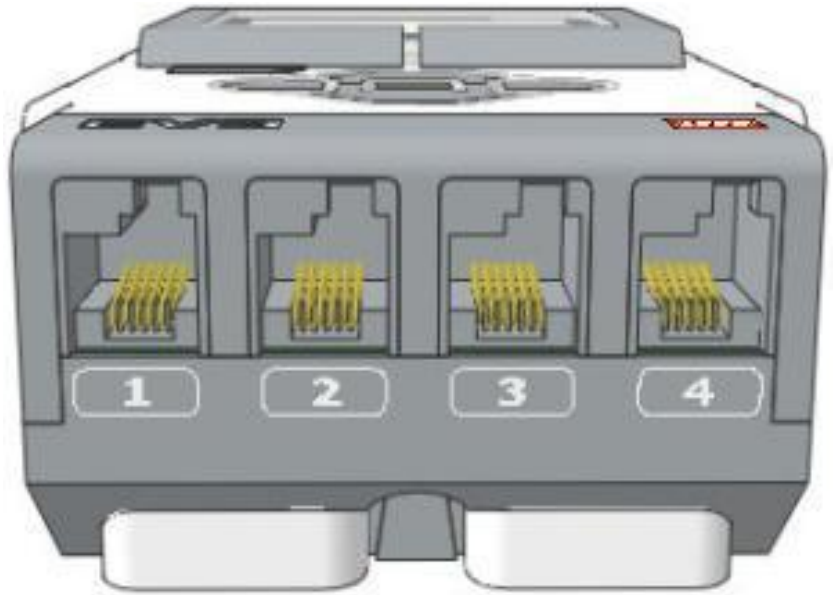
Урок №4 - Изучаем датчик касания

- В состав конструктора Lego mindstorms EV3 входят различные датчики. Главная задача датчиков - представлять информацию из внешней среды модулю EV3, а задача программиста - научиться получать и обрабатывать эту информацию, подавая необходимые команды моторам робота. На протяжении ряда уроков мы будем последовательно знакомиться со всеми датчиками, входящими и в домашний, и в образовательный наборы, научимся взаимодействовать с ними и решать наиболее распространенные задачи управления роботом.

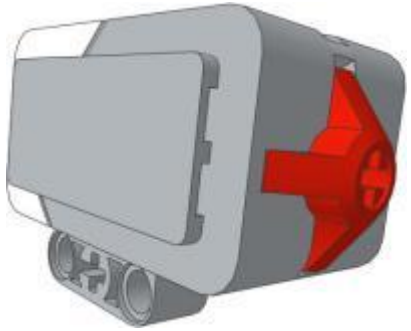
Ваш набор MINDSTORMS EV3 содержит три датчика, которые можно прикрепить к роботу (рис. 6.2), а также некоторые встроенные датчики. *Датчик касания* определяет, нажата ли красная кнопка на датчике или отпущена. *Датчик цвета* определяет цвет поверхности и яркость источника света, о чем вы подробнее узнаете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

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

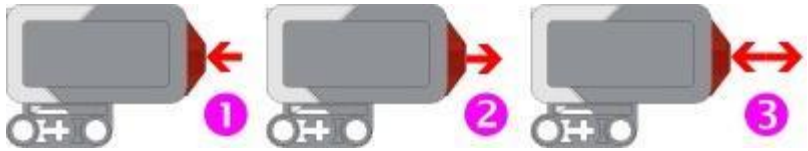
- **4.1. Изучаем первый датчик – датчик касания**
- Для подключения датчиков к модулю EV3 предназначены порты, обозначенные цифрами "**1**", "**2**", "**3**" и "**4**". Таким образом, к одному модулю EV3 одновременно можно подключить до четырех различных датчиков. Все порты абсолютно равнозначны и вы можете подключать датчики к любым портам, главное - будьте внимательны при указании номера порта для соответствующих датчиков в ваших программах.
-



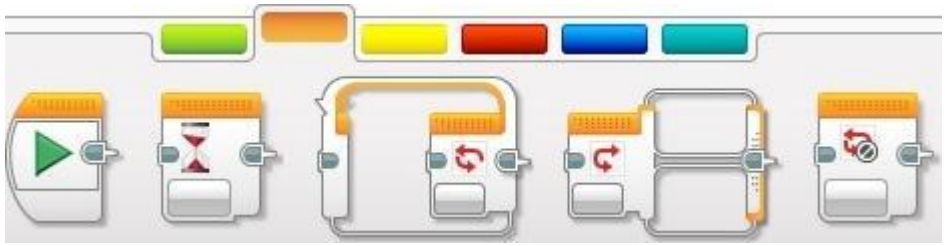
- Первым датчиком, который мы изучим, будет датчик касания **(Рис. 2)**.



- Этот датчик, по сути, представляет собой специальную кнопку, которая может находиться в двух состояниях: **"Нажатие" (Рис. 3 поз. 1)** или **"Освобождение" (Рис. 3 поз. 2)**. Также, последовательный переход в состояние **"Нажатие"**, а затем **"Освобождение"** называется: **"Щелчок" (Рис. 3 поз. 3)** и может обрабатываться программой. как самостоятельное событие.



