INFORME PRÁCTICA 3: WiFi i Bluetooth

La idea general de esta práctica se basa en entender los fundamentos básicos del uso del WIFI i el Bluetooth en periféricos, así como se conectan a los microprocesadores, que en este caso se trata del ESP32-S3. Veremos como crear un servidor web a partir del ESP32 i poder crear una web desde este mismo código para poder verla en un navegador web.
También veremos el uso del Bluetooth en nuestro procesador, así como generar una conexión entre el ESP32 i un dispositivo móvil a través de una aplicación.

Veamos los códigos:

Código Wi-Fi

```
/*
  ESP32 Web Server - STA Mode
  modified on 25 MAy 2019
  by Mohammadreza Akbari @ Electropeak
  https://electropeak.com/learn
*/

#include <WiFi.h>
#include <WebServer.h>

// SSID & Password
const char* ssid = "*****";  // Enter your SSID here
const char* password = "*****";  //Enter your Password here

WebServer server(80);  // Object of WebServer(HTTP port, 80 is defult)

void setup() {
  Serial.begin(115200);
  Serial.println("Try Connecting to ");
  Serial.println(ssid);

  // Connect to your wi-fi modem
  WiFi.begin(ssid, password);

  // Check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected successfully");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());  //Show ESP32 IP on serial

  server.on("/", handle_root);

  server.begin();
  Serial.println("HTTP server started");
  delay(100);
}
```

```
void loop() {
  server.handleClient();
}

// HTML & CSS contents which display on web server
String HTML = "<!DOCTYPE html>\
<html>\
<body>\
<h1>My Primera Pagina con ESP32 - Station Mode &#128522;</h1>\
</body>\
</html>";

// Handle root url (/)
void handle_root() {
  server.send(200, "text/html", HTML);
}
```

Declaración de librerías

```
#include <WiFi.h>
#include <WebServer.h>
```

* ***WiFi.h***: Librería que permite al ESP32 conectarse a redes WIFI
* ***WebServer.h***: Librería que proporciona las herramientas necesarias para crear un servidor web en el ESP32.

Configuración de WIFI

```
const char* ssid = "*****";
const char* password = "*****";
```

* Declaramos las variables:
    - ***ssid***: Contiene el nombre de nuestra red WIFI.
    - ***password***: Contiene su correspondiente contraseña.

Configuración del Servidor Web

```
WebServer server(80);
```

* Se crea un objeto **WebServer** en el puerto HTTP 80, que es el valor por defecto.

Configuración del *setup()*

```
void setup() {
  Serial.begin(115200);
```

* Iniciamos la comunicación serial a la velocidad de siempre, 115200 baudios.

```
  WiFi.begin(ssid, password);
```

* Iniciamos la conexión WIFI utilizando las credenciales declaradas anteriormente.

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
```

* Se genera el *while* para esperar a que el ESP32 se conecte a la red WIFI. Mientras tanto, se imprime un punto para "simular" la espera cada 1000ms (1 segundo).

```
  server.on("/", handle_root);
```

* Se establece una función de manejo para la ruta raíz del servidor. Para ello, define la funcion *handle_root()*

Configuración del *loop()*

```
void loop() {
  server.handleClient();
}
```

* Con la funcion *handleClient* se manejan las solicitudes los clientes que entren al servidor web.

Contenido de la web

```
String HTML = "<!DOCTYPE html>\
<html>\
<body>\
<h1>My Primera Pagina con ESP32 - Station Mode &#128522;</h1>\
</body>\
</html>"
```

* Se crea el contenido en lenguaje HTML que veremos en la una vez se ejecute el programa. Se guarda todo en la variable **HTML**.

Manejador de la Ruta Raíz

