Методические рекомендации по работе с платформой Arduino

Подготовлено студентами Колледжа Современных Технологий им М. Ф. Панова

Год 2024 группой СОБ/21 года

При поддержке В.Г. Макарова

ОГЛАВЛЕНИЕ

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 отличии от STM32, программирование Arduino не требует глубокого погружения в составные части микроконтроллеров, можно писать код и заниматься электроникой с заметной лёгкостью в отличии 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**)

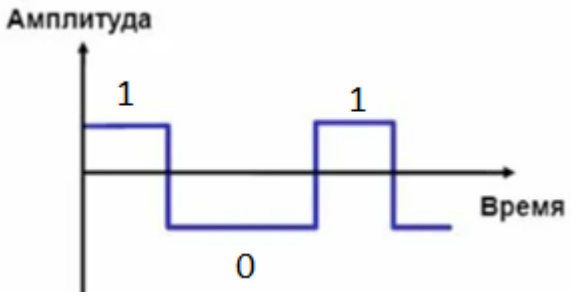
Сопротивление - физическая величина, которая характеризует способность проводника препятствовать протеканию тока.

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 единичек. 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 ардуино, 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 сравнивания 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 удержать 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.

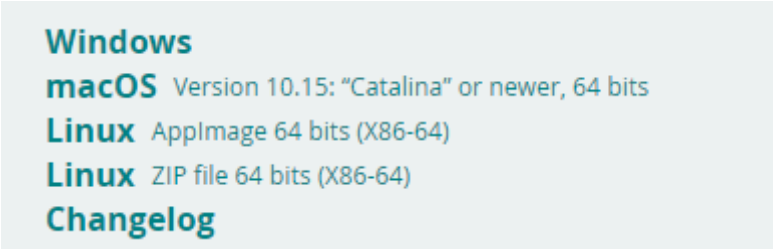
Начало работы

Театр начинается с вешалки, а работа на ардуино 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, MacOC, Linux)

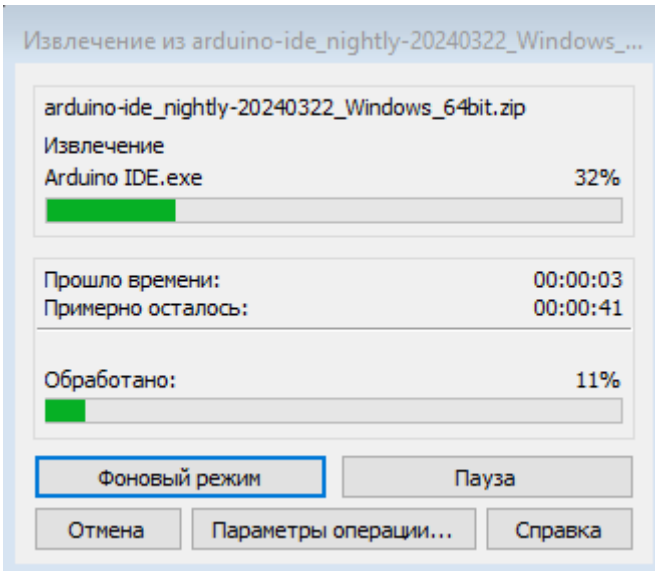
Вкладка “Sowtware”

Выбираем нужную операционную систему



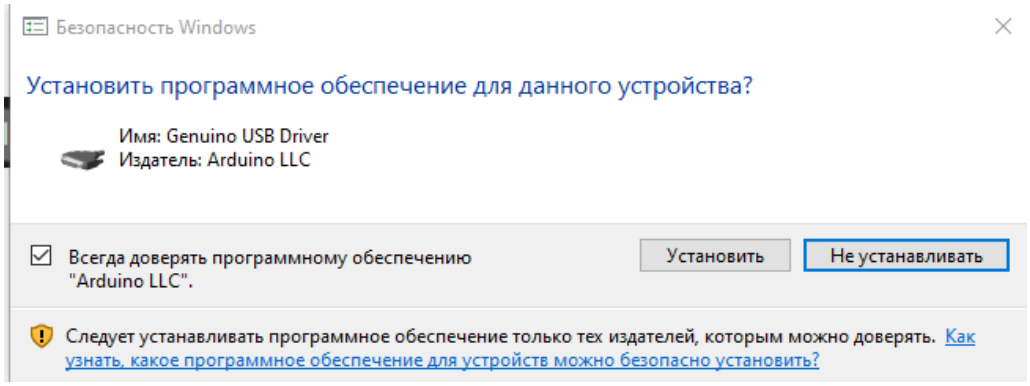
Пропускаем всё нажимая кнопку “JUST DOWNLOAD”

Извлекаем архив



Запускаем файл

Устанавливаем дополнительные компоненты для ардуино

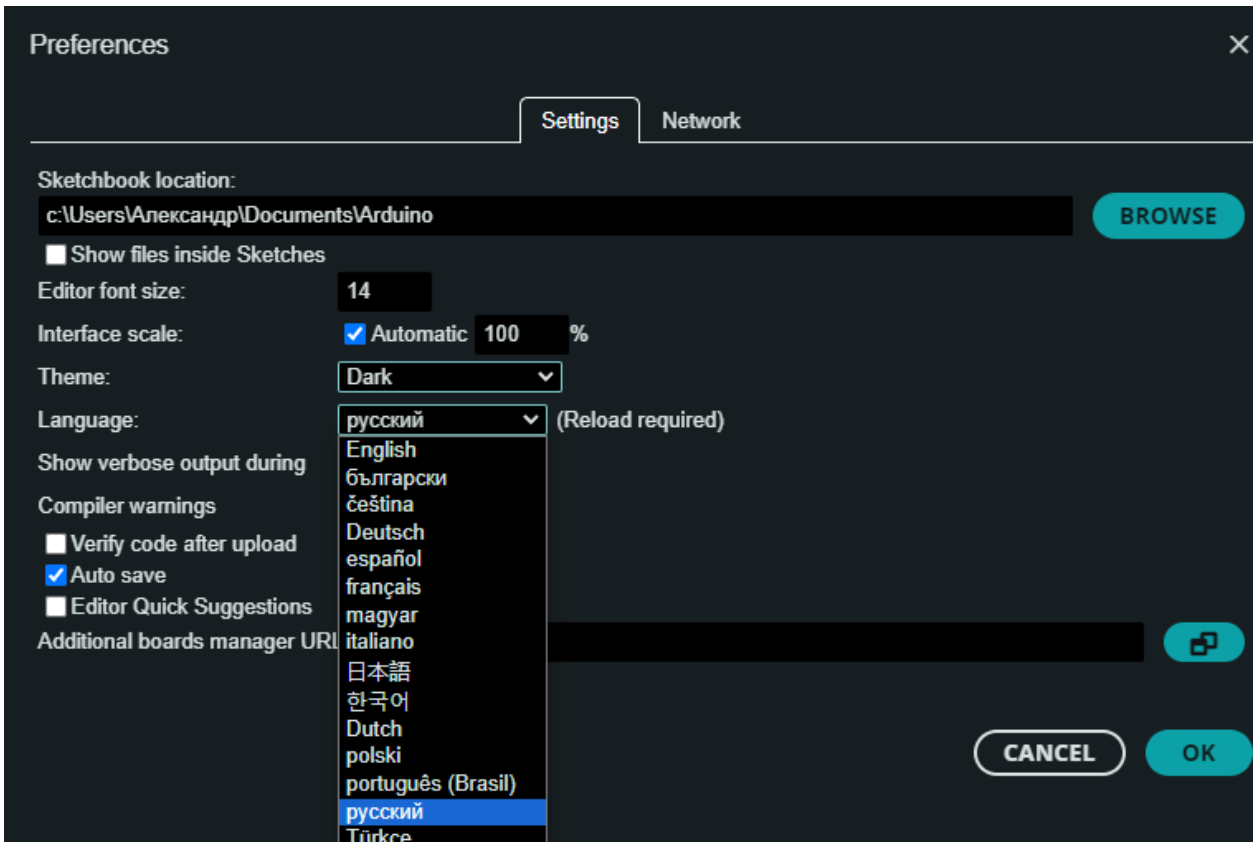


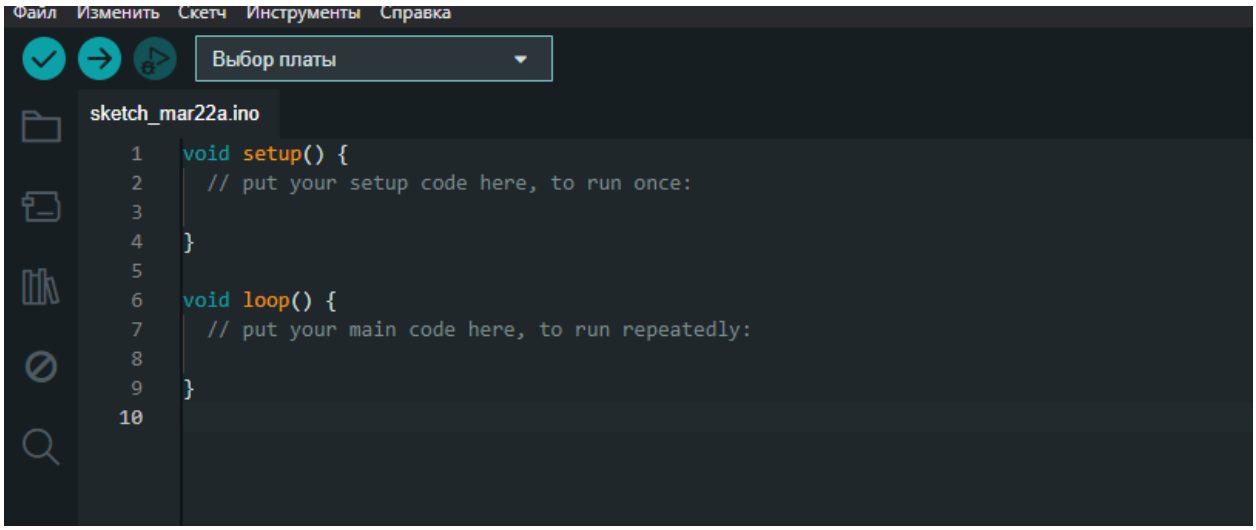
Открываем тот же файл еще раз

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 – Languge – выбираем русский язык

