## Лабораторная работа 8: Работа с Wi-Fi модулем ESP32

**Цель:**

* Изучить основы работы с Wi-Fi модулем ESP32.
* Научиться подключаться к Wi-Fi сетям и настраивать режимы работы.
* Написать программу для сканирования доступных Wi-Fi сетей.
* Реализовать функционал удаленного управления устройством через Wi-Fi (включение/выключение светодиода через веб-интерфейс).

## Теоретическая часть

### Основы работы с Wi-Fi

ESP32 имеет встроенный Wi-Fi модуль, который позволяет подключаться к беспроводным сетям и обмениваться данными с другими устройствами. Модуль поддерживает различные стандарты Wi-Fi (802.11 b/g/n) и режимы работы:

* **Стандартный клиент:** ESP32 подключается к существующей Wi-Fi сети и может получать доступ к интернету или другим устройствам в сети.
* **Точка доступа:** ESP32 создает собственную Wi-Fi сеть, к которой могут подключиться другие устройства.

### Программирование Wi-Fi модуля

Для работы с Wi-Fi в ESP32 используется библиотека WiFi.h. Библиотека предоставляет функции для:

* Подключения к Wi-Fi сетям
* Сканирования доступных Wi-Fi сетей
* Получения информации о подключенной сети
* Создания точки доступа
* Настройки параметров Wi-Fi

### Подключение к Wi-Fi сети

**Схема подключения:**

* ESP32 подключен к Wi-Fi сети.

**Код:**

const char\* ssid = "YOUR\_SSID"; // Имя вашей Wi-Fi сети  
const char\* password = "YOUR\_PASSWORD"; // Пароль вашей Wi-Fi сети  
  
void setup() {  
 Serial.begin(115200);  
 WiFi.begin(ssid, password);  
  
 while (WiFi.status() != WL\_CONNECTED) {  
 delay(500);  
 Serial.print(".");  
 }  
  
 Serial.println("");  
 Serial.println("WiFi connected!");  
 Serial.println("IP address: ");  
 Serial.println(WiFi.localIP());  
}  
  
void loop() {  
 // ... ваш код ...  
}

**Описание:**

* В коде определены константы ssid и password, которые содержат имя и пароль вашей Wi-Fi сети.
* В функции setup() происходит инициализация Wi-Fi и попытка подключения к указанной сети.
* Цикл while проверяет статус подключения, пока не будет установлено соединение.
* После успешного подключения выводится информация о IP-адресе ESP32.

### Сканирование доступных Wi-Fi сетей

**Код:**

void setup() {  
 Serial.begin(115200);  
  
 WiFi.scanNetworks();  
  
 if (WiFi.scanResults() != 0) {  
 Serial.println("Scan results:");  
 for (int i = 0; i < WiFi.scanResults(); i++) {  
 Serial.print(WiFi.scanResults()[i].SSID);  
 Serial.print(" (");  
 Serial.print(WiFi.scanResults()[i].level);  
 Serial.print("%)\n");  
 }  
 } else {  
 Serial.println("No WiFi networks found.");  
 }  
  
 delay(10000);  
}  
  
void loop() {  
 // ... ваш код ...  
}

**Описание:**

* В функции setup() происходит сканирование доступных Wi-Fi сетей.
* Если сети найдены, их названия и уровень сигнала выводятся на Serial.
* Если сети не найдены, выводится соответствующее сообщение.

### Удаленное управление светодиодом через Wi-Fi

**Схема подключения:**

* Светодиод подключен к GPIO пину ESP32 (например, GPIO13).

**Код:**

const char\* ssid = "YOUR\_SSID";  
const char\* password = "YOUR\_PASSWORD";  
  
const int ledPin = 13; // GPIO пин, к которому подключен светодиод  
  
void setup() {  
 Serial.begin(115200);  
 WiFi.begin(ssid, password);  
  
 WiFi.server.onRoot([]() { // Обработчик GET-запросов к корневому URL  
 String message = "<!DOCTYPE html><html><head><title>Управление светодиодом</title></head><body>";  
 message += "<h1>Управление светодиодом</h1>";  
 // Кнопка включения светодиода  
 if (WiFi.server.arg("led") == "on") {  
 digitalWrite(ledPin, HIGH);  
 message += "<p>Светодиод включен</p>";  
 } else if (WiFi.server.arg("led") == "off") {  
 digitalWrite(ledPin, LOW);  
 message += "<p>Светодиод выключен</p>";  
 } else {  
 // Кнопки включения и выключения по умолчанию  
 message += "<a href=\"/?led=on\">Включить</a>";  
 message += "&nbsp;&nbsp;";  
 message += "<a href=\"/?led=off\">Выключить</a>";  
 }  
 message += "</body></html>";  
 WiFi.server.send(200, "text/html", message);  
});  
  
