Руководство пользователя к проекту Общение нескольких Arduino по WiFi при помощи ESP8266

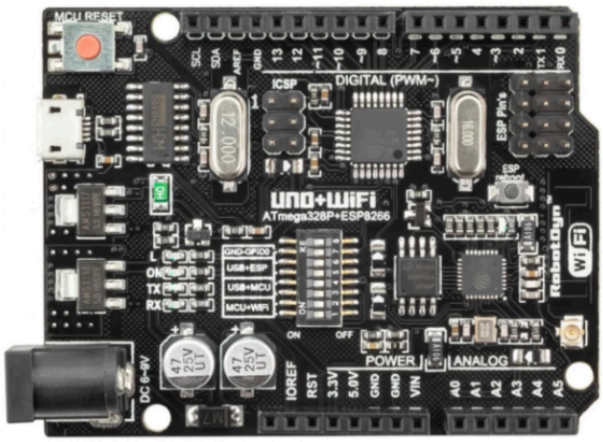
Введение

Это руководство предназначено для помощи пользователям в настройке одностороннего общения между несколькими платами Arduino через Wi-Fi при помощи ESP8266. Мы рассмотрим шаги настройки оборудования, программирования микроконтроллеров и создания сети для передачи данных.

Необходимое оборудование

Для реализации проекта вам понадобятся:

- Две платы Arduino MCU – ATmega 328P со встроенным WiFi – ESP8266.
- Кабели MicroUSB для подключения Arduino к компьютеру.
- Устройство для создания точки доступа (телефон не подходит!), например, ноутбук или роутер.



Примерный вид платы

Необходимые программы

Для реализации проекта потребуется установить:

- Arduino IDE – последняя версия.
- Драйвера для Arduino (при установке IDE предлагается их установить).

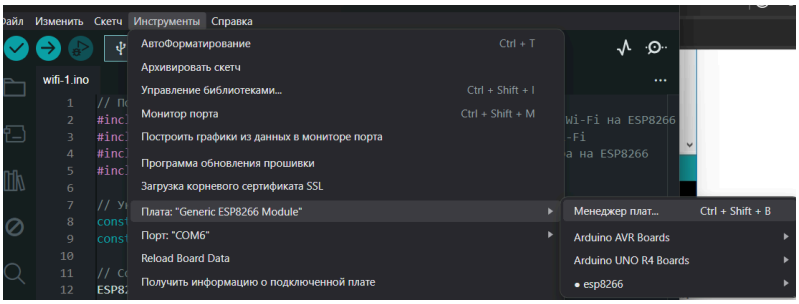
Подключение плат

Платы Arduino подключаются при помощи ранее упомянутых кабелей к компьютеру для первоначальной настройки, в последствие они могут включаться не через MicroUSB, а через кабели питания платы. Дополнительное подключение сенсоров или других устройств – опционально и в данном руководстве не рассматривается.

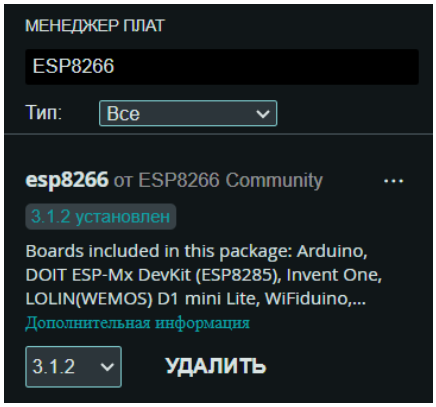
Установка дополнительного ПО

Для начала необходимо добавить плагин esp8266 в Arduino IDE. Для этого:

1. Нажимаем Инструменты → Плата → Менеджер плат.

