Базовые функции микроконтроллера платы Arduino

12 лекция

«Программирование микроконтроллеров на языке С++»

СТРУКТУРА КОДА

Базовая структура программы для Arduino довольно проста и состоит, по меньшей мере, из двух частей. В этих двух обязательных частях, или функциях, заключён выполняемый код.

При запуске Arduino IDE даёт нам заготовку в виде двух обязательных функций: **setup ()** и **loop ()**. Важно заметить, что обязательную в C++ функцию main() процессор Arduino создаёт сам. Результат того, что видит программист:

```
void setup()
{
  statements;
}

void loop()
{
  statements;
}
```

Функция setup ()

Функция **setup()** используется для инициализации переменных, определения режимов работы выводов на плате и т.д.

Функция запускается только один раз, после каждой подачи питания на микроконтроллер.

Она должна быть включена в программу, даже если в ней нет никакого содержания.

```
void setup()
{
  pinMode(pin, OUTPUT);       //  устанавливает 'pin' как выход
}
```

Функция loop()

После вызова функции **setup ()** – управление переходит к функции **loop ()** , которая делает в точности то, что означает её имя - непрерывно выполняется, позволяя программе что-то изменять, отвечать и управлять платой Arduino.

Функции **setup()** и **loop()** должны присутствовать в каждом скетче, даже если эти операторы в программе не используются.

```
void loop()
{
  digitalWrite(pin, HIGH);    // включает 'pin'
  delay(1000);                // секундная пауза
  digitalWrite(pin, LOW);     // выключает 'pin'
  delay(1000);                // секундная пауза
}
```

Цифровой ввод/вывод

Цифровой пин Arduino работает в двух режимах: **вход** и **выход**.

- В режиме **INPUT (вход)** пин принимает внешний сигнал и считывает высокий уровень (HIGH) или низкий уровень (LOW) напряжения, поступающий извне.
- В режиме **OUTPUT (выход)** пин подаёт сигнал, устанавливая высокий (HIGH) или низкий уровень (LOW) напряжения на выводе.

Режим работы задается функцией

pinMode(pin, mode);

где ***pin*** - номер пина, а ***mode*** – выбранный режим:

- INPUT – вход
- OUTPUT – выход
- INPUT_PULLUP – подтянутый к питанию вход

Возвращаемое значение – нет.

Цифровой ввод/вывод

pinMode() применяется в функции setup () для конфигурации заданного вывода, чтобы он работал на вход (INPUT) или на выход (OUTPUT).

```
pinMode(pin, OUTPUT);  //устанавливает pin на выход
```

Все цифровые выводы Arduino по умолчанию находятся в режиме входа с высоким импедансом (сопротивлением), и нет нужды их явно объявлять как INPUT.

В микроконтроллерах ATmega доступны встроенные подтягивающие резисторы (примерно 20–50 кОм), которые можно активировать с помощью параметра INPUT_PULLUP, обеспечивая устойчивый логический уровень при отсутствии внешнего сигнала.

Цифровой ввод/вывод

Функция **digitalWrite (pin, value)**

Передаёт на указанный цифровой вывод логический уровень HIGH или LOW.
Номер вывода можно задавать как константу (0–13) или как переменную.
Возвращаемое значение – нет.

Работа с выводом, сконфигурированным как OUTPUT

- digitalWrite(pin, HIGH) — на вывод подаётся 5 В (или 3.3 В).
- digitalWrite(pin, LOW) — на вывод подаётся 0 В (GND).

Работа с выводом, сконфигурированным как INPUT

- digitalWrite(pin, HIGH) включает внутренний подтягивающий резистор (≈20 кОм).
- digitalWrite(pin, LOW) отключает подтяжку.

Цифровой ввод/вывод

Пример

```
int ledPin = 13;                // светодиод подключен к выводу 13
void setup()
{
      pinMode (ledPin, OUTPUT);  // переключаем цифровой вывод в режим выхода
}
void loop()
{
      digitalWrite (ledPin, HIGH);   // включаем светодиод
      delay(1000);                   // ждем 1 секунду
      digitalWrite (ledPin, LOW);    // выключаем светодиод
      delay(1000);                   // ждем 1 секунду
}
```

Цифровой ввод/вывод

Функция **digitalRead (pin)** считывает значение с заданного цифрового вывода (pin) и возвращает результат HIGH (на выводе высокий уровень напряжения) или LOW (на выводе низкий уровень напряжения).

