KAMMAVARI SANGHAM

K.S.INSTITUTE OF TECHNOLOGY

(APPROVED BY A.I.C.T.E AFFILIATED TO VTU BELGAUM)

#14, RAGHUVANAHALLI, KANAKAPURA MAIN ROAD,BANGALORE-560109

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGG.

NAME OF THE LAB: IOT (INTERNET OF THINGS) LAB

COURSE CODE: 21EC581

K. S. INSTITUTE OF TECHNOLOGY

VISION

"To impart quality technical education with ethical values, employable skills and research to achieve excellence"

MISSION

- To attract and retain highly qualified, experienced & committed faculty.
- To create relevant infrastructure.
- Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence.
- To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society.

K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

VISION:

"To achieve excellence in academics and research in Electronics & Communication Engineering to meet societal need".

MISSION:

- To impart quality technical education with the relevant technologies to produce industry ready engineers with ethical values.
- To enrich experiential learning through active involvement in professional clubs & societies.
- To promote industry-institute collaborations for research & development.

K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO1 : Excel in professional career by acquiring domain knowledge.

PEO2 : Motivation to pursue higher Education & research by adopting technological innovations by continuous learning through professional bodies and clubs.

PEO3 : To inculcate effective communication skills, team work, ethics and leadership qualities.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO1: Graduate should be able to understand the fundamentals in the field of Electronics & Communication and apply the same to various areas like Signal processing, embedded systems, Communication & Semiconductor technology.

PSO2: Graduate will demonstrate the ability to design, develop solutions for Problems in Electronics & Communication Engineering using hardware and software tools with social concerns.

K S INSTITUTE OF TECHNOLOGY

PROGRAM OUTCOMES (PO'S)

Engineering Graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

K S INSTITUTE OF TECHNOLOGY, BENGALURU

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

RUBRICS for Evaluation in Laboratories - 2021 Scheme

Continuous Internal Evaluation (CIE) - 50 Marks = 30+20			
Record, Observation & Viva : 30 Marks			
Record	**Evaluation criteria**		
	Good	**Average**	**Poor**
10 Marks	The Record meets all aspects of assessment-Timeliness, contents, correctness, completeness & neatness.	The Record partially meets all aspects of assessment-Timeliness, contents, correctness, completeness & neatness.	The Record written poorly, does not meet all aspects of assessment-Timeliness, contents, correctness, completeness & neatness.
	9 to 10 Marks	**5 to 8 Marks**	**0 to 4 Marks**
Observation & Conduction	**Evaluation criteria**		
	Good	**Average**	**Poor**
15 Marks	The Observation meets all aspects of assessment-Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is satisfactory.	The Observation partially meets all aspects of assessment-Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is partially satisfactory.	The Observation poorly written and does not meet all aspects of assessment-Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is not satisfactory.
	10 to 15 Marks	**5 to 9 Marks**	**0 to 4 Marks**
Viva	**Evaluation criteria**		
5 Marks	All questions answered Correctly	Answers are partially correct	Poorly answered
	5 Marks	**3 to 4 Marks**	**0 to 2 Marks**
Test - 10 + 10 = 20 Marks (two tests)			
Write-up: 20% of maximum marks		**Conduction**: 40% of maximum marks	**Viva**: 40% of maximum marks

Note: Each test should be conducted for 100 marks and reduce to 10

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI
B.E: Electronics & Communication Engineering / B.E: Electronics & Telecommunication Engineering NEP, Outcome Based Education (OBE) and Choice Based Credit System (CBCS)
(Effective from the academic year 2021 – 22)

V Semester

IoT (Internet of Things) Lab			
Course Code	21EC581	CIE Marks	50
Teaching Hours/Week (L: T:P: S)	0:0:2:0	SEE Marks	50
Credits	1	Exam Hours	03

Course objectives:

- To impart necessary and practical knowledge of components of Internet of Things
- To develop skills required to build real-life IoT based projects.

Sl.No	**Experiments**
1	i) To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to 'turn ON' LED for 1 sec after every 2 seconds. ii) To interface Push button/Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program to 'turn ON' LED when push button is pressed or at sensor detection.
2	i) To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings. ii) To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.
3	To interface motor using relay with Arduino/Raspberry Pi and write a program to 'turn ON' motor when push button is pressed.
4	To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.
5	To interface Bluetooth with Arduino/Raspberry Pi and write a program to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.
6	Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to thingspeak cloud.
7	Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from thingspeak cloud.
8	To install MySQL database on Raspberry Pi and perform basic SQL queries.
9	Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker.
10	Write a program to create UDP server on Arduino/Raspberry Pi and respond with humidity data to UDP client when requested.
11	Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.
12	Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperature data and print it.

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

1. Understand internet of Things and its hardware and software components
2. Interface I/O devices, sensors & communication modules
3. Remotely monitor data and control devices
4. Develop real life IoT based projects

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course is 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% Marks allotted to the procedure part to be made zero.

The duration of SEE is 03 hours

Rubrics suggested in Annexure-II of Regulation book

Suggested Learning Resources:

1. Vijay Madisetti, Arshdeep Bahga, Internet of Things. "A Hands on Approach", University Press
2. Dr. SRN Reddy, Rachit Thukral and Manasi Mishra, "Introduction to Internet of Things: A practical Approach", ETI Labs
3. Pethuru Raj and Anupama C Raman, "The Internet of Things: Enabling Technologies, Platforms, and Use Cases", CRC Press
4. Jeeva Jose, "Internet of Things", Khanna Publishing House, Delhi
5. Adrian McEwen, "Designing the Internet of Things", Wiley
6. Raj Kamal, "Internet of Things: Architecture and Design", McGraw Hill

Course Outcomes:

The students will be able to

CO1: **Make use of** Arduino IDE to interface various sensors, I/O devices & Communication modules.	Applying (K3)
CO2: **Apply** IoT concepts to organise remotely monitored data & control devices using Thingspeak cloud.	Applying (K3)
CO3: **Make use of** MQTT to publish & subscribe sensor data.	Applying (K3)
CO4: **Build** TCP & UDP servers on Arduino & communicate with corresponding clients with sensor data on client request.	Applying (K3)
CO5: **Build** MySQL database on Raspberry Pi & make simple queries.	Applying (K3)

CONTENTS

EXPERIMENT – 1

i. To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to turnON LED for 1 sec after every 2 seconds.

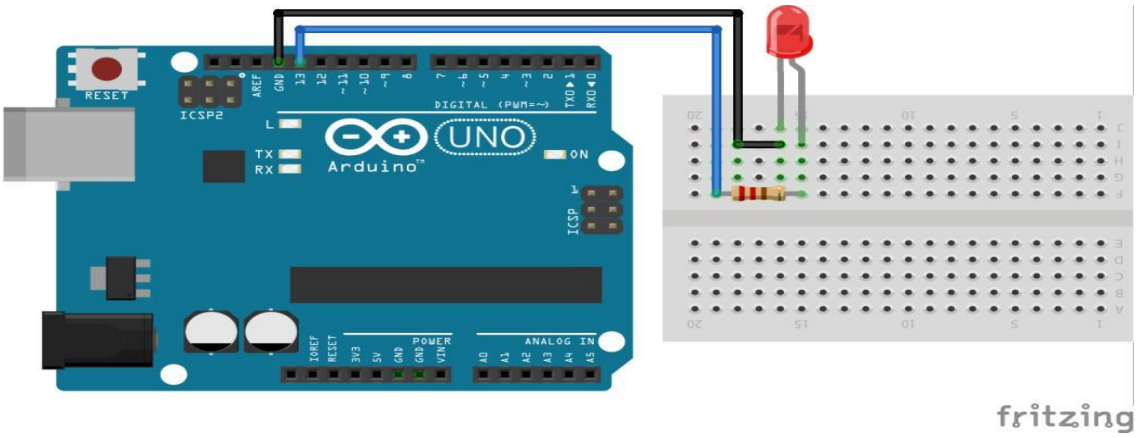
AIM: Write a program to turn ON LED/ Buzzer for 1 sec after every 2 seconds with Arduino/ Raspberry Pi.

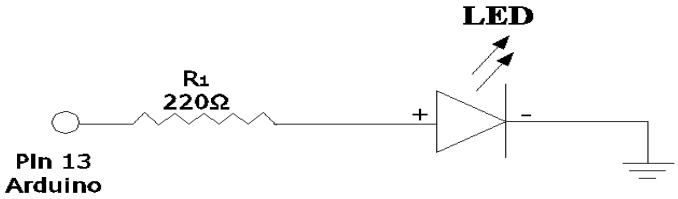
COMPONENTS REQUIRED:

1. Arduino UNO/ Raspberry Pi.
2. LED/ Buzzer.
3. Resistor (220Ω)
4. Connecting cable or USB cable.
5. Breadboard.
6. Jumper wires

CONNECTION DIAGRAM:

(a) LED (b) Buzzer

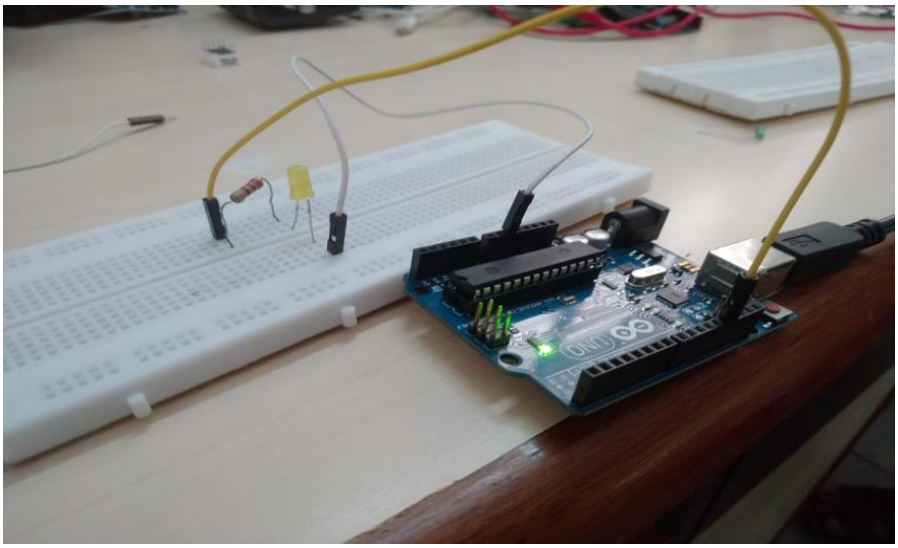




CODING:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(2000);                      // wait for a second
}
```

RESULT: LED/Buzzer is successfully controlled by Arduino UNO



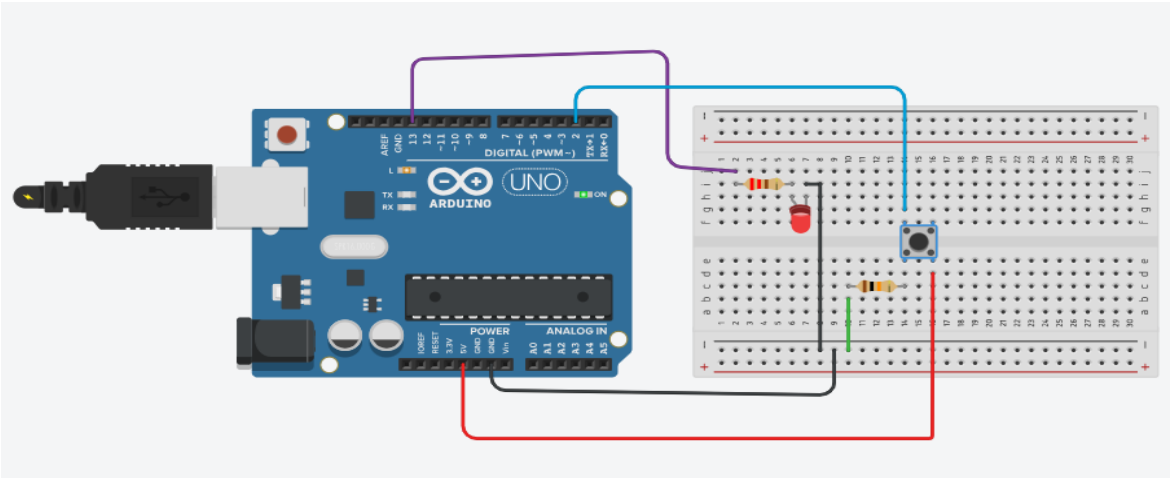
ii. To interface Push button/Digital sensor (IR/LDR) with Arduino UNO and write a program to turn ON LED when push button is pressed or at sensor detection.

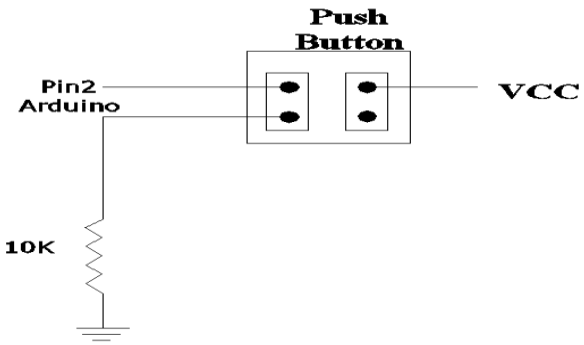
AIM: Write a program to turn ON LED when push button is pressed using Arduino UNO.

COMPONENTS REQUIRED:

1. Arduino UNO.
2. Push button.
3. LED.
4. Resistor (10KΩ, 220Ω)
5. Connecting cable or USB cable.
6. Breadboard.
7. Jumper wires.

CONNECTION DIAGRAM:

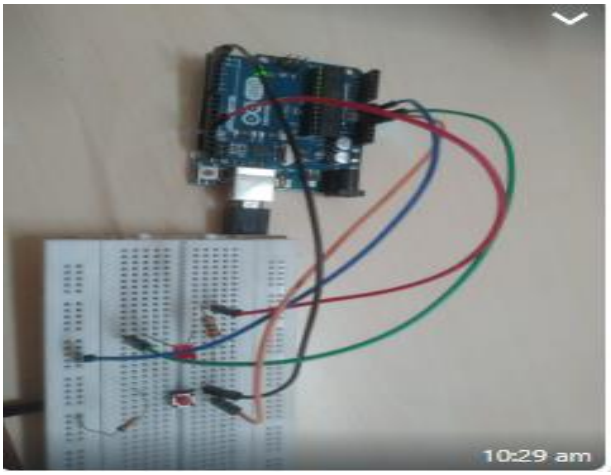




CODING:

```
int PushButtonPin = 2;
int OutputPin = 13;

void setup() {
  pinMode(OutputPin, OUTPUT);
  pinMode(PushButtonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  int SensorValue = digitalRead(PushButtonPin);


  Serial.print("PushButtonPin Value: ");
  Serial.println(SensorValue);
  delay(100);
   if (SensorValue==LOW){ // LOW MEANS Object Detected
    digitalWrite(OutputPin, HIGH);
  }
  else
  {
    digitalWrite(OutputPin, LOW);
  }
}
```

RESULT: Controlling LED by pressing push button using Arduino UNO has successfully executed.

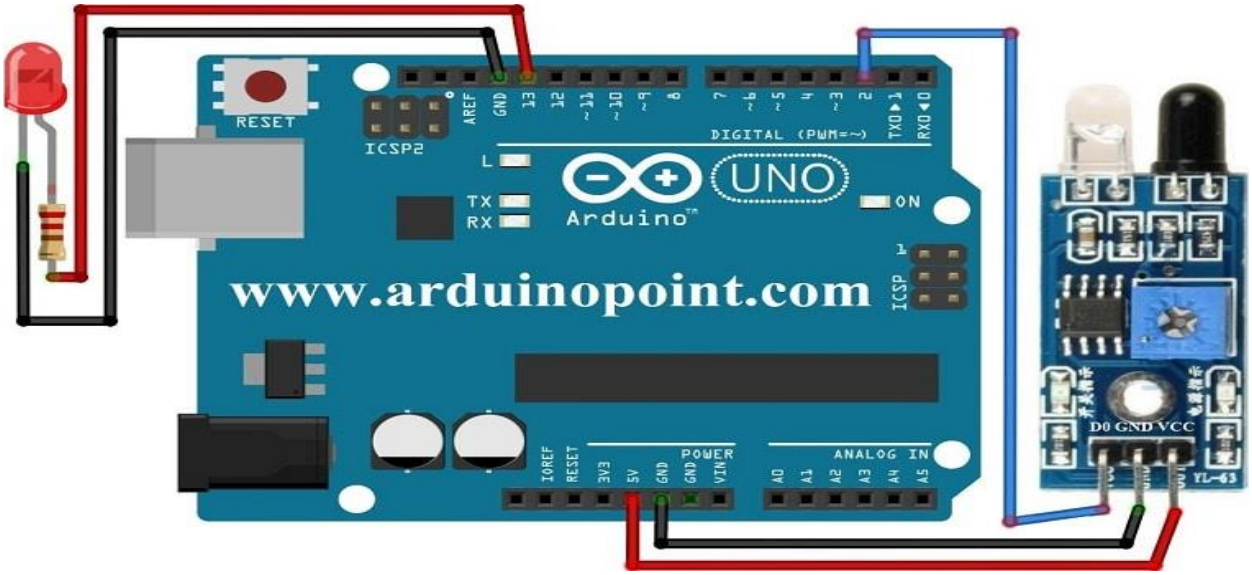


AIM: Write a program to turn ON LED at sensor (IR/LDR) detection using Arduino UNO

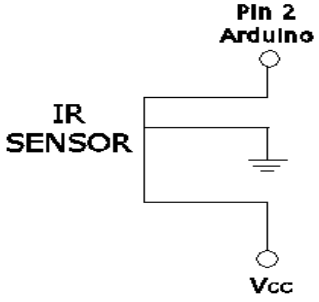
COMPONENTS REQUIRED:

1. Arduino UNO.
2. LDR/IR(DHT-11).
3. LED.
4. Resistor (220Ω)
5. Connecting cable or USB cable.
6. Breadboard.
7. Jumper wires

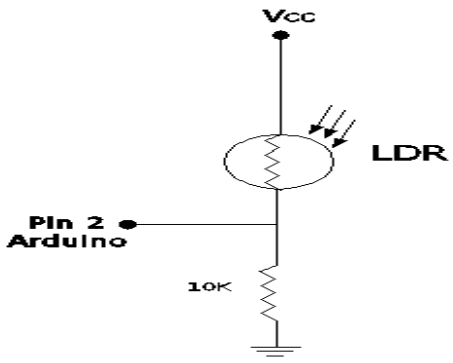
CONNECTION DIAGRAM:



IR SENSOR



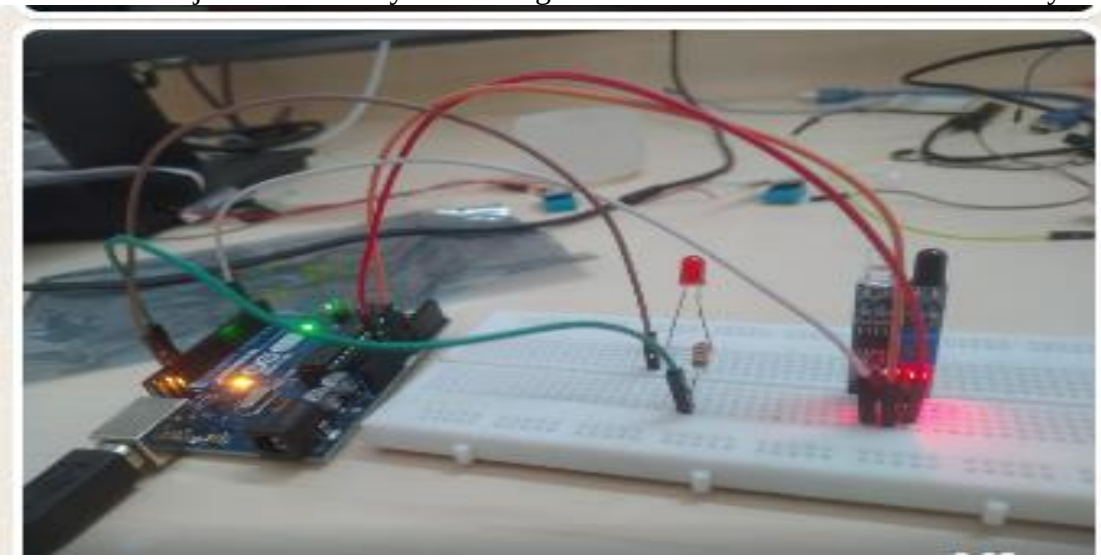
LDR



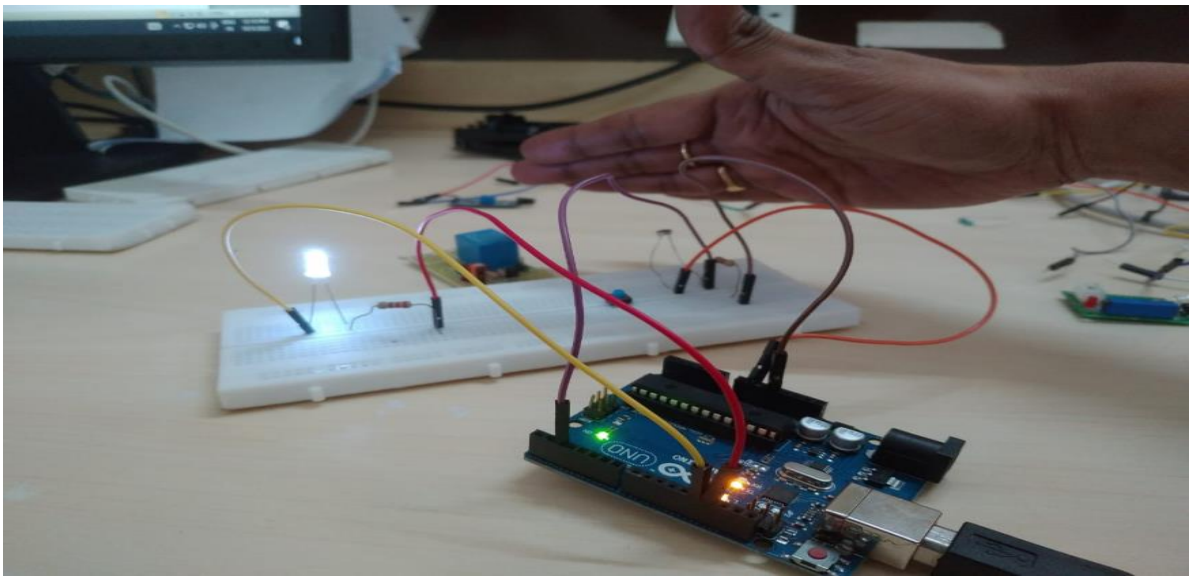
CODING:

```
int IRLDRPin = 2;
int OutputPin = 13;

void setup() {
  pinMode(OutputPin, OUTPUT);
  pinMode(IRLDRPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  int SensorValue = digitalRead(IRLDRPin);


  Serial.print("IRLDRPin Value: ");
  Serial.println(SensorValue);
  delay(100);
   if (SensorValue==LOW){ // LOW MEANS Object Detected
    digitalWrite(OutputPin, HIGH);
  }
  else
  {
    digitalWrite(OutputPin, LOW);
  }
}
```

RESULT: Object detection by interfacing IR with Arduino UNO has successfully executed



RESULT: Controlling LED by interfacing LDR with Arduino UNO has successfully executed



EXPERIMENT – 2

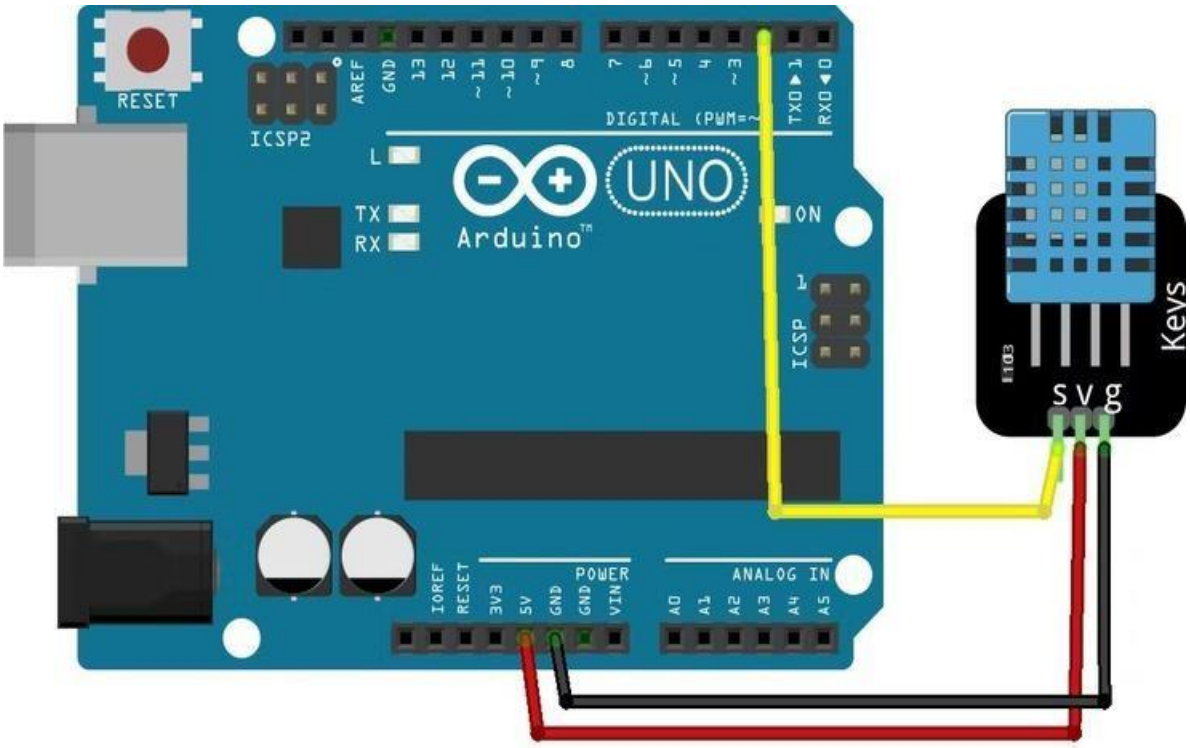
i. To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.

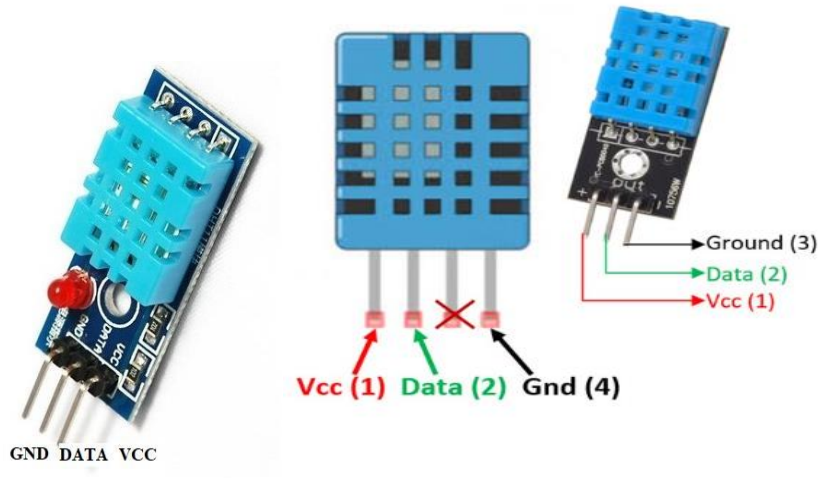
AIM: Write a program to interface DHT11 with Arduino/Raspberry Pi to print temperatureand humidity readings.

COMPONENTS REQUIRED:

1. Arduino UNO/ Raspberry Pi.
2. DHT11.
3. Connecting cable or USB cable.
4. Jumper wires.

CONNECTION DIAGRAM:





Step 1: Install the library for DHT in Arduino IDE.

- Open Arduino IDE and navigate to Sketch > Include Library ManageLibraries.
- Search for “DHTlib” and install the “DHTlib” library in the Arduino IDE

CODING:

```
#include <DHT11.h>
// Create an instance of the DHT11 class and set the digital I/O pin.
DHT11 dht11(2);

void setup()
{
    // Initialize serial communication at 115200 baud.
    Serial.begin(115200);
}
void loop()
{
    // Read the humidity from the sensor.
    float humidity = dht11.readHumidity();

    // Read the temperature from the sensor.
    float temperature = dht11.readTemperature();

    // If the temperature and humidity readings were successful, print them to the
serial monitor.
    if (temperature != -1 && humidity != -1)
    {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.println(" C");
```

```
        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
    }
    else
    {
        // If the temperature or humidity reading failed, print an error message.
        Serial.println("Error reading data");
    }

    // Wait for 2 seconds before the next reading.
    delay(2000);
}
```

Output :


```
Output  Serial Monitor ×
Message (Enter to send message to 'Arduino Uno' on 'COM3')

Temperature: 29.00 C
Humidity: 58.00 %
Temperature: 29.00 C
Humidity: 58.00 %
Temperature: 29.00 C
Humidity: 58.00 %
Temperature: 29.00 C
Humidity: 58.00 %
```

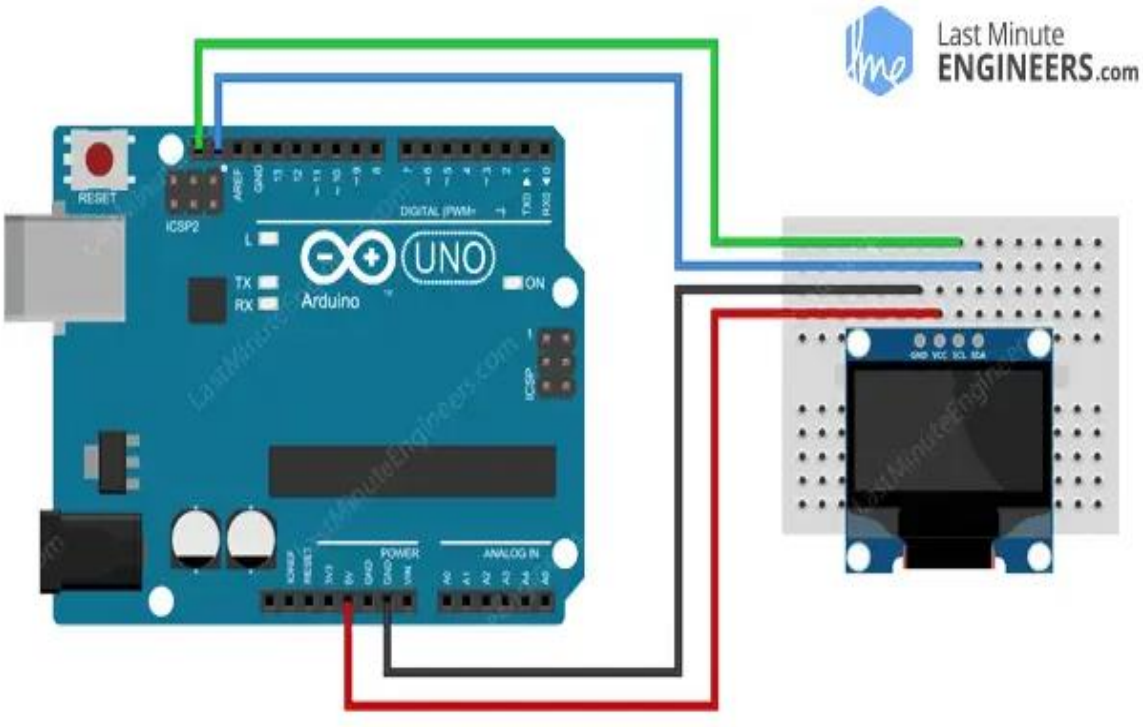
ii. To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.

AIM: Write a program to interface OLED with Arduino to print temperature and humidity reading.

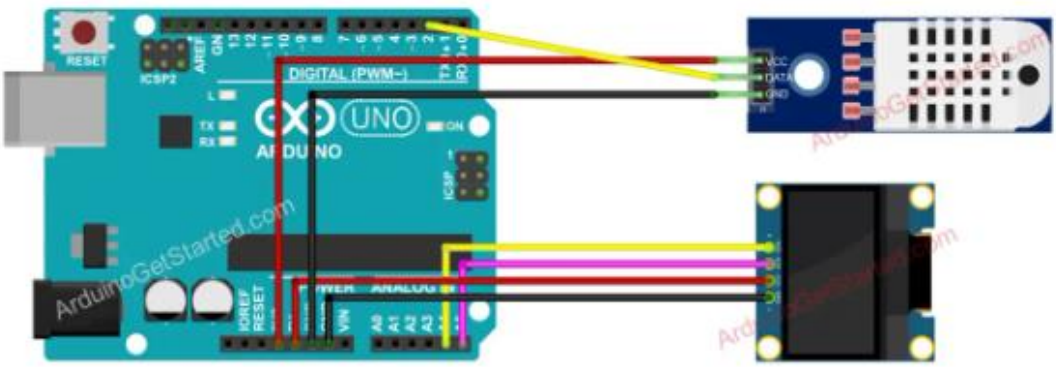
COMPONENTS REQUIRED:

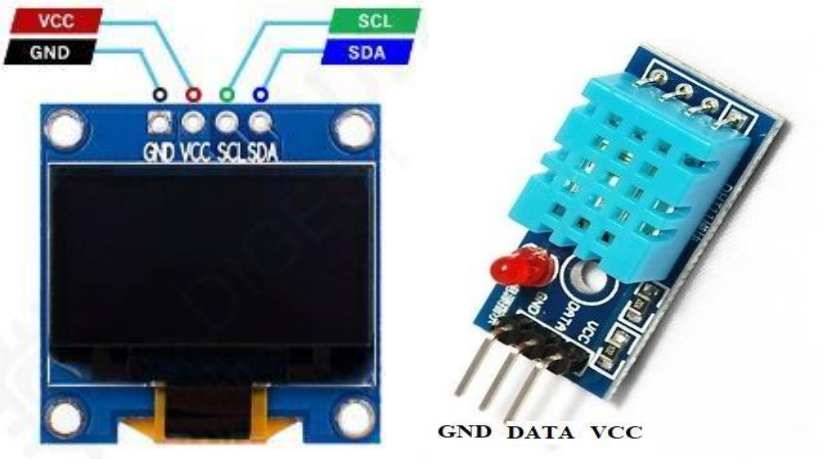
1. Arduino UNO/ Raspberry Pi.
2. DHT11.
3. 4pin OLED Display Module.
4. Connecting cable or USB cable.
5. Jumper wires.

CONNECTION DIAGRAM:

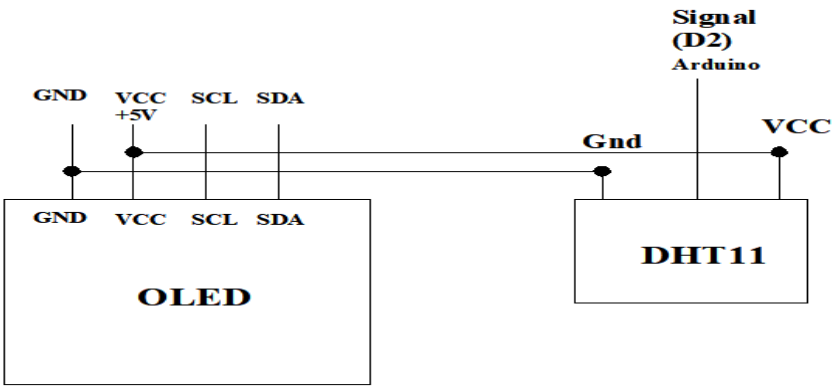


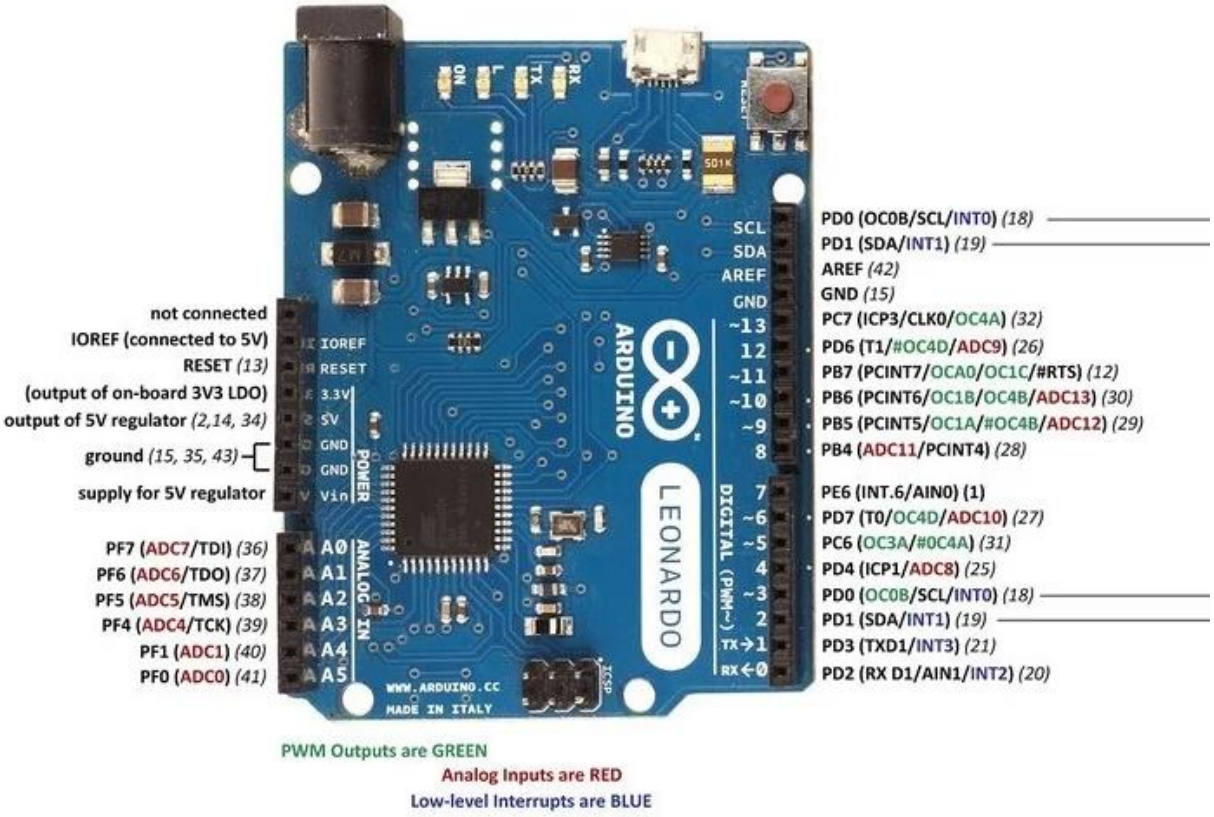
Connecting with oled with Arduino UNO





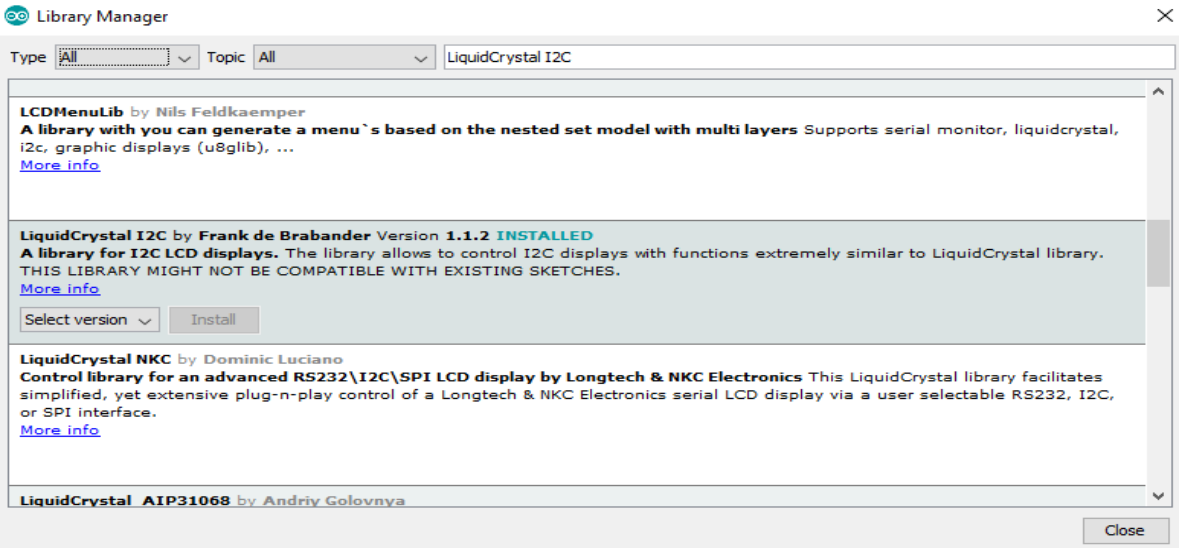
Arduino	Humidity Sensor	OLED
3V3	VCC	
GND	GND	GND
D2	DATA	
5V		VCC
SCL		SCL
SDA		SDA

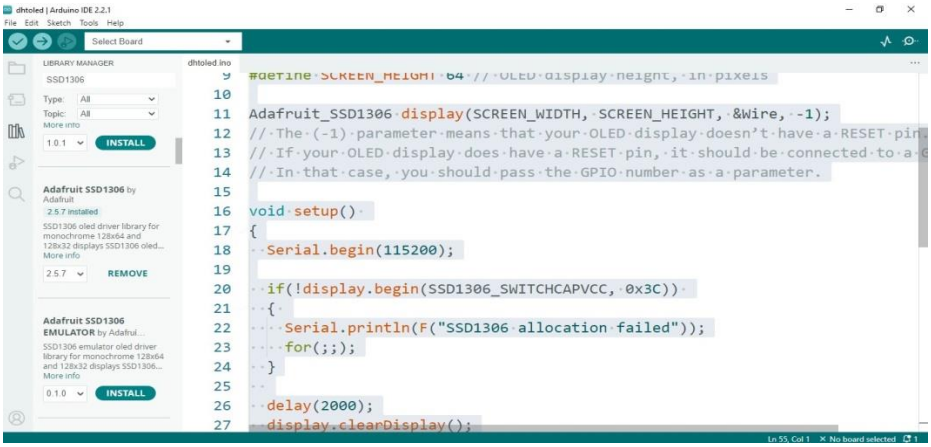

Signal
(D2)
Arduino
GND
VCC
+5V
SCL
SDA
Gnd
VCC
GND
VCC
SCL
SDA
OLED
DHT11


not connected
IOREF (connected to 5V)
RESET (13)
(output of on-board 3V3 LDO)
output of 5V regulator (2,14, 34)
ground (15, 35, 43)
supply for 5V regulator
PF7 (ADC7/TDI) (36)
PF6 (ADC6/TDO) (37)
PF5 (ADC5/TMS) (38)
PF4 (ADC4/TCK) (39)
PF1 (ADC1) (40)
PF0 (ADC0) (41)
PD0 (OC0B/SCL/INT0) (18)
PD1 (SDA/INT1) (19)
AREF (42)
GND (15)
PC7 (ICP3/CLK0/OC4A) (32)
PD6 (T1/#OC4D/ADC9) (26)
PB7 (PCINT7/OCA0/OC1C/#RTS) (12)
PB6 (PCINT6/OC1B/OC4B/ADC13) (30)
PB5 (PCINT5/OC1A/#OC4B/ADC12) (29)
PB4 (ADC11/PCINT4) (28)
PE6 (INT.6/AIN0) (1)
PD7 (T0/OC4D/ADC10) (27)
PC6 (OC3A/#0C4A) (31)
PD4 (ICP1/ADC8) (25)
PD0 (OC0B/SCL/INT0) (18)
PD1 (SDA/INT1) (19)
PD3 (TXD1/INT3) (21)
PD2 (RX D1/AIN1/INT2) (20)
ARDUINO
LEONARDO
POWER
ANALOG IN
DIGITAL (PWM~)
WWW.ARDUINO.CC
MADE IN ITALY
PWM Outputs are GREEN
Analog Inputs are RED
Low-level Interrupts are BLUE

Step 1: Install the library for OLED display in Arduino IDE

- Open Arduino IDE and navigate to **Sketch > Include Library > Manage Libraries.**
- Search for "SH110X" and install the "**SSD1306**" library from Adafruit in the Arduino IDE





Step 2: Import "Adafruit_GFX.h" & "Adafruit_SSD1306.h" header files in the code.
Define header file in the code
#include <Adafruit_GFX.h>,
#include<Adafruit_SH110X.h>
Step 3: Connect OLED display device to Arduino as per the circuit diagram

Coding :

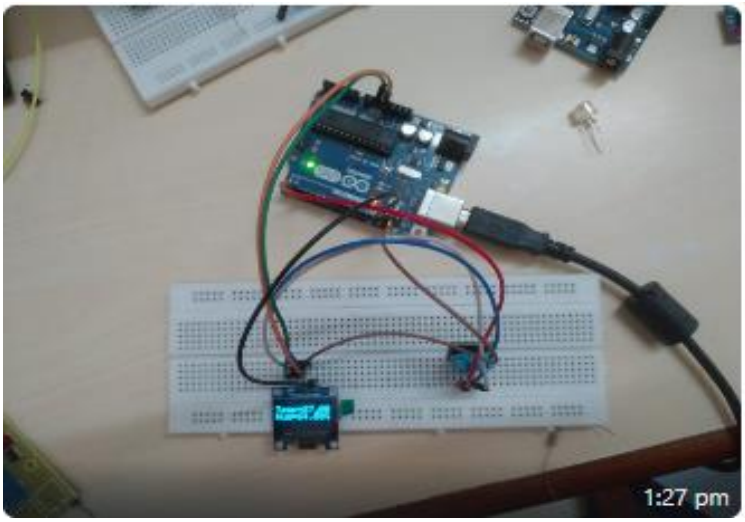
OLED 7 USING HUMDITY SENSOR

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
#include <DHT11.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

DHT11 dht11(2);

Adafruit_SH1106G oled = Adafruit_SH1106G(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup()
{
  Serial.begin(115200);
  Wire.setClock(400000);
  Wire.begin();
  oled.begin(0x3c, true);

  oled.setTextSize(2);
  oled.setTextColor(SH110X_WHITE);
  oled.setCursor(0,10);
  oled.clearDisplay();
  oled.println("Hello");
  oled.display();
  delay(2000);
  oled.clearDisplay();
}
void loop()
{

  float temperature = dht11.readTemperature();
  float humidity = dht11.readHumidity();

  oled.setTextSize(2);
  oled.setTextColor(SH110X_WHITE);
  oled.setCursor(0,10);
  oled.clearDisplay();
  oled.print("Temp=");
  oled.println(temperature);
  oled.print("Hum=");
  oled.print(humidity);
  oled.println("%");
  oled.display();
```

```
  delay(1000);
}
```

