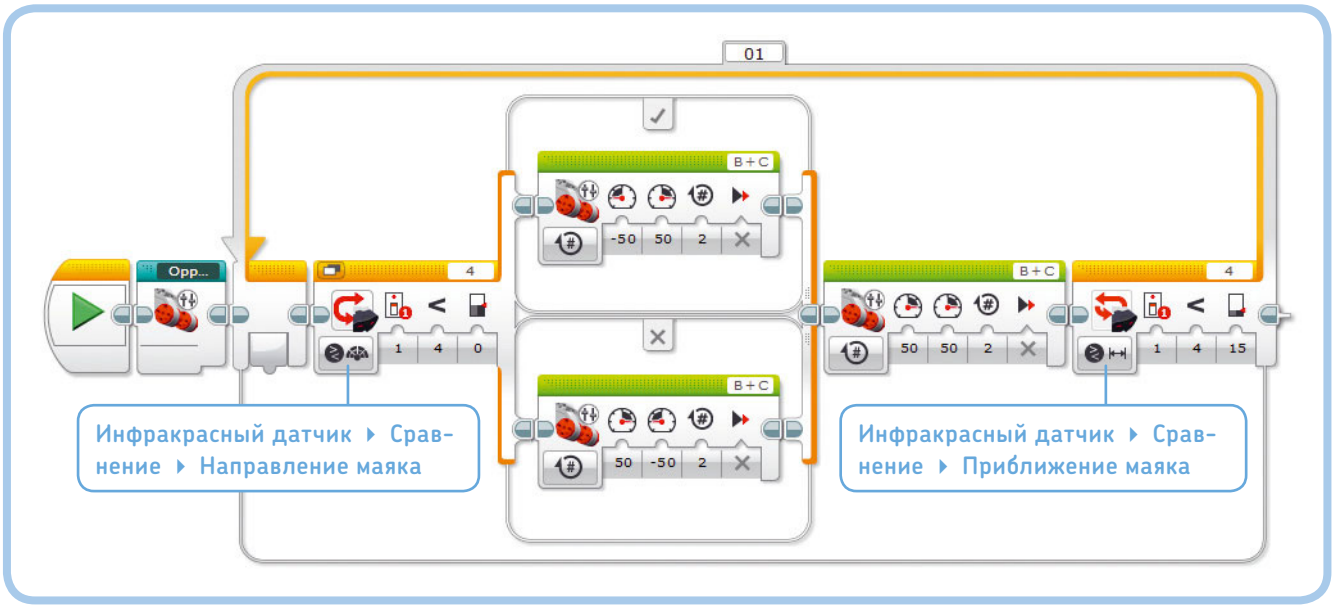


*Рис. 13.6. Программа FindingFood. Блок **Цикл** превратился в контейнер «Мой блок» с именем **Find***

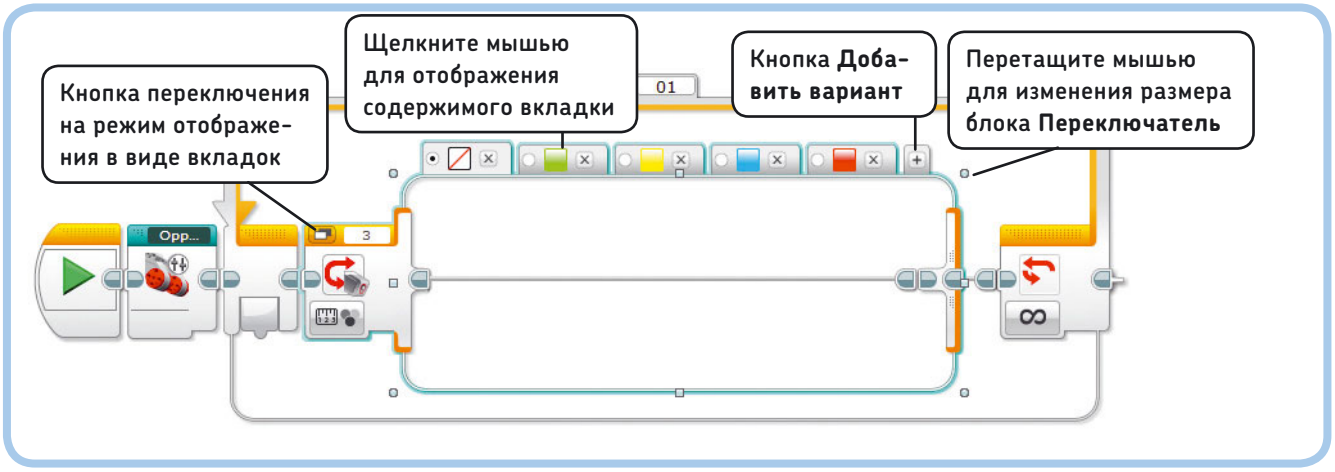


Рис. 13.7. Базовая структура программы ColorBehavior. Блоки будут добавлены на все вкладки

Начнем с создания базовой структуры программы ColorBehavior. Робот должен регулярно проверять, какой цвет он видит, и реагировать на каждый цвет по-разному. Запрограммируйте на это робота с помощью блока **Переключатель** (Switch) в режиме **Датчик цвета ▸ Измерение ▸ Цвет** (Color Sensor — Measure — Color), помещенного в блок **Цикл** (Loop), как показано на рис. 13.7.

Для переключения используется пять вариантов: **Нет цвета** (No Color), **Зеленый** (Green), **Желтый** (Yellow), **Синий** (Blue) и **Красный** (Red). По умолчанию установлен вариант **Нет цвета** (No Color), чтобы блоки на этой вкладке работали, если робот зафиксировал черный, белый или коричневый цвет. Следующий шаг — добавление блоков на каждую вкладку последовательности.

Нет цвета: спокойствие!

При отсутствии зеленого, желтого, синего или красного цветов робот ANTY просто сидит и издает звук, похожий на щебетание. Поместите блок **Звук** (Sound) на вкладку последовательности **Нет цвета** (No Color), как показано на рис. 13.8. Обратите внимание, что файл **Insect chirp** воспроизводит два звука и после воспроизведения второго датчик выполняет повторные измерения.

Зеленый: безопасность!

Когда робот ANTY чувствует зеленую траву на широком поле, он знает, что здесь можно безопасно вздремнуть.

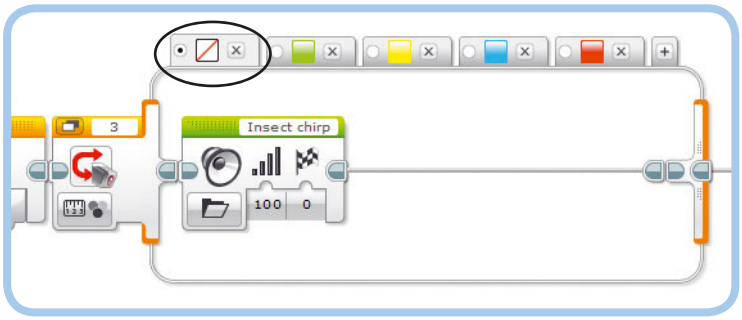


Рис. 13.8. Робот ANTY издает звук, похожий на щебетание, если не обнаруживает объекты зеленого, желтого, синего или красного цвета

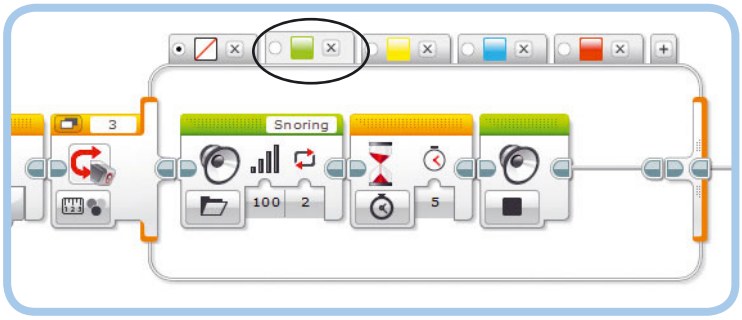


Рис. 13.9. Робот ANTY переходит в спящий режим, когда обнаруживает зеленый объект

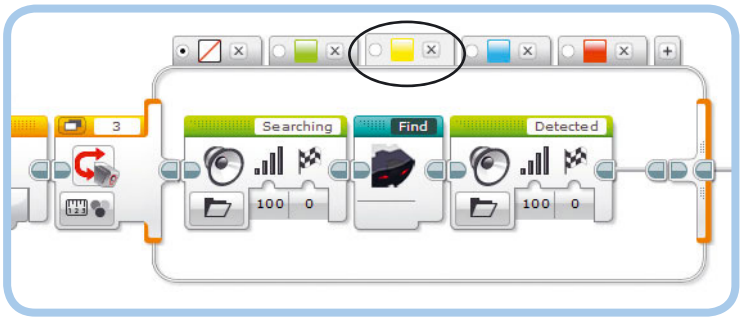


Рис. 13.10. Робот ANTY определяет местонахождение инфракрасного маяка и двигается к нему, если датчик цвета обнаруживает желтый цвет

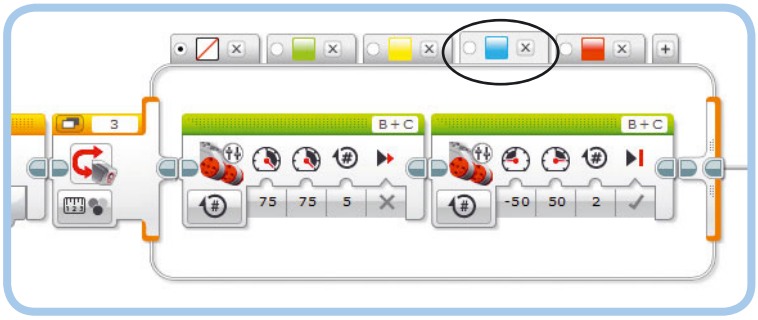


Рис. 13.11. Робот ANTY убегает при виде синего объекта. Обратите внимание, что моторы совершают фиксированное число вращений так, чтобы ноги робота остановились четко в противоположных позициях и робот был готов снова убегать

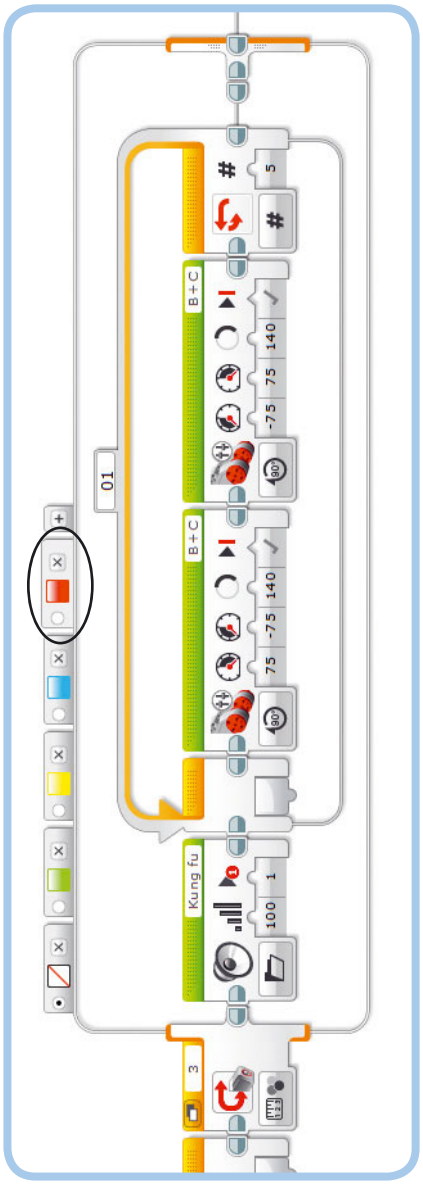


Рис. 13.12. Робот ANTY агрессивно встряхивается, когда «видит» красный предмет

Программа завершена. Запустите ее и протестируйте все варианты поведения робота ANTY, поднося предметы разного цвета к датчику цвета.

Добавьте блоки, которые инструктируют робота перейти в спящий режим, на вкладку **Зеленый** (Green), как показано на рис. 13.9.

Желтый: питание!

Вид желтого объекта вызывает у робота ANTY голод, заставляя искать пищу (маяк) с помощью контейнера **Find**, как показано на рис. 13.10. Если вы еще не создали этот контейнер «Мой блок», прочитайте раздел «Поиск пищи» ранее в этой главе.

ПРАКТИКУМ № 70: ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ!

Сложность: **Время:**

Можно ли создать для робота ANTY программу дистанционного управления? Запрограммируйте робота передвигаться в любом направлении с помощью инфракрасного маяка. Используйте техники, изученные в главе 8.

СОВЕТ **Сделайте так, чтобы при нажатии кнопки в верхней части инфракрасного маяка программа запускала контейнер «Мой блок» с именем Opposite. Нажмите эту кнопку, чтобы изменить положение ног робота, если они расположены неправильно.**

ПРАКТИКУМ № 71: НОЧНОЕ СОЗДАНИЕ!

Сложность: **Время:**

Измените программу ObstacleAvoid, скорректировав скорость передвижения робота ANTY в зависимости от времени суток. Пусть он перемещается ночью с обычной скоростью, на рассвете и в сумерках — крадется, а днем — сидит неподвижно. Используйте режим **Яркость внешнего освещения** (Ambient Light Intensity) для определения времени суток.

СОВЕТ **Протестируйте программу в темной комнате, имитируя ночь. Включите только настольную лампу, чтобы имитировать предрассветное время, а затем включите все освещение, чтобы имитировать дневной свет. Как рассчитать пороговые значения для программы?**

Синий: опасность!

Синие объекты указывают на присутствие хищников. Лучший вариант для робота ANTY — просто убежать с помощью блоков **Независимое управление моторами** (Move Tank), как показано на рис. 13.11. Отображенные на нем блоки заставляют робота сделать пять шагов вперед, а затем два шага влево.

Красный: агрессия!

Красный цвет вызывает у робота ANTY агрессию и заставляет его встряхиваться, чтобы отпугнуть врагов, как показано на рис. 13.12.

Дальнейшее изучение

С помощью EV3 можно конструировать не только транспортные средства и механизмы, но и роботизированных животных и иных существ. В этой главе вы построили шестиногого шагающего робота ANTY. Вы узнали, как запрограммировать его передвигаться с помощью двух больших моторов и датчика касания для определения абсолютного положения каждого мотора.

Вдобавок к конструированию шагающего робота вы создали программу, управляющую его поведением так, что робот ANTY может выполнять различные действия

ПРАКТИКУМ № 72: ГОЛОДНЫЕ РОБОТЫ!

Сложность: **Время:**

Можно ли создать программу, которая заставит робота передвигаться и огибать предметы, пока он не «проголодается»? «Проголодавшись», он должен будет подходить к инфракрасному маяку в поисках «пищи». Повторив эти действия трижды, робот должен перейти в спящий режим.

СОВЕТ **Определяйте, насколько робот ANTY «голоден», по тому, сколько оборотов сделали его моторы с момента последнего «обеда» (50 шагов — хорошее пороговое значение). После того как робот найдет пищу, сбросьте значение датчиков вращения на 0, чтобы восстановить его уровень «энергии».**

в зависимости от сигнала датчика, имитируя поведение настоящего животного.

Теперь попробуйте выполнить задания из практикумов, чтобы еще лучше изучить конструкцию этого робота. Насколько крутое существо вы сможете сконструировать?

СДЕЛАЙ САМ № 22: РОБОТ-ПАУК!

Сборка: **Программирование:**

Сможете ли вы создать робота-паука? Уберите у робота ANTY хвост и переставьте инфракрасные датчики и модуль EV3 так, чтобы тело робота стало более похожим на паучье. Вам нужно будет добавить в конструкцию еще две ноги, ведь у пауков их восемь. Когда все будет готово, прикрепите к телу и ногам робота-паука декоративные элементы, чтобы он выглядел более реалистичным и страшным.

СОВЕТ **Если вы не знаете, как заставить две дополнительные ноги двигаться, просто добавьте неподвижные элементы, похожие на ноги, а робот пусть передвигается на шести рабочих ногах. В этом случае убедитесь, что две новые ноги при ходьбе не касаются земли или других ног.**

СДЕЛАЙ САМ № 23: УСИКИ!

Сборка: **Программирование:**

Насекомые способны ощущать окружающую среду с помощью усиков, которые обычно есть у них на голове. Можно ли приделать роботу ANTY усики, которыми он будет определять наличие препятствий и отходить при приближении к ним вплотную? Датчик касания уже используется, но можете ли вы придумать способ определения контакта с предметом с помощью других датчиков?

СОВЕТ **Сконструируйте усики таким образом, чтобы, когда робот ANTY наталкивается на препятствие, они бы нажимали кнопки на инфракрасном маяке (рис. 13.13), и примените инфракрасный датчик для обнаружения нажатия кнопок.**

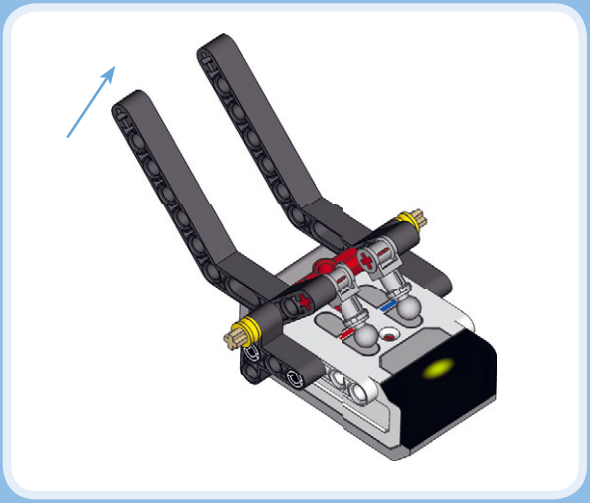


Рис. 13.13. Вы можете использовать удаленный инфракрасный маяк и инфракрасный датчик вместо датчика касания. Касание усика (синяя стрелка) приводит к нажатию кнопки на маяке. С помощью инфракрасного датчика определяйте, какая из кнопок (или обе) нажата

СДЕЛАЙ САМ № 24: ЖУТКИЕ КОГТИ!

Сборка: **Программирование:**

Можно ли улучшить конструкцию робота ANTY, добавив ему когти, чтобы он мог хватать предметы и перетаскивать их свое гнездо? С помощью среднего мотора когти должны раскрываться и сжиматься. В качестве дополнительного задания запрограммируйте робота искать инфракрасный маяк, подходить к нему и хватать его когтями.

