Lab#1 GIT

Basics:

What is version control? Why is it important?

Version control is a process of saving different versions of a project or documents. It is important as the developers have the freedom of experimentation so that if the current version fails, we can just use the previous version.

What's the difference between Git and GitHub?

Git is a version control for the user’s repository and Github is the cloud based host that manages the repositories.

Describe the Git workflow (add, commit, push, pull).

Git workflow:

1. Create or update files locally
2. Use “add” to add the new file to the locally available repository
3. Then use “commit” to save the changes
4. Use “pull” to make sure the current version of repository resonates with the one available locally
5. “push” the repository to the cloud

What is a repository in the context of Git?

Repository is kind of a folder that contains documents, files of a project or datasets based on the users needs.

Commits:

What is a commit in Git?

Commit is a command used for saving the changes done to the local repository before pushing it to the cloud.

How is each commit uniquely identified?

Each commit is uniquely identified using SHA.

Remote Repositories:

What is a remote repository in the context of Git?

Remote Repository is a place where all the documents are stored in the cloud so that it can be accessed using a link wherever necessary.

What are the default names that Git uses for the repository you cloned from and your local repository?

Origin

How do you synchronize changes from a remote repository to your local one, and vice versa?

First we need to add and commit the changes. Then we can use push to send repository from local system to remote one and pull for the other way around.

GitHub Specifics:

What is a pull request?

Pull request is a request made from the local system to the cloud to fetch the latest version of the repository to the local system.

How do you 'fork' a repository on GitHub, and why might you want to?

Octocat -> Spoon-Knife Repository then click Fork. We can use fork to copy any repository and change the owner to use it independently from the original repo.

How can you use GitHub to collaborate on open-source projects?

Collaboration and Best Practices:

Why is it important to write clear commit messages?

To use it as a log so that all updates can be tracked in a well defined manner.

When collaborating with others, why might it be important to frequently pull the latest changes?

Multiple users means multiple updates being made simultaneously. Frequently pulling the latest changes ensures an updated local repository.

Lab #2 Arduino code to blink light and print status simultaneously.

Theory:

* The online simulator wokwi/tinkercad was used to perform the experiment.
* Arduino UNO board was used.
* A led was connected with the GND pin and the pin number 2 of the board.
* Code was written to generate a blinking effect by turning HIGH and LOW with a delay of 1000ms and print the status using Serial.print() method.

Code:

int Led = 2;

void setup()

{

pinMode(Led, OUTPUT);

Serial.begin(2000);

}

void loop()

{

Serial.print("Led is ON. \n");

digitalWrite(Led, HIGH);

delay(1000);

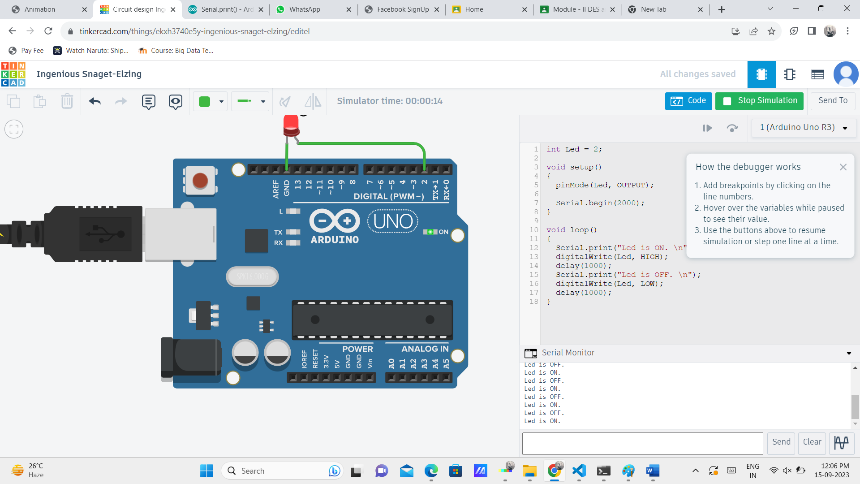
Serial.print("Led is OFF. \n");

digitalWrite(Led, LOW);

delay(1000);

