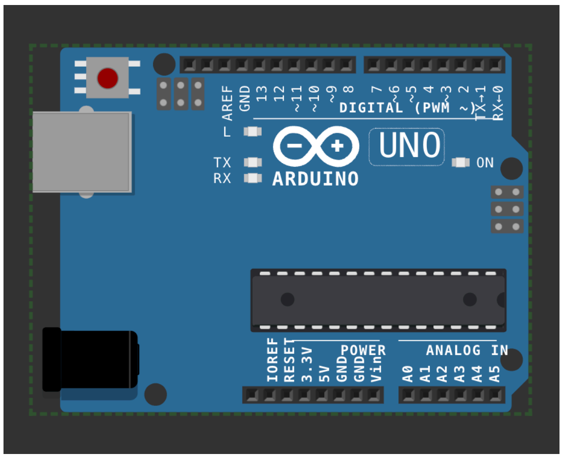
IoT & Automation Lab. Record

LAB 1 : Blinking the inBuilt LED

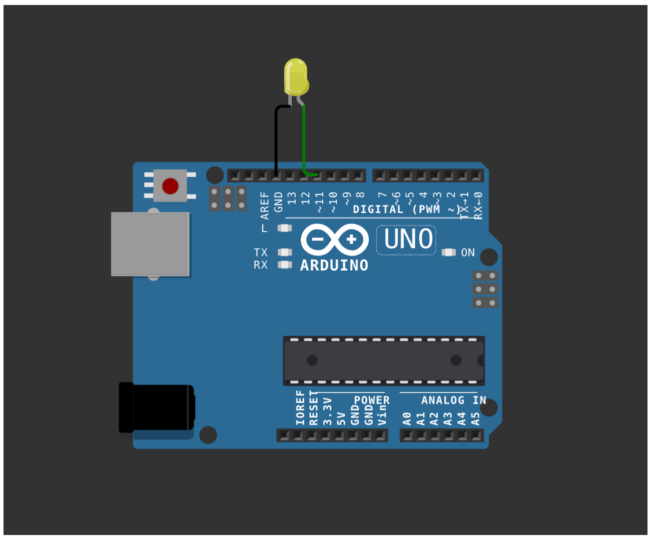
```
void setup() {
  pinMode(LED_BUILTIN,
OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN,
HIGH);
  delay(1000);

digitalWrite(LED_BUILTIN,LOW);
  delay(1000);
}
```



# Wokwi 1

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);
```

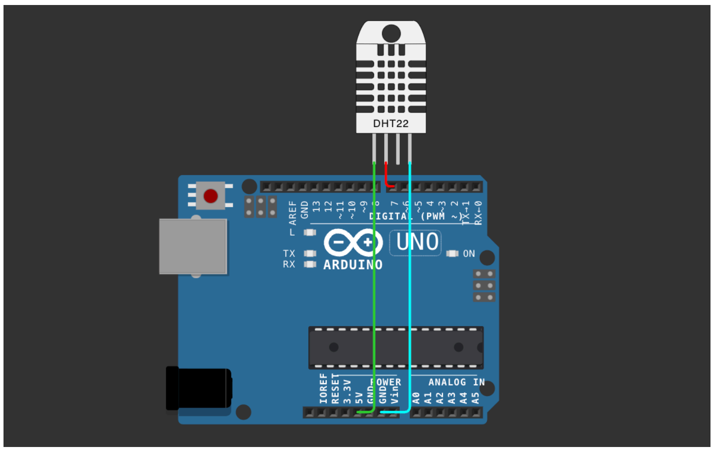


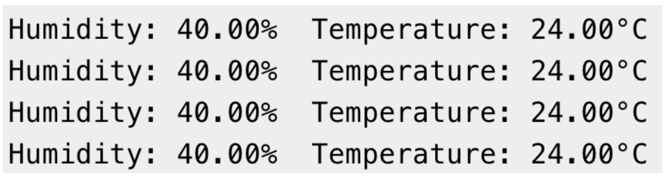
```
  delay(1000);
}
```

## wokwi 2

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);
```





