ПРОЕКТНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА НОМЕР 1.

2023-2024г

Название проекта: Разработка системы управления умным домом на основе микроконтроллера Arduino.

Цель проекта: Создание функциональной системы управления умным домом с использованием микроконтроллера Arduino, обеспечивающей автоматизацию основных процессов в доме для повышения комфорта и безопасности жильцов.

Основная задача заключается в передаче и приеме сигналов (команд) блоку управления.

АРДУИНО.

1.1 Основные понятия.

Arduino - это открытая платформа для создания простых электронных проектов. Она состоит из аппаратной платформы, включающей микроконтроллеры и набор различных входов и выходов, а также средств программирования, позволяющих разрабатывать программное обеспечение для управления этой платформой.

Основные компоненты Arduino:

- **Микроконтроллер:** Arduino использует различные типы микроконтроллеров, такие как ATMega328, ATMega2560 и другие. Эти микроконтроллеры являются мощными, но в то же время достаточно простыми в использовании.
- **Цифровые и аналоговые входы/выходы:** Arduino имеет ряд цифровых и аналоговых портов, которые могут использоваться для подключения датчиков, кнопок, светодиодов, моторов и других устройств.
- **Шина питания:** Arduino имеет встроенные возможности для подачи питания на подключенные устройства, что делает его удобным для использования в различных электронных проектах.
- **USB-интерфейс:** Большинство Arduino-плат имеют встроенный USB-интерфейс, который обеспечивает подключение к компьютеру для загрузки программ и взаимодействия с ними.

1.2 Среда разработки.

Среда разработки Arduino IDE (Integrated Development Environment) представляет собой специализированное программное обеспечение, разработанное для создания и загрузки программного кода на микроконтроллеры Arduino. Она предоставляет удобный интерфейс для написания кода, проверки его на ошибки, компиляции и загрузки на Arduino-плату.

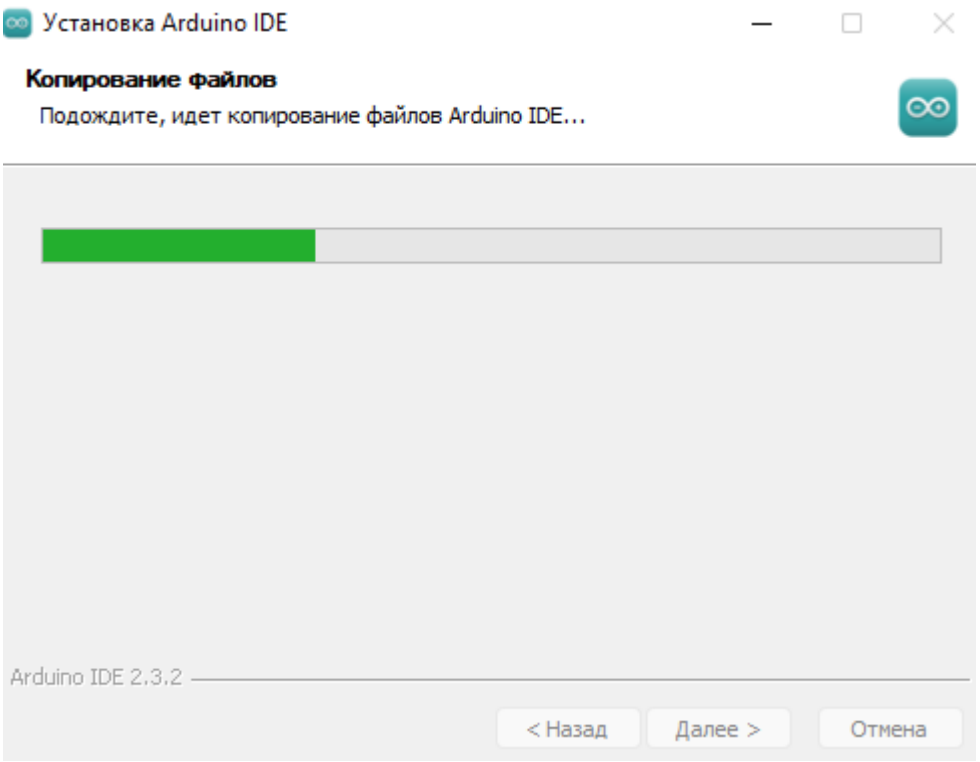
Основные функции Arduino IDE включают в себя:

- **Текстовый редактор:** Встроенный текстовый редактор предоставляет удобное место для написания кода на языке Arduino, подсвечивая синтаксис и автоматический отступ.
- **Компиляция и загрузка:** Arduino IDE автоматически компилирует код и загружает его на подключенную Arduino-плату через USB-интерфейс.

