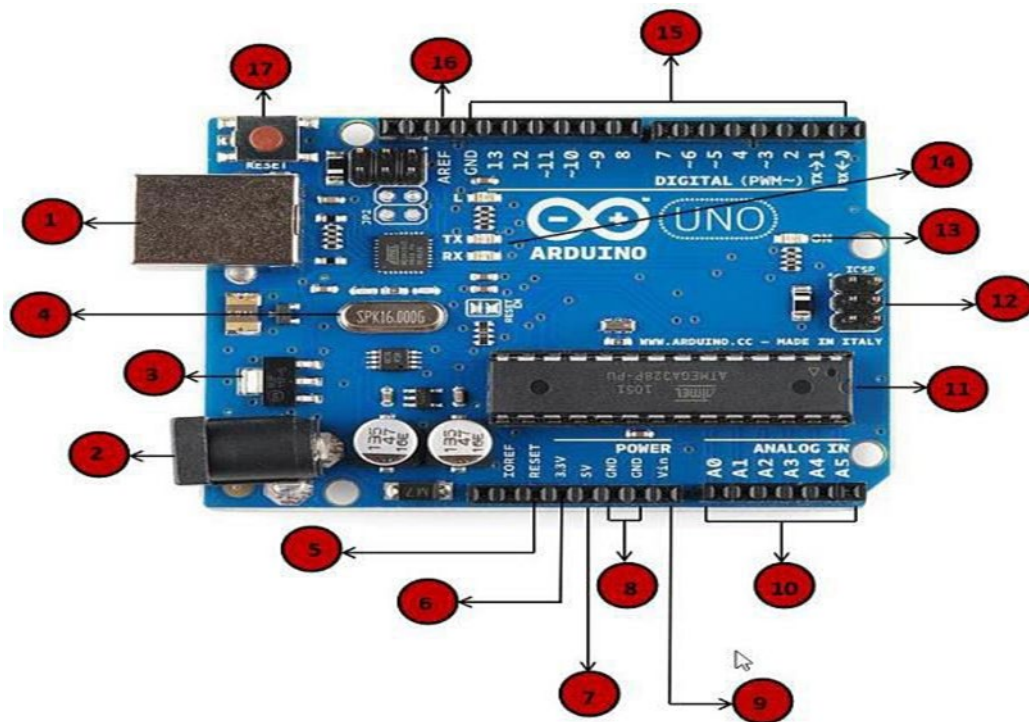


Practical-1

Aim: Study of Arduino board and Interfacing of LED (s) with Arduino.



1

Power USB

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

2

Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3

Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4

Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

5,17

Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7
8,9

Pins (3.3, 5, GND, Vin)

- 3.3V (6) – Supply 3.3 output volt
- 5V (7) – Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
- GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10

Analog pins

The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

11

Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated

circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

12

ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

13

Power LED indicator

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

14

TX and RX LEDs

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication.

Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

15

Digital I/O

The Arduino UNO board has 14 digital I/O pins (15) (of which 6

provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.

16

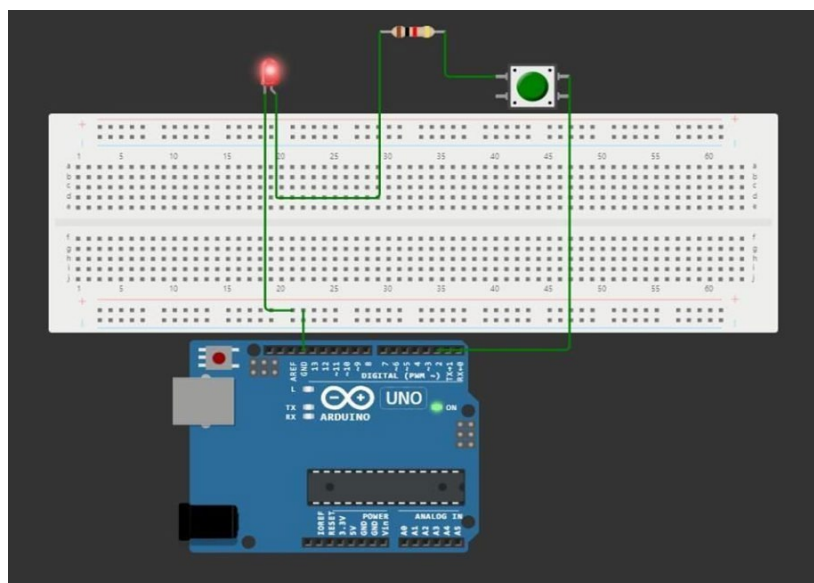
AREF

AREF stands for Analog Reference. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Interfacing:

```
void setup() {  
  // put your setup code here, to run once: pinMode(2, OUTPUT);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly: digitalWrite(2, HIGH);  
  delay(2000); digitalWrite(2, LOW); delay(2000);  
}
```

Output:



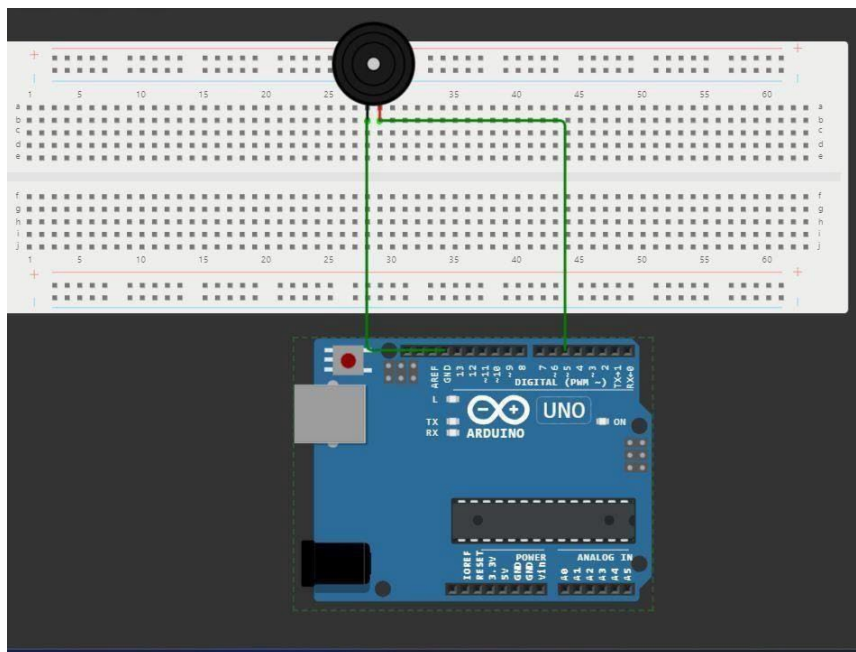
Practical-2

Aim: Study and implementation of Buzzer, Switches, LCD, keypad, LDR, Ultrasonic sensors and PWM interfacing with Arduino.

BUZZER:

```
#define BUZZER 5 void setup()
{
pinMode(BUZZER, OUTPUT);
}
void loop()
{
tone(BUZZER, 85);
delay(1000); noTone(BUZZER); delay(1000);
}
```

Output:

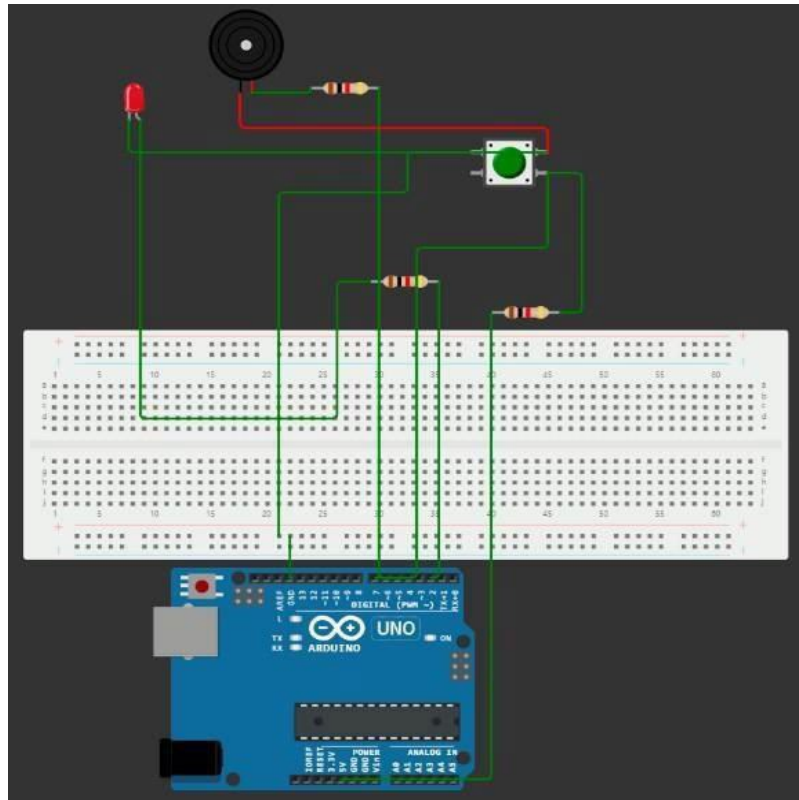


SWITCHES:

```
void setup()
{
    pinMode(2, OUTPUT);
    pinMode(7, INPUT);
    pinMode(4, OUTPUT);
}

void loop()
{
    if(digitalRead(7)==LOW)
    {
        digitalWrite(2, HIGH);
        tone(4,1000);
    }
    else
    {
        digitalWrite(2, LOW);
        noTone(4);
    }
}
```

Output:



LCD :

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);
void setup()
{
    lcd.init();
    lcd.backlight();
    lcd.setCursor(1, 0);
    lcd.print("Hello, Saap!");
}
void loop()
{
    delay(1000);
    lcd.setCursor(2,1);
```

```
lcd.print("How are you?");  
}
```

Output:



KEYPAD :

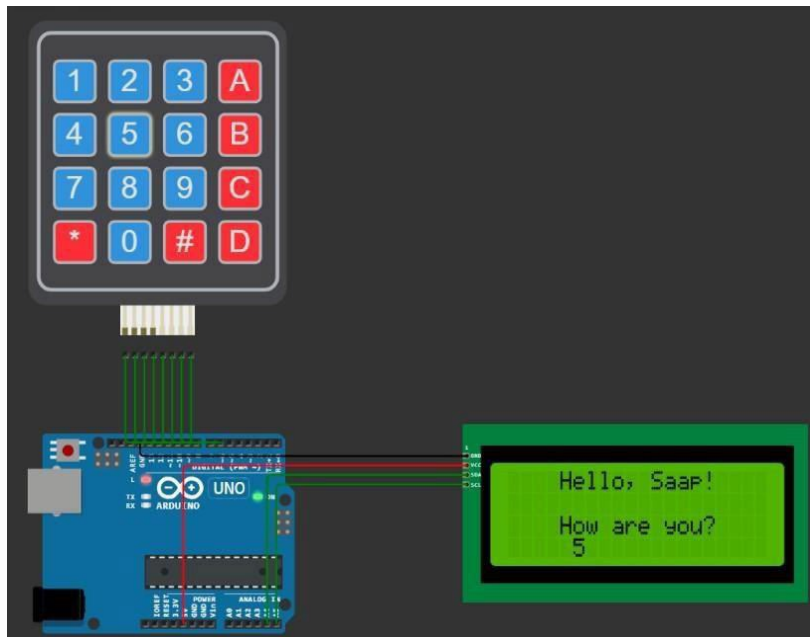
```
#include<LiquidCrystal_I2C.h>  
#include<Keypad.h>  
  
LiquidCrystal_I2C lcd(0x27, 20, 4);  
const byte row=4,col=4;  
char hexaKeys[4][4] = { {'*', '0', '#', 'D'},  
                        {'7', '8', '9', 'C'},  
                        {'4', '5', '6', 'B'},
```



```
        {'1','2','3','A'}
    };

byte rowPins[4] = {10, 11, 12, 13};
byte colPins[4] = {6, 7, 8, 9};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, row,col);
void setup()
{
    Serial.begin(9600);
    lcd.init();
    lcd.backlight();
    lcd.setCursor(4,0);
    lcd.print("Hello,Saap!");
    lcd.setCursor(4,2);
    lcd.print("How are you?");
}
void loop()
{
    char customKey=customKeypad.getKey();
    if (customKey)
    {
        lcd.setCursor(5,3);
        lcd.print(customKey);
    }
}
```

Output:



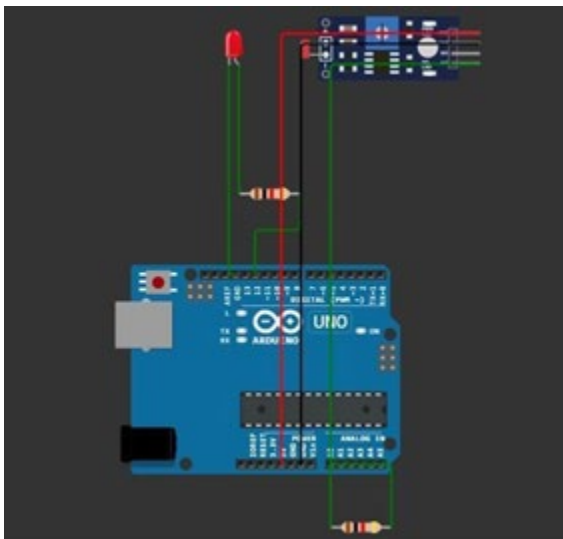
LDR:

```
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    pinMode(A0, INPUT);
    pinMode(12, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    int sensorStatus = analogRead(A0);
    digitalWrite(12, HIGH);
    if (sensorStatus < 200)
    {
        digitalWrite(12, HIGH); // LED is ON
        Serial.println(" LED is ON, status of sensor is DARK");
    }
}
```

