

авиоюньдб ыыйцаот

Методические рекомендации по работе с платформой Arduino

Быстрый старт в программирование
микроконтроллеров



ОГЛАВЛЕНИЕ

Введение	4
Основные термины.....	5
Начало работы	8
Установка драйверов.....	12
Установка библиотек через менеджер библиотек	14
Установка сторонних библиотек.....	16
Проверка работоспособности.....	17
Проект №1	17
Аналогово – цифровой преобразователь.....	19
Проект №2	21
ЖК-дисплей MT-16S2H.....	22
Проект №3	23
OLED дисплей 1306.....	24
Проект №4	25
Проект №5	26
Широтно-импульсная модуляция	27
Проект №6	28
Сервопривод SG90	30
Проект №7	31
UART интерфейс. Bluetooth HC-06	32
Проект №8	33
Датчик распознавания преград. YL-63.....	34
Проект №9	35
Термодатчик на терморезисторе. MF52 NTC-MF52AT 3950B	37
Проект № 10	38
Датчик температуры. DS18B20	39
Проект №11	40
Датчик дыма. MQ-2	41
Проект №12	42
Датчик протечки воды.....	43
Проект №13	43

Датчик освещённости на фоторезисторе	44
Проект №14	45
Внешнее аппаратное прерывание.....	46
Проект №15	48
Датчик влажности. ДН-11	49
Проект №16	50
Часы реального времени. DS1302	51
Проект №17	52
RFID связь. RFID 522 (MFRC522).....	53
Проект №18	55
Сторожевой таймер. Watchdog	56
Проект №19	57
MP3 плеер. MP3-TF-16P	58
Проект №20	59
EEPROM	60
Проект №21	63
Гироскоп и Акселерометр. GY-521 (MPU-6050).....	65
Проект №22	66
Электромагнитное реле	67
Проект №23	68
Основы языка C/C++ и Arduino команд	69
Типы данных. Переменные.....	69
Операторы	71
Основные команды C/C++	72
Основные Arduino команды.....	73
Возможные проблемы и ошибки.....	77

Введение

Arduino — это электронная платформа с открытым исходным кодом, основанная на простом в использовании аппаратном и программном обеспечении.

В основе Arduino лежат микроконтроллеры семейства AVR. Например, в оригинальной Arduino UNO заложен микроконтроллер ATmega328.

Так как платформа open-source (открытая код), можно собирать и дорабатывать собственные платы, но в этом курсе мы рассмотрим только использование различных платформ и использование кода на практически примерах с использованием различных модулей и не только.

В отличие от STM32, программирование Arduino не требует глубокого погружения в составные части микроконтроллеров, можно писать код и заниматься электроникой с заметной лёгкостью в отличие от других платформ.

К основным преимуществам можно отнести:

- 1) Низкая стоимость
- 2) Удобство использования
- 3) Выборка среды разработки
- 4) Взаимозаменяемые чипы
- 5) Низкий порог вхождения
- 6) Огромное и развивающееся комьюнити
- 7) Большой выбор модулей для различных задач

Из недостатков:

- 1) Слабая производительность
- 2) Малое количество пинов

От студентов-первопроходцев в сфере программирования микроконтроллеров STM32 и активных участников, развивающих и популяризирующих электронику и программирование, прилагаем методические рекомендации по работе с Arduino-платформой для создания различных проектов.

Желаем удачи в освоении материалов!

Основные термины

Напряжение - физическая величина, равная отношению работы по перемещению заряда, выполненной электрическим полем, к величине заряда. (1 вольт равен единице работы на один заряд)

Напряжение измеряется в **Вольтах V**

Что бы измерить силу тока, необходимо использовать вольтметр (подключать строго **параллельно**)

Сила тока - физическая величина, которая равна заряду, прошедшему через поперечное сечение проводника в единицу времени.

Сила тока измеряется в **Амперах A**

Что бы измерить силу тока, необходимо использовать амперметр (подключать строго **последовательно**)

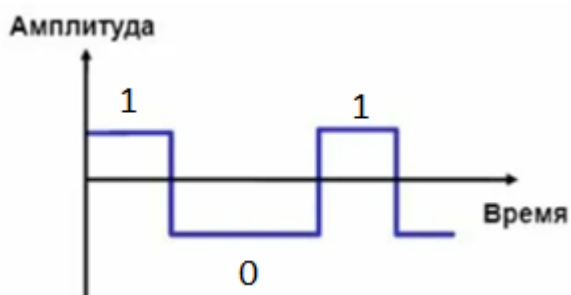
Сопротивление - физическая величина, которая характеризует способность проводника препятствовать протеканию тока.

Сопротивление измеряется в **Омах Ом**

Скейч – файл (расширение .ino) с кодом, который заливают на плату ардуино.

Пины – для подключения модулей или проводов.

Дискретный сигнал – сигнал, который принимает значения только в определенные участки времени. На такой сигнал можно наложить “маску” в виде нулей и единиц. Имеет представление в виде “лесенки”.



Аналоговый сигнал – непрерывный сигнал, представляется в виде синусоиды. Такой сигнал можно встретить в природе: например, звук или цвета (определенная длина волны, отраженная от объекта). Для такого сигнала можно измерить напряжение в любой момент времени.



Компиляция – процесс сборки кода и перевода его в машинный язык. Код собирается полностью.

Интерпретация – процесс сборки кода и перевода его в машинный язык. Код собирается построчно.

Нагрузка – модули, подключаемые к Arduino, могут потреблять различный ток (Максимальный ток, который может выдать один вывод, равен 50мА, а суммарный ток всех выводов не должен превышать значение 200мА.)

Диод — это полупроводниковый элемент электрической сети, который пропускает электрический ток только в одном направлении. Он состоит из двух основных частей: анода + и катода -.

Светодиод — это полупроводниковый элемент электрической сети, который преобразует электрическую энергию в световую.

Резистор – элемент электрической сети, обладает постоянным сопротивлением. Основное предназначение: ограничение силы тока.

Переменный резистор – это резистор, у которого можно изменять сопротивление.

Компаратор – это устройство, предназначено для сравнения двух входных сигналов и выдачи соответствующего результата на основе этого сигнала.

Крутящий момент — это произведение силы на длину рычага. Это показатель, насколько тяжёлый груз сервопривод способен удерживать в покое на рычаге заданной длины.

Интерфейс — это все, что помогает людям управлять устройствами и программами с помощью голоса, нажатий, жестов, с помощью командной строки или иных сигналов.

Редуктор — это механическое устройство, способное изменять угловую скорость и крутящий момент.

Инфракрасные светодиоды (ИК-диоды) — это полупроводниковые электронные устройства, которые излучают инфракрасное излучение с длиной волны от 760 до 1400 нм, невидимой для человеческого глаза (длина волны больше, чем видимый свет).

Фотодиоды – это полупроводниковые приборы, способные переводить световые сигналы в электрические. Их широко применяются в различных сферах, таких как оптическая связь, фотометрия, спектроскопия, медицина и многие другие.

Начало работы

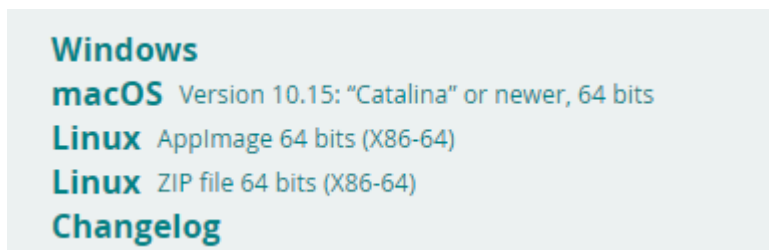
Театр начинается с вешалки, а работа на Arduino начинается с IDE (Интегрированная среда разработки), мы будем использовать Arduino IDE (хотя есть множество других платформ, мы остановился на самом популярном и универсальном варианте)

Для установки надо зайти на [официальный сайт](#) (или перейти по прямой ссылке для скачивания для [Windows](#), [MacOS](#), [Linux](#))

Вкладка “Software”



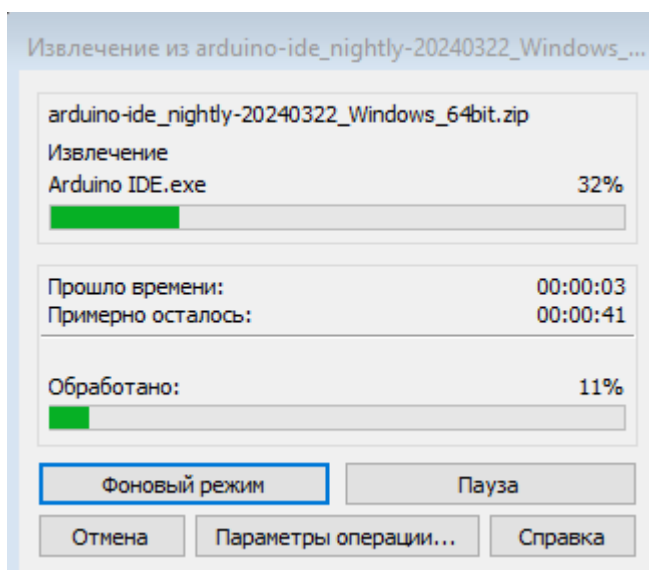
Выбираем нужную операционную систему



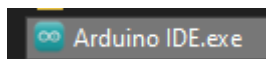
Пропускаем всё нажимая кнопку “JUST DOWNLOAD”

JUST DOWNLOAD

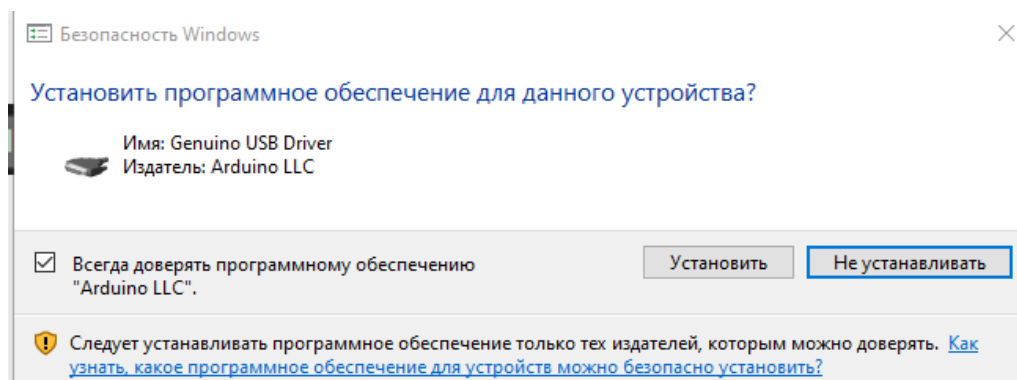
Извлекаем архив



Запускаем файл



Устанавливаем дополнительные компоненты для Arduino

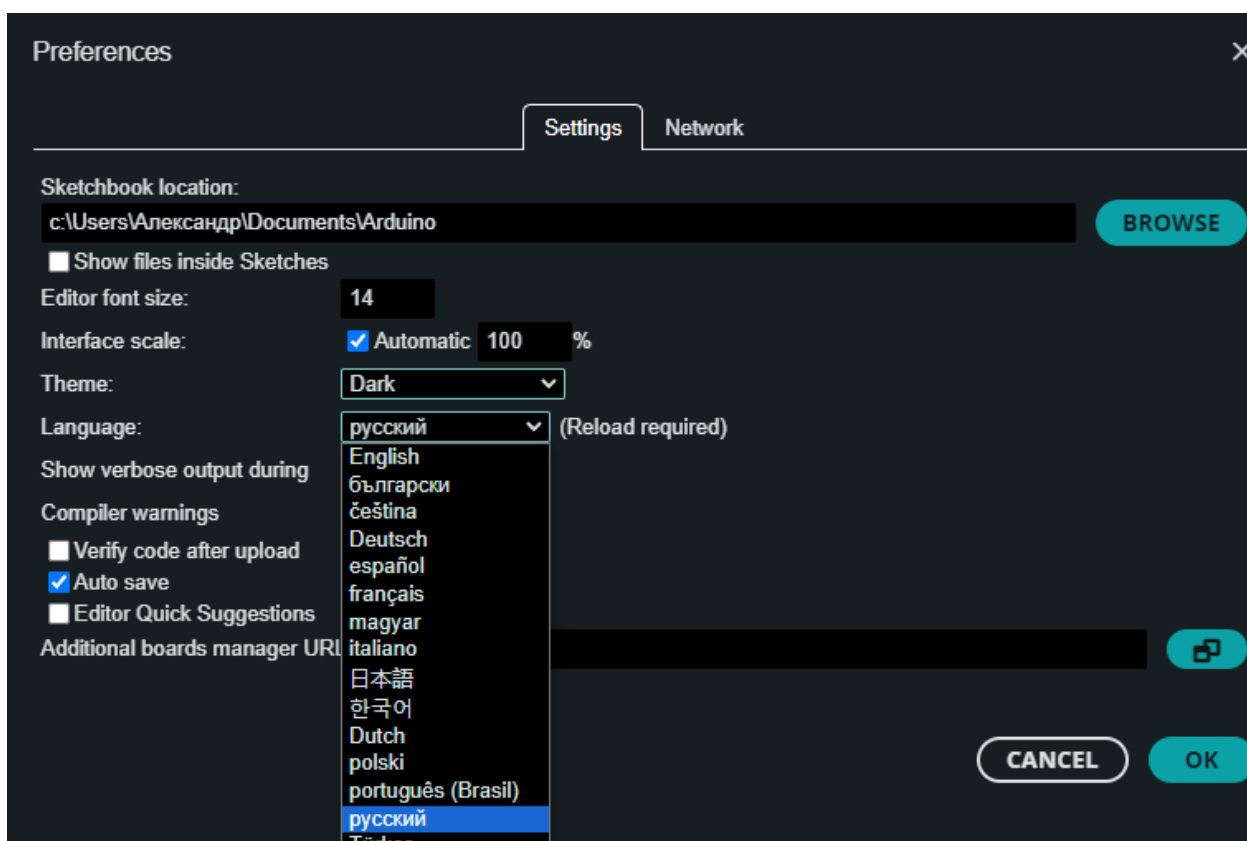


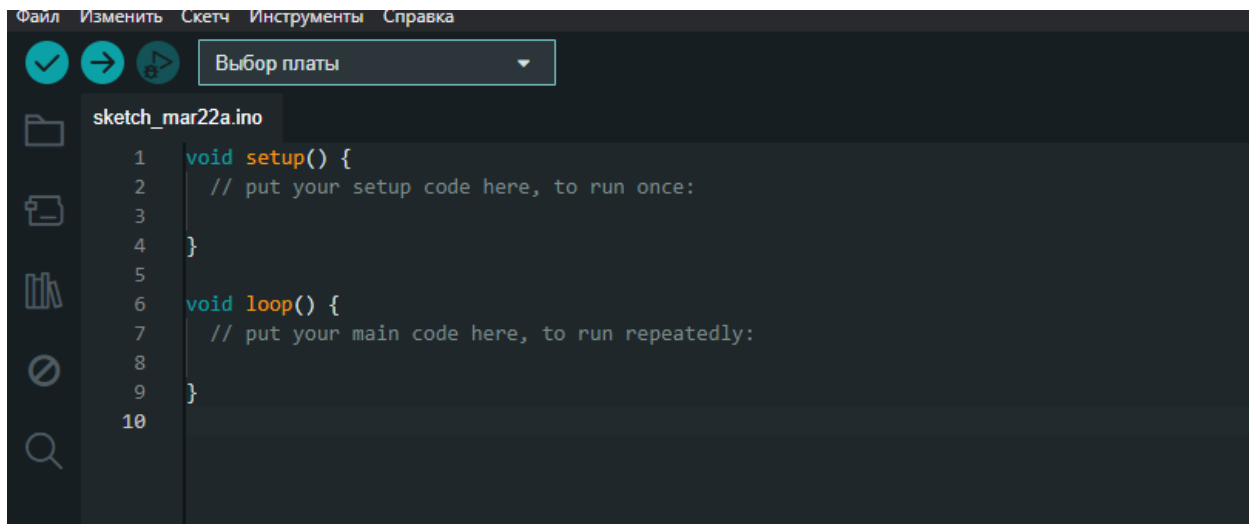
Открываем тот же файл еще раз

Рекомендовано создать ярлык и вытащить его на рабочий стол, чтобы в дальнейшем не искать его по компьютеру

Что бы перевести интерфейс на русский язык необходимо: File – Preferences – Language – выбираем русский язык

Программа перезагрузиться и будет на русском языке





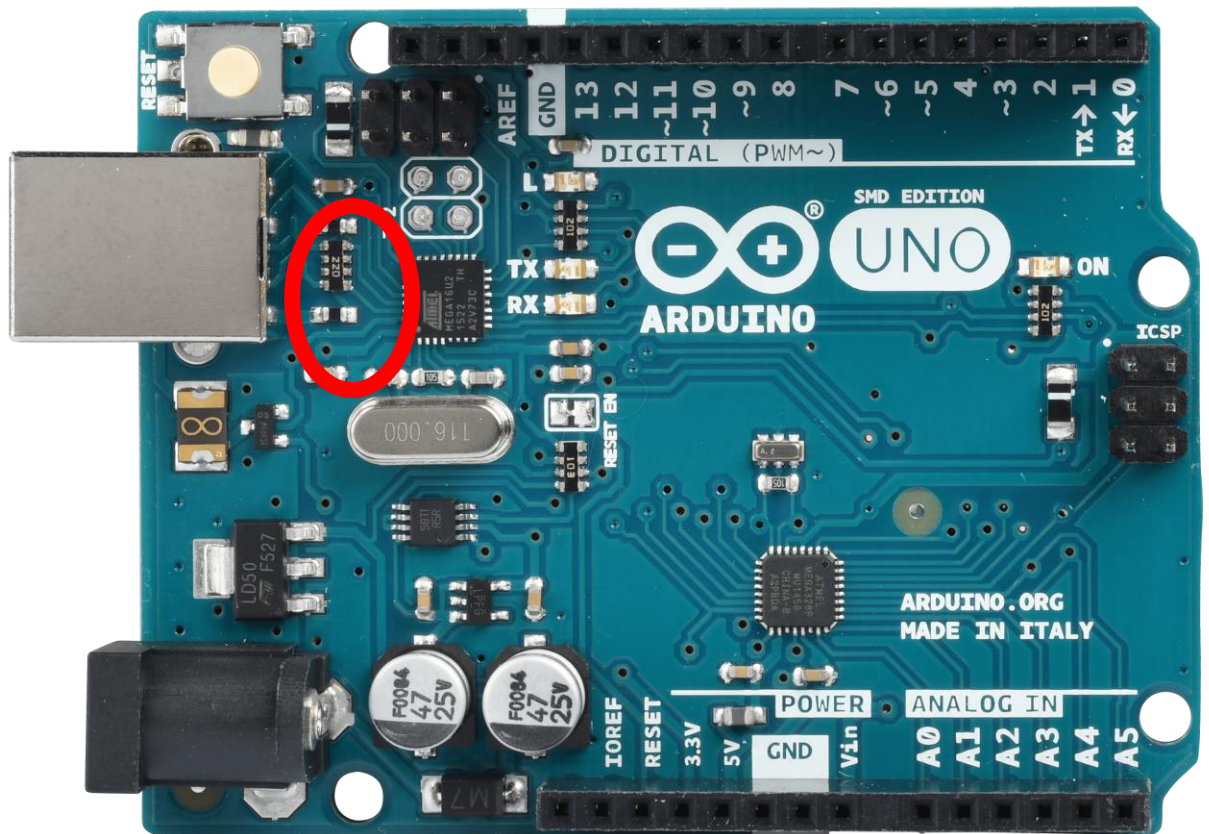
Это начало всех скейчей

Блок “void setup()” это код, который выполняется только один раз, при включении Arduino.

