**Ardublock** — графическая среда программирования, обладающая интуитивно понятным интерфейсом и позволяющая программировать роботов на основе контроллера Arduino. Среда позволяет «визуализировать код», что способствует быстрому освоению основ программирования.

**Вводные понятия**

Программа в Ardublock составляется из **блоков**, которые могут иметь входы и возвращаемое значение. **Входы** располагаются в правой части блоков. Они могут служить разным целям: подстановка аргументов, добавление команд, составление сложного выражения и т.д. Ко входам могут присоединяться другие блоки. **Возвращаемым значением** будем называть результат выполнения действий блока, который может подаваться на вход другим блокам. Графически оно обозначается выпуклой фигурой определенной формы слева от блока. Возвращаемые значения могут иметь различные типы данных.



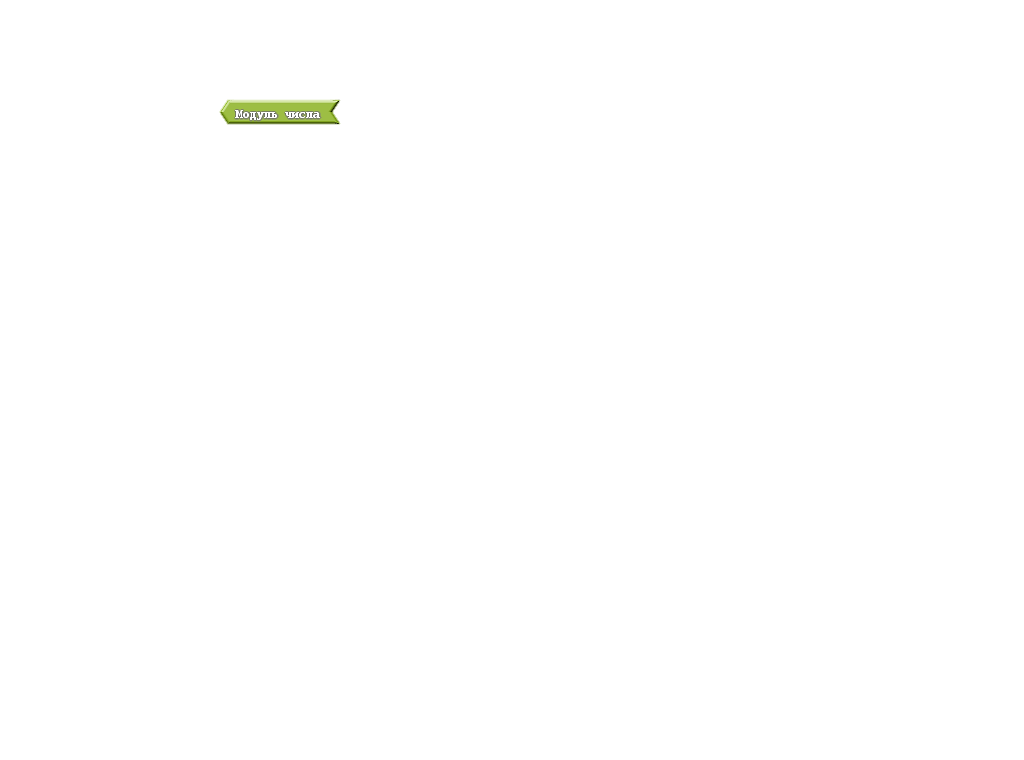
Возвращаемое значение

Входы

**Типы данных**

В Ardublock вы можете использовать 7 типов данных. Под **типом данных** будем понимать множество значений одного класса (числа определенного диапазона, буквы) и набор операций над этими множествами. Давайте познакомимся с каждым типом данных Ardublock.

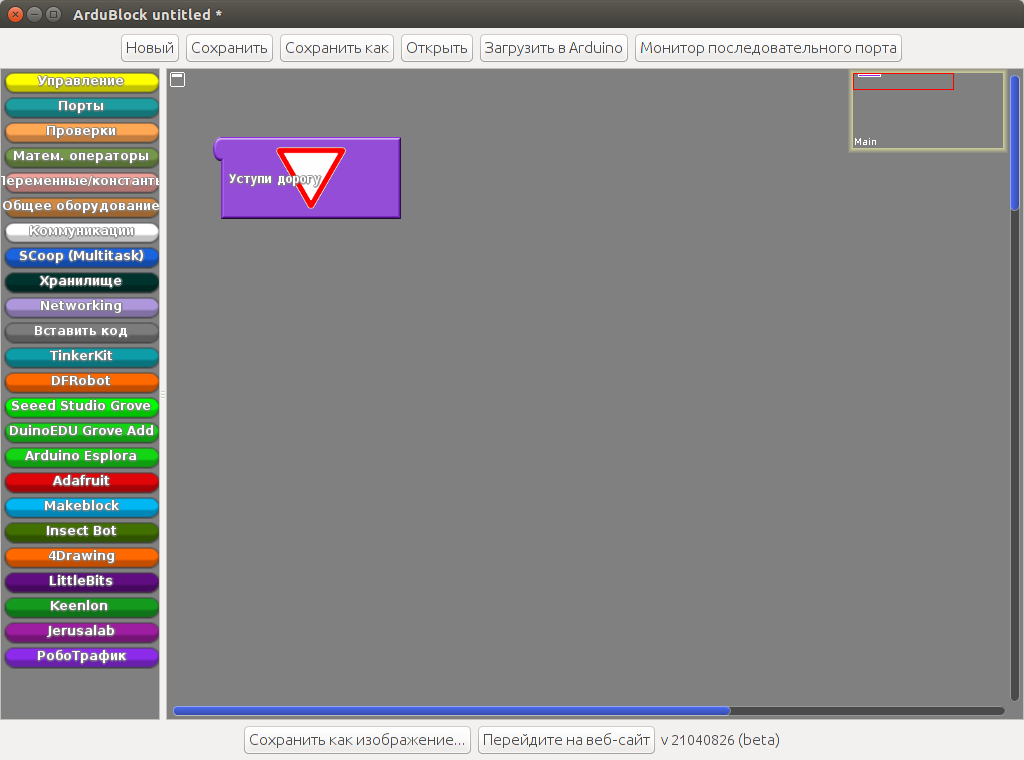
**Целое число**. Класс включает в себя целые числа в диапазоне от -32.768 до 32.767. Обозначается треугольником в левой верхней части блока. Например, блок **Модуль числа** возвращает целочисленное значение (а также принимает целочисленное значение на вход).



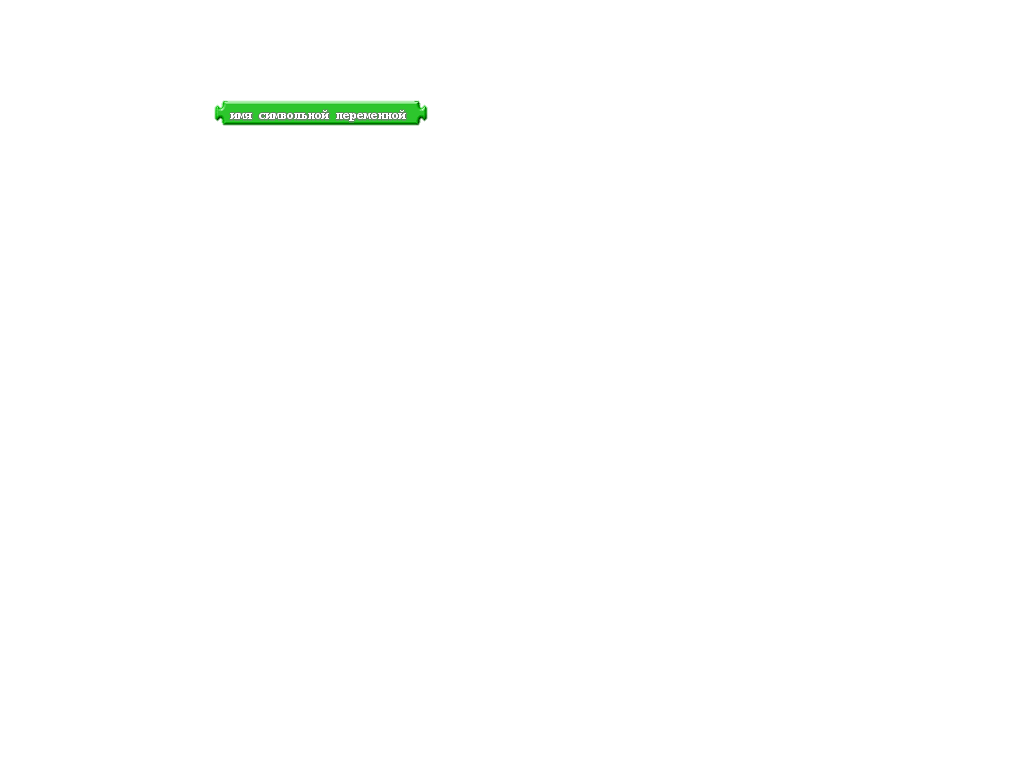
**Длинное целое**. Так мы будем обозначать тип **беззнаковое длинное целое**. Он представляет собой множество целых чисел в диапазоне от 0 до 4.294.967.295. Обозначается также, как целое число (так как используется с теми же блоками).

**Десятичное число.** Состоит из десятичных значений. Значения варьируются в диапазоне +/- 3.4028235E+38. Обозначается аналогично целому числу.

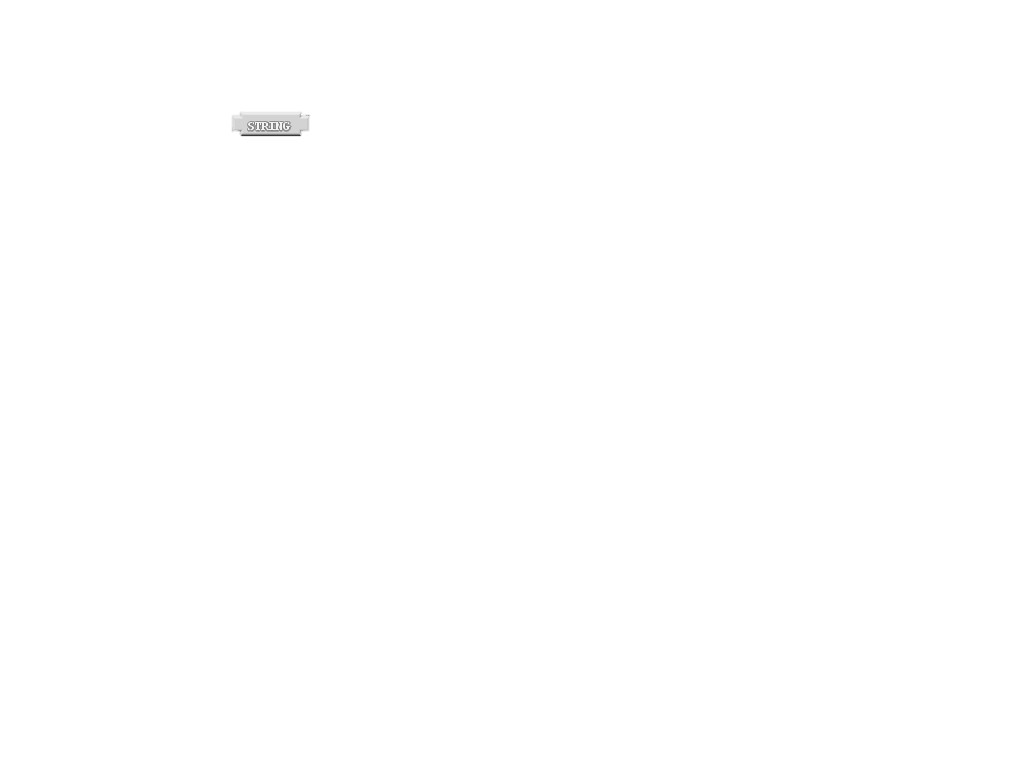
**Логическое значение**. Класс содержит два значения: ИСТИНА и ЛОЖЬ. Используется в логических выражениях и в блоках с условиями. Обозначается закруглённым выступом.



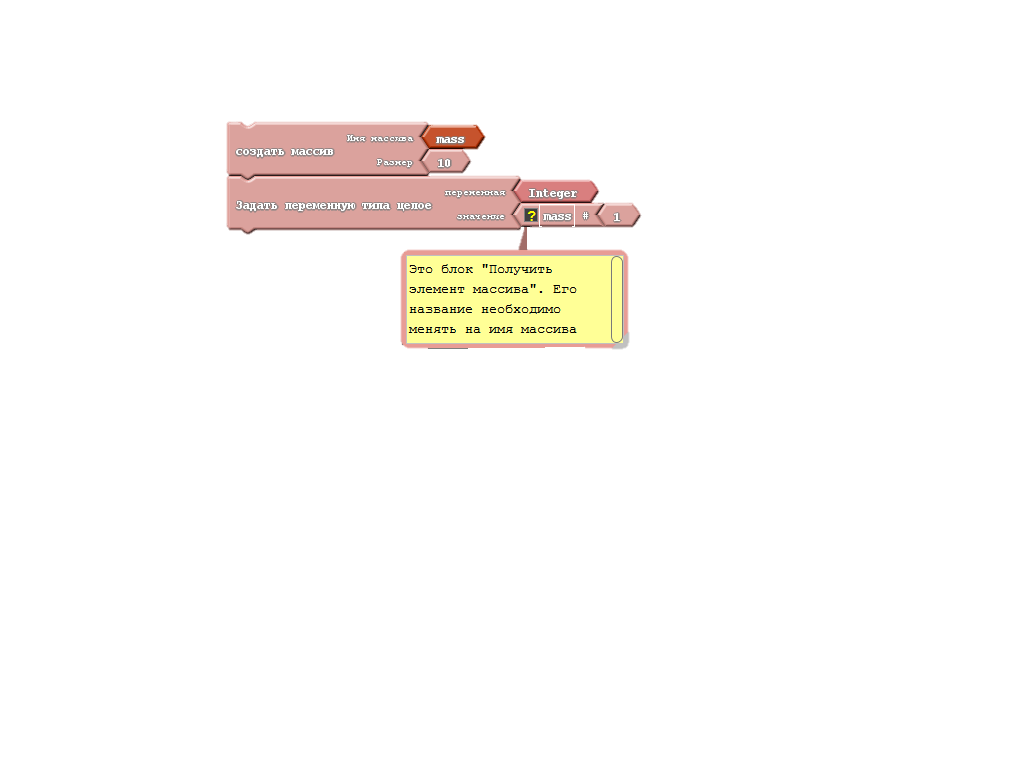
**Символ**. Этот тип данных предназначен для хранения одного символа.

