PRÁCTICA 3A : GENERACIÓN DE UNA PAGINA WEB

APARTADO 1

CÓDIGO

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

// SSID & Password
const char* ssid = "MIWIFI_2G_2jcE";  // Enter your SSID here
const char* password = "9GLX7t3u";  //Enter your Password here

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

void handle_root(void);

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);
}
```

FUNCIONAMIENTO

Al principio hacemos *include* de dos librerías, la primera `#include <WiFi.h>` sirve para permitir que nuestra placa se conecte a internet, y la segunda librería `#include <WebServer.h>` es para hacer un servidor web.

A continuación en una *char* ponemos el SSID de nuestro WiFi y en otra la contraseña de este.

```
const char* ssid = "MIWIFI_2G_2jcE";
const char* password = "9GLX7t3u";
```

Dentro del `void setup()` primero imprimimos por pantalla:
Try Connecting to:
MIWIFI_2G_2jcE

Lo hacemos utilizando el siguiente código:

```
Serial.println("Try Connecting to ");
Serial.println(ssid);
```

Despues conectamos el mòdem WiFi a la placa con:

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

Una vez hecho esto debemos comprobar si el WiFi está conectado a la red WiFi, lo hacemos con el código:

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

Este bucle lo comprueba y si está conectado imprime un punto por pantalla.

A continuación imprimimos por pantalla que el WiFi se ha conectado correctamente y también imprimimos la IP del ESP32.

```
Serial.println("");
Serial.println("WiFi connected successfully");
Serial.print("Got IP: ");
Serial.println(WiFi.localIP());
```

Y para finalizar el `void setup()` empezamos el servidor HTTP con:

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

En el Loop, debemos llamar a la función `handleClient()` que se encarga de recibir las peticiones de los clientes y lanzar las funciones de callback asociadas en el ruteo.

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

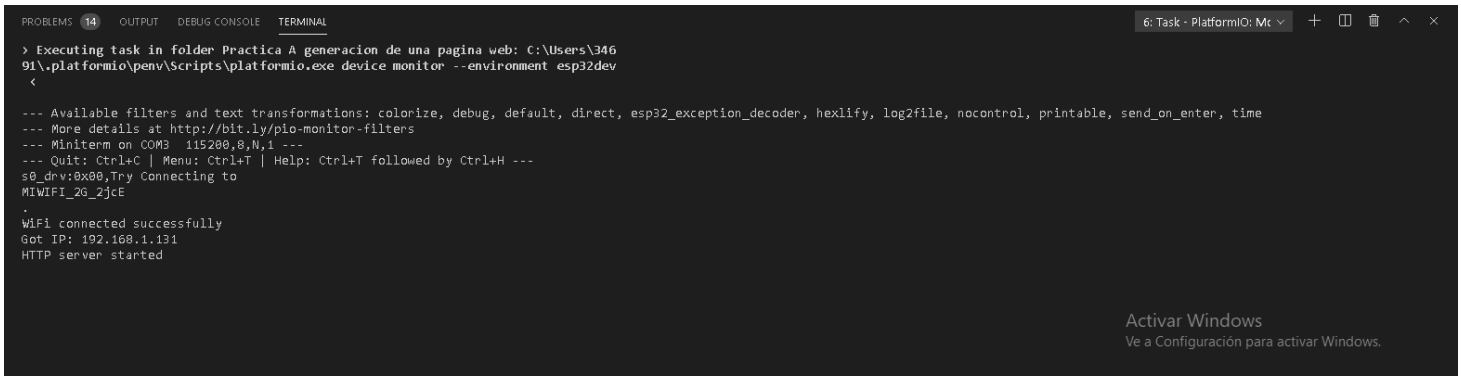
Ya casi para finalizar el programa creamos una string de nombre HTML que son los contenidos de tipo HTML que se van a mostrar en el servidor web, en nuestro caso se mostrara: My Primera Pagina con ESP32 - Station Mode 😊.

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

Y para finalizar, creamos la función `void handle_root()` que se ha llamado anteriormente dentro del `void setup()` . Dentro de esta función utilizamos la función send, la cual tiene tres parámetros: el primero es el código de respuesta (200, 301, 303, 404...), el segundo es el tipo de contenido HTTP (text/plain, text/html, text/json, image/png...) y el último el contenido del cuerpo de la respuesta, que en nuestro caso es el string HTML.

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

SALIDA POR EL TERMINAL



VISUALIZACIÓN DE LA CONEXIÓN A LA PAGINA WEB CON UN NAVEGADOR

My Primera Pagina con ESP32 - Station Mode 😊

APARTADO 2

PRACTICA B INTERRUPCION POR TIMER

CODIGO