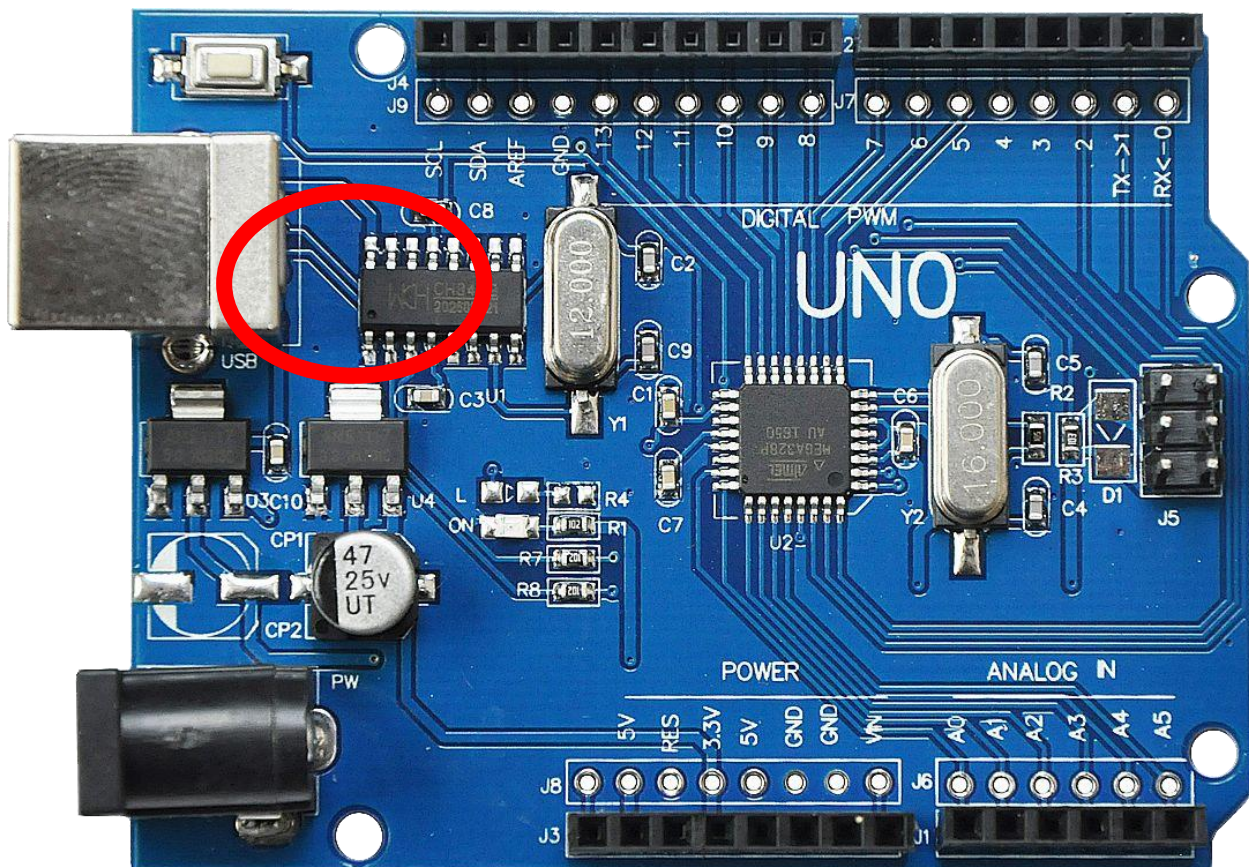
Блок “void loop()” это код, который выполняется всегда, пока есть питание.

Теперь необходимо понять тип вашей платы Arduino UNO.

Есть два типа: Оригинальная плата Arduino UNO



И её китайская копия

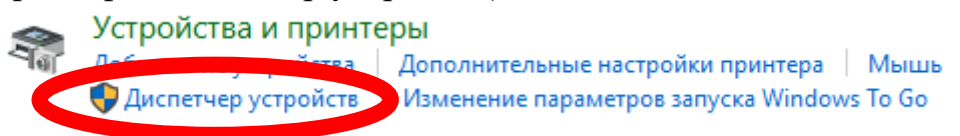


Главное их отличие - это программатор (выделен красным).

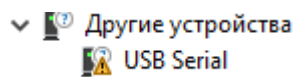
Если у вас оригинальная плата, можно переходить [к первому проекту](#).
При установке Arduino IDE установились необходимые драйверы.

Если у вас китайская копия, то необходимо самостоятельно установить нужные драйверы, то для дальнейшей работы необходимо установить драйвер программатора CH341.

Для начала нужно проверить как плата отображается для компьютера. Открываем диспетчер устройств (Пуск-Паннель управления-Устройства и принтеры- Диспетчер устройств)



Если имеется следующее предупреждение:

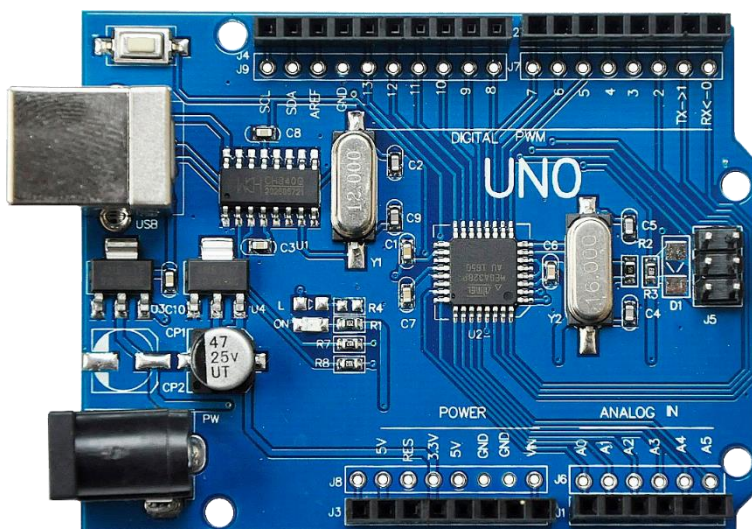


Значит у вас не установлен драйвер. [Инструкция по установке драйвера](#)

Установка драйверов

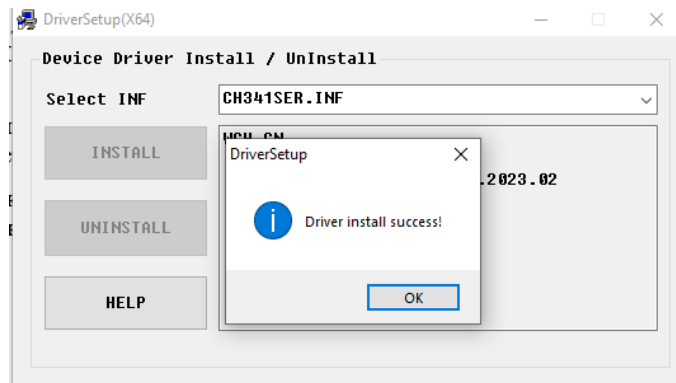
Установка производится на операционную систему семейства Windows, установка на другие операционные системы в методических рекомендация не рассматривается

Если ваша плата Arduino UNO выглядит, так:



По данной [ссылке](#) можно попасть на официальный сайт производителя, а по этим ссылкам можно скачать нужный драйвер под вашу операционную систему: [Windows](#), [Linux](#), [MacOS](#).

Открываем скачанный файл, подключаем Arduino UNO к компьютеру и нажимаем на кнопку “INSTALL”. Если появилось данное сообщение



То установка драйвера завершена, можно переходить [к первому проекту](#)

Установка библиотек через менеджер библиотек

Данная глава расскажет, как работать с менеджером библиотек в Arduino IDE



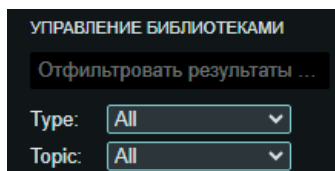
Менеджер библиотек находится в этой вкладке ->

Он позволяет устанавливать библиотеки, одобренные модераторами и проверенными на работоспособность.

В поисковой строке можно ввести нужную библиотеку, фильтрами можно сузить круг поисков

Type фильтр по признакам (не установлена/установлена, официальная/партнерская библиотека и т.д.)

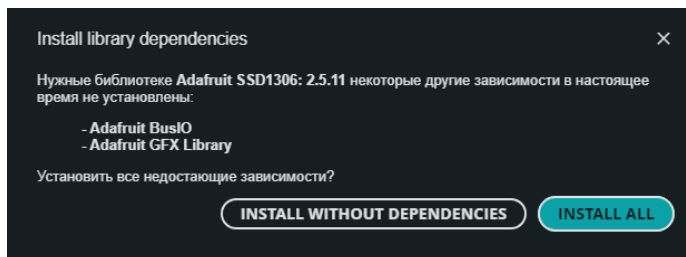
Topic фильтр по категориям (коммуникация, дисплеи, сенсоры и т.д.)



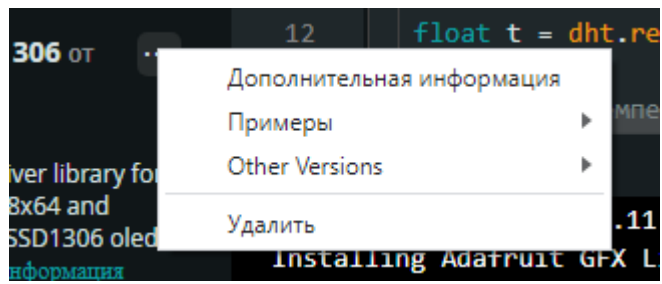
Для установки библиотеки необходимо нажать кнопку “УСТАНОВКА”



Некоторые библиотеки имеют зависимости от других библиотек, в таком случае будет предложено установить зависимости для этой библиотеки (Рекомендуется устанавливать все зависимости во избежание конфликтов)



При нажатии кнопки “Дополнительные действия” можно посмотреть дополнительную информацию (переход на GitHub библиотеки), примеры использования библиотеки, установить старые версии библиотеки (не рекомендуется), а также удалить библиотеку, если таковая не нужна.

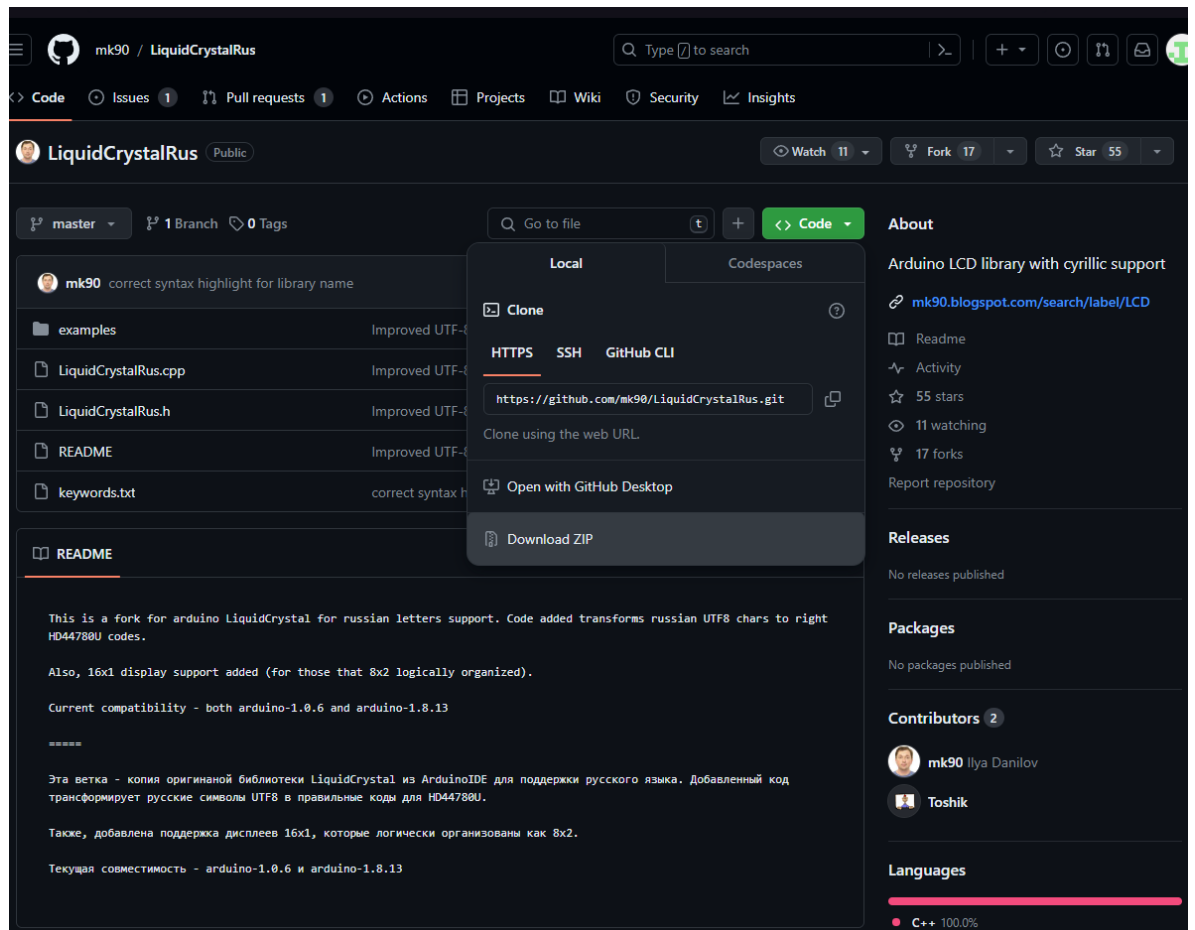


Установка сторонних библиотек

Если необходимая библиотека отсутствует в встроеном менеджере библиотек, то данная глава расскажет, как установить сторонние.

Для начала потребуется ссылка на необходимую библиотеку (будем устанавливать [LiquidCrystalRUS](https://github.com/mk90/LiquidCrystalRus))

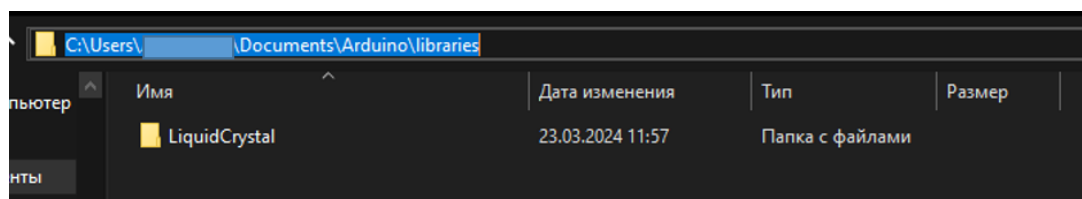
Скачайте библиотеку GitHub



Скачанный архив необходимо распаковать в папку с библиотеками.

Обычно она находится по следующему адресу

C:\Users\[USERNAME]\Documents\Arduino\libraries



Перекидываем папку с библиотекой из архива в папку **libraries**

Сторонняя библиотека установлена

Проверка работоспособности

Начнем с простого. Попробуем помигать светодиодом.

Проект №1

Сложность: легко

Задача: напишем код, который будет мигать светодиодом раз в 5 секунд.

Приступаем к написанию кода.

Схема проекта:

Схемы не будет, ведь мы моргаем встроенным светодиодом :)

Код проекта:

```
1  int LED = 13; // LED это переменная для светодиода, пин подключения 13
2  | | | | | // ( к 13 пину уже подключен светодиод на плате)
3
4  void setup() { // код в setup() выполняется один раз
5  |   pinMode(LED, OUTPUT); // Режим работы пина 13 - выход сигнала
6  | }
7
8  void loop() { // код в loop() будет выполняться "по кругу", пока есть питание на Arduino
9  |   digitalWrite(LED, HIGH); // Зажигаем светодиод
10 |   delay(5000);             // ждем 5 секунд
11 |   digitalWrite(LED, LOW);  // Гасим светодиод
12 |   delay(5000);             // И снова ждем 5 секунд
13 | }
```

Данная кнопка позволяет проверить код на ошибки.



Если вы не выбрали плату, будет следующая ошибка.

```
Вывод
Missing FQBN (Fully Qualified Board Name)

Compilation error: Missing FQBN (Fully Qualified Board Name)
```

Выбираем плату Arduino Uno.



Проверяем код на ошибки ещё раз.

Данная строка означает, что ошибок в коде нет.

Скетч использует 936 байт (2%) памяти устройства. Всего доступно 32256 байт.
Глобальные переменные используют 9 байт (0%) динамической памяти, оставляя 2039 байт для локальных переменных. Максимум: 2048 байт.

Можно заливать скейч на arduino

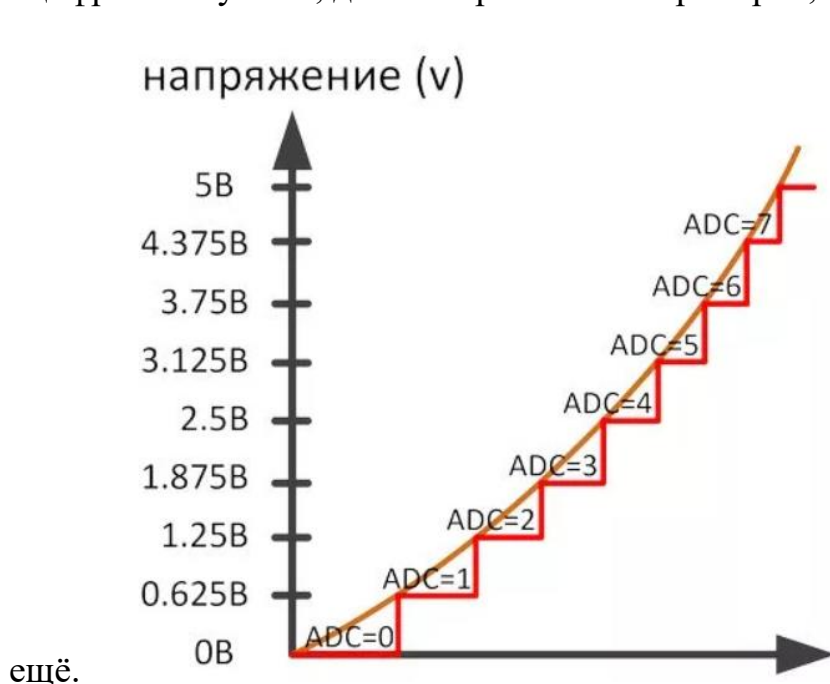
Не забывайте сохранять скейчи. Помните каждый скейч хранится в отдельной папке, название скейча и папки обязательно должно совпадать и не содержать кириллицы.

Аналогово – цифровой преобразователь

Теория

АЦП - это устройство преобразующее аналоговый сигнал в дискретный.

