Date:

EXPERIMENT: 13

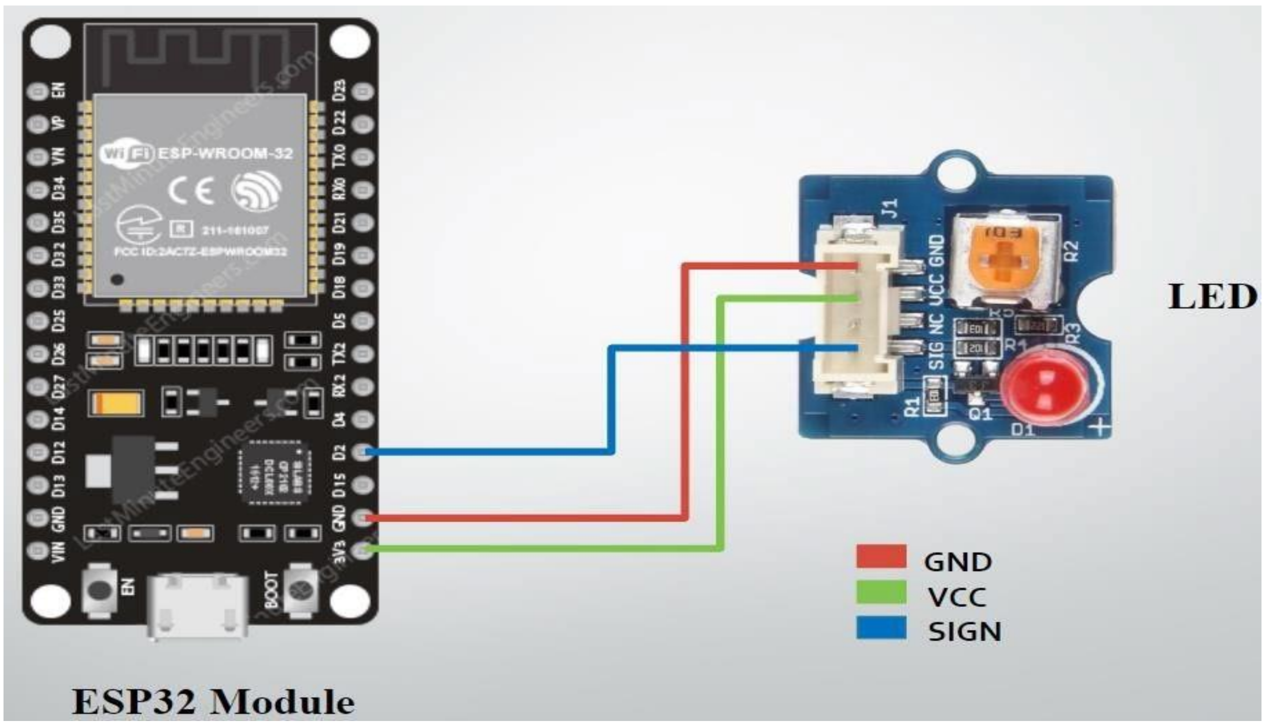
AIM: Learn to utilize Wi-Fi Module on ESP32 and implement codes to : i. scan Wi-Fi networks ii. Set up a simple Wi-Fi web server to blink an LED from the web and iii. set up a Wi-Fi access point and provide a web server on it.

OBJECTIVES:

1. Implement code to scan Wi-Fi networks
2. Implement code to set up a simple Wi-Fi web server to blink an LED from the web
3. Implement code to set up a Wi-Fi access point and provide a web server on it

COMPONENTS:

CONNECTION DIAGRAM:



CODES:

13.1:

```
#include "WiFi.h" void setup()
{
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);    WiFi.disconnect();    delay(100);
  Serial.println("Setup done");
}
void loop()
{
  Serial.println("scan start");    int n = WiFi.scanNetworks();    Serial.println("scan done");
if (n == 0)
  {
    Serial.println("no networks found");
  }    else
  {
    Serial.print(n);
    Serial.println(" networks found");       for (int i = 0; i < n; ++i)
    {
      // Print SSID and RSSI for each network found Serial.print(i + 1);       Serial.print(":
");
     Serial.print(WiFi.SSID(i));
     Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
 Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN) ? " " : "*");       delay(10);
    }
  }
  Serial.println("");    delay(5000);
}
```

13.2:

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

```
const char* ssid = "DEVAL";
const char* password = "12345678";
WiFiServer server(80);
void setup()
{
  Serial.begin(115200);
  pinMode(2, OUTPUT);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.print("Connecting TO ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
 }
  int value = 0;
  void loop() {
    WiFiClient client = server.available();
    if(client) {
      Serial.println("New Client.");
      String currentLine = "";
      while (client.connected()) {
        if (client.available()) {
          char c = client.read();
        Serial.write(c);
        if(c =='\n') {
          if(currentLine.length() ==0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.print("Click <a href=\"/H\"here</a> to turn the LED on pin 2 on.<br>");
            client.print("Click <a href=\"/L\"here</a> to turn the LED on pin 2 off.<br>");
            client.println();
            break;
          }else{
            currentLine = "";
          }
        }else if (c != '\r'){
          currentLine += c;
        }
        if(currentLine.endsWith("GET /H")) {
          digitalWrite(2, HIGH);
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(2, LOW);
        }
      }
    }
    client.stop();
    Serial.println("Client Disconnected");
  }
}
```

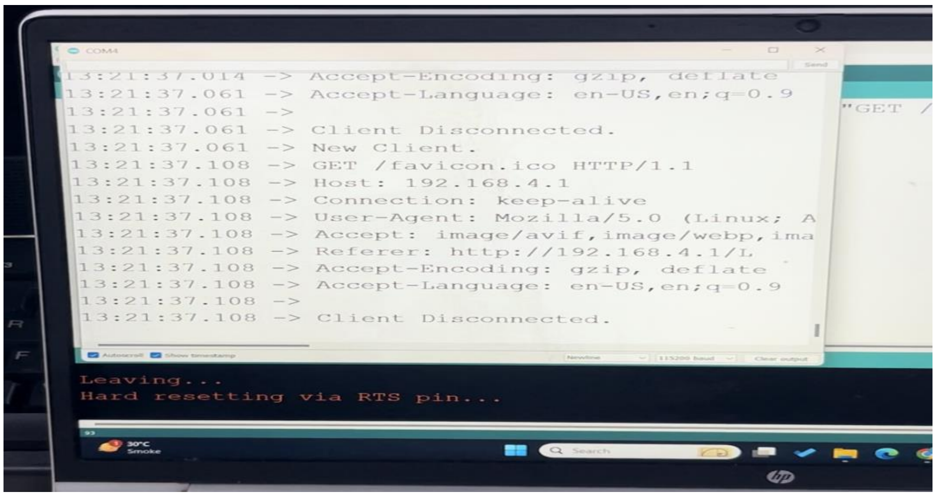
13.3:

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

```
#include <WiFiAP.h>
#define LED_BUILTIN 2   // Led Signal Pin
// Set these to your wifi access point credentials.
const char *ssid = "yourAP";
const char *password = "yourPassword";
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");
}
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("Click <a href=\"/H\">here</a> to turn ON the LED.<br>");
            client.print("Click <a href=\"/L\">here</a> to turn OFF the LED.<br>");
            // 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_BUILTIN, HIGH);               // GET /H turns the LED on
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(LED_BUILTIN, LOW);                // GET /L turns the LED off
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Client Disconnected.");
  }
}.
```

OUTPUTS:

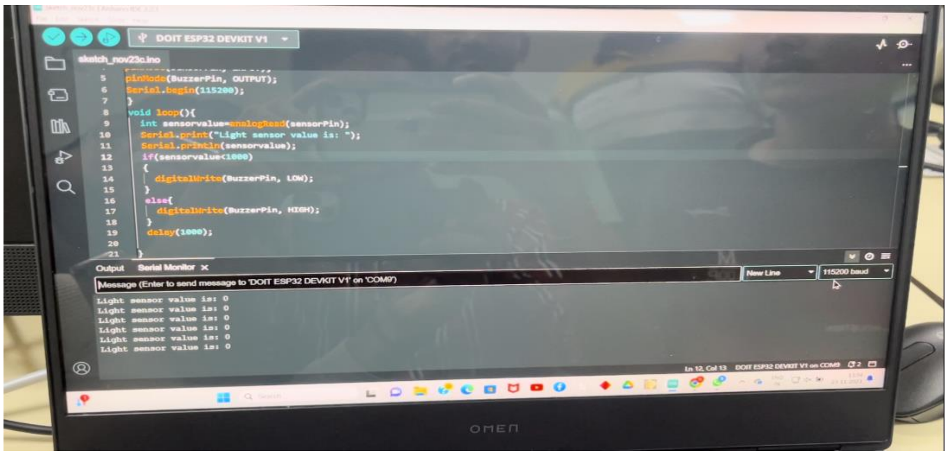
1.



```
13:21:37.014 -> Accept-Encoding: gzip, deflate
13:21:37.061 -> Accept-Language: en-US,en;q=0.9
13:21:37.061 ->
13:21:37.061 -> Client Disconnected.
13:21:37.061 -> New Client.
13:21:37.108 -> GET /favicon.ico HTTP/1.1
13:21:37.108 -> Host: 192.168.4.1
13:21:37.108 -> Connection: keep-alive
13:21:37.108 -> User-Agent: Mozilla/5.0 (Linux; A
13:21:37.108 -> Accept: image/avif,image/webp,ima
13:21:37.108 -> Referer: http://192.168.4.1/L
13:21:37.108 -> Accept-Encoding: gzip, deflate
13:21:37.108 -> Accept-Language: en-US,en;q=0.9
13:21:37.108 ->
13:21:37.108 -> Client Disconnected.
```

```
Leaving...
Hard resetting via RTS pin...
```

