ИНТУИТ

НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

В. О. Дженжер, Л. В. Денисова

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ LEGO-РОБОТОВ НА ЯЗЫКЕ NXT-G

Учебное пособие для студентов и школьников

Москва

Национальный Открытый Университет «ИНТУИТ»

2014

УДК 004.436.2
ББК 3.30в6
В24

Дженжер В. О., Денисова Л. В.

В24 Введение в программирование LEGO-роботов на языке NXT-G. Учебное пособие для студентов и школьников: Учебное пособие / В. О. Дженжер, Л. В. Денисова — М.: Национальный открытый университет «ИНТУИТ», 2014. — 87 с., ил. — (Серия «Лицей информационных технологий»).

ISBN 978-5-9556-0164-9

В пособии рассматриваются основы программирования роботов LEGO на языке программирования NXT-G. Курс рассчитан на студентов и школьников, также будет полезен учителям информатики для организации занятий по робототехнике.

Работа проведена по заданию № 2014/367 на выполнение государственных работ в сфере научной деятельности в рамках базовой части государственного задания Минобрнауки России. Код проекта № 335.

Полное или частичное воспроизведение или размножение каким-либо способом, в том числе и публикация в Сети, настоящего издания допускается только с письменного разрешения Национального открытого университета «ИНТУИТ».

По вопросам приобретения обращаться:

ООО «ИНТУИТ.ру»

Национальный открытый университет «ИНТУИТ»,

Москва, Электрический пер., 8, стр.3.

Телефон: +7 (499) 253-9312, 253-9313, факс: +7 (499) 253-9310

E-mail: info@intuit.ru, http://www.intuit.ru

УДК 004.436.2

ББК 3.30в6

ISBN 978-5-9556-0164-9

© Национальный открытый университет «ИНТУИТ», 2014

© В. О. Дженжер, Л. В. Денисова, 2014

О проекте

Национальный Открытый Университет «ИНТУИТ» — это первое в России высшее учебное заведение, которое предоставляет возможность получить дополнительное образование во Всемирной сети. Web-сайт университета находится по адресу www.intuit.ru.

Мы рады, что вы решили расширить свои знания в области компью-терных технологий. Современный мир – это мир компьютеров и информации. Компьютерная индустрия — самый быстрорастущий сектор экономики, и ее рост будет продолжаться еще долгое время. Во времена жесткой конкуренции от уровня развития информационных технологий, достижений научной мысли и перспективных инженерных решений зависит успех не только отдельных людей и компаний, но и целых стран. Вы выбрали самое подходящее время для изучения компьютерных дисциплин. Профессионалы в области информационных технологий сейчас востребованы везде: в науке, экономике, образовании, медицине и других областях, в государственных и частных компаниях, в России и за рубежом. Анализ данных, прогнозы, организация связи, создание программного обеспечения, построение моделей процессов – вот далеко не полный список областей применения знаний для компьютерных специалистов.

Обучение в университете ведется по собственным учебным планам, разработанным ведущими российскими специалистами на основе международных образовательных стандартов Computer Curricula 2001 Computer Science. Изучать учебные курсы можно самостоятельно по учебникам или на сайте Интернет-университета, задания выполняются только на сайте. Для обучения необходимо зарегистрироваться на сайте университета. Удостоверение об окончании учебного курса или специальности выдается при условии выполнения всех заданий к лекциям и успешной сдачи итогового экзамена.

Книга, которую вы держите в руках, — очередная в многотомной серии «Лицей информационных технологий», выпускаемой НОУ «ИНТУИТ». В этой серии будут выпущены учебники по всем базовым областям знаний, связанным с компьютерными дисциплинами.

Добро пожаловать

в Национальный открытый университет «ИНТУИТ»!

Об авторах

Дженжер Вадим Олегович, кандидат физико-математических наук, доцент, заведующий кафедрой информатики и методики преподавания информатики ФГБОУ ВПО «Оренбургский государственный педагогический университет». Читает курсы: «Компьютерное моделирование», «Нечёткие системы», «Эволюционные алгоритмы», «Алгоритмы робототехники», «Программирование» и др. Ведёт школьный кружок по робототехнике в центре физико-математического образования «Архимед».
e-mail: `vdjenjer@yandex.ru`

Денисова Людмила Викторовна, кандидат педагогических наук, доцент кафедры информатики и методики преподавания информатики ФГБОУ ВПО «Оренбургский государственный педагогический университет». Читает курсы: «Информатика», «Основы робототехники», «Внеучебная деятельность школьника по информатике», «Введение в программирование» и др. Ведёт школьные кружки по робототехнике и программированию в среде Scratch на базе центра физико-математического образования «Архимед».
e-mail: `lv-denisova@yandex.ru`

Оглавление

Введение

Изучение робототехники сегодня начинается уже со школьной скамьи. Однако не всем школам так повезло, и тому есть ряд объективных причин. Во-первых, стоимость одного робототехнического комплекса (здесь и далее имеется в виду LEGO Mindstorms NXT 2.0) превышает стоимость средней компьютерной системы. Во-вторых, руководить занятиями робототехники должен высококвалифицированный педагог, одинаково хорошо разбирающийся и в техническом конструировании, и в микроэлектронике, и в программировании. Подготовка таких специалистов-педагогов сегодня только начинается. Ну и в-третьих, русскоязычные учебники, вышедшие из печати на сегодняшний день, можно пересчитать, по-видимому, на пальцах одной руки.

Настоящее пособие было подготовлено по результатам работы со студентами, получающими специальность учителя информатики. Авторы ставили перед собой задачу дать начальные понятия программирования роботов LEGO Mindstorms NXT 2.0 на языке NXT-G. Кроме того, отдельные главы пособия были использованы на курсах по повышению квалификации и переподготовке учителей информатики на кафедре информатики и методики преподавания информатики Оренбургского государственного педагогического университета.

Не так давно появилась более новая версия робототехнического комплекса LEGO — Mindstorms EV3. Однако имеющийся в российских школах и центрах технического творчества парк роботов Mindstorms NXT 2.0 так велик, что было бы неправильным сбрасывать его со счетов.

Программирование сегодня уже не так привлекательно для школьников как 15–20 лет назад. Это замечают многие школьные и вузовские преподаватели информатики. Нужны новые средства для мотивации учеников в этой области. По нашему мнению, робототехника сегодня как раз и является таким средством. Поэтому в пособии упор делается на алгоритмической составляющей робототехники.

Обращаем внимание студентов на то — и это очень важно! — что при работе с пособием следует не только выполнять задания, но и обяза-

тельно набирать и проверять задачи из примеров. Это поможет лучше понять язык и особенности программирования роботов.

Мы рассматриваем программирование роботов в англоязычной версии среды NXT-G, так как она распространяется свободно.

Идеи многих заданий были почерпнуты в [1–4, 6], авторам которых мы выражаем глубокую признательность.

Для оценки собственной подготовки учащиеся могут использовать тесты, разработанные к каждой теме. В конце курса предусмотрено прохождение итогового тестирования. Самостоятельно пройти тесты можно на сайте Национального открытого университета «ИНТУИТ» по адресу `www.intuit.ru`.

Тема 1

LEGO Mindstorms NXT 2.0. Обзор

Проводится поверхностный обзор робототехнического комплекса, включающего конструктор LEGO Mindstorms NXT 2.0 и среду программирования NXT-G.

Цель: познакомиться с основными компонентами конструктора LEGO Mindstorms NXT 2.0, интерфейсом среды NXT-G и научиться создавать простейшую программу «Hello, world!».

1.1 Состав конструктора LEGO Mindstorms NXT 2.0. Технология NXT

В этом разделе приводятся краткие сведения о составе конструктора LEGO Mindstorms NXT 2.0 и NXT-технологии, основанные на руководстве пользователя, входящем в комплект 8547. Если у Вас имеется указанное руководство, и Вы уже ознакомились с ним, то можно сразу перейти к разделу 1.2.

NXT является интеллектуальным, управляемым компьютером роботом на базе элементов LEGO и системы MINDSTORMS.

> Система MINDSTORMS получила своё название благодаря книге Сеймура Пейперта «Переворот в сознании: Дети, компьютеры и плодотворные идеи», в которой автор анализирует способы формирования мышления детей и роль компьютеров в этом процессе.

Основа конструктора LEGO Mindstorms NXT 2.0 — программируемый блок NXT (его ещё называют «кирпичом»), интерактивные сервомоторы и несколько датчиков. В состав комплекта 8547 входят:

- два датчика нажатия — кнопки;
- датчик цвета — RGB-датчик, позволяющий роботу различать цвета и измерять яркость окружающего света;
- ультразвуковой датчик — «глаза» робота, позволяющие ему измерять расстояние до объекта.

В образовательной версии конструктора имеется также датчик звука, который измеряет уровень громкости звука, а вместо RGB-датчика присутствует датчик освещённости. Существуют и другие датчики, которые выпускаются отдельно от комплекта, например, для измерения температуры, pH среды, гироскопический датчик, компас и др.

Все датчики и моторы подсоединяются к NXT-блоку через порты входа и выхода посредством чёрных шестипроводных кабелей. Сенсоры подключаются к входным портам под номерами 1–4, а моторы — к выходным портам, имеющим на блоке NXT названия A, B, C.

Производитель рекомендует для подключения сенсоров и моторов использовать стандартные порты:

Порт 1: Датчик касания

Порт 2: Датчик касания

Порт 3: Датчик цвета

Порт 4: Ультразвуковой сенсор

Порт A: Мотор для дополнительных функций

Порт B: Мотор для движения

Порт C: Мотор для движения

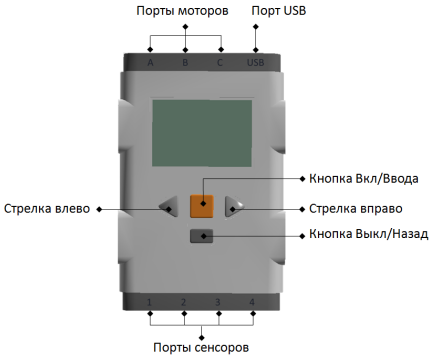


Рис. 1.1. *Внешний вид блока NXT и его разъёмы*

В нашем учебном курсе мы придерживаемся этих рекомендаций. На самом деле можно подключать сенсоры в произвольные порты. Соблюдать стандартные порты обязательно при работе с меню *Try Me* (см. ниже).

Блок NXT работает от шести батарей типа AA. В образовательную версию набора входит аккумуляторная батарея.

Работа с меню NXT происходит при помощи четырёх кнопок (рис. 1.1):

Серые треугольники — кнопки «вперёд» (вправо) и «назад» (влево) — позволяют перемещаться внутри меню до нужного пункта;

Оранжевый квадрат соответствует клавише ввода; она же используется для включения робота;

Тёмно-серый прямоугольник — «отмена» или переход назад к предыдущему пункту. Эта же кнопка используется для выключения робота. Для выключения NXT нажимайте кнопку до тех пор, пока на экране не появится надпись `Turn off?`, после чего для подтверждения выключения нажмите оранжевую кнопку. Также можно нажать и удерживать тёмно-серую кнопку до полного выключения NXT.

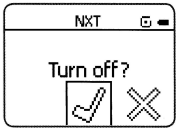


Рис. 1.2. *Экран отключения блока NXT*

Задание 1[1] Научитесь включать и выключать свой NXT. Попробуйте выключить NXT двумя способами, описанными выше.

Вся информация на дисплее NXT отображается на английском языке. При включении экран NXT выглядит как на рис. 1.3:

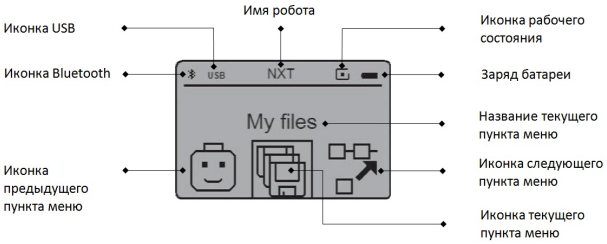


Рис. 1.3. *Экран блока NXT после включения*

	Bluetooth включен, но NXT не определяется другими устройствами Bluetooth.
	Bluetooth включен, и NXT определяется другими устройствами Bluetooth.
	Bluetooth включен, и NXT подключен к устройству Bluetooth.

	USB подключен и работает нормально.
	USB подключен, но имеются неполадки.

Если NXT работает нормально, то иконка рабочего состояния крутится. Если вращение иконки прекратилось, значит NXT завис и требует перезагрузки.

По умолчанию роботу присвоено имя `NXT`. Имя можно изменить. Как это сделать см. в п. 1.2.

Задание 1^2 Включите NXT. Изучите информацию, которая отображается на экране NXT. Перемещайтесь по главному меню NXT при помощи стрелок. Выключите NXT.

Рассмотрим структуру меню NXT.

My Files (Мои Файлы). Здесь хранятся все файлы NXT.

- **Software files (Файлы программ).** Здесь хранятся скомпилированные файлы программ, загруженных пользователем. Отсюда происходит запуск программ на выполнение.
- **NXT files (Файлы NXT).** Служебные файлы NXT.
- **Sound files (Звуковые файлы).** Здесь содержатся различные звуковые файлы, как встроенные, так и записанные пользователем. Вы можете записать собственные звуки, чтобы затем использовать их в программах.
- **Datalog files (Файлы данных).** Содержит данные, собранные NXT при помощи меню NXT Datalog. Данные можно считывать во время работы программы или передавать по Bluetooth другим устройствам.

NXT Program (Программа NXT). Этот раздел позволяет программировать NXT прямо на блоке, без помощи компьютера. Можно создавать короткие программы из не более чем пяти команд. Подходит для начального знакомства с возможностями NXT. Однако сколько-нибудь сложные задачи здесь не решаются.

NXT Datalog (Данные NXT). Получает, отображает на экране и записывает данные с внешних устройств (датчиков) в файлы, которые затем хранятся в меню Datalog files.

View (Обзор). Позволяет отображать на экране данные, полученные с внешних устройств. Но, в отличие от NXT Datalog, не записывает эти данные в файлы. Может использоваться для тестирования и калибровки датчиков.

Bluetooth. Позволяет создать канал беспроводной связи между NXT и другими устройствами с поддержкой Bluetooth. Служит для загрузки программ без помощи USB-кабеля, обмена программами с другими NXT, дистанционного управления другими NXT (не более трёх) или управления NXT при помощи других устройств, например, с телефона.

Settings (Настройки). Настройка различных параметров блока NXT.

- **Volume (Громкость).** Устанавливает громкость динамиков.
- **Sleep (Сон)**. Настраивает параметры автоматического отключения при простое.
- **NXT Version.** Позволяет узнать текущие версии программных и аппаратных средств.
- **Delete files (Удаление файлов).** Служит для удаления файлов. Будьте осторожны: удаляет все файлы из выбранного каталога!

Try Me (Попробуй). Подключив датчики и моторы к соответствующим портам, можно выполнить несколько готовых программ и познакомиться с возможностями конструктора.

Пример 1_1 Настроим NXT так, чтобы он автоматически выключался через две минуты простоя. Для этого выберем из главного меню *Settings\ Sleep*. Затем при помощи стрелок выбираем нужное значение (текущее значение отображается на экране) и нажимаем на ввод (оранжевая кнопка). Если теперь не работать с NXT в течении двух минут, то он выключится сам. Обратите внимание на вариант `Never` (Никогда): в этом режиме NXT не будет выключен, пока Вы сами этого не сделаете. Но это может привести к более быстрому разряду батарей.

Пример 1_2 Попробуем проиграть звуковые файлы на блоке NXT. *Главное меню\ My Files\ Sound files\ Good Job*. В окне отображает-

ся имя выбранного файла и варианты работы с ним, которые можно пролистать при помощи стрелок:

- `Run` — запустить выбранный файл на выполнение (иконка рабочего состояния). Если выбрать этот вариант, звуковой файл будет воспроизведён. В процессе работы файла на экран будет выведено сообщение `Running`, а по завершению — `Done`.
- `Delete` — удалить выбранный файл (иконка корзины). Если выбрать этот вариант, появится предупреждающее сообщение `Are you sure?` (`Вы уверены?`). Вариант по умолчанию — иконка в форме креста (Нет). В случае, если выбранный файл всё же нужно удалить, то при помощи стрелки следует выбрать иконку с галочкой (Да).
- `Send` — переслать (иконка письма). Для того, чтобы переслать файл другим устройствам, следует вначале настроить канал связи Bluetooth.

Задание 1[3] Установите максимальную громкость динамиков. Проверьте громкость, проиграв произвольный звуковой файл NXT. Установите комфортную для себя громкость.

Задание 1[4] Изучите различные варианты автоотключения NXT. Установите таймер автоотключения на 10 минут.

Задание 1[5] Протестируйте датчики касания, ультразвука, мотор при помощи меню *Try Me* (не забудьте про стандартные порты подключения сенсоров).

Работу с другими пунктами меню будем рассматривать по мере необходимости.

1.2 Среда программирования NXT-G

Запустите программное обеспечение Mindstorms NXT 2.0. В открывшемся окне (рис. 1.4) можно просмотреть видео «Getting Started Guide» (Руководство к быстрому старту) и «Software Overview» (Обзор программного обеспечения).

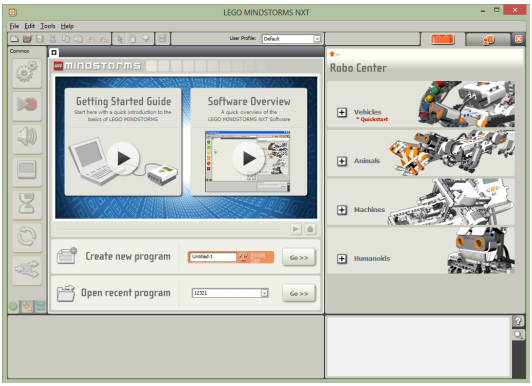


Рис. 1.4. *Окно среды программирования NXT-G после запуска*

1.2.1 Профили

Если при разработке программ на NXT-G один компьютер используется несколькими пользователями, то удобно создать для каждого разработчика свой уникальный профиль, в котором будут храниться программы только этого разработчика. После установки LEGO Mindstorms NXT 2.0 автоматически создаётся один профиль с именем `Default` (по умолчанию) (рис. 1.5).