```
    delay(1000);  
}  
else  
{  
    digitalWrite(12, LOW);  
    Serial.println("  
*****"); delay(1000);  
}  
}
```

Output:



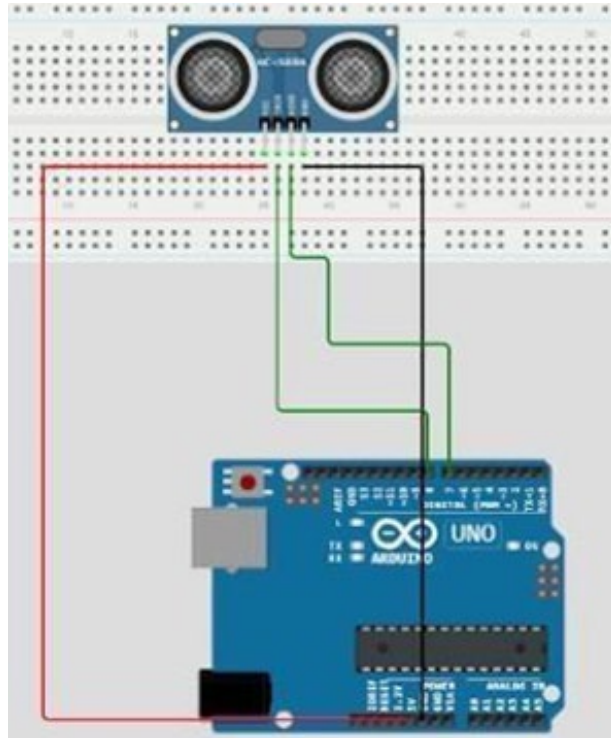
```
LED is ON, status of sensor is DARK  
LED is ON, status of sensor is DARK  
*****  
*****
```

ULTRASONICS:

```
const int pingPin=7;  
const int echoPin=6;  
void setup()  
{  
    Serial.begin(9600);  
}  
void loop()
```

```
{  
    long duration, inches, cm;  
    pinMode(pingPin, OUTPUT);  
    digitalWrite(pingPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(pingPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(pingPin, LOW);  
    pinMode(echoPin, INPUT);  
    duration=pulseIn(echoPin,HIGH);  
    inches = microsecondsToInches(duration);  
    cm=microsecondsToCentimeters(duration);  
    Serial.print(inches);  
    Serial.print("in");  
    Serial.print(cm);  
    Serial.print("cm");  
    Serial.println();  
    delay(100);  
}  
  
long microsecondsToInches(long microseconds)  
{  
    return microseconds / 74 / 2;  
}  
  
long microsecondsToCentimeters(long microseconds)  
{  
    return microseconds / 29 / 2;  
}
```

Output:

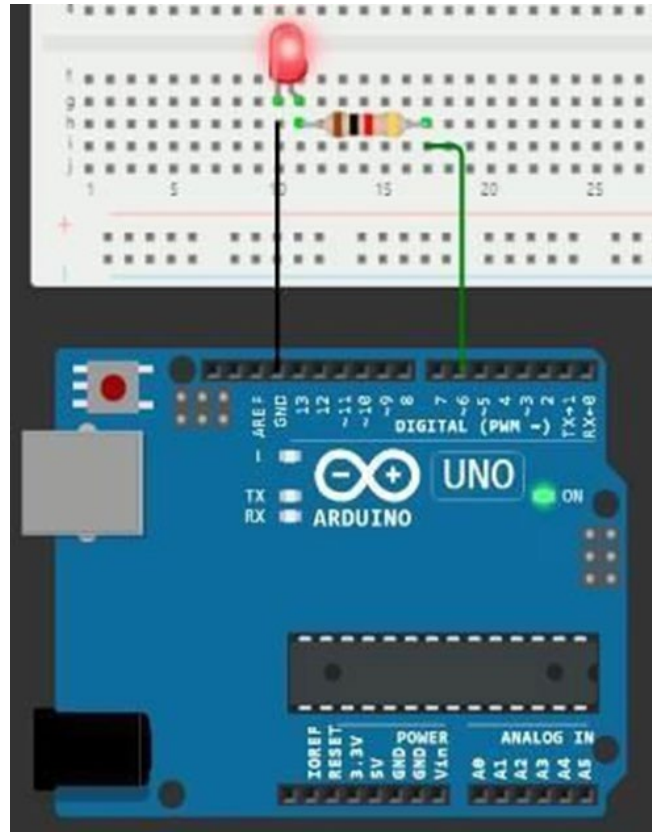


PWM Interfacing:

```
int fade=5,b=0;
void setup()
{
    pinMode(6, OUTPUT);
}
void loop()
{
    analogWrite(6);
    b=b+fade;
    if(b<=0||b>=25)
    {
        fade=-fade;
    }
}
```

```
}  
delay(10);  
}
```

Output:



Practical-3

Aim: Study of serial communication and device control using serial communication with Arduino.

Studying serial communication and device control using serial communication with Arduino is an excellent way to learn about the fundamentals of communication protocols, microcontroller programming, and interfacing with external devices. Serial communication involves the transmission and reception of data between two devices using a specific communication protocol. Arduino, a popular open-source microcontroller platform, provides an easy way to experiment with serial communication and control various devices.

Here's a step-by-step guide to get you started on this topic:

Introduction to Serial Communication:

- Understand the basics of serial communication, including the concepts of data bits, baud rate, start and stop bits, and parity. Serial communication can be asynchronous (UART) or synchronous (SPI, I2C).

Arduino Setup:

- Get an Arduino board (e.g., Arduino Uno) and install the Arduino IDE on your computer. Familiarize yourself with the IDE's interface, code editor, and serial monitor.

Serial Library:

- Learn how to use the Arduino Serial library, which provides functions for sending and receiving data through the serial port.

Serial Communication Modes:

- Explore different modes of serial communication:
- UART (Universal Asynchronous Receiver-Transmitter): Simplex communication for point-to-point connections.
- SPI (Serial Peripheral Interface): Full-duplex synchronous communication for connecting multiple devices in a bus.
- I2C (Inter-Integrated Circuit): Multi-master, multi-slave synchronous communication for short-

distance connections.

Wiring and Connections:

- Learn how to physically connect devices to the Arduino using appropriate wiring configurations for UART, SPI, or I2C.

Writing Arduino Code:

- Write Arduino sketches (programs) to establish serial communication with external devices. Use the Serial library to send and receive data.

Device Control Examples:

- Experiment with different devices for practical learning:
- LED Control: Use serial commands to turn an LED on/off or control its brightness.
- Servo Motor Control: Control the position of a servo motor using serial input.
- Temperature Sensor: Read data from a temperature sensor (e.g., LM35) and send it over serial.

Bi-Directional Communication:

- Implement two-way communication between Arduino and your computer. Send commands from the computer to control devices connected to the Arduino, and receive data back from the Arduino.

Debugging and Serial Monitor:

- Learn how to use the Arduino Serial Monitor to debug your code, monitor data transmission, and receive feedback from your Arduino board.

Advanced Topics:

- Explore more advanced concepts as you become comfortable with serial communication:
- Error handling and data validation.
- Implementing custom communication protocols.
- Interfacing with sensors, actuators, and displays.

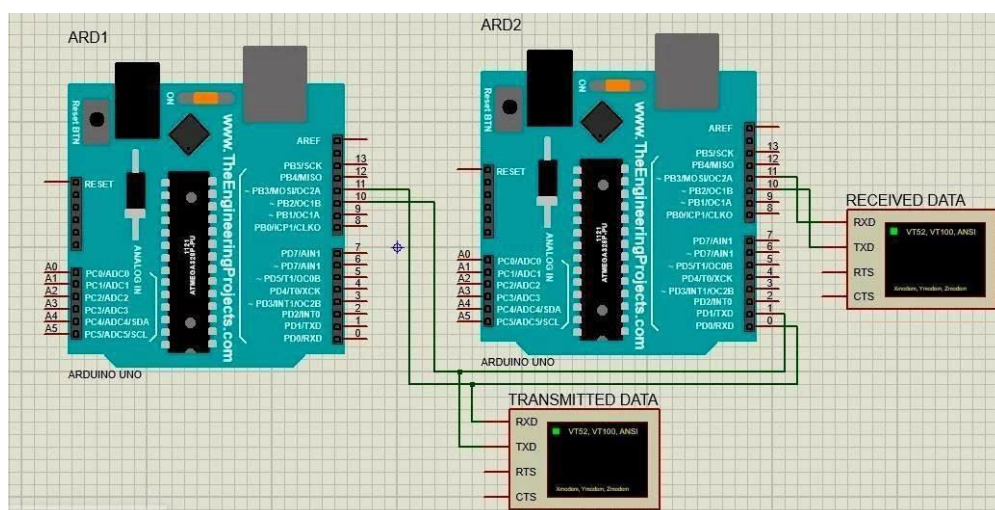
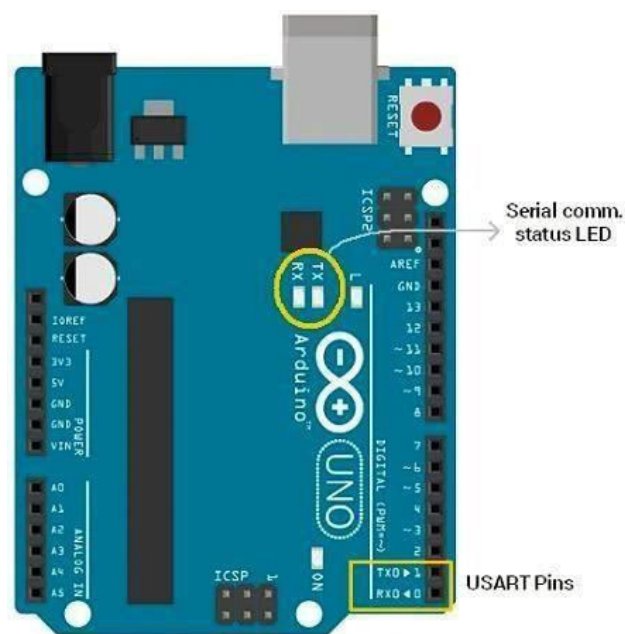
Projects and Applications:

- Create more complex projects involving multiple devices and sensors. For example:

- Wireless communication using Bluetooth or RF modules.
- Data logging and remote control applications.
- Home automation and IoT projects.

Documentation and Resources:

- Utilize online resources, Arduino documentation, forums, and tutorials to expand your knowledge and troubleshoot issues.



Practical-4

Aim: Study and implementation LED (s) Interfacing with NodeMCU.

Interfacing LEDs with the NodeMCU (an ESP8266-based development board) is a great way to learn about basic digital output control and IoT (Internet of Things) concepts. Here's a step- by-step guide to studying and implementing LED interfacing with NodeMCU:

To interface an LED with NodeMCU, you will need the following components:

- A NodeMCU board
- An LED
- A resistor
- A breadboard
- Jumper wires

Once you have your components, you can follow these steps to connect the LED to the NodeMCU:

1. Connect the positive leg of the LED to the digital pin D1 of the NodeMCU.
2. Connect the negative leg of the LED to the ground pin (GND) of the NodeMCU.
3. Connect a resistor in series with the LED. The value of the resistor will depend on the power rating of the LED. For a standard LED, a 220-ohm resistor will work well.
4. Connect the breadboard to the NodeMCU.
5. Use jumper wires to connect the LED and resistor to the breadboard.

Once the LED is connected to the NodeMCU, you can write code to control it. The following code will blink the LED:

```
void setup()
{
    pinMode(D1, OUTPUT);
}

void loop()
{
    digitalWrite(D2, HIGH);
    delay(1000);
```

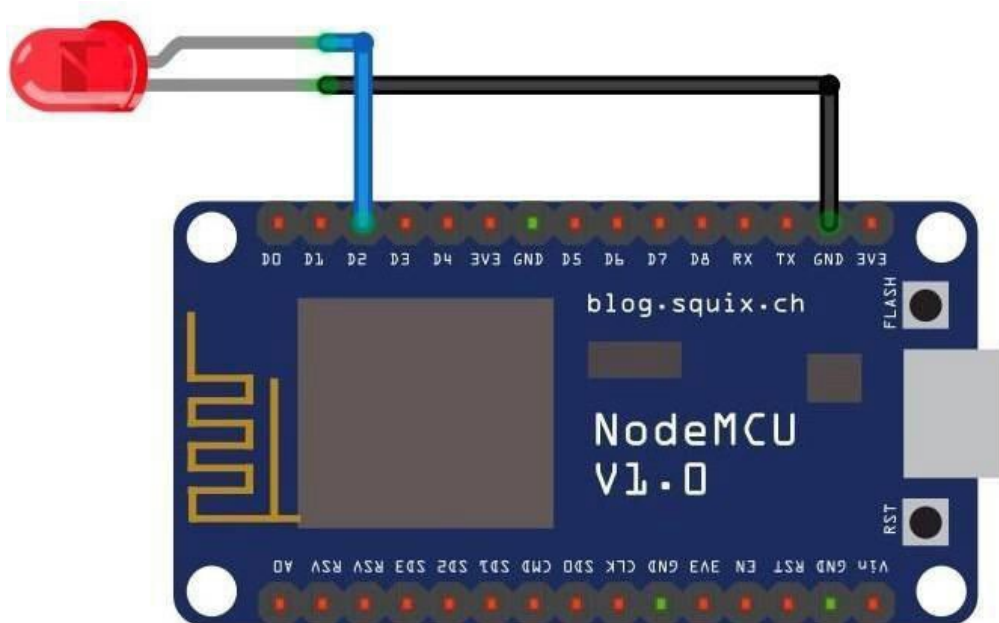
```
digitalWrite(D2, LOW);  
delay(1000);  
}
```

This code will set the digital pin D1 as an output pin. Then, it will turn the LED on for 1 second, then turn it off for 1 second. This will continue to repeat indefinitely.

You can also use the NodeMCU to control the brightness of the LED. The following code will fade the LED in and out:

```
void setup()  
{  
    pinMode(D2, OUTPUT);  
}  
void loop()  
{  
    for (int i = 0; i <= 255; i++)  
    {  
        analogWrite(D2, i);  
        delay(10);  
    }  
    for (int i = 255; i >= 0; i--)  
    {  
        analogWrite(D2, i);  
        delay(10);  
    }  
}
```

Output:



Practical-5

Aim: Implementation of Publishing data on thingspeak cloud.

To implement publishing data on ThingSpeak cloud, you will need the following:

- A ThingSpeak account
- A NodeMCU board
- A sensor
- A breadboard
- Jumper wires

Once you have your components, you can follow these steps to publish data to ThingSpeak cloud:

1. Create a ThingSpeak channel.
2. Get the API key for your ThingSpeak channel.
3. Connect the sensor to the NodeMCU.
4. Write code to read the sensor data and publish it to ThingSpeak cloud.
5. Upload the code to the NodeMCU.

The first step in publishing data to ThingSpeak cloud is to create a ThingSpeak channel. This can be done by visiting the ThingSpeak website and creating an account. Once you have created an account, you can create a channel by clicking on the "Create Channel" button.

The next step is to get the API key for your ThingSpeak channel. The API key is a secret string that is used to authenticate your requests to ThingSpeak cloud. You can find your API key on the "API Keys" tab of your ThingSpeak channel.

Once you have your API key, you can connect the sensor to the NodeMCU. The sensor that you use will depend on the data that you want to publish to ThingSpeak cloud. In the example above, we are using a DHT11 sensor to publish temperature data.

The next step is to write code to read the sensor data and publish it to ThingSpeak cloud. The code that you need to write will depend on the sensor that you are using. In the example above, we are using the DHT.h library to read the temperature data from the DHT11 sensor.

The final step is to upload the code to the NodeMCU. You can do this by using the Arduino IDE.

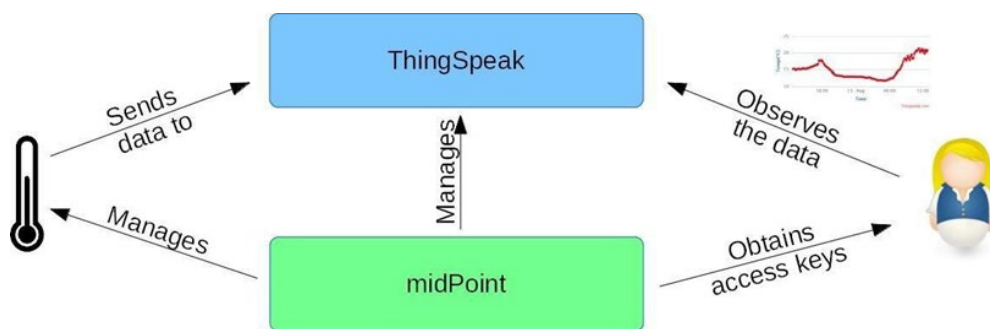
Once the code is uploaded, the NodeMCU will start publishing data to ThingSpeak cloud.

The following code will read the temperature from a DHT11 sensor and publish it to ThingSpeak cloud:

```
#include <DHT.h>
#include <WiFi.h>
#include <ThingSpeak.h>
const char *ssid = "YOUR_WIFI_SSID";
const char *password = "YOUR_WIFI_PASSWORD";
const int DHT_PIN = 2;
const int THINGSPEAK_CHANNEL_ID = 123456;
const char *THINGSPEAK_API_KEY = "YOUR_THINGSPEAK_API_KEY";
DHT dht(DHT_PIN, DHT11);
void setup()
{
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    ThingSpeak.begin(client);
}
void loop()
{
    float temperature = dht.readTemperature();
    if (isnan(temperature))
    {
```

```
Serial.println("Failed to read temperature");  
return;  
}  
Serial.println("Temperature: " + String(temperature));  
ThingSpeak.setField(1, temperature);  
ThingSpeak.writeFields(THINGSPEAK_CHANNEL_ID, THINGSPEAK_API_KEY);  
delay(15000);  
}
```

Output:



Practical-6

Aim: Controlling Appliances using NodeMCU MQTT over the Internet. (Adafruit Cloud)

To control appliances using NodeMCU MQTT over the Internet (Adafruit Cloud), you will need the following:

- A NodeMCU board
- A relay module
- An appliance that you want to control
- A breadboard
- Jumper wires
- An Adafruit IO account

Once you have your components, you can follow these steps to control your appliance:

1. Connect the relay module to the NodeMCU.
2. Connect the appliance to the relay module.
3. Connect the breadboard to the NodeMCU.
4. Use jumper wires to connect the relay module and appliance to the breadboard.
5. Create an Adafruit IO account and create a feed for your appliance.
6. Write code to publish a message to the feed when you want to control the appliance.
7. Upload the code to the NodeMCU.

The following code will publish a message to the feed when you press the button:

```
#include <PubSubClient.h>

const char *ssid = "YOUR_WIFI_SSID";
const char *password = "YOUR_WIFI_PASSWORD";
const char *mqtt_server = "io.adafruit.com";
const int mqtt_port = 1883;
const char *username = "YOUR_ADAFRUIT_IO_USERNAME";
const char *password = "YOUR_ADAFRUIT_IO_PASSWORD";
const char *feed_name = "YOUR_FEED_NAME";
```



```
WiFiClient;
PubSubClient client(wifiClient);
void setup()
{
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    while (!client.connected())
    {
        Serial.println("Connecting to MQTT...");
        if (client.connect(username, password))
        {
            Serial.println("MQTT connected");
        }
        else
        {
            Serial.print("failed, rc=");
            Serial.println(client.state());
            delay(5000);
        }
    }
}
client.subscribe(feed_name);
}
```

```
void loop()
{
    client.loop();
    if (digitalRead(D1) == HIGH)
    {
        String message = "1";
        client.publish(feed_name, message);
        delay(1000);
    }
    else
    {
        String message = "0";
        client.publish(feed_name, message);
        delay(1000);
    }
}

void callback(char *topic, byte *payload, unsigned int length)
{
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++)
    {
        Serial.print((char)payload[i]);
    }
    Serial.println();
    if (strcmp(topic, feed_name) == 0)
    {
        if ((char)payload[0] == '1')
        {
            digitalWrite(D1, HIGH);
        }
    }
}
```

```
    }  
    else  
    {  
        digitalWrite(D1, LOW);  
    }  
}
```

Output:

```
WiFi connected  
MQTT connected  
Subscribing to feed: YOUR_FEED_NAME  
Message arrived [YOUR_FEED_NAME] 1  
Appliance turned on  
Message arrived [YOUR_FEED_NAME] 0  
Appliance turned off
```

Practical-7

Aim: Study and Setup Raspberry Pi board.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT). The Raspberry Pi operates in the open- source ecosystem: it runs Linux (a variety of distributions), and its main supported operating system, Pi OS, is open source and runs a suite of open-source software. The Raspberry Pi Foundation contributes to the Linux kernel and various other open-source projects as well as releasing much of its own software as open source.

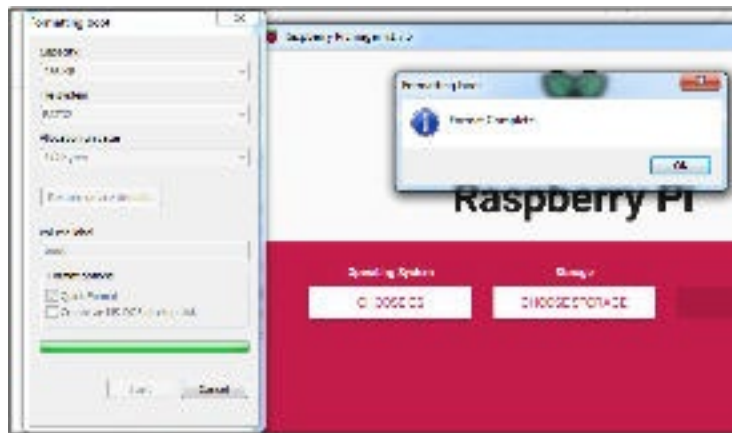
OS for Raspberry Pi: Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, etc.

Raspberry Pi processor: It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit).

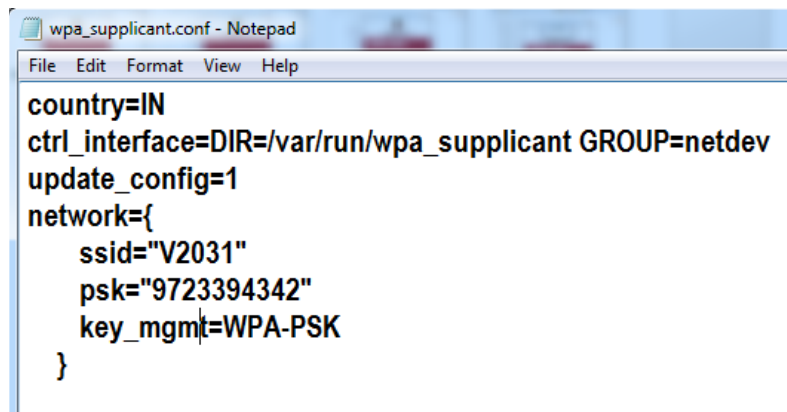
Applications and files required:

1. Raspberry Pi Imager (Windows): [Imager](#)
2. wpa_supplicant.config file
3. Advanced IP Scanner: [IP Scanner](#)
4. PuTTY: [Putty](#)
5. VNC Server: [Server](#)
6. VNC Viewer (Download link provided when successfully singed into the server in the VNC server itself)

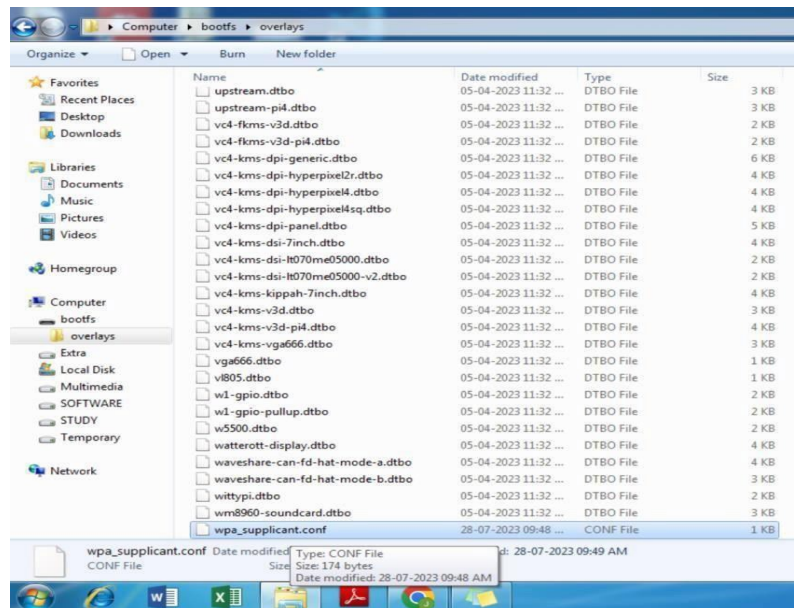
Setup for Raspberry Pi OS on Windows:



1. Format the SD card after inserting it in the laptop or CPU.
2. Open wpa_supplicant.conf file using notepad. Change the SSID configurations in the file and save it.



3. Insert the formatted SD card. Copy this edited config file in the overlays folder.



4. Open the Imager application and the following screen will appear.



5. Choose OS (either custom OS or Raspberry 32-bit). Here, Raspberry Pi 32-bit was selected.



6. Choose the storage as the SD card inserted.



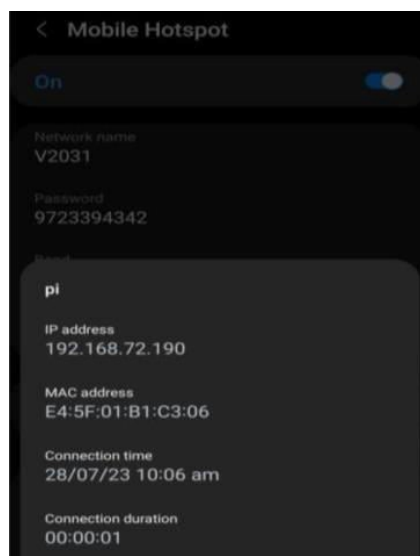
7. Click on the settings and do the following changes in the settings. Set host as “pi”
- Enable SSH
 - Set username and password as “pi”
 - Select Configure LAN
 - Enter the SSID and password as edited in the wpa_supplicant.config file Save the changes
8. Click on the write option. This will start writing OS into the SD card.



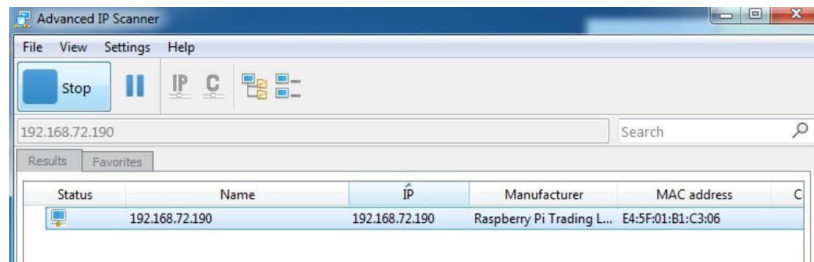
9. Once the writing process is completed, the verifying process will start.



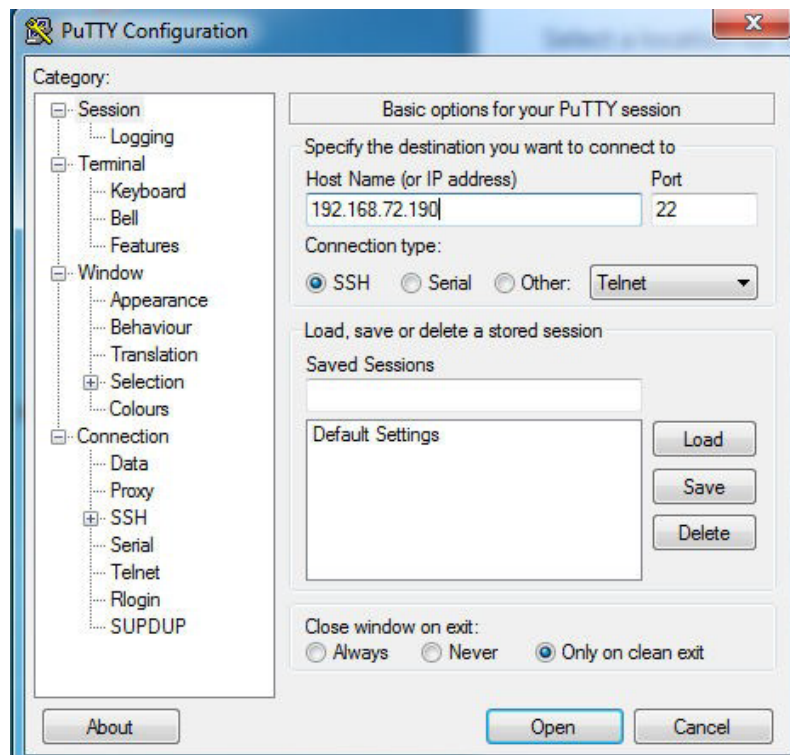
10. Once the verifying is completed, eject the SD card and insert the card in the raspberry Pi board.
11. Check the mobile hotspot if the board and desktop is connected over this specified the network. Note the IP address of the Pi board connected.



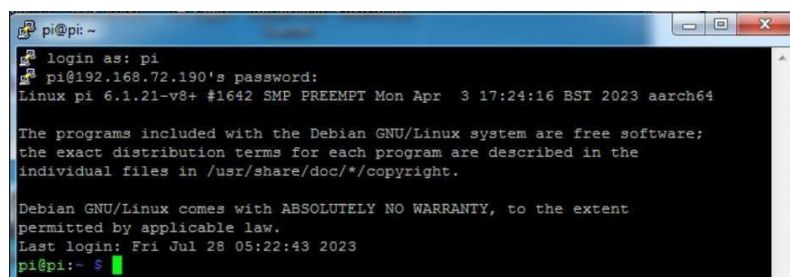
12. Open IP scanner application. Enter the IP address noted in the above step and click on scan.
13. Once the scanning process is completed, note the Raspberry Pi foundation or raspberry Pi Training kit. This confirms the connection of the board with the network.



14. Open putty terminal. Enter the IP address and click on open.



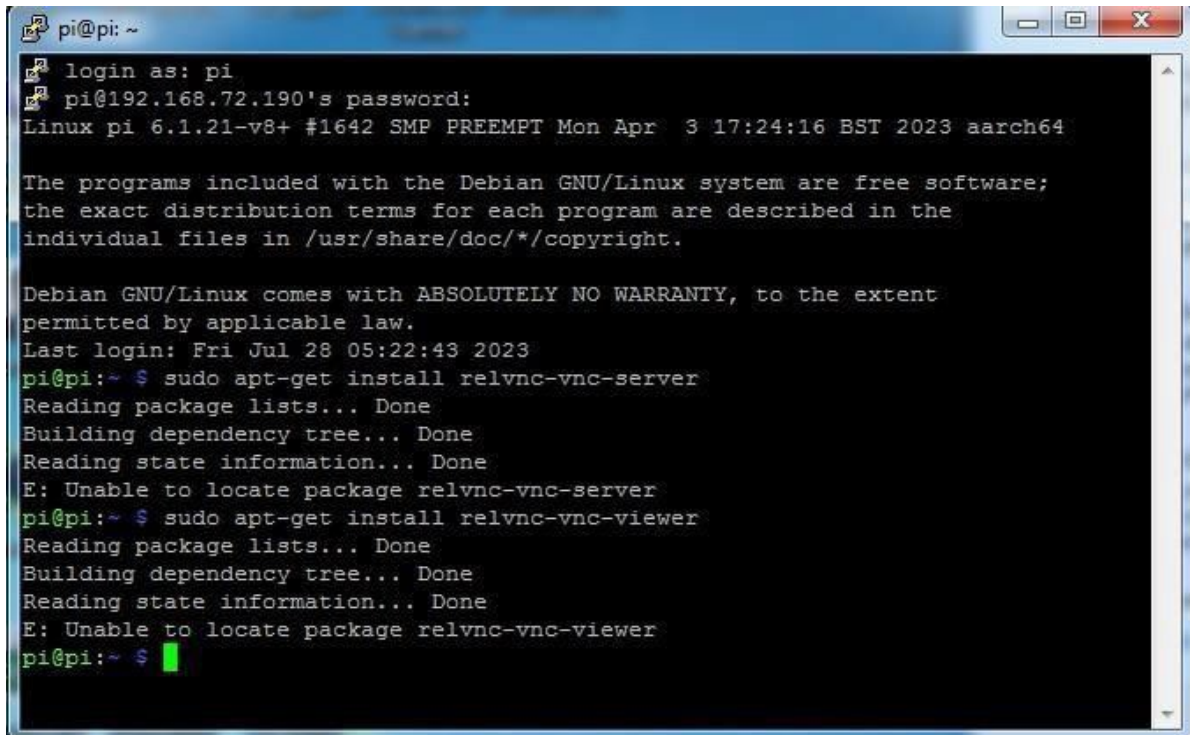
15. This will lead to the terminal. If login credentials are asked, the board is ready to configure on the desktop. Otherwise, look for the alternate solutions for configuring the IP address.
16. Enter login as and password as “pi” as specified in the settings of the imager application in step 7.



17. Run the following commands in the putty terminal to connect VNC to raspberry Pi board:

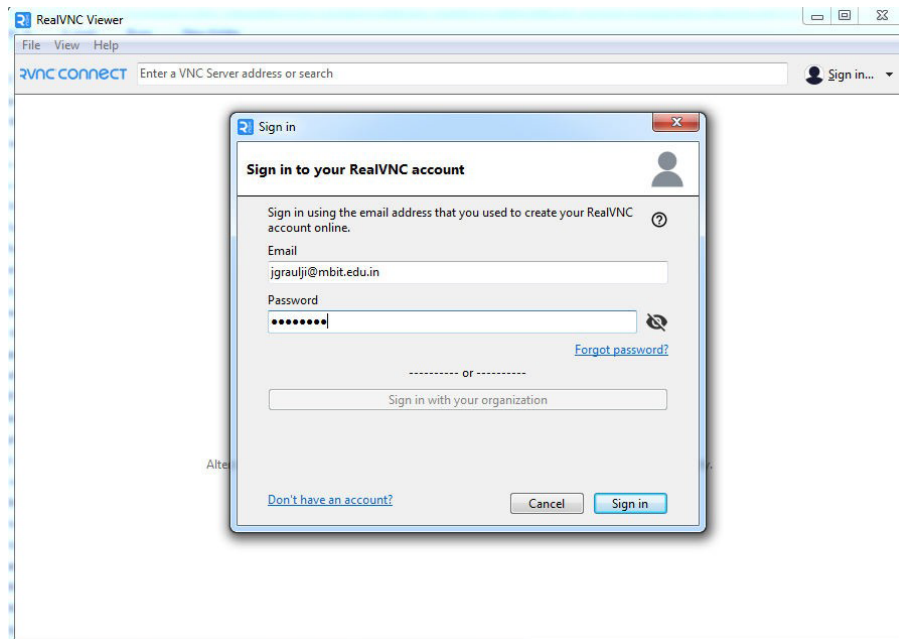
```
sudo apt-get install realvnc-  
vnc-server sudo apt-get install  
realvnc-vnc-viewer
```

[Reference link: Commands for putty terminal](#)

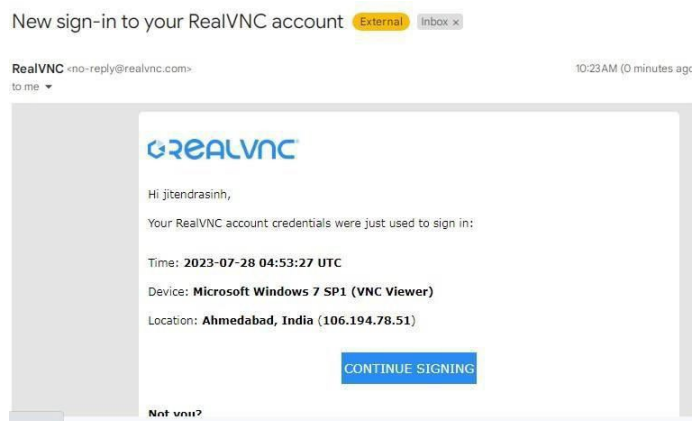


