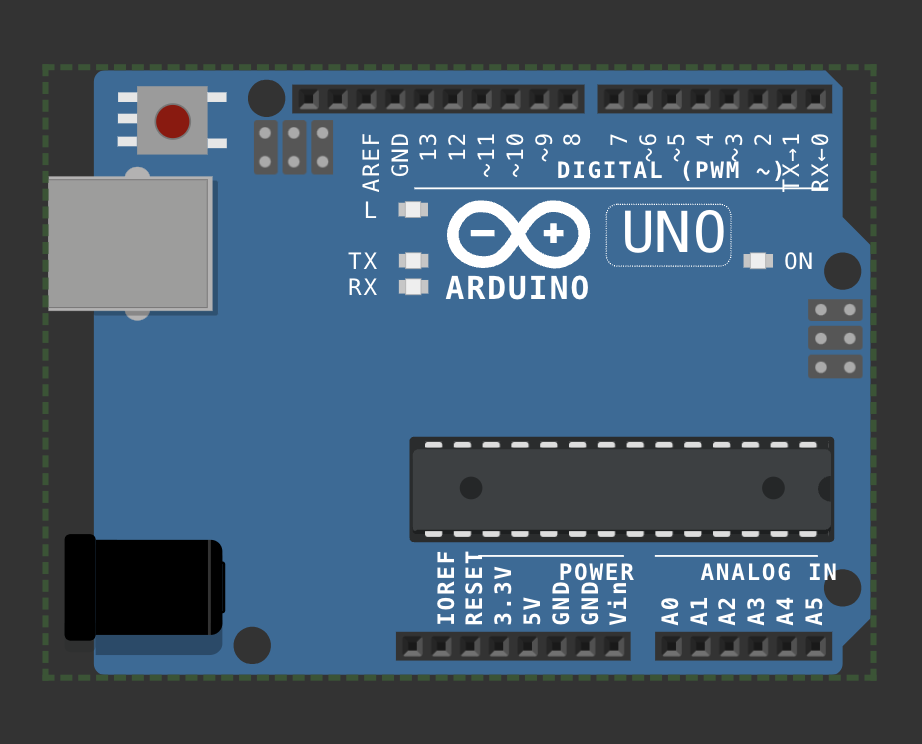
IoT & Automation Lab. Record

LAB 1 : Blinking the inBuilt LED



void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000);