 WiFi.begin();  
  
 while (WiFi.status() != WL\_CONNECTED) {  
 delay(500);  
 Serial.print(".");  
 }  
  
 Serial.println("");  
 Serial.print("WiFi connected! IP address: ");  
 Serial.println(WiFi.localIP());  
  
 WiFi.server.begin();  
 Serial.print("Server started on http://");  
 Serial.println(WiFi.localIP());  
}  
  
void loop() {  
 WiFi.server.handleClient();  
}

**Описание:**

* В коде определены константы ssid и password, которые содержат имя и пароль вашей Wi-Fi сети.
* ledPin указывает на GPIO пин, к которому подключен светодиод.
* В функции setup() происходит инициализация Wi-Fi и попытка подключения к указанной сети.
* Функция WiFi.server.onRoot() назначает обработчик для GET-запросов к корневому URL (“/”).
* Обработчик формирует HTML-страницу с кнопками включения и выключения светодиода.
* При нажатии на кнопку (“/?led=on” или “/?led=off”) происходит соответствующее управление светодиодом и обновляется информация на странице.
* В цикле loop() обрабатываются запросы от клиентов к веб-серверу.

**Запуск программы:**

1. Загрузите код на плату ESP32.
2. Подключитесь к Wi-Fi сети, которую создал ESP32 (обычно имя сети по умолчанию содержит “ESP32”).
3. Откройте браузер на вашем устройстве и перейдите по адресу, который выводит программа в Serial Monitor (обычно вида http://192.168.4.1).
4. Используйте кнопки на веб-странице для управления светодиодом.

**Вариации:**

* Можно добавить дополнительные управляющие элементы на веб-страницу (например, ползунок для регулировки яркости светодиода).
* Можно реализовать управление несколькими светодиодами или другими устройствами, подключенными к ESP32.
* Можно защитить доступ к веб-интерфейсу паролем.

## Задачи:

**Задания:**

**1. Сканирование Wi-Fi сетей с сортировкой по уровню сигнала.**

* Добавьте в код функцию сортировки результатов сканирования Wi-Fi сетей по уровню сигнала.
* Выведите отсортированный список сетей на Serial Monitor или веб-интерфейс.

**2. Управление RGB-светодиодом через Wi-Fi.**

* Подключите к ESP32 RGB-светодиод (например, WS2812B).
* Реализуйте веб-интерфейс для управления цветом и яркостью RGB-светодиода.
* Используйте библиотеку для работы с RGB-светодиодами (например, Adafruit\_NeoPixel).

**3. Создание веб-сервера для отображения информации о датчиках.**

* Подключите к ESP32 датчики (например, температуры, влажности, освещенности).
* Считайте данные с датчиков.
* Создайте веб-сервер, который будет отображать текущие значения датчиков на веб-странице.

## Примеры кода для лабораторной работы:

### 1. Сканирование Wi-Fi сетей с сортировкой по уровню сигнала

**Kод:**

void setup() {  
 // ... (остальной код setup() остается неизменным)  
  
 WiFi.scanNetworks();  
  
 if (WiFi.scanResults() != 0) {  
 Serial.println("Scan results:");  
  
 // Сортировка результатов по уровню сигнала  
 std::sort(WiFi.scanResults(), WiFi.scanResults() + WiFi.scanResults(), [](const WiFiScanResult& a, const WiFiScanResult& b) {  
 return a.level > b.level;  
 });  
  
 for (int i = 0; i < WiFi.scanResults(); i++) {  
 Serial.print(WiFi.scanResults()[i].SSID);  
 Serial.print(" (");  
 Serial.print(WiFi.scanResults()[i].level);  
 Serial.print("%)\n");  
 }  
 } else {  
 Serial.println("No WiFi networks found.");  
 }  
  
 // ... (остальной код setup() остается неизменным)  
}  
  
// ... (остальной код программы)

**Описание:**

* В функцию setup() добавлена сортировка результатов сканирования.
* Для сортировки используется алгоритм std::sort из библиотеки STL.
* Сортировка осуществляется по убыванию уровня сигнала (от наибольшего к наименьшему).
* Отсортированный список сетей выводится на Serial Monitor.