Рис. 1.5. *Профиль по умолчанию*

Пример 1.3 Создадим новый профиль `Student`. Для этого выберем пункт меню *Edit\ Manage Profiles*. В появившемся окне (рис. 1.6) последовательно нажимаем на кнопку Create (по умолчанию для нового профиля будет предложено имя `Profile-1`), заполняем поле Name и нажимаем кнопку Close.

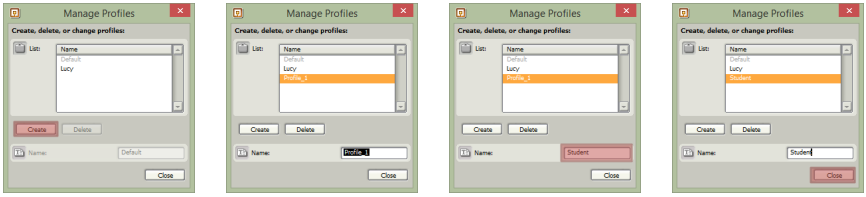


Рис. 1.6. *Создание собственного профиля*

Теперь можно выбрать свой уникальный профиль из списка (рис. 1.7):

Рис. 1.7. *Выбор собственного профиля*

Задание 1[6] Удалите профиль `Student` и создайте профиль со своим именем. Перед дальнейшей работой выберите свой профиль.

1.2.2 Создание и открытие программ

Пользовательский интерфейс среды по умолчанию устанавливает для первой новой программы имя `Untitled-1` (Безымянная-1). В дальнейшем номер каждой новой создаваемой Вами программы автоматически увеличивается на единицу. Однако более целесообразно использовать для своих программ уникальные «говорящие» имена.

В поле Create new program (Создать новую программу) введите `Hello!` и нажмите кнопку Go (Вперёд).

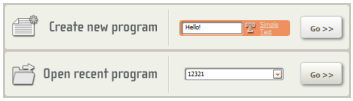


Рис. 1.8. *Создание или открытие программы*

Внимание! Экран NXT отображает текст только на английском языке, поэтому символы кириллицы в имени файла недопустимы.

В дальнейшем открыть уже созданную программу можно, выбрав её из списка в поле Open recent program (Открыть существующую программу) (рис. 1.8).

Для создания и открытия файлов можно использовать и стандартные средства — комбинации клавиш Ctrl+N и Ctrl+O соответственно или воспользоваться меню *File*.

Вкладки в левой верхней части окна среды позволяют открыть Robo Center и My Portal (рис. 1.9). Эти возможности мы оставляем за рамками настоящего пособия и рекомендуем читателю самостоятельно с ними ознакомиться. Просто нажмите на крестик, чтобы увеличить область программирования.

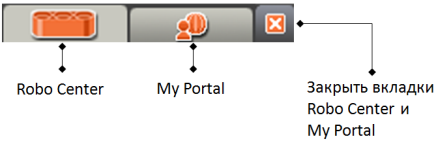


Рис. 1.9. *Доступ к Robo Center и My Portal*

При необходимости эти вкладки всегда можно включить.

1.2.3 Доступ к программным блокам

В NXT-G реализован визуальный способ проектирования программ, что очень удобно для обучения. Программа составляется из блоков. Каждый блок представляет различные типы действий. Блоки можно настраивать. Все блоки организованы и представлены в трёх палитрах программирования (рис. 1.10) — «Common» (Общая), «Complete» (Полная) и «Custom» (Пользовательская). Одновременно можно работать только с одной палитрой.

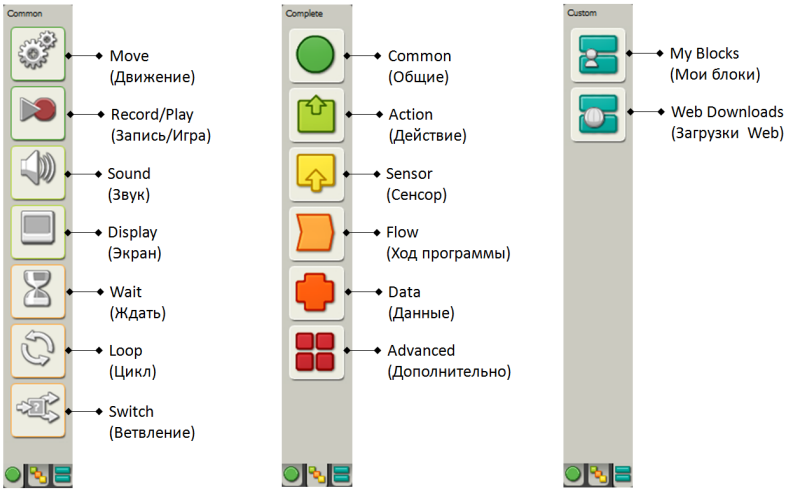


Рис. 1.10. *Палитры блоков NXT-G*

Блоки имеют различную окраску, которая зависит от функционального назначения блока. Блоки, функциональное назначение которых похоже, окрашены одинаково. Например, все блоки, которые отвечают за выполнение некоторого действия (`Action`), окрашены зелёным цветом, а блоки, которые отображают текущие значения датчиков — жёлтым

цветом. Цветовая окраска позволяет легко находить нужные блоки в палитре.

Блоки, которые используются наиболее часто, сгруппированы в палитре «Common». Каждый иконка этой палитры является отдельным программным блоком, кроме иконки `Wait`, которая показывает шесть вариантов этого блока (рис. 1.11):

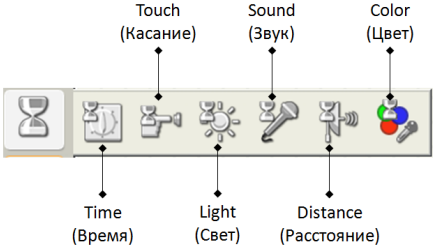


Рис. 1.11. *Выпадающий список вариантов блока Wait*

Любой из вариантов блока `Wait` можно получить путём настроек блока.

Задание 1[7] Изучите всплывающие подсказки блоков палитры «Common», наводя на них указатель мыши.

Полная палитра открывает доступ к 39 блокам, организованным в шесть категорий. Самая первая категория повторяет общую палитру. С остальными категориями мы познакомимся позднее на конкретных примерах.

Пользовательская палитра содержит блоки, которые Вы загрузили и создали самостоятельно. Работу с этой палитрой мы будем изучать в теме 6.

1.3 Первая программа

Продолжим создание программы `Hello!` и тем самым завершим обзор интерфейса среды NXT-G.

Правее палитры программирования находится большая область программирования. Блоки перетаскиваются мышью из палитры на специальную направляющую в виде балки LEGO, которая увеличивается с очередным добавленным блоком.

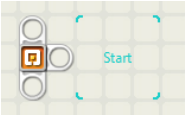


Рис. 1.12. *Исходный узел для построения программы на NXT-G*

Пример 1_4 Создадим классическую для вхождения в язык программирования программу, которая будет выводить на экран сообщение: `Hello, world!`

Добавьте на лист программы блок `Display` из общей палитры. В нижней части окна отображается панель настроек текущего (выделенного) блока. По умолчанию блок настроен на отображение изображений (Action: Image), а именно, улыбающегося смайлика. Смена картинки возможна в свойстве File. Свойство Display позволяет очищать экран (Clear) перед выводом нового изображения.

Нам нужно вывести на экран текст, поэтому изменим свойство Action на Text (рис. 1.13), заменим текст по умолчанию на Hello, world! и выберем позицию отображения Line — 5 (можно просто щёлкнуть мышью в нужном месте свойства Position или задать необходимые координаты).

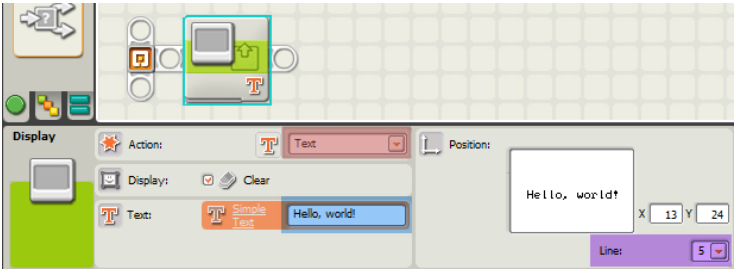


Рис. 1.13. *Настройки блока Display*

Обратите внимание на то, что изменение настроек блока изменяет и внешний вид блока.

Наша программа получилась очень компактной, она полностью умещается на экране. В дальнейшем мы встретимся с более длинным (в прямом смысле этого слова) кодом. Для того, чтобы увидеть код, который не поместился на экране, можно воспользоваться курсорными стрелками. Кроме этого можно щёлкать мышкой по области кода в правом нижнем углу экрана (становится доступной, если щёлкнуть по вкладке с изображением лупы).

Для загрузки и запуска программ на блок NXT в среде NXT-G используются управляющие кнопки или, по-другому, контроллер (рис. 1.14):

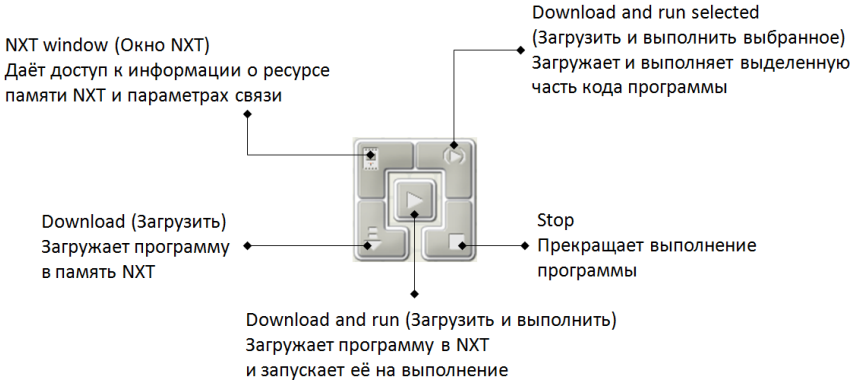


Рис. 1.14. *Контроллер NXT-G*

Подключите робот шнуром USB к компьютеру. Включите робот нажатием на оранжевую кнопку. Запустите программу на выполнение при помощи центральной кнопки контроллера. Что Вы увидели? На экране мелькнула надпись `Hello!` и затем сразу `Done`. Это означает, что программа `Hello!` уже завершила свою работу, а Вы этого так и не увидели. Обратите внимание, что сейчас на экране NXT отображается меню *Software files*.

Для того, чтобы успеть увидеть текст на экране NXT до завершения программы, нужно использовать задержку времени. Добавим в программу новый блок `Wait` и настроим его на отсчёт двух секунд (рис. 1.15):

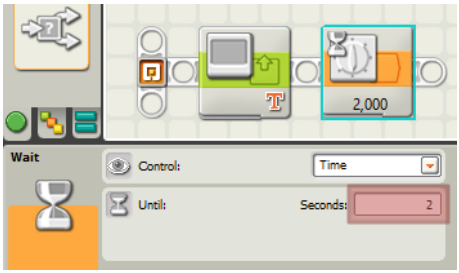


Рис. 1.15. *Настройки блока Wait*

Ещё раз запустите программу на выполнение при помощи центральной кнопки контроллера. Теперь текст остаётся на экране в тече-

ние двух секунд. Поскольку сейчас на экране NXT активна надпись `Run`, то можно повторить выполнение программы нажатием на оранжевую кнопку ввода.

Задание 1^8 Создайте новую программу `Eyes`, в которой на экране NXT будут изображены глаза, смотрящие налево-направо. Используйте файлы `Looking left` и `Looking right` для имитации взгляда налево/направо. Смена направления взгляда должна происходить четыре раза.

Завершим тему присваиванием нового имени Вашему NXT-блоку.

Пример 1_5 Изменение имени NXT. Нажмите на кнопку NXT WINDOW контроллера. В результате откроется окно, правая часть которого NXT Data содержит следующую информацию (рис. 1.16):

- Name (Имя)
- Battery (Заряд батареи)
- Connection (Соединение)
- Free Storage (Свободная память)
- Firmware Version (Версия программного обеспечения, так называемой «прошивки»)

Введите в поле Name новое имя Вашего NXT и нажмите на кнопку ввода рядом с этим полем. На экране NXT автоматически отобразится новое имя.

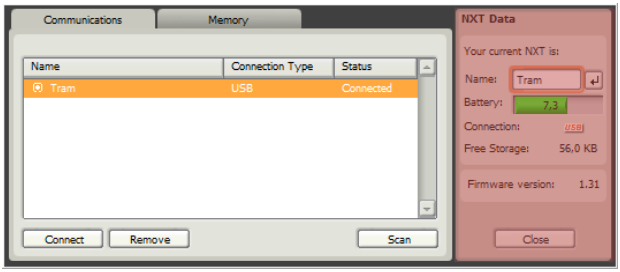


Рис. 1.16. *Окно NXT*

Вкладки Communications и Memory рассмотрим позднее.

Тема 2

Программирование моторов: команда Move

Рассматриваются возможности управления моторами робота LEGO Mindstorms NXT 2.0 при помощи блока `Move`. Изучаются настройки блока.

Цель: научиться управлять роботом при помощи блока `Move`.

Одна из основных функций робота — движение. Двигаться может как весь робот целиком, так и отдельные его части. Движением управляют *сервомоторы* (или *сервоприводы*). В конструкторе LEGO Mindstorms сервомоторы имеют датчики оборотов, подсчитывающие количество градусов поворота главной оси. Важным компонентом сервомотора является *редуктор*, который через систему шестерней превращает чрезвычайно быстрое вращение внутреннего электрического двигателя в более медленное. Наличие датчика оборотов и редуктора позволяют сервомотору совершать прецизионные движения главной оси. Сервопривод LEGO может быть повёрнут с точностью 1°. Внутреннее устройство сервопривода показано на рис. 2.1.

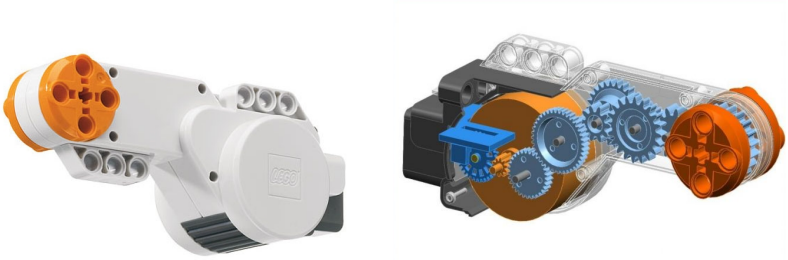


Рис. 2.1. *Сервопривод (слева) и его внутреннее устройство (справа)*

Программирование движения происходит посредством блока `Move` (Движение), который находится в общей и полной палитрах (рис. 2.2).

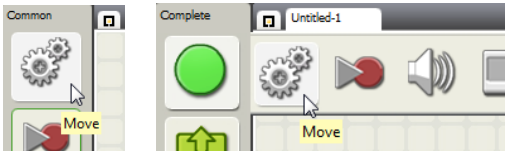


Рис. 2.2. *Блок Move в общей (слева) и полной (справа) палитрах*

Блок `Move` имеет массу настроек, позволяющих управлять поведением мотора (рис. 2.3).

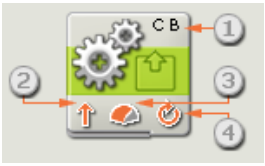


Рис. 2.3. *Настройки блока Move*

На рисунке цифрами отмечены:

1. Моторы, которыми управляет этот блок.
2. Направление вращения моторов.
3. Уровень мощности мотора (скорость). Реальная скорость робота будет зависеть от его конструкции, типа поверхности (скользкая, шершавая и пр.), наклона поверхности, массы робота и т. п.
4. Параметр длительности движения: без ограничения, в градусах, оборотах или секундах.

Настройка блока `Move`, как и других блоков, производится в нижней части экрана после выбора блока (рис. 2.4):

Рис. 2.4. *Панель настроек блока Move*

На рисунке цифрами отмечены:

1. Выбор моторов, которыми нужно управлять (Port). Может быть A, B или C. Один блок `Move` может *управлять сразу двумя мо-*

торами (для остановки — даже тремя); на рисунке как раз выбраны моторы B и C. Заметим, что при создании колёсных или гусеничных роботов лучше левые и правые колёса (гусеницы) назначать моторам B и C, а вспомогательные операции (манипуляторы, поворотные башни и пр.) отводить мотору A. Это связано с устройством блока NXT и позволит реализовывать более точное передвижение робота.

2. Выбор направления вращения мотора/моторов (Direction). Может быть «вперёд», «назад» или «стоп».
3. При управлении двумя моторами служит для поворота: переместите слайдер (бегунок) Steering влево или вправо. Для прямолинейного движения установите его посередине. Если использовать промежуточные значения этого параметра, то робот будет ехать под дуге. Чем дальше слайдер от центра, тем круче поворот.
4. Power задаёт уровень мощности 0–100 %. Мощность не может выходить за этот диапазон.
5. Duration — длительность работы мотора задаётся в:
   - `Rotations` — количествах оборотов двигателя;
   - `Degrees` — градусах, на которые повернётся вал двигателя;
   - `Seconds` — секундах;
   - `Unlimited` — без ограничения. Это очень важный параметр; он означает, что двигатель будет работать *до наступления некоторого события*. Например, мы можем запустить мотор и остановить его, когда робот заедет на красное поле.
6. Next Action — действие моторов после выключения. Здесь возможны два варианта:
   - `Brake` — тормозить. В этом случае при остановке мотор немедленно прекратит своё вращение. На выполнение этой задачи будет потрачена энергия. В частности, при постоянных резких торможениях батарея садится быстрее.
   - `Coast` — после команды на остановку просто отключить питание от двигателя и катиться по инерции. Этот режим выгодно использовать, если не требуется точной остановки.
7. Поле обратной связи. Здесь отображаются текущие значения поворотов моторов при наличии соединения робота с компьютером.

Задание 2[1] Смена настроек блока.

- Запустите среду для программирования роботов LEGO Mindstorms NXT.
- Создайте новый файл. Добавьте в программу блок `Move`.
- Изменяя настройки блока `Move`, понаблюдайте за изменениями на изображении блока в программе.
- Не создавая программу на рис. 2.5 прочитайте её. Какие настройки соответствуют каждому блоку программы?