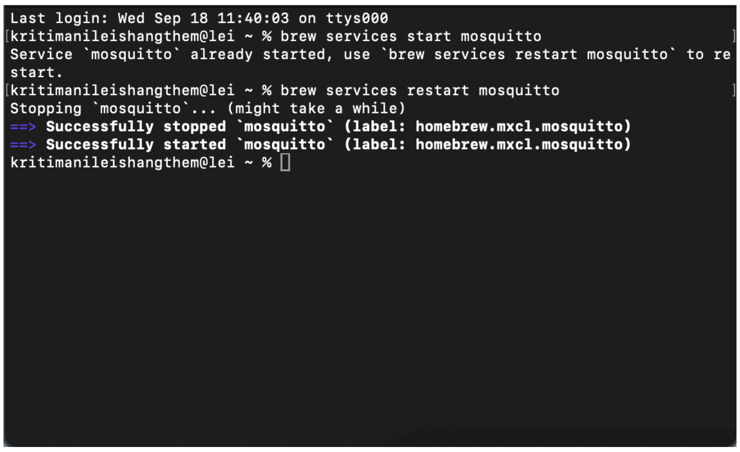
```
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
Humidity: 40.00%  Temperature: 24.00°C
```

```
  Serial.print("%  Temperature: ");
  Serial.print(temp);
  Serial.println("°C");
  delay(1000);
}
```

#Wokwi 3

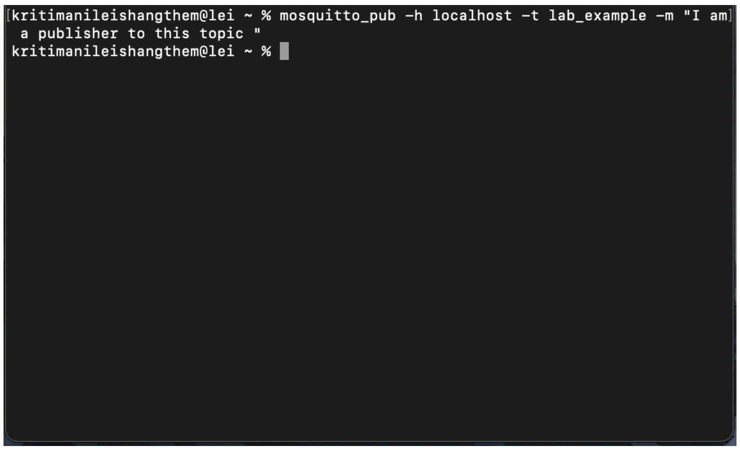
Lab 4: Using Mosquitto MQTT (Pub-Sub):

Starting Mosquitto MQTT:



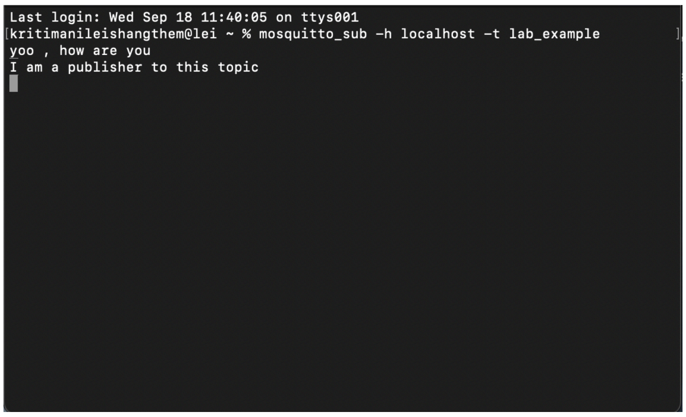
```
Last login: Wed Sep 18 11:40:03 on ttys000
kritimanileishangthem@lei ~ % brew services start mosquitto
Service `mosquitto` already started, use `brew services restart mosquitto` to re
start.
kritimanileishangthem@lei ~ % brew services restart mosquitto
Stopping `mosquitto`... (might take a while)
==> Successfully stopped `mosquitto` (label: homebrew.mxcl.mosquitto)
==> Successfully started `mosquitto` (label: homebrew.mxcl.mosquitto)
kritimanileishangthem@lei ~ %
```

Setting up Publisher & Sending Message:



```
kritimanileishangthem@lei ~ % mosquitto_pub -h localhost -t lab_example -m "I am
 a publisher to this topic "
kritimanileishangthem@lei ~ %
```

Setting up Subscriber & Receiving Message:



```
Last login: Wed Sep 18 11:40:05 on ttys001
kritimanileishangthem@lei ~ % mosquitto_sub -h localhost -t lab_example
yoo , how are you
I am a publisher to this topic
```

The mosquitto server is downloaded and implemented in the UNIX terminal of my laptop.

