Чтобы увидеть режим **Дополнения** (Advanced) в действии, создайте программу, которая заставляет мотор C (красная стрелка) следовать за движениями мотора B (белая стрелка) — его вы будете вращать вручную. Так как мотор C установлен в вашем роботе вверх ногами, стрелка на практике будет вращаться в противоположном направлении, но на такое же количество оборотов. Чтобы решить эту задачу, задайте скорость мотора C следующим образом:

Скорость мотора C = (положение в градусах мотора B – положение в градусах мотора C) × 1,5

Чтобы разобраться, почему такая формула работает, рассмотрим, что происходит, когда вы поворачиваете стрелку мотора B вперед на 70 градусов и мотор C поворачивается на 60 градусов. Скорость мотора C будет равна 15 по формуле (70 – 60) × 1,5. Согласно этому значению, мотор C будет поворачиваться вперед, чтобы догнать мотор B. Если C обгоняет B, результат будет отрицательным, и мотор C повернется назад. Чем больше разница между положениями моторов, тем быстрее будет двигаться мотор C. Как только оба мотора окажутся в одинаковом положении, результат будет равен 0 и мотор остановится.

Составьте программу PositionControl, которая показана на рис. 15.5. Поворачивайте белую стрелку вручную и наблюдайте, как красная стрелка двигается так же, но в обратном направлении.

Практическое применение блока Математика

Поскольку блок **Математика** (Math) является неотъемлемым компонентом многих программ, в которых используются шины данных, будет неплохо, если вы попрактикуетесь в его использовании, прежде чем продолжите чтение книги. Следующие практикумы помогут вам в этом.

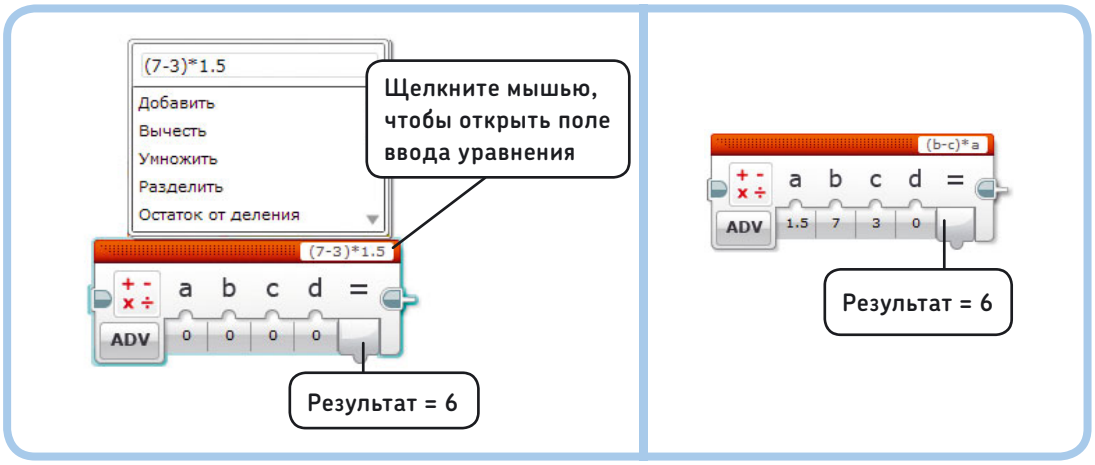


*Рис. 15.4. Блок **Math** в режиме **Дополнения**. Вы можете ввести формулу с числами (слева), с переменными (справа) или комбинацию из переменных и чисел. Операторы указываются вручную, например * для умножения и / для деления, или выбираются из списка*

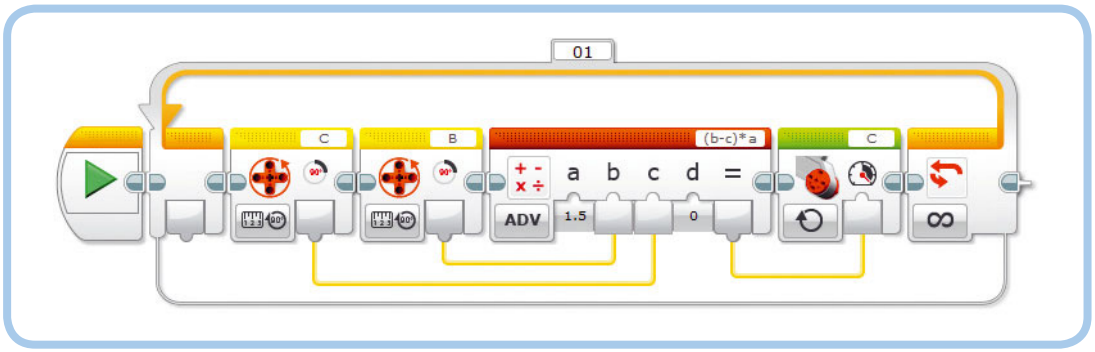


*Рис. 15.5. Программа PositionControl. Вывод **Градусы** мотора B подключен к вводу **b** блока **Математика**; вывод **Градусы** мотора C подключен к вводу **c***

ПРАКТИКУМ № 88: ДОБАВЛЕННОЕ ЗНАЧЕНИЕ!

Сложность: **Время:**

Можете ли вы разработать программу, которая будет постоянно отображать на экране результаты замера приближения инфракрасным датчиком и яркости отраженного света датчиком цвета, а также сумму результатов двух этих измерений?

ПРАКТИКУМ № 89: СКОРОСТЬ НА ОСНОВЕ ЗАМЕРОВ ИНФРАКРАСНОГО ДАТЧИКА!

Сложность: **Время:**

Разработайте программу, похожую на MathSpeed, для управления скоростью и направлением вращения мотора B на основе результатов замера приближения инфракрасным датчиком. Мотор должен вращаться на скорости 50% при 100%-ном значении параметра **Приближение** (Proximity), останавливаться при 50%-ном значении и вращаться на скорости –50% (обратное направление) при значении 0%.

СОВЕТ **Используйте формулу: Скорость = Приближение – 50. Как вы настроите блок Математика (Math), чтобы выполнить эту операцию?**

ПРАКТИКУМ № 90: УДВОЕННАЯ СКОРОСТЬ НА ОСНОВЕ ЗАМЕРОВ ИНФРАКРАСНОГО ДАТЧИКА!

Сложность: **Время:**

Можно ли доработать программу из практикума № 89, чтобы скорость мотора варьировалась в диапазоне от –100 до 100?

СОВЕТ **Используйте следующую формулу: Скорость = (Приближение – 50) × 2.**

ПРАКТИКУМ № 91: РЕГУЛЯТОР УСИЛЕНИЯ!

Сложность: **Время:**

Каким будет результат использования значения 1,5 в программе PositionControl? Попробуйте поэкспериментировать, изменяя значение до минимума (0,1) и максимума (5) и наблюдая за тем, насколько быстро мотор C сможет следовать за вашими движениями.

ПРАКТИКУМ № 92: УПРАВЛЕНИЕ НАПРАВЛЕНИЕМ!

Сложность: **Время:**

Можно ли доработать программу PositionControl таким образом, чтобы красная стрелка вращалась в том же направлении, что и белая?

СОВЕТ **Умножьте значение Градусы (Degrees) мотора C на –1, чтобы поступательное движение измерялось как реверсивное, и наоборот. Аналогично, умножьте результирующую скорость на –1, прежде чем передать ее в блок Большой мотор (Large Motor), чтобы изменить направление.**

Блок Случайное значение

Блок **Случайное значение** (Random), как можно понять из названия, позволяет генерировать случайное значение для использования в программе. В режиме **Логическое значение** (Logic) к выводу блока подключается логическая шина данных, передающая значение либо **Истина** (True), либо **Ложь** (False). Использование блока в данном режиме — это как подбрасывание монетки, результатом которого является **Истина** (True) (орел) или **Ложь** (False) (решка). Когда мы подбрасываем монетку, вероятность того, что выпадет **Истина** (True), составляет 50%, в результате значение **Истина** (True) выпадает приблизительно в половине раз. Блок **Случайное значение** (Random) позволяет вам определить вероятность выпадения значения **Истина** (True) настройкой параметра **Вероятность значения «истина»** (Probability of True) в процентах. Например, ввод числа 33 приведет к передаче значения **Истина** (True) приблизительно в одной трети раз, в то время как в двух третях раз будет выпадать **Ложь** (False).

В режиме **Числовое значение** (Numeric) к выводу блока подключается числовая шина данных, передающая случайное

целое число в диапазоне, определяемом параметрами **Нижняя граница** (Lower Bound) и **Верхняя граница** (Upper Bound). Например, присвоение параметру **Нижняя граница** (Lower Bound) значения 1, а параметру **Верхняя граница** (Upper Bound) значения 6 в результате приведет к 1, 6 или любому другому целому числу между ними. Выпасть может в равной вероятности любое число, как если бы вы бросали игральный кубик.

Блок **Случайное значение** (Random) полезен в том случае, если вы хотите, чтобы ваш робот сделал что-то неожиданное. Например, вы можете использовать блок **Случайное значение** (Random), чтобы робот в произвольном порядке произносил слова «Налево» или «Направо» или чтобы белая стрелка вращалась на произвольной скорости мотора, как показано в программе RandomMotor на рис. 15.6.

Блок **Переключатель** (Switch) в программе RandomMotor запускает либо блоки в верхней части (**Истина** (True)), либо блоки в нижней части (**Ложь** (False)). Вы можете сделать так, чтобы программа в произвольном порядке выбирала тот или иной вариант, переведя блоки **Случайное значение** (Random) и **Переключатель** (Switch) в режим **Числовое значение** (Numeric). Тогда вы сможете определить действия для каждого возможного случайного значения, как продемонстрировано в программе RandomCase на рис. 15.7.

ПРАКТИКУМ № 93: СЛУЧАЙНАЯ ЧАСТОТА!

Сложность: **Время:**

Можно ли сделать так, чтобы модуль EV3 воспроизводил произвольный тон длительностью 0,5 секунды каждый раз, когда вы нажимаете кнопку датчика касания? Сгенерируйте случайное значение и передайте его на ввод **Частота** (Frequency) блока **Звук** (Sound). Наконец, дополните программу, чтобы она выводила частоту на экран модуля EV3.

СОВЕТ **Какие значения принимает ввод Частота (Frequency) блока Звук (Sound)?**

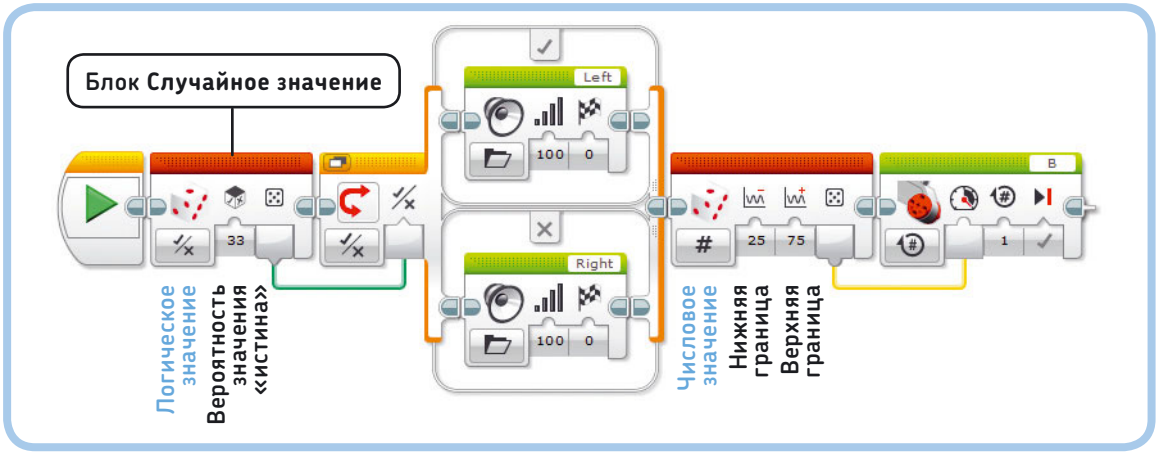


Рис. 15.6. Программа RandomMotor. Если вы запустите программу несколько раз, вы обнаружите, что робот произносит «Налево» примерно в трети случаев. После этого мотор B вращается на один оборот на произвольной скорости между 25 и 75%

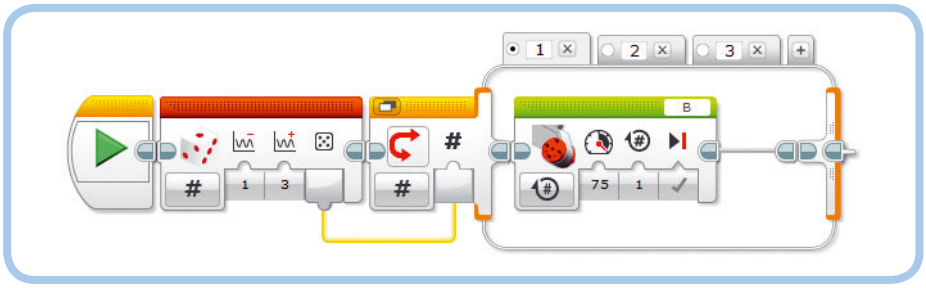


Рис. 15.7. Программа RandomCase в произвольном порядке выбирает, на какой вкладке выполнить блоки. Здесь показан блок только на первой вкладке; вы можете добавить любые блоки, какие вам необходимы

ПРАКТИКУМ № 94: ПРОИЗВОЛЬНО ВЫБРАННЫЙ МОТОР И СКОРОСТЬ!

Сложность: **Время:**

Разработайте программу, которая генерирует случайное число между 10 и 100, чтобы управлять скоростью мотора B или C. Если случайное значение меньше 50, мотор B должен повернуться на один оборот со случайным значением параметра **Мощность** (Power); если значение выше 50, на один оборот должен повернуться на произвольной скорости мотор C.

СОВЕТ **Вам понадобятся блоки Случайное значение (Random), Сравнение (Compare), Переключатель (Switch) (выберите режим просмотра в виде вкладок) и два блока Большой мотор (Large Motor).**

Блок Сравнение

Блок **Сравнение** (Compare) выполняет проверку, равно (=), не равно (≠), больше (>), больше или равно (≥), меньше (<), меньше или равно (≤) одно числовое значение в сравнении с другим. Вы можете присвоить любые значения, которые хотите сравнить, параметрам (**a** и **b**) блока или же передать данные в блок с помощью шин данных.

Блок **Сравнение** (Compare) позволяет подключить одну логическую шину данных, передающую значение **Истина** (True) или **Ложь** (False), на основе результатов сравнения значений **a** и **b**. Так, если вы выберете режим **Равно** (Equal to) (=), блок передаст значение **Истина** (True), если **a** равно **b**.

Программа CompareValues, показанная на рис. 15.8, демонстрирует блок **Сравнение** (Compare) в действии.

С помощью данной программы робот ожидает, когда значение параметра **Приближение** (Proximity) упадет ниже 80. Значение, которое заставило блок **Ожидание** (Wait) прекратить ожидание, переносится в блок **Сравнение** (Compare), и он определяет, меньше ли полученное значение 40. Если это так (**Истина** (True)), SK3TCHBOT говорит «Вниз»; если нет (**Ложь** (False)) — «Вверх».

Блок Логические операции

Блок **Логические операции** (Logic Operations) сравнивает два значения, полученные из логических шин данных, и выдает результат тоже с помощью логической шины данных. В режиме **И** (AND) он выполняет проверку, истинны ли оба входных значения (**a** и **b**). Если это так, выходной результат будет **Истина** (True). Если один или оба входных значения ложны, выходной результат будет **Ложь** (False).

В этом режиме вы можете использовать блок **Логические операции** (Logic Operations), чтобы создать программу, рисующую на экране закрашенный круг, если нажата кнопка датчика касания и результат замера составляет меньше 50. Если не выполняется хотя бы одно из этих условий (или оба), на экране отобразится пустой круг, потому что на выводе блока **Логические операции** (Logic Operations) окажется значение **Ложь** (False). Рис. 15.9 демонстрирует программу LogicAnd.

Логические операции

При настройке блока **Логические операции** (Logic Operations) вы можете выбрать один из четырех режимов: **И** (AND), **ИЛИ** (OR), **Исключающее ИЛИ** (XOR) и **Исключение** (NOT). В каждом режиме блок по-разному сравнивает логические входные значения. Подходящий режим зависит от того, что должна выполнить ваша программа. Табл. 15.1 перечисляет доступные режимы, а также входные значения, при которых вывод будет равен **Истина** (True).

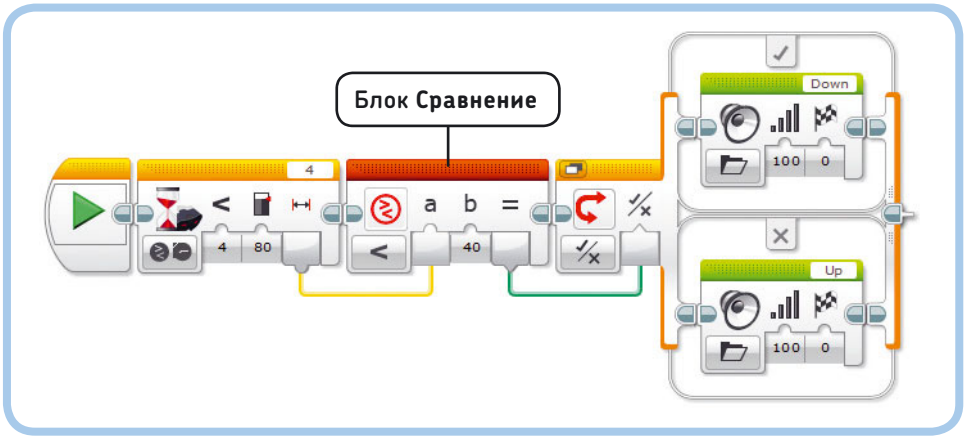


Рис. 15.8. Программа CompareValues

Таблица 15.1. Режимы блока Логические операции и соответствующие выходные значения

Режим		Результат Истина (True), если...
И (AND)		Оба входных значения истинны
ИЛИ (OR)		Одно или оба входных значения истинны
Исключающее ИЛИ (XOR)		Одно значение истинно, а другое — ложно
Исключение (NOT)		Входное значение ложно

В режиме **ИЛИ** (OR) вывод блока передает значение **Истина** (True), если одно или оба входных значения истинны, как это продемонстрировано в программе LogicOr на рис. 15.10. Программа постоянно проверяет, срабатывает ли датчик касания или инфракрасный датчик. Если срабатывают один или оба, выходным результатом будет **Истина** (True), что приведет к завершению цикла. Как только цикл завершается, блок **Звук** (Sound) воспроизводит звуковой сигнал. Этот метод полезен, поскольку позволяет программе ждать до тех пор, пока по крайней мере один из нескольких датчиков не сработает.

Теперь измените режим блока **Логические операции** (Logic Operations) на **Исключающее ИЛИ** (XOR). Когда запустите программу, вы будете слышать звук, когда срабатывает либо датчик касания, либо инфракрасный датчик, но звук не будет воспроизводиться, если оба датчика срабатывают одновременно.

