Программирование микроконтроллеров на примере Arduino-Uno

Сегодня мы познакомимся с таким прибором, как микроконтроллер.

Что это такое и для чего необходимо?

Микроконтро́ллер (англ. Micro Controller Unit, MCU) — микросхема, предназначенная для управления электронными устройствами.

С помощью микроконтроллера можно создать систему «Умный дом», которая будет включать свет, когда Вы заходите в комнату. Или она будет в определенное время включать кофе машину или чайник, чтобы можно было сразу налить горячий напиток.

Сегодня я Вам расскажу об одном из таких микроконтроллеров.

Arduino — это электронный конструктор и удобная платформа быстрой разработки электронных устройств для новичков и профессионалов. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также программному коду. Устройство программируется через USB без использования программаторов.

Arduino позволяет компьютеру выйти за рамки виртуального мира в физический и взаимодействовать с ним. Устройства на базе Arduino могут получать информацию об окружающей среде посредством различных датчиков, а также могут управлять различными исполнительными устройствами.

Arduino — это открытая платформа, которая позволяет собирать всевозможные электронные устройства. Устройства могут работать как автономно, так и в связке с компьютером. Всё зависит от идеи.

Платформа состоит из аппаратной и программной частей; обе чрезвычайно гибки и просты в использовании. Поддерживаются операционные системы Windows, MacOS X и Linux.

Для программирования и общения с компьютером вам понадобится USB-кабель. Для автономной работы потребуется блок питания на 7,5—12 В.

Arduino Uno может питаться как от USB подключения, так и от внешнего источника: батарейки или обычной электрической сети. Источник определяется автоматически.

Платформа оснащена 32 кБ flash-памяти, 2 кБ из которых отведено под так называемый bootloader. Он позволяет прошивать Arduino с обычного компьютера через USB. Эта память постоянна и не предназначена для изменения по ходу работы устройства. Её предназначение — хранение программы и сопутствующих статичных ресурсов.

Также имеется часть памяти, которые используются для хранения временных данных вроде переменных программы. По сути, это оперативная память платформы. Ещё имеется память для долговременного хранения данных. По своему назначению это аналог жёсткого диска для Arduino.

На платформе расположены 14 контактов (pins), которые могут быть использованы для цифрового ввода и вывода. Какую роль исполняет каждый контакт, зависит от вашей программы.

Кроме этого на плате имеется входной контакт *Reset*. Его установка в логический ноль приводит к сбросу процессора. Это аналог кнопки Reset обычного компьютера.

Arduino Uno обладает несколькими способами общения с другими Arduino, микроконтроллерами и обычными компьютерами.

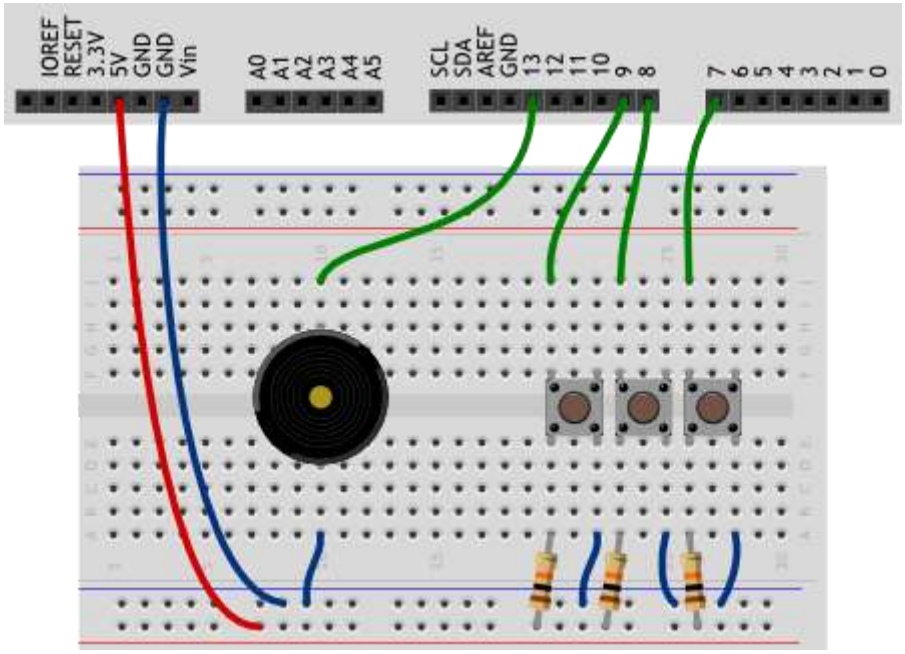
С помощью отдельных плат расширения становится возможной организация других способов взаимодействия, таких как ethernet-сеть, радиоканал, Wi-Fi.

