**INTERNET OF THINGS Lab. Record**

**Subject Code:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| Name: | Gayathri Reddy Epur |
| Registration Number: | FET-BAML-2022-26-018 |
| Course: | IOT |
| Semester: | 5th |
| Faculty: | Biswajeeban Mishra  Pritam Nanda |

|  |  |
| --- | --- |
| Remarks |  |
| Signature |  |



SRI SRI UNIVERSITY

Bidyadharpur, Cuttack, Odisha.

Index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Date** | **Experiment/Case Study** | **Page No.** | **Remark** |
| **1** | 14-8-2024 | Blink Inbuilt Led of your Arduino Board |  |  |
| **2** | 22-8-2024 | Blink an external LED |  |  |
| **3** | 29-8-2024 | Using DHT sensor |  |  |
| **4** | 12-9-2024 | A case study on  a communication Protocol: MQTT |  |  |
| **5** | 19-9-2024 | Building a web app using Node-Red to fetch DHT sensor data and display it on the web app dashboard |  |  |
| **6** | 26-9-2024 | Interfacing Ultrasonic Sensor |  |  |
| **7** | 3-10-2024 | Use of Breadboard |  |  |
| **8** | 17-10-2024 | Use of ESP32, upload code on ESP 32 to blink onboard LED |  |  |
| **9** |  |  |  |  |
| **10** |  |  |  |  |
| **11** |  |  |  |  |
| **12** |  |  |  |  |
| **13** |  |  |  |  |
| **14** |  |  |  |  |
| **15** |  |  |  |  |

***Date: 14-8-2024***

**Experiment No.:1**

Aim:

To learn how to control the built-in LED on an Arduino board using Arduino IDE.

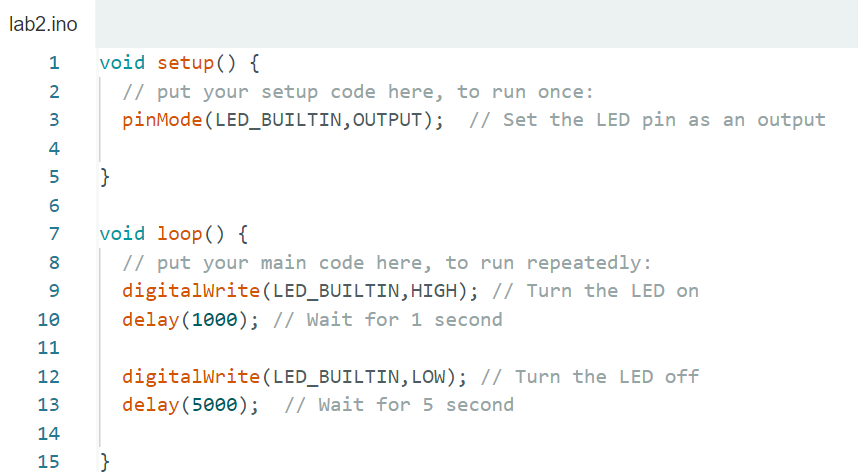
Requirements:

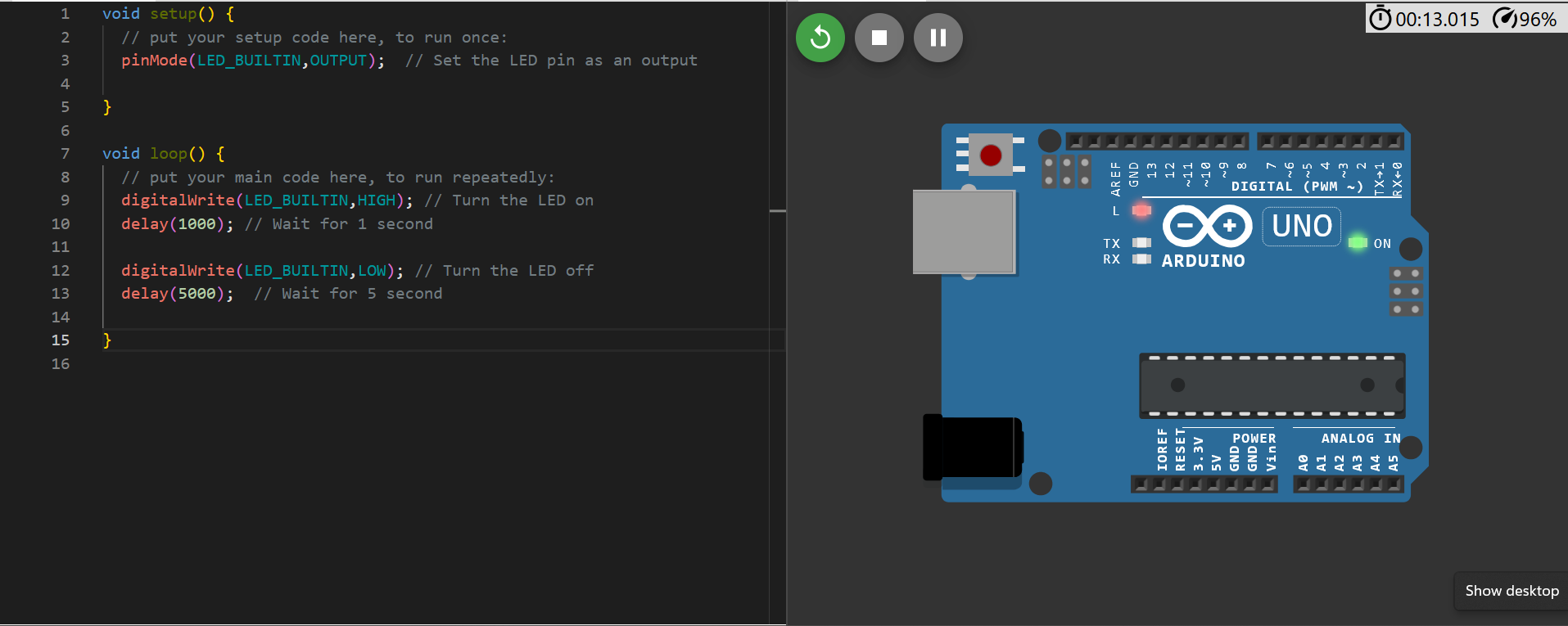
Arduino board (Uno R3), USB cable , Arduino IDE software

Procedure:

1. Connect your Arduino board to your computer using a USB cable.
2. Open the Arduino IDE software on your computer.
3. Write the source code on the software.
4. Connect your Arduino board to your computer.
5. Select the correct board type and port in the Arduino IDE's Tools menu.
6. Click the Upload button in the Arduino IDE to transfer the code to the board.
7. The built-in LED on your Arduino board should now blink on for 1 second and off for 5 second.

Snapshots of the experiment:





***Date: 22-8-2024***

**Experiment No.:2**

Aim:

To learn how to control the built-in LED on an Arduino board using Arduino IDE.