}

Screenshot:

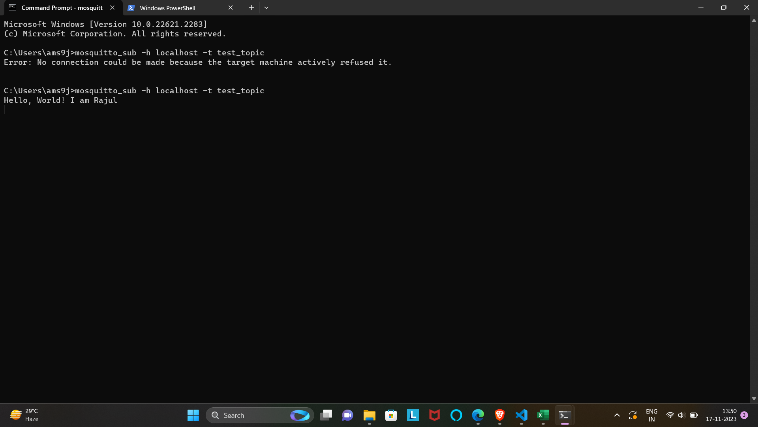
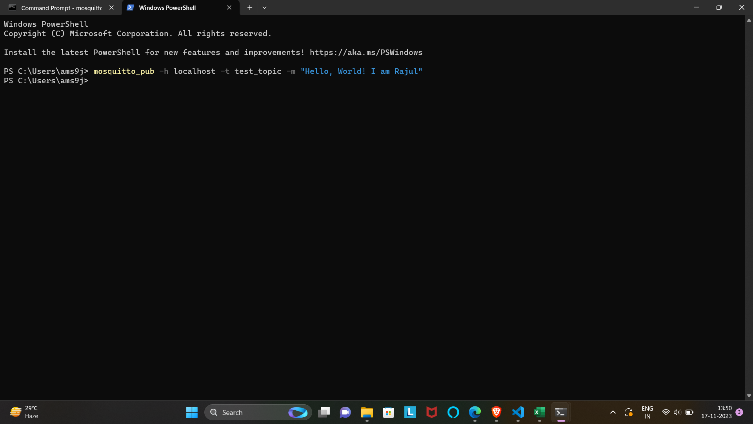


Lab #3 Installation and testing of Mosquitto

Procedure:

1. Visit <https://mosquitto.org/download/>
2. Find the download link for the version compatible with the device. For this case, [mosquitto-2.0.18-install-windows-x64.exe](https://mosquitto.org/files/binary/win64/mosquitto-2.0.18-install-windows-x64.exe) (64-bit build, Windows Vista and up, built with Visual Studio Community 2019)
3. Make sure mosquitto is added to system path.
4. Open cmd and type ‘mosquitto\_sub -h localhost -t test\_topic’.
5. Open another command prompt and run ‘test\_topic’:
   * ‘mosquitto\_pub -h localhost -t test\_topic -m “Hello, World!”’
6. If the first command prompt shows Hello, World! , mosquitto is successsfully installed.

Screenshots:



Lab #4 Red MQTT Installation and testing

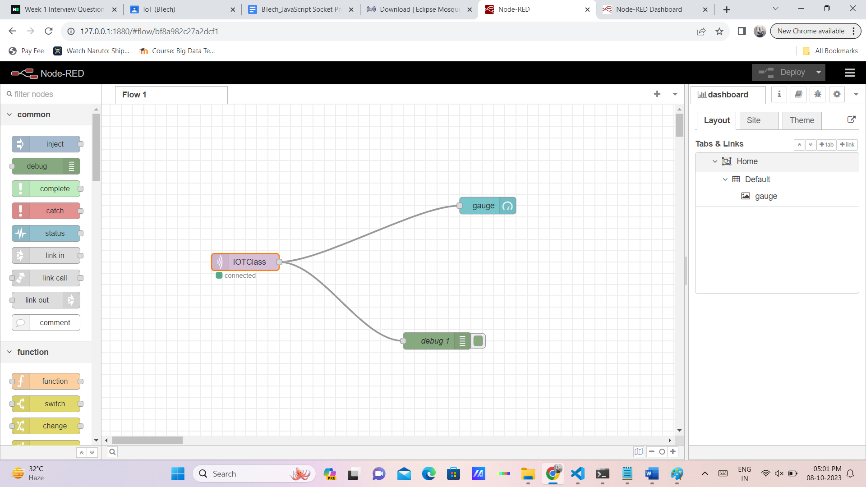
Theory:

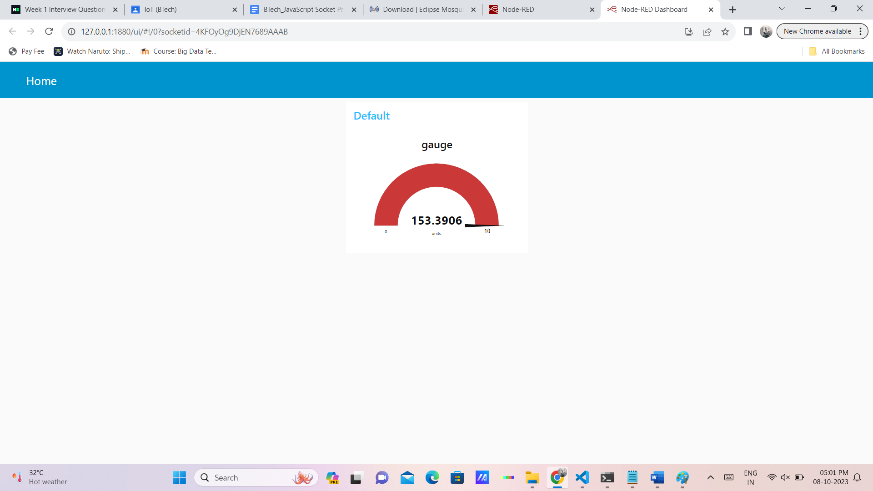
RedMQTT is an application designed to read mobile sensor data and forward it to a local or cloud-based MQTT server.

Procedure:

1. Install RedMQTT on a smartphone.
2. Open node red in pc by typing ‘node-red’ in cmd. Also open hivemq broker.
3. Get the broker name, address and port from hivemq.
4. Open node red tab and add mqtt-in, gauge, and debug to the platform. Connect them as shown in the screenshot.
5. Double click on the mqtt-in. Add broker name, address and port. Then, add a topic name and save it.
6. Go to the RedMQTT app and go to settings. Add the same brover information.
7. Then, go to publish on the app. Select the broker name, enter the same topic name and select the sensor.
8. Now publish the node-red flow and the gauge will show the sensor data.

Screenshots:





Lab #5 Arduino program to read DHT sensor

Program:

#include <DHT.h>

#define DHTPIN 6

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  dht.begin();

  Serial.begin(9600);

}

void loop() {

  // put your main code here, to run repeatedly:

  float H=dht.readHumidity();

  float T=dht.readTemperature();

  if (isnan(H) || isnan(T)) {

    Serial.println("Failed to read from DHT sensor!");

    return;

  }

  String dhtData=String(H) + ","+String(T);

  Serial.println(dhtData);

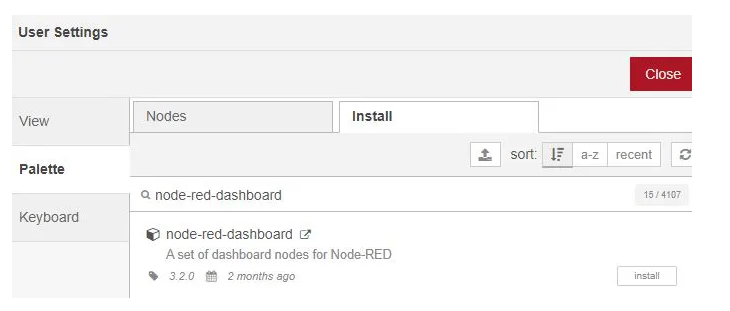
  delay(1000);

