

Управление образования администрации

Ленинск-Кузнецкого городского округа

Муниципальное бюджетное образовательное учреждение

дополнительного образования

«Дворец творчества детей и учащейся молодежи»

**Arduino.**

**Электроника без пайки**

Методическая разработка

2022

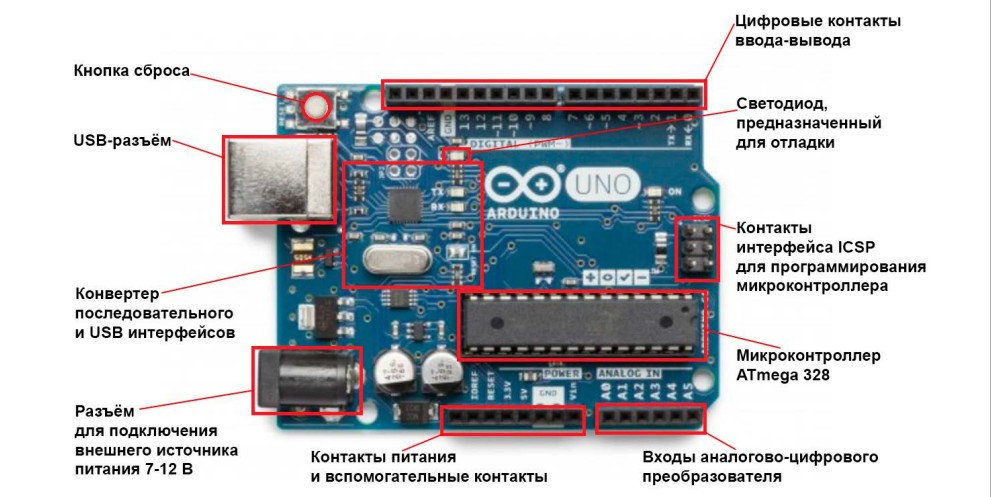
«Arduino. Электроника без пайки»,

методическая разработка

Составитель: Иванов А.В. - педагог дополнительного образования МБОУ ДО «Дворец творчества»

Ленинск-Кузнецкий, 2022. - 16 с.

Методическая разработка предназначена для учащихся младшего и среднего школьного возраста. Знакомит детей c возможностями платы Arduino и особенностями среды программирования. Содержит практическое руководство по базовым проектам.

**Комплектовка платы** **Arduino**

**Платы Arduino**

Модель Arduino Uno является последней версией оригинальной платы Arduino. Это самая распространенная модель Arduino, и обычно, когда кто-то говорит, что использует Arduino, подразумевается именно эта модель. Все остальные модели плат Arduino сконструированы для удовлетворения особых потребностей, таких как большая величина тока на входных и выходных контактах, более высокая производительность, меньший размер, возможность вшивания в элементы одежды и подключения телефонов на Android, простота подключения к беспроводным сетям и т.д. Независимо от конструктивных особенностей, все платы программируются из Arduino IDE, немного различаясь лишь некоторыми особенностями программного обеспечения, которое они могут использовать. Поэтому, узнав, как использовать одну плату Arduino, вы сможете применять полученные знания для работы с другими моделями.

**Язык программирования**

Многие ошибочно полагают, что платы Arduino имеют собственный язык программирования. В действительности программы для них пишутся на языке с простым названием C. Этот язык существует с самых первых дней развития вычислительной техники. А вот что действительно привносит Arduino - это набор простых в использовании команд, написанных на C, которые вы можете использовать в своих программах. Пуристы могут заметить, что в Arduino используется C++, объектно-ориентированное расширение языка C. Строго говоря, они правы, однако наличие всего 1–2 Кбайт памяти обычно означает, что использование объектно-ориентированных приемов при программировании для Arduino не самая лучшая идея, за исключением особых ситуаций, и фактически программы пишутся на C.

**Переменные**

Переменные помогают дать имена числам. В действительности их возможности намного шире, но пока мы будем использовать их только для этой цели. При объявлении переменной в языке C необходимо указать ее тип. Например, если нужна переменная, хранящая целое число, ее следует объявить с типом int (сокращенно от integer — целое со знаком).