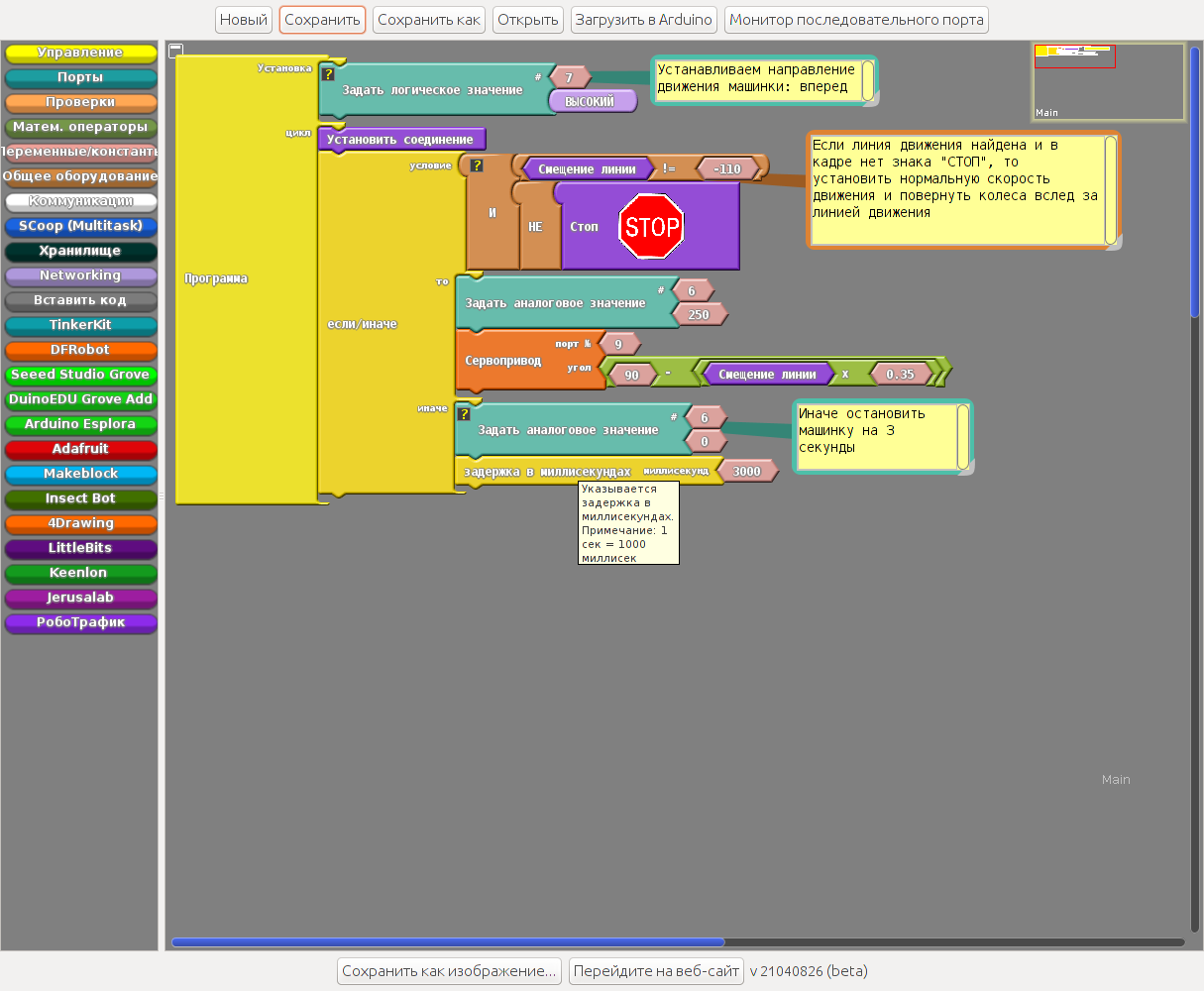


**Строка**. Представляет собой последовательность символов. Используется в основном для вывода сообщений.

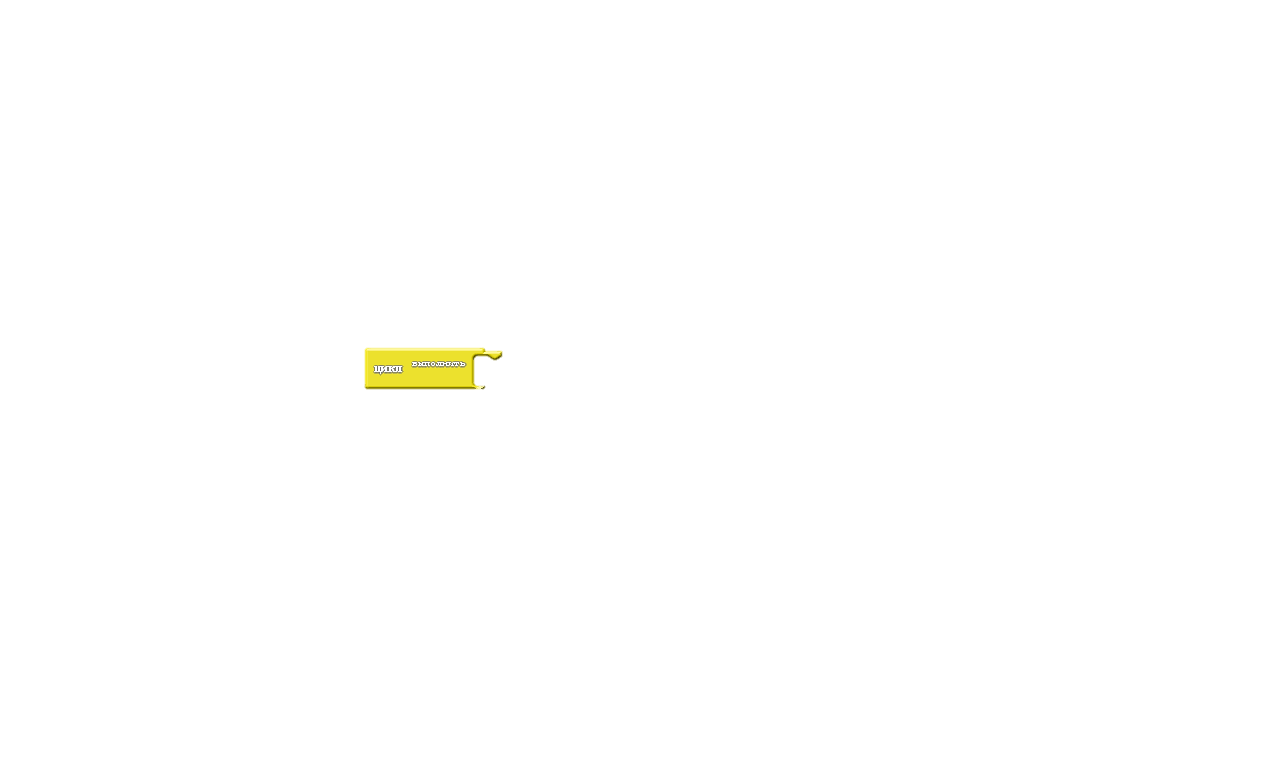
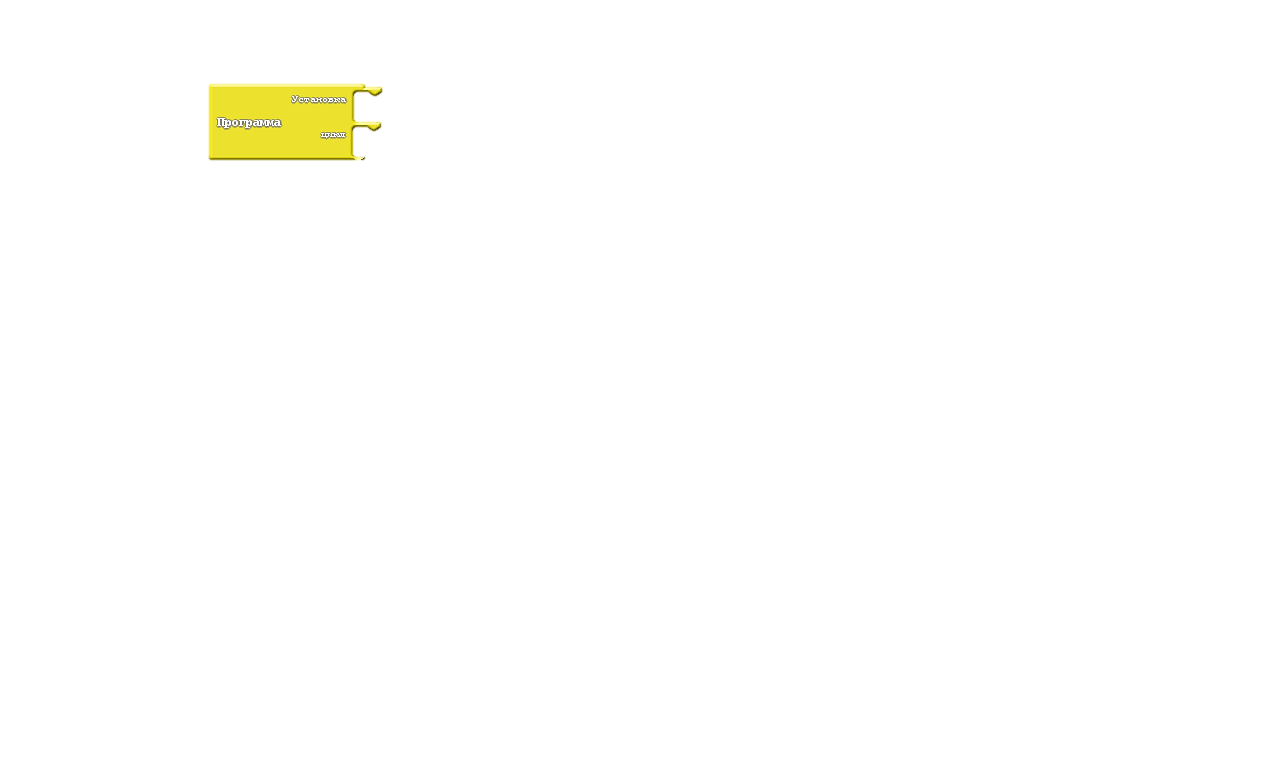


**Целочисленный массив**. Является структурой данных в виде набора целых чисел, расположенных непосредственно друг за другом. Проще говоря, это последовательность целочисленных значений, к каждому из которых можно обратиться по порядковому номеру. **Длиной массива** будем называть количество элементов в нем. Нумерация в массивах может начинаться с 0 или 1. В Ardublock нумерация начинается с 1. Это означает, что первый элемент будет иметь порядковый номер 1. Сам массив в Ardublock возвращает значение, но можно получить любой элемент массива с помощью блока **Получить элемент массива**.



Рассмотрим интерфейс Ardublock. Он состоит из четырех частей: верхней панели, боковой панели, рабочего поля и панели навигации. Верхняя панель служит для создания, сохранения и открытия **скетчей** (так называются программы на графическом языке программирования). На рабочем поле из блоков составляется программа. Боковая панель состоит из наборов блоков, из которых составляются программы. Панель навигации располагается в правом верхнем углу рабочей области и служит для удобного перемещения по ней.

Любая программа в Ardublock начинается с блока **Программа** или **Цикл** из набора **Управление**. Блок **Программа** состоит из двух частей: «Установка» и «Цикл». Первая часть содержит начальные установки программы. Сюда входят: создание переменных и задание их начальных значений, задание начальных значений для портов и другое. Эта часть выполняется один раз при запуске программы. Вторая часть блока **Программа** выполняется бесконечно, пока программа не завершится. Здесь содержатся основная логика программы. Блок **Цикл** состоит из одной части («Выполнить») и эквивалентен второй части блока **Программа**.

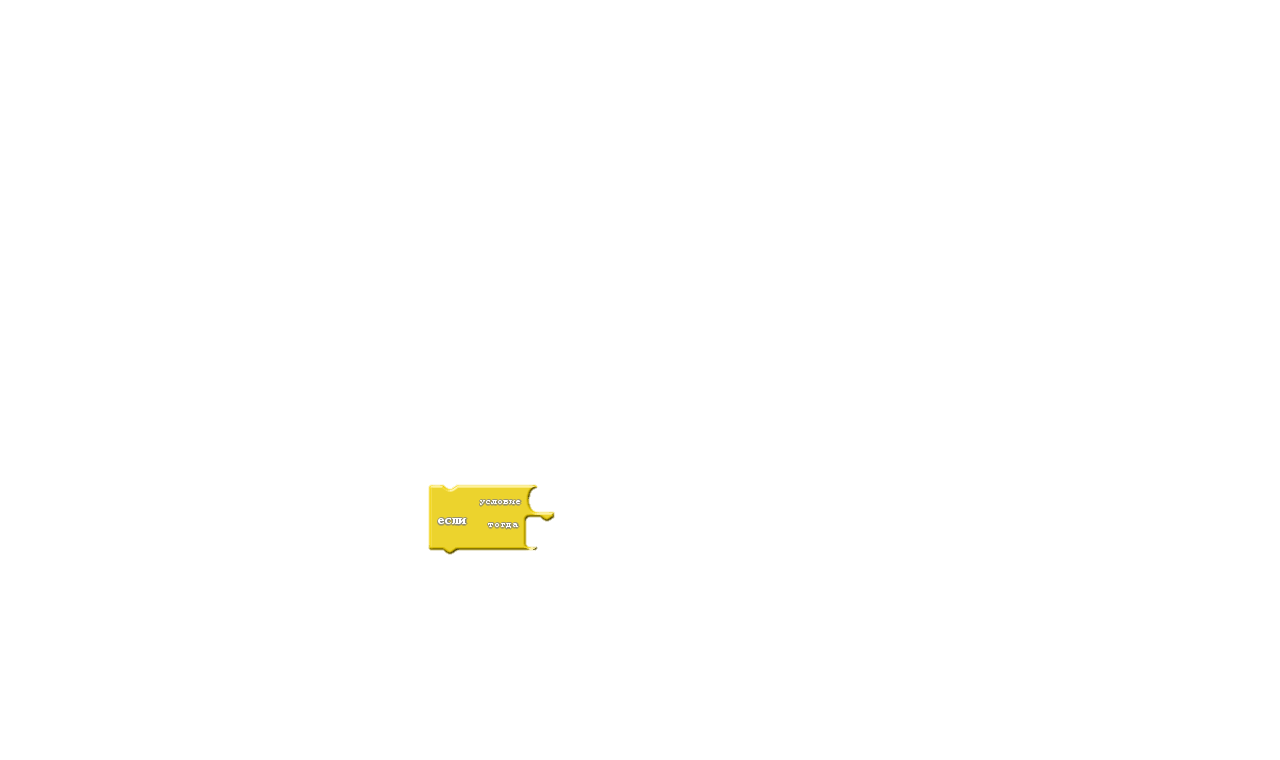
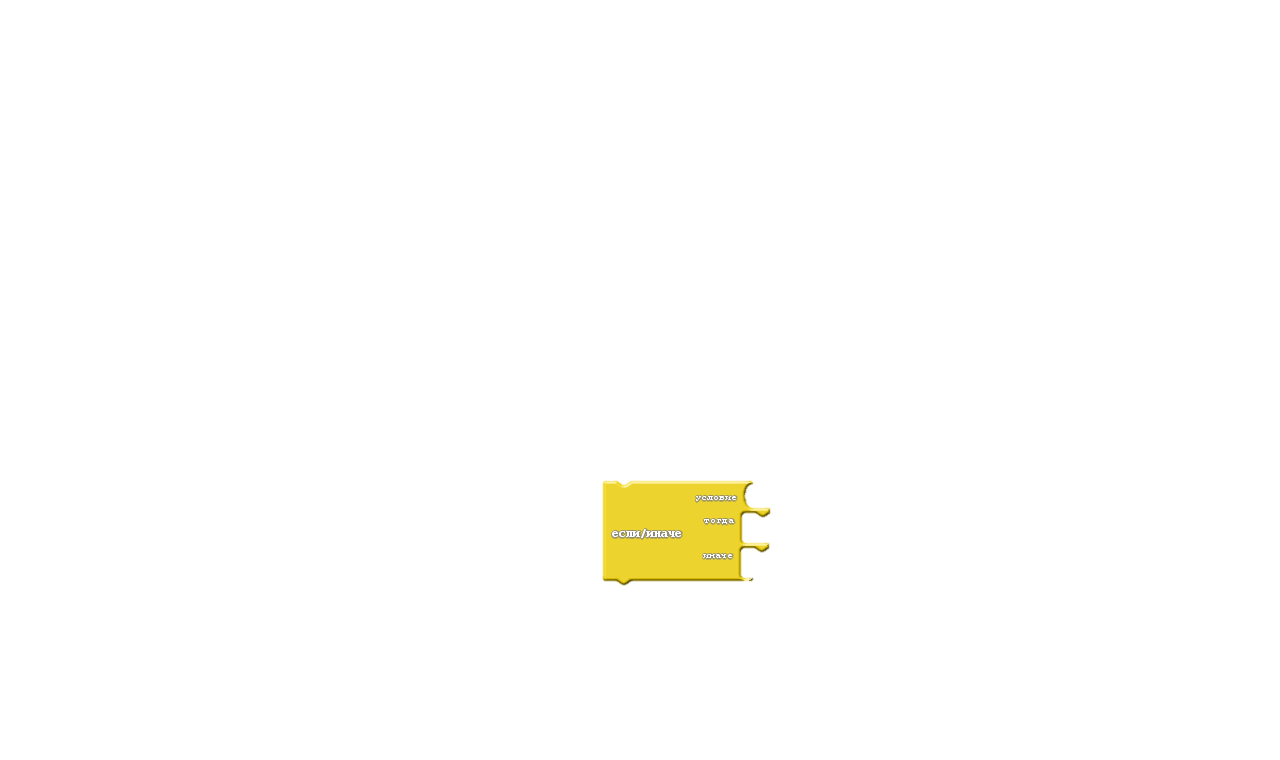


После размещения одного из этих блоков на рабочем поле можно начинать построение программы.

**Управление.** Этот набор состоит из блоков условий, циклов, функций и задержек, а также блоков «Программа» и «Цикл».

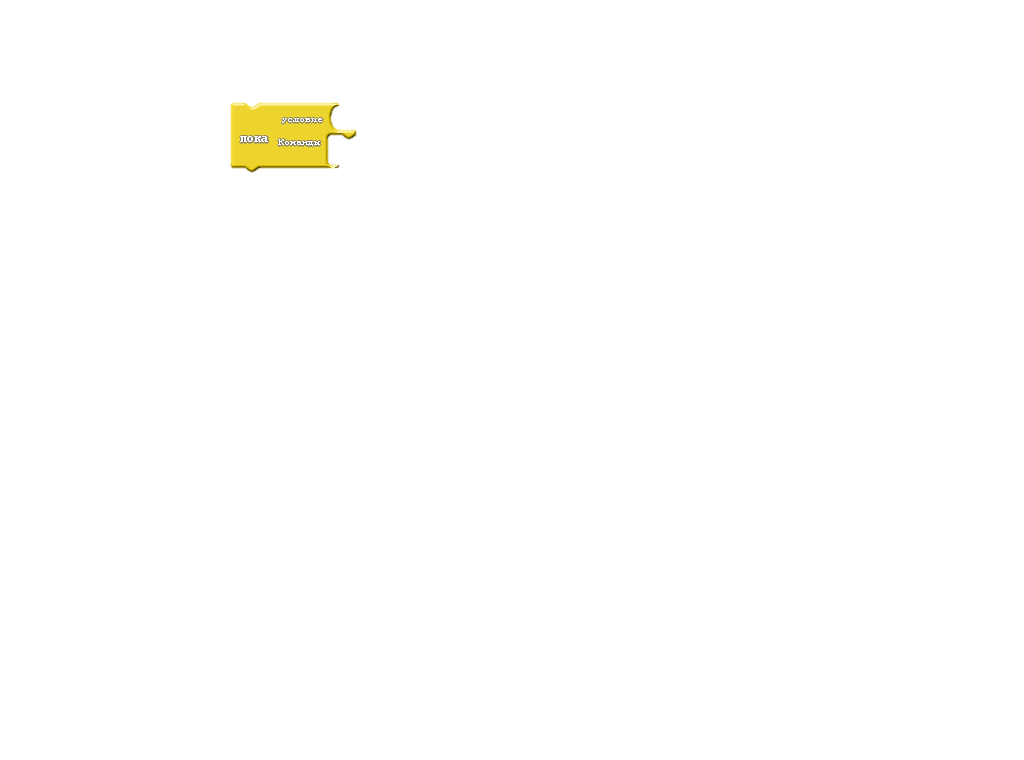
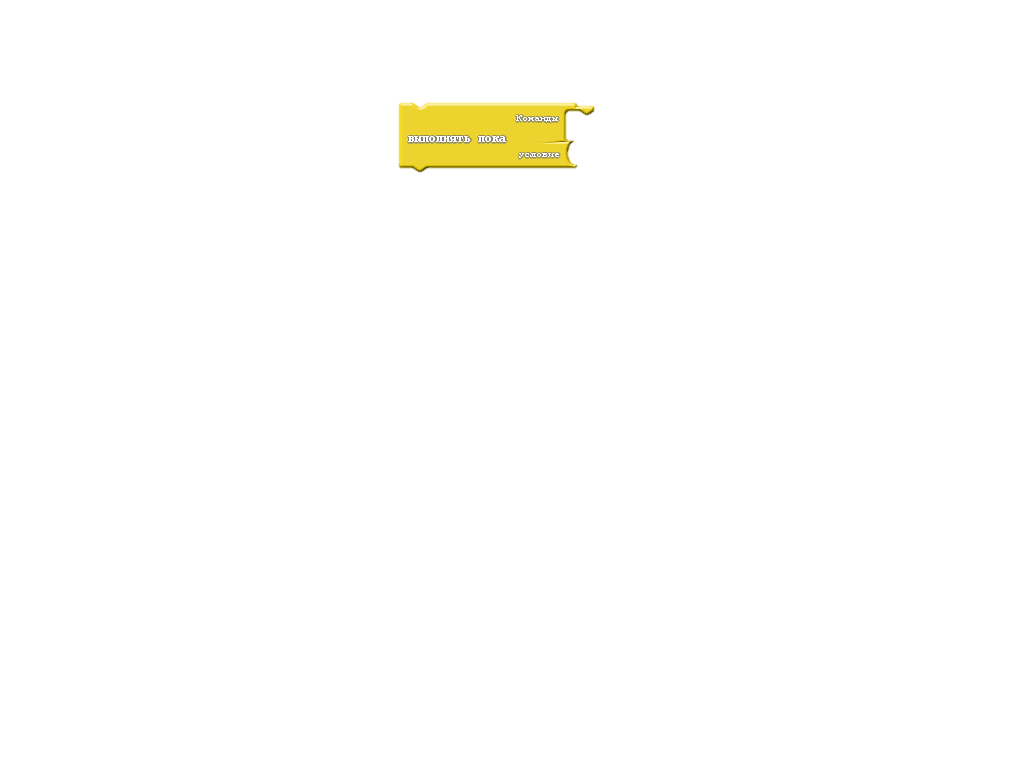
Блок **Если** позволяет выполнить набор команд при соблюдении определенного условия. Ко входу «условие» присоединяется логическое выражение (сравнение, значение логической переменной, логическая константа). Если это логическое выражение возвращает значение ИСТИНА, то выполняются блоки, соединенные со входом «тогда».

