5

Ожидание, повторение, контейнеры и многозадачность

В предыдущей главе вы изучили, как запрограммировать робота на выполнение различных действий, например на передвижение. В этой главе вы узнаете, как приостановить выполнение программы с помощью блока **Ожидание** (Wait), как повторить последовательность действий с помощью блока **Цикл** (Loop), как запустить несколько блоков одновременно и даже как создать собственный блок.

Блок Ожидание

До сих пор вы использовали блоки, которые программируют робота на движение, воспроизведение звуков или выведение информации на экран. Теперь речь пойдет о блоке, который ничего не делает, кроме приостановки выполнения программы в течение заданного промежутка времени. Этот блок носит название **Ожидание** (Wait), он показан на рис. 5.1.

Настройки ожидания

Работа с блоком **Ожидание** (Wait) выполняется так же, как и с любым другим программным блоком: поместите его в область программирования, а затем настройте режим его работы и параметры. В этой главе вы будете использовать только один режим — **Время** (Time).

В режиме **Время** (Time) блок **Ожидание** (Wait) приостанавливает выполнение программы на определенное количество времени, например пять секунд. После того как время истекло, программа переходит к следующему программному блоку. Количество секунд, которое вы указываете как значение параметра **Секунды** (Seconds), может быть либо целым числом, например 14, либо десятичным, например 1,5. Чтобы приостановить выполнение программы на 50 миллисекунд (0,05 секунды), вы должны ввести 0,05.

Использование блока Ожидание

Зачем нужно использовать блок, который ничего, кроме ожидания, не делает? Давайте взглянем на блок **Ожидание** (Wait)

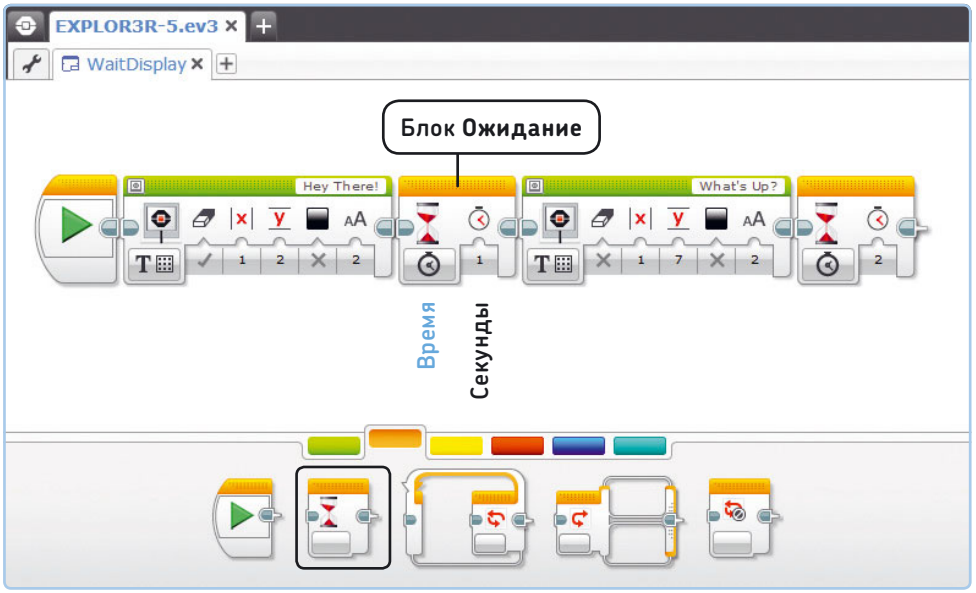


*Рис. 5.1. Программа WaitDisplay содержит два блока **Ожидание**, работающих в режиме **Время** и позволяющих приостановить выполнение программы после отображения текста на экране модуля EV3*

в действии. Создайте новый проект, сохраните его под именем EXPLOR3R-5, а в нем создайте программу WaitDisplay в соответствии с инструкциями, приведенными на рис. 5.1.

Программа WaitDisplay

При запуске программы на экране модуля EV3 должен появиться текст «Hey There!» и следом, через секунду, «What's Up?». Второй блок **Ожидание** (Wait) дает вам время, чтобы успеть прочитать текст, который отображается на экране модуля EV3. Без него программа завершит работу сразу после того, как текст будет выведен на экран, поэтому прочитать сообщение будет невозможно. Блоки **Ожидание** (Wait) могут пригодиться при программировании робота с датчиками, в чем вы убедитесь в следующей главе.

ПРАКТИКУМ № 14: ОСТАВЬТЕ СООБЩЕНИЕ!

Сложность: **Время:**

Дополните программу WaitDisplay так, чтобы робот отображал сообщение, в котором говорится, куда вы отправились, когда вышли из дома. Добавьте блоки **Ожидание** (Wait) между строк текста, чтобы ваше сообщение было удобно читать. Не забудьте проинструктировать членов семьи, как получить доступ к сообщению!

ПРАКТИКУМ № 15: ТАЙМЕР ДЛЯ НАСТОЛЬНЫХ ИГР!

Сложность: **Время:**

Создайте программу, отображающую на экране модуля EV3 таймер, который можно использовать во время настольных игр. Экранный таймер должен показывать, сколько времени осталось для каждого хода (рис. 5.2). Когда время истекло, робот должен сказать «Game Over!», сообщая таким образом, что ход окончен и настала очередь следующего игрока. В качестве альтернативы вы можете использовать блоки **Звук** (Sound), чтобы ваш робот говорил, сколько осталось времени.

СОВЕТ **Используйте несколько блоков Экран (Display) в режиме Изображение (Image) с блоками Ожидание (Wait) между ними. Выбирайте пункты Timer 0, Timer 1 и так далее в списке изображений.**

Рис. 5.2. Таймер для настольных игр из практикума № 15

Блок Цикл

Представьте, что вы идете по траектории квадратной формы, как показано на рис. 5.3. При этом вы повторяете определенный шаблон несколько раз: идете прямо, затем поворачиваете направо, идете прямо, поворачиваете направо и так далее.

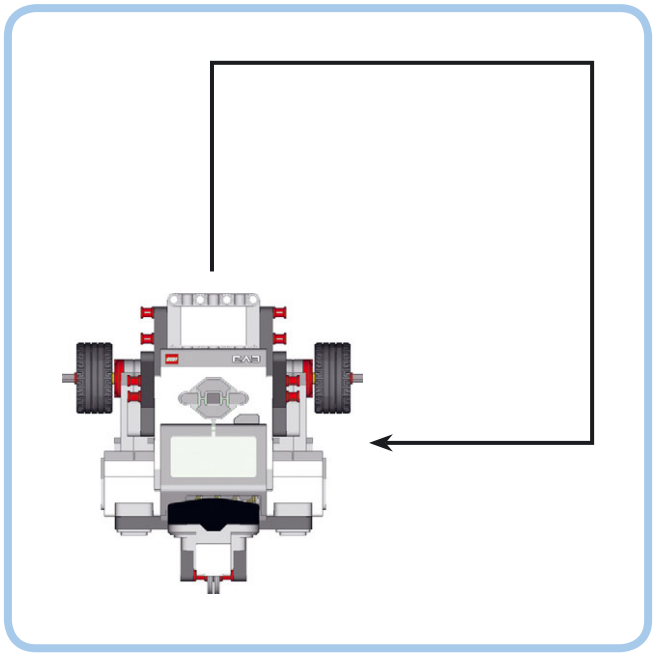


Рис. 5.3. The EXPLOR3R движется по траектории квадратной формы

Для создания такого шаблона движения для вашего робота вы можете использовать один блок **Рулевое управление** (Move Steering), чтобы робот шел прямо, а второй, чтобы он поворачивал направо. Чтобы робот прошел полный квадрат и вернулся в исходное положение, вы должны использовать каждый блок четыре раза, в общей сложности восемь блоков.

Вместо того чтобы использовать восемь блоков **Рулевое управление** (Move Steering), чтобы создать такую программу, гораздо проще применить блок **Цикл** (Loop), который позволяет повторять последовательности блоков, размещенных внутри него. Блоки **Цикл** (Loop) особенно пригодятся в случае, если вы хотите повторить определенные действия много раз.

Использование циклов

Блок **Цикл** (Loop) (рис. 5.4) многократно запускает блоки, которые вы поместили в него. В зависимости от режима, который вы выбираете, он запускает эти блоки либо заданное количество раз, либо в течение определенного количества секунд, либо до тех пор, пока вы вручную не прервете

выполнение программы на модуле EV3. В последующих главах вы научитесь использовать многие из этих режимов.

В верхней части каждого блока **Цикл** (Loop) вы можете указать имя цикла для описания функций блоков, которые вы поместили в него. Вы можете вручную изменить размер блока, если это необходимо, как показано на рис. 5.4. Существует также функция **Параметр цикла** (Loop Index), о которой вы узнаете в главе 14, а пока можете не обращать на нее внимания.

*Рис. 5.4. Блок **Цикл** в режиме **Подсчет**. В данной конфигурации программа будет запускать любые блоки, размещенные в пределах этого цикла, четыре раза. В других режимах блоки, расположенные внутри цикла, могут повторяться в течение указанного количества секунд или неограниченного времени*

Чтобы поместить выбранные блоки в цикл, просто перетащите один блок или их последовательность в блок **Цикл** (Loop), как показано на рис. 5.5.

Блок Цикл в действии

Чтобы попробовать блок **Цикл** (Loop) в действии, создайте программу OneSquare, показанную на рис. 5.6. При ее запуске робот должен воспроизвести звуковой тон, проехать по квадратной траектории, воспроизвести другой звук, а затем остановиться. Если ваш робот не выполняет поворот на 90 градусов, попробуйте изменить значение в градусах во втором блоке **Рулевое управление** (Move Steering) так, как вы делали это в практикуме № 2 на с. 61

Вложенные циклы

Программа OneSquare (рис. 5.6) задает роботу EXPLOR3R квадратную траекторию движения.

Вы можете использовать еще один блок **Цикл** (Loop), зациклив движение по траектории, чтобы робот несколько раз повторил заданный маршрут. В режиме **Неограниченный** (Unlimited) робот будет продолжать движение по квадрату бесконечно. Попробуйте реализовать такое поведение с помощью программы InfiniteSquare, показанной на рис. 5.7. Чтобы это сделать, дополните программу, которую вы только что создали, добавив второй блок **Цикл** (Loop) из палитры программирования и установив режим **Неограниченный** (Unlimited). Перетащите цикл Square, созданный ранее, и второй блок **Звук** (Sound) в новый блок **Цикл** (Loop). Теперь робот будет двигаться по квадратной траектории, говоря «Пока» после завершения каждого квадрата, пока вы не завершите программу нажатием кнопки «Назад» на модуле EV3.

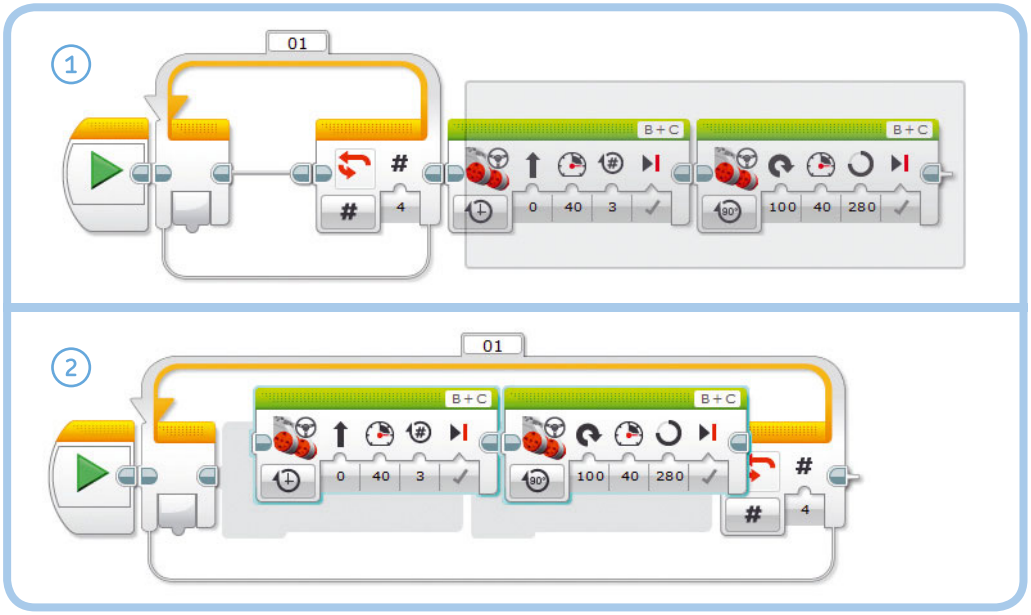


*Рис. 5.5. Чтобы разместить блоки внутри цикла, сначала поместите все необходимые блоки в область программирования (1). Затем выделите блоки, которые вы хотите поместить в цикл, и перетащите их в блок **Цикл** (Loop) (2). При этом блок **Цикл** (Loop) автоматически изменит размер, чтобы создать пространство для перетаскиваемых блоков. При перемещении блока **Цикл** (Loop) по области программирования его содержимое остается внутри него*

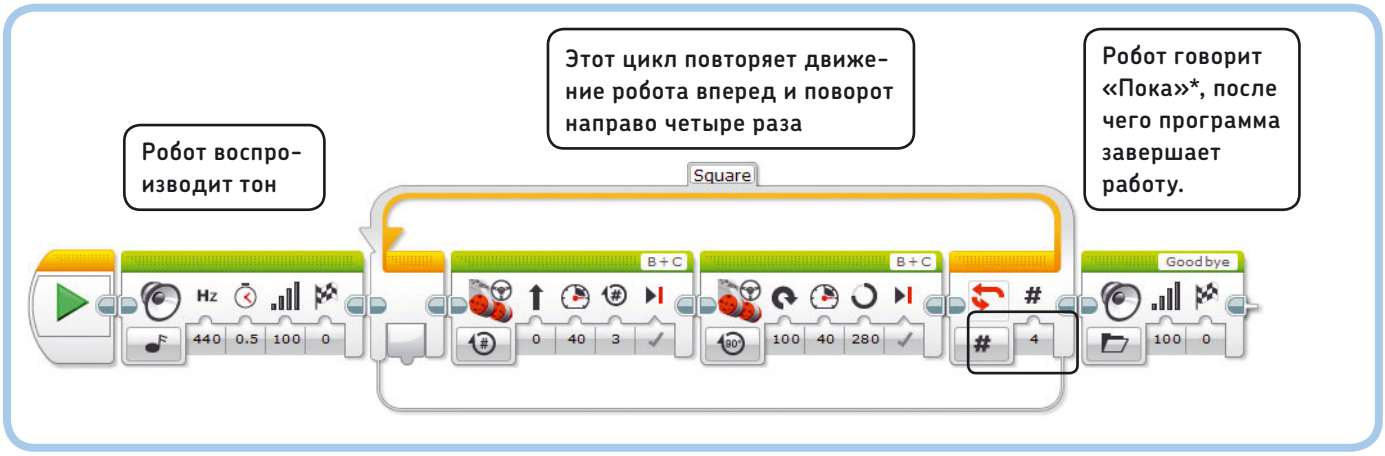


*Рис. 5.6. Программа OneSquare содержит цикл, который выполняется четыре раза. После того как два блока внутри цикла были запущены по четыре раза (результат — движение по квадратной траектории), программа переходит к следующему блоку, в данном случае **Звук** (Sound). В поле имени цикла вы можете ввести слово Square**, как показано на рисунке, чтобы описать предназначение этого цикла*

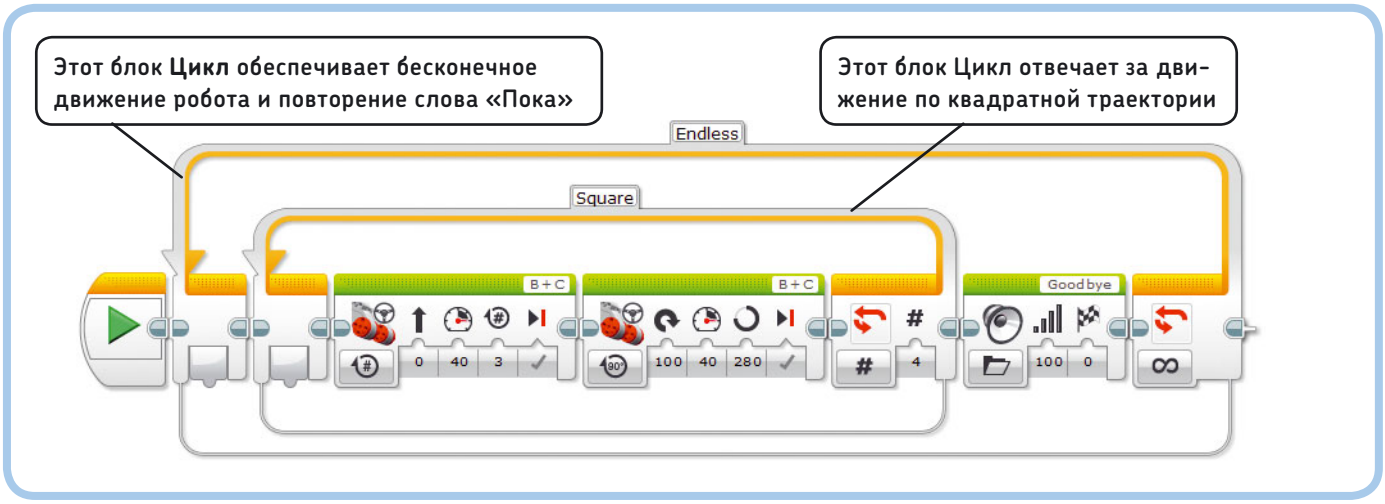


*Рис. 5.7. Программа InfiniteSquare показывает, как один блок **Цикл** вкладывается в другой. Вложенный (внутренний) цикл обеспечивает движение робота EXPLOR3R по квадратной траектории, а внешний отвечает за бесконечное повторение роботом этого перемещения и воспроизведение слова «Пока»*

ПРАКТИКУМ № 16: ОХРАНА КОМНАТЫ

Сложность: **Время:**

Разработайте программу, которая позволит роботу EXPLOR3R постоянно двигаться вперед и назад перед дверью вашей спальни, как будто охраняя ее (рис. 5.8). Используйте один блок **Цикл** (Loop) в режиме **Неограниченный** (Unlimited) и два блока **Рулевое управление** (Move Steering) — один для движения вперед и второй для разворота.

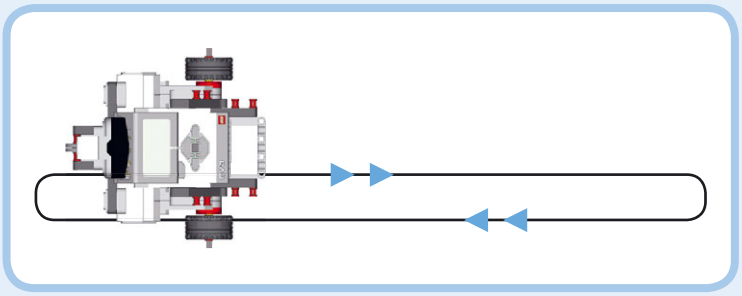


Рис. 5.8. Траектория движения робота EXPLOR3R в практикуме № 16

* Пункт **Goodbye** в группе **Связь** (Communication).
** Квадрат (*англ.*).

ПРАКТИКУМ № 17: ТРЕУГОЛЬНИК!

Сложность: **Время:**

Вы разработали программу, которая обеспечивает вашему роботу движение по квадратной траектории. Как вы могли бы изменить программу OneSquare, чтобы траектория движения робота стала треугольной? А как насчет шестиугольника? Используйте дополнительные блоки **Цикл** (Loop), чтобы повторить каждую траекторию пять раз.