СОВЕТ **Удалите или измените голову робота, чтобы освободить место для среднего мотора.**

ЧАСТЬ V

Разработка сложных программ

14

SK3TCHBOT: использование шин данных

В пятой части книги вы научитесь использовать шины данных (в этой главе), блоки операций с данными (глава 15) и переменные (глава 16), чтобы разрабатывать улучшенные программы для роботов. В главе 17 показано, как комбинировать эти методы и средства, чтобы создать большую программу, позволяющую запускать игру наподобие «Волшебного экрана» на модуле EV3. Вы сможете делать все это с помощью робота под названием SK3TCHBOT, который изображен на рис. 14.1.

В предыдущих главах вы выполняли настройку программных блоков путем ввода необходимых параметров вручную. Одним из основных понятий в данной главе является то, что блоки могут настраивать друг друга, передавая данные с одного блока на другой с помощью *шин данных*. Например, один блок может измерять значение параметра **Приближение** (Proximity) инфракрасного датчика и отправлять результат в блок **Большой мотор** (Large Motor). Блок **Большой мотор** (Large Motor) может использовать полученное значение для установки скорости мотора. Как следствие, обороты мотора будут низкими при малых значениях параметра **Приближение** (Proximity) (значение 27% приводит к показателям скорости в 27%) и высокими — при высоких (85% приводят к 85% скорости).

Эта глава научит вас создавать программы, которые используют шины данных. Поначалу это может показаться сложным, но, как только вы проработаете описанные программы и выполните практикумы в конце глав, вы овладеете всеми этими способами программирования!

Рис. 14.1. Вы будете использовать SK3TCHBOT, для того чтобы изучить множество новых способов программирования, и для игры в «Волшебный экран» на экране модуля EV3

Сборка робота SK3TCHBOT

Вы будете использовать робота SK3TCHBOT для тестирования своих программ с помощью шин данных и переменных. Робот состоит из модуля EV3, трех датчиков и двух больших моторов, которые вы будете использовать как вводы и выводы для своих программ, демонстрирующих шины данных в действии. В главе 17 вы создадите программу, которая превращает SK3TCHBOT в устройство, подобное «волшебному экрану», позволяя вам создавать рисунки на экране поворотами стрелок ввода, прикрепленных к моторам.

Соберите робота SK3TCHBOT, следуя инструкциям далее, но сначала выберите необходимые детали, как показано на рис. 14.2.

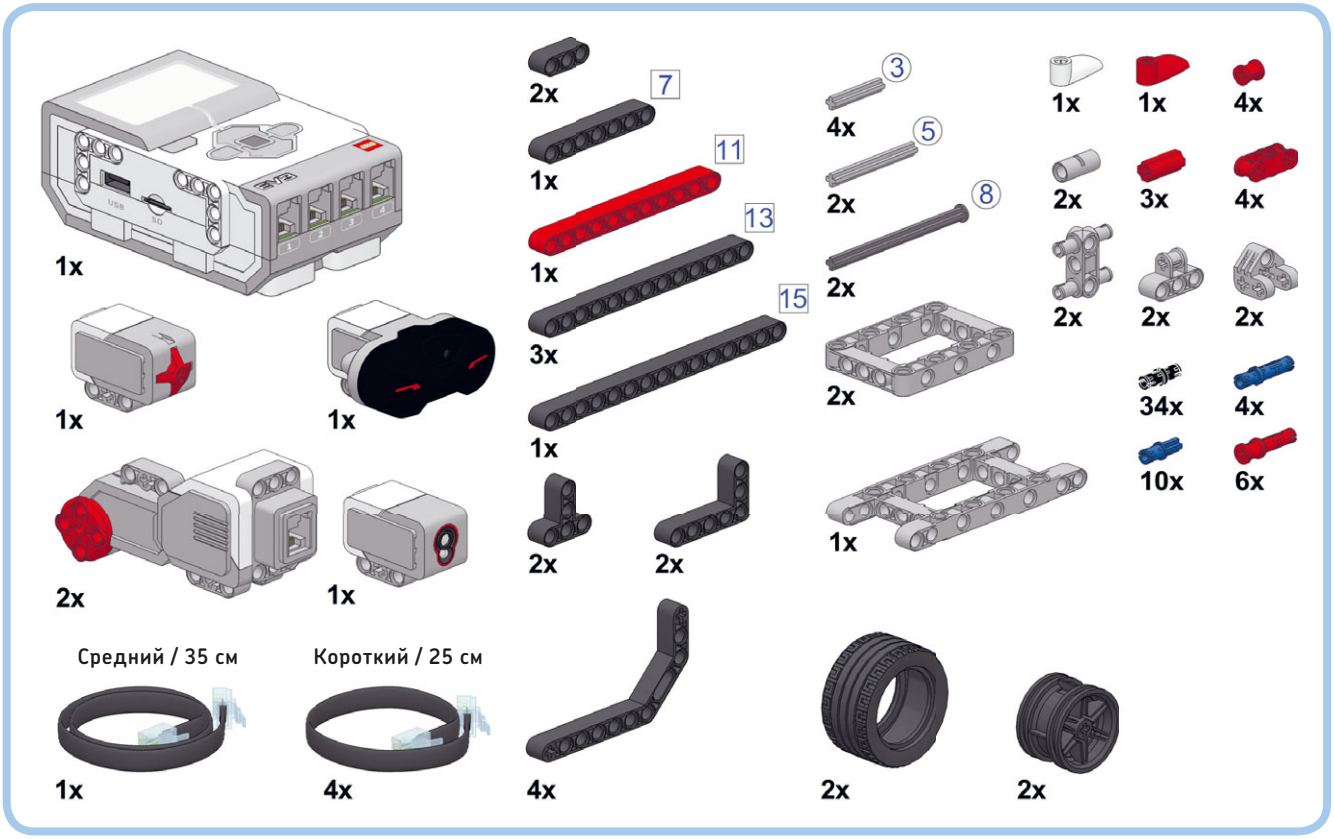
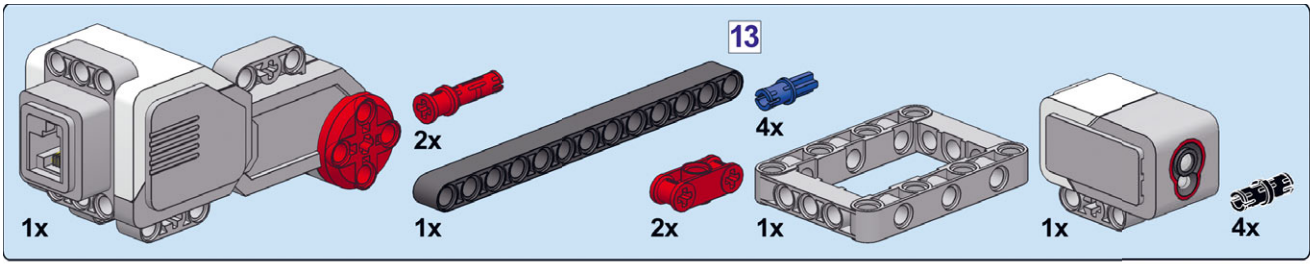
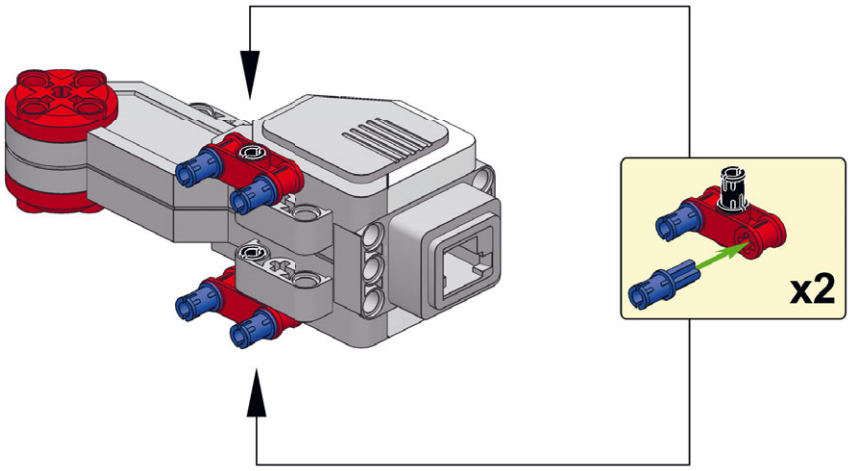


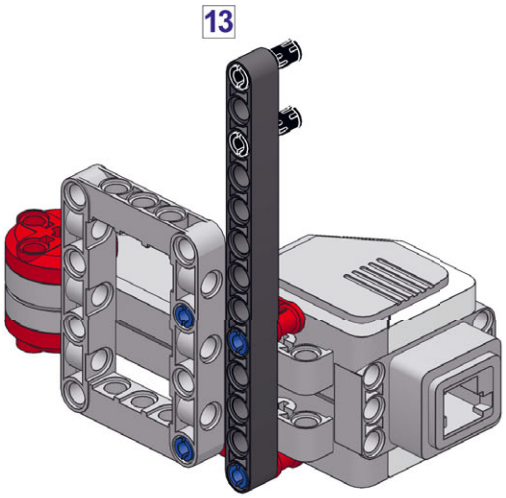
Рис. 14.2. Детали, необходимые для сборки робота SK3TCHBOT



