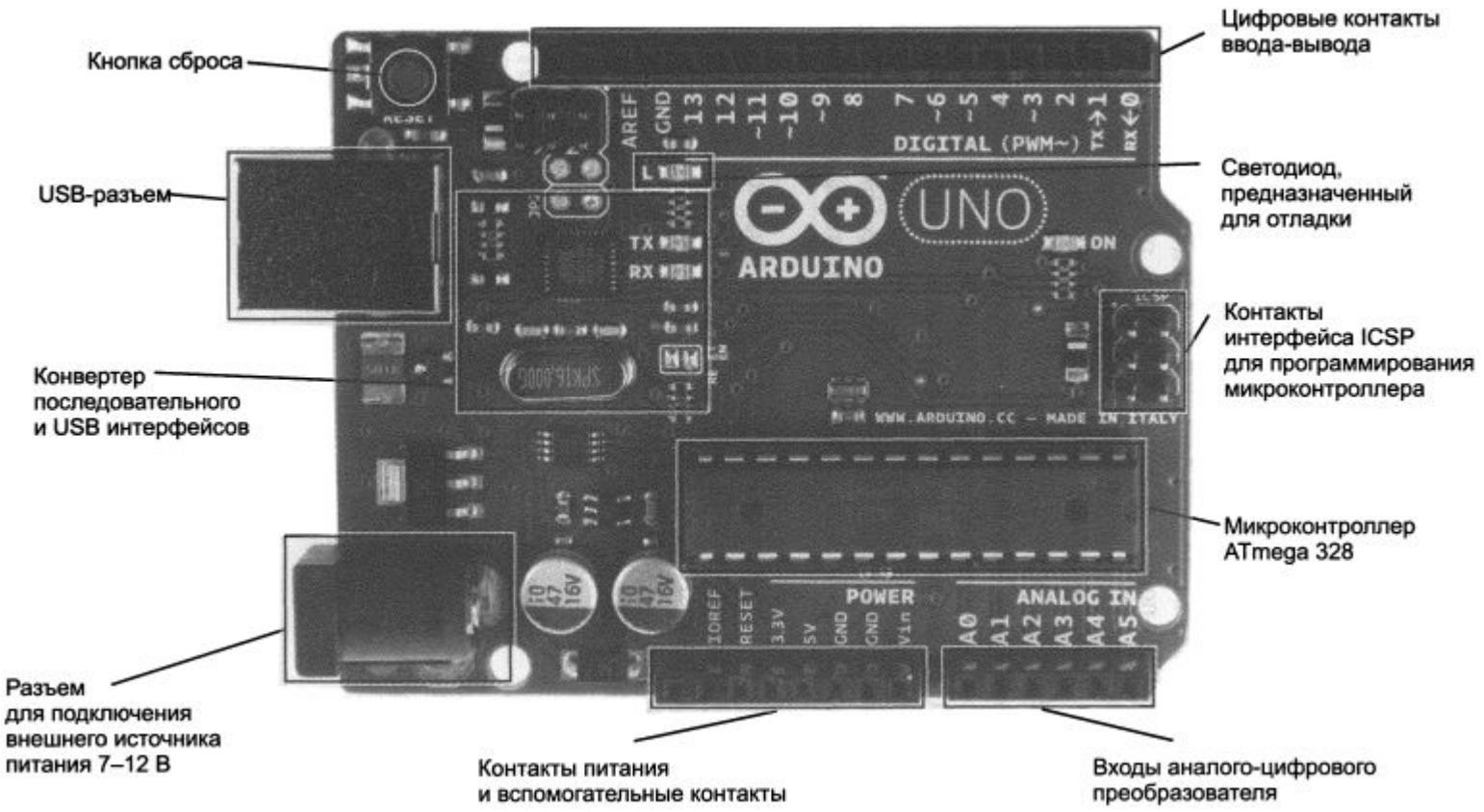
Лекция №11
по курсу
«Системное программирование»
тема: «Программирование систем реального времени на Arduino»

Лектор: д.т.н., Оцоков Шамиль Алиевич,

email: otsokovShA@mpei.ru

Москва, 2021

Arduino


Кнопка сброса
USB-разъем
Конвертер последовательного и USB интерфейсов
Разъем для подключения внешнего источника питания 7–12 В
Цифровые контакты ввода-вывода
Светодиод, предназначенный для отладки
Контакты интерфейса ICSP для программирования микроконтроллера
Микроконтроллер ATmega 328
Контакты питания и вспомогательные контакты
Входы аналого-цифрового преобразователя
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
WWW.ARDUINO.CC – MADE IN ITALY

Симулятор Arduino

Зарегистрироваться в системе [1,2]

1. https://www.tinkercad.com
2. Создание схемы в Tinkercad
3. Программируем скетч виртуального Arduino

1. https://zen.yandex.ru/media/id/5d0992b0a0412200b1332b91/tinkercad-arduino--luchshii-onlain-simuliator-arduino-na-russkom-5f2ac22d7f7edb5a704063b3
2. https://lectmania.ru/1x8d89.html
3. Джереми Блум. Изучаем Arduino- инструменты и методы технического волшебства.