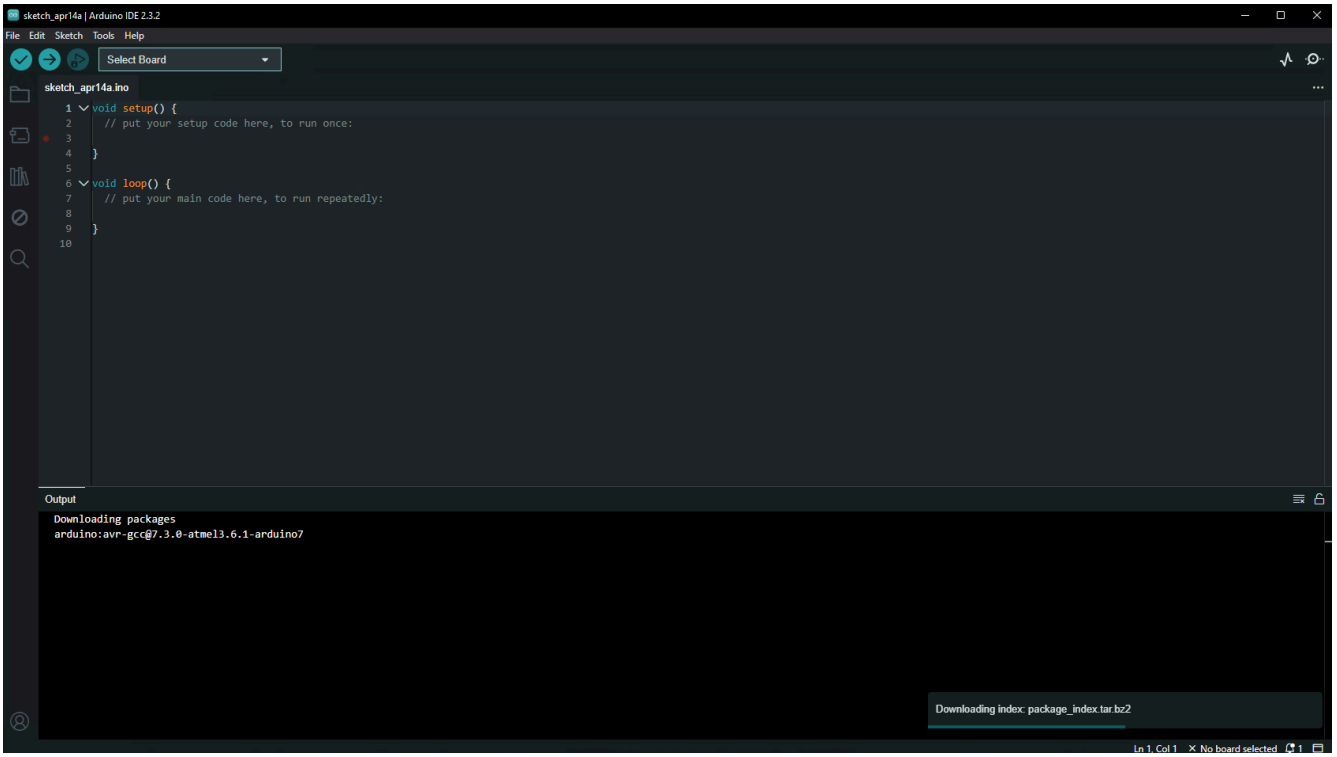
- **Монитор порта:** Встроенный монитор порта позволяет отправлять и принимать данные через последовательный порт Arduino, что полезно для отладки и мониторинга работы программы.
- **Библиотеки:** Arduino IDE поставляется с набором стандартных библиотек, а также предоставляет возможность установки дополнительных библиотек для расширения функциональности.
- Примеры кода, Интеграция с отладочными инструментами и др.

Arduino IDE доступна для различных операционных систем, включая Windows, macOS и Linux, и является бесплатной для загрузки с официального сайта Arduino.

скачать можно тут: https://www.arduino.cc/en/software

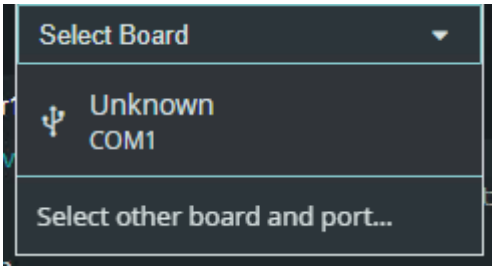


Первый запуск IDE, базовое окно:

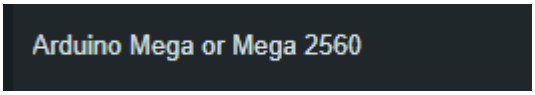


1.3 Выбор платы.