Режим Исключение (NOT)

Если выбран режим **Исключение** (NOT), блок **Логические операции** (Logic Operations) содержит только один ввод. Этот режим попросту инвертирует входящее значение (**a**): если оно истинно, на вывод будет передано значение **Ложь** (False), а если ложно — то **Истина** (True).

Чтобы увидеть этот режим в действии, измените программу LogicOr, которую вы только что создали, удалив из нее блок **Инфракрасный датчик** (Infrared Sensor) и изменив режим блока **Логические операции** (Logic Operations) на **Исключение** (NOT). Вывод блока **Логические операции** (Logic Operations) будет передавать значение **Ложь** (False), если нажата кнопка датчика касания, и **Истина** (True), если кнопка отпущена. Таким образом, цикл будет выполняться, пока вы не отпустите кнопку датчика.

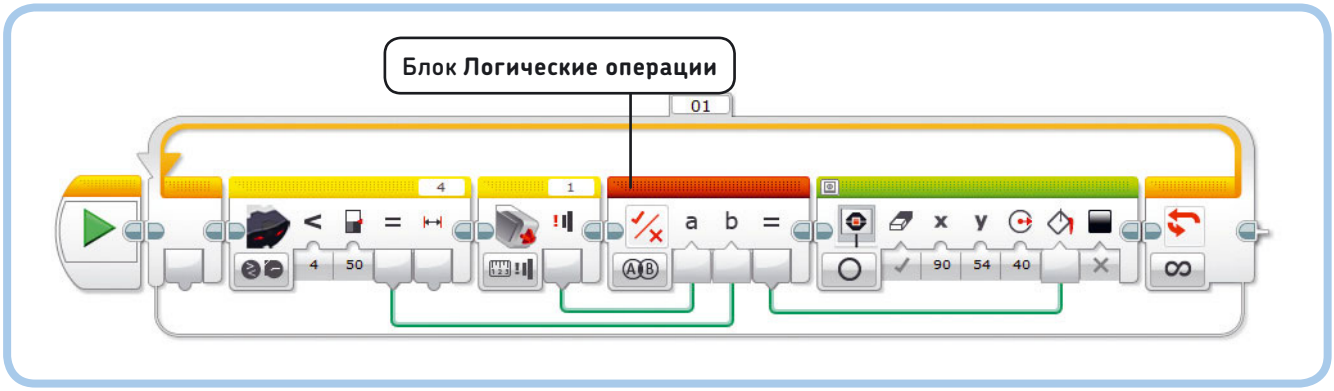


Рис. 15.9. Программа LogicAnd

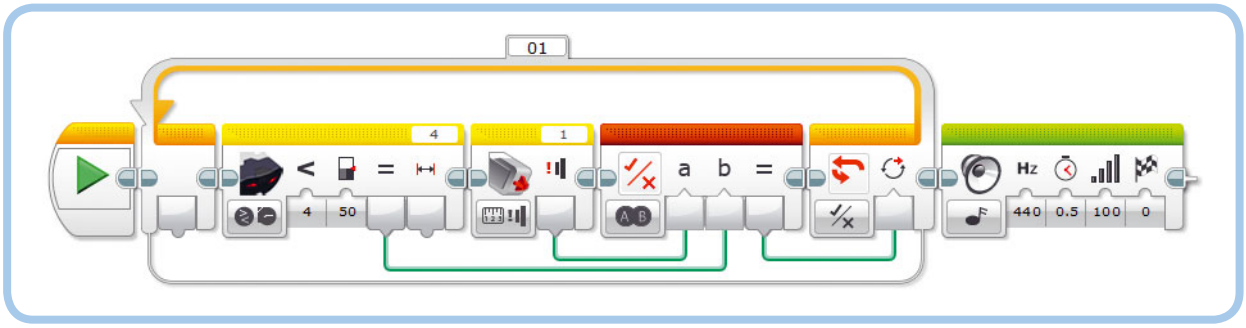


Рис. 15.10. Программа LogicOr

ПРАКТИКУМ № 95: ЛОГИЧЕСКИЕ ДАТЧИКИ!

Сложность: **Время:**

Программа LogicOr воспроизводит звук, когда срабатывает датчик касания или инфракрасный датчик, но она не сообщает, какой датчик стал причиной завершения цикла. Можно ли дополнить программу, чтобы робот произносил слово «Касание», если цикл завершил датчик касания, и «Обнаружено», если был задействован инфракрасный датчик?*

СОВЕТ **Поместите блок Переключатель (Switch) в режиме Логические операции (Logic) сразу же после блока Цикл (Loop) и используйте вывод имеющегося блока Датчик касания (Touch Sensor) для управления переключением.**

* Пункты **Touch** и **Detected** в каталоге звуков Lego.

ПРАКТИКУМ № 96: ДЛЯ ТРЕХ ДАТЧИКОВ!

Сложность: **Время:**

Можно ли разработать программу, которая воспроизводит звук при срабатывании одного из трех датчиков? Программа должна ожидать нажатия кнопки датчика касания, или значения приближения инфракрасного датчика ниже 50%, или яркости отраженного света выше 15%.

СОВЕТ **Взяв за основу программу LogicOr, добавьте блок Датчик цвета (Color Sensor) и еще один блок Логические операции (Logic Operations). Каким образом следует подключать шины данных?**

Блок Интервал

Блок **Интервал** (Range) определяет, находится ли передаваемое через числовую шину данных значение в диапазоне, ограниченном нижним и верхним пределами. Сами граничные значения должны находиться в рамках диапазона.

Если выбран режим **В пределах** (Inside), выходной результат блока будет равен **Истина** (True), если значение параметра **Тестовое значение** (Test Value) находится в пределах диапазона; вывод выдаст **Ложь** (False), если тестовое значение вне этих пределов.

Если выбран режим **За пределами** (Outside), происходит противоположное: выходной результат будет равен **Истина** (True), если тестовое значение находится за пределами указанного диапазона, и **Ложь** (False), если значение в рамках диапазона.

Программа SensorRange (рис. 15.11) содержит блоки **Интервал** (Range) (режим **В пределах** (Inside)) и **Цикл** (Loop) (режим **Логическое значение** (Logic)) и ожидает, пока показание приближения инфракрасного датчика не достигнет значения в диапазоне от 40 до 60, включая эти числа.

Блок Округление

Блок **Округление** (Round) позволяет превратить входное десятичное значение в целое число путем его округления. Вы можете выбрать режим **Округлить к большему** (Round Up) (к примеру, 1,2 или 1,8 становятся 2), **Округлить к меньшему** (Round Down) (1,2 или 1,8 становятся 1) или **До ближайшего** (Nearest) (1,2 становится 1, тогда как 1,5 или 1,8 становятся 2). Также вы можете выбрать режим **Отбросить дробную часть** (Truncate), чтобы отбросить десятичные знаки от числа без округления (1,877 становится 1,8, если вы оставляете только один десятичный знак).

Программа RoundTime (рис. 15.12) отображает время, прошедшее с момента запуска, на экране модуля EV3 путем округления десятичного значения таймера (к примеру, 3,508) до целого числа в секундах (3).

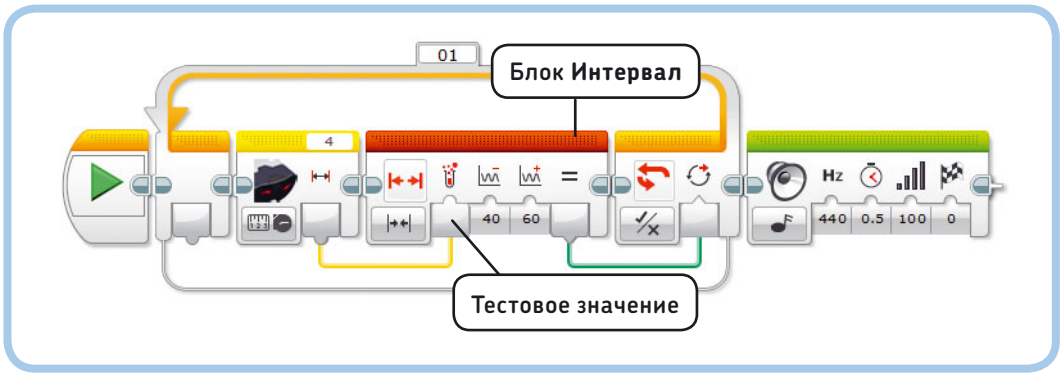


Рис. 15.11. Программа SensorRange

В предыдущих программах мы не использовали блок **Таймер** (Timer), но применять его предельно просто. Блок измеряет время, подобно секундомеру, начиная отсчет с нуля с момента запуска программы. Блок **Таймер** (Timer) в режиме **Измерение** (Measure) выдает текущее время в виде десятичного значения, например 1,500 для полутора секунд. Вы можете сбросить таймер до 0, выбрав режим **Сброс** (Reset) блока.

В этой программе вы можете использовать до восьми различных таймеров. Например, первый таймер для измерения длительности работы программы, второй — для подсчета времени, прошедшего с того момента, как была нажата кнопка датчика касания, обнуляя его при каждом нажатии кнопки. Чтобы указать, значение какого таймера следует считать или сбросить, вы должны выбрать один из восьми *идентификаторов таймера*. В программе RoundTime параметру **Идентификатор таймера** (Timer ID) присвоено значение 1, т.е. используется первый таймер.

Блок Текст

Блок **Текст** (Text) объединяет до трех текстовых вводов шин данных в текстовую строку. Например, если входные текстовые строки содержат значения «EV3 », «это » и «весело», вывод **Результат** (Result) выдаст текст «EV3 это весело»*. Необходимо добавлять пробелы после слов «EV3 » и «это », иначе вывод даст текст «EV3этовесело». Если ввод не содержит значения, он будет проигнорирован.

Можно использовать блок **Текст** (Text) для сочетания текста и чисел и вывода их на экран модуля EV3. Например, вы можете доработать предыдущую программу таким образом, чтобы она выводила текст «Time: 41 s», а не просто число. Чтобы сделать это, вы должны объединить слова «Time:», «s» и числовое значение времени и отобразить результат на экране, как показано в программе TextTime на рис. 15.13.

* Как уже упоминалось ранее в книге, поддерживаются только латинские символы.

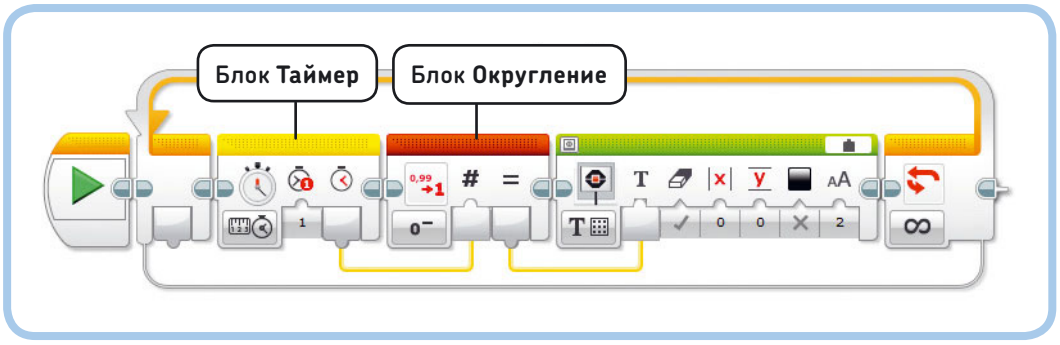


Рис. 15.12. Программа RoundTime

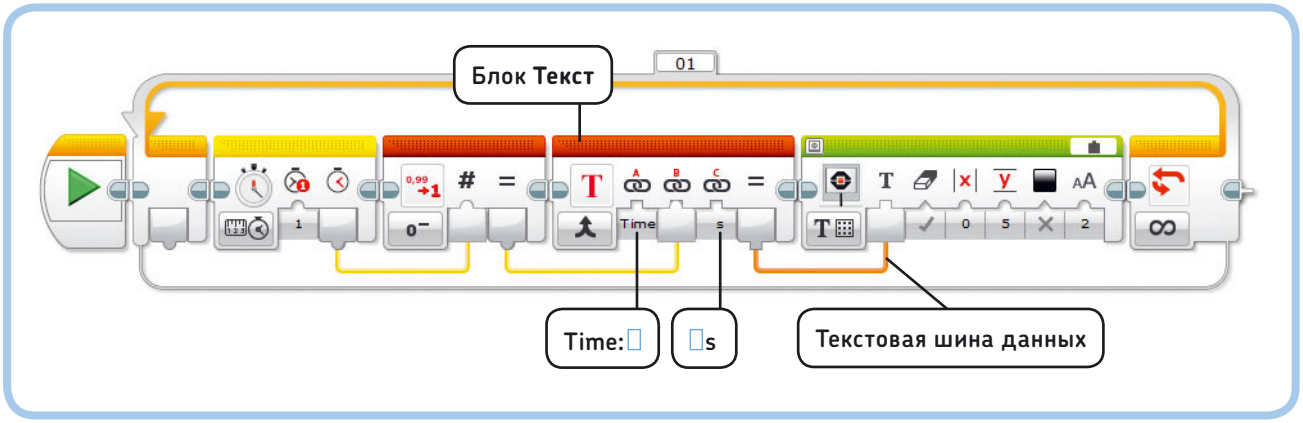


Рис. 15.13. Программа TextTime. Не забудьте о пробелах в словах «Time: », « s» (обозначены на рисунке синими прямоугольниками). При достижении 41 секунды работы программы на экране должен отобразиться текст «Time: 41 s»

ПРАКТИКУМ № 97: ОТСЧЕТ!

Сложность: **Время:**

Можно ли отобразить таймер обратного отсчета на экране модуля EV3? Пусть программа выполняет обратный отсчет от 60 до 0, показывая оставшееся время («46 s to go!»), и пусть воспроизведется оповещение, когда останется 5 секунд. Когда время истечет, робот должен произнести «Игра окончена»*.

СОВЕТ **Используйте программу TextTime в качестве основы. Вместо того чтобы отображать истекшее время, выведите результат этого вычисления: Результат = 60 – Истекшее время. Используйте блоки Сравнение (Compare) для определения, когда оставшееся время составит 5 секунд, а когда оно истечет совсем.**

Создание контейнеров «Мой блок» с шинами данных

До сих пор вы использовали шины данных для передачи информации между существующими программными блоками, например блоками датчиков, действий и операций с данными. Вы можете использовать шины данных со специфическими контейнерами «Мой блок». Благодаря этому вы сможете разрабатывать контейнеры «Мой блок» с *параметрами* (вводами и выводами). Таким образом, контейнер «Мой блок» может обрабатывать значения, переданные в него с помощью шин данных.

В главе 5 вы узнали, как создать базовую версию контейнера «Мой блок». В этом разделе вы научитесь создавать контейнеры «Мой блок» с вводами и выводами и получите некоторые рекомендации по созданию контейнеров «Мой блок».

Контейнеры «Мой блок» с вводами

Для начала вы создадите контейнер «Мой блок» с двумя вводами и именем **DisplayNumber**, как показано на рис. 15.14. Задача этого блока состоит в том, чтобы

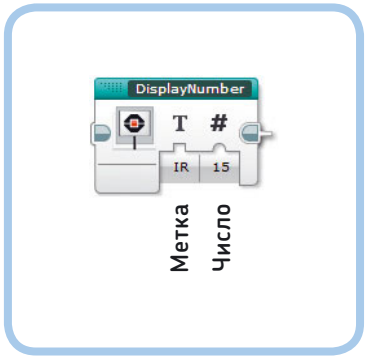


*Рис. 15.14. Контейнер **DisplayNumber**. В этой конфигурации он выводит текст «IR: 15»*

объединять данные, получаемые из текстового ввода **Метка** и числового ввода **Число**, и отображать результат на экране модуля EV3.

Этот блок с легкостью позволит отобразить на экране число с текстовой меткой, которая будет его описывать. Например, если вы присвоите значение IR параметру **Метка** и значение 15 параметру **Число**, блок отобразит на экране текст «IR: 15».

Создайте контейнер «Мой блок», выполнив следующие шаги.

1. Создайте новую программу под названием NumberTest. Она понадобится вам для проверки работы контейнера «Мой блок», когда он будет готов.
2. Установите и настройте блок **Текст** (Text) и блок **Экран** (Display) в области программирования, как показано на рис. 15.15. Вы будете использовать блок **Текст** (Text) для объединения значения ввода **Метка**, двоеточия, за которым следует пробел, и значения ввода **Число** в одну текстовую строку, и блок **Экран** (Display) для дальнейшего вывода ее на экран. Затем с помощью мыши выделите оба блока, заключив их в рамку выделения.

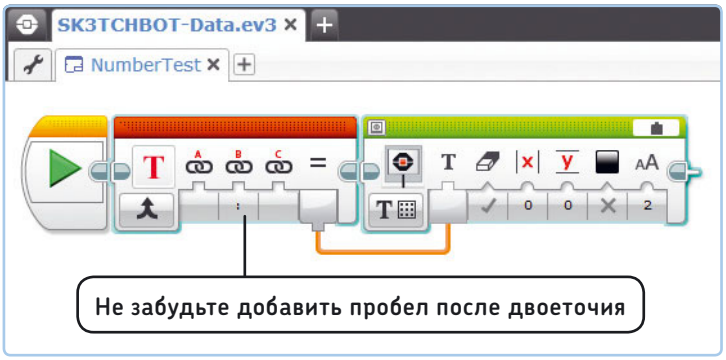


*Рис. 15.15. Настройте блоки **Текст** и **Экран**, как показано на рисунке, а затем выделите оба блока*

* Пункт **Game over** в каталоге звуков Lego.