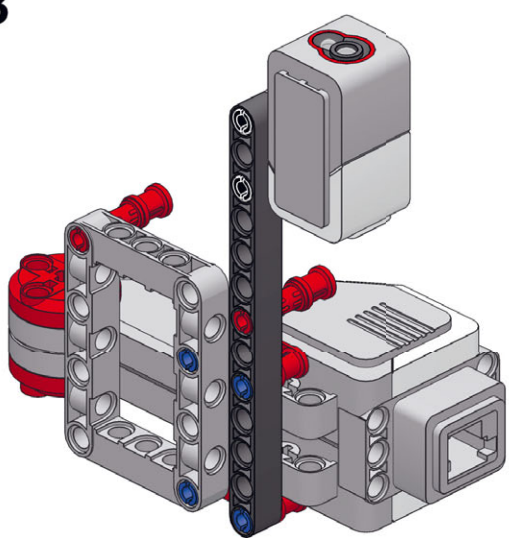
1

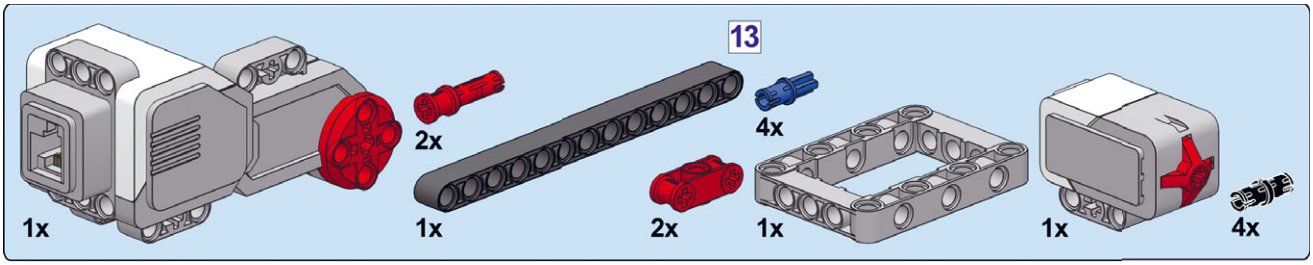


2

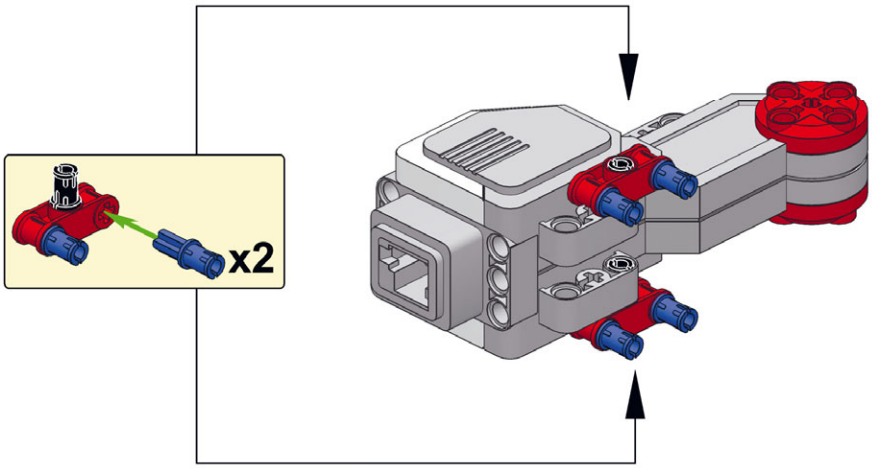


3

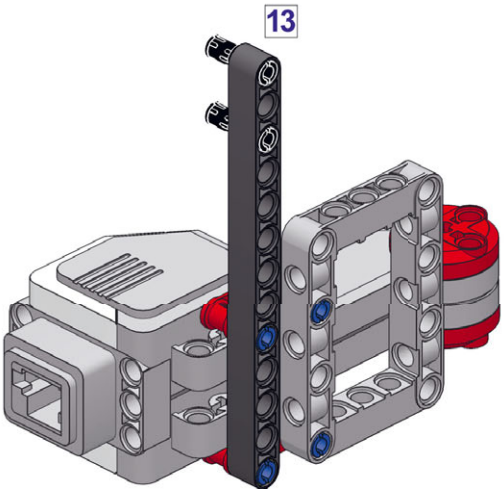




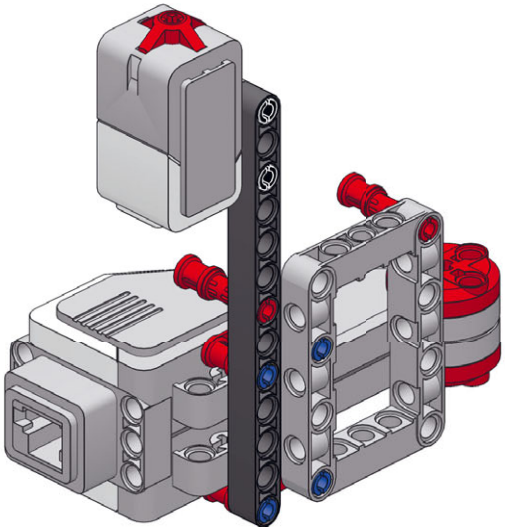
1

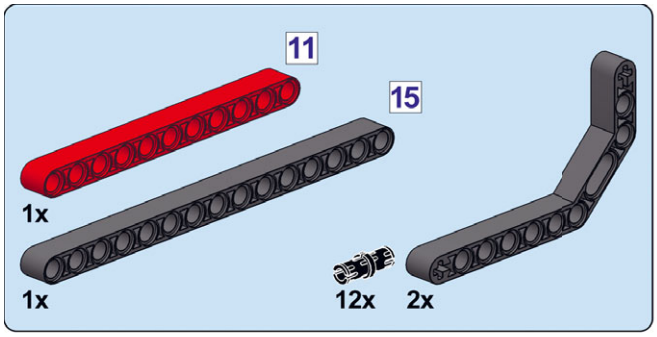


2

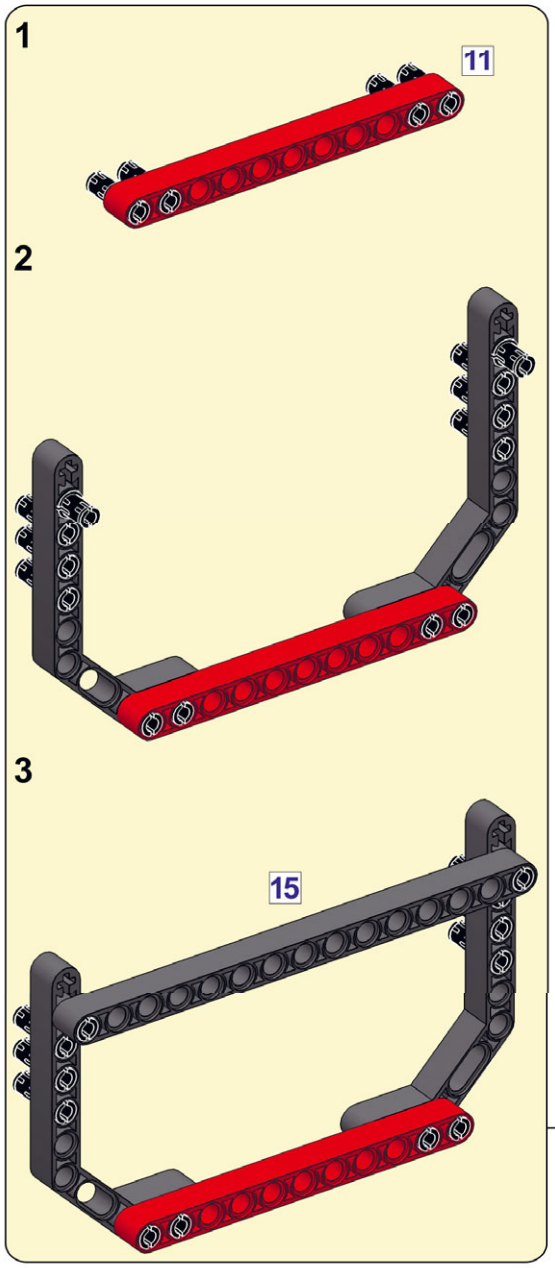


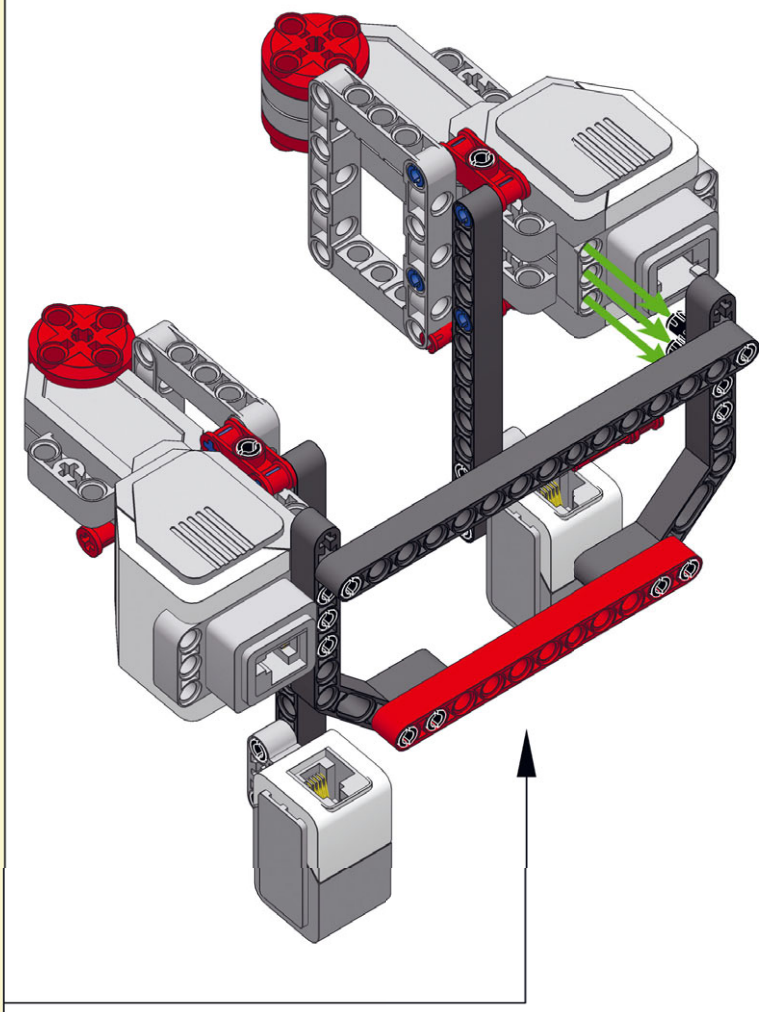
3

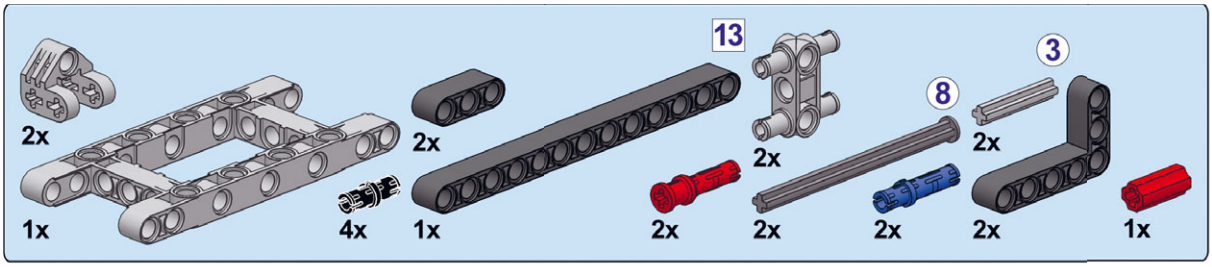




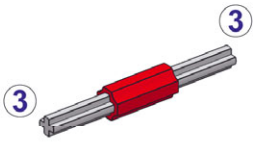
4



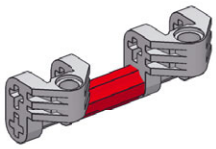




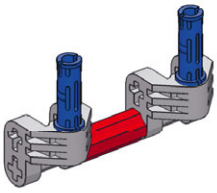
1



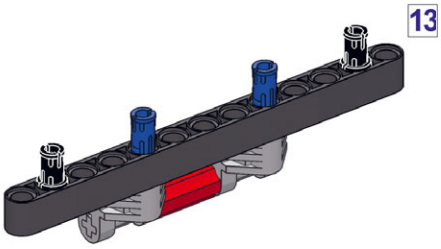
2



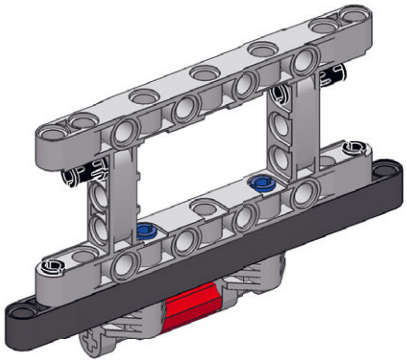
3



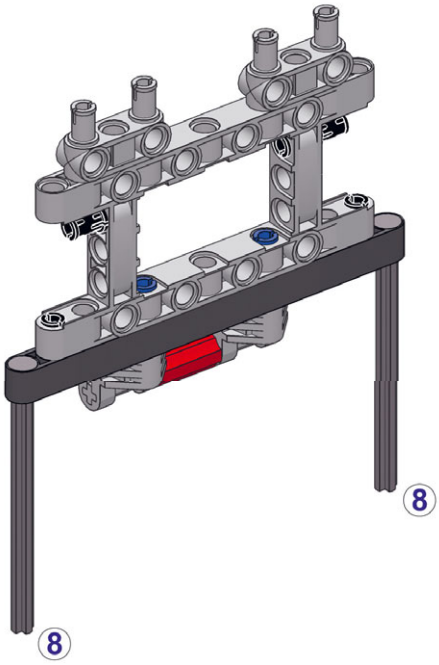
4



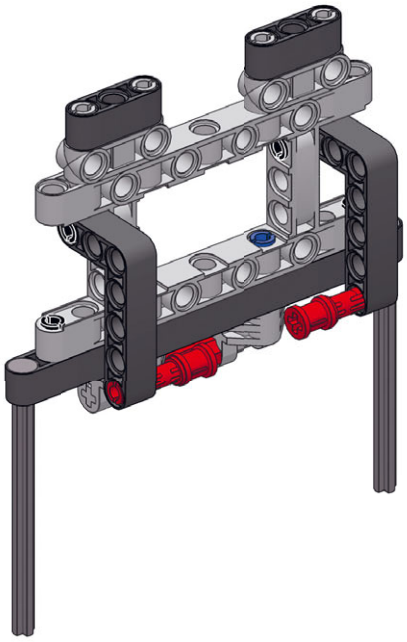
5

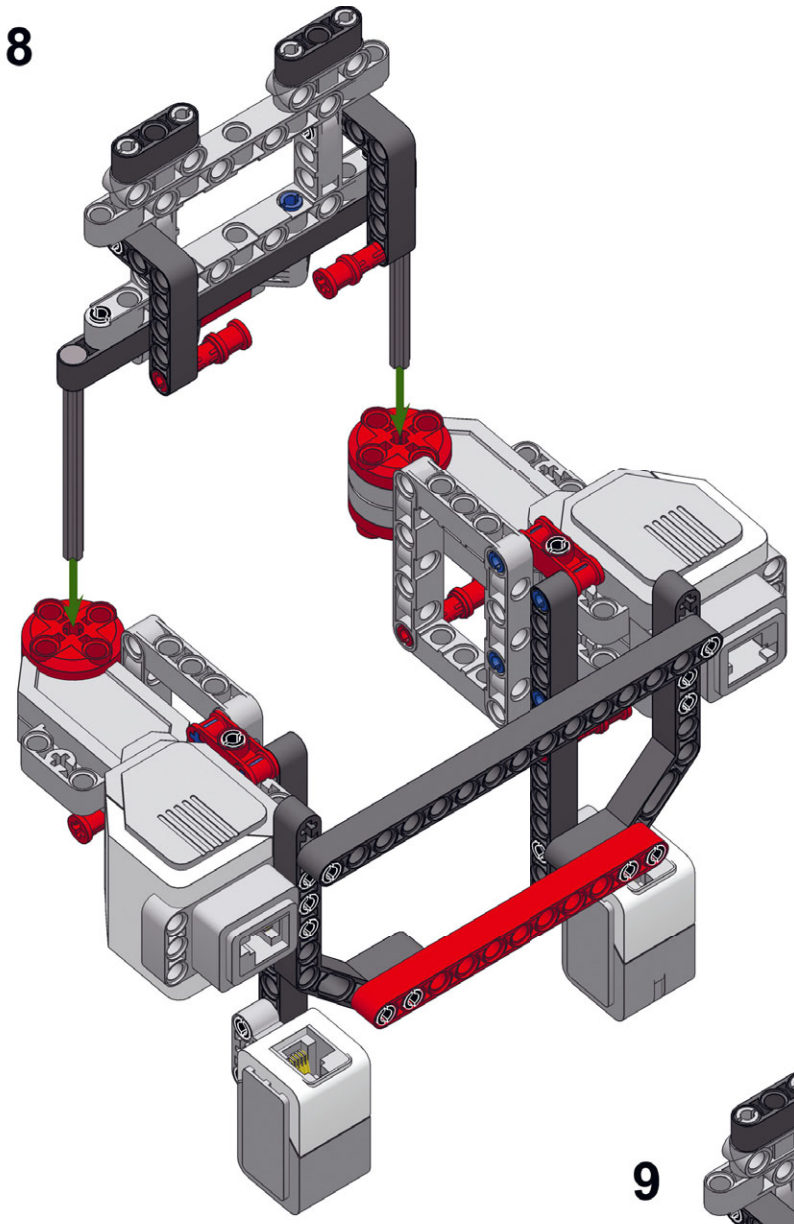


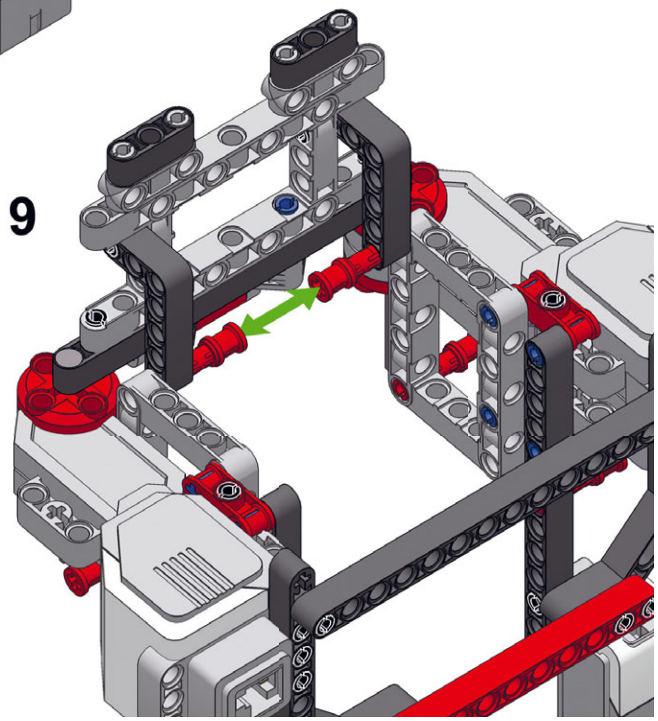
6

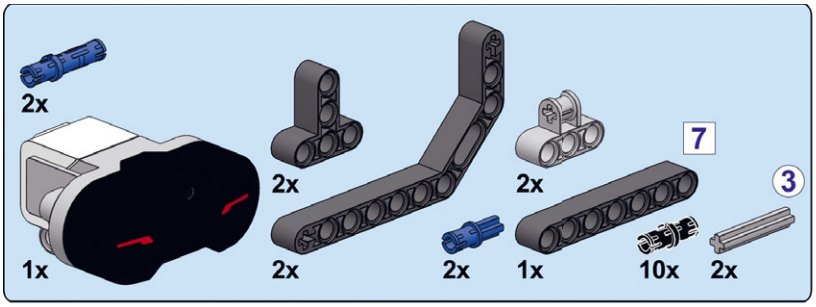


7

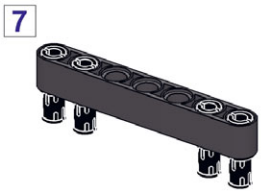



8

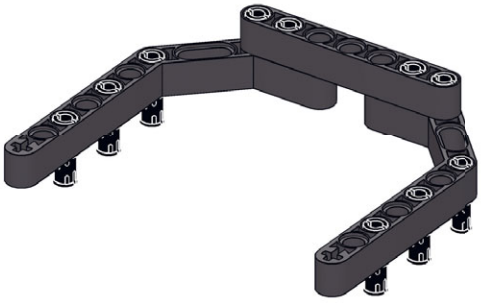

9



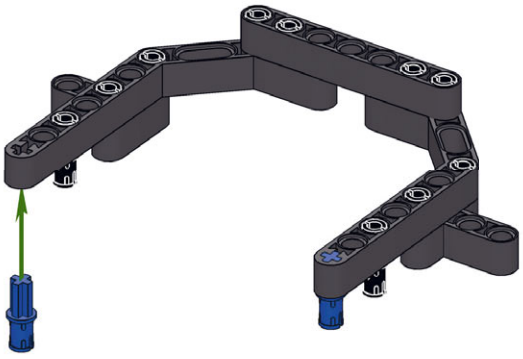
1



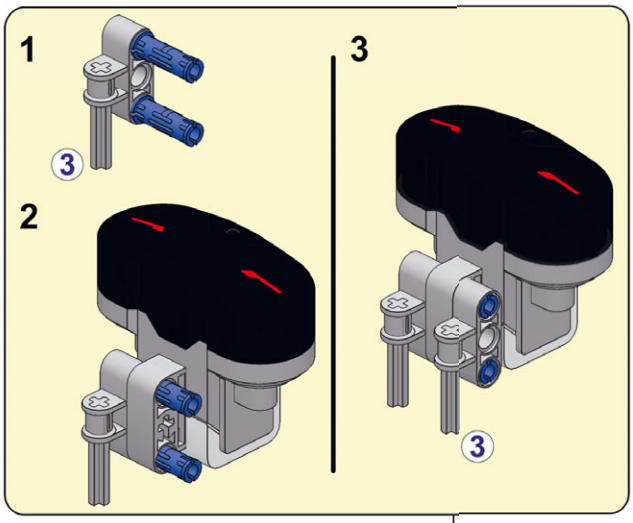
2

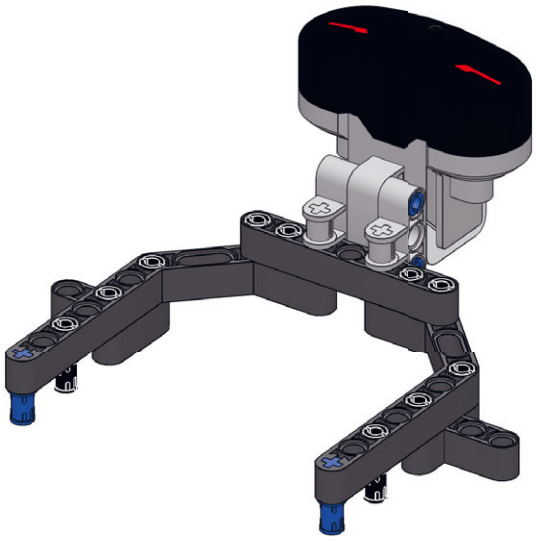


3

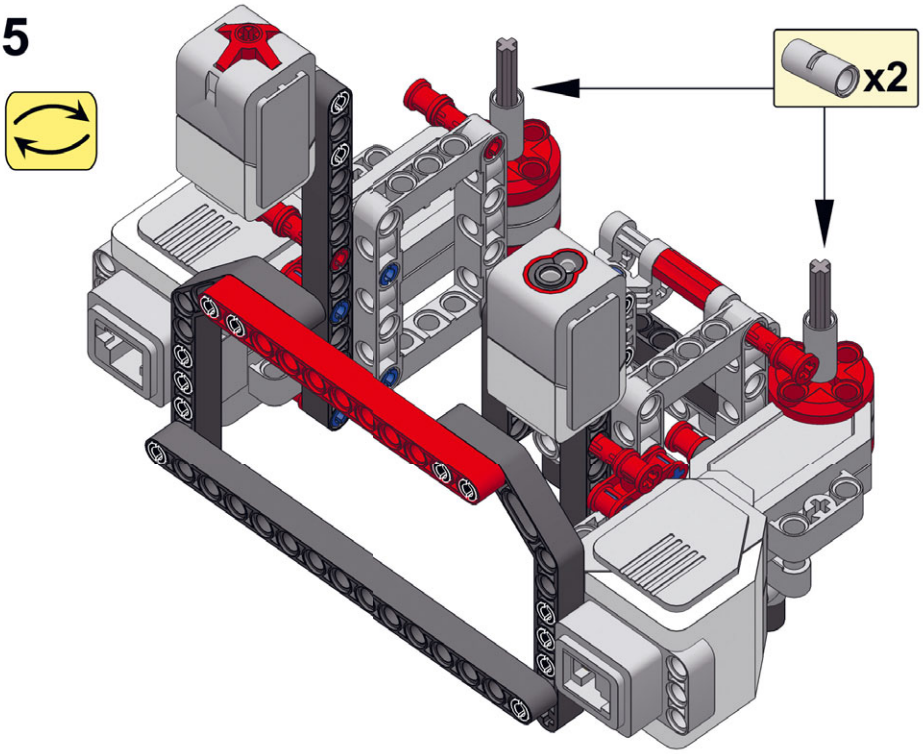


4

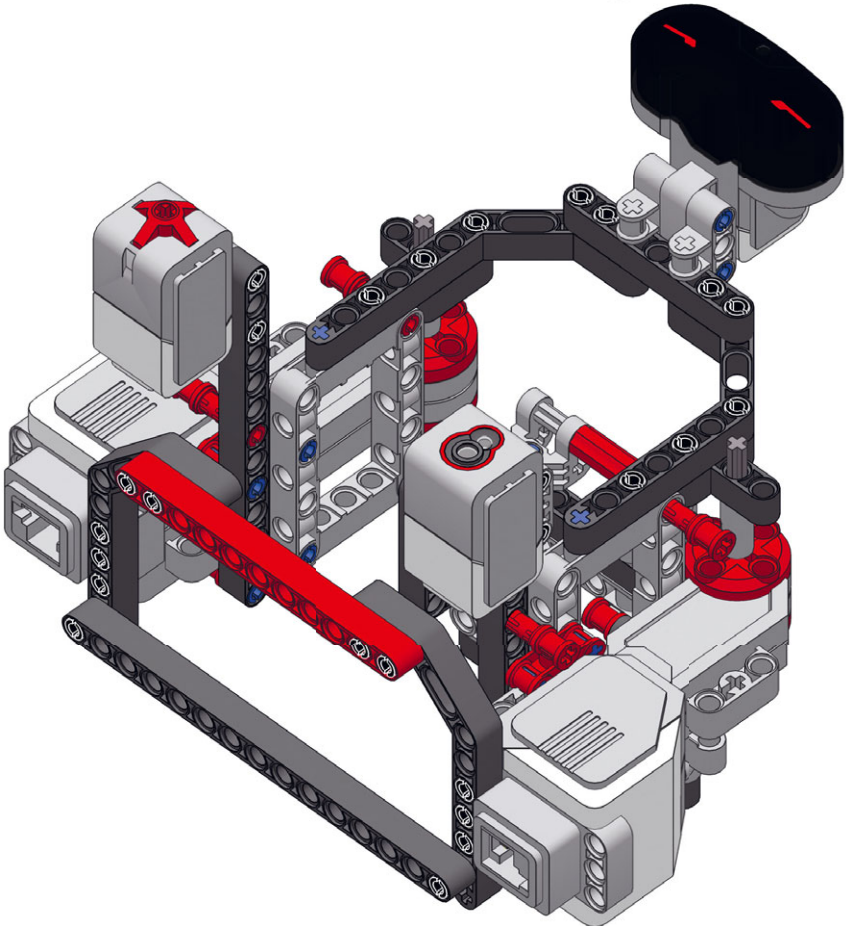


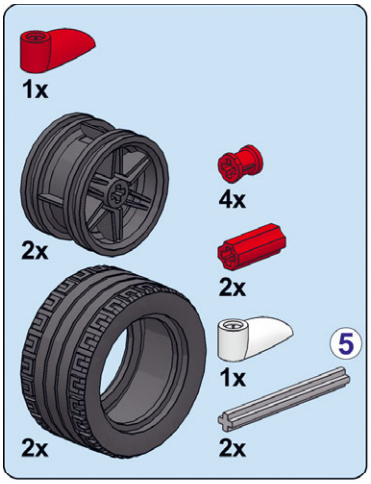


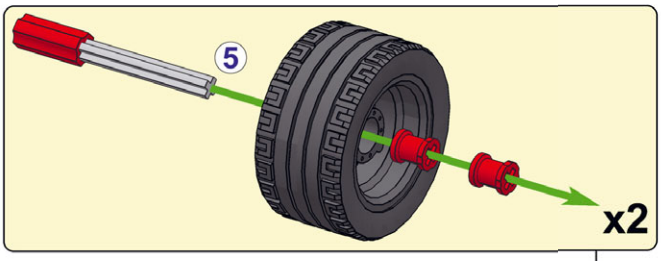
5



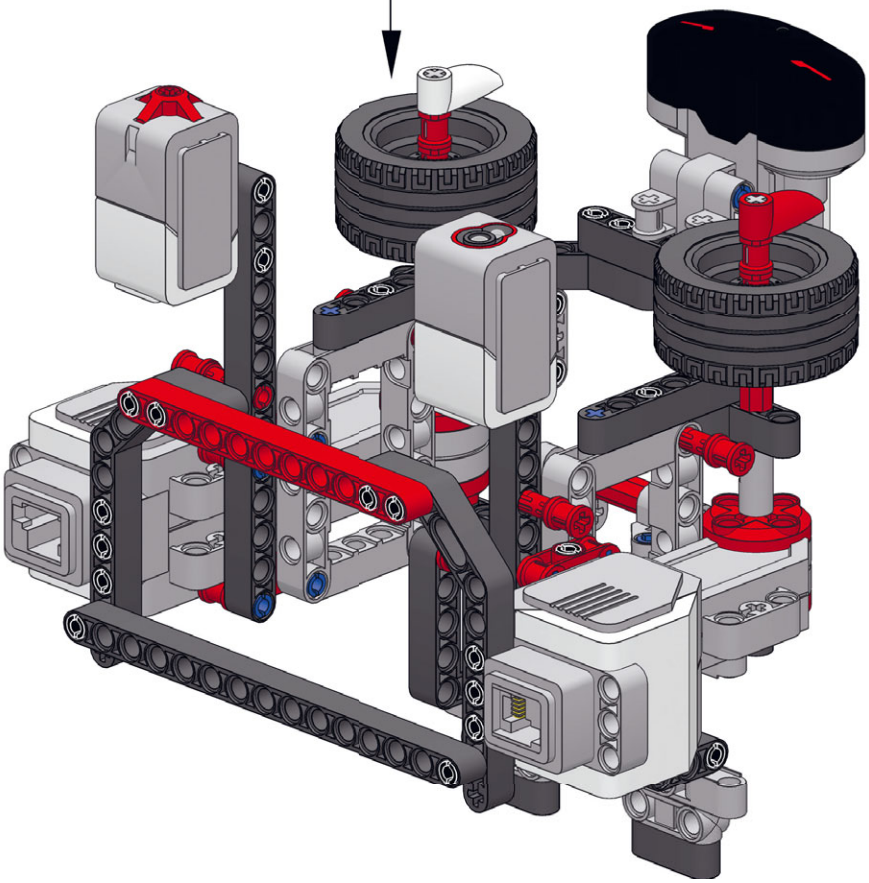
6

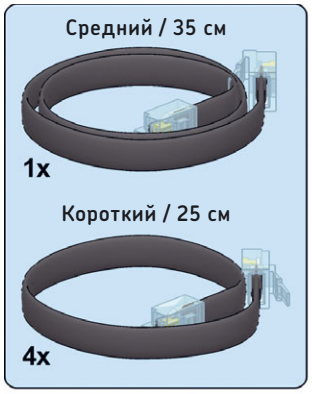




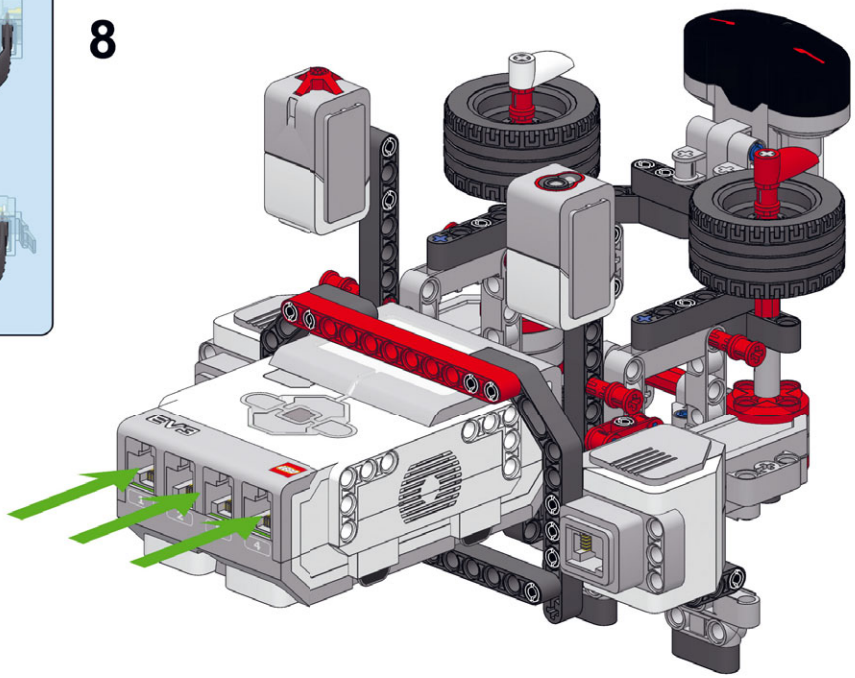


7



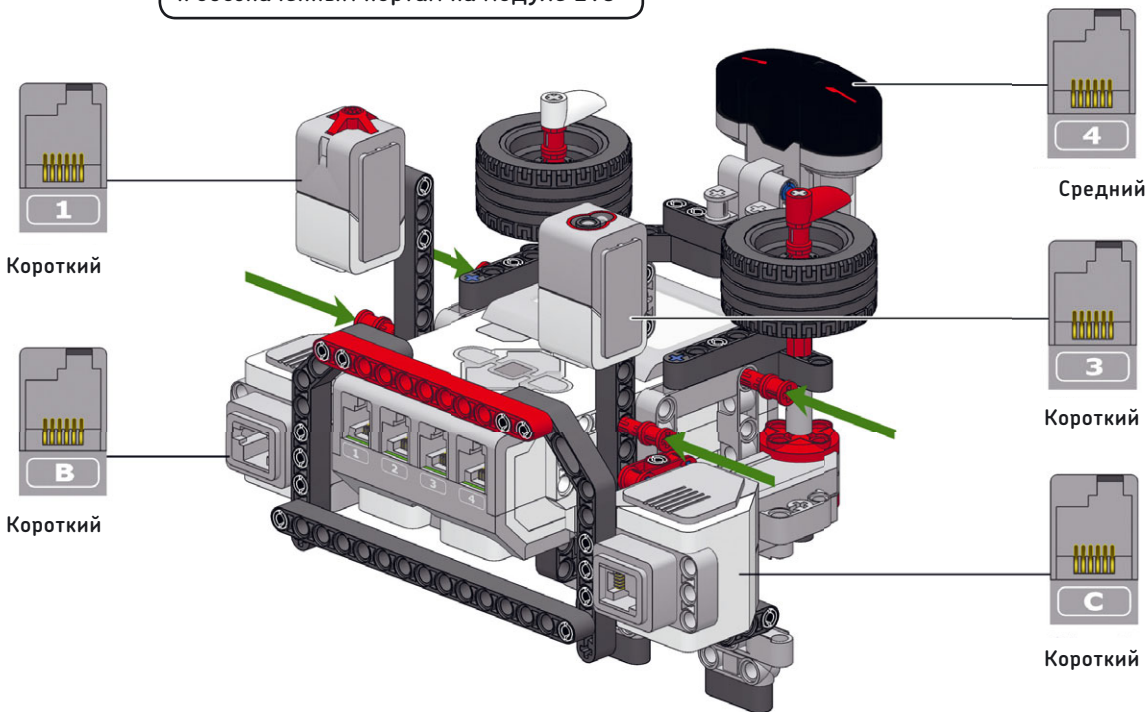


8



9

Вставьте штифты красного цвета в модуль EV3, как показано на рисунке. Затем подключите моторы и датчики к обозначенным портам на модуле EV3



Начало работы с шинами данных

Чтобы увидеть, как работают шины данных, вы создадите небольшую программу, которая инструктирует SK3TCHBOT воспроизводить звук, а затем вращать белую стрелку в течение трех секунд. Параметр **Мощность** (Power) мотора будет настраиваемым, и, следовательно, скорость будет соответствовать значению, фиксируемому инфракрасным датчиком: если значение датчика равно 27%, то и скорость мотора будет 27%; а если датчик считывает 85%, то и скорость мотора будет равна 85%; и так далее.

Вы примените блок **Инфракрасный датчик** (Infrared Sensor), чтобы считывать данные датчика. О блоках датчиков вы узнаете позже. Создайте новый проект под названием SK3TCHBOT-Wire с программой FirstWire, как показано на рис. 14.3 и 14.4, и загрузите ее в своего робота.

Следует отметить, что белая стрелка вращается с различной скоростью каждый раз, когда вы запускаете программу. Например, если вы будете держать руку в непосредственной близости от датчика и запустите программу, колесо будет вращаться медленно. Но оно должно вращаться быстрее, если вы снова запустите программу, а руку отведете на некоторое расстояние.

Поздравляем! Вы только что создали свою первую программу, использующую шины данных. Давайте рассмотрим, как работает каждый блок. Первый блок, **Звук** (Sound), просто проигрывает звук. Как только воспроизведение звука закончено, блок **Инфракрасный датчик** (Infrared Sensor) выполняет одно измерение (скажем, он считывает значение приближения, равное 27%). Желтая шина данных передает измерение датчика в блок **Большой мотор** (Large Motor), который инструктирует мотор B вращаться в течение

