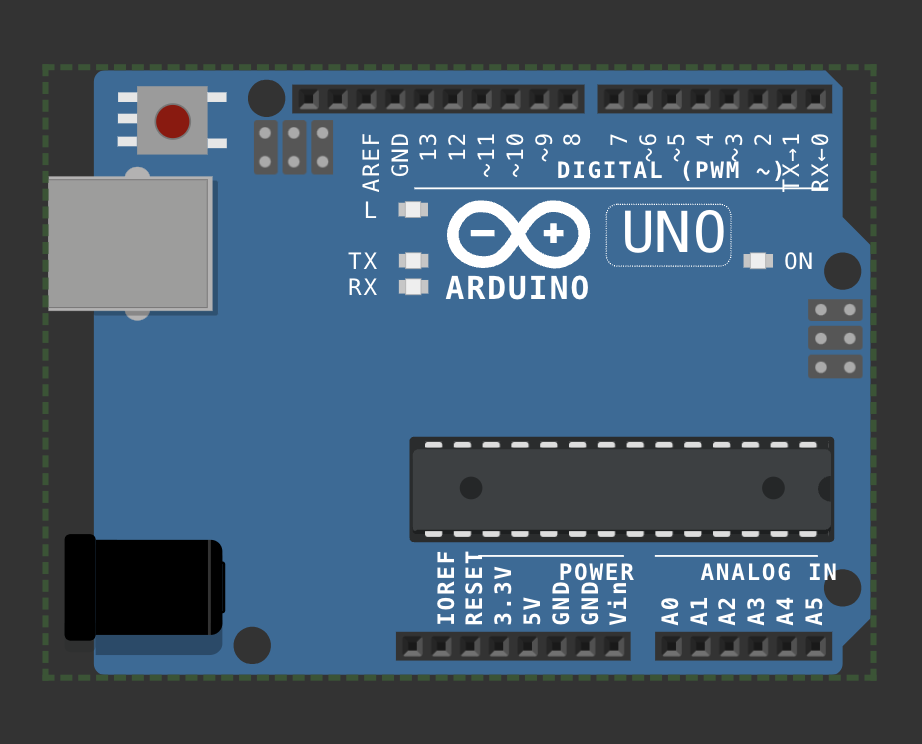
IoT & Automation Lab. Record

Dhruv Patel FET-BAML-2022-26-029

**LAB 1: Blinking The In-Built LED**

void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

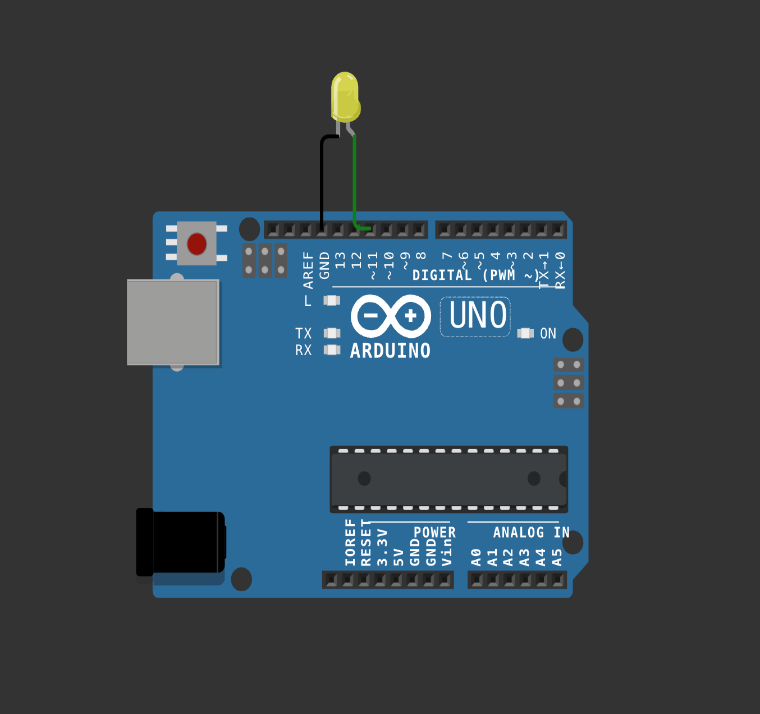
digitalWrite(LED\_BUILTIN, HIGH); delay(1000);

digitalWrite(LED\_BUILTIN,LOW); delay(1000);

