Создаем собственный скетч

На начальном этапе работы с Ардуино можно использовать готовые скетчи.

Примечание. При запуске микроконтроллера Arduino начинают работать встроенные микропрограммы, которые проверяют не началась ли загрузка новой программы с компьютера.

Если новая программа не была загружена пользователем, то микроконтроллер начинает выполнять ранее загруженный скетч.

Не всегда можно найти готовый скетч, который решает нашу задачу, поэтому важно научиться писать программы самостоятельно.

Для этого рассмотрим структуру кода скетча.

Начнем со структуры программы. Любая программа состоит из отдельных блоков, начало блока обозначается левой фигурной скобкой ***{,*** конец блока – правой фигурной скобкой ***}.***

Минимально возможная программа на Ардуино выглядит так:

```
void setup()
{
}
void loop()
{
}
```

В этой программе используются две обязательных функции void setup() и void loop().

Это обязательные компоненты каждого скетча, *без них программа не будет работать,* они вызываются только один раз, даже если какая-то функция не используется, она должна присутствовать.

В этом случае просто не пишем ничего между фигурными скобками, например:

```
void setup()
{
}
```

Если между фигурными скобками пусто, то такая программа ничего не делает.

В блоке ***void setup()*** размещаются команды (функции) для предварительной настройки микроконтроллера, которые он выполнит только один раз в момент подключения платы к источнику питания или сразу после перезагрузки.

В блоке ***void loop()*** размещаются команды (функции), которые будут бесконечно выполняться, пока плата подключена к источнику питания. Микроконтроллер будет выполнять код, дойдёт до последней строки и снова начнёт с первой («loop» от англ. — петля, цикл).

Попробуем дополнить «пустую» программу, чтобы произошло действие, например, загорелся светодиод. В собранной нами модели к некоторым пинам подключены светодиоды, например, рассмотрим 5 пин с подключенным светодиодом. Как им можно управлять?

В пустые блоки добавим выражения. Каждое выражение – приказ процессору что-то сделать.

Мы хотим управлять светодиодом, присоединенным к 5 пину, хотим, чтобы он мигал постоянно. Пин надо перевести в состояние работы на выход (мы хотим выдавать сигнал), в программе это записывается выражением (функцией) в блоке setup:

```
void setup()
{
    pinMode(5, OUTPUT); // 5 пин устанавливается в режим выхода
}
```

Данная программа вызывает функцию с именем ***pinMode,*** эта функция устанавливает заданный по номеру пин в нужный нам режим: вход или выход. Номер пина и режим указываются через запятую в круглых скобках после имени функции, пин и режим выступают для данной функции аргументами. После // представлен комментарий, это часть программы, которая не исполняется, комментарии поясняют работу программы для пользователей. После // комментарий пишется в одну строку. Для записи более длинного комментария используется конструкция ***/* комментарий */***.

Примечание. У разных функций может быть разное число аргументов, в том числе и не быть аргументов, это зависит от сути самой функции.

В нашей программе функция ***pinMode*** сработает один раз, в результате плата будет настроена на работу с 5 пином. Затем начнем управлять выбранным светодиодом. Для этого используем блок loop, он называется основным циклом программы и будет выполняться снова и снова до бесконечности (пока не отключим питание!).

Для того, чтобы светодиод загорелся, на него надо подать напряжение или логическую единицу (HIGH, 5 вольт), чтобы выключился подаем логический ноль (LOW, 0 вольт). В программе подача напряжения описывается выражением digitalWrite(pin, value), у этой функции 2 аргумента – номер пина и логическое значение.

- pin — номер цифрового порта, на который мы отправляем сигнал
- value — значение, которое мы отправляем на порт. Для цифровых портов значением может быть HIGH (высокое, единица) или LOW (низкое, ноль).

Константы: INPUT, OUTPUT, LOW, HIGH, пишутся заглавными буквами, иначе компилятор их не распознает и выдаст ошибку.

Получаем следующий фрагмент кода:

```
void loop()
{
    digitalWrite(5, HIGH); // на 5 пин подаётся высокий сигнал - логическая 1 (5В)
}
```

В этом случае светодиод будет постоянно гореть.

Программа будет иметь вид:

```
void setup()
{
    pinMode(5, OUTPUT);
}
void loop()
{
    digitalWrite(5, HIGH); // на 5 пин подаётся высокий сигнал - логическая 1 (5В)
}
```

Обратите внимание, что каждой открывающейся фигурной скобке { всегда соответствует закрывающаяся }. Фигурные скобки обозначают границы логически завершенного фрагмента кода.

Каждую строку программы внутри цикла необходимо завершать символом «;».

Для проверки того, как работает код, надо скомпилировать и загрузить программу, для этого нажимаем в окне программы на кнопку **Проверить** (**V**) или **Скетч** – **Проверить/Компилировать**, затем на кнопку **Загрузить** (→) или **Скетч** – **Загрузить**.

Продолжим работу с программой. Чтобы светодиод выключить, надо поменять аргумент HIGH на LOW в выражении digitalWrite.

```
void loop()
{
    digitalWrite(5, HIGH); // на 5 пин подаётся высокий сигнал - логическая 1 (5В)
    digitalWrite(5, LOW); // на 5 пин подаётся низкий сигнал - логический 0 (0В)
}
```

В случае работы данной программы светодиод будет включаться и выключаться практически мгновенно. Для того, чтобы видеть включение и выключение светодиода, надо добавить паузы в работе процессора после включения и выключения светодиода. Используем функцию delay, которая заставляет процессор «замереть» на некоторое время, например, на 500 мс, это полсекунды (1000 мс = 1 с). Время в данной функции выступает аргументом, параметр записывается значением в миллисекундах. Фрагмент программы будет выглядеть так:

```
void loop()
{
    digitalWrite(5, HIGH);
    delay (500); //пауза 500 мс
    digitalWrite(5, LOW);
    delay (500);
}
```

Выражения в рамках одного блока выполняются одно за другим. В блоке loop после того, как выполнены все команды, повторяется все сначала и так многократно. В

приведенной программе светодиод, присоединенный к 5 пину будет поочередно включаться и выключаться с паузой 500 мс. Проверьте работу программы, для этого скомпилируйте и запустите ее, попробуйте поменять время включения и выключения.

Поработаем с двумя светодиодами, например, подключенными к 5 и 6 пинам.

```
void setup()
{
    pinMode(5, OUTPUT); //5 пин устанавливается в режим выхода
    pinMode(6, OUTPUT); //6 пин устанавливается в режим выхода
}
```

Сделаем так, чтобы они включались и выключались по очереди – то один, то другой с паузами между включениями. Фрагмент программы будет выглядеть так:

```
void loop()
{
    digitalWrite(5, HIGH);
    delay (500); //пауза 500 мс
    digitalWrite(5, LOW);
    delay (500);
    digitalWrite(6, HIGH);
    delay (500); //пауза 500 мс
    digitalWrite(6, LOW);
    delay (500);
}
```

Проверьте работу программы, для этого скомпилируйте и запустите ее, попробуйте поменять время включения и выключения.

А теперь попробуем сделать так, чтобы светодиоды работали в режиме железнодорожного семафора – горит то один, то другой. Фрагмент программы будет выглядеть так:

```
void loop()
{
    digitalWrite(5, HIGH);
    digitalWrite(6, LOW);
    delay (500); //пауза 500 мс
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    delay (500); //пауза 500 мс
}
```

Поэкспериментируйте самостоятельно, изменяя значения аргумента функции delay. Можно попробовать добавить светодиод, подключенный к портам 9, 10 или 11.