СОВЕТ **В практикуме № 2 на с. 61 вы определили количество градусов, на которое должны повернуться колеса, чтобы робот сделал поворот на 180 градусов. Можете ли вы использовать это значение для расчета количества градусов, необходимых для поворота на 120 градусов, чтобы робот мог ездить по треугольной траектории?**

Создание контейнеров «Мой блок»

В дополнение к использованию предустановленных блоков вы можете создавать свои, называемые *моими блоками*. В этой книге мы назовем их *контейнерами «Мой блок»*, чтобы избежать путаницы, к тому же *мои блоки* представляют собой, по сути, контейнеры, внутри которых расположены предустановленные блоки. Каждый контейнер «Мой блок» позволяет объединить несколько стандартных программных блоков в один. Контейнеры «Мой блок» особенно удобны, если вы хотите использовать определенную последовательность блоков в вашей программе более одного раза. Например, вы можете создать контейнер с инструкциями, чтобы робот говорил: «Привет! Доброе утро!» и изменял цвет индикатора состояния модуля на красный всякий раз, когда вы используете этот контейнер. В обычной ситуации, чтобы робот все это делал, требуется пять программных блоков. Если вам нужно, чтобы эту комбинацию действий робот совершал несколько раз, вместо того чтобы каждый раз помещать все пять блоков, гораздо проще создать один контейнер «Мой блок», сочетающий в себе эти блоки, который можно повторно использовать. Кроме того, использование контейнера «Мой блок» поможет вам в организации вашей программы, поскольку на экране будет меньше блоков.

Контейнеры «Мой блок» на практике

Чтобы продемонстрировать функциональность контейнера «Мой блок», давайте напишем программу, с помощью которой EXPLOR3R будет говорить: «Привет! Доброе утро!»* — двигаться вперед, а затем говорить то же самое еще раз. Поскольку робот будет приветствовать вас два раза, вы создадите контейнер «Мой блок» с именем **Talk**, чтобы упростить повторение этого действия, как показано на рис. 5.9–5.11. После того как вы создадите свой контейнер «Мой блок», вы сможете помещать его в программу всякий раз, когда требуется, чтобы робот EXPLOR3R говорил: «Привет! Доброе утро!»

1. Создайте новую программу под названием MyBlockDemo, в области программирования которой разместите и настроите пять блоков для создания приветствия робота (рис. 5.9). Затем выделите эти пять блоков (контур выделенных блоков окрашен в синий цвет) и выберите команду меню **Инструменты** ▸ **Конструктор Моего Блока** (Tools ▸ My Block Builder).
2. Появится окно **Конструктор Моего Блока** (My Block Builder), показанное на рис. 5.10. Присвойте контейнеру «Мой блок» имя, например **Talk**, в поле **Имя** (Name). Используйте поле **Описание** (Description), чтобы описать ваш контейнер. Это поможет вспомнить его предназначение, когда вы захотите использовать его повторно. И, наконец, выберите значок, например в виде динамика, чтобы создать ассоциацию, что этот контейнер используется для воспроизведения звуков. Нажмите кнопку **Завершить** (Finish).
3. После создания контейнер «Мой блок» появится в области программирования, заменив собой блоки, которые там находились изначально, как показано на рис. 5.11. Поменяйте блоки местами, если возникнет такая необходимость.

Использование контейнеров «Мой блок» в программах

Теперь, когда контейнер «Мой блок» готов, вы можете найти его на светло-голубой вкладке палитры программирования, как показано на рис. 5.12. Добавьте блок **Рулевое управление** (Move Steering) и еще одну копию контейнера «Мой блок», чтобы завершить программу MyBlockDemo. При запуске программы робот должен произнести «Привет! Доброе утро!», проехать вперед, а затем поприветствовать вас снова. Во время воспроизведения звука индикатор состояния должен быть красным, а во время движения робота — зеленым.

Обратите внимание, как использование контейнеров «Мой блок» упрощает программу для понимания или объяснения другу. По этой причине иногда полезно заключить блоки вашей программы в несколько контейнеров «Мой блок», даже если вы используете их всего один раз.

Изменение контейнеров «Мой блок»

После того как вы создали контейнер «Мой блок», вы можете поменять блоки, которые в нем находятся. Для этого дважды

* Пункты **Hello**, **Good**, **Morning** в группе **Связь** (Communication).

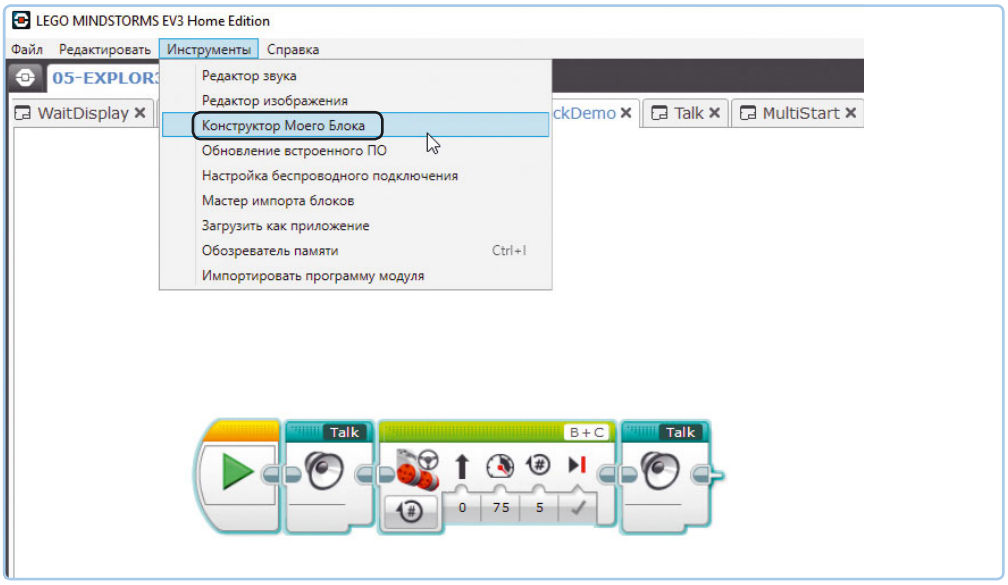


*Рис. 5.9. Поместите блоки в область программирования и настройте их, как показано на рисунке. Затем заключите в рамку выделения все блоки, кроме оранжевого блока **Начало**, и выберите команду меню **Инструменты** ▸ **Конструктор Моего Блока***

щелкните мышью по контейнеру «Мой блок» в области программирования, чтобы показать его содержимое, а затем измените его, как если бы это была обычная программа. Когда вы закончите, щелкните мышью по значку × на вкладке с именем контейнера, чтобы сохранить изменения и вернуться в программу, в которой используется данный контейнер «Мой блок».

Чтобы переименовать контейнер «Мой блок», дважды щелкните мышью по его имени на вкладке и измените текст.

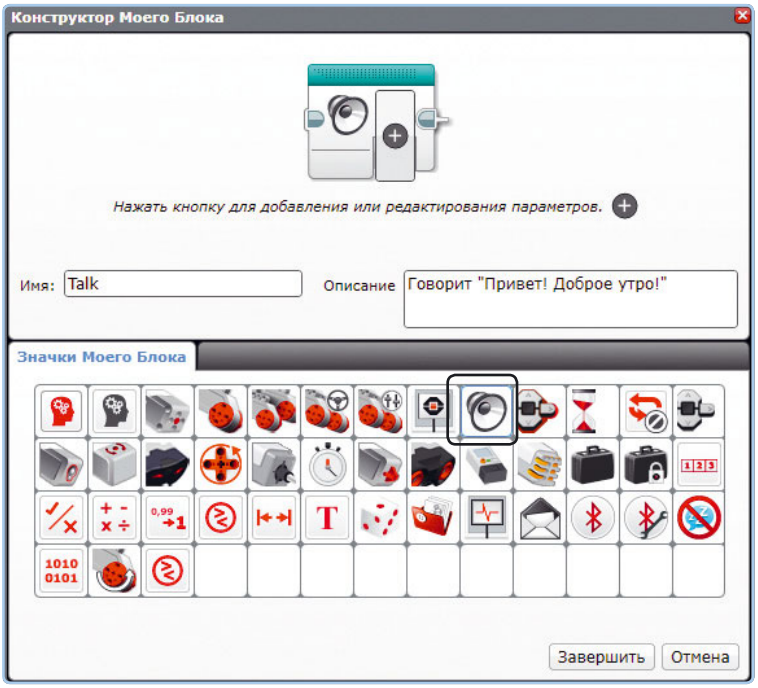


*Рис. 5.10. Окно **Конструктор Моего Блока**. Укажите имя, описание и выберите значок для вашего контейнера «Мой блок». Затем нажмите кнопку **Завершить**, чтобы завершить создание контейнера*

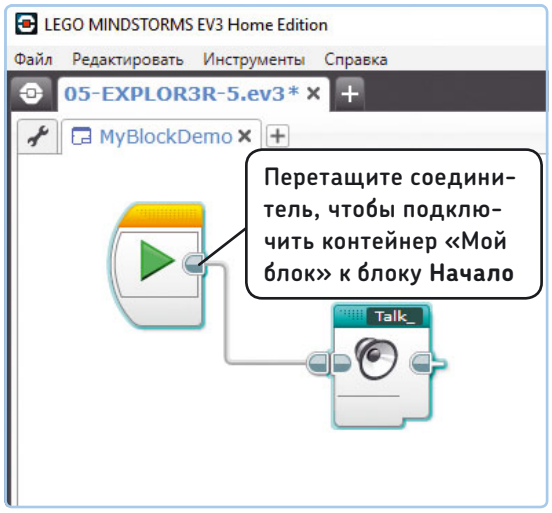


*Рис. 5.11. Законченный контейнер «Мой блок» в области программирования. Если созданный контейнер не примыкает к блоку **Начало**, нажав и удерживая кнопку мыши на правом выходном разъеме блока **Начало**, перетащите соединитель к входному разъему контейнера*

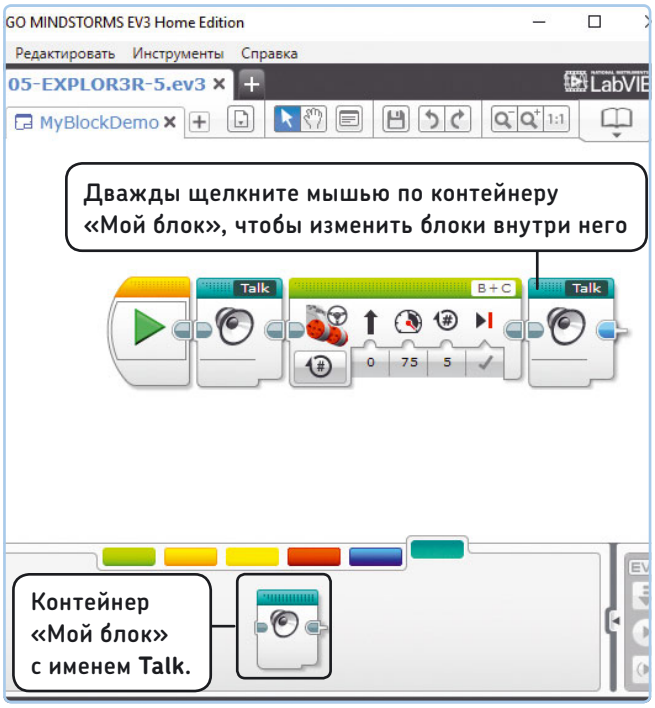


Рис. 5.12. Готовая программа MyBlockDemo

Таким же образом вы переименовываете обычные программы, как показано на рис. 3.11 в главе 3.

Управление контейнерами «Мой блок» в проектах

Вы можете использовать созданные контейнеры «Мой блок» в любой программе в рамках того же проекта. Например, контейнер **Talk** можно использовать в программах в файле проекта EXPLOR3R-5. Но может возникнуть ситуация, когда потребуется использовать контейнеры «Мой блок» в других проектах.

Чтобы скопировать в другой проект контейнер «Мой блок», который вы только что сделали (как показано на рис. 5.13), перейдите на страницу **Свойства проекта** (Project Properties) (1), выберите пункт **Talk.EV3p** на вкладке **Мои блоки** (My Blocks) (**2**), а затем нажмите кнопку **Копировать** (Copy) (**3**). Теперь, не закрывая текущий проект, откройте созданный ранее проект: EXPLOR3R-4 (**4**). Перейдите на вкладку **Мои блоки** (My Blocks) этого проекта на странице **Свойства проекта** (Project Properties) (**5**) и нажмите кнопку **Вставить** (Paste) (**6**). Теперь вы можете использовать контейнер **Talk** в проекте EXPLOR3R-4.

Вместо копирования вы можете нажать кнопку **Экспорт** (Export), которая позволяет сохранить контейнер «Мой блок» в файл. Сохраненный таким образом файл вы можете передать другу по электронной почте, чтобы он добавил этот файл в свой проект, используя кнопку **Импорт** (Import). Чтобы удалить контейнер «Мой блок» из проекта, выберите его и нажмите кнопку **Удалить** (Delete).

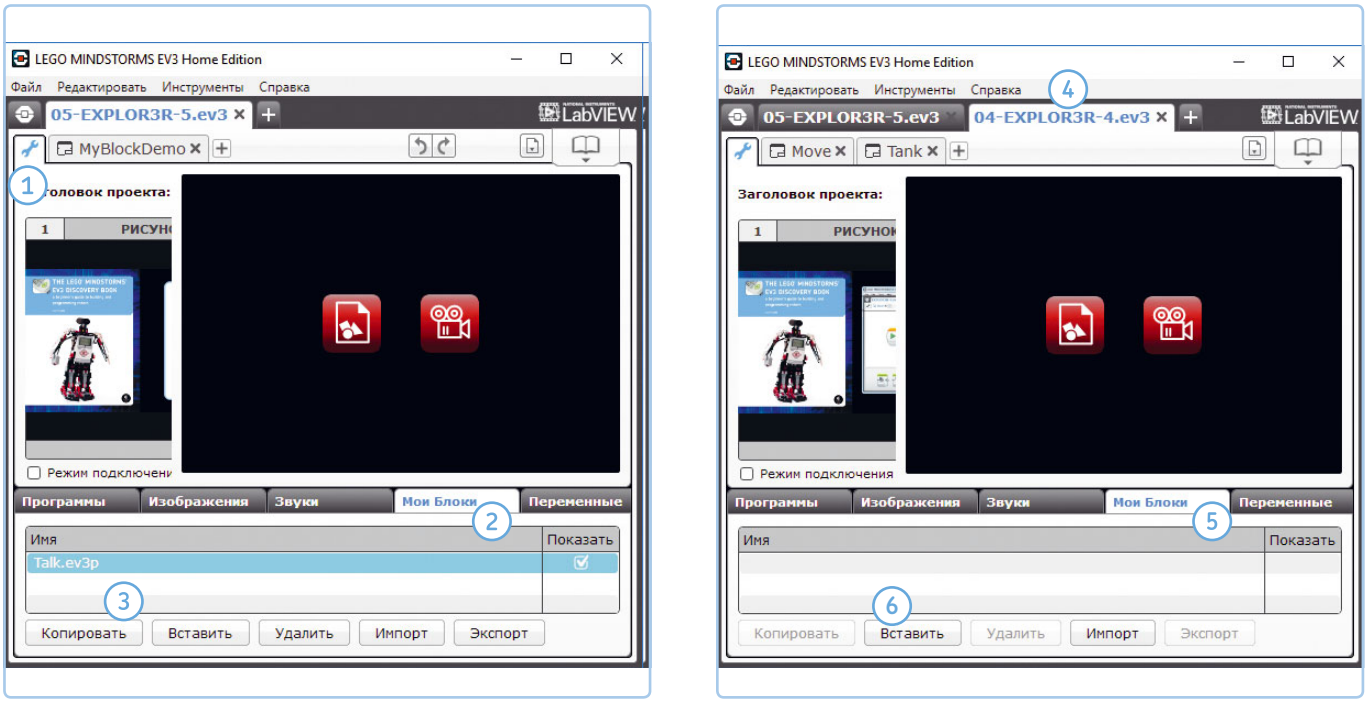


Рис. 5.13. Копирование контейнера «Мой блок» из проекта EXPLOR3R-5 в проект EXPLOR3R-4

ПРАКТИКУМ № 18: МОЙ КВАДРАТ!

Сложность: **Время:**

Откройте программу OneSquare, показанную на рис. 5.6, и поместите блок **Цикл** (Loop) и его содержимое в контейнер «Мой блок», присвоив ему имя **MySquare**. Теперь вы можете использовать этот контейнер всякий раз, когда захотите, чтобы ваш робот ездил по квадратной траектории.

ПРАКТИКУМ № 19: МОЯ МЕЛОДИЯ!

Сложность: **Время:**

Помните мелодию, которую вы сделали с помощью блоков **Звук** (Sound) в практикуме № 6 на с. 64? Преобразуйте эту последовательность блоков **Звук** (Sound) в контейнер «Мой блок», и вы сможете в любое время использовать вашу любимую мелодию в своих программах.

ПРИМЕЧАНИЕ **Вы можете использовать тот же метод для обмена между проектами другими файлами, такими как отдельные программы, пользовательские звуки и изображения.**

Многозадачность

Все блоки, которые вы использовали до сих пор, выполняются по одному в том порядке, в котором они расположены в области программирования. Тем не менее модуль EV3 поддерживает многозадачность, позволяя выполнять инструкции нескольких блоков одновременно. Для этого применяются либо несколько блоков **Начало** (Start) или разделенные соединители, называемые *шинами последовательности*. Вы увидите, что эти методы очень похожи.

Использование нескольких блоков Начало

Самый простой способ заставить работать параллельно (одновременно) две последовательности блоков заключается в добавлении второго блока **Начало** (Start), как показано на рис. 5.14. После нажатия кнопки **Загрузить и запустить** (Download and Run) обе последовательности начинают работать одновременно. Программа заканчивается, когда обе последовательности блоков завершают работу. Чтобы проверить работу одной последовательности без участия второй, нажмите кнопку в виде зеленой стрелки на соответствующем блоке **Начало** (Start).

Когда вы запустите эту программу, робот начнет двигаться и воспроизводить звук.

