ISPC - TSC - Modulo de Electronica Microcontrolada

Vittorio Durigutti
GIthub: https://github.com/vittoriodurigutti

WiFi

[Fecha: 17/09/2024]

Resumen

Busco investigar y dejar registro del uso del módulo de conexión wifi incorporado por defecto en el microcontrolador ESP32, así como sus funciones y potencial para incorporar lo aprendido en los proyectos en curso dentro de la carrera y para utilización de los mismos en futuros proyectos personales.

Metas

1. Entender el funcionamiento del módulo WiFi, así como de las librerías involucradas para su utilización.
2. Realizar bloques de código genérico pero funcional pudiendo este ser utilizado en los proyectos en paralelo.
3. Cargar lo realizado en el microcontrolador y conseguir realizar conexión, utilización de este como access point, recibir respuesta en el monitor serial del estado, y configurar acciones automáticas ante situaciones de uso común.

Detalles del módulo WiFI

El módulo wifi integrado en la ESP32 está basado en la tecnología IEEE 802.11 b/g/n. Opera únicamente con banda de 2,4 GHz pero con los 3 estándares de wifi mencionados. Resumidamente [802.11b] con velocidades de hasta 11 mbps con modulación DSSS, [802.11g] con velocidades de hasta 54 mbps modulación OFDM, y [802.11n] con velocidades de hasta 150 mbps, ahora siendo MIMO, también con modulación OFDM pero con optimizaciones para el ancho de banda, múltiples antenas, entre otras,

Detalles de la librería WiFi.h

Esta librería funciona con toda la familia ESP32 y ESP8266. Permite al microcontrolador operar en tres modos. El [STA] (station mode) que permite conectarlo con redes wifi. [AP] (access point) que permite al microcontrolador emitir/crear una red de wifi. Y un mixto [AP+STA] que permite ambas funciones en simultáneo. Y referido a seguridad soporta **WEP**, **WPA** y **WPA2**,

Tiene una serie de funciones que apuntan a gestionar las conexiones.

> [WiFi.begin(ssid, password)] método que inicia el proceso de conexión, utilizando usuario y contraseña de la red.
>
> [WiFi.status()] es el método que devuelve el estado de la conexión. Las opciones son: [WL_CONNECTED] para encendido, [WL_DISCONNECTED] para el desconectado, [WL_NO_SSID_AVAIL] cuando no se encuentra la SSID, [WL_CONNECT_FAILED] cuando falla la conexión.
>
> WiFi.disconnect() similar a lo anterior, sirve para descontar de la red wifi actual.

Funciones para obtener información de la red:

> [WiFi.localIP()]: Devuelve la dirección IP local
>
> [WiFi.macAddress()]: Devuelve la dirección MAC
>
> [WiFi.RSSI()]: Devuelve la intensidad de la señal Wi-Fi en dBm (decibelios en miliwatts). Valores más cercanos a 0 indican una señal más fuerte (ej. -30 dBm es mejor que -70 dBm).

Punto de Acceso (Access Point):

> [WiFi.softAP(ssid, password)]: Este método permite que el ESP32 cree su propia red Wi-Fi. Es útil cuando quieres que otros dispositivos se conecten directamente al ESP32 sin necesidad de un router.
>
> [WiFi.softAPIP()]: Devuelve la dirección IP del ESP32 cuando actúa como un punto de acceso.

Escaneo de Redes Wi-Fi:

> [WiFi.scanNetworks()]: Escanea todas las redes Wi-Fi disponibles a su alrededor.
>
> [WiFi.SSID(i)]: Obtiene el nombre (SSID) de la red en el índice i del escaneo.

[WiFi.RSSI(i)]: Devuelve la intensidad de la señal de la red en el índice i.

Configuración de Red Manual:

[WiFi.config(local ip, gateway, subnet, dns)]: Si no deseas que el router asigne automáticamente una IP a tu ESP32 (usando DHCP), puedes configurar una IP estática manualmente con este método. Debes definir la IP local, la puerta de enlace (gateway), la máscara de subred, y opcionalmente, la dirección DNS.

