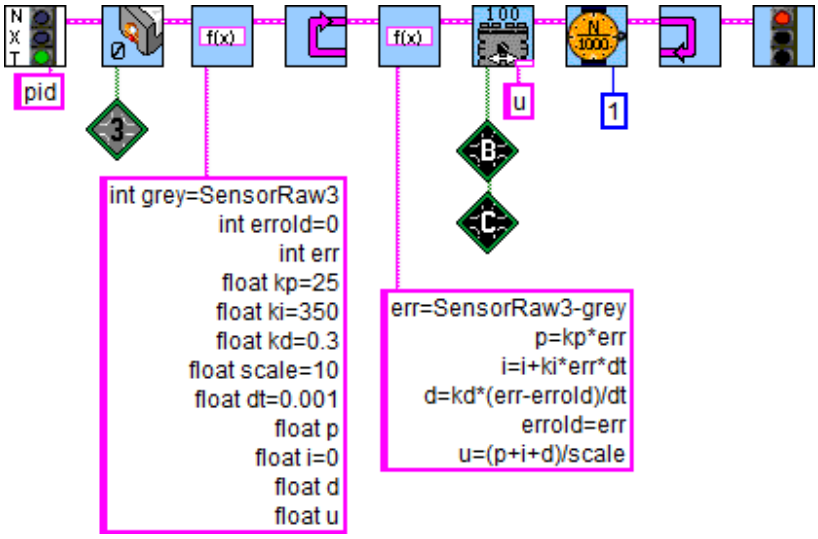


Рис. 7.39. Алгоритм балансировщика основан на ПИД-регуляторе.

Следуя традиции движения по линии, в качестве уставки используем переменную *grey* — средние показания датчика освещенности в положении равновесия. Новый параметр *scale* задает масштабирование управляющего воздействия. По сути, это ослабляющий коэффициент, поскольку вырабатываемое регулятором значение слишком высоко для моторов NXT. Можно было бы внести его внутрь уже имеющихся коэффициентов, но для RobotC этот параметр будет другой, а коэффициенты те же.

С приведенными коэффициентами робот хорошо стабилизируется на однотонном светлом линолеуме или парте. То есть ему не требуется белый цвет поверхности. Для запуска нужно достаточно точно установить сигвея в положение равновесия. Если робот стартует при некотором наклоне вперед или назад, то сразу начнет движение в направлении наклона.

Аналогичный пример на RobotC несколько отличается в силу ряда причин. Во-первых, быстродействие NXT с прошивкой этой среды выше примерно в 1.4 раза, чем у Robolab, поэтому коэффициент scale следует увеличить. Во-вторых, RAW-значения передаются в правильном порядке и потребуется установить реверс моторов или просто подавать отрицательное управляющее воздействие:

```
task main()
{
  int grey=SensorRaw[S3];
  int err, errold=0;
  float kp=25, ki=350, kd=0.3;
  float scale=14;
  float dt=0.001;
  float p, i=0, d, u;
  while (true)
  {
    err= grey-SensorRaw[S3]; //Отклонение с обратным знаком
    p=kp*err;
    i=i+ki*err*dt;
    d=kd*(err-errold)/dt;
    errold=err;
    u=(p+i+d)/scale;
    motor[motorB]=u;
    motor[motorC]=u;
    wait1Msec(1);
  }
}
```

Элементы теории автоматического управления в школе[1]

Важной и интересной методической задачей является «переброска мостика» между областями знаний специалиста и школьника, помогающая учащимся школы увидеть перспективу будущей специальности, т.е. осуществить профориентацию, а студентам увидеть практическую применимость своих профессиональных знаний. Для достижения подобного эффекта были разработаны приемы расчета регуляторов, использующие математический аппарат, не выходящий за рамки школьных программ по математике и физике. В частности, вместо дифференциальных уравнений использованы разностные, хорошо соответствующие дискретному характеру взаимодействия объекта и регулятора при компьютерном управлении.

Рассмотрим, например, задачу построения пропорциональных (П) и пропорционально-дифференциальных (ПД) регуляторов в задаче управления движением мобильного робота вдоль стены. Обозначим через x_t расстояние между роботом и стеной, через θ_t — курсовой угол робота, а через u_t — управляющее воздействие в момент с порядковым номером t, соответственно, где $t = 0, 1, 2, \ldots$ — номера моментов изме-

[1] Автор статьи – д. техн. наук, профессор А. Л. Фрадков.

рений. Считается, что опрос датчиков и изменения величины управляющего воздействия производится через равные промежутки времени h. Для задач управления Lego NXT роботами естественно считать, что управляющим воздействием является разность угловых скоростей вращения колес, пропорциональная скорости изменения курсового угла:

$$\theta_{t+1} = \theta_t + u_t\, hr / b, \tag{1}$$

где r — радиус колес, b — база транспортного средства (расстояние между колесами). Легко также видеть, что расстояние робота от стены за время h изменится на величину $h \sin \theta_t$, т.е. имеет место соотношение

$$x_{t+1} = x_t + v_t\, h\, sin\, \theta_t\,. \tag{2}$$

Считая отклонения курса от номинального $\theta t = 0$ малыми, а среднюю скорость робота постоянной: $vt=v$, динамику изменения переменных состояния робота в первом приближении можно описать линейными уравнениями состояния:

$$x_{t+1} = x_t + vh\theta_t\,,\ \theta_{t+1} = \theta_t + u_t\, hr / b. \tag{3}$$

Исключая переменную θ_t, приходим к разностному уравнению 2-го порядка, непосредственно связывающему управляющую и регулируемую переменные:

$$x_{t+2} - 2\, x_{t+1} + x_t = gu_t\,, \tag{4}$$

где $g = h2vr / b$.

Зададим желаемое расстояние до стены $x^*>0$ и определим цель управления (ЦУ) соотношением

$$x_t \to x^* \text{ при } t \to \infty. \tag{5}$$

Теперь естественным образом введем на содержательном уровне понятие асимптотической устойчивости, как свойства решений системы (4), обеспечивающего достижение ЦУ (5) при любых начальных условиях, достаточно мало отличающихся от целевых. Легко видеть, что при $u_t = 0$ решением уравнения (4) является любое постоянное значение $x_t = x^*$. Но поскольку уравнение (4), соответствующее модели двойного интегратора (двойного сумматора), не обладает свойством асимптотической устойчивости, ЦУ (5) при постоянном управлении не достигается. Это легко демонстрируется как аналитически — суммированием ря-

да натуральных чисел, так и экспериментально, выставляя робот в различные начальные положения.

После такой демонстрации весьма естественно ввести понятия регулятора и обратной связи по результатам измерений и они легко осваиваются школьниками. Сначала рассматривается простейший пропорциональный регулятор (П-регулятор):

$$u_t = K_0(x^* - x_t), \tag{6}$$

где K_0 — коэффициент усиления регулятора. Подстановкой (6) в (4) получаем уравнение замкнутой системы

$$x_{t+2} - 2\,x_t + {}_1 + (1 + gK_0)x_t = gK_0x^*. \tag{7}$$

Переходя к уравнениям для отклонений $yt = x^* - xt$, получим однородное уравнение

$$y_{t+2} - 2\,y_{t+1} + (1 + gK_0)y_t = 0. \tag{7a}$$

Возникающий вопрос об устойчивости системы (7) теперь можно решить в случае общей однородной системы 2-го порядка:

$$x_{t+2} + a_1x_{t+1} + a_0x_t = 0, \tag{8}$$

записав ее общее решение в виде суммы двух геометрических прогрессий со знаменателями λ_1 и λ_2 (возможно, комплексными) равными корням квадратного трехчлена $\lambda_2 + a_1\lambda + a_0$. Условием асимптотической устойчивости оказывается пара неравенств $|\lambda_1| < 1$, $|\lambda_2| < 1$, означающая, что обе геометрические прогрессии являются бесконечно убывающими. Теперь стандартное условие устойчивости в терминах коэффициентов

$$a_0 < 1,\ 1 + a_0 > |\,a_1| \tag{9}$$

школьники могут вывести самостоятельно.

Применяя правило (9) к уравнению системы с П-регулятором (7), убеждаемся в том, что неравенства (9) несовместны, т.е. что не существует числа K_0, делающего систему (7) асимптотически устойчивой. Таким образом, П-регулятор оказывается неработоспособным, и для решения задачи поиски нужно продолжить.

Следующим этапом является построение ПД-регулятора, который можно описать соотношением

$$u_t = K_0(x^* - x_t) + K_1(x_{t+1} - x_t), \qquad (10)$$

где K_1 — коэффициент усиления по разности (дифференциальный). Поскольку в момент времени t измерить x_{t+1} невозможно, для реализации регулятора (10) можно воспользоваться соотношением

$$x_{t+1} - x_t = vh\theta_t = vh\theta_{t-1} + u_{t-1}\, vh2r / b.$$

Подставляя в (10), получим ПД-закон управления в виде

$$u_t = [K_0(x^* - x_t) + K_1\, vh(\theta_{t-1} + u_{t-1}\, hr / b)], \qquad (11)$$

т. е. для применения (11) требуется помнить значения курсового угла и управления на предыдущем шаге. Для исследования устойчивости и выбора коэффициентов ПД-регулятора подставим (10) в (4). Получим уравнение замкнутой системы:

$$x_{t+2} - (2 + gK_1)x_{t+1} + (1 + gK_0 + gK_1)x_t = gK_0x^*. \qquad (12)$$

Проверяя его на устойчивость с помощью критерия (9), убеждаемся в том, что достаточными условиями устойчивости служат два неравенства

$$K_0 > 0,\ K_0 + K_1 < 0. \qquad (13)$$

Таким образом, для построения работоспособного регулятора следует выбирать дифференциальный коэффициент, превосходящий пропорциональный по абсолютному значению и противоположный по знаку. В справедливости этого правила также следует убедиться в экспериментах.

Предложенные методики исследования и реализации мобильных роботов на основе элементарной теории управления способствуют росту интереса учащихся к робототехнике и, более широко, к инженерным наукам.

Глава 8. Задачи для робота

Управление без обратной связи

В задачах управления обычно существуют два объекта: управляющий и управляемый. В простейшем варианте от управляющего объекта поступает команда и управляемый выполняет ее, ничего не сообщая о результате или об изменившихся условиях работы. В этом суть прямой связи (рис. 8.1).

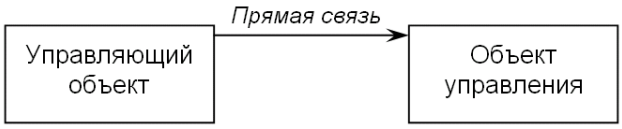


Рис. 8.1. Прямая связь в управлении.

С точки зрения мобильного робота управляющий объект — это его контроллер с запущенной программой, объект управления — это его колеса и корпус (шасси). Управляющие команды контроллер подает на моторы, при прямой связи руководствуясь показаниями своих внутренних часов — таймера.

Первый класс задач, с которых начинается программирование, — это управление перемещениями робота. Рассмотрим их по порядку. В качестве сред программирования используем Robolab 2.9.4 для начинающих и RobotC для подготовленных программистов. В качестве модели робота — любую двухмоторную тележку.

Движение в течение заданного времени вперед и назад

Для движения вперед используются команды управления моторами. Эти команды просто включают моторы. Особенность NXT заключается в том, что после окончания выполнения программы сохраняются все установки в поведении робота, но на моторы перестает подаваться напряжение. Таким образом, просходит пуск и сразу плавное торможение (рис. 8.2).

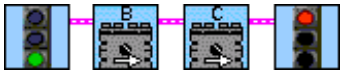


Рис. 8.2. Включение моторов.

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

Обе команды выполняются практически мгновенно. Если сразу следом за ними выключить моторы, то тележка просто дернется и останется стоять на месте (рис. 8.3):

Рис. 8.3. Остановка при попытке начать движение.

```
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 (рис. 8.4):

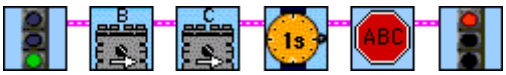


Рис. 8.4. Правильный порядок управления моторами.

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

Движение вперед или назад, очевидно, определяется направлением вращения моторов (рис. 8.5). Для смены направления не требуется остановка:

Рис. 8.5. Проехать секунду вперед, секунду назад и остановиться.

```
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 подачей команды «назад» с моторов снимается напряжение и робот некоторое время едет по инерции (рис. 8.6).

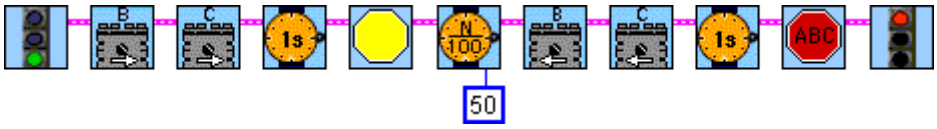


Рис. 8.6. Перед сменой направления полсекунды ехать по инерции.

Более краткий промежуток, чем 1 секунда, задается с помощью команды «N/100» и модификатора. В Robolab 2.9.4 можно задавать время в миллисекундах командой «N/1000»:

```
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;
}
```

В Robolab обычными командами моторы включаются в плавающем режиме, а в RobotC по умолчанию используется режим «торможения»,

который позволяет достичь более точного управления. Но и в Robolab существуют «продвинутые» команды управления моторами в режиме торможения да еще с диапазоном мощностей –100...100.

Повороты

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. Для более точного поворота надо подбирать время в сотых долях секунды (рис. 8.7). Однако при изменении заряда батареек придется вводить новые параметры поворота:

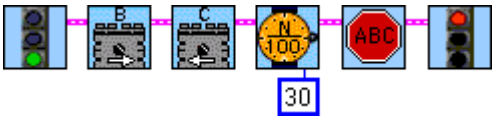


Рис. 8.7. Поворот на месте.

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

Существует другой тип поворотов. Если один из моторов остановить, а другой включить, то вращение будет происходить вокруг стоящего мотора. Поворот получится более плавным (рис. 8.8):

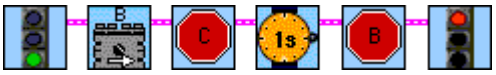


Рис. 8.8. Плавный поворот.

```
task main()
{
  motor[motorB] = 100;
  motor[motorC] = 0;
  wait1Msec(1000); // вращается только мотор B
  motor[motorB] = 0;
}
```

Движение по квадрату

Используя полученные знания управления моторами, можно запрограммировать движение по квадрату или другому многоугольнику с помощью цикла или безусловного перехода (рис. 8.9):

Рис. 8.9. Движение по многоугольнику с плавными поворотами.