Параллельное соединение блоков

Еще один способ реализовать многозадачность заключается в разделении соединителей (шин последовательности), как

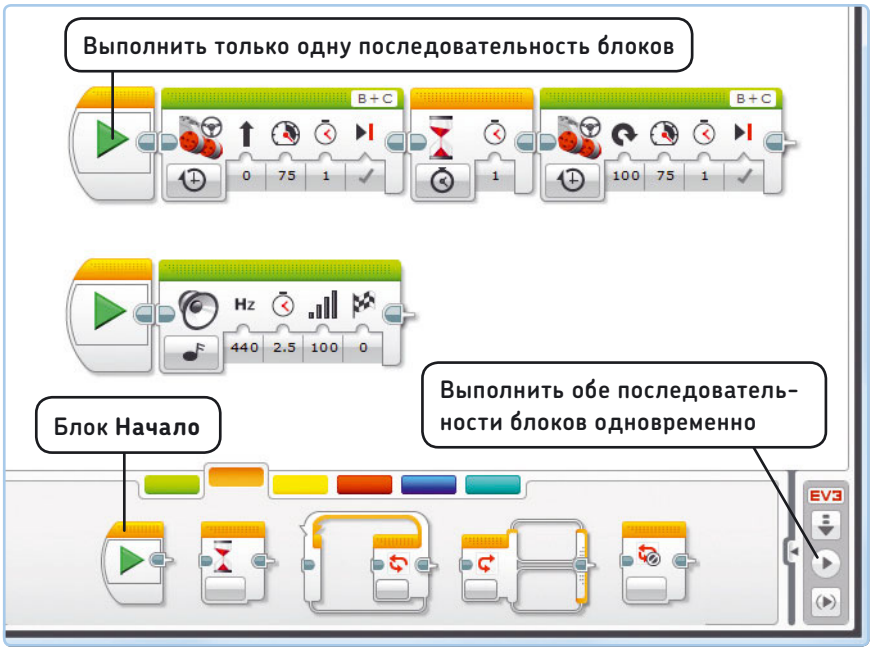


*Рис. 5.14. Реализация многозадачности с помощью двух блоков **Начало** в программе MultiStart. Блок **Начало** доступен на оранжевой вкладке вместе с другими блоками управления операторами*

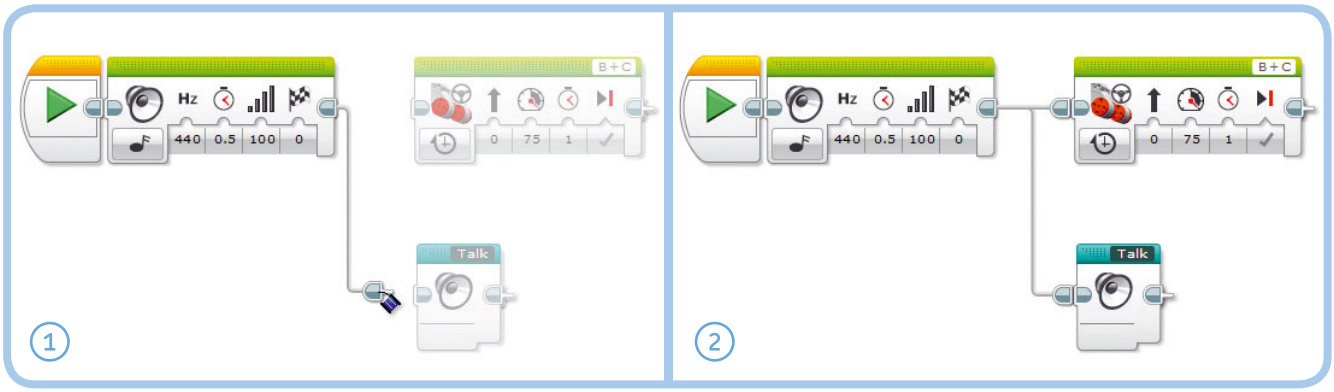


*Рис. 5.15. Реализация многозадачности с помощью соединителей (шин последовательности) в программе MultiSequence. Чтобы достичь этого результата, сначала поместите все необходимые блоки в область программирования, а затем подключите соединителями каждую последовательность блоков. Это возможно, даже если блоки **Звук** и **Рулевое управление** состыкованы. Блоки будут разделены автоматически, когда вы подключите контейнер **Talk***

показано на рис. 5.15. Такой вариант может пригодиться, если вам нужны две параллельные последовательности блоков, которые включаются не с начала программы. В программе MultiSequence показано, как робот воспроизводит мелодию, а затем два действия происходят одновременно: робот едет вперед и произносит «Привет! Доброе утро!» с помощью контейнера **Talk**, который вы создали ранее.

Избегание конфликтов ресурсов

Точно так же, как вы не можете одновременно идти и вперед, и назад, вы не можете запрограммировать робота в одной последовательности двигаться в одном направлении, а в другой — в противоположном направлении. В этом случае возникает *конфликт ресурсов*, когда две последовательности блоков пытаются управлять одним мотором или датчиком одновременно.

К сожалению, программное обеспечение EV3 не сообщит вам, возникает ли конфликт ресурсов в вашей программе; вероятнее всего, программа продолжит работать, но результат будет непредсказуемым. Например, робот не сможет двигаться в заданном направлении. Чтобы избежать этой проблемы, не используйте один и тот же мотор или датчик более чем в одной последовательности.

По возможности старайтесь избегать многозадачности, потому что конфликты ресурсов могут возникнуть неожиданно. Иногда вы можете выполнить ту же задачу, используя только одну последовательность блоков. Например, вместо перемещения и воспроизведения звуков с помощью двух параллельных последовательностей, вы можете выполнить оба действия из одной последовательности, как вы увидите в практикуме № 21.

ПРАКТИКУМ № 20: ДА ЗДРАВСТВУЕТ МНОГОЗАДАЧНОСТЬ!

Сложность: **Время:**

Запрограммируйте робота на бесконечное движение по квадратной траектории и одновременное повторение слов «LEGO, MINDSTORMS, EV3».

Дальнейшее изучение

Завершив изучение первой части этой книги, вы получили прочную основу из нескольких важнейших методов программирования. В этой главе вы узнали, как использовать блоки **Ожидание** (Wait) и **Цикл** (Loop), как создавать и редактировать контейнеры «Мой блок» и как сделать вашего робота многозадачным.

В следующей части книги вы будете создавать роботов, которые могут взаимодействовать с окружающей средой с помощью датчиков. Но прежде чем заняться этим, немного потренируйтесь с навыками, которые вы приобрели в этой главе. Для этого решите следующие практические задачи.

ПРАКТИКУМ № 21: ОДНОЗАДАЧНОСТЬ!

Сложность: **Время:**

Программа MultiStart — простой пример, который иллюстрирует принцип многозадачности. Но не всегда необходимо использовать несколько последовательностей для одновременного движения и воспроизведения звуков. Можете ли вы создать новую программу, которая делает то же самое, что и программа MultiStart, только с одной последовательностью блоков?

Чтобы усложнить задачу, сделайте то же самое для программы MultiSequence.

СОВЕТ Как это сделать, вы узнали в главе 4. А за что отвечает параметр **Тип воспроизведения** (Play Type) на блоке **Звук** (Sound)?

СДЕЛАЙ САМ № 3: ГОСПОДИН EXPLOR3R!

Сборка: **Программирование:**

Добавьте средний мотор, чтобы создать машущую руку для EXPLOR3R. Используйте дополнительные детали LEGO для дальнейшего украшения робота и превращения его в господина EXPLOR3R. Запрограммируйте его постоянно махать рукой и одновременно произносить приветствие, например «Доброе утро!».

ПРАКТИКУМ № 22: СЛОЖНЫЕ ФИГУРЫ!

Сложность: **Время:**

Разработайте программу, согласно которой робот EXPLOR3R будет двигаться по шаблону, показанному на рис. 5.16, воспроизводя различные звуки.

СОВЕТ Если вы посмотрите внимательно, то увидите, что предлагаемый путь можно разделить на четыре одинаковые части, поэтому можно настроить последовательность блоков **Рулевое управление** (Move Steering) только для одной из этих частей. Затем вы можете поместить эти блоки в цикл, настроенный на четырехразовое повторение.

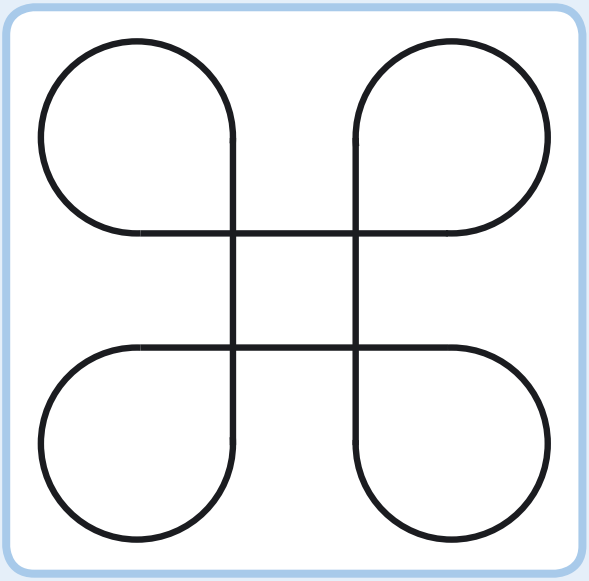


Рис. 5.16. Траектория движения для практикума № 22

ЧАСТЬ II

Программирование роботов с датчиками

6

Предназначение датчиков

Набор LEGO MINDSTORMS EV3 включает в себя три типа датчиков: *касания*, *цвета* и *инфракрасный*. Вы можете использовать эти датчики, чтобы ваш робот реагировал на окружающую его среду. Например, вы можете запрограммировать робота, чтобы он воспроизводил звук, когда видит вас, избегал препятствий во время перемещения или двигался по цветным линиям. В этой части книги вы научитесь создавать роботов, которые используют эти датчики.

Чтобы узнать, как работать с датчиками, мы дополним робота EXPLOR3R, добавив бампер, который определяет препятствия с помощью датчика касания, как показано на рис. 6.1. После того как вы научитесь создавать программы, которые используют датчик касания, в последующих главах вы узнаете, как использовать и другие датчики.

Рис. 6.1. Робот EXPLOR3R использует бампер с датчиком касания для определения объектов, в сторону которых он движется

Что такое датчики?

Роботы LEGO MINDSTORMS не способны видеть или чувствовать, как люди, но, если снабдить их *датчиками*, они смогут собирать и передавать информацию об окружающей среде. При разработке программ, которые интерпретируют информацию, получаемую с датчиков, вы можете сделать так, чтобы ваш робот казался разумным, запрограммировав его на определенные реакции. Например, можно создать программу, благодаря которой робот будет говорить «синий», когда датчик определяет синюю бумагу.

Общее представление о датчиках в наборе MINDSTORMS EV3

Ваш набор MINDSTORMS EV3 содержит три датчика, которые можно прикрепить к роботу (рис. 6.2), а также некоторые встроенные датчики. *Датчик касания* определяет, нажата ли красная кнопка на датчике или отпущена. *Датчик цвета* определяет цвет поверхности и яркость источника света, о чем вы подробнее узнаете в главе 7. *Инфракрасный датчик* (рассматривается в главе 8) измеряет приблизительное расстояние до соседнего объекта, а также принимает сигналы от удаленного инфракрасного маяка.

Кроме того, каждый мотор в комплекте EV3 имеет встроенный *датчик вращения мотора* для определения позиции мотора и скорости, а модуль EV3 может определить, какие из его кнопок нажаты (см. главу 9).

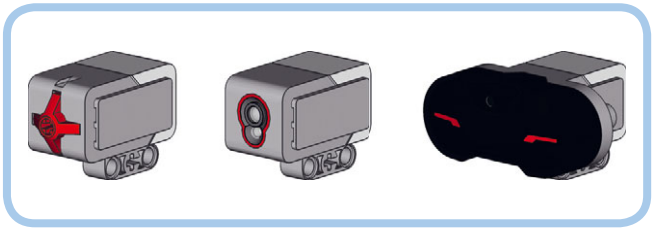


Рис. 6.2. Набор EV3 комплектуется датчиком касания (слева), датчиком цвета (средний) и инфракрасным датчиком (справа)

Обзор датчика касания

Датчик касания позволяет роботу «чувствовать», определяя, нажата или отпущена в настоящий момент красная кнопка на датчике, как показано на рис. 6.3. Модуль EV3 получает эту информацию от датчика и может использовать ее в программах. Например, вы можете запрограммировать вашего робота говорить «Привет!» всякий раз, когда вы нажимаете кнопку датчика касания.

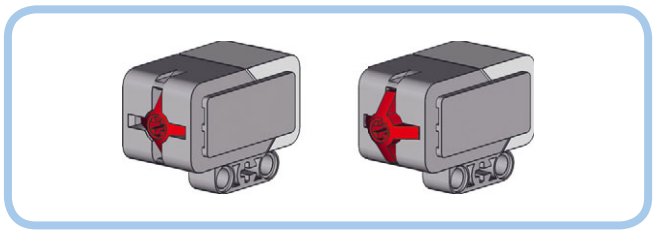
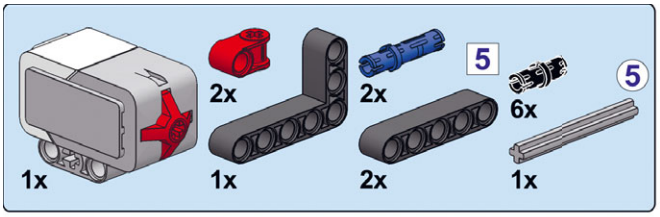


Рис. 6.3. Датчик касания определяет, нажата ли красная кнопка (слева) или отпущена (справа)

Несмотря на свою простоту, датчик касания полезен для многих приложений. Например, роботы могут использовать его для определения препятствий перед ними. С помощью датчика касания вы можете узнать, что какой-то механизм вашего робота достиг заданного положения. В главе 18, к примеру, вы будете использовать датчик касания для определения, поднята ли рука робота во время его движения.

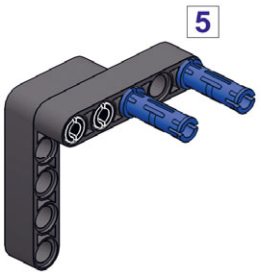
Сборка бампера с датчиком касания

Если вы соберете бампер и прикрепите к нему датчик касания, то каждый раз, когда EXPLOR3R наедет на объект, датчик будет нажат. Программа в модуле EV3 может использовать эту информацию, чтобы направить робота в другую сторону. Соберите бампер и прикрепите его к роботу, как показано на следующих страницах. Обязательно подключите датчик касания к входному порту 1 с помощью короткого кабеля.