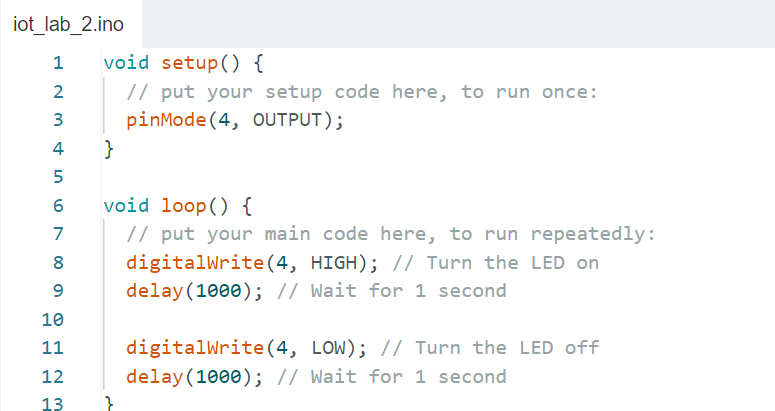
Requirements:

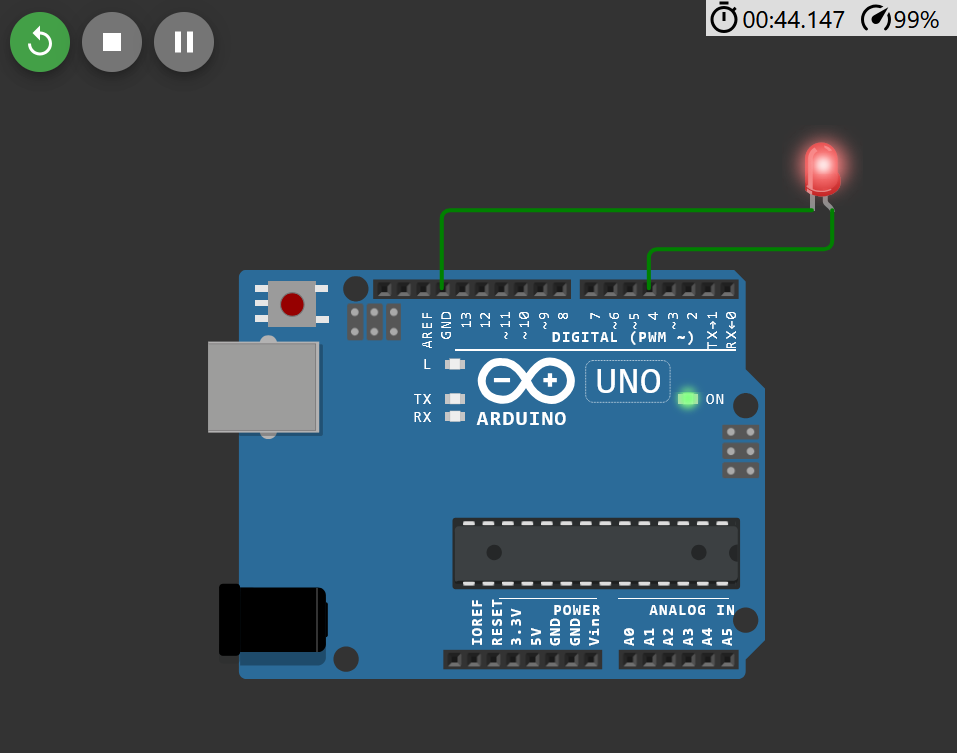
Arduino board (Uno R3), USB cable , Arduino IDE software, External LED (with two leads), Jumper wires

Procedure:

1. Connect your Arduino board to your computer using a USB cable.
2. Open the Arduino IDE software on your computer.
3. Write the source code on the software.
4. Use a jumper wire to connect the anode[+](longer end) of the LED directly to digital pin 4 on the Arduino board.
5. Use another jumper wire to connect the cathode[-](shorter end) of the LED directly to the ground pin on the Arduino board.
6. Select the correct board type and port & click the upload button.
7. The external LED should now blink on and off with a one-second interval.

Snapshots of the experiment:





***Date: 29-8-2024***

**Experiment No.:3**

Aim:

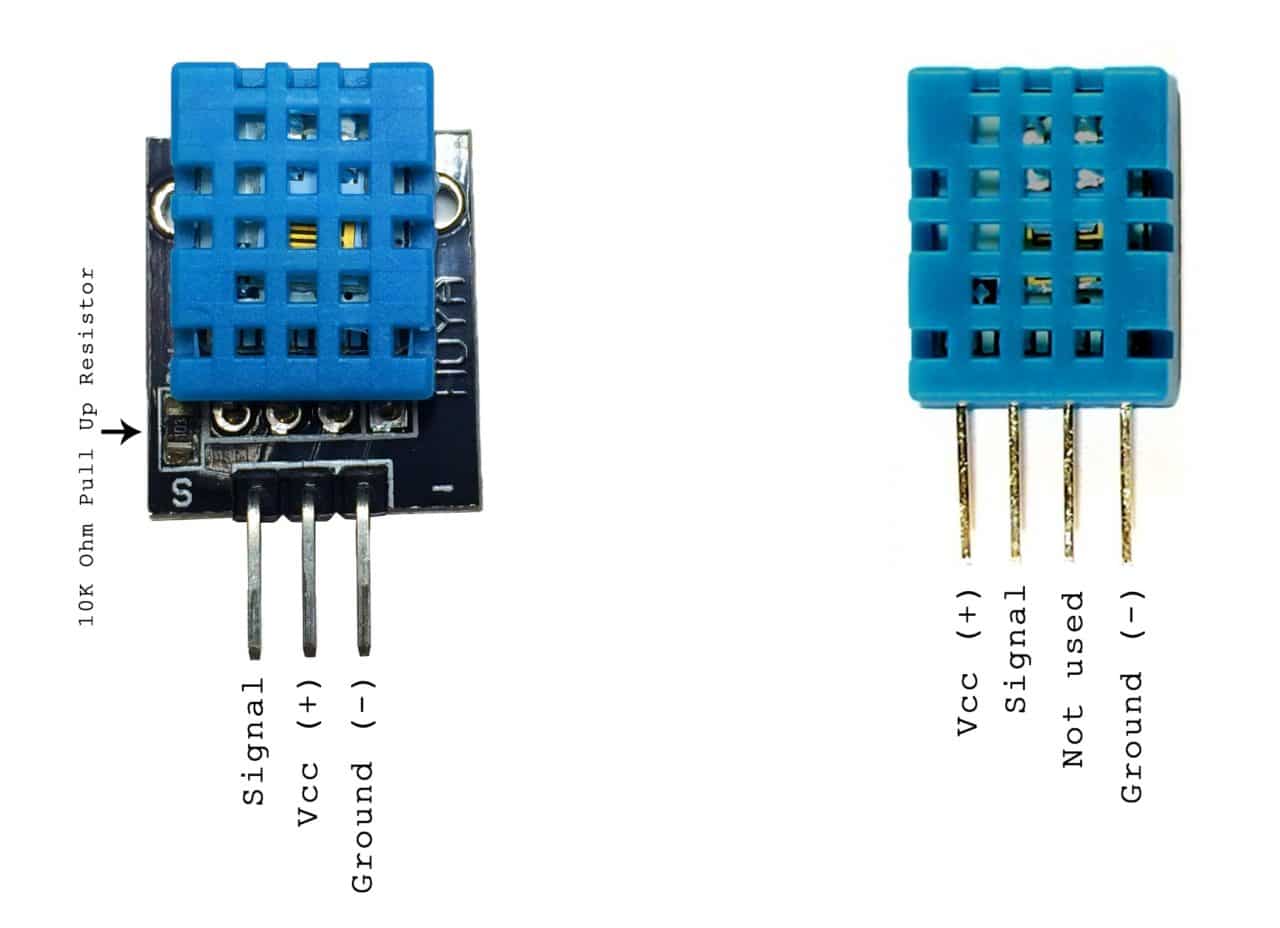
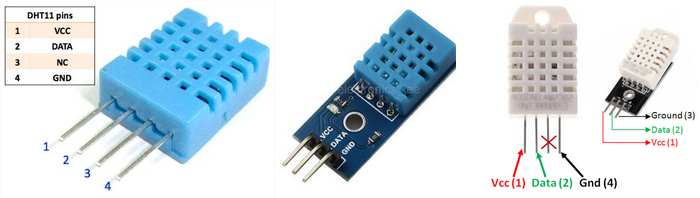
To measure temperature and humidity using a DHT11 or DHT22 sensor with an Arduino board.