Программа перезагрузиться и будет на русском языке





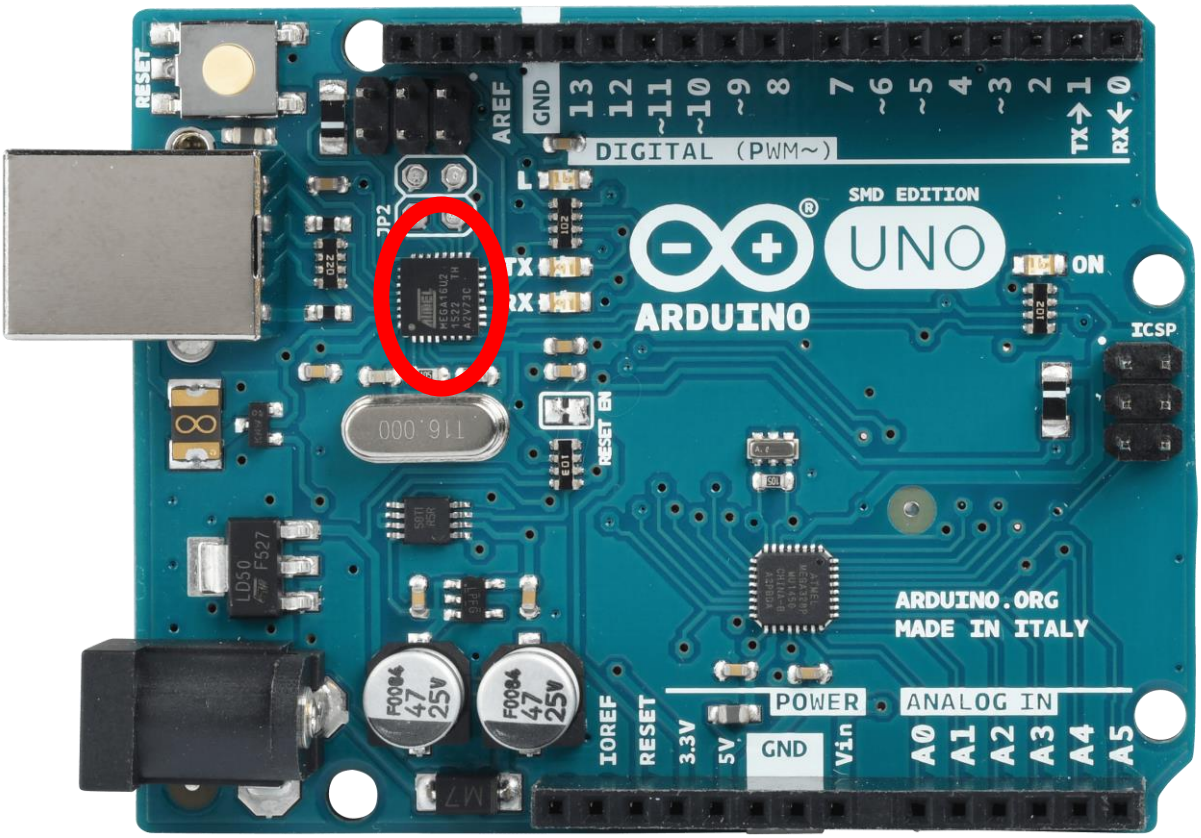
Это начало всех скейчей

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

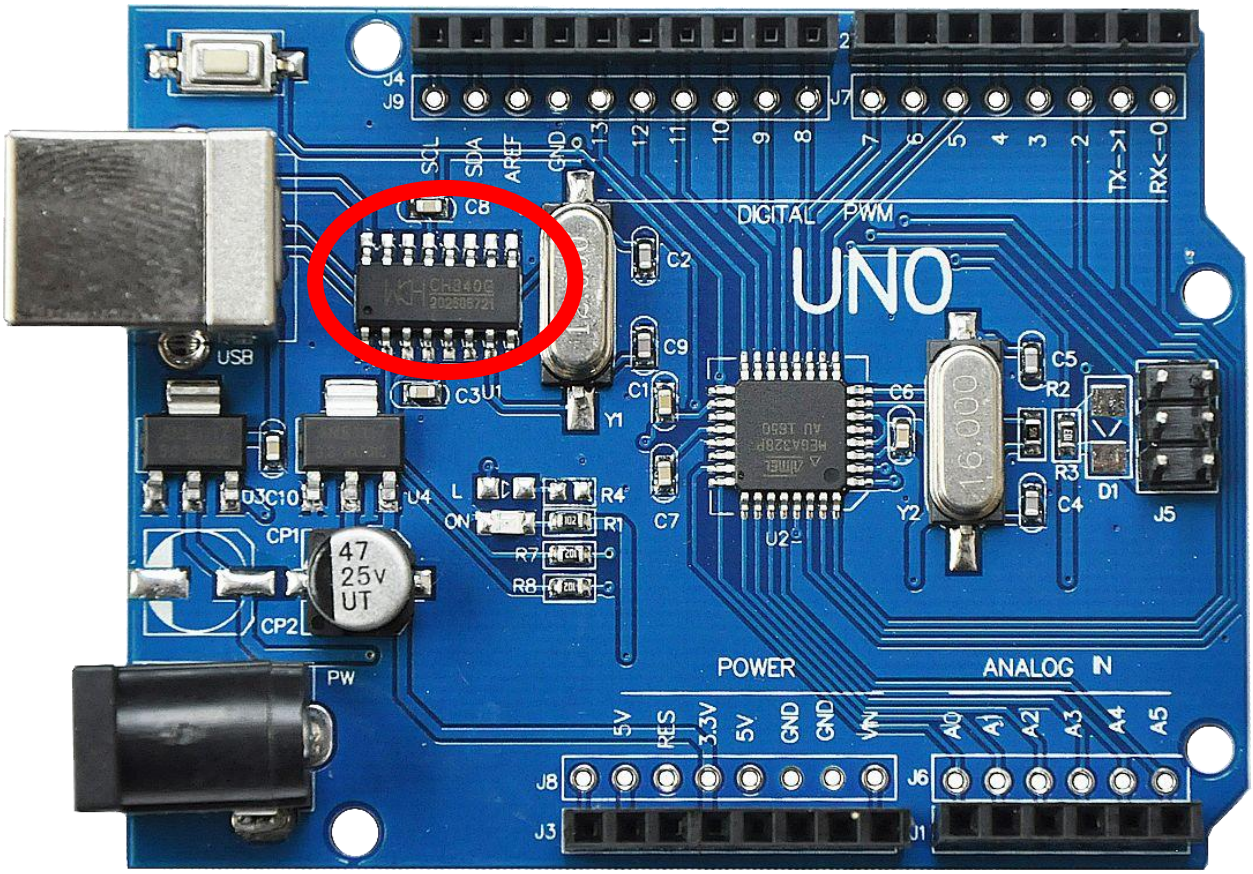
Блок “void loop()” это код, который выполняется всегда, пока есть питание.

Теперь необходимо понять тип вашей платы Arduino UNO.

Есть два типа: Оригиинальная плата Arduino UNO



И её китайская копия

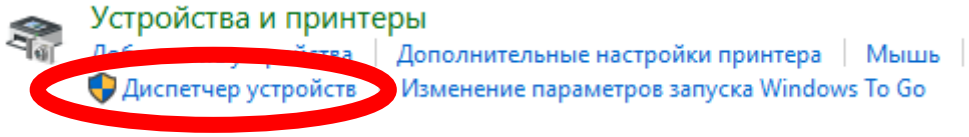


Главное их отличиие - это программатор (выделеен красным).

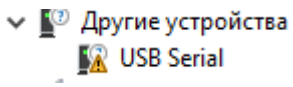
Если у вас оригинальная плата, можно переходиить к первому проекту. При установке Arduino IDE установились необходиимые драйверы.

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

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



Если имеется следующее предупреждение:

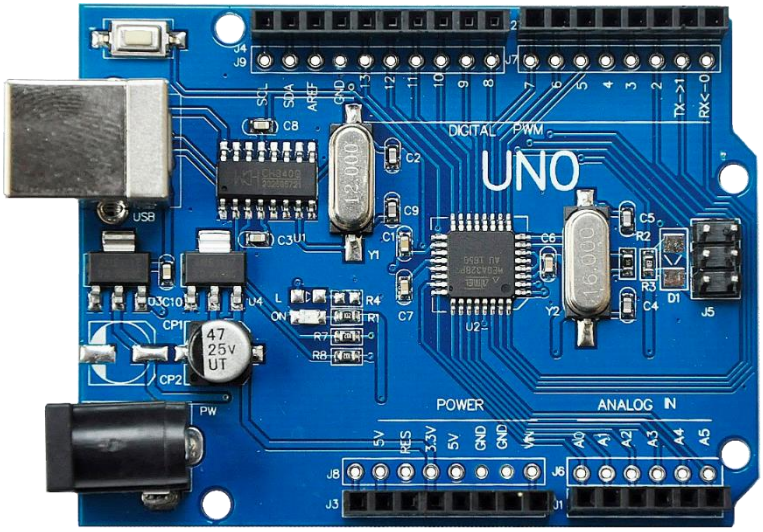


Значит у вас не установлен драйвер. Инструкция по установке драйвера

Установка драйверов

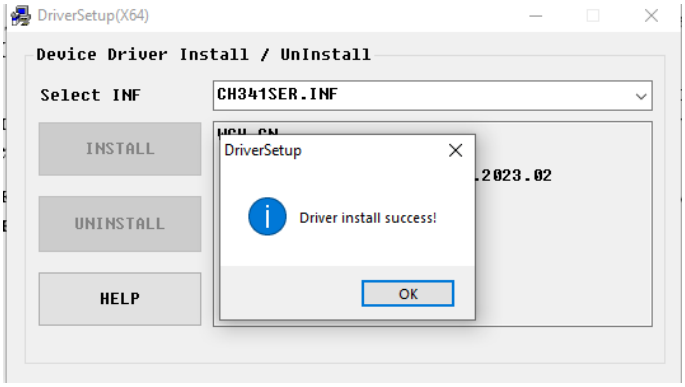
Установка прозводиться на операционную систему семейства Windows, установка на другие операционные сиистемы в методических рекомендация не рассматривается

Если ваша плата Arduin 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

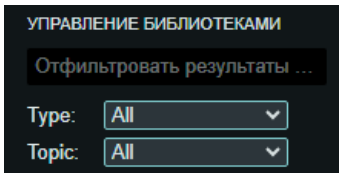
Менеджер библиотек находить в этой вкладке -> 

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

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

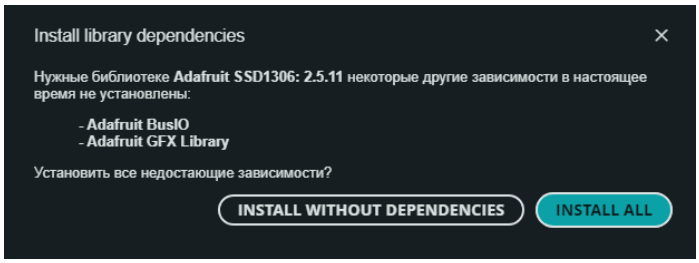
Type фильтр по признакам (не установлена/установлена, официальная/партнерская библиотека и т.д.)

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

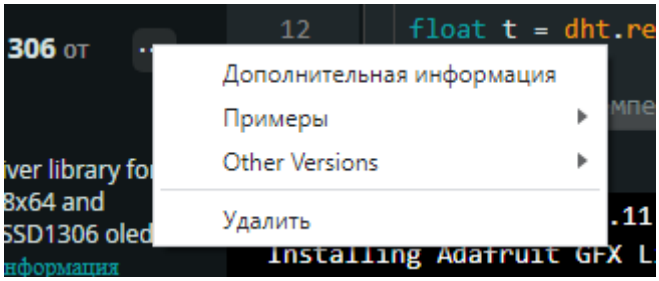


Для установки библиотеки необходимо нажать кнопку “УСТАНОВКА”

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.

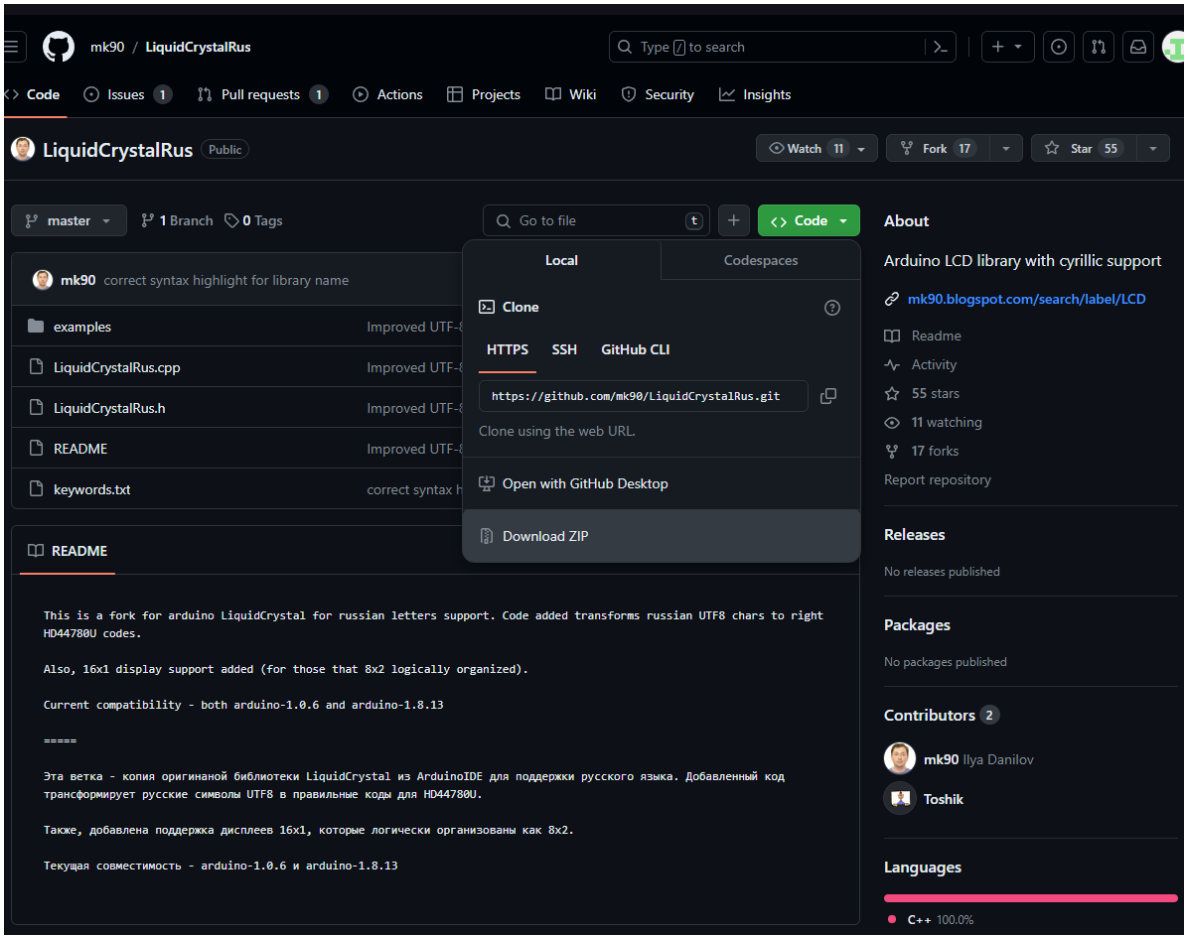


Установка сторонних библиотек

Если необходимая библиотека отсутствует в вустроенном менеджере библиотек, то данная глава расскажет, как устанновить сторонние.