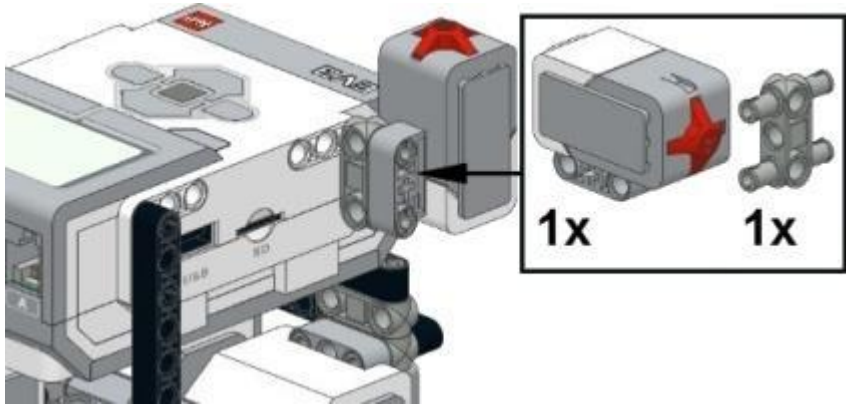
- **4.2. Оранжевая палитра - Управление операторами**
- Какие же инструменты представляет нам среда программирования для получения информации с датчиков и реагирования на эту информацию в программе? Давайте начнем знакомиться с программными блоками, расположенными в Оранжевой палитре, которая называется **"Управление операторами". (Рис. 4)**



- Программные блоки Оранжевой палитры, не смотря на свою малочисленность, очень важны! С помощью этих блоков мы можем обрабатывать массу событий и условий и сложно представить практическую программу, которая может обойтись без этих блоков.

- С самым первым блоком Оранжевой палитры мы уже с вами знакомы: он называется "**Начало**". Именно с него начинаются все программы для роботов.
- Второй программный блок называется "**Ожидание**". Этот блок заставляет программу ожидать выполнения какого-либо условия или наступления какого-либо события. Пока не выполнится условие, установленное в этом блоке, программа не перейдет к выполнению следующих программных блоков! Если перед тем, как начнется выполнение блока "**Ожидание**" были включены, какие-либо моторы, то они будут продолжать вращаться с установленной скоростью.
- Третий программный блок называется "**Цикл**". Этот блок многократно выполняет программные блоки, вложенные внутрь его, пока не будет выполнено условие завершения цикла, заданное в настройках блока.
- Следующий программный блок называется "**Переключатель**". Он служит для того, чтобы в зависимости от заданных условий - выполнить одну последовательность программных блоков, вложенных в один из своих контейнеров.
- Заключительный программный блок называется "**Прерывание цикла**". Его предназначение - досрочное прекращение выполнения заданного цикла.
- Программные блоки "**Ожидание**", "**Цикл**" и "**Переключатель**" имеют множество режимов и соответствующих настроек, знакомиться с которыми мы будем на практических примерах, последовательно и с наглядными пояснениями.

- **4.3. Оранжевая палитра, программный блок "Ожидание"**
- Перед тем, как приступить к решению практических задач, давайте закрепим датчик касания на нашем роботе, как показано на **Рис. 5**, и подключим его кабелем к порту **"1"** модуля EV3.



Датчики и блок Ожидание

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

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

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

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

Использование датчиков с блоком Ожидание

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

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

Обход препятствий с помощью датчика касания

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

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

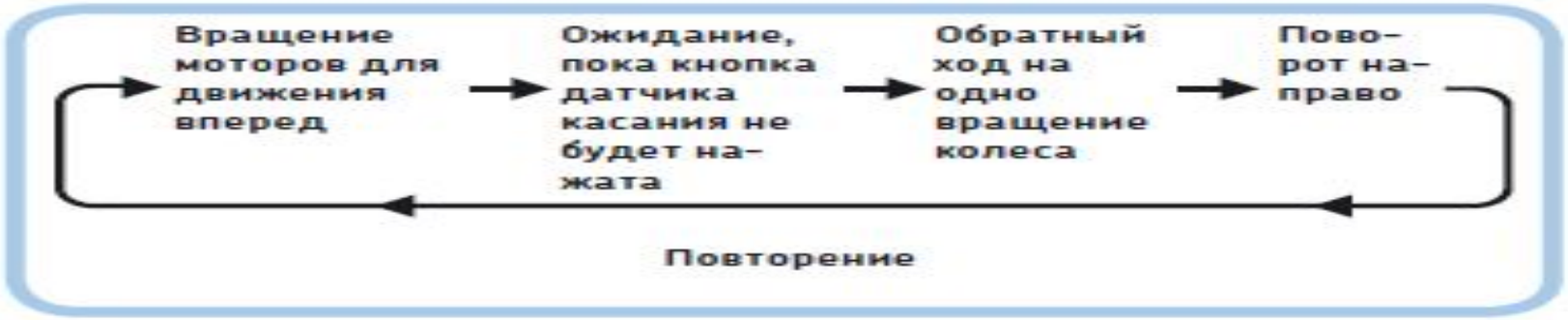


Рис. 6.7. Схема выполнения программы TouchAvoid. После поворота робота направо программа возвращается к началу и повторяется

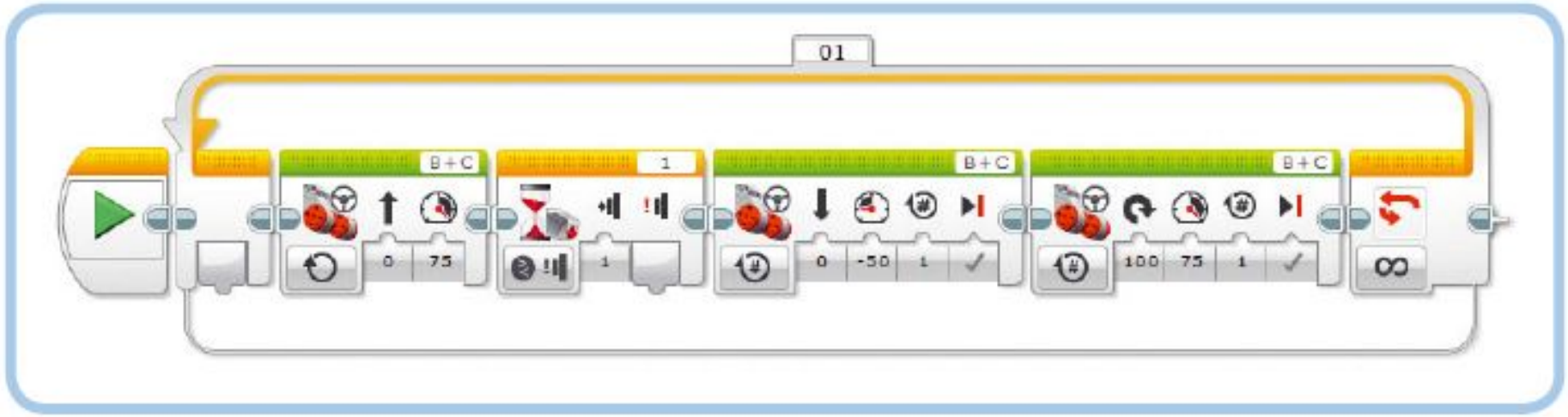


Рис. 6.8. Программа TouchAvoid. Блок ***Ожидание*** *настроен на режим* ***Датчик касания*** ▸ ***Сравнение*** ▸ ***Состояние***