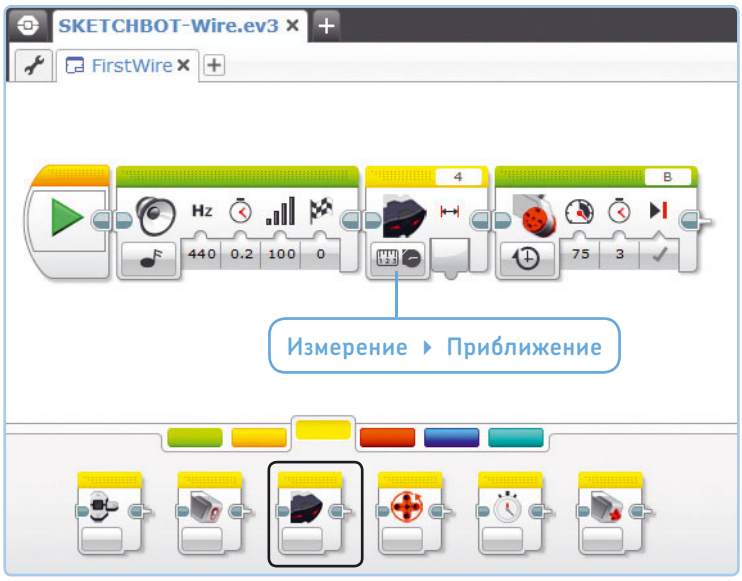


*Рис. 14.3. Шаг 1. Разместите все необходимые блоки в области программирования и скомпонуйте их, как показано на рисунке. Вы найдете блок **Инфракрасный датчик** на желтой вкладке палитры программирования. Убедитесь, что выбран режим **Измерение ▸ Приближение***

ПРАКТИКУМ № 73: ЗВУК НА РАССТОЯНИИ!

Сложность: **Время:**

Удалите блок **Большой мотор** (Large Motor) из программы FirstWire и вставьте вместо него блок **Звук** (Sound), который настроен на произнесение слова «Привет»*. Перетащите шину данных из вывода блока **Инфракрасный датчик** (Infrared Sensor) к вводу **Том**** (Volume) блока **Звук** (Sound). Так вы запрограммируете робота произнести слово «Привет» тихо, когда вы находитесь близко, или громко, когда он определит, что вы далеко.

* Пункт **Hello** в каталоге звуков Lego.

** В русской версии программы слово Volume ошибочно переведено как «Том». Правильно — Громкость.

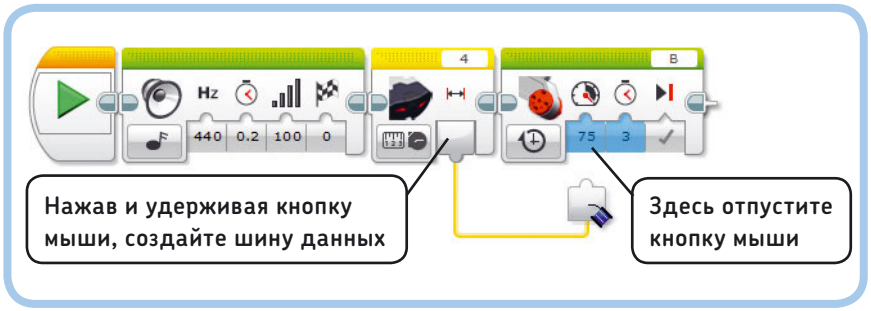


Рис. 14.4. Шаг 2. Нажав и удерживая кнопку мыши, перетащите желтую шину данных с вывода одного блока на ввод другого, как показано на рисунке. Затем запустите готовую программу

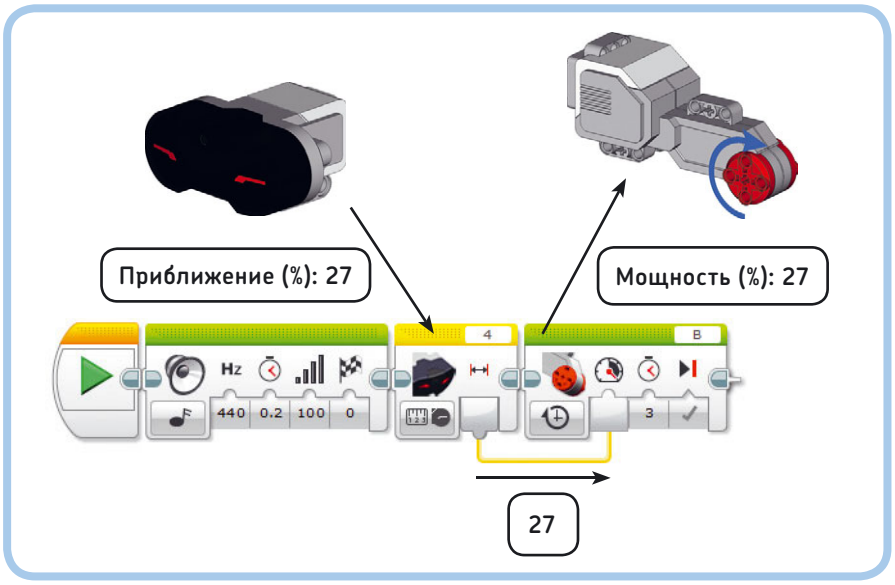


*Рис. 14.5. Блок **Инфракрасный датчик** считывает данные датчика и отправляет полученное значение по шине данных на блок **Большой мотор**, который использует это значение для установки скорости мотора*

3 секунд. Настройка параметра **Мощность** (Power) блока **Большой мотор** (Large Motor) зависит от значения, переданного с помощью шины данных; в данном случае это 27%. На рис. 14.5 можно увидеть, что именно происходит.

Работа с шинами данных

Как вы увидели, шины данных используются для передачи информации между блоками. Данные всегда передаются из *вывода* на *ввод*, как показано на рис. 14.6. В программе, используемой в качестве примера, шина данных передает значение датчика из вывода **Приближение** (Proximity) блока **Инфракрасный датчик** (Infrared Sensor) на ввод **Мощность** (Power) блока **Большой мотор** (Large Motor).

Обратите внимание, что шина данных скрывает значение, которое вы изначально присвоили параметру **Мощность** (Power) (**75**). Программа игнорирует это значение и вместо него использует значение, полученное из шины данных. С другой стороны, так как вы не подключили шины данных к другим вводам (**Секунды** (Seconds) и **Тормозить в конце** (Brake at End)) блока **Большой мотор** (Large Motor), они работают нормально. Так, вы присвоили значение **3** параметру **Секунды** (Seconds), которое заставляет мотор вращаться в течение 3 секунд.

А теперь давайте рассмотрим некоторые другие свойства шин данных.

Просмотр значения в шине данных

Можно просмотреть значение, передаваемое шиной данных, наведя мышь на шину во время работы программы, как

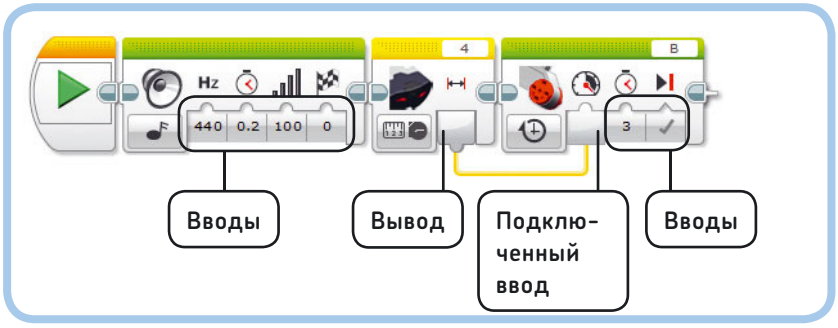


*Рис. 14.6. Шина данных передает данные из вывода (блока **Инфракрасный датчик**) в ввод (блока **Большой мотор**)*

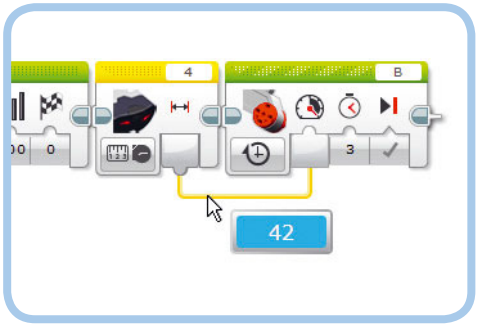


Рис. 14.7. Установите указатель мыши на шину данных, чтобы увидеть передаваемое значение

показано на рис. 14.7. Так вы сможете разобраться, что происходит в вашей программе.

