**TECNOLÓGICO NACIONAL DE MEXICO**

**INSTITUTO TECNOLÓGICO DE LA LAGUNA**

****

**REPORTE DE PRACTICA**

**UNIDAD 5: Puertos y buses de comunicación para microcontroladores**

**PRÁCTICA 21: WiFi con una app movil**

**DOCENTE: LAMIA HAMDAN M.**

| **NUM DE CONTROL** | **NOMBRE** |
| --- | --- |
| 19130514 | Isaias Gerardo Cordova Palomares |
| 19130545 | Oscar Martinez Ruiz |
| 17130763 | Raúl Martín Ayala Salais |
| 19130541 | Pedro Lopez Ramirez |
| 19130535 | Ivan Herrera Garcia |
| 18131263 | Gerardo Alberto Orozco Villegas |

**FECHA DE ENTREGA:** 12/11/2022

**TABLA DE CONTENIDO**

[**1.INTRODUCCIÓN**](#_555xbbp1n8y5) **3**

[**2. COMPETENCIA A DESARROLLAR**](#_30j0zll) **3**

[**3. CIRCUITO LÓGICO Y/O PROGRAMA**](#_vyaxf5uqhjrg) **3**

[**4. METODOLOGÍA**](#_uikunnlbl3c) **3**

[**5. RESULTADOS**](#_4j5zl4o09ir7) **4**

[**6.CONCLUSIONES Y RECOMENDACIONES**](#_swudbzgg3llj) **5**

[**7. REFERENCIAS**](#_s8u6prgrx9vv) **5**

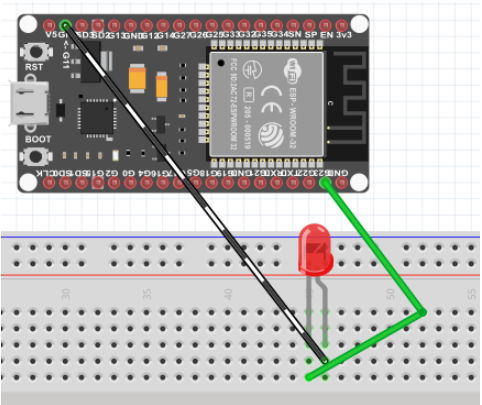
# **1.INTRODUCCIÓN**

Para la realización de la práctica se diseñó y compilar un proyecto en Android en JAVA el cual tomará la dirección que proporciona el ESP32 para acceder a su página y encender o apagar un led.

# **2. COMPETENCIA A DESARROLLAR**

* Identifica las características eléctricas de un microcontrolador.
* Conoce la arquitectura interna del microcontrolador.
* Comprende la estructura de registros del microcontrolador.
* Analiza dispositivos de entrada/salida y puertos del microcontrolador.
* Organiza y clasifica información proveniente de fuentes diversas.

# **3. CIRCUITO LÓGICO Y/O PROGRAMA**



*Montaje del circuito en think cad/fritzing*

# **4. METODOLOGÍA**

**Material necesario:**

* ESP 32 Dev
* Cables
* Protoboard
* Led
* Equipo de cómputo con Android Studio (SDK: Android)

En esta práctica se utilizó el código de la práctica anterior, lo nuevo fue el diseño de la aplicación en android.

[Presione para descargar un archivo zip con el proyecto](https://drive.google.com/file/d/13yNZu3hGcyCHxFj7qgXw-dMsNfcdnXQX/view?usp=sharing)

Se diseñó en JAVA y se utilizó un componente llamado WebView el cual puede mostrar un sitio web ingresando la URL del sitio. En este caso se coloco “http://192.168.4.1/H”

Mientras que las conexiones físicas fueron las mismas que en la práctica 29.

# **5. RESULTADOS**

La práctica fue relativamente sencilla, solo se tuvo que buscar información acerca de los componentes necesarios para diseñar una aplicación que mostrará información de una página web.

El único inconveniente fue que no todos los dispositivos detectan la red inalámbrica, en este caso no todos los Iphone reconocen la red o no pueden conectarse a la misma. Debido a la seguridad de los mismos.