```
pi@pi: ~  
login as: pi  
pi@192.168.72.190's password:  
Linux pi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023 aarch64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Fri Jul 28 05:22:43 2023  
pi@pi:~ $ sudo apt-get install relvnc-vnc-server  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
E: Unable to locate package relvnc-vnc-server  
pi@pi:~ $ sudo apt-get install relvnc-vnc-viewer  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
E: Unable to locate package relvnc-vnc-viewer  
pi@pi:~ $
```

18. Open the browser, search for VNC for raspberry pi board. Make an account on VNC. Note the email ID and password as it will be required for signing in in VNC server.
19. Open VNC server.
20. Open VNC viewer. Enter the credentials of the account made in step 19.



21. After clicking sign in, an email confirmation will be sent to the registered email. Authorise the sign-in through email.



Confirm sign-in

Again, please check you recognize this device, since authorizing will sign it in:

Time

2023-07-28 04:53:27 UTC

Device

Microsoft Windows 7 SP1 (VNC Viewer)

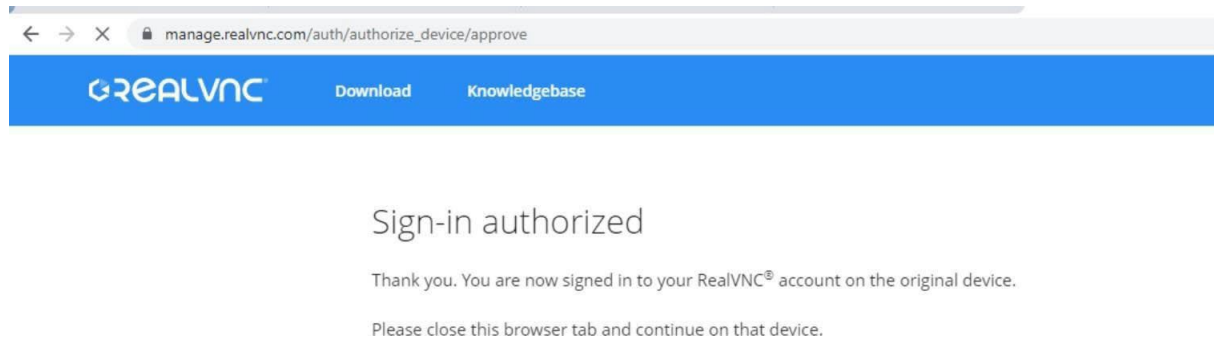
Location

Ahmedabad, India (106.194.78.51)

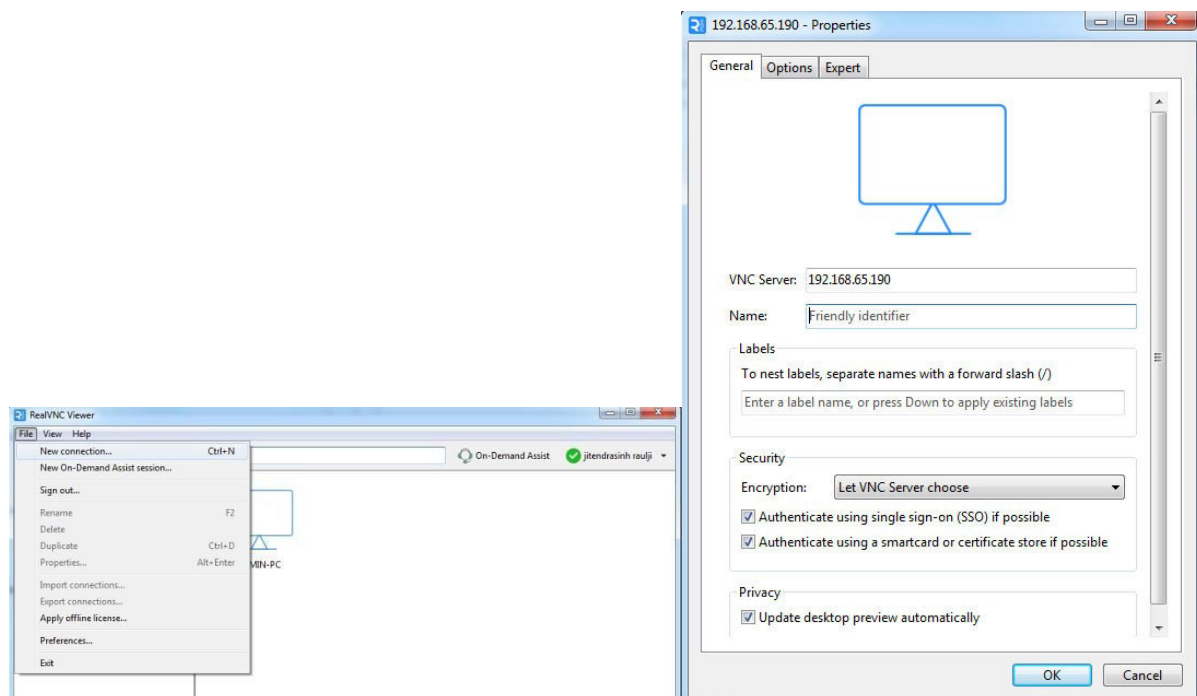
[Authorize sign-in](#)

Don't recognize this device?

We strongly recommend you change your RealVNC® account password as soon as possible. To do this, sign in yourself and navigate to the **Security** page.

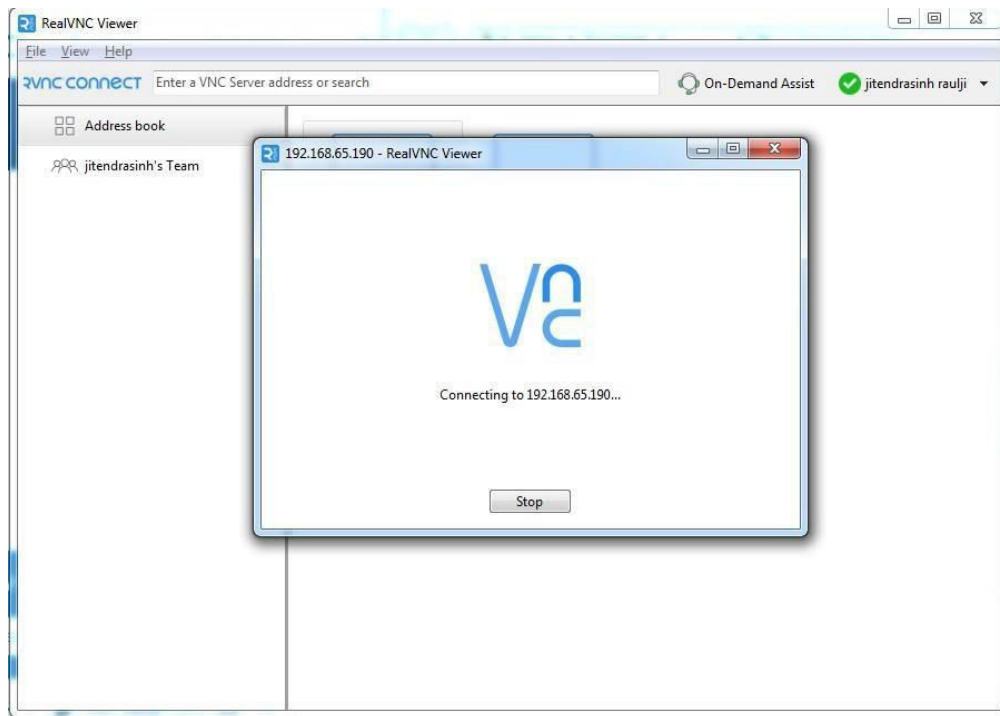


22. Once the VNC sign-in is authorized, the following screen will be displayed.
23. Re-open VNC viewer. File > New connection > Enter the IP address of the Pi board.

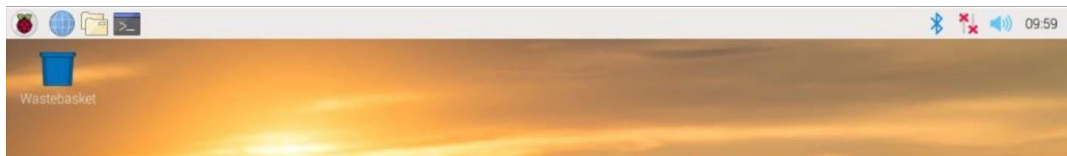


Note: If the connection refuses the connection “network connection timed out”. Check if your system has any firewall which is disrupting the Debian OS to open.

24. Double-click on the connection created. The connection process will start.



25. The following Debian OS will be displayed.



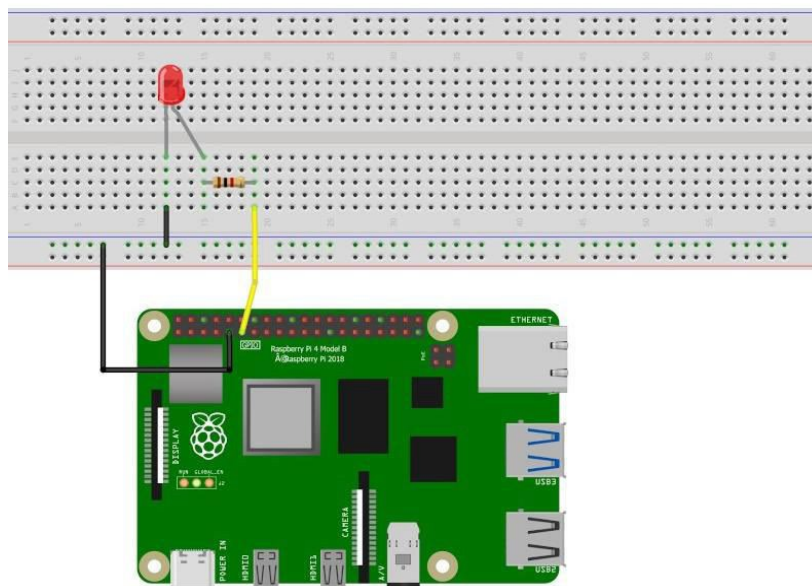
Practical-8

Aim: Study and implementation of LED(s) Interfacing with raspberry pi.

To build this circuit you will need:

- A breadboard
- A Raspberry Pi with GPIO header
- 1 LED – the color doesn't matter
- 1 resistor: any value between 330 Ohm to 1 k Ohm will be fine.
- A set of male to female wires.

Schematics to plug an LED to your Raspberry Pi:



Steps to build the circuit:

1. Connect one wire between one GND (ground) pin of the Raspberry Pi and the blue line of the breadboard.
2. Take the LED and check the 2 legs. You will see that one is shorter than the other. Plug the shorter leg to the blue line (now connected to GND), and the longer to any other connector. You can either directly connect the shorter leg to the blue line, or add an additional short male- to-male connector (like in the picture), the result is the same.
3. Plug one leg of the resistor to the same line as the longer leg of the LED, and the other leg of the resistor to a different line.

4. Finally, to close the circuit plug one wire between the same line as the other leg of the resistor, and the GPIO number 17 . This is the 6th pin on the GPIO header, starting from the left, on the inside side.

Control the LED with Python 3 on Raspberry Pi OS

hardware + software is correctly setup, you can start to control the LED on Raspberry Pi with Python3.

Open Thonny IDE on Raspberry Pi OS (Menu > Programming > Thonny Python IDE)

Simple control of the LED