Использование режима Изменить

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

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

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

Для большинства программ рекомендуется использовать режим **Сравнение** (Compare), так как с ним поведение

программа ожидает нажатия, робот продолжает двигаться вперед.

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

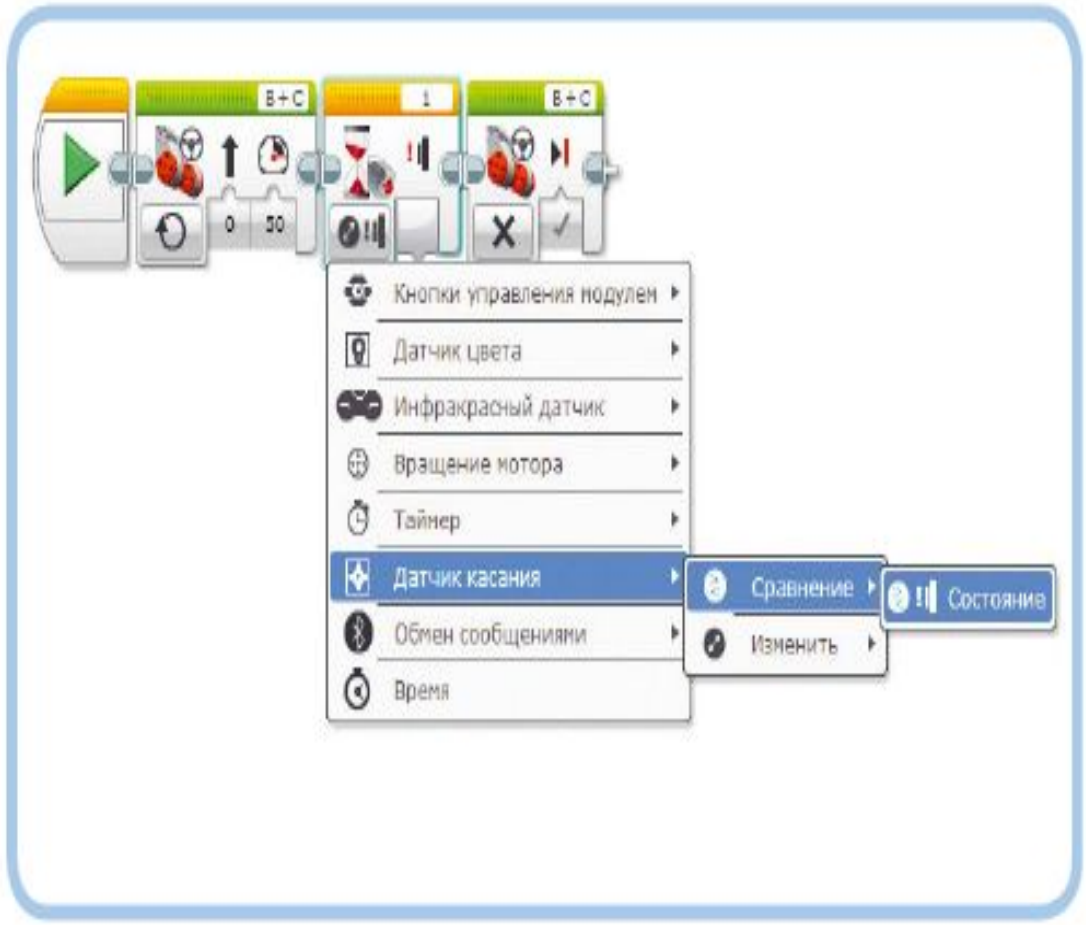


Рис. 6.9. Программа WaitForChange

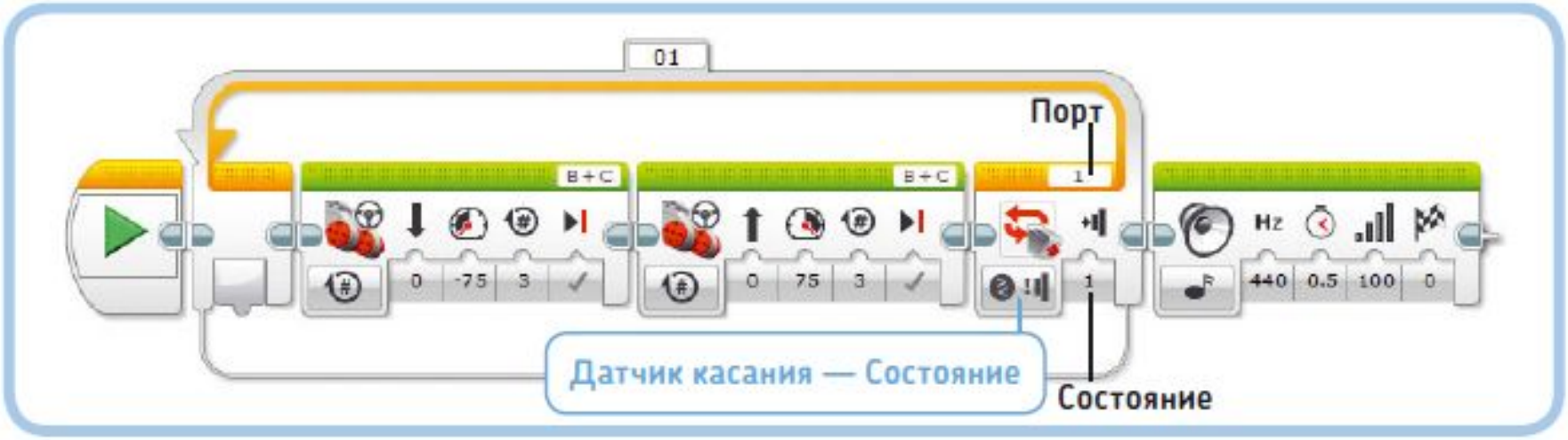


Рис. 6.10. Программа LoopUntilTouch. Чтобы настроить блок ***Цикл****, нажмите кнопку выбора режима и выберите пункт* ***Датчик касания*** ▸ ***Состояние***

Датчики и блок Цикл

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

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

Это обычный режим работы для блока **Цикл** (Loop), но иногда бывает нужно завершить цикл, даже если кнопка датчика не нажата точно в нужное время. Для достижения этой цели присвойте параметру **Состояние** (State) значение **Щелчок** (Bumped) (**2**) и запустите программу еще раз. В этой конфигурации цикл не проверяет, нажата ли кнопка датчика касания в *конце* цикла; скорее, он проверяет, наталкивается ли на что-то бампер (кнопка

датчика нажата и отпущена), в любое время в *течение* цикла. Если вы установите такие настройки, то блоки должны прекратить повторяться после того, как они завершают текущий запуск программы. Во время выполнения цикла EV3 непрерывно контролирует состояние датчика касания, поэтому вам не придется беспокоиться об этом.