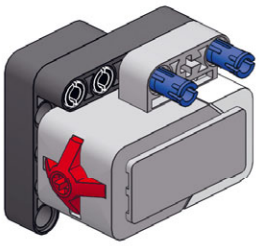


1

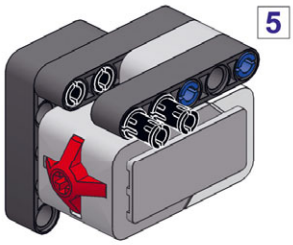
2



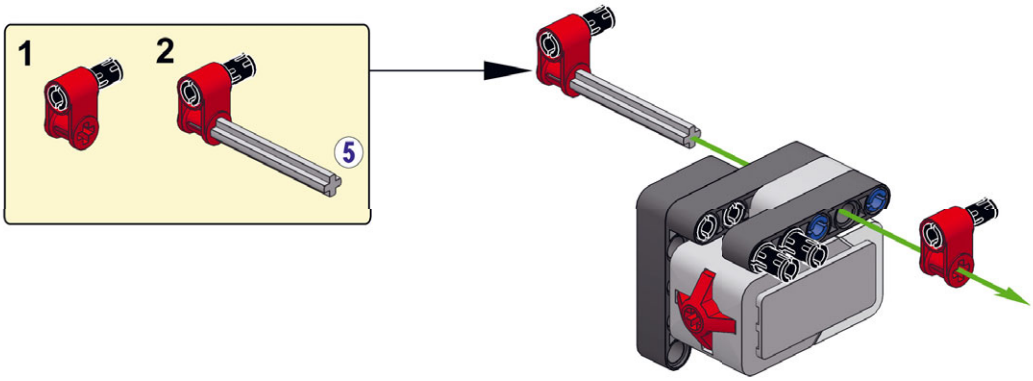
3

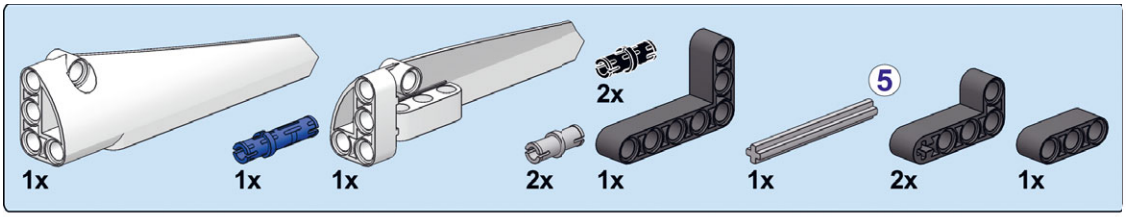


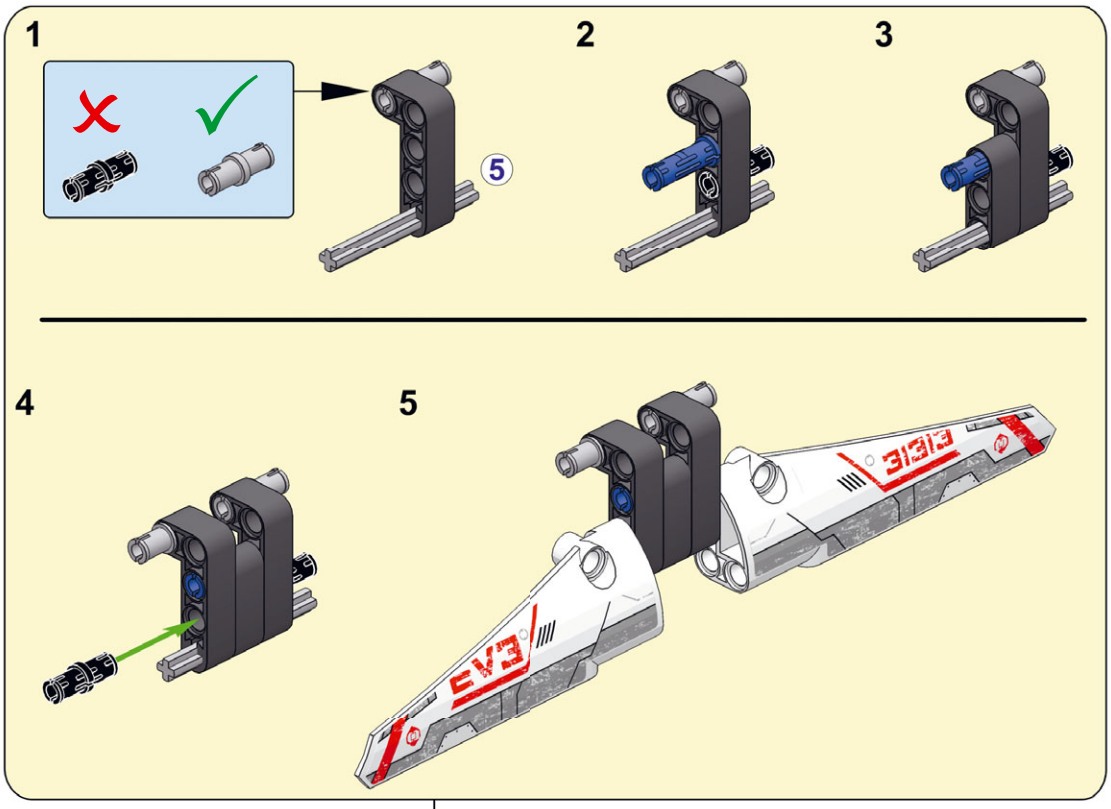
4

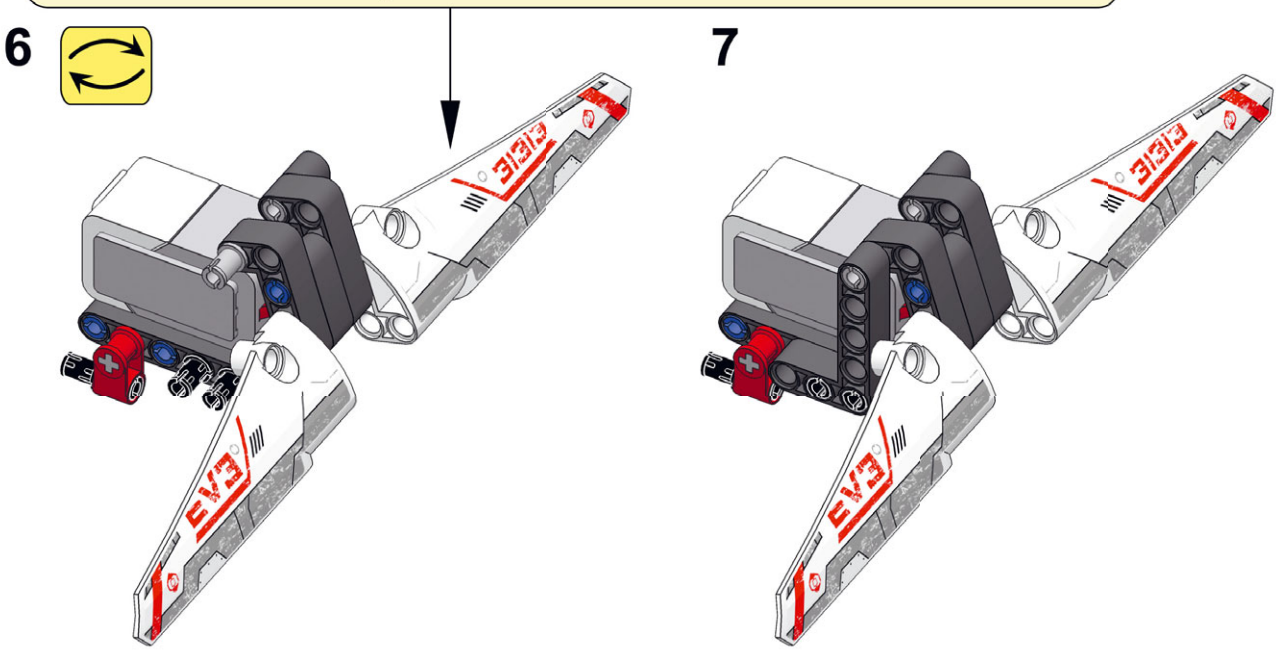


5

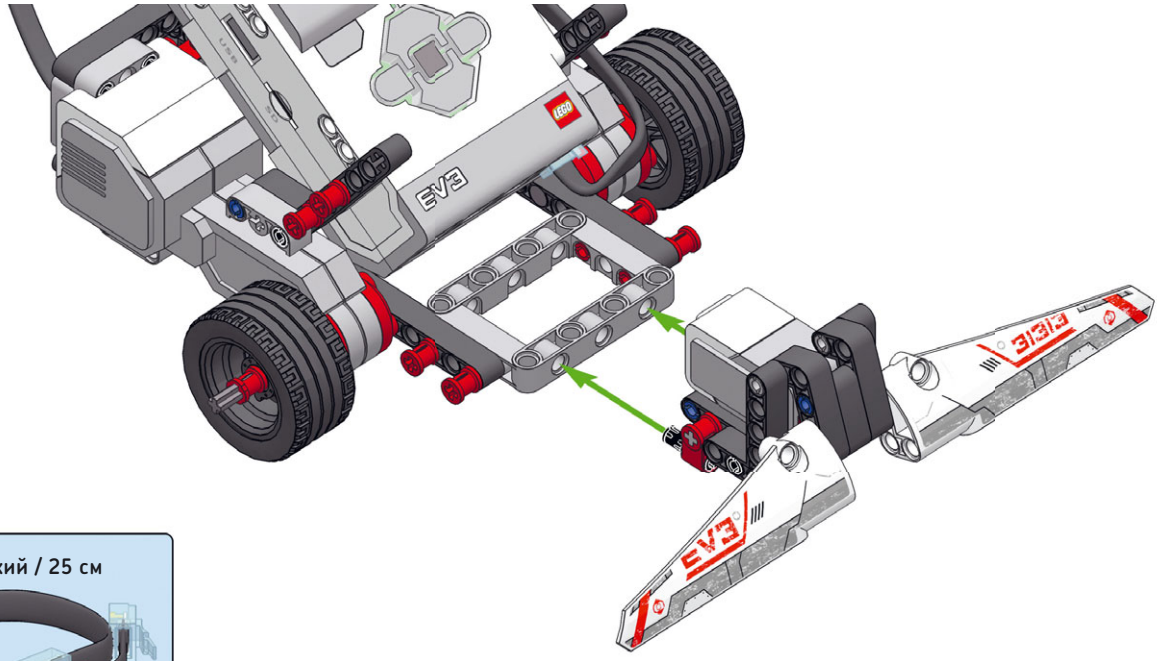


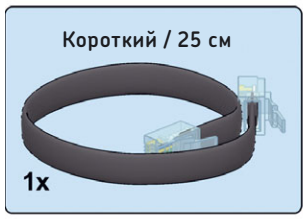

1x
1x
1x
2x
2x
1x
5
1x
2x
1x


1
5
2
3
4
5

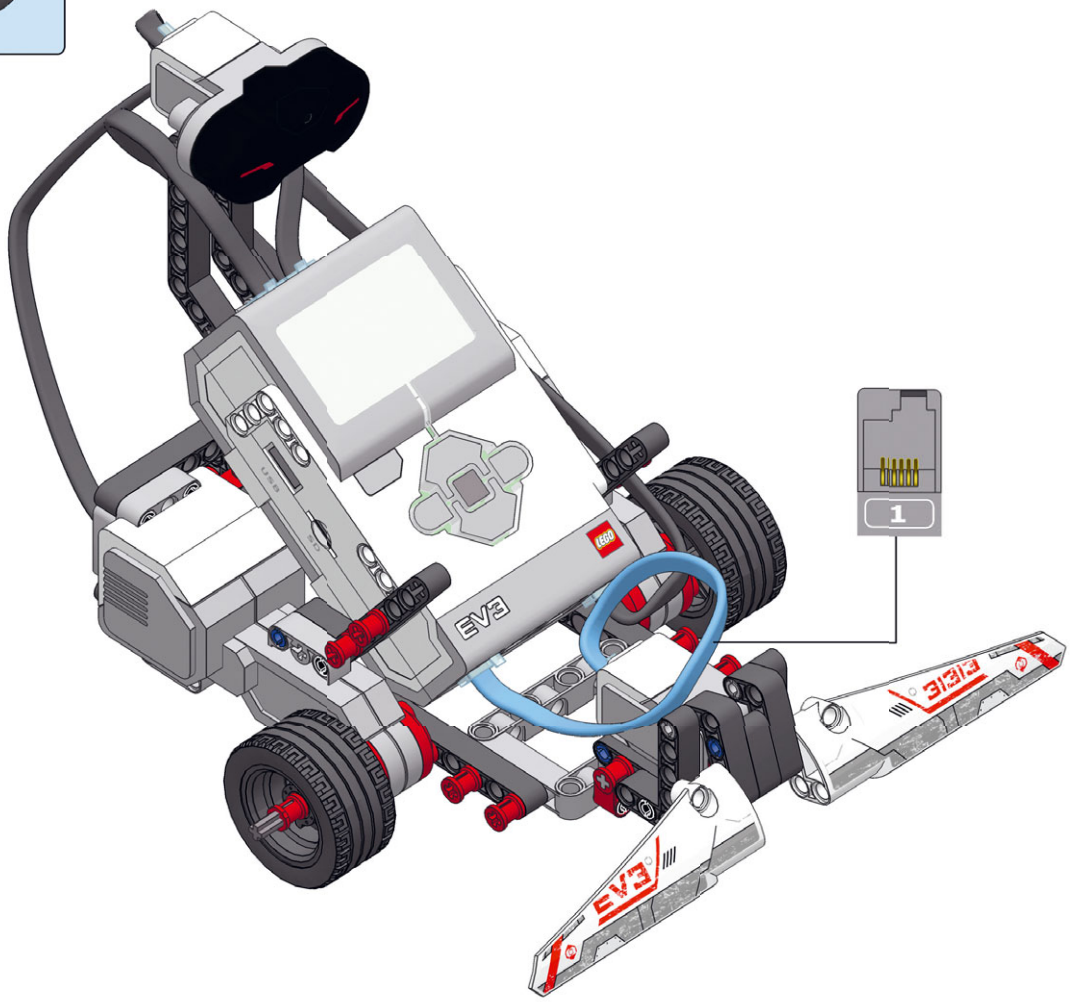

6
7

8





9



Просмотр значений, полученных датчиками

Вы можете просмотреть данные, собранные каждым датчиком, запустив приложение **Port View** на вкладке **Приложения модуля** (Brick Apps) модуля EV3, как показано на рис. 6.4. Для датчика касания значение 1 означает, что кнопка нажата, а 0 указывает на то, что она отпущена.

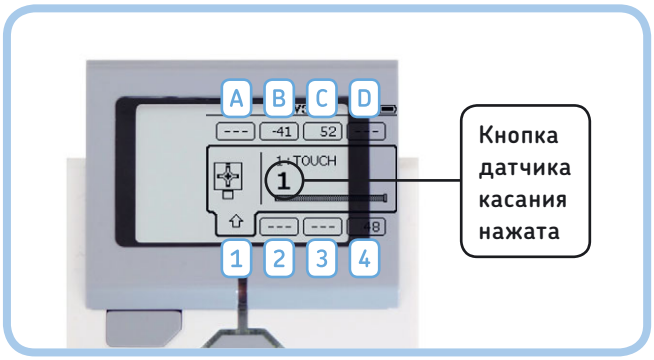


*Рис. 6.4. Приложение **Port View** на вкладке **Приложения модуля**. Модуль EV3 автоматически определяет, какие датчики подключены к EV3, и отображает на экране информацию, получаемую от них. Используйте кнопки «Влево», «Вправо», «Вверх» и «Вниз», чтобы увидеть более подробную информацию о каждом датчике*

Вы можете использовать кнопки на корпусе модуля EV3, чтобы перейти к данным с других датчиков. В правом нижнем углу экрана (порт 4) вы можете увидеть значение расстояния, которое измеряет инфракрасный датчик (48% в данном примере). Два значения в верхней части экрана (–41 и 52) указывают позиции моторов робота, подсоединенных к портам B и C.

Некоторые датчики могут выполнять замеры не одного типа, а больше. Чтобы увидеть другие замеры, к примеру инфракрасного датчика, перейдите к порту 4, нажмите центральную кнопку и выберите режим датчика. По мере чтения книги вы узнаете больше о значении каждого параметра.

Если ваш робот подключен к компьютеру, вы можете просмотреть данные, получаемые от датчиков, на странице

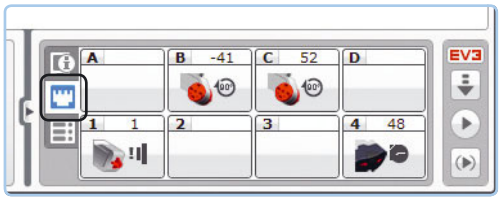


Рис. 6.5. Вы можете просмотреть данные, получаемые от датчиков, в программе EV3. Если значения не обновляются непрерывно, загрузите программу в модуль EV3 робота, чтобы обновить подключение. Щелкните мышью по одному из датчиков, чтобы просмотреть соответствующие данные измерений

аппаратных средств в программном обеспечении EV3, как показано на рис. 6.5. Используйте тот метод, который вам более удобен.

Использование датчиков в программах

Теперь давайте посмотрим, как использовать данные, получаемые от датчиков, в программах. Давайте попробуем использовать датчик касания в программе, с помощью которой робот воспроизводит звук, когда кнопка датчика касания нажата.

Использовать датчики в вашей программе позволяют несколько программных блоков, в том числе **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch). В этой главе вы узнаете, как каждый из них работает с датчиком касания. Те же самые принципы применимы и к другим датчикам из набора MINDSTORMS EV3.

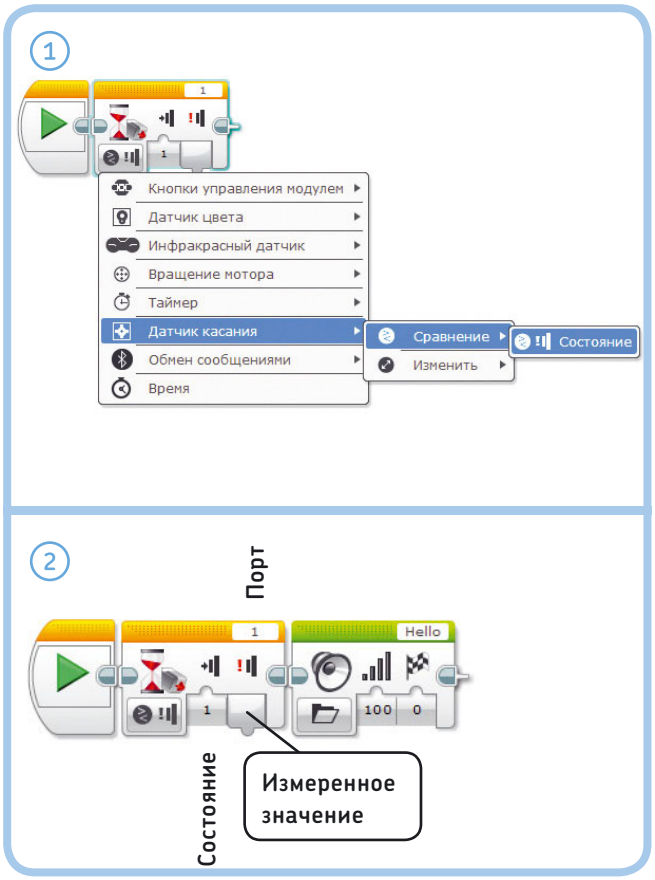


Рис. 6.6. Благодаря программе WaitForTouch робот воспроизводит звук, когда нажата кнопка датчика касания

Датчики и блок Ожидание

Ранее вы использовали блок **Ожидание** (Wait) для приостановки выполнения программы на некоторое время (скажем, пять секунд). Но вы также можете использовать этот блок для приостановки выполнения программы до тех пор, пока не сработает датчик. Например, вы можете настроить блок **Ожидание** (Wait) на приостановку программы до того момента, пока кнопка датчика касания не будет нажата, выбрав режим **Датчик касания** (Touch Sensor), как показано на рис. 6.6.

После выбора этого режима вы должны выбрать подрежим: **Сравнение** (Compare) и **Изменить** (Change). В режиме **Сравнение** (Compare) следует указать параметр **Состояние** (State): должна ли программа ждать, пока кнопка датчика не будет отпущена (0), нажата (1) или щелкнута (2). Если вы выбрали щелчок, программа будет ждать, пока кнопка датчика не будет нажата и сразу же отпущена.

В режиме **Изменить** (Change) программа отслеживает изменения состояния датчика: если кнопка датчика нажата, когда блок начинает работать, программа ожидает, пока кнопка не будет отпущена. И наоборот, если изначально кнопка датчика отпущена, программа приостанавливается до тех пор, пока кнопка датчика не будет нажата.

Параметр **Порт** (Port) позволяет указать, к какому входному порту подключен ваш датчик (в данном случае порт 1). И, наконец, выход **Измеренное значение** (Measured Value) позволяет использовать последний замер датчика в дальнейшем в вашей программе (мы вернемся к этому вопросу в части V).

Использование датчиков с блоком Ожидание

Создайте новый проект под названием EXPLOR3R-Touch с программой WaitForTouch, как показано на рис. 6.6. Для блока **Ожидание** (Wait) выберите режим **Датчик касания** ▸ **Сравнение** ▸ **Состояние** (Touch Sensor ▸ Compare ▸ State). При запуске программы поначалу ничего не будет происходить, но при нажатии кнопки датчика касания (нажатия на бампер) робот должен сказать: «Привет!»

Теперь нажмите кнопку датчика касания и запустите программу снова. Звук воспроизводится сразу, потому что блоку **Ожидание** (Wait) нечего ждать — кнопка датчика уже нажата.

Обход препятствий с помощью датчика касания

Теперь, когда вы познакомились с датчиком касания и блоком **Ожидание** (Wait), вы можете сделать свои программы более интересными. Следующая программа, TouchAvoid, задает движение по комнате и разворот, когда робот EXPLOR3R столкнется с чем-либо, например стеной или стулом. Вы можете увидеть схему программы на рис. 6.7.

Каждое действие в этой схеме вы можете выполнить с помощью одного программного блока. Вы будете использовать блок **Рулевое управление** (Move Steering) в режиме **Включить** (On) для включения моторов, а затем блок **Ожидание** (Wait), чтобы подождать, пока кнопка датчика касания не будет нажата. Обратите внимание, что, когда

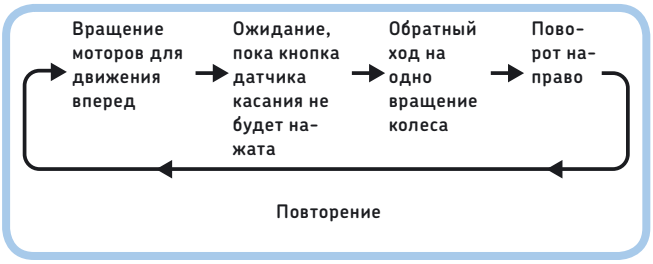


Рис. 6.7. Схема выполнения программы TouchAvoid. После поворота робота направо программа возвращается к началу и повторяется

ПРАКТИКУМ № 23: ПРИВЕТ И ПОКА!

Сложность: **Время:**

Можете ли вы создать программу, с помощью которой робот будет произносить «Привет!», когда вы нажимаете на бампер, а затем «Пока!», когда вы отпускаете бампер?

СОВЕТ **Добавьте еще одну пару блоков Ожидание (Wait) и Звук (Sound) в программу WaitForTouch (рис. 6.6). Первый блок Ожидание (Wait) должен ждать нажатия, а второй — отпускания кнопки датчика. Где вы разместите новые блоки?**

ПРАКТИКУМ № 24: ИЗБЕГАЙТЕ ПРЕПЯТСТВИЙ И ПЛОХОГО НАСТРОЕНИЯ!

Сложность: **Время:**

Дополните программу TouchAvoid, сделав так, чтобы на экране модуля EV3 отображалось счастливое лицо во время движения робота вперед и грустное лицо, когда он едет назад и поворачивает.

СОВЕТ **Поместите два блока Экран (Display) в блок Цикл (Loop).**

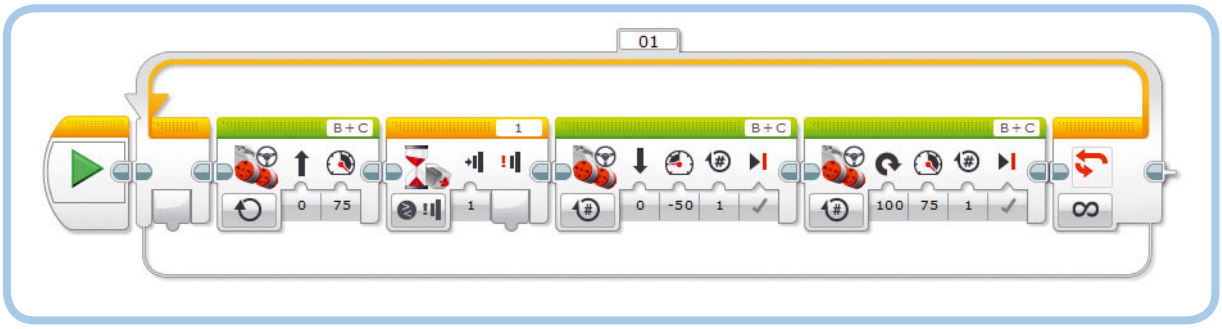


*Рис. 6.8. Программа TouchAvoid. Блок **Ожидание** настроен на режим **Датчик касания** ▸ **Сравнение** ▸ **Состояние***

ПРАКТИКУМ № 25: ДАВАЙТЕ ПОНАЖИМАЕМ!

Сложность: **Время:**

Можете ли вы сделать так, чтобы робот EXPLOR3R двигался в обратном направлении до тех пор, пока бампер не будет нажат, и можете ли вы сделать, чтобы робот остановился, когда вы отпустите бампер? Такое поведение должно продолжаться до тех пор, пока вы не завершите программу вручную. Проверьте свою программу, удерживая бампер нажатым. Вам может показаться, что это вы толкаете робота назад, но на самом деле всю тяжелую работу делает он сам.

СОВЕТ **Вам понадобится блок Цикл (Loop), два блока Ожидание (Wait), а также два блока Рулевое управление (Move Steering), один в режиме Включить (On), а другой в режиме Выключить (Off).**

Использование режима Изменить

До сих пор мы использовали блок **Ожидание** (Wait) в режиме **Сравнение** (Compare), чтобы сделать паузу в выполнении программы до тех пор, пока кнопка датчика касания не займет выбранное вами положение (нажата или отпущена). Теперь мы создадим программу с блоком **Ожидание** (Wait) в режиме **Изменить** (Change), которая будет приостанавливать программу до момента изменения состояния кнопки датчика (или из состояния отпущена к нажата, или от нажата к отпущена).

Соберите и запустите программу WaitForChange, как показано на рис. 6.9.

Если бампер отпущен, когда запускается программа, робот должен ехать, пока не достигнет какого-либо объекта, а затем остановиться. Если при запуске программы бампер уже нажат, робот должен пытаться идти вперед, пока датчик не будет отпущен, а затем остановиться.