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

Уточнив длительность поворотов и число повторений, научим тележку объезжать квадрат по периметру 1 раз (рис. 8.10). Для точности поворотов снизим мощность моторов примерно вдвое. Задержки придется подобрать самостоятельно:

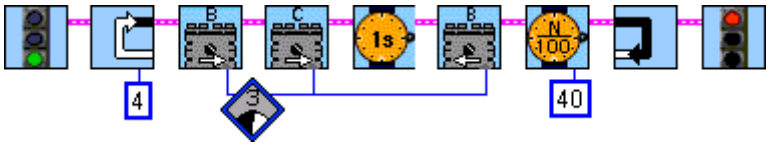


Рис. 8.10. Для поворота на 90 градусов длительность придется подобрать самостоятельно.

```
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;
  }
}
```

Управление с обратной связью

Обратная связь

Появление обратной связи в системе означает то, что управляющий объект начинает получать информацию об объекте управления (рис. 8.11).

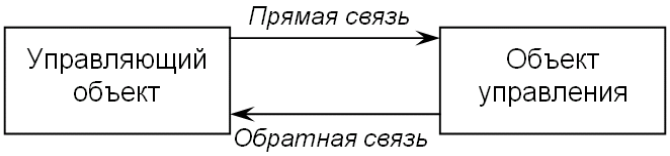


Рис. 8.11. Управление с обратной связью.

Обратная связь осуществляется с помощью датчиков, прикрепленных, например, на корпус робота. Данные поступают в контроллер, который является управляющим объектом.

Точные перемещения

Чтобы поворот не зависел от заряда батареек, можно воспользоваться встроенным в двигатели датчиком оборотов, «энкодером», который позволяет делать измерения с точностью до 1 градуса. Для более эффективного управления задействуем в Robolab «продвинутые» команды, считая что при повороте тележки на 90 градусов левое колесо поворачивается на 250 градусов вокруг своей оси (рис. 8.12):

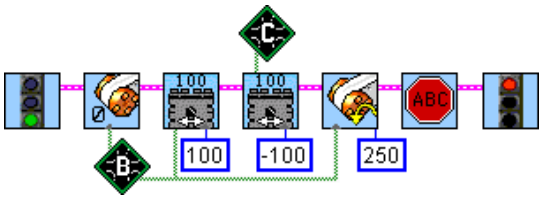


Рис. 8.12. Точный поворот на месте.

```
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;
}
```

Движение вдоль линии

Один датчик

Для первого опыта подойдут робот, созданный для задания «Танец в круге», и то же поле — черная окружность на белом фоне. Если уже все готово для изготовления полноценного поля для траектории, можно обратиться к разделу «Поле» в конце этой части. Единственная поправка: датчик освещенности следует выдвинуть немного вперед, чтобы он образовывал вместе с ведущими колесами равносторонний или хотя бы равнобедренный прямоугольный треугольник (рис. 8.32).

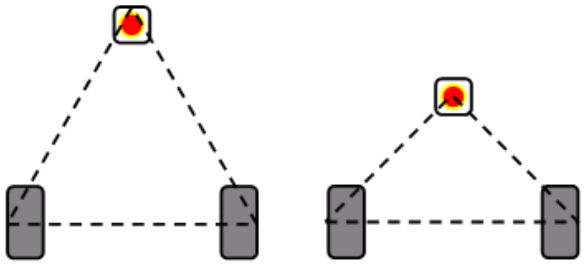


Рис. 8.32. Варианты расположения датчика освещенности относительно ведущих колес.

Задача такова: двигаться вдоль окружности по границе черного и белого. Решается элементарно применением релейного (или пропорционального) регулятора, который рассмотрен в главе «Алгоритмы управления». Только алгоритм будет записан не в виде ветвления, а с использованием блоков «Жди темнее» и «Жди светлее». Базовая конструкиця приведена на рис. 8.33—8.35, а простейшая программа для начинающих — на рис. 8.36. Без модификаторов предполагается, что датчик освещенности подключен к первому порту, а на моторы подается максимальная мощность.

Рис. 8.33. Крепление датчика освещенности к трехколесной тележке.

Рис. 8.34. Ось для регулировки высоты датчика может быть любой длины.

Рис. 8.35. Высота датчика над поверхностью поля — от 5 до 10 мм.

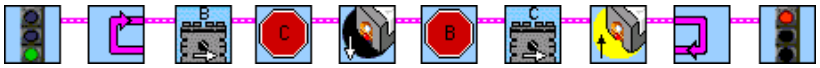


Рис. 8.36. Алгоритм движения по линии с одним датчиком освещенности.

Перед стартом ставим робота на линию так, чтобы датчик был чуть слева. По алгоритму робот плавно поворачивает направо, пока освещенность не понизится на 5 пунктов (по умолчанию). Затем поворачивает налево, пока освещенность не повысится на 5 пунктов. Движение получается похожим на «змейку».

Возможные проблемы

Перечислим трудности, которые могут возникнуть:

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

— робот проскакивает линию, не успевая среагировать. Следует понизить мощность моторов;

— робот реагирует на мелкие помехи на белом, не доезжая до черного. Надо увеличить порог чувствительности датчика (например, не на 5, а

на 8 пунктов). Вообще говоря, это число можно рассчитать. Для этого следует снять показания датчика на белом, затем на черном, вычесть одно из другого и поделить пополам. Например, (56 – 40) / 2 = 8.

Усовершенствованная программа показана на рис. 8.37.

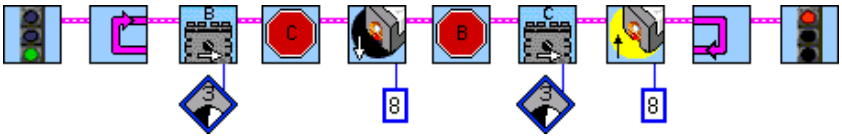


Рис. 8.37. Алгоритм движения по линии с одним датчиком освещенности: понижена скорость, увеличена разность между черным и белым.

Более устойчиво алгоритм работает, если использовать моторы с управлением скоростью –100...100. В этом случае есть возможность отрегулировать плавность поворота в соответствии с кривизной линии (рис. 8.38).

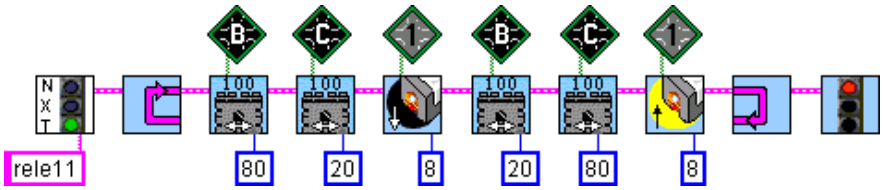


Рис. 8.38. Алгоритм движения по линии с одним датчиком освещенности: улучшено управление моторами.

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

П-регулятор

И, наконец, для сравнения надо посмотреть, как будет работать П-регулятор для одного датчика. Этот пример уже приводился в главе 7. Но его стоит повторить с некоторыми дополнениями (рис. 8.39).

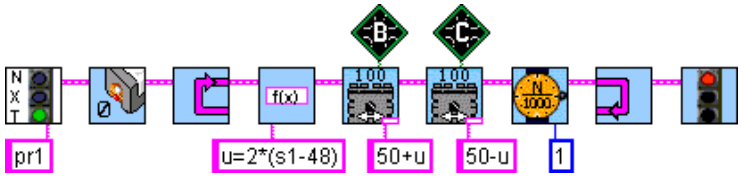


Рис. 8.39. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе.

Число 48, использованное в формуле управления *u*, — это среднее арифметическое показаний датчика освещенности на черном и на белом, например (40 + 56) / 2 = 48. Однако показания датчиков часто меняются по разным причинам: другая поверхность, изменение общей освещенности в помещении, небольшая модификация конструкции и т.п. Поэтому имеет смысл научить робота самостоятельно вычислять среднее арифметическое, т. е. 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. Например, 56 – 7 = 49 можно считать значением серого (рис. 8.40).

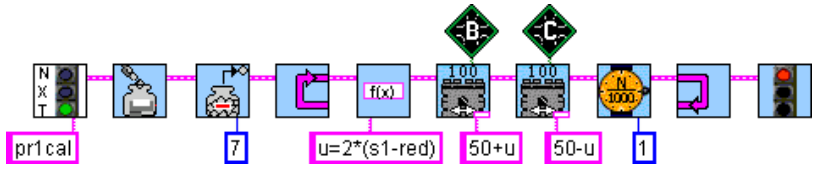


Рис. 8.40. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе с предварительной калибровкой (определением значения серого).

```
task main()
{
  int u, v=50;
  float k=2;
  int red=SensorValue[S1]-7;
  while(true)
  {
    u=k*(SensorValue[s1]-red);
    motor[motorB]=v+u;
    motor[motorC]=v-u;
    wait1Msec(1);
  }
}
```

По умолчанию значение освещенности с датчика на порту 1 считывается в красный контейнер, после чего оно уменьшается на число 7, и в формуле управления *u* используется уже измененное значение красного контейнера red. Если указывать все модификаторы, программа будет выглядеть так, как показано на рис. 8.41.

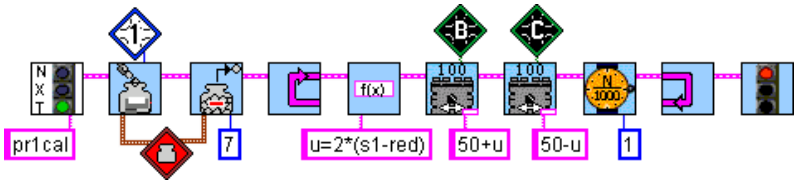


Рис. 8.41. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе — с модификаторами.

Надо иметь ввиду, что такой способ калибровки не учитывает все возможные варианты, а только экономит время на программирование и отладку. Если же времени достаточно, есть другой способ, при котором действительно производится расчет среднего арифметического показаний датчика освещенности на черном и на белом (рис. 8.42).

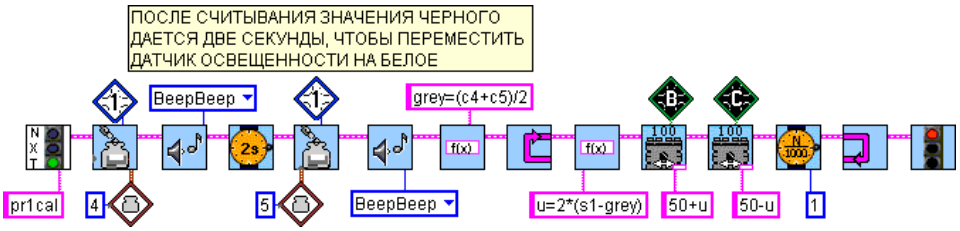


Рис. 8.42. Алгоритм движения по линии с одним датчиком освещенности на пропорциональном регуляторе с расчетом значения серого.

```
task main()
{
  int u, v=50;
  float k=2;
  int c4=SensorValue[S1];
  PlaySound(soundBeepBeep);
  wait1Msec(2000);
  int c5=SensorValue[S1];
  PlaySound(soundBeepBeep);
  int grey=(c4+c5)/2;
  while(true)
  {
    u=k*(SensorValue[S1]-grey);
    motor[motorB]=v+u;
    motor[motorC]=v-u;
    wait1Msec(1);
  }
}
```

Предложенный алгоритм обладает некоторым неудобством: при запуске потребуется быть внимательным и не пропустить звукового сигнала, после которого робота надо переместить так, чтобы датчик освещенности оказался над белым полем. Понятно, что в начале следует

поместить робота точно над черной линией. В контейнере с номером 4 (обозначается *c*4) будет сохранено значение черного, в контейнере с номером 5 (*c*5) — значение белого. В переменную *grey* помещается значение серого, которое используется в регуляторе. Сразу после второго звукового сигнала робот начнет движение.

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

Рассмотрим простейший пример с дополнительным датчиком касания, подсоединенным ко второму порту. Запустить программу имеет смысл, аккуратно установив тележку датчиком освещенности над черной линией (рис. 8.43).

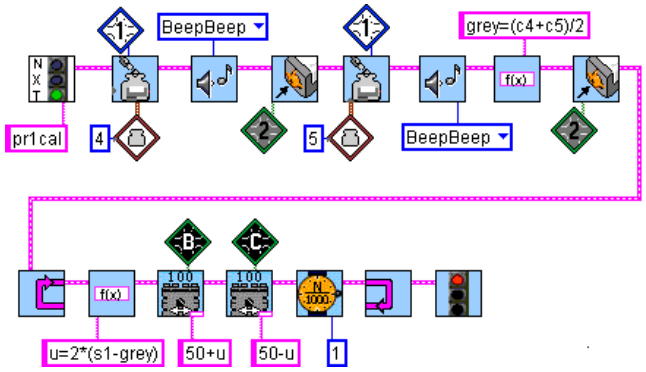


Рис. 8.43. Калибровка датчика освещенности с ожиданием касания.

```
task main()
{
  int u, v=50;
  float k=2;
  int c4=SensorValue[S1];
  PlaySound(soundBeepBeep);
  while(SensorValue[S2]==0);  // Жди, пока не нажато
  wait1Msec(100);             // Защита от залипаний
  while(SensorValue[S2]==1);  // Жди, пока нажато
  wait1Msec(100);
  int c5=SensorValue[S1];
  PlaySound(soundBeepBeep);
  int grey=(c4+c5)/2;
  while(SensorValue[S2]==0);
  wait1Msec(100);
  while(SensorValue[S2]==1);
  while(true)
  {
```

```
    u=k*(SensorValue[S1]-grey);
    motor[motorB]=v+u;
    motor[motorC]=v-u;
    wait1Msec(1);
  }
}
```

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

Аналогичный опыт можно провести, используя датчик расстояния вместо датчика нажатия. Преимущество здесь в том, что старт робота будет осуществляться бесконтактно. Это поможет стартовать в точно выбранном положении. Только надо быть внимательным и несвоевременно не провести рукой возле датчика расстояния (рис. 8.44).