Для начала потребуется ссылка на необходимую библиотеку (буддем устанавливать LiquidCrystalRUS)

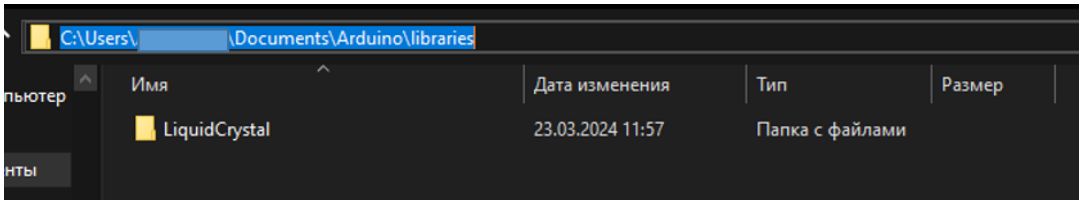
Скачайте библиотеку GitHub



Скачаннный архив необходимо распоковать в папку с библиотеками.

Обычно она находиться по следующему адресу

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



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

Сторонняя библиотека установленна

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:

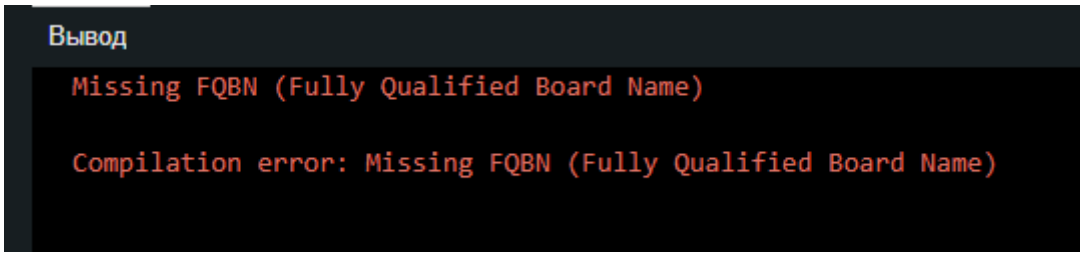
Схемы не будет, ведь мы моргаем встроенным светодиодом :)

Код проекта:

```
int LED = 13; // LED это переменная для светодиода, пин подключения 13
              //( к 13 пину уже подключен светодиод нна плате)

void setup() { // код в setup() выполняется один раз
  pinMode(LED, OUTPUT); // Режим работы пина 13 - выход сигнала
}

void loop() { // код в loop() будет выполнятся "по кругу", пока есть питание на Arduino
  digitalWrite(LED, HIGH);  // Зажигаем светодиод
  delay(5000);              // ждем 5 секунд
  digitalWrite(LED, LOW);   // Гасим светодиод
  delay(5000);              // И снова ждем 5 секунд
}
```

Данная кнопка позволяет проверить код на ошибки.

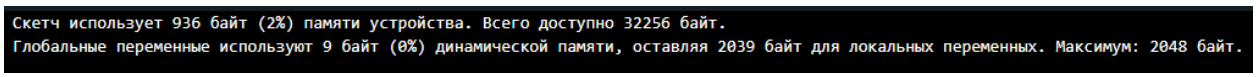
Если вы не выбрали плату, будет следующая ошибка.



Выбираем плату Arduino Uno.

Проверяем код на ошибки ещё раз.

Данная строка означает, что ошибок в коде нет.



Можно заливать скейч на arduino

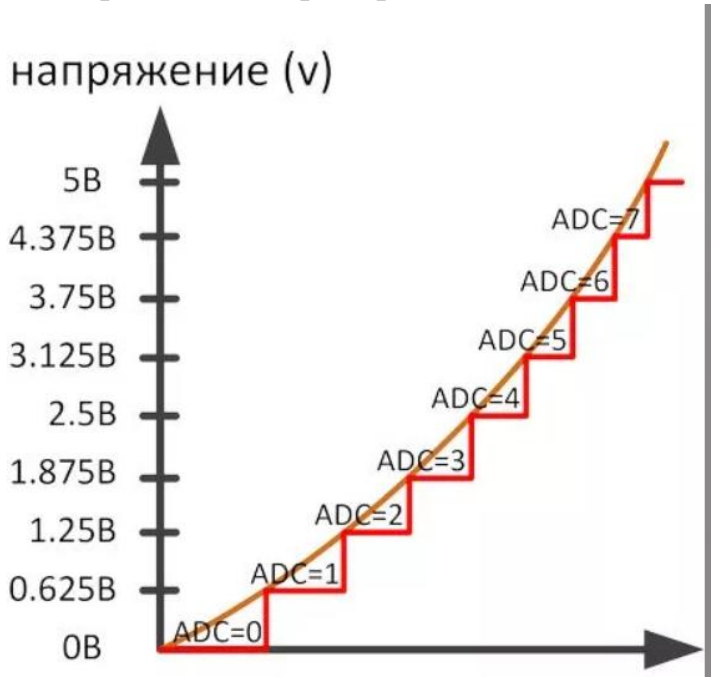
Не забывайте сохранять скейчи. Помните каждый скейч хранится в отдельной папке, название скейча и папки обязательно должно совпадать и не содержать кирилицы.

Аналогово – цифровой преобразователь

Теория

АЦП - это устройство преобразующее аналоговый сигнал в дискретный.

Основной принцип его работы основан насравнениии и преобразовании. Принцып его работы можно описать так: полученное аналоговое значение сравниваеца с таблицой и возвращается в виде цифры, соответсвующей определенному уровню сигнала. АЦП применяют для оцифровки музыки, для измирительных приборов, видеотехнике и много где ещё.



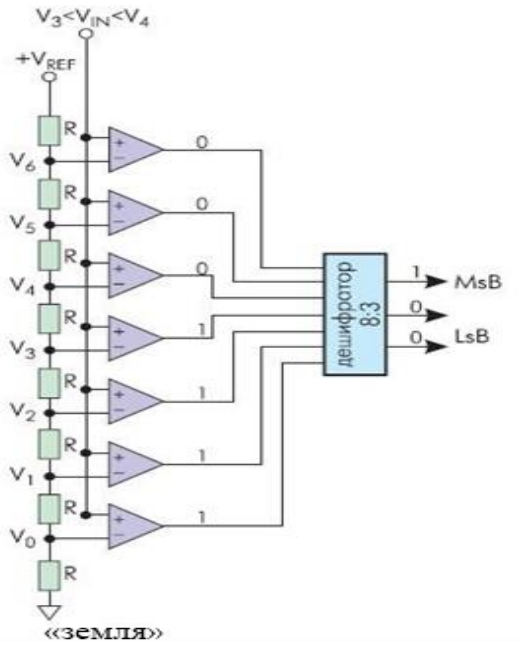
Существует несколько видов АЦП:

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 компораторов подключён к дешифратору, который уже и отправляет дискретный сигнал

Блок-схема ниже



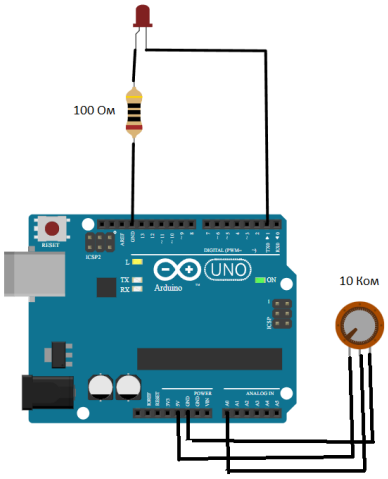
Так же АЦП различаются по битности (максимальное количество символов для кодирования аналогово сигнала – чувствительность) и порогом измеряемых напряжений.

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

Проект №2
Сложность: легко

Задача: При напряжении до 3 V, светодиод не горит, при напряжении 3,3 V – светодиод горит, при напряжении 5 V светодиод мигает.

Схема проекта



Код проекта

```
int LED = 1;                        // LED это переменная для светодиода, пин подключения 1
int Volt = A0;                      //Подключаем аналогвый пин A0

void setup() {
  pinMode(LED, OUTPUT);             // Режим работы пина 1 - выход сигнала
  pinMode(Volt, INPUT);             // Режим работы пина A0 - считывание сигнала
  Serial.begin(9600);
}

void loop() {
  int Voltag = analogRead(Volt); //Переменная Voltag принимает значение оцифрованного с АЦП
  if (Voltag == 690){           // Если значение равно 3.3 вольта, то светодиод горит
    digitalWrite(LED,HIGH);
  }
  if (Voltag == 1023){          // Если значение равно 5 вольта, то светодиод горит
    digitalWrite(LED,HIGH);
    delay(1000);
    digitalWrite(LED,LOW);
    delay(1000);
  }
  else{                          // Иначе светодиод не горит
    digitalWrite(LED,LOW);
  }
  Serial.println(Voltag);
}
```

ЖК-дисплей MT-16S2H

DATASHEET

Теория

Жидкокристаллический дисплей (LCD) mt-16s2h имеет 16 столбцов и 2 строки (16/2). Собственный паралельный 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 ардуино.

Таблица символов можно найти в DATASHEET на странице 8

Проект №3
Сложность: нормально

Задача: Вывод сообщения на дисплей

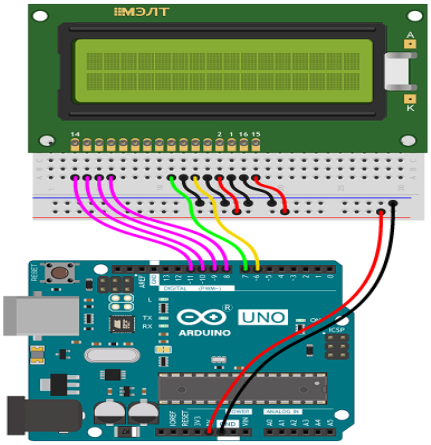
Установите стандартную библеотеку:

- LiquidCrystal

Необходимо установить сторнюю библиотеку:

- LiquidCrystalRUS

Схема проекта:



Код проекта:

Этот код выводит сообщение “Hello word!”

```
// Библиотека для работы с текстовым дисплеем
#include <LiquidCrystalRus.h>

// Задаём имя пинов дисплея в 4 битном режиме
constexpr uint8_t PIN_RS = 6;
constexpr uint8_t PIN_EN = 7;
constexpr uint8_t PIN_DB4 = 8;
constexpr uint8_t PIN_DB5 = 9;
constexpr uint8_t PIN_DB6 = 10;
constexpr uint8_t PIN_DB7 = 11;

/*
 * Создаём объект для работы с текстовым дисплеем
 * LiquidCrystalRus lcd(RS, EN, DB4, DB5, DB6, DB7);
*/

LiquidCrystalRus lcd(PIN_RS, PIN_EN, PIN_DB4, PIN_DB5, PIN_DB6, PIN_DB7);

void setup() {
  // Устанавливаем размер экрана
  // Количество столбцов и строк
  lcd.begin(16, 2);
  // Устанавливаем курсор в колонку 5 и строку 0
  lcd.setCursor(0, 0);
  // Печатаем первую строку
  lcd.print("Hello world!");

}
void loop() {
}
```

OLED дисплей 1306

Теория

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

Рассмотрим дисплей с интерфейсом I2S и размером 128*32 пикселя.