Функция digitalWrite () устанавливает состояние выходного контакта: 5В или 0В. Если светодиод подсоединен к контакту через резистор, то установка значения логической " 1" позволит зажечь светодиод.

Первый параметр функции digitalWrite() — номер контакта, которым требуется управлять.

Второй параметр— значение, которое нужно задать: high (5 В) или low (О В). Контакт остается в этом состоянии, пока не будет изменен следующей командой digitalWrite.

Подключая светодиоды, необходимо соблюдать правильную полярность. Положительный вывод светодиода называется анодом, отрицательный— катодом.

Определить назначение контактов светодиода можно визуально: вывод катода короче, чем анода. Ток через светодиод течет только в одном направлении: от анода к катоду.

Поскольку ток протекает от положительного полюса к отрицательному, анод светодиода следует подключить к источнику тока (цифровой выход +5 В), а катод —к земле.

Простейшая схема маяка

```
// C++ код
int led = 12;
void setup()
{

  pinMode(led, OUTPUT);
}

void loop()
{
  digitalWrite(led, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(led, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Широтно-импульсная модуляция с помощью analogWrite

Задача.

Написать программу, которая плавно увеличивает и уменьшает яркость светодиода.

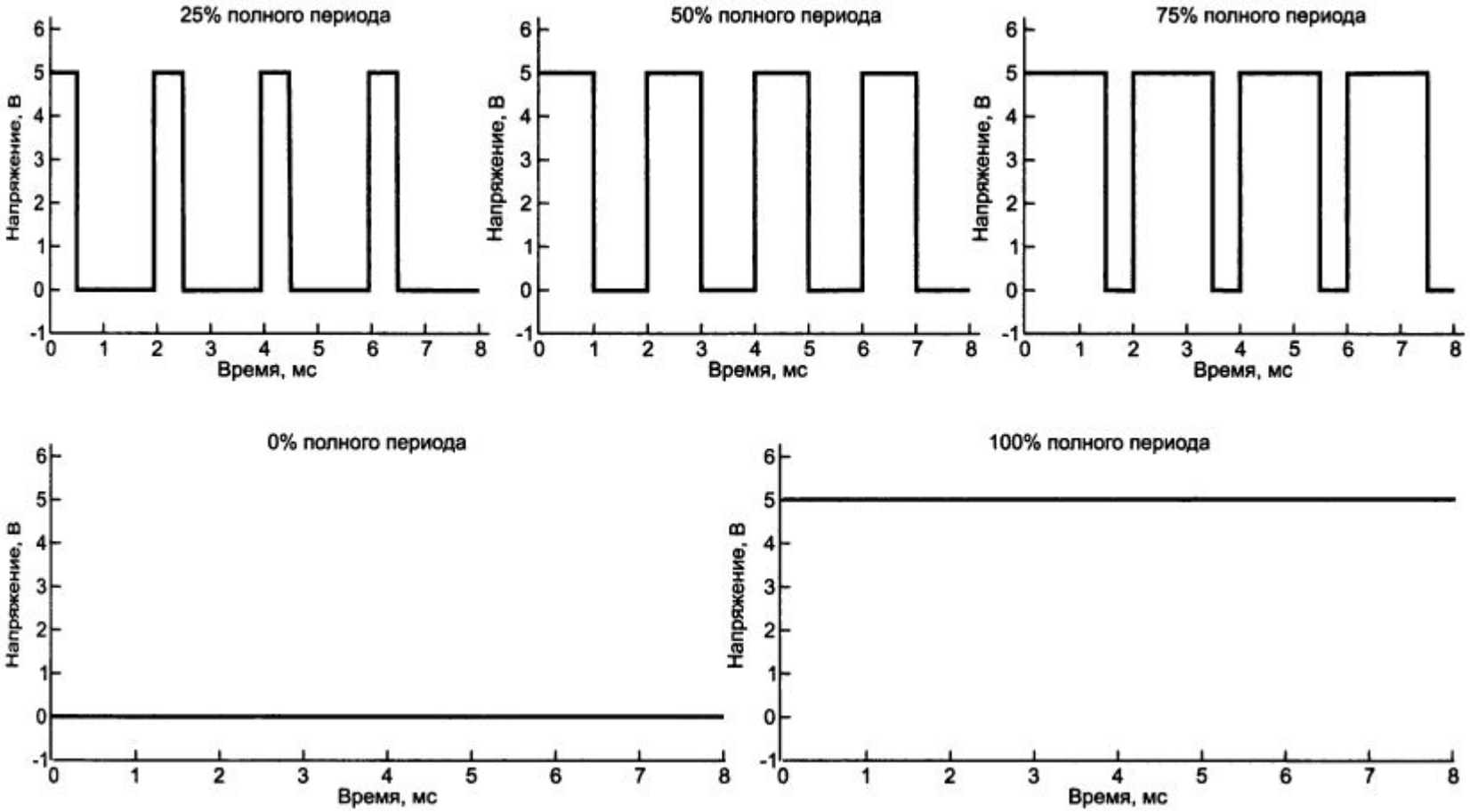
Генерация аналоговых значений на цифровых контактах с помощью широтно-импульсной модуляции (ШИМ). Для некоторых контактов Arduino сформировать ШИМ-сигнал можно командой analogWrite.

Контакты, которые могут выдавать ШИМ-сигнал на определенные периферийные устройства, помечены символом ~ на плате Arduino. На Arduino Uno контакты 3, 5,6, 9, 10, 11 поддерживают выдачу ШИМ-сигнала.

ШИМ представляет собой изменение скважности (отношения периода к длительности импульса) прямоугольной последовательности импульсов. Скважность можно трактовать как процент времени, когда прямоугольный импульс имеет уровень

high, ко всему периоду повторения. Скважность 50% означает, что половину периода сигнал имеет высокий уровень, а половину — низкий.

Широтно-импульсная модуляция с помощью analogWrite



Широтно-импульсная модуляция с помощью analogWrite