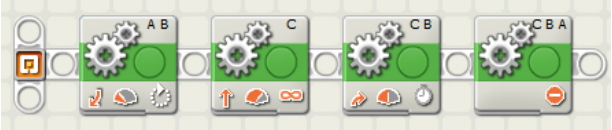


Рис. 2.5. *Пример программы к заданию* 2^1

Для дальнейшей работы нам нужна тестовая модель робота.

Задание 2^2 Сборка тестовой модели.

- Соберите робота Five Minute Bot по инструкции: `www.nxtprograms.com/NXT2/five_minute_bot/index.html`
- Обратите внимание на названия портов, к которым подключены моторы.

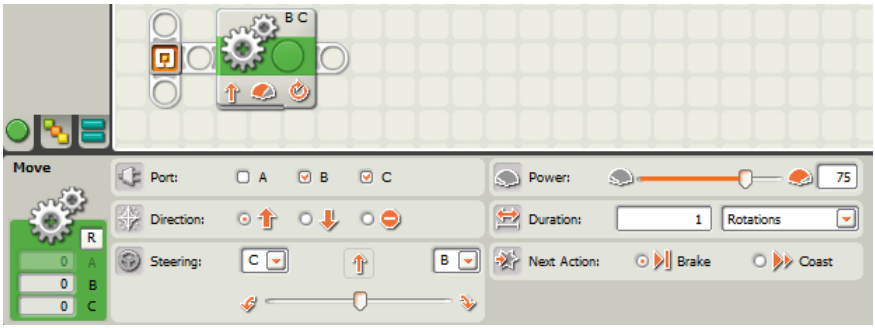


Рис. 2.6. *Пример программы к заданию* 2^3

Задание 2^3 Программа движения робота.

- Создайте программу, показанную на рис. 2.6.
- Внимательно рассмотрите настройки блока `Move`. Что, по-вашему, будет делать робот в соответствии с этой программой?

- Проверьте свои предположения, загрузив программу на робот и запустив её. Выполнил ли робот то, что вы ожидали?
- **Не отключая** USB-кабель вращайте один из моторов рукой и одновременно наблюдайте за показаниями в поле обратной связи. В каких единицах выдаётся информация в этом поле?

Задание 2[4] Движение одним мотором [1].

Измените предыдущую программу так, чтобы робот вращал только один из двух подключенных моторов. Установите длительность вращения в восемь оборотов. Понаблюдайте, как меняется характер движения робота в зависимости от направления движения.

Имейте в виду, что длины USB-кабеля может не хватить на восемь оборотов. Поэтому не забудьте отсоединить его перед запуском программы на выполнение!

Заполните таблицу:

Мотор	Направление	Характер движения
Левый	Вперёд	
Левый	Назад	
Правый	Вперёд	
Правый	Назад	

При выполнении дальнейших заданий Вам понадобится тестовое поле, которое поставляется в комплекте с роботом.

Задание 2[5] Движение двумя моторами [1].

- Изучите доступные параметры продолжительности движения, изменяя их в поле Duration.
- Установите Next Action в Brake и проведите замеры расстояний[1], которые проезжает робот при различных значениях параметра Duration и заполните таблицу:

Кол-во	Тип	Путь (*см*)	Кол-во	Тип	Путь (*см*)
1	Seconds		360	Degrees	
2,5	Seconds		720	Degrees	
1	Rotations			Unlimited	
2	Rotations				

[1] Для замеров пройденного пути используйте линейку на тестовом поле. С одной стороны поля цена деления линейки составляет 1 *см*, а с другой — 1 *дюйм*.

Обратите внимание, что при установке длительности движения в `Unlimited` робот не движется до бесконечности, как мы могли бы ожидать. Напротив, едва начав движение он останавливается и программа завершается. Чтобы разобраться в этом поведении надо понять, как работает блок `Move` в режиме `Unlimited`. При запуске программы первым запускается блок `Move` (он у нас единственный). Если в этом блоке установлена длительность движения, отличная от `Unlimited`, то блок отрабатывает положенное время или количество оборотов и программа завершается. Если установлено движение `Unlimited`, то моторы запускаются *в параллельном режиме*. Это значит, что после включения моторов программа не ждёт, пока выполнится блок `Move`, а переходит к следующему блоку программы; предполагается, что моторы позже будут остановлены «вручную». Итак, запустив моторы, программа переходит *к следующему блоку*. Но его в нашей программе нет, поэтому программа завершается: она достигла своего конца. Вместе с завершением программы автоматически останавливаются все моторы. Этим и объясняется такое необычное, на первый взгляд, поведение робота.

Следует иметь в виду и то, что задание длительности движения в секундах не совсем корректно, т. к. при понижении заряда батареи за то же время мотор совершит меньшее число оборотов.

Задание 2^6 Повороты при помощи Steering.

Установите слайдер Steering в любое положение, отличное от центра, и загрузите программу на робот. Как ведёт себя робот? Изменяя положение слайдера и наблюдая за поведением робота, ответьте на следующие вопросы.

- В какую сторону поворачивает робот?
- От чего зависит крутизна поворота?
- При каком положении слайдера поворот наиболее крутой?
- В каком направлении вращаются ведущие колёса при поворотах разной крутизны?
- Где расположен центр поворота при поворотах разной крутизны?

Итак, вы должны были заметить, что самый быстрый поворот производится двумя моторами при перемещении слайдера Steering до упора. При этом колёса вращаются в противоположных направлениях, центр поворота находится в середине оси, соединяющей колёса.

Когда второй двигатель выключен (слайдер Steering в этот момент недоступен) робот осуществляет плавный медленный поворот од-

ним мотором; центр поворота — неподвижное колесо. Движение робота похоже на работу циркуля.

Промежуточные положения слайдера Steering заставляют робот двигаться по дуге. При этом внутренне колесо, т. е. то, в направлении которого происходит поворот, описывает дугу меньшего радиуса, чем внешнее. Центр поворота совпадает с центром окружностей, которые описывают колёса.

Задание 2^7 Повороты на месте [1].

Напишите программу для поворота робота (не мотора!) на месте на 90°; на 180°; на 270°; на 360°. Назовём способ поворота *быстрым*, если робот вращает два колеса в противоположные стороны (слайдер Steering передвинут до упора); назовём его *плавным*, если поворот осуществляется одним колесом (второе неподвижно).

Для замеров углов поворотов установите робот в центре тестового поля так, чтобы заднее колесо «смотрело» на 180°, а ось ведущих колёс совпадала с красной полосой.

Занесите в таблицу углы поворотов двигателей B и C.

Угол поворота робота	Способ поворота	Угол поворота моторов
90°	Быстрый	
180°	Быстрый	
270°	Быстрый	
360°	Быстрый	
90°	Плавный	
180°	Плавный	
270°	Плавный	
360°	Плавный	

Попробуйте использовать полученные данные для поворота на заданное количество градусов *в другую сторону*. Получилось?

Задание 2^8 Гонки.

Напишите программу для движения робота вокруг коробки от набора LEGO Mindstorms. Устройте соревнование в группе по скоростному объезду коробки: один заезд — три круга.

Тема 3

Состояния и события. Сенсоры

Цель: изучить средства для получения роботом информации из окружающего мира.

Поведение робота — это череда сменяющих друг друга состояний (состояние движения, состояние поиска и т. п.). Смена состояний происходит в зависимости от внешних событий. Робот регистрирует события при помощи различных датчиков. Всё поведение робота можно схематично изобразить в виде диаграммы переходов между состояниями.

Пример 3_1 Диаграмма переходов «Состояние 1 → Событие → Состояние 2» соответствующая движению робота до чёрной линии (рис. 3.1):

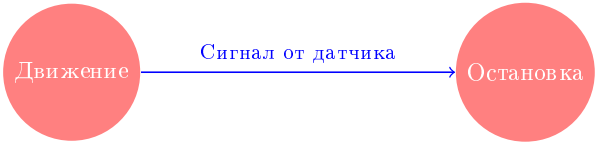


Рис. 3.1. *Диаграмма переходов: движение робота до чёрной линии*

1. Робот находится в состоянии движения.
2. Датчик света регистрирует понижение яркости до некоторого порогового значения — это новое событие, которое можно назвать «робот доехал до чёрной линии».
3. В результате регистрации события робот меняет своё состояние на бездействие (остановка).

Таким образом, можно сказать, что робот должен всё время *ждать возникновения событий*. Для этого в среде NXT-G имеется специальный блок `Wait` (Ждать):

Рис. 3.2. *Варианты блока Wait в общей палитре*

Источниками событий могут являться:

- различные датчики (сенсоры): цвета, освещённости, касания, и пр.;
- таймер;
- кнопки NXT-блока;
- сообщения от других роботов, переданные по Bluetooth соединению.

Далее в этой теме мы рассмотрим работу с некоторыми наиболее часто используемыми датчиками.

3.1 Датчик цвета (Color Sensor)

Датчик цвета (RGB-датчик) совмещает три функции:

1. Позволяет роботу различать цвета.
2. Позволяет роботу различать уровень освещённости путём измерения яркости отражённого света.
3. Цветовая подсветка.

Работа в режиме определения цвета

Пример 3_2 Ожидание события от датчика цвета.

Робот движется по тестовому полю вперёд до тех пор, пока не зарегистрирует синий цвет.

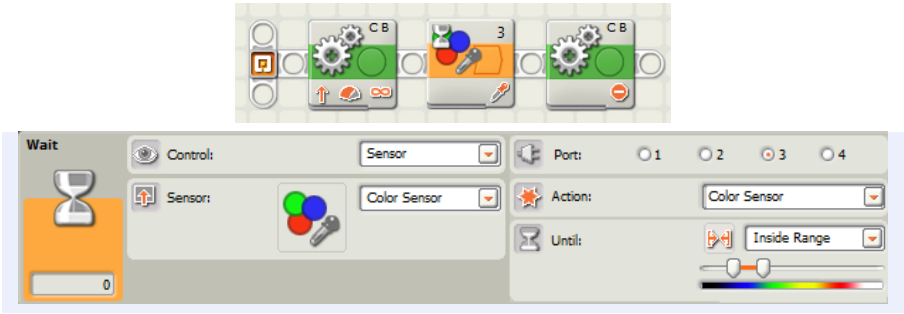


Рис. 3.3. *Пример программы «Движение до синего»*

Ранее в теме (2) мы отметили, что параметр Unlimited (ограничения нет) в настройках блока `Move` позволяет роботу двигаться без каких-либо ограничений до наступления очередного события. После этого можно изменить состояние робота. Изменение состояния **не происходит** автоматически: его надо запрограммировать! В нашем примере после наступления события (регистрация датчиком синего цвета) запрограммирована остановка моторов B и C.

Обратите внимание:

- поле Until надо читать по-русски [анти́л], а переводить следующим образом: «до тех пор пока не станет...»;
- в поле Until возможен один из двух вариантов:
  - Inside Range (Внутри диапазона) — ожидается событие «попадание регистрируемого цвета внутрь указанного диапазона цветов»;
  - Outside Range (Вне диапазона) — ожидается событие «попадание регистрируемого цвета во вне указанного диапазона цветов»;
- цвет в поле Until задаётся при помощи двух слайдеров.

Задание 3[1] Протестируйте датчик цвета:

- Добавьте в конструкцию робота Five Minute Bot датчик цвета, расположенный впереди и направленный вниз, на высоте примерно 1–1,5 *см* от поверхности пола (стола).
- Добавьте на поле программы блок `Wait Color Sensor` и протестируйте его на предметах различных цветов:
  1. Подключите блок NXT к компьютеру через USB-соединение.

2. Выделите блок `Wait` и нажмите на кнопку Download and run selected контроллера NXT (рис. 3.4).
3. **Не отсоединяя** USB-шнур, подносите предметы различной окраски к датчику на расстояние примерно 1 *см*.
4. Следите за результатом в поле обратной связи панели настроек датчика (левый нижний угол панели настроек).
5. Проведите те же тесты при помощи меню *View* NXT. Совпадают ли результаты тестов?

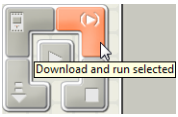


Рис. 3.4. *Компиляция и выполнение выделенного фрагмента*

Задание 3[2] Напишите программу для движения робота по тестовой полосе[1] до обнаружения красного (синего, зелёного и др.) цвета. После того, как заданный цвет обнаружен, робот должен произнести этот цвет.

Задание 3[3] Напишите программу для движения робота по тестовой полосе до обнаружения синего или зелёного цвета. Проверьте работу программы, запуская робот с разных сторон тестовой полосы.

Работа в режиме измерения освещённости

Датчик цвета из нашего набора может работать в двух режимах: собственно датчик цвета, и датчик *света*, измеряющий освещённость. Переключите в настройках блока `Wait Color Sensor` датчик цвета в режим датчика освещённости. Для этого в поле Action выберите Light Sensor (рис. 3.5). Обратите внимание, что внешний вид блока `Wait` изменился.

[1] Тестовая полоса — это таблица разных цветов, идущая по краю тестового поля. Содержит чёрный, синий, зелёный, красный, жёлтый цвета и их градации.

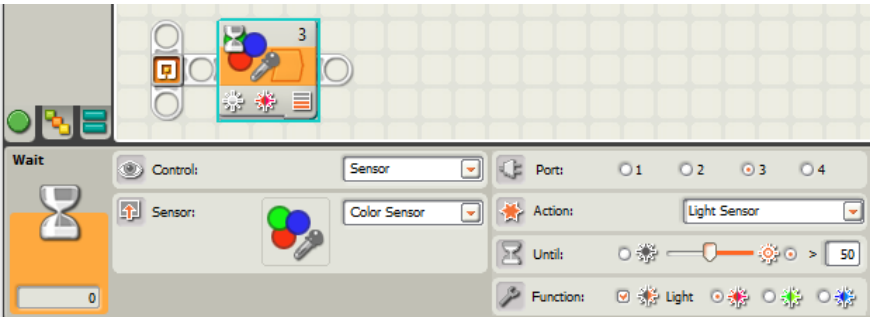


Рис. 3.5. *Настройки блока Wait Color Sensor в режиме измерения освещённости*

Чтобы лучше представить себе, что «видит» датчик освещённости, вообразите, что он перемещается над чёрно-белой фотографией. Результатом измерения будет *яркость* в диапазоне от 0 (чёрный цвет) до 100 (белый цвет), хотя крайних значений на практике зафиксировать не удаётся.

Обратите внимание:

- уровень освещённости в поле Until задаётся при помощи слайдера или вводом числового значения;
- знак в ожидаемом событии переключается щелчком по яркому или тёмному «солнышку» (по умолчанию стоит знак «>»);
- поле Until читается: «до тех пор пока освещённость не станет...». Так, на рис. 3.5 ожидается событие «уровень освещённости > 50», что означает выполнение действия, предшествующего блоку `Wait` до тех пор, пока освещённость не станет > 50;
- поле Function позволяет использовать датчик цвета в качестве лампы красного, зелёного или синего света (включённый флажок Light). Измеряя интенсивность отражённого от поверхности света можно оценить её яркость: от светлой поверхности отразится больше света, чем от тёмной. Вообще говоря, если замеры освещённости происходят при хорошем дневном свете, то подсветку можно выключить (снять флажок Light). Однако производитель рекомендует в режиме измерения освещённости использовать красную подсветку. Также при выключенном свете можно измерять общую освещённость в окружающем пространстве.

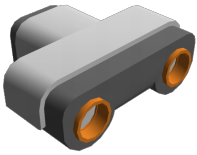
Задание 3^4 Протестируйте работу датчика цвета в режиме измерения освещённости на предметах различных цветов так же, как делали

в предыдущем задании:

- Выпишите значения освещённости, которые датчик показывает для чёрного, синего, голубого, жёлтого, зелёного, красного и чёрного цветов на тестовом поле.
- Используйте в качестве подсветки лампы разных цветов. Как изменяются показания датчика?
- Выключите подсветку и сравните показания датчика с подсветкой и без неё.
- Поэкспериментируйте с окружающими предметами.

Задание 3^5 Используя RGB-датчик **в режиме измерения освещённости**, напишите программу для движения робота по белому полю до обнаружения чёрного цвета. Для этого в поле Until укажите необходимый порог чёрного цвета. После остановки робот должен вывести на экран соответствующую информацию (например, «Black color found!»).

3.2 Датчик ультразвука (Ultrasonic Sensor)



Ультразвуковой сенсор заменяет роботу зрение. Он работает по тому же принципу, что и локатор летучей мыши: измеряет расстояние путём расчёта времени, которое потребовалось звуковой волне для возвращения после отражения от объекта. Как следует из его настроек, датчик способен измерять расстояние от 0 до 255 *см* с точностью $+/-3$ *см*. Однако на практике минимальное расстояние, на котором ультразвуковой датчик выдаёт осмысленные показания составляет около 5–7 *см*. Кроме того, следует иметь в виду, что по сравнению с датчиками цвета, света, касания и др., это медленный датчик. Во-первых, это объясняется малой скоростью звука по сравнению со скоростью света, а во-вторых, медленным протоколом обмена данными, используемым для этого датчика[2].

[2]Более подробно прочитать о скорости измерения ультразвукового датчика можно в статье по адресу `http://nnxt.blogspot.ru/2010/10/nxt-g-ultrasonic.html`

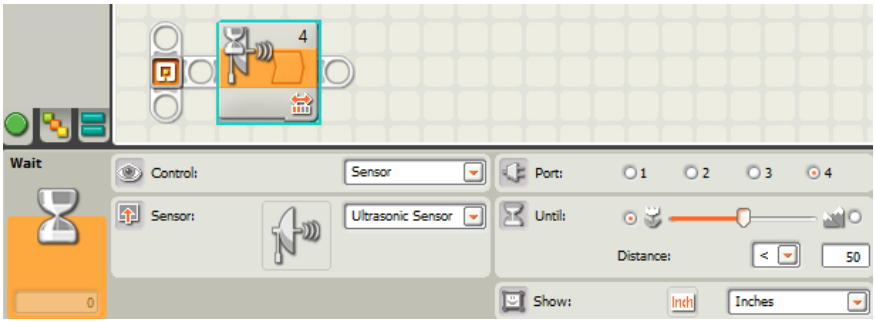


Рис. 3.6. *Настройки блока Wait Ultrasonic Sensor*

Добавьте в конструкцию робота датчик ультразвука, направленный вперёд по ходу движения. Настройки датчика показаны на рис. 3.6.