Блок **Если/Иначе** эквивалентен блоку «Если» за одним исключением – он имеет еще один вход «иначе». Блоки, соединенные с этим входом, выполнятся в том случае, если логическое выражение вернет значение ЛОЖЬ.

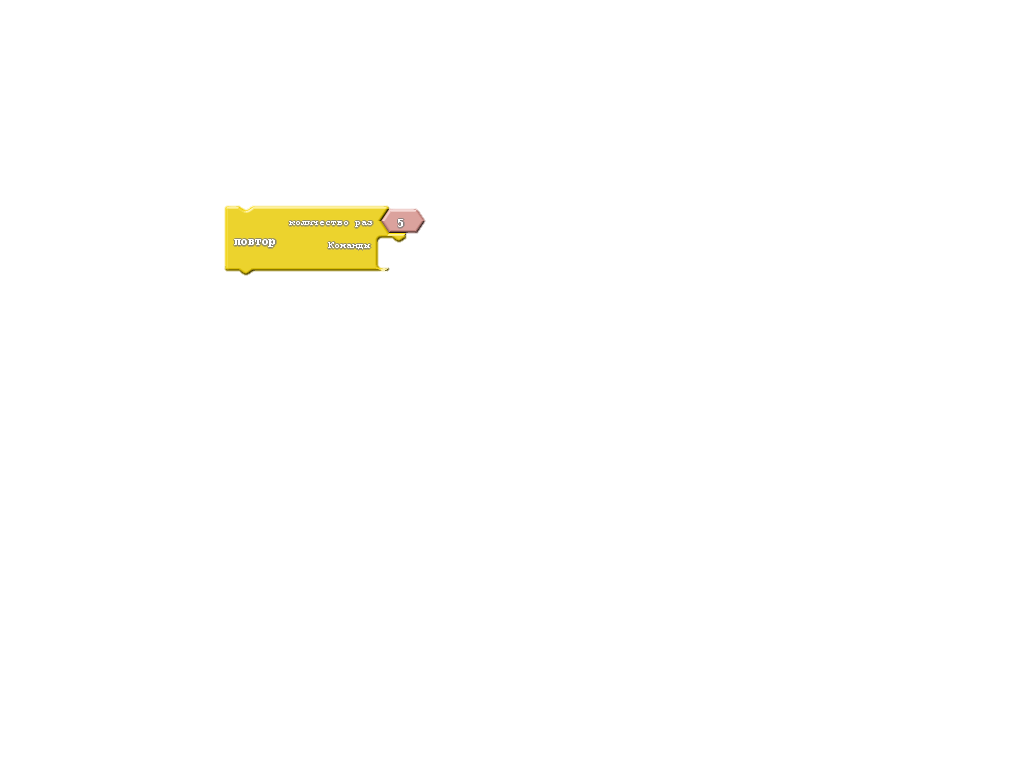
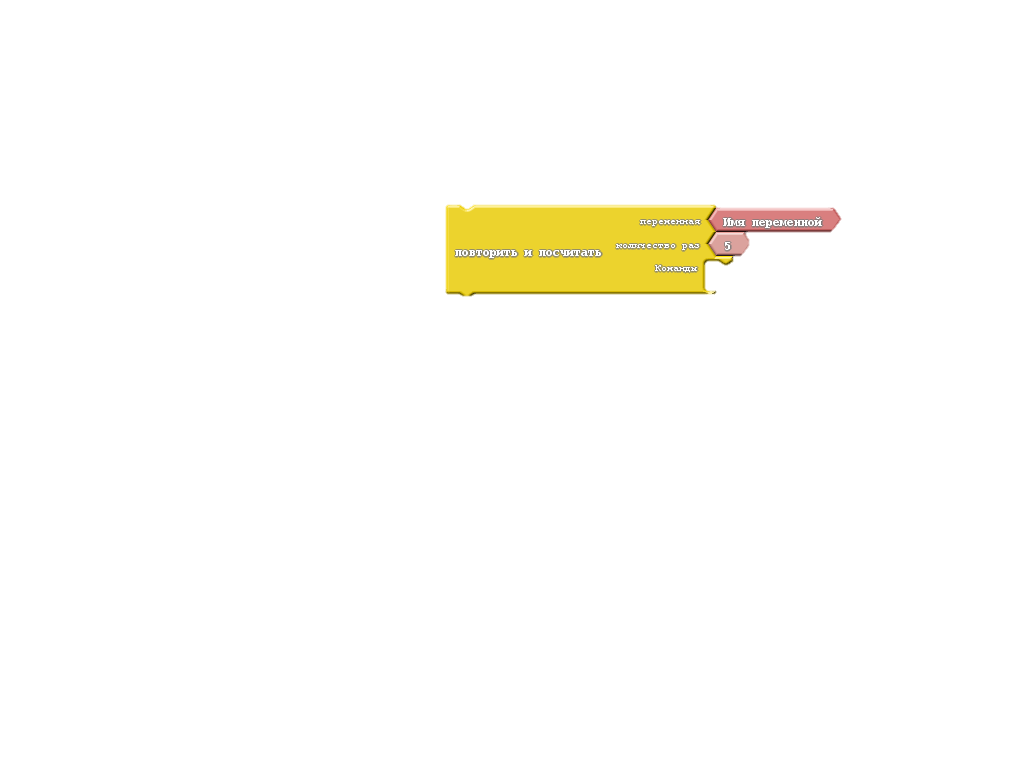
 

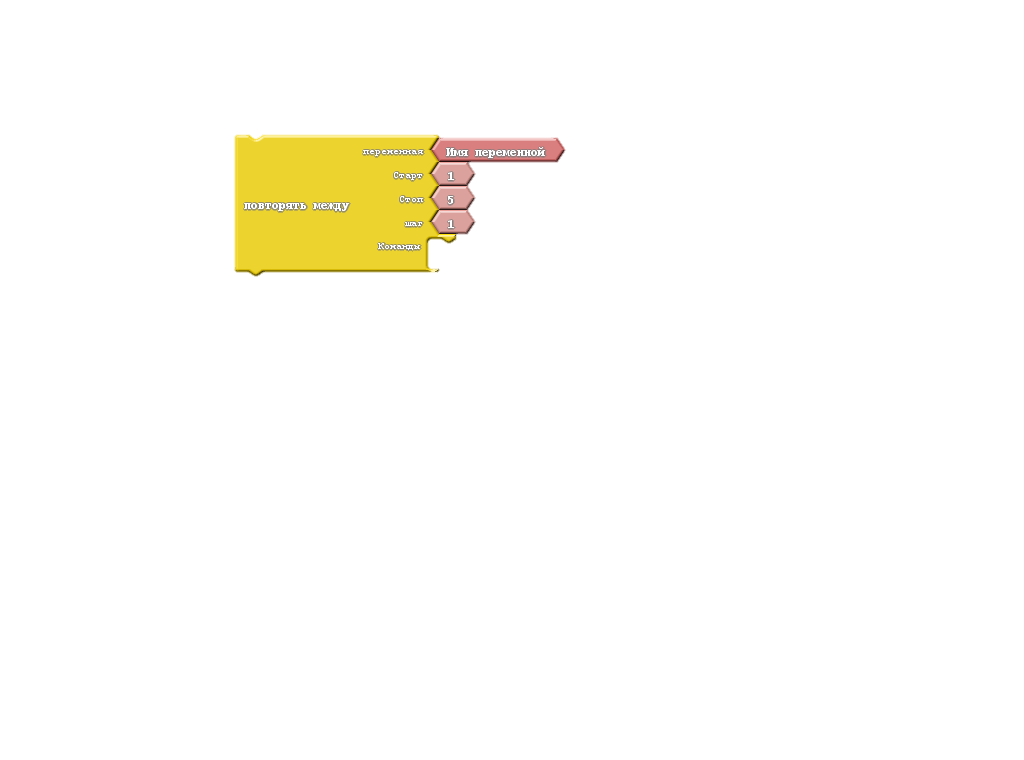
**Блоки циклов** позволяют выполнить набор команд указанное количество раз или выполнять до наступления определенного условия.

В блоке **Пока** команды выполняются, пока выражение, соединенное со входом «условие», возвращает значение ИСТИНА. Блок **Выполнять пока** отличается тем, что набор команд выполнится хотя бы один раз перед проверкой условия (поэтому его называют циклом с постусловием).

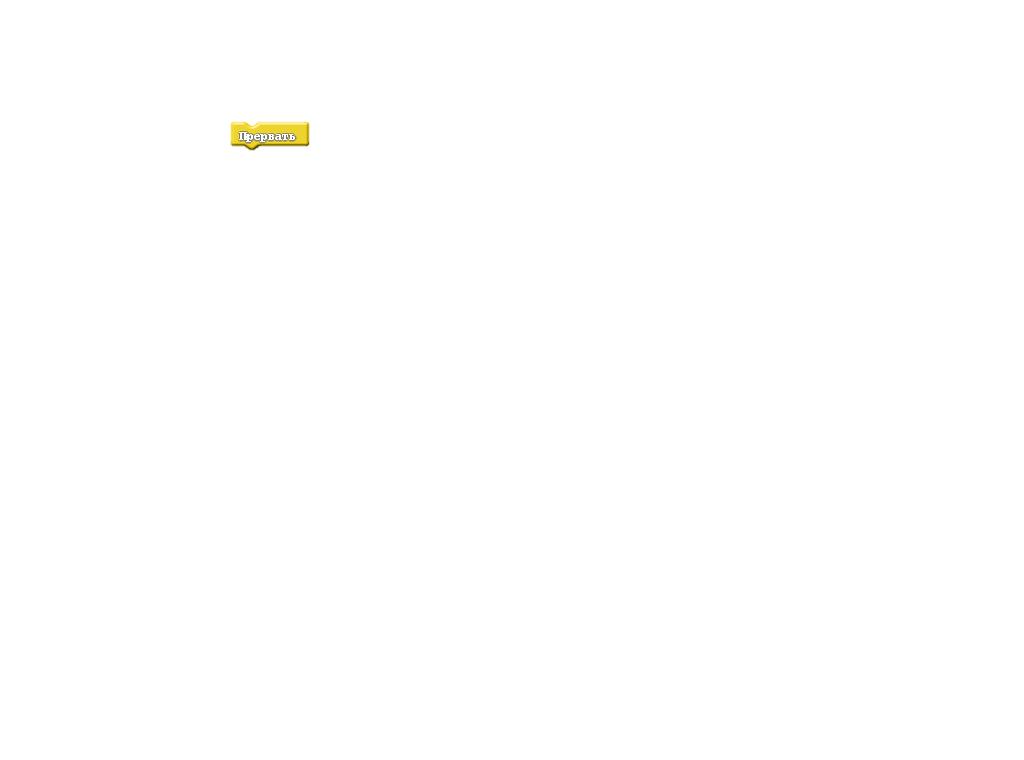
 ****

Блок **Повтор** позволяет выполнить набор команд заданное количество раз. Количество проходов может задаваться переменной, константой, математическим выражением или другим блоком, возвращающим целочисленное значение. Блок **Повторить и посчитать** позволяет не только выполнить набор команд несколько раз, но и записать *реальное* количество повторов в целочисленную переменную. *Реальное количество повторов может отличаться от заданного, так как повтор выполнения команд в цикле можно прекратить блоком* **Прервать***.* Блок **Повторять между** также позволяет выполнить набор команд несколько раз, но работает он немного иначе. У этого блока есть 4 целочисленных входа: «переменная», «Старт», «Стоп» и «шагов». На входе «Старт» задается начальное значение целочисленной переменной, которая соединена со входом «переменная». На входе «Стоп» задается **верхняя граница** для переменной. То есть, когда значение переменной станет большим либо равным **верхней границе**, выполнение команд в цикле прекратится. На входе «шаг» задается шаг приращения переменной за один проход цикла. **Замечание:** *переменная в последнем блоке хранит не число повторов, а текущее значение переменной.*

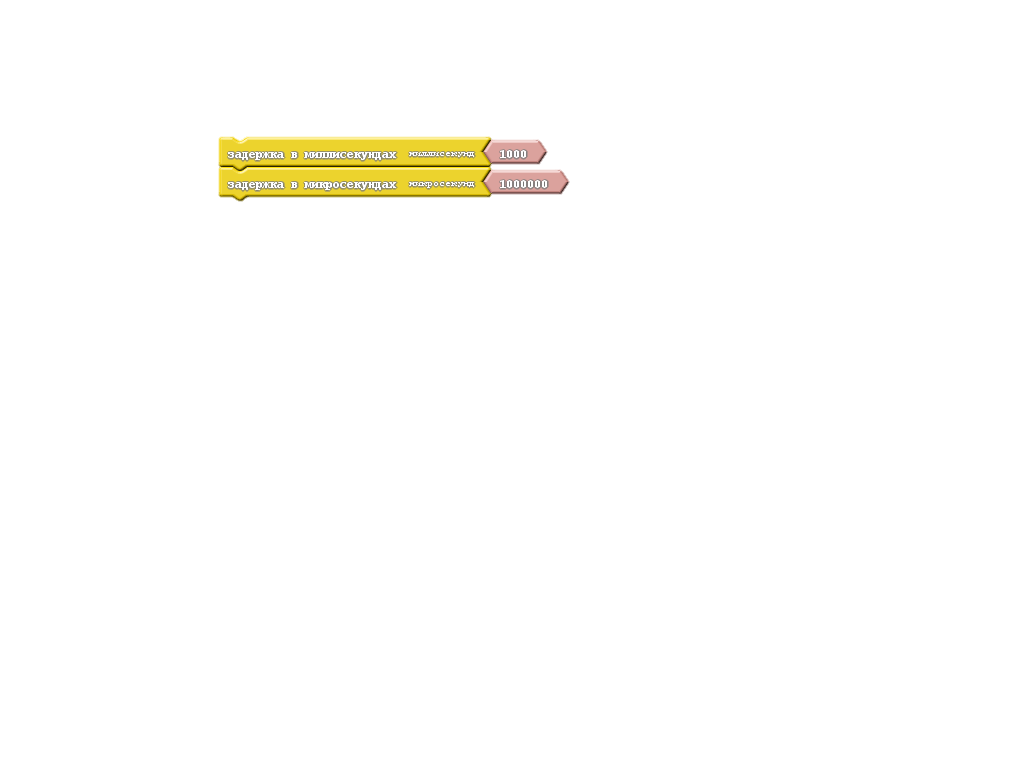
 ****



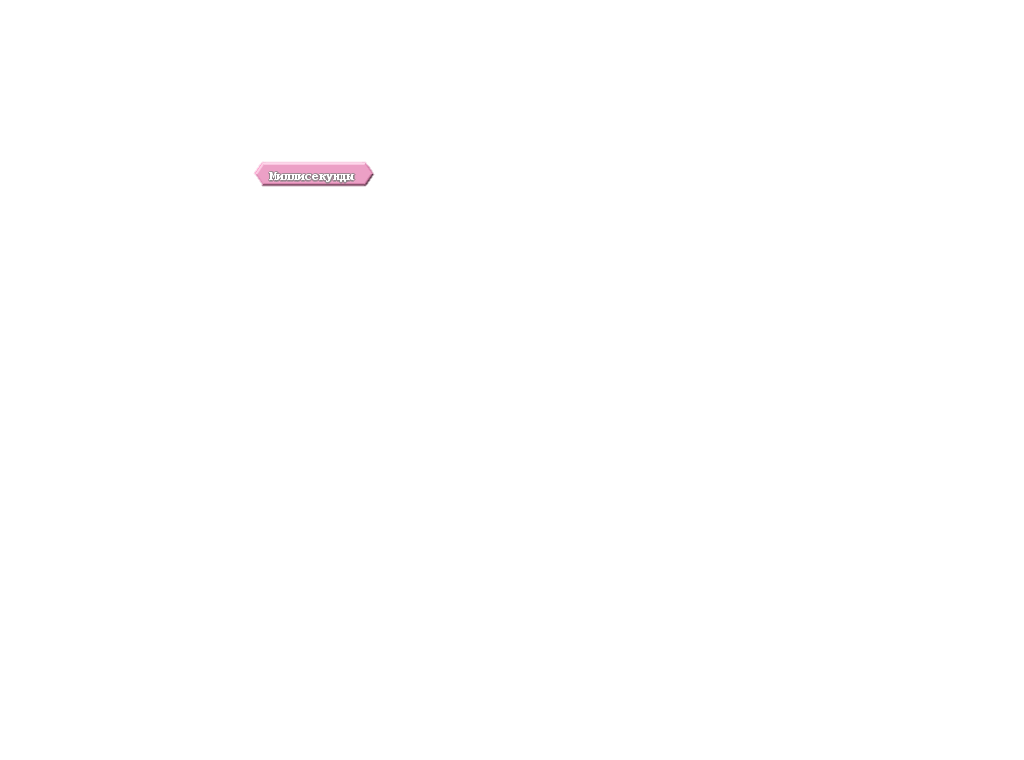
Блок **Прервать** служит для прекращения повтора выполнения команд в блоках циклов. Используется, когда в одном из блоков циклов при наступлении определенного события необходимо прервать выполнение команд.