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

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

Датчики и блок Переключатель

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

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

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

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

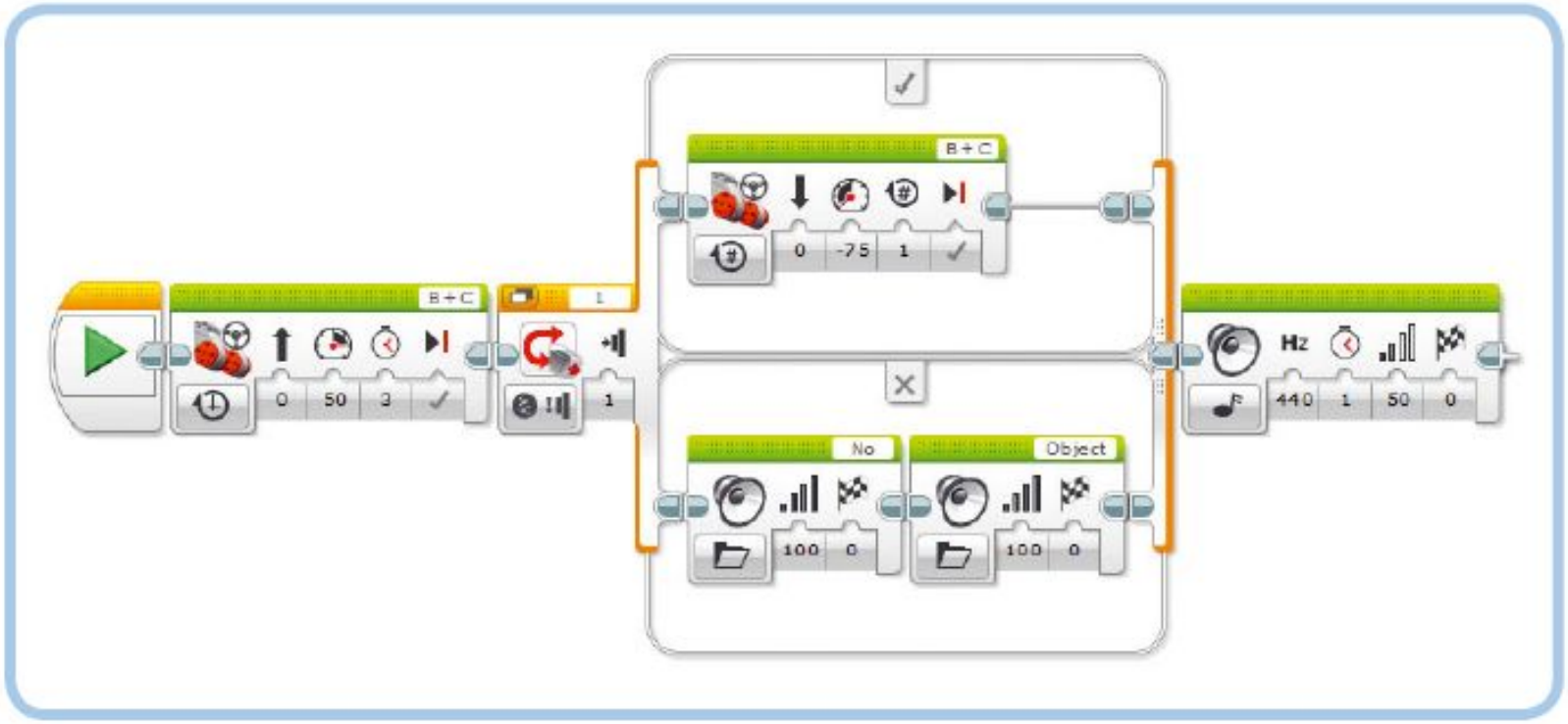


Рис. 6.13. Программа TouchSwitch позволяет роботу решать, что делать, основываясь на информации, полученной от датчика

Настройка блока Переключатель

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

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

Использование датчиков с блоком Переключатель

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

Попробуйте запустить эту программу несколько раз и определить, когда необходимо нажать кнопку датчика касания, чтобы робот двигался в обратном направлении. Ваши эксперименты должны показать, что робот выполняет замер, когда блок **Переключатель** (Switch) работает, и что он использует это единственное измерение для определения, является ли условие истинным. В этой программе чтение данных с датчика производится только после того,

как робот завершит движение вперед. Мелодия играет независимо от того, откатывается ли робот назад или произносит фразу «Нет объекта».