Requirements:

Arduino board (Uno R3), USB cable ,Arduino IDE, DHT11 or DHT22 sensor, Jumper wires

Understanding DHT Sensor pins:

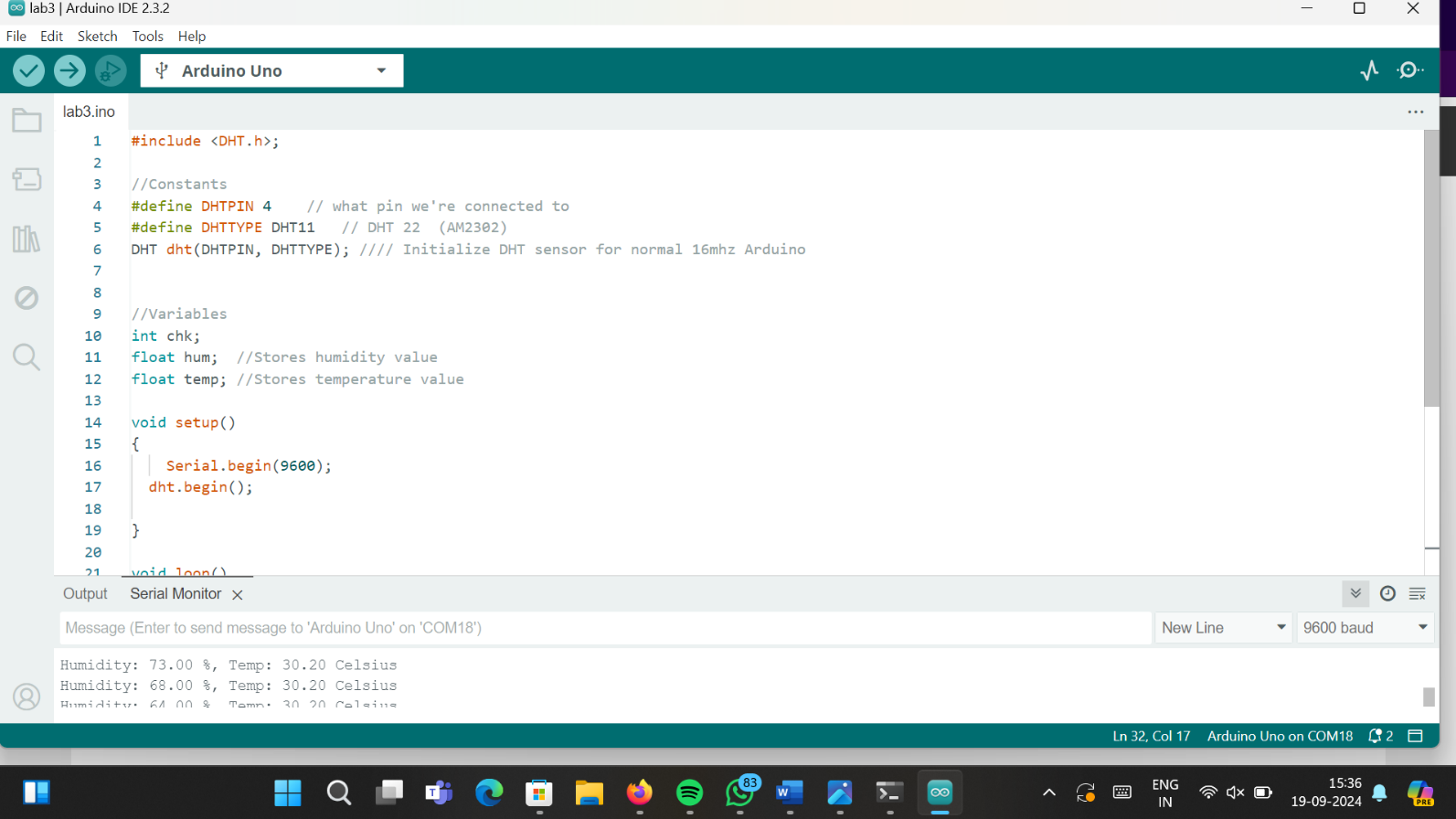
DHT22 Sensor: DHT11 Sensor:

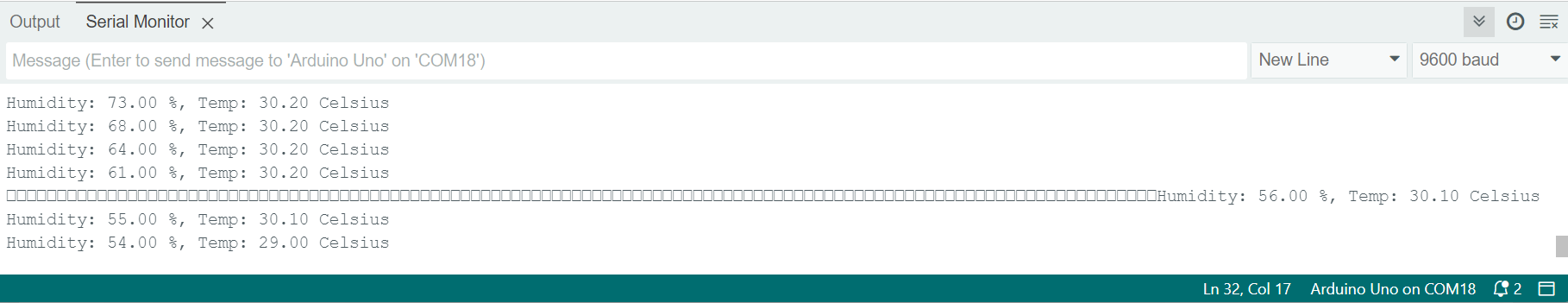


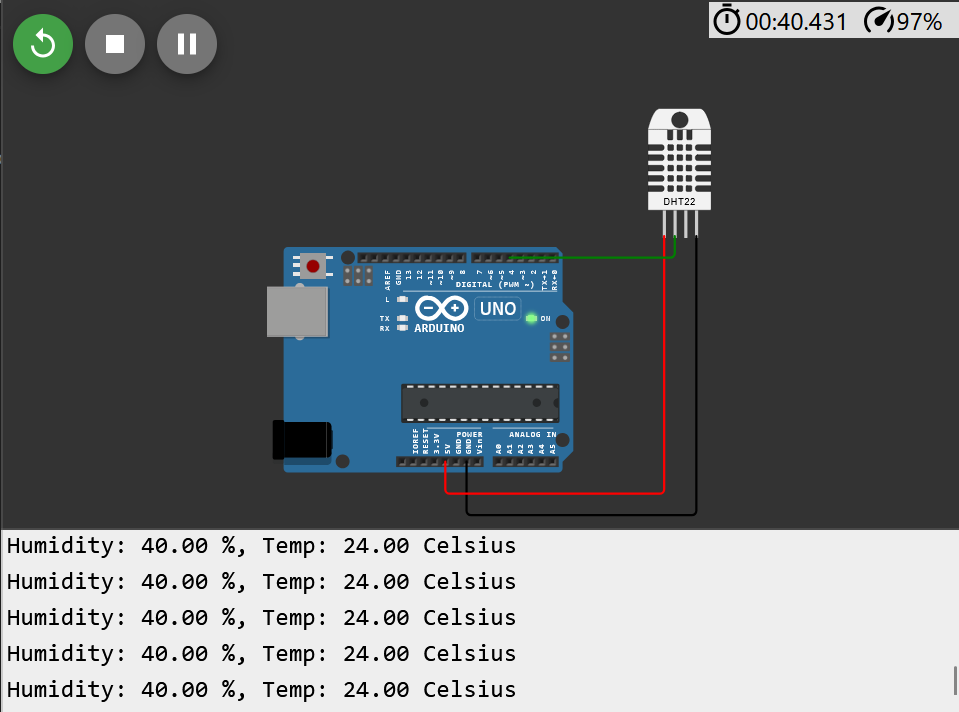
Procedure:

1. Connect your Arduino board to your computer using a USB cable.
2. Search for ‘DHT library’ in the Arduino Library Manager and install the ‘DHT sensor library by Adafruit’.
3. Now write the source code on the software.
4. Compile(click Verify) the code and click the Upload button.
5. Open the Serial Monitor in the Arduino IDE.
6. The temperature and humidity values will be printed periodically.

Snapshots of the experiment:







***Date: 12-9-2024***

**Experiment No.:4**

Aim:

* To understand the MQTT communication protocol.
* To install and configure an MQTT broker (server).
* To write and test MQTT client code to publish and subscribe to messages.