Обратите внимание:

- в поле Until также как и для Color Sensor задаётся условие выхода из блока;
- знак с «<» на «>» или наоборот в ожидаемом событии переключается щелчком по радиокнопке Farther Than (Больше, чем) — изображение горы или Nearer Than (Меньше, чем) — изображение цветка (установлено по умолчанию). Также можно воспользоваться списком;
- поле Show задаёт единицы измерения расстояния. По умолчанию расстояние измеряется в дюймах (inches). Не забудьте переключить этот параметр в сантиметры;

Задание 3^6 Протестируйте датчик ультразвука, поднося к нему предметы на различном расстоянии и наблюдая результаты:

- при помощи меню *View*;
- в поле обратной связи блока `Wait`.

Задание 3^7 Напишите программу, по которой робот движется в направлении препятствия (стены) и останавливается на расстоянии 30 *см* от него.

3.3 Датчик касания (Touch Sensor)

Добавьте в конструкцию робота Five Minute Bot датчик касания — кнопку. Его можно не закреплять, а подключить длинным кабелем и использовать как пульт дистанционного управления.

В настройках блока `Wait Touch` в качестве Action доступны варианты:

- Pressed — датчик нажат;
- Released — датчик отпущен;
- Bumped — выполнен щелчок (то есть кнопка нажата и сразу отпущена).

Задание 3^8 Напишите игру «Кто точнее», смысл которой состоит в том, чтобы остановить робот точно на заданной линии. После запуска программы робот должен начать движение по направлению к чёрной линии. Как только колёса робота коснутся линии, игрок должен нажать кнопку; при этом робот должен остановиться. Выигрывает тот, у кого расстояние от передних колёс до линии меньше. Настройки блока `Move`: неограниченное движение, мощность 80 %. Подберите наилучший вариант для решения задачи.

Задание 3^9 Проведите соревнования по правилам игры «Кто точнее».

Тема 4

Циклы

Цель: изучить (1) способы организации повторяющихся действий в языке NXT-G; (2) способы передачи данных между блоками.

При написании программ часто возникает необходимость в циклическом повторении группы команд. Для этого используется языковая конструкция *цикл*, которой в NXT-G соответствует блок `Loop`.

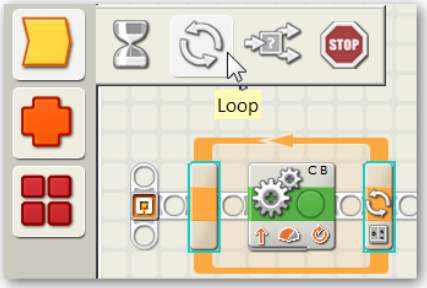


Рис. 4.1. *Блок Loop в меню Flow полной палитры и пример его использования*

Внутрь цикла помещаются блоки, которые нужно выполнить многократно. Заметим, что следует с осторожностью (точнее, с пониманием) размещать в цикле блоки, которые сами в некотором роде обладают цикличностью, например `Move`, `Sound` и некоторые другие.

Программируя цикл, мы должны заранее знать в каком случае он прекратит свою работу. Завершение цикла (выход из цикла) может происходить по разным причинам. В NXT-G возможны следующие способы выхода из цикла, доступные в поле Control:

1. Forever — бесконечно (по умолчанию);
2. Sensor — по срабатыванию датчика;

3. Time — через определённое время;
4. Count — после выполнения цикла заданного количества раз;
5. Logic — при выполнении некоторого логического условия, заданного программистом.

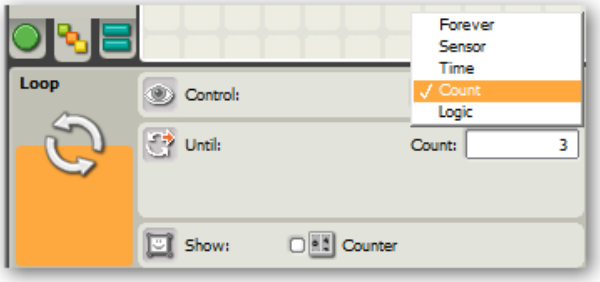


Рис. 4.2. *Выбор варианта выхода из цикла*

В процессе изучения языка NXT-G мы постепенно освоим все эти способы. Начнём с бесконечного цикла.

4.1 Простейшие виды циклов

Задание 4^1 Подготовьте робот с датчиком цвета в передней части; датчик должен быть направлен вниз и находиться на расстоянии около 1 *см* от стола. Расположите робот перед чёрным полем (или чёрной линией) на расстоянии от неё около 20 *см*.

Задание 4^2 *Движение до линии.* Используя датчик цвета в режиме измерения освещённости, напишите программу, позволяющую роботу доехать до чёрной полосы и остановиться (см. задание 3^3).

Задание 4^3 *Движение до линии и обратно.* Измените программу из задания 4^2 так, чтобы после достижения чёрной линии робот отъезжал на белое поле.

Задание 4^4 *Бесконечное циклическое движение до линии и обратно.* Измените программу из задания 4^3 так, чтобы робот выполнял

движение до линии и обратно бесконечное количество раз. Для этого заключите предыдущую программу в бесконечный (forever) цикл.

Задание 4^5 *Циклическое движение до линии и обратно по счётчику.* Измените программу из задания 4^4 так, чтобы робот выполнял движение до линии и обратно три раза. Для этого измените настройки цикла следующим образом:

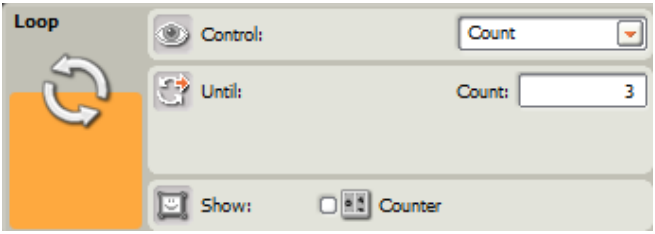


Рис. 4.3. *Настройки цикла со счётчиком*

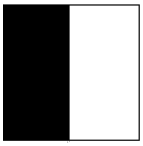
4.2 Движение робота по линии

Пример 4_1 Перейдём к классической задаче о движении робота вдоль линии. Полигон с такой линией имеется в каждом наборе.

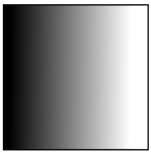
Разберёмся, может ли робот двигаться *по* линии, то есть так, чтобы сенсор всегда находился на чёрном поле. Для этого представим себе, что через некоторое время после старта сенсор «уехал» с чёрного поля и оказался на белом. Мы понимаем, что в этом случае робот должен немного *подрулить* чтобы вернуться обратно на чёрную линию. Но куда именно подрулить: налево или направо? Дело в том, что белый цвет по обе стороны от чёрной линии *одинаков*. Поэтому, попав в такую ситуацию, робот не сможет принять однозначное правильное решение.

Выходом из такой ситуации будет небольшое изменение постановки задачи: робот должен двигаться не *по* линии, а *по границе* чёрного и белого.

Что при этом изменится? Робот и человек по-разному воспринимают границу линии.



Так видит границу человек



А так видит робот

Мы видим, что датчик цвета при своём движении влево или вправо не наблюдает резкой границы чёрного и белого. Это происходит из-за того, что при его перемещении, например, слева направо, количество света, попадающего в сенсор, постепенно увеличивается. Значит показания датчика также будут плавно возрастать.

Нас больше всего устраивает среднее положение датчика, то есть когда он находится прямо над границей. В этом случае датчик наблюдает не чёрный и не белый цвета, а некоторый «средний» — серый. Для определения показания датчика для серого цвета нужно найти среднее арифметическое между белым и чёрным.

Рис. 4.4. *«Чёрное» плюс «белое» пополам равно «серое»*

При движении робота уход влево (в чёрную область) приведёт к уменьшению показаний датчика, а уход вправо (в белую область) — к увеличению. Таким образом мы всегда сможем отличить левое от правого.

Получим примерно такую программу для движения вдоль линии:

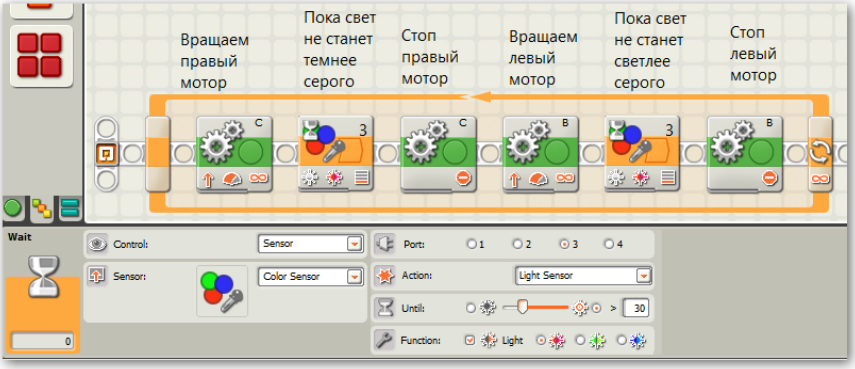


Рис. 4.5. *Движение робота вдоль линии*

Обратите внимание на следующее: перед стартом робот ставится на поле так, чтобы датчик цвета «смотрел» на границу линии: чёрное — слева, белое — справа.

Задание 4^6 Найдите отличия между предложенным решением и программой из задания 4^4.

Задание 4^7 Запрограммируйте робот на движение вдоль чёрной линии, изменив код программы 4^4 и проверьте её работоспособность.

Задание 4^8 Требуется запустить робот по той же линии, но в обратную сторону. Как это сделать *не изменяя программу*?

4.3 Цикл со счётчиком. Передача данных между блоками

Разберём следующий пример.

Пример 4_2 Написать программу движения робота с ускорением.

Для начала напишем программу движения робота с постоянной скоростью:

- Повтори 20 раз:
- Вперёд

В настройках блока Move отметьте Coast (без торможения в конце выполнения блока). В поле Duration оставим значение 1 Rotation.

Проверьте работу программы. Робот движется прямо с постоянной скоростью.

Для создания ускорения будем использовать *передачу данных* из цикла на блок движения. Очевидно, в нашем случае ускорение предполагает постепенное увеличение скорости робота, то есть *увеличение мощности мотора.*

Изменим программу.

В настройках блока Loop включите флажок Show Counter (Показать Счётчик). Обратите внимание на то, как изменился внешний вид цикла: появился разъём Loop Count

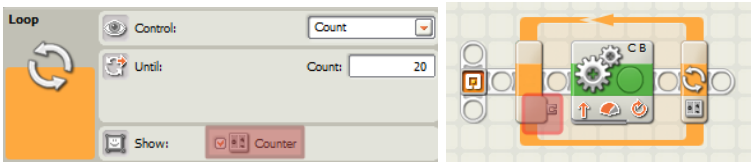


Рис. 4.6. *Включение отображения счётчика цикла (слева) и доступ к нему (справа)*

Откройте концентратор данных блока Move, щёлкнув на кнопке в левой нижней части блока:

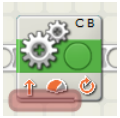


Рис. 4.7. *Концентратор данных в свёрнутом состоянии*

В результате откроются входные (расположенные на левой стороне) и выходные (расположенные на правой стороне) разъёмы, к которым можно подключать *шины данных* от разъёмов других блоков[1] (рис. 4.8). Таким способом можно передавать данные внутри программы на NXT-G *без фактического использования переменных.*

[1] Щёлкнув по тому же месту блока можно закрыть выдвижную панель.

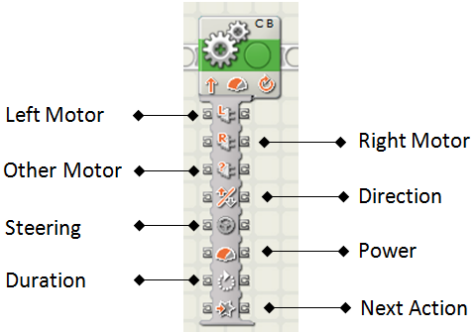


Рис. 4.8. *Концентратор данных в раскрытом состоянии*

Соедините шиной разъём счётчика цикла и разъём Power (Мощность) блока Move (при наведении указателя мыши на разъём появляется соответствующая всплывающая подсказка). Чтобы сделать соединение, щёлкните левой кнопкой мыши на начало, проведите мышь в конец пути и щёлкните ещё раз; можно делать дополнительные щелчки по ходу траектории в местах её желаемых сгибов. В поле Duration укажите Unlimited (без ограничения длительности). Сверните концентратор данных — щёлкните по выпадающей панели блока так, как будто хотите её свернуть. Все разъёмы, к которым не подходят шины, исчезнут. Этот трюк может заметно уменьшить площадь, занимаемую программой.

В результате программа примет вид:

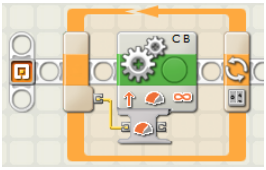


Рис. 4.9. *Движение робота с ускорением*

Проверьте её работу.

Для того чтобы ускорение было более наглядным, добавьте в цикл небольшую задержку:

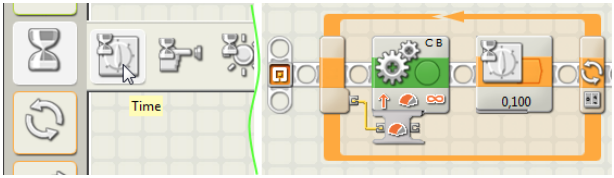


Рис. 4.10. *Движение робота с ускорением с задержкой времени*

выполнения для большей наглядности

Пример 4₃ Вывод информации на экран блока NXT.

Проверим, что на самом деле поступает из цикла на вход блока `Move`. Для этого разработаем упрощённую программу, в которой значение счётчика цикла выводится на экран блока NXT. Напомним, что для вывода используется блок `Display` из панели Common.

Мы собираемся выводить на экран числовую информацию, поэтому её нужно вначале преобразовать к текстовому типу. Делается это при помощи блока `Number to Text` панели `Advanced`.

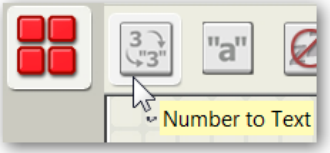


Рис. 4.11. *Доступ к блоку Number to Text панели Advanced*

Получаем следующую схему программы:

- в настройках блока `Loop` значение счётчика устанавливаем равное 10;
- значение счётчика цикла передаём на преобразователь «Число → Текст»;
- полученный текст подаём на вход блока `Display`;
- делаем задержку, чтобы успеть рассмотреть значение на экране;
- после цикла ставим задержку до нажатия на кнопку Start, чтобы рассмотреть последнее выведенное на экран значение.

Пример программы приведен на рис. 4.12.

Обратите внимание на то, что *шины разных данных окрашены по-разному*. Всего в NXT-G используются три типа, каждому из них соответствует своя окраска:

- числовой тип данных — жёлтый цвет;
- текстовый тип данных — красный цвет;
- логический тип данных — зелёный цвет.

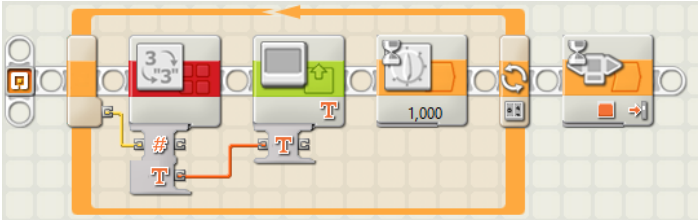


Рис. 4.12. *Вывод значений счётчика на экран*

На рис. 4.13 приведены настройки последнего блока:

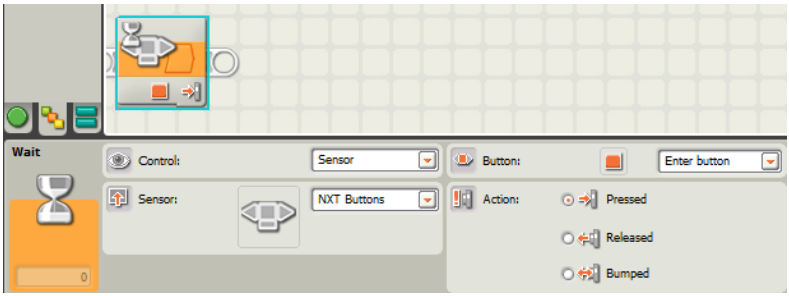


Рис. 4.13. *Настройки блока Wait*

4.4 Цикл с выходом по значению сенсора

В NXT-G есть вариант использования цикла, в котором выход происходит в соответствии с определённым значением какого-либо сенсора. Настройка такого цикла полностью аналогична настройке блока `Wait`, с той лишь разницей, что в параметрах цикла присутствует флажок Show Counter. Пустой цикл с выходом по значению сенсора аналогичен соответствующему блоку `Wait`. Но в теле цикла мы можем разместить блоки, которые должны повторяться многократно до срабатывания сенсора.

Пример 4_4 Превратить программу на рис. 4.12 в простейший секундомер. Для этого тоже нужно отсчитывать время, но остановка должна произойти после нажатия на кнопку. Решение см. на рис. 4.14.

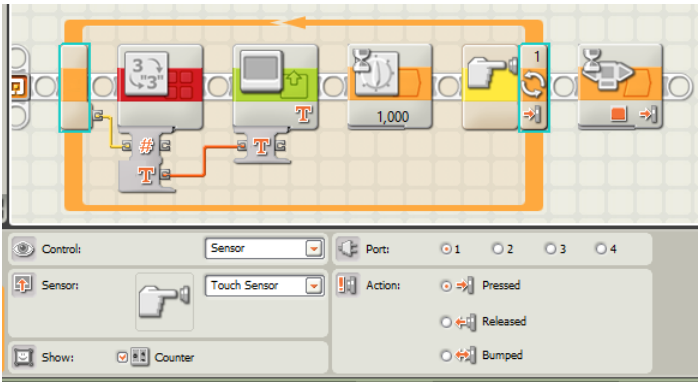


Рис. 4.14. *Простейший секундомер с использованием цикла с выходом по значению сенсора*

Задание 4[9] Напишите программу, по которой робот, используя ультразвуковой датчик, должен доехать до стены и остановиться на расстоянии 30 *см* от неё. Текущее расстояние до стены должно выводиться на экран во время движения.