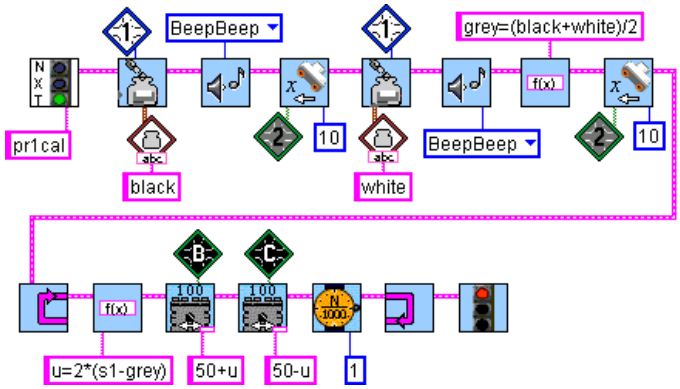


Рис. 8.44. Калибровка датчика освещенности с ожиданием объекта (руки).

В примере использованы именованные контейнеры (black и white), которые по сути являются переменными, как в обычном языке программирования. Обратите внимание, что звуковой сигнал между двумя ожиданиями изменения расстояния способствует тому, чтобы робот не среагировал дважды подряд на одно приближение руки.

Полученный опыт стоит применить для окончательного выталкивания кеглей из круга, если таковые еще остались на границе. Доработайте самостоятельно алгоритм, приведенный на рис. 8.31.

Поле для следования по линии

Более интересную траекторию, чем окружность, стоит сделать самостоятельно на светлой поверхности достаточно большой площади с помощью той же черной изоленты. В качестве поверхности подойдет лист фанеры или оргалита, обратная сторона листа линолеума, белая

клеенка и многое другое. Размеры поля желательно делать не меньше, чем 100 × 150 см. При разметке траектории следует учесть отступ от линии до края поля не менее 20 см, чтобы колеса робота не съезжали с трассы во время движения.

Имея определенный навык, можно наклеить изоленту так, что получится замкнутая кривая. Если не получается с одним куском изоленты, смело пользуйтесь ножницами, чтобы изгибы с малым радиусом кривизны составить из нескольких кусочков. Для начала не стоит рисовать слишком резких поворотов. Линию можно составить как из одной, так и из двух и даже трех полос изоленты. Тогда роботу будет легче ориентироваться и не съехать с курса. Помимо изоленты может быть использована матовая черная самоклеющаяся пленка. И, наконец, оптимальное решение — печать графического файла на баннерной ткани. Стоимость такой печати обычно не превосходит 400 руб. за 1 м^2. Небольшое поле для движения по линии приведено на рис. 8.45.

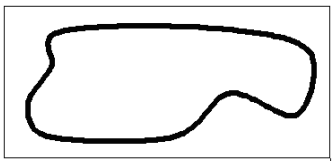


Рис. 8.45. Пример самодельного поля для движения по линии.

На рис. 8.46 приведен пример траектории для состязаний по правилам Открытого турнира на кубок Политехнического музея (г. Москва). Ширина линии составляет 5 см, а минимальный радиус кривизны — 30 см. Актуальные регламенты состязаний размещены на сайте *http://railab.ru*. Регламент состязаний «Гонки по линии» и само поле в векторном формате можно найти на сайте *http://myrobot.ru*[1].

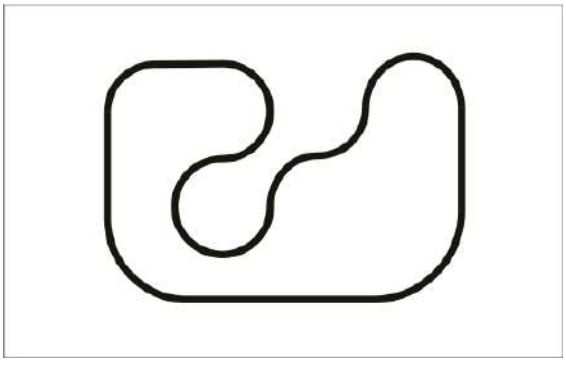


Рис. 8.46. Поле для состязаний «Гонки по линии».

[1] http://myrobot.ru/sport/index.php?n=Reglaments.LineFollowing

Путешествие по комнате

Маленький исследователь

Естественно, нормальная среда обитания для робота, построенного из домашнего конструктора, — это комната с мебелью. И для начала неплохо было бы научиться путешествовать по ней, по возможности не натыкаясь на предметы и не застревая.

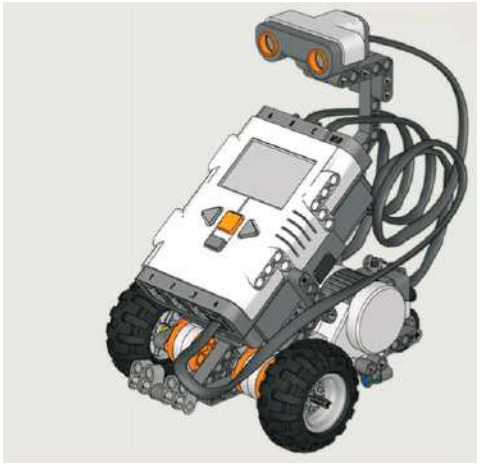


Рис. 8.78. Маленький исследователь из набора 9797 с ультразвуковым датчиком.

Подходящая конструкция для такого робота — это трехколесная тележка с установленным ультразвуковым датчиком наверху (рис. 8.78). Данный датчик следует расположить строго горизонтально относительно пола, иначе любая соринка может быть воспринята как непреодолимое препятствие или, наоборот, что-то серьезное не будет замечено.

Более простой вариант конструкции (рис. 8.79) можно построить на основе тележки, которая рассматривалась в главе 3. Программа очень похожа алгоритм на путешествия в круге. Меняется лишь датчик (рис. 8.80).

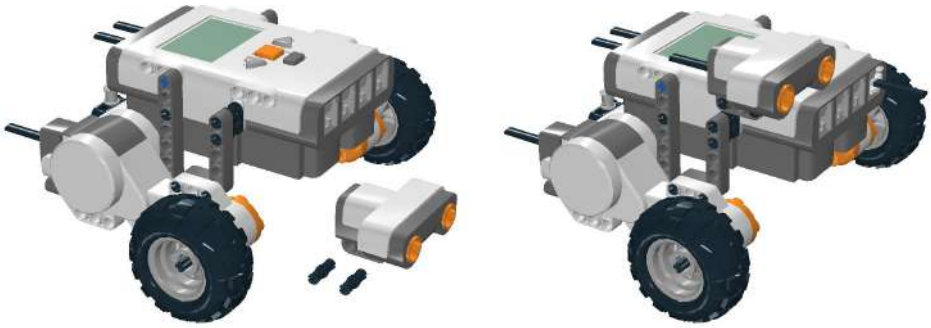


Рис. 8.79. Датчик, прикрепленный к корпусу тележки, должен смотреть строго горизонтально.

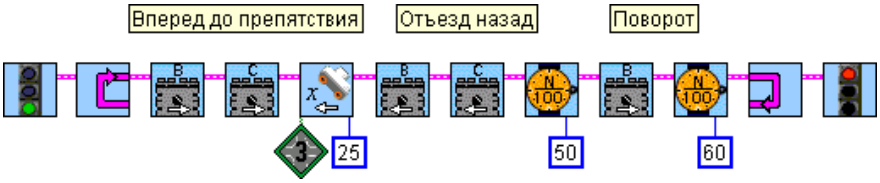


Рис. 8.80. Алгоритм путешествия по комнате.

Можно сделать несколько короче, если заменить отъезд назад с поворотом на месте одним действием: плавным поворотом задним ходом (рис. 8.81).

Рис. 8.81. Алгоритм путешествия по комнате с поворотом задним ходом.

Правда, в некоторых условиях такой поворот может привести к небольшой аварии, так что будьте с ним осторожнее. Кстати, и в первой и во второй программах следует подобрать свои параметры для расстояния до предметов и длительности поворотов.

Защита от застреваний

Присмотревшись к поведению робота повнимательнее, можно заметить, что не все предметы на пути попадают в его поле зрения. Например, если препятствие достаточно низкое, то ультразвуковой датчик его может не заметить. Или покрытая тканевой обивкой поверхность вовсе поглощает ультразвуковые сигналы, т. е. не отражает их на чувствительный элемент.

Не увидев препятствие (тапок или ножку стула), робот может застрять и будет бесконечно пытаться продолжать движение вперед. Однако если поразмыслить, можно прийти к выводу, что в комнате движение не должно быть бесконечным. Скажем, от одной стенки до другой робот может доехать за 10 с. Если за это время он не увидит ни одного препятствия, можно с уверенностью утверждать, что произошло застревание и надо предпринять экстренные меры. Что же делать? Ничего особенного. Просто отъехать назад и развернуться. Поможет в этом «сторожевой таймер» (рис. 8.82). Такие устройства применяются в микроконтроллерах и защищают их от зависаний.

Рис. 8.82. Если на сторожевом таймере «натикает» 10 с, включается защита от застреваний.

Можно заметить в нашей программе повторяющиеся группы блоков. Их стоит объединить в подпрограмму, которая будет осуществлять отъезд назад с разворотом (рис. 8.83). Таким образом, подпрограмма отъезда будет вызываться в двух случаях: 1) при наличии препятствия, 2) при срабатывании сторожевого таймера.

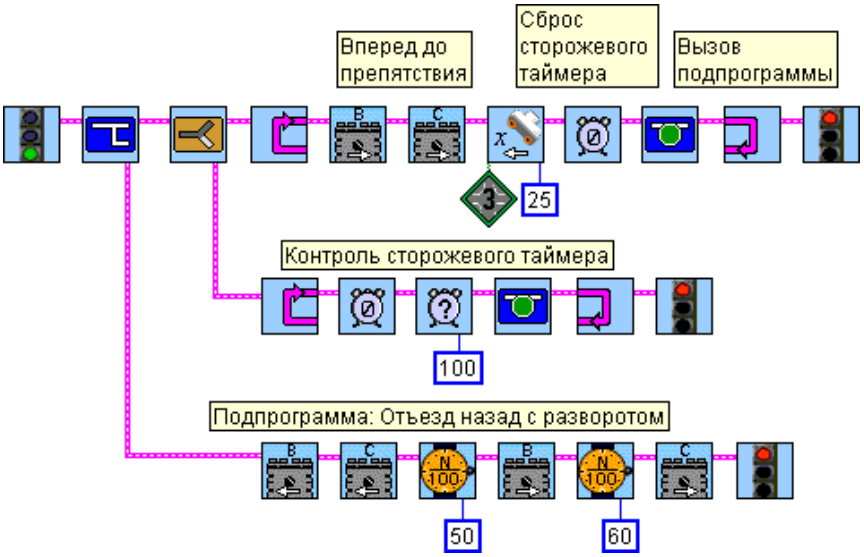


Рис. 8.83. Защита от застреваний с использованием подпрограмм.

Но в этой программе есть один существенный недостаток. Из двух параллельных задач происходит обращение к одним и тем же моторам. Если эти обращения совпадут во времени, может возникнуть непредвиденное поведение робота. В чем-то это даже интересно. Но наиболее корректная программа описана в следующем разделе.

Объезд предметов

Новая конструкция

Первые шаги к объезду предметов сделаны в главе «Алгоритмы управления». Движение вдоль стены с небольшими отклонениями возможно с помощью ПД-регулятора. Однако описанный робот сможет объезжать стены только при малых отклонениях от прямой линии. При резких изгибах робот может потерять контакт со стеной и начать крутиться на месте. Эту проблему можно отчасти разрешить конструктивно.

Рассмотрим вариант, при котором на пути движения будут возникать серьезные повороты, вплоть до прямых углов. Потребуется внести модификации и в конструкцию, и в программу.

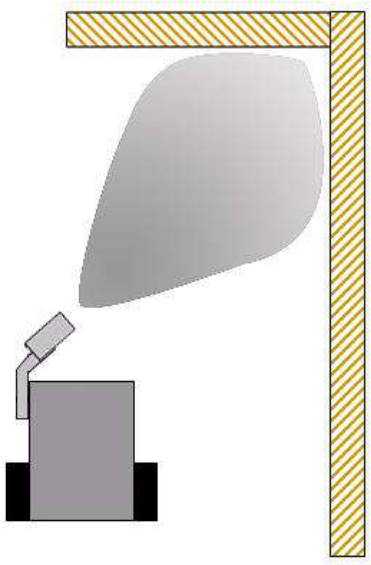


Рис. 8.89. Датчик расстояния устанавливается под острым углом к направлению движения.

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 (рис. 8.89). Это напоминает периферийное зрение человека: кое-что он может увидеть краем глаза. Учитывая это свойство, разместим датчик расстояния не перпендикулярно курсу движения, а под острым углом (рис. 8.90—8.91). 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 можно более эффективно использовать возможности дальномера.

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

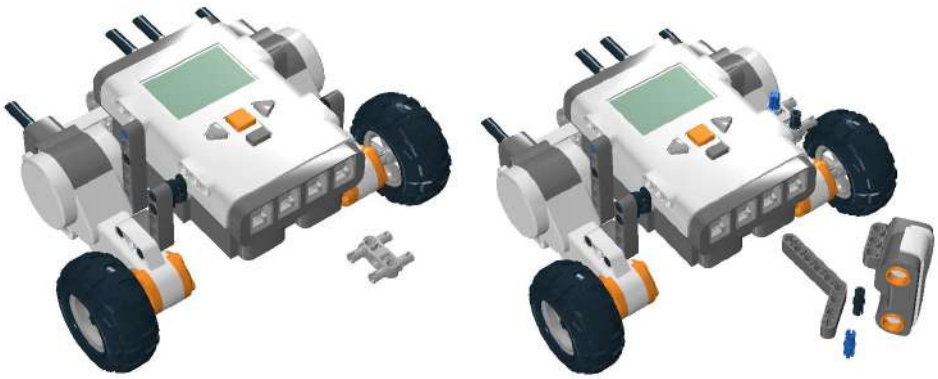


Рис. 8.90. Крепление размещается на левой стороне. Как и в первой конструкции, датчик располагается вертикально.

Рис. 8.91. Увеличенное за счет корпуса робота расстояние до стены способствует расширению области обзора.

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 кру-

титься 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» (рис. 8.92). Если робот движется на расстоянии L от стены, то и поворот, очевидно, он будет выполнять по окружности с радиусом L.