Основной принцип его работы основан на сравнении и преобразовании. Принцип его работы можно описать так: полученное аналоговое значение сравнивается с таблицей и возвращается в виде цифры, соответствующей определенному уровню сигнала. АЦП применяют для оцифровки музыки, для измерительных приборов, видеотехнике и много где



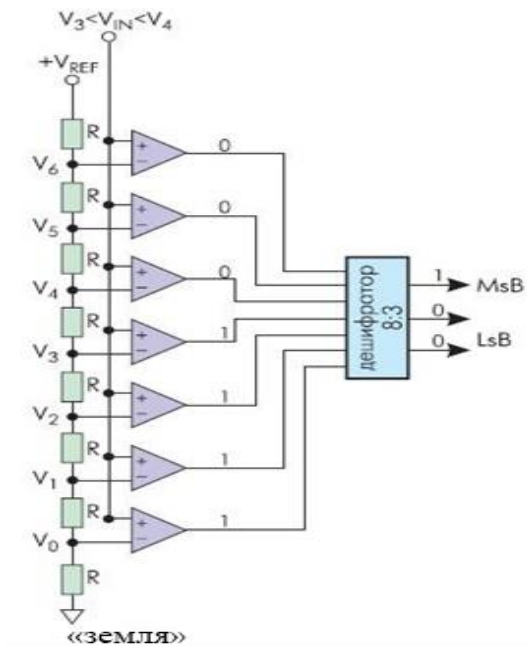
Существует несколько видов АЦП:

- 1) Параллельные прямого преобразования
- 2) Последовательного приближения
- 3) Дельта-Сигма

В рамках понимания внутреннего устройства, рассмотрим самый простой тип: Параллельно прямого преобразования.

В этом типе установлены компараторы, каждый из которых сравнивает напряжение с индивидуальным опорным напряжением, оно формируется за счет резисторов. Выход компараторов подключён к дешифратору, который уже и отправляет дискретный сигнал

Блок-схема ниже



Так же АЦП различаются по битности (максимальное количество символов для кодирования аналогового сигнала – чувствительность) и порогом измеряемых напряжений.

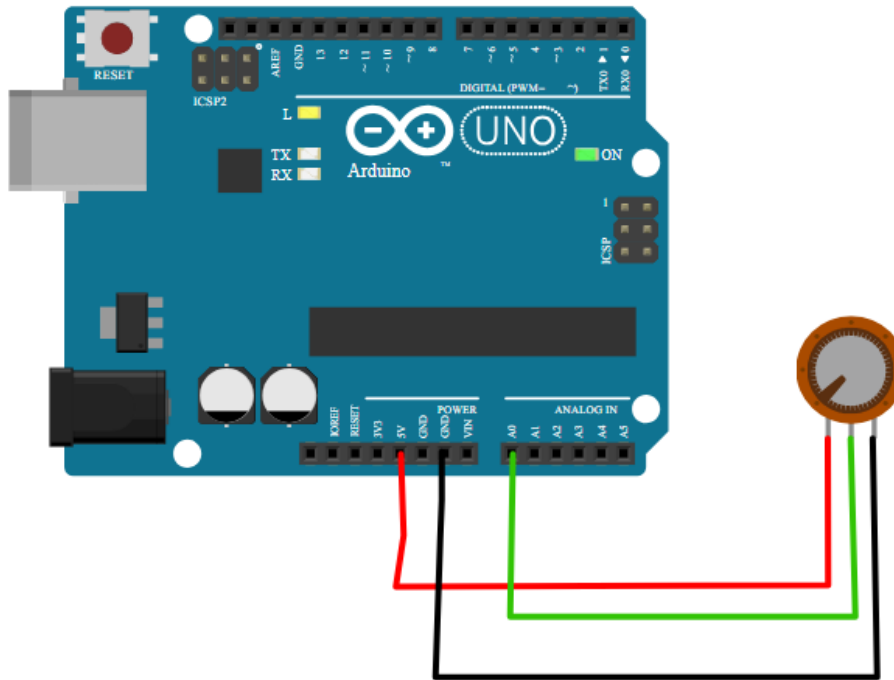
В ардуино установлен 10 битный ацп с порогом в 5 V, следовательно, максимальное цифровое значение 1023 будет достигнуто при 5 V (2^{10} , счёт начитается с 0).

Проект №2

Сложность: легко

Задача: выводим оцифрованное значение на компьютер и выполняем расчет напряжения по формуле.

Схема проекта:



Код проекта:

```
1  int analogPin = A0; // Создаем переменную под аналоговый пин
2  const float MAXvoltage = 5.0; // Создаем константу максимального доступного напряжения
3  void setup() {
4      Serial.begin(9600); // Подключаем связь с компьютером на скорости 9600
5  }
6
7  void loop() {
8      int analogValue = analogRead(analogPin); // Считываем значение с аналогового пина
9      Serial.print("Оцифрованное значение: "); // Выводим значение
10     Serial.println(analogValue);
11     float voltage = (analogValue / 1023.0) * MAXvoltage; // Значение в Вольтах, рассчитываем по формуле
12     Serial.print("Напряжение: "); // Выводим значение
13     Serial.println(voltage); // Выводим значение
14     delay(1000); // Задержка в 1 секунду
15 }
```

ЖК-дисплей MT-16S2H

[DATASHEET](#)

Теория

Жидкокристаллический дисплей (LCD) mt-16s2h имеет 16 столбцов и 2 строки (16/2). Собственный параллельный интерфейс, 2 режима работы (4 и 8 бит), имеет встроенные символы как на латинице, так и на кириллице. От разработчика нам досталось 16 пинов для управления дисплеем.

Информация о каждом из пинов приведена в таблице:

Вывод	Обозначение	Назначение вывода
1	GND	Земля
2	Vcc	Напряжение 5V
3	Vo	Управление контрастностью
4	A0	Адресный сигнал (Данные/Команда)
5	R/W	Запись/Чтение
6	E	Разрешение обращение к модулю (строб данных)
7	DB0	Шина данных (8-ми битный, младший 8-ми битный режим)
8	DB1	Шина данных (8-ми битный)
9	DB2	Шина данных (8-ми битный)
10	DB3	Шина данных (8-ми битный)
11	DB4	Шина данных (8-ми и 4-х битный, младший 4-х битный режим)
12	DB5	Шина данных (8-ми и 4-х битный)
13	DB6	Шина данных (8-ми и 4-х битный)
14	DB7	Шина данных (8-ми и 4-х битный)
15	+LED	+ Подсветки
16	-LED	- Подсветки

Кодирование символов происходит с помощью 16 – ричной систем е исчисления. Но шифровка отображаемых символов (алфавит символов) и их запись в микроконтроллер происходит в уже при загрузке кода в Arduino.

Таблица символов можно найти в [DATASHEET](#) на странице 8

Проект №3

Сложность: нормально

Задача: Вывод сообщения на дисплей

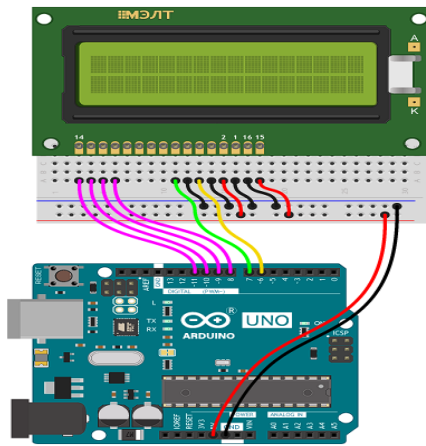
Установите стандартную библиотеку:

- [LiquidCrystal](#)

Необходимо установить стороннюю библиотеку:

- [LiquidCrystalRUS](#)

Схема проекта:



Код проекта:

Этот код выводит сообщение “Hello word!”

```
1 // Библиотека для работы с текстовым дисплеем
2 #include <LiquidCrystalRus.h>
3
4 // Задаём имя пинов дисплея в 4 битном режиме
5 constexpr uint8_t PIN_RS = 6;
6 constexpr uint8_t PIN_EN = 7;
7 constexpr uint8_t PIN_DB4 = 8;
8 constexpr uint8_t PIN_DB5 = 9;
9 constexpr uint8_t PIN_DB6 = 10;
10 constexpr uint8_t PIN_DB7 = 11;
11
12 /*
13  * Создаём объект для работы с текстовым дисплеем
14  * LiquidCrystalRus lcd(RS, EN, DB4, DB5, DB6, DB7);
15  */
16
17 LiquidCrystalRus lcd(PIN_RS, PIN_EN, PIN_DB4, PIN_DB5, PIN_DB6, PIN_DB7);
18
19 void setup() {
20     // Устанавливаем размер экрана
21     // Количество столбцов и строк
22     lcd.begin(16, 2);
23     // Устанавливаем курсор в колонку 5 и строку 0
24     lcd.setCursor(0, 0);
25     // Печатаем первую строку
26     lcd.print("Hello world!");
27 }
28
29 void loop() {
30 }
```

OLED дисплей 1306

Теория

Есть 2 версии: они отличаются интерфейсами взаимодействия (I²C и SPI). Так же есть различие в размере экранов (128*32 и 128*64 пикселей).

Рассмотрим дисплей с интерфейсом I2S и размером 128*64 пикселя.

I²C один из самых удобных интерфейсов

Всего 4 пина (5v, GND, SDA, SLC) и простота настройки делают его фаворитом среди интерфейсов.

Принцип работы:

Обмен осуществляется с помощью двух сигнальных линий: SDA — данные, и SCL — синхроимпульс (фактически, данные идут только по одному проводу, а второй нужен для правильной работы шины)

Формирование пакетов данных описан в рисунке ниже:

1) Сеанс передачи от "Master" к "Slave"



2) Сеанс передачи от "Slave" к "Master" (чтение из "Slave")



- S** "Старт" - условие
- P** "Стоп" - условие
- A** бит подтверждения (ACK)
- A** отсутствие подтверждения

Цветом ячейки обозначено - какое именно устройство выставляет данный бит на шину DATA:

- бит выставляется "Master" - устройством
- бит выставляется "Slave" - устройством

Проект №4

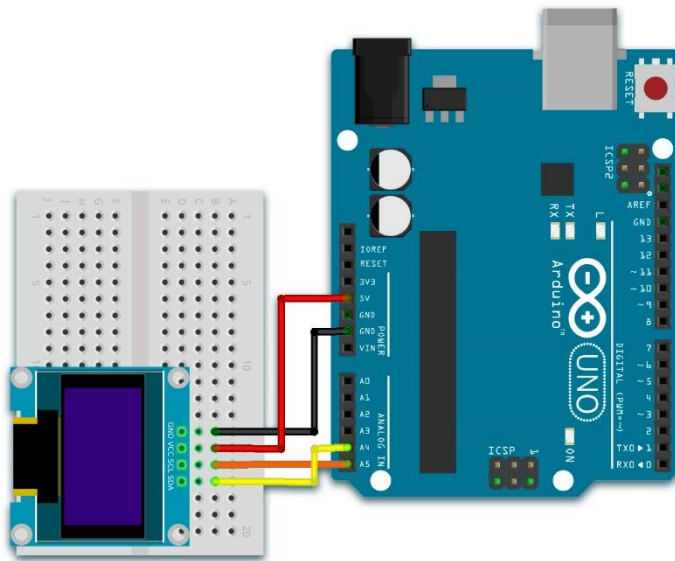
Сложность: нормально

Задача: выводим текст на экран

Для продолжения необходимо скачать библиотеки через менеджер библиотек:

- [Adafruit SSD1306](#)
- [Adafruit GFX Library](#)

Схема проекта:



Код проекта:

```
1  #include <Adafruit_GFX.h>           //OLED библиотека
2  #include <Adafruit_SSD1306.h>
3  #include <Wire.h>                   //Библиотека I2C
4
5  #define SCREEN_WIDTH 128 // Ширина OLED-дисплея, в пикселях
6  #define SCREEN_HEIGHT 64 // Высота OLED-дисплея в пикселях
7  #define OLED_RESET -1 // Т.к. у дисплея нет пина сброса прописываем -1, используем общий сброс Arduino
8
9  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //Объявляем имя и задаем параметры
10
11
12 void setup() {
13   display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //Запуск дисплея
14   display.display();
15   delay(2000); //Пауза для инициализации дисплея
16 }
17
18 void loop() {
19
20   display.clearDisplay(); //Очистить дисплей
21   display.setTextSize(2); //Задаем размер текста
22   display.setTextColor(WHITE); //Задаем цвет текста
23   display.setCursor(35,10); //Задаем координату начала текста в пикселях
24   display.println("HELLO!"); //Пишем сам текст
25   display.display(); //Команда для отображения всего этого на дисплее
26 }
```

Проект №5

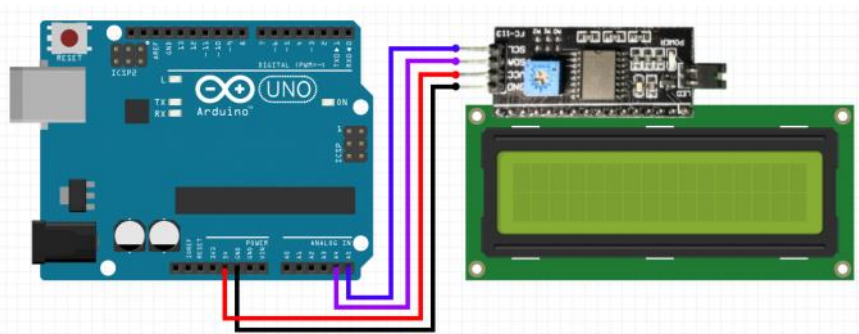
Сложность: нормально

Помните дисплей из предыдущего проекта? Теперь когда нам известно про протокол I²C, и можем использовать конвертор параллельного интерфейса в последовательный (что позволит сильно сэкономить пины)



Задача: Вывод текста на экран дисплея, по протоколу I²C.

Схема проекта:



Код проекта:

Необходимо установить стороннюю библиотеку:

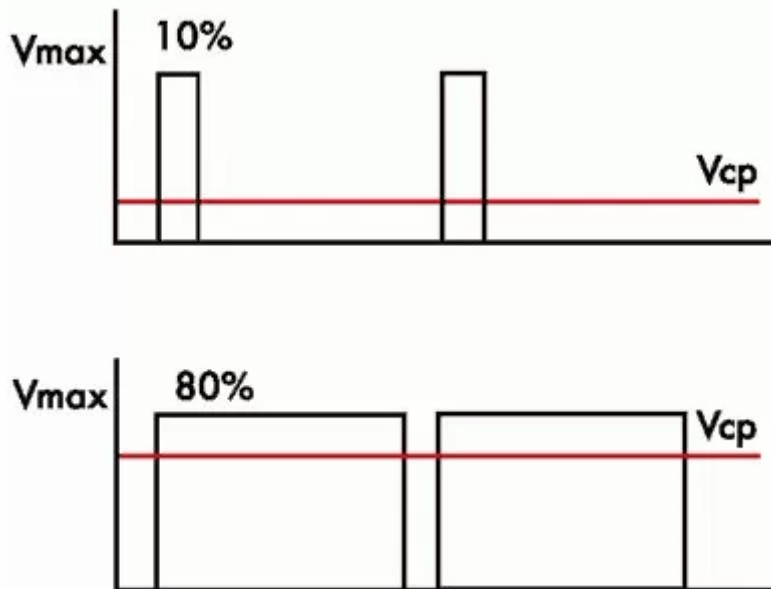
- [LCD_1602_RUS_ALL](#)

```
1  #define _LCD_TYPE 1 // 0 если по параллельному интерфейсу / 1 для работы с I2C
2  #include <LCD_1602_RUS_ALL.h> // подключаем библиотеку
3  LCD_1602_RUS lcd(0x27, 16, 2); // назначаем адрес устройства и количество символов/строк
4
5  void setup() {
6      lcd.init(); // инициализируем дисплей
7      lcd.backlight(); // включаем подсветку
8      lcd.setCursor(0, 0); // курсор в первый символ первой строки
9      lcd.print("Привет!"); // вывод строки
10     lcd.setCursor(0, 1); // курсор в первый символ второй строки
11     lcd.print("Hello!"); // вывод строки
12 }
13
14 void loop() {
15 }
```

Широтно-импульсная модуляция

Теория

ШИМ сигнал - это тип цифрового сигнала, который модулируется для управления мощностью, скоростью и/или положением устройств в автоматизированной системе.



ШИМ имеет несколько критически важных характеристик, которые необходимо учитывать для корректной работы исполняемого устройства.

- 1) Период тактирования — промежутки времени, через которые подаются импульсы.
- 2) Длительность импульса — время пока подается сигнал.
- 3) Скважность — рассчитанное по формуле соотношение длины импульса к импульсному T периоду тактирования.
- 4) Коэффициент заполнения — показатель обратный скважности

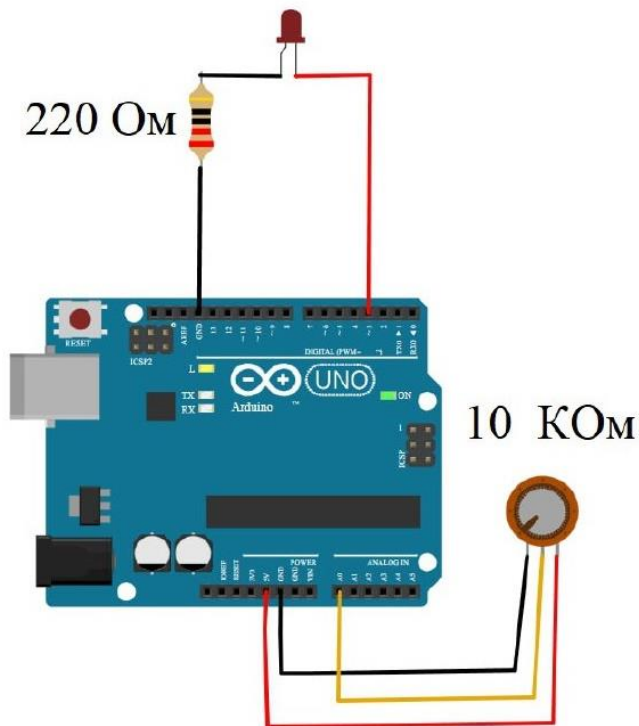
ШИМ генератор на Arduino 8 битный (Скважность регулируется от 0 до 255) и подключен он не ко всем пинам (это нужно учитывать)

Проект №6

Сложность: легко

Задача: Используя ШИМ сигнал попробуем регулировать яркость светодиода при помощи потенциометра и ШИМ сигнала.

Схема проекта:



Код проекта:

```
1  int pwm;
2
3  void setup(){
4
5  }
6  void loop(){
7      pwm = analogRead(A0); // считываем значение с потанциометра
8      pwm = map(pwm, 0, 1023, 0, 255); // конвертируем 10 разрядность в 8 разрядность
9      pwm = constrain(pwm, 0, 255); // обрезаем диапазон, для исключения выхода за рамки
10     analogWrite(3,pwm); // Выводим ШИМ сигнал на 3 пин
11 }
```


Сервопривод SG90

DATASHEET

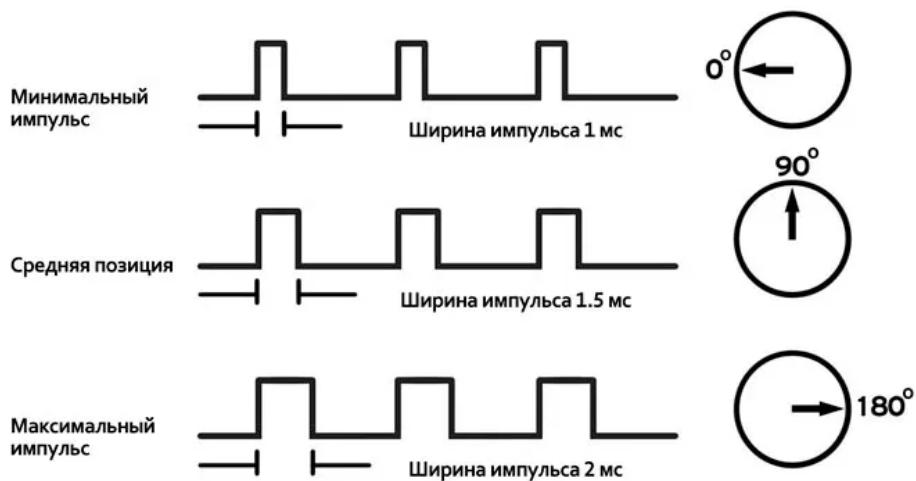
Теория

Сервопривод — это электродвигатель с блоком управления, который за счёт обратной связи может точно поддерживать заданное положение вала или постоянную скорость вращения.