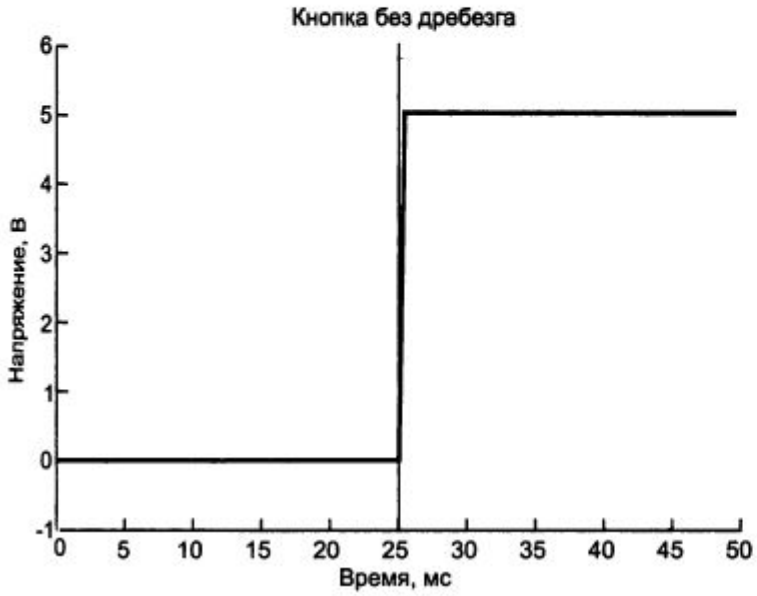
Функция `analogWrite()` устанавливает скважность последовательности прямоугольных импульсов в зависимости от значения, передаваемого ей:

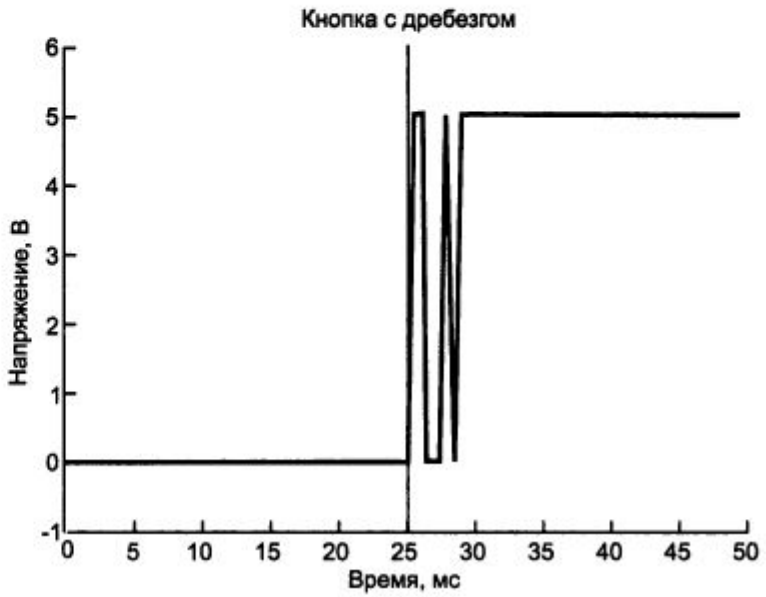
- значение аргумента `analogWrite()`, равное нулю, задает скважность 0% (всегда `LOW`);
- значение 255 — скважность 100% (всегда `HIGH`);
- значение 127 соответствует скважности 50% (половина времени `HIGH`, половина времени `LOW`).

Напряжение, подаваемое на светодиод, на самом деле не понижается, почему же при уменьшении скважности наблюдается спад яркости свечения светодиода? Это связано с особенностью нашего зрения. Если светодиод включается и выключается один раз за 1 мс (при скважности 50%), то вам кажется, что яркость свечения светодиода составляет приблизительно 50% от максимальной, потому что переключение происходит быстрее, чем глаза могут это зафиксировать. Ваш мозг фактически усредняет сигнал и создается впечатление, что светодиод работает на половине яркости.

Эффект «дребезга кнопок»

Обычные кнопки представляют собой механические устройства с пружинным контактом. При нажатии на кнопку сигнал не просто меняется от низкого до высокого, он на протяжении нескольких миллисекунд неоднократно меняет свое значение, прежде чем установится уровень LOW. Отличие ожидаемого процесса от реального иллюстрируют осциллограммы сигнала с кнопки, приведенные на рис.



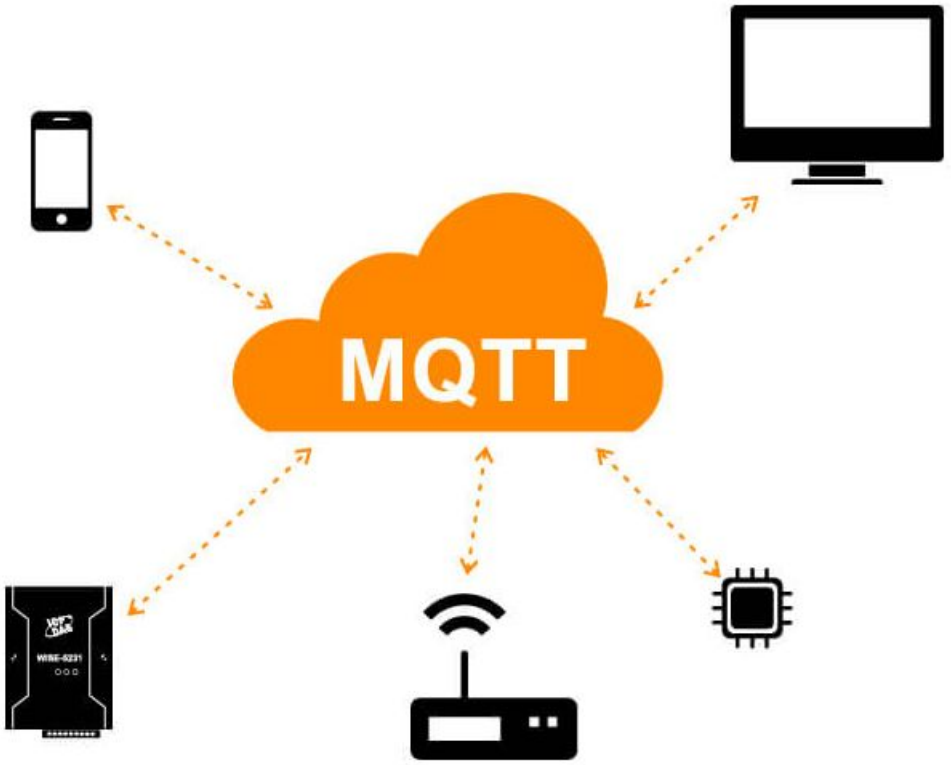


Эффект «дребезга кнопок»

Написать программу для кнопки с дребезгом, которая фиксирует изменение состояния кнопки, некоторое время ждет и затем снова читает состояние переключателя.

1. Сохраняем предыдущее и текущее состояния кнопки (при инициализации `LOW`).
2. Считываем текущее состояние кнопки.
3. Если текущее состояние кнопки отличается от предыдущего, ждем 5 мс, потому что кнопка, возможно, изменит свое состояние.
4. Подождав 5 мс, считываем состояние кнопки и делаем его текущим состоянием кнопки.
5. Если предыдущее состояние кнопки было `LOW`, а текущее — `HIGH`, переключаем состояние светодиода.
6. Устанавливаем предыдущее состояние кнопки в качестве текущего.
7. Возвращаемся к шагу 2.

MQTT