Добавление блоков в переключатель

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

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

Режимы отображения блока Переключатель

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

Многократное выполнение блоков Переключатель

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

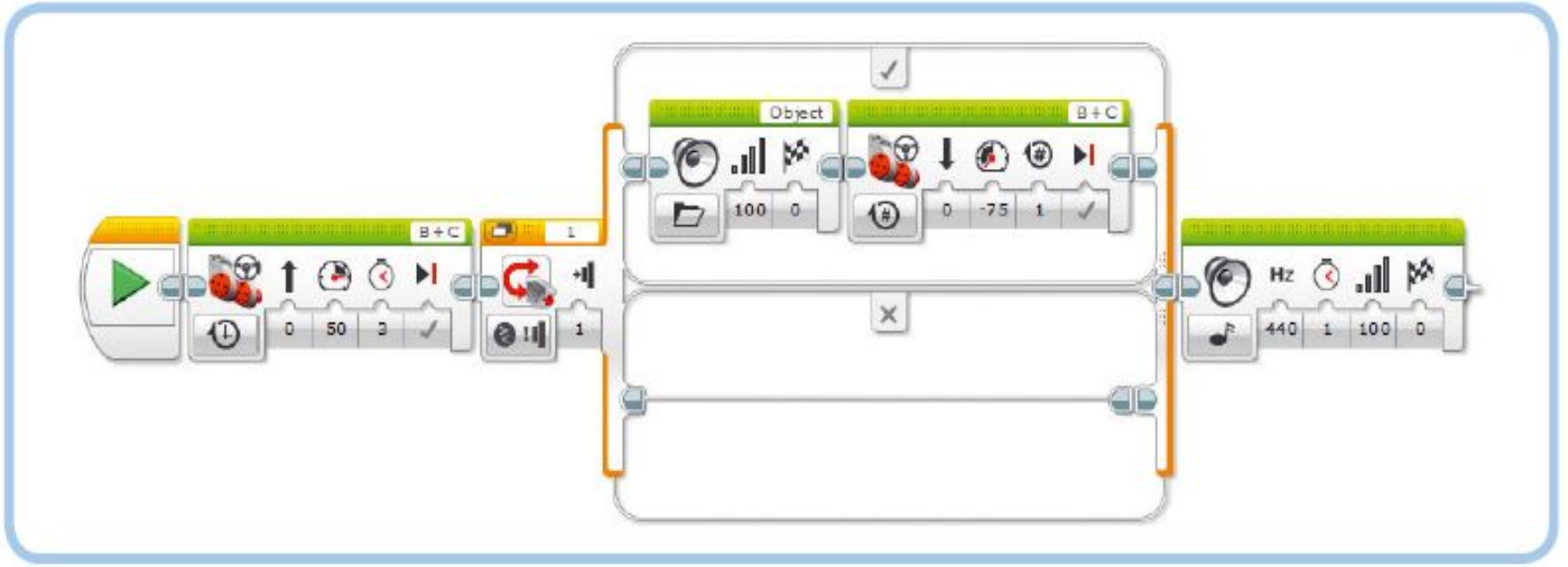


Рис. 6.14. АМодифицированная версия программы TouchSwitch. Блок ***Переключатель*** *не запускает каких-либо блоков, если условие ложно, поэтому по окончании движения в обратном направлении программа сразу же воспроизводит музыку, если кнопка датчика не нажата*

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

Режимы Сравнение, Изменить и Измерение

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

Рис. 6.16. Схема программы для практикума № 28

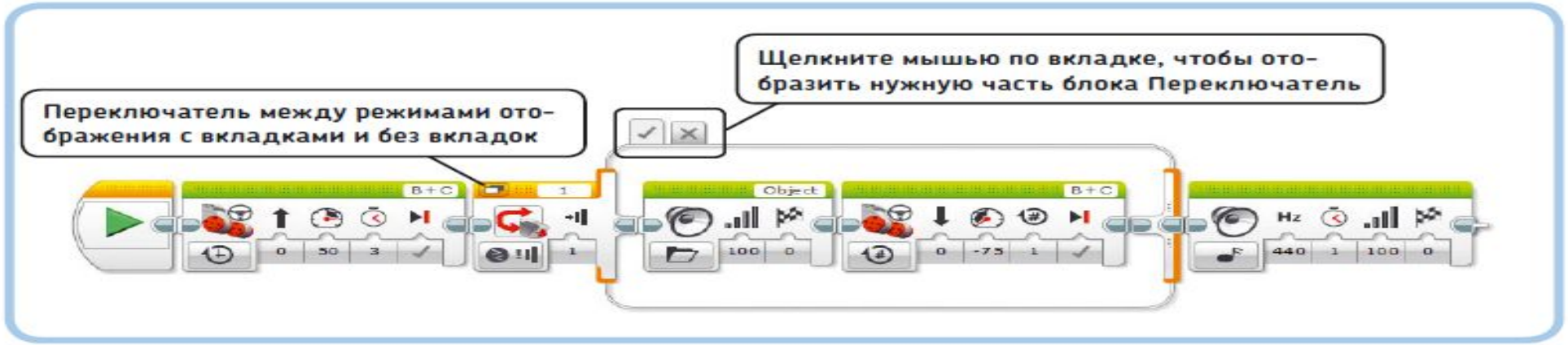


Рис. 6.15. При выборе режима отображения с вкладками размер блока ***Переключатель*** *уменьшится. Режим отображения просто меняет вид блока; он не влияет на то, как программа работает*