```
void handle_root() {
  server.send(200, "text/html", HTML);
  }
```

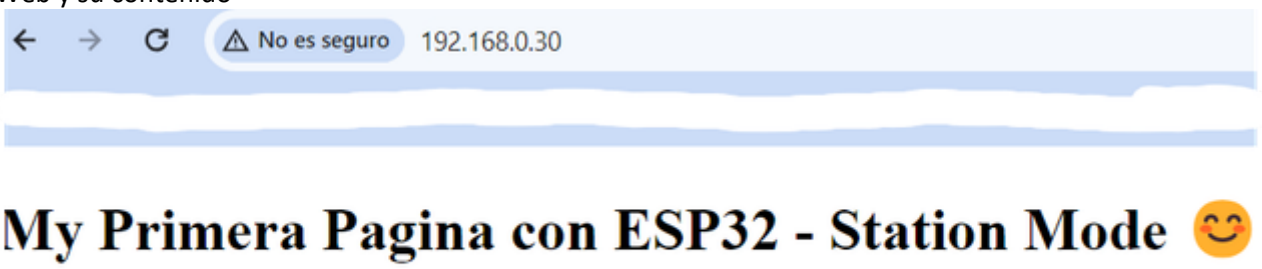
* Cuando se solicita la ruta raíz ("/"), se envía una respuesta HTTP con un código de 200 y el contenido HTML almacenado en la variable HTML.

Imagenes del resultado

* Termnial una vez ejecutado

    - Podemos ver como nos muestra la IP de nuestra web. Para poder ver el contenido, copiamos la IP y la pegamos en cualquier navegador web.

* Web y su contenido



  - Se puede observar la IP que nos muestra en la terminal pegada en un navegador así como el contenido creado HTML.

Código WI-FI modificado

```
#include <WiFi.h>
#include <WebServer.h>

// SSID & Password
const char* ssid = "vodafone52C8"; // Ingresa tu SSID aquí
const char* password = "DZ4EWZLNMN4UZN"; // Ingresa tu contraseña aquí

WebServer server(80); // Objeto de WebServer (puerto HTTP, 80 es el predeterminado)

// HTML contenido que se muestra en el servidor web
String HTML = "<!DOCTYPE html>\
<html lang='es'>\
<head>\
    <meta charset='UTF-8'>\
    <meta name='viewport' content='width=device-width, initial-scale=1.0'>\
    <title>Tipos de Guitarras</title>\
    <style>\
        body {\
            font-family: Arial, sans-serif;\
            margin: 20px;\
        }\
        h1 {\
            color: #333;\
        }\
        h2 {\
            color: #444;\
        }\
        p {\
            color: #666;\
        }\
        ul {\
            list-style-type: none;\
            padding: 0;\
        }\
        li {\
            margin-bottom: 10px;\
```

```
        }\
    </style>\
</head>\
<body>\
    <h1>LA GUITARRA</h1>\
    \
    <h2>Introducción</h2>\
    <p>La guitarra es uno de los instrumentos musicales más populares del mundo. A lo largo de los siglos, ha evolucionado en diversos tipos, cada uno con sus propias características y aplicaciones.</p>\
    \
    <h2>Historia</h2>\
    <p>La historia de la guitarra se remonta a miles de años atrás. Se cree que sus orígenes se encuentran en la antigua Persia, donde se desarrollaron instrumentos de cuerda similares. Con el tiempo, la guitarra se ha transformado y adaptado en diferentes culturas y épocas, dando lugar a una amplia variedad de estilos y diseños.</p>\
    \
    <h2>Tipos de Guitarras</h2>\
    <p>A continuación se presentan algunos de los tipos más comunes de guitarras:</p>\
    <ul>\
        <li><strong>Guitarra Acústica:</strong> Perfecta para tocar en cualquier lugar sin necesidad de amplificación.</li>\
        <li><strong>Guitarra Eléctrica:</strong> Ideal para estilos de música más eléctricos y se puede conectar a amplificadores para aumentar su sonido.</li>\
        <li><strong>Guitarra Clásica:</strong> Con cuerdas de nylon, es adecuada para estilos de música clásica y flamenco.</li>\
        <li><strong>Guitarra Semiacústica:</strong> Combina características de la guitarra eléctrica y la acústica, proporcionando un sonido más cálido.</li>\
        <li><strong>Guitarra de 12 Cuerdas:</strong> Tiene el doble de cuerdas que una guitarra estándar, lo que produce un sonido más rico y completo.</li>\
        <li><strong>Guitarra Resonadora:</strong> Utiliza un cono metálico para amplificar el sonido, siendo común en estilos de blues y country.</li>\
    </ul>\
    \
    <h2>Cómo Elegir la Guitarra Correcta</h2>\
    <p>Al elegir una guitarra, es importante considerar factores como el estilo de música que deseas tocar, tu nivel de experiencia y tu presupuesto. También es recomendable probar diferentes guitarras antes de tomar una decisión final.</p>\
    \
    <h2>Mantenimiento y Cuidado</h2>\
    <p>Para mantener tu guitarra en óptimas condiciones, es importante limpiarla regularmente, cambiar las cuerdas cuando sea necesario y almacenarla en un lugar seguro y adecuado.</p>\
    \
    <h2>Conclusión</h2>\
    <p>La guitarra es un instrumento versátil y emocionante que ha capturado la imaginación de músicos y aficionados de todo el mundo. Con una amplia variedad de tipos disponibles, siempre hay una guitarra adecuada para cada persona y estilo de música.</p>\
    \
    <a href='https://ibb.co/0J5jQsC'><img src='https://i.ibb.co/KynjmK2/Guitarras.jpg' alt='Guitarras' border='0'></a><br /><a target='_blank' href='https://es.imgbb.com/'>pics para editar</a><br />\
```