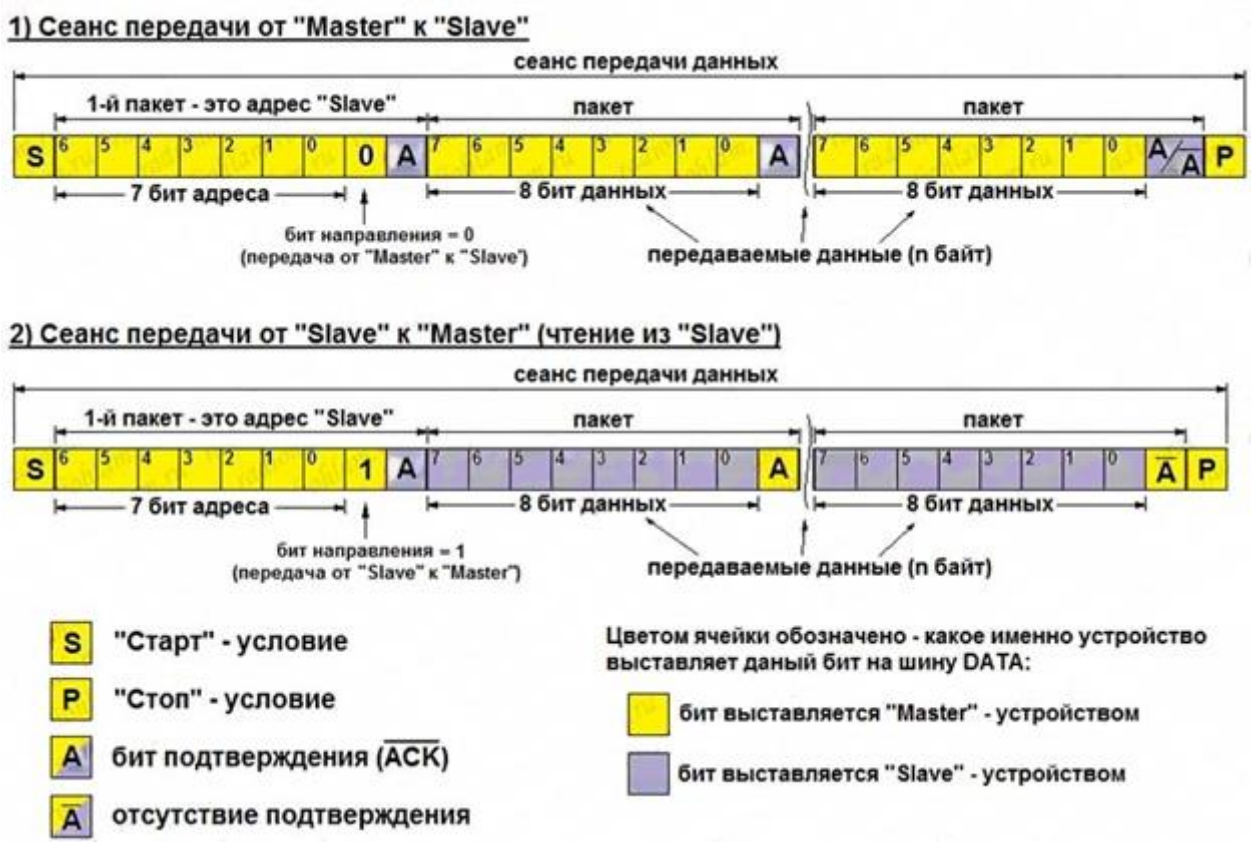
I^2C один из самых удобных интерфейсов

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:



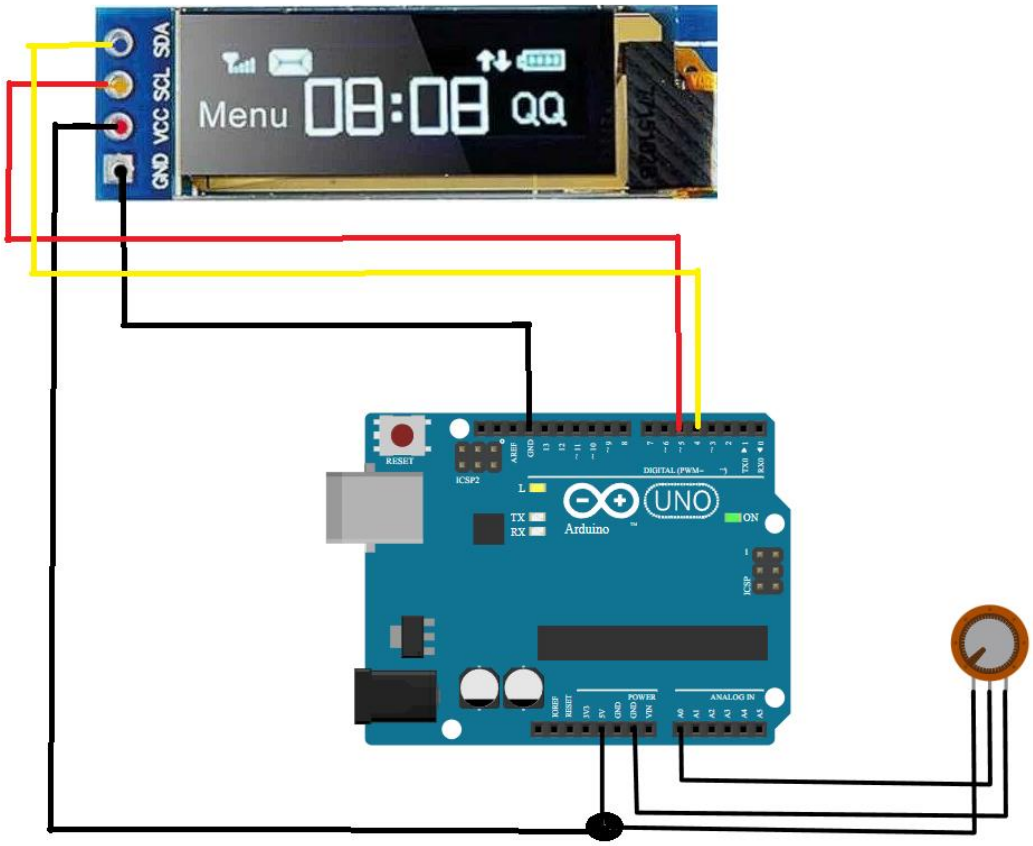
Проект №4
Сложность: нормально

Задача: Вместо вывода данных в консоль, будем выводить данные на дисплей, будем выводить данные с АЦП.

Для продолжения необходимо скачать библиотеки через менеджер библиотек:

- Adafruit SSD1306
- Adafruit GFX Library

Схема проекта:



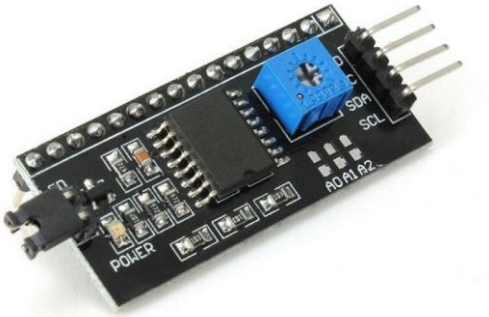
Код проекта:

Данный код выводит данные с АЦП на экран

```
#include <Adafruit_GFX.h>        //OLED библиотека
#include <Adafruit_SSD1306.h>
#include <Wire.h>                //Библиотека I2C

#define SCREEN_WIDTH 128 // Ширина OLED-дисплея, в пикселях
#define SCREEN_HEIGHT 32 // Высота OLED-дисплея в пикселях
#define OLED_RESET    -1 // т.к.у дисплея нет пина сброса прописываем -1, если используется общий сброс Arduino
int v (A0);

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //Объявляем имя и задаем параметры


void setup() {
  pinMode(A0,INPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //Запуск дисплея
  display.display();
  delay(2000);
      display.clearDisplay();              //Очистить дисплей
    display.setTextSize(2);              //Задаем размер текста
    display.setTextColor(WHITE);         //Задаем цвет текстаw
    display.setCursor(35,10);            //Задаем координату начала текста в пикселях
    display.println("Hello user!");            //Пишем сам текст
    display.display();
    delay(2000);                            //Пауза для инизиализации дисплея
  }

void loop() {
    int Voltag =analogRead(v);
    char var[5]; // создаем массив на 5 элементов
    int Volt = sprintf(var, "%d", Voltag);
    display.clearDisplay();              //Очистить дисплей
    display.setTextSize(2);              //Задаем размер текста
    display.setTextColor(WHITE);         //Задаем цвет текстаw
    display.setCursor(15,5);            //Задаем координату начала текста в пикселях
    display.println("Напряжение");
    display.setCursor(45,10);            //Задаем координату начала текста в пикселях
    display.println(Volt);
    display.display();                   //Команда для отображения всего этого на дисплее
  }
```

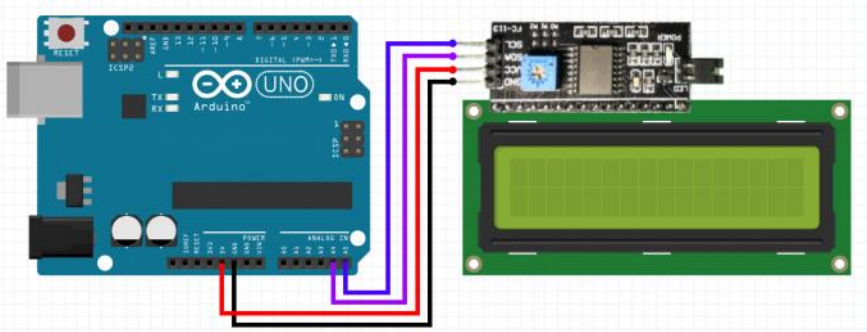
Проект №5
Сложность: нормально

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



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

Схема проекта:



Код проекта:

Необходимо установить сторнюю библиотеку:

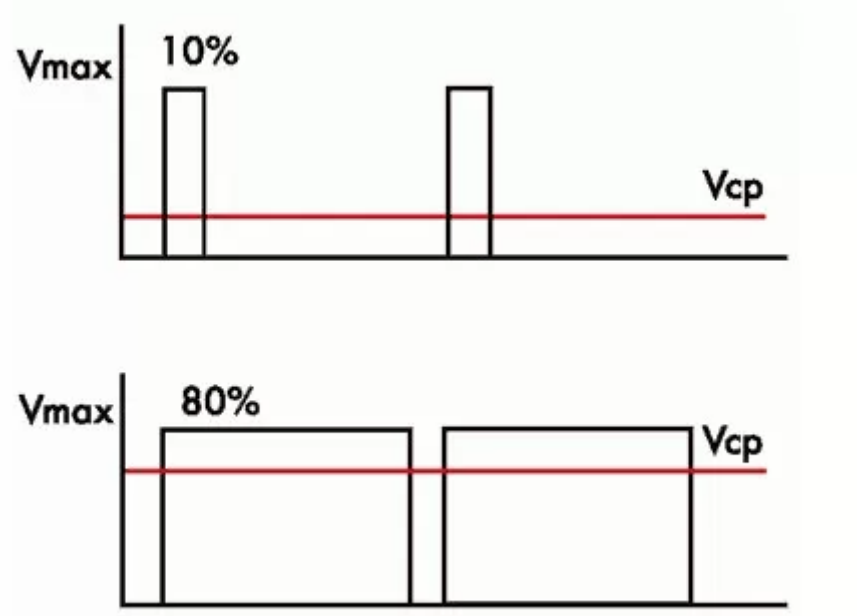
- LCD_1602_RUS_ALL

```
#define _LCD_TYPE 1  // 0 если по параллеельнному иитееерфейсу / 1 для работы с I2C
#include <LCD_1602_RUS_ALL.h> // подключаем библиотеку
LCD_1602_RUS lcd(0x27, 16, 2); // назначаем адрес устройства и количество сиимволов/строк

void setup() {
  lcd.init(); //инициалиизируеем дисплей
  lcd.backlight(); // вкключаем подсветку
  lcd.setCursor(0, 0); // курсор в первый сиимвол первой строки
  lcd.print("Привет!"); // вывод строки
  lcd.setCursor(0, 1); // курсор в первый сиимвол второй строки
  lcd.print("Hello!"); // вывод строки
}

void loop() {
}
```