4.5 Цикл с выходом по условию

Иногда удобно использовать цикл, выход из которого происходит при выполнении некоторого логического условия. Это аналог циклов While и Repeat–Until в языке программирования Паскаль. Настройка цикла производится обычным образом:

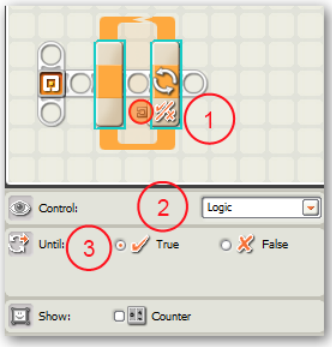


Рис. 4.15. *Настройки цикла с выходом по логическому условию*

Здесь разъём № 1 (Loop Condition) появляется, если в поле № 2 (Control) выбрать Logic. На этот порт цикла нужно подать провод с логическим значением (зелёный).

Поле № 3 (Until) позволяет определить условие выхода из цикла: либо когда условие, поданное на разъём, станет истиной (True), либо когда оно станет ложью (False).

Пример 4_5 Моделирование ожидания в n секунд при помощи таймера на одну секунду.

Этот не вполне содержательный пример приведён здесь исключительно для знакомства с циклом рассматриваемого вида.

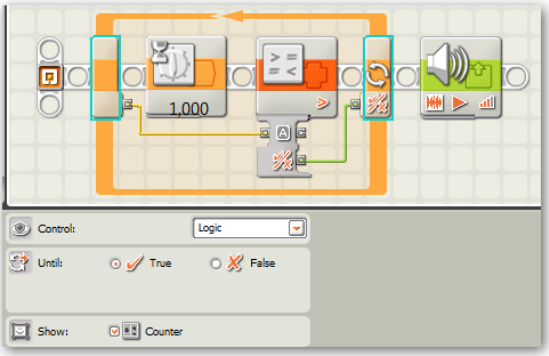


Рис. 4.16. *Моделирование ожидания в n секунд при помощи таймера на одну секунду*

Здесь в цикле, который настроен на выход по условию, выполняются следующие действия:

1. отсчитывается одна секунда при помощи соответствующего блока;
2. номер итерации цикла, поступающий из порта Counter, подаётся на блок сравнения `Compare` из меню `Data` (Данные) полной палитры (рис. 4.17);
3. там он сравнивается с числом n, и результат этого сравнения по зелёному проводу передаётся на второй разъём цикла;
4. если выполняется заданное в блоке сравнения условие, то цикл прекращает работу и мы слышим голос из динамика.

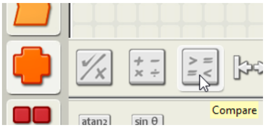


Рис. 4.17. *Блок Compare в полной палитре*

В блоке `Compare` доступны следующие логические операции:

- Less than — «меньше, чем»;
- Greater than — «больше, чем»;
- Equal — «равно».

Данные в поля блока `Compare` могут поступать как по внешнем разъёмам, так и путём ввода в соответствующие поля. Так, в примере 4_5 данные в поле A поступают с разъёма счётчика цикла (в этом случае ввод в поле не доступен), а в поле B вводятся пользователем.

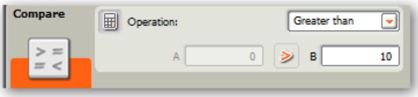


Рис. 4.18. *Настройки блока Compare в примере* 4_5 *(здесь* $n = 10$*):*

Задание 4^{10} Изучите код программы из примера 4_5. Составьте программу и проверьте её работу для разных значений n.

Задание 4^{11} Напишите программу, которая после запуска ждёт нажатия на оранжевую кнопку NXT, после чего что-нибудь говорит и завершается. Для обработки нажатия на оранжевую кнопку воспользуйтесь программным блоком `NXT Buttons` из меню `Sensors` и циклом с выходом по условию. *При выполнении этого задания использовать блок Wait запрещается!* Напомним, что жёлтые блоки группы `Sensors` позволяют узнать текущие показания датчиков.

Задание 4^{12} Исправьте предыдущую программу так, чтобы её завершение происходило после n-го нажатия на оранжевую кнопку. Подсказка: используйте вложенные циклы.

Тема 5

Ветвление в NXT-G

Цель: изучить способы организации ветвлений в языке NXT-G.

Кроме последовательного и циклического исполнения команд язык NXT-G предусматривает *ветвление*. Его можно осуществить при помощи оператора `Switch`, который проверяет логическое условие и пускает программу по одной из двух (или более) ветвей.

В NXT-G оператор ветвления используется реже, чем в обычных языках программирования, из-за наличия удобного оператора `Wait`. Однако в ряде случаев его применение необходимо.

Пример 5_1 Робот должен начать движение после нажатия на оранжевую кнопку Start. Если датчик цвета оказывается над красным объектом, то робот должен сказать «Red» и остановиться, а программа — завершиться. Иначе робот должен перемещаться и говорить: «Detect» (рис. 5.1).

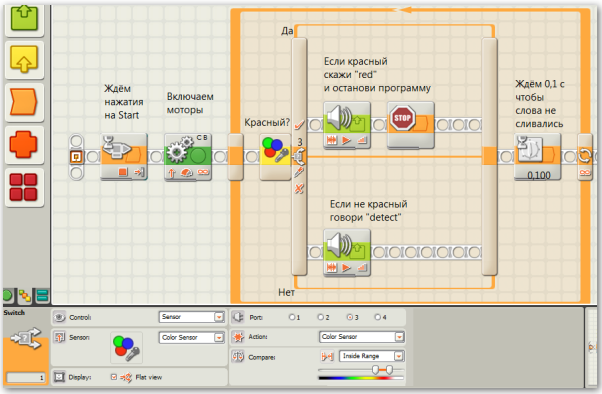


Рис. 5.1. *Программа к примеру* 5_1

Самый первый блок `Wait` в этой программе позволяет запустить робот кнопкой START на блоке.

Робот может говорить при помощи блока `Sound`. Его настройки приведены на рис. 5.2. Блок `Sound` во второй ветке настраивается аналогично.

Рис. 5.2. *Настройки блока Sound из примера* 5_1

В настройке блока `Sound` обратите внимание на флажок Wait for completion. Если он установлен, то вначале проигрывается заданный звук, и *только потом* управление переходит к следующему блоку. Если же флажок снять, то включается звук, после чего управление *немедленно* переходит к следующему блоку.

В конце бесконечного цикла стоит задержка на 0,1 с.

Итак, в поле Control блока `Switch` может быть один из двух вариантов:

- Sensor — поведение робота зависит от логического условия, связанного с текущими значениями сенсора;
- Value — поведение робота зависит от результата сравнения числового, текстового или логического значения, поступившего извне, с заданным значением.

В поле Display флажок Flat view позволяет включить полное отображение блока `Switch` (с двумя ветвями) или отключить его. В последнем случае обе ветви отображаются в виде вкладок, причём в данный момент видна только одна из них (рис. 5.3). Вкладка, помеченная «птичкой», соответствует ветке, по которой пойдёт программа в случае, если проверяемое условие истинно, а вкладка с крестом — ложному значению проверяемого условия.

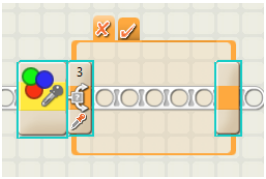


Рис. 5.3. *Компактное отображение блока Switch*

Пример 5_2 Робот говорит «No», если датчик касания не нажат, и «Yes» — если нажат. Обратите внимание на настройки блока `Switch`: в поле Condition можно поменять сравниваемое значение. В блоке `Touch Sensor` в поле Action выбрано Pressed.

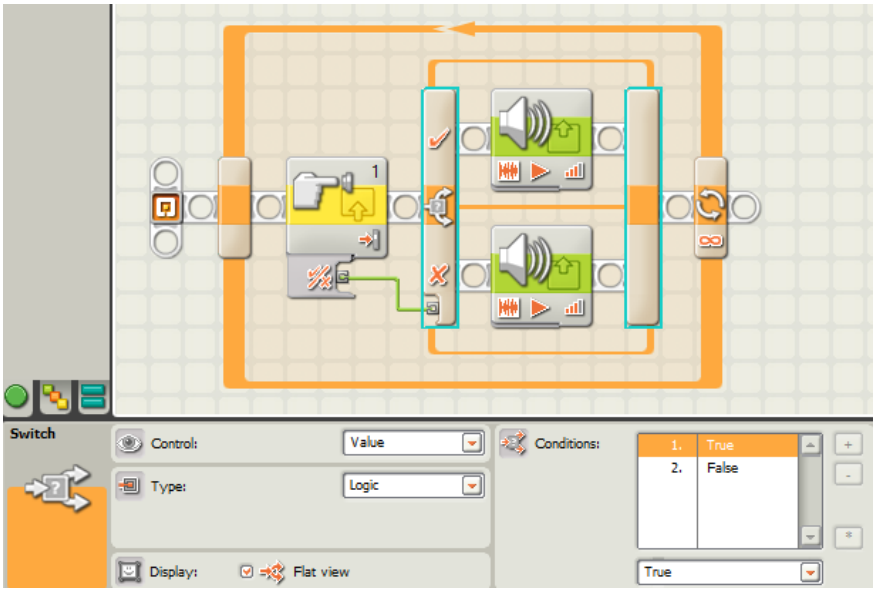


Рис. 5.4. *Пример использования логического типа данных в блоке Switch*

Пример 5_3 На экране NXT последовательно сменяются пять изображений часов в зависимости от счётчика цикла (рис. 5.5). Для изображения часов использованы файлы `Time00-Time04`. Дополнительные вкладки значений блока `Switch` можно добавить нажатием на кнопку «+» в поле Condition. Эта возможность доступна только при выключенном флажке Flat view. Вкладки, соответствующие различным значениям, имеют подписи (появляются при наведении на них указателя мыши). Если обрабатываемых значений больше пяти, то для того, чтобы увидеть все добавленные вкладки, нужно «растянуть» блок `Switch`. Для этого нужно навести указатель мыши на балку LEGO внутри блока, прижать левую кнопку мыши и потянуть вправо.

Подобное использование блока Switch с множественными вкладками соответствует оператору выбора Case (Switch) в текстовых языках программирования.

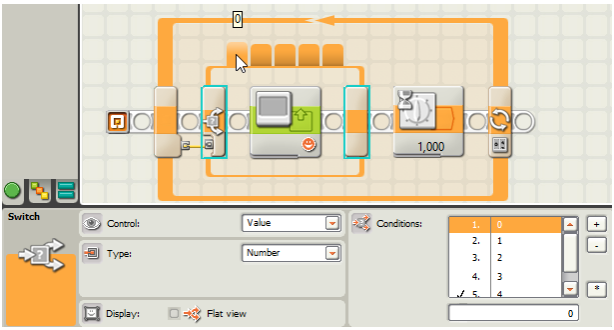


Рис. 5.5. *Пример использования блока Switch со множественным ветвлением*

При обработке сложных логических условий может возникнуть необходимость в использовании *вложенных ветвлений*.

Пример 5_4 Доработаем пример 5_1 так, чтобы робот останавливался ещё и при обнаружении зелёного цвета.

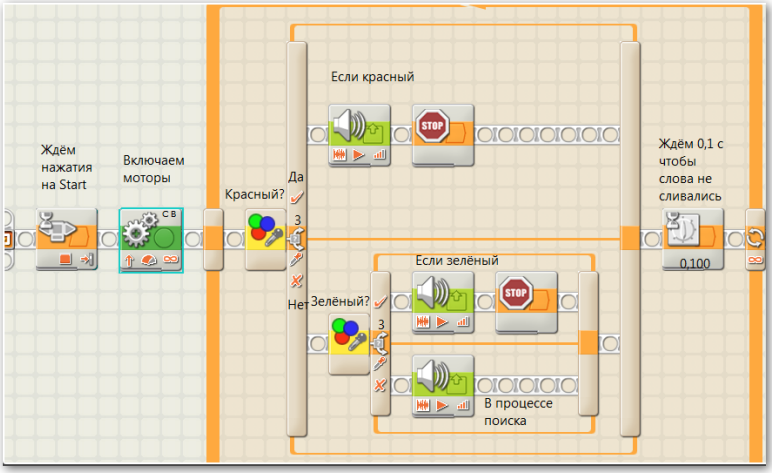


Рис. 5.6. *Программа к примеру* 5_4

Задание 5^1 Напишите программу управления роботом одной кнопкой (датчиком касания). При запуске программы машина должна ожидать нажатия на кнопку Start, после чего начать движение вперёд. При нажатии на датчик касания она должна двигаться назад, при

отпускании — опять двигаться вперёд.

Задание 5^2 Напишите программу управления роботом двумя кнопками (два датчика касания, служащие пультом управления). Машина должна ехать вперёд при нажатии на обе кнопки. Если нажата только правая кнопка, то машина поворачивает направо; если только левая — налево. Если кнопки отпущены, то машина останавливается.

Задание 5^3 Используя блок `Switch` в качестве оператора выбора, напишите программу для исследовательского робота, который перемещается по разноцветному полю и произносит название цвета, над которым проезжает (только основные цвета, для которых в роботе имеется соответствующий звуковой файл). Робот должен остановиться, когда доберётся до красного цвета. Подсказка: вспомните, что все основные цвета закодированы числами 0–5.

Задание 5^4 Напишите программу, которая позволяет роботу оставаться на заданном расстоянии от подвижной стены. Робот оснащён двумя моторами и датчиком ультразвука, направленным вперёд. Перед роботом размещается подвижная стена (журнал, книга, свёрнутый полигон). Робот должен, постоянно двигаясь, сохранять расстояние до стены ≈ 30 *см*. Подсказка: вспомните задание 4^4 и пример 4_1.

Тема 6

Создание собственных блоков

6.1 Подпрограммы: My block

Цель: изучить возможности языка NXT-G по созданию собственных блоков (подпрограмм).

Язык NXT-G позволяет программисту создавать собственные блоки, наподобие блока `Move`. При этом видимый программный код значительно сокращается, что в случае NXT-G очень важно, так как на экране умещается только небольшая часть кода и часто приходится прибегать к прокрутке.

Собственные блоки можно использовать в программе многократно, а также создавать с их участием новые блоки.

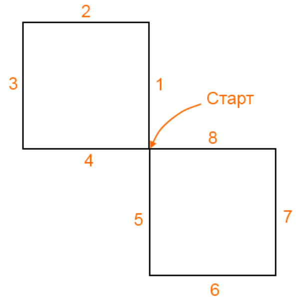
В этом разделе мы рассмотрим простейший вариант блоков: без входных и выходных параметров. В дальнейшем мы изучим более сложные случаи.

Задание 6^1 Записать в тетрадь на сколько градусов нужно повернуть колёса робота, чтобы сам робот повернулся на 90°.

Задание 6^2 Написать программу движения робота по квадрату по следующей схеме:

- Повтори 4 раза:
  - Вперёд;
  - Поворот робота на 90° (на сколько градусов нужно повернуть колёса Вы определили в задании 6^1).

Задание 6^3 Написать программу «Восьмёрка», в которой робот двигается по следующей траектории [4]:



Начало движения — в центре фигуры; цифрами обозначен порядок прохождения сторон. Схема программы:

- Повтори 2 раза
  - Квадрат
  - Поворот на $N°$.

  Подсказка: N найти через пропорцию; см. задание 6_1.

Пример 6_1 Написать ту же самую программу, используя подпрограмму (`My block`). Для этого:

1. выделить мышкой цикл движения робота по квадрату;
2. выбрать пункт меню *Edit* \ *Make A New My Block* (Редактирование \ Создать новый блок);
3. в окне *My Block Builder* (Конструктор блоков) в поле Block Name ввести имя блока — `Square` (только английскими буквами) и описание блока в поле Block Description (можно по-русски), например «Процедура движения по квадрату»;
4. нажать кнопку Next (Далее);
5. на этом шаге нужно создать значок для Вашего блока путём перетаскивания готовых иконок в верхнюю область окна; можно использовать несколько иконок;
6. нажать кнопку Finish (Конец);
7. в результате на листе программы вместо цикла Вы увидите Ваш блок под названием `Square`. Если выполнить на нём двойной щелчок, то на соседней вкладке (на новом листе) откроется содер-

жимое блока, которое при необходимости можно отредактировать. Выбирать свои блоки можно из палитры Custom palette (Моя палитра);

Из перечисленных шагов создания подпрограммы обязательными являются шаги 1, 2, 6. Однако мы советуем выполнять всю последовательность и особое внимание уделить описанию подпрограммы. Некоторое время спустя это поможет вспомнить, что же именно делает эта подпрограмма и разобраться с назначением её параметров. Кроме того, ввод описания блока является правилом хорошего тона. В нашем курсе мы считаем это действие необходимым.

Тогда программа движения по восьмёрке может выглядеть, например, так

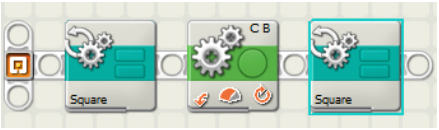


Рис. 6.1. *Движение робота по квадрату с использованием подпрограммы*

или так

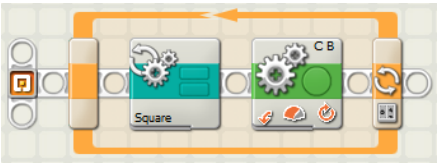


Рис. 6.2. *Движение робота по квадрату с использованием подпрограммы и цикла*

Задание 6^4 Сравните работу двух вариантов программ.

Обратите внимание, доступ к конструктору блоков можно получить и через панель инструментов (рис. 6.3):

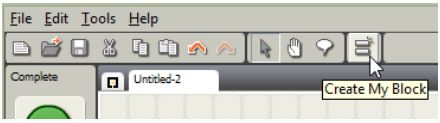


Рис. 6.3. *Кнопка Create My Block на панели инструментов*

Задание 6^5 Написать программу движения робота по правильному треугольнику.

Подсказка: Для вычисления угла поворота используйте пропорцию.