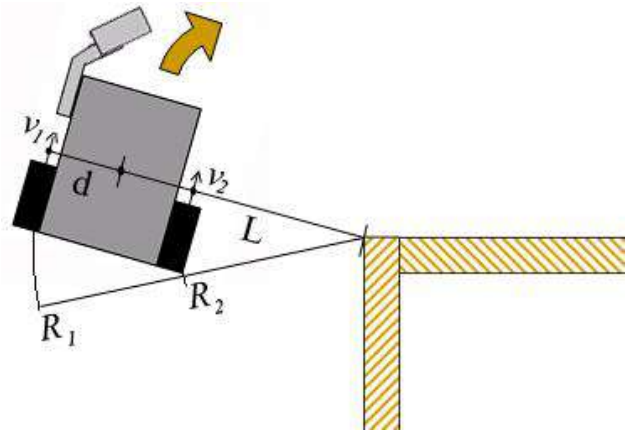


Рис. 8.92. Выполнение поворота при потере контакта со стенкой.

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, $R_1 = L + d$ и $R_2 = L - d$. Пройденные ими пути за единицу времени должны быть пропорциональны радиусам, следовательно, скорости точек крепления колес v_1 и v_2 связаны следующим отношением:

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

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

$$\frac{v+x}{v-x} = \frac{L+d}{L-d},\ \ vL + xL - vd - xd = vL + vd - xL - xd,\ \ 2xL = 2vd,$$

$$x = \frac{vd}{L},\ v_1 = v + \frac{vd}{L} = v(1+\frac{d}{L}),\ v_2 = v - \frac{vd}{L} = v(1-\frac{d}{L}).$$

Линейная скорость v 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=v*8/L;
motor[motorB]=v+u;
motor[motorC]=v-u;
```

Когда расстояние до стены становится больше $2L$ (используем такой порог видимости), т. е. открывается поворот за угол, управляющее воздействие начинает вычисляться по приведенным формулам (рис. 8.93).

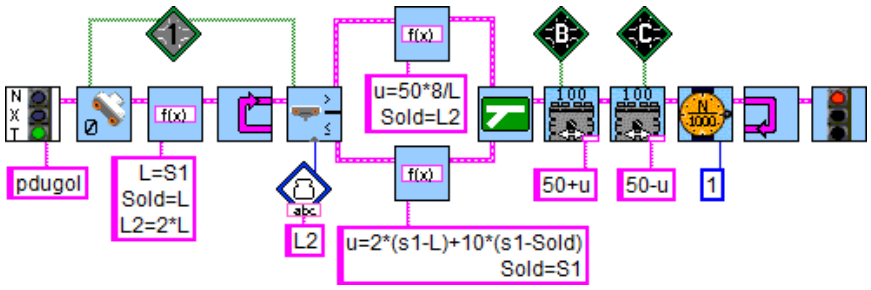


Рис. 8.93. Объезд предметов на заданном расстоянии по правилу правой руки.

```
task main()
{
  float u, k1=2, k2=10;
  int v=50, d=8, Sold, L, Snew;
  Sold=L=SensorValue[S1]; // Запомнили начальное состояние
  while(true) {
    Snew=SensorValue[S1]; // Получили показания датчика
    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);
   }
}
```

```
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.

Роботы-барабанщики

Предыстория

Идея построить робота-барабанщика из Lego появилась в 2009 году на кружке робототехники физико-математического лицея № 239 г. Санкт-Петербурга. Ребятам она понравилась и вскоре было создано несколько моделей, которые запоминали и воспроизводили ритм, импровизировали, управлялись удаленно, играли заранее записанные мелодии и даже аккомпанировали компьютеру в проигрывании midi-файлов. Через 10 месяцев робот-барабанщик получил бронзовую медаль на Всемирной олимпиаде роботов в Южной Корее (фотография награждения команды лицея «Старый барабанщик» на задней стороне обложки). Впоследствии он много путешествовал по России от Москвы до Сибири, выступая на фестивалях, выставках и форумах.

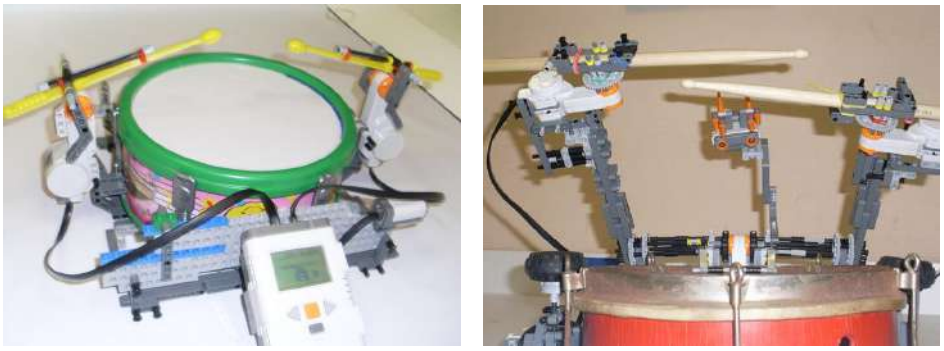


Рис. 8.96. Первые версии робота-барабанщика с игрушечным и пионерским барабанами.

Первые версии робота были сделаны целиком из деталей Lego и программировались через Robolab. Основой служил игрушечный барабан (рис. 8.96). Андроид-барабанщик был построен из пластиковых водопроводных труб (рис. 8.97), перед ним стояла целая барабанная установка, а программировался он на языках RobotC и Java.

Рис. 8.97. Андроид-барабанщик.

Алгоритмы управления роботами-барабанщиками основаны на П-регуляторах, что дает возможность повысить точность позиционирования барабанных палочек. Это описано далее.

Калибровка и удар

Конструкция для барабанщика с одной палочкой изображена на рис. 7.1. Ничего особенного, разве что балку стоит сделать подлиннее (рис. 8.98). Барабанить он будет по столу. Однако прежде чем нанести

удар, надо определиться, где именно находится поверхность «барабана». Для этого необходимо выполнить калибровку. Простейший ее вариант заключается в следующем.

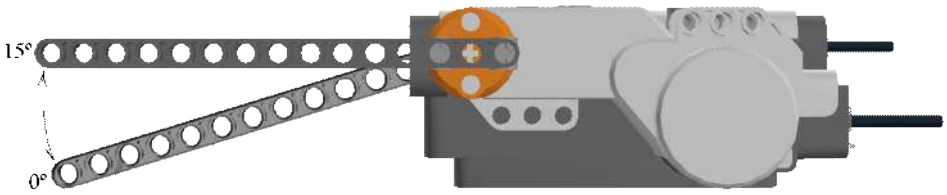


Рис. 8.98. Движения «барабанной палочки».

Робот опускает «палочку» на низкой скорости в течение достаточно большого промежутка времени, около 2 с (рис. 8.99). Поскольку скорость и мощность моторов взаимосвязаны, то, упершись в поверхность, палочка не причинит роботу никакого вреда и просто остановится. Полученное положение мотора запоминается как нулевое. Калибровка произведена.

Рис. 8.99. Простейшая импровизация одной палочкой.

Удар будет представлять из себя поднятие палочки в абсолютное положение под углом 15 градусов (замах) и опускание в нулевое положение (удар). Однако, учитывая люфт моторов Lego, а также инерцию их движения, следует прекращать опускание не в нулевом положении, а значительно раньше, например при достижении угла 10 градусов. Это позволит сократить длительность соприкосновения с поверхностью и воспользоваться энергией отскока, что значительно ускорит движение палочек. Робот получит возможность воспроизводить довольно быстрый ритм.

Управление с помощью датчика

Заменив ожидание таймера ожиданием нажатия датчика касания, можно получить управляемую игрушку (рис. 8.100). Человек будет нажимать на кнопку, а робот — послушно выполнять удар.

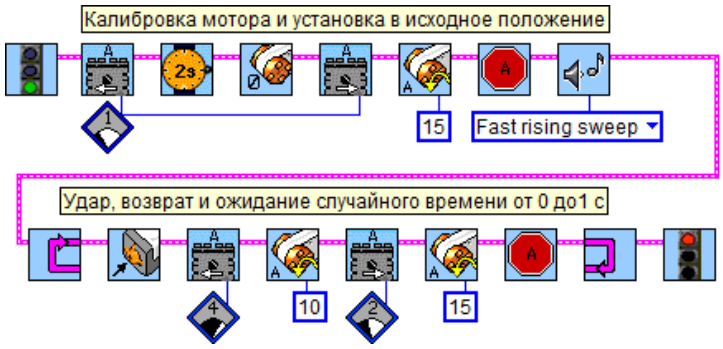


Рис. 8.100. Управление палочкой с помощью датчика касания.

В этом алгоритме стоит естественная защита от слишком частого нажимания датчика: робот будет барабанить с той скоростью, с которой успевает (рис. 8.101).

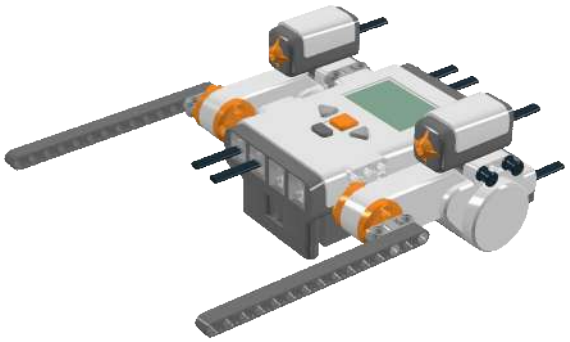


Рис. 8.101. Датчики можно прикрепить на робота или вынести отдельно.

Чтобы настроить управление двумя палочками, потребуется поместить калибровку второго мотора и удар в параллельный процесс. Вытянем алгоритм в цепочку и сдублируем его. Единственное незначительное изменение — это изменение типа и мощности управления моторами при калибровке. Поскольку уже два двигателя будут прижимать палочки к полу, то во избежание поднятия всего робота стоит понизить их мощность до 5—10 %.

Дублируя часть алгоритма, постарайтесь внимательно заменить все команды мотору A на команды мотору B (рис. 8.102).

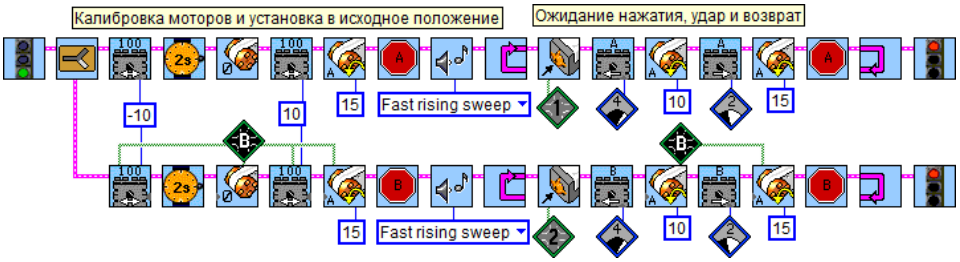


Рис. 8.102. Алгоритм управления палочками с помощью двух датчиков.

Рис. 8.103. Конструкции для управления роботом-барабанщиком: слева — на датчиках касания, справа — на гироскопических датчиках.

Ребята из команды «Старый барабанщик» сконструировали специальную перчатку с закрепленным на ней двумя датчиками касания (рис. 8.103, слева). Быть может, и читатель придумает нечто подобное. А при наличии пары гироскопических датчиков не составит труда настроить управление роботом с помощью настоящих барабанных (рис. 8.103, справа).

Создаем свой ритм

Вернемся к одномоторному барабанщику. Преобразуем удар в процедуру, которая вызывается через определенные промежутки времени. Задавая длительность этих промежутков, создадим ритмический рисунок (рис. 8.104). Лучше будет использовать длительности, кратные 0.2 секунды: 20 сотых, 40 сотых, 60 сотых и т. д. Если задать чересчур малую длительность, то либо робот не успеет произвести удар, либо удар будет слишком тихим. Эту особенность можно использовать для регулирования громкости.

Очевидно, что для повышения точности процесса надо поднимать палочку не до удара, а сразу после него.

Рис. 8.104. Создаем собственный ритимический рисунок.

Удар обладает некоторой длительностью, и ее тоже надо учитывать при определении пауз. Удобнее всего это сделать с помощью системного таймера. Команда «Жди таймер» будет предшествовать каждому удару, а очередное ожидаемое значение задаваться непосредственно перед ней (рис. 8.105).

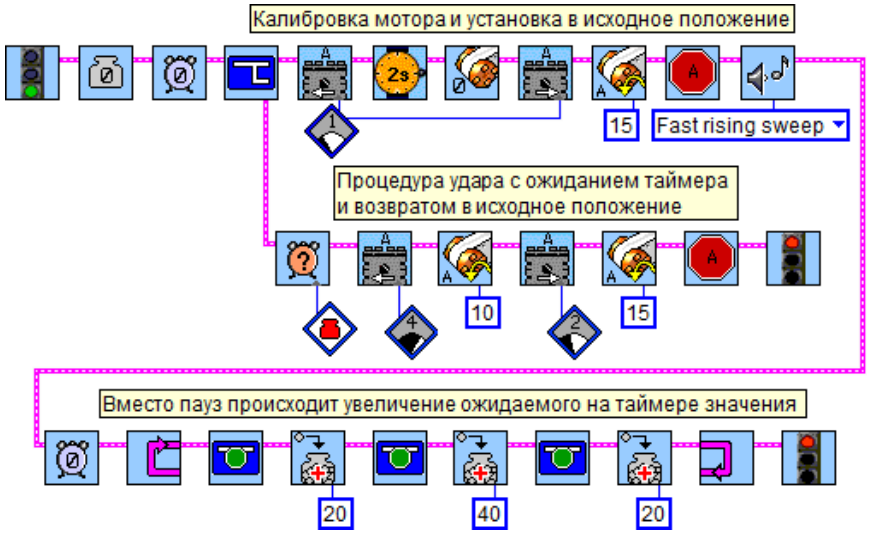


Рис. 8.105. Ритмический рисунок соответствует реальному времени.

В приведенном алгоритме много умолчаний: на энкодере используется мотор А, значение ожидания обнуляется и добавляется в красный контейнер, да и таймер тоже используется по умолчанию красный.

Лабиринт

Виртуальные исполнители

Перед решением задачи прохождения лабиринта стоит познакомиться со средой «Исполнители»[1], созданной проф. К. Ю. Поляковым из Санкт-Петербургского Государственного морского технического университета. Исполнитель робот, реализованный в ней, очень подходит для начального освоения алгоритмики. Среда полностью русифицирована и содержит Си-ориентированный язык программирования с русской и английской лексикой. Задача поиска выхода из лабиринта решается в этой среде различными способами. Наиболее интересный набор задач[2] для робота разработан учителем информатики Д. М. Ушаковым из физико-математического лицея № 239 Центральнго района Санкт-Петербурга. Среда «Исполнители» особенно актуальна для желающих в будущем освоить язык RobotC.