*Demostración de práctica*

[Enlace a vídeo de práctica](https://drive.google.com/file/d/18UTQdmV51cg5N9sTK11lmQxLEmNrcMr8/view?usp=sharing)

**Codigo**

/\*

WiFiAccessPoint.ino creates a WiFi access point and provides a web server on it.

Steps:

1. Connect to the access point "yourAp"

2. Point your web browser to http://192.168.4.1/H to turn the LED on or http://192.168.4.1/L to turn it off

OR

Run raw TCP "GET /H" and "GET /L" on PuTTY terminal with 192.168.4.1 as IP address and 80 as port

Created for arduino-esp32 on 04 July, 2018

by Elochukwu Ifediora (fedy0)

\*/

#include <WiFi.h>

#include <WiFiClient.h>

#include <WiFiAP.h>

//#define LED\_BUILTIN 2 // Set the GPIO pin where you connected your test LED or comment this line out if your dev board has a built-in LED

#define LED 23

// Set these to your desired credentials.

const char \*ssid = "juan electronico";

const char \*password = "123123123";

WiFiServer server(80);

void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

Serial.begin(115200);

Serial.println();

Serial.println("Configuring access point...");

// You can remove the password parameter if you want the AP to be open.

WiFi.softAP(ssid, password);

IPAddress myIP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(myIP);

server.begin();

Serial.println("Server started");

pinMode (LED, OUTPUT);

}

void loop() {

WiFiClient client = server.available(); // listen for incoming clients

if (client) { // if you get a client,

Serial.println("New Client."); // print a message out the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected()) { // loop while the client's connected

if (client.available()) { // if there's bytes to read from the client,

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println();

// the content of the HTTP response follows the header:

client.print("<a href=\"/H\"><button name="button">Click me</button></a>");

client.print("<a href=\"/L\"><button name="button">Click me</button></a>");

// The HTTP response ends with another blank line:

client.println();

// break out of the while loop:

break;

} else { // if you got a newline, then clear currentLine:

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

}

// Check to see if the client request was "GET /H" or "GET /L":

if (currentLine.endsWith("GET /H")) {

digitalWrite(LED, HIGH); // GET /H turns the LED on

}

if (currentLine.endsWith("GET /L")) {

digitalWrite(LED, LOW); // GET /L turns the LED off

}

}

}

// close the connection:

client.stop();

Serial.println("Client Disconnected.");

}

}

# **6.CONCLUSIONES Y RECOMENDACIONES**

Se podría decir que a lo largo de esta practica no se tuvieron mayores complicaciones, solo se tuvo que dedicar algo de tiempo para la investigación en diversos aspectos. Algunas de esas cosas iban desde que Framework y lenguaje de programación utilizar, en un inicio pensábamos realizar esta practica con flutter, mas sin embargo al investigar nos encontramos que la librería no era del todo compatible con nuestra placa de ESP32. Por lo que nos fuimos por algo mas seguro, que fue realizar la aplicación de manera nativa programando desde Android Studio con Java.

# **7. REFERENCIAS**

[1] "ESP32 Wifi + Bluetooth en un solo lugar". Programar fácil con Arduino.<https://programarfacil.com/esp8266/esp32/> (accedido el 10 de noviembre de 2022).

[2] "ESP32 conexión via WiFi".<https://www.juanjobeunza.com/esp32-wifi/> (accedido el 10 de noviembre de 2022).

[3] "arduino-esp32/WiFi.h at master · espressif/arduino-esp32". GitHub.<https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h> (accedido el 10 de noviembre de 2022).

[4] "Arduino/WiFiClient.h at master · esp8266/Arduino". GitHub.<https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiClient.h> (accedido el 10 de noviembre de 2022).

[5] "arduino-esp32/WiFiAP.h at master · espressif/arduino-esp32". GitHub.<https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFiAP.h> (accedido el 10 de noviembre de 2022).

[6] "Fritzing". Welcome to Fritzing.<https://fritzing.org/> (accedido el 10 de noviembre de 2022).