**IoT & Automation Lab Record**

**Lab-1**

**Lab Exercise: Blinking the In-Built LED of Arduino Board**

**Objective:** Familiarize with Arduino programming and syntax by blinking the onboard LED.

**Components:** Arduino Uno.

**Procedure:**

1. Open Arduino IDE and connect Arduino to your computer.
2. Write a basic sketch using the pinMode() and digitalWrite() functions to blink the onboard LED on Pin 13.
3. Upload the code and observe the LED.

****void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000);

digitalWrite(LED\_BUILTIN, LOW);

delay(1000);

}

**Expected Outcome:** The onboard LED should blink at specified intervals.

**Observations and Inferences:** Note if LED blinks as intended, adjust timing if needed.

[**Wokwi Lab-1 @ project**](https://wokwi.com/projects/408548736050079745)

**Lab-2**

**Lab Exercise: Blinking an External Red LED**

**Objective:** Interface an external LED with Arduino and control it through a digital output pin.

**Components:** Arduino Uno, external LED, jumper wires.

****#define vk 12

void setup() {

  pinMode(vk, OUTPUT);

}

void loop() {

  digitalWrite(vk, HIGH);

  delay(500);

  digitalWrite(vk, LOW);

  delay(500);

  }

**Procedure:**

1. Connect the external LED to a digital pin.
2. Write a program to blink the external LED.
3. Upload code, observe the LED behaviour.

**Expected Outcome:** LED should blink on and off at specified intervals.

**Observations and Inferences:** Measure and note LED brightness

[**Wokwi Lab-2 @ Sketch**](https://wokwi.com/projects/408550788966095873)

**Lab-3**

**Lab Exercise: Reading & Displaying Humidity and Temperature Data**

**Objective:** Read temperature and humidity data using a DHT11/DHT22 sensor.

**Components:** Arduino Uno, DHT sensor, breadboard, jumper wires.

**Procedure:**

1. Connect the DHT sensor to Arduino.
2. Install the DHT library and write code to display temperature and humidity.
3. Upload the code and observe the serial output.

#include <DHT.h>

#define vk 7

#define DHTTYPE DHT22

DHT dht(vk, 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);

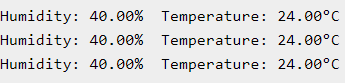
  Serial.println("°C");

  delay(1000);

}

**Expected Outcome:** Display accurate temperature and humidity data.

**Observations and Inferences:** Record the readings and note any variations in data.



[**wokwi Lab-3 @ Sketch**](https://wokwi.com/projects/408552551823923201)

**Lab-4**

**Lab Exercise: Configuring MQTT in our Local Machine**

**Objective:** Explore the Mosquitto MQTT protocol and understand its functionalities.

**In SystemOS [ Windows11 ]:**

**Install Mosquitto:**

* Download and install Mosquitto as a service from the [official Eclipse page](https://mosquitto.org/download/).
* This allows the broker to start automatically in the background at boot.

**Add Mosquitto to System PATH:**

* Go to System Environment Variables and add the path C:\Program Files\mosquitto to allow running Mosquitto commands directly from the Command Prompt or Terminal.

**Control Mosquitto Service:**

* To start Mosquitto at boot: Use net start mosquitto.
* To stop Mosquitto manually in an elevated Command Prompt: Use net stop mosquitto.

**Using Mosquitto in Command Prompt:**

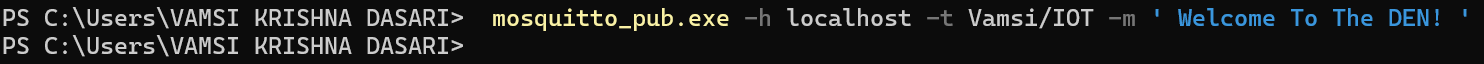
* Open Command Prompt and navigate to Mosquitto’s installation directory:

cd "C:\Program Files\mosquitto"

**Run Mosquitto in Verbose Mode** to view detailed logs and backend processes for debugging:

mosquitto.exe -v

This setup allows you to control the Mosquitto broker and see real-time logs, which is helpful for debugging and monitoring.