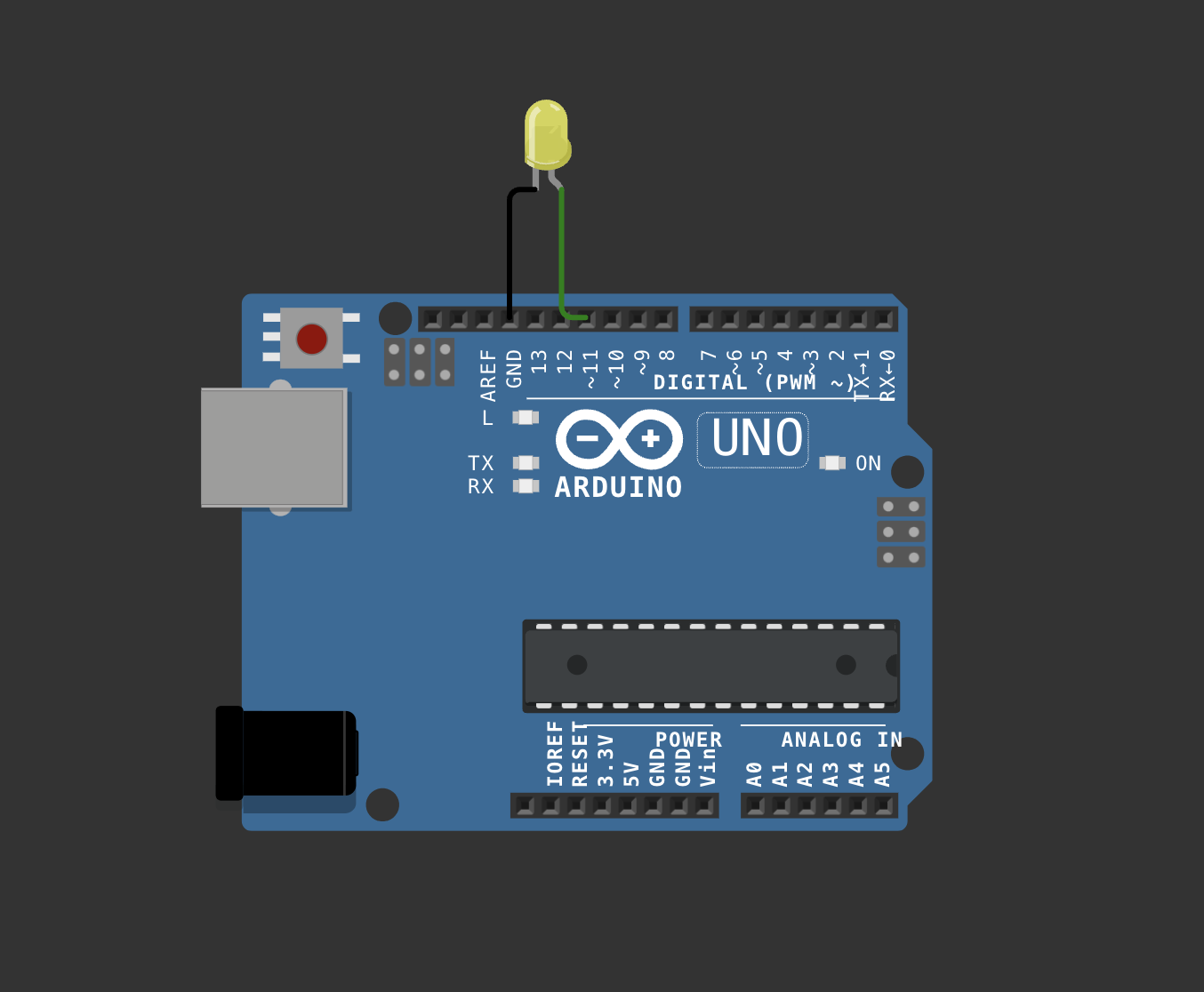
digitalWrite(LED\_BUILTIN,LOW);

delay(1000);

}

[#](http://www.apple.com/uk) [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);

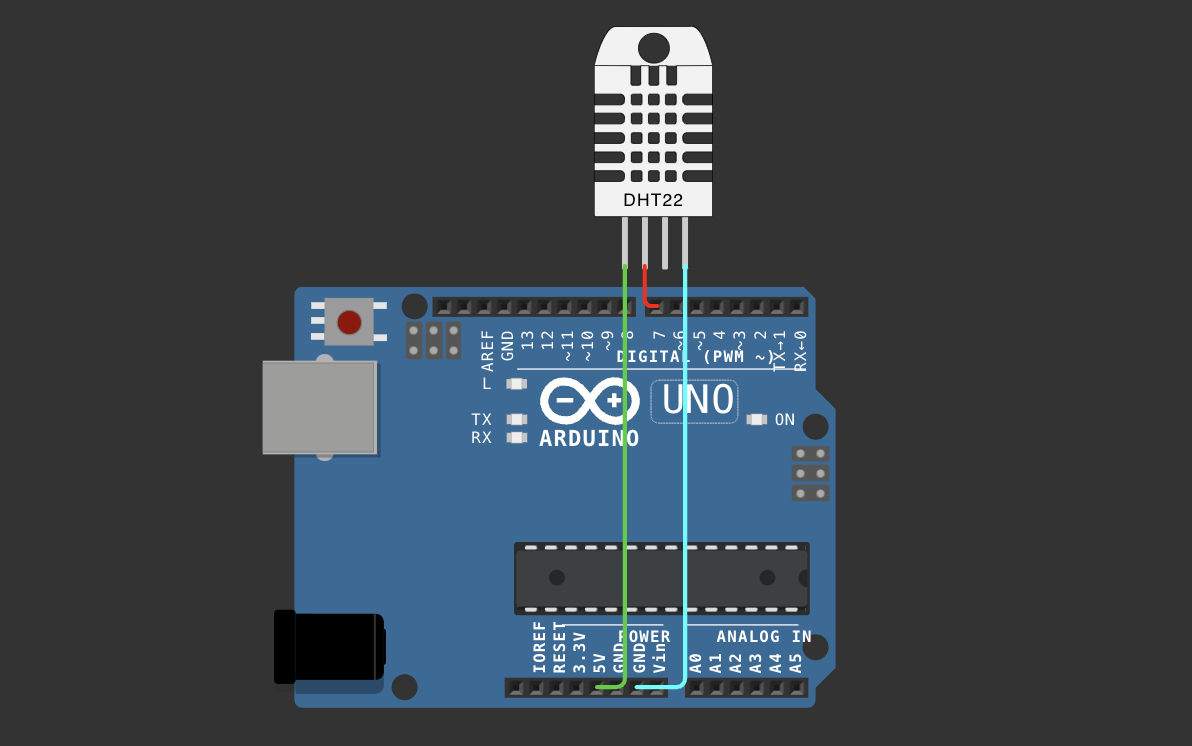
digitalWrite(led\_pin,LOW);