Основные понятия в конструкциях сервоприводов:

- 1) Крутящий момент
- 2) Скорость поворота
- 3) Размер сервопривода
- 4) Интерфейс взаимодействия
- 5) Материал редуктора
- 6) Тип мотора

Что бы управлять сервоприводом нужно использовать ШИМ сигнал.



Проект №7

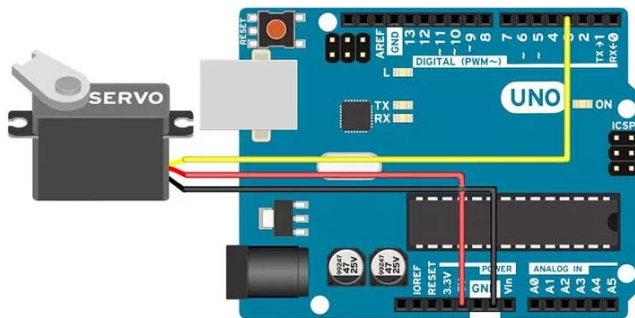
Сложность: легко

Задача: заставим сервопривод вставать на определенный угол 0-30-60-90-120-150-180

Для удобства работы с сервоприводом потребуется библиотека из менеджера библиотек:

- [Servo](#)

Схема подключения:



Код проекта:

```
1 // Подключаем библиотеку Servo
2 #include <Servo.h>
3
4 // Пин для сервопривода
5 int servoPin = 3;
6 // Создаем объект
7 Servo Servo1;
8
9 void setup() {
10   // Нам нужно подключить сервопривод к используемому номеру пина
11   Servo1.attach(servoPin);
12 }
13
14 void loop(){
15   Servo1.write(0); // 0 градусов
16   delay(1000);
17   Servo1.write(30); // 30 градусов
18   delay(1000);
19   Servo1.write(60); // 60 градусов
20   delay(1000);
21   Servo1.write(90); // 90 градусов
22   delay(1000);
23   Servo1.write(120); // 120 градусов
24   delay(1000);
25   Servo1.write(150); // 150 градусов
26   delay(1000);
27   Servo1.write(180); // 180 градусов
28   delay(1000);
29 }
```

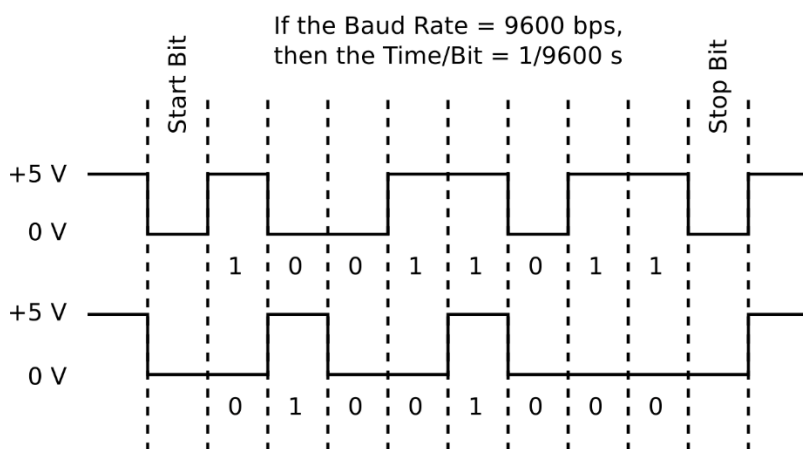
UART интерфейс. Bluetooth HC-06

DATASHEET

Теория

(Универсальный синхронный-асинхронный приемник-передатчик)

Протокол UART — старейший и самый распространенный на сегодняшний день физический протокол передачи данных. Наиболее известен из семейства UART протокол RS-232. Это, наверное, самый древний компьютерный интерфейс, который дожил до наших дней и не потерял своей актуальности.



Проект №8

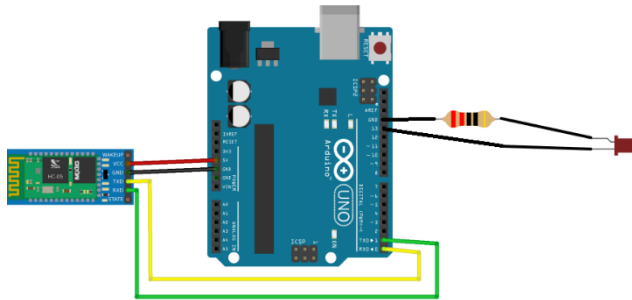
Сложность: легко

Задача: со смартфона, через Bluetooth, управлять светодиодом.

Если пришла “1”, то включаем светодиод.

Если пришёл “0”, то выключаем светодиод

Схема подключения:



Arduino	Bluetooth
Pin 1 (TX)	RXD
Pin 0 (RX)	TXD
GND	GND
5V	VCC

Код проекта:

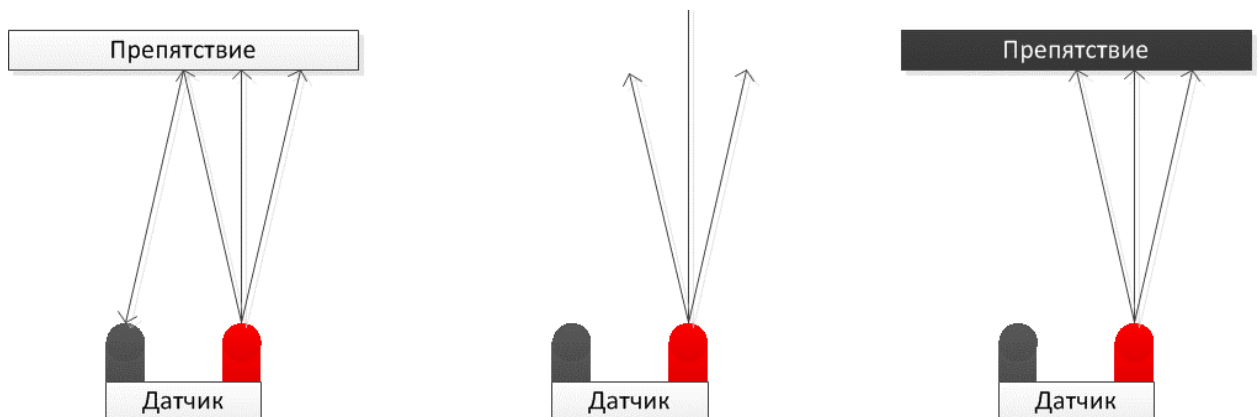
```
1  int val;
2  int LED = 13;
3  void setup()
4  {
5      Serial.begin(9600);
6      pinMode(LED, OUTPUT);
7      digitalWrite(LED, HIGH);
8  }
9  void loop()
10 {
11     if (Serial.available())
12     {
13         val = Serial.read();
14         // При символе "1" включаем светодиод
15         if (val == '1')
16         {
17             digitalWrite(LED, HIGH);
18         }
19         // При символе "0" выключаем светодиод
20         if ( val == '0')
21         {
22             digitalWrite(LED, LOW);
23         }
24     }
25 }
```

Датчик распознавания преград. YL-63

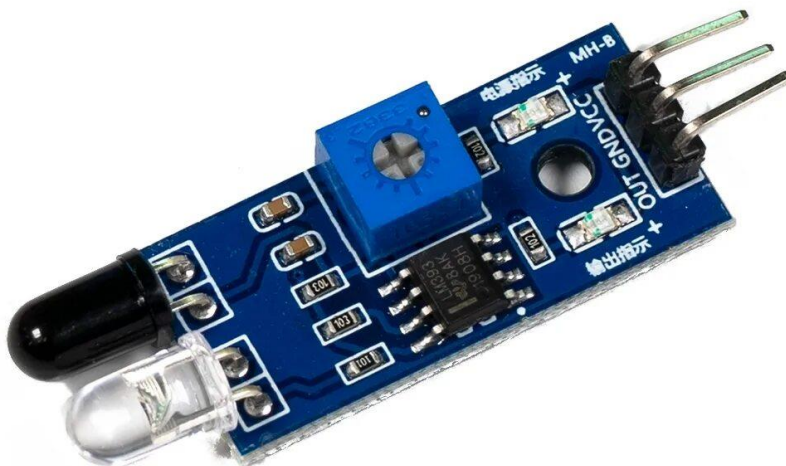
Теория

Модуль YL-63 одержит инфракрасный передатчик (ИК диод) излучающий свет в прямом направлении и приемник (фотодиод), который измеряет отраженное ИК излучение. Если отраженный свет достигает определенного порога на выходе появляется положительный импульс.

Если поверхность белая, то модуль сработает на максимальном расстоянии, если темная или матовая излучение не отразится и модуль не сработает.



Основная микросхема ИК датчика препятствия, это компаратор LM393, который производит сравнение уровней напряжений на входах INB- и INB+. Чувствительность порога срабатывания задается с помощью потенциометра и в результате сравнений на выходе OUTB микросхемы, формируется логический «0» или логическая «1».

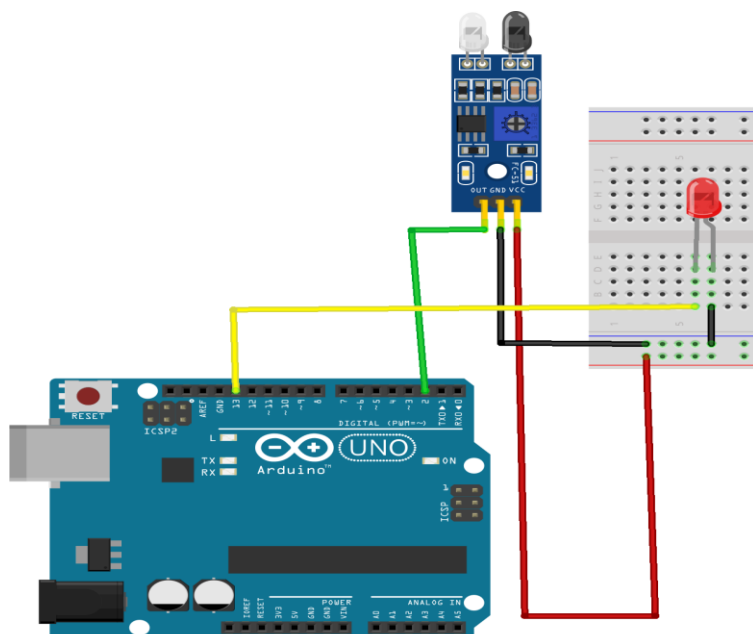


Проект №9

Сложность: легко

Задача: при обнаружении препятствия – загорается светодиод

Схема проекта:



Код проекта:

```

1  const int svPin = 2; // пин данных с датчика
2  const int ledPin = 13; // пин для светодиода
3  int svState = 0;      // переменная для хранения данных
4
5  void setup() {
6
7      pinMode(ledPin, OUTPUT); //светодиод
8
9      pinMode(svPin, INPUT); // прием данных
10 }
11
12 void loop() {
13
14     svState = digitalRead(svPin); // считываем значение
15
16     if (svState == 0) { //
17         digitalWrite(ledPin, HIGH);
18     } else {
19         digitalWrite(ledPin, LOW);
20     }
21 }
22

```

Термодатчик на терморезисторе. MF52 NTC-MF52AT 3950B

Теория

Терморезистор — это резистор, который меняет свое электрическое сопротивление в зависимости от температуры.

Существует два вида термисторов: РТС — с положительным температурным коэффициентом, и NTC — с отрицательным. Положительный коэффициент означает, что с повышением температуры сопротивление термистора растёт. NTC-термистор ведет себя противоположным способом.

Чтобы вычислить значение температуры используют формулу Стейнхарта — Харта:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$

Уравнение имеет параметры А, В и С, которые нужно брать из спецификации к датчику. Но в нашем случае не требуется большой точности, воспользуемся модифицированным уравнением (В-уравнение):

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

В этом уравнении неизвестным остается только параметр В, который для NTC термистора равен 3950. Остальные параметры нам уже известны:

- T_0 — комнатная температура в Кельвинах, для которой указывается номинал термистора; $T_0 = 25 + 273.15$;
- T — искомая температура, в Кельвинах;
- R — измеренное сопротивление термистора в Омах;
- R_0 — номинальное сопротивление термистора в Омах.

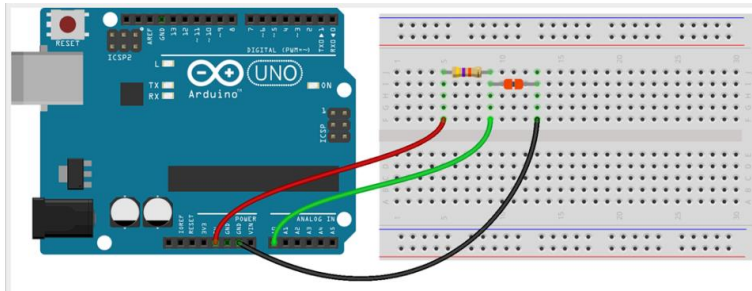
Проект № 10

Сложность: нормально

Задача: Измерение температуры через терморезистор.

Используется терморезистор с номинальным сопротивлением 5 КОм и резистор на 100 Ом.

Схема проекта:



Код проекта:

```
1  #define B 3950 // B-коэффициент
2  #define SERIAL_R 90 // сопротивление последовательного резистора, 100 Ом
3  #define THERMISTOR_R 4350 // номинальное сопротивления термистора, 5 кОм
4  #define NOMINAL_T 25 // номинальная температура (при которой TR = 1-00 кОм)
5
6  const byte tempPin = A0;
7
8  void setup() {
9      Serial.begin( 9600 );
10     pinMode( tempPin, INPUT );
11 }
12
13 void loop() {
14     int t = analogRead( tempPin );
15     float tr = 1023.0 / t - 1;
16     tr = SERIAL_R / tr;
17     Serial.print("Сопротивление=");
18     Serial.print(tr);
19     Serial.print(", Температура=");
20
21     float steinhart;
22     steinhart = tr / THERMISTOR_R; // (R/Ro)
23     steinhart = log(steinhart); // ln(R/Ro)
24     steinhart /= B; // 1/B * ln(R/Ro)
25     steinhart += 1.0 / (NOMINAL_T + 273.15); // + (1/To)
26     steinhart = 1.0 / steinhart; // Invert
27     steinhart -= 273.15;
28     Serial.println(steinhart);
29     delay(100);
30 }
```

Датчик температуры. DS18B20

DATASHEET

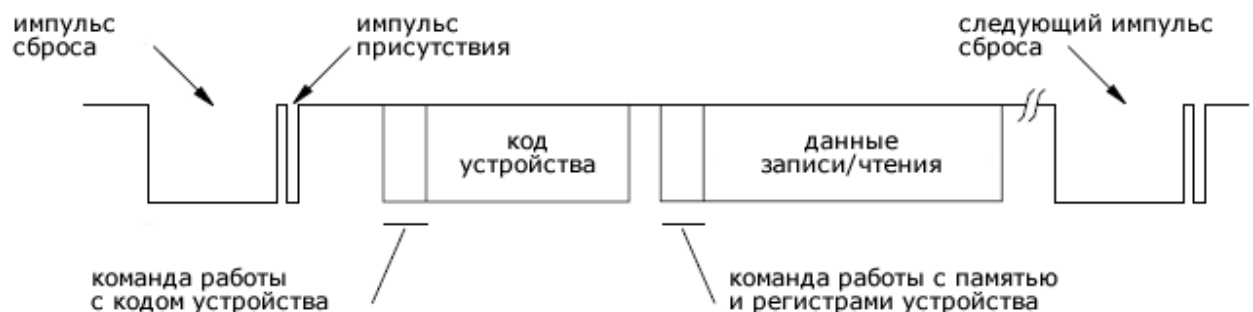
Теория

DS18B20 – это цифровой термометр, способный измерять температуру в диапазоне от -55 до +125 с программируемой точностью 9-12 бит. Каждому датчику присваивается свой уникальный 64-битный адрес, а обмен информацией платой Arduino осуществляется по шине 1-wire. Такой подход позволяет подключать к одной линии целую группу датчиков, вплоть до 264.

Принцип работы:

Обмен данными по магистрали включает три фазы:

- 1) Фаза сброса, включающую импульс сброса от контроллера и ответный импульс подтверждения присутствия от абонента (абонентов);
- 2) Фаза выборки устройства, включающую команду его выборки (по коду, без кода, групповую, поиска) и его код, если командой он предусмотрен;
- 3) Фаза записи/чтения данных, включающую код команды и данные.



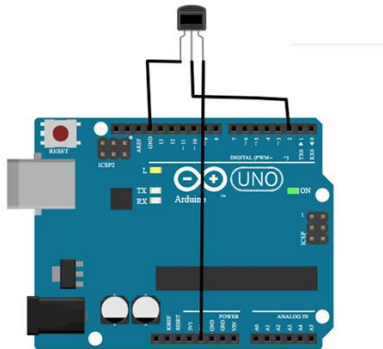
Протокол 1 Wire, нашел своё применение в домофонных ключах (iButton) или используется в системах умного дома для опроса большого количества датчиков.

Проект №11

Сложность: легко

Задача: Будем измерять температуру окружающей среды, а данные выводить в КОНСОЛЬ

Схема подключения:



Код проекта:

Необходимо установить следующие библиотеки:

- [One Wire.h](#)
- [Dallas Temperature.h](#)

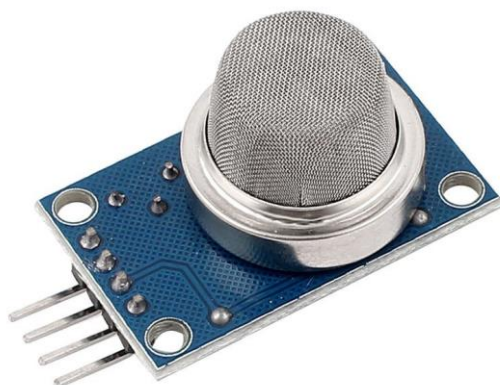
```
1 // Подключаем библиотеку
2 #include <OneWire.h>
3 #include <DallasTemperature.h>
4 // Определяем пин подключения данных
5 #define ONE_WIRE_BUS 2
6
7 // Запускаем протокол oneWire
8 OneWire oneWire(ONE_WIRE_BUS);
9 // Определяем тип датчика
10 DallasTemperature sensors(&oneWire);
11
12 void setup(void)
13 {
14     // Включаем передачу данных через консоль
15     Serial.begin(9600);
16     Serial.println("Hello user!");
17     sensors.begin(); //запускаем счет данных с датчика
18 }
19 void loop(void)
20 {
21     Serial.print("Получаю адрес датчика");
22     sensors.requestTemperatures(); // Получаем адрес датчика
23     Serial.println("Готово");
24     float tempC = sensors.getTempCByIndex(0); // Присваиваем переменной значение
25
26     // Check if reading was successful
27     if(tempC != DEVICE_DISCONNECTED_C)
28     {
29         Serial.print("Температура ");
30         Serial.println(tempC);
31     }
32     else
33     {
34         Serial.println("Ошибка");
35     }
36 }
```


Датчик дыма. MQ-2

[DATASHEET](#)

Теория

Датчик MQ-2 позволяет выявлять в воздухе минимальную концентрацию водорода и углеводородных газов (пропан, метан, бутан). Применяют сенсоры MQ-2 в проектах умного дома для своевременного обнаружения газа или дыма.