```
import RPi.GPIO as GPIO
import time
LED_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.output(LED_PIN, GPIO.HIGH)
time.sleep(1)
GPIO.output(LED_PIN, GPIO.LOW)
GPIO.cleanup()
```

Make the LED blink

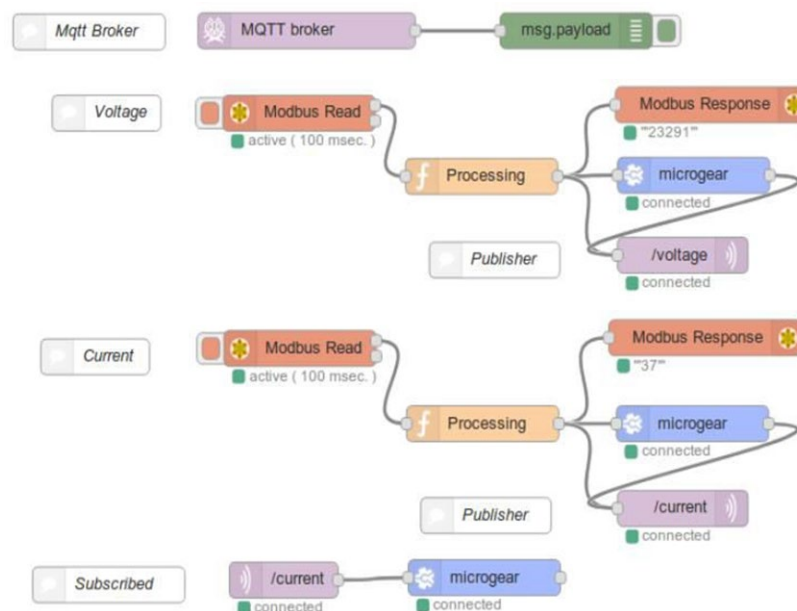
```
import RPi.GPIO as GPIO
import time
LED_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
while True:
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(1)
GPIO.cleanup()
```


Practical-9

Aim: Study of Node-red programming tool.

What is Node-RED and what is it for?

- Node-RED is an open-source development tool based on visual programming that was created by IBM to connect hardware devices, APIs and online services.
- Node-RED is a solid tool, easy to learn, and it does not require any programming knowledge. It has been consolidated as one of the main applications for real-time data management and transformation for IoT and Industry 4.0 solutions.
- Node-RED allows to graphically connect predefined blocks, called nodes, to develop a concrete task. The nodes connection, usually a combination of input nodes, processing nodes and output nodes, when wired together, make up a flow.
- Among all the available nodes we can find standard protocols as MQTT, REST, Modbus, OPC-UA, Bacnet, Websocket; and third party API integrations as Microsoft Azure, Amazon Web Services, Twitter, Facebook and many more.



Install Node-RED:

Node-RED is built on Node.js, taking full advantage of its power and assuring scalability, liability and low hardware requirements. These features allow Node-RED in personal computers, cloud servers and low-cost embedded hardware.

Running the following command will download and run the script.

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

This script will:

- remove the existing version of Node-RED if present.
- if it detects Node.js is already installed, it will ensure it is at least v14. If less than v14 it will stop and let the user decide whether to stay with Node-RED version 1 - or upgrade Node.js to a more recent LTS version. If nothing is found it will install the Node.js 16 LTS release using the NodeSource package.
- install the latest version of Node-RED using npm.
- optionally install a collection of useful Pi-specific nodes.
- setup Node-RED to run as a service and provide a set of commands to work with the service.

Nodes library

The nature of the tool, being open source, and the facility to develop new nodes, come together into a nodes library which grows each day with new community contributions.

Nowadays we can find more than 2500 available nodes in the Node-RED official library, including Smart Home integrations, converters between IoT protocols, geolocation functions, OAuth2 authentication and many more.

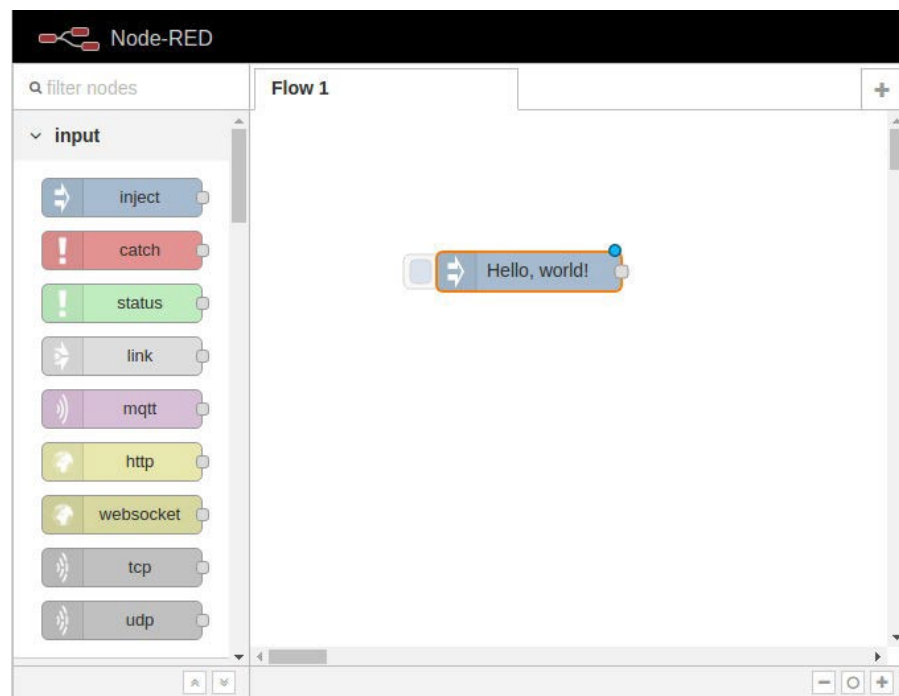
Hello world in Node-RED

Message creation with Inject node.

1. If you are running Node-RED on your computer, you can access `http://127.0.0.1:1880` or whatever address or hostname.
2. If it is our first time in Node-RED, we will see a flow named *Flow 1*
3. On the node palette on the left side of the Node-RED, we will select the *Inject* node and

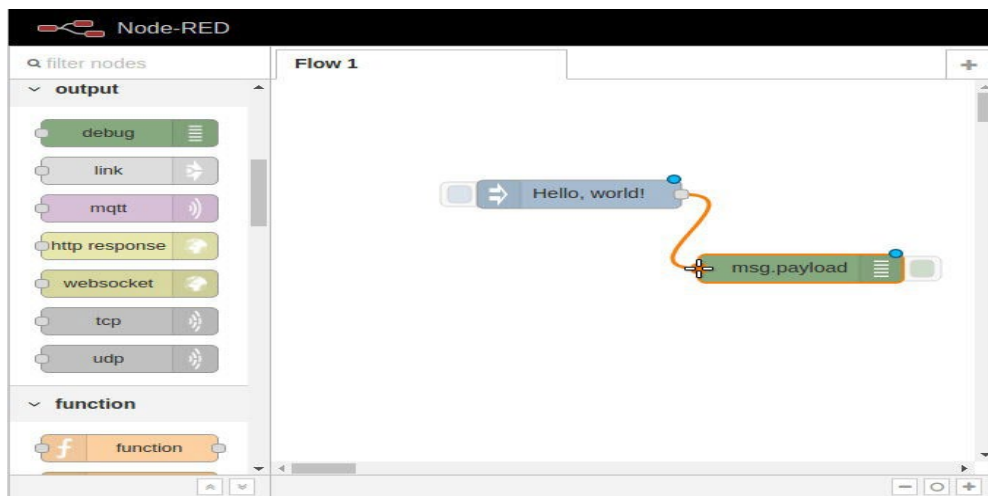
will drag it onto our flow.

4. In order to edit the node, we will double click on it. After that, we will select *string* on the Payload field, and we will write Hello world!
5. Once we finish the previous steps, we click Done.



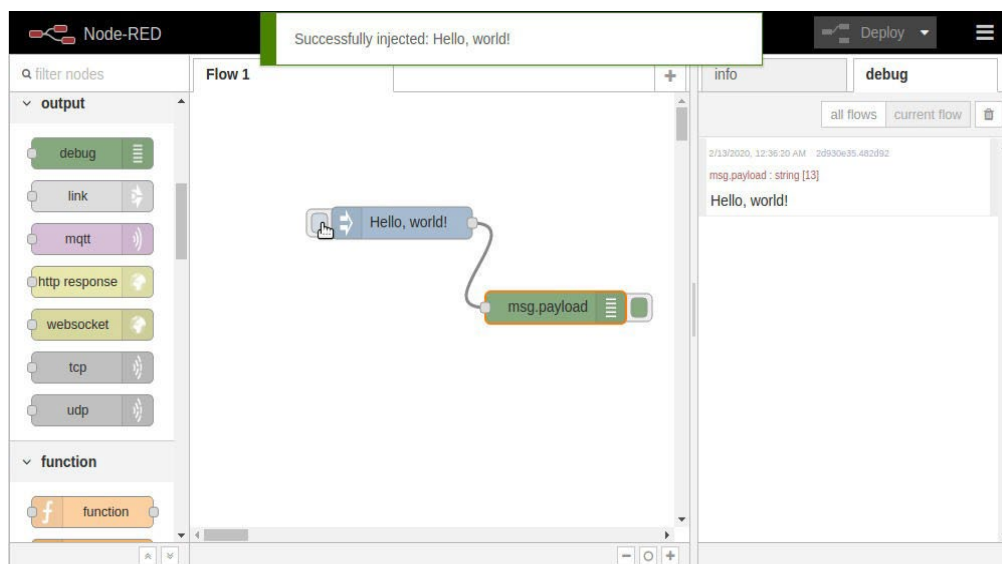
Printing our message

1. To add a destination for our message we will select the *Debug* node. We will click and drag it onto our flow. It's very important to place it on the right-hand side of the *Inject* node.
2. To connect both nodes we just need to click the *Inject* node's output and drag it to the *Debug* node's input. A wire that links both nodes will be created
3. The debug node will automatically print the message to the console window as we will see in the next step.



Deployment

1. In order to start our application, you must click on *Deploy* button
2. Now click the *Debug* tab in the right-hand side of the editor window
3. Finally, to launch our message you will click on the blue button coming out from the left-hand side of the *Inject* node
4. “Hello world” will appear on the Debug screen.



Practical-10

Aim: Study and implementation of processing data from different sensors and visualize data on Node-red dashboard.

The pre-installed Node-RED service and Dashboard plugin on the gateway will be used to create the web visualization.