```
</body>\
</html>";


// Manejar la URL raíz (/)
void handle_root() {
    server.send(200, "text/html", HTML);
}

void setup() {
    Serial.begin(115200);
    Serial.println("Intentando conectar a ");
    Serial.println(ssid);

    // Conectar al modem de Wi-Fi
    WiFi.begin(ssid, password);

    // Verificar que se haya conectado a la red Wi-Fi
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("Conexión Wi-Fi exitosa");
    Serial.print("IP asignada: ");
    Serial.println(WiFi.localIP()); // Mostrar la IP de ESP32 en serie

    server.on("/", handle_root);
    server.begin();
    Serial.println("Servidor HTTP iniciado");
    delay(100);
}

void loop() {
    server.handleClient();
}
```

* En este código, únicamente he añadido más contenido en la parte donde se crea la web en HTML el cual se explican los tipos de guitarras y su mantenimiento y cuidado.

Código Bluetooth

```
//This example code is in the Public Domain (or CC0 licensed, at your option.)
//By Evandro Copercini - 2018
//
//This example creates a bridge between Serial and Classical Bluetooth (SPP)
//and also demonstrate that SerialBT have the same functionalities of a normal Serial

#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
```

```
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test"); //Bluetooth device name
  Serial.println("The device started, now you can pair it with bluetooth!");
}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

Declaración de librerías

```
#include <BluetoothSerial.h>
```

* ***BluetoothSerial.h***: Librería que permite al ESP32 utilizar el Bluetooth.

Verificación de la configuración del Bluetooth

```
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
```

* Verifican que el Bluetooth esté habilitado en la configuración del ESP32-S3. Si no lo está, muestra un mensaje de error.

Inicialización del objeto BluetoothSerial

```
BluetoothSerial SerialBT;
```

* Se crea un objeto llamado ***SerialBT*** el cual tiene forma de la clase ***BluetoothSerial***. Se utilizará para el manejo de la comunicación Bluetooth.

Configuración del *setup()*

```
void setup() {
  Serial.begin(115200);
```

* Iniciamos la comunicación serial a la velocidad de siempre, 115200 baudios.

```
SerialBT.begin("ESP32test");
```

* Se inicia el Bluetooth del objeto creado anteriormente con el nombre de "ESP32test".

```
Serial.println("El dispositivo ha iniciado, ¡ahora puedes emparejarlo con
Bluetooth!")
```

* Se imprime un mensaje el cual indica que ya se puede realizar el emparejamiento con Bluetooth.

Configuración del *loop*

```
if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
```

* En el primer *if* se comprueba que hay datos disponibles en el puerto serie.

```
if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
```

* Si el primer *if* se cumple, en el segundo, se leen los datos del puerto serie y se escriben en el objeto Bluetooth, lo que los envía a través de Bluetooth.

```
delay(20)
```

* Se incluye un retardo de 20 milisegundos entre iteraciones del bucle para evitar saturaciones.