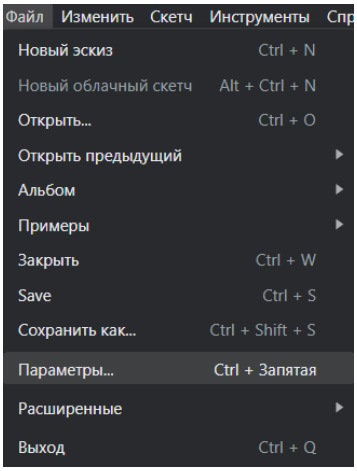


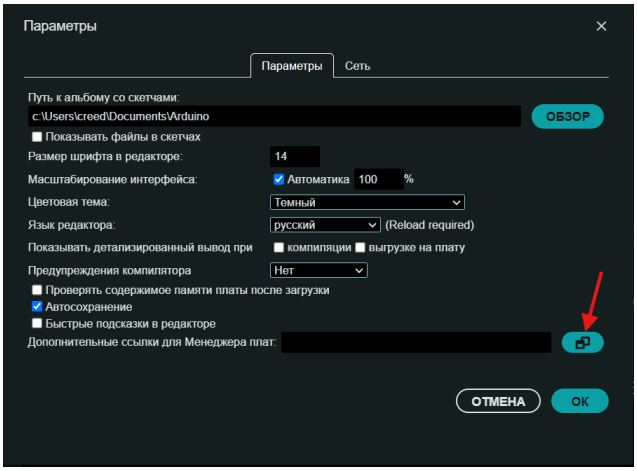
2. В менеджере плат вводим ESP8266 и скачиваем представленный плагин.



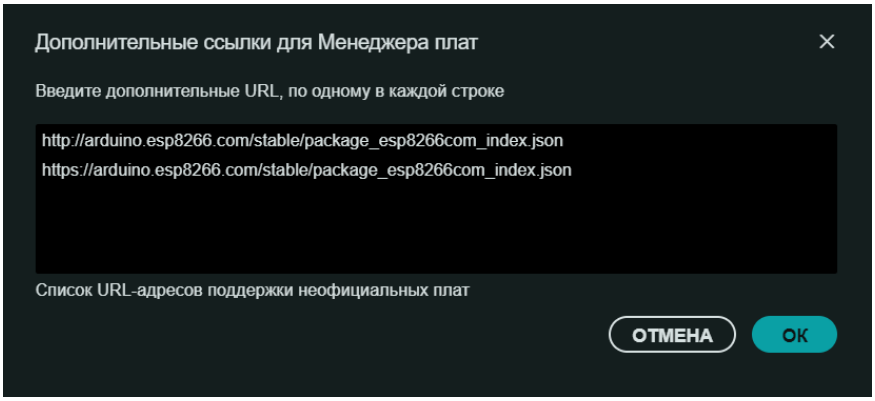
Добавим ссылки для Менеджера плат, который автоматически загрузит необходимые файлы для работы с ESP8266:

1. Нажимаем Файл → Параметры → Дополнительные ссылки для Менеджера плат





2. Вставляем следующие URL http://arduino.esp8266.com/stable/package_esp8266com_index.json, https://arduino.esp8266.com/stable/package_esp8266com_index.json и нажимаем OK в представленном окне, а также в Параметрах:



3. После этого начинается загрузка всех необходимых дополнений.

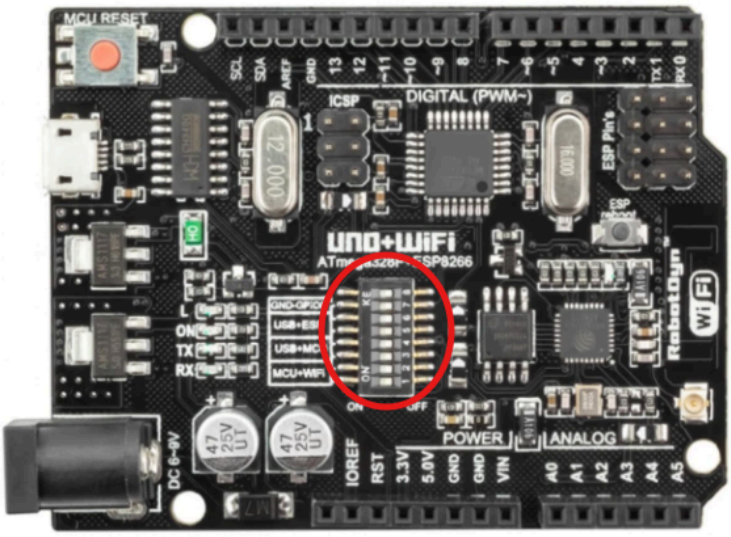
Создание точки доступа

В качестве точке доступа можно использовать ноутбук или роутер. В данном руководстве рассматривается создание точки доступа на ноутбуке.