Итак, сегодня мы на уроке сделаем пианино с помощью Arduino и сыграем свою первую мелодию на этом мини-пианино. Клавишами в нашем пианино могут быть любые электропроводящие предметы (яблоко, пластилин, лист бумаги с нарисованными клавишами пианино и т.д.).

Что нам понадобиться для урока?

- 1 плата Arduino Uno
- 1 беспаечная макетная плата
- 1 пьезопищалка (динамик)
- 3 тактовых кнопки
- 3 резистора номиналом 10 кОм
- 10 проводов «папа-папа»

Схема на макетной плате



Обратите внимание

Ножки тактовой кнопки, расположенные с одной стороны, разомкнуты, когда кнопка не нажата. Ножки, расположенные друг напротив друга на противоположных сторонах макетной платы, находятся на одной «рельсе». Воспользовавшись этим, мы можем расположить резистор с одной стороны макетной платы, а провод, подключаемый к порту Arduino, с другой стороны.

В данном эксперименте мы подключаем кнопки по схеме с подтягивающим резистором.

Для того, чтобы данный вариант программы работал, важно, чтобы кнопки были подключены к портам, находящимся рядом друг с другом, т.е. имеющим соседние номера.

Скетч

```
#define BUZZER_PIN    13 // пин с пищалкой (англ. «buzzer»)
#define FIRST_KEY_PIN 7  // первый пин с клавишей (англ. «key»)
#define KEY_COUNT     3  // общее количество клавиш

void setup()
{
 pinMode(BUZZER_PIN, OUTPUT);
}

void loop()
{
 // в цикле бежим по всем номерам кнопок от 0-го по 2-й
 for (int i = 0; i < KEY_COUNT; ++i) {
  // на основе номера кнопки вычисляем номер её пина
  int keyPin = i + FIRST_KEY_PIN;

  // считываем значение с кнопки. Возможны всего 2 варианта:
  //  * высокий сигнал, 5 вольт, истина — кнопка отпущена
  //  * низкий сигнал, земля, ложь — кнопка зажата
  boolean keyUp = digitalRead(keyPin);

  // проверяем условие «если не кнопка отпущена». Знак «!»
  // перед булевой переменной означает отрицание, т.е. «не».
  if (!keyUp) {
   // рассчитываем высоту ноты в герцах в зависимости от
   // клавиши, которую рассматриваем на данном этапе цикла.
   // Мы получим значение 3500, 4000 или 4500
   int frequency = 3500 + i * 500;

   // Заставляем пищалку пищать с нужной частотой в течение
   // 20 миллисекунд. Если клавиша останется зажатой, пищалка
   // вновь зазвучит при следующем проходе loop, а мы услышим
   // непрерывный звук
   tone(BUZZER_PIN, frequency, 20);
  }
 }
}
```

Пояснения к коду

Благодаря тому, что в начале программы мы определили `FIRST_KEY_PIN` и `KEY_COUNT`, мы можем подключать произвольное количество кнопок к любым идущим друг за другом цифровым пинам, и для корректировки программы нам не придется менять параметры цикла `for`. Изменить понадобится лишь эти константы:

- цикл в любом случае пробегает от 0 до `KEY_COUNT`;
- перед считыванием порта мы задаем смещение на номер первого используемого порта — `FIRST_KEY_PIN`.

Функция digitalRead(pin) возвращает состояние порта, номер которого передан ей параметром pin. Это может быть состояние HIGH или LOW. Или, выражаясь иначе: высокое напряжение или низкое, 1 или 0, true или false

Поскольку мы получаем с порта одно из двух состояний, мы сохраняем его в переменную уже знакомого нам типа boolean, и можем работать с ней как с логическим значением.

Мы используем логический оператор отрицания «не» !. Если keyUp имеет значение 0, выражение !keyUp будет иметь значение 1 и наоборот.

Поскольку мы собрали схему с подтягивающим резистором, при нажатии кнопки мы будем получать на соответствующем порте 0.

Действия, описанные в условном выражении if, выполняются, когда его условие имеет значение «истина» (единица). Поэтому для выполнения действия по нажатию, мы инвертируем сигнал с кнопки.

Источники:

1. Амперка - Ардуино-в помощь к уроку
2. Arduino-Uno плата и микроконтроллер
3.