Output :

Displaying Temperature and Humidity values on OLED using Arduino UNO



EXPERIMENT – 3

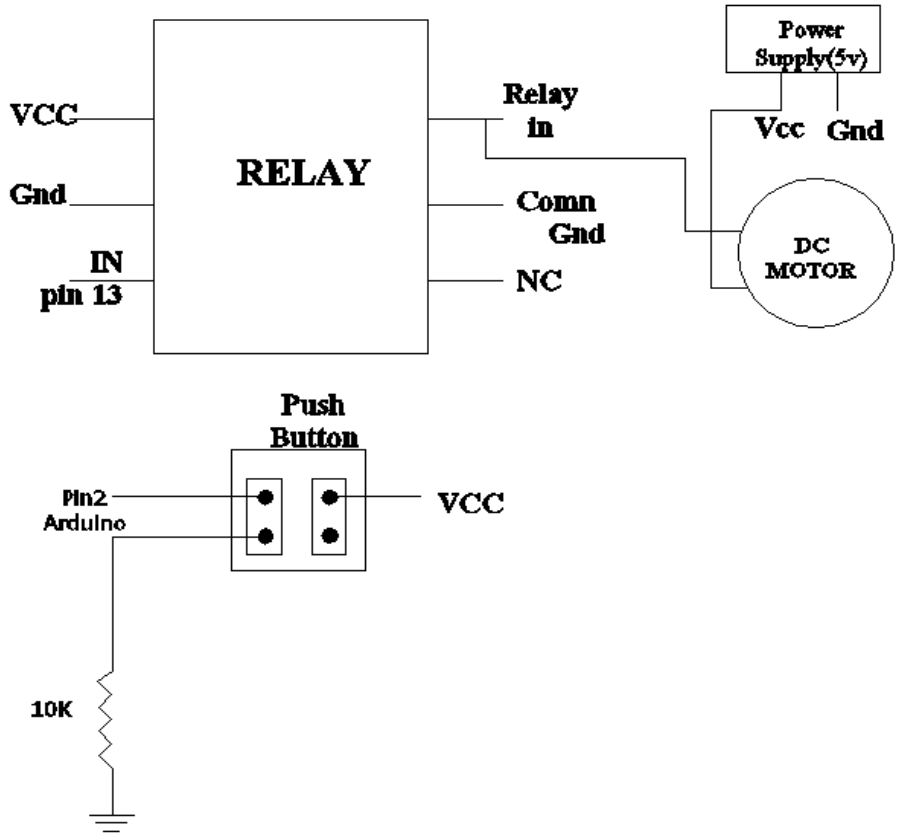
To interface motor using relay with Arduino/Raspberry Pi and write a program to turn ON motor when push button is pressed.

AIM: Write a program to turn ON motor when push button is pressed using relay with Arduino UNO.

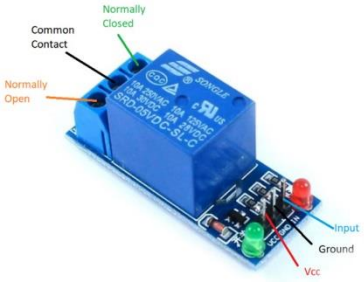
COMPONENTS REQUIRED:

1. Arduino UNO.
2. Relay.
3. Resistor (10kΩ).
4. Motor.
5. Connecting cable or USB cable.
6. Breadboard.
7. Jumper wires.

CONNECTION DIAGRAM:



Pindiagram of Relay

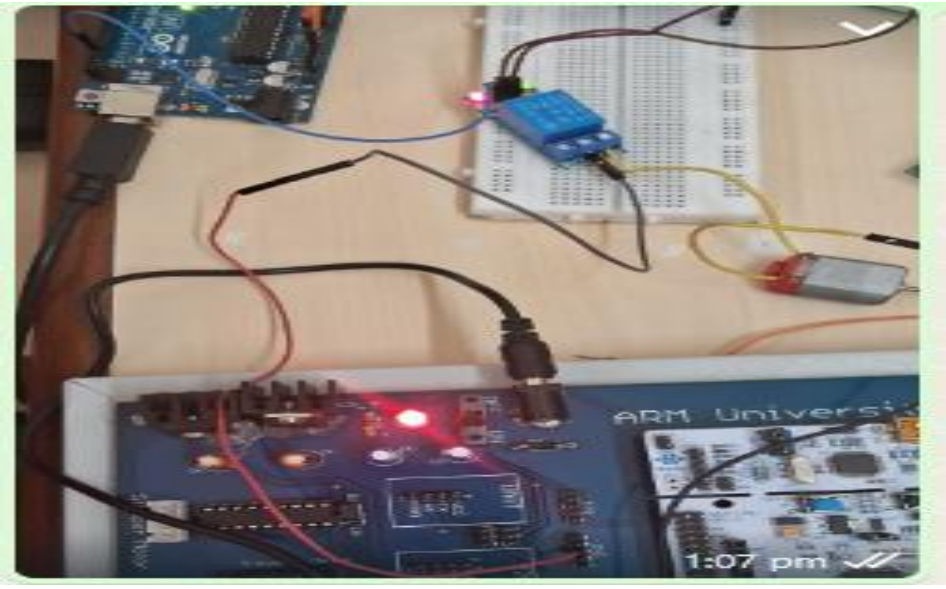


Pin	Pin Name	Description
1	Relay Trigger	Input to activate the relay
2	Ground	0V reference
3	VCC	Supply input for powering the relay
4	Normally Open	Normally open terminal of the relay
5	Common	Common terminal of the relay
6	Normally	Normally closed contact of of the relay

Coding:

```
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin

// variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Output:



EXPERIMENT – 4

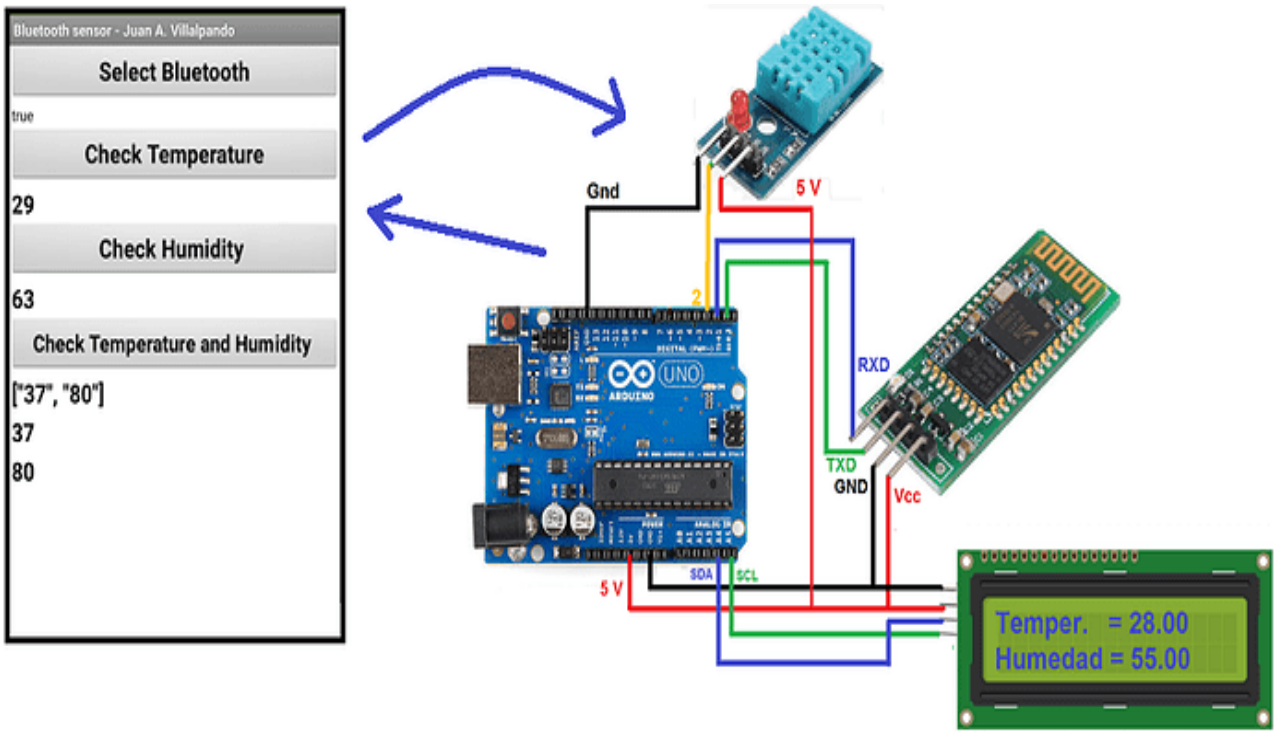
To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.

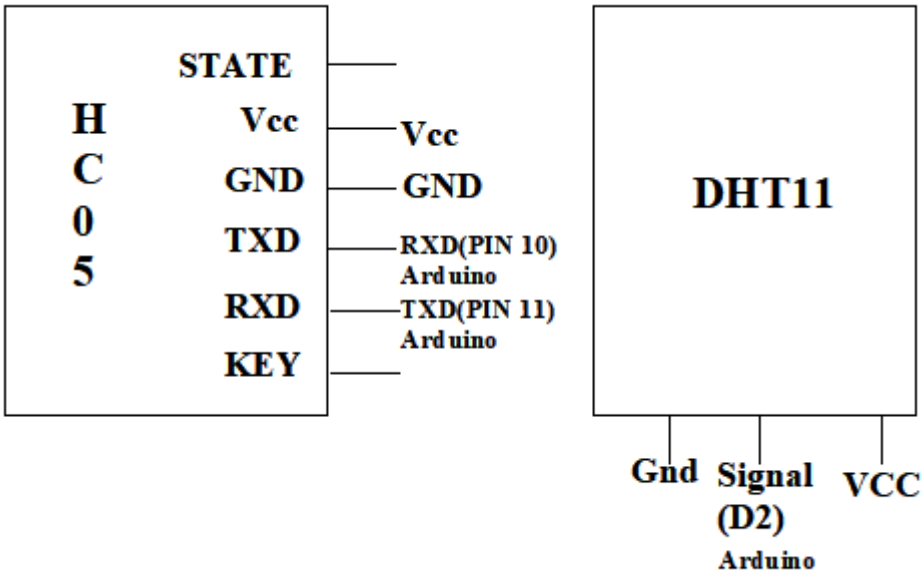
AIM: Write a program to send temperature value using DHT11 sensor to smartphoneusing Bluetooth.

COMPONENTS REQUIRED:

1. Blue Tooth-HC 05
2. DHT11.
3. SMART PHONE /LCD
4. Connecting cable or USB cable.
5. Breadboard.
6. Jumper wires.

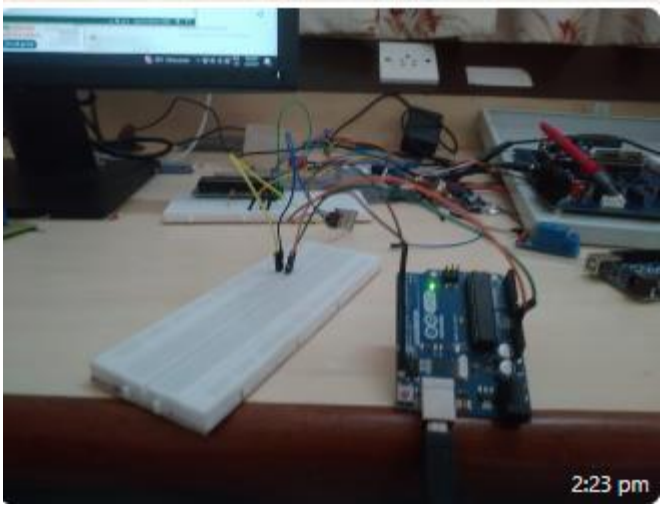
CONNECTION DIAGRAM:





Coding:

```
#include <DHT11.h>
#include <SoftwareSerial.h>       //import Software Serial library
SoftwareSerial myStream(10, 11);  //pins for Rx and Tx respectively

// Create an instance of the DHT11 class and set the digital I/O pin.
DHT11 dht11(2);


void setup() {

  myStream.begin(9600);

}

void loop() {
  float temperature = dht11.readTemperature();

  myStream.print("Temp is");
  myStream.print(',');
  myStream.print(temperature);
  myStream.print(";");
  delay(20);
}
```



RESULT: Sending Temperature data using DHT11 sensor to smartphone using Bluetooth withhas successfully executed.

EXPERIMENT – 5

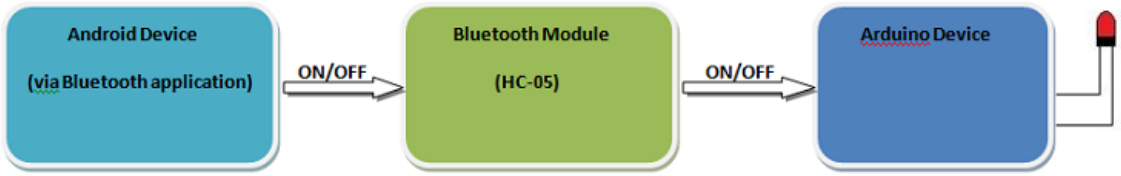
To interface Bluetooth with Arduino and write a program to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.

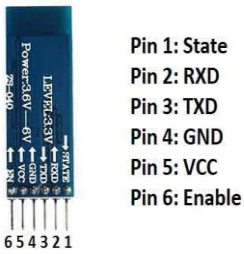
AIM: Write a program on **Arduion/Raspberry Pi** to turn LED ON/OFF from smartphone using Bluetooth when'1'/'0' is received.

COMPONENTS REQUIRED:

1. **Arduion/Raspberry Pi**.
2. LED.
3. Blue tooth
4. Resistor 220Ω
5. Connecting cable or USB cable.
6. Breadboard.
7. Jumper wires.

CONNECTION DIAGRAM:





Pin Uses :

Pin 1: It is status pin. It is used to show the status of the module whether it is connected to some device or not and some other operation like switching between master and slave. This same pin is connected to an onboard LED. That blinks depending upon the different operations.

Pin 2: It is RX(receiver) pin. It is used to receive data from the HC-05. You have to connect this pin to the TX(transmitter) pin of the Arduino UNO board.

Pin 3: It is TX(transmitter) pin. It is used to send data to the HC-05. You have to connect this pin to the RX(receiver) pin of the Arduino UNO board.

Pin 4: It is GND(ground) pin. It is used to connect GND to the HC-05 module.

Pin 5: It is VCC(power) pin. It is used to connect 5V to the HC-05 module.

Pin 6: It is enable pin. It is used to switch between master and slave configuration

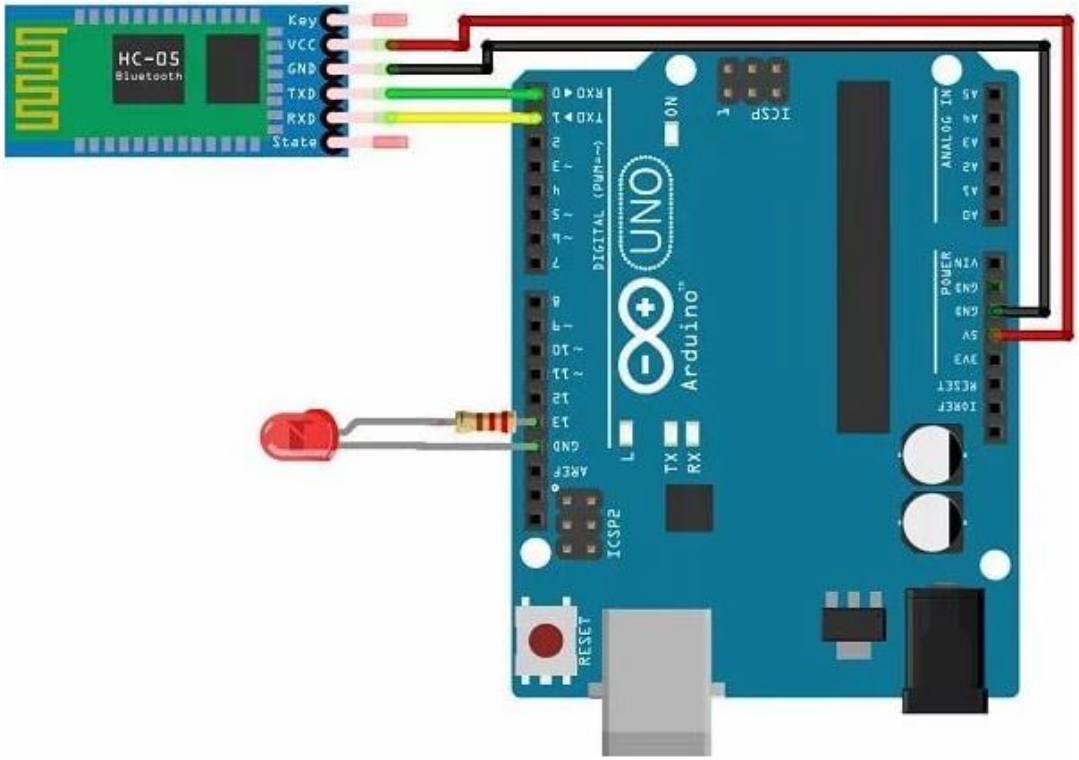
TX ----------> RX (Pin 0)

RX ----------> TX (Pin 1)

VCC ----------> 5v

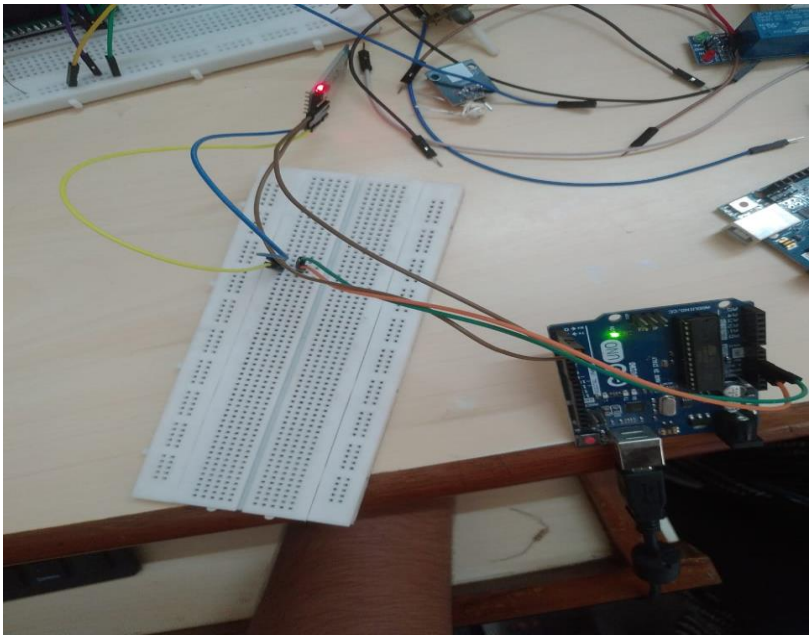
GND ----------> GND

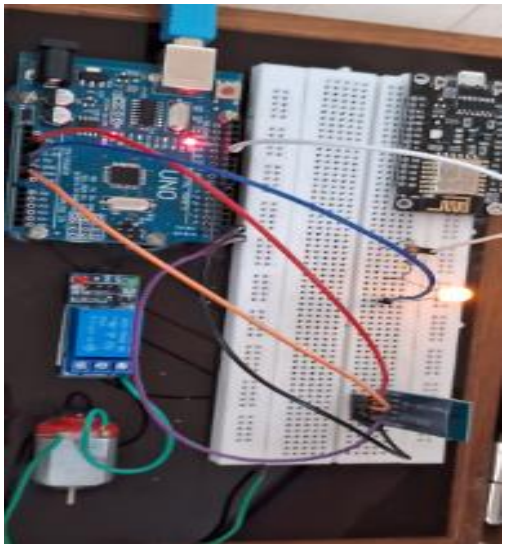
LED Pin	**Arduino UNO**
Pin 1	GND
Pin 2	Pin 13



Coding :

```
int ledPin = 13; // LED connected to digital pin 13

void setup() {
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
  Serial.begin(9600); // initialize serial communication at 9600 bits per second
}