Также можно порекомендовать систему программирования «КуМир»[3], в которой есть исполнитель робот, способный найти выход из лабиринта. В системе «КуМир» используется школьный алгоритмический язык с русской лексикой, ориентированный на язык Паскаль. Система «КуМир» разработана в Научно-исследовательском институте системных исследований (НИИСИ) РАН по заказу Российской Академии Наук.

Обе среды распространяются бесплатно.

Полигон

На первый взгляд, построение поля лабиринта может вызвать затруднения, однако вложенные усилия вполне окупаются результатом. Поиск выхода из лабиринта является классической задачей, которую решают не только робототехники, но и программисты. Одно из самых ярких соревнований — состязания роботов Micromouse — проводится среди студентов. В нем участвуют разные роботы, совсем не из Lego, оборудованные большим числом датчиков и сложными алгоритмами. И размеры лабиринта составляют 16 × 16 квадратных ячеек.

Наш лабиринт может быть поменьше, например, размером 5 × 5 ячеек, а размер ячейки специально подобран для Lego-роботов: около 30 × 30 см. С учетом толщины стенок, как правило, сторона квадрата ячейки колеблется в диапазоне от 28 до 30 см. Число ячеек и структура

[1] *http://kpolyakov.narod.ru/school/robots/robots.htm*
[2] *http://inform239.narod.ru/robot.html*
[3] *http://www.niisi.ru/kumir/*

лабиринта могут быть любыми. Распространена столешница 150 × 150 см, что и соответствует размеру 5 × 5 ячеек. Обеспечим ее съемными внутренними стенками. Стенки лабиринта желательно сделать повыше, до 16 см, однако подойдут и стандартные, высотой 10 см (рис. 8.110).

Итак, особенность лабиринта в том, что его можно в любой момент изменить, сделав задачу более интересной.

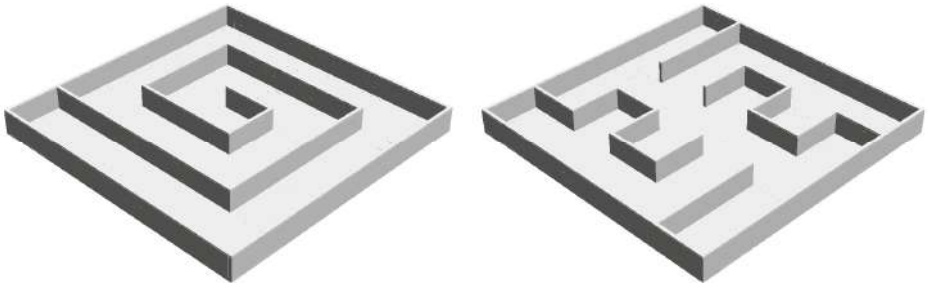


Рис. 8.110. Примеры лабиринтов.

Робот для лабиринта

Каковы характеристики робота? Поскольку приходится иметь дело с замкнутым пространством, робот не должен быть быстрым. В «прямоугольном мире» поворот на 90 градусов должен осуществляться с высокой точностью, как и проезд одной ячейки вперед. Соприкосновение со стеной для робота нежелательно, но не должно вызывать немедленного выхода его из строя. Кроме того, стоит предусмотреть способность двигаться вдоль стены при непрерывном касании. Датчики ультразвука, которые помогут определять расстояние до стен, следует располагать не на самом краю корпуса, чтобы соблюсти минимальное расстояние видимости ультразвукового датчика 5 см.

Исходя из перечисленных условий, оптимальным будет выбор конструкции компактного гусеничного робота. Построить его можно на основе одного конструктора 8547 или конструктора 9797 с добавлением двух гусениц из ресурсного набора 9695. Модель подобного робота предлагается в наборе 8547 в качестве одного из примеров. В нашей версии постараемся сделать робота более компактным за счет извлечения третьего мотора (рис. 8.111—8.117). Самую важную часть — гусеницы — следует закрепить наиболее прочно, по возможности с предельным натяжением.

Рис. 8.111. Крепление моторов с помощью 11-модульной балки и угловых соединительных штифтов.

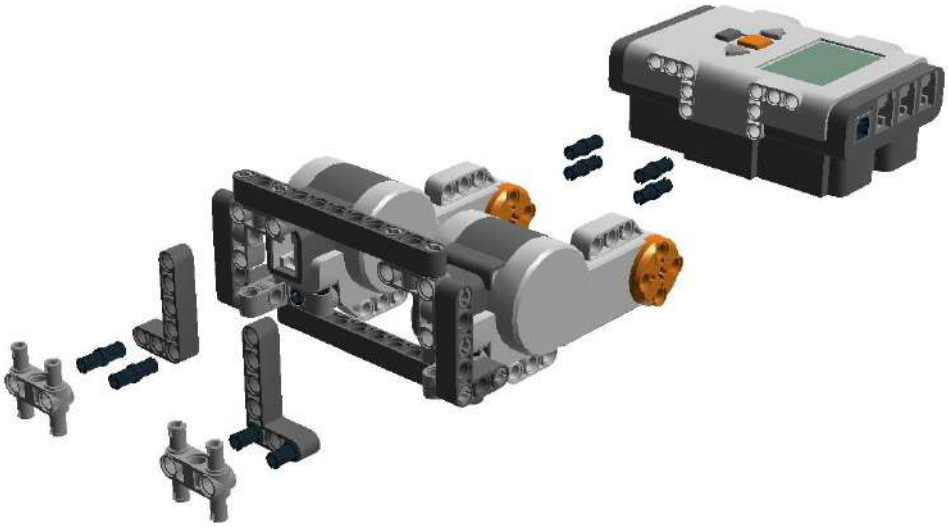


Рис. 8.112. Дополнительные уголки размером 3 × 5 для крепления блока NXT.

Рис. 8.113. Для крепления колесных дисков используются 8-модульные оси в задней части робота.

Рис. 8.114. Вертикальные балки для крепления NXT спереди могут быть любой длины. В оранжевые диски моторов вставляются 10-модульные оси.

Рис. 8.115. Колесные диски прижимаются к желтым втулкам.

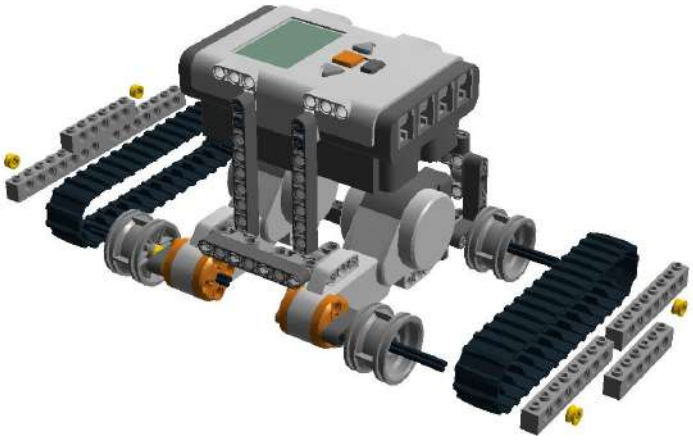


Рис. 8.116. Гусеницы можно закрепить 16-модульными балками с выступами.

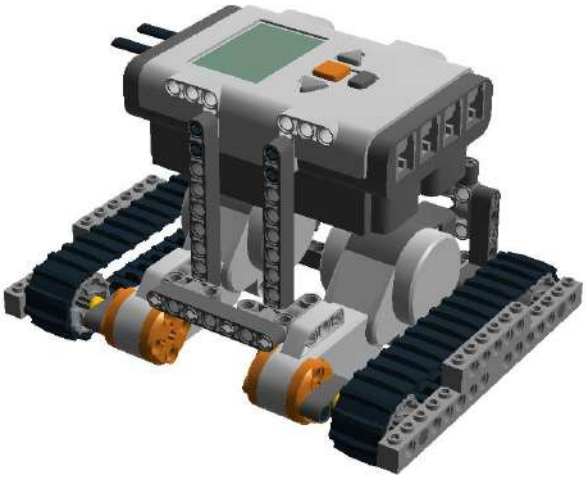


Рис. 8.117. Первая версия тележки готова.

Направление движения робота — вперед оранжевыми дисками моторов, подключенный на порты B (левый) и C (правый). Протестируйте работоспособность модели на примере программ из NXT Program.

Известный лабиринт

Первое, что научится выполнять наш робот, — точные перемещения. Для начала их будет всего три вида: проезд одной клетки вперед, поворот направо на 90 градусов, поворот налево на 90 градусов. Выделим их в три отдельных набора команд (рис. 8.118). Строго говоря, команды такого типа являются высокоуровневыми, т. е. содержат внутри себя какие-то сложные действия, исполняемые операционной системой, которые программисту не видны. В системах с виртуальным исполнителем «Робот» так и происходит. Для реализации этого в физическом мире необходимо научиться управлять роботом на «низком уровне», т. е. все команды управления роботом указывать достаточно подробно.

```
Моторы B, C вперед
Жди 1000 на энкодере B
Моторы стоп
```

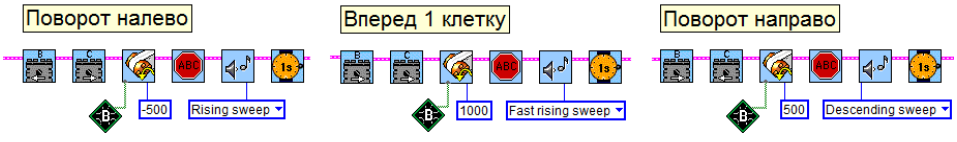


Рис. 8.118. Реализация трех базовых команд для лабиринта на языке Robolab.

Конструкция гусеничного робота такова, что проезд ячейки длиной 30 см требует поворота моторов примерно на 1000 градусов. А поворот на 90 градусов на месте выполняется при повороте одного мотора на 500 градусов вперед, а другого на –500 градусов назад.

Пример на языке Robolab показывает прохождение первых трех ячеек лабиринта (рис. 8.119). Программа составлена из трех блоков команд, повторяющихся в разном порядке.

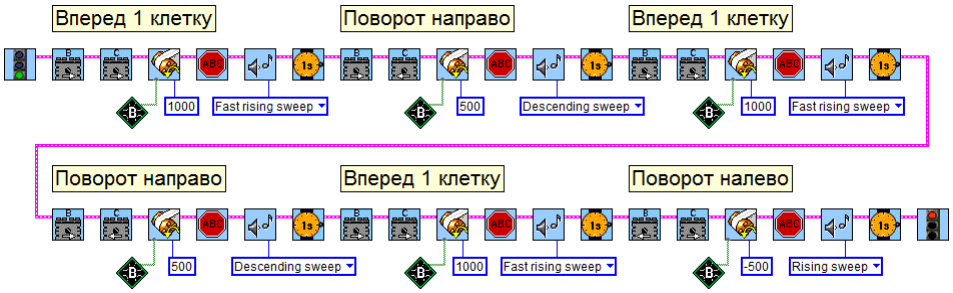


Рис. 8.119. Пример программы прохождения трех ячеек лабиринта.

Первые запуски, скорее всего, приведут к тому, что робот начнет зацепляться за неровности в стенах лабиринта своими бортами. Для защиты от зацепов существует элементарное решение, уже использованное в шагающем роботе для NXT 2.0. Это горизонтальные свободно вращающиеся колесики (рис. 8.120).

Рис. 8.120. Горизонтальные колесики устанавливаются на гладкие штифты или оси в круглые отверстия на изогнутых балках.

После установки колесиков в передней части робота можно добиться выравнивания при движении вдоль стен лабиринта даже при недостаточно точных поворотах.

Возвращаясь к программированию, придется признать, что действуя по предложенному выше алгоритму, можно составить программу прохождения лабиринта размером 5×5, которая уже не уместится в один экран Robolab. Копирование блоков сильно удлиняет программу,

что увеличивает вероятность возникновения ошибок, а при смене структуры лабиринта делает редактирование затруднительным. Опытный программист сразу догадается, что можно использовать подпрограммы!

Для ясности управления следует представить робота как классического исполнителя с тремя базовыми командами: «Вперед», «Налево», «Направо». Каждая из них является высокоуровневой и с точки зрения реального исполнителя — довольно сложной. Команда «Вперед» — это проезд одной клетки лабиринта с последующей остановкой. Команды «Налево» и «Направо» — это повороты на 90 градусов с максимально возможной точностью. Недостатки исполнения команд, связанные с люфтом и трением, придется компенсировать конструктивно.

По принципу нумерования верхних кнопок контроллера NXT три процедуры получат соответствующие номера: 3 — «Вперед», 2 — «Налево», 1 — «Направо». В процедуры стоит включить необязательные, но полезные действия, которые позволят лучше контролировать выполнение отдельных команд: остановки, задержки и звуковые сигналы.

Используя отдельные вызовы процедур, составьте программу прохождения некоторой части лабиринта (рис. 8.121).

Обратите внимание, что для каждой конструкции робота значения градусов энкодера будут свои, даже при внешне одинаковых моделях они будут зависеть от множества незаметных факторов.