About MQTT:

***MQTT*** (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe messaging protocol designed for constrained devices and low-bandwidth networks. It's particularly well-suited for Internet of Things (IoT) applications.

It’s key components are:

***Topic***: A hierarchical string that categorizes messages.

***Publisher***: A device or application that sends messages to a specific topic.

***Subscriber***: A device or application that receives messages from a specific topic.

***Broker***: A server that facilitates communication between publishers and subscribers.

***Mosquitto*** is a popular open-source MQTT broker that implements the MQTT protocol.

Procedure:

**[Task-A] Installing Mosquitto**

***On Linux:***

1. Install and Ubuntu on the device.
2. Install Mosquitto and Client Packages :

*sudo apt install mosquitto mosquitto-clients*

1. Enable and Start the Mosquitto Service :

*sudo systemctl enable mosquitto sudo*

*systemctl start mosquitto*

***On Windows:***

1. Visit the official Mosquitto download page and download the appropriate version of **Mosquitto** (generally the .exe installer for 64-bit Windows).
2. Complete the installation process
3. Run the following command to start Mosquitto in the foreground:

*mosquitto.exe -v*

The -v flag is for verbose output so that you can see log messages.

**[Task-B] Test Mosquitto**

Open Two Terminal Windows to test MQTT communication.

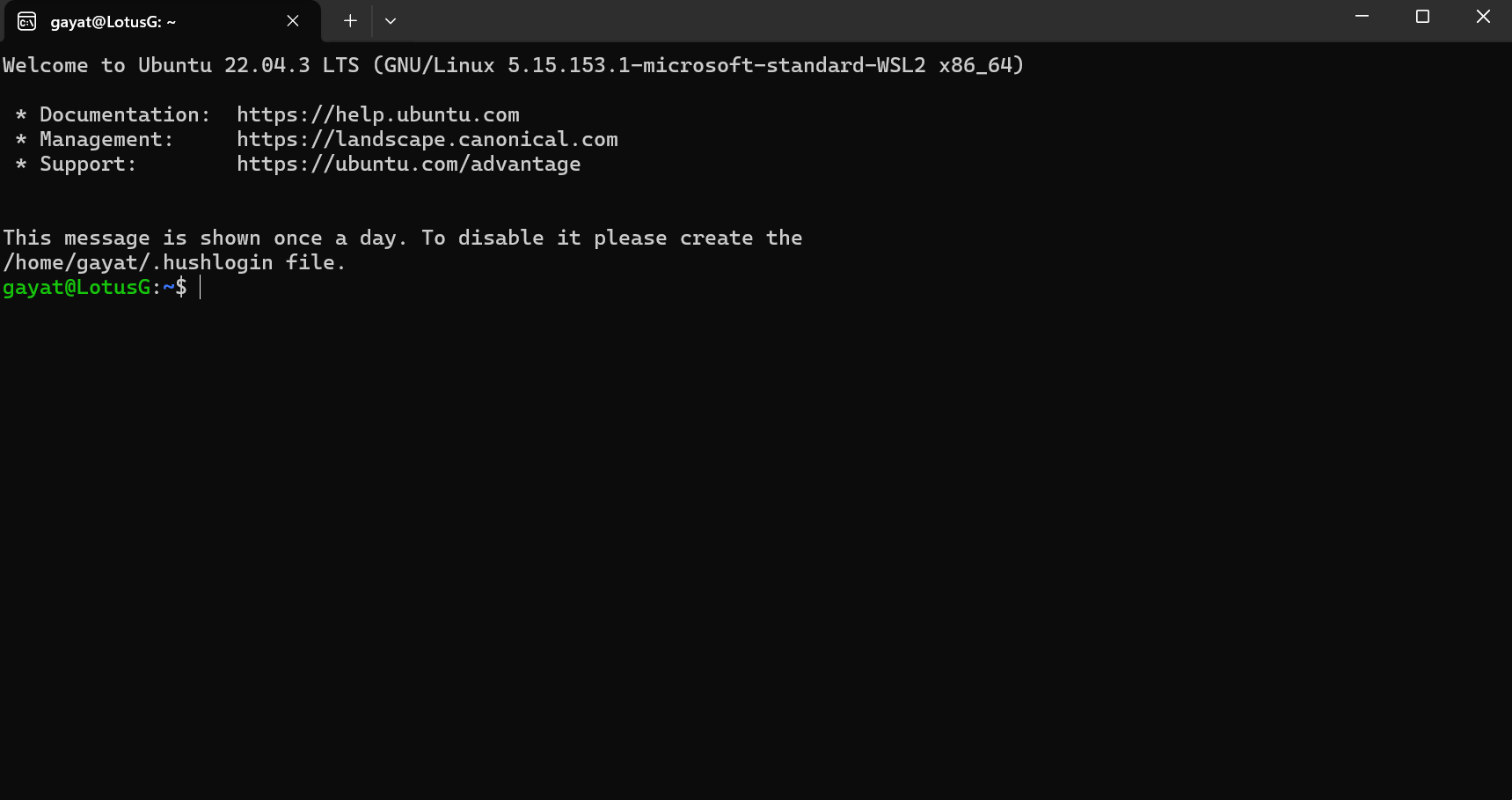
Terminal 1(Subscribing):