delay(1000);

}

[## 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;

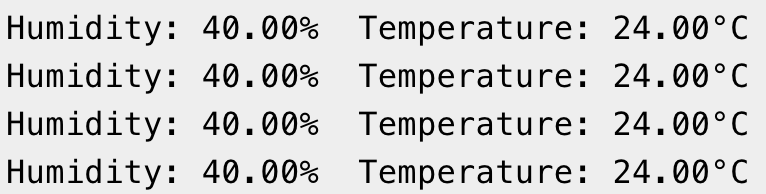
void setup() {

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);

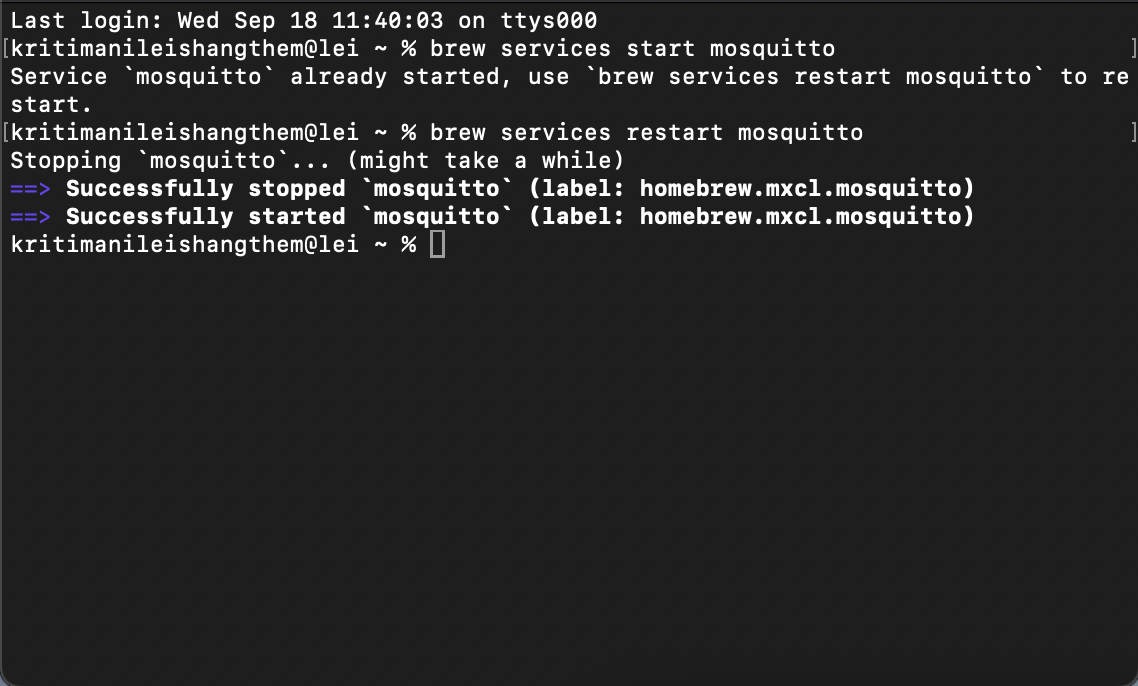
Serial.println("°C");