### 2. Управление RGB-светодиодом через Wi-Fi

**Код:**

#include <Adafruit\_NeoPixel.h>  
  
const int ledPin = 13; // GPIO пин для подключения светодиода  
const int ledCount = 1; // Количество светодиодов (в данном случае 1)  
  
Adafruit\_NeoPixel pixels(ledCount, ledPin, NEO\_GRB + NEO\_KHZ800);  
  
void setup() {  
 // ... (остальной код setup() остается неизменным)  
  
 pixels.begin(); // Инициализация библиотеки Adafruit\_NeoPixel  
 pixels.show(); // Отображение текущего состояния светодиодов (выключены)  
}  
  
void loop() {  
 // ... (остальной код loop() остается неизменным)  
  
 // Обработка GET-запросов к веб-интерфейсу  
 if (WiFi.server.args["red"] && WiFi.server.args["green"] && WiFi.server.args["blue"]) {  
 int red = atoi(WiFi.server.args["red"].c\_str());  
 int green = atoi(WiFi.server.args["green"].c\_str());  
 int blue = atoi(WiFi.server.args["blue"].c\_str());  
  
 pixels.setPixelColor(0, pixels.Color(red, green, blue)); // Установка цвета для первого светодиода  
 pixels.show(); // Отображение нового цвета  
 }  
}  
  
// ... (остальной код программы)

**HTML-код веб-интерфейса:**

<!DOCTYPE html><html><head><title>Управление RGB-светодиодом</title></head><body>  
<h1>Управление RGB-светодиодом</h1>  
<form method="get">  
 <label for="red">Красный:</label>  
 <input type="number" id="red" name="red" min="0" max="255" value="0">  
 <br>  
 <label for="green">Зеленый:</label>  
 <input type="number" id="green" name="green" min="0" max="255" value="0">  
 <br>  
 <label for="blue">Синий:</label>  
 <input type="number" id="blue" name="blue" min="0" max="255" value="0">  
 <br>  
 <br>  
 <button type="submit">Установить цвет</button>  
</form>  
</body></html>

**Описание:**

* **Веб-интерфейс:**
  + На веб-странице размещена форма с тремя полями ввода для значений красного, зеленого и синего цветов.
  + Пользователь может ввести значения цветов в диапазоне от 0 до 255.
  + При нажатии кнопки “Установить цвет” значения из полей ввода отправляются на сервер ESP32 в виде GET-запроса.
* **Обработка запроса:**
  + Сервер ESP32 получает GET-запрос и извлекает значения цветов из параметров запроса.
  + Значения цветов преобразуются из строк в целые числа.
  + Функция pixels.setPixelColor библиотеки Adafruit\_NeoPixel используется для установки цвета первого светодиода.
  + Функция pixels.show обновляет отображение цвета на светодиоде.

### 3. Создание веб-сервера для отображения информации о датчиках

**Код:**

#include <Wire.h>  
#include <Adafruit\_BME280\_Library.h>  
  
// Адрес датчика BME280  
#define BME280\_ADDRESS 0x77  
  
// Создаем экземпляр класса Adafruit\_BME280  
Adafruit\_BME280 bme280(BME280\_ADDRESS);  
  
void setup() {  
 // ... (остальной код setup() остается неизменным)  
  
 // Инициализация датчика BME280  
 if (!bme280.begin()) {  
 Serial.println("Не удалось подключиться к датчику BME280!");  
 while (1);  
 }  
  
 // Настройка режима работы датчика  
 bme280.setTemperatureOversampling(BME280\_TEMPERATURE\_OVERSAMPLING\_8);  
 bme280.setPressureOversampling(BME280\_PRESSURE\_OVERSAMPLING\_8);  
 bme280.setHumidityOversampling(BME280\_HUMIDITY\_OVERSAMPLING\_8);  
 bme280.setFilterCoefficient(BME280\_FILTER\_COEFFICIENT\_16);  
 bme280.setMode(BME280\_MODE\_NORMAL);  
}  
  
void loop() {  
 // ... (остальной код loop() остается неизменным)  
  
 // Считывание данных с датчика  
 float temperature = bme280.readTemperature();  
 float pressure = bme280.readPressure();  
 float humidity = bme280.readHumidity();  
  
 // Формирование HTML-строки с данными  
 String html = "<!DOCTYPE html><html><head><title>Данные с датчиков</title></head><body>";  
 html += "<h1>Данные с датчиков BME280</h1>";  
 html += "<p>Температура: " + String(temperature) + " °C</p>";  
 html += "<p>Давление: " + String(pressure) + " hPa</p>";  
 html += "<p>Влажность: " + String(humidity) + " %</p>";  
 html += "</body></html>";  
  
 // Отправка HTML-страницы клиенту  
 WiFi.server.send(200, "text/html", html);  
}  
  
// ... (остальной код программы)

**Описание:**

* В код добавлены библиотека Adafruit\_BME280 и константы для работы с датчиком BME280.
* В функции setup() происходит инициализация датчика и настройка режима его работы.
* В цикле loop() считываются данные с датчика: температура, давление и влажность.
* Формируется HTML-строка с данными датчиков.
* HTML-строка отправляется клиенту в виде ответа на HTTP-запрос.