Принцип сенсора основан на детекторе, изготовленного из сплава оксида олова и алюминия, который в процессе работы сенсора существенно нагревается. В результате химической реакции, происходящей при попадании молекул углеводородных газов на чувствительный элемент, изменяется сопротивление сенсора. Измеряя изменения сопротивления, можно узнать точное значение концентрации газа в воздухе.

Напряжение аналогового выхода изменяется пропорционально концентрации дыма или газа. Чем выше концентрация газа, тем выше выходное напряжение. Логический сигнал можно откалибровать, держа датчик рядом с дымом, который вы хотите обнаружить. Далее вращайте потенциометр по часовой стрелке (для увеличения чувствительности сенсора), пока не загорится светодиод на модуле.

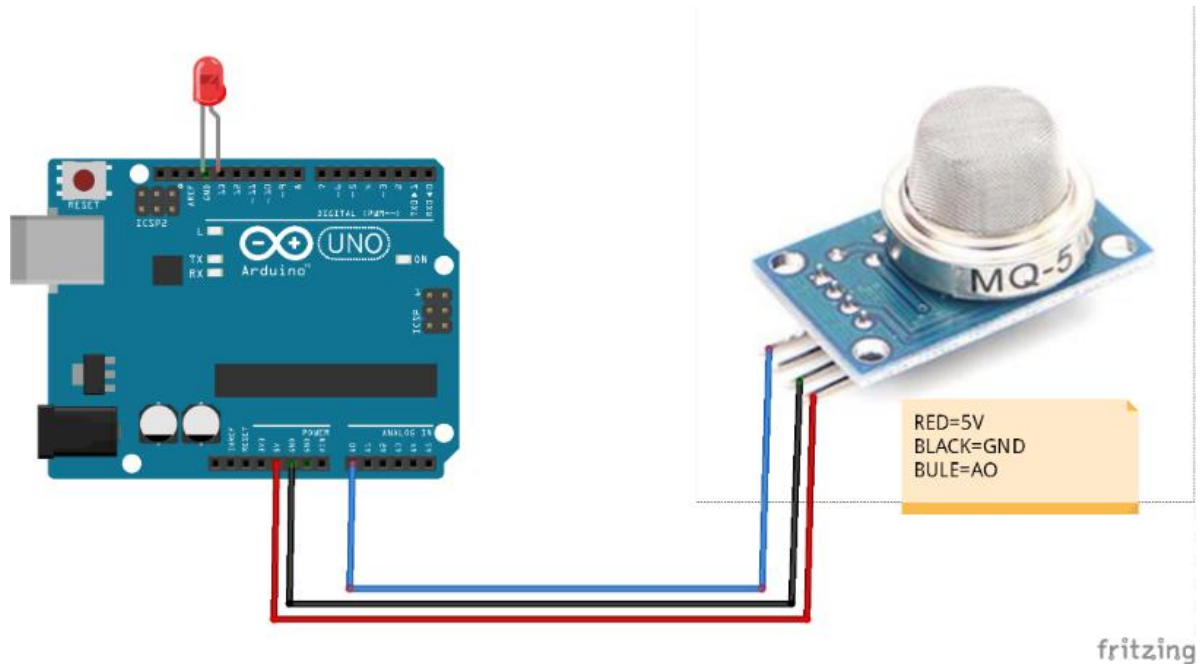
Проект №12

Сложно: нормально

Оповещаем о пожаре.

Задача: при срабатывании датчика дыма загорается светодиод.

Схема подключения:



Код проекта:

```
1  const int gasSensorPin = A0;      // Пин, к которому подключен аналоговый выход датчика газа
2  const int digitalOutPin = 2;      // Пин, к которому подключен цифровой выход датчика газа
3
4  void setup() {
5
6      pinMode(gasSensorPin, INPUT);  // Установка пина аналогового входа как вход
7      pinMode(digitalOutPin, INPUT); // Установка пина цифрового входа как вход
8      Serial.begin(9600);           // Начало работы с последовательным портом на скорости 9600 бит/сек
9  }
10
11 void loop() {
12     int analogValue = analogRead(gasSensorPin); // Считывание аналогового значения с датчика газа
13     int digitalValue = digitalRead(digitalOutPin); // Считывание цифрового значения с датчика газа
14
15
16
17     if (digitalValue == HIGH) {      // Проверка цифрового значения
18         Serial.println("Smoke: Not detected"); // Вывод сообщения в монитор последовательного порта
19     } else {
20         Serial.println("Smoke: Detected!");    // Вывод сообщения в монитор последовательного порта
21     }
22 }
```

Датчик протечки воды

Теория

Датчик протечи воды можно сделать на Arduino, без дополнительных модулей.

Логика работы: два контакта, опускаются в воду, на одном контакте 5 вольт, на другом контакт от АЦП (необходимо подтянуть через резистор к GND)

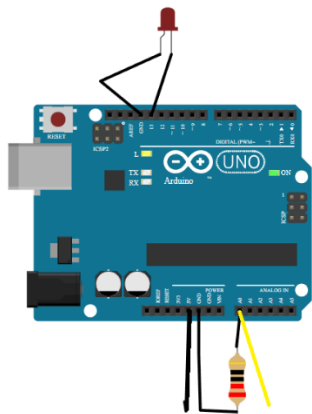
Вода проводит электричество, следовательно, контакты замкнутся.

Проект №13

Сложность: легко

Задача: при протечке воды загорается светодиод.

Схема проекта:



Код проекта:

```
1  int Water = A0;
2  int LED = 13;
3  int waterLvl;
4  void setup() {
5    // put your setup code here, to run once:
6    pinMode(Water, INPUT);
7    pinMode(LED, OUTPUT);
8  }
9
10 void loop() {
11   // put your main code here, to run repeatedly:
12   waterLvl = analogRead(Water);
13   if (waterLvl > 800){ //При достижении порога срабатывания, зажигем светодиод
14     digitalWrite(LED, HIGH);
15   }
16   else{
17     digitalWrite(LED, LOW);
18   }
19 }
```

Датчик освещённости на фоторезисторе

Теория

Фоторезистор — это полупроводниковый прибор, сопротивление (проводимость) которого изменяется в зависимости от уровня освещенности чувствительной части изделия.

Между двумя токопроводящими электродами размещается полупроводник. В том случае если свет не попадает на полупроводник, то его сопротивление имеет высокое значение. Как только на полупроводник попадает свет, его сопротивление снижается.



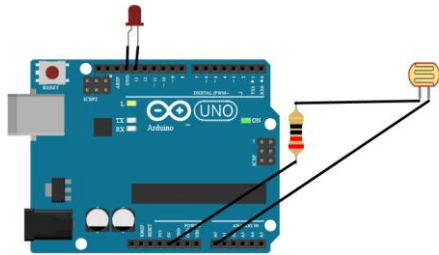
Для производства полупроводящего слоя могут использоваться следующие материалы: сульфид Кадмия, сульфид Свинца, Селенит Кадмия и т.п. От того какой материал был применен для производства полупроводника будет зависеть его спектральная характеристика (вид светового излучения, от которого работает фоторезистор).

Проект №14

Сложность: легко

Задача: включаем освещение в темноте. Если солнце светит светодиод, не горит, а если темно, солнца нет, то зажигаем светодиод.

Схема проекта:



Код проекта:

```
1  int Light = A0;
2  int LED = 13;
3  int LightLvl;
4  void setup() {
5      // put your setup code here, to run once:
6      pinMode(Light,INPUT);
7      pinMode(LED,OUTPUT);
8  }
9
10 void loop() {
11     // put your main code here, to run repeatedly:
12     LightLvl = analogRead(Light);
13     if (LightLvl > 500){ //При достижении порога срабатывания, зажигем светодиод
14         digitalWrite(LED,LOW);
15     }
16     else{
17         digitalWrite(LED,HIGH);
18     }
19 }
```

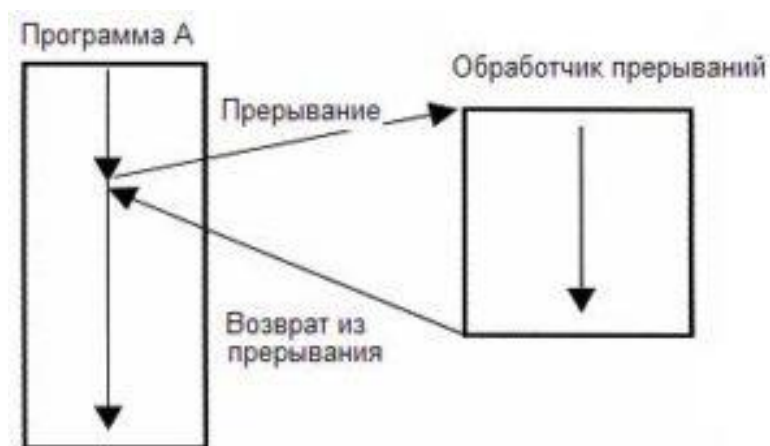
Внешнее аппаратное прерывание

Прерывание в Arduino можно описать так: микроконтроллер "всё бросает", переключается на выполнение функций в обработчике прерывания, выполняет их, а затем возвращается ровно к тому месту основного кода, в котором остановился.

Прерывания бывают разные, точнее их причины: прерывание может вызвать АЦП, таймер или пин МК¹.

External hardware interrupt — это прерывание, вызванное изменением напряжения на пине МК.

Основная суть: системное ядро микроконтроллера не занимается опросом пина и не тратит на это время. Но как только на пине меняется сигнал - микроконтроллер получает сигнал, бросает все дела, обрабатывает прерывание, и возвращается к работе.



Чаще всего прерывания используются для детектирования коротких событий - импульсов, или даже для подсчёта их количества, не нагружая основной код. Аппаратное прерывание может поймать короткое нажатие кнопки или срабатывание датчика во время сложных долгих вычислений или задержек в коде, т.е. грубо говоря - пин опрашивается параллельно основному коду. Также прерывания могут будить МК из режимов энергосбережения.

Рассмотрим основные правила работы с прерываниями:

У микроконтроллера есть возможность получать прерывания с любого пина, НО Ардуино умеет работать только с теми, которые приведены в таблице:

¹ МК – сокр. микроконтроллер

Микроконтроллер/Номер прерывания	INT 0	INT 1	INT 2	INT 3	INT 4	INT 5
Atmega 328/168 (UNO, Nano, Mini)	D2	D3	-	-	-	-
Atmega 32U (Leonard, Micro)	D3	D2			-	-
Atmega 2560 (Mega)	D21	D20	D19	D18	D2	D3

Прерывания имеют свой номер, который отличается от номера пина (Это важно!)

Для работы с прерыванием необходимо объявить функцию

```
void func() { // Название функции (func), может быть любым
}
```

Внутри функции, которую вызывает прерывание, есть некоторые ограничения:

- 1) Переменные, которые возвращают значения, должны быть объявлены как “volatile”
- 2) Не работают задержки delay()
- 3) Не меняют значения millis() и micros(), значение меняется между прерываниями
- 4) Не желательно использовать долгие вычисления (пример: операции с float)

Список команд для работы с прерываниями можно найти в [краткой справке](#)

Проект №15

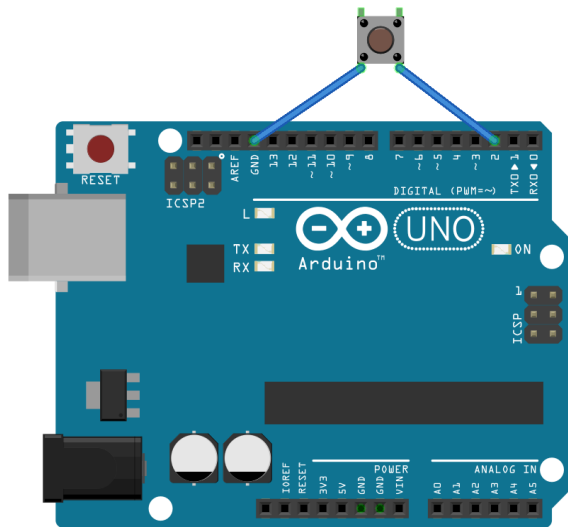
Сложность: легко

Прерывания очень удобно использовать, если в очень большом коде, для задания приоритета действиям или для “параллельного” выполнения задач.

В качестве примера “большого кода”, будет задержка на 5 секунд.

Задача: считаем количество нажатий на кнопку, выводим результат каждые 5 секунд.

Схема проекта:



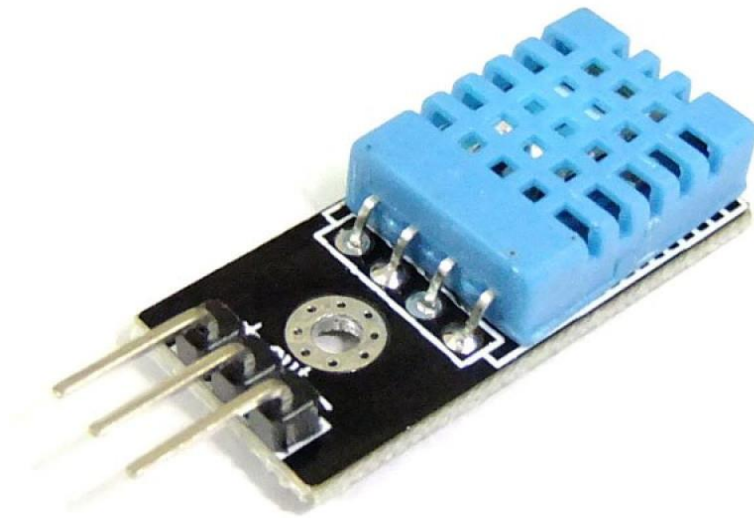
Код проекта:

```
1  volatile int counter = 0; // переменная-счётчик
2
3  void setup() {
4      Serial.begin(9600); // открыли порт для связи
5
6      pinMode(2, INPUT_PULLUP); // подключили кнопку на 2 пин и GND
7
8
9      attachInterrupt(0, btnIsr, FALLING); // FALLING - при нажатии на кнопку будет сигнал 0,
10 }
11
12 void btnIsr() {
13     counter++; // + нажатие
14 }
15
16 void loop() {
17     Serial.println(counter); // выводим
18     delay(5000);             // ждём
19 }
```


Датчик влажности. DH-11

DATASHEET

DHT11 — это цифровой датчик влажности и температуры, состоящий из термистора и емкостного датчика влажности. Также датчик содержит в себе АЦП для преобразования аналоговых значений влажности и температуры. Датчик DHT11 не обладает высоким быстродействием и точностью, но зато прост, недорог и отлично подходит для контроля влажности в помещении.



Если с термистором (терморезистором) мы уже работали ([Термодатчик на терморезисторе](#)), то емкостной датчик влажности, требует пояснений.

Емкостной датчик влажности — это конденсатор с переменной емкостью, который содержит токопроводящие обкладки из медной фольги на текстолите. Этот конденсатор заключен в герметичный чехол, поверх которого расположен влагопоглощающий слой. При попадании частиц воды на этот слой, меняется его диэлектрическая проницаемость, что приводит к изменению емкости конденсатора.

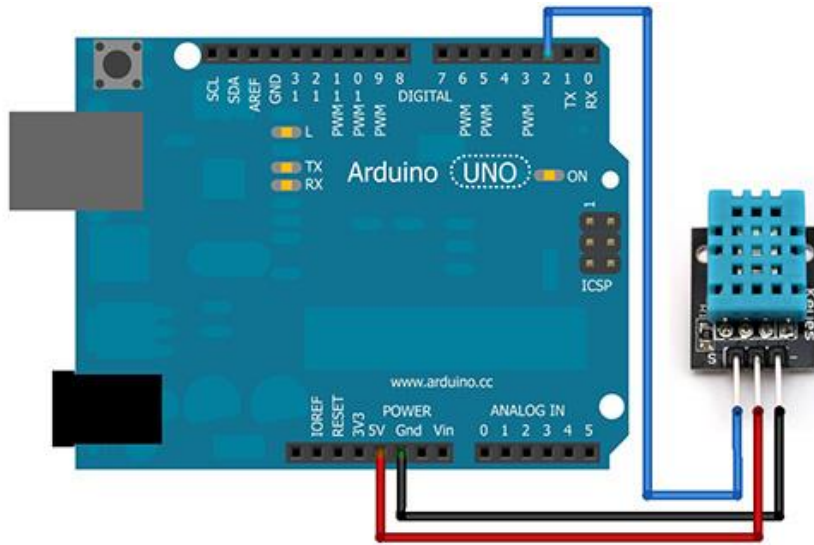
Проект №16

Сложность: легко

Узнаем температуру и влажность в помещении.

Задача: Вывод показаний влажности и температуры в монитор порта

Схема проекта:



Код проекта:

Необходимо установить стороннюю библиотеку:

- [DHT](#)
- [Adafruit_Sensor](#)

```
1  #include <DHT.h>           // подключаем библиотеку для датчика
2  DHT dht(2, DHT11);         // сообщаем на каком порту будет датчик
3
4  void setup() {
5      dht.begin();             // запускаем датчик DHT11
6      Serial.begin(9600);      // подключаем монитор порта
7  }
8
9  void loop() {
10     // считываем температуру (t) и влажность (h)
11     float h = dht.readHumidity();
12     float t = dht.readTemperature();
13
14     // выводим температуру (t) и влажность (h) на монитор порта
15     Serial.print("Влажность: ");
16     Serial.println(h);
17     Serial.print("Температура: ");
18     Serial.println(t);
19     delay(2000);
20 }
```

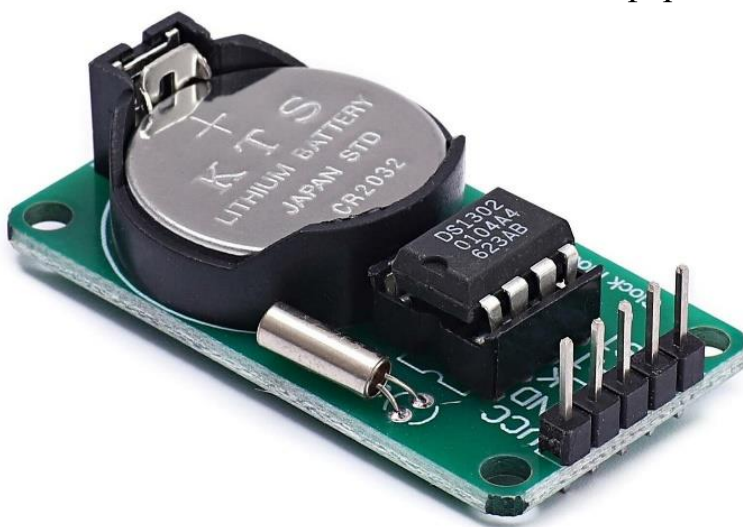
Часы реального времени. DS1302

DATASHEET

Real Time Clock или сокращенно RTC, это часы реального времени.

Используются в проектах, где нужно отслеживать время или для вывода времени для пользователя.

Микросхема DS1302 как раз этим и занимается. Связь с микроконтроллером осуществляется по 3-х проводному интерфейсу. Умеет поддерживать секунды, минуты, часы, день недели, дата, месяц и год, а также следит за количеством дней в месяце и делает поправку на високосный год. Работает в 24-часовом или 12-часовом формате с индикатором AM/PM.



Одной из причин использовать отдельную микросхему счета времени – это банальное энергопотребление. Выгоднее заставить заниматься этим отдельную микросхему с малым потреблением тока, чем оставлять включённым целый прибор.

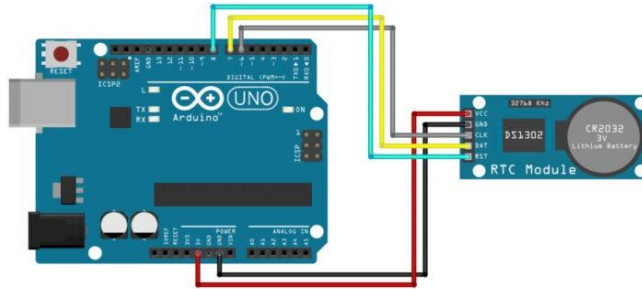
Проект №17

Сложность: легко

Подскажите время?