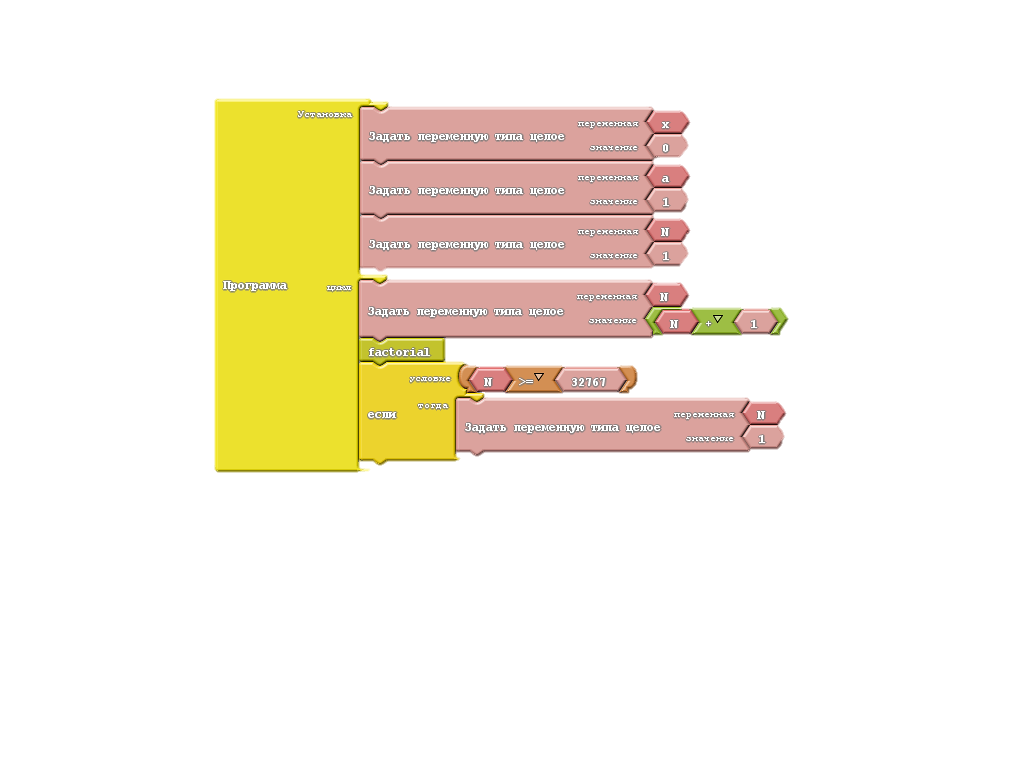
**Блоки задержек** нужны для остановки выполнения программы на определенный временной промежуток. Задержку можно задать в миллисекундах или микросекундах.

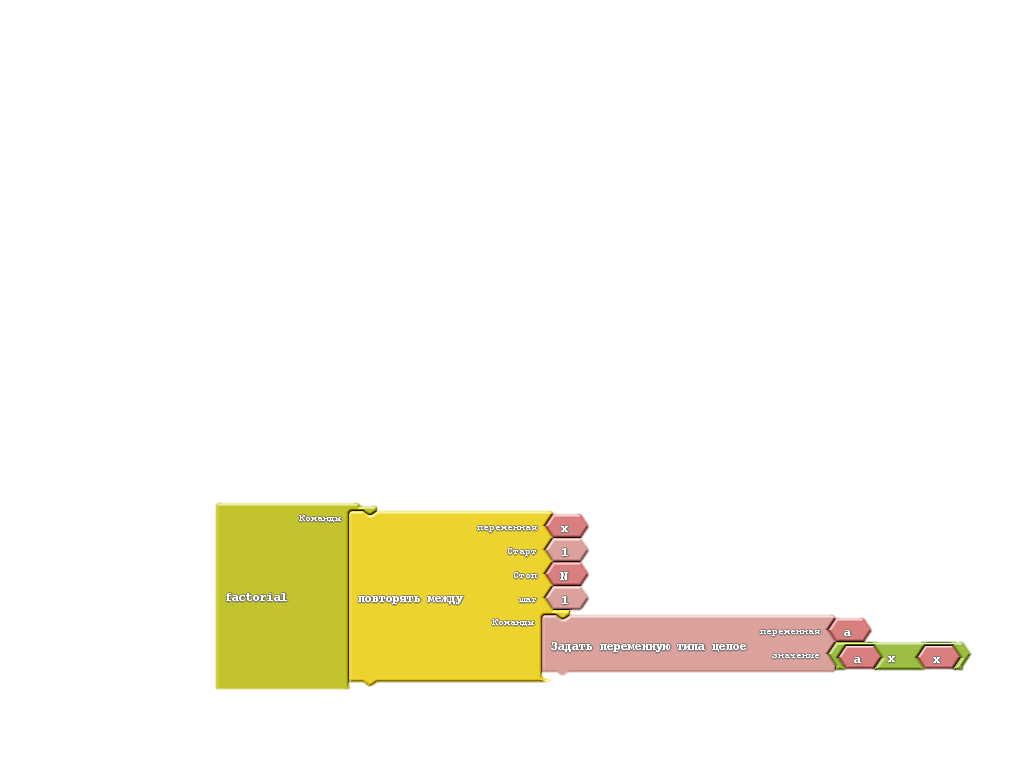


Блок **Таймер** возвращает целочисленное значение, которое показывает, сколько миллисекунд прошло с момента запуска программы.



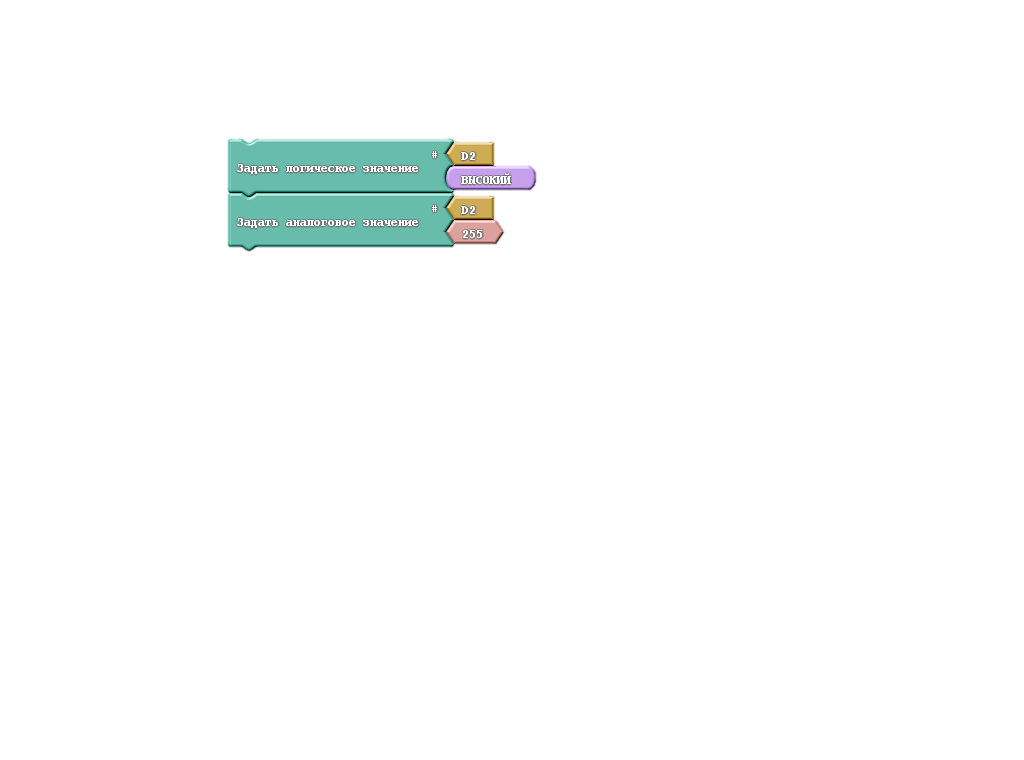
**Блоки функций** позволяют создать или вызвать функцию. Функция представляет собой обособленный набор команд. Блоки создания функций размещаются на рабочей области отдельно (с основной программой они не соединяются). Эти блоки обязательно должны иметь имя, состоящее из латинских букв, цифр и знака подчеркивания (*не может начинаться с цифры*). Блок **Создать функцию с аргументом** позволяет передавать в функцию целочисленный параметр, который можно использовать внутри функции. Для вызова функции нужно соединить блок **Вызвать функцию** или **Вызвать функцию с аргументом** (в зависимости от того, имеет функция аргумент или нет) с основной программой и ввести имя функции. **Пример:** программа вычисляет факториалы чисел в диапазоне от 0 до 32.537 (верхняя граница для значения целочисленной переменной).



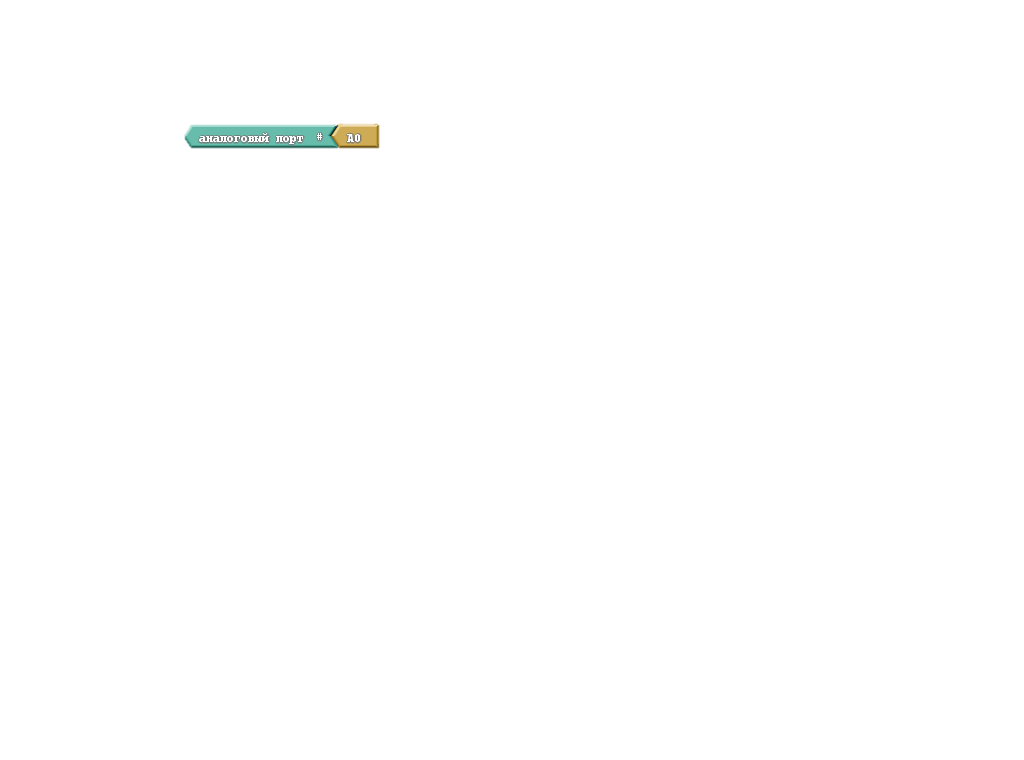
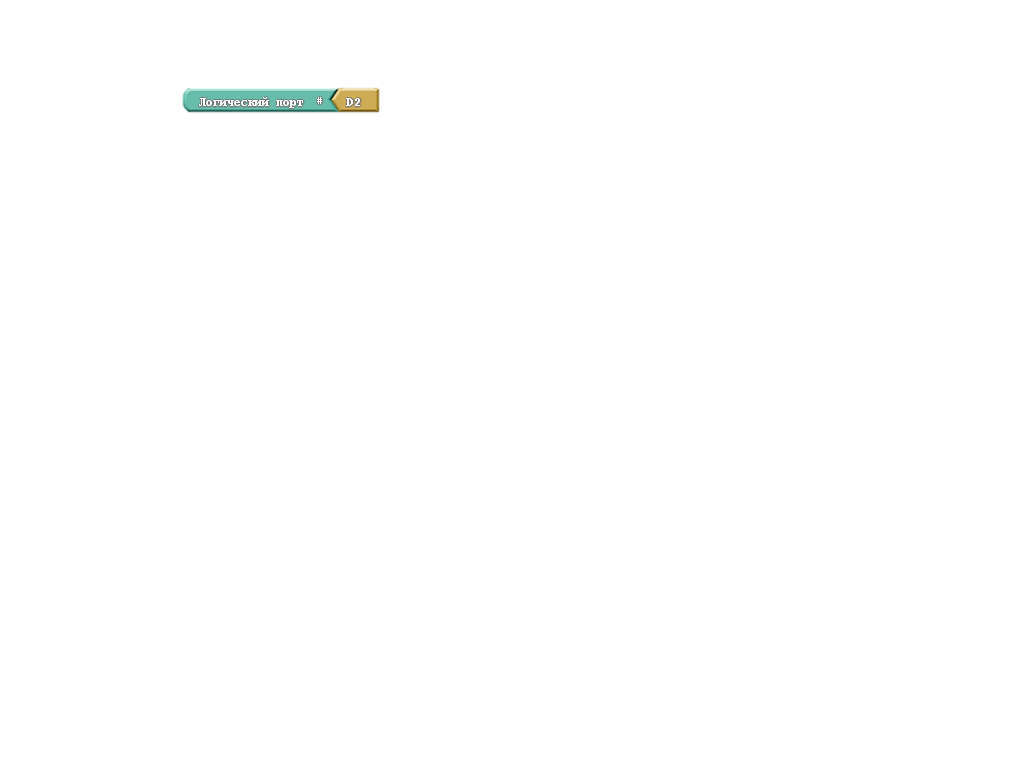


**Порты.** Данный набор состоит из блоков, позволяющих задать или прочитать значение с устройств, подключенных к определенным пинам. На первом входе у них указывается номер пина, к которому они подключены.

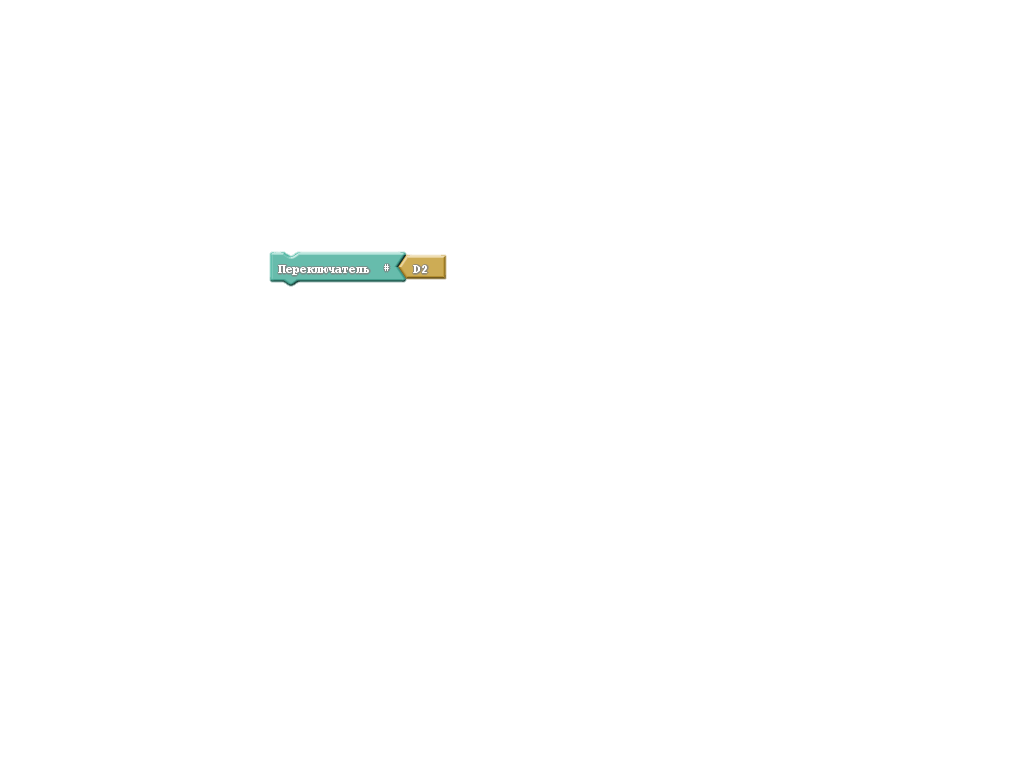
Блоки **Задать логическое значение** и **Задать аналоговое значение** служат для задания соответственно логического и целочисленного значения на устройстве, подключенном к указанному пину.



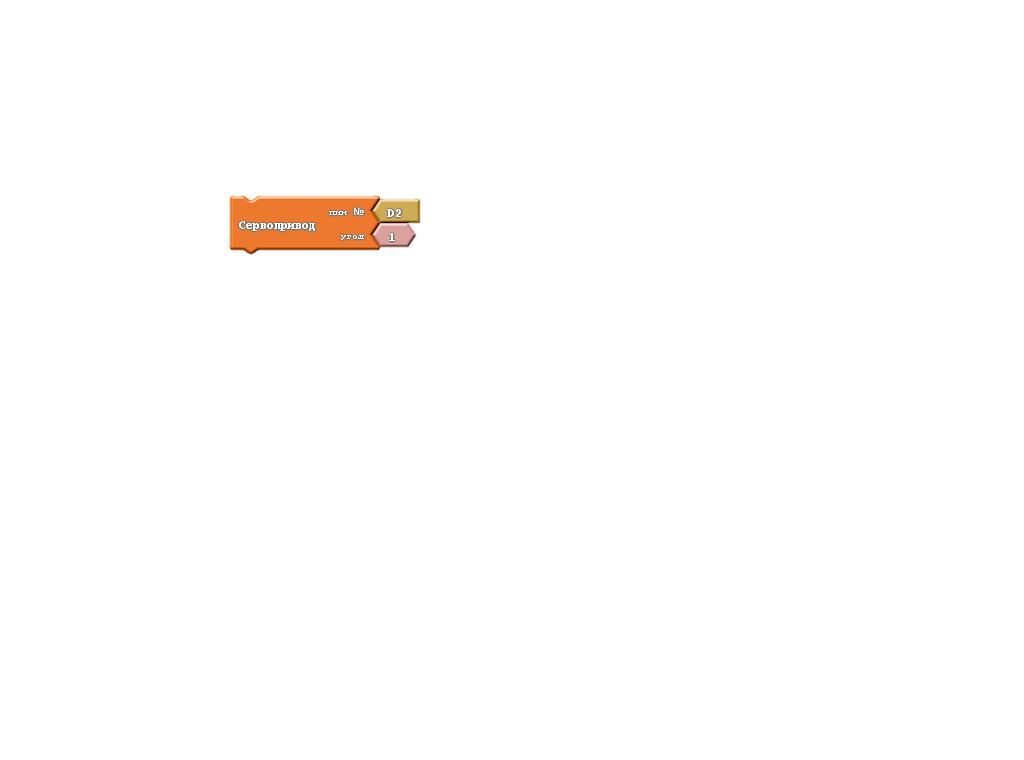
Блоки **Логический порт** и **Аналоговый порт** служат для получения соответственно логических и целочисленных значений с устройств.