**Циклы**

По аналогии с выполнением некоторых операций по условию иногда в скетчах возникает необходимость повторять операции снова и снова. Конечно, вы и так получаете это поведение, помещая команды в функцию loop. Но иногда требуется выполнить команды определенное число раз. Добиться этого можно с помощью команды for, позволяющей использовать переменную-счетчик. Например, напишем скетч, который мигает светодиодом 10 раз.

int ledPin = 13;

int delayPeriod = 200;

void setup() {

pinMode(ledPin, OUTPUT); }

void loop() {

for (int i = 0; i < 10; i++) {

digitalWrite(ledPin, HIGH);

delay(delayPeriod);

digitalWrite(ledPin, LOW);

delay(delayPeriod); }

}

**Функции**

Функция — это способ группировки программных команд в удобный для использования блок. Функции помогают поделить скетч на управляемые и простые в использовании фрагменты.

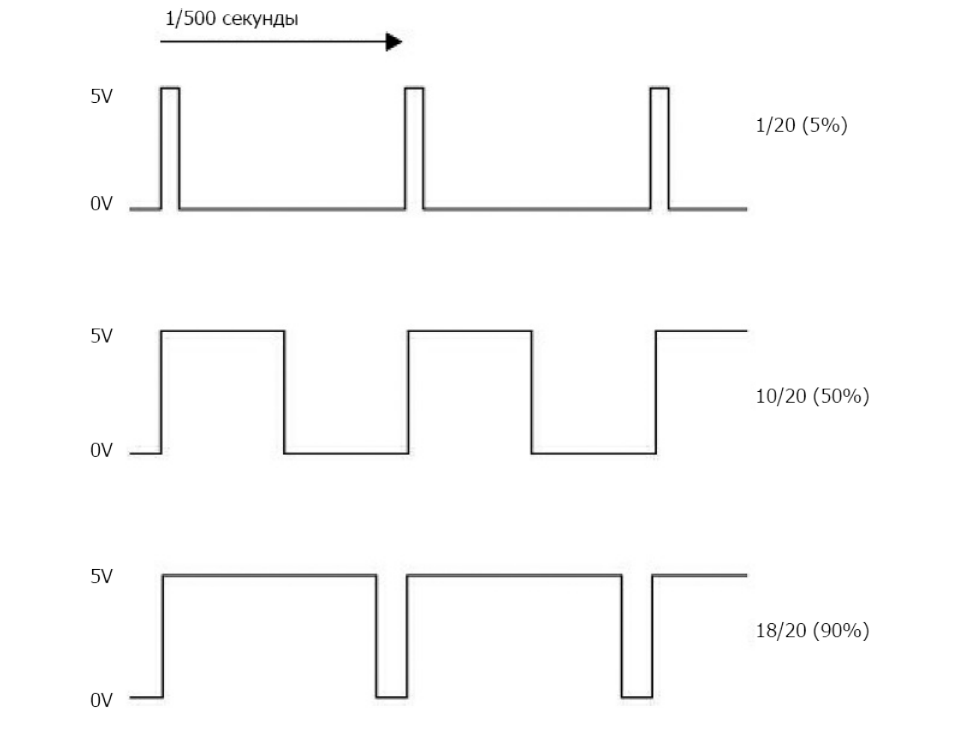
**Массивы и строки**

Массивы предназначены для хранения списков значений. Переменные, которые нам встречались до сих пор, могли хранить только одно значение, обычно типа int. Массив, напротив, может хранить список значений и позволяет обращаться к отдельным значениям по их позициям в списке. В C, как и в большинстве других языков программирования, нумерация позиций в массиве начинается с 0, а не с 1. Это означает, что первый элемент фактически является нулевым элементом.

Все сообщения, или строки, как часто называют коллекции символов, имеют тип char\*. Символ звездочки (\*) говорит о том, что это указатель на что-то. Квадратные скобки ([]) в конце объявления переменной указывают, что данная переменная хранит массив данных типа char\*, а не единственное значение char\*.

**Аналоговые выходы**

Плата Arduino Uno не имеет настоящих аналоговых выходов (такие выходы вы найдете на плате Arduino Due), но она имеет несколько выходов с широтно-импульсной модуляцией (Pulse-Width Modulation, PWM). Они имитируют аналоговые выходы, управляя длительностью импульсов.