Вы увидите значение в шине данных только в том случае, если робот подключен к компьютеру и вы запустили программу, нажав кнопку **Загрузить и запустить** (Download and Run) в окне программы EV3. Значение не отобразится, если запустить программу вручную, используя кнопки на модуле EV3.

Удаление шины данных

Удаление шины данных производится путем отключения ее правого конца от ввода, как показано на рис. 14.8.

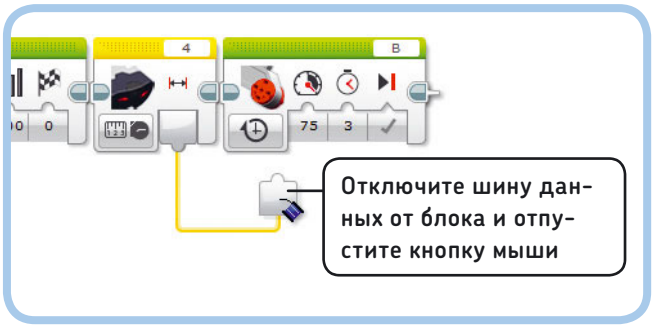


Рис. 14.8. Удаление шины данных

Выбор блока для подключения шины данных

При конфигурации программ с помощью шин данных у вас нет необходимости подключать один блок к другому, располагая их в непосредственной близости; вы можете соединять блоки и в том случае, если между ними установлены другие блоки, как показано на рис. 14.9. Программа WirePause считывает данные датчика, приостанавливается на 5 секунд, затем заставляет мотор вращаться. Скорость мотора основана на измерениях датчика, полученных при запуске программы.

Обратная работа невозможна, как показано на рис. 14.10. Вы не можете заставить блок **Большой мотор**

Рис. 14.10. Вы не можете подключить шину данных справа налево. Программное обеспечение EV3 не позволяет этого сделать

(Large Motor) использовать шину данных, потому что она не содержит значение до тех пор, пока не выполнено измерение. Другими словами, блок с вводом (в данном случае **Большой мотор** (Large Motor)) должен располагаться после блока с выводом (**Инфракрасный датчик** (Infrared Sensor)).

Использование нескольких шин данных

Вы можете использовать вывод для подключения нескольких шин данных, как показано на рис. 14.11. Программа MultiWire передает значение параметра **Приближение** (Proximity) на ввод **Мощность** (Power) одного блока **Большой мотор** (Large Motor) (мотора, подключенного к порту B (белая стрелка)) и на ввод **Градусы** (Degrees) второго блока **Большой мотор** (Large Motor) (другого мотора, подключенного к порту C (красная стрелка)). Например, если результат замера датчика составляет 34%, мотор B вращается на скорости 34%, а мотор C вращается на 34 градуса при скорости 75%.

С другой стороны, вы не можете подключить несколько шин данных к одному вводу. Если ввод уже *занят* (см. рис. 14.6), вы не можете подключить к нему другую шину. При возможности подключения более двух шин к одному и тому же вводу блок не знал бы, какое из значений следует использовать.

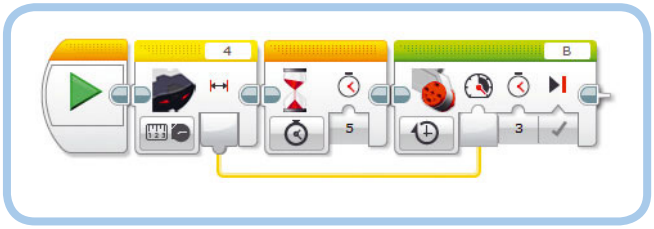


*Рис. 14.9. Программа WirePause выполняет один замер, когда запускается блок **Инфракрасный датчик** (и получает значение, скажем, 34%). Через 5 секунд мотор начинает вращаться на скорости 34%, даже если значение датчика за это время изменилось*

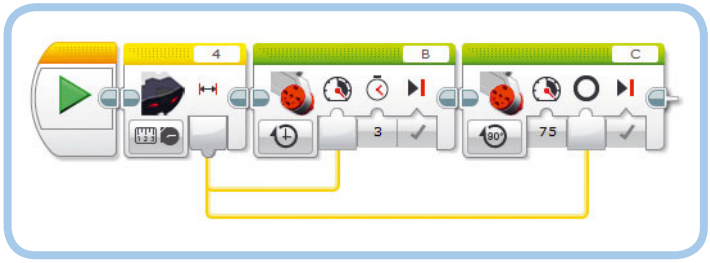


*Рис. 14.11. Две шины данных с одного и того же вывода в программе MultiWire. Инфракрасный датчик выполняет один замер приближения и передает полученное значение на вводы обоих блоков **Большой мотор***

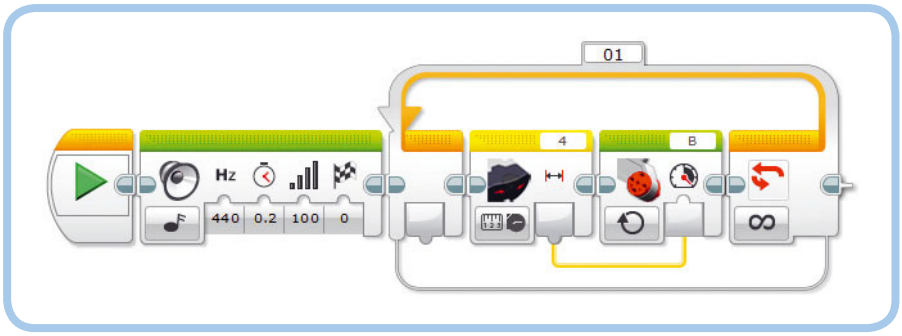


Рис. 14.12. Программа RepeatWire непрерывно подстраивает скорость мотора под измерения инфракрасного датчика

ПРАКТИКУМ № 74: ПОЛОСНАЯ ДИАГРАММА!

Сложность: **Время:**

Программа, изображенная на рис. 14.13, выстраивает полосную диаграмму на экране модуля EV3. Она не закончена, потому что отсутствуют необходимые шины данных. Как их подключить так, чтобы длина полосы представляла собой результат замера инфракрасного датчика?

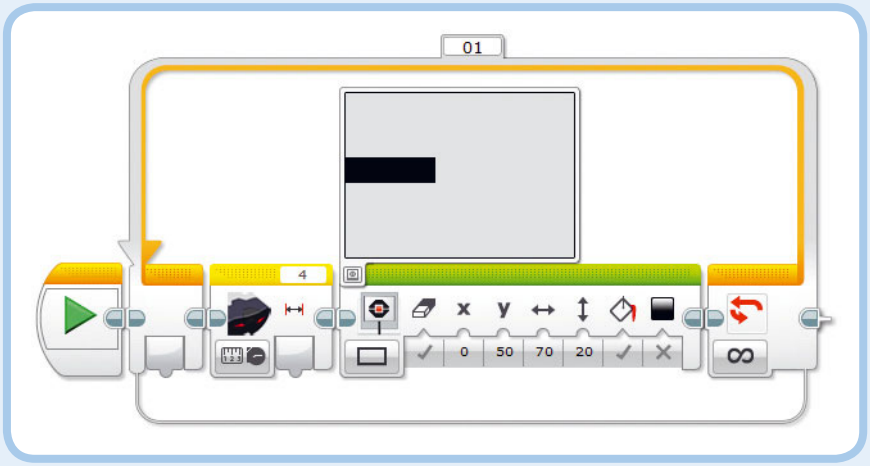


Рис. 14.13. Неполная программа для практикума № 74

ПРАКТИКУМ № 75: РАСШИРЕННАЯ ДИАГРАММА!

Сложность: **Время:**

Расширьте возможности программы, которую вы создали в практикуме № 74, добавив еще две полосные диаграммы, представляющие яркость отраженного света и внешнего освещения, измеряемую с помощью датчика цвета.

СОВЕТ **Используйте два блока Датчик цвета (Color Sensor). Добавьте блок Ожидание (Wait) в цикл, настроенный на ожидание в 0,2 секунды, чтобы предотвратить мерцание на экране. Какой из блоков Экран (Display) должен очищать содержимое экрана, прежде чем на нем появится новое изображение?**

Циклы и шины данных

В программе FirstWire мотор вращался 3 секунды на постоянной скорости на основе значения датчика, которое было получено перед самым началом вращения мотора. Сейчас вы расширите возможности данной программы, чтобы заставлять мотор непрерывно подстраивать свою скорость под значение датчика. Чтобы сделать это, поместите блоки **Инфракрасный датчик** (Infrared Sensor) и **Большой мотор** (Large Motor) в блок **Цикл** (Loop) и выберите для блока **Большой мотор** (Large Motor) режим работы **Включить** (On), как показано на рис. 14.12.

Когда вы запустите программу RepeatWire, скорость мотора должна изменяться постепенно, по мере того как вы медленно отодвигаете свою руку от инфракрасного датчика. Если вы внезапно поднесете руку вплотную к датчику, значение показаний датчика резко уменьшится и мотор быстро остановится.

Скорость постоянно меняется, потому что блокам в цикле почти не требуется времени для запуска. Блок **Инфракрасный датчик** (Infrared Sensor) считывает результаты измерения инфракрасного датчика, а блок **Большой мотор** (Large Motor) переключает мотор на заданную скорость. Затем программа возвращается к началу цикла, чтобы вновь запустить вложенные блоки, постоянно выполняя новые измерения с целью подстройки скорости.

Типы шин данных

До сих пор вы использовали шины данных только для передачи числовых значений, но существует три основных типа данных, которые могут передавать шины: *числовые*, *логические* и *текстовые*. У шин данных каждого типа разный цвет и форма ввода/вывода (круглый/желтый, треугольный/зеленый и квадратный/оранжевый), как показано в табл. 14.1. Форма вывода совпадает с формой ввода, к которому шина подключается, как деталь головоломки. Это сочетание помогает определить, шину данных какого типа можно подключить к конкретному вводу.

Числовые шины данных

Числовые шины данных (желтого цвета) передают числовую информацию, которая может включать целые (как, например, 0, 15 или 1427), десятичные (такие как 0,1 или 73,14) и отрицательные числа (к примеру, –14 или –31,47). Значение приближения инфракрасного датчика представляет собой пример такого числового значения, оно варьируется между 0 и 100.

Логические шины данных

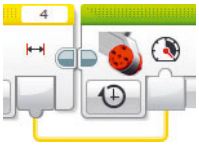
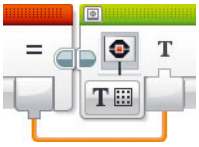
Логические шины данных (зеленого цвета) могут передавать только два значения: **Ложь** (False) или **Истина** (True). Такие шины часто используются, чтобы влиять на параметры, которые могут принимать только два значения, например **Очистить экран** (Clear Screen) блока **Экран** (Display). Датчик касания использует только два возможных значения для кнопки — отпущена и нажата, поэтому для передачи его состояния тоже применяется логическая шина данных. Блок **Датчик касания** (Touch Sensor) в режиме **Измерение ▸ Состояние** (Measure ▸ State) обеспечивает передачу через логическую шину данных значения **Истина** (True), если кнопка датчика нажата, и **Ложь** (False), если отпущена.

Создайте программу LogicClear, чтобы опробовать логическую шину данных в действии, как показано на рис. 14.14. Программа начинается с отображения двух сердитых глаз на экране модуля EV3. Двумя секундами позже робот определяет значение датчика касания и передает его в параметр **Очистить экран** (Clear Screen) блока **Экран** (Display) с помощью логической шины данных. Если кнопка датчика нажата (**Истина** (True)), экран очищается, прежде чем отобразится слово MINDSTORMS. Если кнопка не нажата (**Ложь** (False)), экран не очищается, и слово появится поверх предыдущего изображения.

Текстовые шины данных

Текстовые шины данных (оранжевого цвета) передают текст, например, в блок **Экран** (Display), чтобы вывести его на

Таблица 14.1. Основные типы шин данных

Тип			Пример значения
Числовое значение			–5 0 3,75 75
Логическое значение			Истина Ложь
Текст			Hello I'm a robot 5 apples*

* Допускаются только цифры, латинские буквы и специальные символы.

ПРАКТИКУМ № 76: ПЛАВНАЯ ОСТАНОВКА!

Сложность: **Время:**

Разработайте программу, с помощью которой мотор B вращается на максимальной скорости до тех пор, пока инфракрасный датчик не обнаружит поблизости объект. Используйте логическую шину данных, чтобы при нажатии кнопки датчика касания мотор останавливался резко, а при отпущенной кнопке — плавно.

СОВЕТ **Способом остановки мотора можно управлять с помощью параметра Тормозить в конце (Brake at End).**

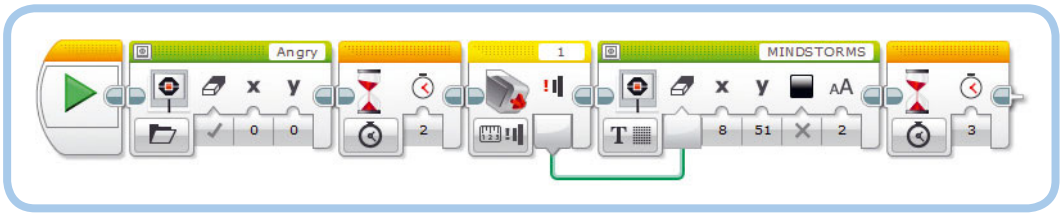


*Рис. 14.14. Программа LogicClear. Вы найдете блок **Датчик касания** среди программных блоков датчиков (желтая вкладка палитры программирования)*

экране модуля EV3. Текст может представлять собой слово или фразу, наподобие «привет, я — робот» или «5 яблок»*. Мы вернемся к шинам данных этого типа в главе 15.

Числовой и логический массивы

В дополнение к числовым, логическим и текстовым шинам данных, программное обеспечение EV3 поддерживает такие типы шин данных как *числовой массив* и *логический массив*. Числовой массив фактически является списком числовых значений, используемых для передачи нескольких чисел по одной шине. Например, вы могли бы использовать массив для отправки пяти последних показаний датчика в контейнер «Мой блок», который будет отображать их все на экране модуля EV3.

Аналогичным образом логический массив является списком логических значений. Мы не будем использовать здесь массивы, но вы можете поупражняться с ними в пробной программе на сайте **goo.gl/sgVGyj**, как только закончите чтение этой книги.

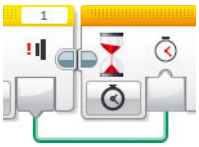
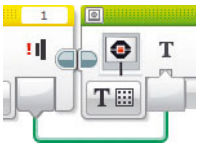
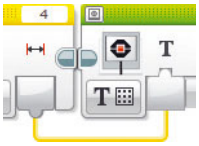
Конвертации шин данных

Как правило, вы должны подключать числовые шины данных к вводам, которые принимают числовые значения (закругленной формы), логические шины данных к логическим вводам (треугольной формы), а текстовые — к текстовым (квадратной формы).

Но программное обеспечение EV3 позволяет выполнить еще три подключения, как показано в табл. 14.2. Можно понять, какие подключения допустимы, просто взглянув на форму вводов. В некоторых случаях формы совпадают не точно, но тем не менее все равно подходят. Например, треугольный ввод вмещается в закругленный, и это означает, что вы можете управлять числовым вводом с помощью логической шины данных. С другой стороны, закругленный ввод не помещается в треугольный, значит, вы не можете

* На самом деле допускаются только цифры, латинские буквы и специальные символы.

Таблица 14.2. Конвертации шин данных

Из типа			В тип	Результат
Логический			Числовой	**Истина** (True) становится равна 1 **Ложь** (False) становится равна 0
Логический			Текстовый	**Истина** (True) становится равна 1 **Ложь** (False) становится равна 0
Числовой			Текстовый	Число конвертируется в формат, поддерживаемый блоком **Экран** (Display), чтобы его можно было отобразить

управлять логическим вводом с помощью числовой шины данных.

В трех случаях, которые показаны в табл. 14.2, данные, передаваемые с помощью шины, конвертируются в подходящий тип, чтобы программа сохраняла работоспособность. Давайте создадим две программы, чтобы разобраться, что это значит.

Конвертация логических значений в числовые

Программа ConvertWire (рис. 14.15) выполняет конвертацию логического значения в числовое. Блок **Ожидание** (Wait) требует числовое значение для параметра **Секунды** (Seconds), но вместо этого он получает логическое значение из блока **Датчик касания** (Touch Sensor). Схема работает, потому что программа конвертирует логическое значение в числовое. Значение **Истина** (True) становится 1, а **Ложь** (False) 0. Следовательно, если вы нажмете кнопку датчика, между звуковыми сигналами вы услышите паузу длиной в 1 секунду, а если не нажмете — паузы не будет.

Отображение чисел на экране модуля EV3

Когда вы используете блок **Экран** (Display) для отображения текста на экране модуля EV3, вы можете либо набрать что-то в текстовом поле, либо выбрать пункт **Проводной** (Wired), как показано на рис. 14.16. Выбор пункта **Проводной** (Wired) создает дополнительный ввод, обеспечивающий поддержку текстовой шины данных.

Программа предполагает, что вы подключите текстовую шину данных к вводу **Текст** (Text), но вы можете подключить и числовую шину данных.

Компьютеры наподобие модуля EV3 хранят числа и текстовые строки двумя разными способами. И поэтому, как правило, вы не можете отправить числовое значение в текстовый ввод. К счастью, программное обеспечение конвертирует числа в текстовый формат, поддерживаемый блоком **Экран** (Display). Эта конвертация позволяет отображать числа на экране модуля EV3, как это демонстрирует программа DisplayNumeric (рис. 14.17).

Программа постоянно обновляет значение датчика и отображает его на экране. Отображение значений — полезный метод тестирования программ. Конечно, вы уже знаете, как отслеживать значения датчика с помощью приложения Port View, но благодаря описанной выше особенности шин данных вы можете, к примеру, выполнять вычисления со значениями датчиков и отображать результаты в режиме реального времени. Вы увидите это в следующей главе.

В дополнение к текстовым и числовым шинам текстовый ввод позволяет подключать логические шины данных. Опять же модуль EV3 хранит логические значения иначе, но программа конвертирует их в текстовый формат, когда это необходимо. Значение **Истина** (True) заставляет блок **Экран** (Display) отображать 1, а **Ложь** (False) — отображать 0. Замените блок **Инфракрасный датчик** (Infrared Sensor) в программе DisplayNumeric блоком **Датчик касания** (Touch Sensor), чтобы опробовать это.