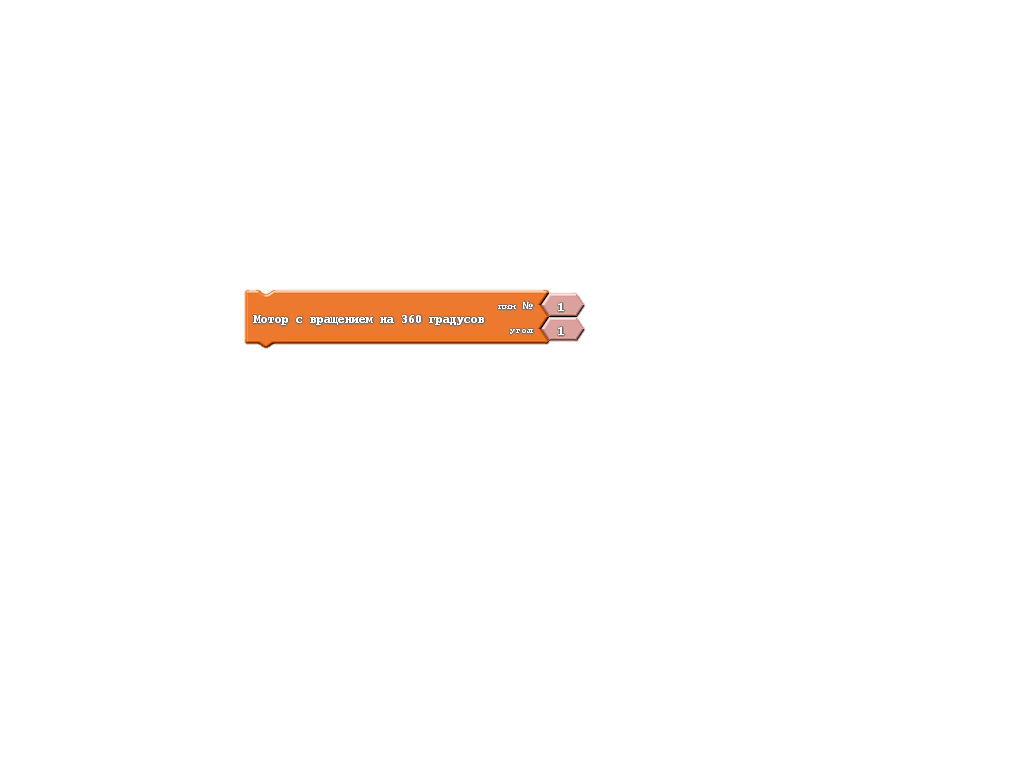
Блок **Переключатель** меняет логическое значение на устройстве на противоположное.



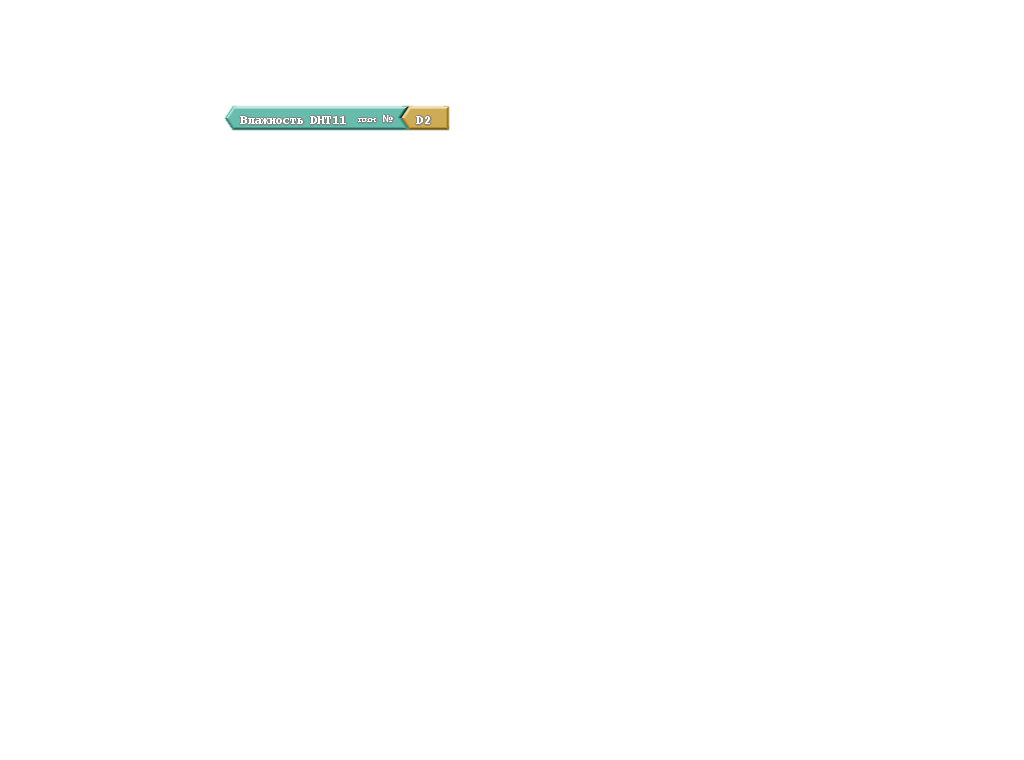
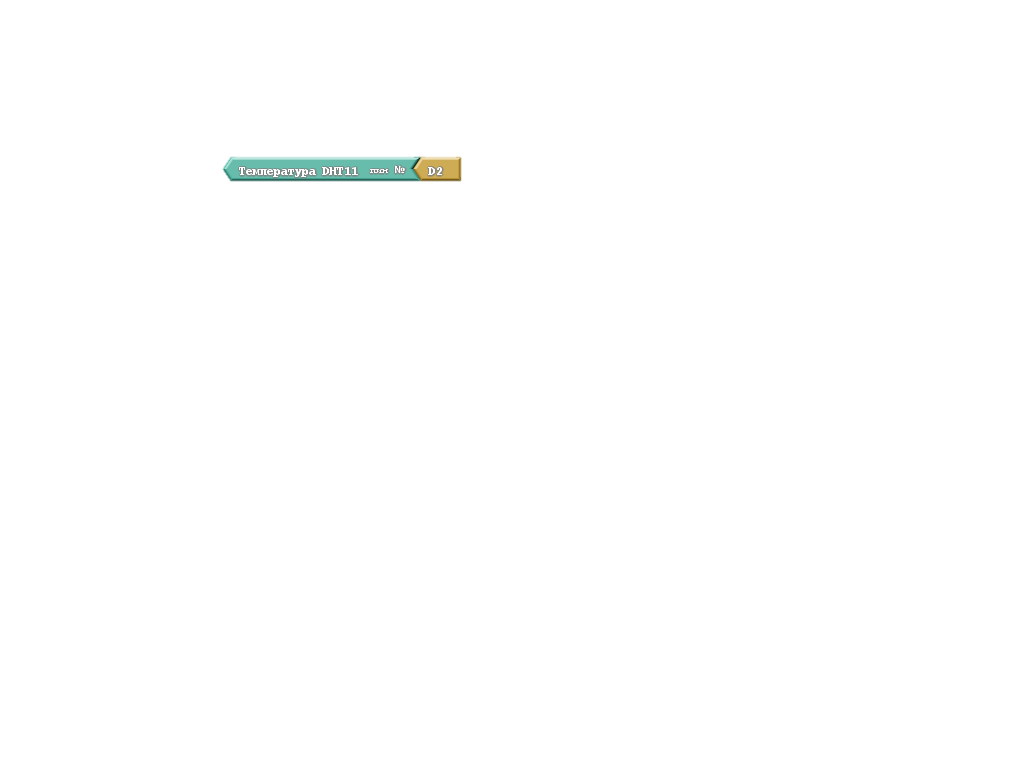
С помощью блока **Сервопривод** можно задать угол поворота вала сервопривода.



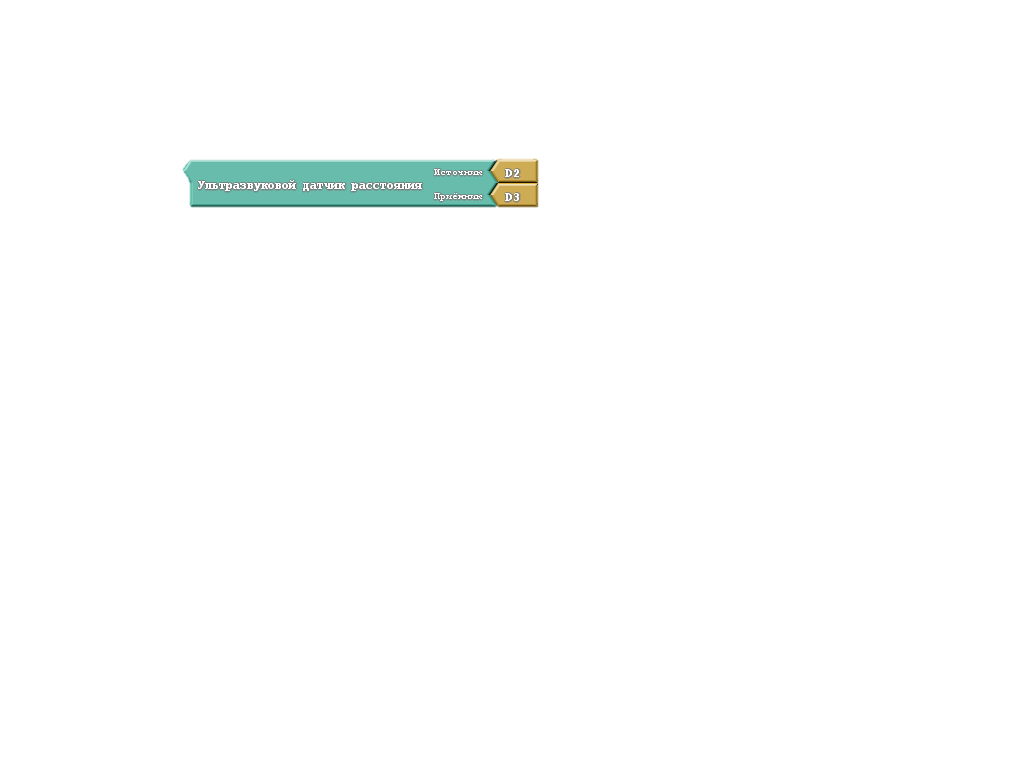
С помощью блока **Мотор с вращением на 360 градусов** можно задать скорость вращения вала мотора.



Блоки **Температура DHT11** и **Влажность DHT11** позволяют получить значения температуры и влажности с соответствующих датчиков.



Блок **Ультразвуковой датчик расстояния** служит для считывания расстояния с соответствующего датчика. Здесь на входе нужно указать два пина (к первому подключен источник ультразвукового импульса, ко второму — сенсор).



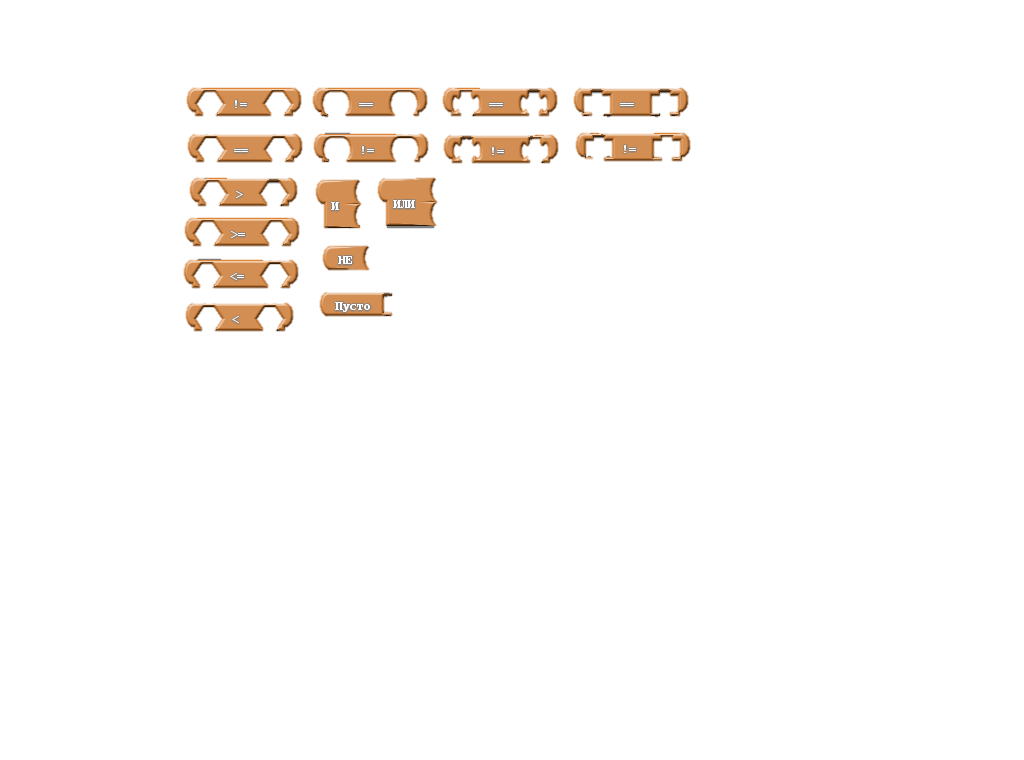
Блоки **Звук** с двумя и тремя входами позволяют манипулировать подключенным генератором звуковых частот. Блок с двумя входами будет передавать значение частоты звука на устройство, пока не встретится блок **Стоп звук**. Блок с тремя входами будет передавать значение указанное количество времени (третий вход «миллисекунд»), но также завершит работу, если встретится блок **Стоп звук**.



**Проверки.** Этот набор включает в себя блоки для сравнения чисел, строк, символов и логических выражений. Все блоки данного набора возвращают логическое значение.

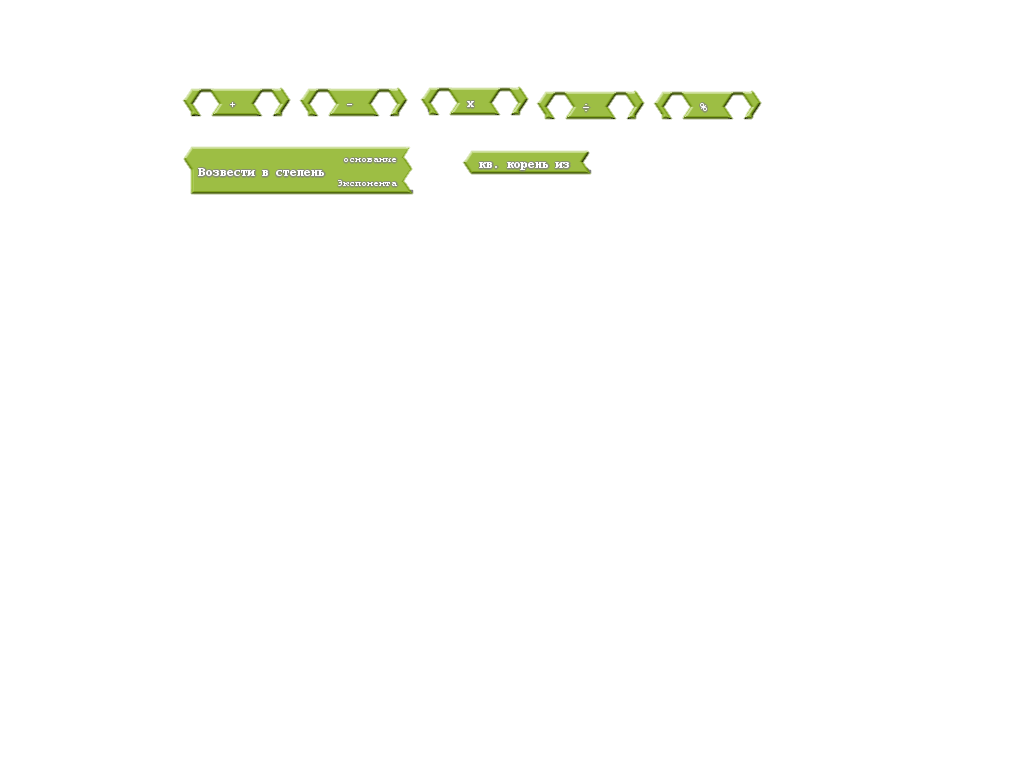
Для каждого из перечисленных типов данных есть блоки «==» и «!=» которые возвращают значение ИСТИНА, если значения на входах равны или не равны соответственно. Также для сравнения чисел есть блоки «>», «>=», «<», «<=». Для изменения значения логического выражения на противоположное существует блок **НЕ**. Для составления сложных логических выражений предназначены блоки **И** и **ИЛИ**. Первый вернет значение ИСТИНА только, если оба логических выражения на его входах вернут значение ИСТИНА. Второй блок вернет значение ИСТИНА в случае, если хотя бы одно из логических выражений на его входах возвращает значение ИСТИНА.

Также в данном наборе есть блок **Пусто**, который возвращает значение ИСТИНА, когда ему на вход присоединяется пустая строка.

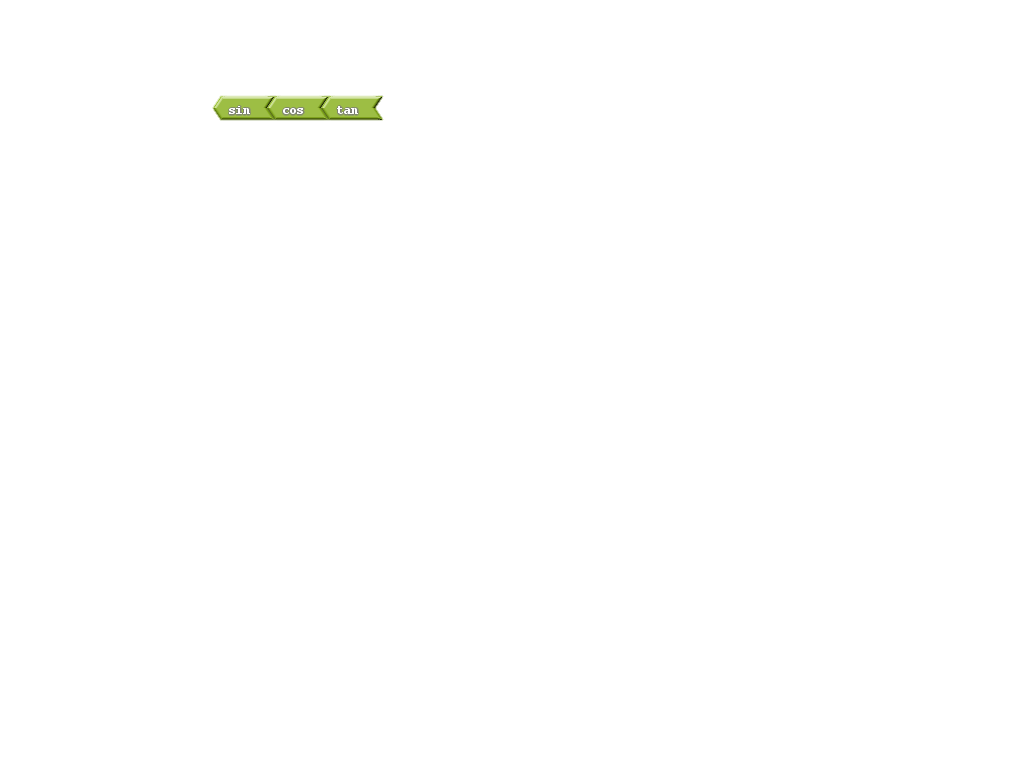


**Математические операторы.** Здесь собраны блоки, с помощью которых производятся некоторые операции над числами (арифметические, тригонометрические и другие). Все они возвращают численные значения.