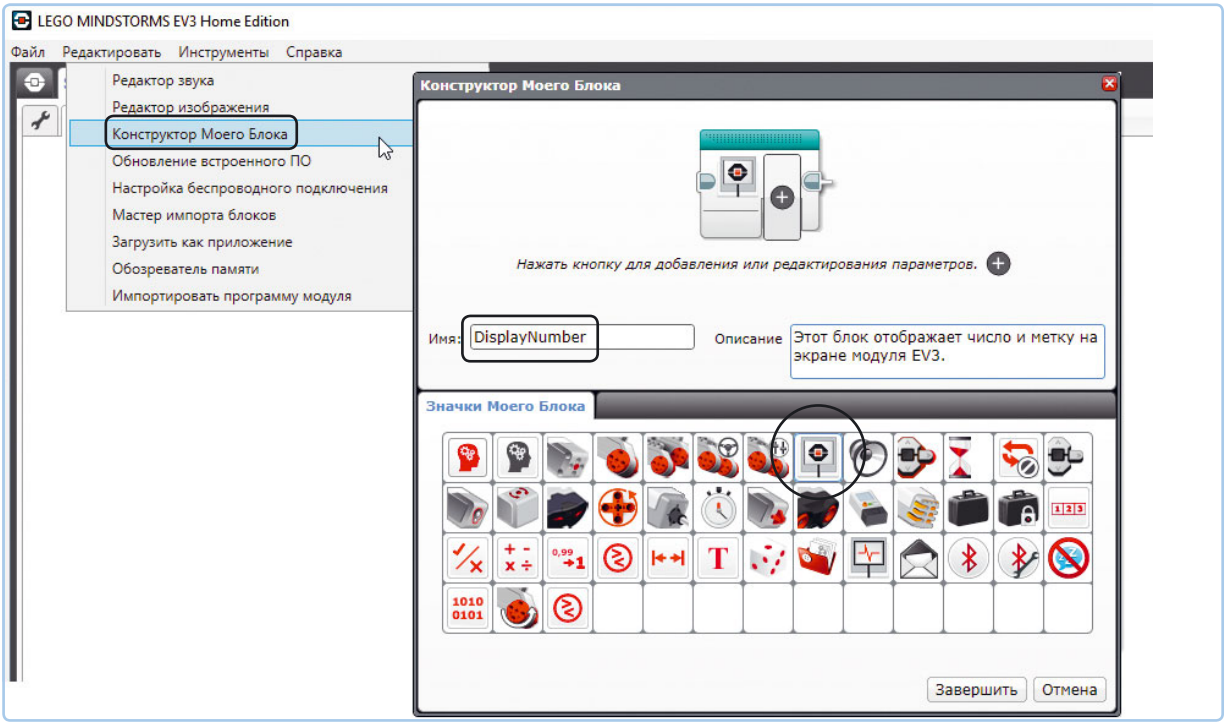


*Рис. 15.16. Откройте окно **Конструктор Моего Блока**, укажите имя и описание контейнера «Мой блок», а затем выберите для него значок. Так как этот блок будет выводить данные на экран модуля EV3, рекомендую выбрать выделенный значок*

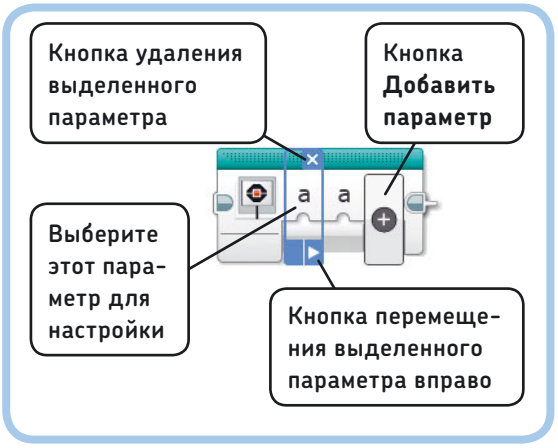


Рис. 15.17. Добавление, удаление и изменение порядка следования параметров

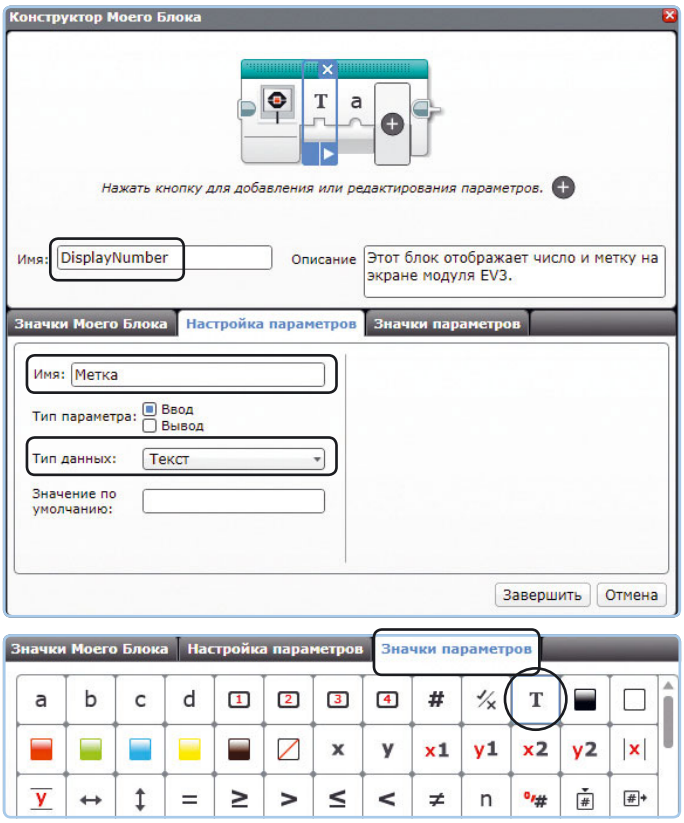


*Рис. 15.18. Настройка ввода **Метка***

3. Создайте новый контейнер «Мой блок», выбрав команду меню **Инструменты ▸ Конструктор Моего Блока** (Tools ▸ My Block Builder).

4. Введите текст **DisplayNumber** в качестве имени контейнера «Мой блок», укажите описание и выберите значок для своего блока, как показано на рис. 15.16.

5. Теперь мы настроим параметры блока. Добавьте два ввода, дважды нажав кнопку **Добавить параметр** (Add Parameter), как показано на рис. 15.17.

6. Перейдите на вкладку **Настройка параметров** (Parameter Setup), выберите первый параметр и настройте его, указав тип параметра **Ввод** (Input), тип данных **Текст** (Text) и имя **Метка**, как показано на рис. 15.18. Выберите информативный значок, например в виде буквы «Т», для этого ввода на вкладке **Значки параметров** (Parameter Icons).

7. Теперь выберите второй параметр и настройте его, указав тип параметра **Ввод** (Input), тип данных **Число** (Number) и имя **Число**, как показано на рис. 15.19. В поле **Значение по умолчанию** (Default Value) указано число 0, и это означает, что параметру **Число** по умолчанию присвоится значение 0, когда позже вы выберете контейнер на палитре программирования. Выберите стандартный тип ввода в группе элементов управления **Стиль параметра** (Parameter Style) (выделен на рисунке). В этом случае будет создан стандартный числовой ввод, который может принимать любое значение (как вводы блока **Математика** (Math)), а не ползунковый регулятор, который принимает значения только в определенном диапазоне (как, например, параметры **Мощность** (Power) и **Рулевое управление** (Move Steering) блока **Рулевое управление** (Move Steering)).

8. Нажмите кнопку **Завершить** (Finish). Теперь вы можете просматривать содержимое созданного контейнера **DisplayNumber** на его собственной вкладке внутри проекта, как показано на рис. 15.20. Вы должны увидеть параметры **Метка** и **Число**, хотя шины еще не подключены. Эти параметры переносят значения, *переданные в* контейнер «Мой блок» *из основной программы*. Таким образом, если указать значения IR и 15, как показано на рис. 15.14, параметр **Метка** передает значение IR, а **Число** — значение 15.

9. Завершите создание контейнера «Мой блок» путем подключения выводов **Метка** и **Число** к блоку **Текст** (Text), как показано на рис. 15.20, и сохраните проект. Помните, что блок **Текст** (Text) объединяет данные с вводов в одну текстовую строку, а блок **Экран** (Display) отображает ее на экране.

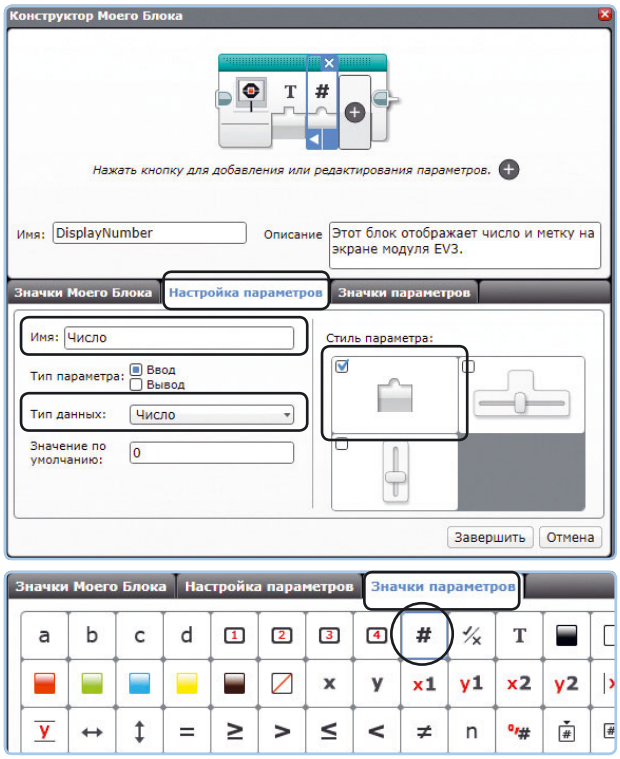


*Рис. 15.19. Настройка ввода **Число***

Поздравляем, вы создали контейнер «Мой блок» с вводами! Он размещается на голубой вкладке палитры программирования. Теперь завершите программу NumberTest, чтобы протестировать новый контейнер, как показано на рис. 15.21. В этом примере основная программа (NumberTest) передает два значения в контейнер **DisplayNumber**. Блоки внутри

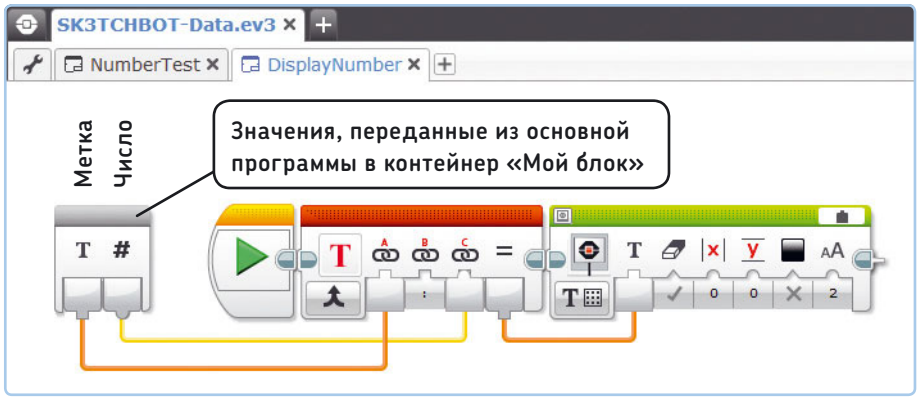


*Рис. 15.20. После нажатия кнопки **Завершить** в окне **Конструктор Моего Блока** вы увидите содержимое контейнера **DisplayNumber** на отдельной вкладке в проекте SK3TCHBOT-Data. Завершите работу с блоком посредством подключения шин данных так, как показано на рисунке*

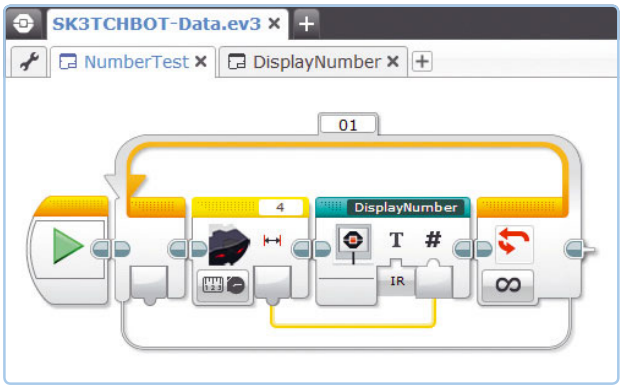


*Рис. 15.21. Программа NumberTest. Блок **Инфракрасный датчик** отправляет значение параметра **Приближение** на числовой ввод **Число** контейнера «Мой блок». Значение параметра **Метка** введено вручную (**IR**). Контейнер «Мой блок» объединяет значение параметра **Метка** с двоеточием, пробелом и значением инфракрасного датчика (параметра **Приближение**) и выводит результат на экран модуля EV3. При получении результата приближения, равного, к примеру, 65 на экране отобразится строка «IR: 65»*

контейнера «Мой блок» извлекают эти значения и отображают их на экране модуля EV3.

Изменение контейнеров «Мой блок»

После того как контейнер «Мой блок» создан, вы можете настроить его функции, дважды щелкнув по нему мышью и изменив вложенные блоки. Например, можно добавить блок **Звук** (Sound) в контейнер **DisplayNumber**, чтобы робот воспроизводил сигнал каждый раз, когда на экране модуля отображается значение.

На момент написания этой книги программное обеспечение LEGO MINDSTORMS EV3 (версия 1.2) не позволяло изменять входные и выходные параметры контейнеров «Мой блок» после того, как завершено его создание в окне **Конструктор Моего Блока** (My Block Builder). Если вы хотите изменить параметры или добавить новые, вам придется создать новый контейнер «Мой блок» с нуля. После того как блок с нужными параметрами создан, скопируйте содержимое старого блока и вставьте его в новый, чтобы сэкономить время.

Контейнеры «Мой блок» с выводами

Кроме контейнеров «Мой блок» с вводами, вы можете создавать пользовательские блоки с выводами. В этом разделе вы соберете контейнер «Мой блок», определяющий направление движущихся объектов с помощью

* Мотор B: 375 оборотов.

ПРАКТИКУМ № 98: МОЙ МОДУЛЬ!

Сложность: **Время:**

Создайте новый контейнер «Мой блок» на основе контейнера **DisplayNumber** с дополнительным текстовым вводом с именем **Модуль** (см. рис. 15.22). Затем добавьте метку и единицу измерения значения. Например, отображение значения, получаемого от датчика вращения, можно сопроводить текстовой меткой «MB» и единицей измерения «Deg»: «MB: 375 Deg» (сокращение для строки «Motor B: 375 Degrees»*).

СОВЕТ **Вам понадобится дополнительный блок Текст (Text).**

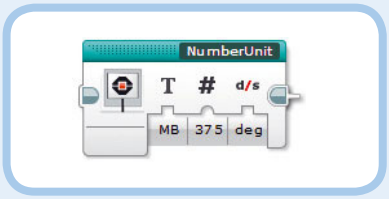


Рис. 15.22. Контейнер «Мой блок» из практикума № 98

ПРАКТИКУМ № 99: ПРОДВИНУТЫЙ ИНТЕРФЕЙС!

Сложность: **Время:**

Контейнер **DisplayNumber**, который вы только что создали, применим для отображения одного значения в верхней части экрана. Можно ли создать продвинутую версию этого контейнера «Мой блок» с возможностями выбора номера строки и удаления содержимого экрана, как показано на рис. 15.23?

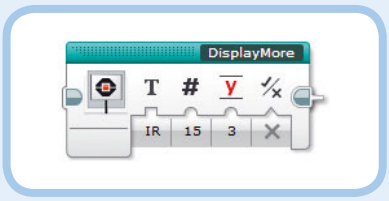


Рис. 15.23. Контейнер «Мой блок» из практикума № 99

СОВЕТ **На экране модуля EV3 пространства достаточно для размещения шести текстовых строк, если вы используете самый большой размер шрифта. Используйте блок Математика (Math), чтобы преобразовать значение на вводе КоличествоСтрок контейнера «Мой блок» в значение для параметра Строка (Row) блока Экран (Display): Строка (Row) = КоличествоСтрок × 2.**

инфракрасного датчика. Контейнер **Direction** будет содержать один логический вывод с именем **Приближение**, как показано на рис. 15.24.

Вывод будет отправлять значение **Истина** (True), если объект приближается к датчику (расстояние до объекта сокращается); в противном случае вывод передаст значение **Ложь** (False) (если расстояние до объекта увеличивается или остается неизменным). Создайте контейнер **Direction** и тестовую программу DirectionSound, следуя приведенным ниже инструкциям.

1. Создайте новую программу с именем DirectionSound и поместите в нее два блока **Инфракрасный датчик** (Infrared Sensor), блок **Ожидание** (Wait) и блок **Сравнение** (Compare), как показано на рис. 15.25. Эти блоки выполняют два замера приближения с разницей в 0,2 секунды и сравнивают их, определяя, является ли второе значение (**a**) меньше, чем первое (**b**). Если это так, объект приближается к датчику, и вывод блока **Сравнение** (Compare) выдает значение **Истина** (True).

2. Выделите все четыре блока и выберите команду меню **Инструменты** ▸ **Конструктор Моего Блока** (Tools ▸ My Block Builder). Укажите слово **Direction** в качестве имени блока и выберите значок в виде инфракрасного датчика, как показано на рис. 15.26.

3. Добавьте один параметр и настройте его, указав тип параметра **Вывод** (Output), тип данных **Логическое значение** (Logic) и имя **Приближение**. Затем выберите подходящий значок, как показано на рис. 15.26. Нажмите кнопку **Завершить** (Finish).

4. Чтобы завершить работу с контейнером «Мой блок», подключите вывод блока **Сравнение** (Compare) к вводу **Приближение**, как показано на рис. 15.27. Затем это значение передается в основную программу, которая содержит контейнер **Direction**.

Теперь вернемся к программе DirectionSound и протестируем наш контейнер «Мой блок», заставив робота воспроизвести высокий тон, когда объект приближается (**Истина** (True)), и низкий, если объект отдаляется или стоит на месте (**Ложь** (False)), как показано на рис. 15.28. В этом примере блоки внутри контейнера **Direction** вычисляют направление движущегося объекта и передают результат в основную программу (DirectionSound) через вывод **Приближение**.