Задание 6^6 (Из [4]). Написать программу для движения робота, который, используя программу из задания 6^5, при своём движении «рисует» мозаику (рис. 6.4). Для того, чтобы сделать движение робота более наглядным, закрепите на робот маркер острием вниз и подложите на стол бумагу.

Рис. 6.4. *Пример траектории движения робота к заданию* 6^6

6.2 Процедуры с параметрами в NXT-G

Цель: изучить способы создания собственных блоков с входными и выходными параметрами. Рассмотреть возможность использования генератора псевдослучайных чисел.

В подпрограммы, реализуемые на NXT-G, можно передавать параметры, что значительно увеличивает гибкость кода. Параметры могут быть как входными, так и выходными. Входные параметры подпрограмма получает, чтобы использовать их в своей работе. Выходные параметры являются результатом выполнения подпрограммы; они могут быть использованы при дальнейшей работе.

Пример 6_2 Написать процедуру движения робота по квадрату. Входной параметр — длина стороны квадрата — задаётся случайным числом в диапазоне от 1 до 5 оборотов мотора.

Вначале самостоятельно напишите простую программу движения робота по квадрату со стороной в 1 оборот мотора (без использования процедуры).

Добавьте в программу новый блок `Random` (случайное) из меню *Data* (данные) полной палитры.

Рис. 6.5. *Блок генерации случайного числа из меню Data*

Разместите этот блок до основного цикла программы:

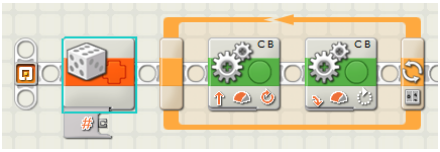


Рис. 6.6. *Создание процедуры с входным параметром: шаг 1*

В настройках блока Random задайте диапазон генерируемого случайного числа от 1 до 5.

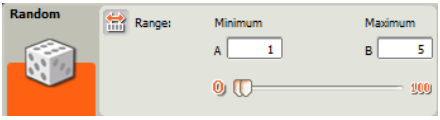


Рис. 6.7. *Настройки блока Random*

Для того, чтобы заставить робота проезжать столько оборотов, сколько выпадет в блоке Random, нужно *передать* это число на разъём Duration (длительность) блока Move (при наведении указателя мыши на разъём появляется всплывающая подсказка).

Обратите внимание на следующее:

- При передаче данных на разъём Duration в настройках блока Move можно указывать только Degrees, Rotations или Seconds, но не Unlimited. При этом варианты Degrees и Rotations обрабатываются одинаково — как Degrees. Поэтому выберите Degrees.
- Число оборотов, которое должен произвести мотор, следует перевести в градусы из расчёта: один оборот мотора — 360°.
- Поместите в цикл блок Math (Математика) из меню *Data*, и настройте его на умножение (Multiplication) на 360.

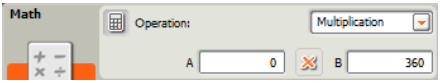


Рис. 6.8. *Настройки блока Math*

В блоке Math доступны шесть операций:

- Addition — сложение;
- Substraction — вычитание;

- Multiplication — умножение;
- Division — деление;
- Absolute Value — абсолютная величина;
- Square Root — квадратный корень.

Соедините *шиной данных* выходной разъём блока `Random` и входной разъём A блока `Math`.

Затем соедините шиной выходной разъём Result блока `Math` (числовые выходные разъёмы помечаются символом «#») и входной разъём Duration блока `Move`, который «соответствует» длине стороны квадрата.

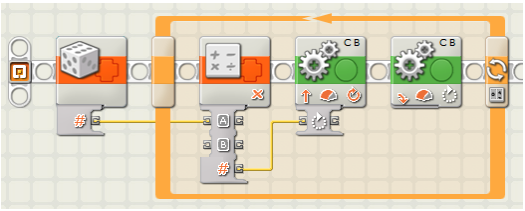


Рис. 6.9. *Создание процедуры с входным параметром: шаг 2*

Проверьте работу программы.

Оформим часть программы в виде процедуры (подпрограммы) с параметром. Для этого следует выделить всё, что по смыслу должно попадать в процедуру, **кроме параметра** (в данном случае — кроме блока `Random`),

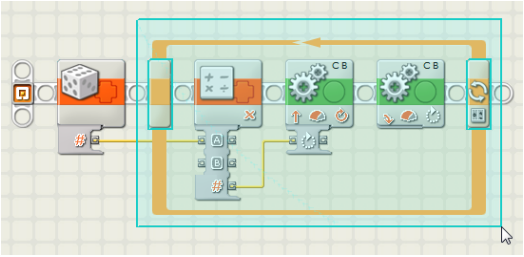


Рис. 6.10. *Создание процедуры с входным параметром: шаг 3*

и нажать кнопку Create My Block (создать блок) на панели инструментов или воспользоваться меню *Edit \ Make A New My Block*.

В появившемся окне введите в соответствующие поля имя подпрограммы (например, `sq_param`) и описание подпрограммы по-русски, а затем создайте иконку блока. В конце нажмите на кнопку Finish.

Обратите внимание на то, что у нового блока появились входной и выходной разъёмы для передачи данных:

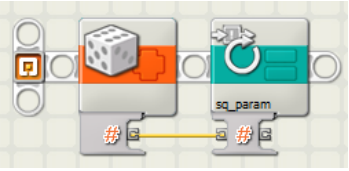


Рис. 6.11. *Входной и выходной разъёмы подпрограммы*

Выполните двойной щелчок на блоке `sq_param` — в новой вкладке открылось содержимое процедуры. Измените название входного параметра на более подходящее — «Длина стороны квадрата»:

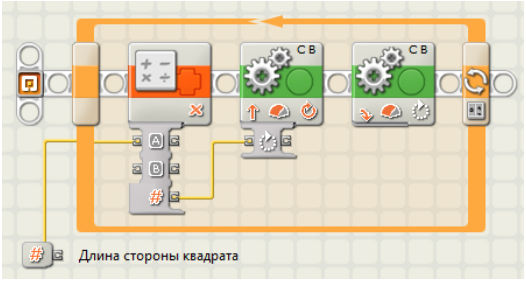


Рис. 6.12. *Изменение названия входного параметра подпрограммы*

Сохраните файл процедуры и закройте его.

Удалите блок `sq_param` из программы, а затем снова добавьте его из палитры My Blocks. Обратите внимание на изменение всплывающей подсказки при наведении указателя мыши на разъём блока `sq_param`:

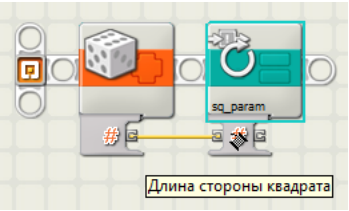


Рис. 6.13. *Всплывающая подсказка входного параметра подпрограммы*

Выведем на экран блока NXT текущую длину стороны квадрата, измеряемую в оборотах двигателя. После блока `Random` «вытяните» мышкой ещё одну направляющую для блоков (начать создание новой направляющей — щелчок левой кнопкой мыши; завершить — двойной щелчок мыши):

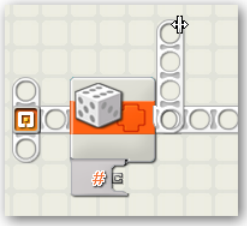


Рис. 6.14. *Создание новой направляющей для блоков*

Вывод числовой информации на экран NXT был подробно рассмотрен в примере 4_3.

После добавления всех нужных блоков и их соединения получим следующую программу:

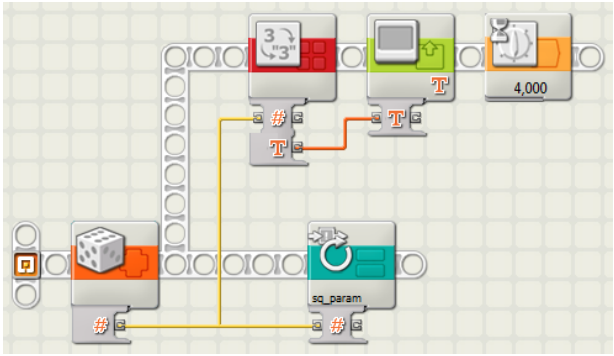


Рис. 6.15. *Движение робота по квадрату с использованием подпрограммы с входным параметром и выводом информации на экран*

Настройки блока `Display` тоже посмотрите в примере 4_3 главы 4.

Задание 6^7 Написать программу движения робота по «квадратной» спирали: один шаг, поворот, два шага, поворот и т. д. (рис. 6.16).

Задание 6^8 Написать программу движения робота по «квадратной» спирали: две ветки длиной 1 шаг, затем две ветки длиной 2 шага и т. д. (рис. 6.16). Организовать вывод на экран длину очередной ветки (в шагах).

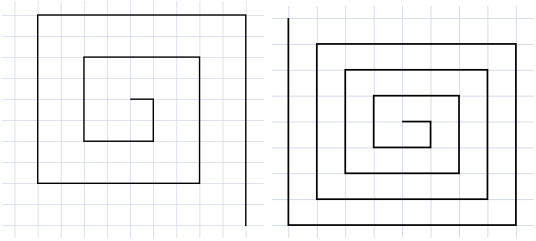


Рис. 6.16. *Квадратная спираль к заданию* 6^7 *(слева) и к заданию* 6^8 *(справа)*

Задание 6^9 Написать процедуру движения робота по сторонам правильного N-угольника. В качестве параметров передаются (1) число сторон правильного N-угольника и (2) длина его стороны. Параметры должны генерироваться случайно и выводиться на экран блока NXT в разных строчках.

Рассмотрим процесс создания собственного блока *с входным и выходным параметрами.*

Пример 6_3 Разработать блок для перевода градусов в радианы.

Для решения этой задачи мы должны не только составить программу вычисления радиан, но и предусмотреть два момента: (1) в наш код данные должны откуда-то поступать и (2) результат должен куда-то выводиться. Это необходимо сделать с той целью, чтобы в программе явно присутствовали входная и выходная шины данных. Договоримся, что данные (градусы) будут поступать из генератора случайных чисел, а результат (радианы) будет выводиться на экран. Получим следующую программу, из которой для нас наиболее интересны математические блоки умножения и деления; их настройки показаны на рис. 6.17:

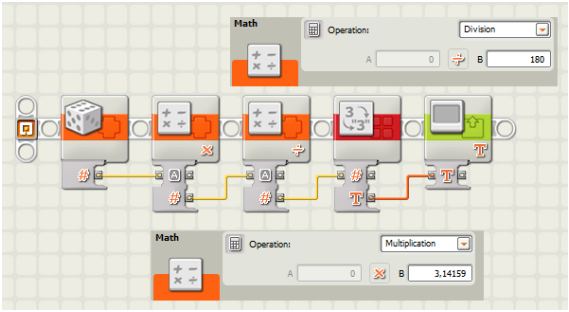


Рис. 6.17. *Программа для перевода градусов в радианы*

Построим из блоков умножения и деления собственный блок. Для этого выделим два названных блока и нажмём на панели инструментов кнопку Create My Block.

Введём имя блока и его описание (рис. 6.18). Обратите внимание на то, что в теле блока появился выходной параметр (Result).

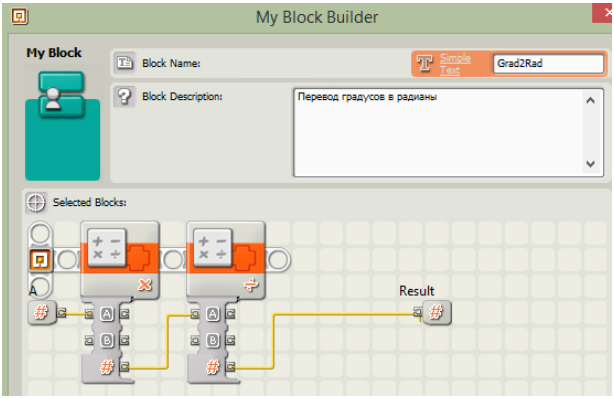


Рис. 6.18. *Первый шаг создания блока для перевода градусов в радианы*

Окончательно после изменения имён входных и выходных переменных блок примет вид (рис. 6.19):

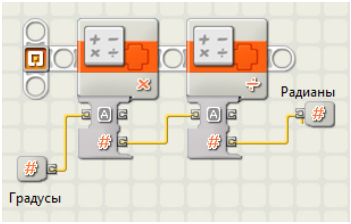


Рис. 6.19. *Блок для перевода градусов в радианы*

Тема 7

Переменные и константы

Цель: изучить возможности использования переменных и констант в языке NXT-G.

Для хранения данных в программе используются переменные и константы. Переменная представляет собой область памяти, которой назначено некоторое имя. Определяющей характеристикой переменной является её тип: в переменную можно записывать данные только её собственного типа. Например, в NXT-G имеется три основных типа: числовой (Number), текстовый (Text) и логический (Logic). Соответственно каждая переменная будет иметь один из названных типов. Тип однозначно определяет (1) размер области памяти, отводимой для хранения переменной и (2) множество операций, которые можно выполнять с этой переменной.

Переменные можно использовать и для передачи данных из собственных блоков в главную программу и наоборот. Для этого и в программе, и в созданном блоке нужно определить переменные с одним именем и типом.

Константа, как и переменная, есть поименованная область памяти определённого типа. Однако значение переменных можно менять в процессе выполнения программы, в то время как константы определяются перед запуском программы и при её выполнении изменяться не могут. В языке NXT-G это различие визуально проявляется в том, что у переменных есть как входные, так и выходные разъёмы, в то время как у констант — только выходные.

Благодаря наличию сквозных разъёмов у большинства стандартных блоков во многих случаях можно составить код без переменных (как мы это и делали до сих пор). Но существуют задачи, при решении которых нельзя обойтись без использования переменных.

Рассмотрим один из примеров. Вначале подробно опишем работу с переменными.

Пример 7_1 На белом поле нарисовано несколько параллельных чёрных полос. Написать программу для подсчёта количества этих полос.

Будем решать задачу по шагам. Вначале создадим программу, по которой робот будет двигаться до чёрной линии (см. задание 3^3 на стр. 31). Теперь, прежде, чем двинуться дальше, нужно сохранить информацию о найденной линии, «посчитать» её. Для хранения этих данных воспользуемся переменной «Линии». Для её создания используем блок `Variable` (Переменная) в пункт меню *Data* полной палитры:

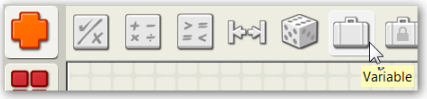


Рис. 7.1. *Доступ к блоку Variable в меню Date*

Посмотрите на настройки блока (рис. 7.2). По умолчанию среда создаёт три переменные: логического, числового и текстового типа. Имена переменных задаются в соответствии с их типом, что неудобно: имена должны объяснять назначение и смысл данных, хранящихся в переменной.

По этому поводу существует известный анекдот о том, как у программиста было трое детей, которых звали A1, A2 и A3.

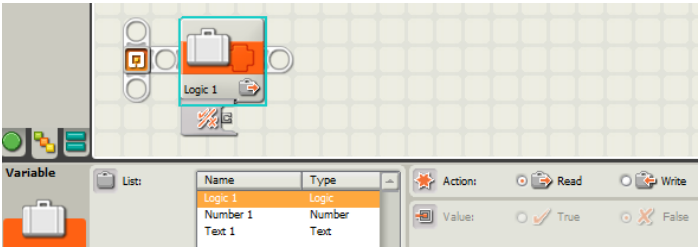


Рис. 7.2. *Настройки блока Variable по умолчанию*

Поэтому принято вначале определять собственные переменные, задавать им понятные имена и тип, а затем уже использовать. Создадим переменную «Линии» числового типа.

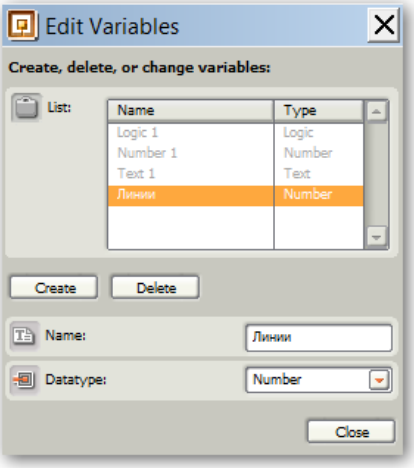


Рис. 7.3. *Окно редактора переменных*

Для этого:

- зайдите в главном меню в пункт *Edit* и далее в *Define variables* (Определить переменную);
- в открывшемся окне редактора переменных *Edit Variables* (рис. 7.3) нажмите кнопку CREATE (Создать);
- имя переменной можно задать произвольное (в том числе и по-русски), но принято именовать переменные в соответствии с тем, для каких целей они используются в задаче. В полях Name (Имя) и Data type (Тип данных) введите, соответственно, «Линии» и «Number»;
- нажмите кнопку CLOSE.

Снова щёлкните по блоку `Variable` в программе и посмотрите, как изменились настройки. Рассмотрим их подробнее.

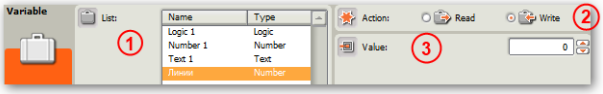


Рис. 7.4. *Настройки блока Variable после создания переменной*

Настройки переменной состоят из следующих полей:

1. Поле List (Список) содержит список имеющихся переменных с описанием их имени Name и типа Type. Обратите внимание, что

созданная нами переменная с именем «Линии» уже есть в списке и имеет числовой тип.

2. Action (Действие). Оно может быть или Read (Считывать) или Write (Записывать). В первом случае из переменной можно только читать ранее записанные данные. Во втором — и считывать и записывать в неё. В программе может быть несколько блоков для одной и той же переменной, причём в одном данные считываются, а в другом — записываются. Если выбран режим Write, то данные можно и читать, и писать в переменную.
3. Value (Значение). Можно задать начальное значение переменной. В зависимости от выбранного типа значения будут меняться. В нашем случае значение должно быть равно нулю; это означает, что пока ни одной линии не найдено.

Продолжим создание программы подсчёта количества чёрных полос.