**Procedure:**

1. Read about the Mosquitto MQTT protocol.
2. Document its components (broker, client), setup, and messaging.
3. Create a basic publisher-subscriber model using Mosquitto.

**Expected Outcome:** Knowledge of MQTT fundamentals.

**Observations and Inferences:** Note how Mosquitto is used for IoT communications.

**Lab-5**

**Lab Exercise: Node-Red Web App for DHT Data**

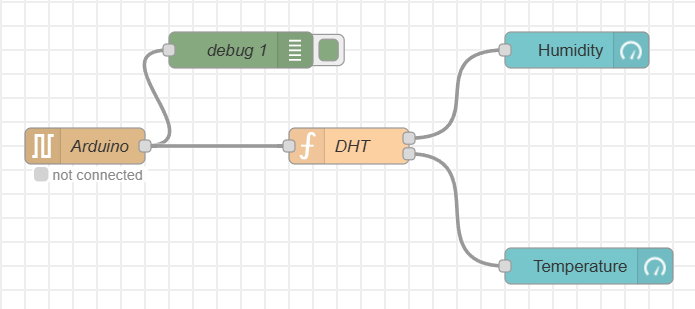
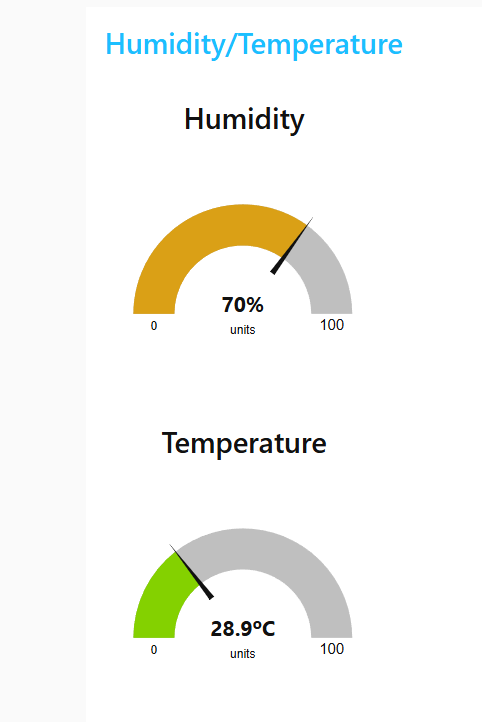
**Objective:** Build a web dashboard in Node-Red to display DHT sensor data.

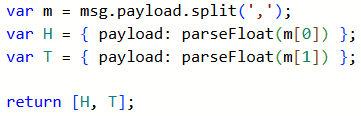
**Components:** Node-Red environment, Arduino with DHT sensor.

**Procedure:**

1. Set up Node-Red and connect Arduino with DHT data.
2. Use the Node-Red dashboard nodes to display data.
3. Test and document the data on the web interface.

**Expected Outcome:** Real-time temperature and humidity data on a web dashboard.

**Observations and Inferences:** Document data refresh rate and layout effectiveness.



**Serial In Node**: Configure it to read from the correct serial port where your Arduino is connected (e.g., COM5). Set the baud rate to 9600.

* + Add the node and name it "Arduino".

**Function Node**: Add a function node and insert the following code to process the incoming serial data:

**Debug Node**: Connect it to the output of the serial node to check if data is flowing correctly.

**Gauge Nodes**:

Add two gauge nodes, one for **humidity** and another for **temperature**.

* + **Humidity Gauge**:
    - Set the title to "Humidity".
    - Set the value format to {{value}}%.
    - Set the minimum value to 0 and the maximum to 100.
  + **Temperature Gauge**:
    - Set the title to "Temperature".
    - Set the value format to {{value}}ºC.
    - Set the minimum value to 0 and the maximum to 100.

The flow is structured as follows:

1. Serial In -> Function Node -> Humidity Gauge
2. Serial In -> Function Node -> Temperature Gauge

**Deploy and Test**

1. **Deploy the flow** in Node-RED.
2. Open the Node-RED dashboard (usually available at http://<your-IP>:1880/ui).
3. **Check the Gauges**: You should see live humidity and temperature readings from the DHT11/22 sensor displayed in the dashboard.