Código excepcional: RGB

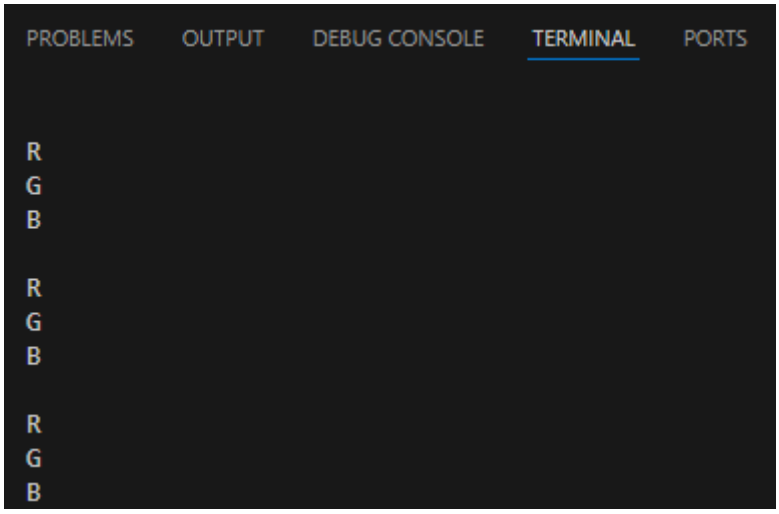
Durante la ejecución de la práctica 3, en el laboratorio hemos aprendido sobre el led integrado que tiene el ESP32-S3. Conocido como LED_BUILTIN, se trata de una luz led que tiene la opción de iluminarse en formato RGB, es decir, se puede iluminar en 3 distintos colores: rojo, verde y azul.
Buscando por diferentes foros de internet, encontramos un código el cual lo permite.

Adjunto código:

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>  //Incluimos la librerÃƒÂ-a Adafruit_NeoPixe

// pin del led RGB en PCB
#define PIN      48  // ESP32S3 Generic -n8r2-

// Un solo led en PCB
#define NUMPIXELS 1

// Inicializamos el objeto "pixels"
// Argumento 1 = nÃƒÂºmero de pixeles
// Argumento 2 = nÃƒÂºmero del pin de datos
// Argumento 3 = tipo de pixel: NEO_KHZ800  800 KHz bitstream (most NeoPixel products
w/WS2812 LEDs)
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
// Time (en milliseconds) pausa entre pixels
#define DELAYVAL 500

void setup() {
  delay(500);
  // Inicializamos el objeto pixels
  pixels.begin();
  Serial.begin(115200);
  delay(500);
  // InformaciÃƒÂ³n chip
  Serial.println("\n\n================================");
  Serial.printf("Chip Modelo: %s\n", ESP.getChipModel());
  Serial.printf("Chip Revision: %d\n", ESP.getChipRevision());
  Serial.printf("E %d core\n", ESP.getChipCores());
```

```
  Serial.printf("Flash Chip capacidad: %d \n", ESP.getFlashChipSize());
  Serial.printf("Flash Chip velocidad: %d \n", ESP.getFlashChipSpeed());

  esp_chip_info_t chip_info;
  esp_chip_info(&chip_info);
  Serial.printf("\nCaracteristicas:\n %s\n %s\n %s\n %s\n %s\n",
      (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "flash en ESP32" : "Sin flash
en ESP32",
      (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "2.4GHz WiFi" : "Sin Wifi",
      (chip_info.features & CHIP_FEATURE_BLE) ? "Bluetooth LE" : "Sin Bluetooth LE",
      (chip_info.features & CHIP_FEATURE_BT) ? "Bluetooth" : "Sin Bluetooth",
      (chip_info.features & CHIP_FEATURE_IEEE802154) ? "IEEE 802.15.4" : "NO IEEE
802.15.4");
  // MAC Address
  String MACString = "";
  uint64_t chipid = ESP.getEfuseMac();
  for (int i=0; i<6; i++) {
    if (i > 0) MACString.concat(":");
    uint8_t Octet = chipid >> (i * 8);
    MACString.concat(String(Octet, HEX));
  }
  Serial.println("MAC: " + MACString);
}

void loop() {
  // apagamos
  pixels.clear();
  pixels.show();
  Serial.println("");
  delay(1000);

  // encendemos con el color rojo
  pixels.setPixelColor(0, pixels.Color(150, 0, 0));
  // enviamos
  pixels.show();
  Serial.println("R");
  delay(DELAYVAL);

  // encendemos con el color verde, con brillo moderado
  pixels.setPixelColor(0, pixels.Color(0, 150, 0));
  pixels.show();
  Serial.println("G");
  delay(DELAYVAL);

  // encendemos con el color azul, con brillo moderado
  pixels.setPixelColor(0, pixels.Color(0, 0, 150));
  pixels.show();
  Serial.println("B");
  delay(DELAYVAL);

}
```

Imagen del terminal



El terminal nos muestra las letras 'R', 'G' y 'B', que corresponden a **RED** (rojo), **GREEN** (verde) y **BLUE** (azul). Se imprimen por pantalla a la vez que el LED se ilumina del color correspondiente.