1. Открываем через Пуск Параметры → Сеть и Интернет → Мобильный хот-спот.
2. Задаем название сети и её пароль, в будущем они понадобятся нам в скетчах.
3. При наличии различных диапазонов для точки доступа выбираем 2.4 ГГц.

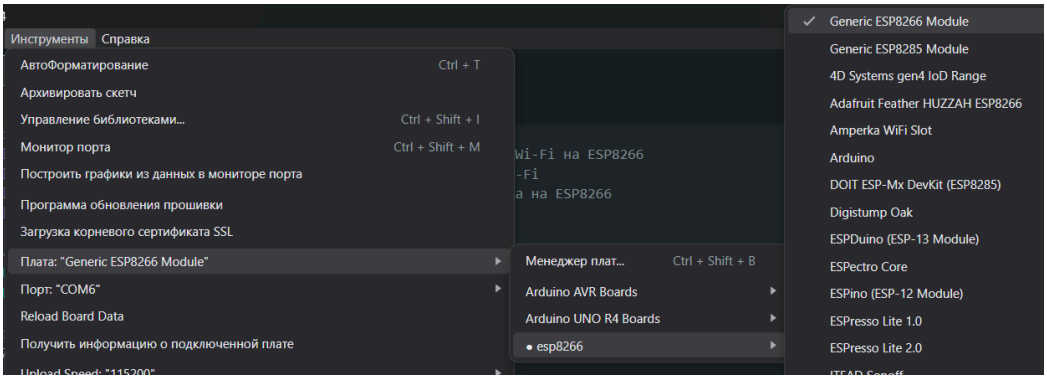
Загрузка скетча сервера

1. На плате устанавливаем переключатели с номерами 5, 6, 7 в положении ON.



2. Открываем скетч сервера.
3. Подключаем первую плату к компьютеру.

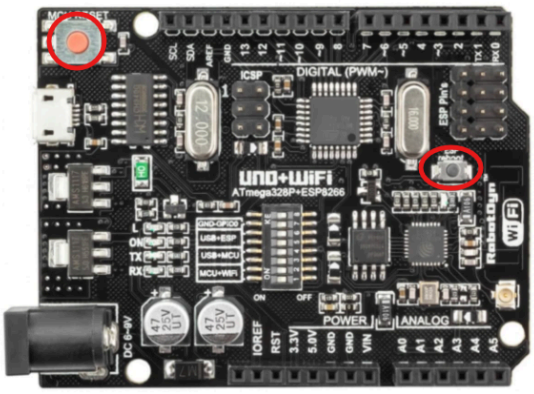
4. Переходим в Инструменты → Плата → esp8266 и выбираем Generic ESP8266 Module.

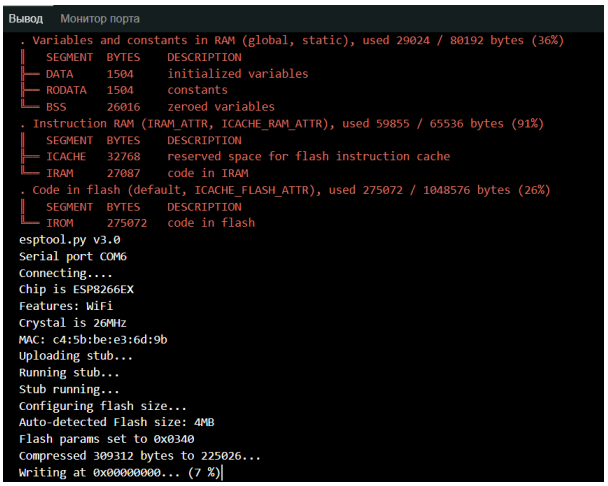


5. В скетче для сервера (Приложение 1) указываем Имя сети и Пароль сети (строки 8 и 9) от точки доступа.

```
7  // Указание SSID и пароля Wi-Fi сети
8  const char* ssid = "TestNetwork";     // Имя вашей сети Wi-Fi
9  const char* password = "1234567890";  // Пароль от вашей сети Wi-Fi
```