Callbacks para Eventos: Puedes utilizar eventos para manejar situaciones de conexión y desconexión

[WiFi.onEvent()]: Este método te permite manejar diferentes eventos de la conexión Wi-Fi, como conexión exitosa, desconexión, y otros, mediante la asignación de funciones de callback.

Detalles de la librería WiFiClient.h

Esta es parte del framework de Wifi con el que trabaja los apartados de ESP32. Permite manejar las conecciones de red como cliente estableciendo conexiones TCP/IP (cualquiera de los protocolos basados en TCP

Código de ejemplo definiendo la dirección, puerto, etc.

```
WiFiClient client;
if (client.connect("example.com", 80)) {
  client.println("GET / HTTP/1.1");
  client.println("Host: example.com");
  client.println("Connection: close");
  client.println();
}
```

Detalles de la librería Preferences.h

Esta librería permite guardar datos de forma persistente en la memoria NO volátil (NVS) del microcontrolador. Estos no se borran al apagar o reiniciar el dispositivo. Permite guardar, recuperar y eliminar variables de configuración (por ejemplo, el SSID y la contraseña de una red Wi-Fi) sin usar la memoria flash como la librería EEPROM.

Codigo de ejemplo genérico

```
Preferences preferences;
preferences.begin("my-app", false); // Inicia el espacio de almacenamiento
preferences.putString("ssid", "MiRedWiFi");  // Guarda un SSID
preferences.putString("password", "123456"); // Guarda la contraseña
String ssid = preferences.getString("ssid", "");  // Recupera el SSID
preferences.end();  // Cierra el almacenamiento
```

Código de ejemplo, explicado por bloques

Librerias

```
#include <WiFi.h>
#include <WiFiClient.h>
 #include <Preferences.h>
```

Objeto para manejar la memoria no volatil

```
Preferences preferences;
```

Definición de la red local y el servidor remoto

```
const char* host = "192.168.1.100";
const int port = 3005;
```

Definiciones para guardar los datos, y la conexión automatica

```
WiFiClient client;        // Cliente para la conexión TCP
String ssid;              // SSID de la red Wi-Fi
String password;          // Contraseña de la red Wi-Fi
bool autoConnect = false; // Conexión automática habilitada/deshabilitada
```

Declaraciones adelantadas para conectar a Wifi, la lista de redes, conexion y desconexion con el servidor

```
void connectWiFi();
void listNetworks();
void connectToServer();
void disconnectWiFi();
```

Evento para monitorear el estado de la conexión Wi-Fi

```
void WiFiEvent(WiFiEvent_t event) {
  switch(event) {
    case SYSTEM_EVENT_STA_CONNECTED:
```

```
      Serial.println("Conectado a la red Wi-Fi.");
      break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
      Serial.println("Desconectado de la red Wi-Fi.");
      if (autoConnect) {
        Serial.println("Intentando reconectar en 20 segundos...");
        delay(20000);
        connectWiFi();  // Intentar reconectar
      }
      break;
    case SYSTEM_EVENT_STA_GOT_IP:
      Serial.print("Conexión exitosa. Dirección IP: ");
      Serial.println(WiFi.localIP());
      Serial.println("Usted se encuentra conectado a internet.");
      connectToServer();  // Intentar conectar al servidor remoto
      break;
  }
}
```

```
void setup() {
  Serial.begin(115200);
  WiFi.onEvent(WiFiEvent);  // Configurar los eventos Wi-Fi
```

Cargar credenciales guardadas

```
 preferences.begin("wifi-creds", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  autoConnect = preferences.getBool("autoConnect", false);

  if (ssid != "") {
    Serial.println("Red Wi-Fi guardada encontrada. Intentando
conectar...");
    connectWiFi();
  } else {
    Serial.println("No hay redes guardadas. Listando redes cercanas...");
    listNetworks();
  }
```