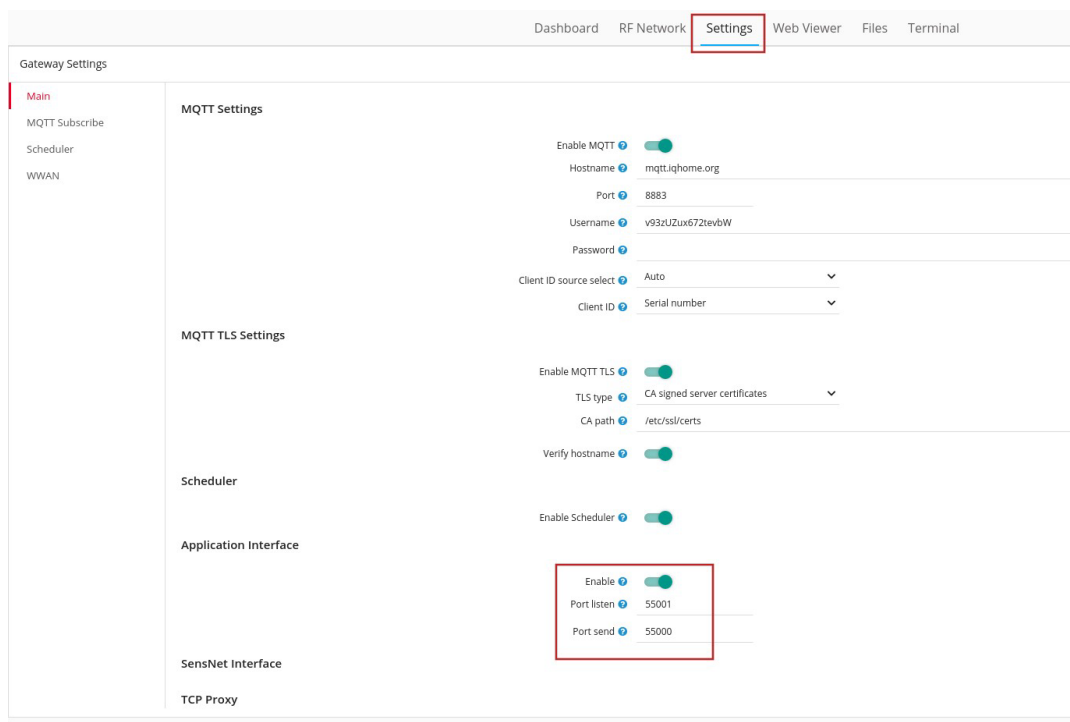
What you need:

- IQ Home Gateway
- IQ Home Sensor

Enable the Application Interface

To collect sensor data with Node-RED, first, we need to enable the Application Interface feature on the gateway.

- Connect to the Gateway using the “**Link It!**” Software
- Go to the “**Settings**” tab
- Enable “**Application Interface**” and set a “**Port Send**” value (e.g. 55000)



Gateway Settings

Dashboard RF Network **Settings** Web Viewer Files Terminal

Main

MQTT Settings

MQTT Subscribe

Scheduler

WWAN

Enable MQTT ☒

Hostname mqtt.lqhome.org

Port 8883

Username y93zUZux672tevbW

Password

Client ID source select Auto

Client ID Serial number

MQTT TLS Settings

Enable MQTT TLS ☒

TLS type CA signed server certificates

CA path /etc/ssl/certs

Verify hostname ☒

Scheduler

Enable Scheduler ☒

Application Interface

Enable ☒

Port listen 55001

Port send 55000

SensNet Interface

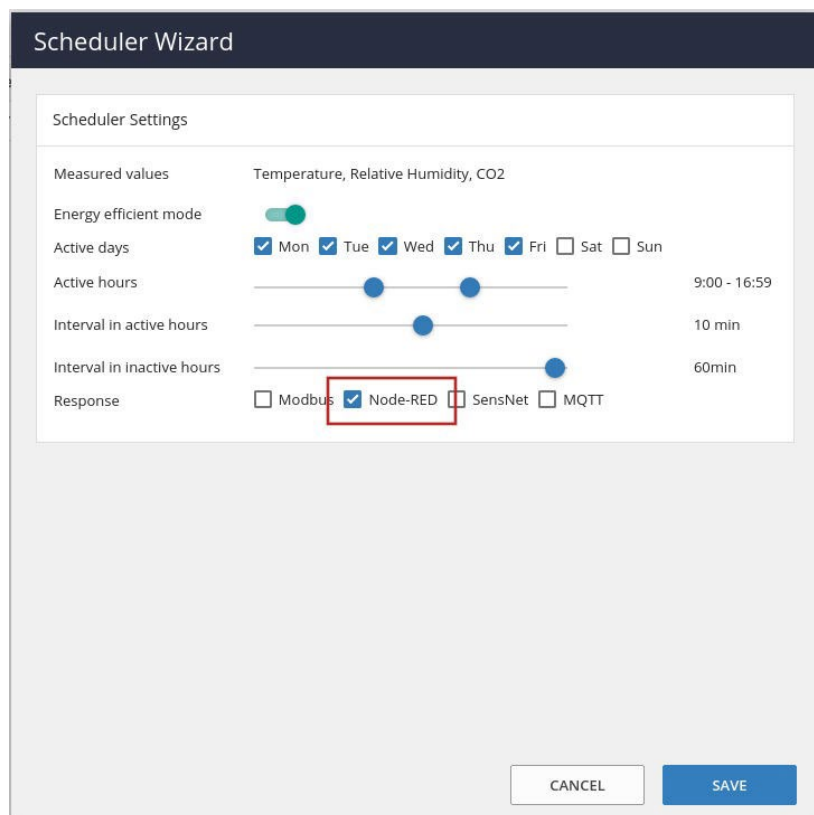
TCP Proxy

The **Port Send** value selected here will be used in Node-RED.

Then enable the Node-RED scheduler for the sensors.

1. Open the “RF Network” tab.
2. Switch to “Sensor Data”
3. Click on the clock icon in the top right corner labeled “Create Scheduler”.
4. Enable the “**Node-RED**” Response option.

You can also set the time intervals between the sensor measurements



The image shows a 'Scheduler Wizard' dialog box with the following settings:

- Scheduler Settings**
- Measured values:** Temperature, Relative Humidity, CO2
- Energy efficient mode:** ☒
- Active days:** ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☐ Sat ☐ Sun
- Active hours:** 9:00 - 16:59
- Interval in active hours:** 10 min
- Interval in inactive hours:** 60min
- Response:** ☐ Modbus ☒ Node-RED ☐ SensNet ☐ MQTT

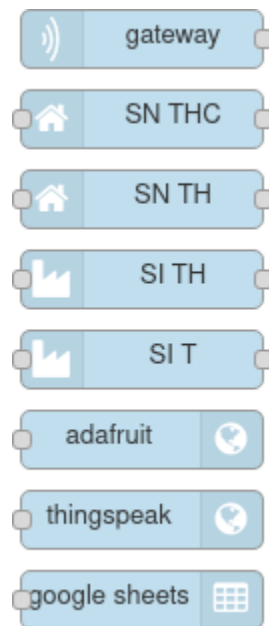
Buttons: CANCEL, SAVE

Set up a Node-RED network to forward the sensor data

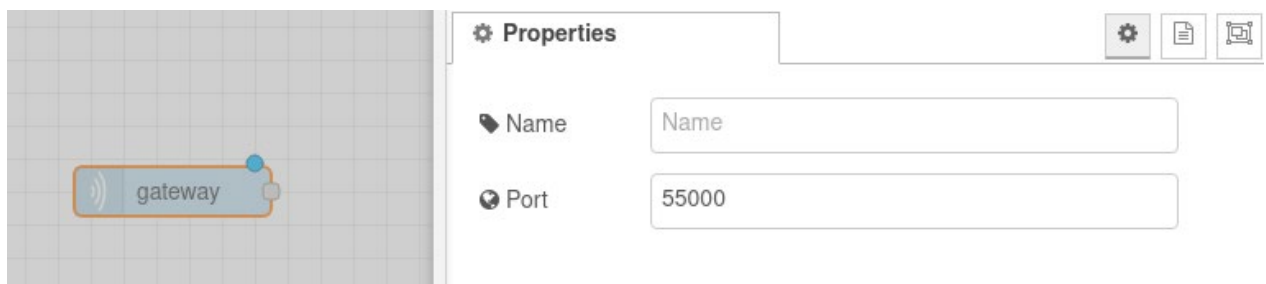
we will be using a Temperature Sensor [SI-T-02/SC] and a Temperature and Relative Humidity Sensor [SN-TH-02].

1. Switch to the **Node-RED** tab in **LinkIt!**

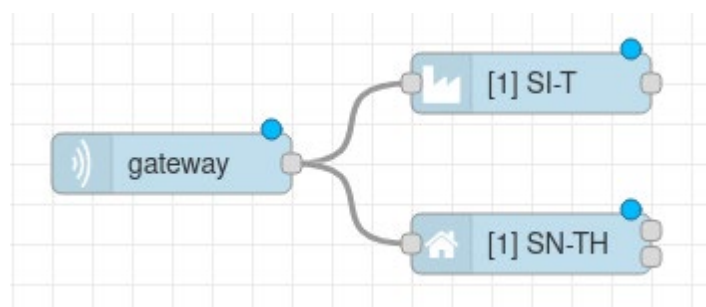
The **IQ Home** nodes can be found in the bottom of the panel on the left side of your screen.



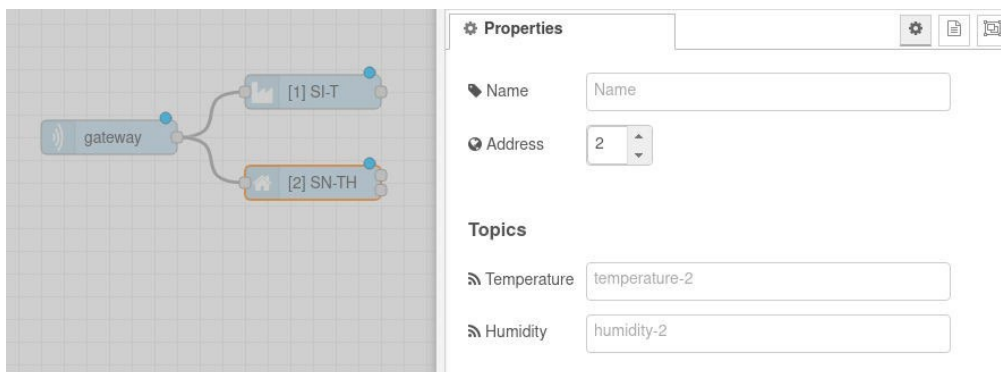
2. Add an iqhome **gateway** node. If you changed the used port in the first step, you can set it here by double-clicking on the node



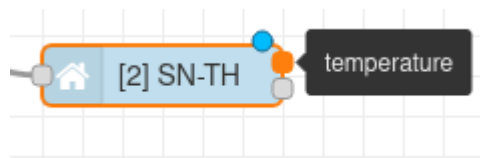
3. Add the sensor nodes corresponding to the sensors you are using in your IQHome network. In this example, we are using the **SI-T-02/SC** and **SN-TH-02** sensors, so we will add the **SI-T** and **SN-TH** nodes.



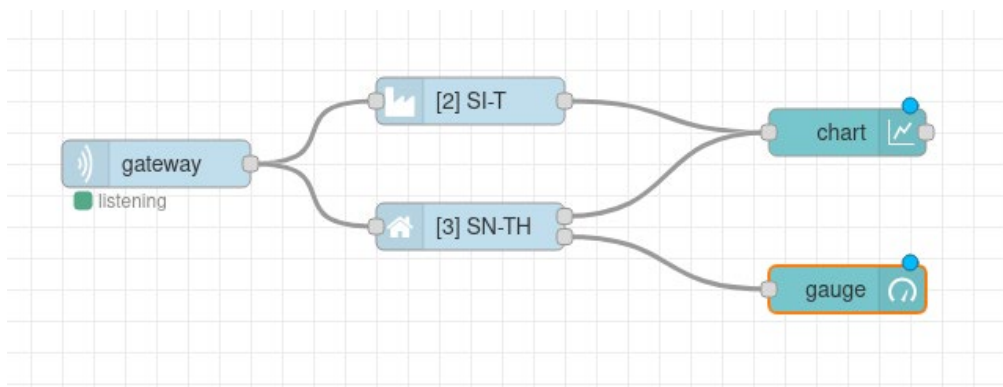
4. Set the **device addresses** corresponding to your sensor's addresses as seen in the **LinkIt! RF Network** tab by double-clicking the sensor nodes. We are using the default topic (feed) names generated by the sensor nodes, so you can leave the boxes under "Topics" empty. If you used a different topic (feed) name on the Adafruit website, you have to write the same topic names here.



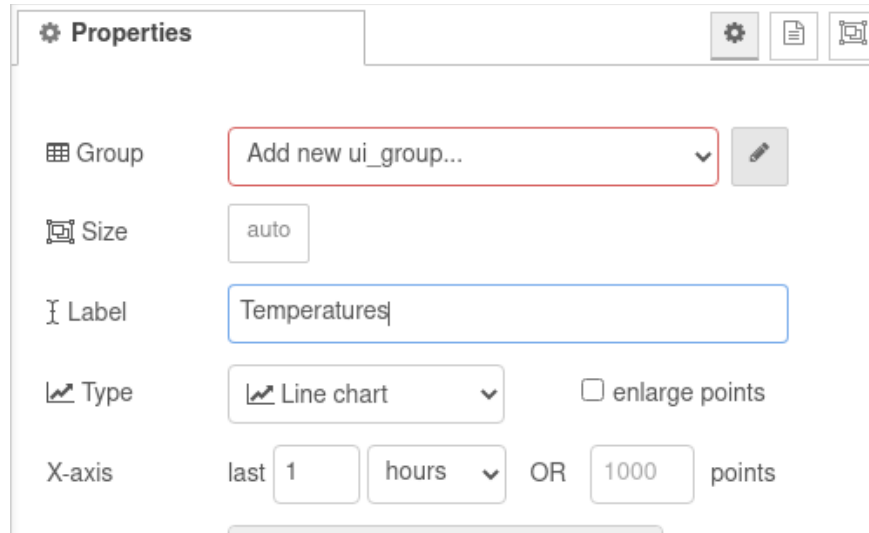
Each sensor node has outputs depending on what types of measurements can that sensor make.



5. For this demo, we are going to create a line chart for the two temperature values and a gauge for the humidity. Add a **gauge** and a **chart** node and connect them to the corresponding sensor outputs.



6. Configure the **chart** node by double-clicking on it. Give a label to the chart, then click on the **pencil icon** next to **Add new ui_group...**



Properties

Group: Add new ui_group...

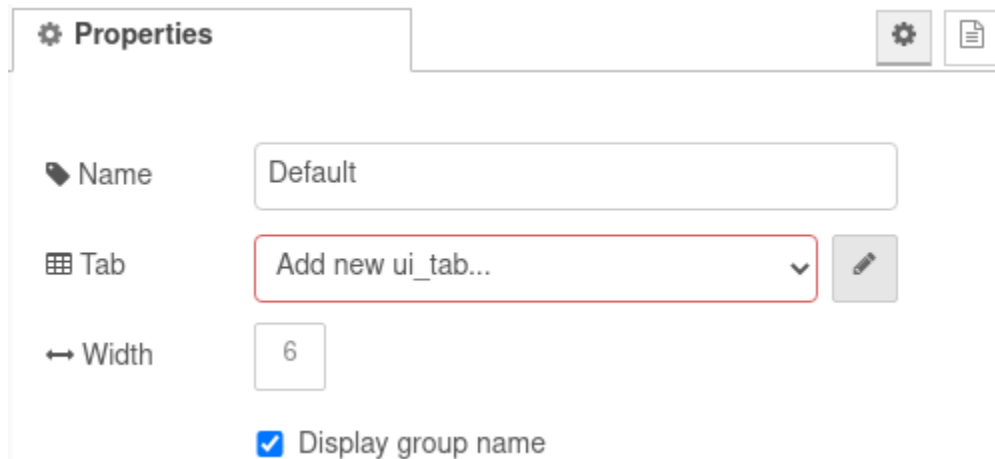
Size: auto

Label: Temperatures

Type: Line chart ☐ enlarge points

X-axis: last 1 hours OR 1000 points

7. Then click on the **pencil icon** next to **Add new ui_tab...** and click the red **Add** button twice.



Properties

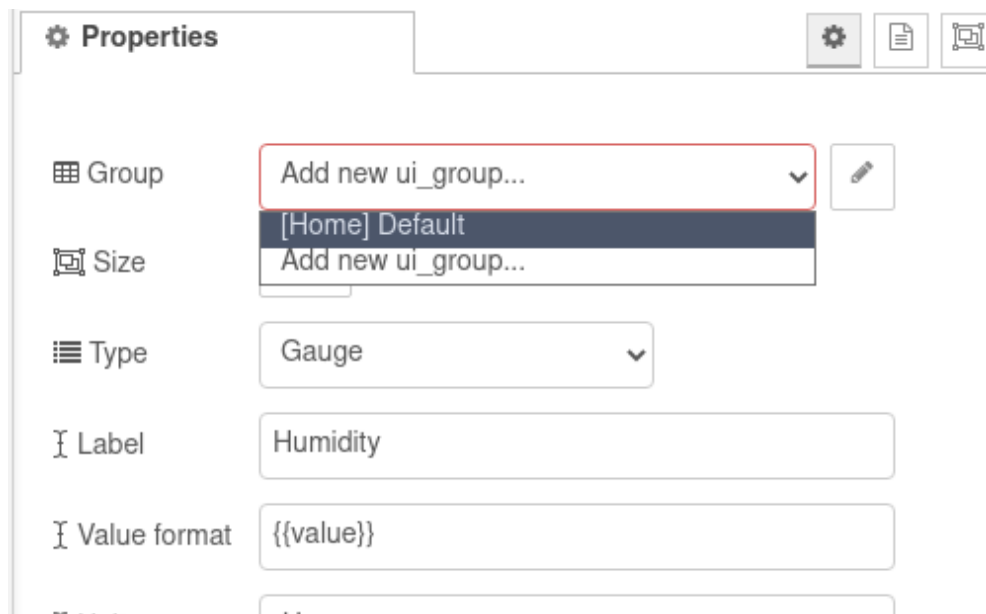
Name: Default

Tab: Add new ui_tab...

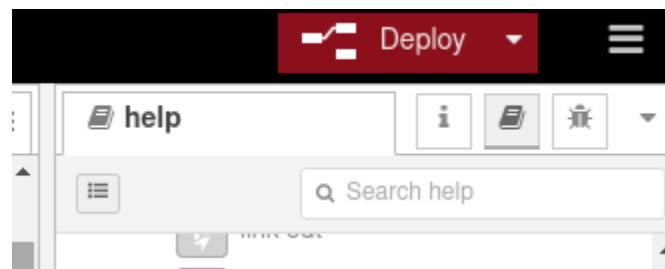
Width: 6

☒ Display group name

8. Configure the **gauge** node by double-clicking on it. Select the **Group** created in the previous step, and give a **label** to the gauge.



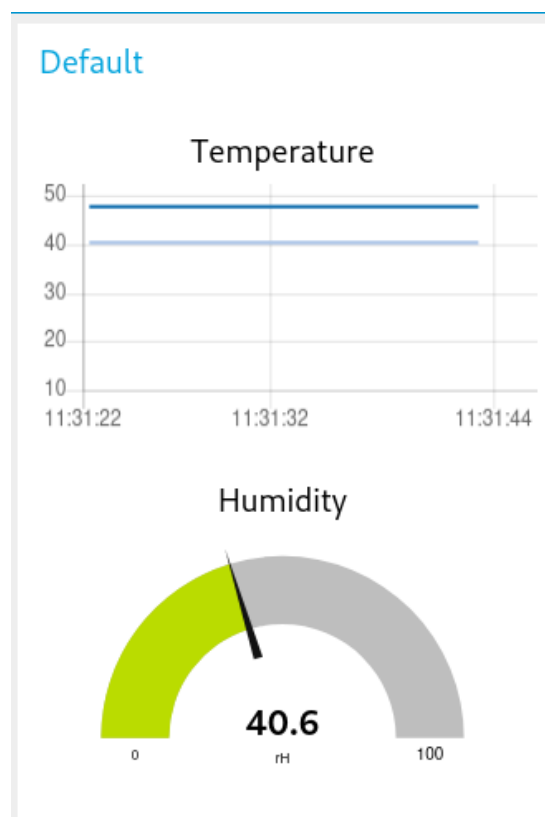
9. **Deploy** your Node-RED network by clicking the **Deploy** button in the top right corner of your window