**Global Variables and Objects:**

DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

1. **DHT**: This is the name of the class provided by the library. It's a blueprint for creating objects that represent DHT sensors in your code.
2. **dht**: This is the name we're giving to the instance of the **DHT** class you're creating. It's like a variable that you'll use to interact with the DHT sensor.
3. **(DHTPIN, DHTTYPE)**: These are arguments we're passing to the constructor of the **DHT** class.
   1. **DHTPIN** is a constant with the value **,** which tells the library which pin on the Arduino the data line of the sensor is connected to.
   2. **DHTTYPE** is another constant with the value **DHT11/22**, which tells the library which type of DHT sensor you're using (the library also supports the DHT22 sensor).

**Lab-6**

**Lab Exercise: HC-SR04 Ultrasonic Sensor**

**Objective:** Measure distance using an ultrasonic sensor.

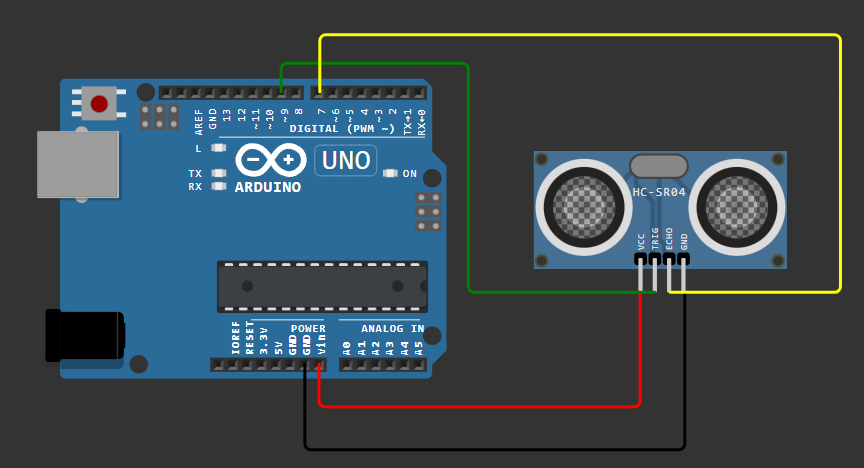
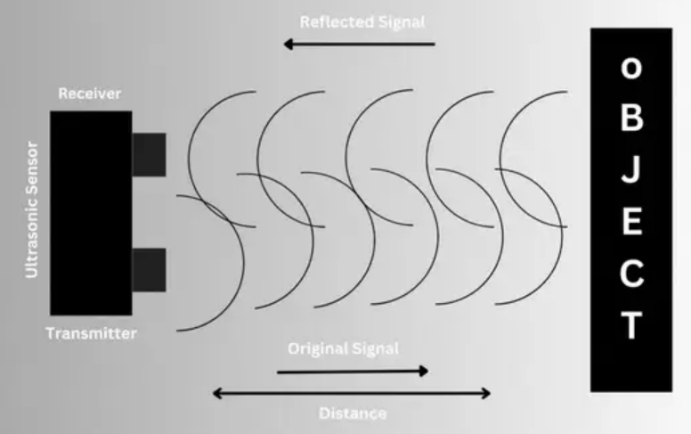
**Components:** Arduino, ultrasonic sensor, breadboard, jumper wires.