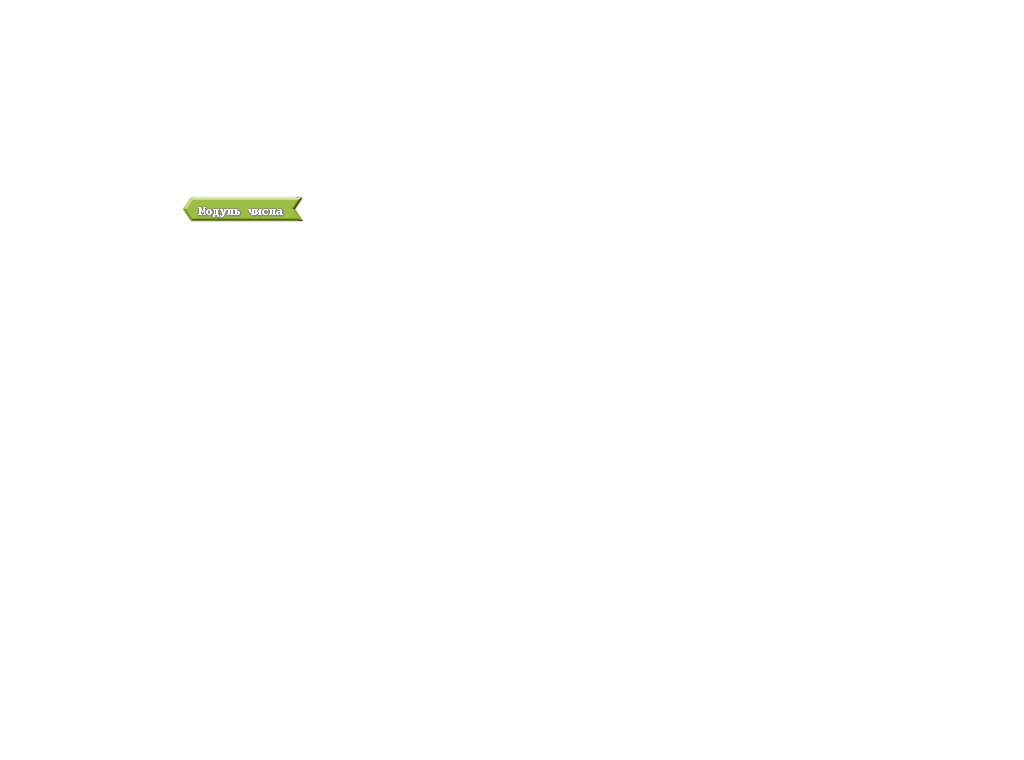
**Арифметические операции.** Сложение, вычитание, умножение, деление и операция взятия остатка от деления осуществляются с помощью блоков «+», «-», «\*», «÷» и «%» соответственно. Блок **Возвести в степень** возвращает возведенное в степень (заданную на втором входе блока) значение, которое подано на первый вход блока. Для операции взятия квадратного корня числа существует блок **кв. корень из числа**.



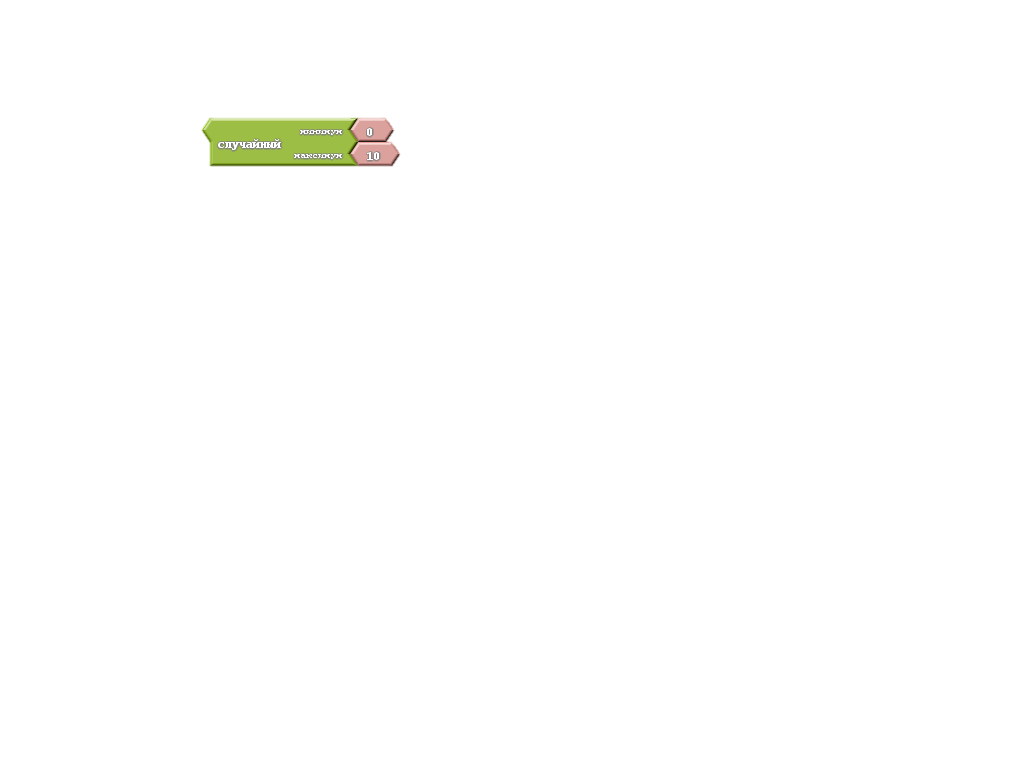
**Тригонометрические функции.** Блоки **sin**, **cos** и **tan** служат для получения синуса, косинуса и тангенса угла (поданного на вход в радианах) соответственно.



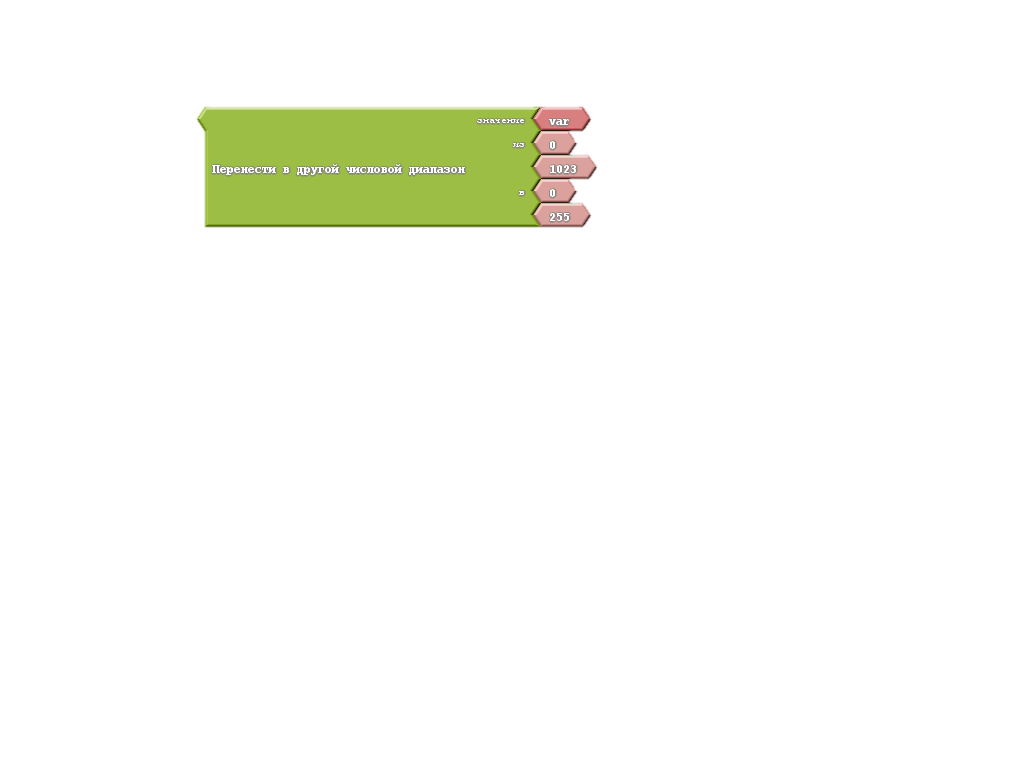
Чтобы получить модуль (абсолютную величину) числа, нужно использовать блок **Модуль числа**.



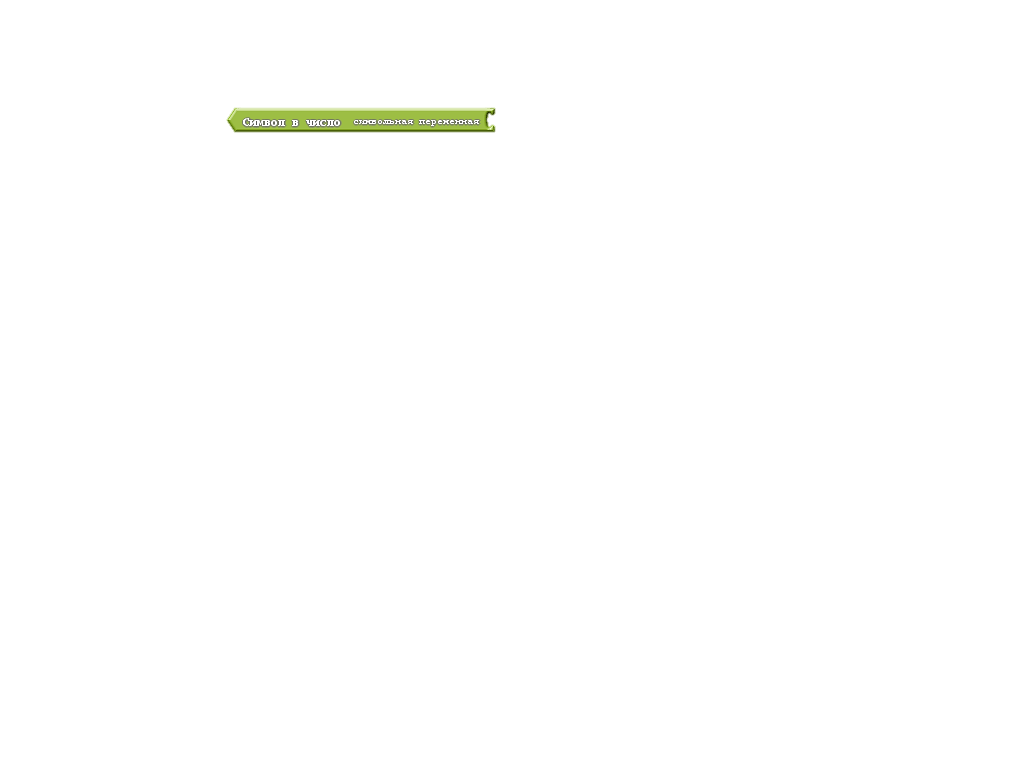
Блок **случайный** возвращает случайное число из указанного диапазона. Нижняя и верхняя границы диапазона задаются посредством первого и второго входа блока соответственно.



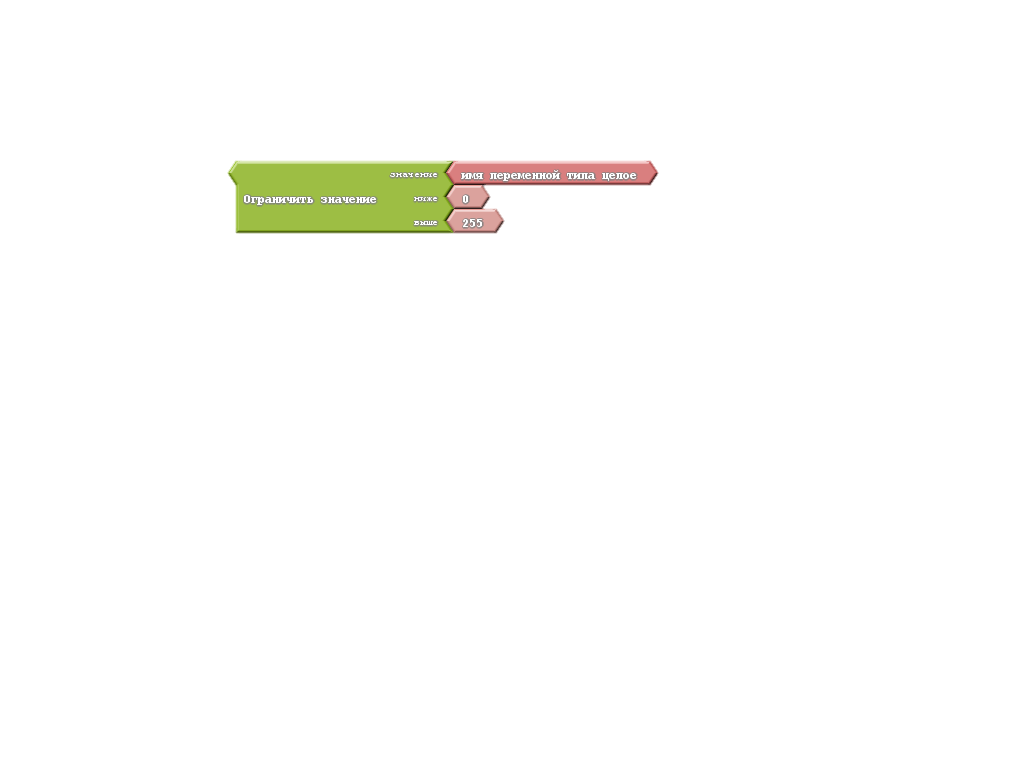
Блок **Перенести в другой числовой диапазон** принимает на вход числовое значение в одном диапазоне (исходном), переносит его в другой (выходной) и возвращает. Исходный диапазон задается посредством двух входов **из**. Выходной диапазон задается посредством двух входов **в**.



Блок **Символ в число** позволяет преобразовать символьное значение, которое подается на вход блоку, в численное (длинное целое). При этом символ должен иметь код цифры.

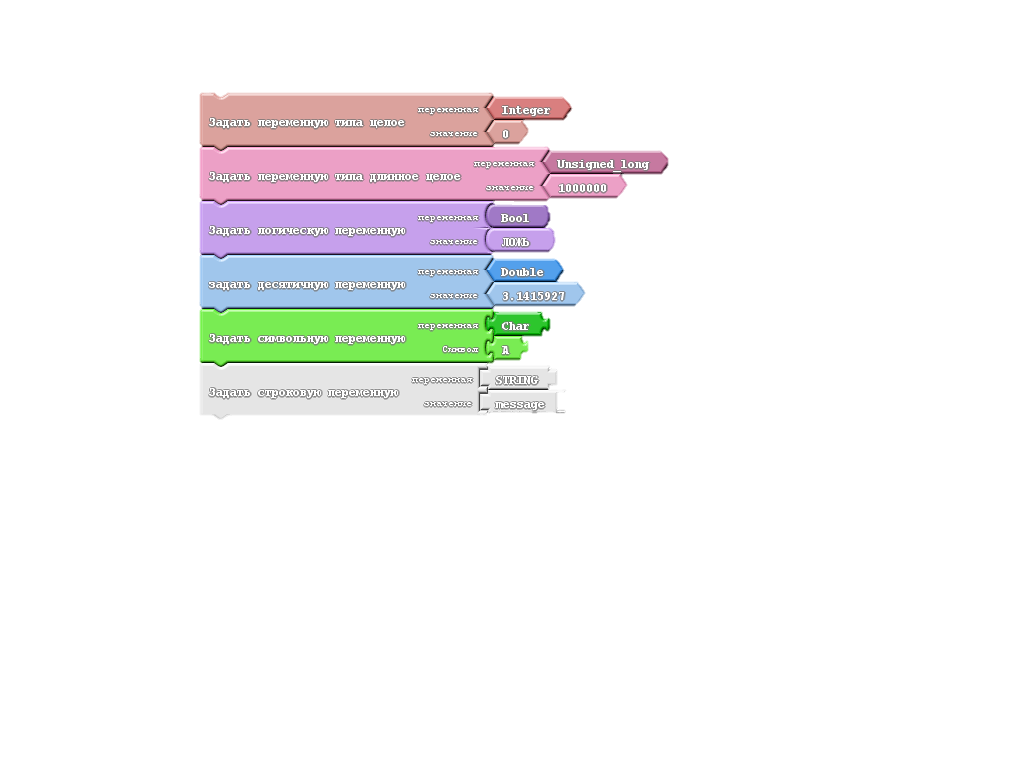


Для того, чтобы предотвратить выход численного значения из определенного диапазона, существует блок **Ограничить значение.** Если значение, которое подается на первый вход, меньше нижней границы диапазона, то оно становится равным нижней границе. А если оно больше верхней границы, то становится равным верхней границе.