}

**LINK** - [**WOKWI 1**](https://wokwi.com/projects/410064097034032129)

**LAB 2: Blinking An External LED**



#define led pin 11

void setup() {

pinMode(led pin, OUTPUT);

}

void loop() {

digitalWrite(led\_pin, HIGH);

delay(1000);

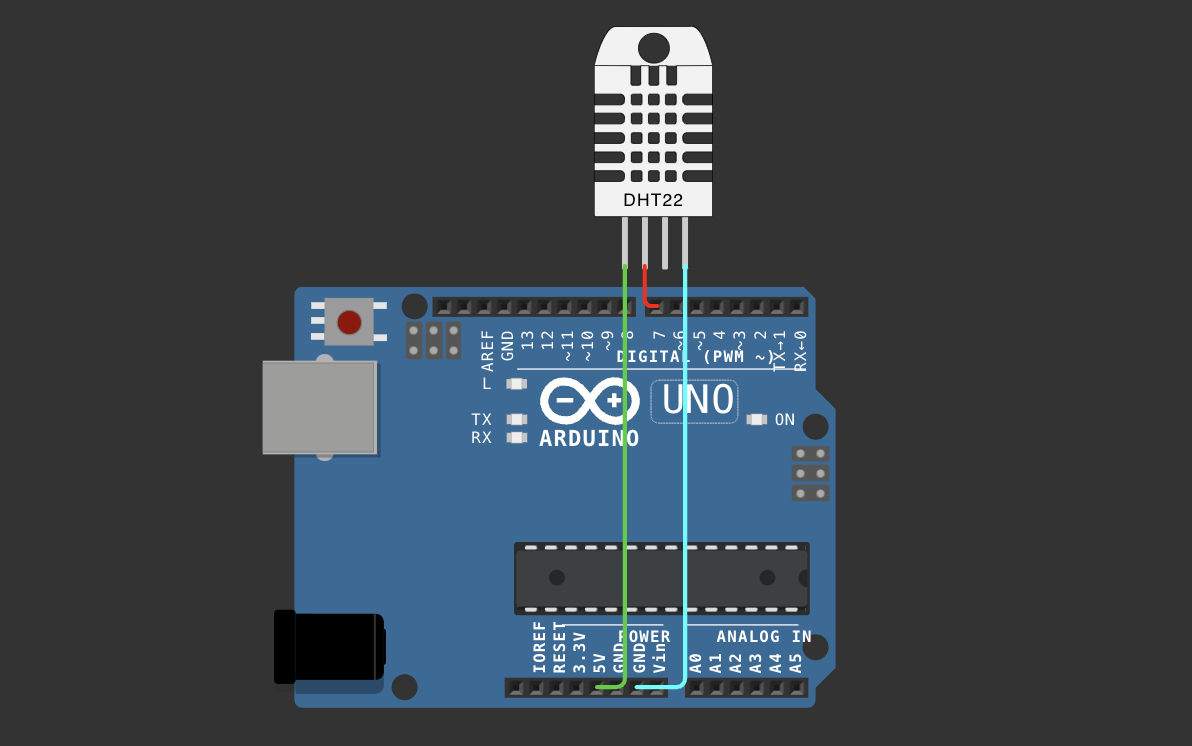
digitalWrite(led\_pin,LOW);

delay(1000);

}

[Link - wokwi 2](https://wokwi.com/projects/410063574396342273)

# LAB 3: Using DHT sensor



#include <DHT.h>

#define pin 7

#define DHTTYPE DHT22

DHT dht(pin, DHTTYPE);

float humid, temp;

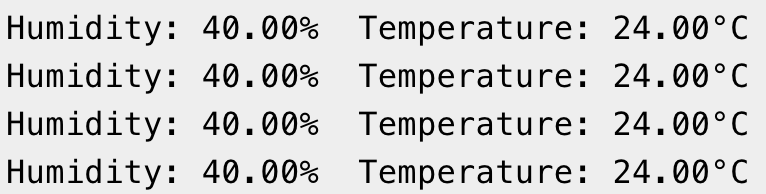
void setup() {

**Serial**.begin(9600);

dht.begin();