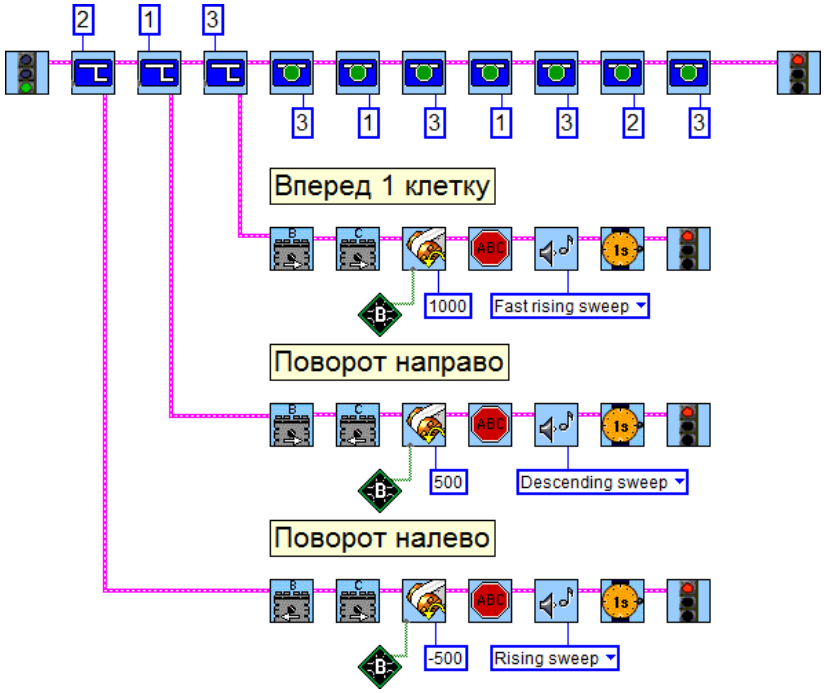


Рис. 8.121. Три базовых действия в процедурах и начало программы прохождения лабиринта.

```
void left()  // Поворот налево
{
  motor[motorB]=-100;
  motor[motorC]=100;
  nMotorEncoder[motorC]=0;
  while(nMotorEncoder[motorC]<500);
  motor[motorB]=motor[motorC]=0;
  PlaySound(soundUpwardTones);
  wait1Msec(1000);
}
task main() // Основной алгоритм
{
  while(true) {
    if(SensorValue[S1]>30) {  // Если справа проем
      right();
      forward();
    }
    else
      if(SensorValue[S2]>30) // Если спереди свободно
        forward();
      else
        left();
  }
}
```

Следующим этапом может стать защита от застреваний. Поскольку повороты физической модели неидеальны, на любом из них робот может зацепиться за угол стены. Пусть читатель самостоятельно построит защиту от застреваний, аналогичную предложенной для робота, путешествующего по комнате.

Удаленное управление

Передача данных

Контроллер NXT оснащен устройством беспроводной передачи данных Bluetooth второго класса. Это значит, что бесперебойная связь гарантирована на расстоянии до 10 м. Но качество и скорость обмена данными во многом будут зависеть от команд и алгоритмов, которые используются при программировании. Соединить можно как два контроллера между собой, так и контролер с компьютером или мобильным телефоном, оснащенным Bluetooth. Существует специальное программное обеспечение, которое с компьютера или мобильного телефона позволяет передавать на NXT команды управления подключенными к не-

му устройствами: моторами, датчиками и пр. В этом случае нет необходимости запускать какую-либо программу на NXT-приемнике, достаточно установить соединение. Если же необходимо передавать данные, то на приемнике и получателе запускаются разные программы, которые отправляют и обрабатывают полученные данные.

Обмен информацией следует разделить на четыре составляющих:

1) установка соединения,
2) передача данных или управляющих команд,
3) прием данных или выполнение управляющих команд,
4) завершение соединения.

Bluetooth-устройство, которое инициирует подключение, называется ведущим (master). Устройство, которое принимает подключение, называется ведомым (slave). К одному «мастеру» может быть подключены до трех ведомых NXT, по одному на каждый виртуальный порт (соответственно на 1-й, 2-й и 3-й).

Соединение можно устанавливать вручную, а можно и программно, эта функция реализована в RobotC.

При первом соединении двух устройств запрашивается числовой пароль, который нужно ввести на каждом из них. Впоследствии он не требуется. Подключаемые устройства узнают друг друга по имени (поэтому все имена должны быть уникальными) и способны обмениваться данными. Рассмотрим порядок действий при установке соединения вручную.

1. Включить два NXT и убедиться, что у них уникальные имена.
2. На обоих контроллерах включить Bluetooth, вследствие чего в левом верхнем углу экрана должен появиться фирменный значок.
3. На ведомом контроллере включить опцию «Виден всем» (Visibility → Visible).
4. На ведущем контроллере включить режим поиска соседних устройств (Search).
5. Из списка найденных устройств выбрать ведомый и подключиться к нему.
6. При первом подключении потребуется ввести числовой код (по умолчанию «1234») и нажать «галочку». При последующих подключениях нужный контроллер можно будет найти по имени в разделе My Contacts.
7. Выбрать любой порт: 1, 2 или 3.
8. При успешном соединении ведомый издаст звуковой сигнал (если включен звук) и на экране каждого из контроллеров рядом со значком Bluetooth появится ромбик «<>». Если подключения нет, то останется значок «<», т. е. левая половинка ромбика.

После того, как подключение состоялось, можно передавать файлы или запускать программы, поддерживающие обмен данными. При этом оба контроллера, и master, и slave на равных правах могут посылать и получать информацию.

Со времен инфракрасной связи в RCX по «почте» передавался 1 байт, а в NXT допускаются большие числа: от -2^{15} до $2^{15} - 1$, т.е. в диапазоне –32768...32767. Ноль в «почтовом ящике» означает, что ничего не было принято, поэтому отправлять его не имеет смысла.

На NXT-приемник и NXT-передатчик загружаются разные программы (рис. 8.125).

Рис. 8.125. Передача числа 5 по нажатию датчика (слева) и ответный звуковой сигнал (справа).

В общем случае перед приемом данных необходимо изначально инициализировать (т. е. обнулить) почтовый ящик, чтобы очистить его от писем, которые могли остаться от предыдущих сеансов. Команда «Жди письма» (Wait for mail) сама обнуляет его и ждет очередного письма с заданным значением. Если конкретное число не задано, ожидается письмо с ненулевым значением.

Теперь рассмотрим пример с передачей двух различных чисел (рис. 8.126).

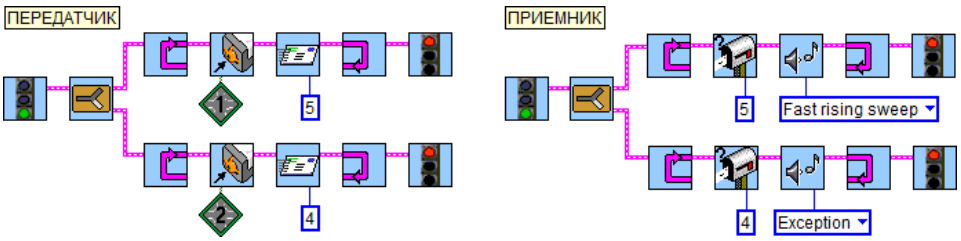


Рис. 8.126. Передача чисел 4 или 5 по нажатию соответствующего датчика и ответные звуковые сигналы.

На передатчике две параллельные задачи осуществляют отправку значений 4 или 5 в зависимости от того, какой из двух датчиков касания был нажат. Для повторного срабатывания после нажатия требуется отпустить датчик. На приемнике также две параллельные задачи «заглядывают» в почтовый ящик и вызывают соответствующий звуковой сигнал. При этом вызовы сигналов могут «наслаиваться» друг на друга,

поскольку письма с разными значениями могут приходить довольно часто.

Передача данных является довольно медленной операцией, на гарантированную доставку одного сообщения тратится 0.03 — 0.05 с. Можно, конечно, отправлять сообщения слишком часто, не заботясь о том, будут ли они приняты. В некоторых случаях работает и такой подход. Мы будем стремиться к устранению потерь пакетов при передаче. В двух приведенных выше примерах естественную задержку перед отправкой письма создает сам человек, нажимая на датчик с некоторыми перерывами.

Теперь рассмотрим пример удаленного управления мотором, в котором задержки задаются программно (рис. 8.127). Чтобы увидеть его в действии, необходимо на приемник и передатчик прикрепить по мотору, подключив каждый к порту A (8.128).

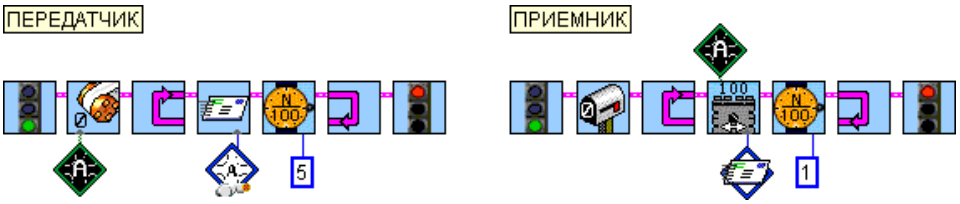


Рис. 8.127. Передача значения энкодера и управление скоростью удаленного мотора.

Для передачи значения энкодера мотора A используется соответствующий модификатор (Value of Encoder A из палитры NXT Commands), значение которого отправляется в виде письма с задержкой 0.05 с на доставку пакета. На приемнике после обнуления почты используется модификатор (Value of Mail из палитры Modifiers), который несет в себе значение полученного письма (т. е. содержимое почтового ящика). Это значение подается в качестве скорости управляемого мотора A.

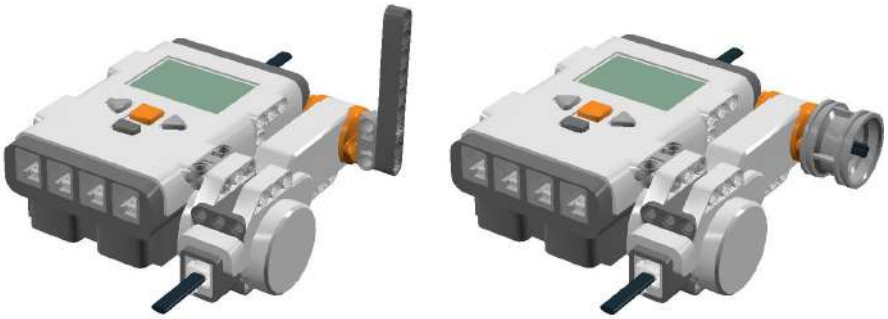


Рис. 8.128. Приемник вращает колесо со скоростью, заданной на передатчике поворотом балки.

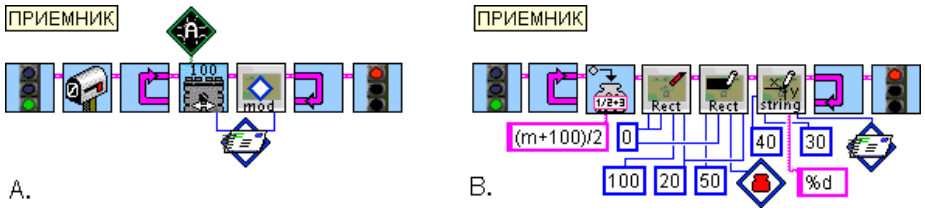


Рис. 8.129. Отображение принимаемого значения: слева — в виде числа, справа — в виде диаграммы.

Усовершенствуем приемник, добавив отображение принимаемого значения на экране в виде числа или даже диаграммы (рис. 8.130).

На рис. 8.129 слева показано, что одновременно с управлением скоростью мотора на экране приемника отображается принимаемое значение. Задержка отсутствует, поскольку вывод на экран сам по себе является довольно существенной задержкой.

На рис. 8.129 справа принимаемое значение в нижней части экрана отображается в виде изменяющегося прямоугольника, а в центральной — в виде числа. Поскольку максимальная ширина прямоугольника на экране 100 точек, что в 2 раза меньше диапазона скоростей мотора –100...100, то полученное по почте значение *m* смещается в неотрицательный диапазон и сокращается вдвое. Координата X одной пары вершин отображаемого прямоугольника находится в позиции 50 (визуальная точка отсчета), а координата противоположной пары (значение красного контейнера) колеблется в диапазоне 0...100.

Рис. 8.130. Диаграмма на экране NXT.

Читатель может добавить в программу управление скоростью мотора, показанное на рис. 8.129, самостоятельно.

Теперь рассмотрим удаленное управление положением мотора (рис. 8.131). Такая возможность полезна при взаимодействии с роботом-манипулятором, который находится на удалении от оператора.

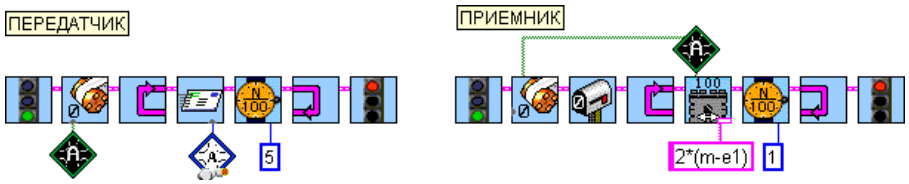


Рис. 8.131. Управление положением удаленного мотора.

Дополнительный режим джойстика

Пальцев на руке пять, а для удержания джойстика требуется два или три. Таким образом, для повышения эффективности управления мы можем ввести дополнительные датчики, которые будут отвечать, например, за изменение скорости («турбо-режим») или за удар по мячу с помощью третьего мотора. Рассмотрим вариант с датчиком касания, закрепленным на верхнем моторе (рис. 8.143). На кнопку удобно нажимать большим пальцем.

Рис. 8.143. Датчик касания на джойстике.

Алгоритм управления следует несколько изменить. До сих пор мы тратили на каждый двигатель по 8 бит информации, что при имеющейся точности движений является избыточным. Займем 1 бит для передачи показания датчика касания, у которого вариантов всего два: 0 (отпущен) и 1 (нажат). Заменив младший бит четности, мы потеряем в точности управления в 2 раза, что будет практически незаметно (рис. 8.144).

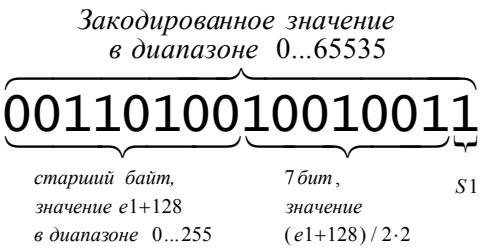


Рис. 8.144. Замена бита четности на показания датчика касания *S*1.

Чтобы избавиться от младшего бита, достаточно выполнить незатейливую процедуру: увеличенные показания энкодера разделить на 2 и умножить на 2. Поскольку деление происходит в целых числах, младший бит неизбежно будет обнулен, а его место займет значение *S*1:

$$L = (e1 + 128) \cdot 256 + (e2 + 128) / 2 \cdot 2 + S1 - 32\,768.$$

На принимающей стороне несколько изменится анализ младшего байта, а старшего останется прежним:

$v = (m + 32\,768) / 256 - 128,$
$u = (m + 32\,768) \,\%\, 256 / 2 \cdot 2 - 128,$
$k = (m + 32\,768) \,\%\, 2,$
$v = v \cdot (k + 1),$
$u = u \cdot (2 - k) / 2.$

В переменную k будет записано переданное значение $S1$. Используя ее, смоделируем режим «турбо», т. е. увеличение скорости v и понижение чувствительности к управлению u (рис. 8.145). Выбрав курс робота в обычном режиме, можно включить «турбо» нажатием кнопки, и робот выполнит рывок в нужном направлении.