Mostrar menú inicial

```
  Serial.println("\nMenú de opciones:");
  Serial.println("1: Conectar a una red Wi-Fi");
```

```
  Serial.println("2: Borrar/redactar credenciales Wi-Fi");
  Serial.println("3: Configurar parámetros de red (WiFi.config)");
  Serial.println("4: Desconectar del Wi-Fi");
}

void loop() {
  if (Serial.available() > 0) {
    String option = Serial.readStringUntil('\n');
    option.trim();

    if (option == "1") {
      listNetworks();
    } else if (option == "2") {
      preferences.remove("ssid");
      preferences.remove("password");
      Serial.println("Credenciales Wi-Fi borradas.");
    } else if (option == "3") {
```

Configuración de IP estática

```
      IPAddress local_IP(192, 168, 1, 184);
      IPAddress gateway(192, 168, 1, 1);
      IPAddress subnet(255, 255, 255, 0);
      WiFi.config(local_IP, gateway, subnet);
      Serial.println("Configuración de red personalizada establecida.");
    } else if (option == "4") {
      disconnectWiFi();
    } else {
      Serial.println("Opción no válida.");
    }
```

Mostrar menú después de cada operación

```
    Serial.println("\nMenú de opciones:");
    Serial.println("1: Conectar a una red Wi-Fi");
    Serial.println("2: Borrar/redactar credenciales Wi-Fi");
    Serial.println("3: Configurar parámetros de red (WiFi.config)");
    Serial.println("4: Desconectar del Wi-Fi");
  }
}
```

Función para conectar al Wi-Fi usando las credenciales guardadas o ingresadas

```
void connectWiFi() {
  if (ssid != "" && password != "") {
    WiFi.begin(ssid.c_str(), password.c_str());
    Serial.print("Conectando a la red: ");
    Serial.println(ssid);
  } else {
    Serial.println("No se encontraron credenciales guardadas.");
  }
}
```

Función para listar redes Wi-Fi cercanas

```
void listNetworks() {
  int numNetworks = WiFi.scanNetworks();
  if (numNetworks == 0) {
    Serial.println("No se encontraron redes.");
  } else {
    Serial.println("Redes Wi-Fi disponibles:");
    for (int i = 0; i < numNetworks; ++i) {
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.println(" dBm)");
    }
    Serial.println("Ingrese el número de la red a la que desea
conectarse:");

    while (Serial.available() == 0) {}  // Espera a la entrada del usuario
    int selectedNetwork = Serial.parseInt() - 1;
    if (selectedNetwork >= 0 && selectedNetwork < numNetworks) {
      Serial.print("Seleccionó: ");
      Serial.println(WiFi.SSID(selectedNetwork));
      Serial.print("Ingrese la contraseña: ");
      while (Serial.available() == 0) {}
      password = Serial.readStringUntil('\n');
      password.trim();

      ssid = WiFi.SSID(selectedNetwork);
      connectWiFi();
```

Preguntar si quiere guardar la red

```
Serial.println("¿Desea guardar esta red para conexiones automáticas?
```

```
(y/n)");
      while (Serial.available() == 0) {}
      String save = Serial.readStringUntil('\n');
      save.trim();
      if (save == "y" || save == "Y") {
        preferences.putString("ssid", ssid);
        preferences.putString("password", password);
        preferences.putBool("autoConnect", true);
        Serial.println("Credenciales guardadas.");
      } else {
        Serial.println("Credenciales no guardadas.");
      }
    } else {
      Serial.println("Selección inválida.");
    }
  }
}
```

Función para desconectar del Wi-Fi

```
void disconnectWiFi() {
  WiFi.disconnect();
  Serial.println("Usted se encuentra desconectado de la red.");
}
```

Función para conectarse a un servidor en el puerto 3005

```
void connectToServer() {
  if (client.connect(host, port)) {
    Serial.println("Conexión al servidor exitosa.");
    client.println("GET / HTTP/1.1");
    client.println("Host: 192.168.1.100");
    client.println("Connection: close");
    client.println();
  } else {
    Serial.println("Error al conectarse al servidor.");
  }
}
```