Режим Сравнение

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

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

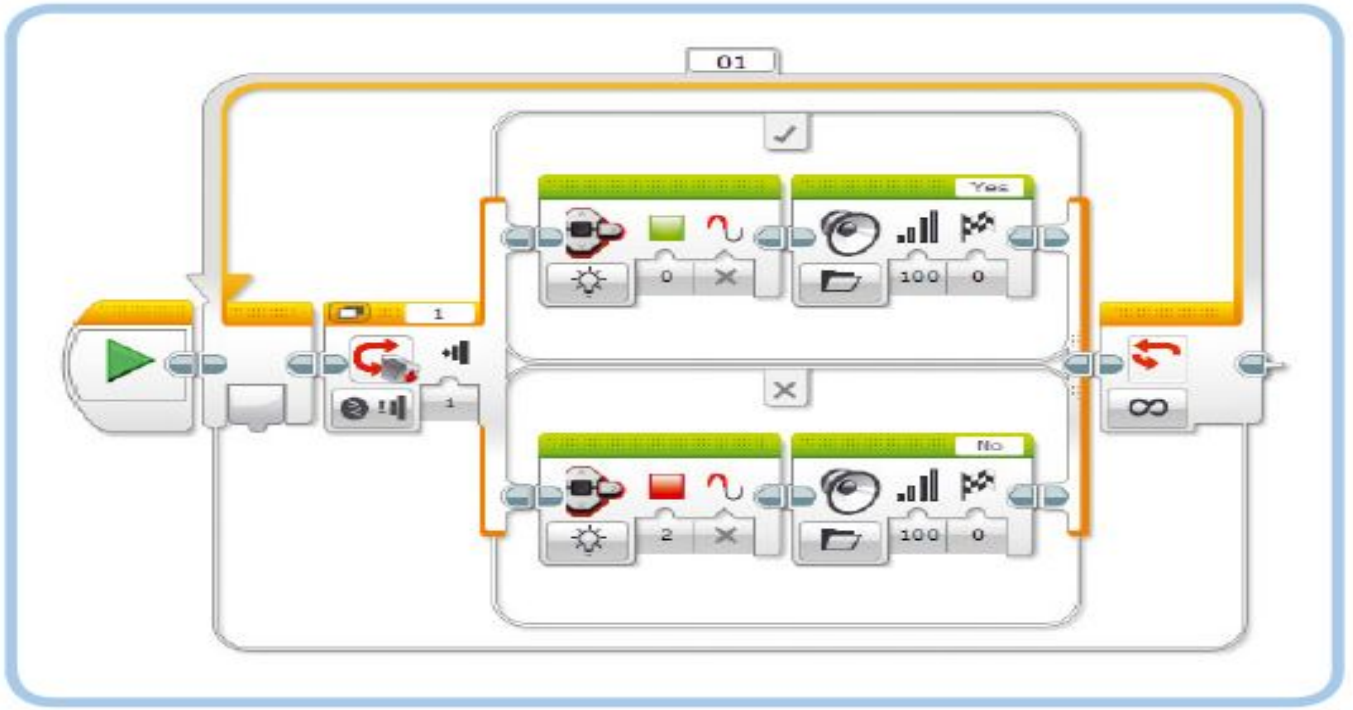


Рис. 6.17. Программа RepeatSwitch

Режим Изменить

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

Режим Измерение

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

Акт

обнаружит такое, которое отличается от первого. Например, датчик цвета.