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 окритично важных хараректеристик, которое необходимо учитывать для коректной 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

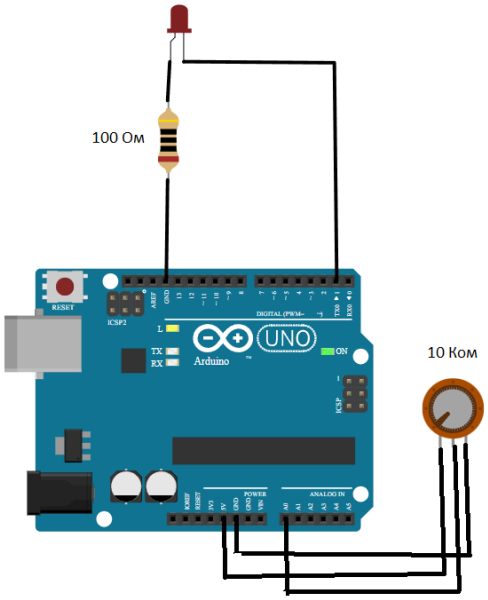
ШИМ генератор на ардуино 8 битный (Скважность регулируется от 0 до 255) и подключен он не ко всем пинам (это нужно учитывать)

Проект №6

Сложность: легко

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

Схема проекта:



Код проекта:

```
int pwm
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:
  pwm = analogReasd(A0); // считываем значение с потанцеометра
  pwm = map(pwm, 0, 1023, 0, 255); // конвертируем 10 разрядность в 8 разрядность
  pwm = constrain(pwm, 0, 255); // обрезаем диапозон, для исключения выхода за рамки
  analogWrite(3,pwm); // Выводим ШИМ сигнал на 3 пин
}
```

Сервопривод SG90

DATASHEET

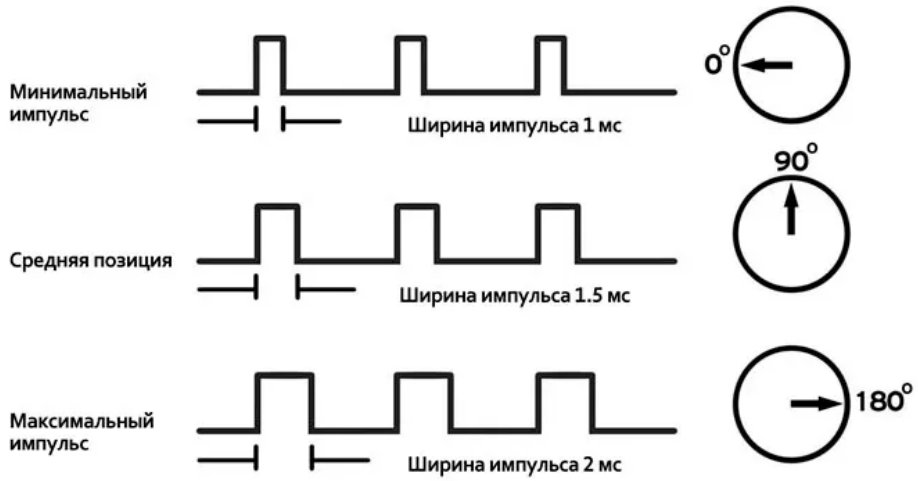
Теория

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.



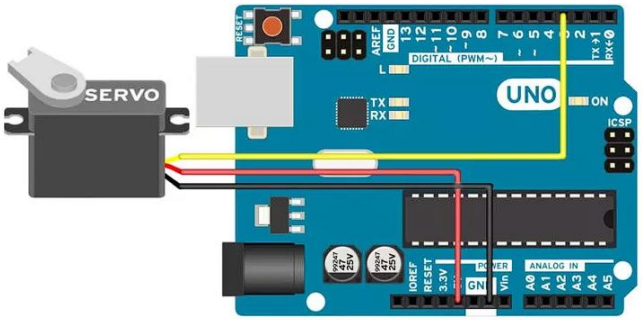
Проект №7
Сложность: легко

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:

```
// Подключаем библиотеку Servo
#include <Servo.h>

// Пин для сервопривода
int servoPin = 3;
// Создаем объект
Servo Servo1;

void setup() {
  // Нам нужно подключить сервопривод к используемому номеру пина
  Servo1.attach(servoPin);
}

void loop(){
  Servo1.write(0);// 0 градусов
  delay(1000);
  Servo1.write(30);// 30 градусов
  delay(1000);
  Servo1.write(60);// 60 градусов
  delay(1000);
  Servo1.write(90);// 90 градусов
  delay(1000);
  Servo1.write(120);// 120 градусов
  delay(1000);
  Servo1.write(150);// 150 градусов
  delay(1000);
  Servo1.write(180);// 180 градусов
  delay(1000);
}
```

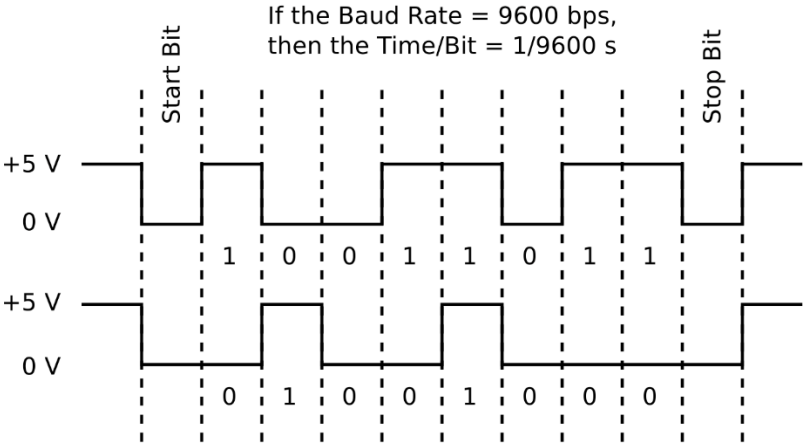
UART интерфейс. Bluetooth HC-06

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.



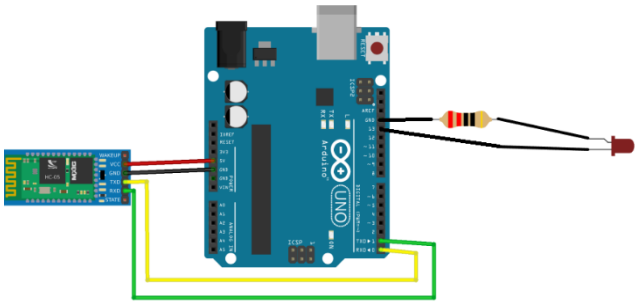
Проект №8
Сложность: легко

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:

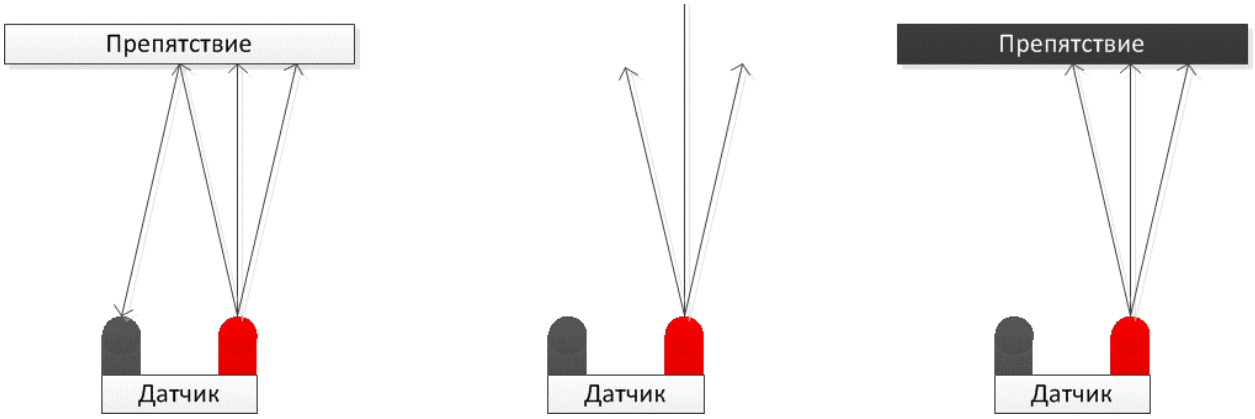
```
int val;
int LED = 13;
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);
}
void loop()
{
  if (Serial.available())
  {
    val = Serial.read();
    // При символе "1" включаем светодиод
    if (val == '1')
    {
      digitalWrite(LED, HIGH);
    }
    // При символе "0" выключаем светодиод
    if ( val == '0')
    {
      digitalWrite(LED, LOW);
    }
  }
}
```

Датчик распознавания преград. YL-63

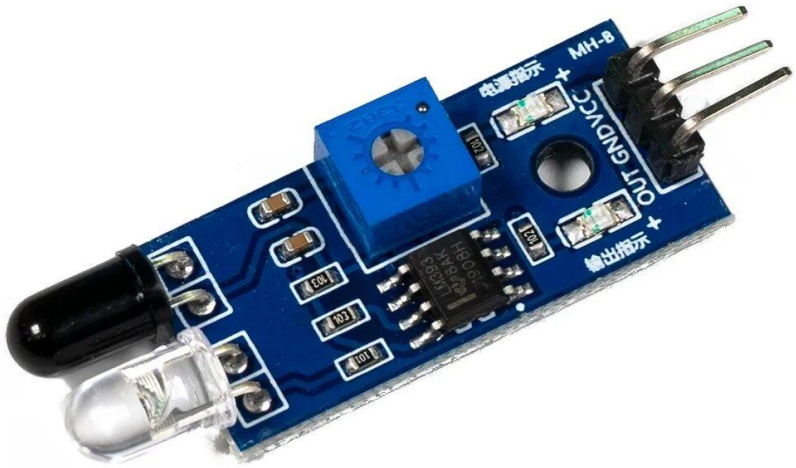
Теория

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.



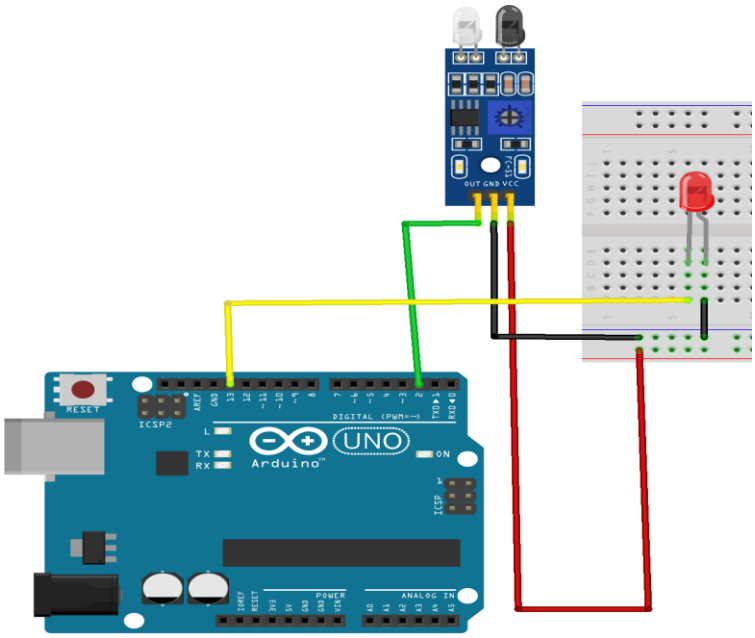
Основная микросхема ИК датчика препятствия, это компаратор LM393, который производит сравнение уровней напряжений на входах INB- и INB+ Чувствительность порога срабатывания задается с помощью потенциометра и в результате сравнений на выходе OUTB микросхемы, формируется логический «0» или логическая «1».