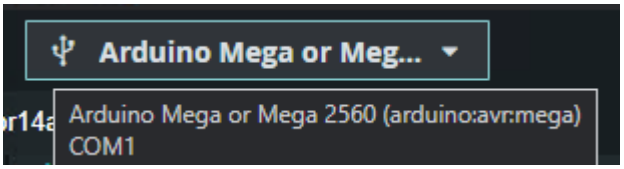
нажмите на **SELECT BOARD** -> UNKNOWN, COM1 для выбора платы:



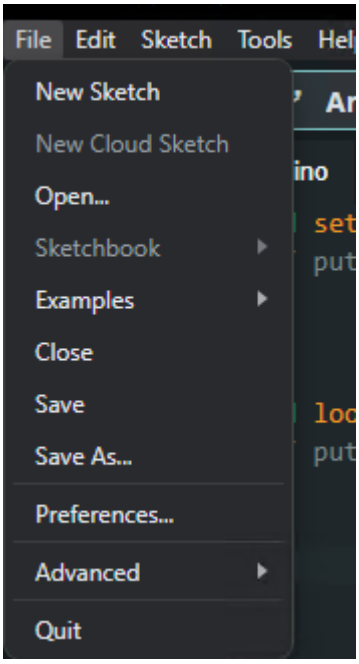
В поиске найдите такую вот плату:



После выбора в SELECT BOARD измениться (примерно так)



SKETCH в проекте это новый файл с кодом. Для его создания - можно перейти в верхней панели в раздел File -> New Sketch. (но мы не будем уже создавать новый файл sketch)



1.4 Setup/Loop.

- **setup():**
  - функция вызывается один раз при запуске Arduino.
  - Она используется для инициализации переменных, настройки пинов ввода-вывода, установки начальных параметров и выполнения других операций, которые требуются только один раз при запуске устройства.
  - Внутри setup() обычно выполняется начальная конфигурация устройства, например, настройка скорости последовательного порта или установка пинов ввода-вывода в режим входа или выхода.
- **loop():**
  - функция вызывается после завершения функции setup() и выполняется бесконечный цикл.
  - Она используется для выполнения основной логики программы, которая должна выполняться повторно или в цикле.
  - Внутри loop() обычно находится основной код программы, который будет выполняться снова и снова, пока Arduino не будет отключен или сброшен.
  - Основной код может включать в себя считывание датчиков, управление актуаторами, отправку и прием данных через последовательный порт и другие операции.

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

void loop() {
  // put your main code here, to run repeatedly:
}
```

1.5 Серийный порт.

Серийный порт (Serial port) - это интерфейс ввода-вывода, который позволяет передавать данные между устройствами посредством последовательной передачи битов. В Arduino он обычно используется для связи с компьютером или другими устройствами.

```
// Определение используемого серийного порта
#define SERIAL_PORT Serial


void loop() {}
```

1.6 Команды.

Вот список основных команд работы с объектом **SERIAL_PORT** (*обычно это объект типа Serial*), который используется для взаимодействия с последовательным портом в Arduino:

- **begin(baudrate):** Начинает передачу данных через серийный порт с указанной скоростью передачи baudrate в битах в секунду. Например, SERIAL_PORT.begin(9600) устанавливает скорость передачи 9600 бит/с.
- **print(data):** Отправляет данные data в байтовом виде через серийный порт без добавления символа новой строки в конце. Эта команда может использоваться для отправки чисел, символов и строк.
- **println(data):** Отправляет данные data через серийный порт с добавлением символа новой строки (\n) в конце. Это полезно для форматированного вывода данных, так как новая строка переводит курсор на новую строку в мониторе порта.
- **available():** Возвращает количество доступных для чтения байтов из приемного буфера серийного порта. Это позволяет проверить, есть ли данные, которые могут быть прочитаны.

- **read():** Считывает следующий байт из приемного буфера серийного порта и возвращает его как целочисленное значение. Эта команда часто используется в цикле для последовательного чтения входящих данных.
- **flush():** Очищает буферы приема и передачи серийного порта. Это полезно, если вам нужно убедиться, что все данные были отправлены или приняты до перехода к следующей операции.
- **write(data):** Отправляет байт данных data через серийный порт. Это можно использовать для отправки отдельных байтов, например, управляющих команд или двоичных данных.

```
SERIAL_PORT. название-команды()
```

1.7 Как сделать задержку??.

Существует специализированная команда для создания задержки выполнения определенного запроса к порту:

```
delay(1000);
```

РЕАЛИЗАЦИЯ ПРОЕКТА.

План действий таков:

- запустить IDE
- выбрать плату для работы
- создать sketch файл (или исп-ть уже созданный ранее)
- сделать объявление серийного порта (#define …)
- реализовать две основные функции setup, loop.
- сформулируйте выводы о проделанной работе (подготовьте грамотный отчет с картинками, кодом, и др)

setup должен что делать:

- начало работы с портом и установка скорости (поставьте как в примере 9600)

loop должен делать:

- отправление фразы (любой) через порт
- блок ожидание получения данных
- блок обработки полученных данных (исп-я while)
- блок вывода принятой фразы в монитор порта
- сделать небольшую задержку перед повторной отправкой данных.