Вывод должен быть задан либо как переменная, либо как константа (0-13).

```
int buttonPin = 7;          // номер пина сохраняем в переменную
int state = digitalRead(buttonPin);  // считываем сигнал
```

```
int state = digitalRead(7);  // считываем сигнал с пина 7
```

Цифровой ввод/вывод

Пример

```
int ledPin = 13; // светодиод подсоединен к цифровому выводу 13
int inPin = 7;   // кнопка подсоединена к цифровому выводу 7
int val = 0;     // переменная для хранения считанного значения

void setup()  {
pinMode(ledPin, OUTPUT);      // настраиваем светодиод как выход
pinMode(inPin, INPUT);      // настраиваем кнопку как вход
}

void loop()    {
val = digitalRead(inPin);   // считываем состояние кнопки HIGH/LOW
digitalWrite(ledPin, val);    // передаём это значение на светодиод
}
```

Аналоговый ввод/вывод

Функция **analogRead (pin)**

pin - номер аналогового входа, с которого будет считываться напряжение (0 – 5В для большинства плат, 0 - 7В для Mini и Nano, 0 – 15В для Mega).

Возвращает значение целого числа (int) от 0 до 1023.

- Большинство плат Arduino имеют 6 аналоговых входов (8 у платы Mini и Nano, 16 у Mega) с 10-битным аналого-цифровым преобразователем (АЦП). Напряжение поданное на аналоговый вход, обычно от 0 до 5 вольт будет преобразовано в значение от 0 до 1023, это 1024 шага с разрешением 5В/1024≈0.0049В (4.9 мВ на шаг).
- Считывание значение с аналогового входа занимает примерно 100 микросекунд (0.0001 сек), т.е. максимальная частота считывания приблизительно 10,000 раз в секунду.
- Входной диапазон и разрешающая способность могут меняться с помощью функции analogReference().

Аналоговый ввод/вывод

Функция **analogWtite (pin, value)**

pin - вывод, на котором будет формироваться напряжение.

value - коэффициент заполнения – лежит в пределах от 0 (всегда выключен) до 255 (всегда включен).

Возвращаемые значения **-** нет

Функция формирует заданное аналоговое напряжение на выводе в виде ШИМ-сигнала. Может использоваться для варьирования яркости свечения светодиода или управления скоростью вращения двигателя. После вызова analogWrite(), на выводе будет непрерывно генерироваться ШИМ-сигнал с заданным коэффициентом заполнения до следующего вызова функции analogWrite() (либо до момента вызова digitalRead() или digitalWrite(), взаимодействующих с этим же выводом). Частота ШИМ составляет приблизительно 490 Гц.

Аналоговый ввод/вывод

На большинстве плат Arduino (на базе микроконтроллеров ATmega168 или ATmega328) функция analogWrite() работает с выводами 3, 5, 6, 9, 10 и 11. На более старых версиях Arduino (на базе микроконтроллера ATmega8) функция analogWrite() работает только с выводами 9, 10 и 11.

Примечание: Аналоговые выводы не похожи на цифровые, и нет необходимости предварительно объявлять их как INPUT или OUTPUT (если только вы не планируете использовать их в качестве цифровых портов 14-18).

- При работе с analogWrite() предварительный вызов функции pinMode() для переключения выводов в режим «выход» не требуется.
- Функция analogWrite() не имеет ничего общего с аналоговыми выводами и функцией analogRead().