There are 3 photos of which 1 of each represent the part of an mqtt (a broker and client(subscriber and publisher)):
1. Start the mosquitto server by the typing the code [brew services start mosquitto].
2. in another terminal we made a subcriber to the [lab_example] topic using the code [mosquitto_sub -h localhost -t lab_example]
3. in another terminal i made a publisher to the topic [lab_example] using [mosquitto_pub -h localhost -t lab_example -m "\\I am a publisher to this topic\\"]
4. stop the server with the code [brew services stop mosquitto]

## We can also allow subscriber to subcribe to our topic using our network .

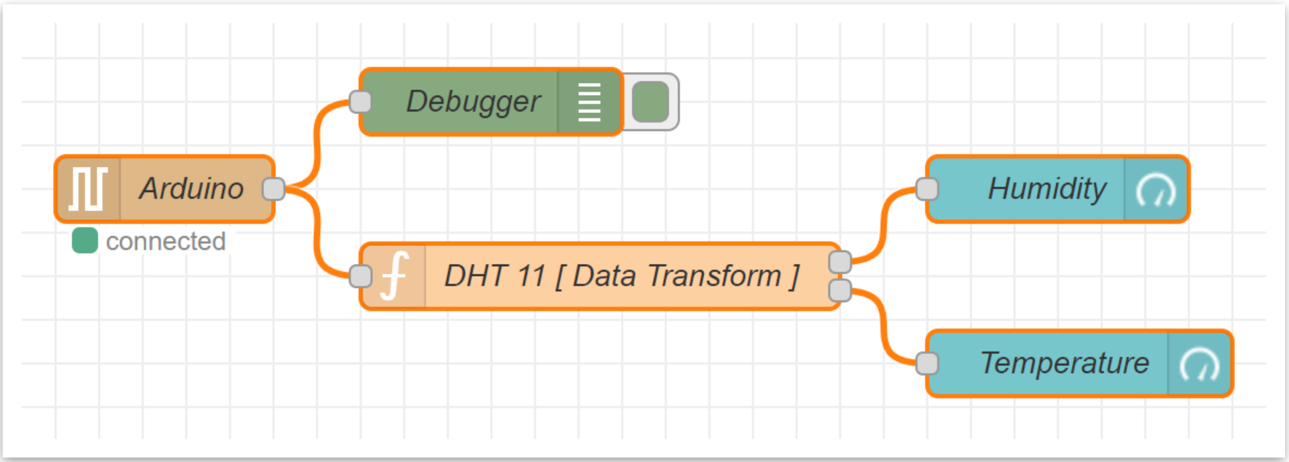
Lab 5: Building a web app using Node-Red to fetch DHT sensor data and display it on the web app dashboard