Чем длиннее положительный импульс, тем выше среднее напряжение на выходе. Так как импульсы следуют с частотой 500 раз в секунду, а большинство устройств, которые вам доведется подключать к выходам PWM, не обладают мгновенной реакцией, возникает эффект изменения напряжения.

**Использование библиотек**

Так как платы Arduino обладают весьма ограниченным объемом памяти, имеет смысл включать в программу, которая в конечном итоге окажется в плате, только тот код, который действительно потребуется. Один из способов добиться этого - использовать библиотеки. В Arduino и вообще в языке C под библиотекой понимается коллекция функций. Например, Arduino IDE включает библиотеку для работы со светодиодным жидкокристаллическим дисплеем. Она занимает примерно 1,5 Кбайт памяти для программ. Нет никакого смысла подключать эту библиотеку, если она не используется, поэтому такие библиотеки подключаются только при необходимости. Подключение выполняется добавлением директив #include в начало скетча. Добавить инструкции include для подключения любых библиотек, поставляемых в составе Arduino IDE, можно с помощью пунктов меню Sketch—>Import Library… (Скетч— >Подключить библиотеку).

В состав Arduino IDE входит большая коллекция официальных библиотек, в том числе:

• EEPROM — для сохранения данных в электрически стираемую программируемую постоянную память (ЭСППЗУ) (Electrically Erasable Programmable Read-Only Memory);

• Ethernet — для реализации сетевых взаимодействий;

• Firmata — стандартная библиотека для реализации взаимодействий через последовательный порт;

• LiquidCrystal — для работы с алфавитно-цифровыми жидкокристаллическими дисплеями;

• SD — для чтения и записи данных на карты флеш-памяти;

• Servo — для управления сервоприводами;

• SPI — для реализации взаимодействий по шине последовательного периферийного интерфейса;

• Software Serial — для реализации взаимодействий по последовательным линиям с использованием любых цифровых выходов;

• Stepper — для управления шаговыми электромоторами;

• WiFi — для доступа к беспроводной сети WiFi;

• Wire — для реализации взаимодействий с периферией по протоколу I2C.

Некоторые библиотеки предназначены для конкретных моделей плат Arduino:

• Keyboard — позволяет платам Arduino имитировать USB-клавиатуру (Leonardo, Due и Micro);

• Mouse — позволяет платам Arduino имитировать USB-мышь (Leonardo, Due и Micro);

• Audio — утилиты для проигрывания звука (только Due);

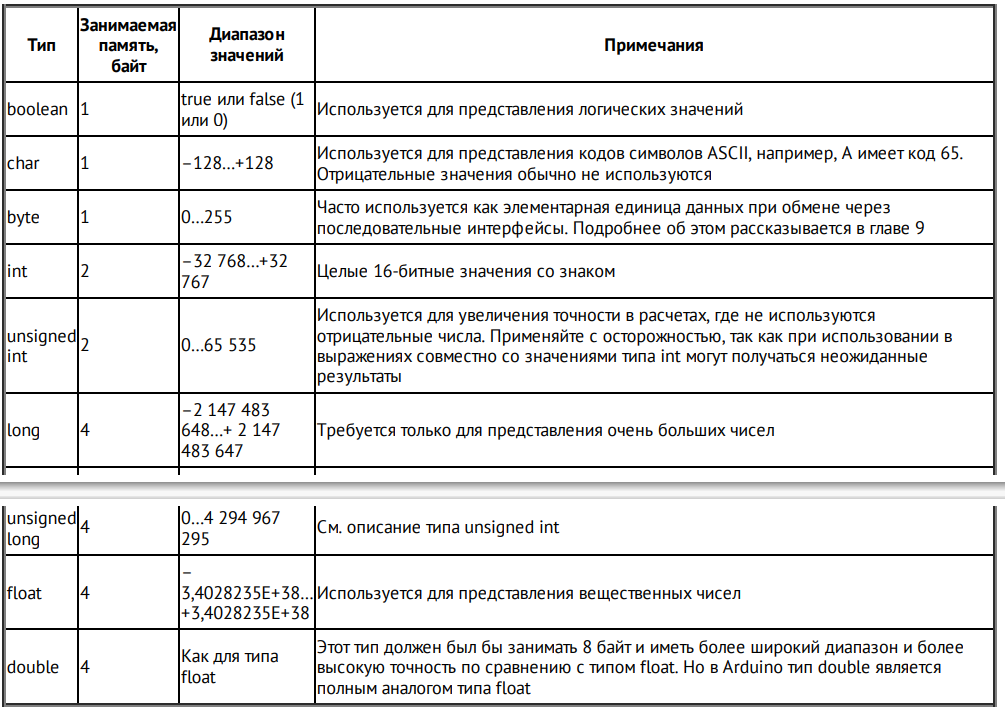
• Scheduler — для управления выполнением нескольких потоков (только Due);