Проект №9
Сложность: легко

Задача: При обнаружении препятствия – загорается светодиод

Схема проекта:



Код проекта:

```
const int svPin = 2; // пин данных с датчика
const int ledPin =  13; // пин для светодиода
int svState = 0;         // переменная для хранения данных

void setup() {

  pinMode(ledPin, OUTPUT); //светодиод

  pinMode(svPin, INPUT); // прием данных
}

void loop() {

  svState = digitalRead(svPin); // считываем значение

  if (svState == 0) { //
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}

```

Термодатчик на терморезисторе. MF52 NTC-MF52AT 3950B

Теория

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

Существует два вида термисторов: PTC — 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 A, B и C, которые нужно брать из спецификации к датчику. Но в нашем случае не требуется большой точности, воспользуемся модифицированным уравнением (B-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 B, который для NTC термистора равен 3950. Остальные параметры нам уже известны:

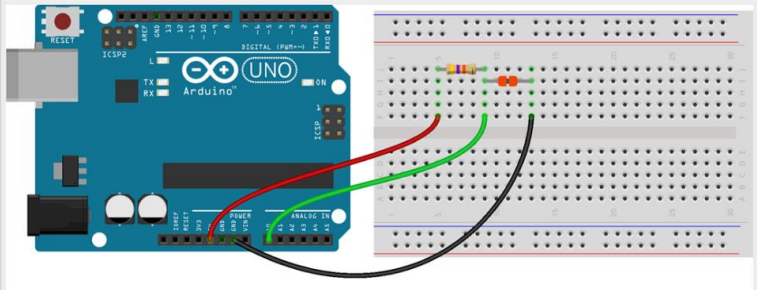
- T0 — комнатная температура в Кельвинах, для которой указывается номинал термистора; T0 = 25 + 273.15;
- T — искомая температура, в Кельвинах;
- R — измеренное сопротивление термистора в Омах;
- R0 — номинальное сопротивление термистора в Омах.

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:

```
#define B 3950 // B-коэффициент
#define SERIAL_R 90 // сопротивление последовательного резистора, 100 Ом
#define THERMISTOR_R 4350 // номинальное сопротивления термистора, 5 кОм
#define NOMINAL_T 25 // номинальная температура (при которой TR = 1-00 кОм)

const byte tempPin = A0;

void setup() {
    Serial.begin( 9600 );
    pinMode( tempPin, INPUT );
}

void loop() {
    int t = analogRead( tempPin );
    float tr = 1023.0 / t - 1;
    tr = SERIAL_R / tr;
    Serial.print("Сопротивление=");
    Serial.print(tr);
    Serial.print(", Температура=");

    float steinhart;
    steinhart = tr / THERMISTOR_R; // (R/Ro)
    steinhart = log(steinhart); // ln(R/Ro)
    steinhart /= B; // 1/B * ln(R/Ro)
    steinhart += 1.0 / (NOMINAL_T + 273.15); // + (1/To)
    steinhart = 1.0 / steinhart; // Invert
    steinhart -= 273.15;
    Serial.println(steinhart);
    delay(100);
}
```

Датчик температуры. DS18B20

DATASHEET

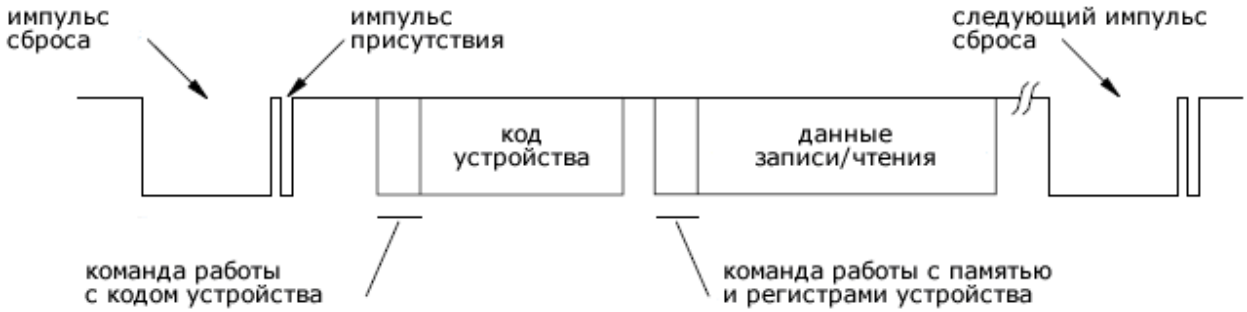
Теория

DS18B20 – это цифровой термометр, способный измерять температуру в диапазоне от -55 до +125 с программируемой точностью 9-12 бит. Каждому датчику присваивается свой уникальной 64-битный адрес, а обмен информацией платой ардуино 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.

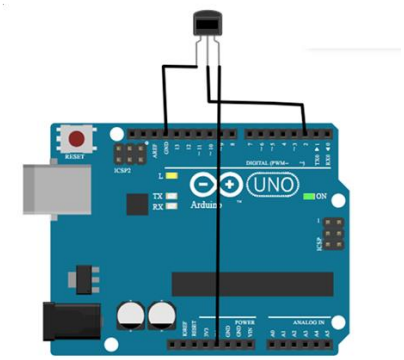


Протокол 1 Wire, нашел своё применение в домофонных ключах (iButton) или используется в системах умного дома для опроса большого количества датчиков.

Проект №11
Сложность: легко

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

Схема подключения:



Код проекта:

Необходимо установить следующие библиотеки:

- OneWire.h
- DallasTemperature.h

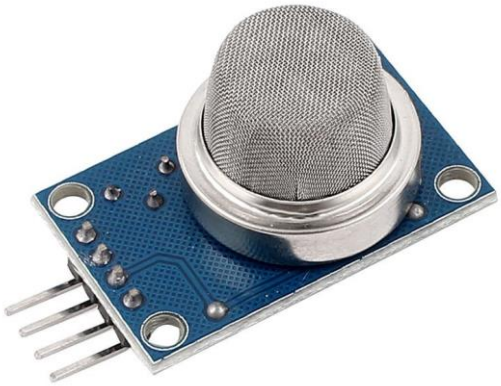
```
// Подключаем библиотеку
#include <OneWire.h>
#include <DallasTemperature.h>
// Определяем пин подключения данных
#define ONE_WIRE_BUS 2

// Запускаем протокол oneWire
OneWire oneWire(ONE_WIRE_BUS);
// Определяем тип датчика
DallasTemperature sensors(&oneWire);

void setup(void)
{
  // Включаем передачу данных через консоль
  Serial.begin(9600);
  Serial.println("Hello user!");
  sensors.begin(); //запускаем счет данных с датчика
}
void loop(void)
{
  Serial.print("Получаю адресс датчика");
  sensors.requestTemperatures(); // Получаем адресс датчика
  Serial.println("Готово");
  float tempC = sensors.getTempCByIndex(0); // Присваеваем переменной значение

  // Check if reading was successful
  if(tempC != DEVICE_DISCONNECTED_C)
  {
    Serial.print("Температура ");
    Serial.println(tempC);
   }
   else
   {
    Serial.println("Ошибка");
   }
}
```

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.

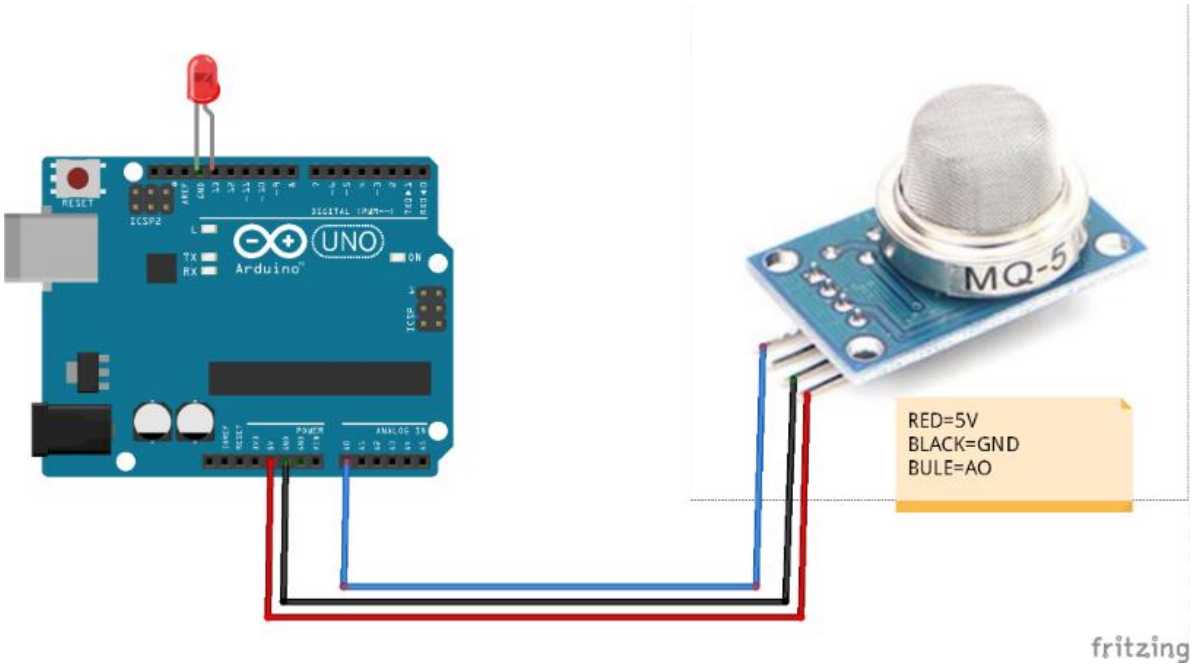
Напряжение аналогового выхода изменяется пропорционально концентрации дыма или газа. Чем выше концентрация газа, тем выше выходное напряжение. Логический сигнал можно откалибровать, держа датчик рядом с дымом, который вы хотите обнаружить. Далее вращайте потенциометр по часовой стрелке (для увеличения 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:

```
const int gasSensorPin = A0;       // Пин, к которому подключен аналоговый выход датчика газа
const int digitalOutPin = 2;       // Пин, к которому подключен цифровой выход датчика газа

void setup() {

  pinMode(gasSensorPin, INPUT);      // Установка пина аналогового входа как вход
  pinMode(digitalOutPin, INPUT);     // Установка пина цифрового входа как вход
  Serial.begin(9600);                // Начало работы с последовательным портом на скорости 9600 бит/сек
}

void loop() {
  int analogValue = analogRead(gasSensorPin);  // Считывание аналогового значения с датчика газа
  int digitalValue = digitalRead(digitalOutPin);  // Считывание цифрового значения с датчика газа



  if (digitalValue == HIGH) {           // Проверка цифрового значения
    Serial.println("Smoke: Not detected");  // Ввывод сообщения в монитор последовательного порта
  } else {
    Serial.println("Smoke: Detected!");      // Вывод сообщения в монитор последовательного порта
  }
}
```

Датчик протечки воды

Теория

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

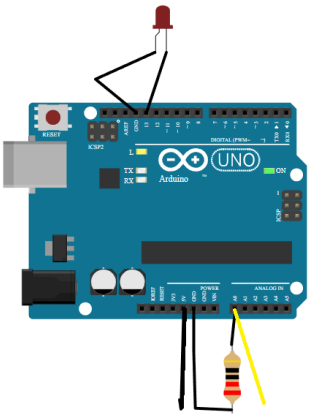
Логика следующая: два контакта, опускаются в воду, на одном контакте 5 вольт, на другом контакт от АЦП (необходимо подтянуть через резистор к GND)

Вода проводит электричество, следовательно, контакты замкнуться.

Проект №13
Сложность: легко

Задача: При протечке воды загорается светодиод.