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*

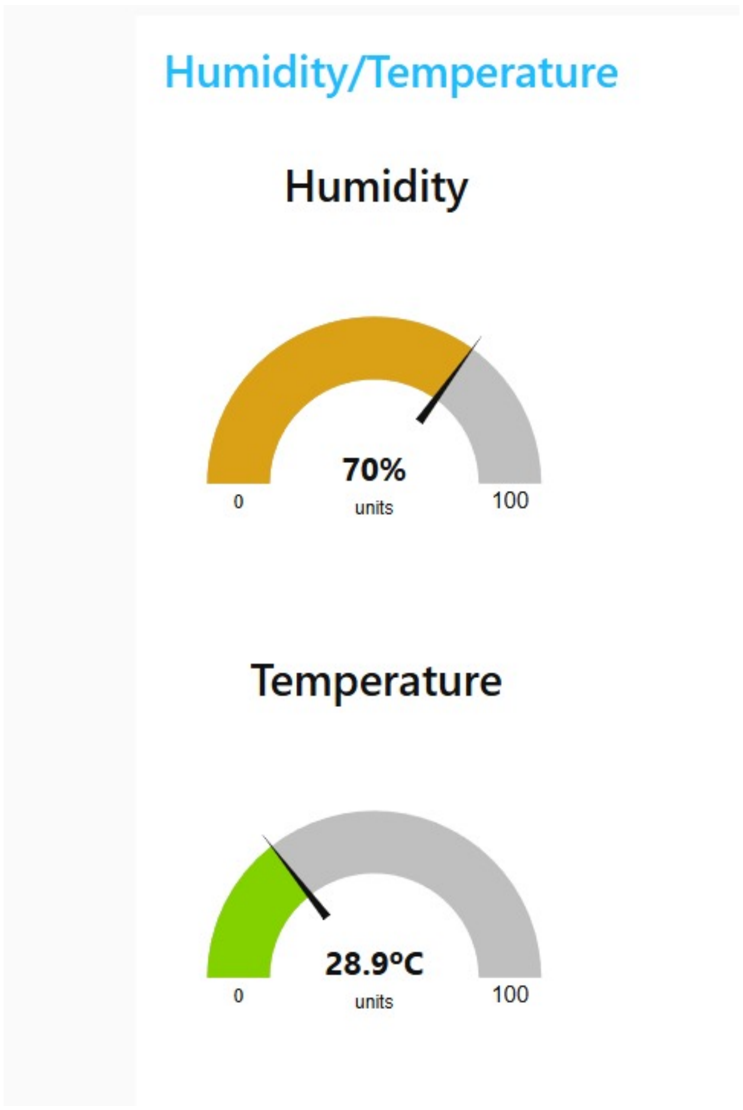
DEPLOYMENT:

- 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:

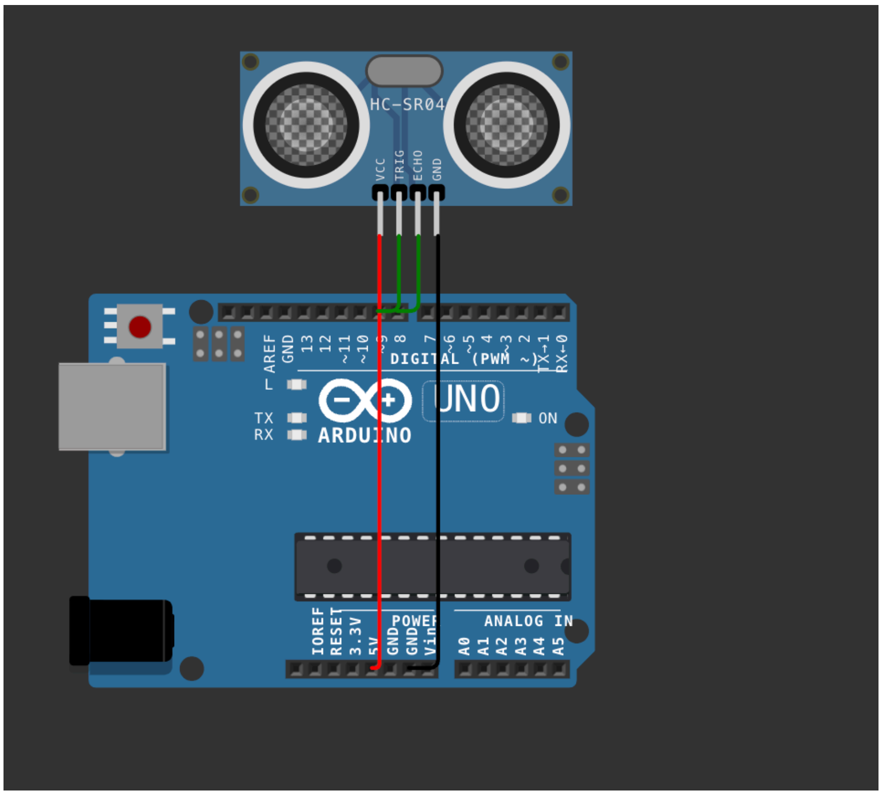


LAB 6: Working with ultrasonic sensors .

## Measuring distance of a somethings using ultrasonic sensors

CODE:

```
#include <DHT.h>

#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);
  float distanceCm = duration / 58.0;

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

  delay(1500);
}
```



Lab#7 Use of Breadboard

Breadboard Usage Note

A breadboard is a device for creating temporary electronic circuits without soldering. It is an essential tool for prototyping and testing electronic circuits quickly and easily.