delay(1000);

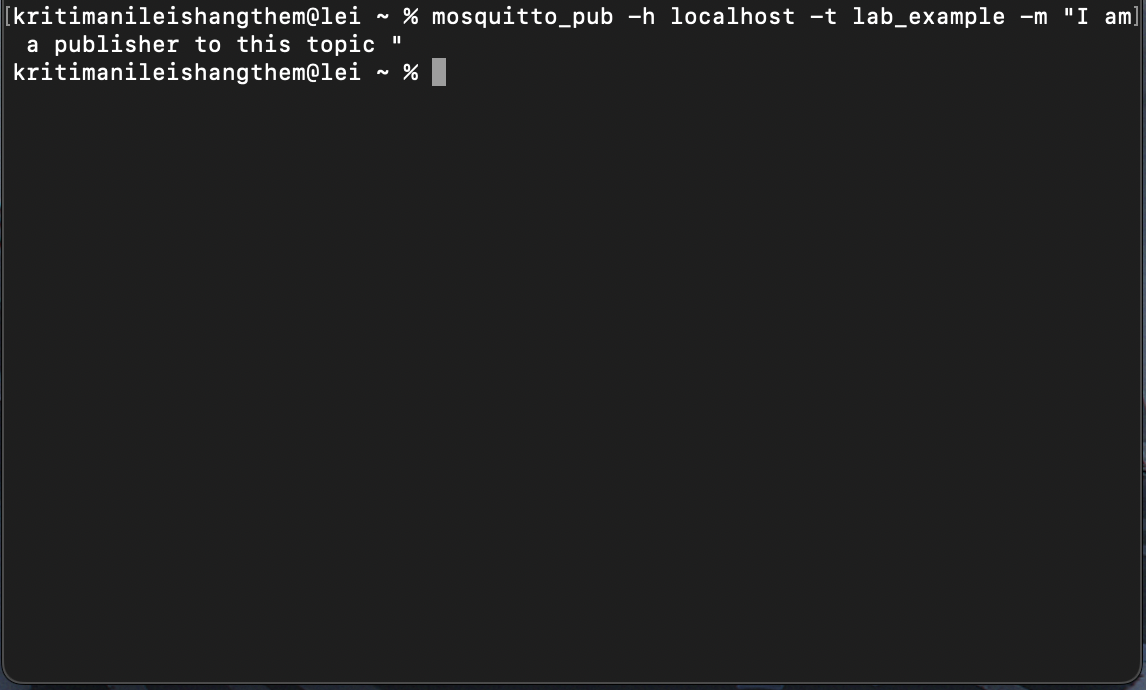
}

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

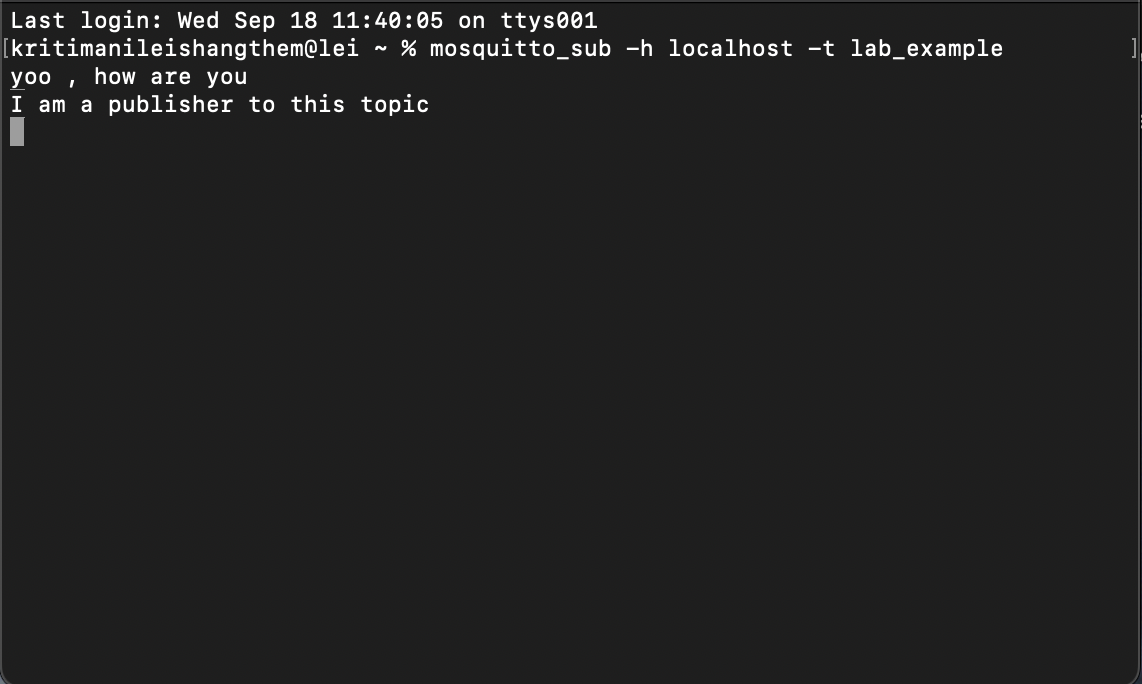
Lab 4: Using Mosquitto MQTT (Pub-Sub):



Starting Mosquitto MQTT:



Setting up Publisher & Sending Message:



Setting up Subscriber & Receiving Message:

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]:

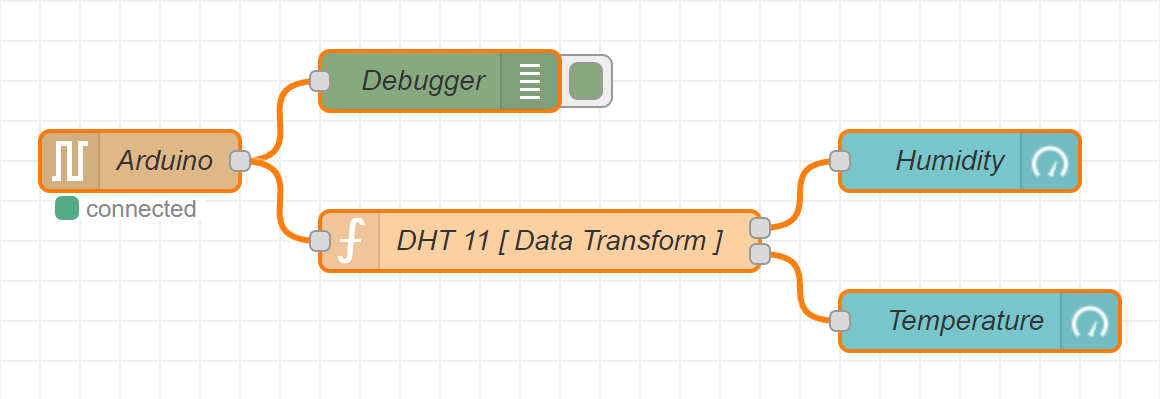
* Inside the nodered window, a flow was created w/ the nodes as:

            > 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));