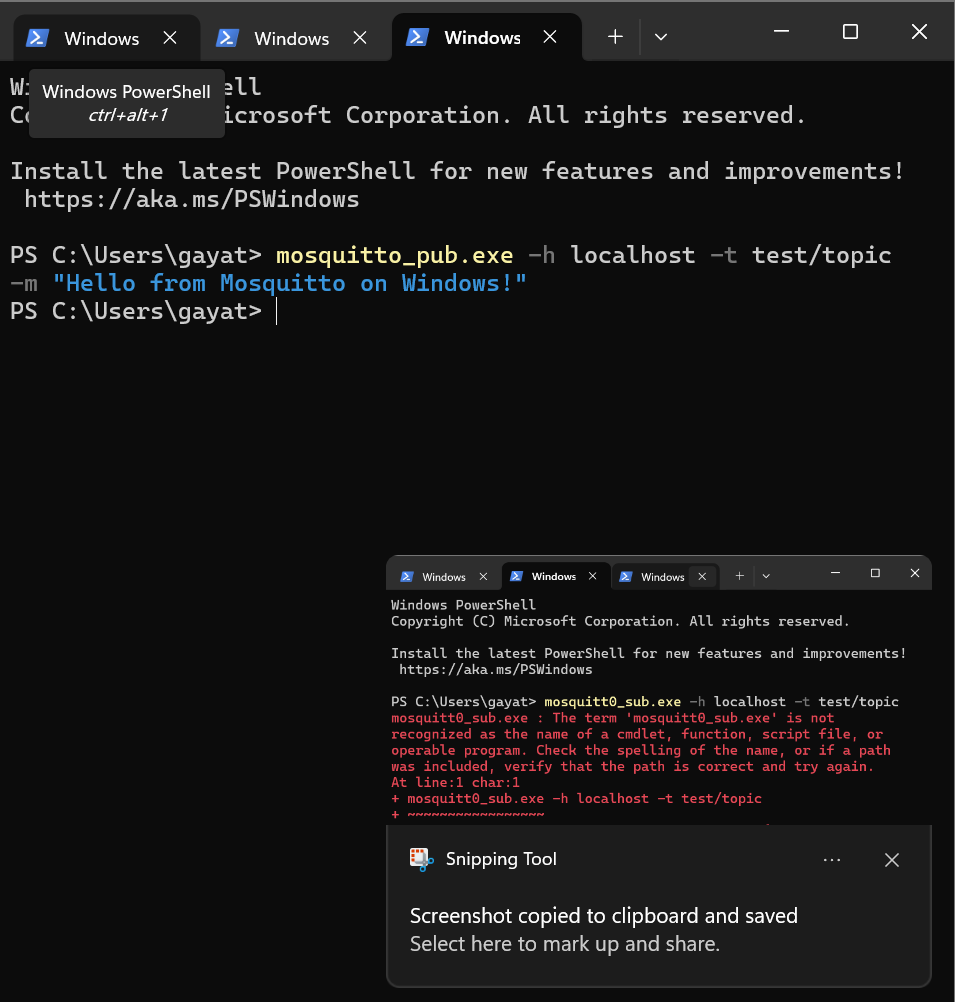
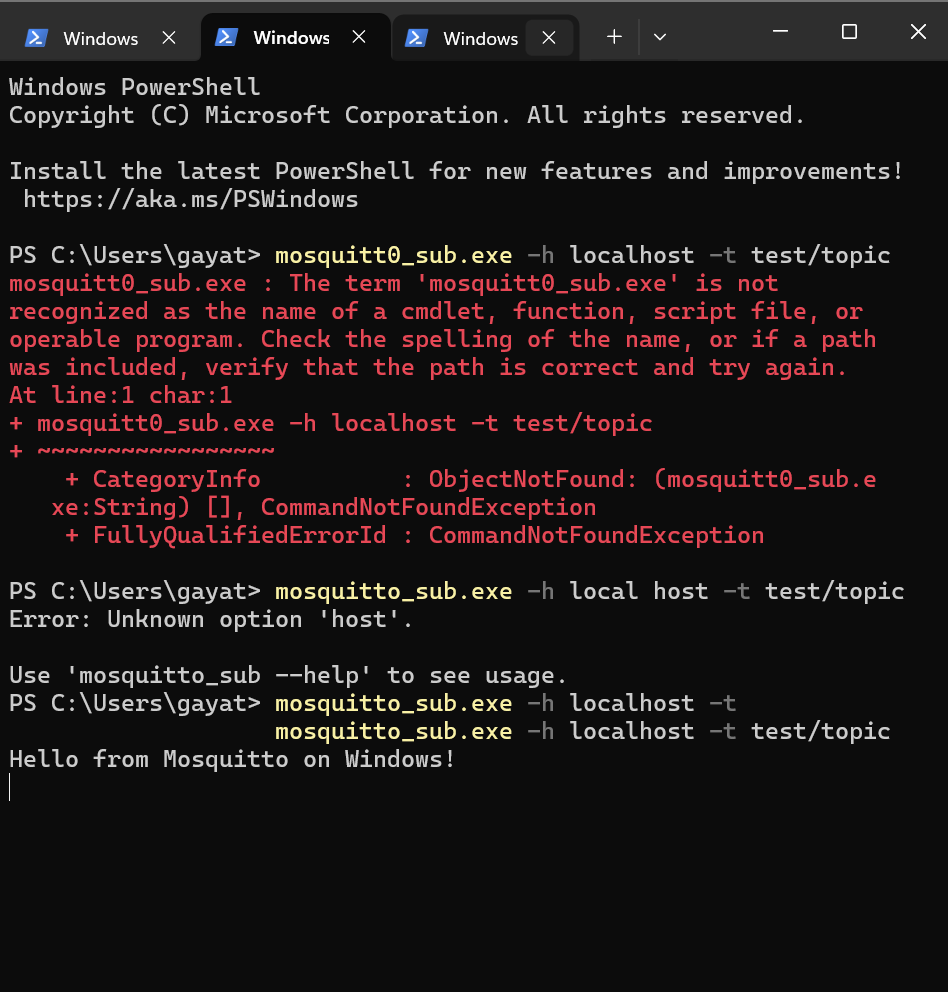
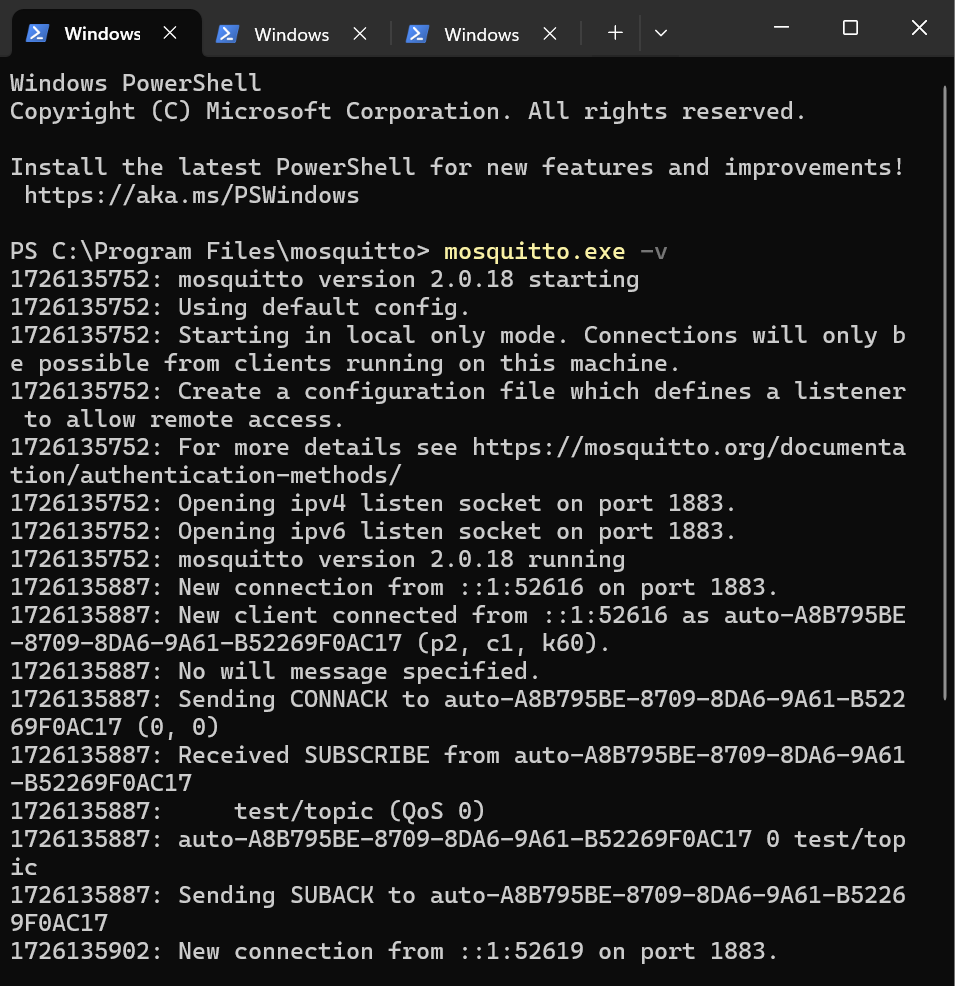
*mosquitto\_sub -h localhost -t test/topic*

Terminal 2(Publising):

*mosquitto\_pub -h localhost -t test/topic -m "Hello from Mosquitto!"*

Snapshots of the experiment:





***Date: 19-8-2024***

**Experiment No.:5**

Aim:

To build a web dashboard in Node-Red to display DHT sensor data.

