Сборник IoT-рецептов

Содержание

1 Подготовка к работе

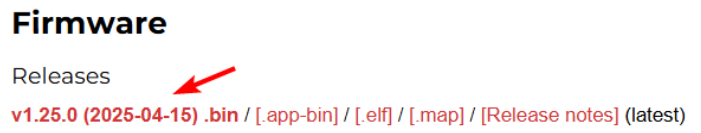
1.1 Назначение пинов ESP32

1.2 Программирование ESP32 с помощью Python

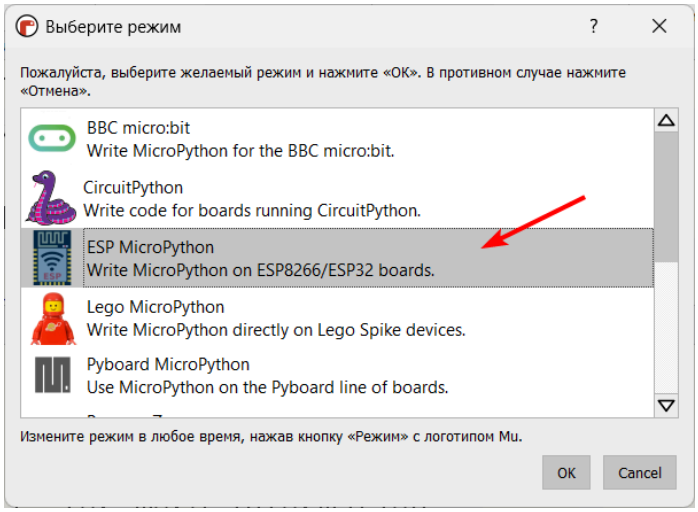
Скачайте и установите среду разработки Mu.

Скачайте USB-to-UART драйвер:

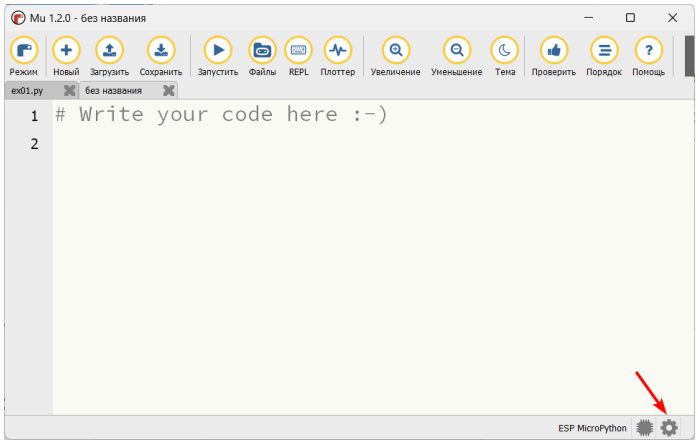
Скачайте MicroPython-прошивку последней версии (отмечается словом latest):



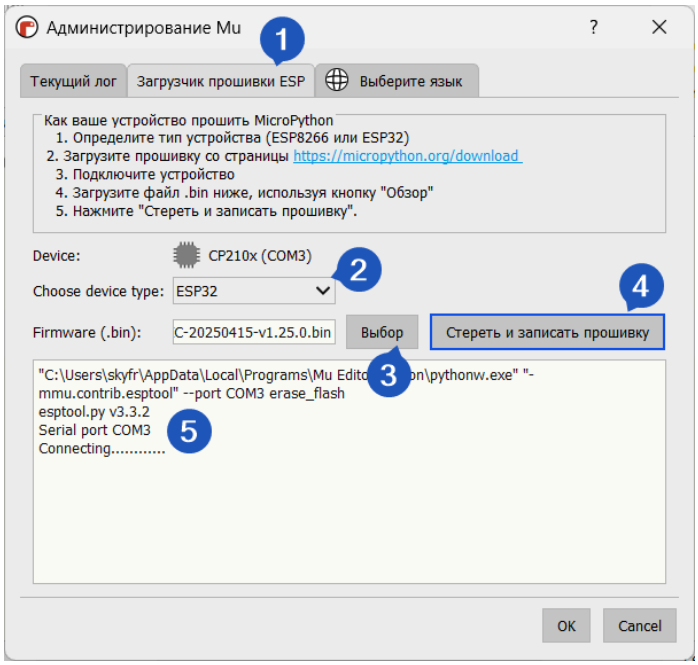
Подключите плату ESP32 к компьютеру и запустите редактор Mu. Нажмите кнопку **Режим**. Выберите режим *ESP MicroPython* и нажмите кнопку **OK**:



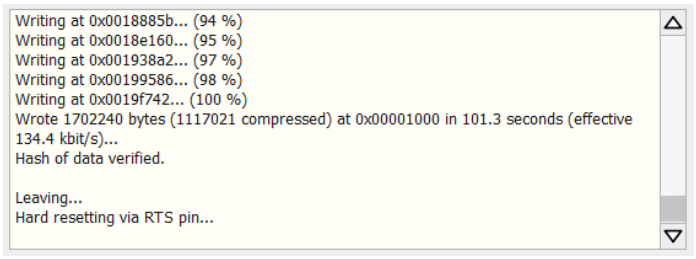
Нажмите на иконку шестерёнки, чтобы открыть настройки:



Перейдите на вкладку **Загрузчик прошивки ESP** (1). Выберите в списке тип устройства - *ESP32* (2). Нажмите кнопку **Выбор** (3) и выберите файл с MicroPython-прошивкой. Нажмите кнопку **Стереть и записать прошивку** (4). Когда появится сообщение `Connecting...` (5), нажмите и удерживайте 2 секунды кнопку `BOOT` на микроконтроллере.



Появления следующего текста



означает, что загрузка завершена. Нажмите кнопку **OK**.

Проверьте работу прошивки следующей программой:

```
from time import sleep
from machine import Pin

led = Pin(2, Pin.OUT)

led.on()
sleep(3)
led.off()
```

Нажмите кнопку **Запустить**. На микроконтроллере загорится встроенный светодиод и погаснет через 3 секунды.

1.3 Установка Python-модулей

1.4 Моделирование схем с микроконтроллером

1.5 Рисование схем с прототипом устройств

2 Подключение датчиков

2.1 Кнопка

Задача: получать в коде состояние кнопки - нажата / не нажата. Состояние для нажатой кнопки - 1, для не нажатой - 0.

Схема подключения:

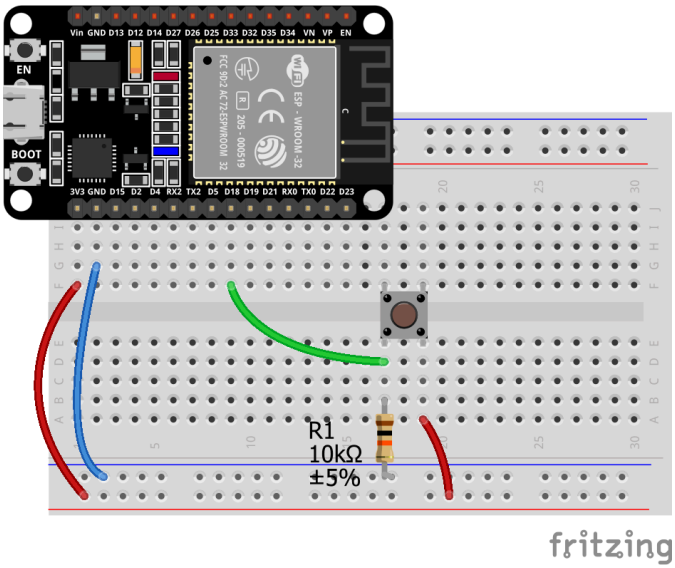


Рисунок 1. Кнопка

Код:

```
from machine import Pin
from time import sleep
```

```
### Обязательный код
BUTTON_PIN = 18 # пин подключения кнопки
button = Pin(BUTTON_PIN, Pin.IN) # объект-кнопка
### Конец обязательного кода

while True:
    button_state = button.value() # считываем
значение кнопки
    print(button_state)
    sleep(0.1)
```

Можно обойтись без стягивающего резистора R1. Для этого при создании объекта кнопки указывается тип `Pin.PULL_DOWN`:

```
button = Pin(BUTTON_PIN, Pin.PULL_DOWN)
```

2.2 Кнопка-переключатель

Задача: нажатие кнопки последовательно переключает состояние переменной из `True` в `False` и наоборот.