}

void loop() {

delay(200);

humid = dht.readHumidity();

temp = dht.readTemperature();

**Serial**.print("Humidity: ");

**Serial**.print(humid);

**Serial**.print("% Temperature: ");

**Serial**.print(temp);

**Serial**.println("°C");

delay(1000);

}

LINK - [Wokwi 3](https://wokwi.com/projects/410064543004003329)

# LAB 4: Using Mosquitto MQTT (Pub-Sub):

Starting Mosquitto MQTT:

# 

Setting up Publisher & Sending Message:

# 

Setting up Subscriber & Receiving Message:

# 

# LAB 5: Building A Web App Using Node-Red To Fetch DHT Sensor Data And Display It On The Web App Dashboard:

Installing & Initialising node red:

* Open node.js > npm install node-red-dashboard
* [postinstallation] > elevated cmd: node-red

In client application, browsed localhost:1880 [ accessing node red]:

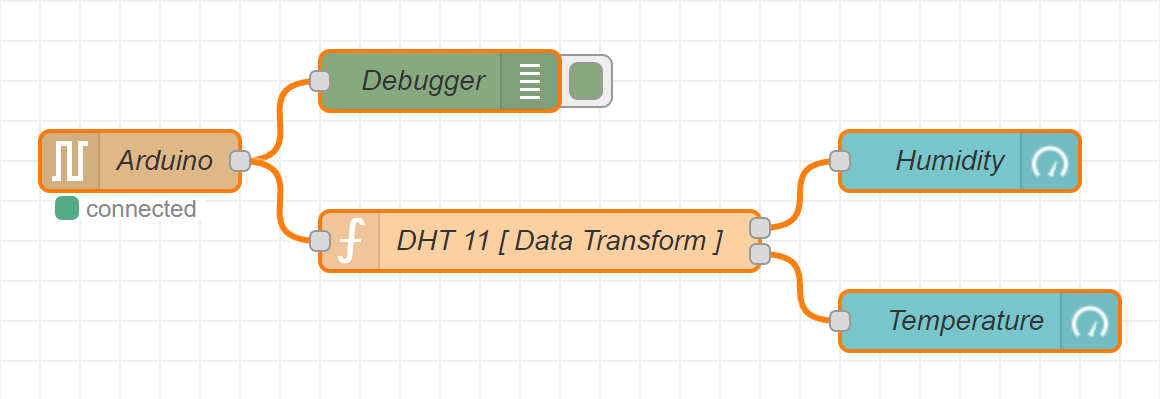
* Inside the nodered window, a flow was created w/ the nodes as:

            > serial-in ( arduino uno r3 board )

            > debugger

            > dht function

> 2 gauges (humidity& temperature)



* Serial in node: configured it to read from the correct serial port where my arduino is connected (e.g., com7) > set the baud rate to 9600.
* Configure the dht function as:

var m = msg.payload.split(',');

if (m.length === 2) {

var h = { payload: parsefloat(m[0]) };

            var t = { payload: parsefloat(m[1]) };

            return [h, t];

} else {

return null; }

* Adjusting Gauge Nodes:

Humidity:

        - Title as “ Humidity ”.

        - Value format as ‘ {{value}}% ’.

        - Range Value: 0 ~ 100 %.

Temperatue:

        - Title as ' Temperature '.

        - Value format as ‘ {{value}}°C ’.

*\*\*Ensure that Humidity & Temperature are in the same group*

Deployment:

* Uploaded DHT11 /22 Sketch to the Arduino Board through its IDE:

#include <dht.h>

#define dhtpin 3

#define dhttype dht11

dht dht(dhtpin, dhttype);

void setup() {

serial.begin(9600);

dht.begin();

}

void loop()  {

float h = dht.readhumidity();

float t = dht.readtemperature();

if (isnan(h) || isnan(t)) {

serial.println("failed to read from dht sensor!");

}

else {

serial.println(string(h) + "," + string(t));