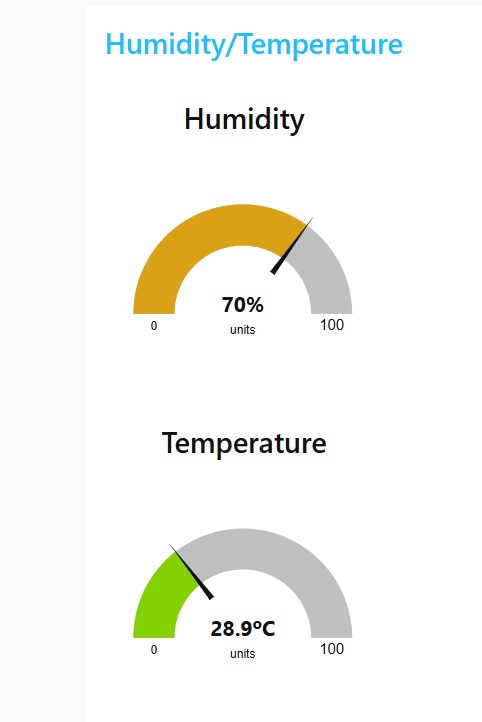
}

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 .

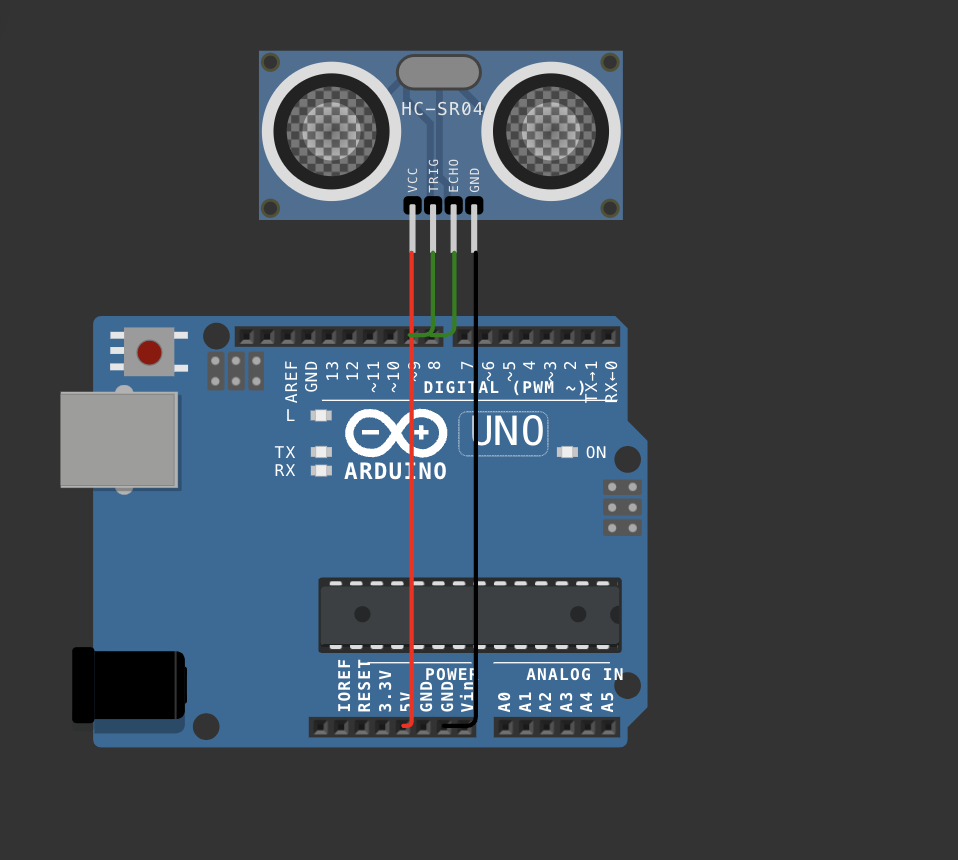
## Measuring distance of a somethings using ultrasonic sensors

# 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

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);

}