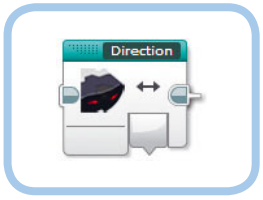


Рис. 15.24. Контейнер Direction

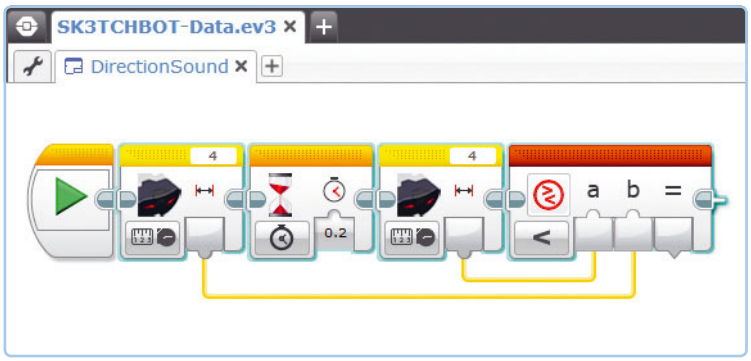


*Рис. 15.25. Разместите и настройте блоки в контейнере **Direction**, как показано на рисунке. Когда все будет готово, выделите блоки мышью и выберите команду меню **Инструменты** ▸ **Конструктор Моего Блока***

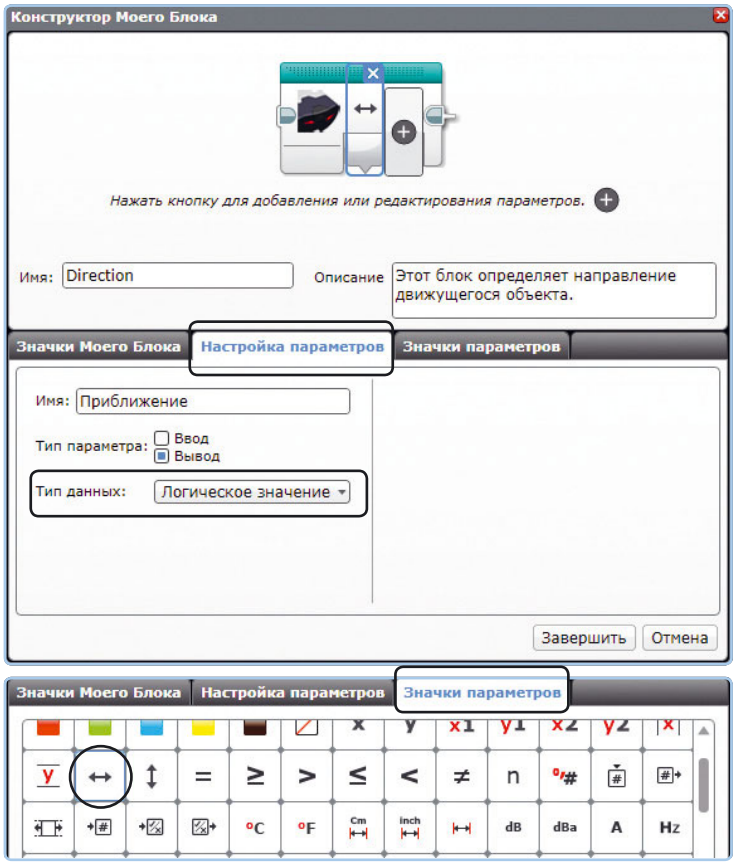


Рис. 15.26. Выполните настройку контейнера «Мой блок» и его логического вывода, как показано на рисунке

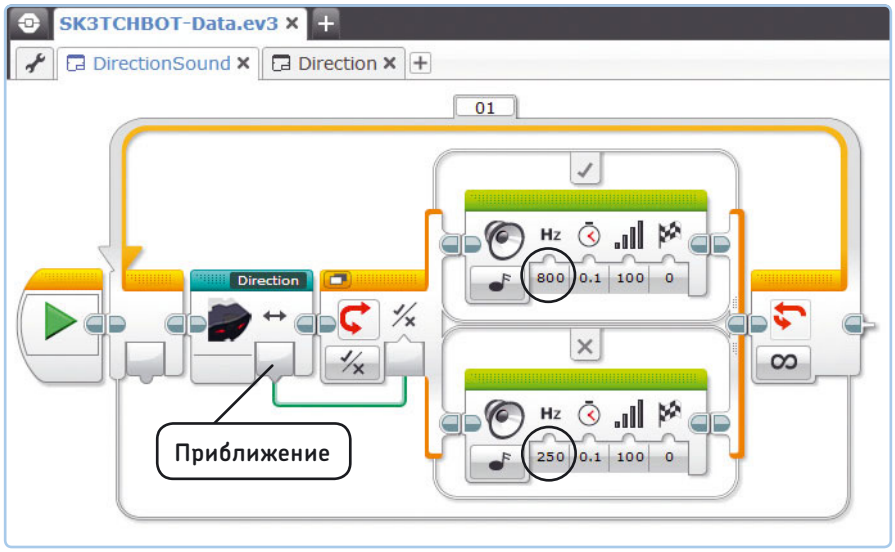


*Рис. 15.27. Программа DirectionSound использует логическое значение, передаваемое на вывод **Приближение**, для воспроизведения высокого тона, если какой-либо объект приближается, и низкого тона, если объект не приближается*

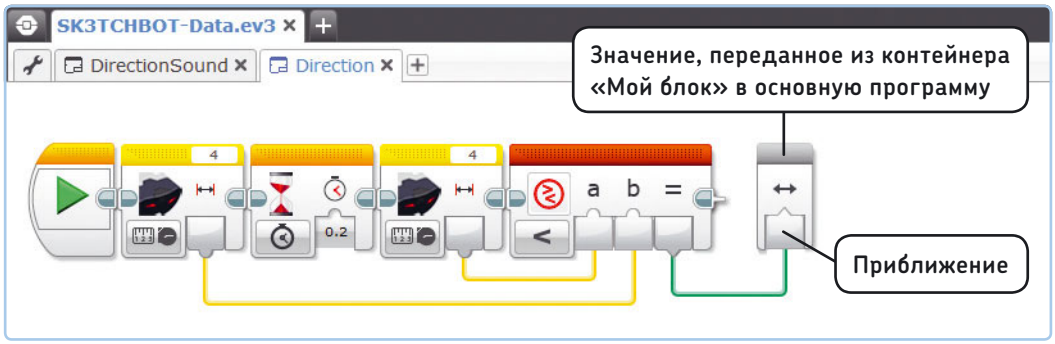


*Рис. 15.28. Подключите вывод блока **Сравнение** к вводу **Приближение***

ПРАКТИКУМ № 100: СРЕДНЕЕ ЗНАЧЕНИЕ ПРИБЛИЖЕНИЯ!

Сложность:

Время:

Создайте контейнер «Мой блок», который вычисляет среднее значение из двух замеров приближения. Затем он должен выдавать результат с помощью числовой шины данных. Как и в контейнере **Direction**, измерения следует делать с интервалом в 0,2 секунды.

ПРАКТИКУМ № 101: СКОРОСТЬ ПРИБЛИЖЕНИЯ!

Сложность: **Время:**

Создайте контейнер «Мой блок» с одним числовым выводом, который определяет скорость движущегося объекта путем вычисления скорости изменения значений приближения. Для этого блок должен выполнить два замера, как и в контейнере **Direction**, а затем вычислить скорость изменений по следующей формуле:

$$\text{Скорость} = \frac{\text{Второе измерение} - \text{Первое измерение}}{\text{Время между измерениями}}$$

Чтобы выполнить проверку блока, отобразите значение скорости на экране и измените подсветку модуля на зеленый цвет, если объект приближается, оранжевый, если объект не двигается, и красный, если объект удаляется от датчика. Как можно определить направление, используя значение скорости?

Контейнеры «Мой блок» с вводами и выводами

В следующем примере вы создадите контейнер «Мой блок» с вводами и выводами, показанный на рис. 15.29. Контейнер **IsEven** определяет, является ли четным числовое значение на вводе **Число**. Если это так, логический вывод **Четность** передаст значение **Истина** (True); в противном случае — значение **Ложь** (False). Создайте блок, выполнив следующие шаги.

1. Создайте новую программу с именем EvenSound и разместите блоки **Математика** (Math) и **Сравнение** (Compare) в области программирования. Пока оставьте их настройки без изменений.
2. Выделите оба размещенных блока и выберите команду меню **Инструменты** ▸ **Конструктор Моего Блока** (Tools ▸ My Block Builder). Введите текст **IsEven** в качестве имени блока и выберите соответствующий значок, как показано на рис. 15.29.
3. Добавьте один числовой ввод и присвойте ему имя **Число** и один логический вывод с именем **Четность**. Выберите значки, как показано на рис. 15.29. Нажмите кнопку **Завершить** (Finish).
4. Теперь завершите работу с контейнером «Мой блок», настроив оба блока и подключив шины данных, как показано на рис. 15.30. В блоке **Математика** (Math) используется оператор деления по модулю (%), который вычисляет остаток от деления двух чисел. В этом примере он сообщает остаток от деления входного значения **a** на 2. Четные числа делятся на 2, поэтому остаток будет равен 0, и блок **Сравнение** (Compare) выдаст значение **Истина** (True). Для нечетных чисел остаток не будет равен 0, поэтому блок **Сравнение** (Compare) выдаст значение **Ложь** (False). Например, выражение 7/2 даст в остатке 1, и блок **Сравнение** (Compare) выдаст значение **Ложь** (False), сообщающее, что на ввод передано нечетное число.

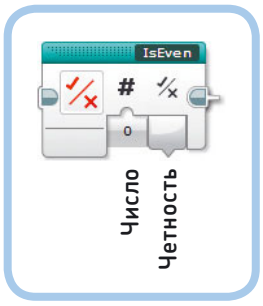


Рис. 15.29. Контейнер IsEven

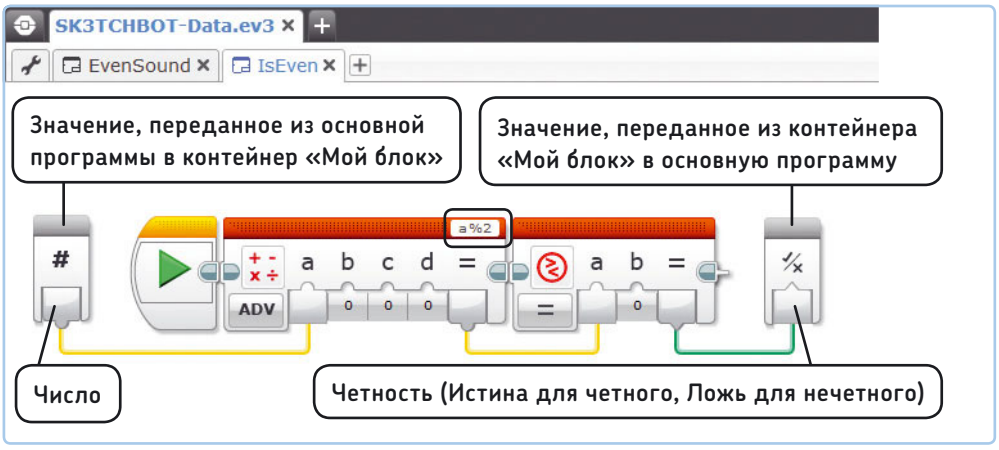


*Рис. 15.30. Настройка блоков в контейнере **IsEven***

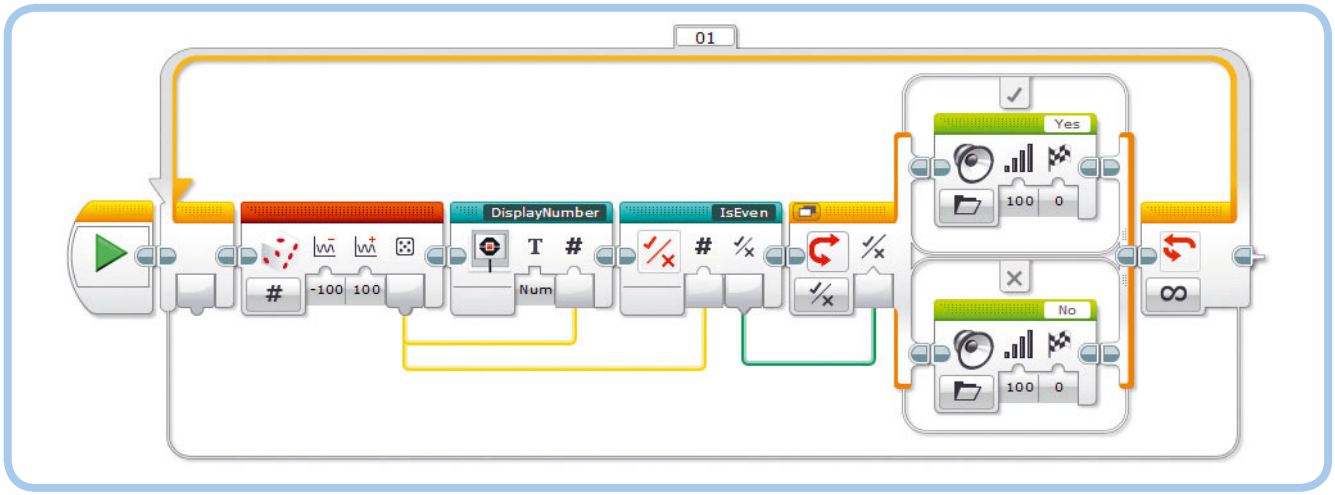


Рис. 15.31. Программа EvenSound

Теперь, когда вы закончили работу с контейнером «Мой блок», протестируйте его, завершив программу EvenSound, показанную на рис. 15.31. Программа отображает случайное число в диапазоне от –100 до 100 на экране модуля EV3 и определяет, четное оно или нет, используя контейнер **IsEven**. Робот произносит «Да» при обнаружении четных чисел и «Нет» — при нечетных.

ПРАКТИКУМ № 102: МАТЕМАТИКА КРУГА!

Сложность: **Время:**

Можете ли вы создать контейнер «Мой блок», который будет вычислять длину окружности и площадь круга, используя радиус круга в качестве входного значения? Создайте контейнер «Мой блок» с одним числовым вводом с именем **Радиус** и двумя числовыми выводами **Окружность** и **Площадь**.

СОВЕТ **Используйте следующие формулы, учитывая, что число π (pi) приблизительно равняется 3,14:**

Окружность = 2 × π × Радиус
Площадь = Радиус^2 × π

Рекомендации по использованию контейнеров «Мой блок»

Контейнеры «Мой блок» с вводами и выводами позволяют выполнять множество различных задач. Их можно использовать в следующих случаях.

Повторяющиеся задачи: некоторые действия одинаковы для многих программ, как, например, вывод значений на экран модуля. Если вы один раз создадите контейнер «Мой блок» для выполнения подобной задачи, в долгосрочной перспективе вы сэкономите много времени.

Организация блоков: вы можете собрать блоки сложной программы в контейнеры «Мой блок», чтобы было проще разобраться в ней и по отдельности протестировать ее части. Например, если требуется создать программу для автономного робота-манипулятора, можно собрать один контейнер «Мой блок» для определения местоположения объекта, второй — для перемещения к нему и третий для подъема и переноса объекта.

Обработка информации: поскольку вы создаете программы, которые обрабатывают значения с помощью блоков операций с данными, будет сложно понять, как на самом деле работает программа, когда вы взглянете на нее спустя некоторое время. Чтобы решить эту проблему, вы можете раздробить сложные вычисления на более мелкие части и сгруппировать их в контейнеры «Мой блок». Например, в программе EvenSound (см. рис. 15.31) сокрытие операции деления по модулю в контейнере **IsEven** упрощает понимание, как работает программа.

Способы создания контейнеров «Мой блок»

Вы можете создавать контейнеры «Мой блок» двумя способами.

Преобразовать группу блоков в контейнер «Мой блок»: используйте этот способ, если вы уже настроили и протестировали блоки в вашей программе. Дробление большой программы на подразделы, каждый из которых обладает собственной функциональностью, — весьма полезный метод. Если группа блоков подключена к другим блокам в вашей программе с помощью шин данных, вам понадобится настроить параметры контейнера «Мой блок» для каждой шины данных. На самом деле, инструмент **Конструктор Моего Блока** (My Block Builder) автоматически настраивает такие параметры, но вам все-таки придется добавить имя и значок для каждого из них.

Создать контейнер «Мой блок» с нуля: используйте этот способ, если знаете, что должен делать ваш контейнер «Мой блок», но пока не уверены в том, как настроить блоки внутри него, чтобы выполнить задачу. Например, если вы хотите создать блок, который будет определять, является число четным или нечетным, но не знаете, как точно он будет работать, вы можете начать с создания контейнера «Мой блок» с одним числовым вводом и одним логическим выводом. Затем поэкспериментируйте с блоками внутри контейнера, пока не решите поставленную задачу. Обратите внимание, что вы не можете запустить инструмент **Конструктор Моего Блока** (My Block Builder), не выделив хотя бы один блок, поэтому вам придется начать с временного блока, например **Ожидание** (Wait), и позже удалить его.

Использование блоков в разных проектах

Когда вы создаете контейнер «Мой блок», он становится доступен для использования только внутри текущего проекта. Чтобы использовать его в рамках другого проекта, вам придется скопировать его или экспортировать из вашего текущего проекта и вставить или импортировать его в другой проект. (См. раздел «Управление контейнерами “Мой блок” в проектах» главы 5.)

Дальнейшее изучение

В этой главе вы узнали о блоках операций с данными, а также о том, как создавать контейнеры «Мой блок» с вводами и выводами для подключения шин данных. Эти методы позволят вашему роботу обрабатывать и объединять значения, получаемые от датчиков, чтобы использовать их в качестве входных данных для действий, таких как перемещение и звуковое оповещение. Вы увидите множество практических примеров в оставшихся главах этой книги. А теперь укрепите приобретенные навыки, выполняя приведенные ниже практикумы.

ПРИМЕЧАНИЕ **Помните, что вы можете найти решения для многих практикумов по ссылке eksmo.ru/files/Lego_Mindstorms_Primers.zip.**

ПРАКТИКУМ № 103: ЭТО ЦЕЛОЕ ЧИСЛО?

Сложность: **Время:**

Можно ли создать контейнер «Мой блок», который определяет, каким числом представлено входящее числовое значение — целым или десятичным? Создайте контейнер «Мой блок» с именем **IsInteger** и с параметрами, аналогичными контейнеру **IsEven**, но используйте другие вложенные блоки.

СОВЕТ **Округлите входящее значение и сравните его с исходным. Почему в некоторых случаях входящее значение идентично округленному?**

ПРАКТИКУМ № 104: ДВОЙНАЯ ПОТЕРЯ СКОРОСТИ!

Сложность: **Время:**

Можно ли создать программу, с помощью которой оба мотора, B и C, будут вращаться до тех пор, пока один из них не заглохнет? Когда программа будет завершена, вложите блоки, отвечающие за ожидание остановки мотора, в контейнер «Мой блок» с именем **WaitForStall**, чтобы вы могли использовать его в любой из ваших программ. Он будет особенно полезен для таких роботов, как EXPLOR3R и FORMULA EV3.

СОВЕТ **Часть этой программы похожа на программу LogicOr, показанную на рис. 10.15. Используйте два блока Вращение мотора (Motor Rotation) в режиме Сравнение ▸ Текущая мощность (Compare ▸ Current Power).**

ПРАКТИКУМ № 105: ТЕСТ НА РЕФЛЕКС!

Сложность: **Время:**

Можно ли создать программу для проверки вашей реакции? Сделайте так, чтобы цвет индикатора состояния оставался зеленым в течение произвольного времени, а затем неожиданно загорался красным. Как только вы увидите красный свет, вы должны нажать кнопку датчика касания. Затем программа должна отобразить, сколько времени прошло с момента, когда была включена красная подсветка, до нажатия кнопки. Вложите программу в цикл, чтобы увидеть, сможете ли вы улучшить время вашей реакции! Когда будете готовы, дополните программу таким образом, чтобы избегать жульничества с преждевременным нажатием кнопки датчика касания.

СОВЕТ **Работа программы заключается в следующем: установить зеленый цвет подсветки. Подождать некоторое время. Установить красный цвет подсветки. Сбросить таймер. Подождать нажатия кнопки датчика касания. Отобразить значение таймера на экране. Подождать 3 секунды, чтобы вы могли прочитать результат.**

СДЕЛАЙ САМ № 26: РОБОТИЗИРОВАННЫЕ ЧАСЫ!

Сборка: **Программирование:**

Можно ли сконструировать работающие часы EV3? Используйте три мотора из набора MINDSTORMS EV3, чтобы управлять часовой, минутной и секундной стрелками на ваших часах. Используйте блок **Таймер** (Timer) и вычисляйте положение каждой стрелки, основываясь на количестве секунд, которые прошли с момента запуска программы.