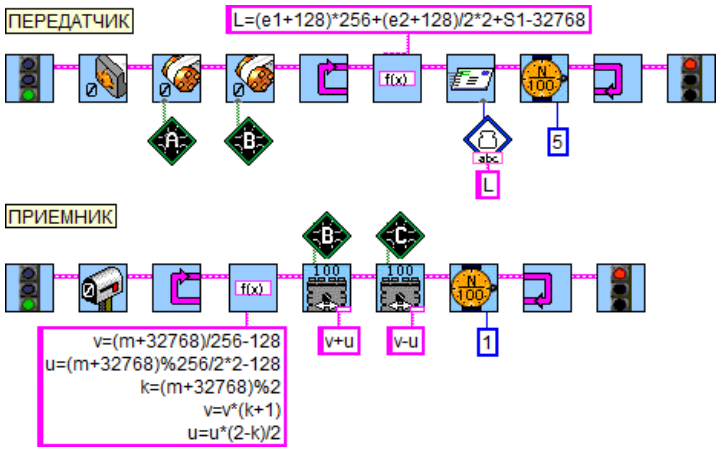


Рис. 8.145. Режим «турбо» включается датчиком касания на передатчике.

Другой пример использования дополнительного бита — это нанесение удара по мячу с помощью третьего мотора. По правилам игры в управляемый футбол, которая с 2011 г. проводится на Математико-механическом факультете Санкт-Петербургского государственного университета (СПбГУ), размер робота в момент удара по мячу не должен превышать цилиндра диаметром 22 см и высотой 22 см.

Для того, чтобы запрограммировать удар, необходимо логически отделить первый полученный сигнал к удару от всех последующих до тех пор, пока «клюшка» не будет приведена в исходное положение. В пылу игры человек может несколько раз нажать на кнопку и продержать ее в нажатом состоянии существенно дольше, чем это необходимо. Задача робота по требованию спокойно от начала до конца выполнить очередной удар и приступить к следующему только в том случае, если после завершения продолжает поступать соответствующий сигнал. Алгоритм приемника приведен на рис. 8.146. Для передатчика подойдет программа из предыдущего примера (рис. 8.145).

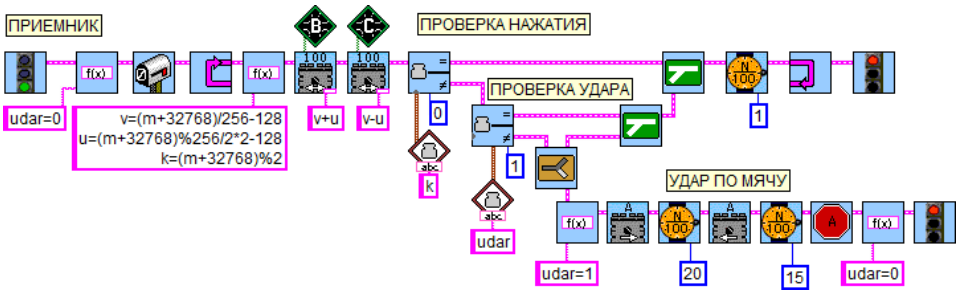


Рис. 8.146. Удар по мячу роботом с третьим мотором.

Для управления третьим мотором используется параллельная задача, вход в которую контролируется контейнером-семафором *udar*. На время выполнения удара семафор «запрещает» повторный вызов задачи и обеспечивает корректную работу программы.

Пример робота с ударным механизмом изображен на рис. 8.147. Его разработал ученик физико-математического лицея № 239 Илья Балташов. Игра в футбол с такими роботами происходит в формате 3 × 3 или 5 × 5 на поле размером 4 × 6 м. В качестве покрытия используется ковролин. Мяч для гольфа оказался наиболее подходящим для такого размера роботов, хотя можно использовать и мячик из набора 9797.

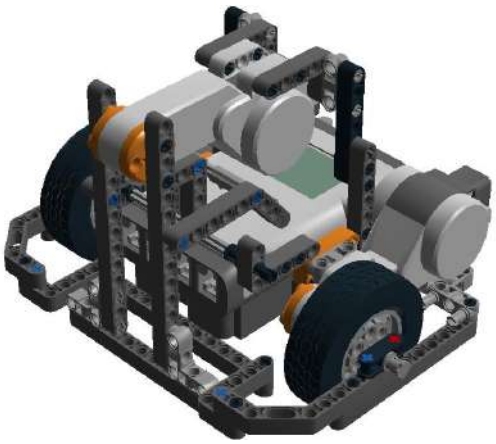


Рис. 8.147. Робот с ударным механизмом для игры в футбол.

Существует множество приложений для ноутбуков и мобильных телефонов, с помощью которых можно управлять таким роботом (например, nxtremote). Их преимущество — высокое быстродействие. Их недостаток — отсутствие возможности развития алгоритма. Если же составлять программу самостоятельно, то, используя методы кодирования, можно достичь интересных результатов во взаимодействии человека с роботом.

```
      ClearMessage(); // Может не работать в RobotC 3.0
      temp = message;
    }
    if (message!=0)
    {
      k=message-1;    // Сообщение приходит на 1 больше
      v=messageParm[1];
      u=messageParm[2];
      motor[motorA]=v+u;
      motor[motorB]=v-u;
      nxtDisplayTextLine(0, "k=%d", k);
      nxtDisplayTextLine(1, "v=%d", v);
      nxtDisplayTextLine(2, "u=%d", u);
      wait1Msec(10);
    }
  }
}
```

В среде RobotC реализованы команды подключения к другому устройству в программном режиме, а также ряд других возможностей Bluetooth, позволяющих повысить стабильность работы. Примеры находятся в папке Sample Programs\NXT\Bluetooth Communication.

Роботы-манипуляторы

Использование манипуляторов стало обыденным явлением еще в 20-м веке. На сегодняшний день ни одно крупное промышленное производство не обходится без них. Кроме того, манипуляторы устанавливают и на мобильных роботах, чтобы расширить возможности управления в труднодоступных для человека местах. При однообразных и монотонных действиях робот также может заменить человека, например раскладывать пирожные по коробочкам.

Итак, определим манипулятор как управляемое устройство, предназначенное для выполнения сложных действий, аналогичных движениям руки человека. В том числе, это механизм для управления положением предметов.

Стрела манипулятора

Для освоения управления манипулятором следует вернуться к контролю положения двигателя с помощью П-регулятора, которое описано в главе «Алгоритмы управления». Только конструкция будет несколько отличаться от описанной (рис. 8.148—8.149).

Рис. 8.148. Крепление мотора к корпусу NXT горизонтально на уровне поверхности. Штифт-полуось вставляется в вал мотора снизу.

Рис. 8.149. Первое колено манипулятора.

Стрела манипулятора расположена горизонтально. Алгоритм управления состоит из двух параллельных задач. В первой работает П-регулятор, который удерживает мотор в положении alpha. Во второй положение alpha изменяется со временем (рис. 8.150).

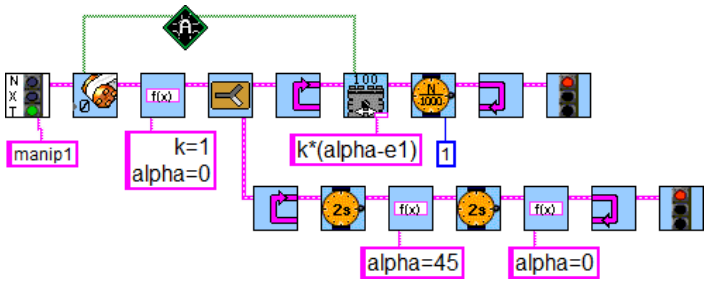


Рис. 8.150. Управление горизонтальным положением стрелы манипулятора.

Учитывая, что нулевое положение определяется на старте, задайте отрицательный угол во второй параллельной задаче и осуществите последовательный переход через нуль. Если все получается, следует приступить к установке второго мотора с захватом.

Манипулятор с захватом

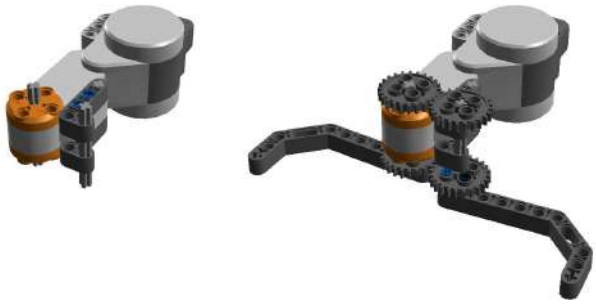


Рис. 8.151. Установка захвата на второй мотор.

Рис. 8.152. Мотор с захватом закрепляется на диске первого мотора.

Программирование робота с двумя степенями свободы (рис. 8.151—8.152) осуществляется аналогично. Новая переменная *beta* будет определять положение второго мотора. Расширьте вторую задачу, подобрав подходящие значения для открывания и закрывания захвата (рис. 8.153). Не забывайте, что стартовое положение определяет все.

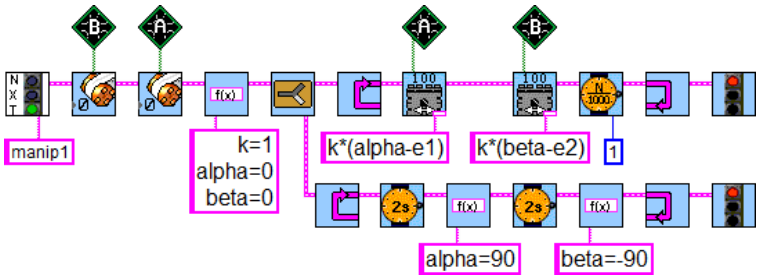


Рис. 8.153. Заготовка для управления двумя моторами на основе П-регуляторов. Необходимо продолжить цикл в параллельной задаче.

Заключение

Если читатель добрался до Заключения, проделав все опыты, изложенные в этой книге, можно быть уверенным, что впереди у него множество собственных находок и изобретений. Мы коснулись лишь малой части замечательной науки, которая все больше становится современной реальностью.

Не останавливайтесь на достигнутом, находите новые задачи и решения, создавайте своих оригинальных роботов. Быть может, эти небольшие открытия сослужат хорошую службу и нашей стране, и всему человечеству.

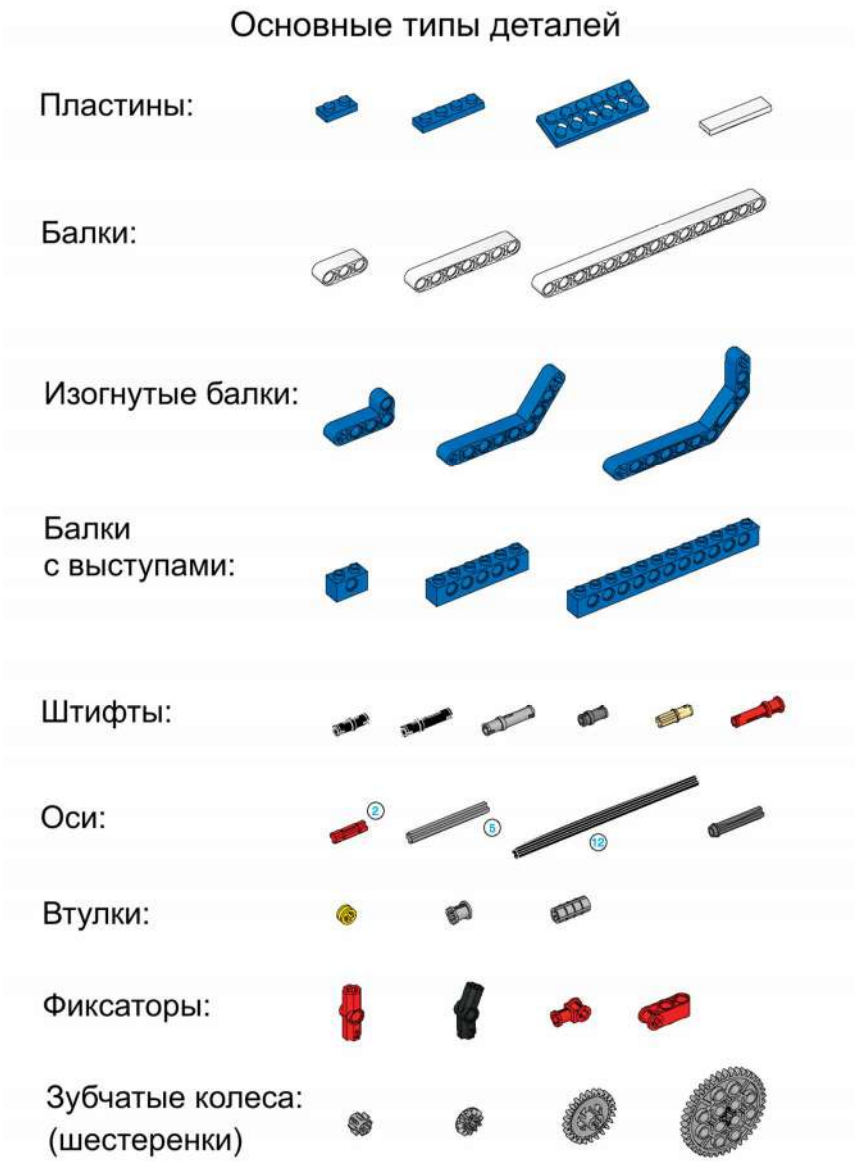
Ваши вопросы и предложения автору отправляйте по адресу robobook@mail.ru.