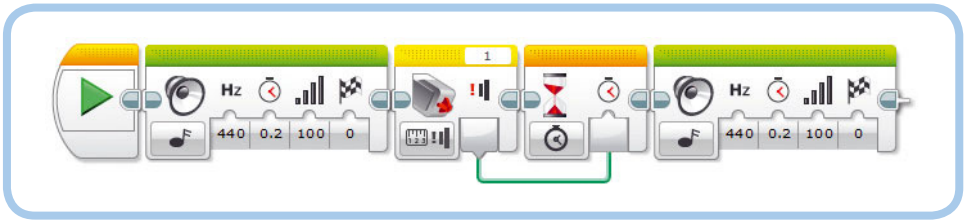


Рис. 14.15. Программа ConvertWire

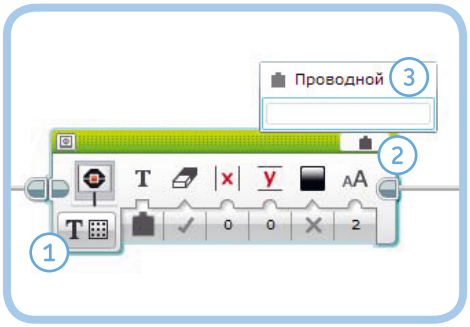


*Рис. 14.16. Перетащите блок **Экран** из палитры, выберите режим **Текст** ▸ **Сетка** (1), щелкните мышью по текстовому полю (2) и выберите пункт **Проводной** (3), чтобы отобразить ввод **Текст***

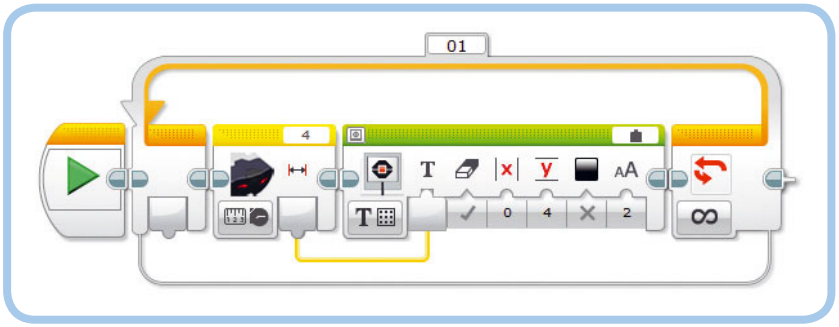


Рис. 14.17. Программа DisplayNumeric

Использование блоков датчиков

Во второй части книги вы научились работать с датчиками, создавая программы с блоками **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch). Обработкой значений датчиков занимаются *программные блоки датчиков*. Эти блоки полезны в том случае, если вы хотите получить значение датчика и передать его в другой блок с помощью шины данных, как вы это делали в программе FirstWire.

Для каждого датчика на вкладке **Датчик** (Sensor) палитры программирования доступен соответствующий блок, как показано на рис. 14.18. Каждый блок можно использовать в режиме **Измерение** (Measure) или **Сравнение** (Compare).

Режим Измерение

Блок датчика в режиме **Измерение** (Measure) выполняет одно измерение и передает полученное значение в другой блок с помощью шины данных. Вы определяете тип измерения путем выбора одного из режимов работы датчика. Например, вы можете передать значение инфракрасного датчика в блок **Большой мотор** (Large Motor), используя блок **Инфракрасный датчик** (Infrared Sensor) в режиме **Измерение** ▸ **Приближение** (Measure ▸ Proximity).

Все, что вы знаете о режимах работы и значениях датчиков на данный момент, применимо и для блоков датчиков. В табл. 14.3 приводятся сводные значения датчиков для каждого режима работы. Используйте таблицу в качестве справочной информации при разработке собственных программ, будь то блоки **Ожидание** (Wait), **Цикл** (Loop), **Переключатель** (Switch) или блоки датчиков.

Режим Сравнение

Как и в режиме **Измерение** (Measure), блок датчика в режиме **Сравнение** (Compare) выполняет одно измерение и выводит показания датчика с помощью шины данных. Кроме того, он сравнивает полученное значение с пороговым и выводит результат с помощью логической шины данных. Вывод **Результат сравнения** (Compare Result) передает значение **Истина** (True), если условие (например, «значение приближения составляет более 40%») истинно, и значение **Ложь** (False), если условие ложно.

Вы определяете условие посредством присвоения значения параметру **Пороговое значение** (Threshold Value) и выбора режима **Сравнение** (Compare) в настройках блока, как вы это делали для блоков **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch).

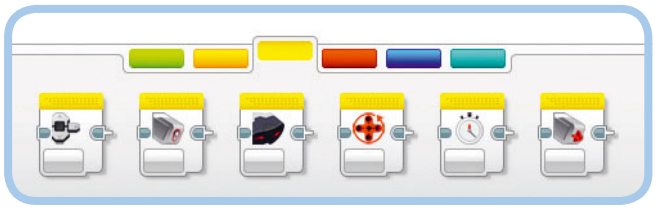


Рис. 14.18. Слева направо, блоки: ***Кнопки управления модулем, Датчик цвета, Инфракрасный датчик, Вращение мотора, Таймер*** *и* ***Датчик касания***

Теперь вы разработаете программу, чтобы увидеть, как это работает. Программа SensorCompare (рис. 14.19) содержит блок **Инфракрасный датчик** (Infrared Sensor) в режиме **Сравнение** ▸ **Приближение** (Compare ▸ Proximity), анализирующий значение датчика и проверяющий, выше ли оно 40%. Значение **Приближение** (Proximity) управляет скоростью мотора B, который вращается в течение 5 секунд. Вывод **Результат сравнения** (Compare Result) управляет параметром **Импульсный** (Pulse) блока **Индикатор состояния модуля** (Brick Status Light). Если результат сравнения истинный (значение датчика действительно выше 40%), параметр **Импульсный** (Pulse) приобретает значение **Истина** (True), что заставит индикатор состояния мигать. Если результат сравнения ложный, индикатор будет светиться непрерывно.

Режим Сравнение и значения маяка

Режимы **Приближение маяка** (Beacon Proximity) и **Направление маяка** (Beacon Heading) объединяются в один режим под названием **Маяк** (Beacon), если вы используете блок **Инфракрасный датчик** (Infrared Sensor) в режиме **Измерение** (Measure) (рис. 14.23). Эти же режимы доступны по отдельности, если вы выберете режим **Сравнение** (Compare). В остальном разницы никакой нет. Позже в этой главе вы примените режим **Маяк** (Beacon).

Режим Сравнение и датчик касания

Блок **Датчик касания** (Touch Sensor) в режиме **Сравнение** (Compare) может оповещать о щелчке по кнопке датчика (кнопка была нажата и сразу же отпущена) с последнего момента, когда какой-либо блок был использован для взаимодействия с датчиком касания. То есть, если вы щелкнете по кнопке датчика и позже проверите состояние датчика, блок сообщит вам о щелчке: вывод **Измеренное значение** (Measured value) передаст числовое значение 2. Далее следует некоторая неясность — если вы проверите, отпущена ли кнопка датчика после щелчка, блок выдаст значение **Ложь** (False) (потому что по кнопке щелкнули), даже если кнопка уже отпущена.

Обычно достаточно знать, нажата или отпущена кнопка датчика касания при запуске блока, поэтому лучше использовать блок **Датчик касания** (Touch Sensor) в режиме

Таблица 14.3. Значения датчика для каждого режима работы

Блок датчика	Режим работы	Мин.	Макс.	Значение	Стр.	Заметки
Кнопки управления модулем	Кнопки управления модулем	0	5	0 = Ничего, 1 = Влево, 2 = По центру, 3 = Вправо, 4 = Вверх, 5 = Вниз	119	Определяет только одну кнопку за один раз
Датчик цвета	Цвет	0	7	0 = Нет цвета, 1 = Черный, 2 = Синий, 3 = Зеленый, 4 = Желтый, 5 = Красный, 6 = Белый, 7 = Коричневый	99	
	Яркость отраженного света	0	100	0 = Самая низкая яркость 100 = Самая высокая яркость	103	
	Яркость внешнего освещения	0	100	0 = Самая низкая яркость 100 = Самая высокая яркость	107	
Инфракрасный датчик	Приближение	0	100	0 = Очень близко 100 = Очень далеко	111	
	Маяк (Приближение)	1	100	1 = Очень близко 100 = Очень далеко	115	Значение *не определено*, если не обнаружен сигнал маяка. См. рис. 14.26
	Маяк (Направление)	–25	25	–25 = Слева 0 = По центру 25 = Справа	115	Значение равно 0, если не обнаружен сигнал маяка или если нельзя определить направление сигнала
	Удаленный	0	11	Число представляет собой комбинацию нажатых кнопок на удаленном инфракрасном маяке	114	
Вращение мотора	Градусы	Н/Д	Н/Д	Количество градусов, на которое повернулся мотор с момента запуска программы	120	Значение может быть сброшено на 0 с помощью режима **Сброс** (Reset)
	Обороты	Н/Д	Н/Д	Количество оборотов, которые выполнил мотор с момента запуска программы	120	Значение представляет собой десятичное число, например 1,5 — для полутора оборотов. Значение можно сбросить на 0 с помощью режима **Сброс** (Reset)
	Текущая мощность	–100	100	Данное число отображает скорость вращения мотора. 100%-ная скорость равна 170 оборотам в минуту для большого мотора и 267 оборотам в минуту для среднего мотора	121	Этот режим измеряет скорость вращения; потребляемая мощность или питание не измеряется. Измеренное значение не зависит от уровня заряда батареи модуля EV3
Таймер	Время	0	Н/Д	Время в секундах, прошедшее с момента запуска программы	256	Значение представляет собой десятичное число, например 1,5 для полутора секунд. Значение можно сбросить на 0 с помощью режима **Сброс** (Reset)
Датчик касания	Состояние	**Ложь** (False)	**Истина** (True)	False = Отпущено True = Нажато	88	Значение реализуется с помощью логического канала данных

Измерение (Measure). В этом режиме блок выдает значение **Истина** (True), если кнопка датчика в данный момент нажата, и **Ложь** (False), если отпущена, независимо от того, что произошло ранее во время выполнения программы.

Диапазон значений шин данных

Во время разработки программ, включающих шины данных, важно учитывать, что происходит, когда передаваемое значение шины данных выходит за пределы допустимого диапазона значений. Например, индикатор состояния модуля принимает три значения, которые устанавливают зеленый (0), оранжевый (1) или красный (2) цвет, но что произойдет, если вы будете управлять

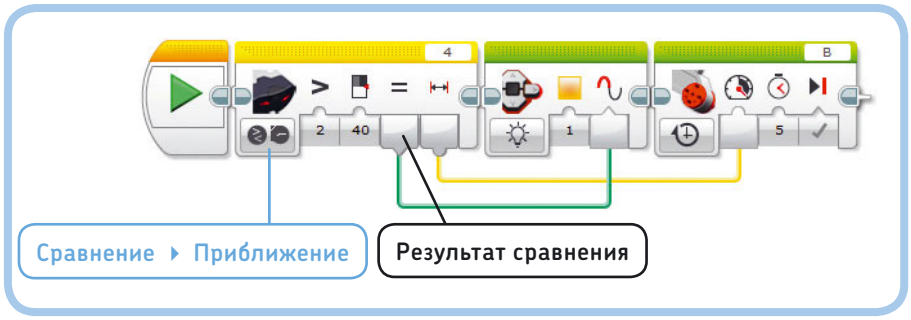


Рис. 14.19. Программа SensorCompare

ПРАКТИКУМ № 77: РЕГУЛЯТОР ДАТЧИКА!

Сложность: **Время:**

Можете ли вы составить программу управления скоростью белой стрелки (мотор B) путем поворота красного диска (мотор C)? Поверните красную стрелку вручную, чтобы протестировать вашу программу.

СОВЕТ **Используйте программу RepeatWire (см. рис. 14.12) в качестве основы и блок Вращение мотора (Motor Rotation) в режиме Измерение ▸ Градусы (Measure ▸ Degrees).**

ПРАКТИКУМ № 78: РАСШИРЕННАЯ ВЕРСИЯ ПРОГРАММЫ MY PORT!

Сложность: **Время:**

Дополните программу DisplayNumeric (см. рис. 14.17), чтобы выводить на экран модуля EV3 следующие значения: датчика цвета (значение яркости отраженного света), датчика касания и датчиков вращения (положения). Эти значения должны обновляться четыре раза в секунду.

Когда будете готовы, поместите блоки внутри цикла в контейнер «Мой блок» с именем **MyPortView**. Вы можете использовать его в своих программах для робота SK3TCHBOT всегда, когда требуется просмотреть данные, получаемые от каждого датчика.

СОВЕТ **Поместите блок Ожидание (Wait) в цикл и приостановите выполнение программы на 0,25 секунд.**

ПРАКТИКУМ № 79: СРАВНИТЕ РАЗМЕР!

Сложность: **Время:**

Можете ли вы отобразить круг в центре экрана модуля EV3 и управлять его размером и цветом с помощью замеров инфракрасного датчика? Используйте параметр **Радиус** (Radius) блока **Экран** (Display), чтобы управлять размером круга; используйте параметр **Заполнить** (Fill), чтобы отобразить закрашенный круг, если значение приближения превышает 30%, и пустой круг в остальных случаях. Поместите блоки **Инфракрасный датчик** (Infrared Sensor) и **Экран** (Display) в блок **Цикл** (Loop), чтобы круг на экране постоянно обновлялся.

СОВЕТ **Используйте блок Инфракрасный датчик (Infrared Sensor) в режиме Сравнение ▸ Приближение (Compare ▸ Proximity).**

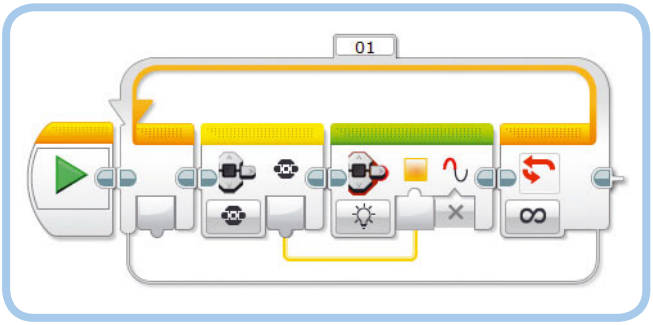


Рис. 14.20. Программа ColorRange

цветом, используя шину данных, передающую значение 4? Чтобы выяснить это, составьте программу ColorRange, показанную на рис. 14.20, и обратитесь к табл. 14.3, чтобы увидеть, какое значение шины данных будет использоваться для каждой кнопки EV3.

После запуска программы индикатор будет подсвечиваться зеленым, если ни одна кнопка не нажата (0), оранжевым, если нажата кнопка «Влево» (1), и красным, если нажата центральная кнопка (2). На все остальные кнопки (3, 4 и 5) индикатор тоже будет реагировать красной подсветкой.

Вы найдете допустимые значения для каждого ввода в справочной системе, выбрав команду меню **Отобразить справку EV3** (Help ▸ Show EV3 Help), но в документации не сказано, что произойдет, если вы выйдете за пределы допустимого диапазона. Как правило, вам достаточно лишь помнить, что программное обеспечение EV3 использует ближайшее допустимое значение, но вам придется провести простой эксперимент, наподобие программы ColorRange, чтобы удостовериться, что данное обобщение применимо к вашей программе. Это правило применимо в следующем случае: значение 2 (красная подсветка) — ближайшее допустимое, если шина данных передает значение 3, 4 или 5.

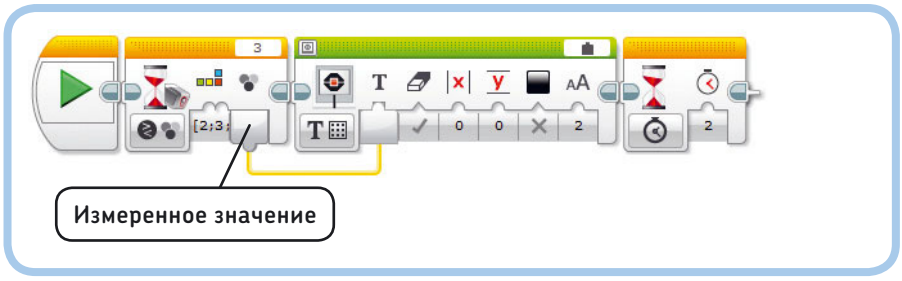


Рис. 14.21. Программа WireWait

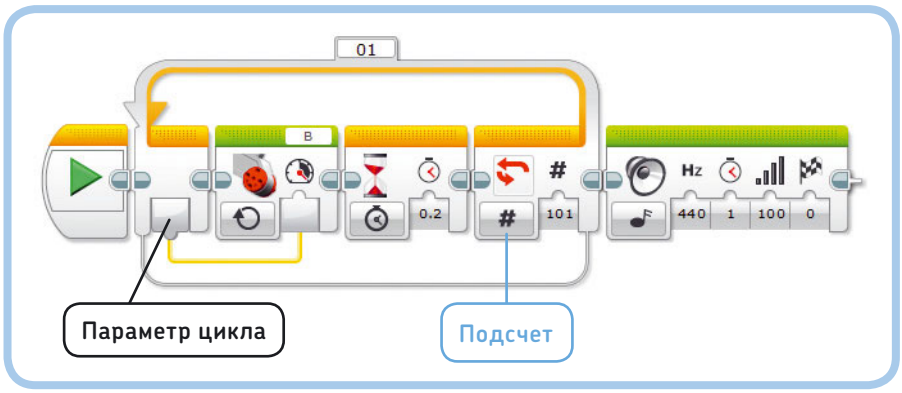


Рис. 14.22. Программа Accelerate

Расширенные функции блоков управления операторами

Теперь, когда вы знаете, как работают шины данных, вы готовы к изучению возможностей блоков **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch), которые требуют использования шин данных. Кроме того, вы научитесь использовать блок **Прерывание цикла** (Loop Interrupt).

Шины данных и блок Ожидание

Напоминаю, что блок **Ожидание** (Wait) приостанавливает программу до тех пор, пока датчик не зафиксирует определенное пороговое значение или не будет достигнуто определенное условие. Вывод **Измеренное значение** (Measured Value) передает значения, согласно которым блок прекращает ожидание. Например, программа WireWait (рис. 14.21) ожидает, пока датчик цвета обнаружит синий (2), зеленый (3), желтый (4) или красный цвет (5), после чего выводит результат последнего измерения на экран модуля.