6. Нажимаем кнопку Загрузить в плату.
7. Как только компиляция будет подходить к концу сначала нажимаем красную кнопку MCU RESET, а сразу после задерживаем кнопку ESP REBOOT до появления сообщения Writing at 0x00000000. После этого кнопку можно отпускать.



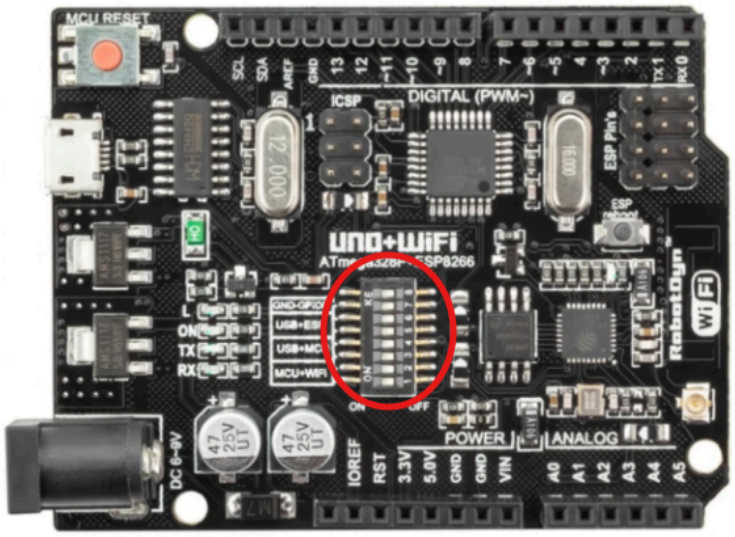


8. По окончании загрузки переключаем выключатель с номером 7 в положении OFF и нажимаем MCU RESET.
9. В мониторе порта будет выведен IP, который будет использоваться на клиенте как IP хоста.

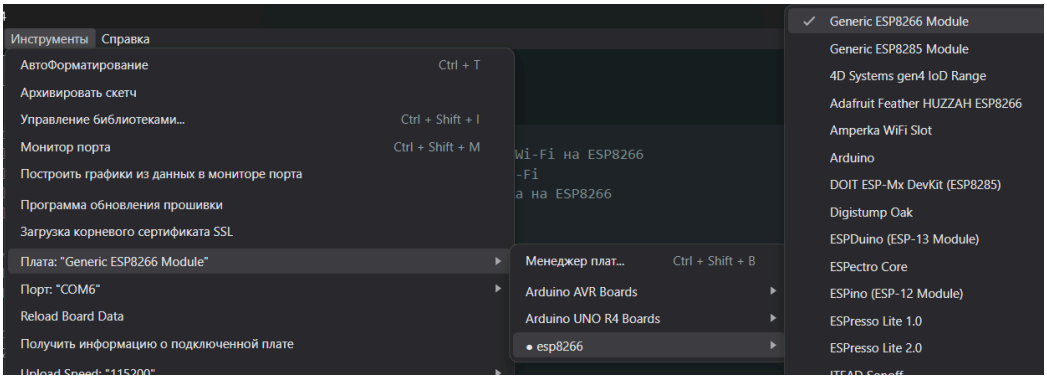
```
.........
Connected to TestNetwork
IP address: 192.168.137.132
HTTP server started
```

Загрузка скетча клиента

1. На плате устанавливаем переключатели с номерами 5, 6, 7 в положении ON.