Для большинства программ рекомендуется использовать режим **Сравнение** (Compare), так как с ним поведение программа ожидает нажатия, робот продолжает двигаться вперед.

После того как датчик срабатывает, вы используете блок **Рулевое управление** (Move Steering), чтобы отъехать назад, а затем другой такой же блок, чтобы развернуться; оба блока должны находиться в режиме **Включить на количество оборотов** (On for Rotations). После того как робот развернется, программа возвращается к началу, поэтому вы должны поместить описанные четыре блока внутрь блока **Цикл** (Loop), работающего в режиме **Неограниченный** (Unlimited). Теперь создайте программу, как показано на рис. 6.8.

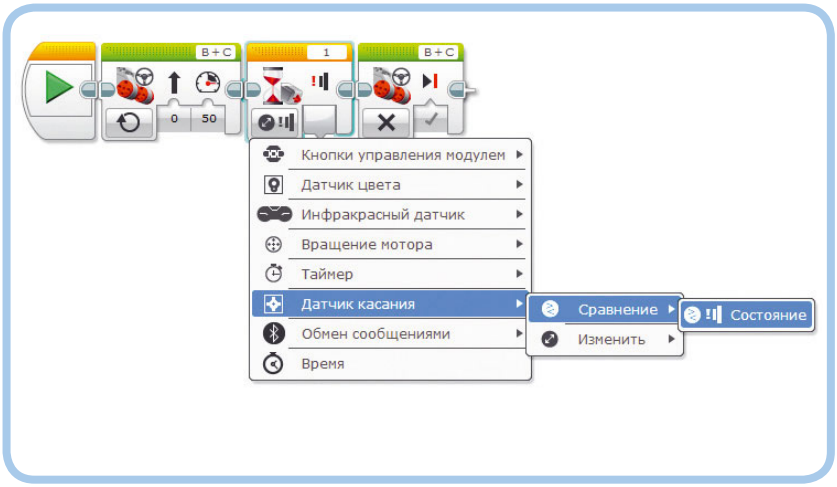


Рис. 6.9. Программа WaitForChange

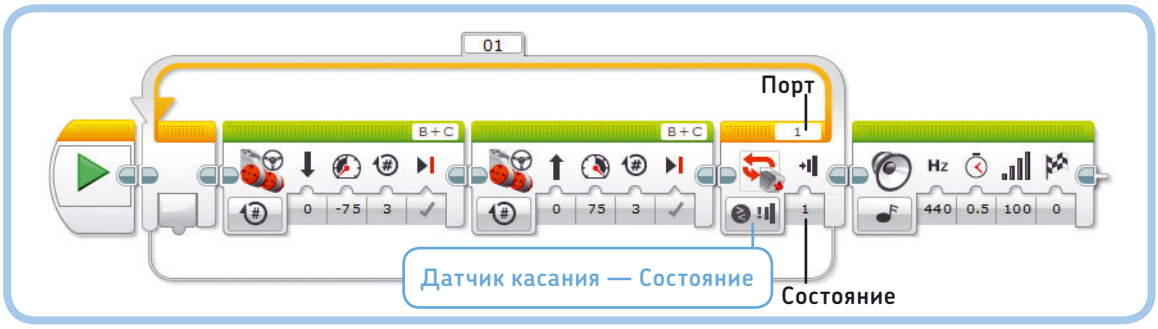


*Рис. 6.10. Программа LoopUntilTouch. Чтобы настроить блок **Цикл**, нажмите кнопку выбора режима и выберите пункт **Датчик касания** ▸ **Состояние***

вашего робота легче предугадать. Вне зависимости от исходного состояния датчика робот всегда будет ждать изменения своего поведения, пока датчик касания не достигнет заданного вами состояния.

Датчики и блок Цикл

Как вы узнали в главе 5, вы можете настроить блок **Цикл** (Loop) на повторение определенное число раз, или в течение некоторого количества секунд, или на бесконечное зацикливание. Вы также можете запрограммировать блок **Цикл** (Loop) на остановку цикла на основе входного сигнала от датчика. Например, вы можете сделать так, чтобы ваш робот двигался вперед и назад, пока кнопка датчика касания не будет нажата. Чтобы таким образом настроить блок **Цикл** (Loop), выберите режим **Датчик касания** ▸ **Состояние** (Touch Sensor ▸ State), как показано на рис. 6.10. Как и прежде, присвойте значение **1** параметру **Порт** (Port).

Создайте программу LoopUntilTouch и запустите ее, чтобы посмотреть, как она работает. Следует заметить, что программа проверяет показания датчика только один раз за полностью пройденную последовательность блоков в цикле. Для завершения цикла нужно, чтобы кнопка датчика была нажата только после того, как робот продвинется вперед. Если кнопка датчика не будет нажата в этой фазе цикла, робот повторит движение вперед и назад, прежде чем снова проверить состояние датчика касания.

Это обычный режим работы для блока **Цикл** (Loop), но иногда бывает нужно завершить цикл, даже если кнопка датчика не нажата точно в нужное время. Для достижения этой цели присвойте параметру **Состояние** (State) значение **Щелчок** (Bumped) (**2**) и запустите программу еще раз. В этой конфигурации цикл не проверяет, нажата ли кнопка датчика касания в *конце* цикла; скорее, он проверяет, наталкивается ли на что-то бампер (кнопка датчика нажата и отпущена), в любое время в *течение* цикла. Если вы установите такие настройки, то блоки должны прекратить повторяться после того, как они завершают текущий запуск программы. Во время выполнения цикла EV3 непрерывно контролирует состояние датчика касания, поэтому вам не придется беспокоиться об этом.

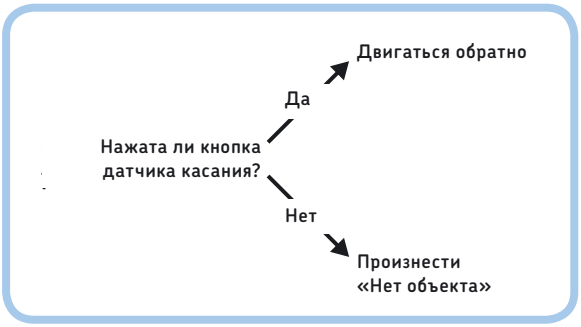


Рис. 6.11. Робот может действовать в зависимости от результата, измеренного датчиком

ПРАКТИКУМ № 26: ВЕСЕЛЫЕ МЕЛОДИИ!

Сложность: **Время:**

Используйте блок **Цикл** (Loop), чтобы робот проигрывал мелодию до тех пор, пока бампер не будет нажат. В этот момент робот должен вскрикнуть и быстро развернуться.

СОВЕТ **Вы можете использовать контейнер «Мой блок», который сделали в практикуме № 19 на с. 78 для воспроизведения мелодии. Если вам еще предстоит создать свою мелодию, просто выберите звуковой файл из списка в блоке Звук (Sound).**

*Рис. 6.12. Блок **Переключатель** проверяет, является ли условие истинным или ложным, и запускает соответствующие блоки. Вы указываете состояние, нужный режим и параметры на блоке **Переключатель***

Датчики и блок Переключатель

Вы можете использовать блок **Переключатель** (Switch), чтобы робот действовал в зависимости от результата, измеренного датчиком. Например, вы можете сделать так, чтобы робот двигался в обратном направлении при нажатии кнопки датчика касания или произносил «Нет объекта*», когда кнопка датчика в отпущенном состоянии (рис. 6.11).

Блок **Переключатель** (Switch) проверяет, является ли данное *условие* (например, «кнопка датчика касания нажата») истинным или ложным, как показано на рис. 6.12.

Блок **Переключатель** (Switch) в этом примере содержит две последовательности блоков; он запускает ту или иную последовательность в зависимости от того, истинно условие или нет. Если условие истинно, то запускается верхняя часть блока **Переключатель** и робот движется в обратном направлении; если условие ложно, выполняется нижняя часть блока и робот произносит «Нет объекта».

* Как и во всех остальных случаях, используется английский аналог выражения — No Object.

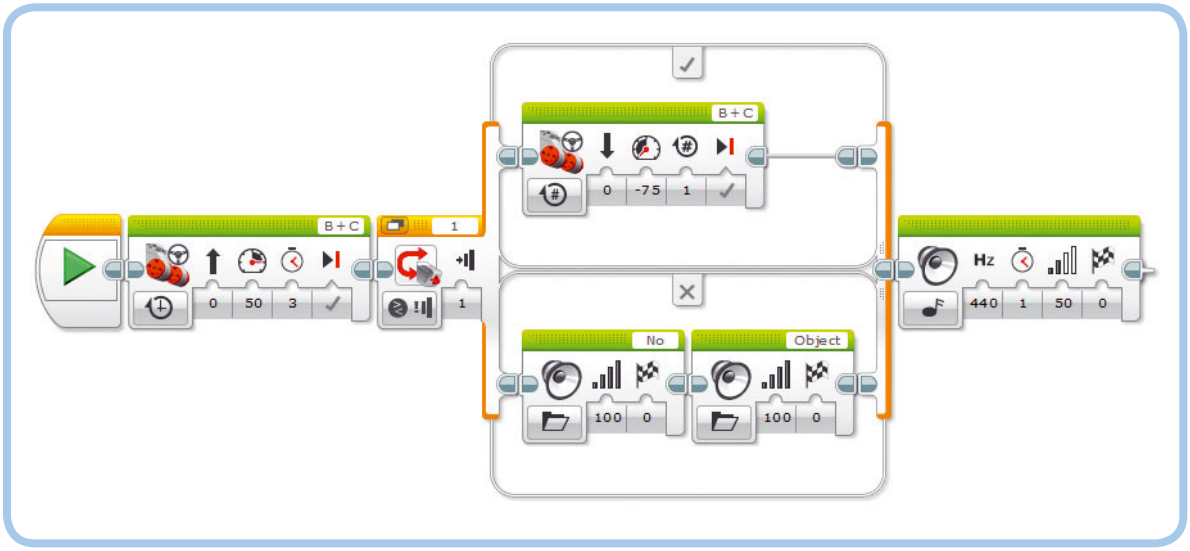


Рис. 6.13. Программа TouchSwitch позволяет роботу решать, что делать, основываясь на информации, полученной от датчика

Настройка блока Переключатель

Вы определяете состояние блока **Переключатель** (Switch) путем выбора его режима и настройки параметров. Когда выполнение программы доходит до блока **Переключатель** (Switch), робот проверяет, является ли условие истинным. Затем он решает, какую последовательность программных блоков следует запустить.

Для каждого датчика используется собственный режим. В данном случае нужно выбрать режим для датчика касания, а именно **Датчик касания ▸ Сравнение ▸ Состояние** (Touch Sensor ▸ Compare ▸ State) (единственный доступный вариант). После того как вы выбрали этот режим, вы можете с помощью параметра **Состояние** (Status) указать, должна ли кнопка датчика быть нажата (**1**) или отпущена (**0**) в случае истинности условия. Как и ранее, присвойте параметру **Порт** (Port) значение **1**, чтобы указать, к какому порту модуля EV3 подключен датчик касания.

Использование датчиков с блоком Переключатель

Программа TouchSwitch, которую вы создадите, обеспечивает движение робота вперед в течение 3 секунд. Затем, если кнопка датчика касания нажата, робот немного отъезжает назад. Если кнопка датчика не нажата, робот говорит: «Нет объекта». Наконец, независимо от решения блока **Переключатель** (Switch) робот воспроизводит мелодию. Теперь создайте программу, как показано на рис. 6.13.

Попробуйте запустить эту программу несколько раз и определить, когда необходимо нажать кнопку датчика касания, чтобы робот двигался в обратном направлении. Ваши эксперименты должны показать, что робот выполняет замер, когда блок **Переключатель** (Switch) работает, и что он использует это единственное измерение для определения, является ли условие истинным. В этой программе чтение данных с датчика производится только после того, как робот завершит движение вперед. Мелодия играет независимо от того, откатывается ли робот назад или произносит фразу «Нет объекта».

Добавление блоков в переключатель

Внутри блока **Переключатель** (Switch) вы можете поместить неограниченное количество блоков. Если хотя бы в одной части размещено несколько блоков, они просто будут запускаться один за другим. Вы можете оставить вторую часть блока **Переключатель** пустой, как показано на рис. 6.14.

Запустите модифицированную программу, чтобы посмотреть, что произойдет. Если условие истинно (бампер нажат), робот должен произнести «Объект» и поехать в обратном направлении, а программа должна воспроизводить музыку. Если условие ложно (кнопка датчика не нажата), программа, не обнаружив блоков в нижней части блока **Переключатель** (Switch), немедленно перейдет к блоку **Звук** (Sound) после переключения.

Режимы отображения блока Переключатель

Как правило, блок **Переключатель** (Switch) отображается в режиме без вкладок, с двумя последовательностями вложенных блоков сверху и снизу. При разработке сложных программ, содержащих блоки **Переключатель** (Switch), легко запутаться в схеме работы вашей программы. В таких случаях вы можете выбрать режим отображения с вкладками, чтобы уменьшить размер блока **Переключатель** (Switch), как показано на рис. 6.16. Обе части блока **Переключатель** (Switch) остаются в программе, но находятся на отдельных вкладках, которые можно переключать, щелкая по ним мышью.

Многократное выполнение блоков Переключатель

Каждый раз, когда ваша программа достигает блока **Переключатель** (Switch), он проверяет состояние датчика

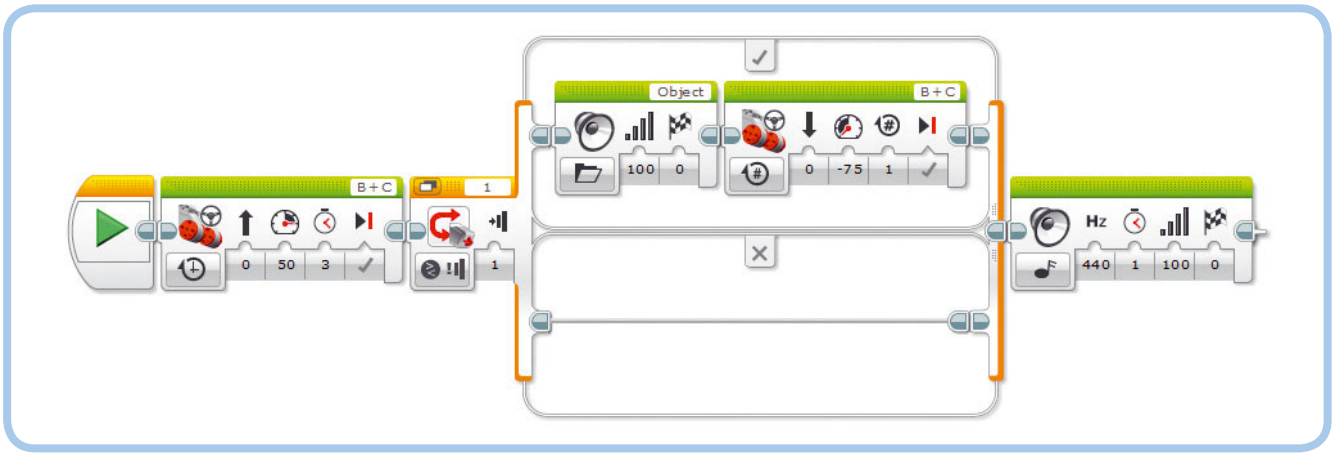


*Рис. 6.14. АМодифицированная версия программы TouchSwitch. Блок **Переключатель** не запускает каких-либо блоков, если условие ложно, поэтому по окончании движения в обратном направлении программа сразу же воспроизводит музыку, если кнопка датчика не нажата*

ПРАКТИКУМ № 27: СТОЙ ИЛИ ИДИ!

Сложность: **Время:**

Запрограммируйте робота стоять на месте в течение трех секунд. Затем, если кнопка датчика касания отпущена, робот должен развернуться и двигаться вперед на пять оборотов колес. Но, если кнопка датчика нажата, робот не должен делать ничего, и программа должна немедленно завершить работу.

касания, чтобы определить, из какой части следует запускать блоки, истинной или ложной. Чтобы робот проверил состояние более одного раза, вы можете вложить блок **Переключатель** (Switch) внутрь блока **Цикл** (Loop). Например, вы можете запрограммировать робота, чтобы он произносил: «Да», если кнопка датчика касания нажата, и «Нет», если отпущена. Если поместить блок **Переключатель** (Switch) с подобной конфигурацией в блок **Цикл** (Loop), робот будет постоянно проверять показания датчика и продолжать говорить «Да» или «Нет» соответственно. Теперь создадим программу RepeatSwitch, показанную на рис. 6.17.

Режимы Сравнение, Изменить и Измерение

При использовании датчика в программах с блоками **Ожидание** (Wait), **Цикл** (Loop) и/или **Переключатель** (Switch) часто требуется выбирать один из трех режимов работы каждого блока: **Сравнение** (Compare), **Изменить** (Change) и **Измерение** (Measure). В книге вы найдете много примеров использования этих режимов, но сначала рассмотрим внимательнее, как они работают.

ПРАКТИКУМ № 28: ТРУДНЫЕ РЕШЕНИЯ!

Сложность: **Время:**

Давайте еще поработаем с блоком **Переключатель** (Switch). Создайте программу для выполнения схемы решений, показанной на рис. 6.15. Как настроить блок **Переключатель** (Switch) и почему вы должны установить блок **Ожидание** (Wait) в конце программы?

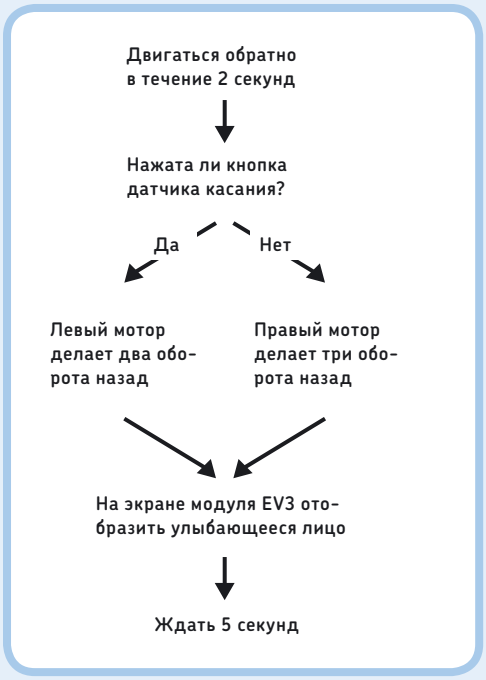


Рис. 6.16. Схема программы для практикума № 28

*Рис. 6.15. При выборе режима отображения с вкладками размер блока **Переключатель** уменьшится. Режим отображения просто меняет вид блока; он не влияет на то, как программа работает*

Режим Сравнение

Режим **Сравнение** (Compare) ( ) позволяет сверять значение датчика с *условием*, указанным в настройках блока. Условием является утверждение, например, «кнопка датчика касания нажата», «измеренная яркость света составляет менее 37%» или «датчик цвета зафиксировал красный или синий».

- Блок **Ожидание** (Wait) в режиме **Сравнение** (Compare) продолжает снимать показания с датчиков, пока условие не станет истинным. Когда это происходит, программа переходит к следующему блоку в последовательности (см. рис. 6.6).
- Блок **Цикл** (Loop) в режиме **Сравнение** (Compare) проверяет новые значения датчика каждый раз при готовности запустить блоки внутри цикла. Если условие истинно, то программа переходит к блоку, расположенному после цикла; если оно ложно, то цикл запускается снова (см. рис. 6.10). Блоки **Цикл** (Loop) всегда находятся в режиме сравнения.
- Блок **Переключатель** (Switch) в режиме **Сравнение** (Compare) запускает последовательность блоков в верхней части, если условие истинно; в противном случае запускаются блоки, находящиеся внизу (рис. 6.13).