• USBHost — для подключения USB-периферии (только Due).

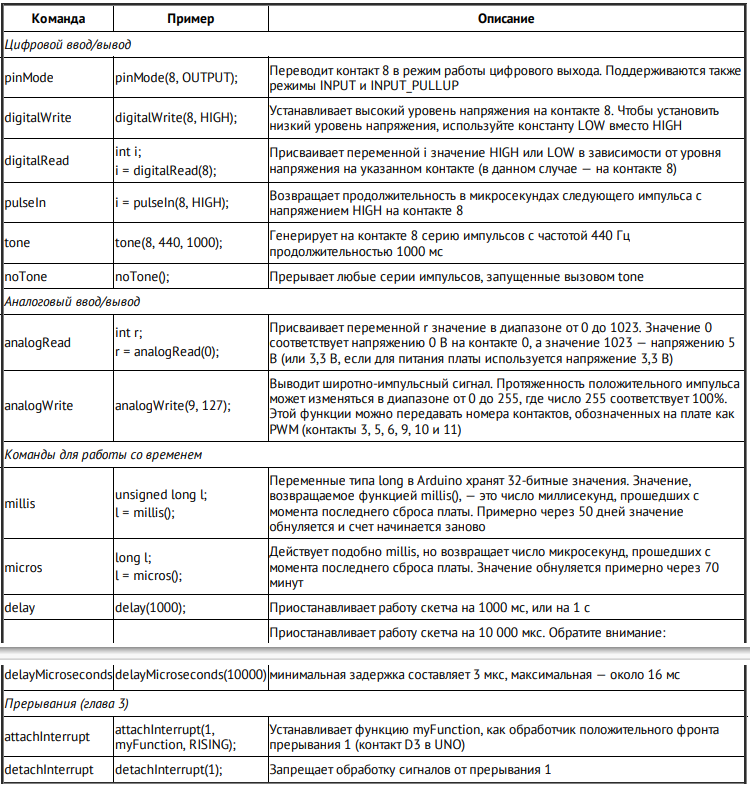
**Типы данных в Arduino**

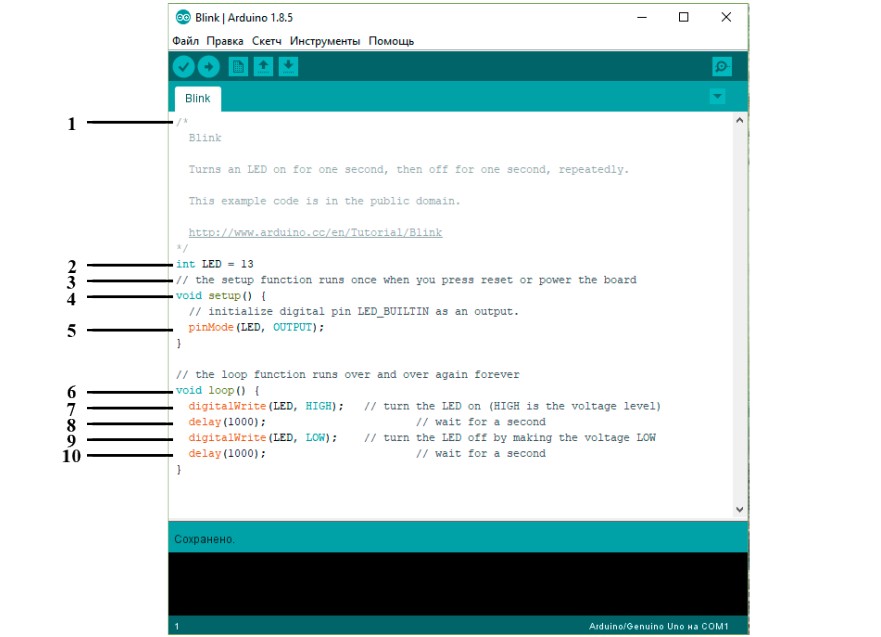
Для переменных типа int в Arduino C отводится 2 байта памяти. Если только скетч не предъявляет особых требований к экономии памяти, значения int используются практически для любых видов информации, даже для логических значений и маленьких целых чисел, которые можно было бы хранить в однобайтовом значении.