СОВЕТ **Во время тестирования робота увеличьте значение таймера в 10 или более раз, чтобы упростить проверку движения часовой стрелки.**

16

Использование констант и переменных

Если вы добрались до этой главы, значит, вы уже почти овладели всеми навыками программирования, описанными в этой книге. Вы научились применять множество различных программных блоков и освоили работу с основными инструментами, такими как шины данных. Эта глава завершает раздел, посвященный программированию, и из нее вы узнаете, как использовать константы и память EV3 для работы с переменными.

Использование констант

Блок **Константа** (Constant) представляет собой начальную точку шины данных, значение которой вы можете указать вручную. Тип шины данных определяется выбором режима (**Текст** (Text), **Числовое значение** (Numeric) или **Логическое значение** (Logic)), а значение вводится в поле **Значение** (Value), как показано на рис. 16.1.

Блок **Константа** (Constant) удобно использовать, если вам необходимо сконфигурировать параметры нескольких блоков с одинаковым значением. Например, программа ConstantDemo (рис. 16.1) заставляет моторы робота SK3TCHBOT вращаться с одинаковой скоростью, но в разных направлениях. В довершение всего блок **Константа** (Constant) отправляет значение 50 первому блоку **Большой мотор** (Large Motor) и блоку **Математика** (Math), который умножает значение на –1 перед тем, как передать его второму блоку **Большой мотор** (Large Motor). Чтобы одновременно изменить скорость вращения обоих моторов, достаточно просто указать другое значение в блоке **Константа** (Constant). Если бы этого блока не было, пришлось бы менять значение дважды.

Использование переменных

Представьте, что переменная — это чемодан, в котором можно хранить информацию. Когда программе нужно запомнить

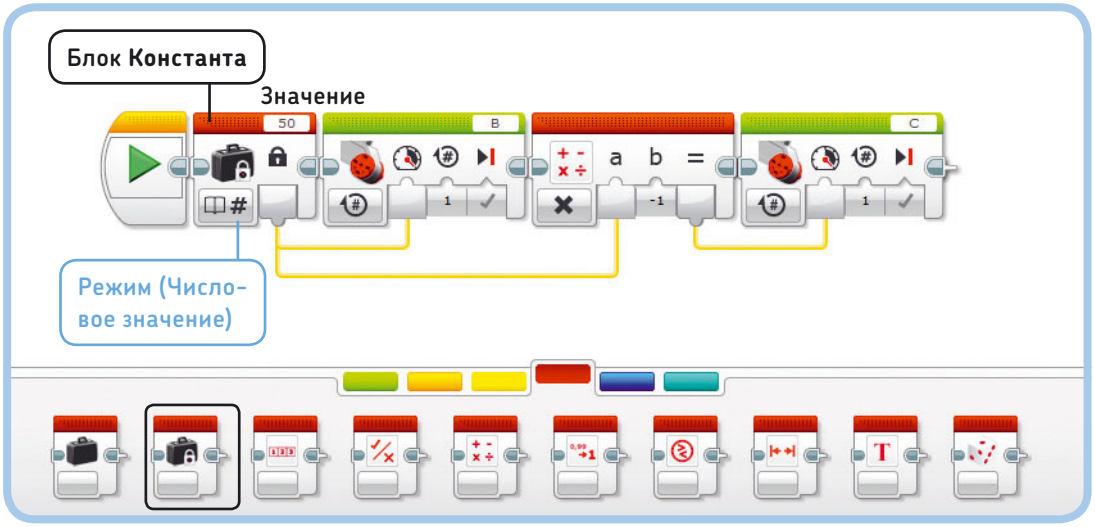


Рис. 16.1. Программа ConstantDemo. Для работы с программами этой главы создайте новый проект с именем SK3TCHBOT-Variable

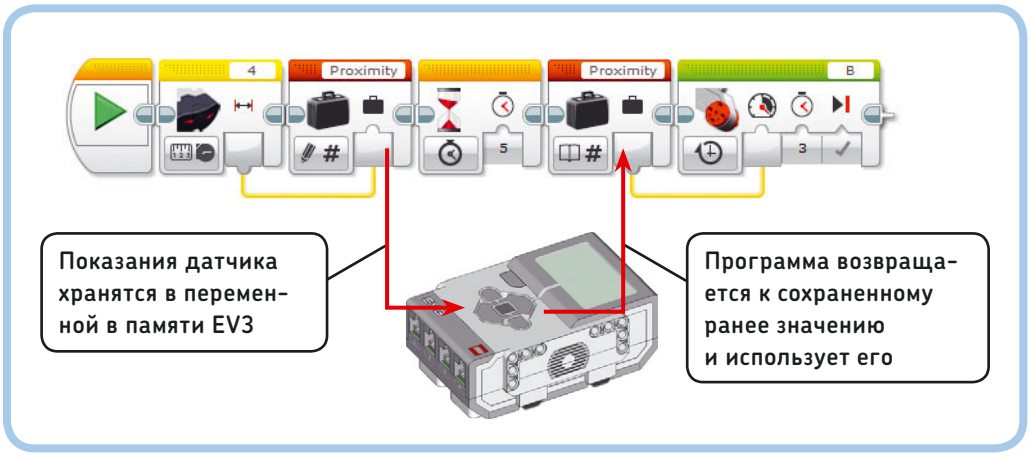


Рис. 16.2. Значение, к примеру расстояние, показанное на рисунке, можно хранить в переменных в памяти модуля EV3. Далее вы узнаете, на что способна эта программа и как ее написать

значение (к примеру, показания датчика), чтобы обратиться к нему в дальнейшем, она помещает значение в этот чемодан и убирает его в сторонку. Когда программе необходимо применить это значение, она открывает «чемодан» и использует сохраненное значение. Переменная хранится в памяти модуля EV3, пока не понадобится.

К хранящейся в переменной информации можно получить доступ из других частей программы. К примеру, можно поместить показания приближения инфракрасного датчика в этот «чемодан» и использовать их для управления скоростью вращения мотора через 5 секунд.

Программа может получить доступ к сохраненной информации в любой момент во время работы, но, как только она завершит работу, данные будут утеряны. Для сохранения и применения информации из переменных используйте блок **Переменная** (Variable), который можно узнать по значку в виде чемодана. На рис. 16.2 вкратце показано, что происходит при использовании переменных.

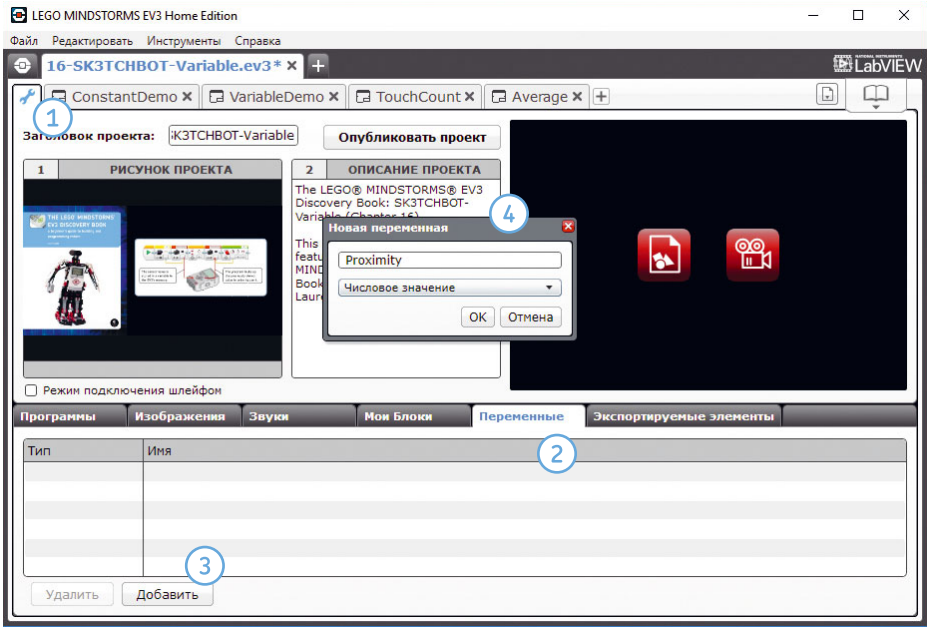


*Рис. 16.3. Определение переменной на странице **Свойства проекта**. Шаг 1: откройте страницу **Свойства проекта**. Шаг 2: перейдите на вкладку **Переменные**. Шаг 3: чтобы создать новую переменную, нажмите кнопку **Добавить**. Шаг 4: введите имя новой переменной — **Proximity** и выберите тип данных **Числовое значение**. Затем нажмите кнопку **OK***

ПРИМЕЧАНИЕ **В списке Название переменной (Name) содержатся только те переменные, тип которых подходит к выбранному режиму блока. Например, чтобы просмотреть только что созданную числовую переменную Proximity, блок должен находиться в режиме Считывание ▸ Числовое значение (Read ▸ Numeric) или Записать ▸ Числовое значение (Write ▸ Numeric).**

Определение переменной

У каждой переменной есть имя, тип данных и содержащееся в ней значение. Например, переменной с числовым значением можно присвоить имя **Proximity** и значение **56**. Кроме числовых переменных существуют логические (содержат значения **Истина** (True) или **Ложь** (False)) и текстовые (содержат строку текста, например «Hello»*).

* Обратите внимание, что как в именах, так и в значениях переменных можно использовать только латинские буквы, цифры и некоторые специальные символы.

Перед использованием переменной в программе необходимо *определить* ее, указав имя и тип. Это можно сделать на странице **Свойства проекта** (Project Properties), как показано на рис. 16.3, или использовать блок **Переменная** (Variable), как вы скоро увидите.

Определив переменную в проекте, вы сможете использовать ее в любой программе в рамках данного проекта.

Для удаления переменной откройте вкладку **Переменные** (Variables) на странице **Свойства проекта** (Project Properties), выберите переменную, которую хотите удалить, и нажмите кнопку **Удалить** (Delete).

Использование блока Переменная

Определив переменную, в дальнейшем ее можно использовать в программе с помощью блока **Переменная** (Variable). Блок **Переменная** (Variable) позволяет считывать данные из памяти EV3 или записывать (сохранять) данные в память, как показано на рис. 16.4.

Для настройки блока **Переменная** (Variable) сначала перейдите в меню выбора режима и укажите, следует ли считывать данные переменной (**Считывание** (Read), значок в виде книги) или записывать данные в переменную (**Записать** (Write), значок в виде карандаша). Затем выберите тип переменной, которую необходимо считать или записать (**Числовое значение** (Numeric), **Логическое значение** (Logic) или **Текст** (Text)). Наконец, в списке **Название переменной** (Name) выберите переменную, которую следует использовать.

У блока **Переменная** (Variable) есть параметр **Значение** (Value). В режиме **Записать** (Write) вы можете присвоить этому параметру значение, которое нужно сохранить. Если вместо ввода значения подключить шину данных, в параметре будет сохранено переданное через нее значение. Если в переменной уже сохранено какое-либо значение, оно будет стерто и вместо него будет записано новое.

В режиме **Считывание** (Read) блок **Переменная** (Variable) извлекает информацию из памяти модуля EV3 и присваивает ее в качестве значения параметру **Значение** (Value) так, что эти данные можно передать другим блокам через шину данных. При считывании значение не меняется, поэтому, если считать его повторно с помощью другого блока **Переменная** (Variable), оно останется неизменным.

Определение переменных с помощью блока Переменная

Другой способ задать переменную — выбрать команду **Add Variable** (добавить переменную) в раскрывающемся списке **Название переменной** (Name) блока **Переменная** (Variable), как показано на рис. 16.5. При этом создается новая переменная того же типа, что и выбранный режим блока. Например, когда блок находится в режиме **Считывание** ▸ **Числовое значение** (Read ▸ Numeric), новая переменная будет числовой. Для создания таким способом логической переменной сначала следует изменить режим на **Считывание** ▸ **Логическое значение** (Read ▸ Logic) или **Записать** ▸ **Логическое значение** (Write ▸ Logic).

*Рис. 16.4. Хранение и считывание значений с помощью блока **Переменная***

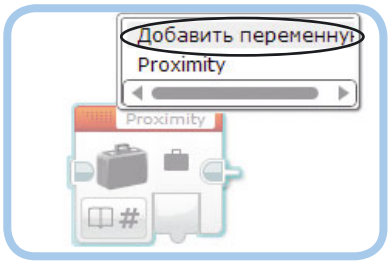


*Рис. 16.5. Для определения переменной непосредственно в блоке **Переменная** выберите команду **Добавить переменную**. Введите имя переменной в появившемся диалоговом окне и нажмите кнопку **OK***

Создание программ с переменными

Теперь, когда вы познакомились с принципами определения и использования переменных, можно приступать к разработке программы VariableDemo, показанной на рис. 16.6. Эта программа сохраняет значение приближения инфракрасного датчика в переменную с именем **Proximity**. Через 5 секунд она извлекает значение из переменной и использует его для управления скоростью мотора B, а это значит, что скорость мотора зависит от показаний

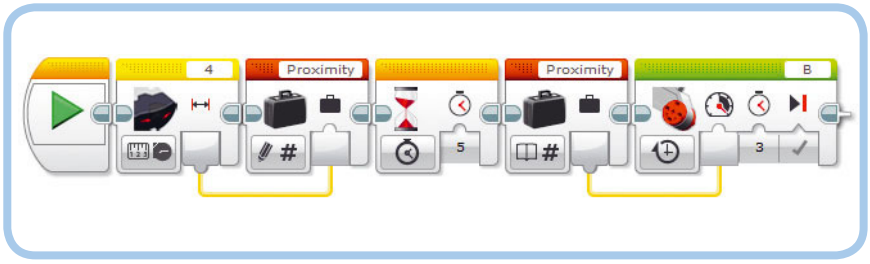


Рис. 16.6. Программа VariableDemo

ПРАКТИКУМ № 106: СТАРОЕ ИЛИ НОВОЕ!

Сложность: **Время:**

Можете ли вы разработать программу, которая многократно сравнивала бы новые показания датчика с показаниями, сохраненными в переменной при запуске программы? Если новые показания окажутся меньше, робот должен сказать «Да», а если больше — «Нет»*. Незаконченная версия программы показана на рис. 16.7.

СОВЕТ **С чего вы начнете при разработке программы с переменными? Какой блок Переменная (Variable) должен считывать или записывать значение? Как подключить шины данных?**

ПРАКТИКУМ № 107: ПРЕДЫДУЩЕЕ ИЛИ НОВОЕ!

Сложность: **Время:**

Программа, описанная в практикуме № 106, сравнивает новые показания датчика со значением, измеренным при запуске программы. Можете ли вы разработать программу, которая сравнивала бы каждое новое значение с предыдущим? Если новое значение окажется меньше, робот должен сказать «Да», а если больше — «Нет»**. Следовательно, при приближении объекта к инфракрасному датчику спереди робот будет говорить «Да»***.

СОВЕТ **Напишите программу, похожую на показанную на рис. 16.7, с одной переменной Proximity. Каждый раз в цикле роботу нужно будет сравнивать значение переменной Proximity с новыми показаниями датчика. Последним блоком цикла должен быть блок Переменная (Variable), запрограммированный сохранять новые показания датчика в качестве значения переменной Previous, тогда при следующем цикле в переменной Previous будет храниться значение, полученное при предыдущем повторе цикла.**

* Пункты **Yes** и **No** в каталоге звуков Lego.
** Пункты **Yes** и **No** в каталоге звуков Lego.
*** Пункт **Yes** в каталоге звуков Lego.

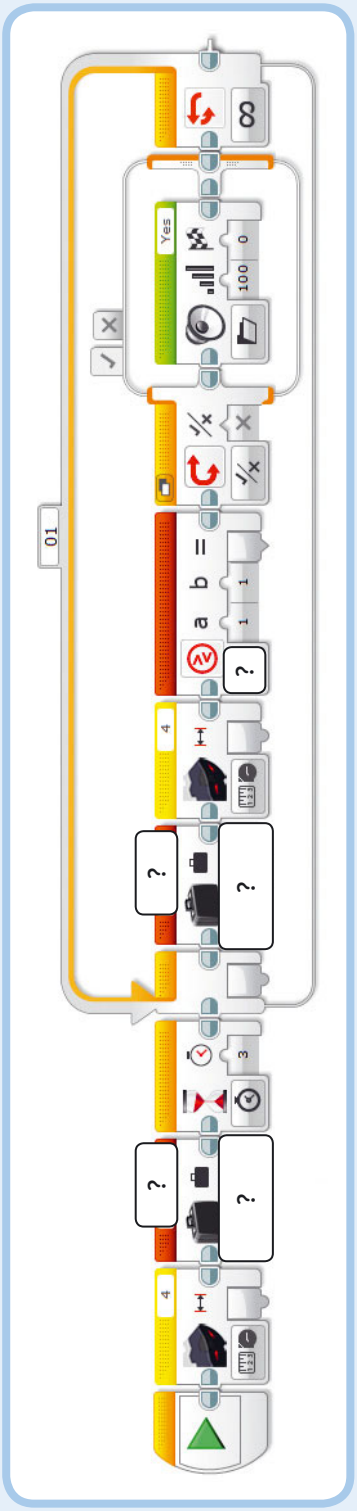


Рис. 16.7. Незаконченная версия программы для практикума № 106

инфракрасного датчика пятисекундной давности. Перед написанием программы определите переменную **Proximity**, если вы еще этого не сделали, как показано на рис. 16.3.

Программа VariableDemo иллюстрирует идею использования переменных, но это очень простая программа. Завершив ее, попрактикуйтесь с переменными в практикумах № 106 и № 107.

ПРИМЕЧАНИЕ **Не путайте блоки Константа (Constant) и Переменная (Variable). У обоих есть значок чемодана, но в блоке Константа (Constant) еще есть значок замка, напоминающий, что константу нельзя изменить во время работы программы.**

Изменение значений переменных

Иногда возникает необходимость изменить значение переменной, к примеру, чтобы с ее помощью отследить наибольшее значение и общее число нажатий кнопки датчика касания. Часто вам потребуется увеличивать значение переменной на 1, что называется *приращением*. Программа TouchCount, которую вы сейчас напишете, покажет, как с помощью переменной определить, сколько раз щелкали по кнопке датчика касания (нажимали кнопку и сразу же отпускали).

Начнем с определения новой числовой переменной с именем **Count**, в которой будет храниться число нажатий кнопки датчика. Программа будет ждать, пока кнопку датчика касания не нажмут, после чего значение переменной **Count** будет увеличиваться на 1. Чтобы программа продолжала считать, нам понадобится блок **Цикл** (Loop).

Но как увеличить значение переменной на 1? Как показано на рис. 16.8, для считывания значения переменной **Count** используется блок **Переменная** (Variable). Затем это значение передается в блок **Математика** (Math), где к нему будет добавлена единица. Полученный результат отправляется в другой блок **Переменная** (Variable), созданный, чтобы записывать (хранить) новое значение переменной **Count**, которое теперь увеличилось на 1. Блок **Цикл** (Loop) повторяет этот процесс в течение 5 секунд, после чего итоговое значение переменной **Count** отображается на экране модуля EV3. Цикл должен быть выполнен по крайней мере один раз, а значит, окончательное значение всегда будет 1 или больше.