Схема подключения:

См. Рисунок 1.

Код:

```
from machine import Pin
from time import sleep

BUTTON_PIN = 18
button = Pin(BUTTON_PIN, Pin.IN)

button_state = False # текущее состояние кнопки
prev = False # предыдущее состояние кнопки
state = False # переменная с чередующумся
логическим значением

while True:
    # сохраняем текущее состояние кнопки
    button_state = button.value()
    """
    Если состояние изменилось с False на True,
    значит кнопка нажата
    """
    if button_state == True and prev == False:
        # меняем значение переменной на
противоположное
        state = not state

    # текущее состояние кнопки сохраняем как
предыдущее
    prev = button_state

    print(state)
    sleep(0.1)
```

2.3 Потенциометр

Задача: прочитать показания фоторезистора.

Схема подключения:

2.3.a Прототип

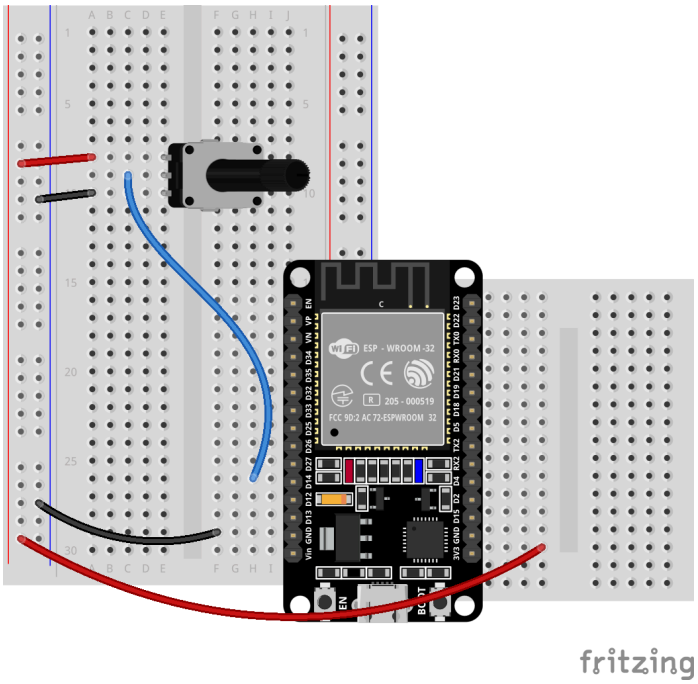


Рисунок 2. Потенциометр

2.3.b Принципиальная

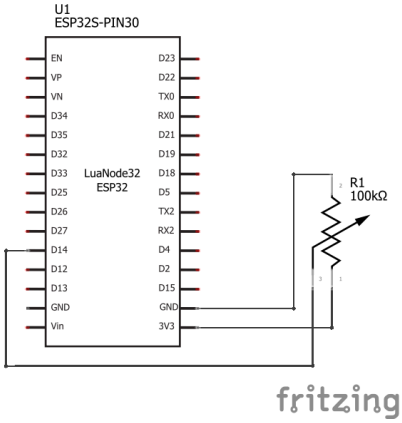


Рисунок 3. Потенциометр

Код:

```
from machine import Pin, ADC
from time import sleep

### Обязательный код
pot = ADC(Pin(14))
pot.width(ADC.WIDTH_12BIT)
pot.atten(ADC.ATTN_11DB)
### Конец обязательного кода

while True:
    sensor_val = pot.read()
    print(sensor_val)

    sleep(0.1)
```

2.4 Датчик освещённости

Задача: получить уровень освещенности с помощью фоторезистора.