}

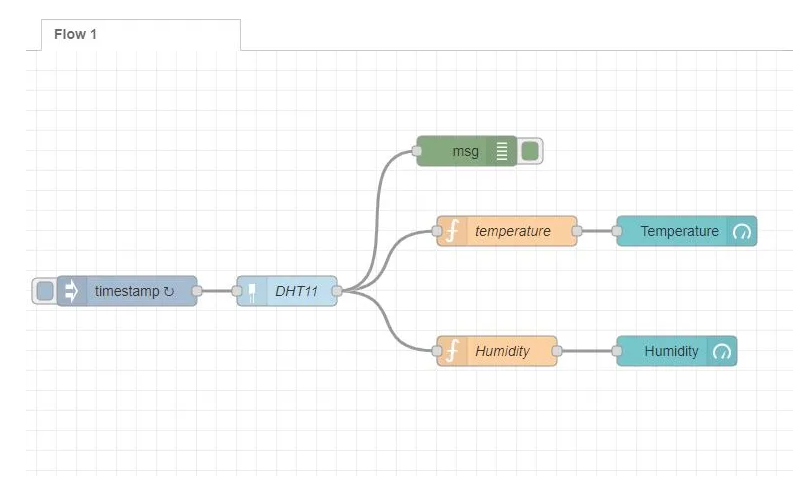
Lab #6 Visualizing DHT sensor data on Node-RED

Procedure:

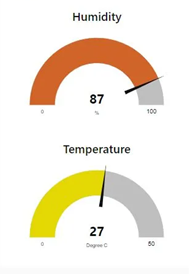
1. Open palette and install node red dashboard along with node-red-contrib-dht-sensor



1. Create the flow like in the image using the following tags:
   1. Inject
   2. Function
   3. Debug
   4. rpi-DHT22
   5. Gauge
   6. Set the properties of above tags as said in the following link: https://iotstarters.com/building-node-red-dashboard-with-dht11-sensor/



Output:



Lab #7 Arduino program to use push button to turn LED on and off.

Program:

int sw=2;

int led=3;

int a=0;

void setup() {

  pinMode(led, OUTPUT);

  pinMode(sw, INPUT);

}

void loop() {

  a=digitalRead(sw);

  if(a==HIGH)

  {

    digitalWrite(led,HIGH);

  }

  else

  {

    digitalWrite(led,LOW);

  }

}

Lab #8 Arduino program to use push button to get output from LED and DHT sensor (ON – LED on and show Temperature, OFF – LED off and show Humidity)

Program:

#include <DHT.h>

#define DHTPIN 6

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

int LED=7;

int sw=2;

void setup()

{

  pinMode(LED, OUTPUT);

  pinMode(sw, INPUT);

  dht.begin();

**Serial**.begin(9600);

}

void loop()

{

  int a=digitalRead(sw);

  if(a==HIGH)

  {

    float H=dht.readHumidity();

    if (isnan(H)) {

**Serial**.println("Failed to read from DHT sensor!");

      return;

    }

    String dhtData="Humidity = "+String(H);

**Serial**.println(dhtData);

    digitalWrite(LED,HIGH);

    delay(1000);

  }

  else

  {

    float T=dht.readTemperature();

    if (isnan(T)) {

**Serial**.println("Failed to read from DHT sensor!");

      return;

   }

    String dhtData="Temperature = "+String(T);