void loop() {
  if (Serial.available() > 0) { // if data is available to read
    char received = Serial.read(); // read the incoming data
    if (received == '1') {
      digitalWrite(ledPin, HIGH); // turn on the LED
    } else if (received == '0') {
      digitalWrite(ledPin, LOW); // turn off the LED
    }
  }
}
```





RESULT: LED ON/OFF from smartphone using Bluetooth based on received '1'/'0' data with **Arduino** has successfully executed

EXPERIMENT – 6

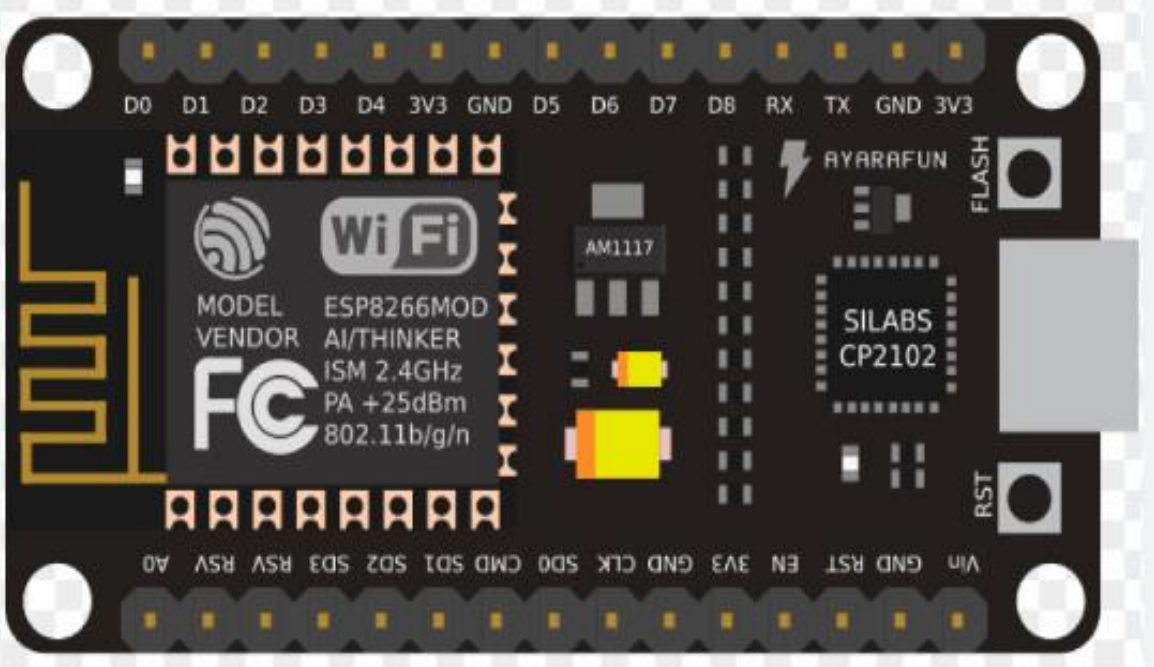
Write a program on Arduino/Raspberry Pi to upload temperature and humidity data to Thingspeak cloud.

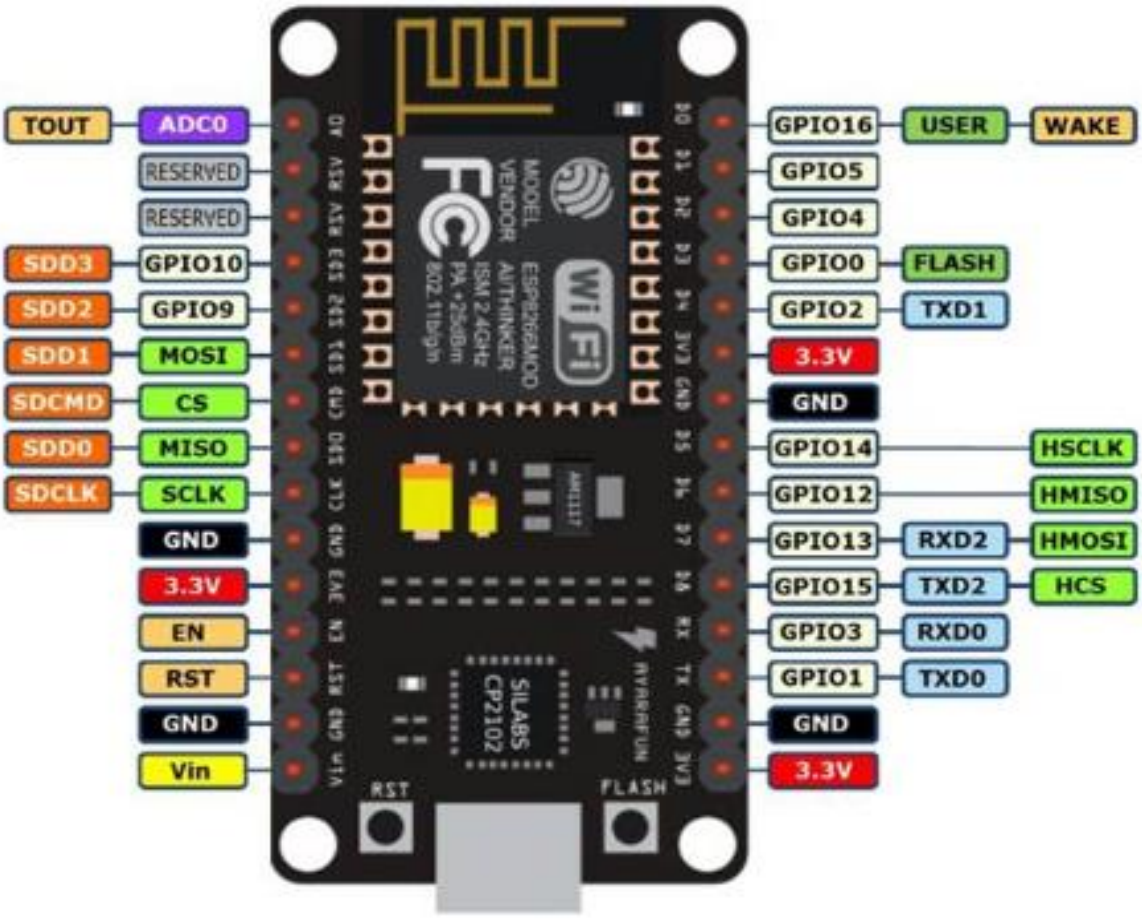
AIM: Write a program on Node MCU ESP8266 to upload temperature and humidity value using DHT11 sensor to Thingspeak cloud

COMPONENTS REQUIRED:

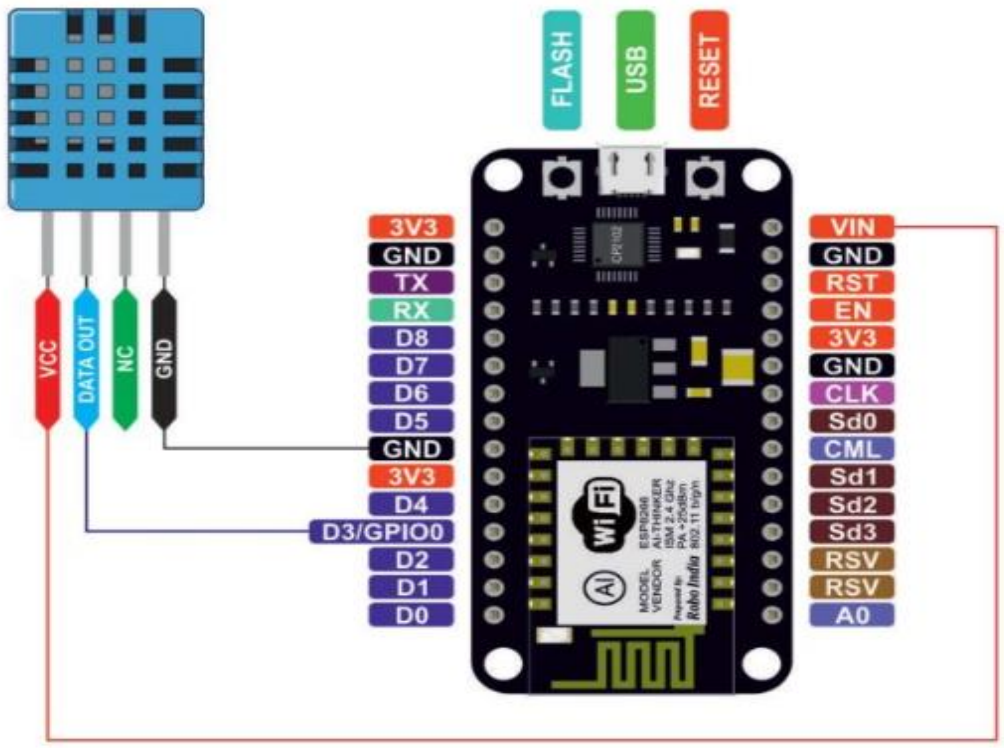
1. NODEMCU 8266.
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires

Pin diagram : NODE MCU 8266





CONNECTION DIAGRAM:



Sl.NO.	NodeMCU	DHT11
1	Vin	VCC
2	GND	GND
3	D3/D4	Data Out

To connect an ESP8266 module to an Arduino board, you'll need to follow these steps:

1. Install the Arduino IDE*: Make sure you have the latest version of the Arduino IDE installed on your computer.
2. Install ESP8266 Board Support Package. The ESP8266 is not natively supported by the Arduino IDE. You need to install the board support package. Here's how:

- Open the Arduino IDE.
- Go to "File" > "Preferences".
- In the "Additional Boards Manager URLs" field, add the following URL:
- http**://arduino.esp8266.com/stable/package_esp8266com_index.json**
- Click "OK".
- Go to "Tools" > "Board" > "Boards Manager".
- Search for "esp8266" and install the "esp8266" package.

3. Select the ESP8266 Board*: After installing the board package, you can select the ESP8266 board in the Arduino IDE.

- Go to "Tools" > "Board" and select the appropriate ESP8266 board (e.g., "NodeMCU 1.0").

4. Wiring Connections*: Connect your ESP8266 to the Arduino as follows:

- ESP8266 VCC → Arduino 3.3V
- ESP8266 GND → Arduino GND
- ESP8266 TX → Arduino RX (Pin 0)
- ESP8266 RX → Arduino TX (Pin 1)
- ESP8266 CH_PD → Arduino 3.3V (Enable pin)

Note: Make sure to not connect ESP8266 5V pins directly to Arduino's pins. The ESP8266 operates at 3.3V and can be damaged by 5V logic levels.

5. Upload Code: You can now write and upload code to your ESP8266 using the Arduino IDE. Create a new sketch, write your code, and click the "Upload" button.

6. Serial Communication: To view the serial output from your ESP8266, open the Serial Monitor in the Arduino IDE. Choose the appropriate baud rate (usually 115200) to match your ESP8266 code.

Remember that the ESP8266 is quite capable by itself, featuring Wi-Fi capabilities, so you can use it as a standalone device or connect it to the Arduino for more complex projects. Depending on your project's needs, the code you write for the ESP8266 might involve Wi-Fi setup, data transmission, or other functionalities.

Steps for things speak

Step 1: Go to https://thingspeak.com/ and create your ThingSpeak Account if you don't have. Login to Your Account.
Step 2: Create a Channel by clicking 'New Channel

Step 3: Enter the channel details.

Name: Any Name

Description: Optional

Field 1: Sensor reading – This will be displayed on the analytics graph. If you need more than 1 Channels you can create for additional Sensor Data.

Save this setting.

Step 4: Now you can see the channels. Click on the 'API Keys' tab. Here you will get the Channel ID and API Keys. Note this down.

Step 5: Open Arduino IDE and Install the ThingSpeak Library. To do this go to Sketch>Include Library>Manage Libraries. Search for ThingSpeak and install the library. ThingSpeak Communication Library for Arduino, ESP8266 and ESP32 https://thingspeak.com

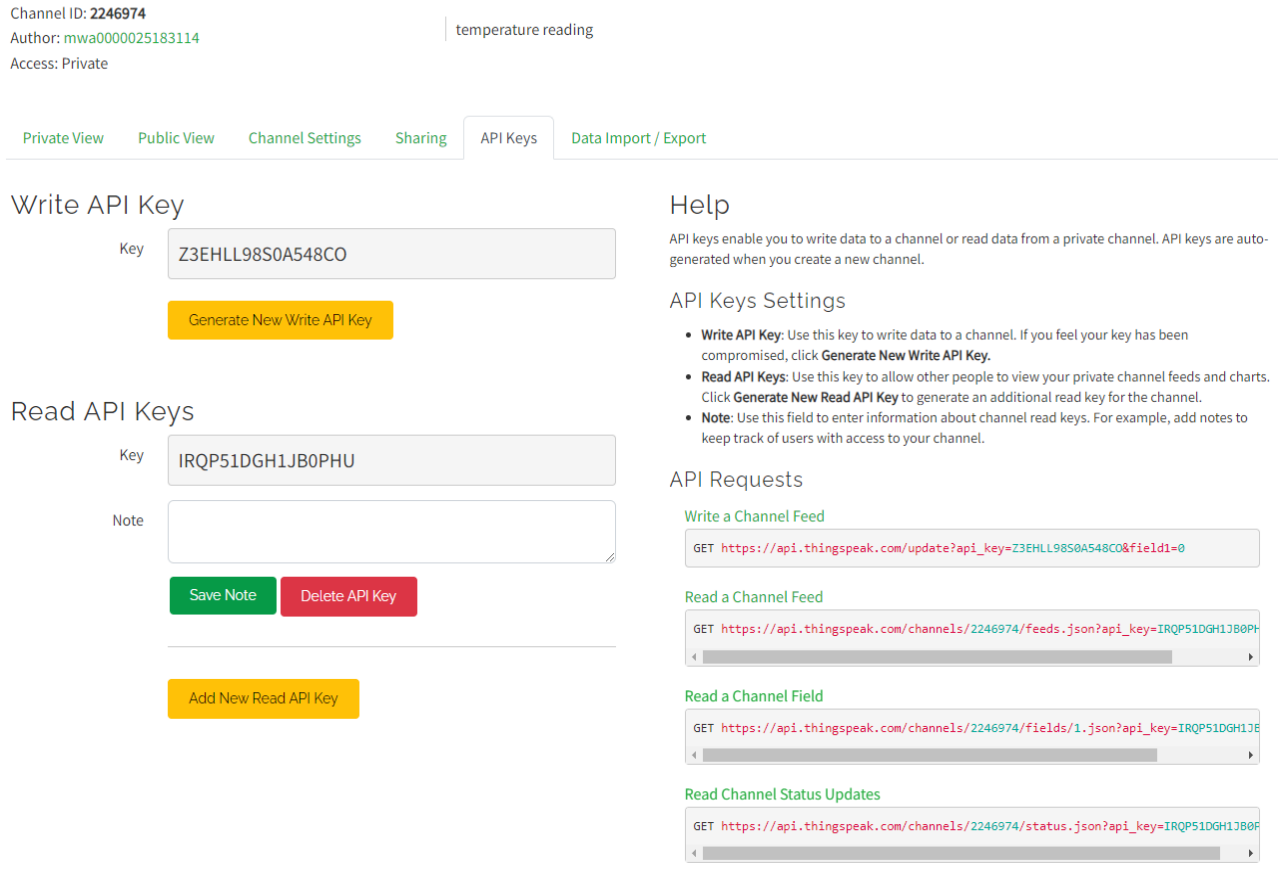
Step 6: Need to modify the code .In the below code you need to change your Network SSID, Password and your ThingSpeak Channel and API Keys.

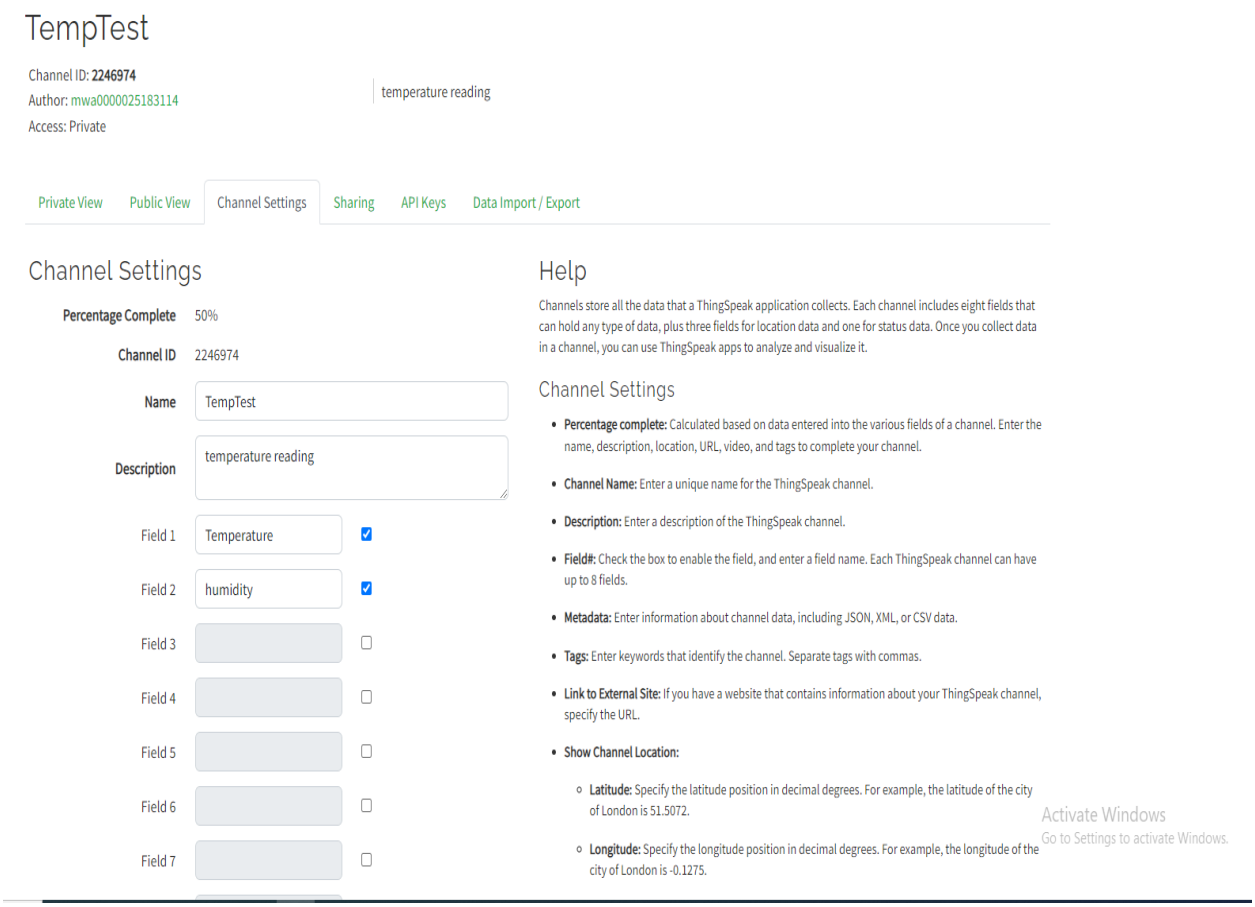
Installing ThingSpeak Library

- Click on sketch □ Include Library □ Add Zip Library
- Navigate to the Zip named **thingspeak-Arduino master.zip** and click Ok

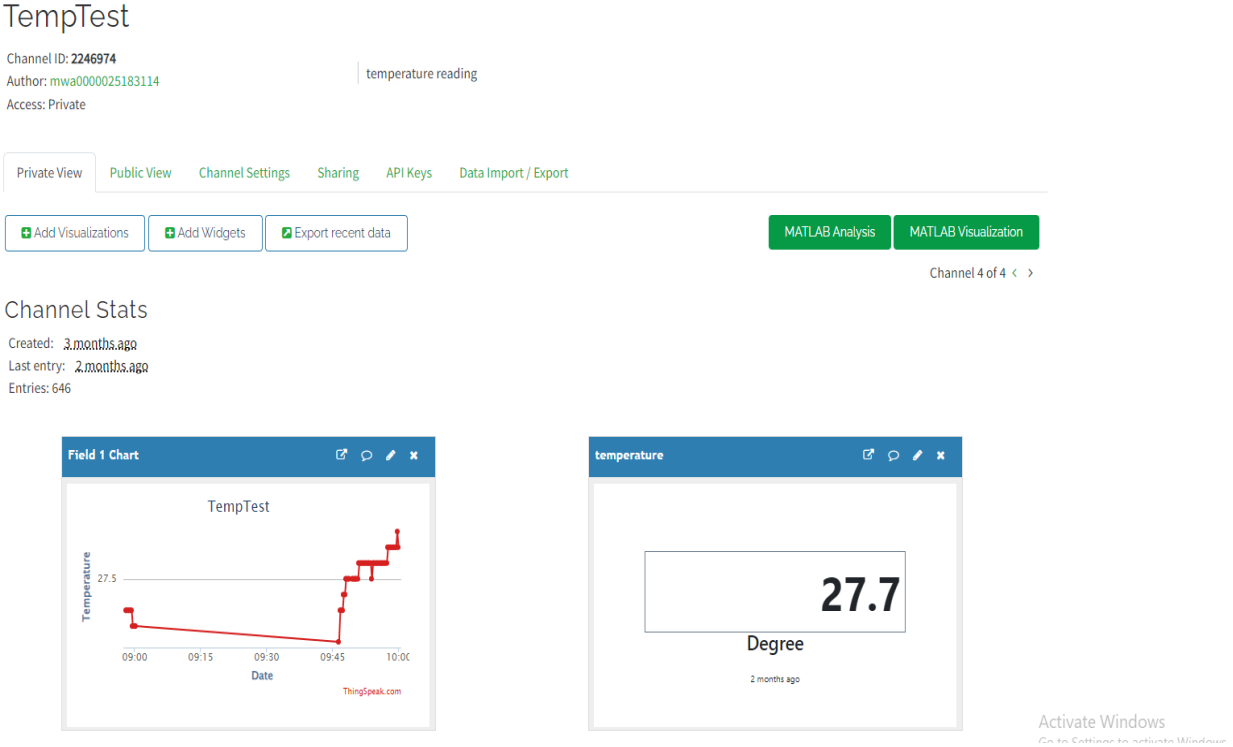
Go to the weblink: https://thingspeak.com/
Click signup button
Create mathwork account
Check your E-mail to verifythe account.
Click the Green button **I agree**
You are redirected to thingspeak
Website in your account a new channel button as

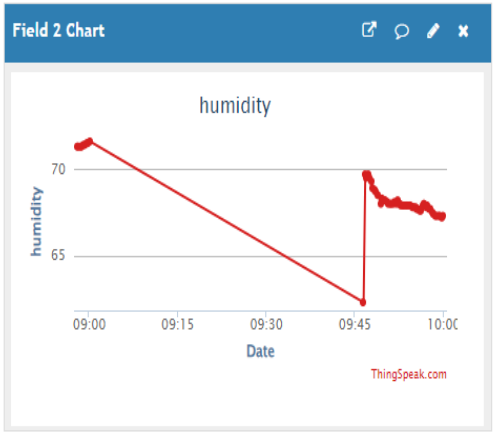
Channel setting

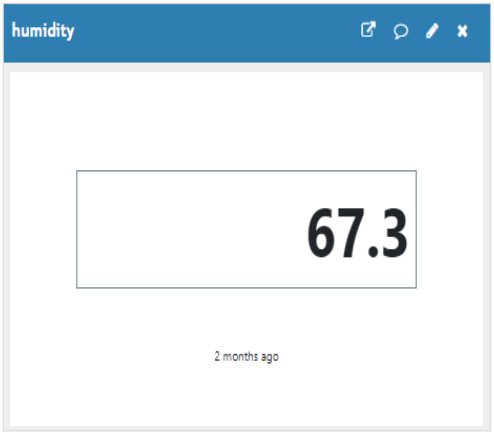




Private/Public View







Coding :

```
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>
#include <DHT.h>

// Replace with your network credentials
const char* ssid = "KWVOF";
const char* password = "ece@1234";

// Replace with your ThingSpeak channel details
const unsigned long channelID = 2246974; // Replace with your channel ID
const char* writeAPIKey = "Z3EHLL98S0A548CO";  // Replace with your Write API Key

// Initialize the DHT sensor
#define DHTPIN 2          // Pin where the DHT11 sensor is connected (GPIO2 on most
ESP8266 boards)
#define DHTTYPE DHT11     // DHT11 sensor type
DHT dht(DHTPIN, DHTTYPE);

WiFiClient client;

void setup() {
  Serial.begin(9600);
  delay(10);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
```

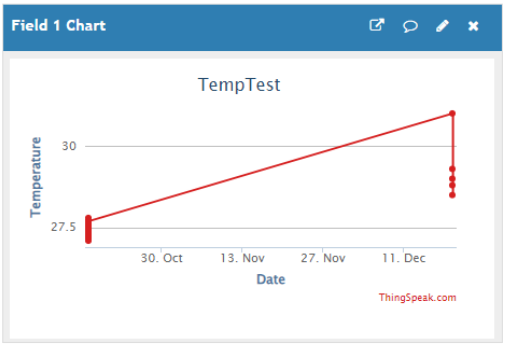
```
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Initialize ThingSpeak
  ThingSpeak.begin(client);

  // Initialize DHT sensor
  dht.begin();
}

void loop() {
  // Read temperature and humidity from DHT11 sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();



  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    delay(2000);
    return;
  }

  Serial.print("Temperature: ");
  Serial.println(temperature);
  Serial.print("Humidity: ");
  Serial.println(humidity);

  // Write data to ThingSpeak
  ThingSpeak.setField(1, temperature);
  ThingSpeak.setField(2, humidity);

  if (ThingSpeak.writeFields(channelID, writeAPIKey)) {
    Serial.println("Data sent to ThingSpeak successfully.");
  } else {
    Serial.println("Failed to send data to ThingSpeak.");
  }