Схема подключения:

2.4.a Прототип

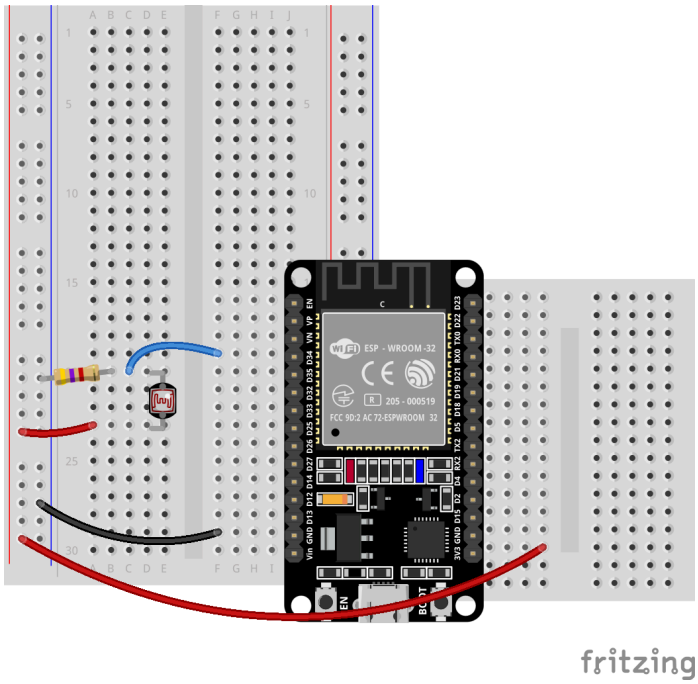


Рисунок 4. Фоторезистор

2.4.b Принципиальная

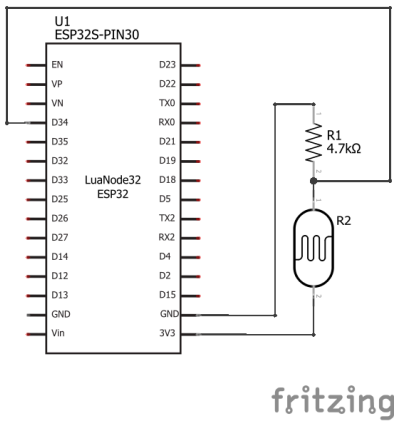


Рисунок 5. Фоторезистор

Код:

```
from machine import Pin, ADC
from time import sleep

### Обязательный код
ldr = ADC(Pin(34))
ldr.width(ADC.WIDTH_9BIT)
ldr.atten(ADC.ATTN_11DB)
### Конец обязательного кода

while True:
    sensor_val = ldr.read()
    print(sensor_val)

    sleep(0.1)
```

2.5 Датчик температуры и влажности

2.6 Датчик расстояния

3 Подключение исполнительных устройств

3.1 Двигатель постоянного тока

3.2 Серводвигатель

3.3 Шаговый двигатель

4 Подключение устройств вывода

4.1 Светодиод

Задача: программно включать и выключать светодиод.

Схема подключения:

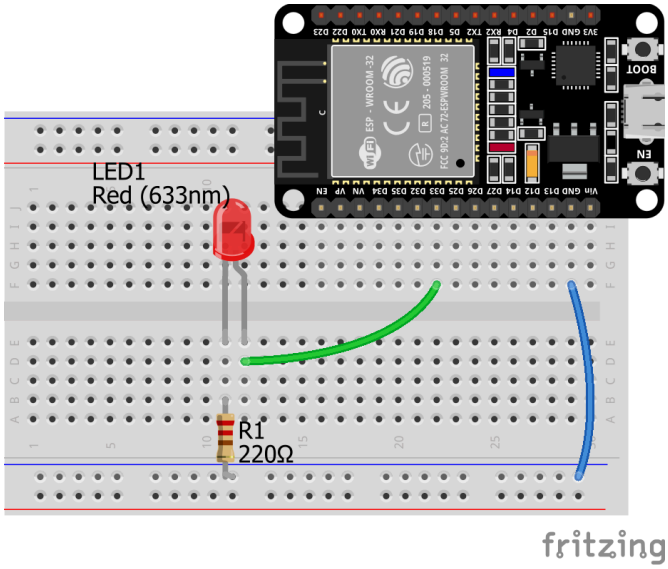


Рисунок 6. Светодиод

Код:

```
from machine import Pin
from time import sleep

LED_PIN = 33
led = Pin(LED_PIN, Pin.OUT)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

4.2 Управление яркостью светодиода

Задача: программно менять яркость светодиода.

Схема подключения:

См. Рисунок 6.

Код:

```
from machine import Pin, PWM
from time import sleep

### Обязательный код
```

```
LED_PIN = 33
led = Pin(LED_PIN, Pin.OUT)

led_pwm = PWM(led, 5000)
### Конец обязательного кода

while True:
    led_pwm.duty(1023)
    sleep(1)
    led_pwm.duty(512)
    sleep(1)
    led_pwm.duty(0)
    sleep(1)
```

4.3 ЖК-дисплей SSD1306

Задача: вывести на экран ЖК-дисплея информацию.

Скачайте python-модуль *ssd1306.py* по ссылке. Откройте его в редакторе Mu Editor. Подключите ESP32 к компьютеру.

Откройте вкладку **Файлы** (1). Перетащите модуль *ssd1306.py* из панели **Файлы на вашем компьютере** в панель **Файлы на вашем устройстве**. Теперь модуль для работы с ЖК-дисплеем SSD1306 будет доступен для импорта. Повторно нажмите кнопку **Файлы**, чтобы закрыть панель файлов.

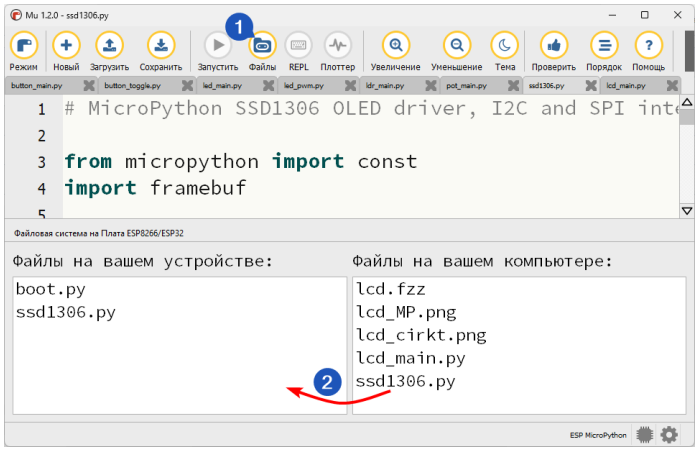


Схема для сборки:

4.3.a Прототип

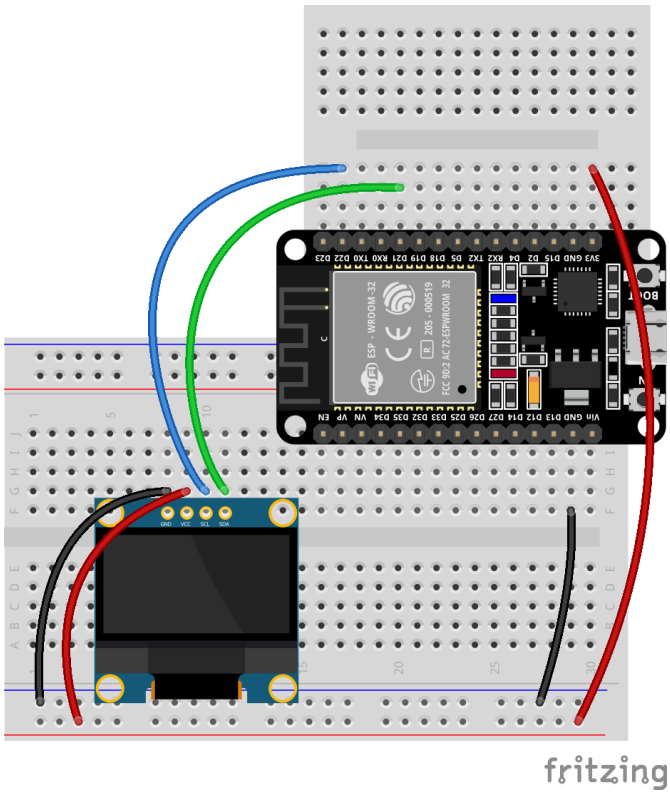


Рисунок 7. SSD1306

4.3.b Принципиальная

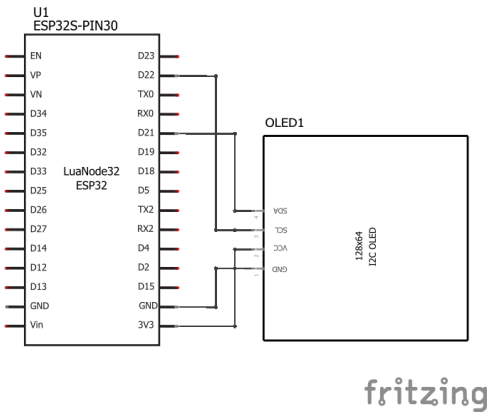


Рисунок 8. SSD1306

Код:

```
from machine import Pin, SoftI2C
import ssd1306

### Обязательный код
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))

oled_width = 128
oled_height = 64
display = ssd1306.SSD1306_I2C(oled_width,
oled_height, i2c)
### Конец обязательного кода

display.text("Hello", 0, 0)
display.show()
```

5 Передача данных

5.1 Подключение к WiFi

Задача: подключить ESP32 к WiFi с указанным SSID и паролем.

> ⚠ Предупреждение
>
> Для корректной работы, точка доступа должна использовать диапазон частот 2.4 ГГц.

Схема для сборки: не требуется.

> ⚠ Предупреждение
>
> Если ESP32 используется для считывания аналоговых сигналов, то одновременно с WiFi можно использовать для этого пины с 32 по 39. Это связано с тем что АЦП2 занят обеспечением работы WiFi-модуля. Перечисленные же ранее пины используют другой АЦП.

Код:

```
import machine
import network

### Обязательный код
wifi = network.WLAN(network.WLAN.IF_STA)

SSID = "esp"
KEY = "12345678"

if not wifi.isconnected():
    print('connecting to network...')
    wifi.active(True)
    wifi.connect(SSID, KEY)

    while not wifi.isconnected():
        machine.idle()

print('network config:', wifi.ipconfig('addr4'))
### Конец обязательного кода
```

Чтобы подключение происходило сразу после включения питания, сохраните код в файл `boot.py` и скопируйте на ESP32.

5.2 Протокол MQTT

Задача: подключиться к MQTT брокеру, чтобы получать и отправлять сообщения по данному протоколу.

Для подключения к MQTT брокеру понадобится модуль `umqttsimple.py`. Скачайте его по ссылке и скопируйте на ESP32.

Схема для сборки: не требуется.

Код:

Перед подключением к брокеру, ESP32 нужно подключить к интернет. Один из вариантов - подключение к WiFi (Глава 5.1).

```
import machine
import network
import time
from umqttsimple import MQTTClient

wifi = network.WLAN(network.WLAN.IF_STA)

SSID = "esp"
KEY = "12345678"

if not wifi.isconnected():
    print("connecting to network...")
    wifi.active(True)
    wifi.connect(SSID, KEY)

    while not wifi.isconnected():
        machine.idle()

print("network config:", wifi.ipconfig("addr4"))

mqtt_server = "broker.emqx.io"
mqtt_user = ""
mqtt_pass = ""
client_id = "esp01"

topic_sub = "esp_home/val"
topic_pub = "esp_home/msg"


def sub_cb(topic, msg):
    topic = topic.decode("utf-8")
    msg = msg.decode("utf-8")
    print((topic, int(msg)))


def restart_and_reconnect():
    print("Ошибка подключениек к MQTT брокеру.
Перезагрузка...")
    time.sleep(10)
    machine.reset()


try:
    client = MQTTClient(client_id, mqtt_server,
user=mqtt_user, password=mqtt_pass)
    client.set_callback(sub_cb)
    client.connect()
    client.subscribe(topic_sub)
except OSError as e:
    restart_and_reconnect()
```

Чтобы получать сообщения по подписке понадобится следующий код:

```
while True:
    try:
        client.check_msg()
        # client.publish(topic_pub, msg)
    except OSError as e:
        restart_and_reconnect()
```

Периодическая отправка сообщения без блокирования цикла:

```
import time

last_message = 0
```

```
message_interval = 5

while True:
    try:
        if (time.time() - last_message) >
message_interval:
            client.publish("тема", "сообщение")
            last_message = time.time()
    except OSError as e:
        restart_and_reconnect()
```

6 Разработка клиентских IoT-приложений

6.1 Дэшборд для управления устройствами

https://ioty-mqtt.a9i.sg

wss://broker.emqx.io:8084/mqtt

6.2 Приложение с GUI

6.3 Мобильное приложение

6.4 Использование алгоритмов машинного обучения