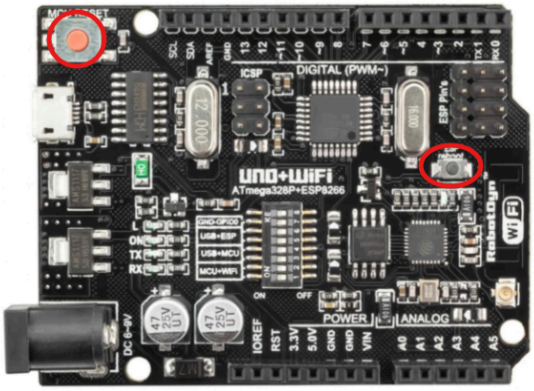
2. Открываем скетч сервера.
3. Подключаем первую плату к компьютеру.
4. Переходим в Инструменты → Плата → esp8266 и выбираем Generic ESP8266 Module.

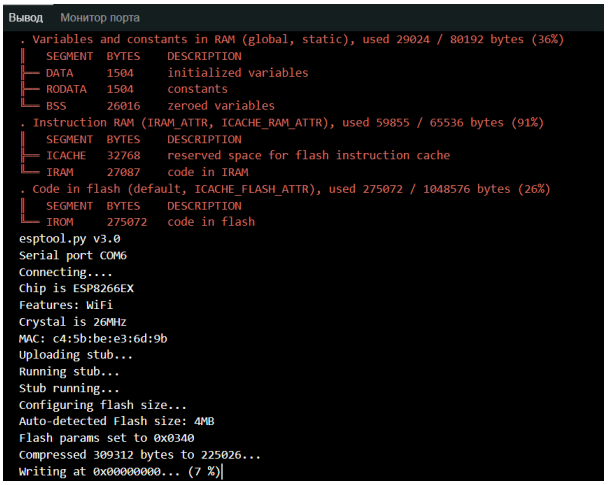


5. В скетче для сервера (Приложение 2) указываем Имя сети, Пароль сети и адрес хоста (строки 6, 7, 8) от точки доступа и платы сервера.

```
5  // Указание SSID и пароля Wi-Fi сети, а также адреса хоста
6  const char* ssid = "TestNetwork";     // Имя Wi-Fi сети
7  const char* password = "1234567890";  // Пароль от Wi-Fi сети
8  const char* host = "192.168.137.132";   // Адрес хоста, к которому будет отправляться запрос
```

6. Нажимаем кнопку Загрузить в плату.
7. Как только компиляция будет подходить к концу сначала нажимаем красную кнопку MCU RESET, а сразу после задерживаем кнопку ESP REBOOT до появления сообщения Writing at 0x00000000. После этого кнопку можно отпускать.





8. По окончании загрузки переключаем выключатель с номером 7 в положении OFF и нажимаем MCU RESET.
9. В мониторе порта будут выведены данные, полученные от сервера.

Приложение 1
Скетч для платы-сервера

```
// Подключение необходимых библиотек
#include <ESP8266WiFi.h>        // Библиотека для управления модулем Wi-Fi на ESP8266

#include <WiFiClient.h>         // Библиотека для создания клиента Wi-Fi

#include <ESP8266WebServer.h>   // Библиотека для создания веб-сервера на ESP8266

#include <math.h>               // Математическая библиотека

// Указание SSID и пароля Wi-Fi сети
const char * ssid = "TestNetwork"; // Имя вашей сети Wi-Fi
const char * password = "1234567890"; // Пароль от вашей сети Wi-Fi
// Создание экземпляра класса ESP8266WebServer для работы с портом 80
ESP8266WebServer server(80);
// Обработчик для корневого URL-адреса ("/")
void handleRoot() {
    // Отправка ответа клиенту с кодом состояния 200 и содержимым
    server.send(200, "text/plain", String(sin(millis())));
}
void setup() {
    // Настройка последовательного порта для вывода информации на скорости 115200 бод
    Serial.begin(115200);
    Serial.println();
    // Установка режима модуля Wi-Fi в режим станции
    WiFi.mode(WIFI_STA);
    // Запуск процесса подключения к указанной сети Wi-Fi
    WiFi.begin(ssid, password);
    // Ожидание успешного подключения к сети
    while (WiFi.status() != WL_CONNECTED) {
        delay(500); // Задержка перед следующей проверкой статуса
        Serial.print("."); // Вывод точки в консоли для индикации ожидания
    }
    // Вывод сообщения о подключении к сети
    Serial.println("");
    Serial.print("Connected to ");
    Serial.println(ssid); // Вывод имени подключенной сети
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP()); // Вывод IP-адреса устройства
    // Регистрация обработчика для корневого URL
    server.on("/", handleRoot);
    // Запуск сервера
    server.begin();
    Serial.println("HTTP server started"); // Сообщение о старте сервера
}
void loop() {
    // Обработка входящих запросов к серверу
    server.handleClient();
}
```