**Serial**.println(dhtData);

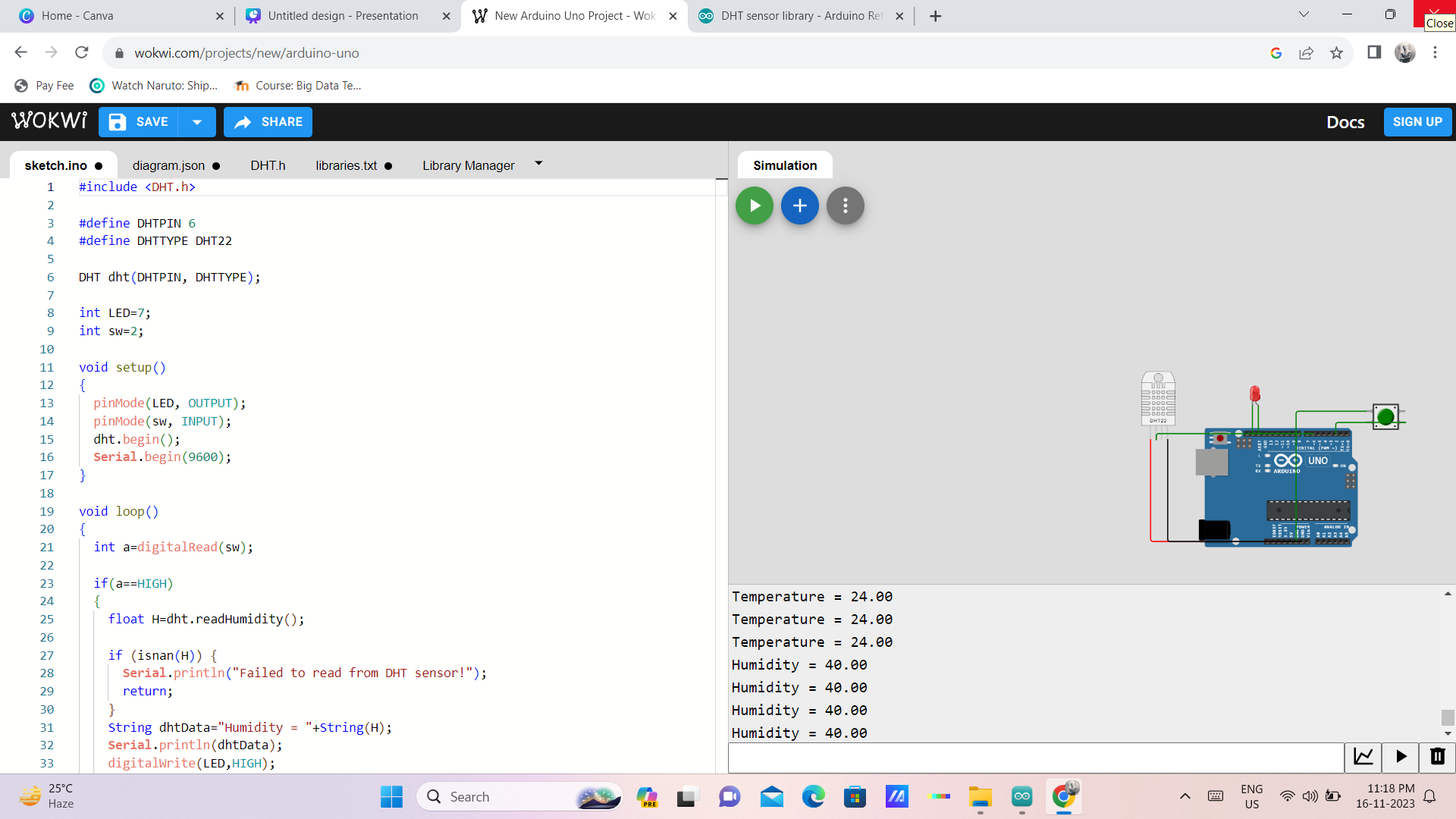
    digitalWrite(LED,LOW);

    delay(1000);

  }

}

OUTPUT:



Lab #9 Arduino program to measure distance from an object using ultrasonic sensor.

Program:

int trigger=2;

int echo=3;

int distance=0;

long duration;

void setup()

{

  pinMode(trigger, OUTPUT);

  pinMode(echo, INPUT);

  Serial.begin(9600);

  Serial.println("Let's find out the distu.");

  delay(100);

}

void loop()

{

  digitalWrite(trigger,LOW);

  delayMicroseconds(2);

  digitalWrite(trigger,HIGH);

  delayMicroseconds(10);

  digitalWrite(trigger,LOW);

  duration=pulseIn(echo,HIGH);

  distance=duration\*0.0344/2;

  Serial.print("Distance is: ");

  Serial.print(distance\*0.394);

  Serial.print(" inch \n");

  delay(2000);

}