2.

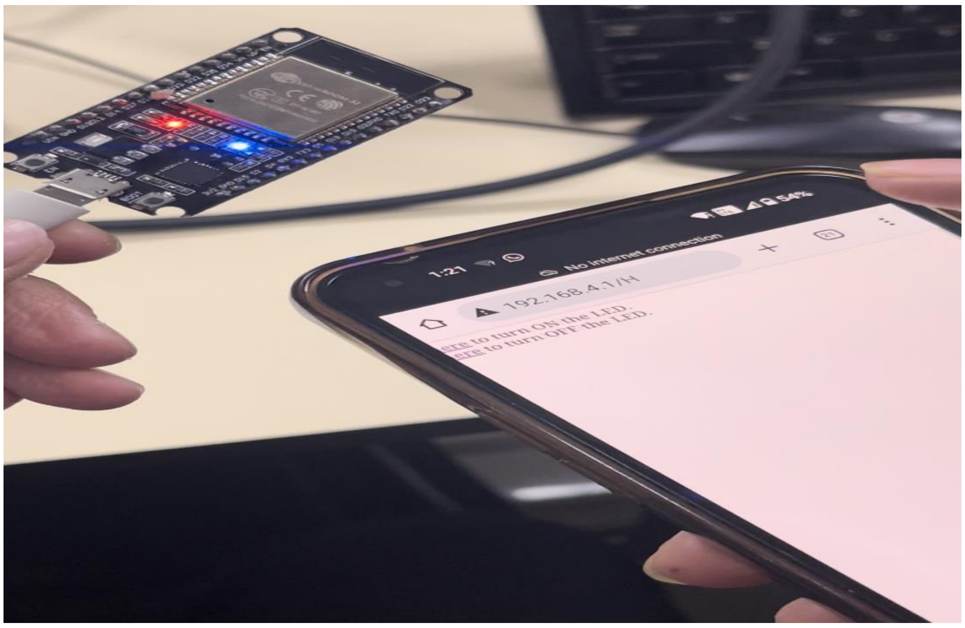


```
pinMode(BuzzerPin, OUTPUT);
Serial.begin(115200);
}
void loop(){
  int sensorvalue=analogRead(sensorPin);
  Serial.print("Light sensor value is: ");
  Serial.println(sensorvalue);
  if(sensorvalue<1000)
  {
    digitalWrite(BuzzerPin, LOW);
  }
  else{
    digitalWrite(BuzzerPin, HIGH);
  }
  delay(1000);
}
```

```
Light sensor value is: 0
Light sensor value is: 0
Light sensor value is: 0
Light sensor value is: 0
Light sensor value is: 0
Light sensor value is: 0
```

DOIT ESP32 DEVKIT V1
sketch_nov23c.ino
Output
Serial Monitor
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM9')
New Line
115200 baud
Ln 12, Col 13
DOIT ESP32 DEVKIT V1 on COM9
OMEN

3.



OBSERVATIONS:

CONCLUSION:

DRIVE LINK OF VIDEO:

1. **https://drive.google.com/file/d/1BijVP344BNwNPi5kYSOlj9Vl82d9H2gu/view?usp=drivesdk**
2. **https://drive.google.com/file/d/1uRaXNCl9b9Tlpnn3KL3dx1GNNeqDVIic/view?usp=drivesdk**
3. **https://drive.google.com/file/d/1A_H-JYHqqUVvbB9ogBIpG60f0YEKkbiR/view?usp=drivesdk**

SUBMITTED BY:

1. 23CS041- DHRUV LOKADIYA
2. 23CS045- MITUL MISTRY
3. 23CS046- KATHAN MODH