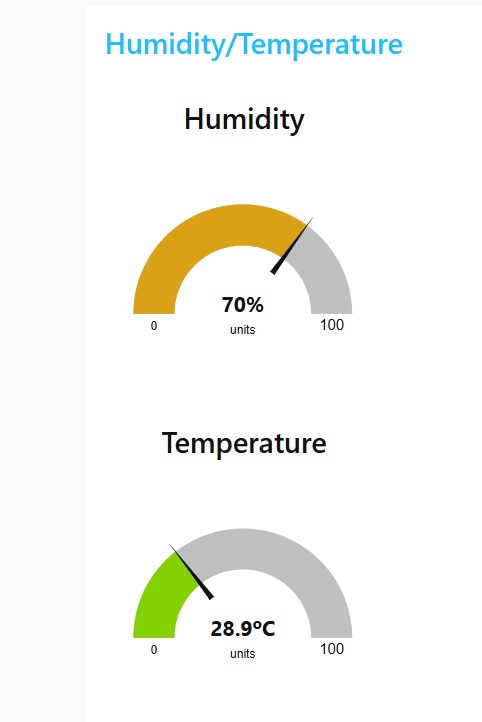
}

delay(2000);

}

* After uploading this sketch, close the IDE.
* Deploy the flow in NodeRED.
* Check the Dashboard in the upper-right corner, for the Humidity and Temperature Gauge.

OUTPUT ON THE DASHBOARD:



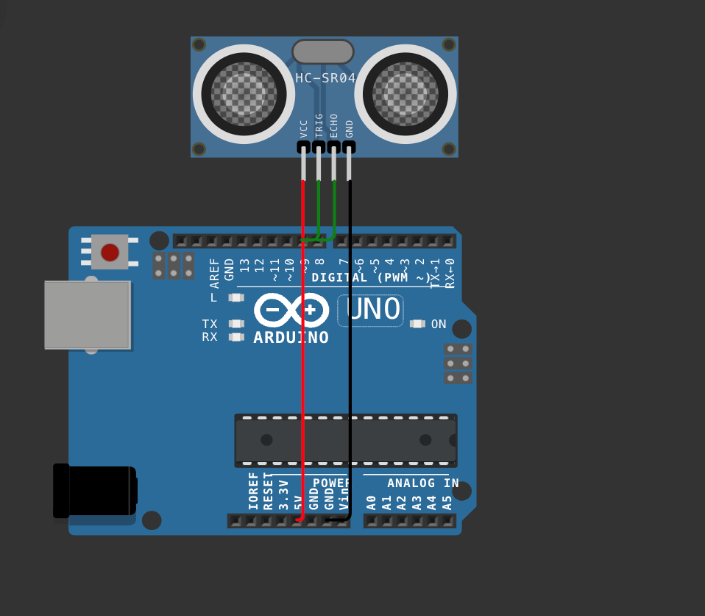
LAB 6: Working With Ultrasonic Sensors:

Measuring Distance of a Somethings Using Ultrasonic Sensors

**UltraSonic [ HC-SR04 ]: For Measuring Distance**

**Step 1:** Connect 4 jumper wires to the UltraSonic Sensor as [VCC: 5v], [Trigger: 9], [Echo: 8], & GND.

**Step 2:** After configuring the HC-SR04 w/ UNO R3, this sketch is to be uploaded and executed in the IDE:



#define PIN\_TRIG 9

#define PIN\_ECHO 8

void setup() {

Serial.begin(9600);

pinMode(PIN\_TRIG, OUTPUT);

pinMode(PIN\_ECHO, INPUT);

}

void loop() {

digitalWrite(PIN\_TRIG, HIGH);

delayMicroseconds(10);

digitalWrite(PIN\_TRIG, LOW);

int duration = pulseIn(PIN\_ECHO, HIGH);

Serial.print("Distance in CM: ");

Serial.println(duration / 58);

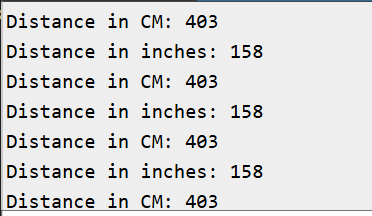
Serial.print("Distance in inches: ");

Serial.println(duration / 148);

delay(1000);

}

**Output:** Distance in CM: 403; Distance in inches: 158



Lab 7: Use Of Breadboard

Breadboard usage guide:

A breadboard is an essential tool for building and testing electronic circuits without soldering. It allows you to prototype and experiment with circuits quickly and conveniently.