Задача: Вывод даты и времени в монитор порта.

Схема проекта:



Код проекта:

Необходимо установить библиотеку из менеджера библиотек:

- [RTC by Makuna](#)

```
1  #include <ThreeWire.h> // Подключаем библиотеку ThreeWire
2  #include <RtcDS1302.h> // Подключаем библиотеку RtcDS1302
3
4  ThreeWire myWire(7,6,8); // Указываем вывода IO, SCLK, CE
5  RtcDS1302<ThreeWire> Rtc(myWire);
6
7  void setup ()
8  {
9      Serial.begin(9600); // Установка последовательной связи на скорости 9600
10     Serial.print("Дата: "); // Отправка данных на последовательный порт
11     Serial.println(__DATE__); // Получение даты и времени с ПК
12     Serial.print("Время: "); // Отправка данных на последовательный порт
13     Serial.println(__TIME__); // Получение даты и времени с ПК
14     Rtc.Begin(); // Инициализация RTC
15     RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__); // Копирование даты и времени в compiled
16     Rtc.SetDateTime(compiled); // Установка времени
17     Serial.println(); // Отправка данных на последовательный порт
18 }
19
20 void loop ()
21 {
22     Rtc.GetDateTime();
23     RtcDateTime now = Rtc.GetDateTime();
24     Serial.print("Дата RTC "); // Отправка данных на последовательный порт
25     Serial.print(now.Month()); // Отправка месяца
26     Serial.print("."); // Отправка данных на последовательный порт
27     Serial.print(now.Day()); // Отправка дня
28     Serial.print("."); // Отправка данных на последовательный порт
29     Serial.print(now.Year()); // Отправка года
30     Serial.print(" Время RTC "); // Отправка данных на последовательный порт
31     Serial.print(now.Hour()); // Отправка часа
32     Serial.print(":"); // Отправка данных на последовательный порт
33     Serial.print(now.Minute()); // Отправка минут
34     Serial.print(":"); // Отправка данных на последовательный порт
35     Serial.println(now.Second()); // Отправка секунд
36     delay(2000); // Пауза 2 с
37 }
```

RFID связь. RFID 522 (MFRC522)

DATASHEET

Радиочастотная идентификация — это современная технология бесконтактной идентификации объектов, основанная на применении радиочастотного электромагнитного излучения малой мощности. Бесконтактная технология RFID используется для автоматизированного учёта и контроля объектов, снабженных специализированными миниатюрными устройствами - радиометками.

Одно из широко известных развитий технологии – NFC, это оплата покупок и подключение устройств. Так же охранные системы, СКУД, интерактивные предметы в квестах.

Рассмотрим принцип работы с RFID метками, а именно, с той которая идет в комплекте (MIFARE Classic 1K). Вся память разбита на сектора, а сектора на блоки. Всего секторов 16 по 4 блока каждый (64 блока по 16 байт каждый – 1 килобайт памяти). Все сектора, кроме нулевого, имеют одинаковое строение – 3 блока данных + 1 блок безопасности. Почти каждый блок может быть прочитан и перезаписан. Рисунок упрощает понимание.

MIFARE Classic 1K

Сектор	Блок	Содержимое
15	63	Блок безопасности
	62	16 Байт данных
	61	16 Байт данных
	60	16 Байт данных
...
01	07	Блок безопасности
	06	16 Байт данных
	05	16 Байт данных
	04	16 Байт данных
00	03	Блок безопасности
	02	16 Байт данных
	01	16 Байт данных
	00	Информация изготовителя

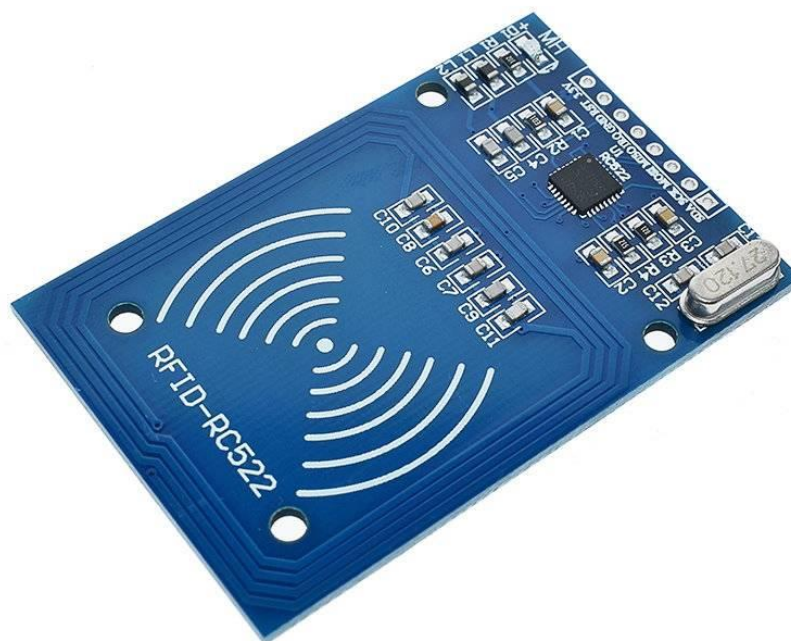
Нулевой блок хранит в себе уникальный идентификатор “UID”, тип метки и прочую информацию, записанную заводом-изготовителем. Нулевой сектор не может быть перезаписан, если речь идет о “классических” метках.

Таким образом UID позволяет отличить две с виду идентичные метки. UID как правило состоит из 4х байт, свободно считываемых из метки.

Важно!!! Существуют “перезаписываемые” метки, UID в которых можно менять, путем перезаписи нулевого блока. Если в вашей системе используется только UID – учтите возможность очень простого копирования UID в метки-болванки (в том числе злоумышленниками).

Блоком безопасности является каждый 4й блок, каждый блок безопасности отвечает за свой сектор (предыдущие 3 блока данных) – он хранит 2 ключа доступа по 6 байт (ключи А и В), а также специальные “Access bits” (Биты доступа). Ключи А и В могут быть использованы для аутентификации и последующего доступа к блокам данных в пределах сектора. То есть, для того чтобы получить доступ к любому из блоков внутри сектора необходимо “разблокировать” этот сектор, при помощи одного из ключей.

Карты имеют большой потенциал для работы, но в рамках методических рекомендаций, будет рассмотрено только работа с уникальными идентификаторами.



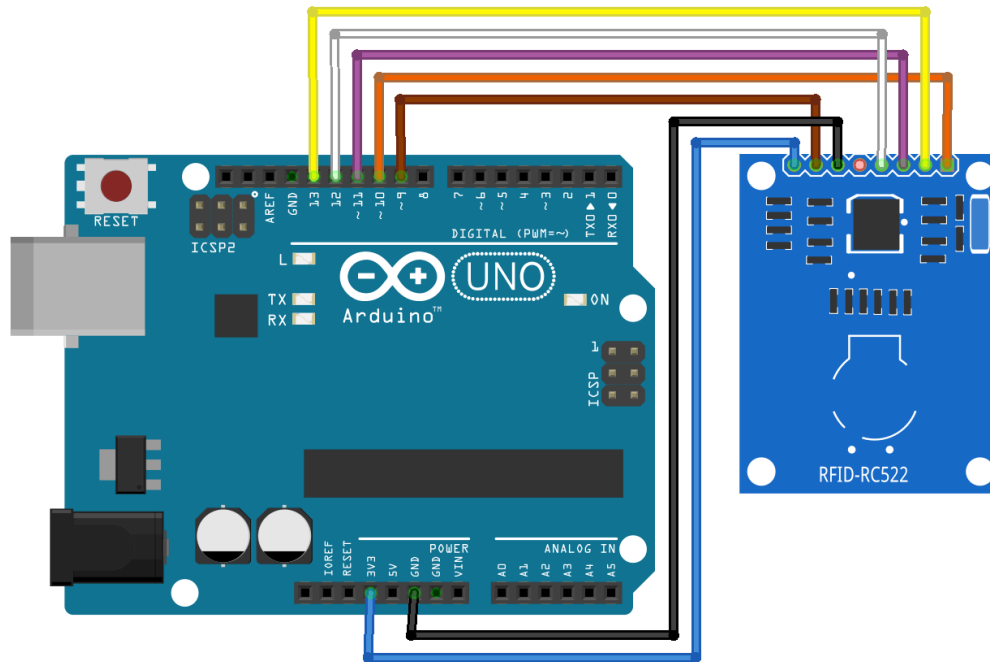
Проект №18

Сложность: нормально

Сканируем карточки на предмет уникальности.

Задача: считываем уникальный идентификатор карты при помощи модуля RFID 522.

Схема проекта:



Код проекта:

Необходимо установить библиотеку из менеджера библиотек:

- [MFRC522](#)

```
1 #include <SPI.h> // подключаем библиотеку
2 #include <MFRC522.h>
3
4 #define RST_PIN 9 // Reset пин
5 #define SS_PIN 10 // Пин синхронизации
6
7 MFRC522 mfrc522(SS_PIN, RST_PIN); // Передаем пины библиотеке
8
9 void setup() {
10     Serial.begin(9600); // Открываем последовательный порт
11     while (!Serial); // Ничего не делаем пока не открыли порт
12     SPI.begin(); // инициализируем SPI интерфейс
13     mfrc522.PCD_Init(); // инициализация модуля
14 }
15
16 void loop() {
17     if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) { //Если не приложена карта или невозможно прочитать,
18         // начать заново
19         return;
20     }
21     for (byte i = 0; i < mfrc522.uid.size; i++) { // Читаем и выводим UID
22         Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
23         Serial.print(mfrc522.uid.uidByte[i], HEX);
24     }
25     Serial.println();
26     mfrc522.PICC_HaltA(); //Завершаем работу с картой
27 }
28 }
```

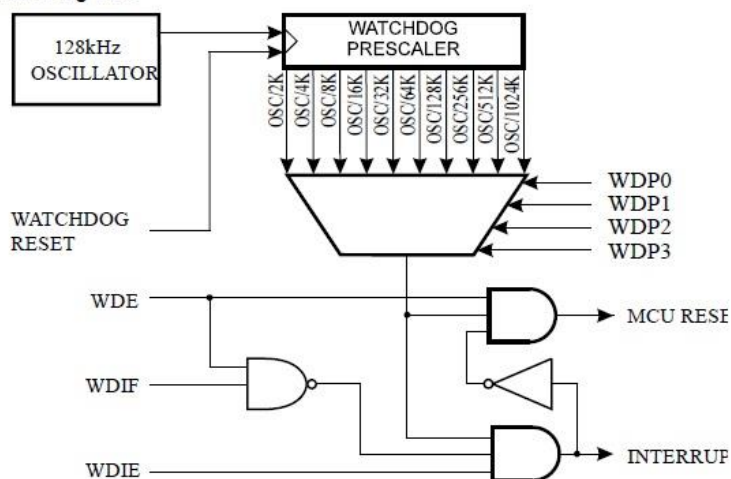

Сторожевой таймер. Watchdog

Обычные таймеры работают вместе с программой и тактируются от одной шины. Watchdog таймер тактируется независимо от других таймеров и способен работать вне программного цикла. Это сторожевой таймер (а если перевести дословно, «сторожевой пёс»), который представляет собой аппаратно-реализованную схему контроля над зависанием системы. Это таймер, который периодически сбрасывается контролируемой системой. Если вдруг сброса не произошло за определённый интервал времени после предыдущего сброса данного таймера, то происходит принудительная перезагрузка микроконтроллера.

Интервал работы Watchdog-таймера не бесконечен, а расположен в диапазоне от ~16 миллисекунд до ~ 8 секунд.

Таймер работает от собственного генератора, сигнал приходит на предделитель, который и содержит 10 различных выходов с различными коэффициентами. Подробнее на рисунке ниже:

Figure 15-7. Watchdog Timer



Поэтому интервал работы Watchdog-таймера не бесконечен, а расположен в диапазоне от ~15 миллисекунд до ~ 8 секунд.

Способы использования Watchdog-таймера:

- Аппаратная перезагрузка из-под программы;
- Обработка прерываний;
- Пробуждение контроллера из состояния сна

Проект №19

Сложность: легко

ВНИМАНИЕ!

Некоторые платы Arduino могут уходить в бесконечную перезагрузку из-за ошибки загрузчика. В качестве проекта будет использован код проверки работоспособности. При обнаружении зависания МК, необходимо загрузить код из [проекта 1](#).

Задача: Тестирование.

Схема проекта:

Схемы не будет, ведь мы проверяме внутреннюю периферию:)

Код проекта:

```
1  #include <avr/wdt.h> //подключаем встроенную библиотеку
2
3  void setup() {
4      wdt_disable(); // бесполезная строка до которой не доходит выполнение при цикличной загрузке
5      Serial.begin(9600);
6      Serial.println("Setup..");
7
8      Serial.println("Wait 5 sec..");
9      delay(5000); // Задержка, чтобы было время перепрошить устройство в случае цикличной перезагрузки
10     wdt_enable (WDTO_8S); // НЕ устанавливать значение менее 8 сек.
11     Serial.println("Watchdog enabled.");
12 }
13
14 int timer = 0;
15
16 void loop(){
17     // Каждую секунду мигаем светодиодом и значение счетчика пишем в Serial
18     if(!(millis()%1000)){
19         timer++;
20         Serial.println(timer);
21         digitalWrite(13, digitalRead(13)==1?0:1); delay(1);
22     }
23     // wdt_reset(); //команда отключения таймера watchdog
24 }
```


MP3 плеер. MP3-TF-16P

DATASHEET

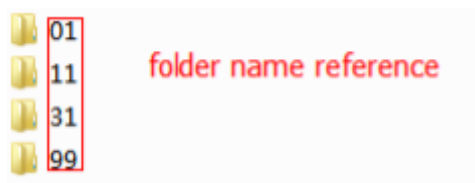
Модуль позволяет воспроизводить заранее записанную музыку на MiniSD карту (до 32GB и файловой системы FAT32 и eFAT). Питается от 3.3-5 V, интерфейс взаимодействия USART. Имеет встроенный усилитель, как для наушников, так и для динамика (MAX 3W). Есть возможность подключения резистивной клавиатуры. Воспроизведения файлов формата MP3 и WAV. Можно создать до 100 папок с 255 треками в каждой.

Для корректной работы файлы нужно правильно именовать.

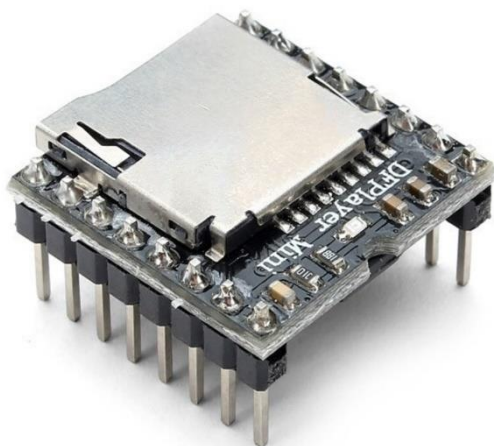
Название должно быть по формуле: [номер трека] _ [название трека]

Пример:  0007_korol-i-shut-lesnik.mp3

Папки же должны иметь следующий вид:



Модуль имеет сразу два вида выходного сигнала: первый - со встроенным усилителем, а второй вариант - через внешний усилитель.



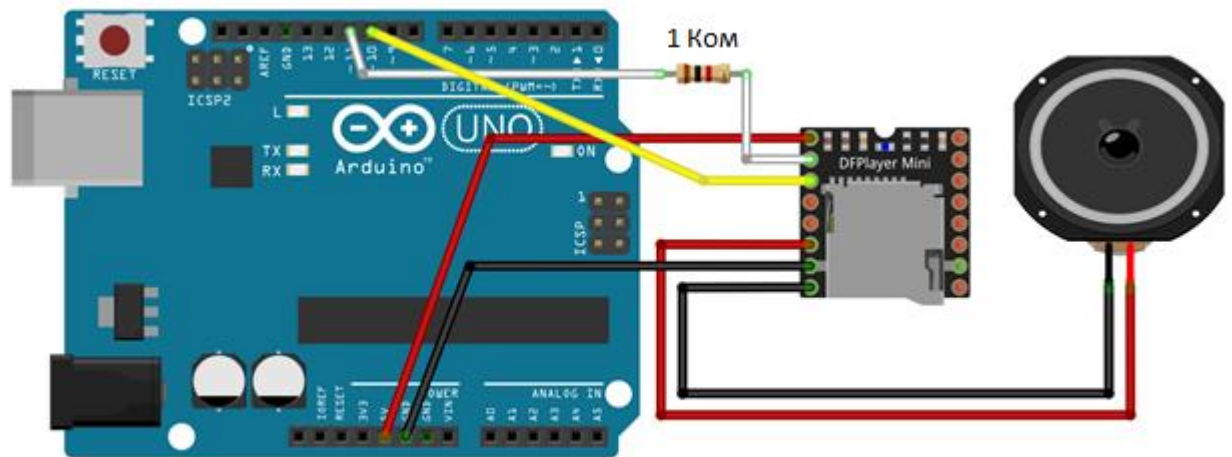
Проект №20

Сложность: легко

Танцуют все!

Задача: воспроизводим музыкальную композицию, записанную на Sd карту.

Схема проекта:



Код проекта:

Необходимо установить библиотеку из менеджера библиотек:

- [DFPlayerMini_Fast](#)

```
1  #include <SoftwareSerial.h>           // Создания программного UART
2  #include <DFPlayerMini_Fast.h>        // библиотека для MP3 плеера
3
4  SoftwareSerial mySerial(10, 11);       // Создаем контакты RX, TX
5  DFPlayerMini_Fast myMP3;              // Создаем объект для работы плеера
6  void setup() {
7      mySerial.begin(9600);              // Открываем последовательный порт
8      myMP3.begin(mySerial);             // инициализация
9      myMP3.volume(20);                  // Указываем громкость (0-30)
10     myMP3.play(1);                     // Воспроизведение 1 трек
11 }
12 void loop()
13 {
14     //В бесконечном цикле ничего не делаем
15 }
```

EEPROM

или жизнь после перезагрузки

«Если ты что-то записал в компьютерной памяти, запомни, где ты это записал.

Лео Бейзер

Может возникнуть нужда сохранить в большой объем данных, которые не должны меняться, например:

- Данные для калибровки
- Текст названий для меню
- Посчитанные данные
- Изображение для дисплея
- Константы

Хранение таких данных в оперативной памяти не лучшая идея. Изменяться такие данные не будут, а место займут. Хранение таких данных лучше доверить постоянной памяти. Для таких задач отлично подходит EEPROM.

EEPROM (англ. Electrically Erasable Programmable Read-Only Memory - электрически стираемое перепрограммируемое постоянное запоминающее устройство (ЭСПЗУ)), она же энергонезависимая память.

Если по простому, это память к которой мы имеем полный доступ из выполняющейся программы, т.е. пока работает программа, можно читать и писать туда данные, а после перезагрузки эти данные не сотрутся.