  // Delay before reading and writing again (adjust as needed)
  delay(1000); // Read and write data every 60 seconds
}
```

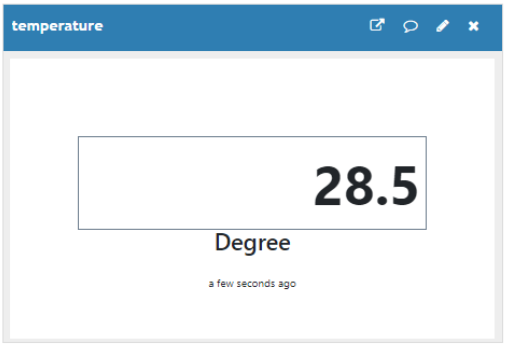
Note : In this code:

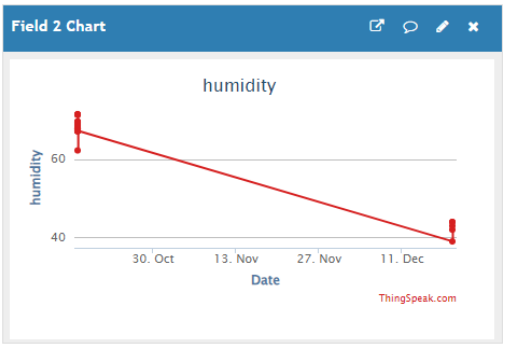
- Replace "Your_SSID", "Your_PASSWORD", YOUR_CHANNEL_ID, and "Your_Write_API_Key" with your actual Wi-Fi credentials, ThingSpeak channel ID, and Write API Key.
- The code connects to Wi-Fi, initializes the ThingSpeak library, and initializes the DHT11 sensor.
- It reads temperature and humidity data from the DHT11 sensor, checks for valid readings, and then writes the data to ThingSpeak using ThingSpeak.setField() and ThingSpeak.writeFields().
- Data is written to ThingSpeak every 60 seconds (adjust the delay as needed).
- Upload this code to your ESP8266 board, and it will read temperature and humidity from the DHT11 sensor and send the data to your ThingSpeak channel periodically.

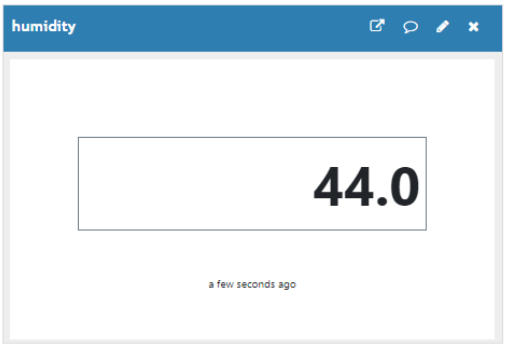
Output:

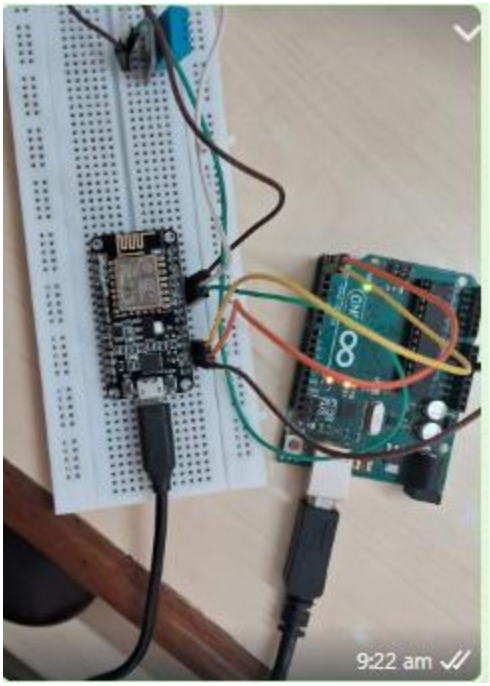
```
Temperature: 29.80
Humidity: 41.00
Data sent to ThingSpeak successfully.
Temperature: 29.70
Humidity: 41.00
Data sent to ThingSpeak successfully.
Temperature: 29.70
Humidity: 41.00
Data sent to ThingSpeak successfully.
Temperature: 29.70
Humidity: 42.00
Data sent to ThingSpeak successfully.
Temperature: 29.70
Humidity: 42.00
Data sent to ThingSpeak successfully.
Temperature: 29.60
Humidity: 42.00
Data sent to ThingSpeak successfully.
```











RESULT: Sending temperature and humidity data using DHT11 sensor to Thingspeak cloudwith NODEMCU 8266 has successfully executed.

EXPERIMENT – 7

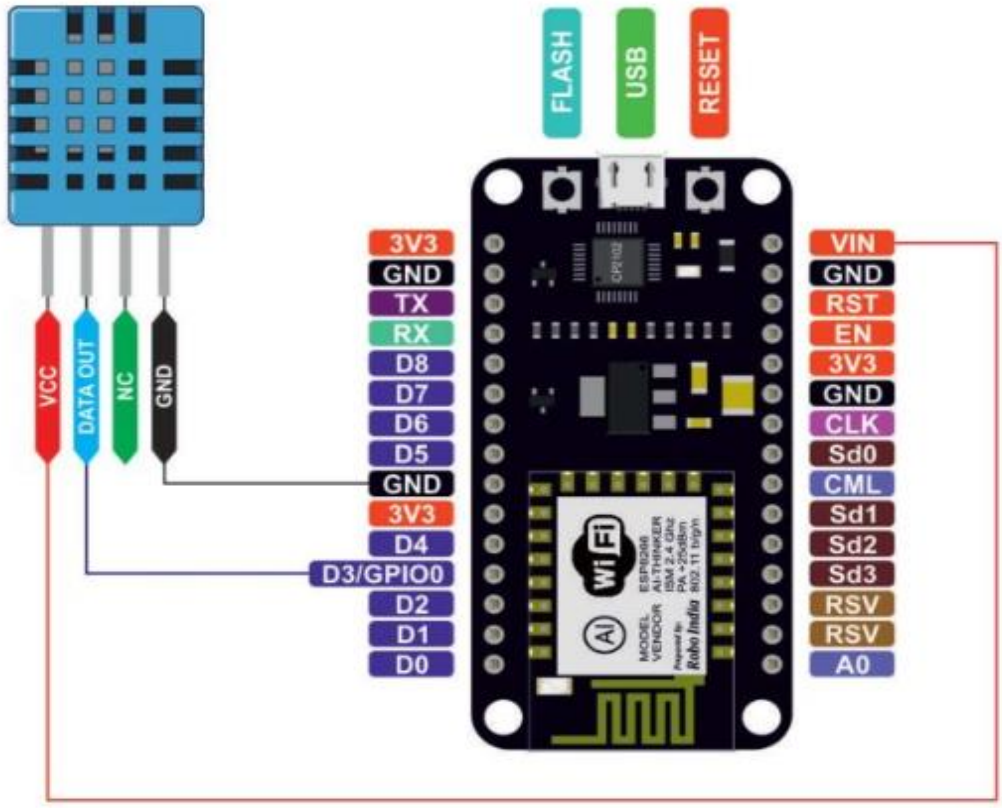
Write a program on Arduino/Raspberry Pi to retrieve temperature and humidity data from Thingspeak cloud.

AIM: Write A Program on Node MCU 8266 to retrieve temperature and humidity values using DHT11sensor from Thingspeak cloud.

COMPONENTS REQUIRED:

1. NODEMCU 8266.
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires

CONNECTION DIAGRAM:



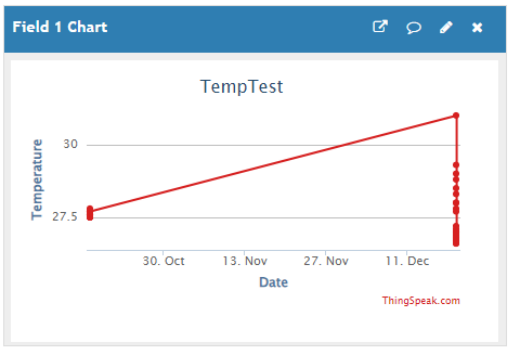
Sl.NO.	NodeMCU	DHT11
1	Vin	VCC
2	GND	GND
3	D4	Data Out

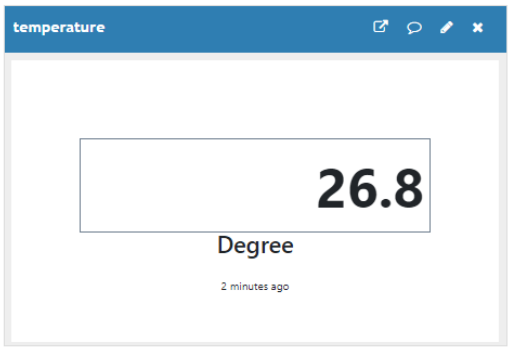
Coding :

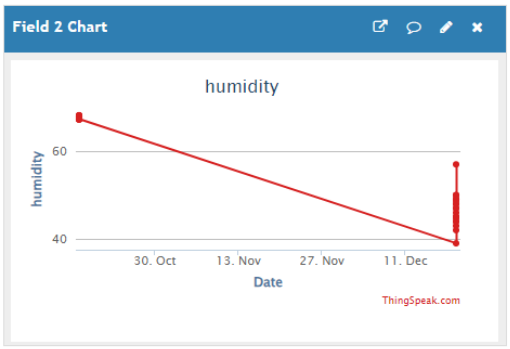
```
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>
// Replace with your network credentials
const char* ssid = "KWVOF";
const char* password = "ece@1234";

// Replace with your ThingSpeak channel details
const unsigned long channelID = 2246974; // Replace with your channel ID
const char* readAPIKey = "IRQP51DGH1JB0PHU";     // Replace with your Read API Key

WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Initialize ThingSpeak
  ThingSpeak.begin(client);
}
void loop() {
  float temperature, humidity;

  // Read temperature from Field 1
  temperature = ThingSpeak.readFloatField(channelID, 1, readAPIKey);

  // Read humidity from Field 2
  humidity = ThingSpeak.readFloatField(channelID, 2, readAPIKey);

  // Check if reading was successful
  if (!isnan(temperature) && !isnan(humidity)) {
    Serial.print("Temperature: ");
    Serial.println(temperature);
    Serial.print("Humidity: ");
    Serial.println(humidity);
```

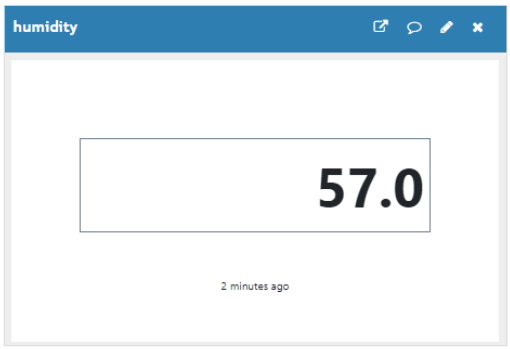
```
  } else {
    Serial.println("Failed to read data from ThingSpeak.");
  }


  // Delay before reading again
  delay(10000); // Read data every 60 seconds (adjust as needed)
}
```

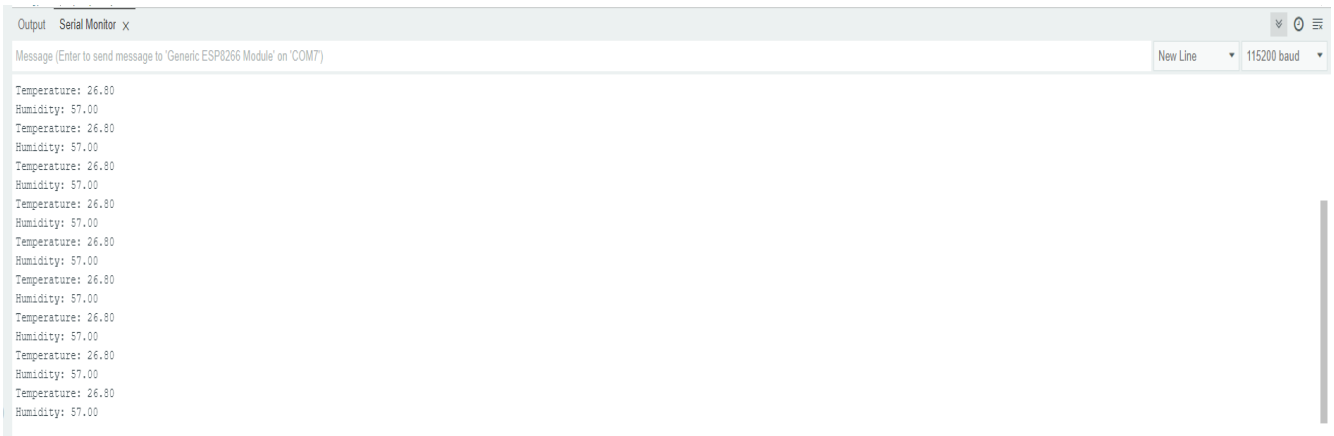
From thinkspeak:











RESULT: Retrieving temperature and humidity data from Thingspeak cloud with NODEMCU 8266 hassuccessfully executed.

EXPERIMENT – 8

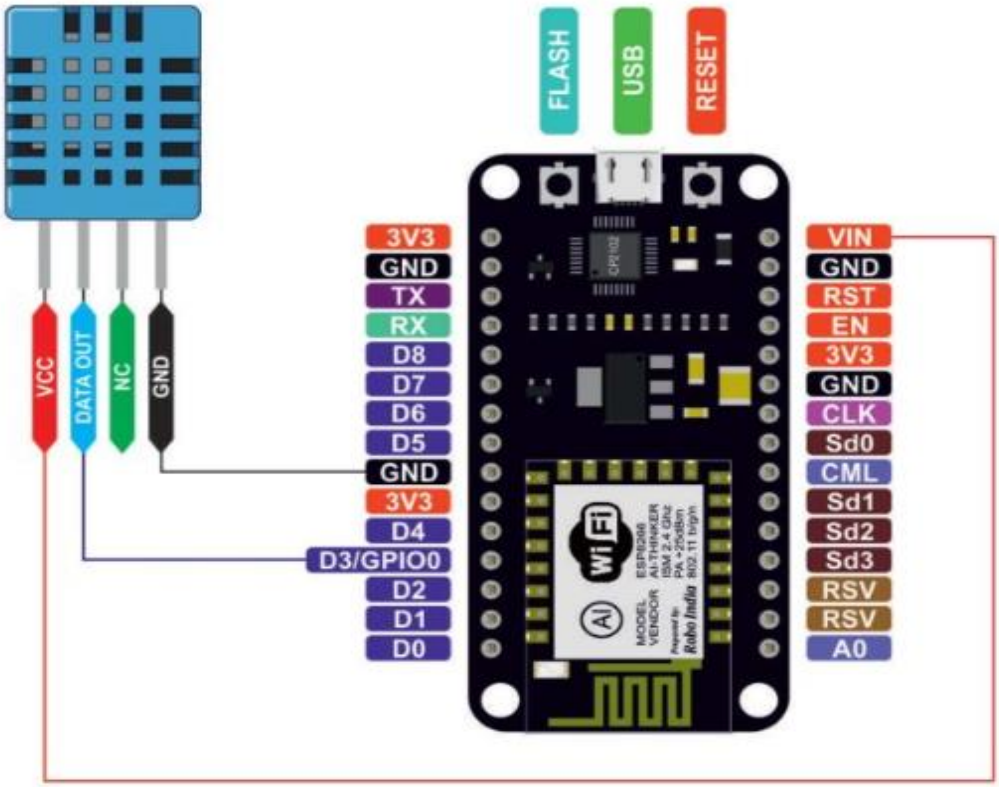
Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker.

AIM: Write a program on NODE MCU ESP8266 to publish temperature and humidity values using DHT11sensor to MQTT broker.

COMPONENTS REQUIRED:

1. NODEMCU 8266.
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires

CONNECTION DIAGRAM:



Sl.NO.	NodeMCU	DHT11
1	Vin	VCC
2	GND	GND
3	D3	Data Out

Step 1. Install Arduino libraries.

Open Arduino IDE and go to **Sketch -> Include Library -> Manage Libraries**. Find and install the following libraries:

- PubSubClient by Nick O'Leary
- WiFiEsp by bportaluri
- Adafruit Unified Sensor by Adafruit
- DHT sensor library by Adafruit
- Arduino ThingsBoard SDK by ThingsBoard
- ArduinoJSON by bblanchon
- Arduino Http Client

Note that this tutorial was tested with the following versions of the libraries:

- PubSubClient 2.6
- WiFiEsp 2.1.2
- Adafruit Unified Sensor 1.0.2
- DHT sensor library 1.3.0
- Arduino ThingsBoard SDK 0.4
- ArduinoJSON 6.10.1

https://io.adafruit.com/IOTECLAB/dashboards/h-and-t

- login : IOTECLAB
- PASSWORD: KSIT123

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username IOTECLAB

Active Key aio_lbRi81Syoa7S

REGENERATE KEY

Hide Code Samples

Arduino

```
#define IO_USERNAME  "IOTECLAB"
#define IO_KEY       "aio_lbRi81Syoa7SkCxsGBZUYHCRDgfT"
```

Linux Shell

```
export IO_USERNAME="IOTECLAB"
export IO_KEY="aio_lbRi81Syoa7SkCxsGBZUYHCRDgfT"
```

Scripting

```
ADAFRUIT_IO_USERNAME = "IOTECLAB"
ADAFRUIT_IO_KEY = "aio_lbRi81Syoa7SkCxsGBZUYHCRDgfT"
```

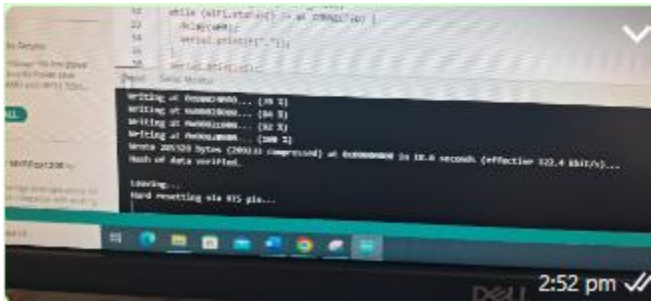
Coding:

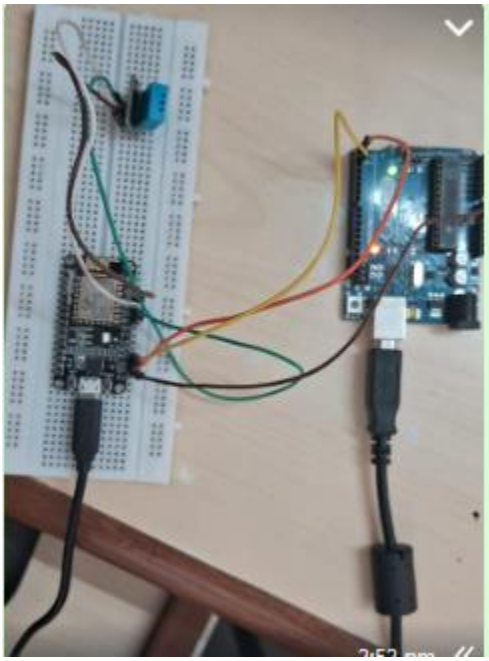
```
#include <SimpleDHT.h>                    // Data ---> D3 VCC ---> 3V3 GND ---> GND
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
// WiFi parameters
#define WLAN_SSID       "KWVOF"
#define WLAN_PASS       "ece@1234"
// Adafruit IO
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "IOTECLAB"
#define AIO_KEY         "aio_lbRi81Syoa7SkCxsGBZUYHCRDgfT"
WiFiClient client;
// Setup the MQTT client class by passing in the WiFi client and MQTT server and
login details.
```

```
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);
Adafruit_MQTT_Publish Temperature = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Temperature");
Adafruit_MQTT_Publish Humidity = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/Humidity");

int pinDHT11 = 0;
SimpleDHT11 dht11(pinDHT11);
byte hum = 0;  //Stores humidity value
byte temp = 0; //Stores temperature value
void setup() {
  Serial.begin(115200);
  Serial.println(F("Adafruit IO Example"));
  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  delay(10);
  Serial.print(F("Connecting to "));
  Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();
  Serial.println(F("WiFi connected"));
  Serial.println(F("IP address: "));
  Serial.println(WiFi.localIP());

  // connect to adafruit io
  connect();
}
// connect to adafruit io via MQTT
void connect() {
  Serial.print(F("Connecting to Adafruit IO... "));
  int8_t ret;
  while ((ret = mqtt.connect()) != 0) {
    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
      default: Serial.println(F("Connection failed")); break;
    }

    if(ret >= 0)
```

```
      mqtt.disconnect();