```
#include

volatile int interruptCounter;

int totalInterruptCounter;

hw_timer_t * timer = NULL;

portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR onTimer() {

portENTER_CRITICAL_ISR(&timerMux);

interruptCounter++;

portEXIT_CRITICAL_ISR(&timerMux);

}

void setup() {

Serial.begin(9600);

timer = timerBegin(0, 80, true);

timerAttachInterrupt(timer, &onTimer, true);

timerAlarmWrite(timer, 1000000, true);

timerAlarmEnable(timer);

}

void loop() {

if (interruptCounter > 0) {

portENTER_CRITICAL(&timerMux);

interruptCounter--;

portEXIT_CRITICAL(&timerMux);
```



```
totalInterruptCounter++;

Serial.print(An interrupt as occurred. Total number: );

Serial.println(totalInterruptCounter);

}

}
```

FUNCIONAMIENTO

Empezamos declarando el contador de interrupciones con **volatile**, lo que evitara que se elimine a causa de optimizaciones del compilador.

```
volatile int interruptCounter;
```

Declaramos tambien un contador para ver cuantas interrupciones han ocurrido desde el principio del programa, este no requiere **volatile**, ya que solo vamos a usarlo en el *loop* principal (*main loop*).

```
int totalInterruptCounter;
```

Para la configuracion del timer que haremos mas adelante, primero necesitaremos un puntero.

```
hw_timer_t * timer = NULL;
```

Y la ultima variable que tenemos que declarar es del tipo `portMUX_TYPE`, esta la usaremos para la sincronizaciÃ³n entre el *main loop* y la ISR.

```
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

A continuacion vamos a declarar el siguiente **void**.

```
void setup() {
```

```
Serial.begin(9600);

timer = timerBegin(0, 80, true);

timerAttachInterrupt(timer, &onTimer, true);

timerAlarmWrite(timer, 1000000, true);

timerAlarmEnable(timer);

}
```

Dentro de este tenemos la inicializaciÃ³n de nuestro timer llamando a la funcion `timerBegin`. Esta funcion recibe como entrada el numero del temporizador que queremos usar (en nuestro caso el 0), el valor del prescaler (en nuestro caso 80) e indicamos si el contador cuenta hacia adelante (*true*) o si lo hace hacia atras (*false*).

```
timer = timerBegin(0, 80, true);
```

Ahora usaremos la funcion `timerAttachInterrupt`, esta recibe como entrada un puntero al temporizador, la direccion a la funcion `onTimer` que mÃ¡s tarde especificaremos y sirve para manejar la interrupcion y finalmente el valor *true* para especificar que la interrupcion es de tipo *edge*.

```
timerAttachInterrupt(timer, &onTimer, true);
```

A continuacion usaremos la funcion `timerAlarmWrite`, esta tambien recibe como entrada tres valores: Como primera entrada recibe el puntero al temporizador, como segunda el valor del contador en el que se tiene que generar la interrupcion y finalmente un indicador de si el temporizador se ha de recargar automaticamente al generar la interrupcion.

```
timerAlarmWrite(timer, 1000000, true);
```

Finalmente llamamos a la funcion `timerAlarmEnable`, en esta pasamos como entrada nuestra variable de temporizador.

```
timerAlarmEnable(timer);
```


El *main loop* completo es el siguiente:

```
void loop() {

if (interruptCounter > 0) {

portENTER_CRITICAL(&timerMux);

interruptCounter--;

portEXIT_CRITICAL(&timerMux);

totalInterruptCounter++;

Serial.print(An interrupt as occurred. Total number: );

Serial.println(totalInterruptCounter);

}

}
```

Este programa basicamente consiste en incrementar el contador con el numero total de interrupciones (`totalInterruptCounter`) i imprimirlo al puerto serie.

Para acabar explicaremos como funciona la funcion ISR (*interrupt service routine*). Esta consistira en incrementar el contador de interrupciones que indicara al bucle principal que se ha producido una interrupcion. Esto se produce dentro de una secciÃ³n critica, declarada con `portENTER_CRITICAL_ISR` y `portEXIT_CRITICAL_ISR`. El codigo completo del ISR queda:

```
void IRAM_ATTR onTimer() {

portENTER_CRITICAL_ISR(&timerMux);

interruptCounter++;

portEXIT_CRITICAL_ISR(&timerMux);

}
```

IMPRESION SERIE

Este programa va a imprimir por pantalla el numero de veces que hay una interrupcion. Por lo tanto imprimira algo como:

```
An intrrupt as occurred. Total number: 1

An intrrupt as occurred. Total number: 2

An intrrupt as occurred. Total number: 3

An intrrupt as occurred. Total number: 4

An intrrupt as occurred. Total number: 5

An intrrupt as occurred. Total number: 6

An intrrupt as occurred. Total number: 7

...
```


Y asi indefinidamente hasta que no se pare manualmente.