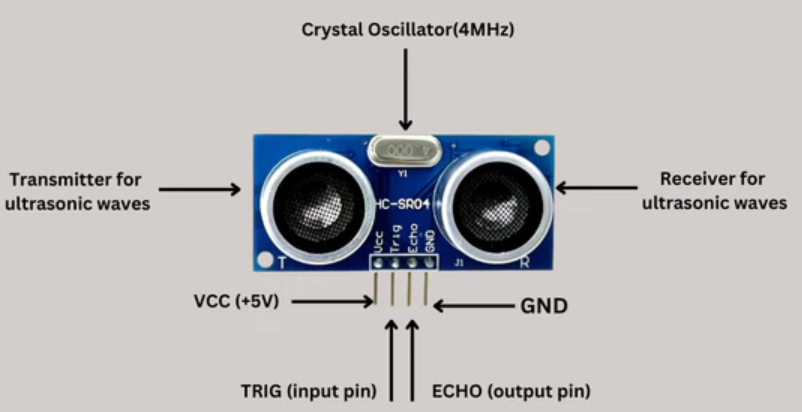
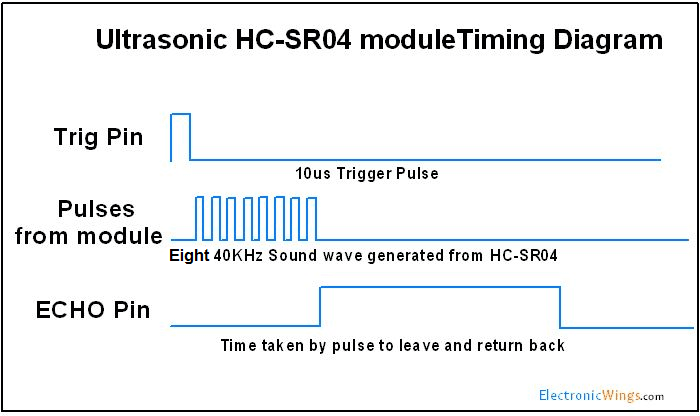
**Procedure:**

1. Connect the ultrasonic sensor to Arduino.
2. Write code to calculate and display distance based on echo time.
3. Upload code and test for different objects and distances.

**Expected Outcome:** Accurate distance measurements on the serial monitor.

**Observations and Inferences:** Note sensor accuracy and limitations at varying distances.



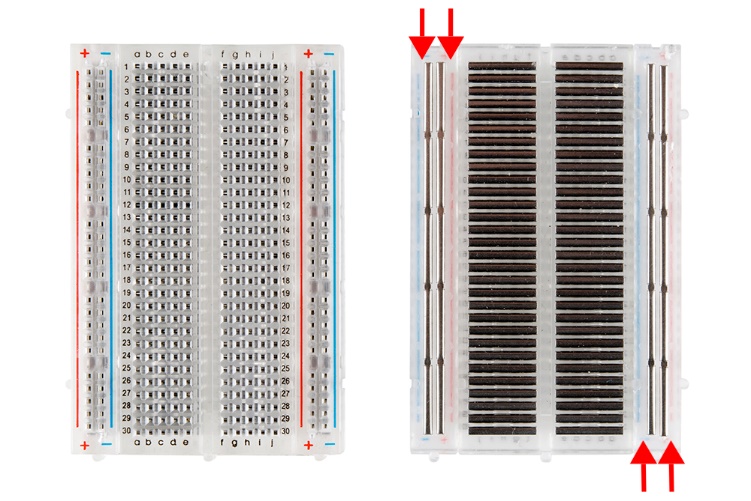


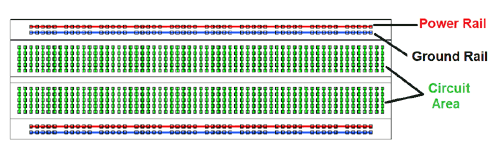
[**Wokwi Lab-6 @ Sketch**](https://wokwi.com/projects/413102749629529089)

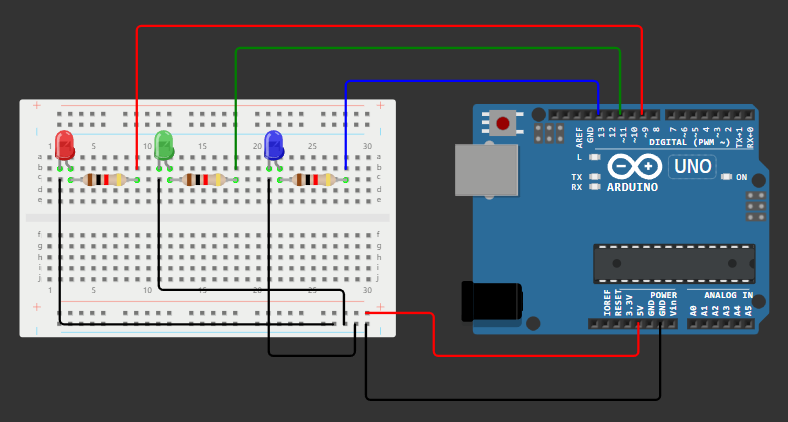
**Lab-7**

**Lab Exercise: Use of Breadboard**

**Objective:** Learn basic breadboard interfacing and circuit assembly.

**Components:** Breadboard, jumper wires, resistors, LEDs.





In this project, we did the process to blink three LEDs using for loop. The three LEDs will light up one after the other.