Аналоговый ввод/вывод

- **Пример** Формирование на выводе, управляющим светодиодом, напряжения пропорционального напряжению на потенциометре. (задание яркости светодиода пропорционально значению, снимаемому с потенциометра)

```
int ledPin = 9;      // светодиод подключен к цифровому выводу 9
int analogPin = 3;   // потенциометр подключен к аналоговому выводу 3
int val = 0;         // переменная для хранения считанного значения

void setup() {
 pinMode(ledPin, OUTPUT);   // настраиваем пин светодиода как выход }

void loop() {
 val = analogRead(analogPin);   // считываем напряжение с потенциометра
analogWrite(ledPin, val / 4);  /*регулируем яркость светодиода , значения из
диапазона 0–1023 возвращаемые analogRead  переводятся в диапазон 0–255,
который понимает analogWrite*/ }
```

.

Функции времени

Функция delay (ms) останавливает выполнение программы на заданное время в миллисекундах, где 1000 миллисекунд равно 1 секунде.

```
delay(1000);        // ожидаем секунду
```

- **Предупреждение.** Функция задержки используется для решения многих задач. С помощью функции delay() заставить мигать светодиод достаточно просто. Несмотря на это, использование функции delay() в коде программы имеет существенные недостатки. В процессе действия delay() такие операции, как считывание данных с датчиков, математические вычисления или операции с выводами не могут выполняться. Фактически, функция delay() приводит к остановке практически всех операций. Альтернативный способ контролировать время - использование функции millis().

Функции времени

Пример

```
int ledPin = 13;                // светодиод подключен на порт 13

void setup()  {
pinMode(ledPin, OUTPUT); // устанавливается режим порта-выход
}

void loop()    {
digitalWrite(ledPin, HIGH);     // включаем светодиод
delay(1000);                    // ожидаем секунду
 digitalWrite(ledPin, LOW);    // выключаем светодиод
delay(1000);                    // ожидаем секунду
}
```

Функции времени

- Опытные программисты обычно избегают использования delay() для установки временных интервалов больше нескольких десятков миллисекунд (за исключением очень простых программ Arduino).
- Тем не менее, некоторые события и участки кода могут работать и в процессе выполнения микроконтроллером функции delay(), т.к. эта функция не влияет на работу прерываний. Так, по прежнему будут срабатывать прерывания, записываться данные, поступающие на вывод RX по последовательному интерфейсу, а также будет поддерживаться ШИМ-сигнал, формируемый функцией analogWrite().

Функции времени

Функция **millis()**

Возвращает количество миллисекунд с момента начала выполнения текущей программы на плате Arduino.

Счётчик автоматически сбрасывается на ноль примерно через 50 дней (из-за переполнения).

Функция без параметров, т.е. функция не принимает аргументы, достаточно просто вызвать millis().

```
time = millis();     //выводит количество миллисекунд
                     //с момента начала выполнения программы
```

- **Предупреждение**. Помните, что значение, возвращаемое функцией millis(), имеет тип unsigned long. При попытке выполнения математических операций между этим значением и значениями другого типа (например, int) будет сгенерирована ошибка.

Функции времени

Пример

```
unsigned long time;

void setup() {
Serial.begin(9600);
}

void loop()    {
        Serial.print("Time: ");
        time = millis();
        Serial.println(time); //выводит количество миллисекунд с момента
                                    // начала выполнения программы
        delay(1000); // ждет 1 секунду, перед следующей итерацией цикла,
                            // задаем чтобы не отправлять большой массив данных
}
```

Функции времени

Функция **micros()**

Возвращает количество микросекунд с момента начала выполнения текущей программы на плате Arduino.

Счётчик автоматически сбрасывается на ноль примерно через 70 минут (из-за переполнения).

```
time = micros();  //выводит количество микросекунд
                  //с момента начала выполнения программы
```

На платах Arduino с 16 MHz (например, Uno, Nano, Mega) функция **micros()** имеет **разрешение 4 µs и** возвращаемое значение всегда кратно 4.

На платах с 8 MHz (например, Lilypad) разрешение функции составляет **8 µs**, и значения кратны 8.

Функции времени

Пример

```
unsigned long time;

void setup() {
Serial.begin(9600);
}

void loop()    {
        Serial.print("Time: ");
        time = micros();
        Serial.println(time); //выводит количество микросекунд с момента
                                    // начала выполнения программы
        delay(1000); // ждет 1 секунду, перед следующей итерацией цикла,

}
```