Requirements:

Arduino board (Uno R3), USB cable, Node-Red environment, DHT11 or DHT22 sensor, Jumper wires

Procedure:

1. Open node.js > npm install node-red-dashboard
2. [postinstallation] > elevated cmd: node-red
3. In client application, browse localhost:1880 to access node red
4. Inside the nodered window, a flow was created w/ the nodes as:
   1. serial-in ( arduino uno r3 board )
   2. debugger
   3. dht function
   4. 2 gauges (humidity& temperature)
5. Configured Serial in node to read from the correct serial port where my Arduino is connected & set the baud rate to 9600.
6. Configure the dht function as:

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

if (m.length === 2) {

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

* 1. var t = { payload: parsefloat(m[1]) };
  2. return [h, t];

} else {

return null;

}

1. Adjusting two gauge nodes:
   1. **Humidity Gauge**:
      1. Set the title to "Humidity".
      2. Set the value format to {{value}}%.
      3. Set the minimum value to 0 and the maximum to 100.
   2. **Temperature Gauge**:
      1. Set the title to "Temperature".
      2. Set the value format to {{value}}ºC.
      3. Set the minimum value to 0 and the maximum to 100.
2. Upload the 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));

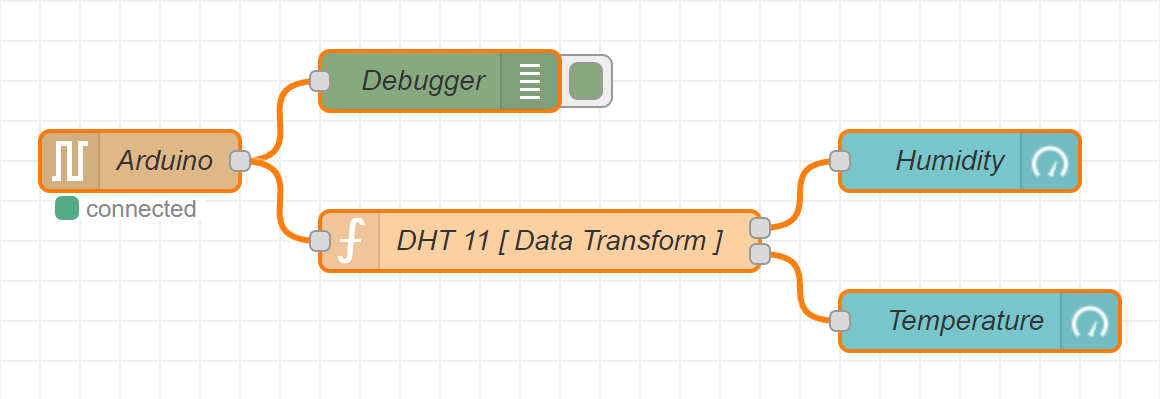
}

delay(2000);

}

1. after uploading this sketch, close the ide.
2. deploy the flow in nodered.
3. check the dashboard in the upper-right corner, for the humidity and temperature gauge.

Snapshots of the experiment:



***Date: 26-9-2024***

**Experiment No.:6**

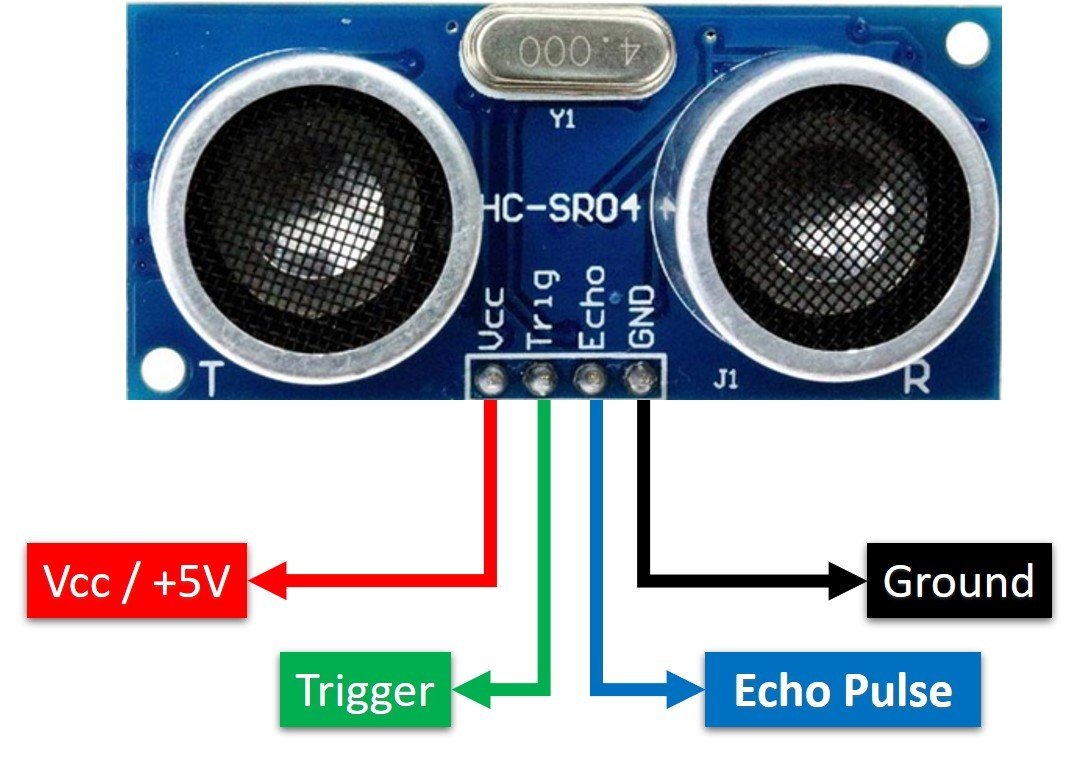
Aim:

To measure distance using an ultrasonic sensor and display the result on the serial monitor.

Requirements:

Arduino board (Uno R3), USB cable ,Arduino IDE, Ultrasonic sensor (HC-SR04), Jumper wires

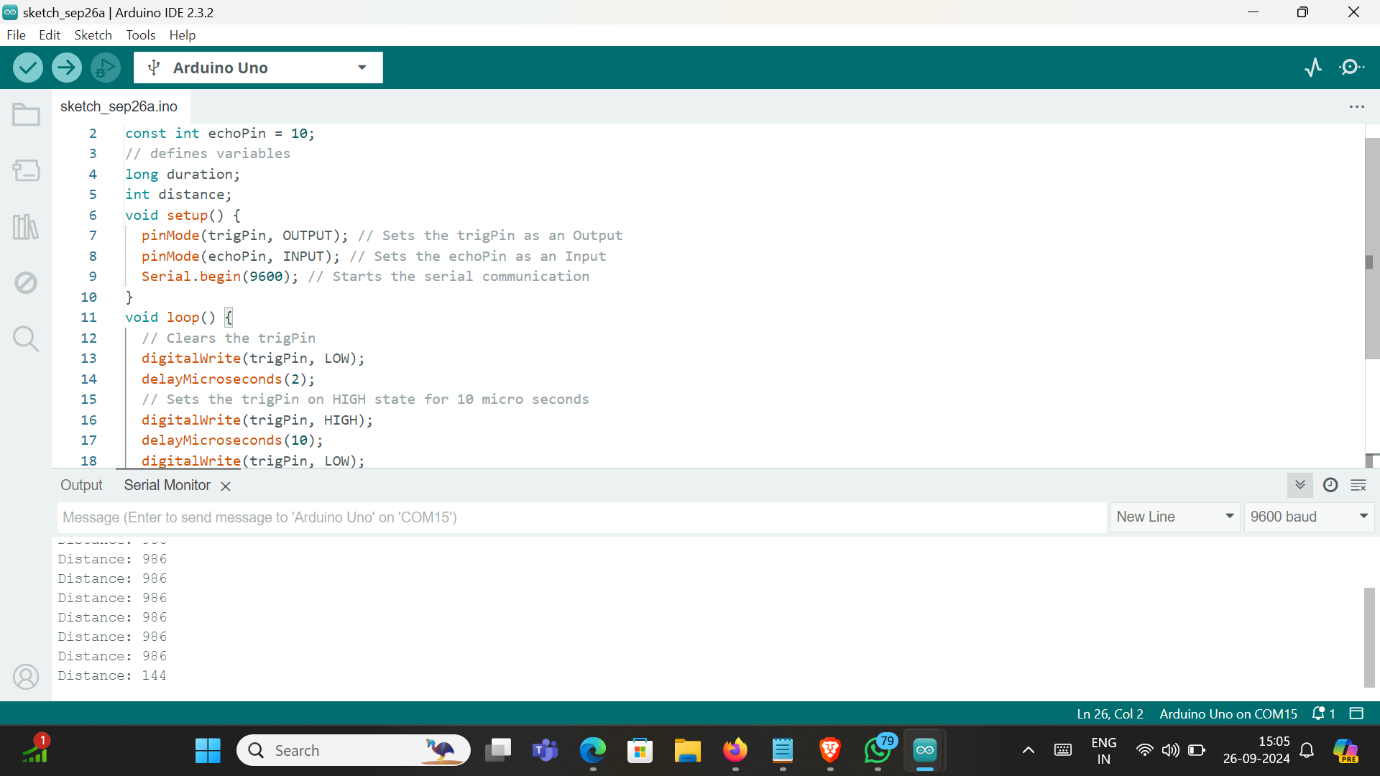
Understanding Ultrasonic sensor (HC-SR04) pins:

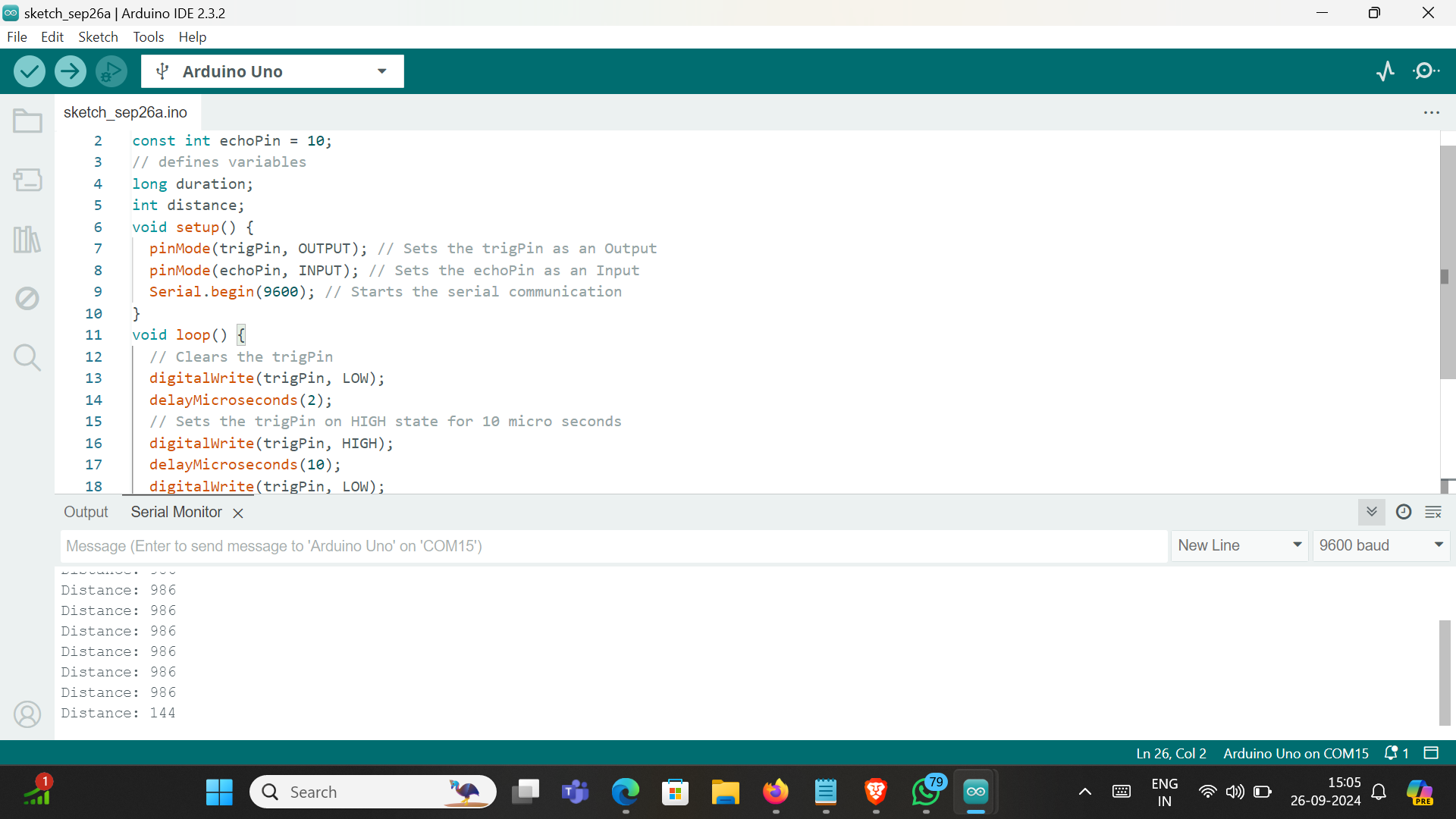


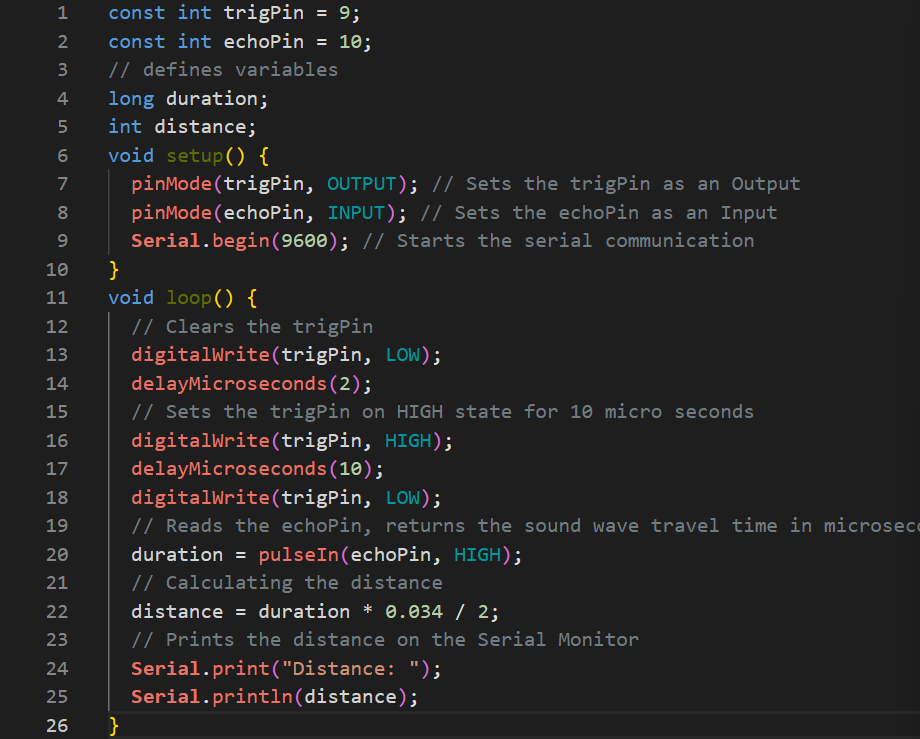
Procedure:

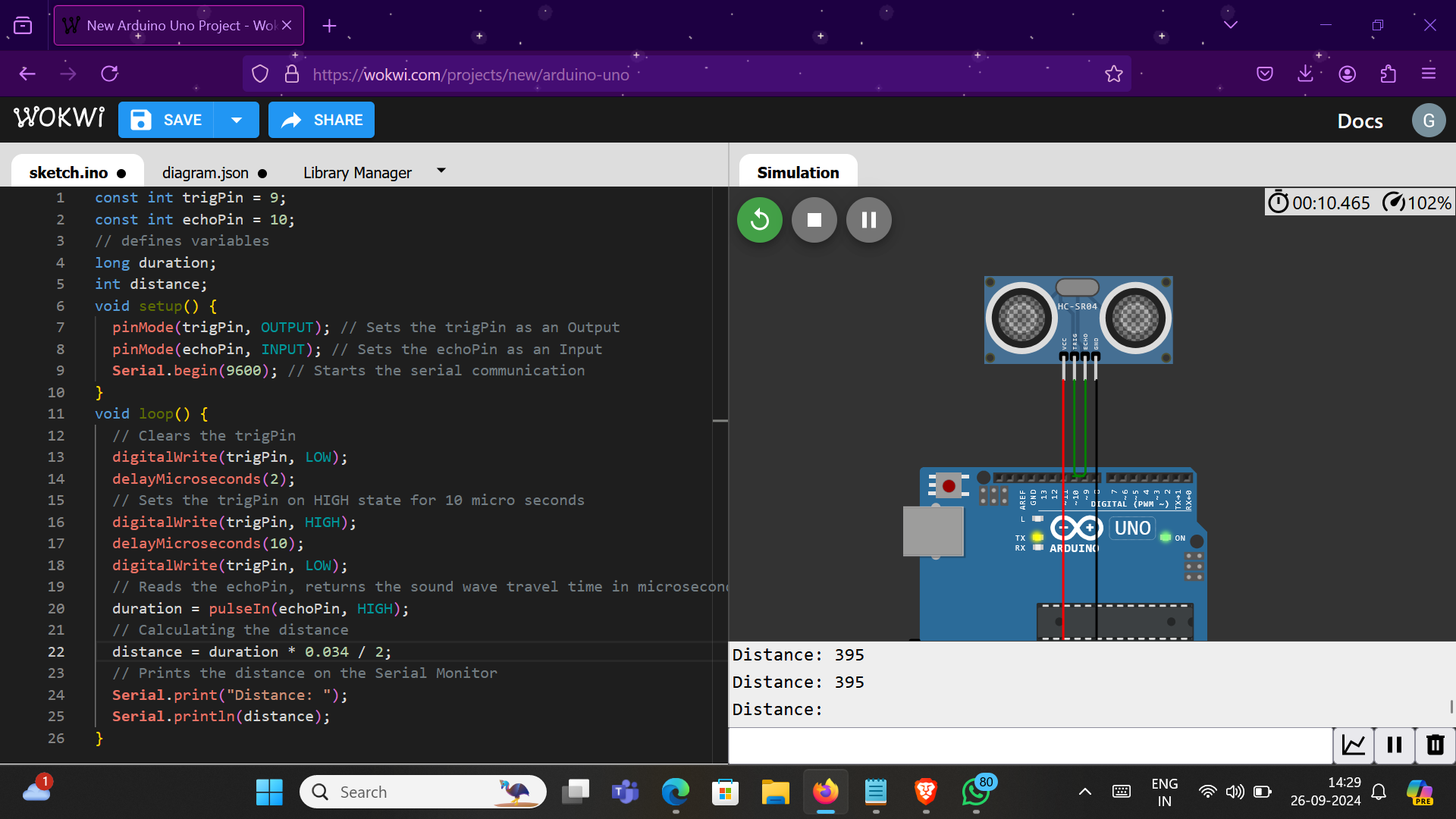
1. Connect your Arduino board to your computer using a USB cable.
2. Connect the sensor:
   1. Connect the Vcc pin to the 5V pin on the Arduino.
   2. Connect the GND pin to the GND pin.
   3. Connect the Trig pin to a digital pin.
   4. Connect the Echo pin to another digital pin.
3. Write the Source code, verify and upload the code to the Arduino board.
4. Open the Serial Monitor
5. The distance measurement in centimeters is printed periodically.

Snapshots of the experiment:





**



***Date:3/10/2024***

**Experiment No.:7**

Aim:

To Learn basic breadboard interfacing and circuit assembly.

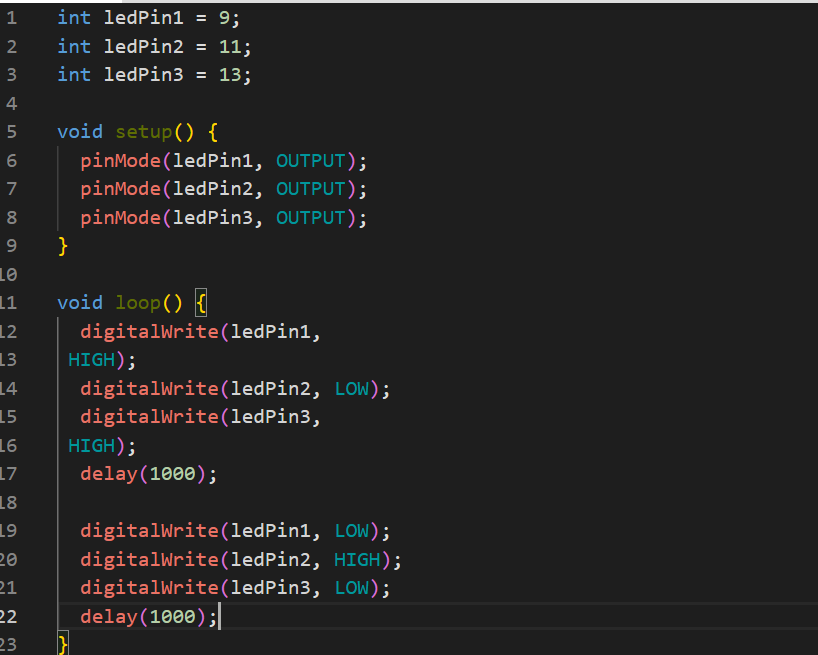
Requirements:

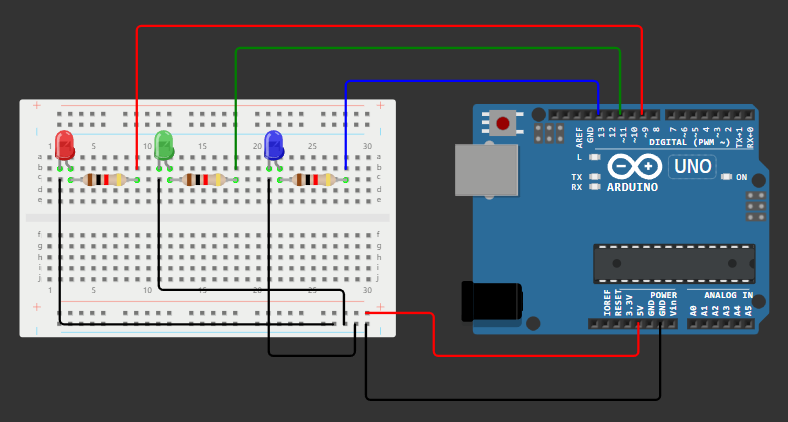
Arduino board (Uno R3), USB cable, Breadboard, jumper wires, resistors, LEDs.

Procedure:

1. Connect one leg of each resistor to the 5V power rail.
2. Connect the other leg of each resistor to the longer leg (anode) of an LED.
3. Connect the shorter leg (cathode) of each LED to a different digital pin on the Arduino (e.g., pins 2, 3, and 4).
4. Connect the other leg of each resistor to the ground rail.
5. Connect the Arduino to the Breadboard:
   1. Power: Connect the 5V pin on the Arduino to the 5V power rail on the breadboard.
   2. Ground: Connect the GND pin on the Arduino to the ground rail on the breadboard.
6. Write the Source Code in Arduino IDE or Wokwi.
7. Connect the Arduino to your computer via USB.
8. Open the Arduino IDE, select the correct board and port, and upload the code.
9. The LEDs should now blink in a specific pattern as defined by the code.

Snapshots of the experiment:





***Date:17/10/2024***

**Experiment No.:8**

Aim:

To Use the ESP32 microcontroller to blink its onboard LED.

Requirements:

ESP32 board, Arduino IDE, USB cable

Procedure:

1. Go to File > Preferences in the Arduino IDE.
2. In Additional Boards Manager URLs, paste: https://dl.espressif.com/dl/package\_esp32\_index.json and click OK.
3. Open Boards Manager (Tools > Board > Boards Manager) and search for ESP32 by Espressif Systems. Click Install.
4. Download and install the CP210x USB-to-UART Bridge VCP Driver to enable communication between your computer and the ESP32 board.
5. After restarting, select ESP32 Dev Module from the Tools > Board menu.
6. Choose the correct COM port for your ESP32 (e.g., COM7).
7. Press and hold the BOOT button while uploading the code.
8. Release the BOOT button once the upload begins (when you see "Connecting...").
9. Use the following code in the Arduino IDE to blink the onboard 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);

}