10. Open the **Node-RED Dashboard** by switching to the **Dashboard** tab in **LinkIt!**, and clicking on the link next to **IPv4**

Dashboard RF Network Setl	
Gateway Information	
Serial	
Hostname	
Model	GW-IND-01
Interface	wlan0
MAC	
IPv4	192.168.6.249
IPv6	
RF MID	
Uptime	19 hours 1 minutes
Firmware	Up to date

11. If you turn on your IQHome gateway and sensors, you will see the incoming data in your Node-RED Dashboard:



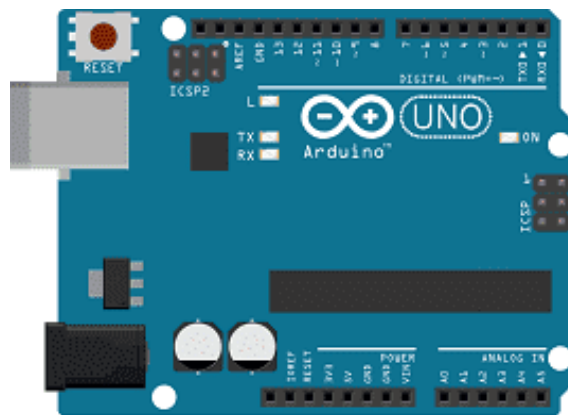
Practical-11

Aim: Write a sketch that will upload data (like temperature, Light status, etc) on thingspeak cloud.

- Thanks to the Arduino board and the ESP8266 module connected to the Internet of Things (IoT: Internet of Things), it is possible to monitor in real time the temperature and humidity measured by the DTH11 sensor.
- To create sensor data recording applications, you can use ThingSpeak which is an API and an open source application for the Internet of Things, allowing you to store and collect data from connected objects.

Necessary components

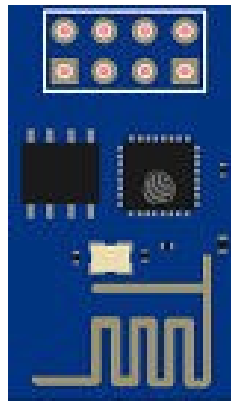
Arduino UNO



The Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. The Arduino UNO is programmed using the Arduino software (IDE), which is available for Windows, Mac, and Linux. The board can be powered by an external power supply or by USB. It is compatible with a wide range of sensors, actuators, and other devices, which can be connected to it using a variety of communication protocols, such as I2C, SPI, and UART. The Arduino UNO is a popular choice for beginners and hobbyists, as it is easy to use and

has a large community of users and developers.

ESP8266 module



An ESP8266 module can be used in combination with an Arduino board to send data to ThingSpeak. The ESP8266 is a low-cost Wi-Fi module that can be used to connect an Arduino to the internet. It can be programmed using the Arduino IDE, which allows you to write sketches that can read sensor data, connect to a Wi-Fi network, and send data to ThingSpeak using its API.

To use an ESP8266 module with an Arduino, you will need to connect the ESP8266 to the Arduino's serial pins (RX, TX) and power pins (VCC, GND). Then, you can upload a sketch to the Arduino that will configure the ESP8266 to connect to your Wi-Fi network and send data to ThingSpeak.

Once the ESP8266 is connected to Wi-Fi and ThingSpeak, it can send data from various sensors connected to the Arduino. This data can be then visualized on ThingSpeak's website, or can be used to trigger other actions or notifications.

DHT11 sensor



The DHT11 sensor is a low-cost humidity and temperature sensor that can be used with an Arduino board. The sensor has four pins: VCC, GND, data out and NC. The VCC and GND pins are used to power the sensor, while the data out pin is used to send the sensor data to the Arduino. The NC is not connected.

To use the DHT11 sensor with an Arduino, you will need to connect the VCC pin to the 5V pin on the Arduino, the GND pin to a GND pin on the Arduino, and the data out pin to a digital pin on the Arduino (for example, pin 2).

Once the sensor is connected, you can use the Arduino's DHT library to read the sensor data. The library provides functions that can be used to read the temperature and humidity from the sensor.

Connecting wires



When connecting wires to an Arduino or other electronic devices, it's important to ensure that the wires are connected to the correct pins and in the correct orientation. Typically, the wires are connected to the power, ground, and input/output (I/O) pins on the device. The power and ground pins provide a source of power for the device, while the I/O pins are used to send and receive data.

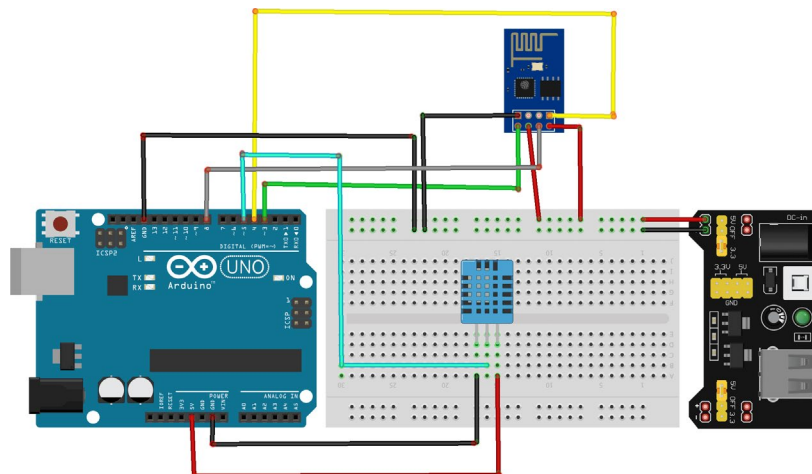
Mounting:

To perform the assembly

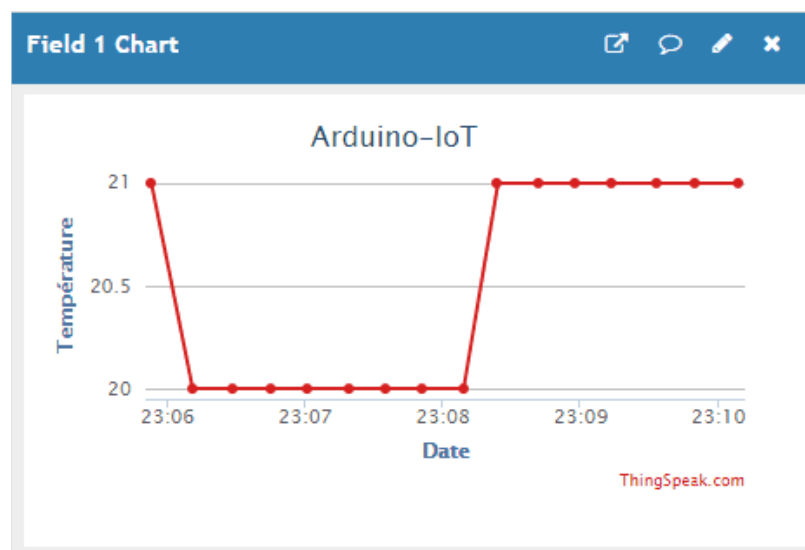
For ESP8266 module:

- The RX pin to pin 4 of the Arduino board
- The TX pin to pin 3 of the Arduino board

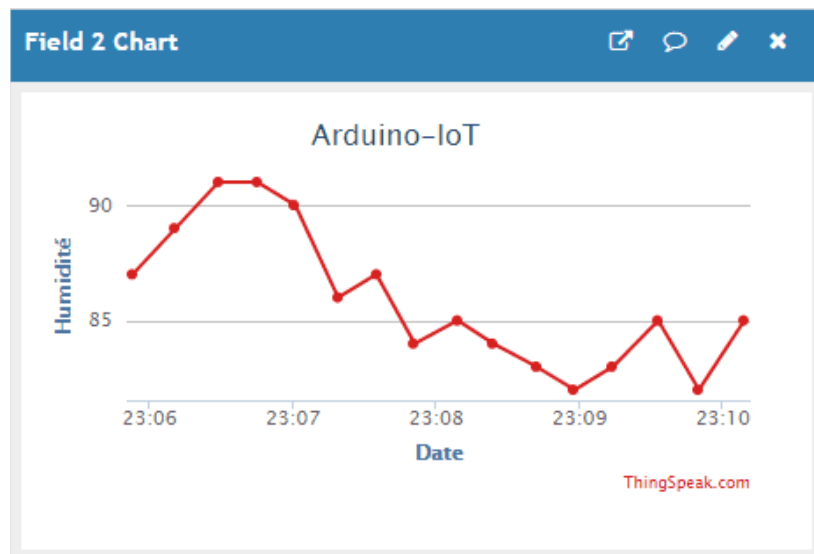
- The GND pin to the GND of the Arduino board
- The two pins 3V3 and EN to the 5V pin of the power supply module
- The RST pin to pin 8 of the Arduino board
- The two pins 3V3 and EN to the 5V pin of the power supply module
- For DHT11 sensor:
- the DATA pin to pin N ° 5 of the Arduino board
- the VCC pin to the 3.3V pin of the Arduino board
- the GND pin to the GND pin of the Arduino board



The temperature values sent by the Arduino board to the thinkspeak.com site



The humidity values sent by the Arduino board to the thingspeak.com site



Program:

```

#include <Adafruit_ESP8266.h>
#include <SoftwareSerial.h>

#define ESP_RX 3
#define ESP_TX 4
#define ESP_RST 8

SoftwareSerial softser(ESP_RX, ESP_TX);
// Must declare output stream before Adafruit_ESP8266 constructor; can be
// a SoftwareSerial stream, or Serial/Serial1/etc. for UART.
Adafruit_ESP8266 wifi(&softser, &Serial, ESP_RST);
// Must call begin() on the stream(s) before using Adafruit_ESP8266 object.
#define ESP_SSID "HUAWEI Y5 2019"
// Your network name here #define ESP_PASS "b582058c4d86"
// Your network password here
String API = "KM7MYIGX8G2X6GA0";
// CHANGE ME
#define HOST "api.thingspeak.com"
// Find/Google your email provider's SMTP outgoing server name for unencrypted email
  
```

```
#define PORT 80

// Find/Google your email provider's SMTP outgoing port for unencrypted email
String field = "field1";

int count = 0;

// we'll use this int to keep track of which command we need to send next
bool send_flag = false;

// we'll use this flag to know when to send the email commands
#include <dht11.h>

#define DHT11PIN 5

// broche DATA -> broche 4 dht11;

void setup()
{
char buffer[50];

// This might work with other firmware versions (no guarantees)
// by providing a string to ID the tail end of the boot message:
// comment/replace this if you are using something other than v 0.9.2.4!
wifi.setBootMarker(F("Version:0.9.2.4\r\n\r\nready"));
softser.begin(9600);

// Soft serial connection to ESP8266
Serial.begin(57600);
while(!Serial);

// UART serial debug
// Test if module is ready
Serial.print(F("Hard reset..."));
if(!wifi.hardReset())
{
    Serial.println(F("no response from module."));
    for(;;);
}
```



```
Serial.println(F("OK."));
Serial.print(F("Soft reset..."));
if(!wifi.softReset())
{
    Serial.println(F("no response from module."));
    for(;;);
}
Serial.println(F("OK."));
Serial.print(F("Checking firmware version..."));
wifi.println(F("AT+GMR"));
if(wifi.readLine(buffer, sizeof(buffer)))
{
    Serial.println(buffer);
    wifi.find();
    // Discard the 'OK' that follows
}
else
{
    Serial.println(F("error"))
    Serial.print(F("Connecting to WiFi..."));
    f(wifi.connectToAP(F(ESP_SSID), F(ESP_PASS)))
    // IP addr check isn't part of library yet, but
    // we can manually request and place in a string.
    Serial.print(F("OK\nChecking IP addr..."));
    wifi.println(F("AT+CIFSR"));
    if(wifi.readLine(buffer, sizeof(buffer)))
    {
        Serial.println(buffer);
        wifi.find();
        // Discard the 'OK' that follows
```

```
Serial.print(F("Connecting to host...")) ;
Serial.print("Connected..");
wifi.println("AT+CIPMUX=0");
// configure for single connection,
//we should only be connected to one SMTP server wifi.find();
wifi.closeTCP();
// close any open TCP connections
wifi.find();
Serial.println("Type \"send it\" to send an email");
}
else
{
    // IP addr check failed
    Serial.println(F("error"));
}
}
else
{
    // WiFi connection failed
    Serial.println(F("FAIL"));
void loop()
{
    send_flag = true;
    if(send_flag)
    {
        // the send_flat is set, this means we are or need to start sending SMTP commands
        if(do_next())
        {
            // execute the next command
            count++;
            // increment the count so that the next command will be executed next time.
```

```
    }  
  }  
}  
boolean do_next()  
{  
    switch(count)  
    {  
        case 0:  
            Serial.println("Connecting...");  
            return wifi.connectTCP(F(HOST), PORT);  
            break;  
        case 1:  
            DHT11.read(DHT11PIN);  
            char charVal2[100];  
            char charVal1[100]="GET /update?api_key=KM7MYIGX8G2X6GA0&field1=";  
            //itoa(sensor, charVal2, 10);  
            dtostrf((float)DHT11.temperature, 4, 2, charVal2);  
            strcat(charVal1,charVal2);  
            strcat(charVal1,"&field2=");  
            dtostrf((float)DHT11.humidity, 4, 2, charVal2);  
            strcat(charVal1,charVal2);  
            return wifi.cipSend(charVal1);  
            //Sending data to the Thinkspeak  
            sitedelay(60000);  
    }  
}
```

Practical-12

Aim: Case Study on IoT Application.

- **Home Automation:** This home automation system based on IoT automates the functioning of household appliances over the Internet.

Introduction

In recent years, the Internet of Things (IoT) has revolutionized the way we interact with our homes. Home automation systems powered by IoT technology allow homeowners to control and monitor various household appliances and systems remotely, utilizing internet connectivity. These systems enhance convenience, energy efficiency, safety, and overall quality of life. This case study explores the architecture, components, functionality, benefits, and challenges associated with implementing IoT in home automation. By integrating IoT technologies, homeowners can remotely control and monitor various appliances, devices, and systems within their homes.

Selection of Components

The implementation process began with the selection of appropriate components and devices for the home automation system. This involved researching and evaluating different sensors, actuators, communication protocols, and the central control system. Factors such as compatibility, reliability, and ease of integration were considered during the selection process. It is an important phase as the correct selection of components will lead to the better home automation system. During this process we should also consider its disadvantages and we should try to overcome it for better performance.

Installation and Integrity

Once the components were chosen, the installation and integration phase began. Sensors were strategically placed throughout the house to capture environmental data, including temperature, humidity, light intensity, and occupancy. Actuators were installed to control various appliances, such as lights, thermostats, and door locks. The central control system was set up and configured to receive data from sensors, process user commands, and communicate with the actuators. A secure communication network, such as Wi-Fi, was established to enable seamless connectivity

between the components.

Methodologies

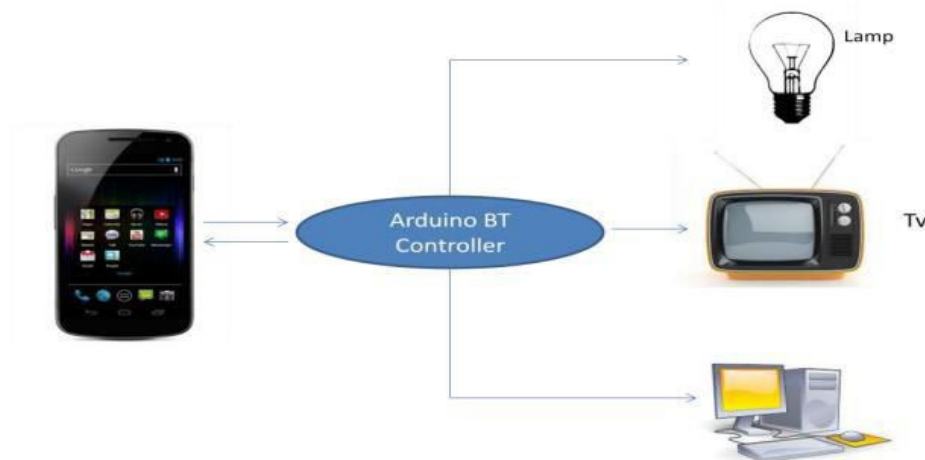
1) GSM based Home Automation System

It provides 3 means to control the home: the GSM network, the Internet and through speech. The real time monitoring has been an important feature that can be used in the home automation systems. As a change in the status of the devices occurs, the user can be informed in real time. The user commands are transferred to a server which is usually done by a PC. The server processes the user commands and sends them to the relevant units. This can help control the appliances. GSM is used as a communication medium to help establish connection in places where there may not be proper internet connectivity. The server uses AT commands to communicate with the GSM modem. The mobile interface is developed using J2ME. The server has 4 engines running – the web server, database, main control program and speech recognition program. The system can be controlled using SMS. It can send confirmation messages. Speech processing is done with a dynamic time wrapping algorithm. The voice activation has been tested and found to be too impractical. As a more stable alternative, the voice input can be activated through a wireless unit the user carries along in the house. Each application node has four parts – the transmitter, receiver, I/O device and a microcontroller. The main control program in the server takes status information from the devices' transceiver in real time.

2) Bluetooth Based Home Automation

It makes use of a cell phone and Bluetooth technology. Bluetooth technology is secured and low cost. It makes use of an Arduino Bluetooth board. An interactive python program is used in the cell phone to provide the user interface. The I/O ports of the Bluetooth board and relays are used for interfacing with the devices which are to be controlled. The Bluetooth is password protected to ensure that the system is secure and not misused by any intruders. The Bluetooth has a range of 10 to 100 meters, 2.4 GHz bandwidth and 3Mbps speed. The python app on the phone is portable. It is also a fast and cost effective system. There is a diagnostic system that can detect problems in the circuitry. A feedback system will report status of devices after every signal toggle. The main drawback with respect to Bluetooth is that it takes a long time to discover and access devices in its vicinity. It does not provide energy conservation tips. Real time access cannot be achieved. Anywhere access to the devices cannot be achieved. Access is

limited to within the Bluetooth range.



Beside from these 2 methodology there are more methods through which we can automate our home they are through wireless system, through ZigBee and through mixed type.

User Experience and Benefits

After the installation and setup, homeowners experienced several benefits through the IoT-based home automation system. They could conveniently control and monitor their home from anywhere using their smartphones. For example, they could turn on lights and adjust the thermostat before arriving home, ensuring a comfortable environment upon their arrival. The system provided energy-saving recommendations based on real-time data analysis, helping homeowners optimize their energy consumption. Users could set schedules for appliances, such as turning off lights during daytime or adjusting the thermostat based on occupancy, leading to reduced energy bills. The integrated security features allowed homeowners to remotely monitor surveillance cameras and receive instant alerts in case of any security breaches. They could also control door locks remotely, adding an extra layer of safety and convenience. The integration with voice assistants further enhanced the user experience, enabling homeowners to control appliances using voice commands. This hands-free interaction improved accessibility and convenience within the home.

Challenges

During the implementation process, several challenges were encountered. Interoperability issues between different IoT devices and protocols required thorough testing and troubleshooting to

ensure seamless integration. Security and privacy concerns demanded robust measures to safeguard user data and prevent unauthorized access. The cost of components, installation, and maintenance posed a financial challenge, requiring careful budgeting and planning. Additionally, ongoing maintenance and firmware updates were necessary to ensure the system's reliability and performance. Lessons learned from this implementation include the importance of thorough research and testing when selecting components, addressing security concerns proactively, and considering long-term scalability and maintenance requirements.

Conclusion

In conclusion, IoT-based home automation systems have revolutionized the way we interact with our homes. These systems empower users to control and monitor household appliances and devices remotely via the internet, offering convenience, energy efficiency, and enhanced security. With the ability to create automation routines and integrate with voice assistants, these systems provide a seamless and customizable smart home experience. However, it's crucial for users to prioritize security measures to protect their data and privacy while enjoying the benefits of home automation. As technology continues to advance, the possibilities for IoT-based home automation are expected to expand further, making our homes even smarter and more efficient.