Добавьте в программу блок `Math`, настроенный на прибавление единицы к переменной «Линии», при этом результат снова должен быть сохранён в той же переменной. Выведите это значение на экран, как мы уже делали в примере 4_3 (стр. 43). Чтобы программа продолжала подсчёт, требуется заключить её в цикл. Получим следующий код:

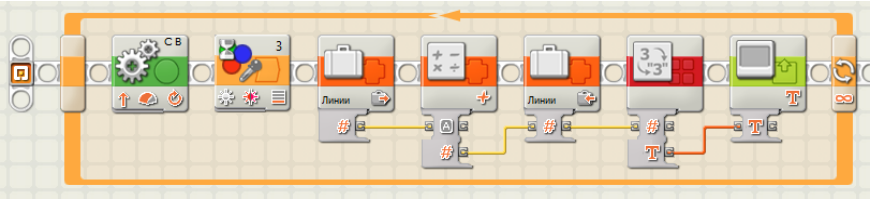


Рис. 7.5. *Программа подсчёта количества чёрных полос*

Задание 7[1] Запустите программу на выполнение. Правильно ли ведётся подсчёт линий?

Если на поле были достаточно широкие полосы, то оказывается, что робот успевает посчитать одну линию несколько раз, пока проезжает над ней датчиком цвета. Надо каким-то образом приостановить подсчёт. Самый простой вариант, который первым приходит на ум — это использование задержки времени. Но насколько она должна быть длинной? Очевидно, что чем шире полоса, тем дольше требуется ждать. Этот способ не очень хорош.

Другой вариант основан на том, что мы знаем — ждать надо до тех пор, пока чёрная полоса не закончится, то есть до более светлого участка. В итоге программа примет вид:

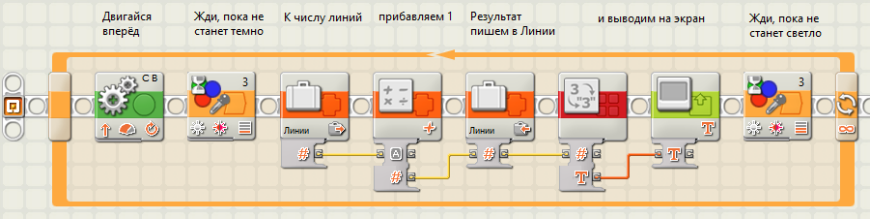


Рис. 7.6. *Окончательный вариант программы подсчёта количества чёрных полос*

Задание 7[2] Когда остановится робот? Придумайте способ программной остановки робота и попробуйте реализовать его.

Задание 7[3] Напишите программу для робота, чтобы он останавливался на пятой полосе.

Задание 7[4] Напишите программу для робота, которая подсчитывает количество нажатий на кнопку (кнопку присоединить к свободному порту). Текущее число нажатий должно выводиться на экран. Придумайте способ останова программы по условию.

Для использования констант в NXT-G имеется блок `Constant` из меню `Date`.

Рис. 7.7. *Доступ к блоку Constant из меню Date*

Обратите внимание на изображение блока — это чемоданчик с замком, что означает невозможность изменения его содержимого.

Рассмотрим настройки блока:

- Action — Choose from list (Выбрать из списка) или Custom (Создать пользовательскую). По умолчанию выбран второй вариант. Это означает, что константу можно определить в самом блоке, заполнив остальные поля. Чтобы стал доступен вариант Choose from list, нужно заранее определить константу в редакторе констант (*Edit\ Define Constants*) так же, как мы это делали для переменной. Этот

вариант делает возможным неоднократное использование одной и той же константы в разных частях программы.

- Data Type — как и для переменной можно выбрать логический, числовой или текстовый тип данных.
- Value — значение константы.
- Name — имя константы; после ввода имени оно автоматически отображается на блоке.

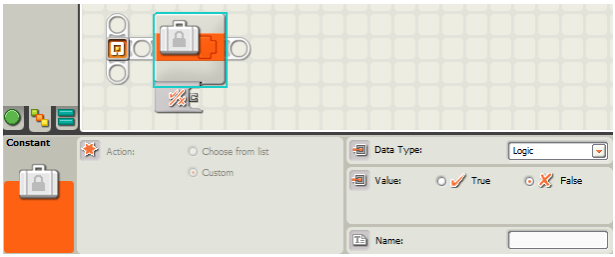


Рис. 7.8. *Настройки блока Constant по умолчанию*

В целом работа с блоком `Constant` аналогична блоку `Variables`.

Тема 8

Потоки

Цель: изучить возможности использования потоков в языке NXT-G.

В жизни нам часто приходится делать несколько дел сразу. Например

- идти и есть мороженое;
- пить чай, разговаривать и смотреть фотографии;
- вести мяч, оценивать ситуацию на площадке и придумывать план действий;
- спускаться по лестнице и разговаривать по телефону.

Для нас это совершенно обычно, и, если разобраться, то окажется, что на самом деле мы умеем производить одновременно гораздо больше действий. Это называется *параллельным выполнением задач*. Все системы человеческого организма приспособлены к параллельности.

Современные компьютеры широко используют в своей работе технологию *многопоточности*. При этом программа составляется из нескольких потоков (`threads`), которые выполняются параллельно. Наш робот тоже умеет выполнять многопоточные программы.

На самом деле мы уже использовали многопоточность *неявно*: когда запускаются моторы, они начинают работать, а управление немедленно передаётся дальше, на следующий блок программы. Таким образом моторы работают в параллельном режиме. Таким же образом можно запустить блок `Sound`.

Важно понимать, что параллельность не только придаёт гибкость коду, но и *является потенциальным источником ошибок*. Боль-

шинство из них возникает, когда в двух или более параллельных потоках пытаются одновременно использовать один и тот же ресурс. Например, в первом потоке мы запускаем мотор вперёд, а во втором одновременно пытаемся запустить тот же мотор назад. Или один поток пишет в какую-то переменную некоторые значения, а второй поток одновременно пишет в ту же переменную другие значения. Ещё одна известная проблема возникает при несинхронном запуске или окончании одного из параллельных потоков. Все описанные случаи носят общее название *проблемы синхронизации*: как организовать совместный доступ из разных потоков к общему ресурсу. При написании программ от нас потребуется известная аккуратность для того, чтобы избежать подобных ситуаций.

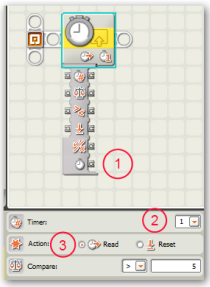
Пример 8_1 Двигаемся и разговариваем.

Рассмотрим следующую задачу. Робот должен доехать до чёрной линии и остановиться, сообщив об этом. Во время движения робот должен постоянно произносить какую-либо фразу, например, «Tracking object». Чтобы поведение робота было более «человеческим», сделаем так, что фраза будет произноситься не через равные промежутки времени, а через случайные.

Для решения воспользуемся новым блоком `Timer` из меню *Sensor*, который отсчитывает миллисекунды, а саму программу оформим в виде двух потоков команд.

Рис. 8.1. *Доступ к блоку Timer в меню Sensor*

После его добавления можно произвести настройку:



Из порта № 1 считывается количество миллисекунд, прошедших с момента последнего перезапуска таймера. Поле № 2 (Timer) указывает на номер таймера. Всего в одной программе можно использовать до трёх таймеров. Поле № 3 (Action) показывает действие, которое можно выполнить: считать данные с таймера (Read) или сбросить таймер в ноль (Reset). Другие порты и поля нам пока не понадобятся.

Кроме того, для понимания программы нужно вспомнить правила работы с циклом, выход из которого происходит по логическому условию (см. § 4.5 на стр. 45).

Рассмотрим код программы.

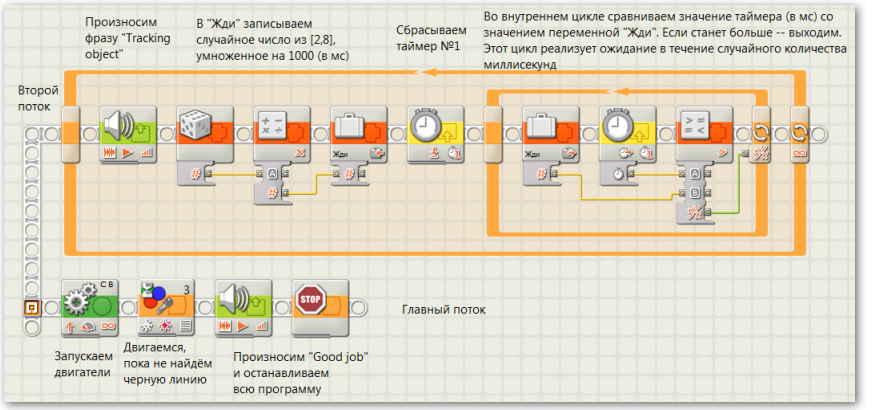


Рис. 8.2. *Программа к примеру* 8_1

Возможность создания нового потока мы уже рассматривали в примере 6_2. Чтобы создать новый поток программ, не исходящий из начальной точки, можно установить указатель мыши на нужном участке основной цепочки, нажать клавишу Shift и переместить указатель мыши вверх или вниз. Затем можно размещать новые блоки.

Однако бывает более удобным вначале составить код нового потока, а только затем подвести к нему серую полосу.

Задание 8^1 Внимательно изучите код, прочитайте комментарии. Подумайте, как следует настроить каждый из блоков. Наберите программу, опробуйте её работу. Можно ли было реализовать этот алгоритм в одном потоке? Обоснуйте ответ.

Задание 8^2 Подключите к роботу два датчика касания. Напишите программу, которая в двух потоках (по одному на каждый датчик) подсчитывает и отображает на экране *суммарное* количество нажатий на кнопки. (Например, если на первую кнопку нажали в сумме 3 раза, а на вторую — 5 раз, то на экране должно быть число 8.) Отображение числа на экране реализуйте в третьем потоке.

Задание 8^3 Измените предыдущую программу так, чтобы на экран выводилась не сумма, а *произведение* количества нажатий на каждую кнопку. (Например, если на первую кнопку нажали в сумме 3 раза, а на вторую — 4 раза, то на экране должно быть число 12.)

Задание 8^4 Добавьте в предыдущие программы условие останова: при превышении определённого значения (в сумме или произведении) программа должна завершаться.

Тема 9

Управление движением робота при помощи системы с отрицательной обратной связью

Задача управления является очень актуальной в современной науке и технике. Пусть имеется система (*объект управления*), которую мы должны поддерживать в заданном состоянии. Для этого у нас имеется *регулятор*, который (1) производит сбор информации о текущем состоянии системы в момент времени t и (2) вычисляет управляющий сигнал $U(t)$. Этот сигнал подаётся объекту управления, возвращая его в заданное состояние. Такая схема носит название *системы с отрицательной обратной связью,* поскольку при отклонении от равновесия регулятор стремится вернуть систему в нормальное состояние. В этой теме мы рассмотрим два вида регуляторов: более простой *релейный* и более устойчивый *пропорциональный*.

9.1 Релейный регулятор

Цель: научиться строить систему управления автономным роботом на основе простейшего релейного регулятора.

Пример 9_1 Движение робота вдоль стены.

Пусть имеется робот, оснащённый датчиком ультразвука, и не очень ровная стена (с небольшими выступами и впадинами). Требует-

ся написать программу управления движением робота вдоль стены на заданном расстоянии.

Следует заранее продумать положение датчика ультразвука. Как упоминалось выше (см. с. 33), этот датчик медленный. Поэтому его следует располагать чуть впереди корпуса робота. Кроме этого, его следует сместить как можно дальше от стены (см. рис. 9.1), потому что на малых расстояниях (5–7 *см*) показания датчика ультразвука становятся слишком неточными. Далее мы увидим, что направление датчика также следует подкорректировать.

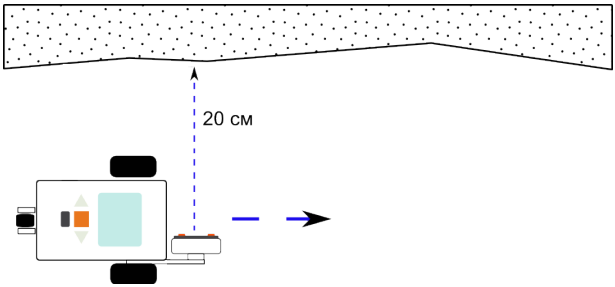


Рис. 9.1. *Примерное расположение датчика ультразвука*

Алгоритм движения робота, записанный в словесной форме, может быть примерно таким (рис. 9.2):

1. двигаться прямо;
2. если расстояние до стены больше заданного, то повернуть к стене;
3. если расстояние до стены меньше заданного, то повернуть от стены;
4. повторять шаги 2–3 бесконечно (или до наступления некоторого события).

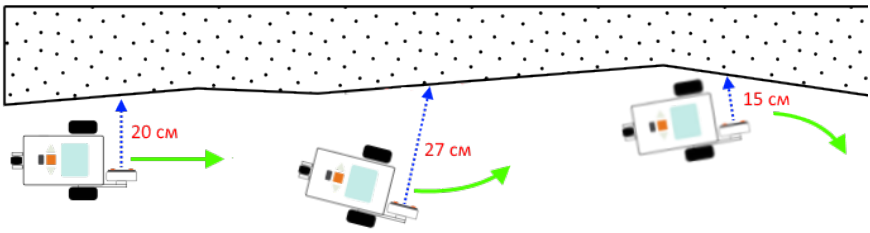


Рис. 9.2. *Иллюстрация алгоритма движения вдоль стены*

Одним из самых очевидных решений этой задачи является *релейный регулятор*.

Реле в электротехнике — это замыкатель с автоматическим возвратом в исходное состояние (хотя существует множество других разновидностей). Другими словами, при превышении сигналом (например, током) некоторого *предельного значения* происходит замыкание реле. Как только ток снизится, реле размыкается. В нашем случае определение релейный по отношению к регулятору означает лишь то, что мы описываем поведение системы лишь *для двух случаев*: значение сигнала (1) меньше заданного и (2) больше заданного.

Для определённости примем в качестве заданного расстояния 20 *см*. Получим следующую программу (рис. 9.3):

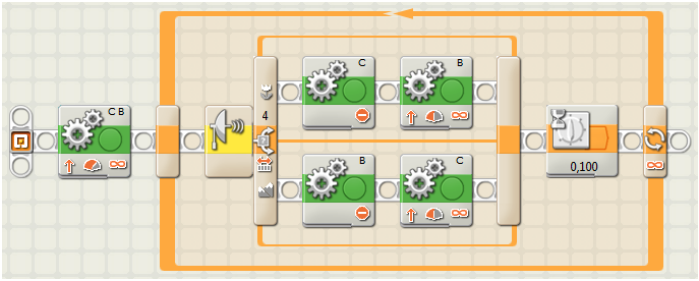


Рис. 9.3. *Реализация релейного алгоритма*

Как видно из текста программы, после включения обоих моторов запускается бесконечный цикл, в котором реализован вышеописанный словесный алгоритм. Хорошо заметной особенностью этого подхода является «рыскающее» движение: робот всегда поворачивает с одной и той же интенсивностью, независимо от того, насколько далеко или близко он оказался по отношению к стене. Поэтому траектория *всегда* будет зигзагообразной, так как во время поворотов робот всегда будет «прыгать» вокруг среднего значения 20 *см*. Так как датчик ультразвука является «медленным», в цикле используется небольшая задержка (0,1 с) для того, чтобы показания датчика успевали обрабатываться блоком NXT. Величина задержки фактически определяет время, в течение которого робот будет двигаться в неизменном направлении. Другими словами, увеличивая время задержки мы получим более крупные «зубцы» траектории. Как отмечалось ранее, делать задержку меньше 0,06 с не имеет смысла, потому что в этом случае датчик ультразвука просто не успеет провести измерения.

При использовании релейного регулятора возможны частые уходы робота с дистанции. Одна из причин состоит в том, что при выбранном нами расположении датчика ультразвука робот не сможет различать положения, симметричные относительно нормального положения. То есть два положения, показанные на рис. 9.4 будут

идентичными, и расстояние до стены в обоих случаях окажутся больше 20 *см*. В соответствии с вторым шагом алгоритма робот должен повернуть к стене (влево). И если для второго положения это верно, то для первого — ошибочно: робот ещё больше уйдёт с трассы.

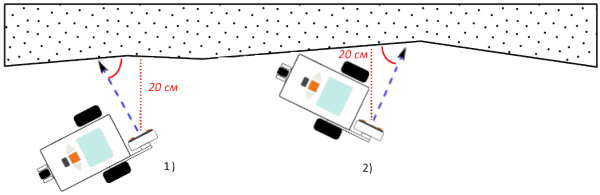


Рис. 9.4. *Куда теперь поворачивать?*

Эту проблему можно решить, если расположить датчик не перпендикулярно к направлению движения (т. е. строго влево), а под углом 45° к направлению движения (рис. 9.5).

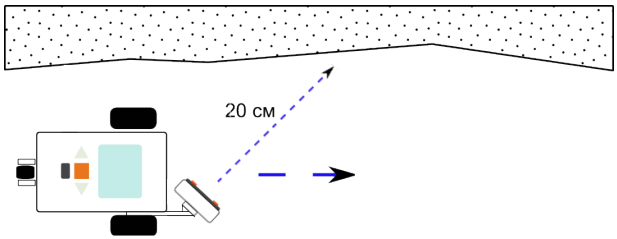


Рис. 9.5. *Разворот датчика ультразвука под углом* 45° *к направлению движения*

Так нам удастся избежать случая, показанного на рис. 9.4. Действительно, при подруливании влево расстояние до стены будет (при небольших углах поворота $< 45°$) уменьшаться, а при поворотах вправо, наоборот, увеличиваться. Однако не стоит рассчитывать, что наш робот сможет проехать вдоль стены любой формы. Даже простой поворот на 90° может вызвать у него неожиданные трудности. Таким образом, наиболее значимым достоинством релейного регулятора в нашем случае является простота его алгоритма. В следующем разделе мы рассмотрим более интересный алгоритм управления.

Задание 9^1 Запрограммируйте движение робота вдоль стены с разными положениями датчика ультразвука. В каком случае движение

более устойчиво?

Задание 9^2 Сравните это решение с примером 4_1 на с. 38. Реализуйте релейный алгоритм движения вдоль стены без использования ветвления.