**Команды Arduino**



В библиотеке Arduino доступно большое число команд. В таблице ниже перечислены наиболее часто используемые из них вместе с примерами.





1. Многострочный комментарий. Комментарии важны для пояснения кода программы. Все, что написано между этими символами, не будет обрабатываться компилятором. Многострочные комментарии начинаются с /\* и заканчиваются \*/. Многострочные комментарии удобны, когда текст пояснения большой, например, описание программы.

2. Код объявления переменной. Переменная - это ячейка памяти, содержащая информацию. Существуют переменные различных типов. В нашем примере указана переменная типа int, что означает целое число. Целочисленной переменной LED присвоено значение 13 - номер цифрового контакта, к которому подключен светодиод на плате Arduino. Всюду в остальной части программы можно использовать переменную LED, когда мы хотим управлять контактом 13. Переменные в данном случае удобны, потому что при необходимости поменять контакт ввода-вывода достаточно изменить только одну строку, а остальная часть кода не изменится.

3. Однострочный комментарий. Если поместить // на любую строку, компилятор проигнорирует весь текст строки после этого символа. Однострочный комментарий обычно поясняет определенную строку кода.

4. Функция setup(), одна из двух функций, которые должны быть включены в каждую программу Arduino. Функция - это фрагмент кода, выполняющий определенную задачу. Код в теле функции setup о выполняется один раз в начале программы. Это полезно для установки начальных параметров настройки, назначения режимов портов ввода-вывода, инициализации коммуникационных интерфейсов и т.д.

5. Цифровые контакты Arduino могут быть запрограммированы на ввод или вывод. Сконфигурировать их направление позволяет команда pinMode(), имеющая два параметра, указанных в круглых скобках. Первый параметр pinMode определяет номер контакта. Поскольку переменная LED уже назначена ранее в программе, конфигурация задается для контакта 13. Второй параметр устанавливает направление контакта: input (вход) или output (выход). По умолчанию все контакты настроены на ввод. Чтобы сконфигурировать их на вывод, следует явно указать значение этого параметра output. Поскольку нам нужно управлять светодиодом, контакт 13 должен быть выходом. Настройка конфигурации контакта сохраняется до тех пор, пока вы не измените его назначение на ввод.

6. Вторая обязательная функция во всех программах Arduino - loop(). Это оператор цикла.

7. Функция digitalWrite() устанавливает состояние выходного контакта: 5 или 0 В. Если светодиод подсоединен к контакту через резистор, то установка значения логической «1» позволит зажечь светодиод (вы узнаете больше об этом в следующей главе). Первый параметр функции digitalWrite() — номер контакта, которым требуется управлять. Второй параметр— значение, которое нужно задать: HIGH (5 В) или LOW (0 В). Контакт остается в этом состоянии, пока не будет изменен следующей командой digitalWrite().

8. Функция delay() имеет один аргумент — время задержки выполнения программы в миллисекундах. При вызове delay() Arduino останавливает выполнение программы на определенный интервал времени. В нашем примере задержка равна 1000 мс (1 с). Это приводит к свечению светодиода в течение одной секунды до выполнения следующей команды.

9. Здесь вызвана функция digitalWrite(), чтобы выключить светодиод, устанавливая состояние контакта в LOW.

10. Снова делаем задержку на одну секунду, чтобы светодиод был погашен перед повторением цикла.