Схема проекта:



Код проекта:

```
int Water = A0;
int LED = 13;
int waterLvl;
void setup() {
  // put your setup code here, to run once:
  pinMode(Water,INPUT);
  pinMode(LED,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  waterLvl = analogRead(Water);
  if (waterLvl > 800){ //При достяжении порога срабатывания, зажигем светодиод
    digitalWrite(LED,HIGH);
  }
  else{
    digitalWrite(LED,LOW);
  }
}
```

Датчик освещённости на фоторезисторе

Теория

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

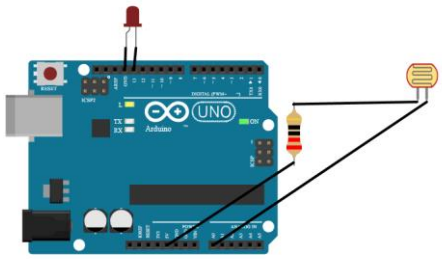
Между двумя токопроводящими электродами размещается полупроводник. В том случае если свет не попадает на полупроводник, то его сопротивление имеет высокое значение. Как только на полупроводник попадает свет, его сопротивление снижаться.

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:

```
int Light = A0;
int LED = 13;
int LightLvl;
void setup() {
  // put your setup code here, to run once:
  pinMode(Light,INPUT);
  pinMode(LED,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  LightLvl = analogRead(Light);
  if (LightLvl > 500){ //При достяжении порога срабатывания, зажигем светодиод
    digitalWrite(LED,LOW);
  }
  else{
    digitalWrite(LED,HIGH);
  }
}
```

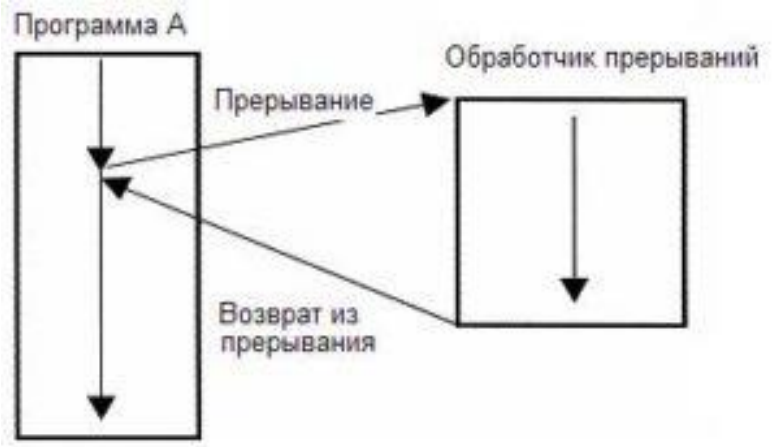
Внешнее аппаратное прерывание

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 МК[1].

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(){ // Назвавние 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

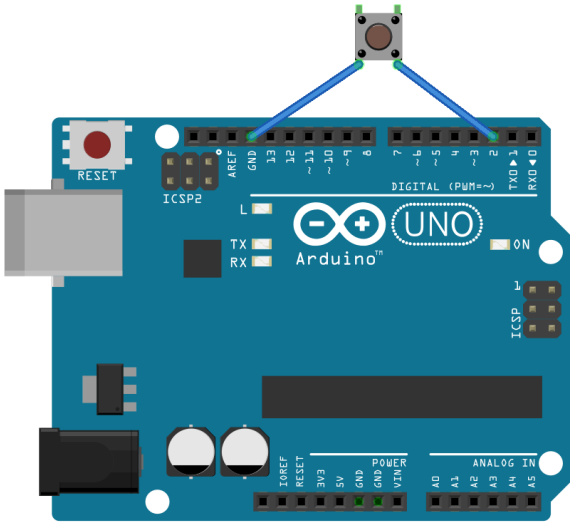
Проект №15
Сложность: легко

Прерывания очень удобно использовать, если в очень большом коде, для задавания 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:



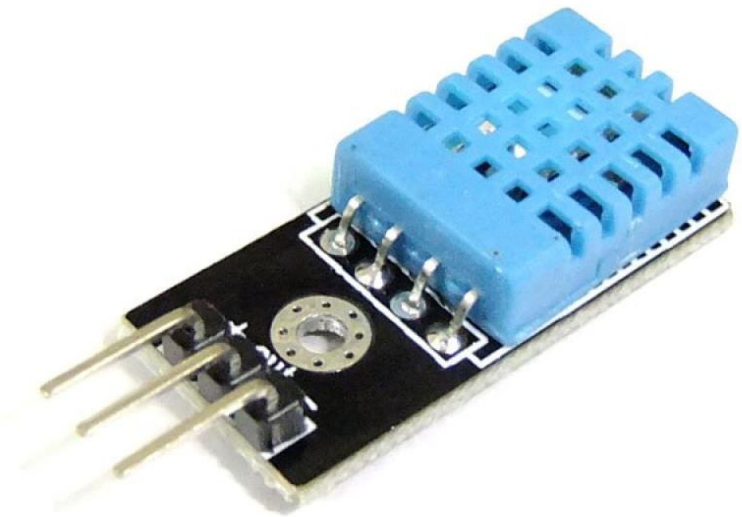
Код проекта:

```
volatile int counter = 0;  // переменная-счётчик

void setup() {
  Serial.begin(9600); // открыли порт для связи

  pinMode(2, INPUT_PULLUP);  // подключили кнопку на 2 пин и GND


  attachInterrupt(0, btnIsr, FALLING); // FALLING - при нажатии на кнопку будет сигнал 0,
}

void btnIsr() {
  counter++;  // + нажатие
}

void loop() {
  Serial.println(counter);  // выводим
  delay(5000);              // ждём
}
```

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 обладают высоким быстродействием и точностью, но зато прост, недорог и отлично подходят для контроля влажности в помещении.



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

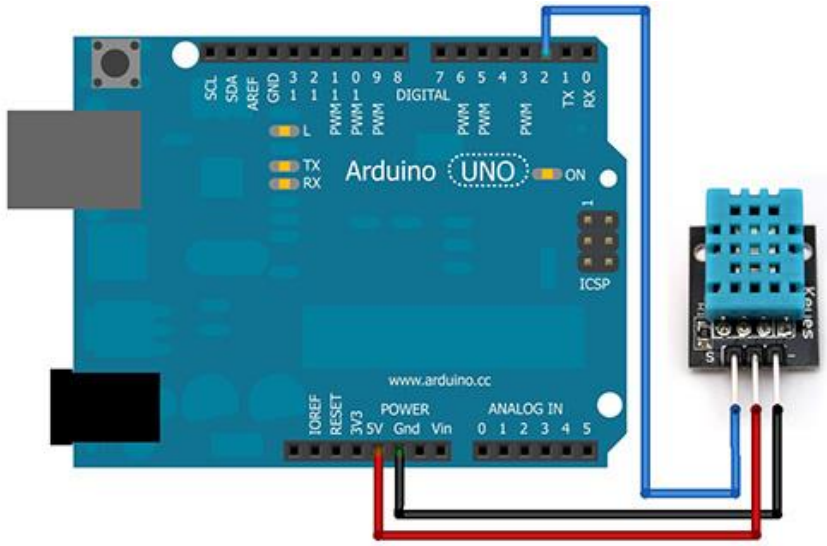
Емкостной датчик влажности — это конденсатор с переменной емкостью, который содержит токопроводящие обкладки из медной фольги на текстолите. Этот конденсатор заключен в герметичный чехол, поверх которого расположен влагопоглощающий слой. При попадании частиц воды на этот слой, меняется его диэлектрическая 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 сторнюю библиотеки:

- DHT
- Adafruit_Sensor

```
#include <DHT.h>      // подключаем библиотеку для датчика
DHT dht(2, DHT11);  // сообщаем на каком порту будет датчик

void setup() {
  dht.begin();               // запускаем датчик DHT11
  Serial.begin(9600);   // подключаем монитор порта
}

void loop() {
  // считываем температуру (t) и влажность (h)
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // выводим температуру (t) и влажность (h) на монитор порта
  Serial.print("Влажность: ");
  Serial.println(h);
  Serial.print("Температура: ");
  Serial.println(t);
  delay(2000);
}
```

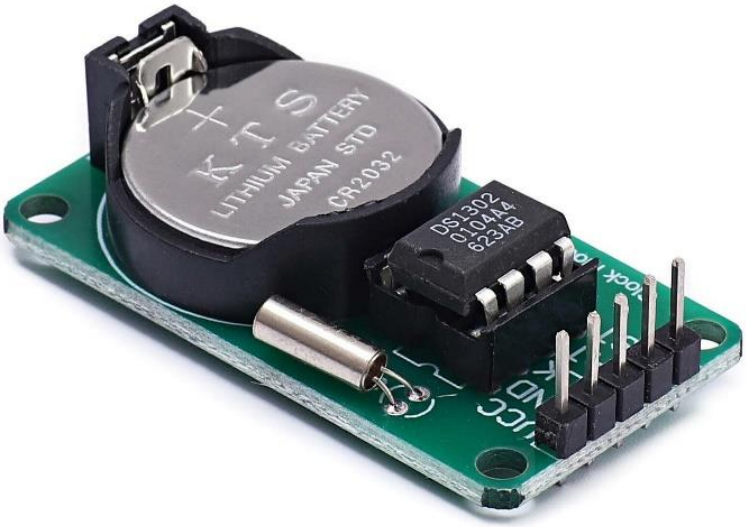
Часы реального времени. DS1302

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 так же следит за количеством дней в месяце и делает поправку на високосный год. Работает в 24-часовом или 12-часовом формате с индикатором AM/PM.



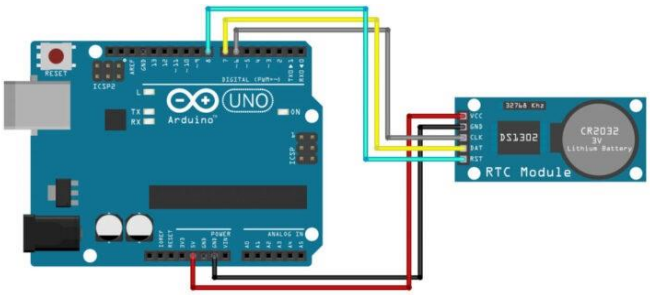
Одной из причин использовать отдельную микросхему счета времени – это банальное энергопотребление. Выгоднее заставить заниматься этим отдельную микросхему с малым потреблением тока, чем оставлять включённым целый прибор.

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**

```
#include <ThreeWire.h>                          // Подключаем библиотеку ThreeWire
#include <RtcDS1302.h>                          // Подключаем библиотеку RtcDS1302

ThreeWire myWire(7,6,8);                        // Указываем вывода IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);

void setup ()
{
    Serial.begin(9600);                         // Установка последовательной связи на скорости 9600
    Serial.print("Дата: ");                     // Отправка данных на последовательный порт
    Serial.println(__DATE__);                   // Получение даты и времени с ПК
    Serial.print("Время: ");                    // Отправка данных на последовательный порт
    Serial.println(__TIME__);                   // Получение даты и времени с ПК
    Rtc.Begin();                                // Инициализация RTC
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__); // Копирование даты и времени в compiled
    Rtc.SetDateTime(compiled);                  // Установка времени
    Serial.println();                           // Отправка данных на последовательный порт
}

void loop ()
{
  Rtc.GetDateTime();
  RtcDateTime now = Rtc.GetDateTime();
  Serial.print("Дата RTC ");                    // Отправка данных на последовательный порт
  Serial.print(now.Month());                    // Отправка месяца
  Serial.print(".");                            // Отправка данных на последовательный порт
  Serial.print(now.Day());                      // Отправка дня
  Serial.print(".");                            // Отправка данных на последовательный порт
  Serial.print(now.Year());                     // Отправка года
  Serial.print(" Время RTC ");                  // Отправка данных на последовательный порт
  Serial.print(now.Hour());                     // Отправка часа
  Serial.print(":");                            // Отправка данных на последовательный порт
  Serial.print(now.Minute());                   // Отправка минут
  Serial.print(":");                            // Отправка данных на последовательный порт
  Serial.println(now.Second());                 // Отправка секунд
  delay(2000);                                  // Пауза 2 с
}
```

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 называние 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 ШИМ[2] 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

attachInterrpupt(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**

interrups() 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 usigned 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.

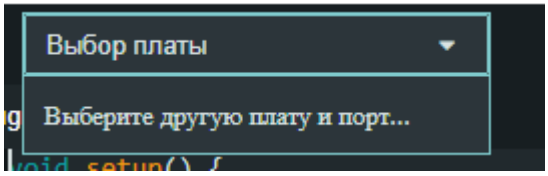
Возвращает:

- 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.