С развитием промышленности увеличивается количество устройств, которые нужно контролировать и получать от них различные данные. Для решения проблем взаимодействия большого количества устройств и проблем объединения устройств в одну сеть была создана концепция Интернета вещей (англ. Internet of Things, IoT) – это когда устройства объединяются по какому-то признаку в одну сеть, потом уже несколько подобных сетей объединяются в другую большую сеть и так далее.

Устройства в таких сетях взаимодействуют друг с другом по средствам различных интерфейсов и протоколов передачи данных. Так как мы говорим о промышленном применении концепции IoT, в которой должны использоваться промышленное оборудование со своими протоколами и

MQTT

Что такое MQTT?
MQTT или Message Queue Telemetry Transport – это легкий, компактный и открытый протокол обмена данными созданный для передачи данных на удалённых локациях, где требуется небольшой размер кода и есть ограничения по пропускной способности канала. Вышеперечисленные достоинства позволяют применять его в системах M2M (Машинно-Машинное взаимодействие) и IIoT (Промышленный Интернет вещей).

Также существует версия протокола MQTT-SN (MQTT for Sensor Networks), ранее известная как MQTT-S, которая предназначена для встраиваемых беспроводных устройств без поддержки TCP/IP сетей, например, Zigbee.

Основные особенности протокола MQTT:

Асинхронный протокол
Компактные сообщения
Работа в условиях нестабильной связи на линии передачи данных
Поддержка нескольких уровней качества обслуживания (QoS)
Легкая интеграция новых устройств
Протокол MQTT работает на прикладном уровне поверх TCP/IP и использует по умолчанию 1883 порт (8883 при подключении через SSL).

MQTT

Устройства MQTT используют определенные типы сообщений для взаимодействия с брокером, ниже представлены основные:

- Connect – установить соединение с брокером
- Disconnect – разорвать соединение с брокером
- Publish – опубликовать данные в топик на брокере
- Subscribe – подписаться на топик на брокере
- Unsubscribe – отписаться от топика

Схема простого взаимодействия между подписчиком, издателем и брокером

MQTT

Семантика топиков