    Serial.println(F("Retrying connection..."));
    delay(10000);
  }
  Serial.println(F("Adafruit IO Connected!"));
}

void loop() {
  // ping adafruit io a few times to make sure we remain connected
  if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      connect();
  }
  dht11.read(&temp, &hum, NULL);
  Serial.print((int)temp); Serial.print(" *C, ");
  Serial.print((int)hum); Serial.println(" H");
  delay(5000);

   if (! Temperature.publish(temp)) {                     //Publish to Adafruit
      Serial.println(F("Failed"));
    }
       if (! Humidity.publish(hum)) {                     //Publish to Adafruit
      Serial.println(F("Failed"));
    }
    else {
      Serial.println(F("Sent!"));
    }
}
```

OUTPUT:





RESULT: Published temperature and humidity data to MQTT broker with NODEMCU 8266 has successfully executed.

EXPERIMENT – 9

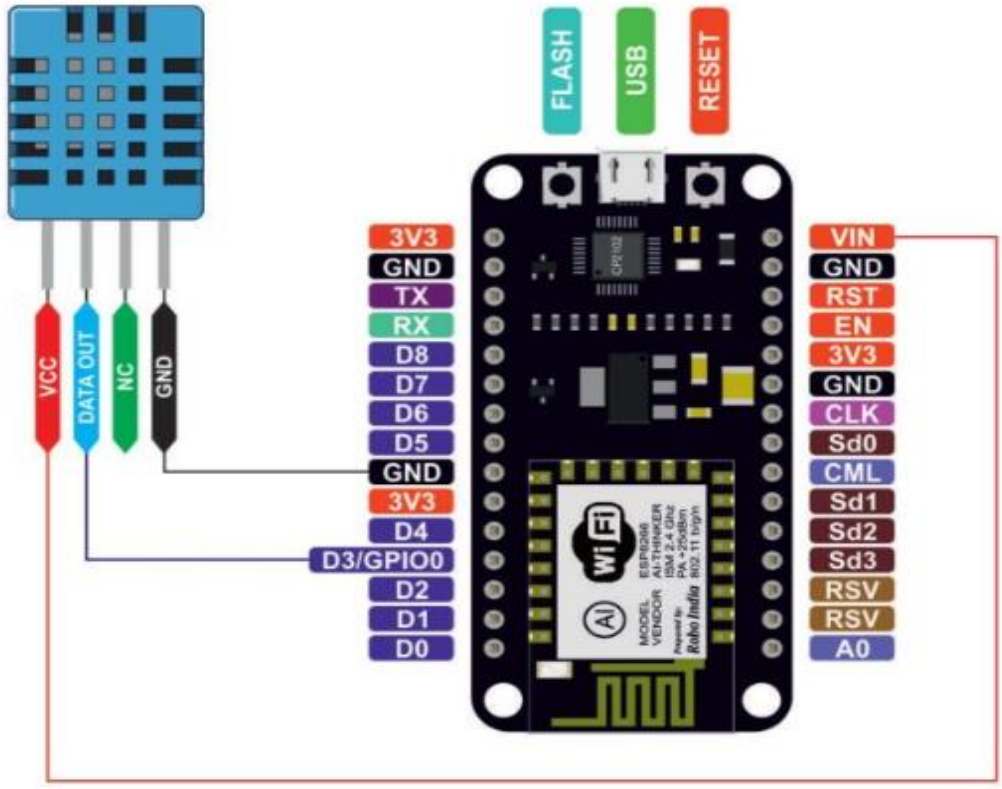
Write a program on Arduino/Raspberry Pi to subscribe to MQTT broker for temperaturedata and print it.

AIM: Write a program on Arduino/Raspberry Pi to publish temperature data to MQTT broker.

COMPONENTS REQUIRED:

1. NODEMCU 8266.
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires

CONNECTION DIAGRAM:



Coding:

```
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>

#define WIFI_SSID "KWVOF"
#define WIFI_PASS "ece@1234"
#define ADAFRUIT_SERVER "io.adafruit.com"
#define ADAFRUIT_PORT 1883
#define ADAFRUIT_USERNAME "IOTECLAB"

#define ADAFRUIT_KEY  "aio_lbRi81Syoa7SkCxsGBZUYHCRDgfT"

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, ADAFRUIT_SERVER, ADAFRUIT_PORT,
ADAFRUIT_USERNAME, ADAFRUIT_USERNAME, ADAFRUIT_KEY);

Adafruit_MQTT_Subscribe Temperature = Adafruit_MQTT_Subscribe(&mqtt,
"IOTECLAB/feeds/Temperature");

void MQTT_connect() {
int8_t ret;

// Stop if already connected.
if (mqtt.connected()) {

return;
}

Serial.print("Connecting to MQTT... ");
while ((ret = mqtt.connect()) != 0) {
Serial.println(mqtt.connectErrorString(ret));
Serial.println("Retrying MQTT connection in 5 seconds...");
mqtt.disconnect();
delay(5000); // wait 5 seconds
}

Serial.println("MQTT Connected!");
}

void setup() {
Serial.begin(115200);
delay(10);

// Connect to Wi-Fi Serial.println();
```

```
Serial.println();
Serial.print("Connecting to ");
Serial.println(WIFI_SSID);
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println();
Serial.println("WiFi connected");
Serial.println("IP Address: ");
Serial.println(WiFi.localIP());
mqtt.subscribe(&Temperature);
}

void loop() {
  // Ensure MQTT connection is established
  MQTT_connect();


  // Check for new MQTT messages
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {

    if (subscription == &Temperature) {
      Serial.print(F("Received temperature value is: "));
      Serial.println((char *)Temperature.lastread);
    }
  }

  delay(1000); // Add a delay to reduce rapid loop execution
}
```

RESULT: Program on NODEMCU ESP8266 to subscribe to MQTT broker for temperature data and printing the data has successfully executed.

EXPERIMENT – 10

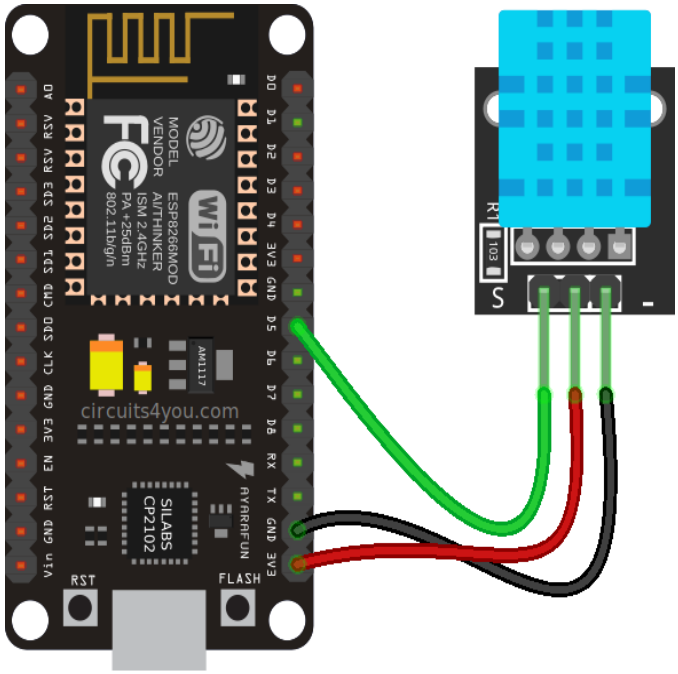
Write a program to create UDP server on Arduino/Raspberry Pi and respond withhumidity data to UDP client when requested.

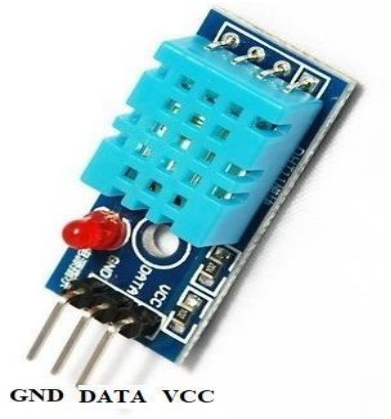
AIM: Write a program to create UDP server on NodeMCU (ESP8266) and respond withhumidity data to UDP client when requested.

COMPONENTS REQUIRED:

1. NodeMCU (ESP8266).
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires

CONNECTION DIAGRAM:





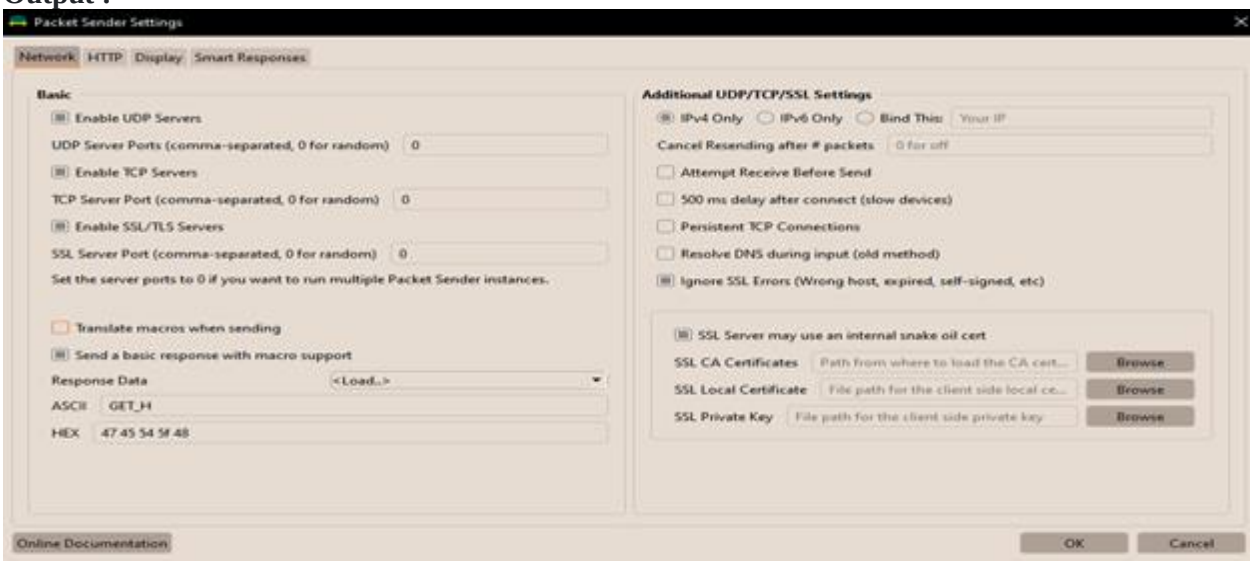
Sl.NO.	NodeMCU	DHT11
1	Vin	VCC
2	GND	GND
3	D5	Data Out

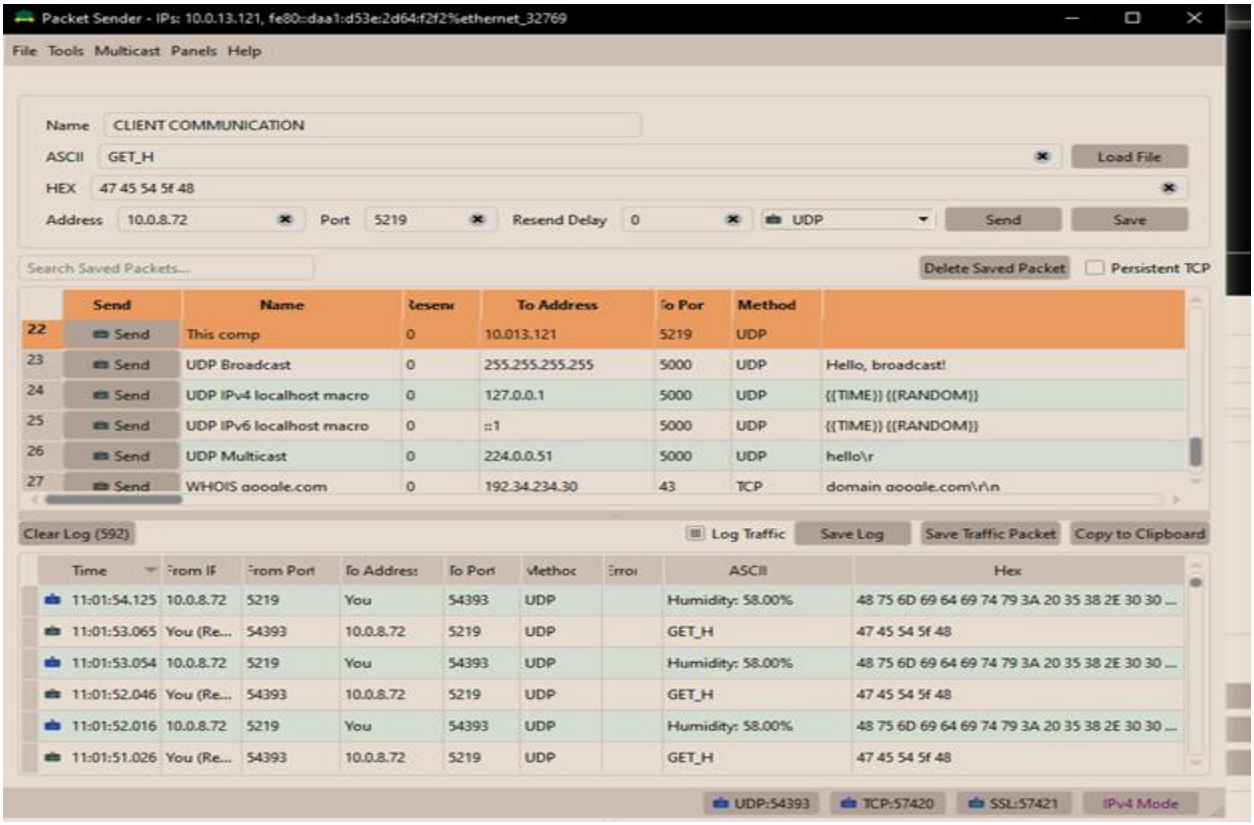
Coding:

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include "DHT.h"
DHT dht(14,DHT11); //D5 on board numbering system & 14 GPIO numbering system

const char* ssid = "KWVOF";
const char* password = "ece@1234";
const int udpPort = 5219; // UDP port to listen on 5219

WiFiUDP udp;

void setup() { Serial.begin(115200);

Serial.println("Connecting to WiFi...");
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}

Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
Serial.println("UDP server started");
udp.begin(udpPort);
```

```
}

void loop() {
char packetBuffer[255]; // buffer to hold incoming packet

int packetSize = udp.parsePacket();
if (packetSize) {
// receive incoming UDP packet
int len = udp.read(packetBuffer, 255);
if (len > 0) {
  packetBuffer[len] = 0;
}

// check if the received packet is a request for humidity data
if (strcmp(packetBuffer, "GET_H") == 0)
 {
// replace the following line with your humidity sensor reading logic
 float humidity = readHumidity(); // function to read humidity data

// send humidity data back to the client
  udp.beginPacket(udp.remoteIP(), udp.remotePort());
  udp.printf("Humidity: %.2f%%", humidity);
  udp.endPacket();
}
}
delay(5000); // add a small delay to avoid excessive response
}

float readHumidity()
{
// Read data from a DHT sensor
float humidity = dht.readHumidity();
return humidity;

}
```

Output :





RESULT: Received humidity data to UDP client from UDP server on NodeMCU (ESP8266) has successfully executed.

EXPERIMENT – 11

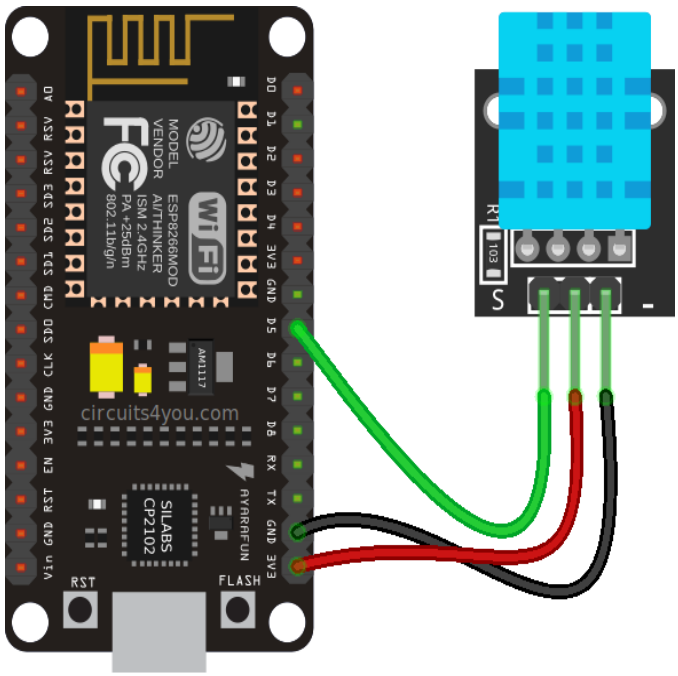
Write a program to create TCP server on Arduino/Raspberry Pi and respond with humidity data to TCP client when requested.

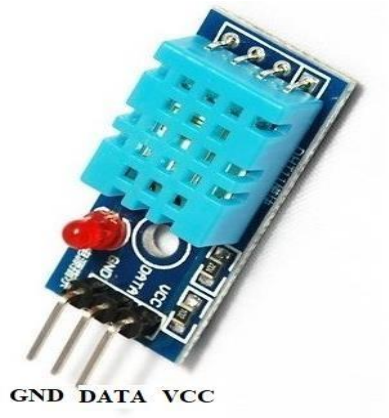
AIM: Write a program to create TCP server on NodeMCU (ESP8266) and respond with humidity data to TCP client when requested.

COMPONENTS REQUIRED:

1. NodeMCU (ESP8266).
2. DHT11.
3. Connecting cable or USB cable.
4. Breadboard.
5. Jumper wires.

CONNECTION DIAGRAM:



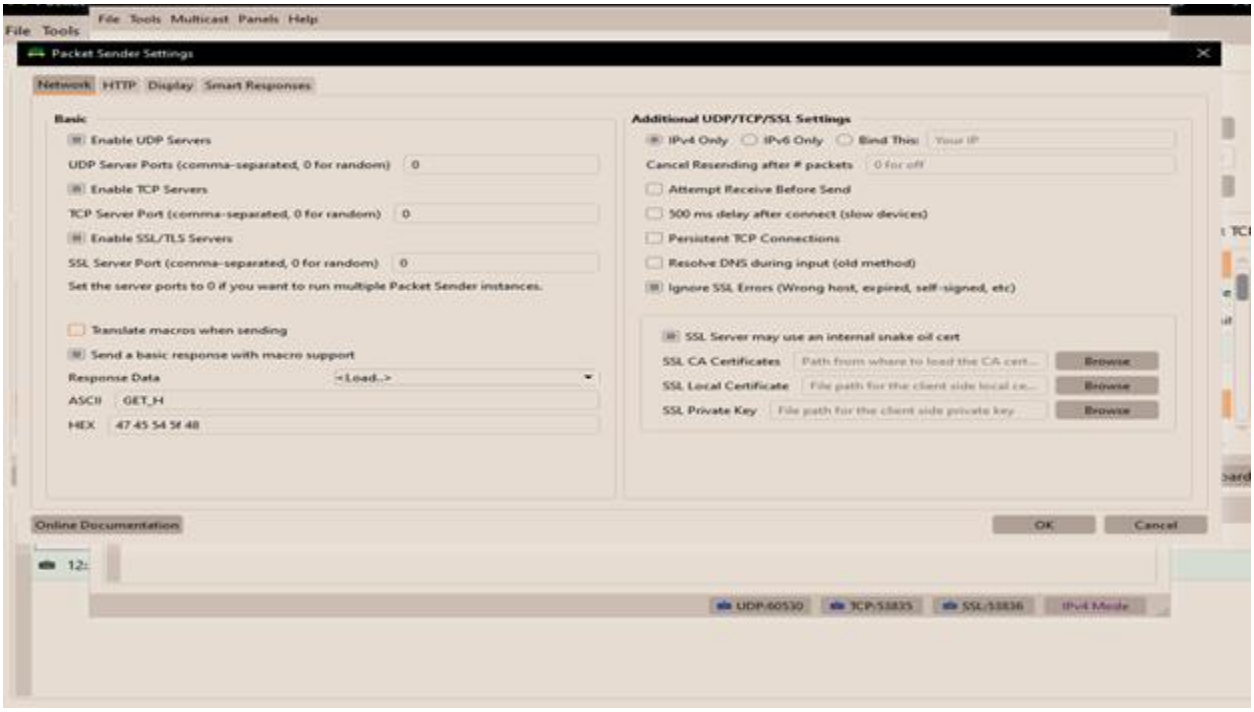


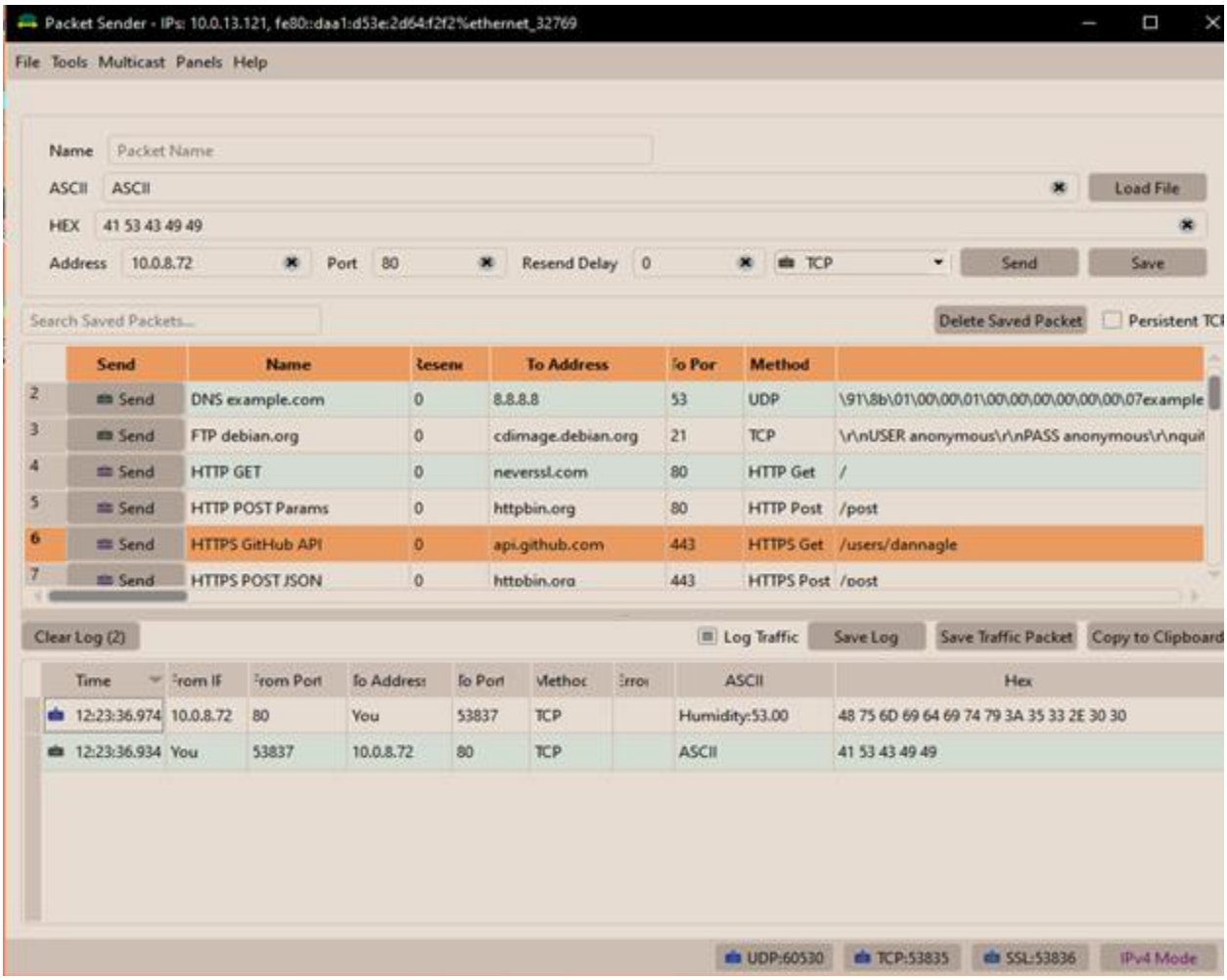
Sl.NO.	NodeMCU	DHT11
1	Vin	VCC
2	GND	GND
3	D5	Data Out

Coding:

```
#include <ESP8266WiFi.h>
#include "DHT.h"
DHT dht(14,DHT11); //D1 on board numbering system & 5 GPIO numbering system
// Replace with your network credentials
const char* ssid = "KWVOF";
const char* password = "ece@1234";

WiFiServer server(80); // Create a server instance on port 80
void setup() {
Serial.begin(115200);
delay(10);

// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");

server.begin(); // Start the server
}

void loop() {
WiFiClient client = server.available(); // Check for a client connection

if(client) {
```

```
Serial.println("New client connected");
float humidity = dht.readHumidity();
String response = "Humidity:"; // Replace this with your actual humidity data
//send the humidity data
client.print(response);
client.print(humidity);

//close the connection
client.stop();
Serial.println("client disconnected");
}

}
```





RESULT: Received humidity data to TCP client from TCP server on NodeMCU (ESP8266)has successfully executed

EXPERIMENT – 12

To install MySQL database on Raspberry Pi and perform basic SQL queries.

AIM: To perform basic SQL queries by installing MySQL database on Rsapberry Pi.

Please follow the following steps to know how to install MYSQL server on raspberry pi or to know how to install MariaDB server on raspberry pi.

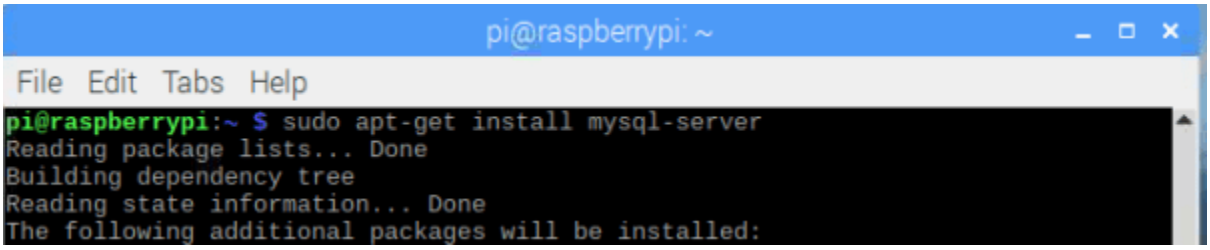
1)Please open the raspberry pi terminal.

2)Execute the following command to update the existing packages.

```
sudo apt-get update
```

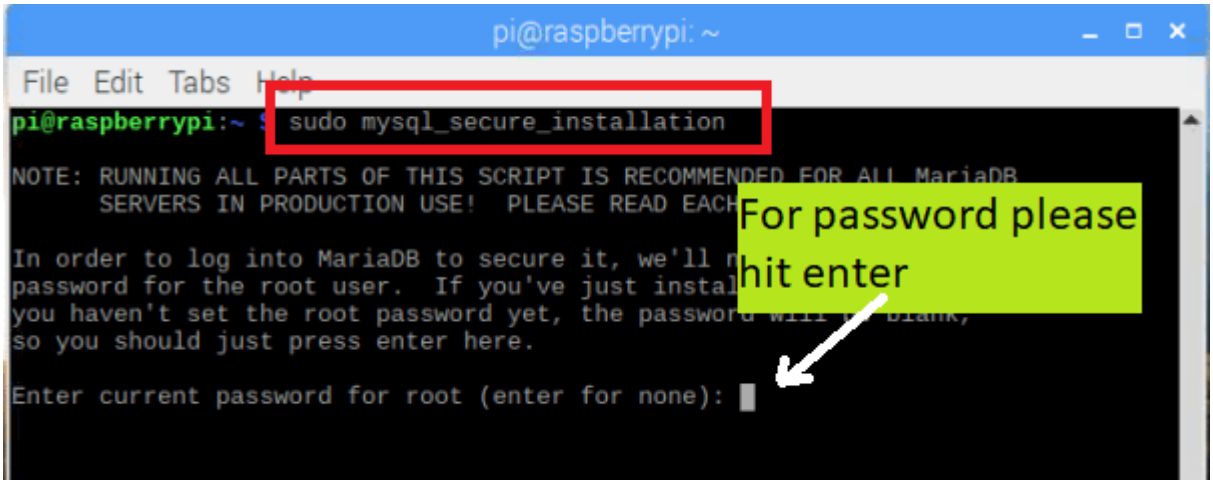
3)Now execute the following to install MySQL server which is shown below. While installing if it is asking do you want to continue then please enter y and hit enter

```
sudo apt-get install mysql-server
```



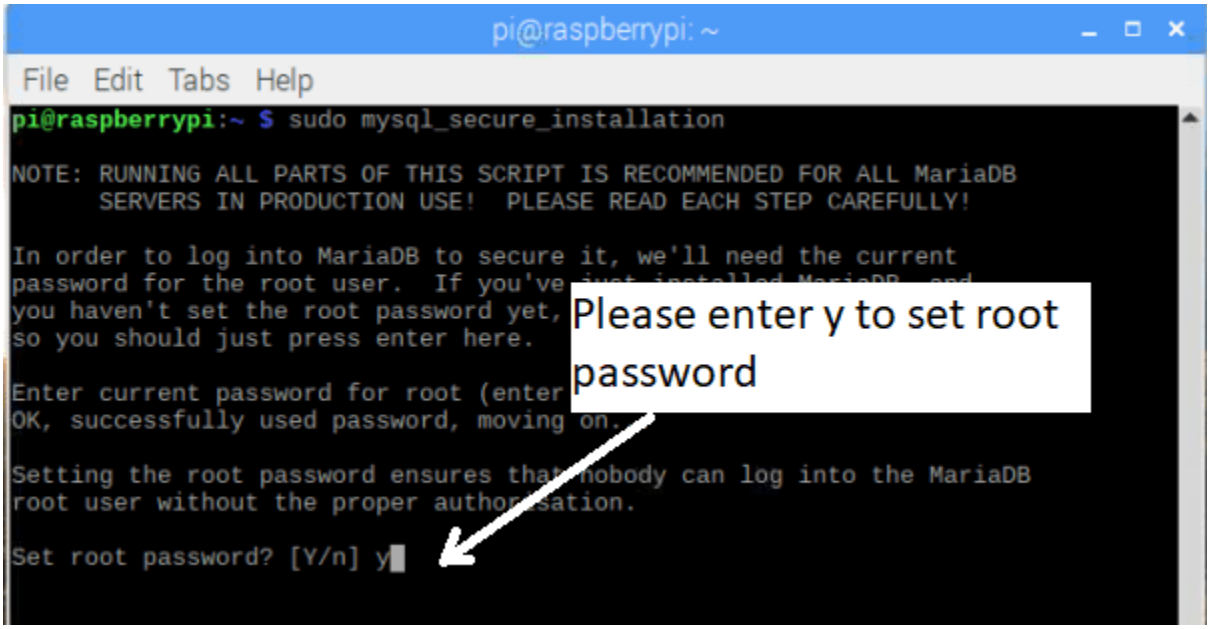
4)Now please execute the following command for secure installation which is shown below.

```
sudo mysql_secure_installation
```



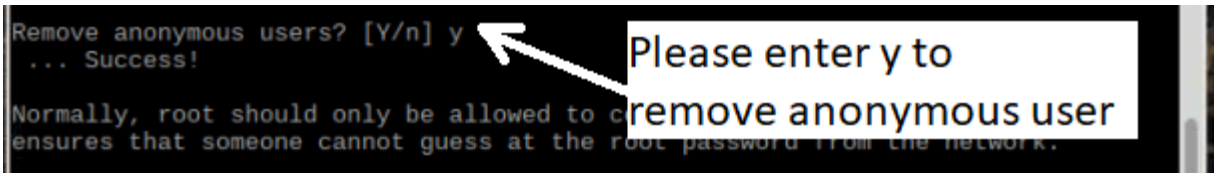
5)Please hit Enter for current root password.

6)Now please Enter y and hit Enter for setting a new password which is shown below

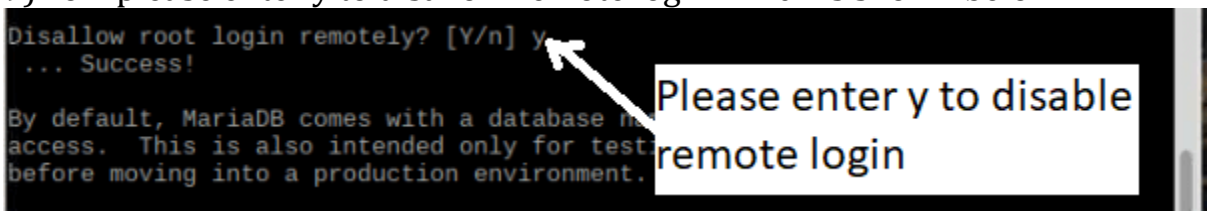


7)Now Please enter New password which is shown below

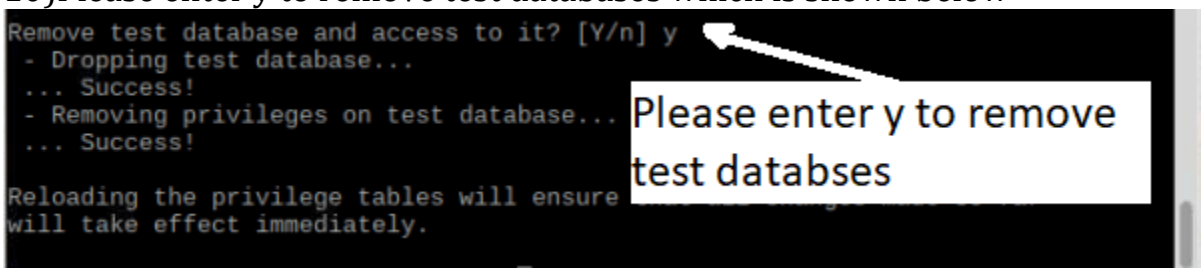
8)Now please enter y to remove anonymous user and hit Enter.



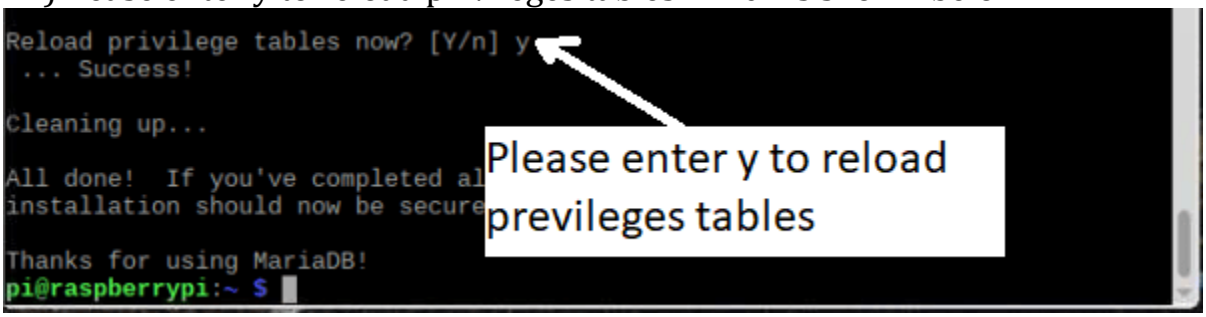
9)Now please enter y to disallow remote login which is shown below



10)Please enter y to remove test databases which is shown below

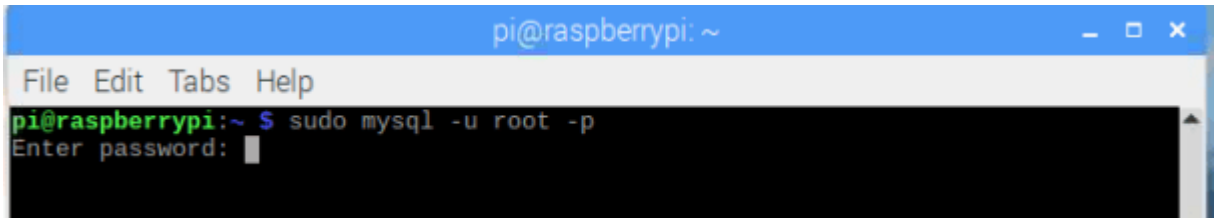


11)Please enter y to reload privileges tables which is shown below



12)Now please execute the following command to login into the database and Enter the password which you have entered in step 7.

sudo mysql -u root -p



13)Please execute the following command to see databases present in the mysql database.

show databases;

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.01 sec)

MariaDB [(none)]>
```

14)Execute the following to create Demo database in mysql server which is shown below.

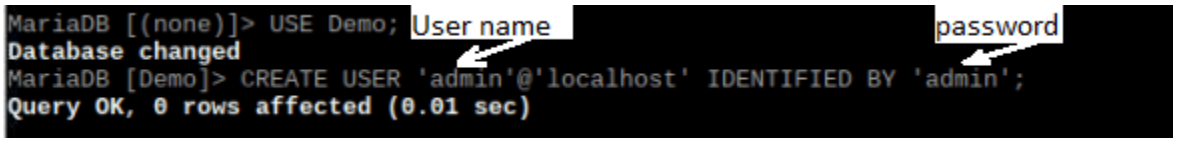
CREATE DATABASE Demo;

15)Now please execute the following to go in Demo database

USE Demo;

16)Please execute the following command to create database user

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';



17)Execute the following command to grant all previleges

GRANT ALL PRIVILEGES ON Demo.* TO 'admin'@'localhost';

```
MariaDB [(none)]> USE Demo;
Database changed
MariaDB [Demo]> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';
Query OK, 0 rows affected (0.01 sec)

MariaDB [Demo]> GRANT ALL PRIVILEGES ON Demo.* TO 'admin'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

18)Now execute the following command save all the changes

FLUSH PRIVILEGES;

```
MariaDB [Demo]> GRANT ALL PRIVILEGES ON Demo.* TO 'admin'@'localhost';
Query OK, 0 rows affected (0.00 sec)

MariaDB [Demo]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

19)Now please execute the following command to come out of database.

quit

```
MariaDB [Demo]> quit
Bye
pi@raspberrypi:~ $
```

20)Execute the following command to restart the MYSQL server

sudo service mysql restart

```
pi@raspberrypi:~ $ sudo service mysql restart
pi@raspberrypi:~ $
```

How to insert and fetch data from MySQL database

Please follow the following steps to insert and fetch from the MySQL database.
1)Open the raspberry pi terminal

2)Execute the following command to login to the database and enter the password which is shown below.

sudo mysql -u root -p

```
pi@raspberrypi:~ $ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.38-MariaDB-0+deb9u1 Raspbian 9.0

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use Demo
Database changed
MariaDB [Demo]> create table login(
    -> username varchar(25),password varchar(25));
Query OK, 0 rows affected (0.05 sec)

MariaDB [Demo]>
```

3)Execute the following command to use Demo database which is shown above.

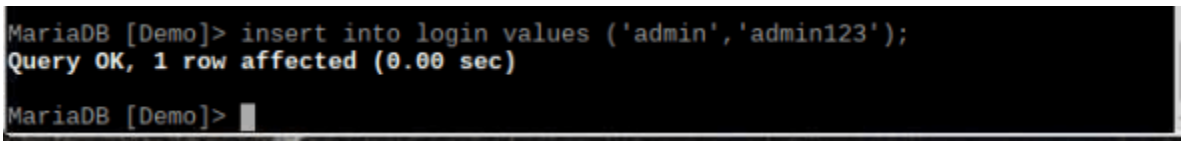
USE Demo;

4)Execute the following command to create login table which has two coloums i.e is username and password which is shown above.

create table login(username varchar(25), password varchar(25));

5)Execute the following command to insert data into login table which is shown below.

insert into login values('admin','admin123');



6)To see the inserted values please execute the following command which is shown below

select * from login;

RESULT: Performed basic SQL queries by installing MySQL database on Rsapberry Pi