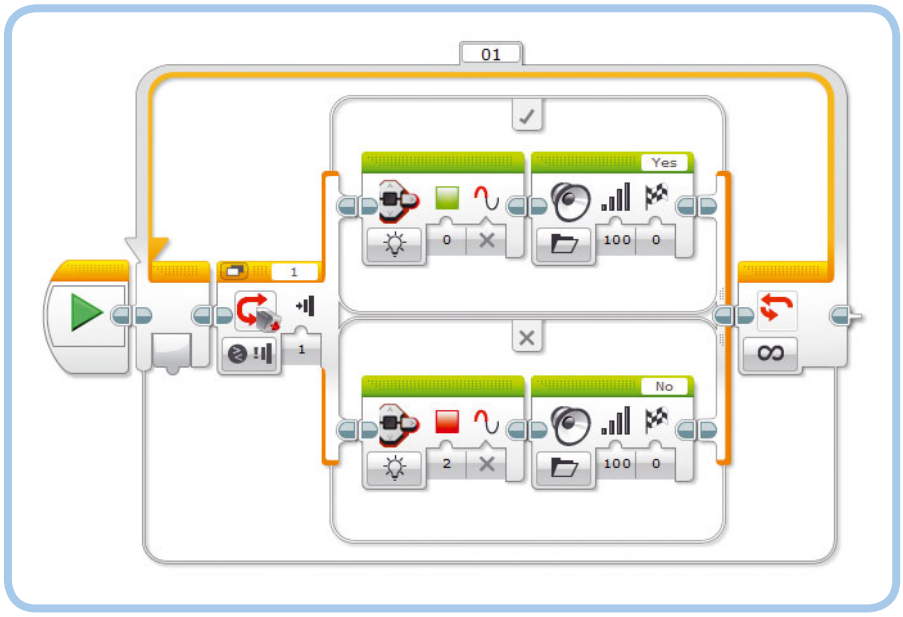


Рис. 6.17. Программа RepeatSwitch

Режим Изменить

Режим **Изменить** (Change) ( ) доступен только в блоке **Ожидание** (Wait). Блок **Ожидание** (Wait) в режиме **Изменить** (Change) снимает первое и дальнейшие показания, пока не обнаружит такое, которое отличается от первого. Например, если при запуске блока кнопка датчика касания нажата, он ждет, пока кнопка датчика не будет отпущена. Затем программа переходит к следующему блоку (см. рис. 6.9).

Режим Измерение

Режим **Измерение** (Measure) ( ) доступен только в блоке **Переключатель** (Switch). Блок **Переключатель** (Switch) в этом режиме содержит набор блоков для запуска для каждого возможного значения датчика. Вы увидите, как это работает, в главе 7, когда вы будете создавать программу, выполняющую разные действия для каждого цвета, который может распознать датчик цвета.

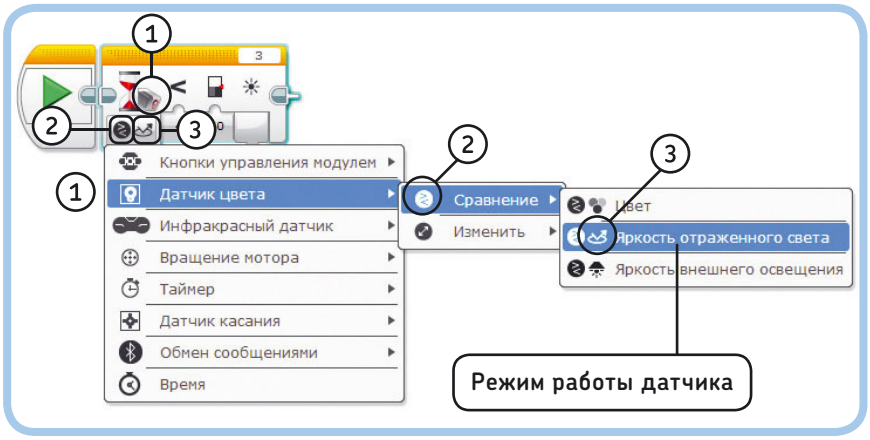


*Рис. 6.18. Если в примерах программ в этой книге используются блоки **Ожидание**, **Цикл** и **Переключатель**, выбирайте пункты меню так, чтобы они совпадали со значками, показанными на блоках. Начните с выбора подходящего датчика (1), выберите способ взаимодействия, **Сравнение**, **Изменить** или **Измерение** (2), и, наконец, выберите режим работы датчика (3)*

Настройка режимов

Как правило, из описания становится ясно, какой режим следует использовать для примеров программ, описанных в этой книге, кроме того, информацию вы можете видеть на схемах программ. Если вы не уверены, какой режим выбрать, просто взгляните на значки, указанные в каждом блоке, как изображено на рис. 6.18.

После того как вы выбрали датчик (1) и нужный режим (**Сравнение** (Compare), **Изменить** (Change) или **Измерение** (Measure)) (2), вы выбираете способ работы датчика (3). Датчик касания измеряет только один показатель (состояние красной кнопки), а датчик цвета имеет три режима работы, как показано на рис. 6.18. В последующих главах вы попрактикуетесь с каждым режимом и сможете создавать собственные программы с датчиками в кратчайшие сроки.

Дальнейшее изучение

В этой главе вы узнали, что роботы используют датчики для сбора информации из окружающей среды. Вы также узнали, как разрабатывать программы с датчиками, используя блоки **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch).

До сих пор вы работали только с датчиком касания. Для работы с другими датчиками вы можете применять все методы программирования, которые вы изучили в этой главе. В главе 7 вы научитесь работать с датчиком цвета. В главе 8 узнаете об инфракрасном датчике. В главе 9 вы разберетесь со встроенным датчиком вращения мотора и кнопками модуля EV3. Прежде чем продолжить работу, попрактикуйтесь в развитии своих навыков программирования с использованием датчика касания, решая представленные ниже практические задачи.

ПРАКТИКУМ № 29: ВЫБОР НАПРАВЛЕНИЯ!

Сложность: **Время:**

Дополните программу TouchAvoid, показанную на рис. 6.8, чтобы робот сразу после первого столкновения с объектом двигался вправо, а после повторного нажатия кнопки датчика — влево. Следующее препятствие снова должно приводить к повороту робота направо и так далее.

СОВЕТ **Удвойте количество блоков внутри блока Цикл (Loop), чтобы в цикле их стало восемь, и измените значение параметра Рулевое управление (Move Steering) во втором наборе блоков.**

ПРАКТИКУМ № 30: ОЖИДАНИЕ, ЦИКЛ ИЛИ ПЕРЕКЛЮЧАТЕЛЬ?

Сложность: **Время:**

Запрограммируйте робота ждать нажатия кнопки датчика касания. Затем, если кнопка датчика нажата дольше пяти секунд, пусть робот произносит: «Да». В противном случае он должен сказать «Нет».

СОВЕТ **Вам потребуется сочетание блоков Ожидание (Wait) и Переключатель (Switch).**

ПРАКТИКУМ № 31: КНОПКИ МОДУЛЯ!

Сложность: **Время:**

Из главы 9 вы узнаете, что большинство кнопок модуля EV3 вы можете использовать так же, как датчик касания. Не забегая вперед, можете ли вы сделать так, чтобы робот говорил «Влево» при нажатии кнопки «Влево» на модуле EV3 и «Вправо» — при нажатии кнопки «Вправо»? Можете ли вы запрограммировать робота произносить слова «Вверх» и «Вниз», когда вы нажимаете кнопки «Вверх» и «Вниз» соответственно?

СДЕЛАЙ САМ № 4: ОХРАННАЯ СИГНАЛИЗАЦИЯ!

Сборка: **Программирование:**

Можете ли вы перепрограммировать вашего робота на сигнал тревоги, который предупреждал бы вас о проникновении в комнату злоумышленников? Используйте датчик касания как переключатель, который активируется, когда кто-то входит в комнату. Когда это происходит, пусть робот воспроизведет громкий звуковой сигнал.

СОВЕТ **Положите что-нибудь тяжелое, например книгу, около передней части робота, чтобы бампер был нажат. Затем создайте хитроумную конструкцию, которая автоматически убирает книгу, когда кто-то входит в комнату. Ваш робот должен воспроизводить звук, как только кнопка датчика касания отпускается.**

СДЕЛАЙ САМ № 5: ВЫКЛЮЧАТЕЛЬ!

Сборка: **Программирование:**

Сконструируйте робота, который переключает выключатель света в вашей комнате, когда вы нажимаете кнопку датчика касания. Каждый раз, когда вы нажимаете кнопку датчика, робот должен включать или выключать свет. Чтобы робот был способен сделать это, вы можете добавить средний мотор к EXPLOR3R. Или вы можете создать совершенно нового робота, специально предназначенного для решения этой задачи.

7

Использование датчика цвета

В этой главе вы установите на EXPLOR3R (рис. 7.1) и научитесь использовать датчик цвета, с помощью которого робот сможет определять цвет бумаги, следовать по линии и реагировать на световые сигналы.

Датчик цвета может определить цвет поверхности (в режиме **Цвет** (Color)), яркость света, отраженного от поверхности (в режиме **Яркость отраженного света** (Reflected Light Intensity)), или яркость окружающего освещения (в режиме **Яркость внешнего освещения** (Ambient Light Intensity)), как показано на рис. 7.2.

Вы создадите программы для EXPLOR3R, чтобы опробовать каждый режим работы, используя блоки **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch), как ранее с датчиком касания. По мере чтения книги и сборки роботов, описанных в ней, вы увидите больше применений для этого датчика, например в виде модели LAVA R3X в главе 19, измеряющей яркость отраженного света с целью обнаружить рукопожатие.

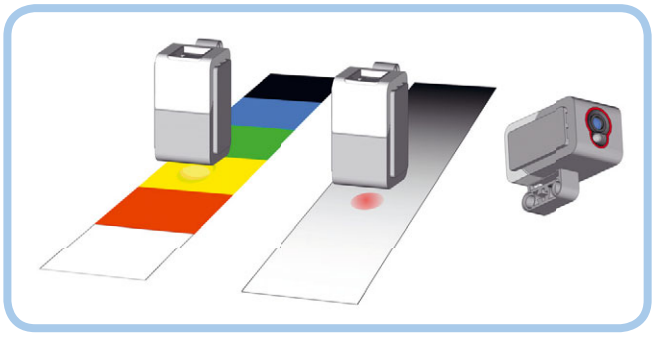


*Рис. 7.2. Три режима работы датчика цвета: **Цвет** (слева), **Яркость отраженного света** (средний) и **Яркость внешнего освещения** (справа). Датчик справа направлен вверх для определения яркости света в комнате*

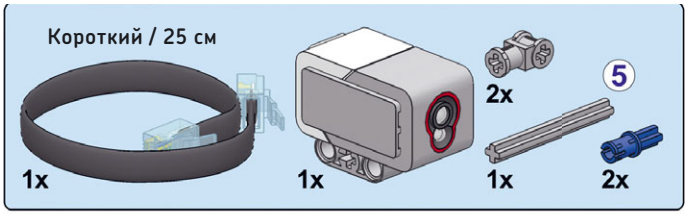
Рис. 7.1. Используя датчик цвета, EXPLOR3R может определять цвета и следовать по линии

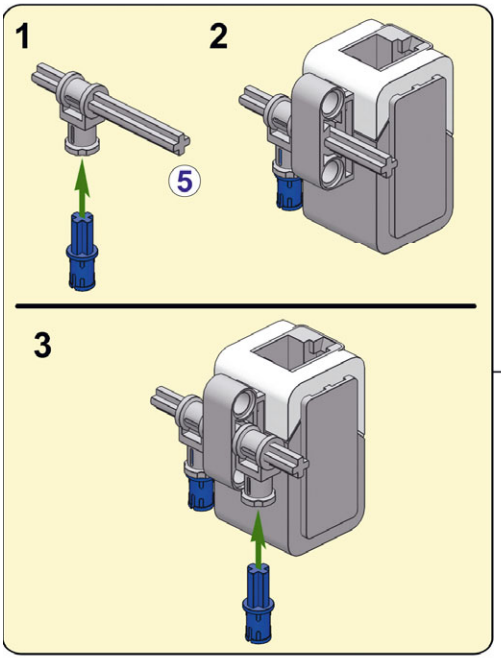
Подключение датчика цвета

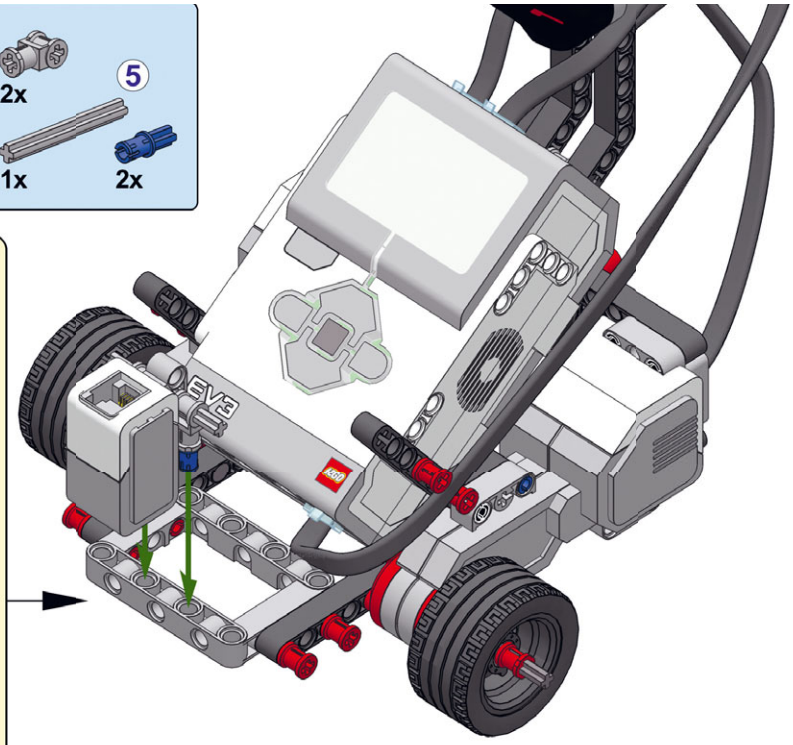
Перед тем как начать разработку программ, снимите подключенный датчик касания с робота (не убирайте его далеко, позднее он вам понадобится). Затем подключите датчик цвета, используя следующую инструкцию по сборке.

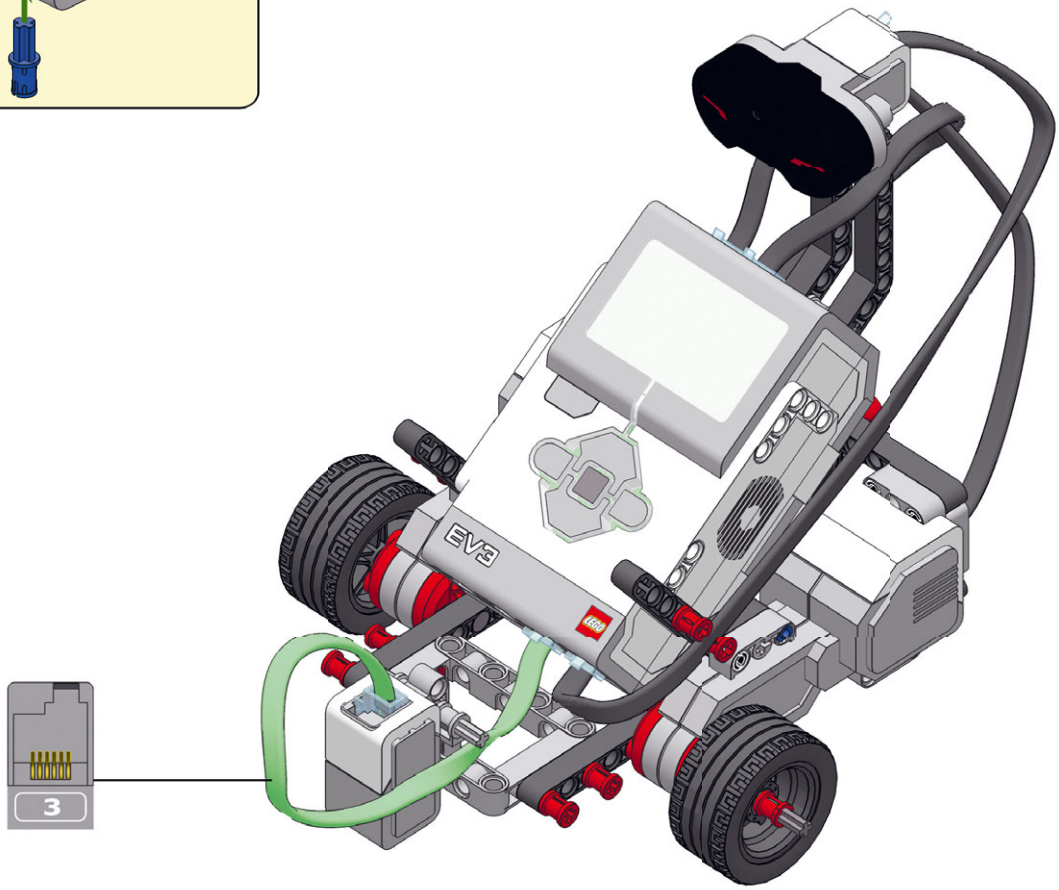
Цветовой режим

Первый режим работы, который вы будете использовать, это **Цвет** (Color). В этом режиме датчик может определять цвет поверхности на расстоянии около 1 см. Датчик устанавливается в направлении прямо вниз, поэтому он может определять цвет непосредственно под роботом. Для проверки определения цвета скачайте и распечатайте на цветном принтере файл *Датчик_цвета.pdf* по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip** и поставьте робота на


Короткий / 25 см
1x
1x
2x
5
1x
2x


1
2
5
3


EV3


EV3
3

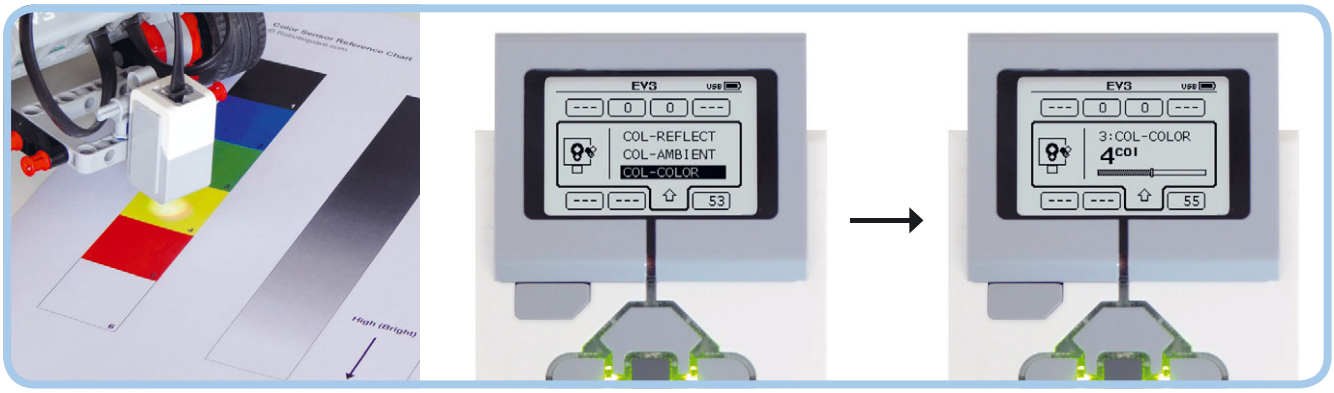


*Рис. 7.3. Запустите приложение **Port View**, чтобы увидеть результаты замеров, произведенных датчиком. Перейдите к датчику на входном порте 3 и нажмите центральную кнопку. Затем выберите пункт **COL-COLOR** и снова нажмите центральную кнопку. Теперь вы должны увидеть число в диапазоне от 0 до 7, которое обозначает цвет. В данном случае от датчика получено значение 4, что соответствует желтому цвету*

распечатку. Затем перейдите к приложению **Port View** на модуле EV3, чтобы увидеть значение обнаруженного цвета, указанное в виде числа (рис. 7.3). Если у вас нет цветного принтера или если датчик не определяет цвета правильно, попробуйте использовать трассу для выполнения миссий, которая поставляется в комплекте с набором EV3.