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

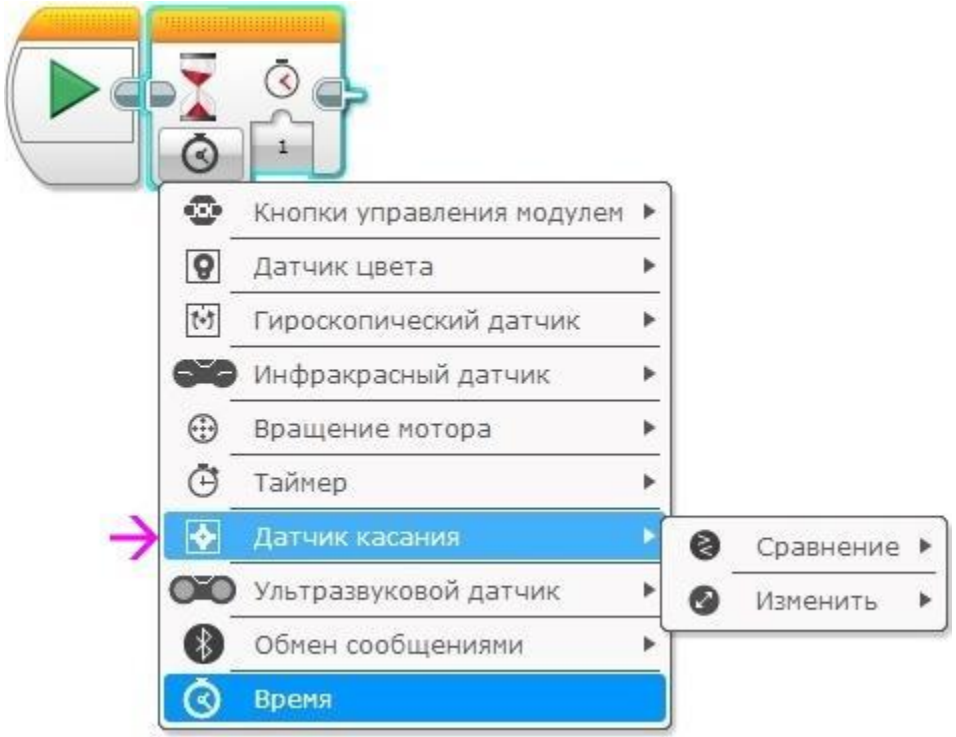
Настройка режимов

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

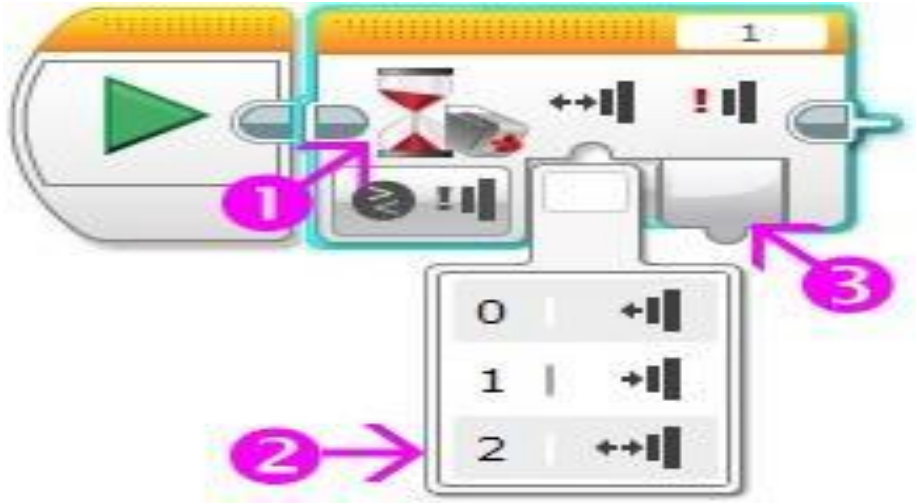
После того как вы выбрали датчик (1) и нужный режим (**Сравнение** (Compare), **Изменить** (Change) или **Измерение** (Measure)) (2), вы выбираете способ работы датчика (3). Датчик касания измеряет только один показатель (состояние красной кнопки), а датчик цвета имеет три режима работы, как показано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

Акти

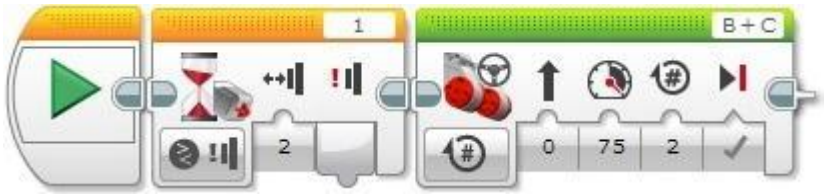
- **Задача №6:** необходимо написать программу, запускающую движение робота по щелчку кнопки.
- **Решение:**
- Само условие задачи подсказывает нам возможное решение: перед началом движения - необходимо дождаться нажатия-отпускания кнопки датчика касания. Возьмем программный блок **"Ожидание"**, изменим режим программного блока на **"Датчик касания"** - **"Сравнение" (Рис. 6)**.



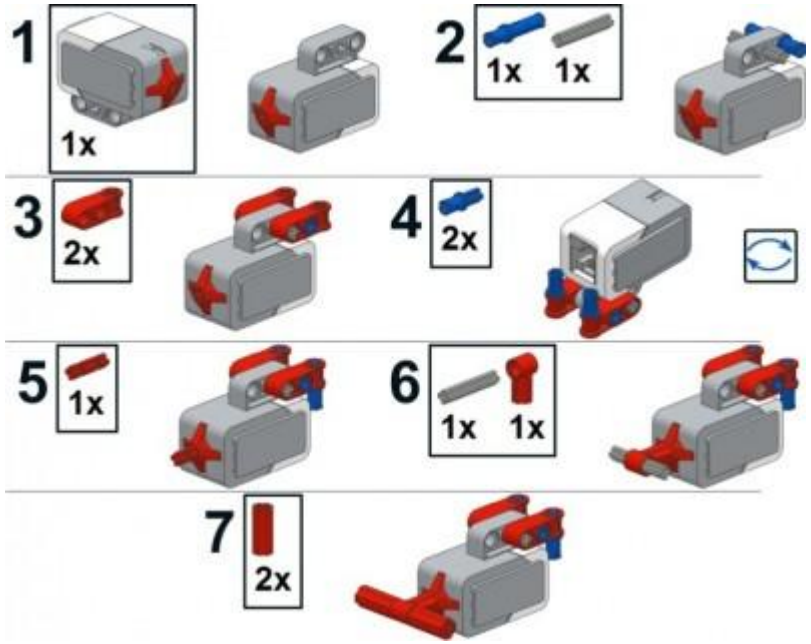
- Как можно увидеть - программный блок **"Ожидание"** сменил свое отображение! Рядом с песочными часами появилось изображение датчика касания **(Рис. 7 поз. 1)**, помогающее в программе визуально оценивать установленный режим работы. Настройка программного блока **"Состояние"** задает требуемое состояние датчика, достижение которого прекратит выполнение блока **"Ожидание" (Рис. 7 поз. 2)**. Настройка **"Состояние"** может принимать следующие значение: **"0"** - **"Отпущено"**, **"1"** - **"Нажатие"**, **"2"** - **"Щелчок"**. Для решения нашей задачи выберем состояние **"Щелчок"**. Вывод **"Измеренное значение" (Рис. 7 поз. 3)** при необходимости позволяет передать окончательное состояние датчика для обработки в другой программный блок.