Задание 9^3 Реализуйте релейный алгоритм движения по линии с использованием ветвления.

9.2 P-регулятор

Цель: научиться строить систему управления автономным роботом на основе пропорционального регулятора.

Трудности использования релейного регулятора, о которых говорилось в п. 9.1, требуют поиска более приемлемого решения задачи управления. Одним из вариантов является *пропорциональный регулятор* (или P-регулятор). В этом случае управляющее воздействие на моторы робота *не постоянно*, как в релейном регуляторе, а изменяется *пропорционально отклонению* от заданного расстояния до стены. Другими словами, чем больше отклонение, тем активнее должны работать моторы, выравнивая траекторию робота. В идеале робот должен ехать прямо, если датчик регистрирует заданное расстояние. При небольшом отклонении следует небольшое подруливание. Если отклонение больше, то и подруливание больше. Алгоритм P-регулятора является классическим в теории систем автоматического управления.

Пример 9_2 Управление движением вдоль стены на основе P-регулятора.

Для P-регулятора управляющее воздействие $U(t)$ на моторы робота в момент времени t вычисляется по формуле:

$$U(t) = K_p \cdot E(t), \tag{9.1}$$

где

- $E(t) = X_0 - X(t)$ — отклонение робота от заданного положения (другими словами, ошибка, Error, или ещё говорят *невязка*);
- X_0 — заданное расстояние до стены (у нас 20 *см*);

- $X(t)$ — текущее показание датчика;
- K_p — усиливающий коэффициент (коэффициент пропорциональности). Всегда положительный.

Очевидно, графиком для расчёта управляющего воздействия U в зависимости от ошибки E будет прямая (см. рис. 9.6):

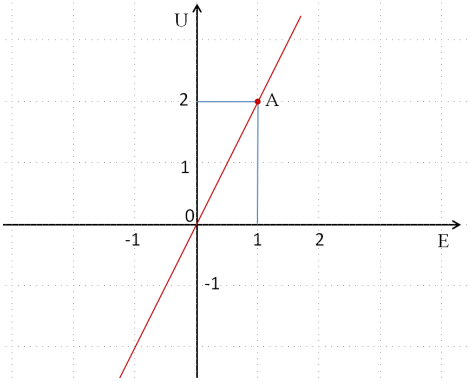


Рис. 9.6. *График для P-регулятора: зависимость U от E*

Из графика очень просто определить коэффициент K_p. Выберем произвольную точку на графике. В нашем случае это точка $A(1, 2)$. Тогда

$$K_p = U_A/E_A = 2/1 = 2.$$

Ошибка может быть как положительная (если мы ближе к стене, чем надо), так и отрицательная (если мы отъехали от стены дальше, чем необходимо). Таким образом, управляющее воздействие $U(t)$ тоже может быть как положительным, так и отрицательным.

Пусть мы имеем P-регулятор для робота, движущегося вдоль стены. Тогда мощность моторов $PowerB(t)$ и $PowerC(t)$ в момент времени t вычисляется по формулам:

$$PowerB(t) = Nm + U(t) \tag{9.2}$$

$$PowerC(t) = Nm - U(t), \tag{9.3}$$

где

- Nm — нормальная мощность моторов: мощность, с которой должны крутиться оба двигателя, если отклонение от курса равно нулю. В нашей программе можно положить $Nm = 50$;
- $U(t)$ — управляющее воздействие на моторы, вычисляемое по формуле (9.1).

  Знаки перед $U(t)$ для Вашего робота могут поменяться на противоположные в зависимости от того, какой мотор находится слева, а какой справа.

Рассмотрим смысл коэффициента пропорциональности K_p. Как видно, поворот производится в силу того, что от мощности одного мотора управляющее воздействие вычитается, тогда как к другому — прибавляется. Таким образом коэффициент K_p может усиливать или ослаблять воздействие регулятора на моторы: если он больше единицы, то происходит усиление, а если меньше — ослабление. Большой K_p сделает робот очень чутким к ошибкам, что приведёт к резким рывкам для исправления траектории. Малый K_p сделает движения робота плавнее, но на крутых поворотах робот может потерять стену и сойти с траектории. Конкретные значения коэффициента K_p, наиболее подходящие в каждом конкретном случае, будут зависеть от конструктивных особенностей робота, скорости движения (нормальной мощности), сложности трассы, используемых датчиков. Величины $PowerB(t)$ и $PowerC(t)$ должны лежать в диапазоне $[0; 100]$. Поэтому при больших ошибках (и, соответственно, больших управляющих сигналах) мощность моторов будет ограничиваться так, чтобы она не выходила из указанного диапазона. В этом случае пропорциональный регулятор не будет работать корректно, поскольку не сможет скомпенсировать большие ошибки. Таким образом, одной из особенностей P-регулятора является адекватная работа только при небольших ошибках. Для того, чтобы учесть это в нашей задаче нужно иметь стену без резких поворотов и двигаться с небольшой скоростью.

Слишком расплывчатые фразы «резкие повороты» и «небольшая скорость» используются здесь намеренно. Конкретные значения этим нечётким величинам можно придать в результате проведения экспериментов с конкретным роботом в конкретных условиях.

Реализуем алгоритм P-регулятора согласно приведённым формулам при помощи вложенных процедур (My blocks) на языке NXT-G.

Блок вычисления отклонения (ошибки) от заданного расстояния, который мы назовём Error, имеет следующие параметры:

- входные — текущее показание датчика расстояния $X(t)$;
- выходные — ошибка $E(t)$).

Блок `Error` на языке NXT-G выглядит так:

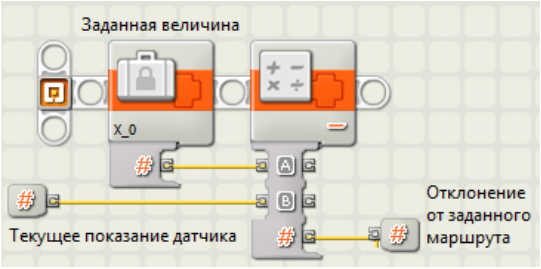


Рис. 9.7. *Подпрограмма вычисления ошибки отклонения от заданного расстояния*

Мы используем его в подпрограмме для вычисления управляющего воздействия $U(t)$. Она имеет следующие параметры:

- входные — текущее показание датчика $X(t)$;
- выходные — управляющее воздействие на моторы.

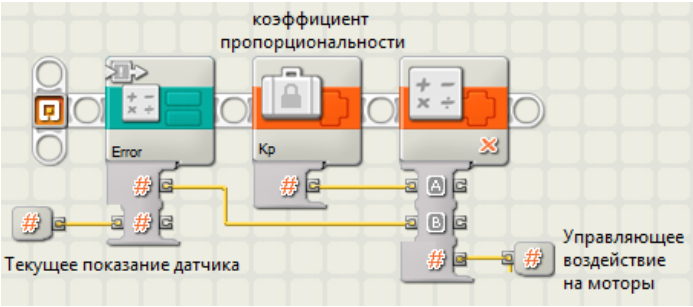


Рис. 9.8. *Подпрограмма вычисления управляющего воздействия* $U(t)$

Этот блок мы используем для разработки блока P-регулятора. Он будет иметь следующие параметры:

- входные — текущее показание датчика;
- выходные — мощность моторов B и C.

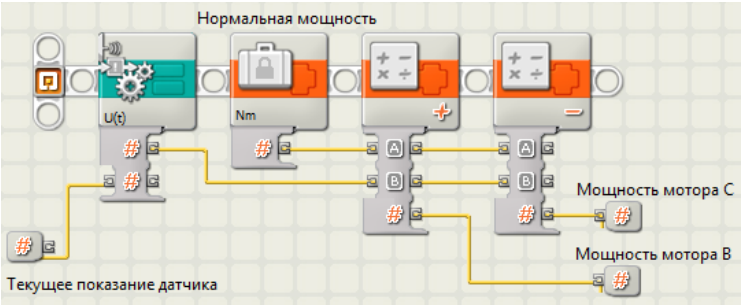


Рис. 9.9. *Подпрограмма вычисления P-регулятора*

Наконец, полная программа, реализующая бесконечное движение вдоль стены, получается с использованием блока P-регулятора и формул (9.2) и (9.3):

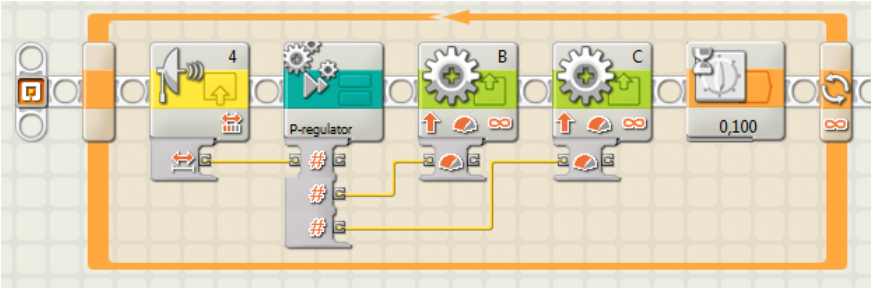


Рис. 9.10. *Пример программы движения вдоль стены с использованием P-регулятора*

Задание 9^4 Реализуйте предложенный алгоритм движения вдоль стены с использованием P-регулятора. Проверьте его работу. Настройте регулятор при помощи подбора коэффициента K_p.

При необходимости понизьте нормальную мощность, измените положение датчика ультразвука. Хороший результат даёт не горизонтальное, а вертикальное положение сенсора (когда «глаза» ультразвукового датчика расположены один над другим).

Задание 9^5 Реализуйте алгоритм следования по линии с использованием P-регулятора. Настройте регулятор, добиваясь наиболее уве-

ренного прохождения роботом крутых поворотов.

Задание 9^6 Проведите соревнование в своей группе на скоростное прохождение трассы.

Задание 9^7 После того, как робот стал двигаться по линии более или менее уверенно, попробуйте (1) увеличить подобранный коэффициент K_p вдвое; (2) уменьшить подобранный коэффициент K_p вдвое. Как изменяется характер движения робота? Удаётся ли ему пройти трассу? Какие участки наиболее проблемны?

Задание 9^8 Реализуйте алгоритм следования за рукой с использованием P-регулятора (см. задание 5^4 на с. 52). Настройте регулятор так, чтобы движение робота стало как можно более плавным.

Тем не менее, использование P-регулятора не позволяет решить задачу полностью: движение робота будет сильно зависеть от его конструкции, настроек регулятора и пр. Чтобы движение стало ещё стабильнее, нужно обратиться к более интеллектуальным алгоритмам. В первую очередь здесь следует назвать пропорционально-дифференциальный (PD) и пропорционально-интегрально-дифференциальный (PID) регуляторы. В настоящем пособии начального уровня указанные алгоритмы изучаться не будут. Интересующихся мы отсылаем к книге С. А. Филиппова [6], где подробно рассмотрены эти и другие интереснейшие задачи.

9.3 Кегельринг

Цель: рассмотреть одну из задач соревновательной робототехники и изучить возможности использования датчика оборотов.

В мире робототехники проводится множество различных соревнований. Наиболее известными и зрелищными являются сумо роботов, поиск выхода из лабиринта, скоростное движение по линии и танцы роботов. Большое количество соревнований проводятся на всемирной олимпиаде роботов (WRO), где перед роботами ставятся сложные ин-

теллектуальные задачи. Мы рассмотрим одну из соревновательных задач начального уровня — так называемый «Кегельринг».

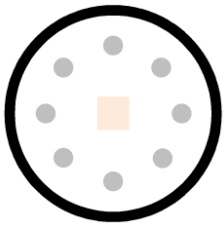


Рис. 9.11. *Поле для соревнования «Кегельринг»*

Имеется робот, оснащённый датчиками цвета (или света) и ультразвука. Также имеется ринг и расставленные внутри него кегли — алюминиевые банки (рис. 9.11). Робот должен вытолкнуть все кегли за пределы ринга. Начальное положение робота — центр ринга. Побеждает робот, вытолкнувший все кегли из ринга за минимальное время. На решение задачи роботу даётся 2 минуты.

Наиболее очевидное общее решение задачи распадается на две отдельные подзадачи:

1. поиск кегли;
2. выталкивание кегли.

Поиск кегли

Для поиска банки добавьте в конструкцию робота датчик ультразвука, направленный вперёд. Запрограммируйте поведение робота по следующему алгоритму:

1. робот, находясь в центре ринга, разворачивается на месте, пока датчик не зафиксирует расстояние до объекта менее заданного;
2. остановка.

Задание 9^9 Составьте программу поиска роботом кегли.

Выталкивание кегли

Если кегля найдена, то алгоритм её выталкивания состоит в следующем:

1. двигаться вперёд до чёрной линии;

2. вернуться назад.

Возникает вопрос: насколько сильно робот должен возвращаться назад? Очевидно, на то же расстояние, которое он проехал до линии. Как определить это расстояние?

Моторы LEGO Mindstorms содержат встроенный датчик оборотов. Снимать показания этого датчика можно при помощи блока `Rotation Sensor` из меню *Sensor* полной палитры. Этот блок подсчитывает количество градусов (с точностью до 1°) или количество полных оборотов вала мотора.

Рис. 9.12. *Доступ к блоку Rotation Sensor в меню Sensor*

Блок `Rotation Sensor` имеет довольно много входных и выходных разъёмов. Для решения нашей задачи понадобится только один разъём, выдающий текущее количество градусов или оборотов, совершённых валом мотора.

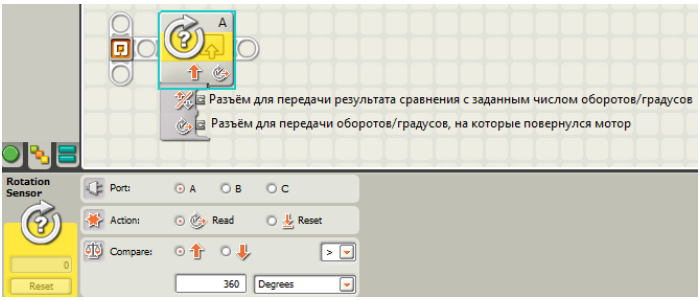


Рис. 9.13. *Настройки блока Rotation Sensor*

Поле Action задаёт тип действия: Read — считать данные; Reset — сбросить показания датчика в ноль.

Пример 9₃ Демонстрация работы датчика оборотов.

- Запрограммируйте робот так, как показано на рис. 9.14. Обратите внимание, что первый блок `Rotation Sensor` сбрасывает показания датчика оборотов в ноль, а второй работает в режиме чтения.
- Загрузите программу на робот при помощи кнопки Download and Run.

- Не отсоединяя USB-кабель вращайте колесо робота вперёд/назад и наблюдайте за показаниями на экране.
- Сравните показания на экране с показаниями в поле обратной связи датчика оборотов.

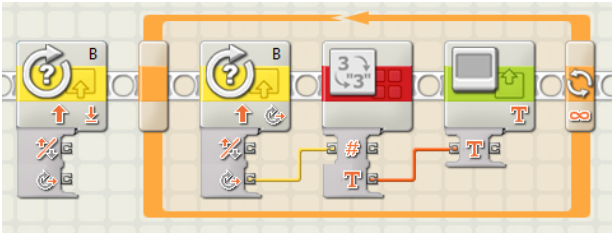


Рис. 9.14. *Демонстрация работы датчика оборотов*

Задание 9^{10} Измените программу так, чтобы данные выводились в оборотах.

Кегельринг. Общий алгоритм

Итак, общий алгоритм кегельринга состоит в следующем:

1. Разворот на месте, пока датчик ультразвука не обнаружит банку.
2. Двигаться вперёд до чёрной линии.
3. Вернуться назад на столько же оборотов, сколько он проехал вперёд, выталкивая очередную банку.

   Напоминаем, что на концентратор данных блока `Move` можно передавать только градусы, но не обороты.

4. Перейти к пункту 1.

Задание 9^{11} Реализуйте алгоритм кегельринга целиком. Устройте соревнование «Кегельринг» по описанным выше правилам.

Задание 9^{12} Создайте программу для имитации дискового телефона. Пользователь «набирает номер» из трёх цифр (от нуля до девяти). После набора каждой цифры диск должен вернуться в исходное состояние. В конце набора робот проговаривает набранный номер вслух (по цифрам).

Литература

[1] Блог А. Колотова «NiNoNXT» // URL: http://nnxt.blogspot.ru

[2] Вводный курс по программированию NXT. Часть I. Учебное пособие научно-технической конференции LEGO «Инженерная культура: от школы к производству» — Научно-Методический Центр Университета TUFTS, 2012. — 40 с.

[3] Вводный курс по программированию NXT. Часть II. Учебное пособие научно-технической конференции LEGO «Инженерная культура: от школы к производству» — Научно-Методический Центр Университета TUFTS, 2012. — 34 с.

[4] Копосов Д. Г. Первый шаг в робототехнику: практикум для 5–6 классов / Д. Г. Копосов. — М.: БИНОМ. Лаборатория знаний, 2012. — 286 с.

[5] Сервомотор [Электронный ресурс] // LEGO education: [сайт]. URL: http://education.lego.com/en-us/lego-education-product-database/mindstorms/9842-interactive-servo-motor

[6] Филиппов С. А. Робототехника для детей и родителей. — СПб.: Наука, 2013. — 319 с.

[7] NXT® motor internals [Электронный ресурс] // Philo's Home Page: [сайт]. URL: http://www.philohome.com/nxtmotor/nxtmotor.htm

Учебное издание

Дженжер Вадим Олегович

Денисова Людмила Викторовна

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ LEGO-РОБОТОВ НА ЯЗЫКЕ NXT-G

Учебное пособие

Компьютерная верстка В. О. Дженжер, Л. В. Денисова

Дизайн обложки Ю. Овчинникова

Подписано в печать 1.10.2014. Формат А5.

Гарнитура «Computer Modern Roman». Бумага офсетная. Печать офсетная.

Усл. печ. л. 5,5. Тираж 1000 экз.

ООО «ИНТУИТ.ру»

Национальный Открытый Университет «ИНТУИТ»

Москва, Электрический пер., 8, стр. 3.

Телефон: +7 (499) 253-9312, 253-9313, факс: +7 (499) 253-9310

E-mail: info@intuit.ru, http://www.intuit.ru