Приложение 2
Скетч для платы-клиента

```
// Подключение необходимых библиотек
#include <ESP8266WiFi.h>  // Подключает библиотеку для работы с Wi-Fi на ESP8266
#include <WiFiClient.h>   // Подключает библиотеку для создания клиента Wi-Fi

// Указание SSID и пароля Wi-Fi сети, а также адреса хоста
const char* ssid = "TestNetwork";     // Имя Wi-Fi сети
const char* password = "1234567890";  // Пароль от Wi-Fi сети
const char* host = "192.168.137.132";   // Адрес хоста, к которому будет отправляться запрос

void setup() {
  // Настройка последовательного порта для вывода информации на скорости 115200 бод
  Serial.begin(115200);
  Serial.println();

  // Установка режима модуля Wi-Fi в режим станции
  WiFi.mode(WIFI_STA);
  // Запуск процесса подключения к указанной сети Wi-Fi
  WiFi.begin(ssid, password);

  // Ожидание успешного подключения к сети
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);         // Делаем паузу в полсекунды между попытками проверки соединения
    Serial.print(".");  // Вывод точки в консоли для индикации ожидания
  }

  // Вывод сообщения о подключении к сети
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);  // Показываем имя подключённой сети
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  // Выводим IP адрес устройства
}

void loop() {
  WiFiClient client;  // Создаем объект клиента Wi-Fi

  if (!client.connect(host, 80)) {        // Пробуем соединиться с хостом на порт 80
    Serial.println("Connection failed");  // Если соединение не удалось, выводим сообщение об
ошибке
    return;                               // Прерываем выполнение функции
  }

  // Формируем GET-запрос к указанному хосту
  client.print(String("GET /") + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection:
close\r\n\r\n");

  unsigned long timeout = millis();  // Сохраняем текущее время для тайм-аута

  while (client.available() == 0) {           // Ждем поступления данных от сервера
    if (millis() - timeout > 5000) {          // Проверяем, истекло ли 5 секунд с момента начала
ожидания
      Serial.println(">>> Client Timeout !");  // Если да, выводим сообщение о таймауте
      client.stop();                          // Закрываем соединение
      return;                                 // Прерываем выполнение функции
    }
  }

  String response = "";  // Переменная для хранения полного ответа от сервера

  while (client.available()) {        // Пока есть данные для чтения
    response += client.readString();  // Читаем и добавляем их в переменную response
  }

  response.replace("\r\n", "\n");  // Убираем лишние символы перевода каретки (\r), оставляя
только переводы строки (\n)
```

```
  // В качестве разделения строк использовались именно \r\n, поэтому остановимся только на одном
\n

  int last_index = 0;  // Индекс последнего найденного символа \n новой строки
  int cur_index = 0;   // Индекс текущего символа \n новой строки
  String body = "";    // Переменная для хранения тела ответа

  for (int i = 0; i < 6; i++) {                      // Проходимся по ответу на запрос, в бой
строке находится ответ от сервера
    cur_index = response.indexOf('\n', last_index);  // Находим следующий символ \n начиная с
last_index

    if (i == 5) {                                        // Если дошли до 6 строки, то она
содержит тело ответа
      body = response.substring(last_index, cur_index);  // Извлекаем тело ответа
    }

    last_index = cur_index + 1;  // Сдвигаемся к следующему индексу
  }

  Serial.println(body);    // Выводим тело ответа

  delay(1000);  // Пауза в одну секунду перед следующим циклом
}
```