Шины данных и блок Цикл

Блоки **Цикл** (Loop) поддерживают две функции, которые требуют использования шин данных: **Параметр цикла** (Loop Index) и **Логическое значение** (Logic). Вы попрактикуетесь с каждой функцией в следующей программе.

Использование вывода Параметр цикла

Вывод **Параметр цикла** (Loop Index) (рис. 14.22) предоставляет сведения о количестве завершенных запусков содержимого блока **Цикл** (Loop). Значение параметра начинается с нуля, увеличиваясь на единицу каждый раз после запуска блоков в цикле.

Теперь вы разработаете программу Accelerate, в которой используется параметр **Параметр цикла** (Loop Index) в качестве вводного значения, определяющего скорость мотора. Когда вы запускаете программу, значение параметра будет равно нулю, поэтому мотор будет вращаться на скорости 0 (не вращается) в течение 0,2 секунд. Когда блок **Цикл** (Loop) возвращается к началу, значение параметра **Параметр**

цикла (Loop Index) повышается до 1, после чего вновь запускается блок **Большой мотор** (Large Motor), определяя в этот раз скорость мотора равной 1. Во время следующей итерации цикла скорость мотора возрастет до 2, и так далее.

Блок **Цикл** (Loop) настроен таким образом, что будет запускаться 101 раз, поэтому скорость будет равна 100, когда программа запустится в последний раз. Затем вы услышите сигнал, и программа завершит работу. Вы можете настроить программу так, чтобы цикл запускался 150 раз, но мотор перестанет набирать скорость, когда значение параметра достигнет 100, потому что будет достигнута максимальная скорость.

Завершение цикла в режиме Логическое значение

Ранее вы узнали, что можно прекратить повторение блока **Цикл** (Loop) при достижении определенного количества итераций, секунд или же некоторого значения датчика. В режиме **Логическое значение** (Logic) вы можете остановить цикл путем использования специальной шины данных.

Блок **Цикл** (Loop) проверяет значение в шине данных каждый раз при запуске блоков в цикле. Если значение шины данных **Ложь** (False), цикл запускается вновь. Если значение — **Истина** (True), цикл завершается. Другими словами, цикл повторяется до тех пор, пока значение в шине данных не станет **Истина** (True).

Программа LogicLoop (см. рис. 14.23) демонстрирует данный метод: в ней используется блок **Цикл** (Loop) в режиме **Логическое значение** (Logic) и блок **Инфракрасный датчик** (Infrared Sensor) в режиме **Измерение ▸ Маяк** (Measure ▸ Beacon). Блок **Инфракрасный датчик** (Infrared Sensor) предоставляет также выводы **Приближение** (Proximity) и **Направление** (Heading), но здесь вы будете использовать только вывод **Обнаружено** (Detected), который информирует, зафиксировал ли датчик сигнал маяка. Если сигнал обнаружен, шина данных передает значение **Истина** (True), которое приводит к завершению цикла; если сигнал отсутствует, шина данных передает значение **Ложь** (False), и цикл выполняется вновь. Другими словами, программа ожидает, пока датчик не поймает сигнал, а затем воспроизводит звук.

После запуска программы вы увидите, что датчик может успешно обнаружить сигнал на расстоянии до 3 метров.

ПРАКТИКУМ № 80: УДАЛЕННОЕ УСКОРЕНИЕ!

Сложность: **Время:**

Можете ли вы запрограммировать мотор B работать с ускорением до тех пор, пока инфракрасный датчик не обнаружит сигнал от маяка?

СОВЕТ **Объедините программы Accelerate (см. рис. 14.22) и LogicLoop (см. рис. 14.23) в одну.**

Шины данных и блок Переключатель

Как вы помните из главы 6, блоки **Переключатель** (Switch) применяются для того, чтобы робот «принимал решения». Робот использует результаты замеров датчиков для определения, является ли текущее состояние (к примеру, «значение приближения выше 30%») истинным. Если оно истинно, в блоке **Переключатель** (Switch) запускается верхняя последовательность блоков (✓); а если ложно — нижняя (✕). Такое поведение демонстрирует программа SwitchReminder на рис. 14.24.

Режим Логическое значение

Вместо использования результатов замеров датчиков для выбора последовательности вы можете применить логическую шину данных для управления блоком **Переключатель** (Switch), работающим в режиме **Логическое значение** (Logic), как показано на рис. 14.25. Если передается значение **Истина** (True), запускаются блоки в верхней части переключателя; а если **Ложь** (False) — блоки в нижней части.

Программа LogicSwitch1 (рис. 14.25) постоянно проверяет, фиксирует ли инфракрасный датчик сигнал от удаленного маяка. Если это так (**Истина** (True)), робот отображает слово «Успех!»* на экране модуля EV3, а мотор B вращается; если нет (**Ложь** (False)) — слово «Ошибка!»**, и мотор не вращается. Датчик перестает получать сигнал через несколько секунд

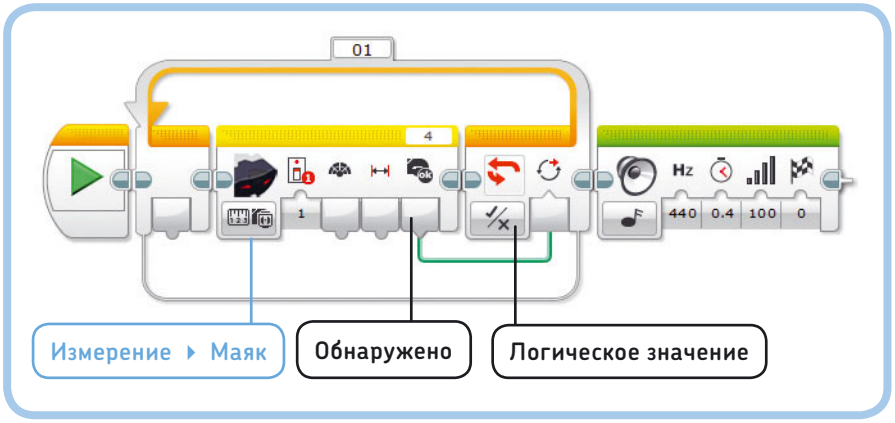


Рис. 14.23. Программа LogicLoop воспроизводит звук, когда инфракрасный датчик фиксирует сигнал удаленного маяка

* Пункт **Success!** в каталоге звуков Lego.

** Пункт **Error!** в каталоге звуков Lego.

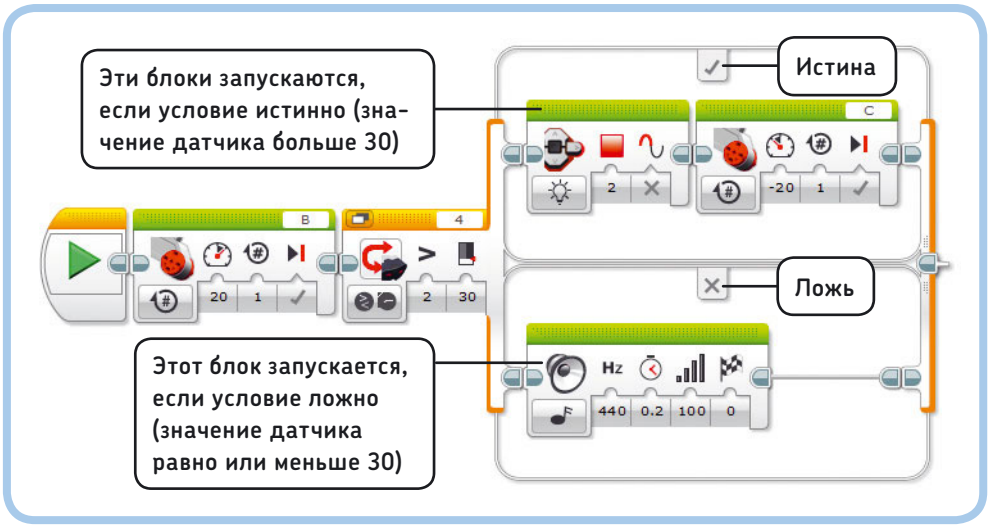


Рис. 14.24. Программа SwitchReminder

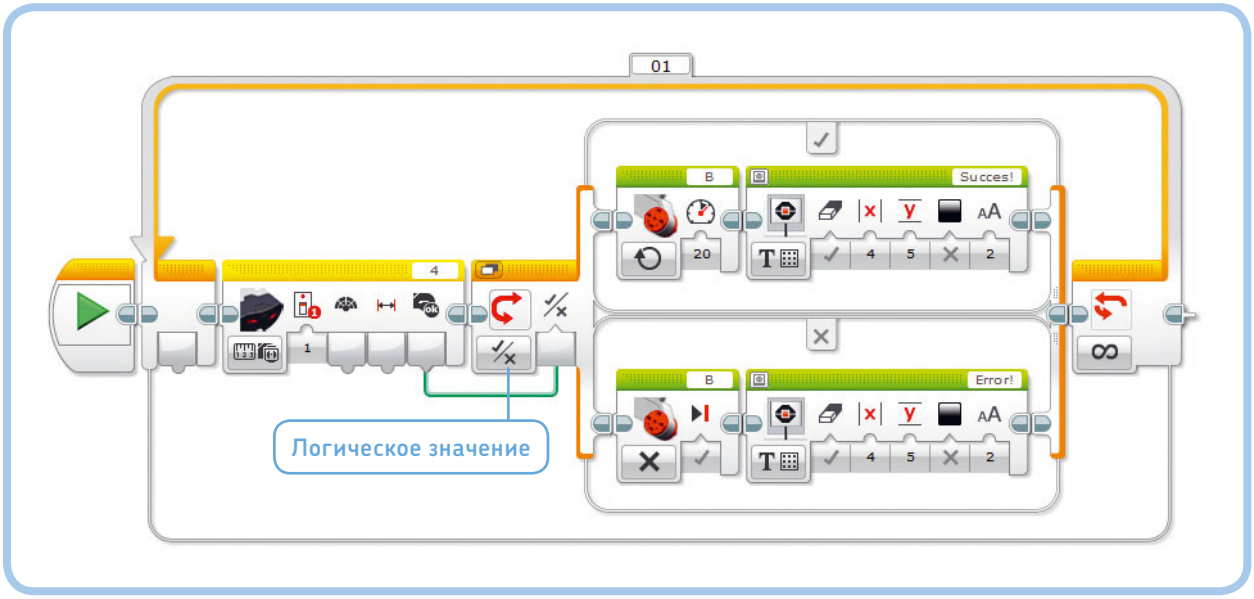


Рис. 14.25. Программа LogicSwitch1

после того, как вы отпустите кнопку, поэтому появление сообщения об ошибке займет около секунды.

Режим Числовое значение

Если вы переведете блок **Переключатель** (Switch) в режим **Числовое значение** (Numeric) и подключите к нему числовую шину данных, то сможете для каждого значения выбрать определенные действия, как это было описано в главе 7 (см. рис. 7.10). Например, вы можете запрограммировать робота произносить «Привет!», если значение в шине данных равно 3, «Доброе утро!», если оно равно 10, и «Нет!» для всех остальных значений (значение по умолчанию). Вы разработаете такую программу в следующей главе.

Подключение шин данных к блокам внутри переключателей

В некоторых случаях может потребоваться подключать шины данных от выводов снаружи блока **Переключатель** (Switch) к вводам блоков внутри переключателя. Например, вы можете изменить свою предыдущую программу, чтобы управлять скоростью мотора на основе расстояния от маяка без необходимости использования другого блока **Инфракрасный датчик** (Infrared Sensor).

Чтобы сделать это, выберите в блоке **Переключатель** (Switch) режим отображения с вкладками и подключите шину данных, как показано на рис. 14.26. Завершенная программа LogicSwitch2 управляет скоростью мотора с помощью

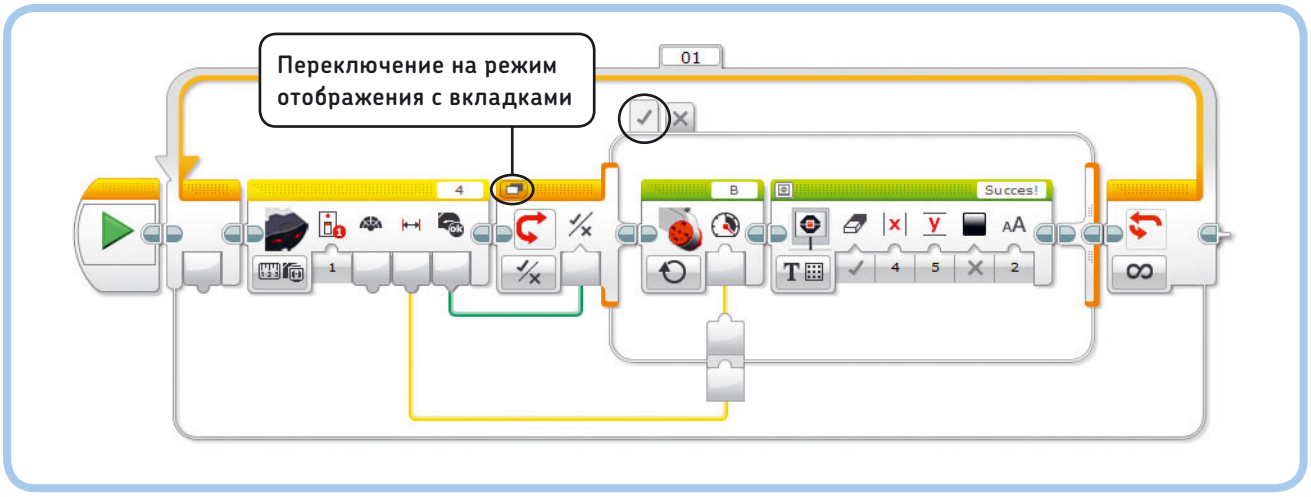


*Рис. 14.26. Программа LogicSwitch2. Начните с программы LogicSwitch1, выберите режим отображения с вкладками, перейдите на вкладку **Истина** и подключите числовую шину данных, как показано на рисунке. Пара вводов/выводов на границе блока **Переключатель** появится автоматически, когда вы попробуете подключить шину. Блоки на вкладке **Ложь** остаются без изменений*

удаленного маяка, если сигнал обнаружен; в противном случае программа останавливает мотор.

Как показано в табл. 14.3, значение приближения маяка *не определено*, если сигнал не обнаружен. Если вывод **Приближение** (Proximity) подключен к блоку **Большой мотор** (Large Motor), а сигнал не обнаружен, большой мотор не получает значения и вращается неконтролируемо. Вы можете избежать этой проблемы, следя за тем, что значение используется только тогда, когда оно содержит точный результат замеров датчика — только если вывод **Обнаружено** (Detected) передает значение **Истина** (True).

Поэтому вам и нужно использовать блок **Переключатель** (Switch) в программе LogicSwitch2. Значение датчика используется для управления скоростью мотора только в том случае, если обнаружен сигнал от маяка. В противном случае запускаются блоки на вкладке **Ложь** (False), и блок **Большой мотор** (Large Motor) в режиме **Выключить** (Off)) останавливает мотор.

В режиме **Направление маяка** (Beacon Heading) значение на выводе будет 0, даже если маяк находится непосредственно перед датчиком или если сигнал вообще не регистрируется. Можно провести различие между двумя этими случаями, используя такой же метод, как в нашей последней программе. Например, вы можете отобразить значение параметра **Направление маяка** (Beacon Heading) на экране, если сигнал обнаружен, а если он отсутствует, на экране появится слово «Ошибка!».

ПРИМЕЧАНИЕ **Вы можете подключить шины данных к блокам внутри блока Переключатель (Switch) только в том случае, если выбран режим отображения с вкладками. Переключение в режиме без вкладок удаляет шины данных. Подключение шин данных к содержимому блоков Цикл (Loop) осуществляется таким же образом.**

Блок Прерывание цикла

Окончательный способ заставить блок **Цикл** (Loop) остановить повторение заключается в использовании блока **Прерывание цикла** (Loop Interrupt). Блок **Цикл** (Loop), как правило, проверяет состояние датчика или логическое значение каждый раз, когда завершается выполнение вложенных блоков, а блок **Прерывание цикла** (Loop Interrupt) завершает указанный цикл немедленно.

Вы выбираете, какой цикл требуется прервать, выбрав из списка его имя, как показано на рис. 14.27. Вы можете использовать блок **Прерывание цикла** (Loop Interrupt) внутри цикла для его остановки изнутри или завершения цикла на параллельной последовательности. После того как цикл прерван, программа продолжает работу с блоками, размещенными после цикла.

Прерывание цикла изнутри

Блок **Прерывание цикла** (Loop Interrupt) может быть полезен для выхода из блока **Цикл** (Loop) в любой позиции цикла вместо ожидания завершения работы всех блоков в цикле. Рассмотрим программу BreakFromInside, показанную на рис. 14.27, которая систематически вращает мотор B на один поворот, а затем воспроизводит слово «LEGO». Если значение, полученное от инфракрасного датчика, составляет менее 50% после того, как робот произнес слово «LEGO», цикл завершается нормально, и сразу же после этого робот произносит слово «MINDSTORMS».

Благодаря блоку **Прерывание цикла** (Loop Interrupt) вы можете завершить цикл, нажав и удерживая кнопку датчика касания после того, как мотор B совершил вращение. Если вы так поступите, программа перейдет к блоку после цикла, и вы услышите только слово «MINDSTORMS».

Составьте и настройте программу, как показано на рис. 14.27, и запустите ее несколько раз, чтобы определить, какой датчик и когда необходимо привести в действие, чтобы цикл завершил работу.