С помощью этого метода можно изменять значение переменной как вам заблагорассудится. Данный пример показывает, как добавить к значению 1, но, применив этот же метод, можно и вычитать из значения.

Присвоение начального значения переменным

При программировании с использованием переменных важно присвоить им начальное значение. Вы делали это в программе TouchCount, установив значение переменной **Count** равным нулю в начале программы. Присвоение переменным начального значения повышает надежность программы, гарантируя, что каждый раз при запуске она будет работать одинаково, поскольку начинается с одного и того же момента.

Однако начальное значение необязательно должно равняться нулю. Например, если задать начальное значение

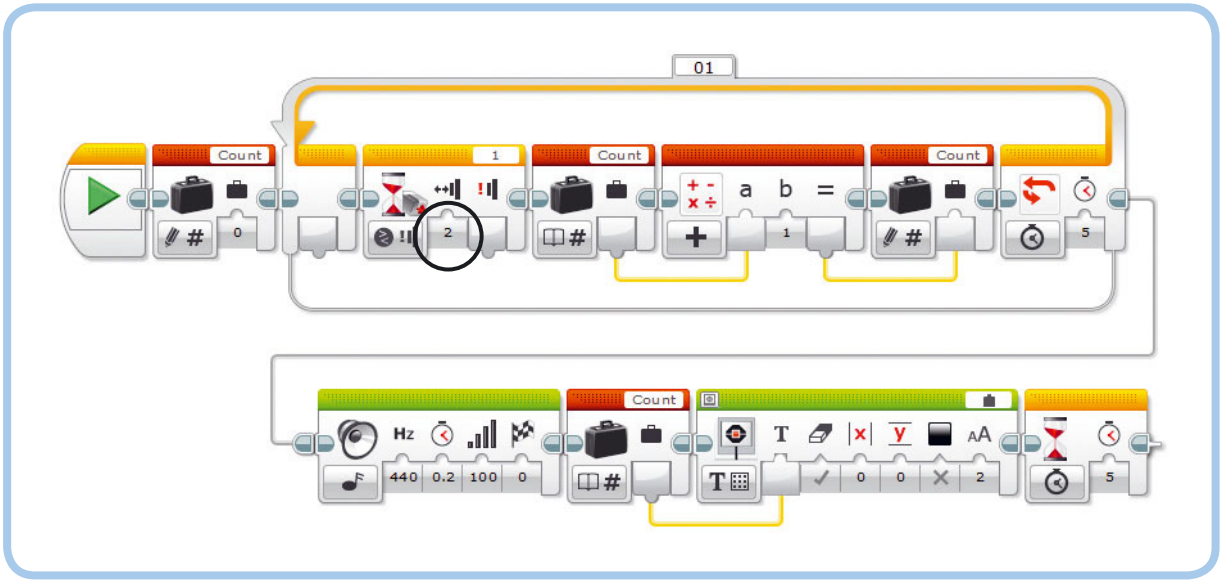


*Рис. 16.8. Программа TouchCount считывает количество нажатий на кнопку датчика касания в течение 5 секунд, а затем отображает итоговое число на экране. Обратите внимание, что для большей наглядности я подключил блоки **Звук** и **Цикл** друг к другу с помощью соединителя, но вы можете просто поместить блок **Звук** сразу после блока **Цикл***

переменной **Count** равным 5, программа будет начинать отсчет с 5.

Если вы не зададите в своей программе начальное значение числовой переменной, программное обеспечение EV3 будет использовать в качестве начального значения 0.

Тем не менее рекомендуется всегда задавать начальные значения в программах, даже если вы хотите, чтобы значения переменных начинались с 0.

Вычисление среднего значения

В следующем примере используем переменную для вычисления *среднего значения* 50 показаний датчика. Чтобы вычислить среднее показание (среднее значение), разделим сумму всех показаний на их количество (в данном случае 50). Программа Average, показанная на рис. 16.9, вычисляет эту сумму путем многократного добавления показаний датчика к переменной с именем **Sum**.

Начальное значение переменной равно нулю, а блоки в цикле каждый раз добавляют в переменную **Sum** последние показания датчика. После 50 повторов сумма делится на 50, чтобы вычислить среднее значение показаний датчика, которое, в свою очередь, отображается на экране.

Вычисление среднего значения может быть полезно для получения более точных показаний датчика. Например, у вас есть робот, огибающий препятствия, который поворачивает, если значение приближения становится меньше 50. В некоторых случаях датчик может показать, что препятствие находится ближе (на значении 40, скажем), хотя на самом деле там ничего нет. Обычно такие данные заставили бы робота повернуть, но если взять среднее значение трех показаний, то получится (100 + 100 + 40) / 3 = 80, и это ложное показание не заставит робота изменить курс.

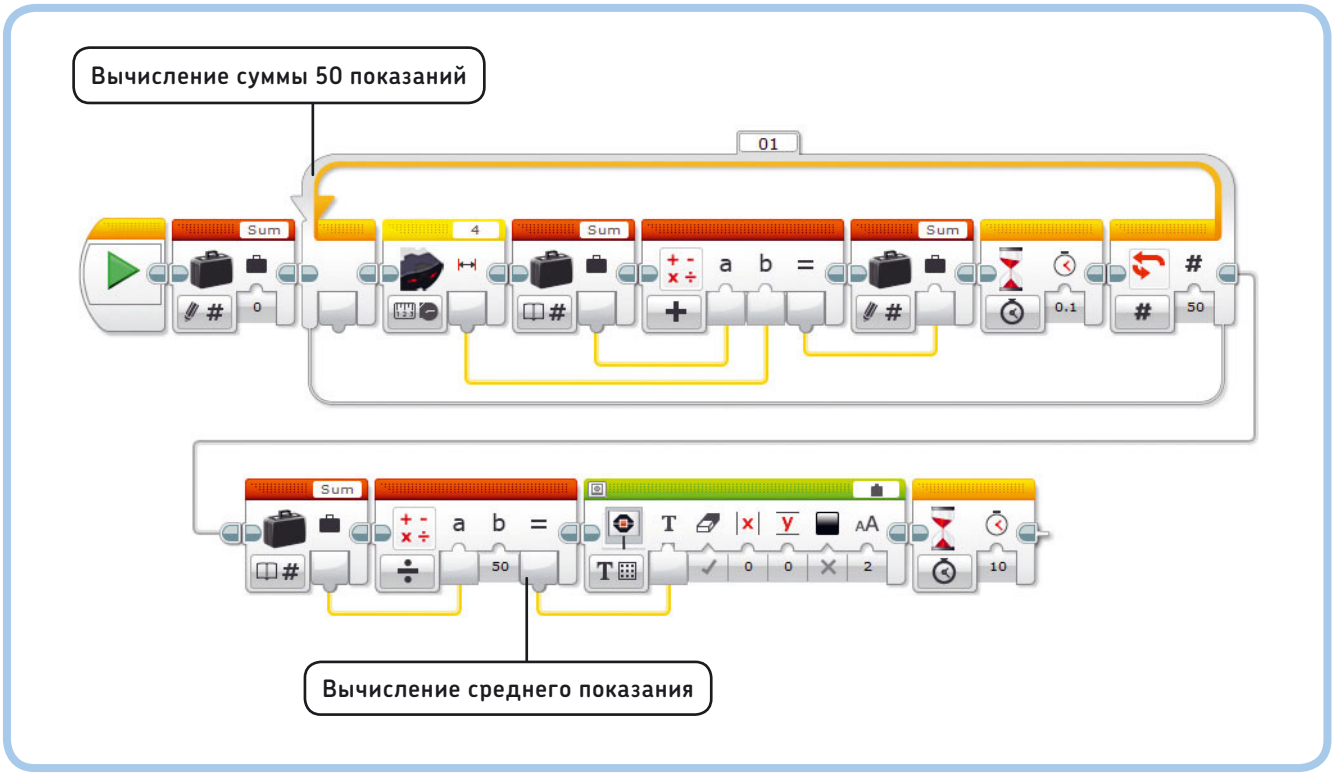


*Рис. 16.9. Программа Average. Блок **Ожидание** в блоке **Цикл** отвечает за паузу в 0,1 секунды между снятием показаний, чтобы программа могла вычислить среднее значение датчика за период 50×0,1 = 5 секунд*

Дальнейшее изучение

Поздравляю! Вы изучили всю теорию по программированию в этой книге и готовы перейти к следующей главе, в которой напишете большую программу, превращающую SK3TCHBOT в устройство наподобие детской игры «Волшебный экран». Но прежде изучите практикумы. Они могут оказаться сложнее, чем те, которые вы выполняли ранее, но помните, что в каждом случае существует несколько решений. Попробуйте их выполнить, а решения многих заданий можно найти по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip**.

ПРАКТИКУМ № 108: ПРЯМОЙ И ОБРАТНЫЙ СЧЕТ!

Сложность: **Время:**

Можно ли на основе программы TouchCount создать другую, которая будет использовать переменную для отслеживания количества нажатий кнопок «Влево» и «Вправо» на корпусе модуля EV3? При нажатии кнопки «Вправо» значение переменной должно увеличиваться на 1, а при нажатии кнопки «Влево» — уменьшаться на 1. Выведите значение переменной на экран EV3.

СОВЕТ **Используйте блок Ожидание (Wait), отслеживающий нажатия кнопок «Влево» и «Вправо». Затем используйте блок Переключатель (Switch), чтобы определить, какая кнопка была нажата. Внутрь блока Переключатель (Switch) добавьте блоки, необходимые для изменения значения переменной, и блоки, ожидающие, когда кнопку EV3 отпустят.**

ПРАКТИКУМ № 109: ОГРАНИЧЕННОЕ СРЕДНЕЕ!

Сложность: **Время:**

Можно ли расширить программу Average, показанную на рис. 16.9, таким образом, чтобы она вычисляла среднее значение всех замеров приближения с момента запуска программы до тех пор, пока вы не нажмете кнопку датчика касания?

СОВЕТ **Настройте цикл так, чтобы он повторялся, пока не будет нажата кнопка датчика касания. Задайте переменную с именем Measurements, чтобы отслеживать количество измерений датчика, добавленных к переменной Sum. Когда цикл завершится, вычислите среднее значение, поделив сумму на количество измерений.**

ПРАКТИКУМ № 110: ПОДТВЕРЖДЕНИЕ СЛУЧАЙНОСТИ!

Сложность: **Время:**

В этом практикуме вам нужно проверить, насколько хорошо работает блок **Случайное значение** (Random) в режиме **Логическое значение** (Logic). Для этого робот должен сгенерировать 10 000 случайных логических значений и определить, сколько из них являются истинными, а сколько — ложными. Поэкспериментируйте со значением параметра **Вероятность значения «истина»** (Probability of True) и посмотрите, удастся ли вам добиться ожидаемого результата.

СОВЕТ **Создайте две переменные типа Числовое значение (Numeric) с именами TrueCount и FalseCount. Создайте случайное логическое значение с помощью блока Случайное значение (Random). Если оно окажется истинным, то увеличьте значение переменной TrueCount; если ложным — переменной FalseCount. Выведите на экран значения обеих переменных после 10 000 повторений. Какие значения вы ожидаете увидеть?**

ПРАКТИКУМ № 111: МАКСИМАЛЬНОЕ СБЛИЖЕНИЕ!

Сложность: **Время:**

Можно ли создать программу, которая будет определять наименьшее зафиксированное значение приближения из 50 измерений? Задайте в программе паузу на 0,1 секунды между измерениями и выведите на экран наименьшее значение.

СОВЕТ **Создайте переменную с именем Lowest, в которой будет храниться наименьшее зарегистрированное показание. Регулярно сравнивайте сохраненное измерение с новыми показаниями датчика. Если новое показание окажется меньше, сохраните его значение в переменной Lowest. Какое значение нужно указать в качестве начального для переменной Lowest в начале программы?**

СДЕЛАЙ САМ № 27: СЧЕТЧИК КЛИЕНТОВ!

Сборка: **Программирование:**

Можно ли создать робота, который будет подсчитывать количество находящихся в комнате людей? Спроектируйте устройство, которое будет открывать дверь, когда гости нажимают кнопку датчика касания, чтобы войти в комнату, и попросите их поднести руку к инфракрасному датчику, чтобы открыть дверь, когда они захотят выйти. Затем измените программу, которую вы написали на практикуме № 108, чтобы подсчитать число людей в комнате: при нажатии кнопки датчика касания количество на счетчике должно увеличиваться, а при поднесении руки к инфракрасному датчику — уменьшаться.

СОВЕТ **Вместо того чтобы создавать робота, который будет поворачивать дверную ручку, гораздо проще спроектировать перемещающегося робота, который будет открывать дверь, толкая ее вперед, а закрывать — потянув назад. Убедитесь, что дверь закрывается не до конца. Как вариант, с помощью скотча или шнура закрепите дверную ручку, чтобы язычок замка не выступал и дверь не закрывалась.**

17

Играем в игры на EV3

Теперь напишем программу, для создания которой потребуется применить многие из изученных ранее техник программирования. Выше я познакомил вас со всеми техниками программирования и блоками на коротких примерах, а теперь вы узнаете, как объединить их в более сложную программу.

В этой главе вы создадите программу, которая позволит вам запускать на модуле EV3 игру наподобие «Волшебного экрана». С помощью стрелок и больших моторов вы будете рисовать на экране модуля EV3, а датчики касания и цвета будут использоваться для дополнительного ввода данных (рис. 17.1).

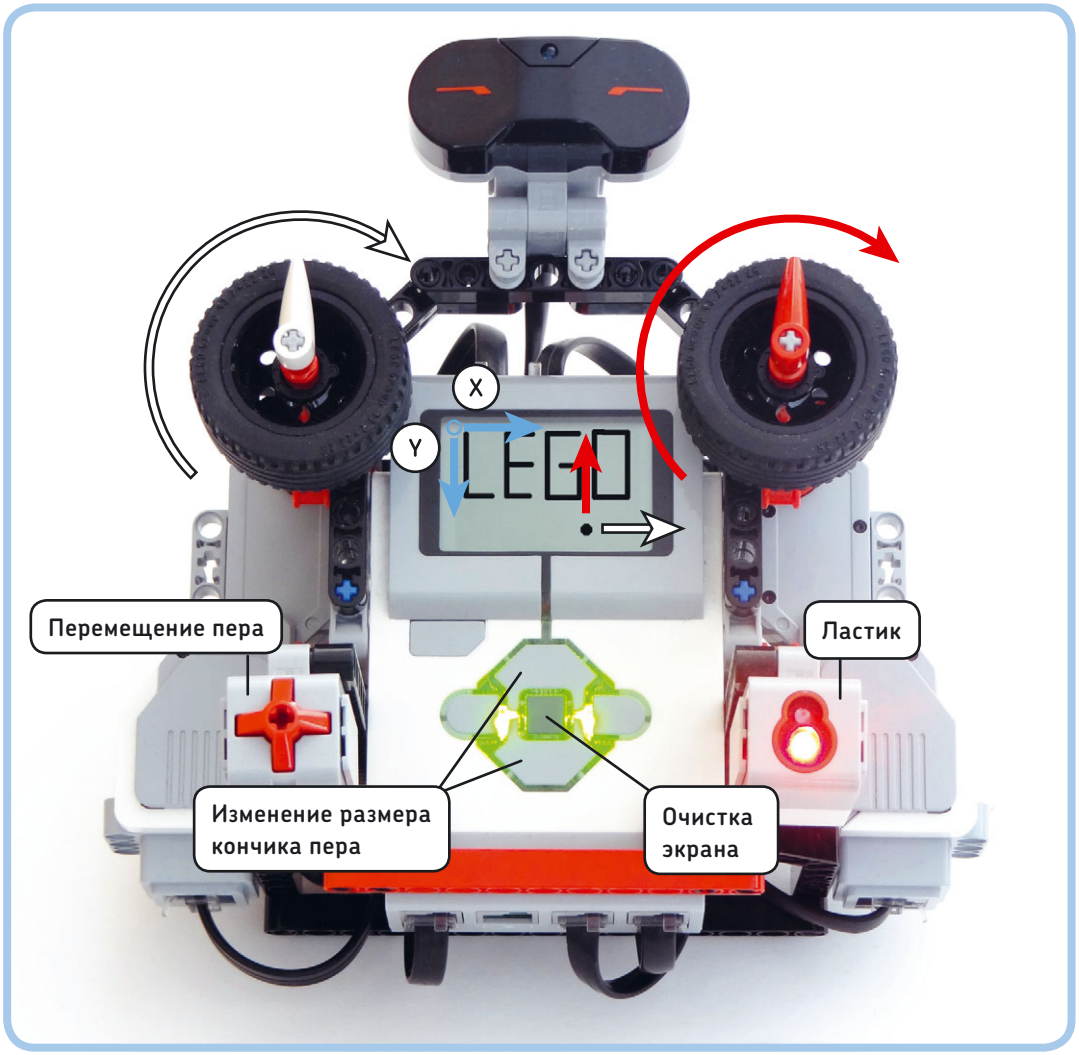


Рис. 17.1. Играем на SK3TCHBOT в «Волшебный экран»

Шаг первый: создание основной программы

В игре «Волшебный экран» можно рисовать на экране устройства линии, поворачивая стрелки, как будто вы рисуете пером на листе бумаги. Левая стрелка управляет положением пера по горизонтали (ось *x*), а правая — по вертикали (ось *y*). Такое поведение на удивление просто воссоздать в программе EV3, как показано на рис. 17.2.

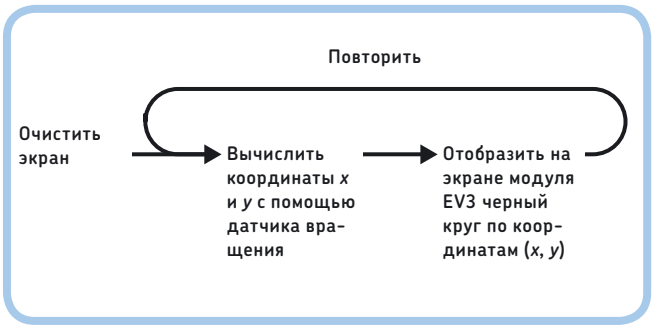


Рис. 17.2. Схема простой программы волшебного экрана

Программа постоянно отображает на экране модуля EV3 новую точку (кружок) в позиции, соответствующей координатам *x* и *y*. Программа определяет эти координаты с помощью датчиков вращения в больших моторах. Белая стрелка (мотор B) контролирует положение по оси *x*, а красная (мотор C) — по оси *y*. По мере того как программа добавляет точки на экране, можно создавать рисунки, «перемещая перо» с помощью стрелок на корпусе робота SK3TCHBOT.