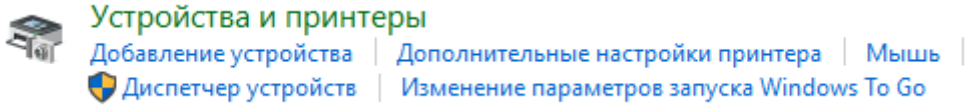
Arduino не отображается в селекторе выбора



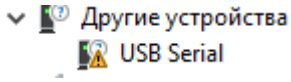
Решение:

Проверьте, что компьютер видит плату через “диспетчер устройств”

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



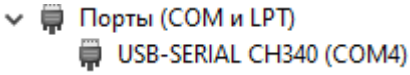
Если имеется следующее предупреждение:



Значит у вас не установлен драйвер.

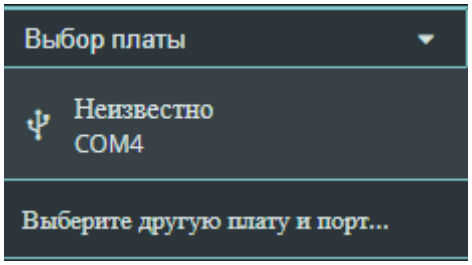
[Инструкция по установке драйвера]

Если компьютер ддолжен видить плату Arduino, как устройство с COM портом:

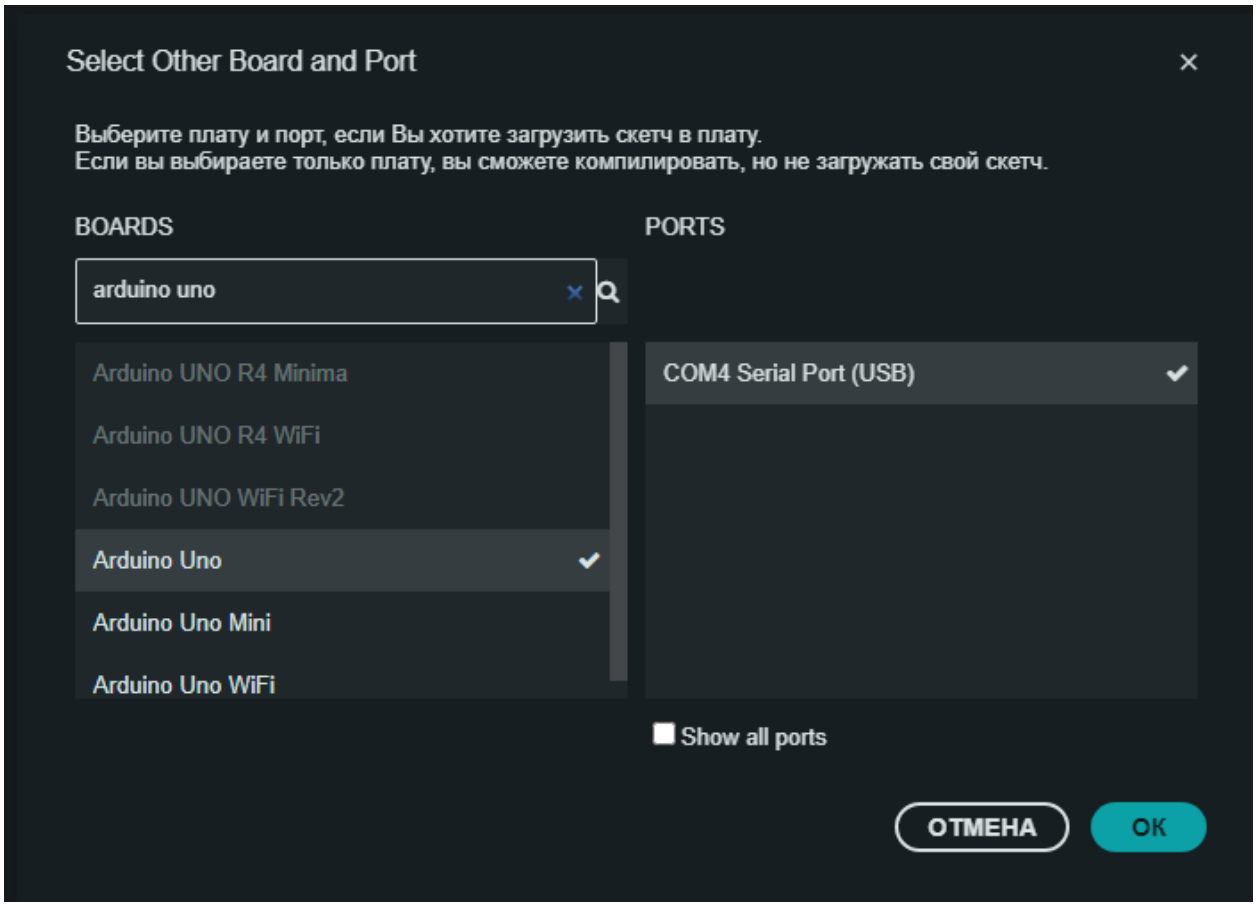


Необходдимо запомнить номер COM порта (в данном случае COM4)

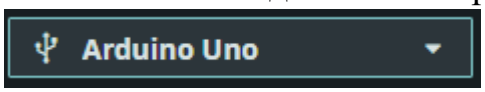
Переходим в Arduino IDE, в селеекторе выбора платы указываем нужный порт



В всплывающем окне вбиваем название платы и нажимаем “OK”



Название платы должно отображаться **жирным шрифтом**.



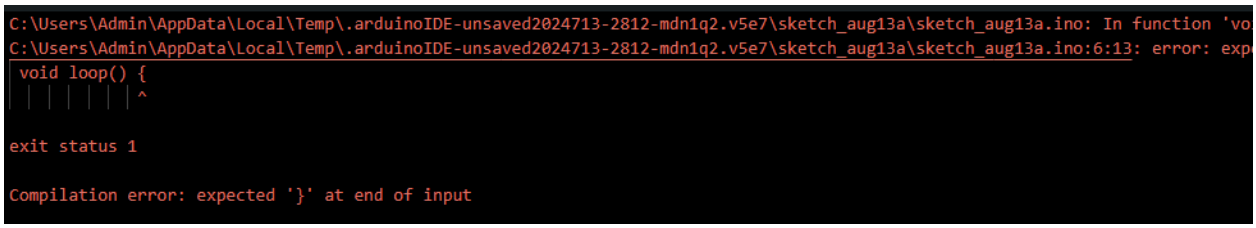
Ошибка загрузки

```
Failed uploading: no upload port provided
```

Решение:

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

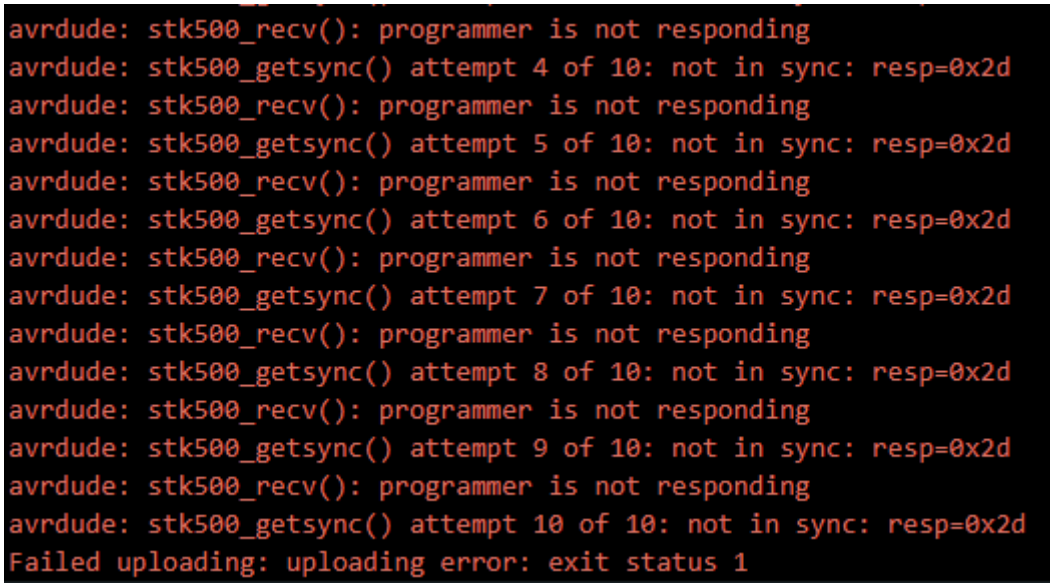
Ошибка сборки проекта



Решение:

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

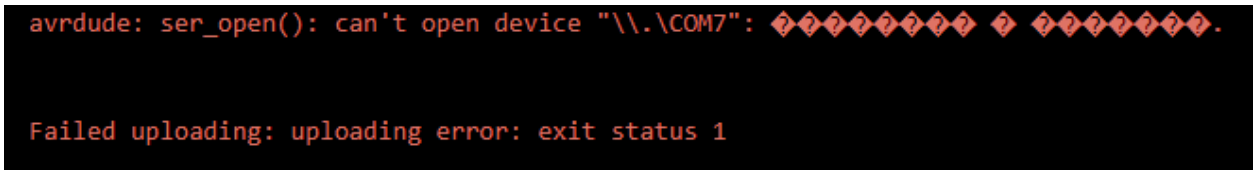
Выбран не верный COM порт для загрузки



Решение:

Выберите нужную плату и повторите попытку загрузки

COM порт занят или отсутствует



Решение:

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

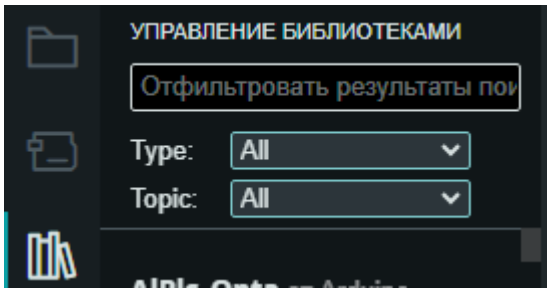
Отсутствует библиотека

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

Данная ошибка означает, что не установленна библеотека

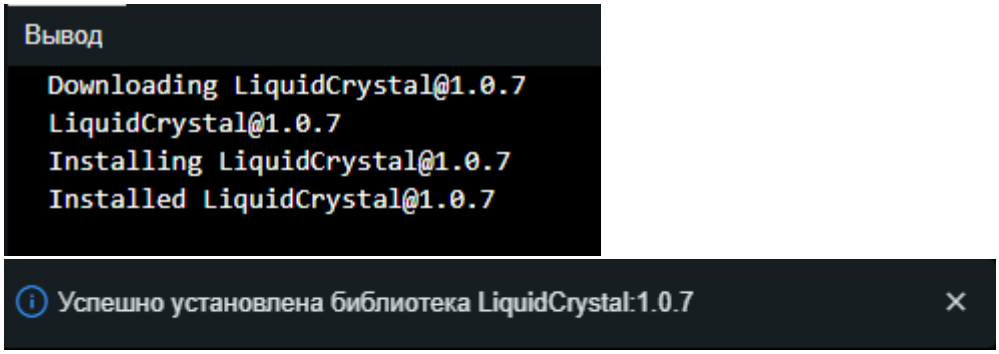
Решение:

Введиите название библиотеки в менеджер библиотек



Нажмите кнопку “УСТАНОВКА”

Дождитесь окончания установки и попробуйте залить скейч заново



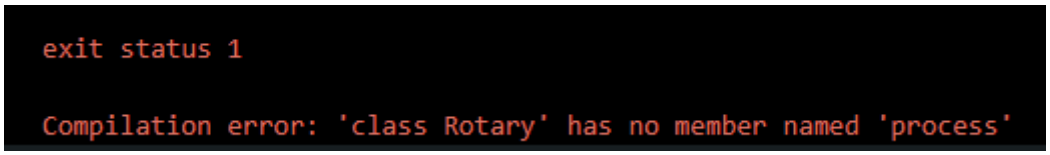
Отсутствует возможность подключиться к плате

```
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



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.