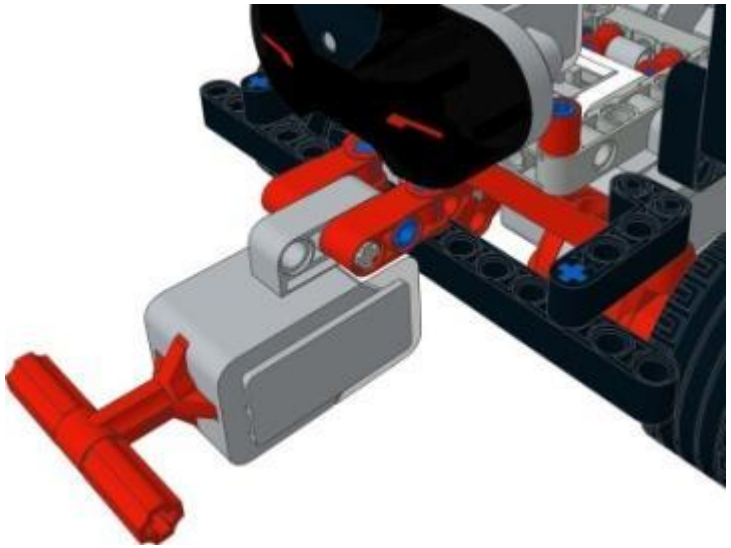
- Итак: при такой настройке блока ожидания выполнение нашей программы будет остановлено до нажатия-отпускания кнопки датчика касания. Только после **"Щелчка"** выполнение будет передано следующему программному блоку. Установим после блока ожидания один программный блок **"Рулевое управление"**, загрузим программу в робота и убедимся в правильности её выполнения! **(Рис. 8)**

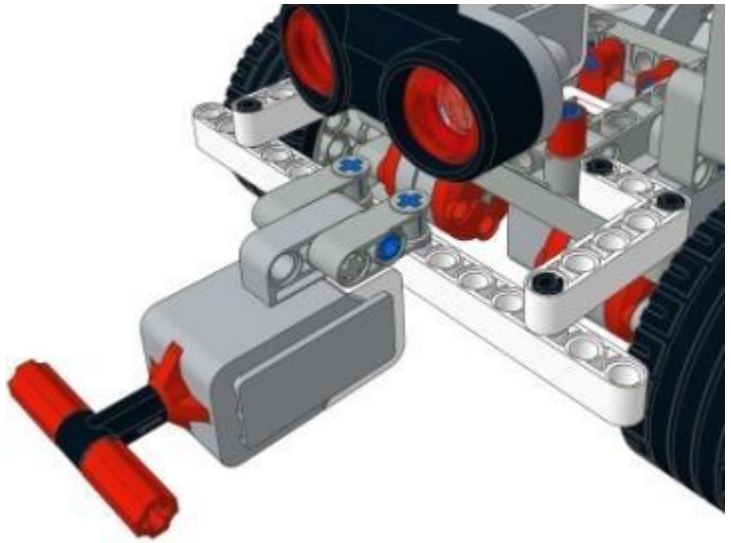


- **Задача №7:** необходимо написать программу, останавливающую робота, столкнувшегося с препятствием.
- Из датчика касания давайте соберем небольшой бампер, который будет нам сигнализировать о том, что наш робот столкнулся с препятствием. Ниже приведены подробные инструкции для сборки, как из домашней, так и из образовательной версии конструктора Lego mindstorms EV3. Можете поэкспериментировать и придумать собственный вариант конструкции.
- **Lego mindstorms EV3 home**

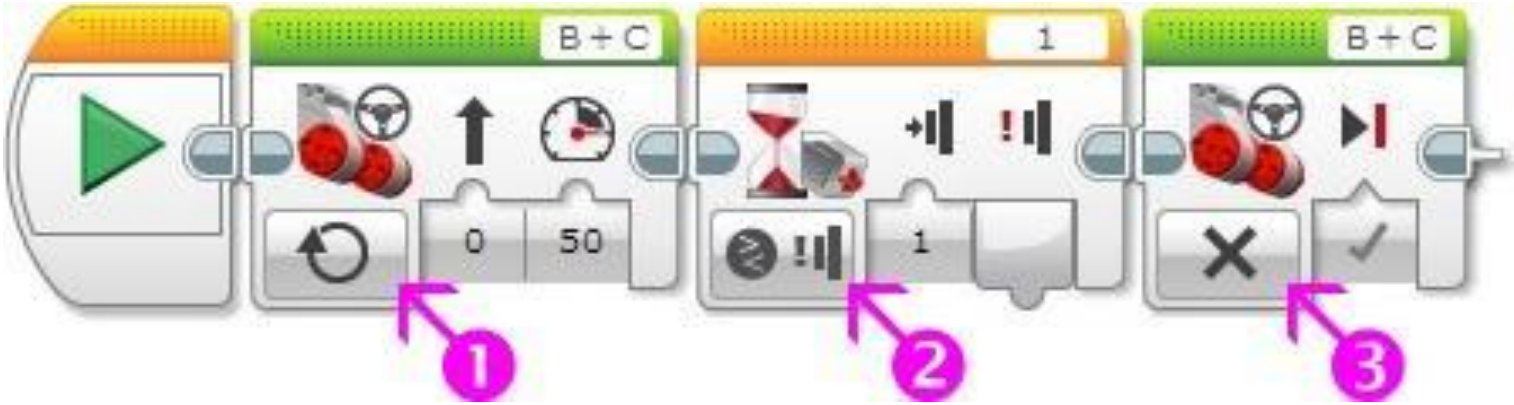


- Получившийся элемент закрепим на передней балке нашего робота и соединим датчик касания с портом "1" модуля EV3.





- Конструкция готова! Приступим к созданию программы. По условию задачи: робот должен двигаться вперед, пока не наткнется на препятствие. В этом случае датчик касания будет нажат! Для решения снова воспользуемся программным блоком "**Ожидание**".
- **Решение:**
- Начать прямолинейное движение вперед **(Рис. 9 поз. 1)**.
- Ждать, пока датчик касания не будет нажат **(Рис. 9 поз. 2)**.
- Прекратить движение вперед **(Рис. 9 поз. 3)**.



- Для решения следующей задачи нам понадобится программный блок **"Цикл"** Оранжевой палитры.
- **Задача №8:** необходимо написать программу, заставляющую робота двигаться вперед, при наезде на препятствие - отъезжать назад, поворачивать вправо на 90 градусов и продолжать движение вперед до следующего препятствия.
- **Подсказка:** напишите и протестируйте программу **движения - отъезда - поворота**, а затем поместите эти блоки внутрь программного блока **"Цикл"**.