**TECNOLÓGICO NACIONAL DE MEXICO**

**INSTITUTO TECNOLÓGICO DE LA LAGUNA**

****

**REPORTE DE PRACTICA**

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

**PRÁCTICA 18 y 19: WiFi**

**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:** 13/10/2022

**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**

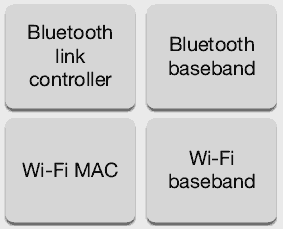
# **1.INTRODUCCIÓN**

En la unidad se vio el funcionamiento del dispositivo ESP32 con Bluetooth.

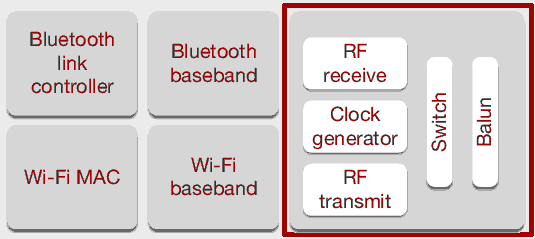
Investigando hemos descubierto que este dispositivo puede funcionar tambien en la modalidad de WiFi.

[1] “ESP 32 es muy superior, en capacidades a un Arduino UNO y a un ESP8266, pero eso no quiere decir que sea ideal para cualquier proyecto, ya que para sacarle todo el jugo a este microprocesador es necesario un proyecto mas complejo del que estamos acostumbrado”

**Conectividad Inalambrica**

****

El chip cuenta con conectividad WiFi, siendo compatible con 802.11 b/g/n en la banda de los 2.4GHz, alcanzando velocidades de hasta 150Mbits/s.



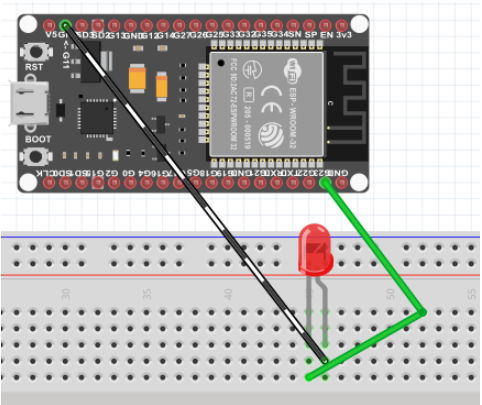
El bloque de radio está estrechamente ligado a los módulos de comunicación inalámbrica. De hecho, este es el que realmente transmite y recibe información.

Es decir, toma los datos digitales provenientes de los módulos wifi y bluetooth; y los convierte en señales electromagnéticas que viajan por el aire para comunicarse el teléfono o router.

# **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**

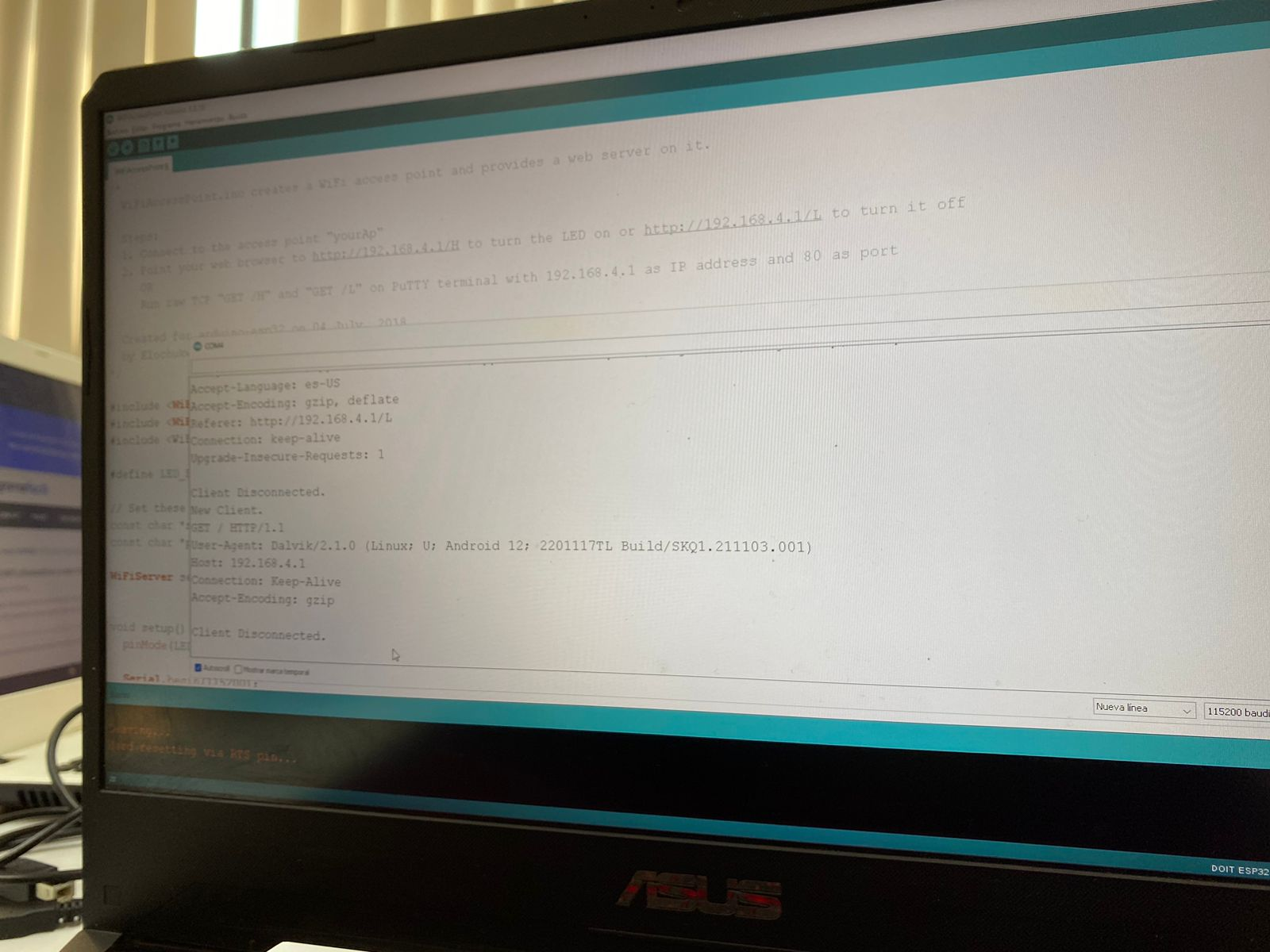
**Material necesario:**

* ESP 32 Dev
* Cables
* Protoboard
* Led

# **5. RESULTADOS**

Como resultado de las practicas en conjunto, se obtuvo una conexión exitosa de arduino

A continuación anexaremos la siguiente liga para que se pueda ver la conexión de nuestro ESP 32 :



[Video Demostración de la conexión entre ESP32 y WiFi](https://drive.google.com/file/d/1yn0KvpmSknUnBHvkFFJ9nEYnOFoED4mZ/view?usp=share_link)

**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**

En general , podríamos decir que fue una practica entre sencilla y compleja, mas que nada nos costo algo de tiempo asimilar un poco el como funciona la conexión del esp 32 via wifi. Mas sin embargo una vez que pillamos el truco, logramos concretar las practicas sin mayor dificultad/

# **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).