Топики представляют собой символы с кодировкой UTF-8. Иерархическая структура топиков имеет формат «дерева», что упрощает их организацию и доступ к данным. Топики состоят из одного или нескольких уровней, которые разделены между собой символом «/».

Пример топика в который датчик температуры, расположенный в спальной комнате публикует данные брокеру:

/home/living-space/living-room1/temperature

Подписчик может так же получать данные сразу с нескольких топиков, для этого существуют wildcard. Они бывают двух типов: одноуровневые и многоуровневые. Для более простого понимания рассмотрим в примерах каждый из них:

Одноуровневый wildcard. Для его использования применяется символ «+»
К примеру, нам необходимо получить данные о температуры во всех спальных комнатах:
/home/living-space/+/temperature
В результате получаем данные с топиков:
/home/living-space/living-room1/temperature
/home/living-space/living-room2/temperature
Многоуровневый wildcard. Для его использования применяется символ «#»
В результате получаем данные с топиков:

/home/living-space/living-room1/temperature
/home/living-space/living-room1/light1

MQTT

Web клиент Mqtt

http://www.hivemq.com/demos/websocket-client/

```
static MqttClient client;
static void Subscribe()
{
// create client instance (both host name and IP address work nicely)
client = new MqttClient("broker.mqtt-dashboard.com");
// register to message received
client.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;
string clientId = Guid.NewGuid().ToString();
client.Connect(clientId);
// subscribe to the topic "/home/temperature" with QoS 2
client.Subscribe(new string[] { "home/temperature" }, new byte[] { MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
}
```

MQTT

Web клиент Mqtt

http://www.hivemq.com/demos/websocket-client/

```
private static void Client_MqttMsgPublishReceived(object sender,
uPLibrary.Networking.M2Mqtt.Messages.MqttMsgPublishEventArgs e)
{
string response;

response = Encoding.UTF8.GetString(e.Message);
Console.WriteLine(response+" "+e.Topic);
}

static void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{
// handle message received
Console.WriteLine("Received = " + Encoding.UTF8.GetString(e.Message) + " on topic " + e.Topic);
}
```