**Expected Outcome:** Familiarity with breadboard layout and component arrangement.

[**Wokwi Lab-7 @ Sketch**](https://wokwi.com/projects/413105854597684225)

**Lab-8**

**Lab Exercise: ESP32 for Onboard LED Blink**

**Objective:** Use the ESP32 microcontroller to blink its onboard LED.

**Components:** ESP32 board, Arduino IDE.

**Procedure:**

1. Install ESP32 Support in Arduino IDE:

* Go to File > Preferences in the Arduino IDE.
* In Additional Boards Manager URLs, paste: **https://dl.espressif.com/dl/package\_esp32\_index.json** and click OK.
* Open Boards Manager (Tools > Board > Boards Manager) and search for **ESP32 by Espressif Systems**. Click Install.

2. Install USB Driver:

* Download and install the **CP210x USB-to-UART Bridge VCP Driver** to enable communication between your computer and the ESP32 board.

3. Select Board and Port in Arduino IDE:

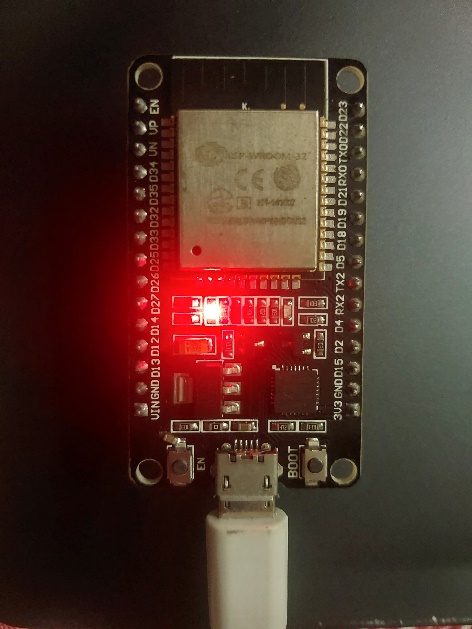
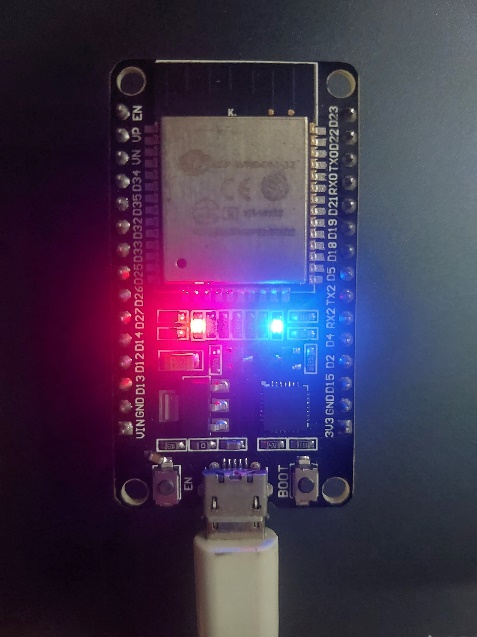
* After restarting, select ESP32 Dev Module from the Tools > Board menu.
* Choose the correct COM port for your ESP32 (e.g., COM7).

4. Put ESP32 in Boot Mode:

* Press and hold the BOOT button while uploading the code.
* Release the BOOT button once the upload begins (when you see "Connecting...").

5. Code Execution for LED Blink:

* Use the following code in the Arduino IDE to blink the onboard LED.

**Expected Outcome:** ESP32 onboard LED blinks at specified intervals.