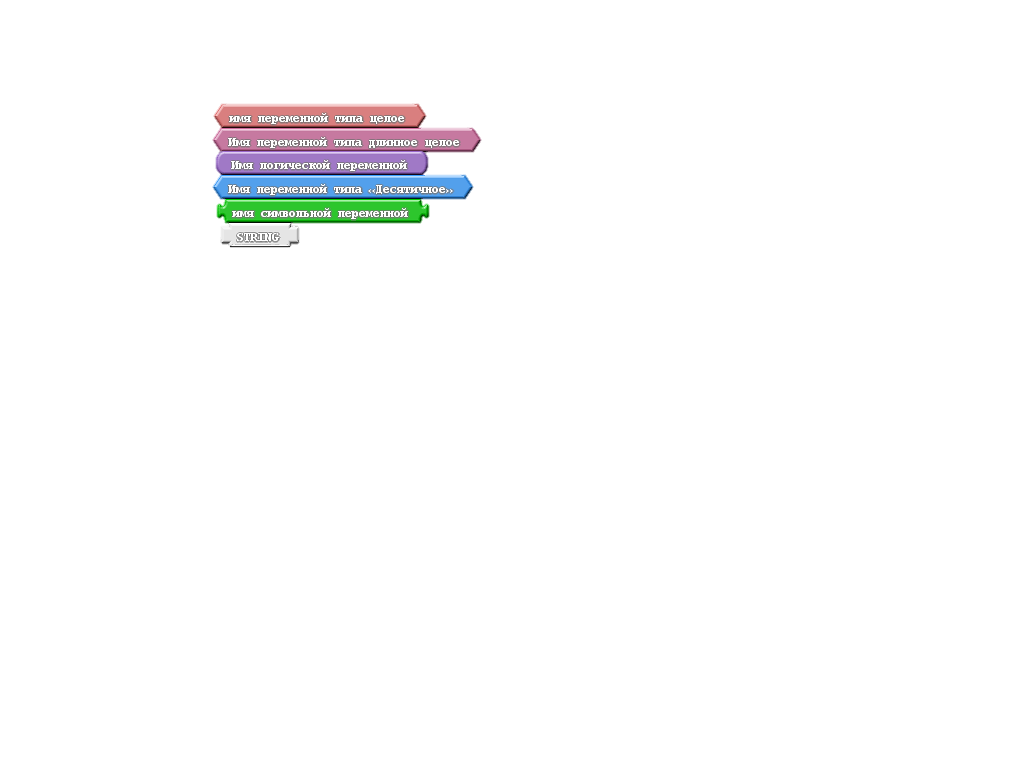


**Переменные/константы.** Набор состоит из блоков для инициализации, использования и задания значений переменных, а также блоков для работы с константами. В Ardublock вы можете оперировать следующими **типами данных**: строка, целое число, длинное целое число, десятичное число, символ, логическое значение и массив чисел.

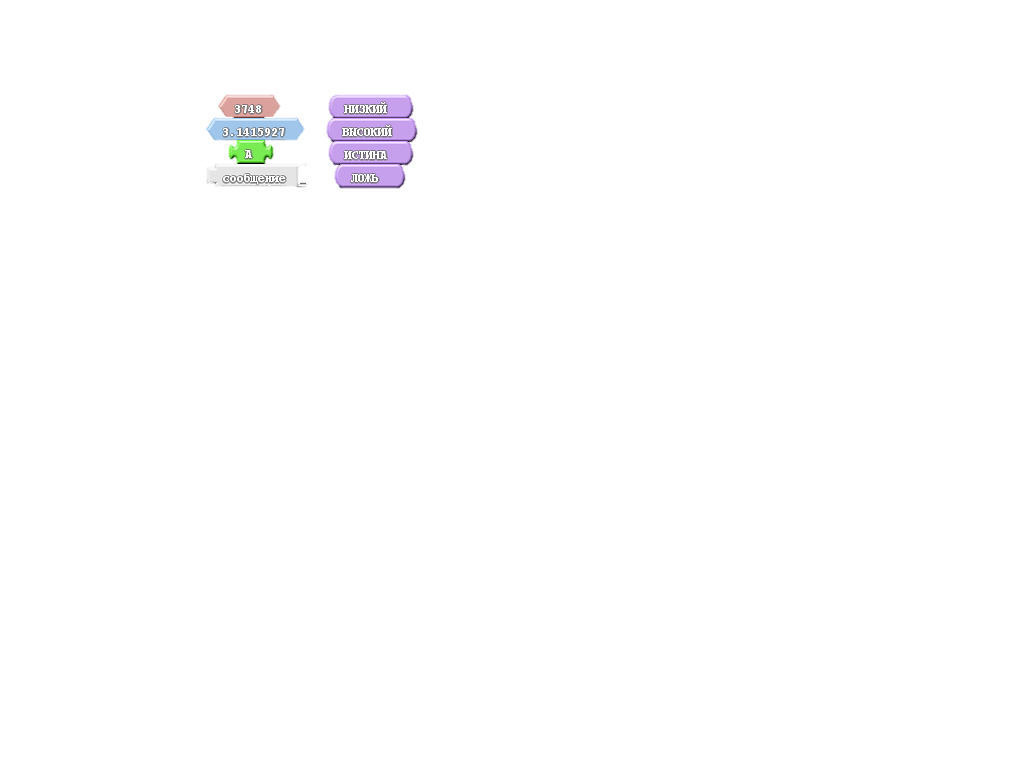
Блоки типа **Задать переменную** позволяют создать или изменить значение существующей переменной. Переменная может быть создана в любом месте программы. Переменная всегда будет **глобальной** (её можно будет использовать в любом месте программы и в любой функции). **Примечание:** *в большинстве случаев переменным следует задавать начальное значение, поэтому их создание лучше производить в части* ***Установка*** *блока* ***Программа***.



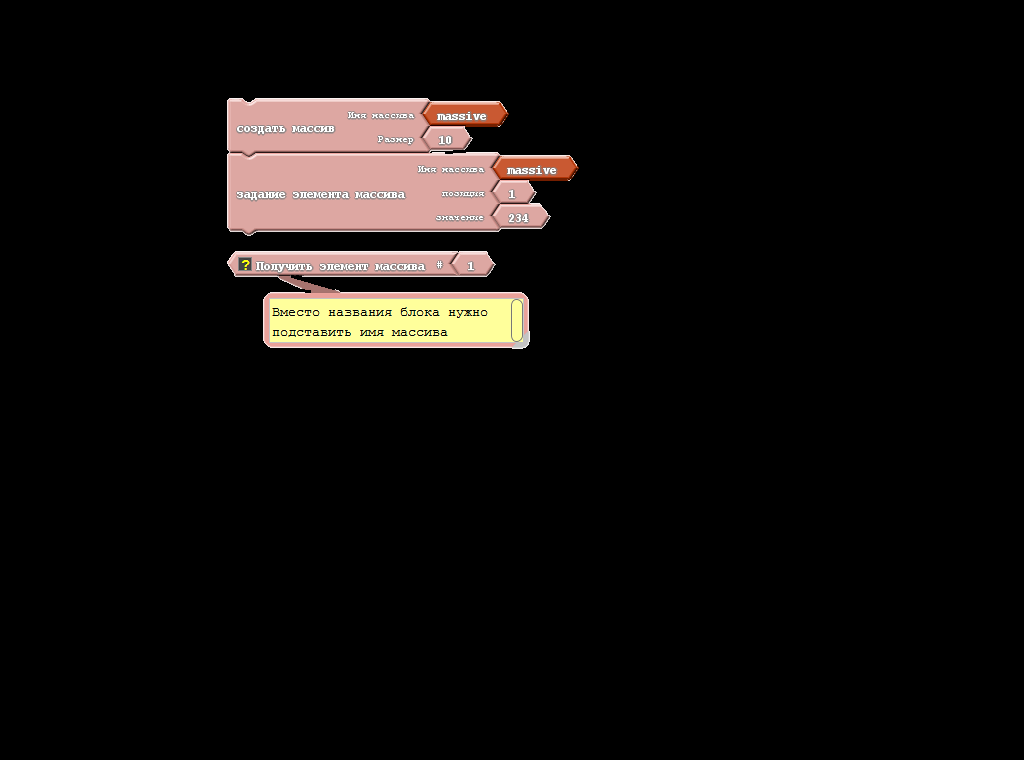
Следующие несколько блоков позволяют использовать переменные (изменять их значение, подставлять в различные выражения и подавать на вход другим блокам). Они имеют редактируемое имя блока (вместо названия блока нужно ввести имя переменой).



Далее представлены блоки **констант**. Их также можно использовать во всевозможных выражениях и подавать на вход другим блокам.

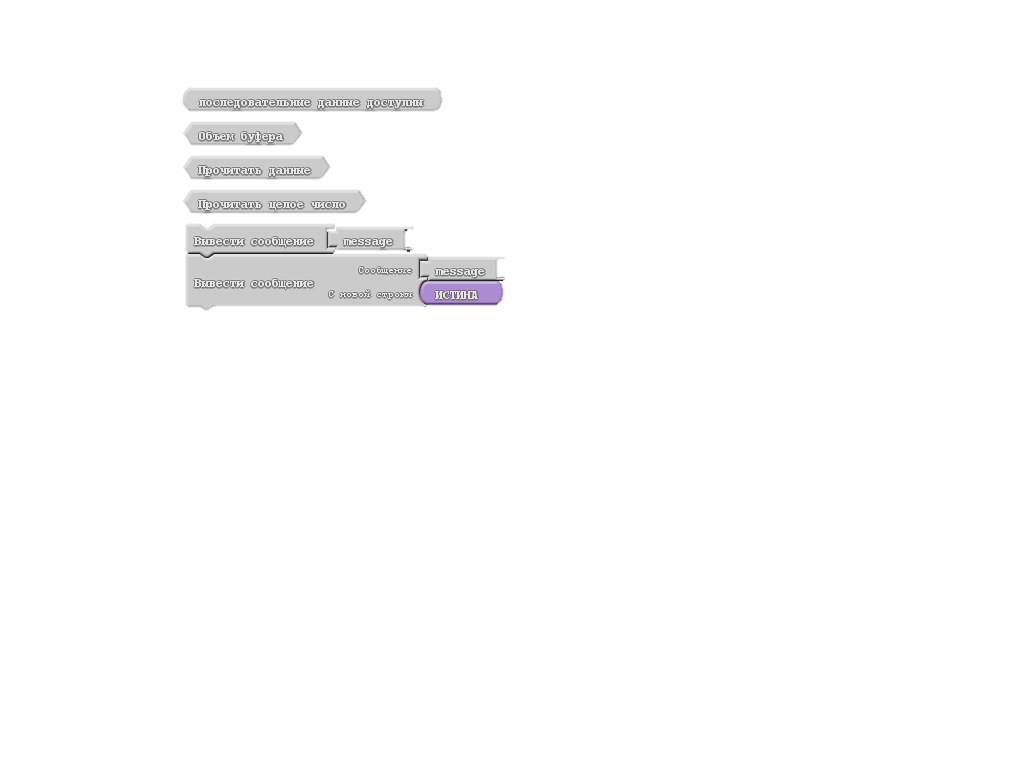


И, наконец, блоки для работы с массивами. Создавать массив следует в части **Установка** блока **Программа**. Задавать начальные значения элементов массива (если необходимо) следует там же. Для изменения значения элемента массива служит блок **Задание элемента массива**. На втором и третьем входах у него нужно указать номер элемента и значение, которое нужно задать. Для получения значения элемента массива существует блок **Получить элемент массива**. На вход ему подаётся номер элемента. Имя массива указывается путем редактирования названия блока.



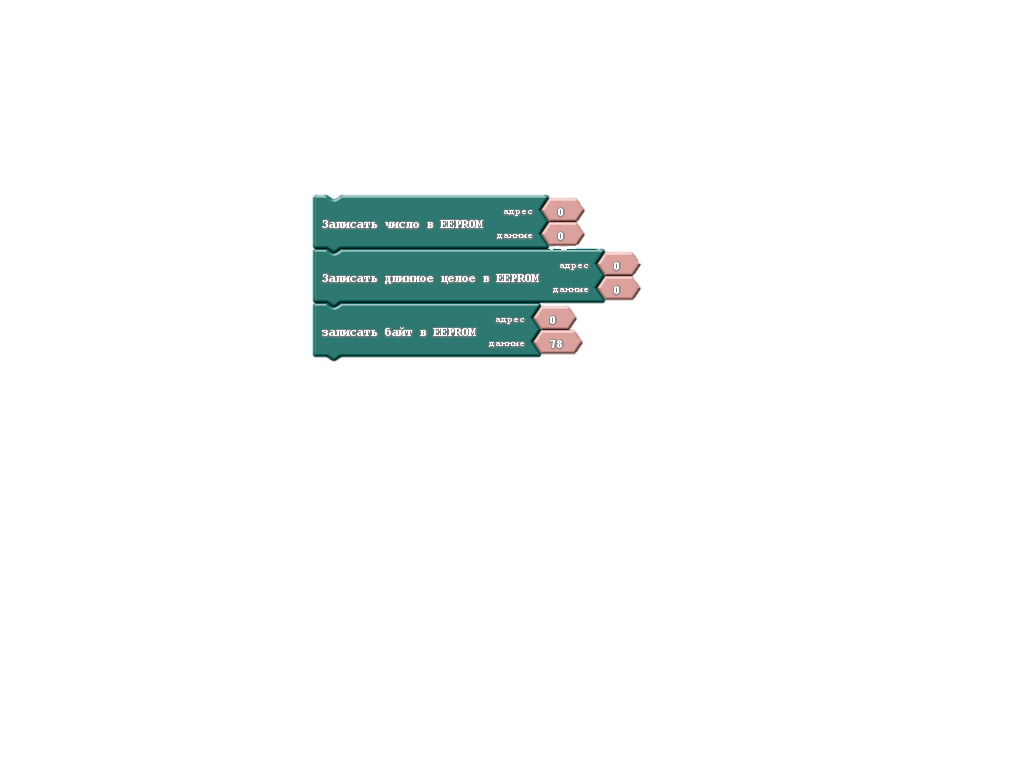
**Коммуникации.** Набор содержит блоки, отвечающие за передачу данных через последовательный интерфейс (или Serial). Все платы Arduino имеют хотя бы один последовательный порт (его называют UART или USART). Для передачи данных через Serial используются порты ввода/вывода (0 и 1). Необходимо помнить, что, если в программе используется передача данных через Serial, то нельзя использовать 0 и 1 порты для других целей. Передача данных осуществляется посредством записи в буфер и чтения из него. Размер буфера может достигать до 64 байт.

Блок **Последовательные данные доступны** вернет значение ИСТИНА, если буфер последовательного порта не пуст. Блок **Объем буфера** вернет объем информации (в байтах), содержащейся в буфере. Для чтения данных из буфера существует два блока: **Прочитать данные** и **Прочитать целое число**. Первый считывает один байт из буфера и возвращает его код. Второй вернет первое найденное в буфере целое число (вернет 0, если за секунду не найдет ни одного числа). Для отправки данных (или сообщения) также существует два блока. Оба они называются **Вывести сообщение** (отличие состоит в том, что один из блоков позволяет вывести сообщение с новой строки).



**Хранилище**. Этот набор блоков служит для работы с EEPROM и SD картами.

**EEPROM.** Для записи в EEPROM существует три блока (см. рисунок). С их помощью можно записать целое (от -32.768 до 32.767), длинное целое (от 0 до 4.294.976.295) или байт (от 0 до 255). На первый вход подается адрес, с которого нужно начать запись. Байт занимает одну ячейку памяти, целое число – две, а длинное целое – четыре. Это означает, что целое число при записи займет указанный и следующий за ним адрес (а длинное целое, соответственно, указанный и три, следующих за ним).