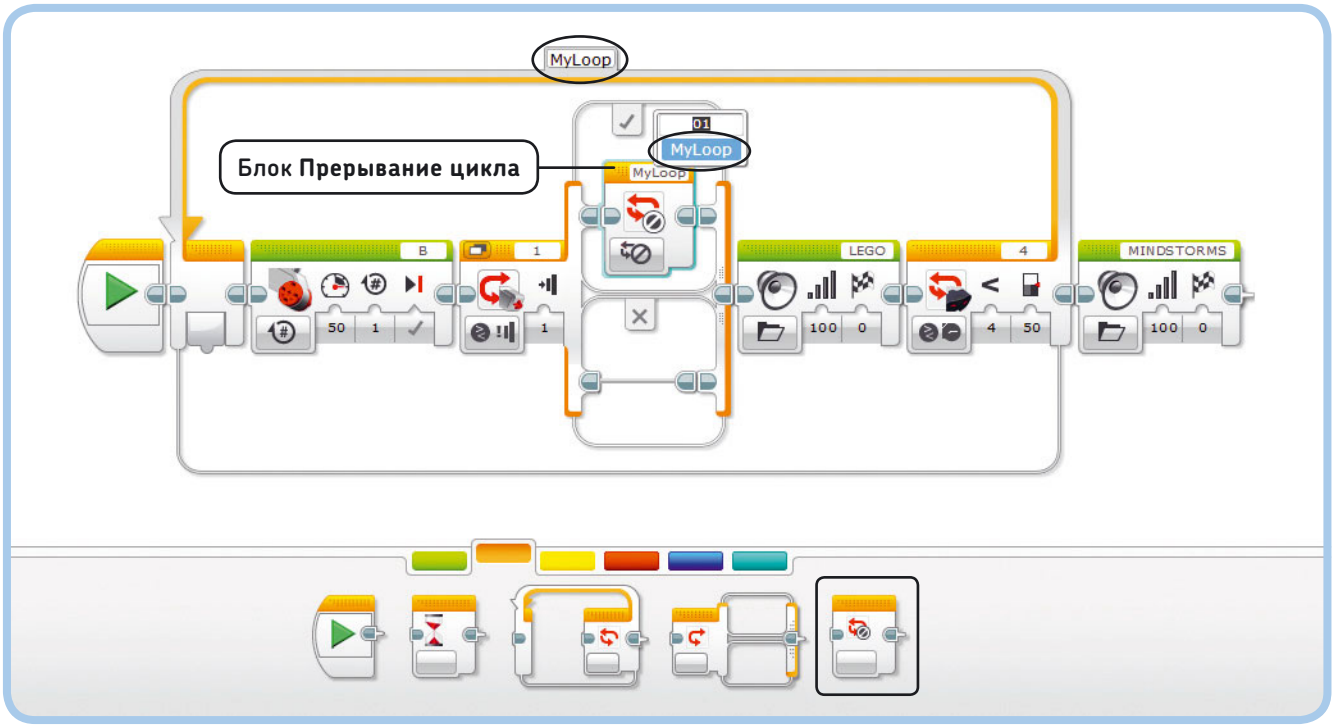


*Рис. 14.27. Программа BreakFromInside содержит цикл с именем **MyLoop**, который завершается, если кнопка датчика касания нажимается сразу же после вращения мотора или если инфракрасный датчик срабатывает сразу после того, как робот произнес слово «LEGO»*

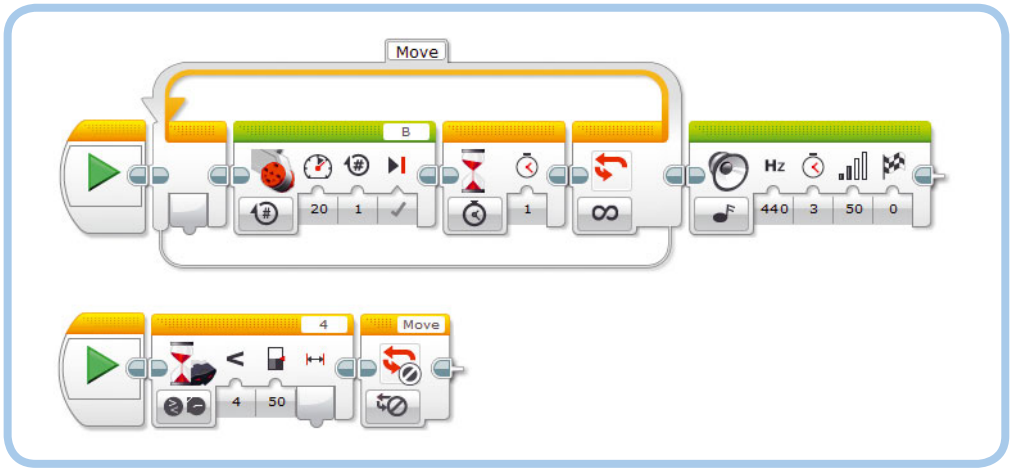


*Рис. 14.28. Программа BreakFromOutside прерывает цикл с именем **Move**, если срабатывает инфракрасный датчик*

Прерывание цикла снаружи

Вы можете прервать выполнение блока **Цикл** (Loop) из последовательности, запущенной параллельно. В этом случае цикл завершит работу и программа запустит блоки после цикла. В то же время программа будет пытаться завершить работу блока, который был запущен при выполнении блока **Прерывание цикла** (Loop Interrupt). Так, программа BreakFromOutside (рис. 14.28) содержит блок **Цикл** (Loop), который систематически поворачивает мотор B на одно вращение. На параллельной последовательности блок **Прерывание цикла** (Loop Interrupt) запускается после того, как срабатывает инфракрасный датчик. Затем цикл завершается, и блок **Звук** (Sound) немедленно воспроизводит звук.

Если вы нажмете кнопку датчика, когда мотор совершил пол-оборота, он закончит вращение во время воспроизведения звука. Блоки **Большой мотор** (Large Motor) и **Звук** (Sound) запустятся одновременно. Теперь присвойте параметру **Продолжительность** (Duration) блока **Звук** (Sound) значение 0,1 секунды и вновь запустите программу. Если вы приведете в действие датчик, когда мотор совершил пол-оборота,

программа завершится почти сразу же, останавливая мотор прежде, чем он закончит оборот.

Рекомендую вам использовать этот способ с осторожностью. Прерывание цикла, который выполняется параллельно, может привести к тому, что робот будет вести себя непредсказуемо. (Что произойдет, если, к примеру, вы запрограммируете мотор совершить еще один оборот сразу же после цикла? Будет ли продолжать выполняться предыдущий блок или же запустится новый?)

Обратите внимание, что прерывание цикла изнутри не приводит к неоднозначности поведения: программа BreakFromInside прерывает цикл, но не завершает работу другого запущенного блока.

Дальнейшее изучение

В этой главе вы научились использовать шины данных для переноса информации из одного блока в другой. Кроме того, вы научились считывать значения датчиков с помощью блоков датчиков и работать с расширенными функциями блоков **Ожидание** (Wait), **Цикл** (Loop) и **Переключатель** (Switch).

Большинство программ, которые вы создали на данный момент с помощью шин данных, относительно просты, и шины данных пока не представляются очень полезными. Но они необходимы для программирования усовершенствованных роботов, которые рассматриваются в части VI книги. Следующие практикумы предоставят вам возможность на практике реализовать навыки программирования, полученные в этой главе.

ПРАКТИКУМ № 81: ПРЕРЫВАНИЕ ПРЕРЫВАНИЙ!

Сложность: **Время:**

Можете ли вы создать программу наподобие BreakFromInside, которая постоянно вращает мотор и воспроизводит звук, пока нажата кнопка датчика касания, а датчик цвета фиксирует зеленый цвет? Одновременное срабатывание этих датчиков должно завершить цикл после вращения мотора. В следующей главе вы узнаете еще об одном способе завершения цикла, когда срабатывает несколько датчиков, а пока просто используйте блок **Прерывание цикла** (Loop Interrupt).

СОВЕТ **Добавьте блок Переключатель (Switch) в программу BreakFromInside (см. рис. 14.27).**

ПРАКТИКУМ № 82: РЕЖИМ РАБОТЫ ДАТЧИКА!

Сложность: **Время:**

Создайте программу, которая будет вращать белую стрелку на скорости, зависимой от состояния инфракрасного датчика, но только когда вы одновременно нажимаете кнопку датчика касания и касаетесь датчика цвета. Если любой из этих датчиков не сработал, мотор не должен вращаться.

СОВЕТ **В этой главе вы узнали о шинах данных и блоках датчиков, но иногда вам все равно придется использовать блоки Ожидание (Wait), Цикл (Loop) и Переключатель (Switch) для работы с датчиками.**

ПРАКТИКУМ № 83: ПИТАНИЕ VS СКОРОСТЬ!

Сложность: **Время:**

Разработайте программу, которая вращает мотор B на скорости 30% (51 оборот в минуту) с помощью блока **Большой мотор** (Large Motor), и мотор C на скорости 30% с помощью блока **Нерегулируемый мотор** (Unregulated Motor). Затем непрерывно отображайте скорость обоих моторов, используя блоки **Вращение мотора** (Motor Rotation) в режиме **Текущая мощность** (Current Power). Напомним, что этот режим показывает *скорость вращения мотора*, как описано в главе 9.

Теперь посмотрим, что произойдет, когда вы попытаетесь замедлить моторы руками. Вы должны увидеть, что скорость мотора C быстро падает: 30% мощности недостаточно для преодоления торможения. Но мотор B продолжает вращаться на почти 30% скорости, потому что блок программирует модуль EV3 использовать дополнительное питание для мотора при его замедлении.

ПРАКТИКУМ № 84: РЕАЛЬНОЕ НАПРАВЛЕНИЕ!

Сложность: **Время:**

Можете ли вы создать программу, которая будет отображать значение параметра Направление маяка (Beacon Heading), если зафиксирован сигнал маяка, и слово «Ошибка!»*? если сигнал не обнаружен?

СОВЕТ **В качестве основы возьмите программу LogicSwitch2 (см. рис. 14.26).**

ПРАКТИКУМ № 85: SK3TCHBOT НАБЛЮДАЕТ ЗА ВАМИ!

Сложность: **Время:**

Создайте программу, которая будет подсчитывать количество людей, проходящих рядом с вашим роботом. Разместите робота таким образом, чтобы инфракрасный датчик мог фиксировать людей, находящихся перед ним. Вложите два блока **Ожидание** (Wait) внутрь блока **Цикл** (Loop) и настройте первый блок так, чтобы он ожидал до тех пор, пока кто-то не пройдет мимо; используйте второй блок, чтобы подождать, пока человек не окажется вне поля зрения. В конечном счете, цикл должен выполняться однократно каждый раз, когда человек проходит мимо. Теперь при отображении значения параметра **Параметр цикла** (Loop Index) на экране модуля EV3 вы будете видеть, сколько людей прошло мимо. Чтобы убедиться, что ваша программа работает правильно, поместите блок **Звук** (Sound) внутрь цикла, чтобы слышать звуковой сигнал каждый раз, когда кто-то идет мимо.

СДЕЛАЙ САМ № 25: БИОНИЧЕСКАЯ РУКА!

Сборка: **Программирование:**

Можете ли вы спроектировать захватное устройство, которое затем можно будет прикрепить к вашей руке? Используйте датчики и кнопки модуля EV3 для управления движениями захватного устройства. Добавьте блок **Инфракрасный датчик** (Infrared Sensor) и запрограммируйте робота таким образом, чтобы рука предупреждала вас, когда датчик фиксирует приближение к стене. Используйте методы программирования, которые вы изучили в этой главе, чтобы выводить результаты замеров датчика на экран или настроить звуковое оповещение на основе замеров датчика.

ПРАКТИКУМ № 86: ОСЦИЛЛОГРАФ!

Сложность: **Время:**

Можете ли вы превратить ваш модуль EV3 в измерительное устройство, которое записывает показания датчика на экране, как показано на рис. 14.29? Отобразите значение параметра **Приближение** (Proximity) в виде небольших окружностей, чьи координаты *x* определяются с помощью значения параметра **Параметр цикла** (Loop Index), а координаты *y* — с помощью значения параметра **Приближение** (Proximity). В процессе работы цикла вы должны видеть, как новые результаты измерений отображаются один за другим и образуют диаграмму, подобную показанной на рисунке.

Так как экран содержит 178 пикселов по горизонтали, блок **Цикл** (Loop) должен сработать 178 раз, чтобы отобразить 178 кругов, и использовать паузу в 0,05 секунды между замерами. Затем программа должна очистить экран и начать все сначала.

СОВЕТ **Чтобы блок Экран (Display) очищал экран после 178 измерений, сделайте так, чтобы на экран выводился белый прямоугольник, присвоив параметру Цвет (Color) значение Истина (True).**

Рис. 14.29. Диаграмма из практикума № 86. Подобная диаграмма может быть получена в результате постоянного перемещения объекта вперед и назад перед датчиком

* Пункт **Error!** в каталоге звуков Lego.

15

Использование блоков операций с данными и контейнеров с шинами данных

Теперь, когда вы знаете, как работают шины данных, можно сделать много интересного, используя большее количество программных блоков. Например, вы можете сделать так, чтобы модуль EV3 объединял и обрабатывал данные датчиков, чтобы их можно было использовать как входные значения для выполнения других действий. С помощью приемов, которые вы изучите в этой главе, вы сможете запрограммировать своего робота на выбор случайного действия или же на то, чтобы он выполнял некие действия, когда два датчика срабатывают одновременно, вместо выполнения заранее запрограммированных действий.

В этой главе вы узнаете, как применить блок **Математика** (Math), чтобы роботы выполняли расчеты, которые можно использовать в программах. Например, робот может вычислить расстояние, которое он должен пройти, основываясь на показаниях датчика. Еще я представлю вам несколько новых блоков программирования, таких как **Случайное значение** (Random), **Сравнение** (Compare) и **Логические операции** (Logic Operations), и покажу, как создавать контейнеры «Мой блок» с вводами и выводами.

Эти приемы и программные блоки необходимы для разработки сложных программ для роботов. Как и ранее, в этой главе вы будете использовать робота SK3TCHBOT для проверки создаваемых программ.

Практикумы в этой главе сначала могут показаться немного сложными, но они помогут вам овладеть необходимыми навыками программирования, что позволит разрабатывать гораздо более интересные программы и собирать действительно умных роботов!

Применение блоков операций с данными

Палитра программирования содержит ряд блоков, которые вы еще не использовали, например блоки операций с данными (рис. 15.1). Они включают в себя блоки **Математика** (Math), **Случайное значение** (Random), **Сравнение** (Compare) и **Логические операции** (Logic Operations). Каждый блок обладает собственным функционалом, но все они обрабатывают значения, передаваемые шинами данных, и генерируют новые значения на их основе. Данный раздел объясняет, как использовать эти блоки в программах.

Рис. 15.1. Блоки операций с данными

Блок Математика

Блок **Математика** (Math) (рис. 5.2) позволяет модулю EV3 выполнять арифметические операции, такие как сложение, вычитание, умножение и деление. Вы вводите два числа в поля **a** и **b** блока и используете раскрывающийся список выбора режима, чтобы выбрать, какую операцию (например, деление) следует применить к ним (в случае с делением значение **a** будет делиться на **b**). Шина данных выводит результат. Чтобы вручную не присваивать значения параметрам **a** и **b**, вы можете передавать их через шину данных.

Блок Математика в действии

Программа на рис. 15.3 демонстрирует блок **Математика** (Math), который выполняет операцию по умножению результата замеров датчика цвета для управления скоростью мотора B. Значение датчика цвета (число в диапазоне от 0 до 7) не подходит для управления скоростью, потому что с этим значением мотор будет вращаться очень медленно (в лучшем случае 7% скорости). Блок **Математика** (Math) увеличивает значение датчика в десять раз и передает результат (число в диапазоне от 0 до 70) во ввод **Мощность** (Power) блока **Большой мотор** (Large Motor). Как следствие, мотор вращается на скорости 10% при обнаружении черного цвета, 20% — синего и так далее, вплоть до 70% скорости при обнаружении коричневого цвета.

Создайте новый проект под названием SK3TCHBOT-Data и в нем программу MathSpeed, как показано на рис. 15.3.

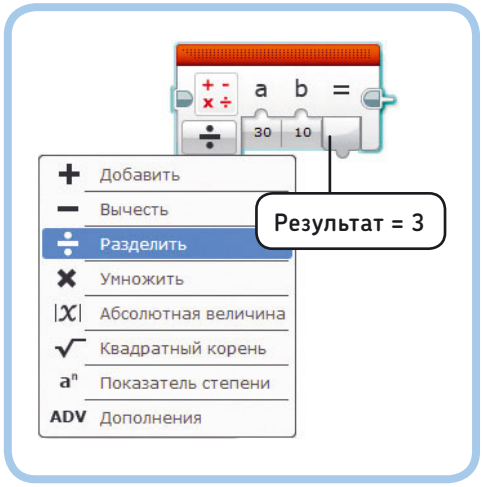


*Рис. 15.2. Блок **Математика**. Выходное значение этого блока равно 30 ÷ 10 = 3*

ПРАКТИКУМ № 87: 100%-НАЯ МАТЕМАТИКА!

Сложность: **Время:**

Благодаря блоку **Математика** (Math) в программе MathSpeed мы можем использовать определенный цвет для управления скоростью мотора, но скорость доходит только до 70%. Можно ли изменить программу так, чтобы использовать полный диапазон значений скорости? Добавьте блок **Экран** (Display) в свою программу, чтобы убедиться, что результирующий вывод блока **Математика** (Math) достигает 100% при обнаружении коричневого цвета.

СОВЕТ **Вы можете использовать десятичные числа в полях a и b блока Математика (Math).**

вычитания, второй для умножения, но в режиме **Дополнения** (Advanced) вы сможете проводить эти расчеты, используя всего лишь один блок **Математика** (Math), как показано на рис. 15.4.

В поле ввода уравнения вы вводите расчеты, которые блок **Математика** (Math) должен выполнить, таким же образом, как и на калькуляторе. Например, ввод (7–3)*1,5 в результате даст ответ 6. Так же как и на калькуляторе, вы используете круглые скобки, чтобы вычитание выполнялось прежде умножения.

Вы можете использовать в уравнении символы переменных (**a**, **b**, **c** и **d**), и присвоить значение каждой переменной

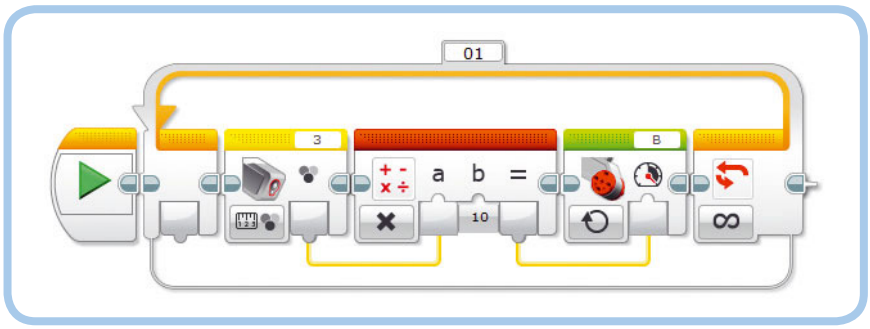


Рис. 15.3. Программа MathSpeed

Режим Дополнения

Некоторые вычисления требуют выполнения более одного арифметического действия. Например, вы можете вычесть два числа и умножить результат на третье число. Можно было бы использовать два блока **Математика** (Math), один для с помощью параметров **a**, **b**, **c** и **d** соответственно. Например, введя формулу (b–c)*a в качестве уравнения и присвоив значение 7 параметру **b**, 3 — параметру **c** и 1,5 — параметру **a**, мы получим такой же результат: 6. Наконец, вы можете присваивать значения каждой переменной, используя числовые шины данных.