Датчик определяет следующие цвета: черный (1), синий (2), зеленый (3), желтый (4), красный (5), белый (6) и коричневый (7). Значение 0 указывает на то, что датчик не может определить цвет, потому что находится слишком далеко от поверхности или слишком близко к ней.

Нахождение внутри контура

Программы могут использовать блоки **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch) для выбора последовательности действий на основе информации, полученной от датчика. Например, вы можете сделать робота, который будет двигаться в пределах черного контура, показанного на рис. 7.4. Для этого робот должен ехать вперед, пока не обнаружит черную линию. Затем он должен дать задний ход, развернуться и двигаться вперед в другом направлении.

Создание тестовой трассы

Сначала вам нужно создать круговую трассу, которую вы будете использовать для тестирования датчика. Трасса состоит из набора элементов, напечатанных на стандартной бумаге формата А4. Скачайте и распечатайте документ *Тестовая_трасса.pdf* по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip**. Эти элементы нужно вырезать по пунктирной

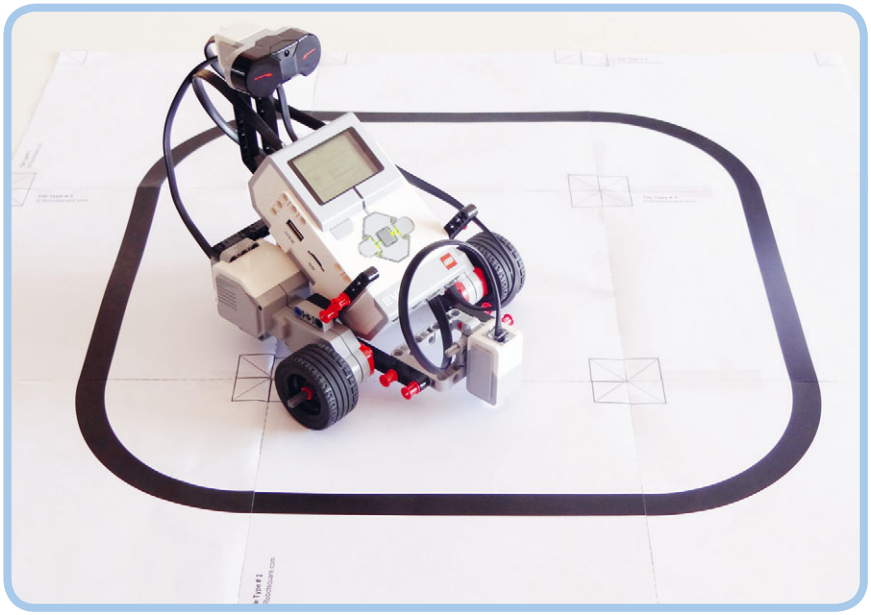


Рис. 7.4. Робот EXPLOR3R перемещается, не выезжая за пределы черного контура тестовой трассы

линии и склеить скотчем, чтобы получилось цельное полотно, как показано на рис. 7.4.

Если у вас нет возможности напечатать тестовую трассу, сделайте собственную, используя черный скотч на светлой поверхности, например на белой напольной плитке или большом листе фанеры.

Создание программы

На рис. 7.5 показана схема программы, которую вы должны создать, чтобы робот перемещался в пределах черного контура тестовой трассы. Она похожа на программу уклонения от препятствий, которую вы разрабатывали для датчика касания (см. рис. 6.7 в главе 6), только на этот раз программа отслеживает черную линию, а не нажатие кнопки.

Для создания этой программы вы будете использовать блок **Ожидание** (Wait) в режиме **Датчик цвета** ▸ **Сравнение** ▸ **Цвет** (Color Sensor ▸ Compare ▸ Color). В этой конфигурации вы можете выбрать цвета, которые датчик должен искать, с помощью параметра **Набор цветов** (Set of Colors). Когда обнаруживается какой-либо из них, блок

СДЕЛАЙ САМ № 6: БУЛЬДОЗЕР!

Сборка: **Программирование:**

С помощью программы StayInCircle робот будет перемещаться в разных направлениях. Если вы положите несколько деталей LEGO в круг, ваш робот может вытолкнуть их. Но сначала вам нужно снабдить EXPLOR3R специальным отвалом* бульдозера. Можете ли вы собрать такой отвал из деталей LEGO?

* Передняя часть бульдозера, которой он сгребает землю, мусор и т. п.

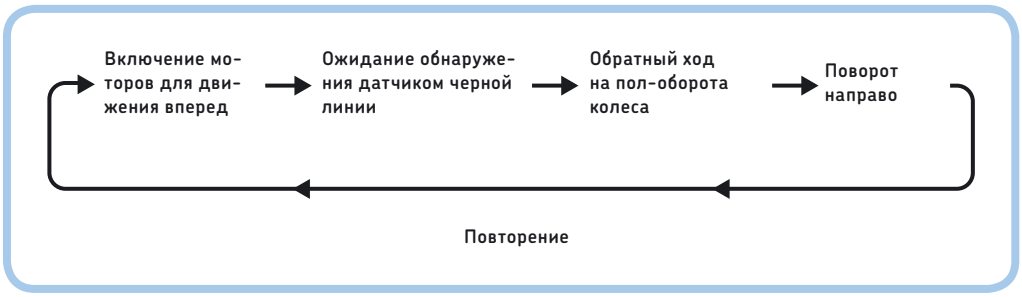


Рис. 7.5. Схема программы StayInCircle

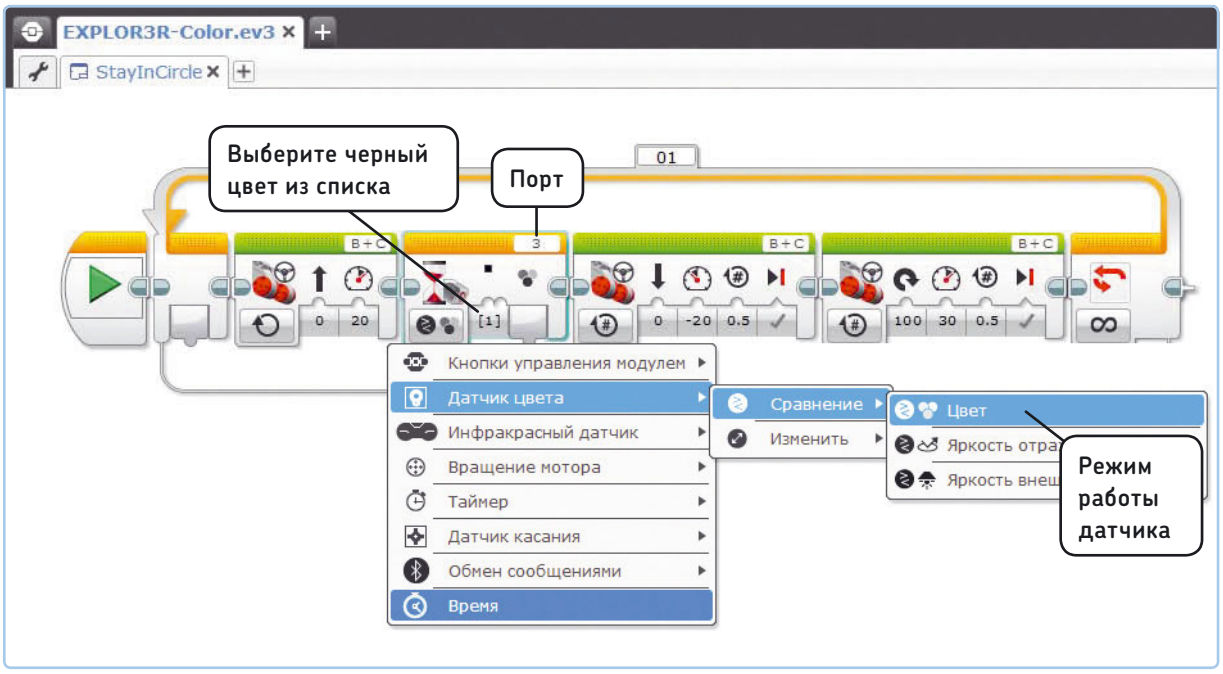


*Рис. 7.6. Новый проект EXPLOR3R-Color с одной программой StayInCircle. Чтобы настроить блок **Ожидание**, сначала выберите режим; затем откройте раскрывающийся список **Набор цветов** (Set of Colors) и выберите черный цвет (пункт **1**) из списка*

останавливается в ожидании, и программа переходит к следующему блоку. Рис. 7.6 демонстрирует готовую программу StayInCircle с блоком **Ожидание** (Wait), настроенным на ожидание черной линии. Поместите робота внутрь черного контура на тестовой трассе и запустите программу, чтобы увидеть ее в действии.

Движение по трассе

В своем следующем проекте вы будете использовать датчик цвета, чтобы создать робота, движущегося по трассе. Такой робот будет двигаться строго по линии тестовой трассы. Давайте рассмотрим принцип работы этой программы.

Когда робот следует по черной линии на белом фоне, датчик будет определять один из двух цветов: белый или черный. Поэтому, чтобы создать программу, определяющую трассу в черно-белой среде, вы можете использовать блок **Переключатель** (Switch), который отслеживает черный цвет. Когда датчик определяет черный цвет, блок **Переключатель** (Switch) запускает блок **Рулевое управление** (Move Steering) для выполнения одного движения; если блок **Переключатель** (Switch) определяет белый цвет, робот выполняет другое движение, как это показано на рис. 7.7.

Если робот обнаруживает белый цвет, он не будет знать, с какой стороны трассы он находится. Вы должны убедиться, что он всегда остается только с одной стороны линии; в противном случае он будет отклоняться от трассы в белую область. Вы можете сделать это, всегда поворачивая EXPLOR3R направо, когда он обнаруживает черный цвет, и налево, когда обнаруживает белый цвет. Если вы сделаете так, чтобы робот немного проезжал вперед, как бы подсказывая ему, и зациклите это поведение, робот будет следовать по трассе. Программа Colorline реализует эту стратегию, как показано на рис. 7.8.

ПРИМЕЧАНИЕ **Если ваш робот отклоняется от трассы в углах или на крутых поворотах, замедлите его движение (укажите 20% скорости вместо 25%).**

Теперь давайте посмотрим на поведение робота, когда он следует по линии трассы. Обратите внимание на то, что робот на самом деле следует по *краю* линии. Программа поддерживает настройки рулевого управления робота на основании данных, получаемых от датчика, заставляя его двигаться зигзагом у края линии: как только он обнаруживает черную линию, он пытается уйти от нее направо; как только он видит белую область, он пытается вернуться к трассе, двигаясь налево.

В результате робот движется таким образом, что линия трассы находится слева от датчика. Это означает, что, если поместить робота на трассе таким образом, чтобы он двигался по кругу по часовой стрелке, он придерживается внутреннего края окружности; если вы

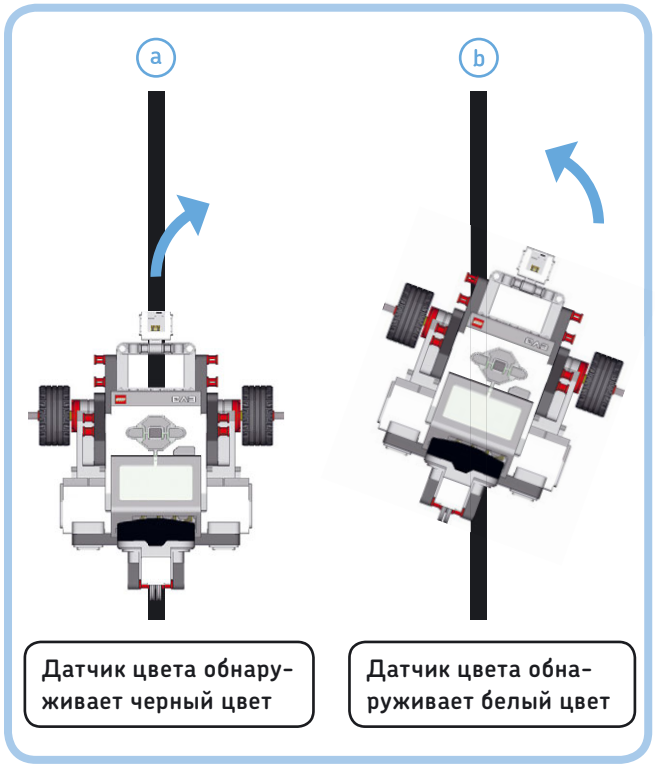


Рис. 7.7. Робот EXPLOR3R поворачивает вправо, если обнаруживает черную линию (a), и влево, если определяет белую область (b). Таким образом он движется вперед, поэтому, если зациклить это поведение, ваш робот в конечном итоге будет следовать по трассе

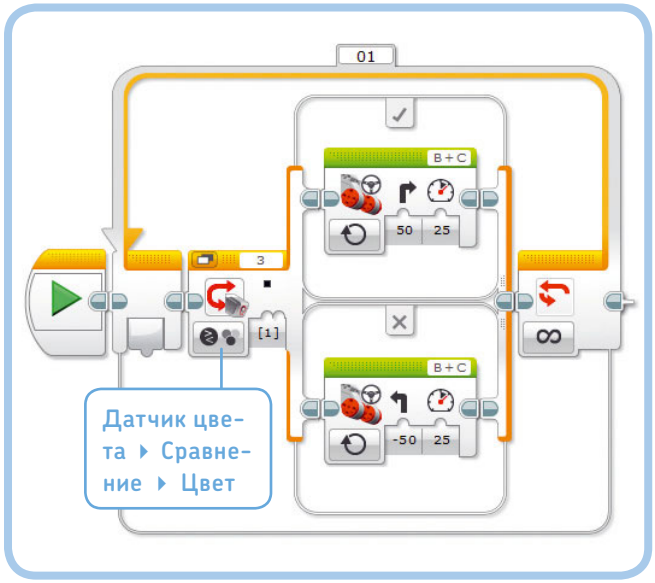


*Рис. 7.8. Программа Colorline. Обратите внимание, что блоки **Рулевое управление** находятся в режиме **Включить**. Как только робот начинает поворачивать, программа немедленно возвращается к началу для определения, был ли обнаружен другой цвет или поворот следует продолжить в том же направлении. Режим **Включить** лишь включает моторы, и программа продолжает выполняться*

пустите его против часовой стрелки, то он поедет по внешнему краю. Чтобы убедиться в этом, поверните робота во время работы программы, чтобы он двигался по линии в другую сторону; он должен начать двигаться по другому краю трассы.

Блок Переключатель в режиме Измерение

Блок **Переключатель** (Switch) в программе Colorline задает роботу движение прямо, если датчик обнаруживает черную линию. Если он определяет какой-либо другой цвет, например зеленый или красный, робот двигается влево. Изменив режим работы блока **Переключатель** (Switch) на **Датчик цвета** ▸ **Измерение** ▸ **Цвет** (Color Sensor ▸ Measure ▸ Color), вы можете настроить блок так, чтобы робот выполнял различные действия для каждого цвета.

Программа ShowColor на рис. 7.9 меняет цвет индикатора состояния модуля в зависимости от информации, полученной от датчика цвета. Блок **Переключатель** (Switch) в этой программе содержит четыре *варианта*, каждый из них включает один или несколько блоков. Каждый вариант соответствует различным замерам цвета: индикатор состояния подсвечивается зеленым цветом, если датчик обнаруживает зеленый; красным, если красный, и оранжевым, если желтый. Если цвет не обнаружен (⧄), индикатор состояния гаснет. Программа запускает блоки, относящиеся к тому варианту, который она определила, когда добралась до блока **Переключатель** (Switch).

Но что произойдет, если датчик обнаружил черный, синий, белый или коричневый? Если ни один из вариантов поведения не соответствует значению датчика, блок **Переключатель** (Switch) запускает вариант по умолчанию, который отмечен точкой, как показано на рис. 7.9. Здесь запускаются те же блоки, которые выполняются, когда цвет не обнаружен.

Теперь создайте программу и поместите датчик поверх распечатки файла *Датчик_цвета.pdf*, чтобы проверить работу датчика.

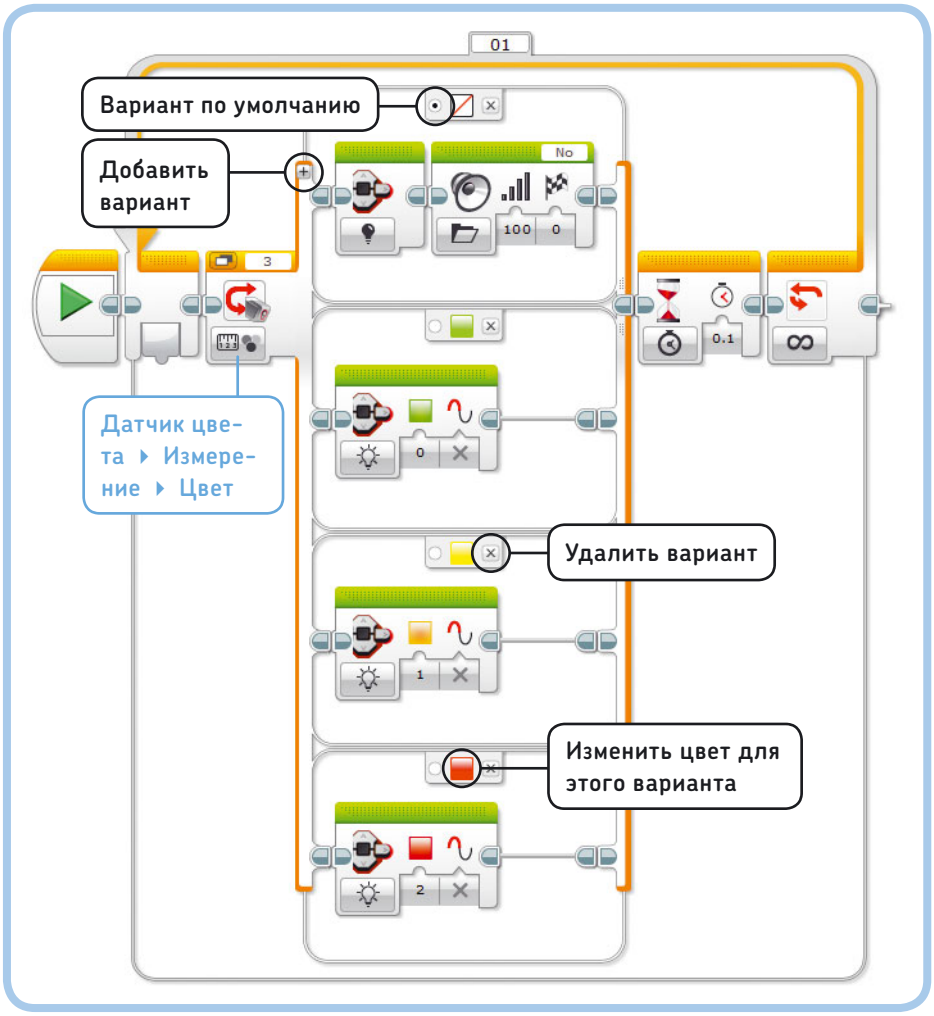


*Рис. 7.9. Программа ShowColor. Чтобы настроить варианты блока **Переключатель**, сначала выберите команду **Датчик цвета** ▸ **Измерение** ▸ **Цвет** и добавьте два случая, два раза нажав кнопку **+**. Теперь, когда ваш блок **Переключатель** содержит четыре варианта, выберите для каждого из них свой цвет. Не забудьте указать, что вариант, при котором цвет не обнаруживается, используется по умолчанию, установив переключатель в соответствующее положение (см. рисунок)*

ПРАКТИКУМ № 32: СОЗДАЙТЕ СОБСТВЕННУЮ ТРАССУ!