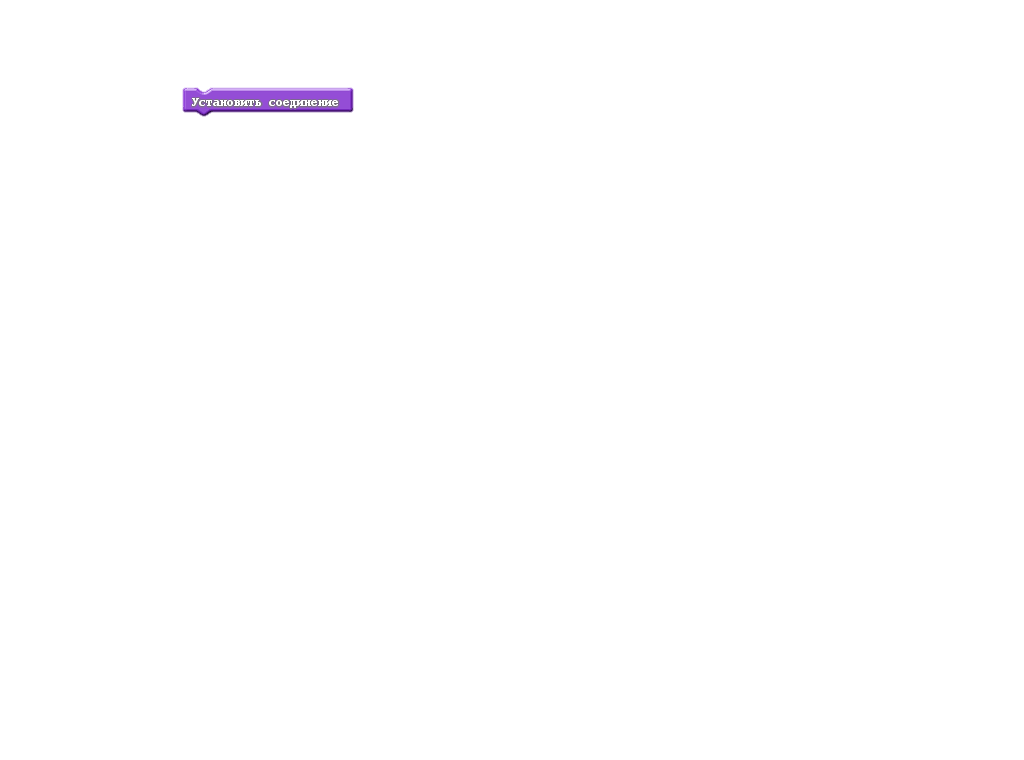


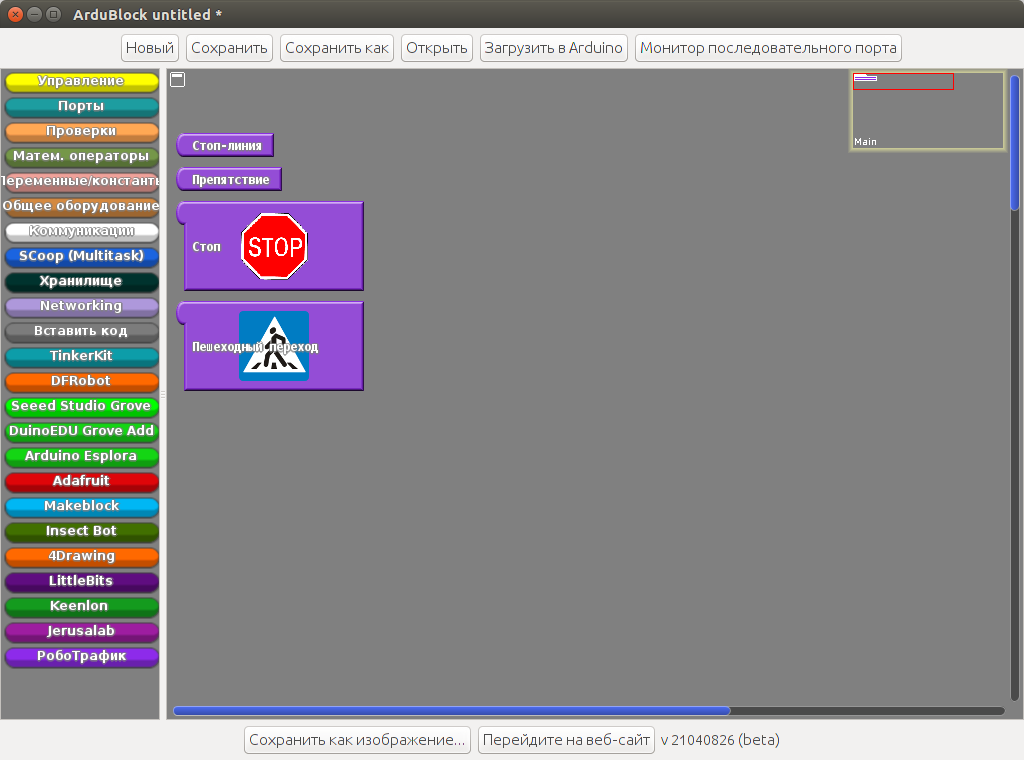
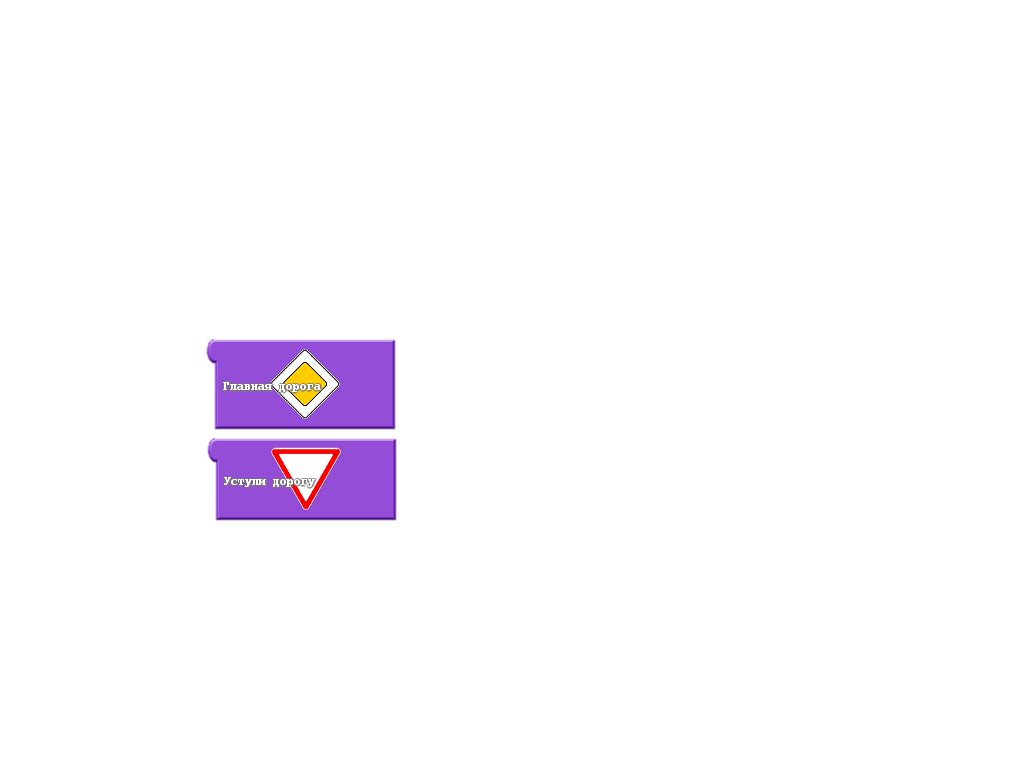
Для чтения данных из EEPROM также есть три блока. Прочитать, аналогично, можно целое, длинное целое и байт. При чтении стоит помнить о правилах записи разных типов данных и о том, сколько каждый тип данных занимает ячеек памяти.



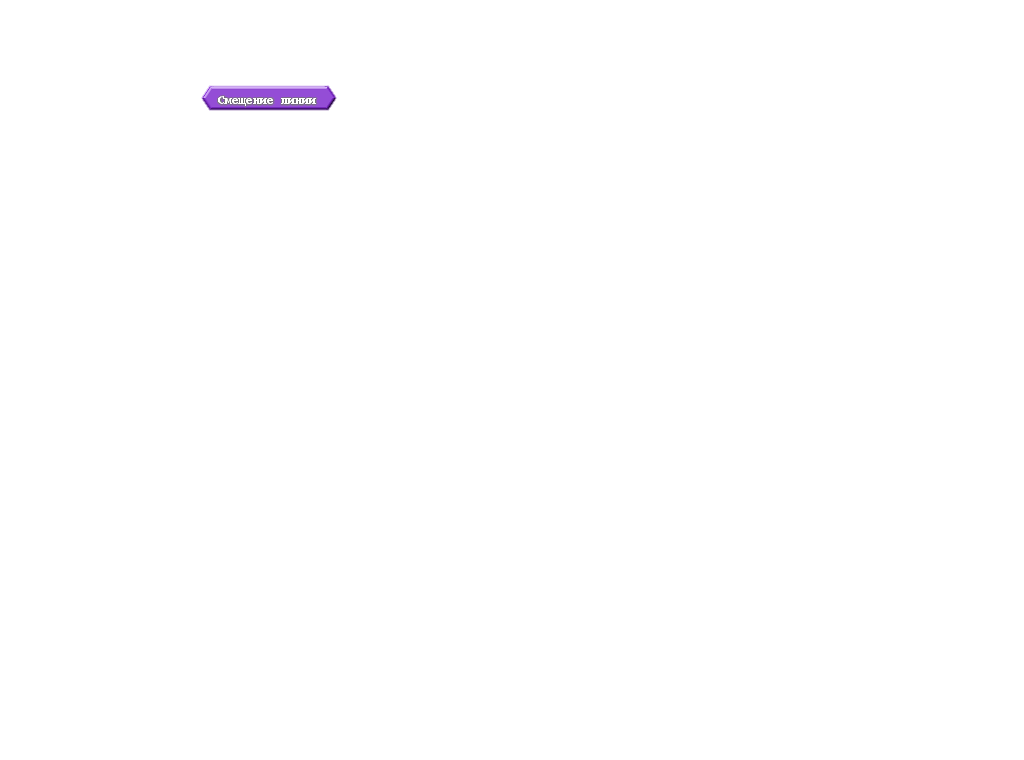
**РобоТрафик.** Набор содержит блоки для работы с робомобилем компании Elecard.

Блок **Установить соединение** является обязательным для работы с робомобилем. Он позволяет начать получение данных с камеры автомобиля (для распознавания знаков, линии движения и т.д.).

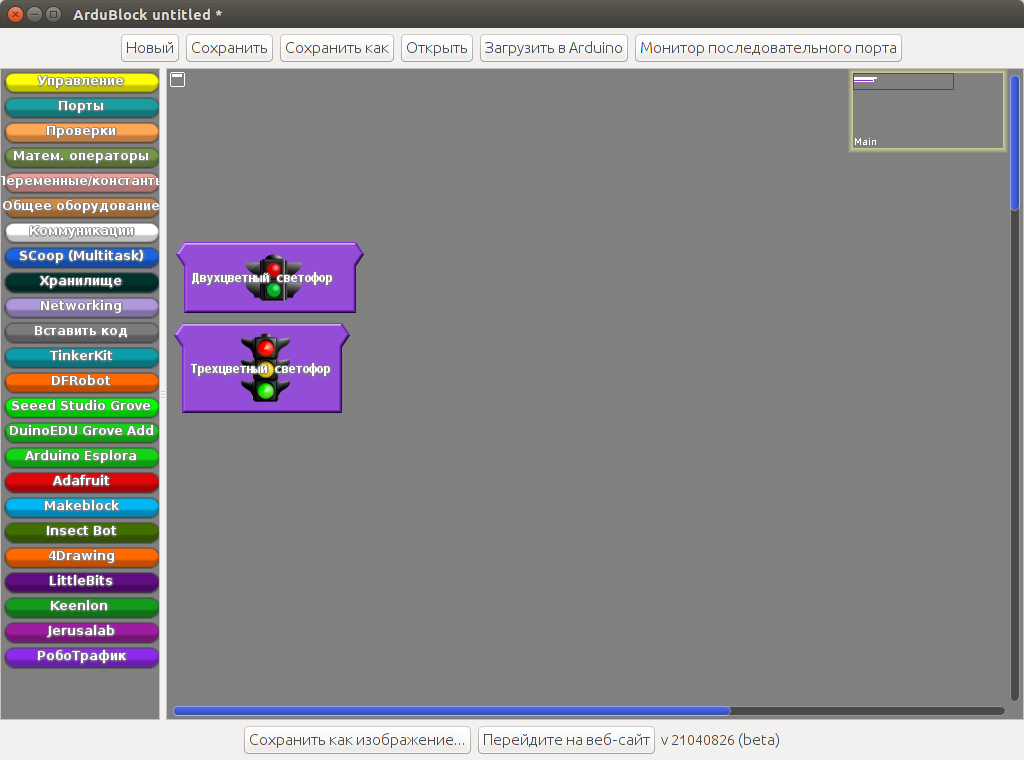


**Блоки, возвращающие логическое значение.** Такие блоки служат для реагирования на дорожные знаки, дорожную разметку, препятствия и другое. Они используются в логических выражениях, а также в качестве входов для других блоков (условных, например). По их названиям можно определить, какой блок за что отвечает. Стоит пояснить лишь предназначение блока **Стоп-линия.** Он нужен для обнаружения так называемой стоп-линии (линия, на которой должны оказаться передние колеса автомобиля при остановке перед светофором). 

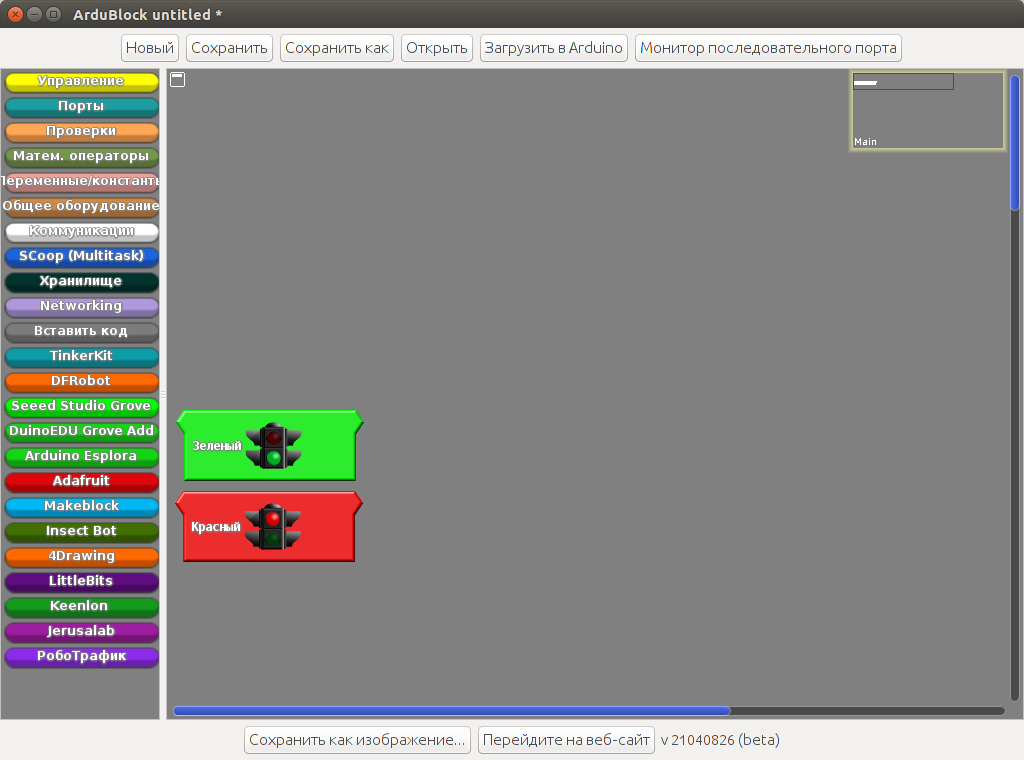
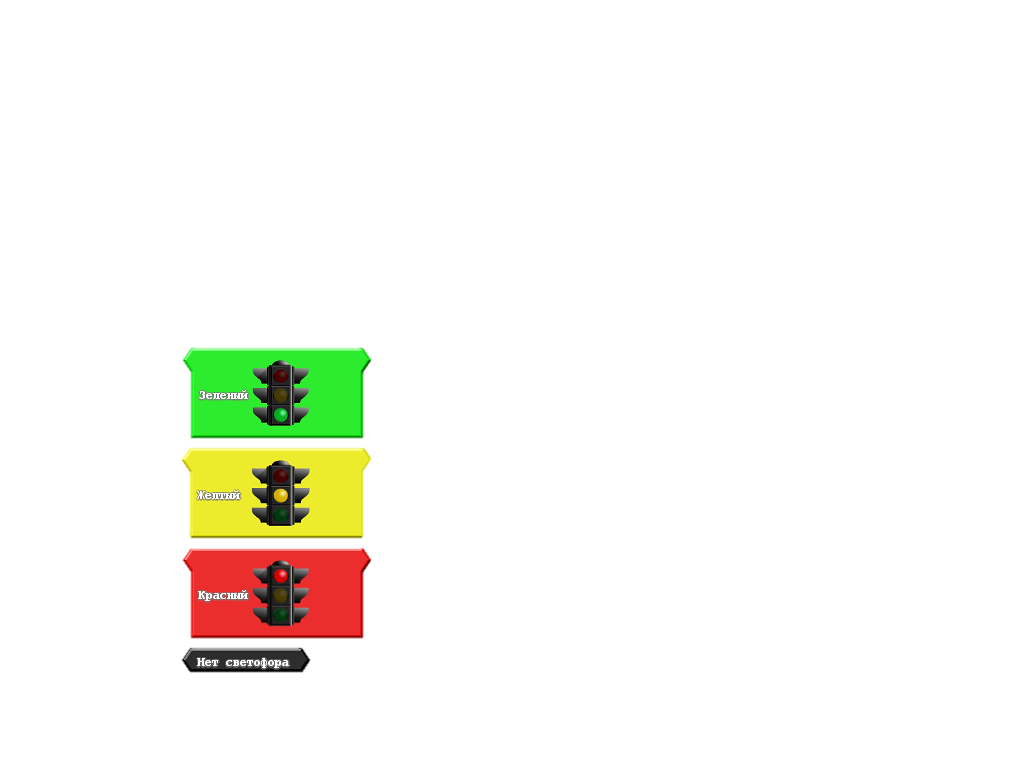
Блок **Смещение линии** возвращает значение от -100 до 100, которое соответствует положению черной линии относительно продольной оси непосредственно перед автомобилем, или -110, если линия не будет найдена.



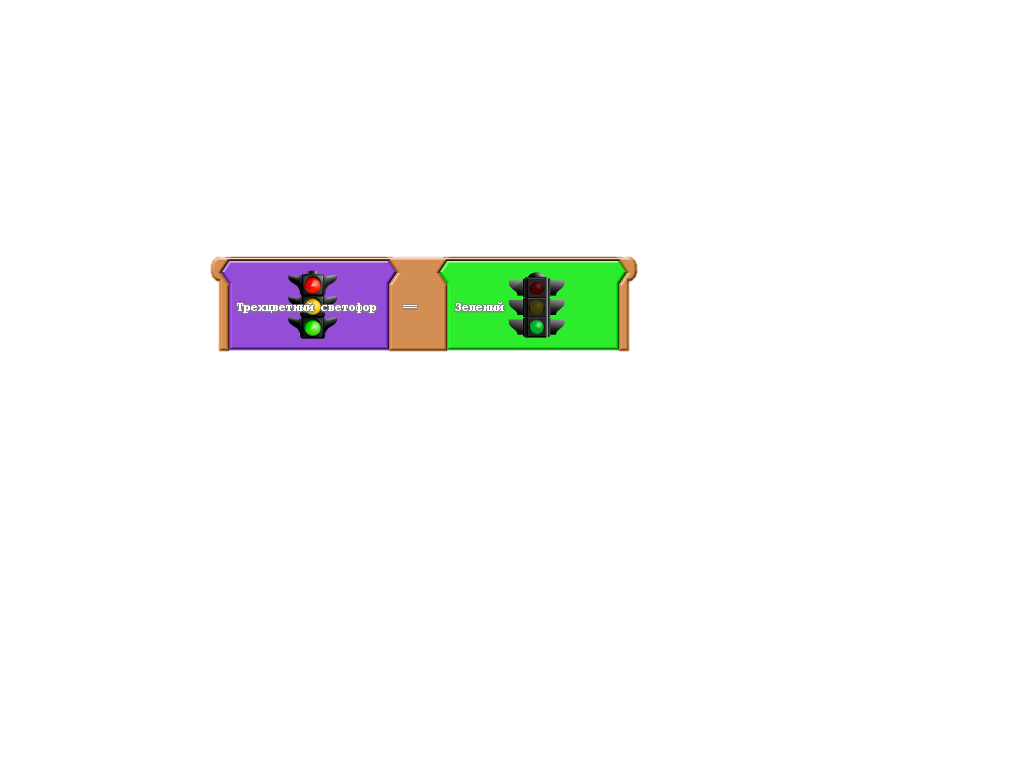
**Блоки светофоров.** На трассе могут располагаться два вида светофоров: двухцветные и трехцветные. Блоки **Двухцветный светофор** и **Трёхцветный светофор** позволяют получить текущее состояние того или иного светофора в кадре (зеленый, желтый, красный или нет светофора).



Блоки **Зеленый**, **Желтый**, **Красный** и **Нет светофора** используются с блоками светофоров в операциях сравнения.



Например, для того, чтобы узнать, видит ли камера автомобиля зеленый цвет на светофоре в данный момент, нужно составить следующее логическое выражение.



Блок **Подключить энкодер** используется для инициализации чтения показаний с энкодера. На вход ему подается диаметр колеса робомобиля в мм. Этот блок является **обязательным,** если в программе используется блок **Прочитать скорость с энкодера,** который возвращает скорость движения робомобиля в см/с. Полученные с **энкодера** показатели скорости движения робомобиля могут использоваться как для статистики, так и для регулирования скорости движения. В примере скорость движения устанавливается не более 10 см/с. **speed** – это глобальная переменная, в которую где-то в программе записывается скорость вращения мотора в условных единицах.