Где это можно использовать? Например:

- Хранение настроек, измененных “из меню” устройства, без перепрошивки
- Калибровка и сохранение калибровочных данных
- Использование в качестве расширения оперативной памяти, в случае нехватки
- “Чёрный ящик”, запись показаний с датчиков для расшифровки в случае провала
- Запись состояния работы, для восстановления работы после сбоя

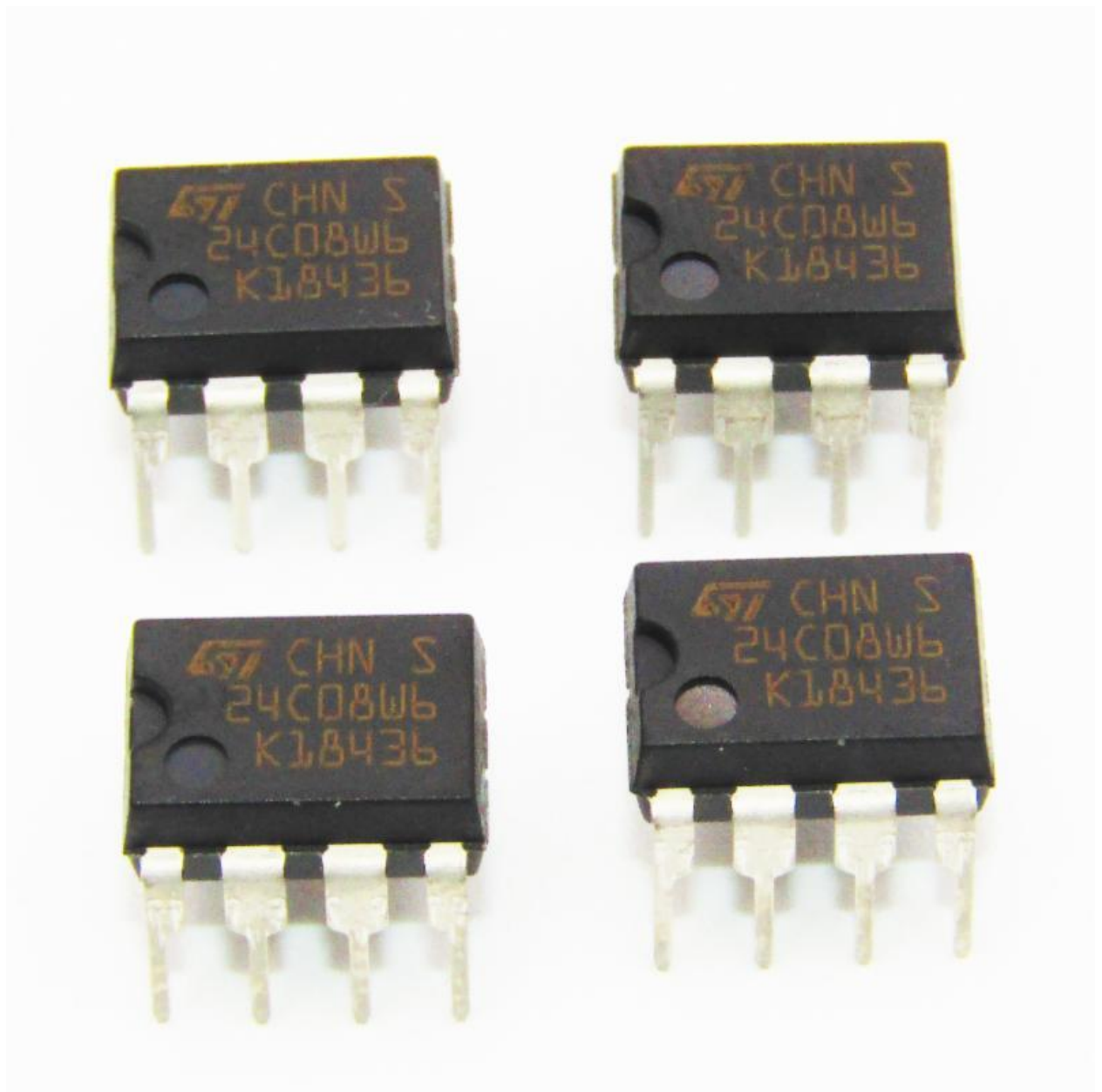
Внимание! EEPROM имеет ресурс по количеству перезаписи ячеек. Производитель гарантирует 100 000 тысяч циклов успешной записи, с сохранностью данных 100 лет при температуре в 24 градуса Цельсия. Независимые тесты показали 3-6 миллионов циклов перезаписи. Но устойчивость к внешним факторам может быть непредсказуемой.

EEPROM представляет собой область памяти, состоящую из элементарных ячеек с размером в один байт (как SRAM). Объем EEPROM разный у разных моделей МК:

- ATmega328 (Arduino UNO, Nano, Pro Mini): 1 кБ
- ATmega2560 (Arduino Mega): 4 кБ
- ATtiny85 (Digispark): 512 Б

Каждый байт в EEPROM памяти имеет свой адрес (главное их не перепутать). Так же скорость работы с EEPROM не зависит от частоты контроллера.

- Запись одного байта занимает ~3.3 мс (миллисекунды)
- Чтение одного байта занимает ~0.4 мкс (микросекунды)



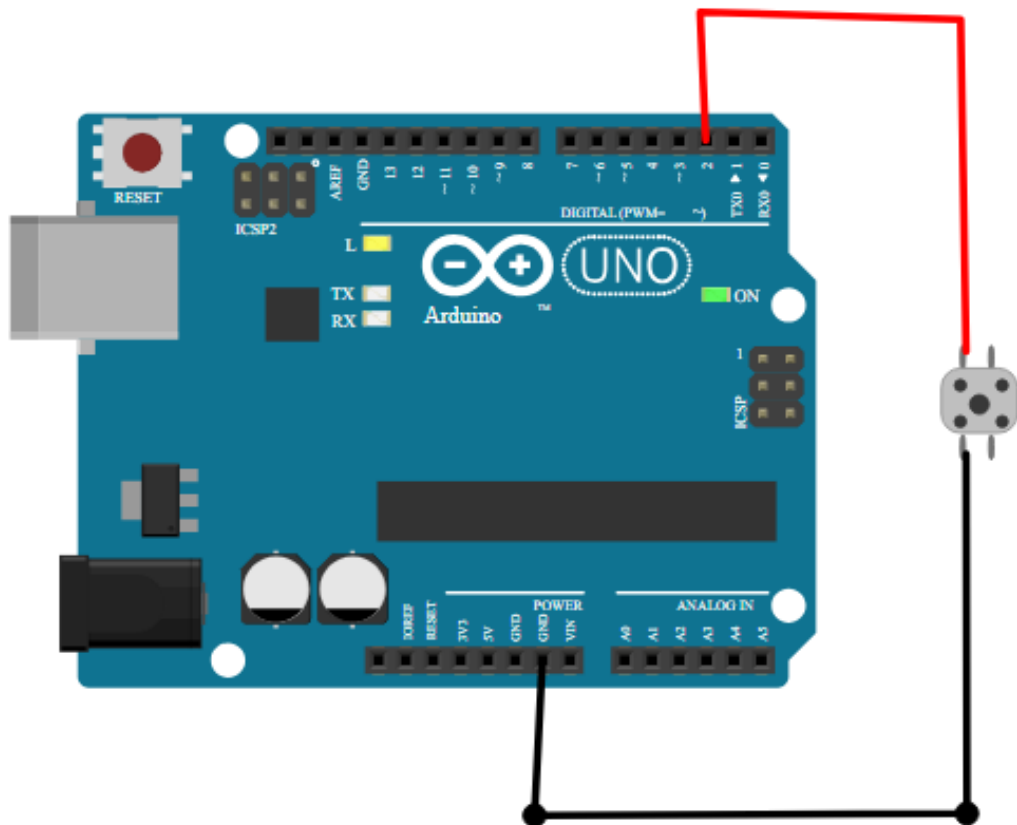
Проект №21

Сложность: нормально

Что в черном ящике?

Задача: считаем количество нажатий на кнопку и записываем это значение в две переменные, одну оставляем в оперативной памяти, а другую записываем в EEPROM.

Схема проекта:



Код проекта:

```
1  #include <EEPROM.h>           //Подключаем встроенную библиотеку для EEPROM
2  | | | | | | | | | | | | | | //Переменные
3  int button = 0;
4  int counter = 0;
5  int counter_EEPROM = 0;
6  bool eepromFlag = false;
7  uint32_t eepromTimer = 0;
8
9  void setup(){
10     pinMode(2, INPUT_PULLUP);    //Переменная для кнопки
11     EEPROM.get(0, counter_EEPROM); //Получаем значение с EEPROM
12     Serial.begin(9600);           // Открываем последовательный порт
13     Serial.print("Значение в оперативной памяти: ");
14     Serial.println(counter);
15     Serial.print("Значение в EEPROM: ");
16     Serial.println(counter_EEPROM);
17 }
18 void loop(){
19     checkEEPROM();               //проверяем память
20
21     button = digitalRead(2); //Считываем значение
22     if (button == LOW){       // Проверяем кнопку
23         counter++;
24         LetsClick();           //Сброс таймера для сохранения в память
25         while (digitalRead(2) == LOW);
26     };
27 }
28 void LetsClick() {
29     eepromFlag = true;         // поднять флаг
30     eepromTimer = millis();     // сбросить таймер
31 }
32
33 void checkEEPROM() {
34     // если флаг поднят и с последнего нажатия прошло 5 секунд
35     if (eepromFlag && (millis() - eepromTimer >= 5000) ) {
36         eepromFlag = false;     // опустили флаг
37         EEPROM.put(0, counter);  // записали в EEPROM
38         Serial.print("Значение в оперативной памяти: ");
39         Serial.println(counter);
40         Serial.println("Сохранено в чёрный ящик");
41         //после этого можно перезагружать микроконтроллер
42     }
43 }
```


Гироскоп и Акселерометр. GY-521 (MPU-6050)

DATASHEET

Что объединяет счет шагов, поворот экрана смартфона и навигацию?

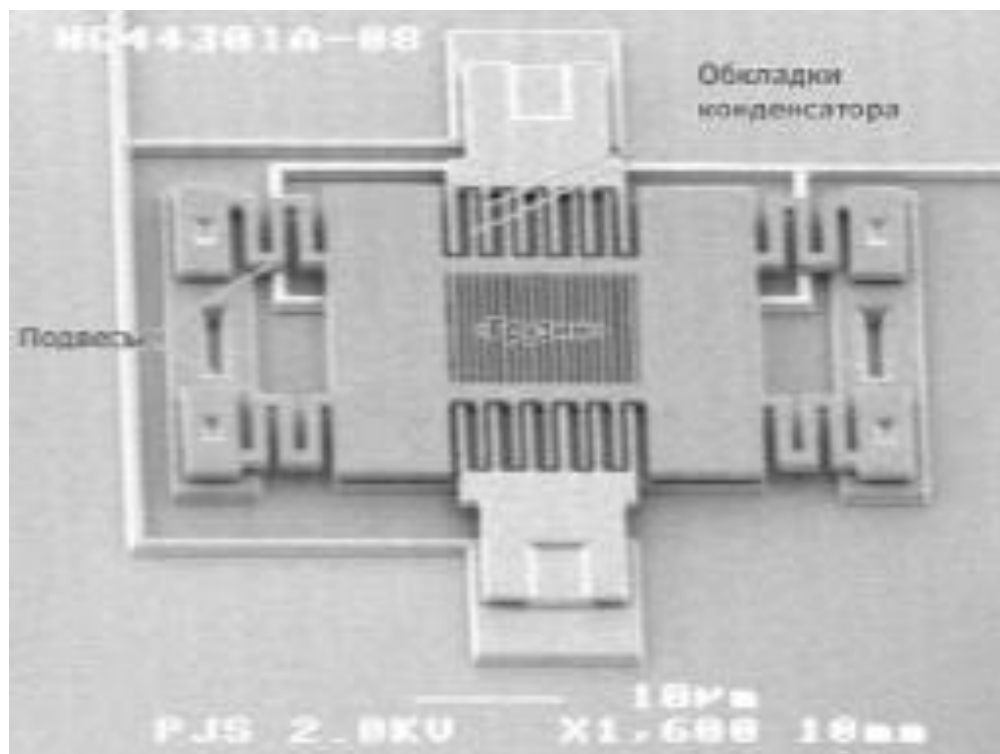
Во всех случаях необходимо измерять и отслеживать положение объекта в пространстве. Если у вас стоит похожая задача, то этот модуль отлично подходит.

Гироскоп – инструмент, который позволяет измерить реакцию тела на перемещение углов и вообще ориентации.

Акселерометр же служит измерителем проекции ускорения.

Модуль работает по интерфейсу I2C, хорошо нам знаком по предыдущим проектам

Ниже приведена фотография, сделанная под микроскопом:



На ней виден “грузик”, положение и ускорение которого, можно отслеживать при помощи подвижных контактов конденсаторов.

Примерно так устроен акселерометр и гироскоп в одной микросхеме.

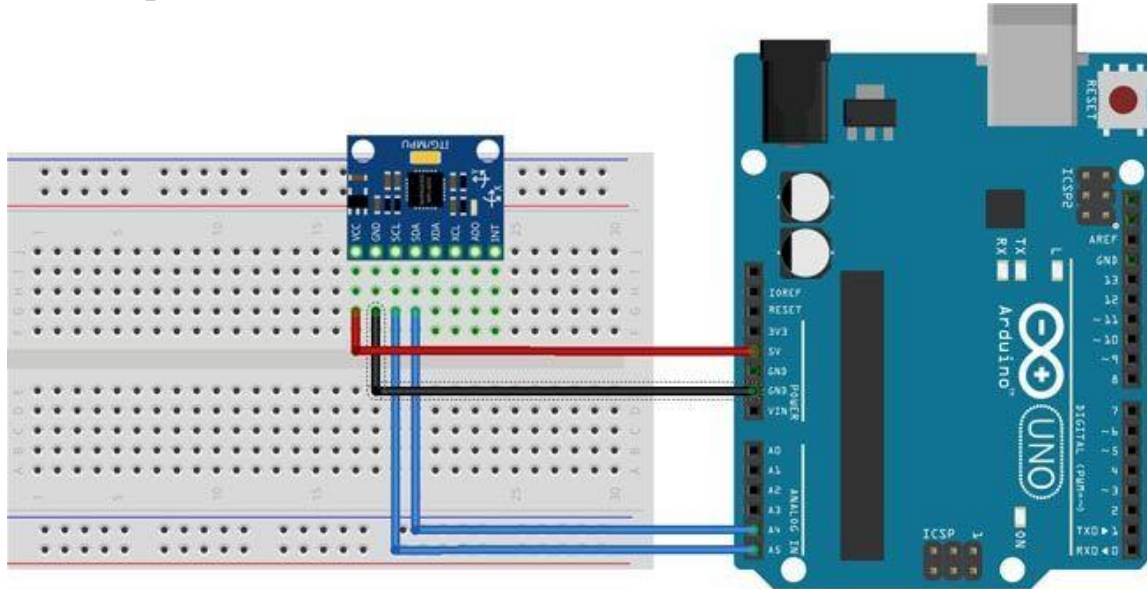
Проект №22

Сложность: легко

Где я?

Задача: отслеживать ускорение и положение в пространстве.

Схема проекта:



Код проекта:

Необходимо установить библиотеку из менеджера библиотек:

- [MPU6050](#)

```
1  #include "MPU6050.h"           //Подключаем библиотеки
2  #include "I2Cdev.h"
3  #include "Wire.h"
4  MPU6050 accelgyro;             //Переменные для акселерометра и гироскопа
5  int16_t ax, ay, az;
6  int16_t gx, gy, gz;
7
8  void setup(){
9  Wire.begin();
10 Serial.begin(9600);
11 Serial.println("Инициализирую устройства...");
12 accelgyro.initialize();
13 delay(500);
14 }
15 void loop(){                    //Выводим значения
16 accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
17 Serial.print("a/g:\t");
18 Serial.print(ax); Serial.print("\t");
19 Serial.print(ay); Serial.print("\t");
20 Serial.print(az); Serial.print("\t");
21 Serial.print(gx); Serial.print("\t");
22 Serial.print(gy); Serial.print("\t");
23 Serial.println(gz);
24 }
```

Электромагнитное реле

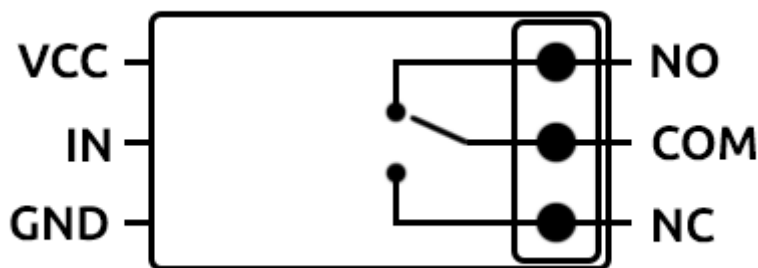
Реле - универсальный способ коммутировать нагрузку. Универсальность в чистой механике, то есть реле физически замыкает контакты. Это позволяет коммутировать нагрузку как переменного, так и постоянного тока в широком диапазоне напряжений. По току производитель обещает 10 А.

Само реле напрямую к микроконтроллеру подключать нельзя, поэтому для управления силовая схема развязывается с логической.

Различают 2 типа реле:

- Высокого уровня (работает, при подаче логической 1)
- Низкого уровня (работает, при подачи логического 0)

Внутреннюю схему реле, можно представить так:



Слева находятся пины питания и управления самого реле:

- VCC (DC+, +) – питание
- GND (DC-, -) – “земля”
- IN (S) – логический управляющий сигнал

Справа находятся выходы самого реле, это одна контактная группа с переключением:

- COM– общий контакт
- NO – нормально разомкнутый относительно COM контакт
- NC – нормально замкнутый относительно COM контакт

Принцип работы:

Подавая сигнал на вход, реле будет переключать COM контакт

Проект №23

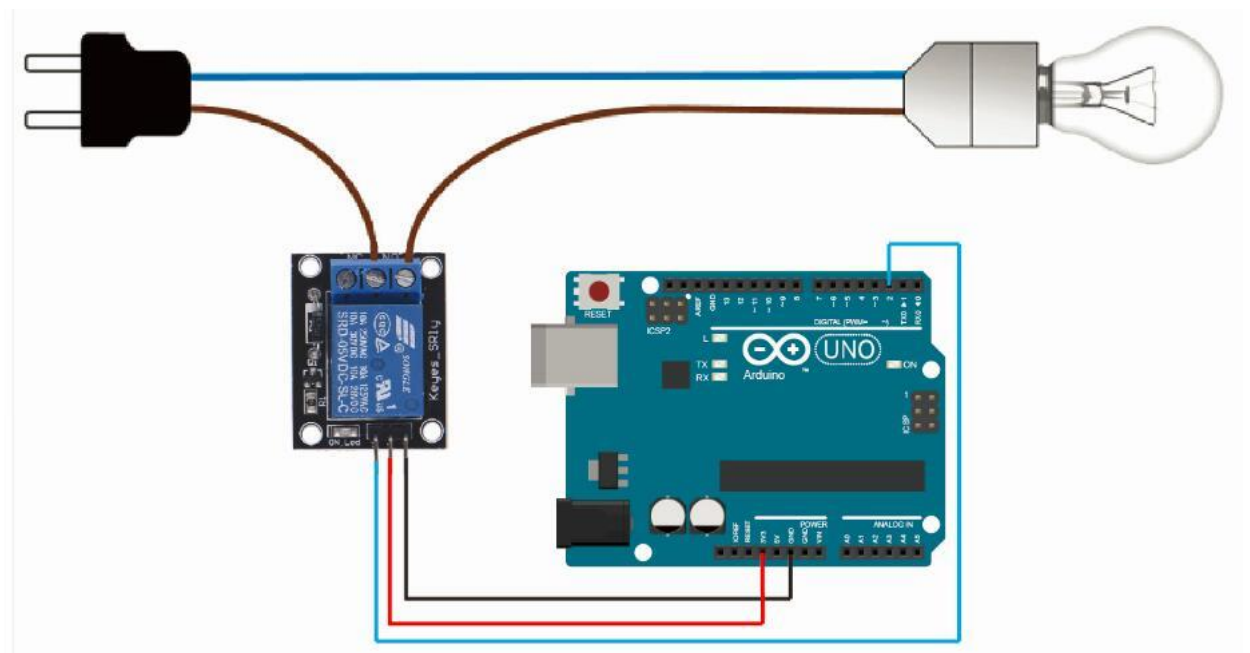
Сложность: средняя

ВНИМАНИЕ! При работе с сетевым напряжением необходимо соблюдать технику безопасности. Выполнение проекта производить строго под присмотром опытного специалиста или взрослого. Материал предоставлен в ознакомительных целях. ⚡

Управляем мощным прибором при помощи Ардуино.