Key features:

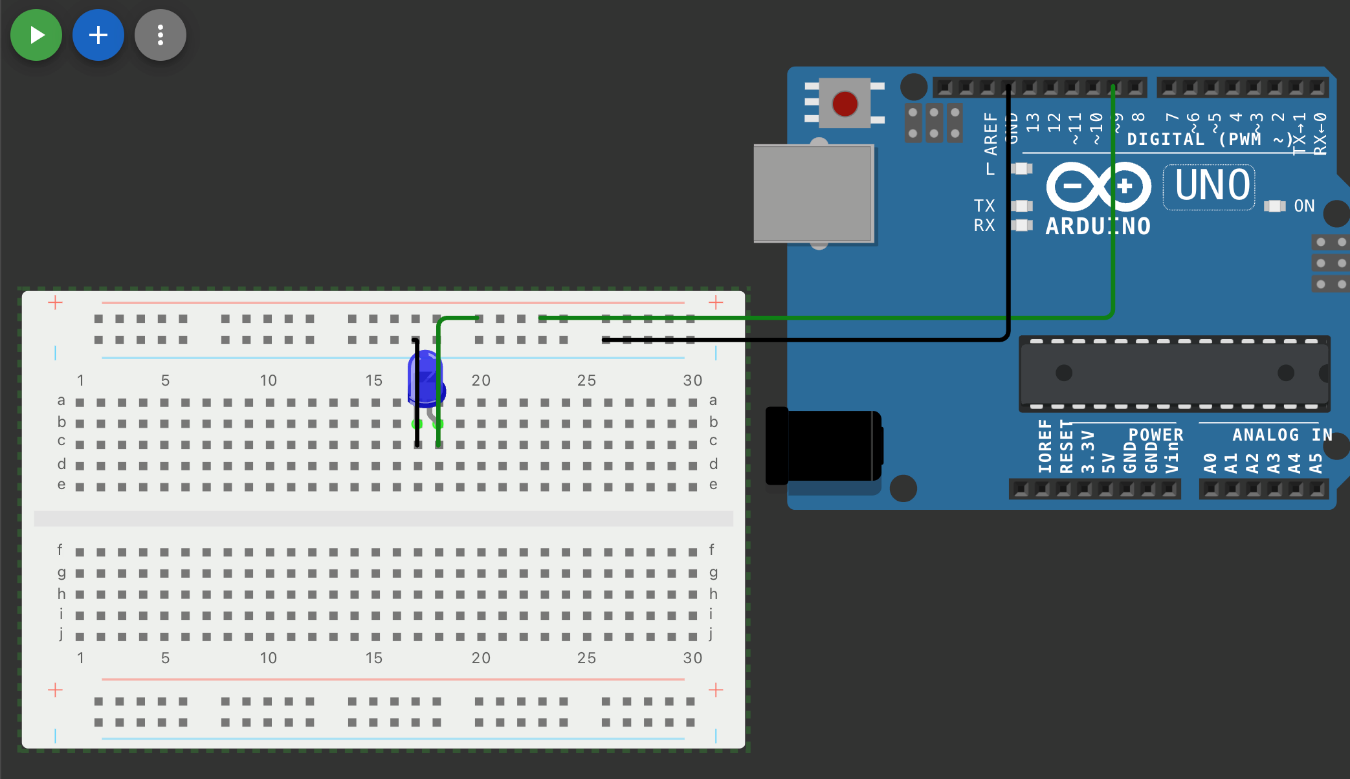
1. Rows And Columns: Breadboards are organized into numbered rows and lettered columns. The central rows are connected horizontally, while the outer rows, or power rails, run vertically to help distribute power.
2. Power Rails: Located along the edges and marked with red (+) and blue (-) lines, power rails are designed to supply power across the board.

How to use a breadboard:

1. Insert Components: Place the leads of each component into the breadboard holes.
2. Make Connections: Use jumper wires to connect components as needed.
3. Power Connections: Connect the power source to the power rails to easily distribute power across the circuit.
4. Avoid Overload: Breadboards are suitable for low-power circuits. High currents can damage the contacts, so avoid overloading them.

Tips:

* Plan The Layout: Arrange components and wires to keep the layout organized and clear.
* Check Connections: Ensure each wire and component is securely inserted to maintain good connections.