Сложность: **Время:**

Тестовая трасса, которую вы только что сделали, — отличное начало, но EXPLOR3R может освоить гораздо более сложные трассы. Перейдите по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip**, чтобы скачать файл *Настраиваемая_трасса.pdf* и создать собственную трассу. Вы можете выбрать любые из тридцати типов элементов, включая прямые линии, углы и перекрестки. Напечатайте элементы трассы, которые вам нравятся, обрежьте их по пунктирным линиям и не забудьте использовать скотч, чтобы склеить поле воедино.

Для начала напечатайте четыре угла (четыре копии с. 3), зигзагообразную линию (с. 15), а также прямую линию, которую пересекает линия синего цвета (с. 18). С помощью этих элементов можно собрать трассу, показанную на рис. 7.10. Запустите программу Colorline, которую вы сделали, чтобы проверить EXPLOR3R на новой трассе.

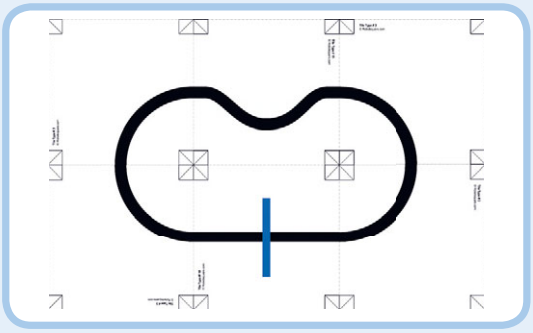


Рис. 7.10. Трасса для робота из практикумов № 32 и № 33

СОВЕТ **Некоторые принтеры позволяют печатать все необходимые страницы по очереди одной командой. Для этого надо ввести номера страниц в поле Страницы (Pages) настроек принтера следующим образом: 3,3,3,3,15,18.**

ПРАКТИКУМ № 33: ОСТАНОВИСЬ НА СИНИЙ!

Сложность: **Время:**

Измените программу Colorline так, чтобы робот следовал по черной линии трассы, которую вы сделали в практикуме № 32, пока она не пересечется с синей линией. Когда робот обнаружит синий цвет, он должен остановиться и воспроизвести звук.

СОВЕТ **Смените режим блока Цикл (Loop) на обнаружение синего цвета.**

ПРАКТИКУМ № 34: НАЗОВИ ЦВЕТ!

Сложность: **Время:**

Создайте программу, с помощью которой робот сможет сообщать вам, какой цвет он видит. Задайте такие параметры, чтобы он произносил «Blue», если датчик обнаружил синий цвет, и так далее. Если он не обнаружил цвет, настройте произнесение какой-либо другой фразы. Затем превратите блок **Переключатель** (Switch), который определяет цвет, в контейнер «Мой блок» с именем **SayColor**. Вы сможете использовать этот блок в любое время, когда захотите узнать, какой цвет обнаружил робот.

СОВЕТ **Ваша программа будет похожа на программу ShowColor.**

Режим Яркость отраженного света

Датчик цвета может измерять и яркость цвета в режиме **Яркость отраженного света** (Reflected Light Intensity). Например, датчик может определить разницу между белой, серой и черной бумагой, освещая бумагу и измеряя количество отраженного света. **Яркость отраженного света** измеряется в процентах от значения 0% (очень низкий коэффициент отражения: темный) до 100% (очень высокий коэффициент отражения: светлый).

Черная бумага отражает небольшое количество света, в результате чего замеры этого показателя будут ниже 10%. Белая бумага может показать результат измерения выше 60%. Чтобы проверить

ПРАКТИКУМ № 35: СУПЕРОТРАЖАТЕЛЬ!

Сложность: **Время:**

Подберите по крайней мере один материал, значение яркости отраженного света которого достигает 100%. Что это за материал и почему значение так высоко?

эти значения, запустите приложение **Port View** на модуле EV3, укажите порт **3**, а затем выберите пункт **COL-REFLECT** (см. рис. 7.3). Поставьте робота на распечатку файла *Датчик_цвета.pdf* и наблюдайте, как изменяется значение по мере передвижения робота по цветовой схеме с различными оттенками серого.

Установка порогового значения

В режиме **Цвет** (Color) датчик мог определять цвет тестовой трассы, черный или белый. Теперь наблюдайте за измерением отраженного света, передвигая робота руками (очень медленно!) от черной линии в белую область тестовой трассы. Вы заметите, что измеряемое значение постепенно увеличивается примерно от 6% (черный) до 62% (белый). Когда датчик частично фиксирует черную линию и частично белую бумагу, значение будет находиться в середине указанного диапазона, как будто датчик фиксирует серую поверхность. Это более детальное измерение вы можете использовать для улучшения вашего робота, движущегося по трассе.

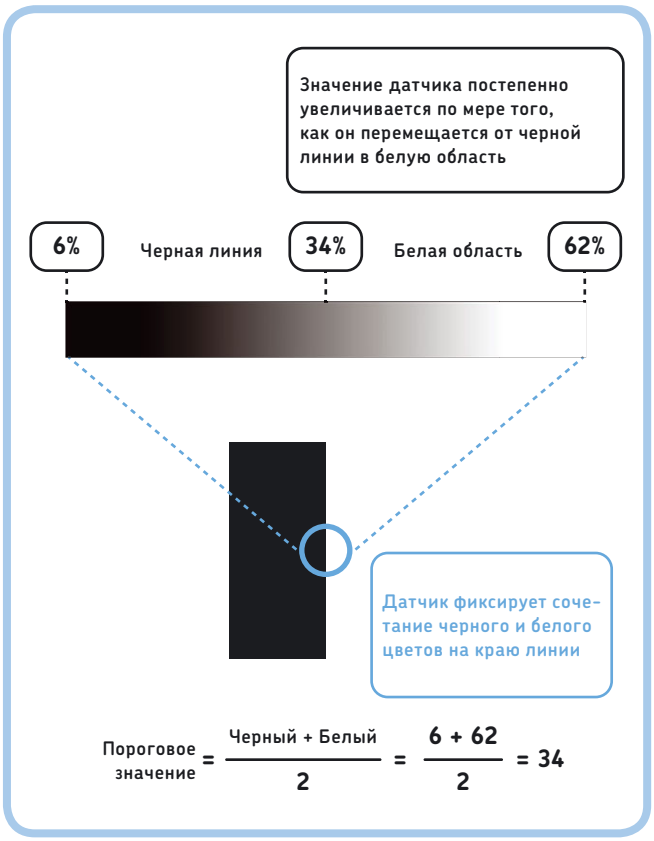


Рис. 7.11. На краю линии трассы датчик фиксирует сразу и черный и белый цвета и выдает среднее значение между ними, как будто перед ним серая поверхность. Пороговым считается среднее значение показаний датчика, которое он получил при измерении на черной линии (меньшее число) и на белой области (большее число). Чтобы вычислить среднее, сложите эти два значения и поделите сумму на 2

Чтобы сообщить роботу, какой цвет вы решили измерять — белую область или черную линию трассы, задайте белый и черный цвета в программе, используя *пороговое значение*. Измеренное значение выше порогового будет считаться белым цветом, а значение ниже этого порога — черным. Другими словами, вы принимаете темно-серые тона в качестве черных, а светло-серые в качестве белых, как показано на рис. 7.11.

Сравнение замеров датчика с пороговым значением

Теперь создадим для движения по трассе новую программу, которая использует режим **Яркость отраженного света** (Reflected Light Intensity) и пороговое значение, чтобы определить, где находится датчик — на черной линии или белой области. Как и прежде, робот должен повернуть направо, если он обнаружил черную линию, и налево — если белый цвет.

Для достижения этой цели используйте блок **Переключатель** (Switch) в режиме **Датчик цвета ▸ Сравнение ▸ Яркость отраженного света** (Color Sensor ▸ Compare ▸ Reflected Light Intensity), как показано на рис. 7.12. Введите ранее вычисленный порог в качестве значения параметра (Threshold Value). Параметр (Compare Type) определяет, какие показания датчика делают условие *истинным*, то есть какие значения будут запускать последовательность блоков в верхней части блока **Переключатель** (Switch). Вы можете выбрать для запуска верхние блоки, когда значение соответствует одному из следующих условий.

0. Равно (=) пороговому значению
1. Не равно (≠) пороговому значению
2. Больше чем (>) пороговое значение
3. Больше или равно (≥) пороговому значению
4. Меньше (<) порогового значения
5. Меньше или равно (≤) пороговому значению

Вам нужно, чтобы робот поворачивал направо, если датчик обнаруживает черный цвет (значение *меньше* порогового), поэтому вы можете выбрать это значение в качестве условия, запускающего верхнюю последовательность блоков. А теперь создайте и запустите программу ReflectedLine1 (см. рис. 7.12) и протестируйте робота точно так же, как в предыдущей программе.

ПРИМЕЧАНИЕ **Обязательно рассчитайте собственные пороговые значения, а не используйте приведенные в диаграмме. У вас могут получиться другие результаты замеров для черного и белого цвета, в зависимости от таких факторов, как уровень освещенности в помещении, уровень заряда батареи робота и тип используемой бумаги.**

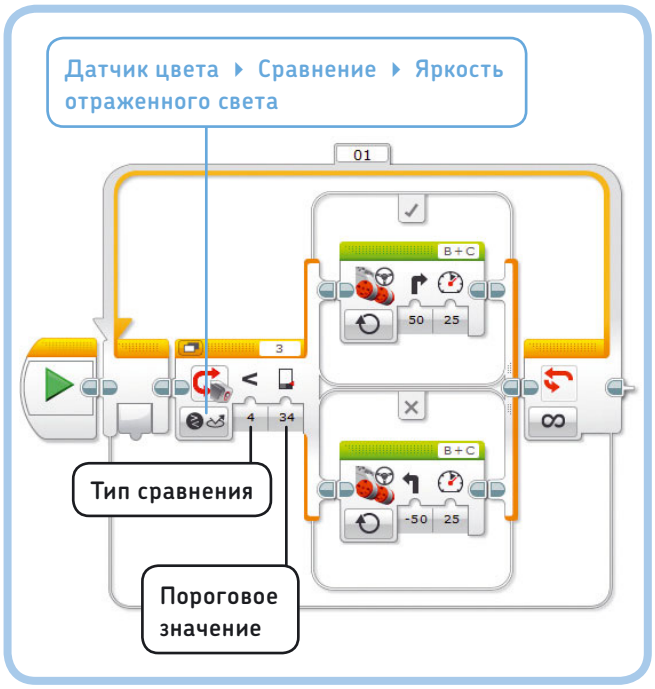


*Рис. 7.12. Согласно инструкциям программы ReflectedLine1 робот EXPLOR3R движется по линии тестовой трассы, используя режим **Яркость отраженного света**. Для параметра **Тип сравнения** используется значение **Меньше (<)** (под номером 4)*

Плавное движение по трассе

Преимущество режима **Яркость отраженного света** (Reflected Light Intensity) заключается в том, что робот, двигаясь по краю линии трассы, может измерять не только черный и белый, но и сочетание черного и белого цветов. В программе ReflectedLine1, если робот обнаруживает белый цвет, он делает резкий поворот влево, чтобы вернуться на трассу.

Но резкие повороты не нужны, если датчик находится близко к линии трассы. Если робот получил данные о светло-сером цвете, то для возвращения на трассу достаточно мягкого левого поворота. Все это ведет к тому, что робот плавно движется по трассе, а не дергается влево-вправо по нескольку раз. Иногда бывает недостаточно совершить мягкий поворот, чтобы вернуться к линии трассы, в таком случае робот должен сделать резкий поворот, когда он далеко от трассы и фиксирует белый цвет.

Чтобы определить, являются ли полученные роботом данные с датчика черным, темно-серым, светло-серым или белым цветом, нужно использовать два дополнительных пороговых значения, на полпути между тремя известными значениями, как показано на рис. 7.13.

Схема на рис. 7.14 иллюстрирует, как робот должен определить, в каком направлении повернуть и нужно ли сделать резкий или мягкий ход. Запустите программу ReflectedLine2 (рис. 7.15), чтобы убедиться, что робот движется по линии более плавно, чем раньше. Когда вы запустите программу, индикатор состояния модуля поможет вам определить, делает ли робот резкий поворот (индикатор подсвечивается красным цветом) или мягкий (зеленым цветом).

Режим Яркость внешнего освещения

Датчик цвета можно применить для определения уровня освещенности в помещении или яркости источника света, если выбрать режим **Яркость внешнего освещения** (Ambient Light Intensity). Этот режим можно использовать, чтобы робот

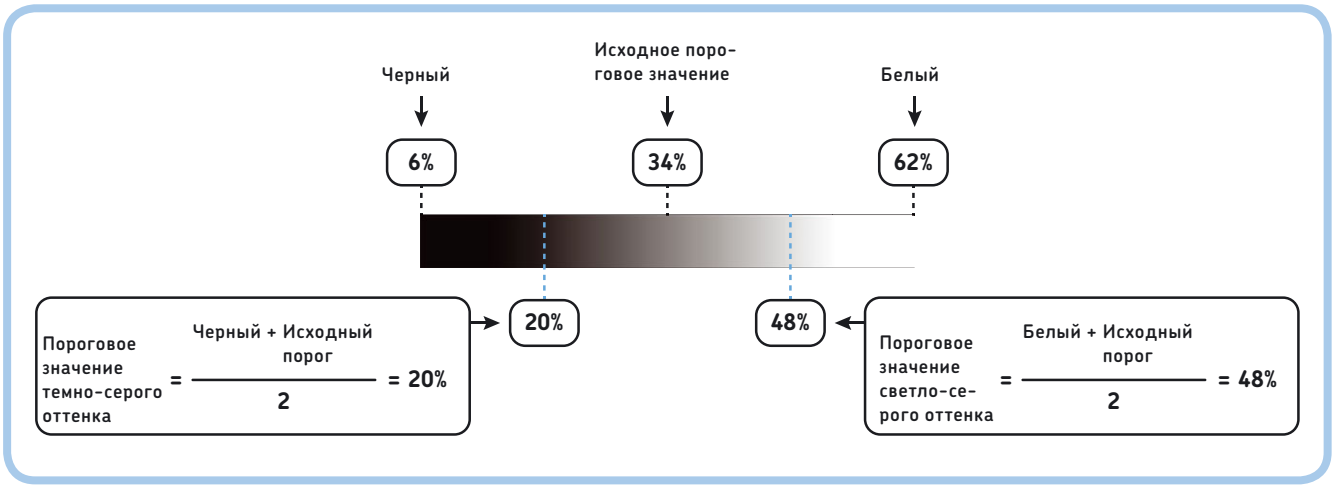


Рис. 7.13. Вам необходимо рассчитать два дополнительных пороговых значения, чтобы различать черный, темно-серый, светло-серый и белый цвета. Как и прежде, каждый порог представляет собой среднее от двух известных значений. Например, пороговое значение темно-серого цвета (20%) представляет собой среднее значение между черным (6%) и первоначальным пороговым значением (34%)

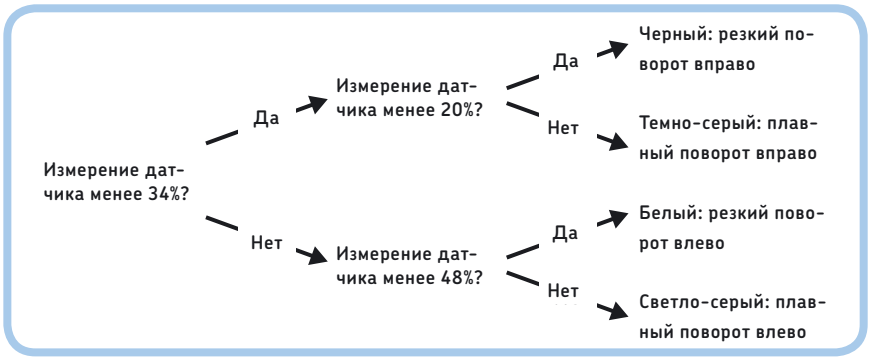


Рис. 7.14. Схема работы программы ReflectedLine2

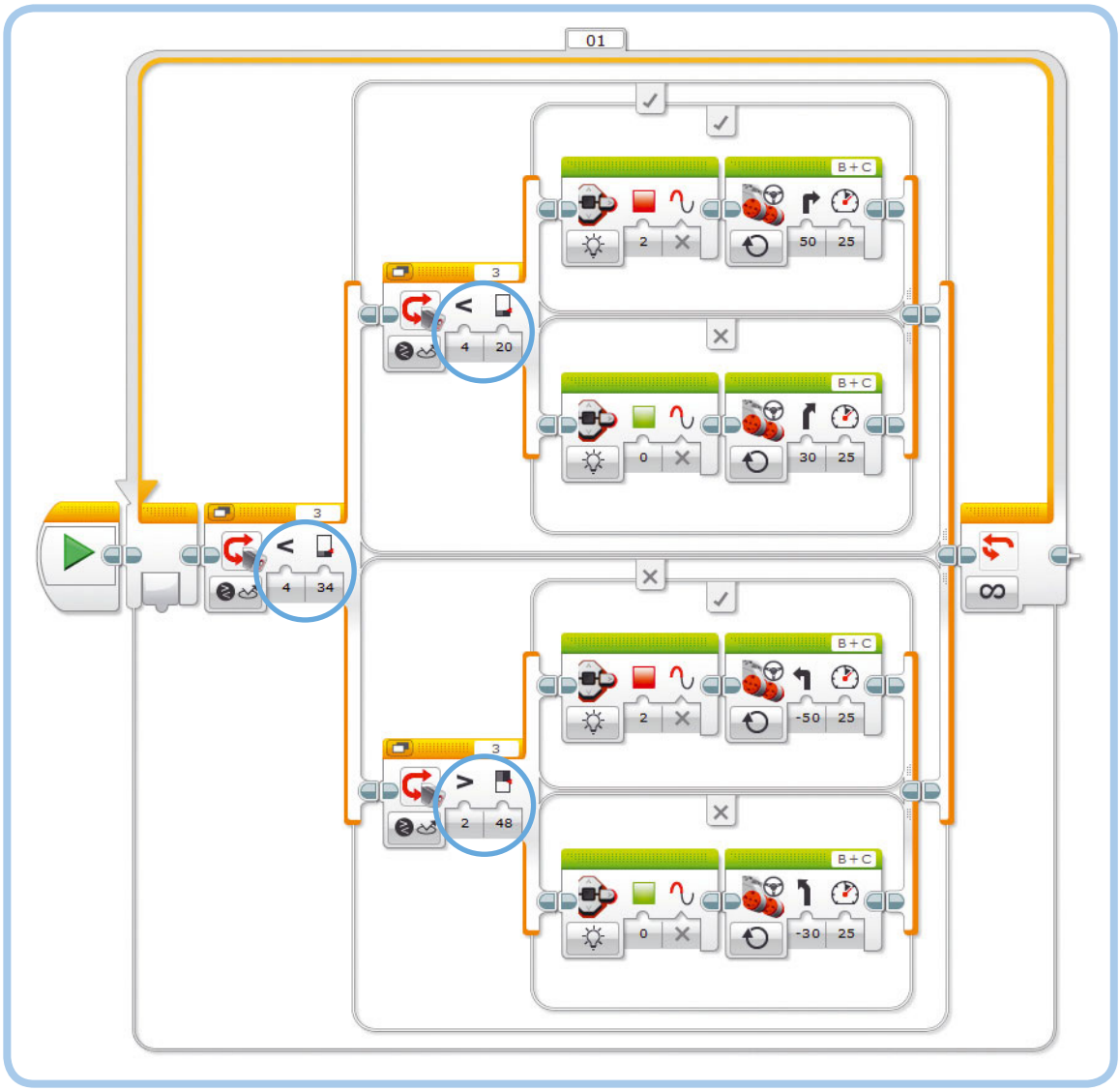


*Рис. 7.15. Программа ReflectedLine2. Обратите внимание, что блок **Переключатель** в нижней части сконфигурирован с возможностью запуска блоков в его верхней части, если показание датчика больше (>) светло-серого порогового значения. Для указания этого условия выберите вариант **2** в раскрывающемся списке **Тип сравнения***