Задача: зажигаем лампочку каждые 10 секунд.

Схема проекта:



Код проекта:

```
1  int Lampa = 2;           //Пин для управления реле
2
3
4  void setup() {
5      pinMode(Lampa,OUTPUT); //Режим работы пина
6  }
7
8  void loop() {            //Каждые 10 секунд меняем состояние пина
9      digitalWrite(Lampa,HIGH);
10     delay(10000);
11     digitalWrite(Lampa,LOW);
12     delay(10000);
13 }
```

Основы языка C/C++ и Arduino команд

Краткая справка по основным правилам и командам языка C/C++ и Arduino.

Типы данных. Переменные

Название	Альт. название	Вес	Диапазон	Особенность
boolean	bool	1 байт	0 или 1, true или false	Логическая переменная
char	-	1 байт	-128... 127	Хранит номер символа из таблицы символов ASCII
-	int8_t	1 байт	-128... 127	Целые числа
byte	uint8_t	1 байт	0... 255	Целые числа
int	int16_t, short	2 байта	-32 768... 32 767	Целые числа
unsigned int	uint16_t, word	2 байта	0... 65 535	Целые числа
long	int32_t	4 байта	-2 147 483 648... 2 147 483 647	Целые числа
unsigned long	uint32_t	4 байта	0... 4 294 967 295	Целые числа
float	-	4 байта	-3.4E+38 3.4E+38	Числа с плавающей точкой (десятичные дроби). Точность: 6-7 знаков
double	-	4 байта	-1.7E+308...1.7E+308	Для AVR то же самое, что float
-	int64_t	8 байт	-(2^64)/2... (2^64)/2-1	Целые числа
-	uint64_t	8 байт	2^64-1	Целые числа

Спецификаторы переменных

- **const** Константа. После объявления переменную невозможно изменить
- **static** Позволяет объявить переменную в функции, переменная не будет объявляться заново при повторном вызове функции, сохраняя свое значение.
- **volatile** Указывает компилятору, что переменную не нужно оптимизировать, что её значение может меняться извне (спецификатор должен быть применён к переменным, которые меняют своё значение в прерывании)
- **extern** указывает компилятору, что эта переменная объявлена в другом файле программы, но мы хотим пользоваться именно ей, а не создавать новую с таким же именем в этом файле программы. Позволяет читать/записывать в переменные, созданные в других файлах (библиотеках)!

Операторы

, - запятая, это тоже оператор, в следующих случаях:

- 1) Перечисление элементов в массивах
- 2) Перечисление аргументов в функциях
- 3) Выполнение последовательности действий

Арифметические

- = присваивание
- % остаток от деления
- * умножение
- / деление
- + сложение
- - вычитание

Логические

- == равенство
- != неравенство
- >= больше равно
- <= меньше равно
- > больше
- < меньше
- ! логическое НЕ (аналог **not**)
- && логическое И (аналог **and**)
- || логическое ИЛИ (аналог **or**)

Составные

- ++ инкремент $a++$ (аналог $a = a + 1$)
- -- декремент $a--$ (аналог $a = a - 1$)
- += составное сложение $a+=$ (аналог $a = a + 10$)
- -= составное вычитание: $a -= 10$ (аналог $a = a - 10$)
- *= составное умножение: $a *= 10$ (аналог $a = a * 10$)
- /= составное деление: $a /= 10$ (аналог $a = a / 10$)

Основные команды C/C++

//** - многострочный комментарий

// - однострочный комментарий

#include – директива для подключения библиотеки

#define – директива, объявления констант (указатель для предпроцессора заменить указанное название на указанную переменную)

if, else if, else – операторы сравнения.

Синтаксис

```
int a = 0; // объявляем переменную a
int b = 1; // объявляем переменную b
int c;     // объявляем переменную c
if (a == 0) { // если a равно 0
    c = a + b; // то c равно сумме a и b
}
else if (a == 1) { // если нет, но если a равно 1
    c = a - b; // то c равно разнице a и b
}
else { // иначе c равно 2
    c = 2;
}
```

for – оператор цикла “счетчика”, принимает три настройки: инициализация, условие и изменение.

Синтаксис

```
for (int i = 0; i < 100; i++) { // пока i меньше 100, i прибавляет 1 к себе самой
    // тело цикла
}
```


Основные Arduino команды

pinMode(pin, mode) Устанавливает режим работы пина (pin) в режим mode.

mode может принимать следующие значения:

- 1) INPUT - вход (все пины сконфигурированы так по умолчанию)
- 2) OUTPUT - выход (при использовании analogWrite ставится автоматически)
- 3) INPUT_PULLUP - подтяжка к питанию (например для обработки кнопок)

Цифровые пины

digitalRead(pin) Считывание состояние пина.

Возвращает:

- 0 (LOW) – на пине 0 V
- 1 (HIGH) – на пине 5V

analogWrite(pin, value) Запускает генерацию ШИМ² сигнала на пине **pin**.

value принимает значение от 0 до 255 (Коэффициент заполнения 0-100%)

ШИМ генерация доступна на следующих пинах:

- ATmega 328/168 (Nano, UNO, Mini): D3, D5, D6, D9, D10, D11
- ATmega 32U4 (Leonardo, Micro): D3, D5, D6, D9, D10, D11, D13
- ATmega 2560 (Mega): D2 - D13, D44 - D46

Аналоговые пины

analogRead(pin) Считывает и возвращает оцифрованное напряжение с пина **pin**.

Возвращает:

- Значение от 0 до 1023
- При отсутствии напряжения на пине, возвращает опорное напряжение

² ШИМ *абр.* Широтно-импульсная модуляция См. [Широтно-импульсная модуляция](#)

Аппаратное прерывание

attachInterrupt(pin,ISR,mode) Подключает прерывание на номер прерывания **pin**, назначает функцию **ISR**, как обработчик и установить режим прерывания **mode**.

ISR принимает название функции.

mode принимает следующие значения:

- LOW - срабатывает при сигнале LOW на пине
- RISING - срабатывает при изменении сигнала на пине с LOW на HIGH
- FALLING - срабатывает при изменении сигнала на пине с HIGH на LOW
- CHANGE - срабатывает при изменении сигнала (с LOW на HIGH и наоборот)

Не все пины способны на прерывание:

Микроконтроллер/Номер прерывания	INT 0	INT 1	INT 2	INT 3	INT 4	INT 5
Atmega 328/168 (UNO,Nano,Mini)	D2	D3	-	-	-	-
Atmega 32U (Leonard, Micro)	D3	D2			-	-
Atmega 2560 (Mega)	D21	D20	D19	D18	D2	D3

digitalPinToInterrupt(pin) Возвращает номер прерывания, согласно номеру пина выбранной платы.

Возвращает:

- Номер прерывания на выбранном пине выбранной платы

Разрешена следующая конструкция *attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)*

detachInterrupt(pin) Отключает прерывание выбранном пине **pin**

interrupts() Разрешает все прерывания

noInterrupts() Запрет на все прерывания.

ВНИМАНИЕ!!!

Запрет на все прерывание может сломать логику некоторых библиотек и аппаратные функции. Применять не рекомендуется.

Другие команды

Serial.begin(speed) Запускает связь по Serial на скорости **speed** (baud rate, бит в секунду)

Список скоростей доступных через Arduino IDE:

- 300
- 1200
- 2400
- 4800
- 9600 (используется чаще всего)
- 19200
- 38400
- 57600
- 115200 (часто встречается)
- 230400
- 250000
- 500000
- 1000000
- 2000000

Serial.print(val, [format]) Отправляет в порт число или строку, как строковое значение

format позволяет задать тип выводимых данных (только для чисел)

Принимаемые значения: BIN, OCT, DEC, HEX

Serial.println(val, [format]) Аналог **Serial.print(val)**, но автоматически переводит строку после отправки данных

Serial.write(val) или **Serial.write(buf, len)** Отправляет в порт численное значение или строку (**val**), или отправляет количество байт (**len**) из буфера **buf**.

ВАЖНО! Отправляет данные как байт.

Serial.end() Завершает связь по Serial.

delay(time) Приостанавливает выполнение кода (**исключение: прерывания**)

time принимает тип данных `unsigned long` и может приостановить выполнение на срок от 1 мс до ~50 суток (4 294 967 295 миллисекунд) с разрешением в 1 миллисекунду. *Работает на системном таймере 0.*

delayMicroseconds(time) Аналог **delay()**, останавливает выполнение кода от 4 до 16'383 микросекунд с разрешением 4 микросекунды.

millis() Возвращает количество миллисекунд, прошедших с запуска. После переполнения сбрасывается в 0.

Возвращает:

- `unsigned long` от 1 до 4 294 967 295 миллисекунд (~50 суток), имеет разрешение 1 миллисекунда

micros() Возвращает количество микросекунд, прошедших с запуска. после переполнения сбрасывается в 0.

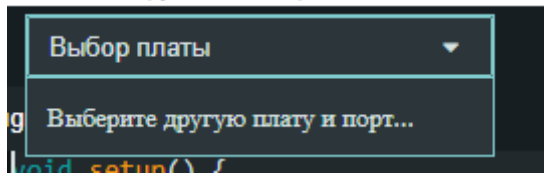
Возвращает:

- `unsigned long (uint32_t)`, от 4 до 4 294 967 295 микросекунд (~70 минут), имеет разрешение в 4 микросекунды.

Возможные проблемы и ошибки

Не собирается проект? Компьютер не видит Arduino? Красные буквы в консоли? Тогда этот раздел для вас.

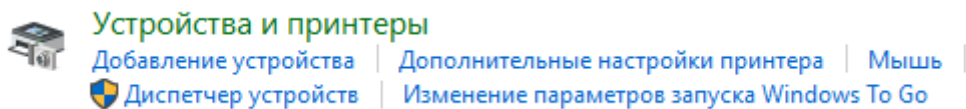
Arduino не отображается в селекторе выбора



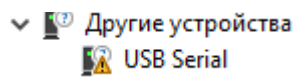
Решение:

Проверьте, что компьютер видит плату через “диспетчер устройств”

(Пуск-Паннель управления-Устройства и принтеры- Диспетчер устройств)



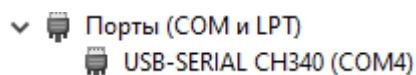
Если имеется следующее предупреждение:



Значит у вас не установлен драйвер.

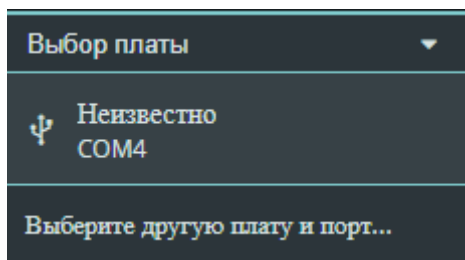
[Инструкция по установке драйвера](#)

Если компьютер должен видеть плату Arduino, как устройство с COM портом:

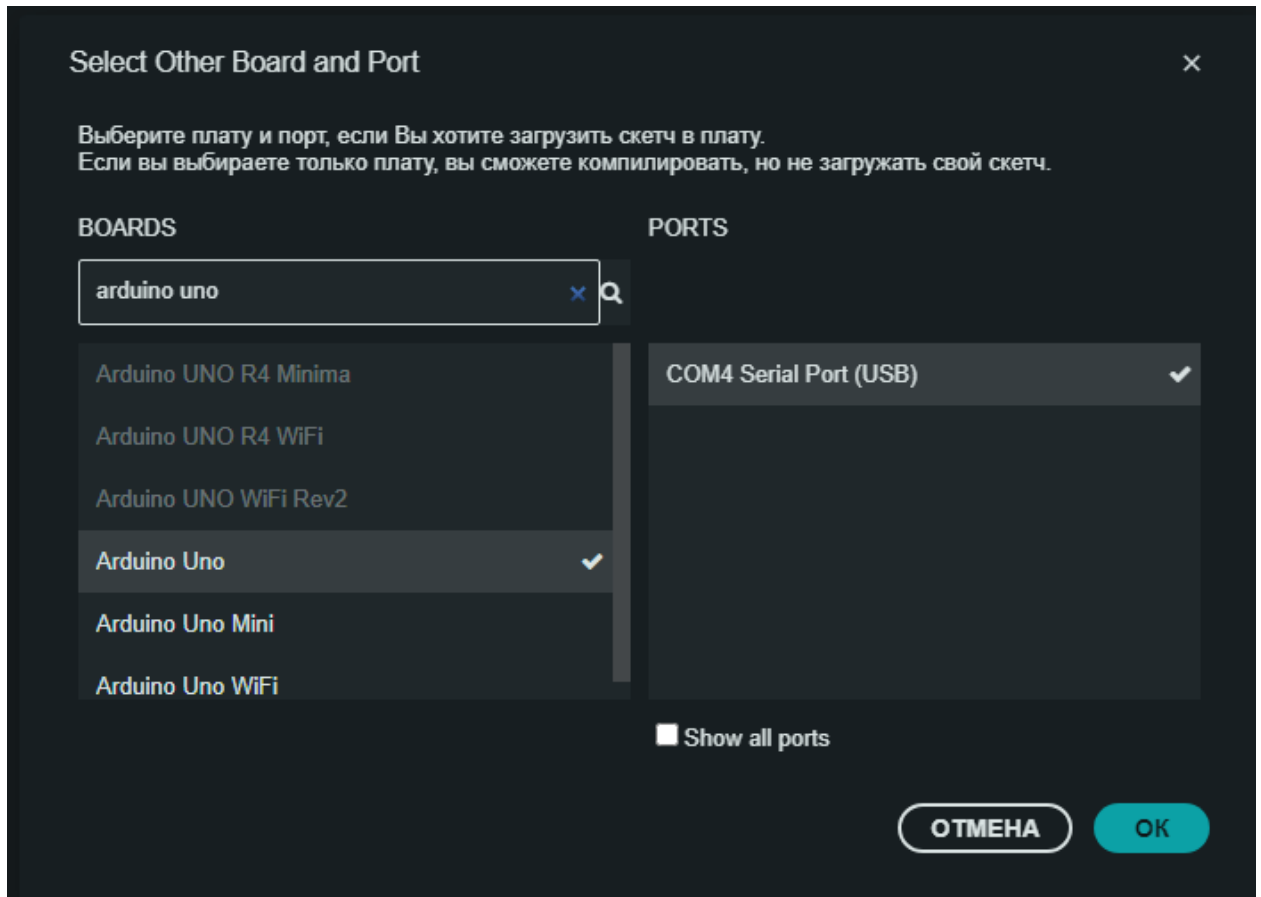


Необходимо запомнить номер COM порта (в данном случае COM4)

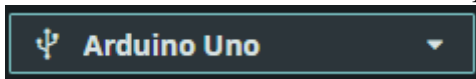
Переходим в Arduino IDE, в селекторе выбора платы указываем нужный порт



В всплывающем окне вбиваем название платы и нажимаем “ОК”



Название платы должно отображаться **жирным шрифтом**.



Ошибка загрузки

Failed uploading: no upload port provided

Решение:

Не выбрана плата для загрузки скетча. Выберите нужную плату и повторите попытку.

Ошибка сборки проекта

```
C:\Users\Admin\AppData\Local\Temp\.arduinoIDE-unsaved2024713-2812-mdn1q2.v5e7\sketch_aug13a\sketch_aug13a.ino: In function 'vo
C:\Users\Admin\AppData\Local\Temp\.arduinoIDE-unsaved2024713-2812-mdn1q2.v5e7\sketch_aug13a\sketch_aug13a.ino:6:13: error: exp
void loop() {
  | | | | | ^
exit status 1

Compilation error: expected '}' at end of input
```

Решение:

Возникает, если невозможно собрать проект из-за ошибки в коде. Ищите ошибку (в коде подсветиться красным строчки, в которых ошибки)

Выбран не верный COM порт для загрузки

```
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x2d
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x2d
Failed uploading: uploading error: exit status 1
```

Решение:

Выберите нужную плату и повторите попытку загрузки

COM порт занят или отсутствует

```
avrdude: ser_open(): can't open device "\\.\COM7": ?????.
Failed uploading: uploading error: exit status 1
```

Решение:

Проверьте, что у вас открыта одна вкладка Arduino IDE и правильно выбрана плата. (Если вы используете клон платы Arduino, существует вероятность, что при следующем подключении смениться номер COM порта)

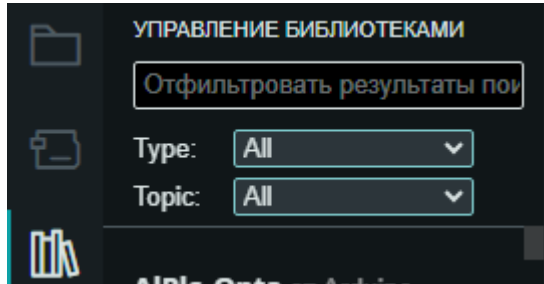
Отсутствует библиотека

```
fatal error: LiquidCrystalRus.h: No such file or directory
```

Данная ошибка означает, что не установлена библиотека

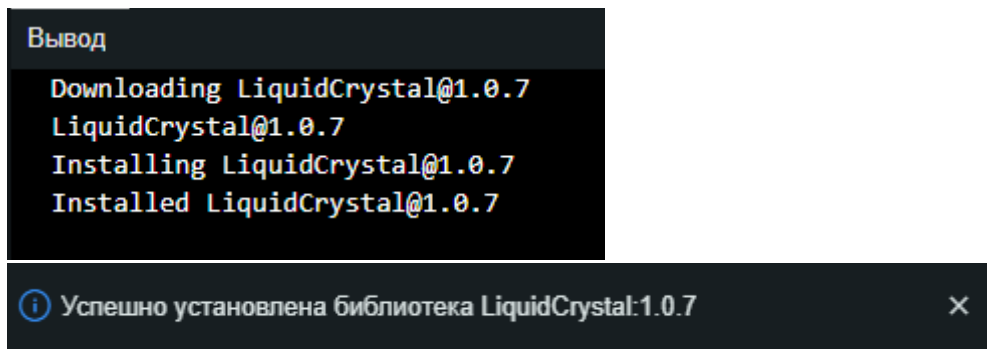
Решение:

Введите название библиотеки в менеджер библиотек



Нажмите кнопку “УСТАНОВКА”

Дождитесь окончания установки и попробуйте залить скейч заново



Отсутствует возможность подключиться к плате

```
No connection established  
Compilation error: No connection established
```

Решение:

Проверьте, что у вас открыта одна вкладка Arduino IDE и правильно выбрана плата, перезагрузите Arduino IDE.

Отсутствуют необходимые файлы

```
exit status 1
```

```
Compilation error: 'class Rotary' has no member named 'process'
```

Проблема возникает, если вы скачивали готовый проект и забыли распаковать архив, открыв только файл .ino .

Решение:

Распаковать весь архив в отдельную папку, повторите попытку загрузить скейч.