Литература

1. *Ананьевский М. С., Болтунов Г. И., Зайцев Ю. Е., Матвеев А. С., Фрадков А. Л., Шиегин В. В.* Санкт-Петербургские олимпиады по кибернетике. Под ред. *Фрадкова А. Л., Ананьевского М. С.* СПб.: Наука, 2006.
2. *Boogaarts M., Torok R., Daudelin J.*, et al. The LEGO Mindstorms NXT Idea Book. San Francisco: No Starch Press, 2007.
3. *Isogawa Y.* LEGO Technic Tora no Maki, Version 1.00 Isogawa Studio, Inc., 2007, //Электронный ресурс [*http://www.isogawastudio.co.jp/legostudio/toranomaki/en/*].
4. Constructopedia NXT Kit 9797, Beta Version 2.1, Center for Engineering Educational Outreach, Tufts University, 2008, //Электронный ресурс [*http://www.legoengineering.com/library/doc_download/150-nxt-constructopedia-beta-21.html*].
5. *Kelly J. F.* Lego Mindstorms NXT. The Mayan adventure. Apress, 2006.
6. *Wang E*. Engineering with LEGO Bricks and ROBOLAB. Third edition. College House Enterprises, LLC, 2007.
7. *Perdue D. J.* The Unofficial LEGO MINDSTORMS NXT Inventor's Guide. San Francisco: No Starch Press, 2007.
8. *Белиовская Л.Г., Белиовский А.Е.* Программируем микрокомпьютер NXT в LabVIEW. М: ДМК Пресс, 2010.
9. *Азимов А.* Я, робот. Серия: Библиотека приключений. М: Эксмо, 2002.

Приложения

П.1. Названия деталей



П.2. Правила состязаний

Регламент соревнований роботов «Кегельринг»[1]

(по версии Ассоциации спортивной робототехники)

10. Условия состязания:

— в наиболее короткое время робот, не выходя за пределы круга, очерчивающего ринг, должен вытолкнуть расположенные в нем кегли;

— на очистку ринга от кеглей дается максимум две минуты;

— если робот полностью выйдет за линию круга более чем на 5 секунд, попытка не засчитывается;

— во время проведения состязания участники команд не должны касаться роботов, кеглей или ринга.

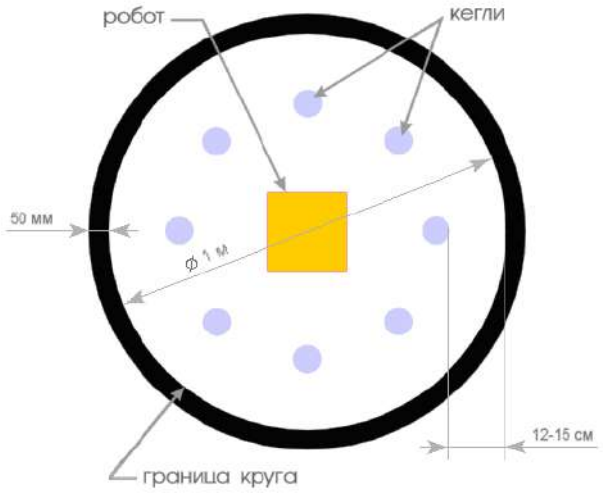


Рис. П.2.1. Поле для кегельринга.

11. Ринг:

— цвет ринга — светлый;

— цвет ограничительной линии — черный;

— диаметр ринга 1 м (белый круг);

— ширина ограничительной линии 50 мм;

12. Кегли:

— кегли — жестяные цилиндры, изготовленые из пустых стандартных жестяных банок, используемых для напитков;

[1] Идея взята с сайта *http://www.myrobot.ru*

— диаметр кегли 70 мм;
— высота кегли 120 мм;
— вес кегли — не более 50 г.

13. Робот:

— максимальная ширина робота 20 см, длина — 20 см;
— высота и вес робота не ограничены;
— робот должен быть автономным;
— во время соревнования размеры робота должны оставаться неизменными и не должны выходить за пределы 20 × 20 см;
— робот не должен иметь никаких приспособлений для выталкивания кеглей (механических, пневматических, вибрационных, акустических и др.);
— робот должен выталкивать кегли только своим корпусом;
— запрещено использование каких-либо клейких приспособлений на корпусе робота для сбора кеглей.

14. Игра:

— робот помещается строго в центр ринга.
— на ринге устанавливается восемь кеглей.
— кегли равномерно расставляются внутри окружности ринга. На каждую четверть круга должно приходиться не более двух кеглей. Кегли ставятся не ближе 12 и не далее 15 см от черной ограничительной линии. Перед началом игры участник состязания может поправить расположение кеглей. Окончательная расстановка кеглей принимается судьей соревнования;
— главная цель робота состоит в том, чтобы вытолкнуть кегли за пределы круга, ограниченного линией;
— кегля считается вытолкнутой, если никакая ее часть не находится внутри белого круга, ограниченного линией;
— один раз покинувшая пределы ринга кегля считается вытолкнутой и может быть снята с ринга в случае обратного закатывания;
— робот должен быть включен или инициализирован вручную в начале состязания по команде судьи, после чего в его работу нельзя вмешиваться. Запрещено дистанционное управление или подача роботу любых команд.

15. Правила отбора победителя:

— каждой команде дается не менее двух попыток (точное число определяется судейской коллегией в день проведения соревнований);
— в зачет принимается лучшее время из попыток или максимальное число вытолкнутых кеглей за отведенное время;
— победителем объявляется команда, чей робот затратил на очистку ринга от кеглей наименьшее время, или если ни одна команда не справилась с полной очисткой ринга — команда, чей робот вытолкнул за пределы ринга наибольшее количество кеглей.

П.3. Интернет-ресурсы по Lego Mindstorms NXT

— *http://www.mindstorms.com* (официальный сайт компании Lego)

— *http://www.mindstorms.su* (неофициальный российский сайт Lego Mindstorms)

— *http://learning.9151394.ru* (содержит вводный курс Lego Mindstorms NXT на русском языке)

— *http://www.lugnet.com* (форум пользователей Lego Mindstorms NXT)

— *http://www.nxtprograms.com* (примеры разработок роботов из Lego Mindstorms NXT)

— *http://www.legoengineering.com* (поддержка пользователей Mindstorms)

— *http://nnxt.blogspot.ru/* (робототехника для школ и вузов Нижнего Новгорода)

— *http://www.isogawastudio.co.jp/legostudio/toranomaki/en/* (LEGO Technic Tora no Maki, энциклопедия конструирования)

Языки и среды программирования для Lego Mindstorms NXT

— RobotC: *http://www.robotc.net*

— NBC/NXC (Next Byte Codes & Not eXactly C): компилятор и документация к NBC
http://bricxcc.sourceforge.net/nbc/

— Интегрированная среда разработки Bricxcc
http://bricxcc.sourceforge.net/

— LEJOS: Java for Lego Mindstorms: *http://lejos.sourceforge.net/*

— Среда LabVIEW для Lego Mindstorms NXT:
www.ni.com/mindstorms

— Обновление для Robolab 2.9 до версии 2.9.4:
http://www.legoengineering.com/patches/RL294PowerPatch_PC.zip

— Обновление для Robolab 2.9.4 с поддержкой новых датчиков сторонних производителей: *http://legoengineering.com/library/cat_view/41-applications-patches-a-firmware/43-robolab.html*

— QReal Robots, среда программирования роботов с 2D-симулятором, разработанная на матмехе СПбГУ: *http://qreal.ru*

Правила состязаний роботов

— *http://www.myrobot.ru/sport* (Мой робот: роботы, робототехника, микроконтроллеры)

— *http://railab.ru/* (лаборатория робототехники и искуственного интеллекта Политехнического музея)

— *http://wroboto.ru/* (Международные состязания роботов)

— *http://www.wroboto.org/ (*Всемирная олимпиада роботов)

— *http://239.ru/robot* (Центр робототехники физико-математического лицея №239 Центрального района Санкт-Петербурга)

Неофициальный гид изобретателя Lego Mindstorms NXT

Интернет-ресурсы по Lego Mindstorms NXT из книги David Perdue, «The Unofficial Lego Mindstorms NXT Inventor's Guide». см. сайт http://nxtguide.davidjperdue.com/

Общие ресурсы

— Обновления программ (*http://mindstorms.lego.com/en-us/support/files/default.aspx*)

— LUGNET (*http://www.lugnet.com*)

— MOC pages (*http://www.mocpages.com*)

— Brickshelf (*http://www.brickshelf.com*)

— Peeron LEGO Inventories (*http://www.peeron.com*)

— Brickset (*http://www.brickset.com*)

— NXT Programs: Fun Projects for your LEGO MINDSTORMS NXT (*http://www.nxtprograms.com/index.html*)

— MINDSTORMS NXT Building Instructions (*http://ricquin.net/lego/instructions/*)

— Technica (*http://isodomos.com/technica/technica.html*)

— Blackbird's Technicopedia (*http://www.ericalbrecht.com/technic*)

Ресурсы для программистов

— Programming Solutions for the LEGO MINDSTORMS NXT: Which approach is best for you? NBC and NXC (*http://bricxcc.sourceforge.net/nbc*)

— NBC Debugger for NXT (*http://www.sorosy.com/lego/nxtdbg*)

— BricxCC (*http://bricxcc.sourceforge.net*)

— Programmable Brick Utilities (*http://bricxcc.sourceforge.net/utilities.html*)

— leJOS NXJ (*http://lejos.sourceforge.net*)

— RobotC (*http://www.robotc.net*)

— Writing Efficient NXT-G Programs: *http://www.firstlegoleague.org/sitemod/upload/Root/WritingEfficientNXTGPrograms2.pdf*

— OnBrick NXT Remote Control (*http://www.pspwp.pwp.blueyonder.co.uk/science/robotics/nxt/)*

— NXTender (*http://www.tau.ac.il/~stoledo/lego/NXTender*)

— NXT Programming Software (*http://www.teamhassenplug.org/NXT/NXTSoftware.html*)

Ресурсы для Bluetooth

— MINDSTORMS Bluetooth Resources *http://www.mindstorms.com/bluetooth*
— NXTBluetoothCompatibilityList: *http://www.vialist.com/users/jgarbers/NXTBluetoothCompatibilityList*
— Analysis of the NXT Bluetooth-Communication Protocol: *http://www.tau.ac.il/~stoledo/lego/btperformance.html*

NXT-Блоги

— The NXT STEP (*http://www.thenxtstep.com*)
— nxtasy.org (*http://www.nxtasy.com*)

Ресурсы по автоматизированному конструированию (LEGO computer-aided design resources):

— LEGO Digital Designer (*http://ldd.lego.com*)
— Google SketchUp NXT Parts Library: *http://groups.google.com/group/LegoTechnicandMindstormsNXTParts*
— LDraw (*http://www.ldraw.org*)
— Tutorial: Setting up LDraw to Create Virtual NXT Robots: from *http://nxtblog.davidjperdue.com*
— LeoCAD (*http://www.leocad.org*)
— Bricksmith (*http://bricksmith.sourceforge.net*)
— L3P (*http://www.hassings.dk/l3/l3p.html*)
— LDView (*http://ldview.sourceforge.net*)

Методы конструирования (Building techniques)

— NXT-based Creations (*http://legoengineering.com/library/cat_view/30-building-instructions/38-nxt-based-creations.html*)
— LEGO Education Constructopedia: *http://legoengineering.com/library/doc_details/150-nxt-constructopedia-beta-21.html*

Изучаем геометрию Lego:

— *http://www.syngress.com/book_catalog/174_lego_robo/chapter_01.htm*
— LEGO Design (*http://www.owlnet.rice.edu/~elec201/Book/legos*)

— Sergei Egorov's LEGO Geartrains (*http://www.malgil.com/esl/lego/geartrains.html*)

Образовательные ресурсы

— LEGO Education (*http://www.legoeducation.com*)
— MINDSTORMS Education NXT blog: http://www.legoeducation.com/community/9/blogs/nxt/default.aspx
— LEGO ED West (*http://www.legoedwest.com*)
— LEGO Engineering (*http://www.legoengineering.com*)
— FIRST LEGO League (*http://www.firstlegoleague.org*)
— US FIRST Curriculum Collection: *http://www.usfirst.org/community/*
— Robotics Academy (*http://www-education.rec.ri.cmu.edu*)

Наборы Lego, детали Lego и заказные детали (custom hardware)

— LEGO Store (*http://shop.lego.com*)
— LEGO Education Store (*http://www.legoeducation.us*)
— BrickLink (*http://www.bricklink.com*)
— HiTechnic (*http://www.hitechnic.com*)
— Mindsensors.com (*http://www.mindsensors.com*)

Хранение деталей Lego

— Robotics Learning Store (*http://www.roboticslearning.com/store*)
— Plano Molding Company (*http://www.planomolding.com*)

Персональные вебсайты

— David J. Perdue (*http://www.davidjperdue.com*)
— Philippe Hurbain (*http://www.philohome.com*)
— Dave Astolfo (*http://www.astolfo.com*)
— Daniele Benedettelli (*http://daniele.benedettelli.com*)
— Michael Gasperi (*http://extremenxt.com/lego.htm*)
— Matthias Paul Scholz (*http://mynxt.matthiaspaulscholz.eu*)
— Steve Hassenplug (*http://www.teamhassenplug.org*)
— Laurens Valk (*http://www.laurensvalk.com*)
— Jürgen Stuber (*http://www.jstuber.net*)
— Mario Ferrari (*http://www.marioferrari.org/lego.html*)
— Miguel Agullo (*http://miguelagullo.net/technicpuppy/*)

События Lego

- World Robot Olympiad (*http://www.wroboto.org*)
- LEGO World (*http://www.legoworld.nl*)
- BrickFest (*http://www.brickfest.com*)
- NWBrickCon (*http://www.nwbrickcon.org*)
- BrickFair (*http://www.brickfair.com*)

Научное издание

Сергей Александрович Филиппов

РОБОТОТЕХНИКА ДЛЯ ДЕТЕЙ И РОДИТЕЛЕЙ
Издание 3-е, дополненное и исправленное
Утверждено к печати
Ученым советом Института проблем машиноведения РАН

Редактор издательства А. Б. Иванова
Художник О. Скворцова
Книга печатается с оригинал-макета,
подготовленного автором

Санкт-Петербургская издательская фирма «Наука» РАН
199034, Санкт-Петербург, Менделеевская линия, 1
E-mail: main@nauka.nw.ru
Internet: www.naukaspb.spb.ru
Лицензия ИД № 02980 от 06 октября 2000 г.
Подписано к печати 30.01.2013. Формат 70 × 90 1/16.
Бумага офсетная. Печать офсетная. Гарнитура Таймс.
Объем 20 усл. печ. л. Тираж 3000 экз. Стр. 319

Отпечатано в типографии ООО «Дитон»
Санкт-Петербург, Б. Сампсониевский пр., 60, литер М
Тел.: (812) 333-15-42
Факс: (812) 333-15-41

ISBN 978-5-02-038200-8