Key Features

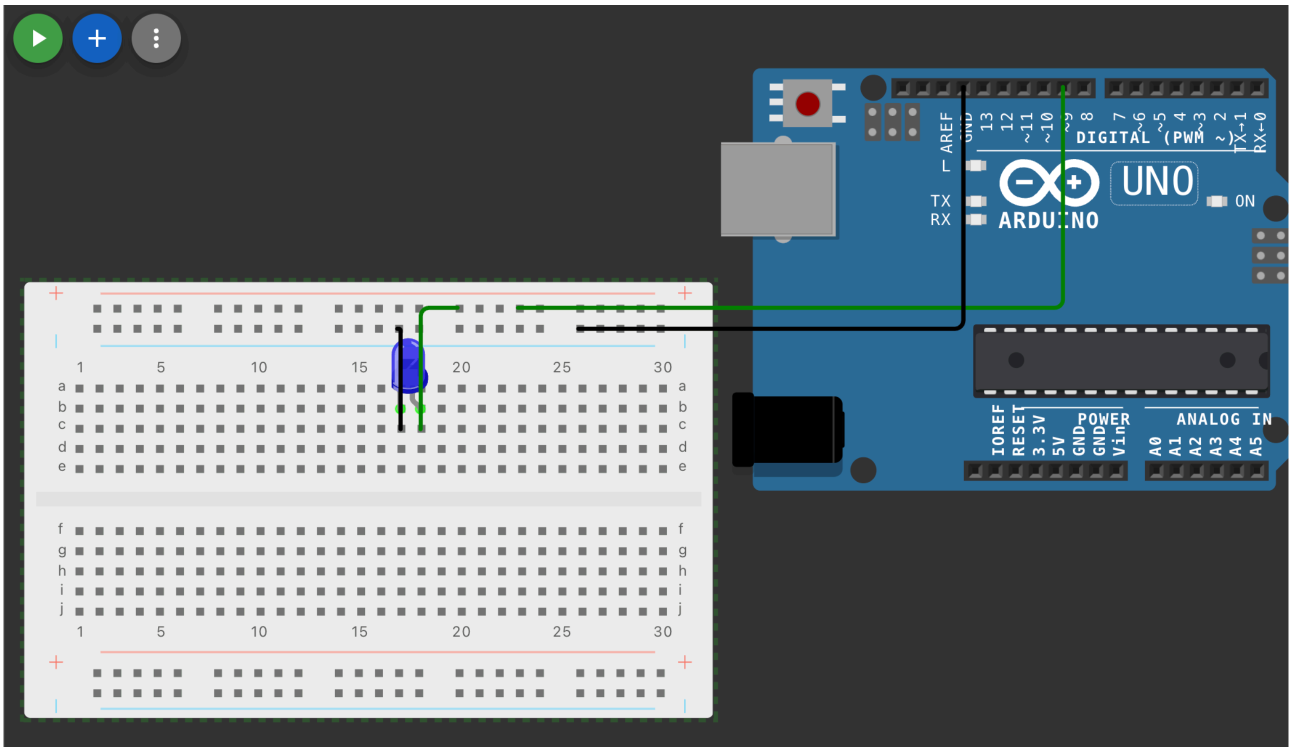
1. **Rows and Columns**: Breadboards have rows (numbered) and columns (lettered). The rows in the middle are typically connected horizontally, while the side rows (power rails) run vertically and are useful for connecting power supplies.
2. **Power Rails**: These are long rows along the edges, usually marked with a red (+) and blue (-) line, used for distributing power across the board.

HOW to use a Breadboard

1. **Insert Components**: Push the leads of components into the holes.
2. **Make Connections**: Use jumper wires to connect components.
3. **Power Connections**: Connect your power source to the power rails first to distribute power across the board easily.
4. **Avoiding Overload**: Breadboards can handle only low current and low power. High currents can damage the contacts, so avoid overloading.

Tips

- **Plan the Layout**: Organize the layout of components and wires to minimize clutter.
- **Check Connections**: Ensure each component and wire is fully inserted.



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 .

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);
}
```

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);
}
```