Для начала создайте новый проект под названием SK3TCHBOT-Games, а в нем — программу Etch-A-Sketch. Чтобы отобразить точку, используйте блок **Экран** (Display) в режиме **Фигуры** ▸ **Круг** (Shapes ▸ Circle), но вам нужно будет создать два контейнера «Мой блок», позволяющих очищать экран и рассчитывать координаты, где должна отображаться точка.

Контейнер «Мой блок» № 1: Очистка экрана

У блока **Экран** (Display) нет функции очистки экрана без вывода на него нового изображения, но ее можно сымитировать, отобразив один белый пиксел и очистив остальную часть экрана. Создайте блок **Экран** (Display), как показано на рис. 17.3, и превратите его в контейнер «Мой блок» с именем **Clear**.

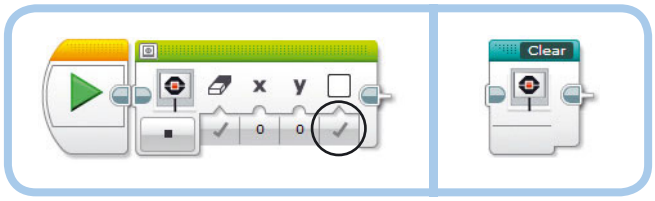


*Рис. 17.3. Контейнер **Clear** очищает экран EV3. Готовый контейнер «Мой блок» показан справа*

Контейнер «Мой блок» № 2: Определение координат

Теперь создайте контейнер «Мой блок» с именем **Coordinates**, как показано на рис. 17.4. У него есть два числовых вывода (**X** и **Y**), которые вы будете использовать для управления положением каждой новой точки. Давайте посмотрим, как этот блок вычисляет координату *x*.

Напомним, что датчик вращения сообщает, на сколько градусов повернулся мотор с момента запуска программы. В принципе, можно использовать значение датчика от мотора B, чтобы управлять непосредственно координатой *x*. Но тогда наше перо будет двигаться слишком быстро: поворот белой стрелки всего на 180 градусов заставит перо переместиться через весь экран. (Ширина экрана по горизонтали составляет 178 пикселов.)

Для уточнения движения разделите количество градусов на 3, чтобы при повороте белой стрелки на 180 градусов линия появлялась только поперек одной трети экрана. Добавьте к получившемуся частному число 89 (половина от 178), чтобы при значении, равном нулю градусов, при запуске программы точка появлялась в центре экрана.

Процедура определения координат *y* аналогична, за исключением того, что вы будете использовать мотор C и добавите к частному число 64 (по вертикали экран содержит 128 пикселов). Так как мотор C перевернут, то, поворачивая его в направлении, указанном красной стрелкой на рис 17.1, вы получите отрицательное значение градусов. Следовательно, перо будет двигаться вверх (вертикальная красная стрелка), а это противоположно положительному направлению координаты *y* (вертикальная синяя стрелка).

Чтобы создать контейнер «Мой блок», поместите в рабочую область два блока **Вращение мотора** (Motor Rotation) и два блока **Математика** (Math) и превратите их в контейнер «Мой блок» с двумя числовыми выводами и именем **Coordinates**, как показано на рис. 17.4.

Завершение основной программы

Теперь, когда вы создали основные компоненты (см. схему на рис. 17.2), завершите программу, как показано на рис. 17.5. Кроме двух контейнеров «Мой блок» программа содержит

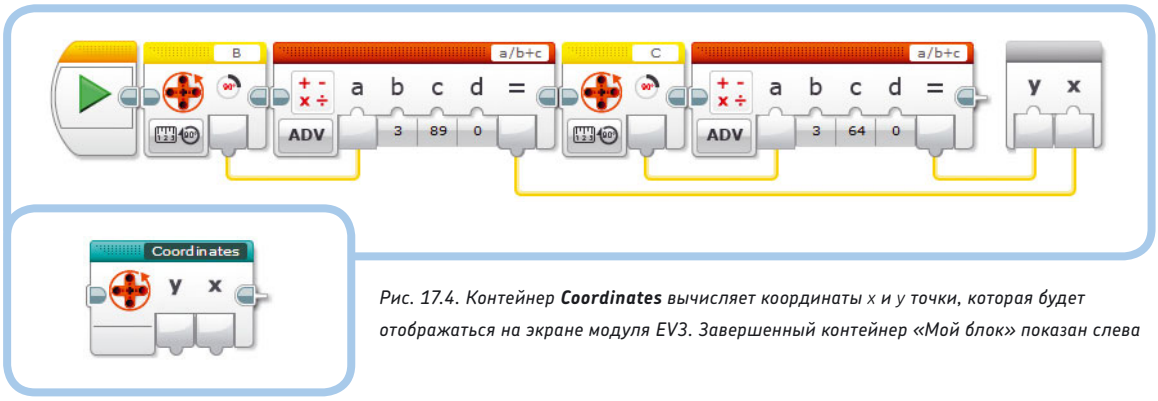


*Рис. 17.4. Контейнер **Coordinates** вычисляет координаты x и y точки, которая будет отображаться на экране модуля EV3. Завершенный контейнер «Мой блок» показан слева*

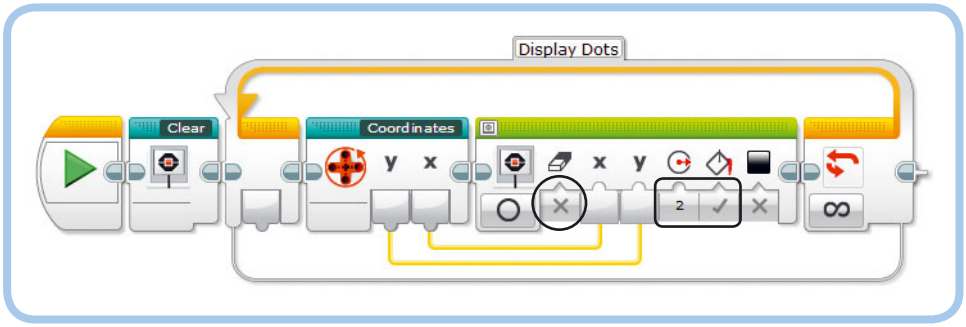


*Рис. 17.5. Основная программа Etch-A-Sketch. Обратите внимание, что блок **Экран** запрограммирован не для очистки экрана; он гарантирует, что ранее напечатанные точки останутся видимыми, когда на экран добавится еще одна точка*

блок **Экран** (Display), запрограммированный на отображение точки (кружочка) по заданным координатам, и блок **Цикл** (Loop) для повтора программы. Загрузите программу в робота и протестируйте ее перед тем, как развивать дальше в следующем разделе.

ПРИМЕЧАНИЕ Если у вас возникли трудности при создании этой программы, загрузите готовую программу по ссылке eksmo.ru/files/Lego_Mindstorms_Primers.zip и сравните ее с вашей.

Шаг второй: добавление элементов управления пером

Основная программа «Волшебного экрана» интересна, но можно придать ей еще больше функциональных возможностей, немного доработав, как показано на рис. 17.6.

Используйте датчик касания, чтобы на время прекращать процесс рисования, датчик цвета, чтобы превратить перо в ластик, центральную кнопку модуля EV3 для очистки экрана, а кнопки «Вверх» и «Вниз» для изменения размера пера.

Чтобы упростить поиск потенциальных проблем, обязательно тестируйте программу каждый раз после добавления новых функций. Когда закончите, в новых практикумах вас ждут задания по дальнейшему дополнению программы.

Перемещение пера без рисования

В процессе рисования иногда требуется «оторвать перо от бумаги», чтобы перейти в другую позицию, не чертя за собой линию. Для этого укажите в программе, что рисовать можно, только если кнопка датчика не нажата (рис. 17.7). В противном случае ничего не произойдет. (На вкладке **Ложь** (False) блока **Переключатель** (Switch) нет блоков.)

Превращение пера в ластик

Теперь добавьте в программу ластик, который будет рисовать белые точки при активации датчика цвета, как показано на рис. 17.8. Рисуя белые точки, можно эффективно стирать фрагменты изображения, что очень удобно, если вы вдруг сделаете ошибку.

Рис. 17.6. Схема расширенной версии программы Etch-A-Sketch

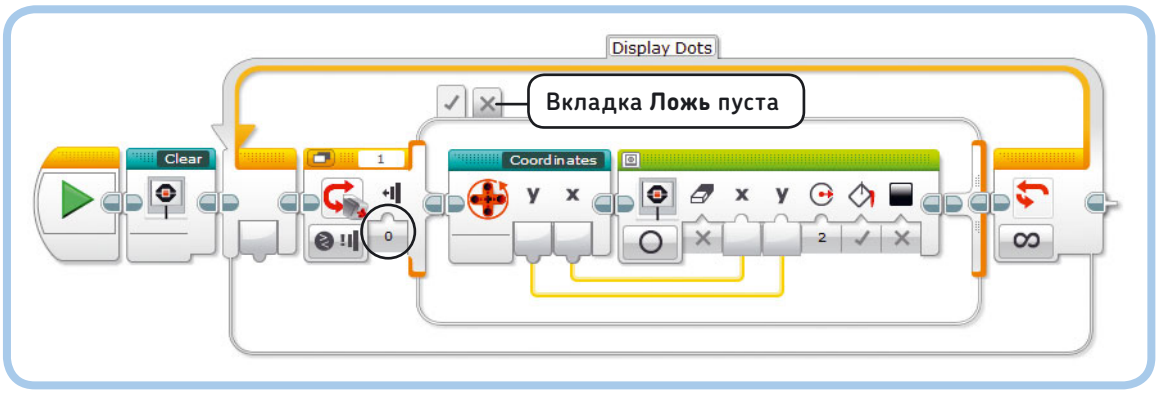


Рис. 17.7. Программа рисует новые точки только в том случае, если кнопка датчика касания не нажата. Чтобы переместить перо в другую позицию, не чертя при этом линию, нажмите и удерживайте кнопку датчика касания, поверните стрелки и отпустите кнопку датчика касания, чтобы продолжить рисовать

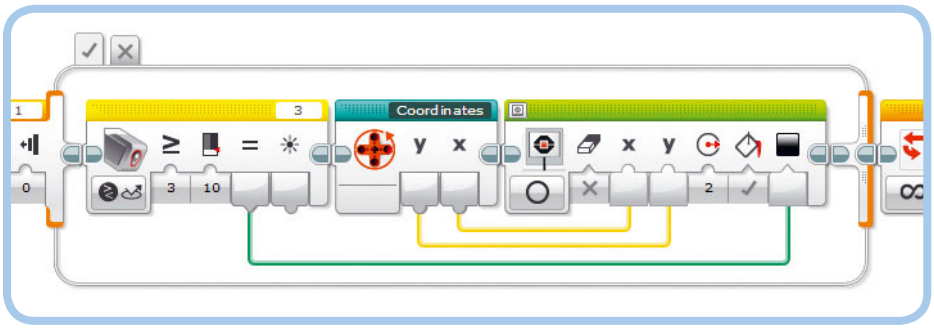


*Рис. 17.8. Добавьте блок **Датчик цвета**, настроенный, как показано на рисунке, и подключите логическую шину данных к блоку **Экран**. Теперь, когда вы поднесете палец к датчику цвета, перо будет работать как ластик*

Если поднести к датчику цвета палец, показатель яркости отраженного света вырастет до значений выше 10%, а параметр **Результат сравнения** (Compare Result) блока **Датчик цвета** (Color Sensor) примет значение **Истина** (True), в результате чего параметр **Цвет** (Color) блока **Экран** (Display) примет значение **Белый** (White).

Когда вы уберете палец, вывод передаст значение **Ложь** (False), и блок **Экран** (Display) продолжит добавлять на экран черные кружки.

Очистка экрана

Для очистки экрана нажатием центральной кнопки добавьте блоки **Переключатель** (Switch) и **Clear** (Очистить) (рис. 17.9). Если ни одна кнопка на корпусе модуля EV3 не нажата, то ничего не произойдет. Вариант **Ничего** (No Buttons) задан по умолчанию, и в нем не используется ни одного блока.

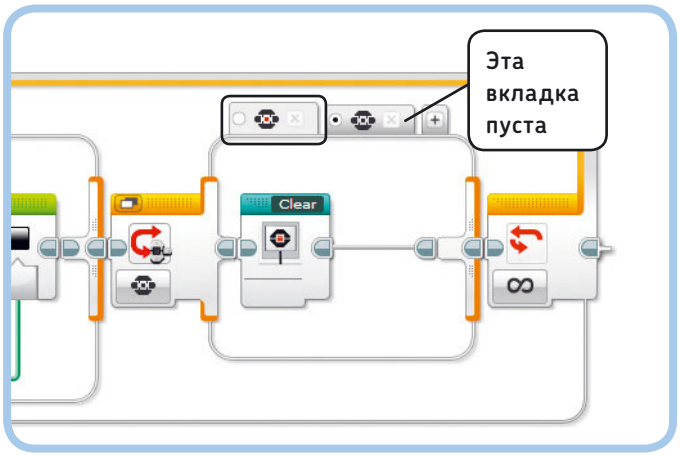


Рис. 17.9. Эти блоки очищают экран, когда вы нажимаете центральную кнопку

Установка размера пера

И наконец, дополним программу, чтобы можно было задавать размер пера с помощью кнопок «Вверх» и «Вниз» на корпусе модуля EV3. Для этого создадим переменную с именем **Size**, которая будет управлять значением параметра **Радиус** (Radius) блока **Экран** (Display). Затем добавим блоки, которые позволят менять значение переменной **Size** с помощью кнопок на корпусе модуля EV3.

Для начала определите переменную типа **Числовое значение** (Numeric) с именем **Size** и добавьте в текущую программу два блока **Переменная** (Variable), как показано на рис. 17.10. В первом блоке **Переменная** (Variable) задается начальное значение 1.

Второй блок передает значение размера на ввод **Радиус** (Radius) блока **Экран** (Display). Следовательно, радиус будет равен 1, пока вы не поменяете значение переменной.

Затем добавьте две ситуации (вкладки) к блоку **Переключатель** (Switch), который вы настроили ранее. Добавьте одну вкладку для кнопки «Вверх», а вторую — для кнопки «Вниз».

При нажатии кнопки «Вверх» переменная **Size** должна увеличиваться на 1, как показано на рис. 17.11. Согласно величине экрана, нужно ограничить значение размера числом 25. Другими словами, вы будете увеличивать переменную, только если ее текущее значение меньше 25, как это определено в блоке **Сравнение** (Compare). Если значение переменной **Size** достигнет 25, а вы попытаетесь его еще увеличить, блок **Сравнение** (Compare) выдаст значение **Ложь** (False), вы услышите звуковой сигнал, и значение не изменится.

Блок **Ожидание** (Wait) прекратит выполнение программы до тех пор, пока вы не отпустите кнопку «Вверх». Если не установить режим ожидания до тех пор, пока кнопка не будет отпущена, программа продолжит повторять действие, увеличивая значение переменной с каждым новым циклом.

Значение переменной должно уменьшаться (на 1) при нажатии кнопки «Вниз», но только если текущее значение переменной **Size** больше 1, чтобы в итоге у вас не получился круг с отрицательным или равным нулю значением радиуса (который не будет работать). Если значение переменной **Size** равно 1, а вы попытаетесь еще его уменьшить, блоки **Сравнение** (Compare) выведут значение **Ложь** (False), вы услышите звуковой сигнал, и значение не изменится (рис. 17.12).

Поздравляем! Вы завершили программу Etch-A-Sketch. После проверки работоспособности программы попробуйте еще дополнить ее, выполняя задания из практикумов № 112–114.

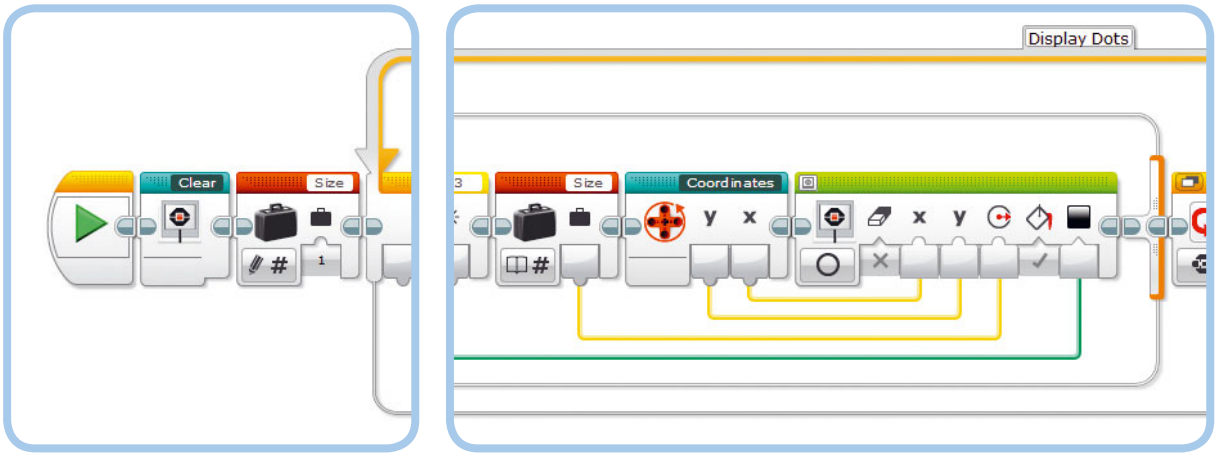


*Рис. 17.10. Добавьте в программу два блока **Переменная**, как показано на рисунке*

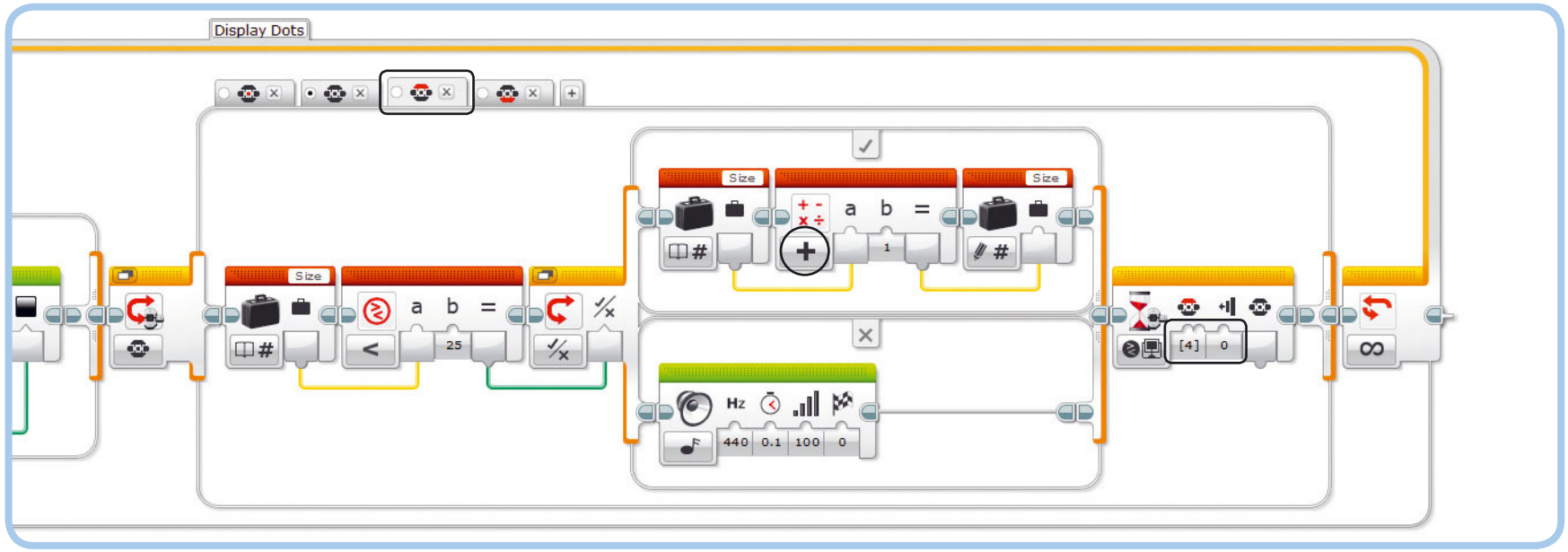


*Рис. 17.11. Если текущее значение переменной **Size** меньше 25, то при нажатии кнопки «Вверх» оно увеличится на 1. Обратите внимание, что вторая вкладка (**Ничего**) остается выбранной по умолчанию*

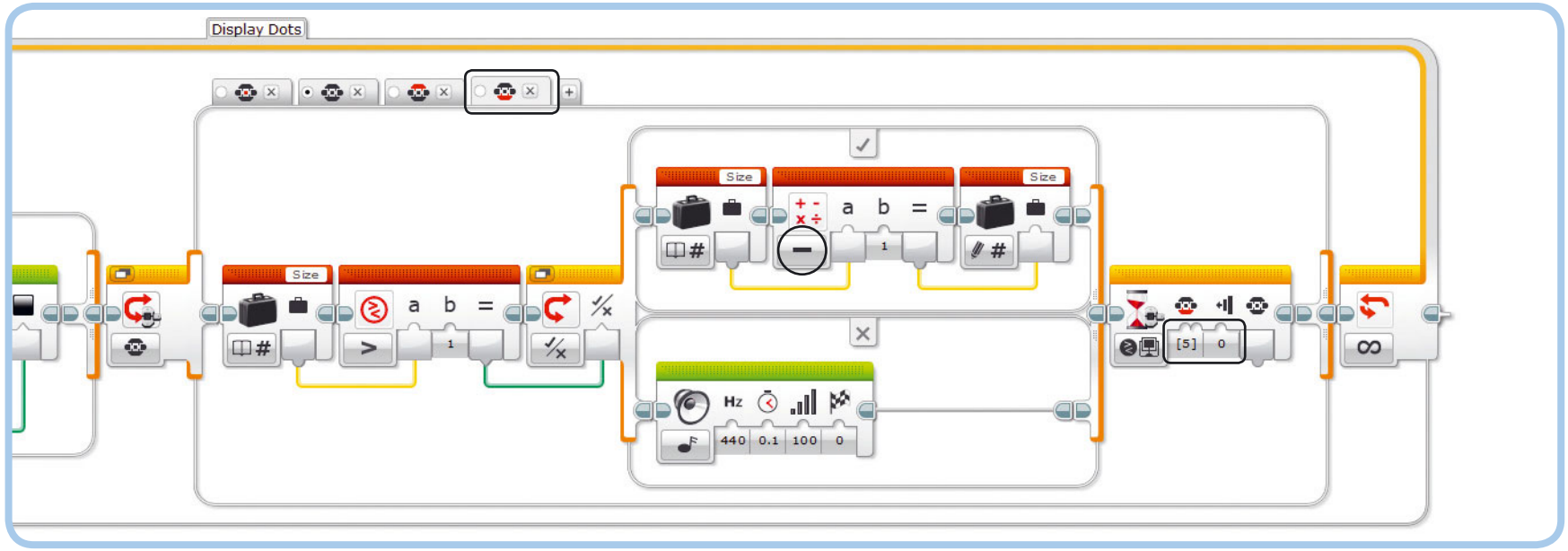


*Рис. 17.12. Если текущее значение переменной **Size** больше 1, то при нажатии кнопки «Вниз» оно уменьшится на 1*

ПРАКТИКУМ № 112: РОБОТ-ХУДОЖНИК!

Сложность: **Время:**

Нарисовать красивый рисунок с помощью программы Etch-A-Sketch непросто, но можно применить маленькую хитрость — заставить программу рисовать идеально прямые линии. Для этого нужно с помощью программных блоков управлять движением моторов B и C. Можно ли запрограммировать робота, чтобы он нарисовал на экране домик? Как нарисовать диагональные линии?

СОВЕТ **Управляйте движением моторов в программе, работающей параллельно с Etch-A-Sketch (см. раздел «Многозадачность» в главе 5).**

ПРАКТИКУМ № 113: РЕАКЦИЯ НА СИЛУ!

Сложность: **Время:**

Можно ли расширить контейнер **Coordinates** так, чтобы пользователь не мог вывести перо за пределы экрана? Перо должно автоматически возвращаться на экран, если пользователь повернет стрелки слишком сильно.

СОВЕТ **Укажите, что скорость мотора B в активном режиме должна составлять –5% при значениях координаты *x* больше 178 и 5%, если значение *x* оказывается меньше 0. Если значение координаты *x* находится в диапазоне от 0 до 178, запрограммируйте остановку мотора и присвоение значения Ложь (False) параметру Тормозить в конце (Brake at End). Такое же поведение задайте и в отношении координаты *y*.**

ПРАКТИКУМ № 114: ПЕРО-УКАЗКА!

Сложность: **Время:**

Диаметр самых маленьких точек, которые программа Etch-A-Sketch может отобразить на экране, 2 пиксела (радиус 1 пиксел). Можно ли дополнить программу так, чтобы в качестве самых маленьких точек на экране использовались отдельные пикселы? Это позволит прорисовать мельчайшие детали.

СОВЕТ **На экране можно нарисовать отдельную точку, если отобразить один пиксел по координатам *x* и *y*. Это можно сделать с помощью блока Экран (Display) в режиме Фигуры ▸ Точка (Shapes ▸ Point).**

Дальнейшее изучение

В этой главе вы разработали программу, в которой объединены многие изученные ранее техники программирования. Если вам понравилось, рекомендую изучить практикумы № 115 и 116. Далее вы узнаете, как сконструировать и запрограммировать еще двух классных роботов.

ПРАКТИКУМ № 115: АРКАДНАЯ ИГРА!

Сложность: **Время:**

На этом практикуме вы узнаете, как написать программу аркадной игры для SK3TCHBOT. Для начала загрузите программу по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip**, запустите ее и посмотрите, что произойдет. На экране модуля EV3 должна появиться цель (местоположение ее случайно) и игрок, которым управляет пользователь (рис. 17.13).

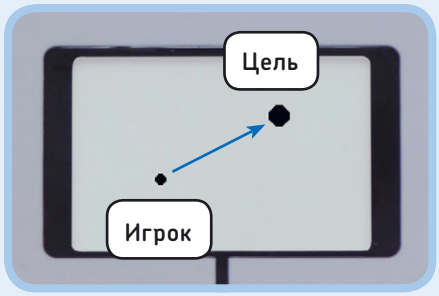


Рис. 17.13. Аркадная игра

Для попадания в цель нужно управлять игроком с помощью стрелок на корпусе робота SK3TCHBOT. Если вам удастся поразить цель в течение 4 секунд, вы заработаете 1 очко. Если вам потребуется более 4 секунд, вы потеряете 1 очко. После каждого попадания в другой позиции экрана появляется новая цель. После 10 ходов программа сообщит вам результат.

Изучите программу, чтобы понять, как она работает, а схема, приведенная на рис. 17.14, вам в этом поможет. Если вы любите трудности, попробуйте написать программу самостоятельно!

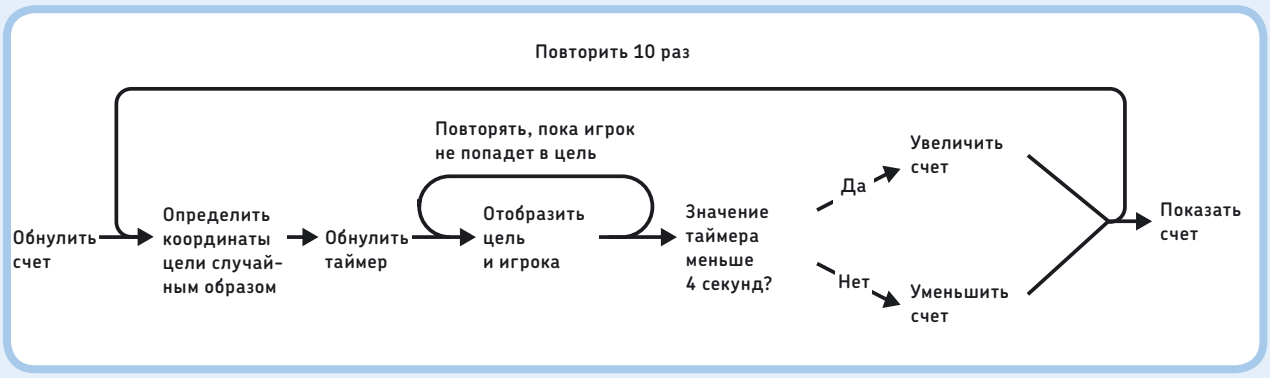


Рис. 17.14. Схема аркадной игры из практикума № 115

ПРАКТИКУМ № 116: ЗАРЯДКА ДЛЯ УМА!

Сложность: **Время:**

В этом практикуме вы освоите программу для тренировки ума, которую можно запустить на экране модуля EV3. Вы можете скачать ее по ссылке **eksmo.ru/files/Lego_Mindstorms_Primers.zip**. Программа отображает на экране модуля EV3 в случайном порядке примеры на сложение, вычитание, умножение или деление (например, 7 × 3).

Пользователь решает, верен ли отображаемый ответ, и нажимает кнопку «Влево» (неверно) или кнопку «Вправо» (верно), как показано на рис. 17.15. Правильный ответ высвечивается примерно в половине случаев.

Программа создана с использованием только тех программных блоков, которые обсуждались ранее в этой книге, но ее все равно трудно понять поначалу.

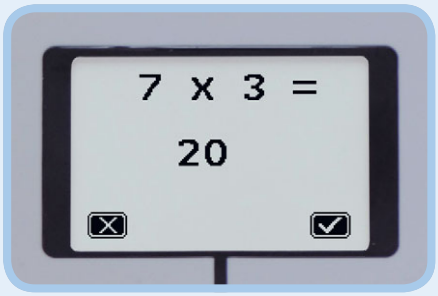


Рис. 17.15. Игра для тренировки ума. Нажав кнопку «Влево» в этом примере, вы заработаете 1 очко, так как данный ответ — неверный

Обратите внимание на комментарии, они помогут изучить программу.

Как только вы разберетесь, как она работает, дополните программу, чтобы отслеживать, сколько верных и неверных ответов вы даете в течение 30 секунд. Сможете улучшить результат, немного потренировавшись?

СДЕЛАЙ САМ № 28: ПЛОТТЕР!

Сборка: **Программирование:**

Можно ли сконструировать робота, способного делать чертежи на листе формата А4? С помощью одного мотора перо будет перемещаться по всей ширине листа для рисования горизонтальных линий (*x*), а с помощью другого мотора — перемещать бумагу, чтобы чертить вертикальные линии (*y*). Наконец, используйте блок **Средний мотор** (Medium Motor), чтобы поднимать и опускать перо. Это позволит перемещать перо и продолжать рисование в другом месте. Чтобы упростить управление устройством, создайте контейнеры «Мой блок» для перемещения пера с вводами, позволяющими указать координаты *x* и *y*.

СОВЕТ **Для вдохновения изучите механизм работы обычного струйного принтера. Принтер перемещает картридж влево и вправо по неподвижной направляющей (так печатаются горизонтальные линии) и передвигает бумагу вперед-назад с помощью колесиков (так печатаются вертикальные линии).**

ЧАСТЬ VI

Роботы-машины и гуманоиды

18

SNATCH3R: автономный робот-манипулятор

Из предыдущих глав вы узнали много нового о том, как программировать роботов EV3. Теперь можно приступить к созданию и программированию более сложных роботов. В этой главе вы построите и запрограммируете SNATCH3R, робота-манипулятора, способного находить и поднимать предметы. Он показан на рис. 18.1.

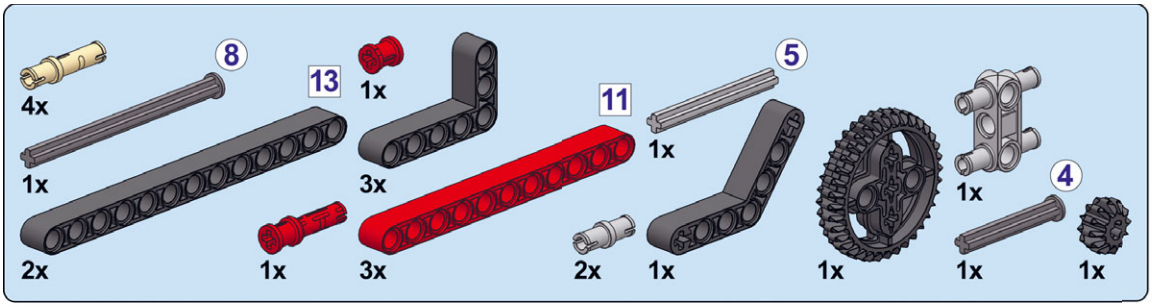
Сначала вы создадите программу, позволяющую управлять роботом удаленно, — так вы сможете протестировать его механические возможности. Затем вы запрограммируете робота таким образом, чтобы он находил и поднимал инфракрасный маяк. С помощью шин данных и переменных можно создать такой алгоритм, чтобы робот исследовал окружающую обстановку и был в состоянии обнаружить маяк на расстоянии вплоть до 2 метров, даже если тот находится позади робота.

Практикумы продемонстрируют вам, как работают эти программы, и вы сможете дополнить их новыми функциями. Например, перед вами будет стоять задача запрограммировать робота так, чтобы он, ориентируясь на датчик цвета, двигался по определенным линиям и собирал предметы вдоль маршрута.

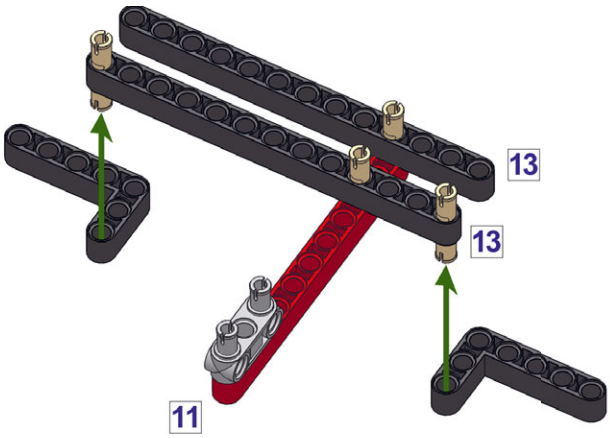
Рис. 18.1. Робот SNATCH3R может захватывать и поднимать легкие предметы, например пустую пластиковую бутылку

Захватное устройство

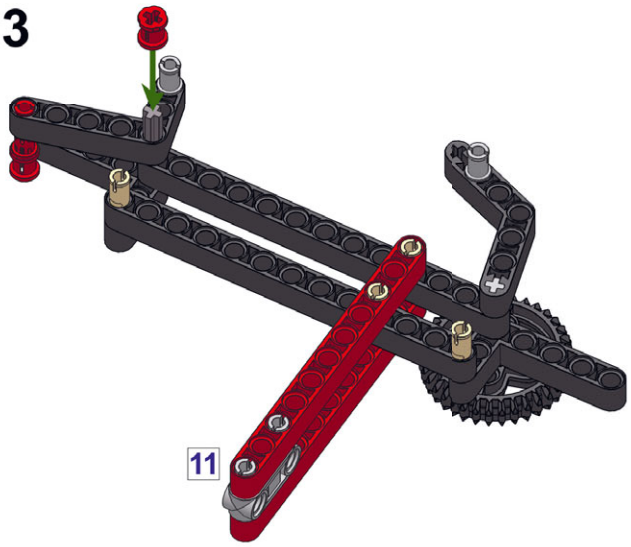
В роботе SNATCH3R вращением колес управляют два больших мотора, но самая потрясающая особенность этого робота — его многофункциональное захватное устройство. Как правило, для захвата и подъема предметов требуется два мотора: один для захвата, а второй, соответственно, для подъема. Роботу SNATCH3R для выполнения обеих задач достаточно одного среднего мотора, что обеспечивается особенностями используемых для его создания балок, осей и зубчатых колес LEGO. Чтобы лучше понять, как этот робот функционирует, вы с помощью инструкций на следующей странице построите упрощенную версию захватного устройства.



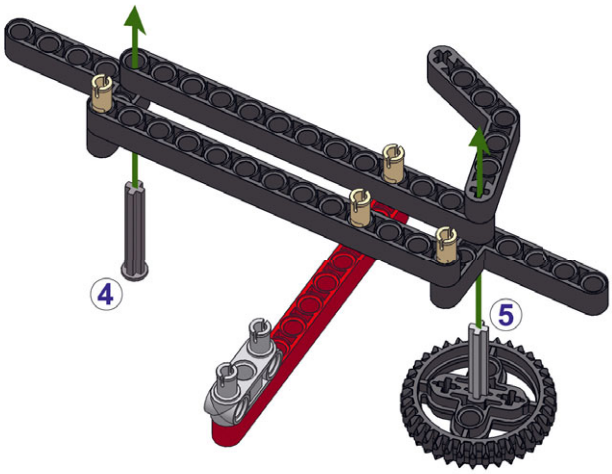
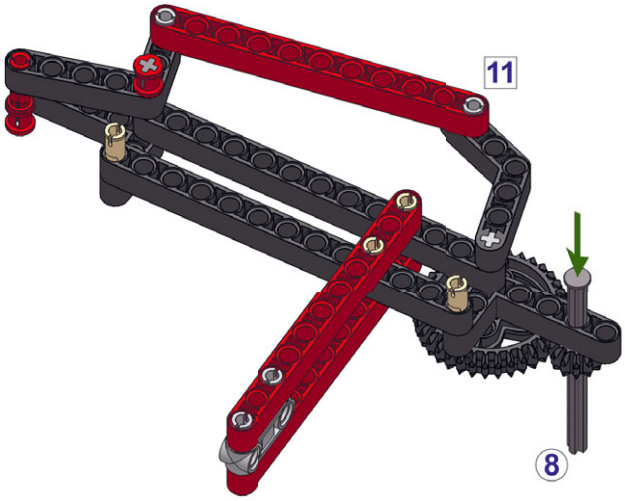
1



3



2



4