LAB 8: Use Of Esp32, Upload Code on Esp 32 To Blink Onboard Led

Step 1: Install esp32 board from the board manager

Step 2 : Choose the esp32 board from the port

Step 3: Compile and upload the code.

Step 4: Now, for establishing the connection, we would need to configure our mainframe w/ the cp210x usb-to-uart bridge virtual com port (vcp) driver avaliable at [cp210x usbtouart driver](https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip).

Step 5: After a superfluous reboot, the mainframe is ready to be used w/ an espressif32, as in the arduino ide, we first select the correct com port (*here, com12*), and esp32 dev module as the board.

* Note: While executing a sketch, the board reuires to be in download mode /boot mode, so for, the boot button is to be pressed while uploading the code onto the board, exactly post connecting... For 3-4 seconds.

Code 1 : Blinking Inbuilt Led

#define led\_pin 2

Void setup() {

pinmode(LED\_PIN, OUTPUT);

}

Void loop() {

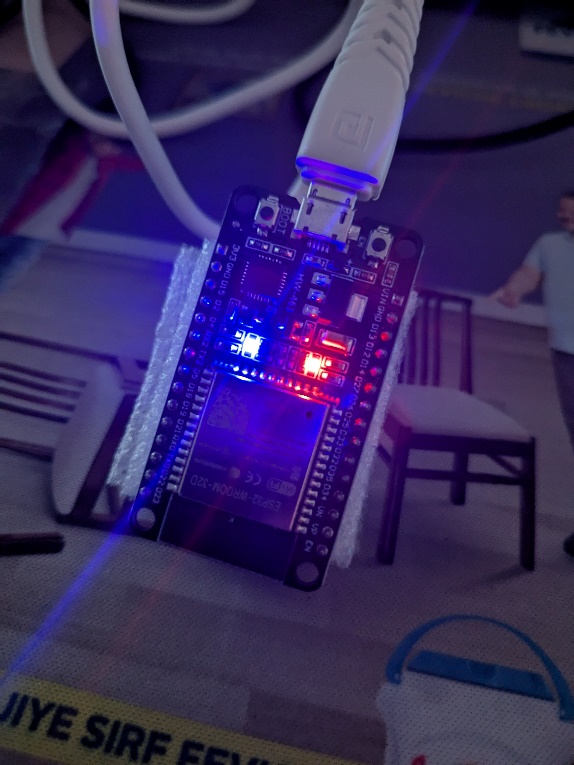
digitalwrite(LED\_PIN, HIGH);

delay(1000);

digitalwrite(LED\_PIN, LOW);

delay(1000);

}



Output:

Code 2 : Finding Nearby Using Esp32

#include "wifi.h"

Void setup() {

serial.begin(9600);

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");

serial.println("Nr | SSID | RSSI | CH | Encryption");

for (int i = 0; i < n; ++i) {

serial.printf("%2d", i + 1);

serial.print(" | ");

serial.printf("%-32.32s", WiFi.SSID(i).c\_str());

serial.print(" | ");

serial.printf("%4ld", WiFi.RSSI(i));

serial.print(" | ");

serial.printf("%2ld", WiFi.channel(i));

serial.print(" | ");

switch (WiFi.encryptiontype(i)) {

case WiFi\_auth\_open: serial.print("open"); break;

case WiFi\_auth\_wep: serial.print("WEP"); break;

case WiFi\_auth\_wpa\_psk: serial.print("WPA"); break;

case WiFi\_auth\_wpa2\_psk: serial.print("WPA2"); break;

case WiFi\_auth\_wpa\_wpa2\_psk: serial.print("WPA+WPA2"); break;

case WiFi\_auth\_wpa2\_enterprise: serial.print("WPA2-EAP"); break;

case WiFi\_auth\_wpa3\_psk: serial.print("WPA3"); break;

case WiFi\_auth\_wpa2\_wpa3\_psk: serial.print("WPA2+WPA3"); break;

case WiFi\_auth\_wapi\_psk: serial.print("WAPI"); break;

default: serial.print("unknown");

}

serial.println();

delay(10);

}

}

serial.println(“");

wifi.scandelete();

delay(5000);

}

Output:

