Maratha Mandal's Engineering College, Belagavi

Department of Electronics and Communication Engineering

Laboratory Manual

Subject: **IoT (Internet of Things) Lab**

Subject Code: 21EC581

Prepared By

Prof.Vaibhav Kakade
Assistant Professor
Department of Electronics & Communication Engineering

Syllabus

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

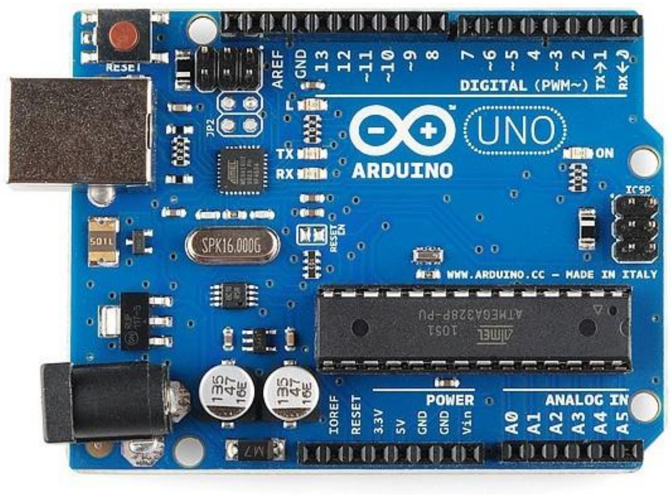
Suggested Learning Resources:

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

Contents

Introduction to Arduino UNO

Arduino Uno is an open-source microcontroller board based on the ATmega328P microcontroller. It is designed to be easy to use and is a popular choice for beginners in the field of electronics and programming. The board has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

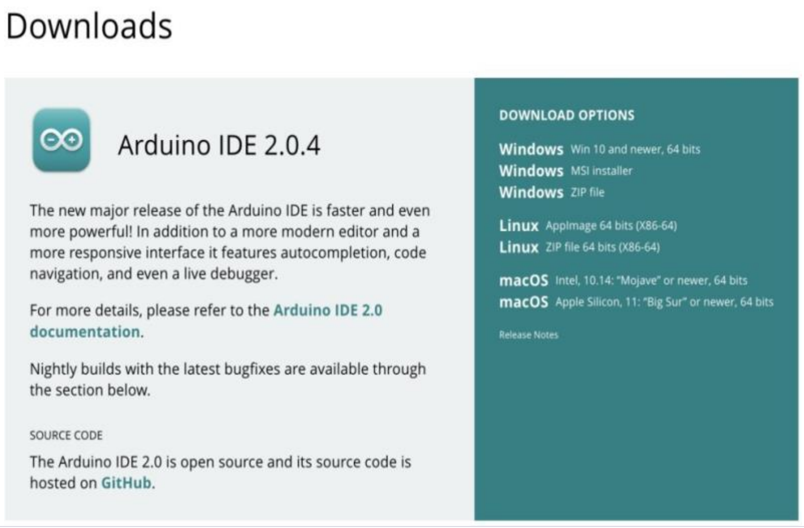
The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

Installation steps for Arduino IDE

Here are the steps to install the Arduino IDE on your computer:

1.	Go to the official Arduino website at https://www.arduino.cc/en/software

Click on the download link for your operating system (Windows, Mac, or Linux).


Downloads

Arduino IDE 2.0.4

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

DOWNLOAD OPTIONS

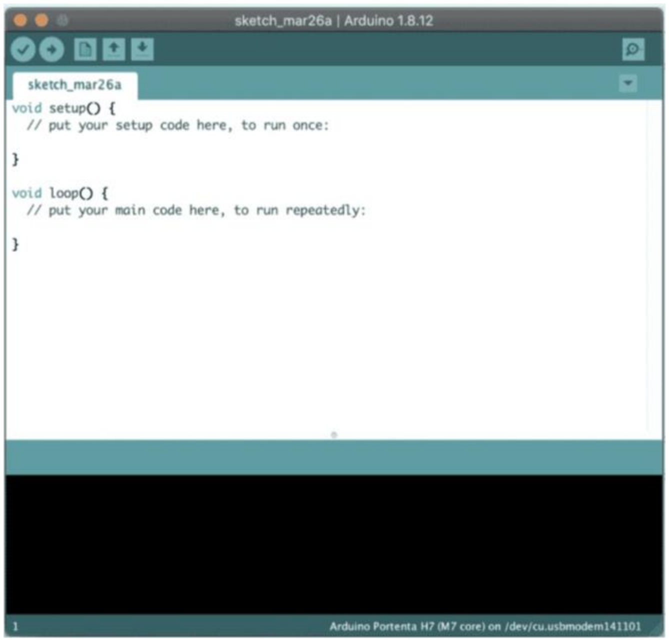
Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

2. Once the download is complete, run the installer file.Follow the prompts to complete the installation.
3. When the installation is finished, launch the Arduino IDE.


sketch_mar26a | Arduino 1.8.12

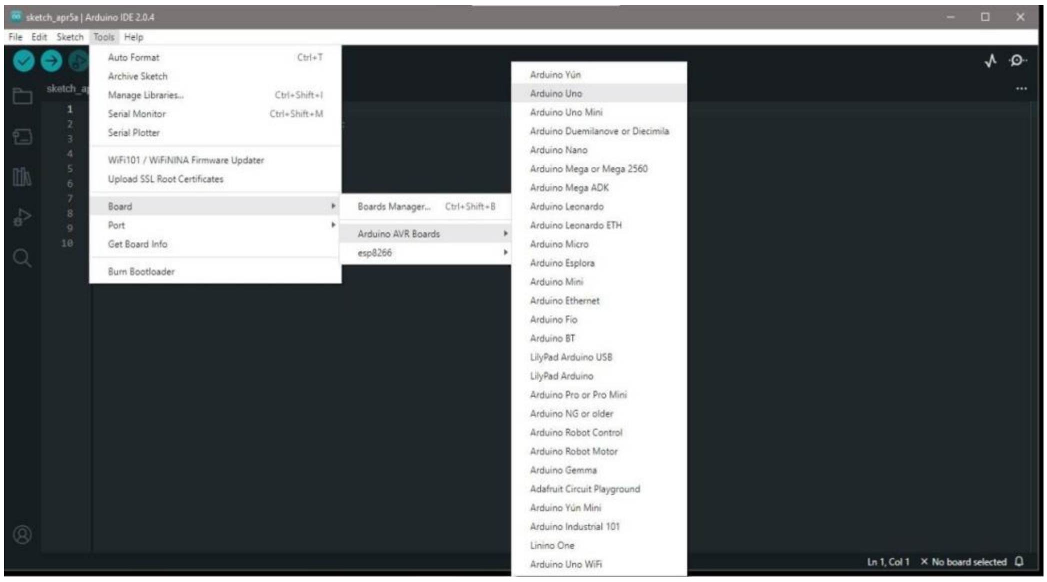
sketch_mar26a

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

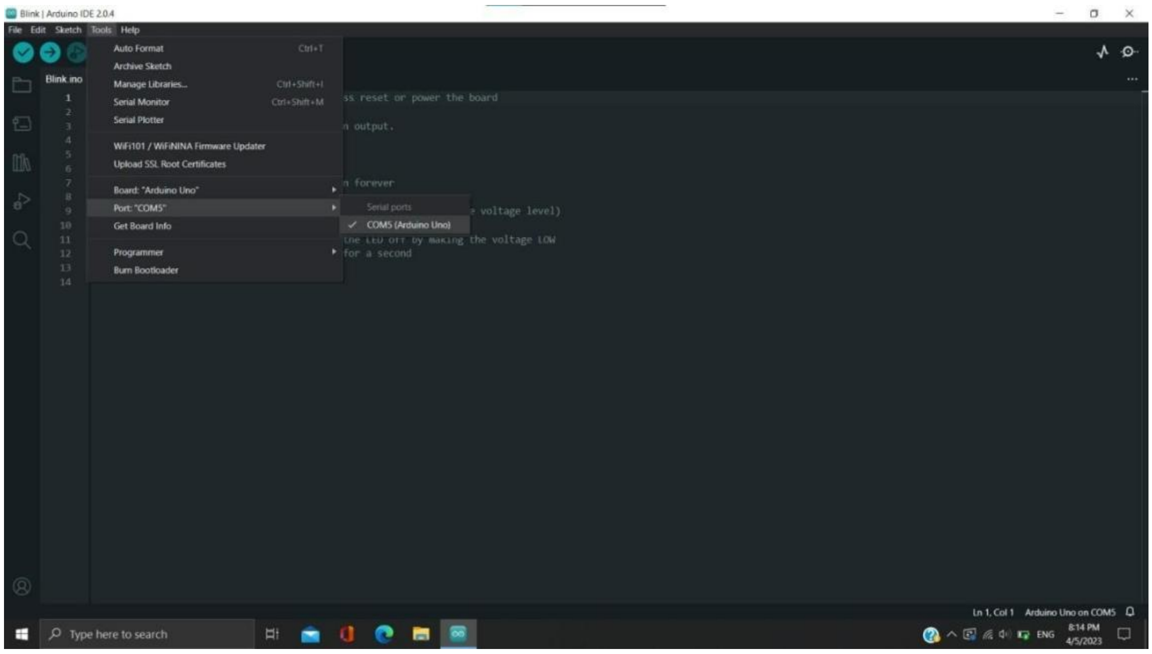
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101

Sample Program to toggle LED in Arduino UNO

1. Go to Tools >> Board >> Arduino AVR Boards >> select Arduino Uno

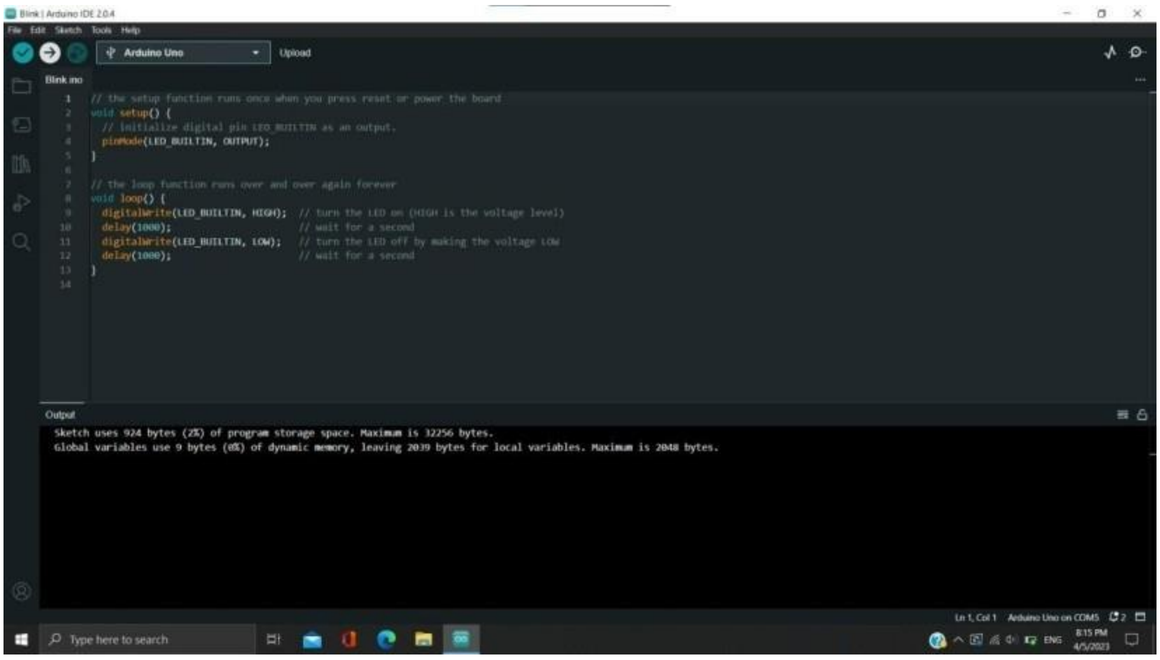


2. Select the Port from Tools menu


Blink | Arduino IDE 2.0.4
File Edit Sketch Tools Help
Blink.ino
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM5"
Get Board Info
Programmer
Burn Bootloader
Serial ports
✓ COM5 (Arduino Uno)
Ln 1, Col 1 Arduino Uno on COM5
Type here to search
ENG 8:14 PM 4/5/2023

```
ss reset or power the board
n output.
n forever
e voltage level)
the LED off by making the voltage LOW
for a second
```

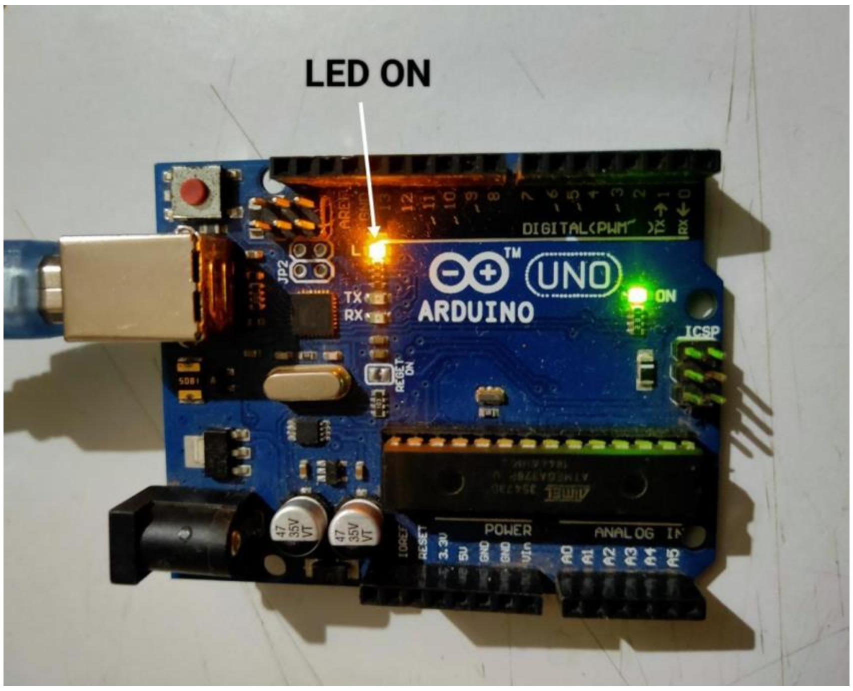
3. Compile the code and click on upload to dump the code in Arduino Uno


Blink | Arduino IDE 2.0.4
File Edit Sketch Tools Help
Arduino Uno
Upload
Blink.ino
Output
Ln 1, Col 1 Arduino Uno on COM5
Type here to search

```
// 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(1000);                      // wait for a second
}
```

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

4. Result



<u>Experiment No 1 A</u>

Aim: To interface LED/Buzzer with Arduino and write a program to 'turn ON' LED for 1 sec after every 2 seconds.

Components Required:

Sl No	Components	Quantity
1	Arduino Uno Board	1
2	LED or Buzzer	1
3	USB cable for Arduino UNO	1
4	Connecting wires	1

Circuit Connection:

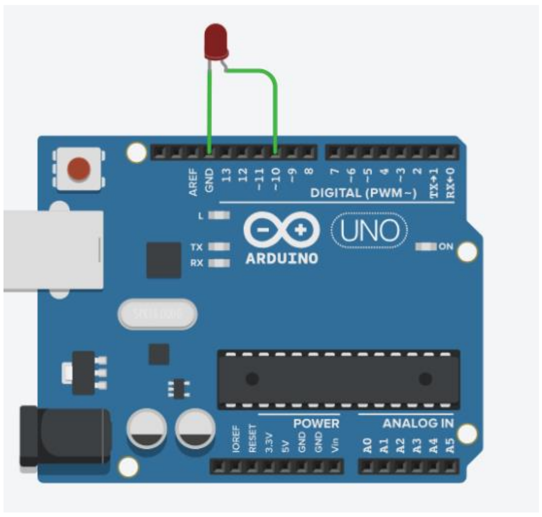


Figure: Interfacing LED

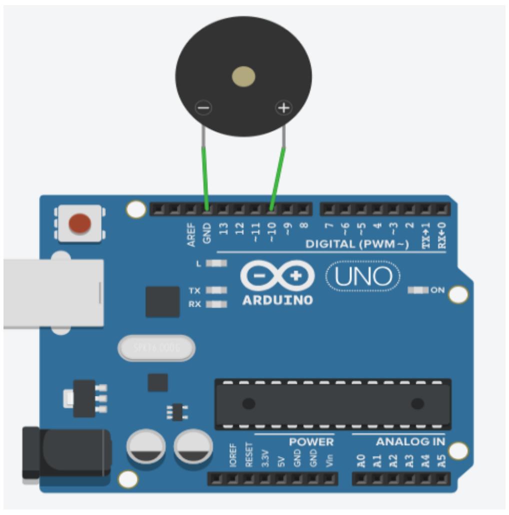
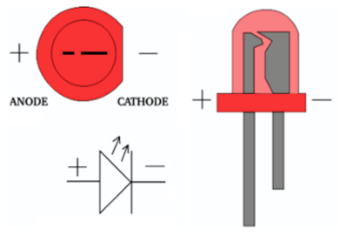


Figure: Interfacing Buzzer

Theory: LED, which stands for Light Emitting Diode, is a semiconductor device that emits light when an electric current passes through it. LEDs are widely used for various applications due to their energy efficiency, long lifespan, and versatility. Here are some key points about LEDs.

Basic Operation: LEDs work on the principle of electroluminescence. When electrons and holes (positive counterparts of electrons) recombine within the semiconductor material, they release energy in the form of photons, which produces light.



Item	Min	Max	Unit
Forward Current	20	30	mA
Forward Voltage	1.8	2.2	V

Arduino Code to interface LED/ Buzzer

```
void setup()

{

   pinMode(10, OUTPUT); // initialize digital pin 10 as an output.

}

void loop() {

  digitalWrite(10, HIGH);  // turn the LED on (HIGH is the voltage level)

  delay(1000);                  // wait for 1 second

  digitalWrite(10, LOW);   // turn the LED off by making the voltage LOW

  delay(2000);                  // wait for 2 second

}
```

Result: LED will blink continuously; It will be ON for 1 second and OFF for 2 seconds. If we connect buzzer in place of LED then buzzer will sound for 1 second and buzzer will be OFF for 2 seconds.

Experiment No 1 B

Aim: To interface Push button with Arduino and write a program to 'turn ON' LED when push button is pressed.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO BOARD	1
2	LED	1
3	USB cable for ARDUINO	1
4	Connecting wires	--
5	Push Button	1
6	Bread Board	1
7	Resistance 220ohm and 1Kohm	1

Circuit Connection:

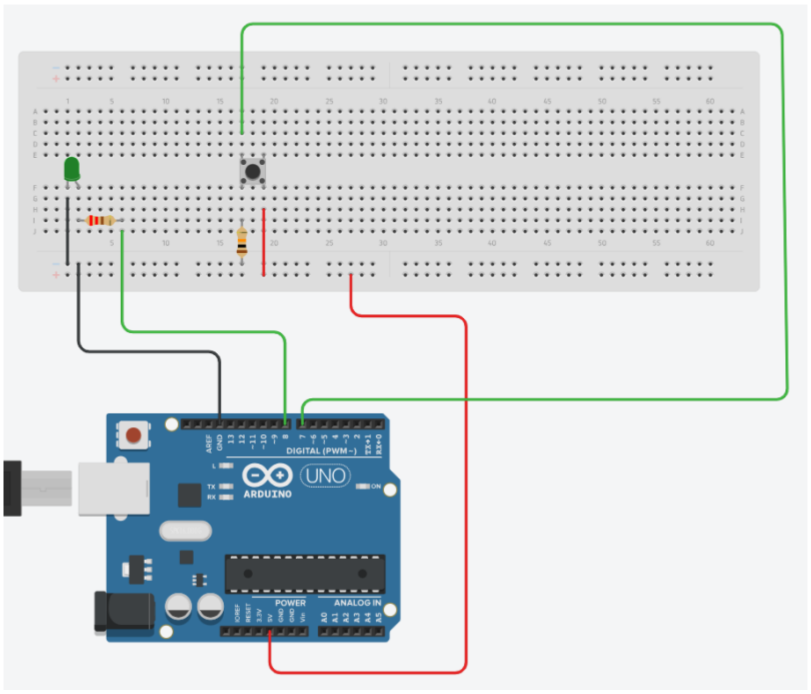


Figure: Controlling LED using Push Button switch

Arduino Code to interface push button to control LED

```
#define LED_PIN 8
```

```
#define BUTTON_PIN 7
void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);

}

void loop()
{
  if (digitalRead (BUTTON_PIN)==HIGH)
  {
  digitalWrite(LED_PIN, HIGH);
  }
  else
  {
   digitalWrite(LED_PIN, LOW);

  }

}
```

Result: LED will be ON when push button is pressed and LED will be OFF when button is released.

Experiment No 1 C

Aim: To interface Digital sensor (LDR) with Arduino and write a program to 'turn ON' LED when at sensor detection.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO BOARD	1
2	LED	1
3	USB cable for ARDUINO	1
4	Connecting wires	--
5	LDR	1
6	Bread Board	1
7	Resistance 220ohm and 1Kohm	1

Circuit Connection:

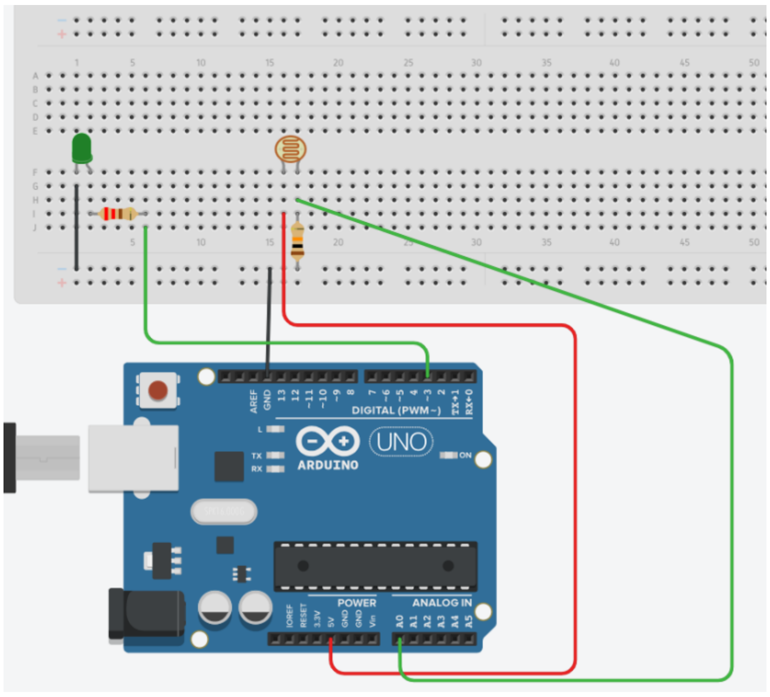


Figure: Controlling LED using LDR

Theory:

LDR (Light Dependent Resistor) as the name states is a special type of resistor that works on the photoconductivity principle means that resistance changes according to the intensity of light. Its resistance decreases with an increase in the intensity of light.

It is often used as a light sensor, light meter, Automatic street light, and in areas where we need to have light sensitivity. LDR is also known as a Light Sensor. LDR are usually available in 5mm, 8mm, 12mm, and 25mm dimensions.

It works on the principle of photoconductivity whenever the light falls on its photoconductive material, it absorbs its energy and the electrons of that photoconductive material in the valence band get excited and go to the conduction band and thus increasing the conductivity as per the increase in light intensity.

Also, the energy in incident light should be greater than the bandgap gap energy so that the electrons from the valence band got excited and go to the conduction band.

The LDR has the highest resistance in dark around 1012 Ohm and this resistance decreases with the increase in Light.

Arduino Code to interface LDR to control LED

```
int ldr=A0;//Set A0(Analog Input) for LDR.
int value=0;
void setup() {
Serial.begin(9600);
pinMode(3,OUTPUT);
}
void loop() {
value=analogRead(ldr);//Reads the Value of LDR(light).
Serial.println("LDR value is :");//Prints the value of LDR to
Serial Monitor.
Serial.println(value);
if(value<300)
  {
    digitalWrite(3,HIGH);//Makes the LED glow in Dark.
  }
  else
  {
    digitalWrite(3,LOW);//Turns the LED OFF in Light.
  }
}
```

Result: LED will be ON when no light fall on LDR that is during dark LED will be ON and during bright light LED will be turned OFF.

Experiment No: 2A

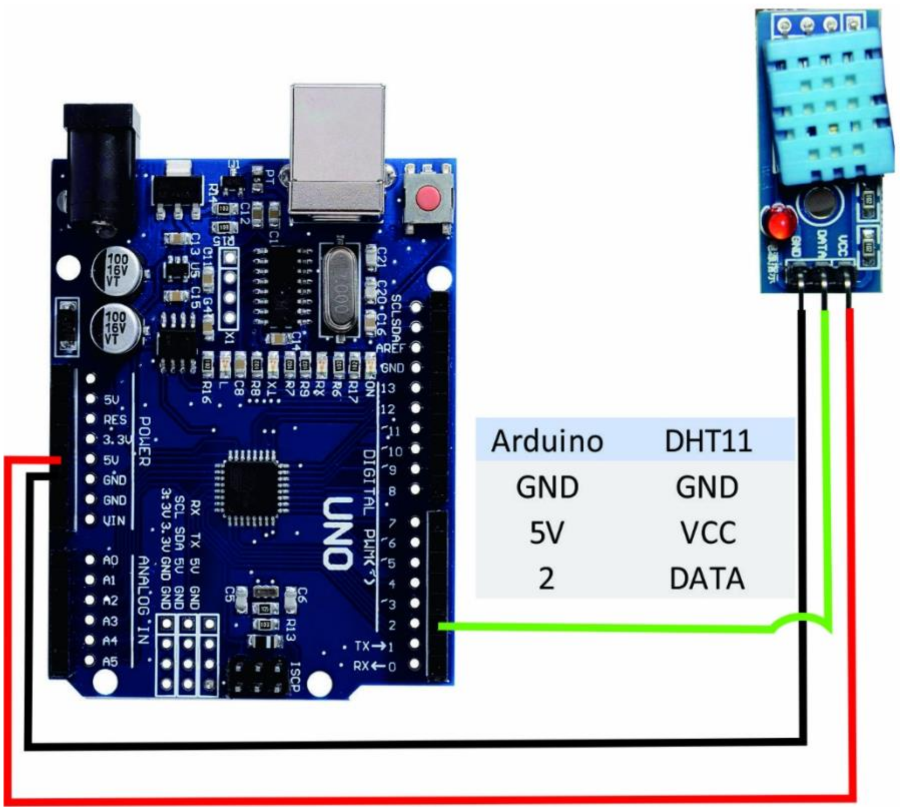
Aim: To interface DHT11 sensor with Arduino and write a program to print temperature and humidity readings.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO BOARD	1

2	DHT11 Temperature sensor	1
3	USB cable for ARDUINO Board	1
4	Connecting wires	--

Circuit Connection:



Theory:

The **DHT11 sensor** is a **low-cost** digital temperature and humidity sensor. It operates at a **voltage of 3.3V to 5V** and can measure temperatures ranging from **0°C to 50°C** with an accuracy of ±2°C. Additionally, it can measure relative humidity ranging from **20% to 90%** with an accuracy of ±5% .

The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature.

DHT11 Specifications

- Operating Voltage: 3.5V to 5.5V

- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the LiquidCrystal library makes it easy to talk to character LCD displays.

There are thousands of libraries available for download directly through the Arduino IDE, and you can find all of them listed at the Arduino Library Reference.

Steps to Add DHT11 Sensor Library

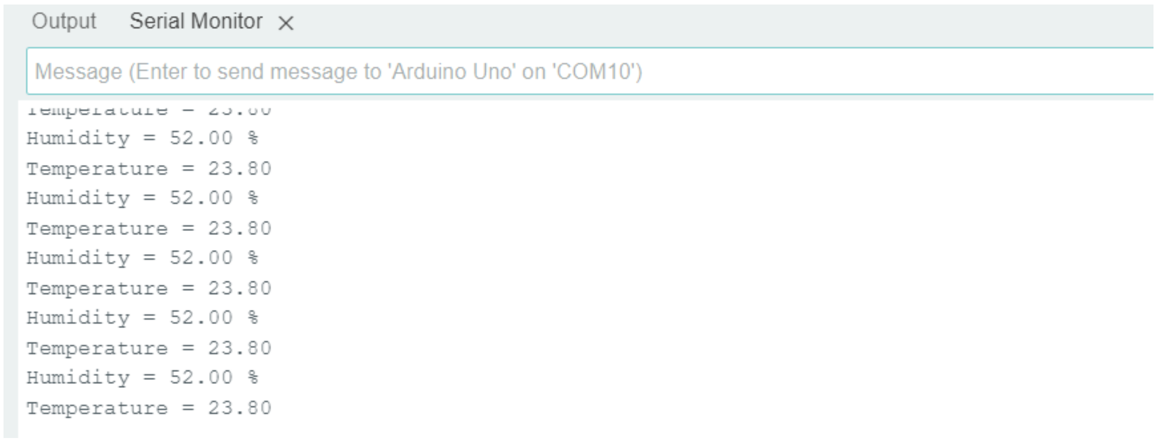
1) Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2) Search DHT then find the DHT Sensor library by Adafruit
3) Click install button to install library
4) If ask click on install all button to install library dependencies

Arduino Code to interface DHT 11 sensor to read Temperature & Humidity:

```
#include <DHT.h>
#define DHTPIN 2 //DHT11 out pin is connected to pin2 of arduino
#define DHTTYPE DHT11
DHT dht(DHTPIN,  DHTTYPE);
float humidity;
float temp;
```

```
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  humidity= dht.readHumidity();
  Serial.print("Humidity = ");
  Serial.print(humidity, 2);
  Serial.println (" %");

  temp=dht.readTemperature();
  Serial.print("Temperature = ");
  Serial.println(temp,2);
  delay(500);
}
```

Output:



```
Temperature = 23.80
Humidity = 52.00 %
Temperature = 23.80
Humidity = 52.00 %
Temperature = 23.80
Humidity = 52.00 %
Temperature = 23.80
Humidity = 52.00 %
Temperature = 23.80
Humidity = 52.00 %
Temperature = 23.80
```

Result: Temperature and Humidity values are read and displayed on to serial monitor by interfacing DHT11 sensor with Arduino.

Experiment No: 3

Aim: To interface Bluetooth with Arduino and write a program to send sensor data to smartphone using Bluetooth.

Component Required:

Sl No	Components	Quantity
1	Arduino UNO board	1
2	Bluetooth Module	1
3	USB cable for Arduino	1

4	DHT11 sensor	1
5	Connecting wires	---
6	Smart Phone with RC Controlled Bluetooth app	1

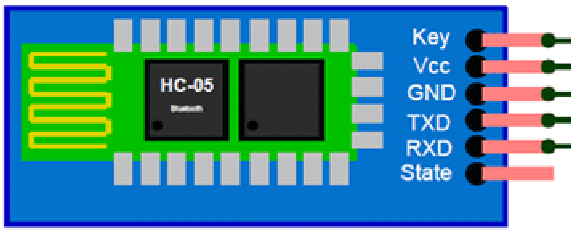
Circuit Connection:

Pin Connection details

Arduino Pins	Bluetooth Module	Sensor DHT11
5V	VCC	
GND	GND	GND
5V	--	VCC
4	--	OUT
Rx	Tx	
Tx	RX	

Theory:

The HC-05 is a popular module which can add two-way (full-duplex) wireless functionality. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available which makes this process a lot easier.



The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. So if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you.

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

Procedure to check output:

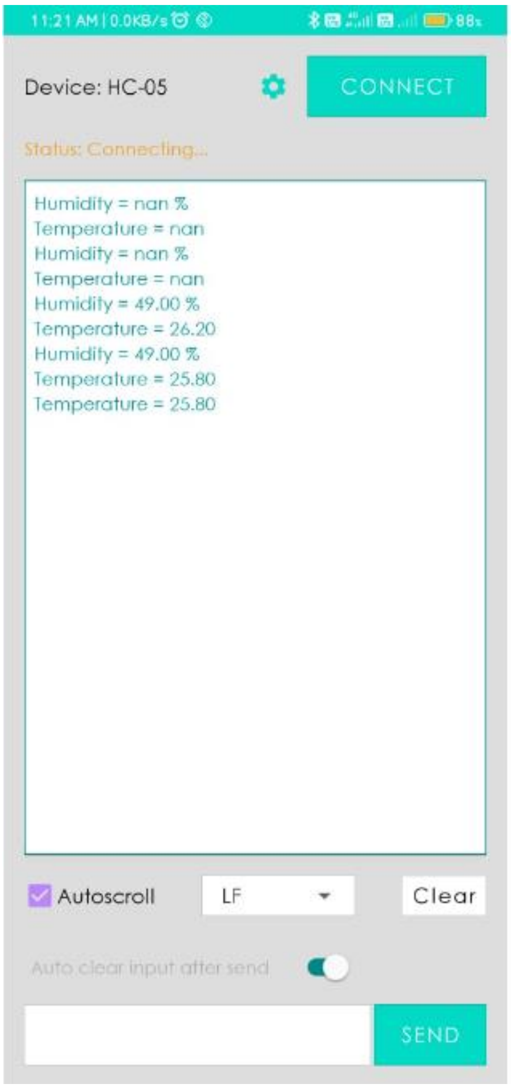
1. Switch on Bluetooth on Mobile and pair HC05 device.
2. Install Bluetooth Serial Monitor app from Google Play Store.
3. Scan Classic Bluetooth button, Select device HC05. On monitor displayed the temperature and humidity readings.

Arduino Code to interface HC05 Bluetooth module and DHT11 Sensor

```
#include <DHT.h>


#define DHTPIN 4

#define DHTTYPE DHT11

DHT dht(DHTPIN,  DHTTYPE);


float humidity;

float temp;

char data;
```

```
void setup() {
  Serial.begin(9600);
  dht.begin();

}

void loop() {
  data =temp;

  humidity= dht.readHumidity();
  Serial.print("Humidity = ");
  Serial.print(humidity, 2);
  Serial.println (" %");

  temp=dht.readTemperature();
  Serial.print("Temperature = ");
  Serial.println(temp,2);

  delay(500);

}
```

Result: Bluetooth is interfaced with Arduino and sent sensor DHT11 Temperature data to smartphone using Bluetooth.



```
Humidity = nan %
Temperature = nan
Humidity = nan %
Temperature = nan
Humidity = 49.00 %
Temperature = 26.20
Humidity = 49.00 %
Temperature = 25.80
Temperature = 25.80
```

Experiment No: 4

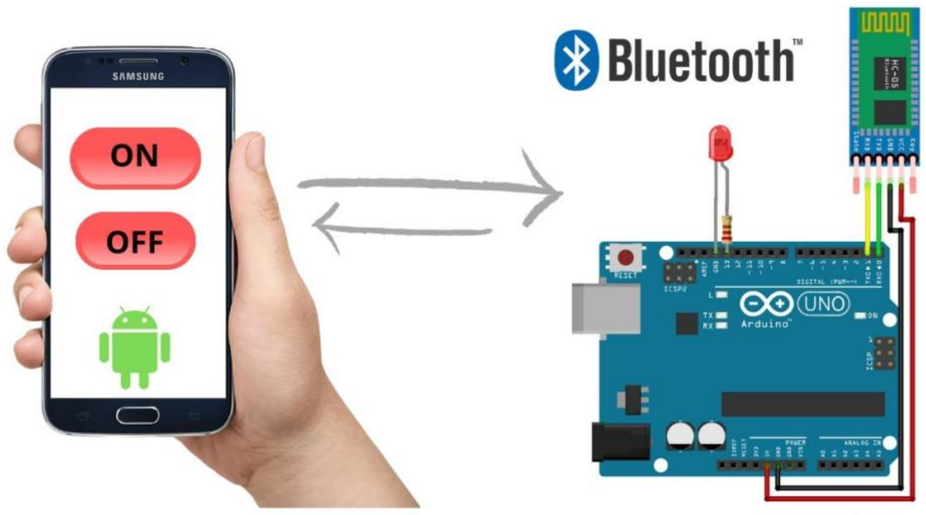
Aim: To interface Bluetooth with Arduino and write a program to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.

Component Required:

Sl No	Components	Quantity
1	ARDUINO UNO BOARD	1
2	Bluetooth Module HC-05	1
3	USB cable for ARDUINO	1
4	Connecting wires	--
5	LED	1
6	Smart Phone with RC Controlled Bluetooth app	1

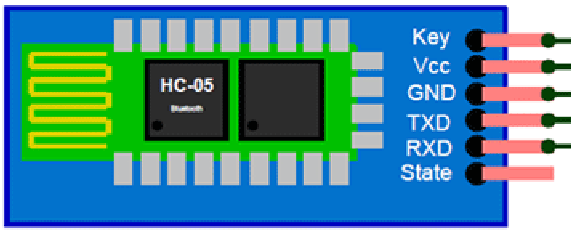
Pin Connection details		
Arduino Pins	**Bluetooth Module**	**LED**
5 V	VCC	
GND	GND	Cathode
Rx	Tx	
Tx	RX	
13	--	Anode

Circuit Connection:



Theory

Bluetooth is a low-power wireless connectivity technology used to stream audio, transfer data and broadcast information between devices. Operating frequency is: 2.4 to 2.485 GHz(UHF)



The HC-05 is a popular module which can add two-way (full-duplex) wireless functionality. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop. There are many android applications that are already available which makes this process a lot easier. The module communicates with the help of USART at 9600 baud rate hence it is easy to interface with any microcontroller that supports USART. So if you looking for a Wireless module that could transfer data from your computer or mobile phone to microcontroller or vice versa then this module might be the right choice for you.

HC-05 Pinout Configuration	
Pin Name	**Description**
Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default, it is in Data mode
Vcc	Powers the module. Connect to +5V Supply voltage
Ground	Ground pin of module, connect to system ground.
TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
State	The state pin is connected to on board LED, it can be used as feedback to check if Bluetooth is working properly.
LED	Indicates the status of Module • Blink once in 2 sec: Module has entered Command Mode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
Button	Used to control the Key/Enable pin to toggle between Data and command Mode

Mobile app

Connecting the Android device to the HC-05 creates a serial communication channel very similar to the serial monitor in the Arduino IDE. This means we need a Bluetooth version of the serial monitor.

Procedure to connect Bluetooth with mobile app

1. Download " **Serial Bluetooth Terminal** " app from play store and install in your mobile.

2. Pair your phone with HC-05. for doing this go to *Settings->Bluetooth->Scan device->select HC-05* and pair it. Pass code to pair is '1234'.

3. Now, open the app and connect the HC-05 module in **terminal mode**.

4. Type "0" in terminal to turn OFF LED.

5. Type "1" in terminal to turn ON LED,

Result: If you send '0' it will make to LED ON and if you send '1' it will make the LED OFF. The Arduino code also sends the current state of LED also.

Arduino Code to control LED with HC05 Bluetooth Module

```
#define led 13
char data;

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}
void loop() {
  if(Serial.available()>0);{
    data=Serial.read();
  }
  if(data =='0'){
     Serial.println("LED OFF");
    digitalWrite(led, LOW);
  }
  if(data=='1'){
    Serial.println("LED ON");
    digitalWrite(led, HIGH);
  }
  delay(300);
}
```

Result: Arduino program written and verified to turn LED ON/OFF when '1'/'0' is received from smartphone using Bluetooth.

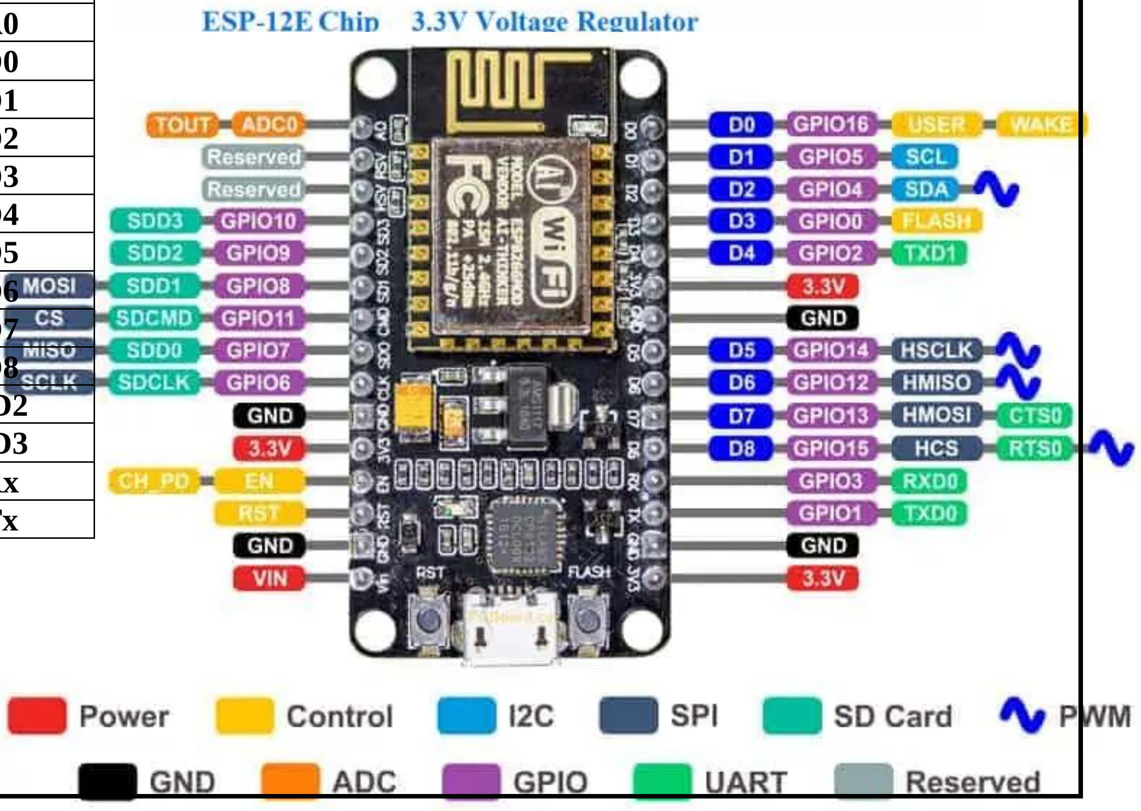
Introduction to Node MCU ESP8266 module

The NodeMCU (*N*ode *M*icro-Controller *U*nit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.

NodeMCU ESP8266 Specifications & Features and Pinout

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

PIN	CODE
A0	A0
GPIO 16	D0
GPIO 5	D1
GPIO 4	D2
GPIO 0	D3
GPIO 2	D4
GPIO14	D5
GPIO 12	D6
GPIO 13	D7
GPIO 15	D8
GPIO 9	SD2
GPIO10	SD3
GPIO3	Rx
GPIO1	Tx



Steps to install Node MCU

1. Download and install Arduino IDE
2. Open the IDE and follow this path. **File -> preferences -> Additional board manager URL.**
3. Now paste the URL in the dialog box **:** **http://arduino.esp8266.com/stable/package_esp8266com_index.json**
4. Then, click the "**OK**" button.
5. Now follow this path**. Tools -> Board -> Boards Manager**
6. Search for **ESP8266** and install the "**ESP8266 by ESP8266 Community**"
7. After this, restart your Arduino IDE.
8. Then, go to **Tools** > **Board** and check that you have ESP8266 boards available.
9. First, make sure you have an ESP8266 selected in **Tools** > **Board**. If you're using the ESP8266-12E NodeMCU Kit as shown in previous pictures, select the **NodeMCU 1.0 (ESP-12E Module)** option.

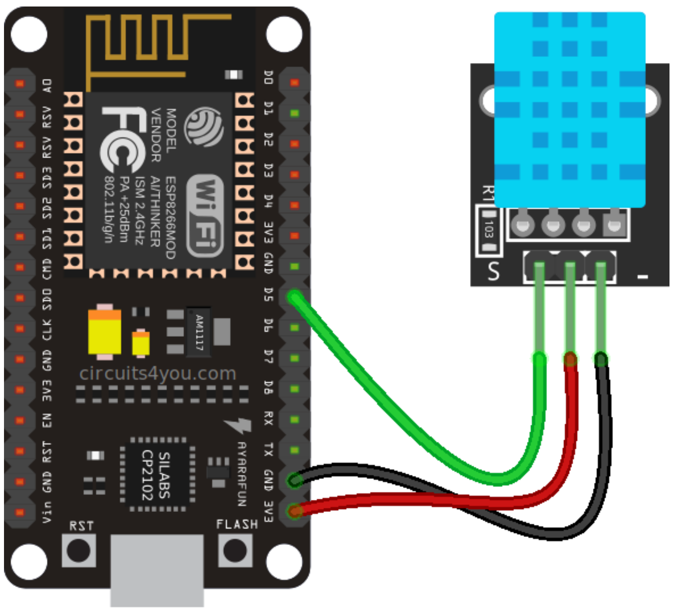
Experiment No: 5

Aim: Write a program on Arduino to upload temperature and humidity data to thingspeak cloud.

Component Required:

Sl No	Components	Quantity
1	ESP8266 12E Node MCU Kit	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--
5	Breadboard	1

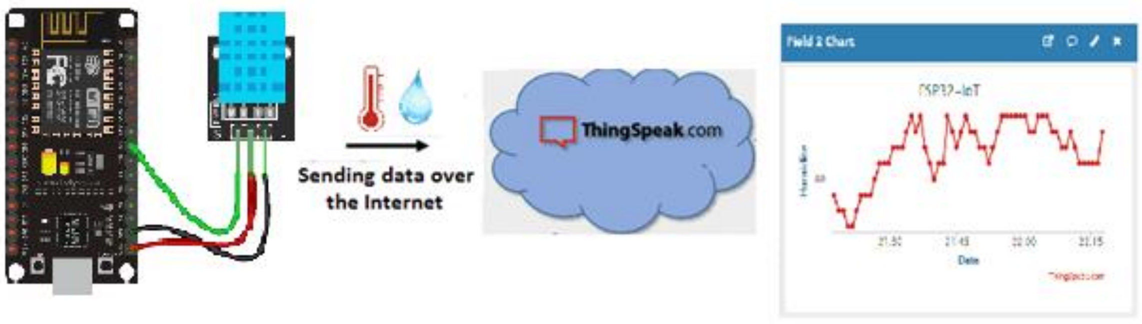
Circuit Connection



NODE MCU	DHT11
GND	GND
3V3	VCC
D5	DATA

Theory

ThingSpeak is an open-source Internet of Things (IoT) application and API that allows users to collect and store sensor data in the cloud and perform analytics on that data. It allows users to create “channels” to collect data from multiple sensors, and also has built-in support for visualizing and analyzing the data. **ThingSpeak** can be used for a variety of applications, such as monitoring environmental conditions, tracking the location of assets, and controlling devices remotely. It is available for free and also has paid subscription plans for additional features and support. The device that sends the data must be configured with the correct channel information, such as the channel ID and write API key.



Steps to Connect

Step 1: ThingSpeak Setup for Temperature and Humidity Monitoring

a. For creating your channel on Thingspeak, you first need to Sign up on Thingspeak.

b. In case if you already have an account on Thingspeak, just sign in using your id and password.

c. For creating your account go to www.thingspeak.com.

d. After this, verify your E-mail id and click on continue.

Step 2: Create a Channel for Your Data

a. Once you Sign in after your account verification, create a new channel by clicking "New Channel" button.

b. After clicking on "New Channel", enter the Name and Description of the data you want to upload on this channel.

c. Enter the name of your data 'Temperature' in Field1 and 'Humidity' in Field2.

d. Click on the save channel button to save your details.

Step 3: API Key

a. To send data to Thingspeak, we need a unique API key, which we will use later in our code to upload our sensor data to Thingspeak Website.

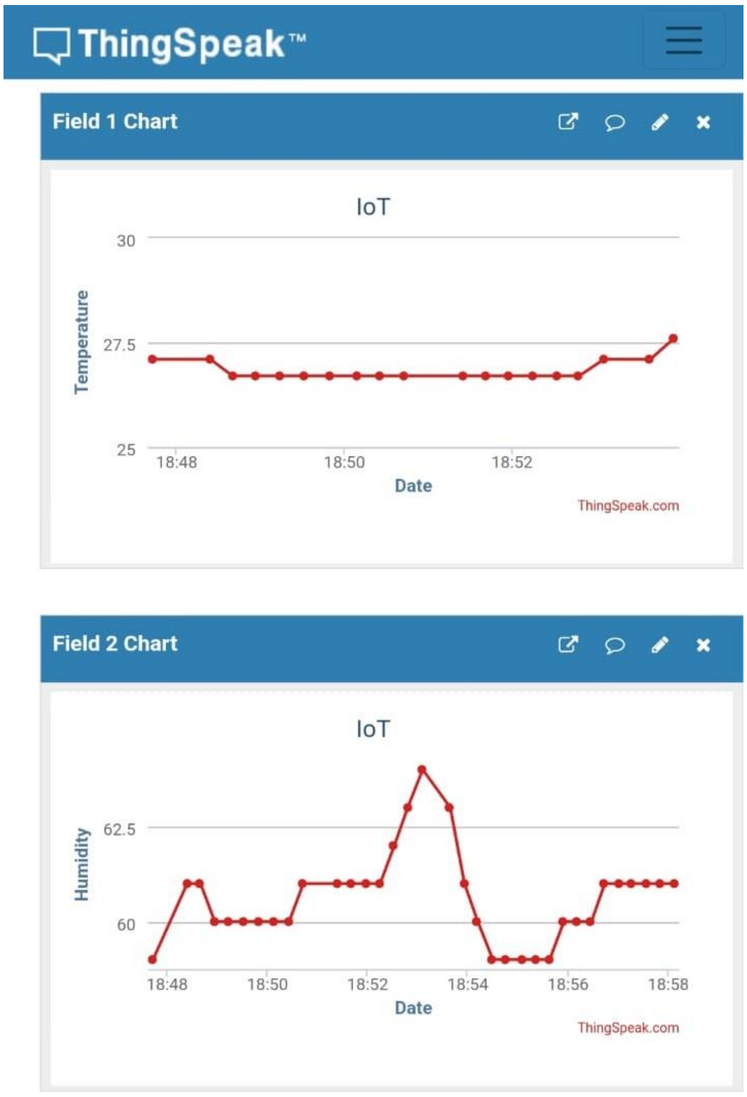
b. Click on "API Keys" button to get your unique API key for uploading your sensor data.

c. Now copy your "Write API Key". We will use this API key in our code.

Arduino Code to send temperature and humidity data to ThingSpeak Cloud

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


const char* ssid = "Vaibhav";
const char* password ="123456789";
String apiKey = "A6THKZ4JF1R5FSFG"; //Write API Key from website
const char* server = "api.thingspeak.com";


#define DHTPIN D5
#define DHTTYPE DHT11
DHT dht(DHTPIN,  DHTTYPE);
float humidity;
float temp;

WiFiClient client; //need to communication with the ThingSpeak server

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);
  Serial.print("Wifi connecting to ...SSID:");
  Serial.println(ssid);
  Serial.print("connecting");

  while(WiFi.status()!=WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }

 Serial.println("Wifi connected successfully");
 Serial.print("nodemcu Ip Adress");
 Serial.println(WiFi.localIP());

}

void loop() {
  temp=dht.readTemperature();
  humidity= dht.readHumidity();


 if(client.connect(server, 80)){

    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(temp);
    postStr += "&field2=";
    postStr += String(humidity);
    postStr += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
    delay(1000); //necessary for posting to ThingSpeak

    Serial.print("Temperature = ");
    Serial.println(temp,2);
    Serial.print("Humidity = ");
    Serial.print(humidity, 2);
    Serial.println (" %");
    Serial.println("Data sent to ThingSpeak");
  }
  client.stop();
  Serial.println("Waiting for 15sec...");
  delay(15000); //ThingSpeak needs min. 15sec delay between each data
post
}
```

Output:



Result : Temperature and humidity data uploaded to thingspeak cloud and readings observed.

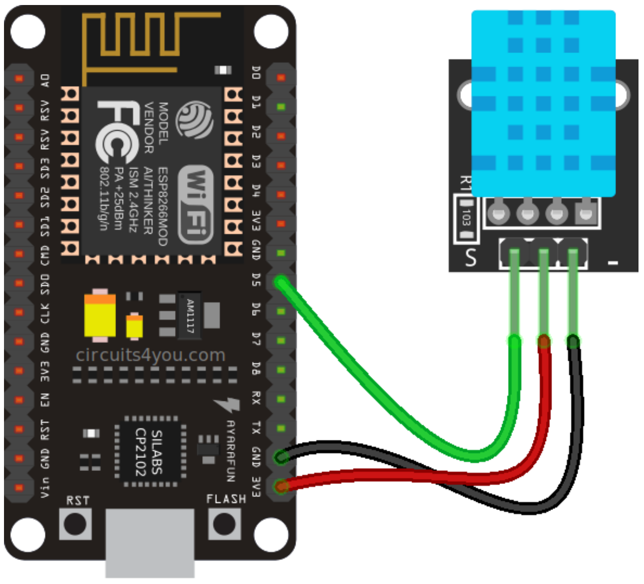
Experiment No 6

Aim: Write a program on Arduino to retrieve temperature and humidity data from Thingspeak cloud.

Component Required:

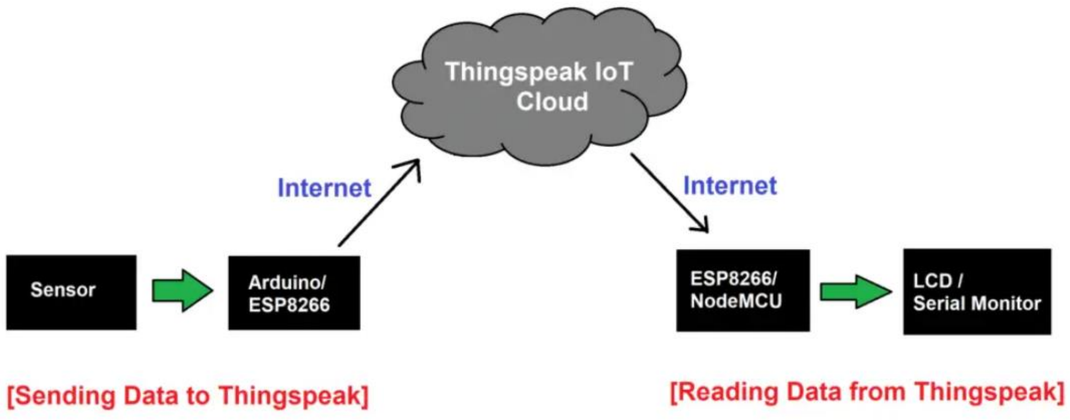
Sl No	Components	Quantity
1	Node MCU	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--

Circuit Connection:



NODE MCU	DHT11
GND	GND
3V3	VCC
D5	DATA

Theory :
To read values from Thingspeak we need to upload some data in real time, to do this, first upload temperature and humidity data to Thingspeak using previous experiment using NodeMCU 8266



Inst alling the ThingSpeak Library:

1. To send or receive sensor readings to ThingSpeak, we'll use the ThingSpeak Arduino library. Go to **Sketch** > **Include Library** > **Manage Libraries…** and search for "**ThingSpeak**" in the Library Manager. Install the ThingSpeak library by MathWorks.

OR

2. Upload the Library from drive Go to **Sketch > Include Library > Add ZIP Library.… > open t**o install Library

<u>Channel Settings that you need to do read data:</u>

Go to your Thingspeak account and do the following setting to receive temperature and humidity data.

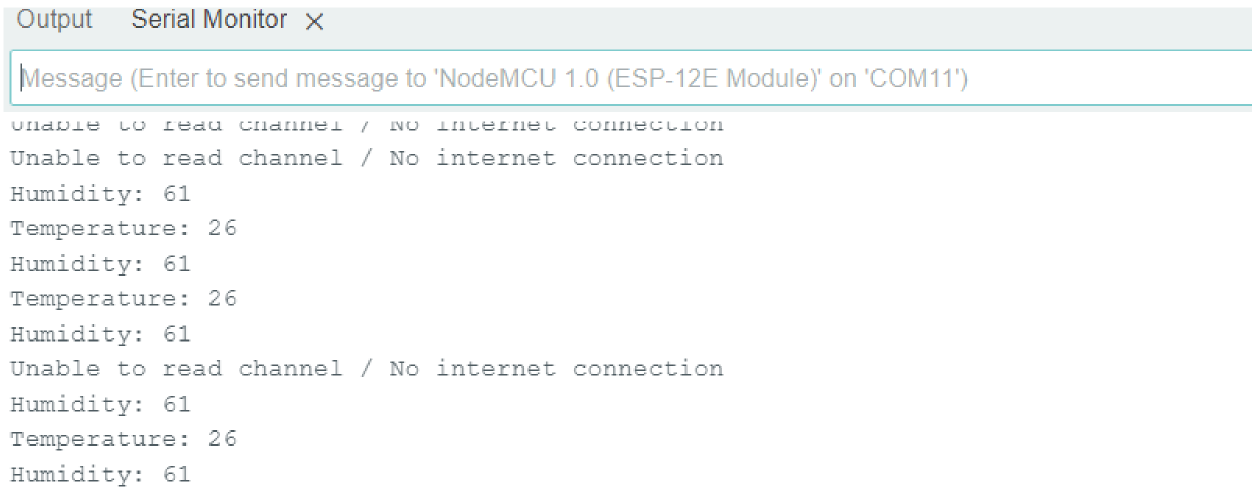
1. Go to channel setting put 'tick' mark for both filed 1 and filed 2 and scroll down to bottom and save it.
2. You need your channel ID to read the fields on your channel you wish to read so that copy your channel id and paste in the code
3. You need your **Read API key** from your channel and copy **Read API key**, use this **Read API key** in our code.
4. **Write following program and upload in the Node MCU82666**
5. After successful upload Open the serial monitor; you will be able to see the values read from your channel.

<u>Arduino Code to retrieve Temperature and Humidity values from ThingSpeak Cloud</u>

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

const char* ssid = "Vaibhav";
const char* password ="123456789";

//---------Channel Details---------//
// enter your Channel ID
unsigned long counterChannelNumber = 2404058;
// enter your Read API Key
const char * myCounterReadAPIKey = "AW591CS2C08HS7JM";
const int FieldNumber1 = 1;  // The field you wish to read
const int FieldNumber2 = 2;  // The field you wish to read

WiFiClient  client; //need to communication with the ThingSpeak
server

void setup()
{
  Serial.begin(9600);
   WiFi.begin(ssid, password);

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);


    while(WiFi.status()!=WL_CONNECTED){
    delay(500);
    Serial.println("Wifi connecting.....");
  }
  Serial.println("Wifi connected successfully ");
}

void loop(){
  //---------------- Channel 1 ----------------//
  long temp = ThingSpeak.readLongField(counterChannelNumber,
FieldNumber1, myCounterReadAPIKey);
  int statusCode = 0;
  statusCode = ThingSpeak.getLastReadStatus();
  if (statusCode == 200)
  {
    Serial.print("Temperature: ");
    Serial.println(temp);
  }
  else
  {
    Serial.println("Unable to read channel / No internet 
connection");
  }
  delay(100);

  //---------------- Channel 2 ----------------//
  long humidity = ThingSpeak.readLongField(counterChannelNumber, 
FieldNumber2, myCounterReadAPIKey);
  statusCode = ThingSpeak.getLastReadStatus();
  if (statusCode == 200)
  {
    Serial.print("Humidity: ");
    Serial.println(humidity);
  }
  else
  {
    Serial.println("Unable to read channel / No internet 
connection");
  }
  delay(100);
  //-------------- End of Channel 2 -------------//
}
```

Result: Temperature and Humidity present on ThingSpeak cloud was retrived and displayed on Serial Monitor.

Output:


Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM11')

```
Unable to read channel / No internet connection
Unable to read channel / No internet connection
Humidity: 61
Temperature: 26
Humidity: 61
Temperature: 26
Humidity: 61
Unable to read channel / No internet connection
Humidity: 61
Temperature: 26
Humidity: 61
```

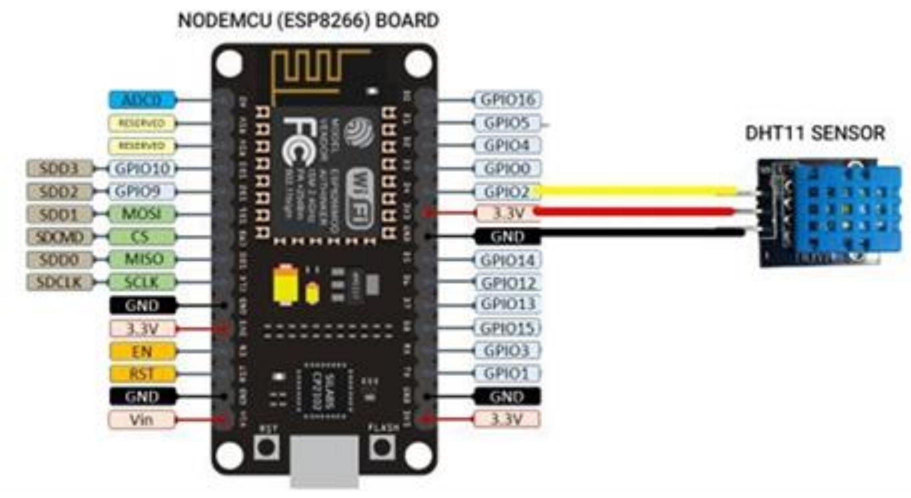
Experiment No: 7

Aim: Write a program to create UDP server on Arduino and respond with humidity data to UDP client when requested.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--

Circuit Connection:



NODE MCU	DHT11
GND	GND
3V3	VCC
D4	DATA

Theory :

User Datagram Protocol (UDP) is a network communication protocol that operates at the transport layer of the Internet Protocol (IP) suite. It is a connectionless and lightweight protocol designed for fast and efficient data transmission, but it does not provide the same level of reliability and error-checking as Transmission Control Protocol (TCP).

UDP is a lightweight, connectionless, and fast protocol that prioritizes low-latency data transmission over reliability. It is suitable for applications where occasional packet loss or out-of-order delivery can be tolerated, and real-time communication is essential. However, for applications that require guaranteed delivery and error recovery, TCP is a better choice.

Procedure

To run Python code for UDP Server

1) Install python software
2) Open python IDLE
3) In python IDLE go to **File →   New File** (it opens new script windows) **→ Type the code**
4) Click on **Run** button and **Save** the program from script window
5) See the output from IDLE shell (command prompt)

Arduino Code to create UDP server on Arduino and respond with humidity

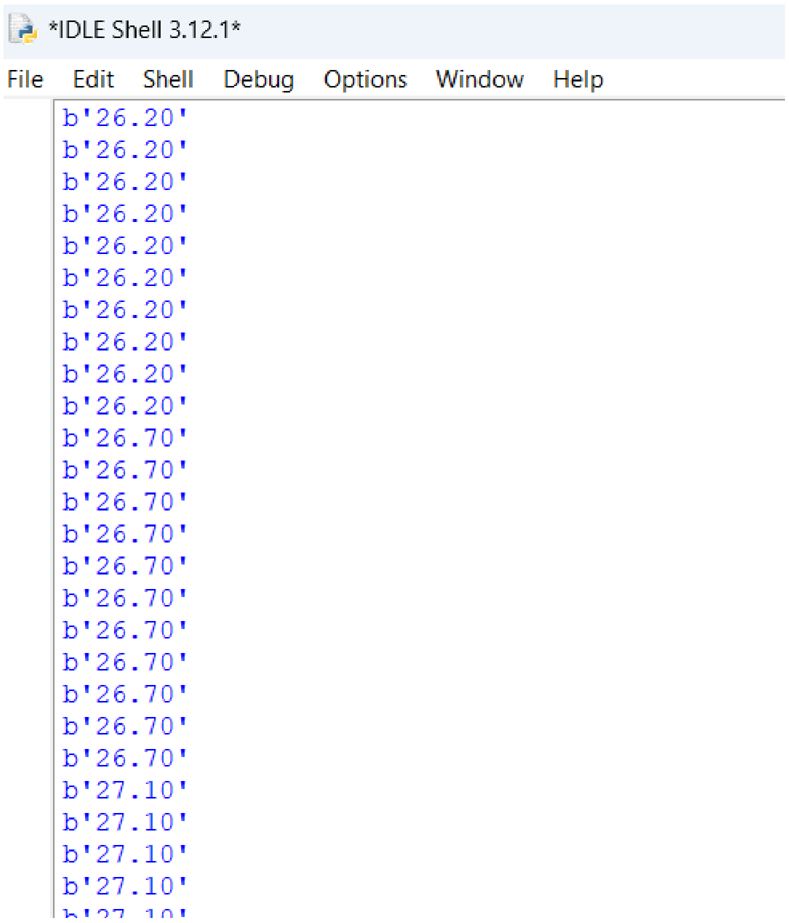
data to UDP client when requested.

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

#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temp;
char packet[1024];
const char* ssid = "Vaibhav";
const char* password ="123456789";
const char* udpServerIP = "192.168.199.118";  // Replace <IP>
with the actual IP address of your computer
unsigned int port = 9000;

WiFiUDP Udp; // Create a UDP object

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
 Serial.println("Wifi connected successfully");
}

void loop() {
  Udp.beginPacket(udpServerIP, port);

  // Read temperature from DHT sensor and convert it to a string
  temp = dht.readTemperature();
  //confirm value in serila monitor
  Serial.print("Temperature in degree Cel. :");
  Serial.println(temp);

  String tempStr = String(temp);

  // Copy the temperature string into the packet buffer
  tempStr.toCharArray(packet, 1024);

  // Send the packet over UDP
  Udp.write(packet);

  // End the UDP packet and send it
  Udp.endPacket();
  delay(500);
}
```

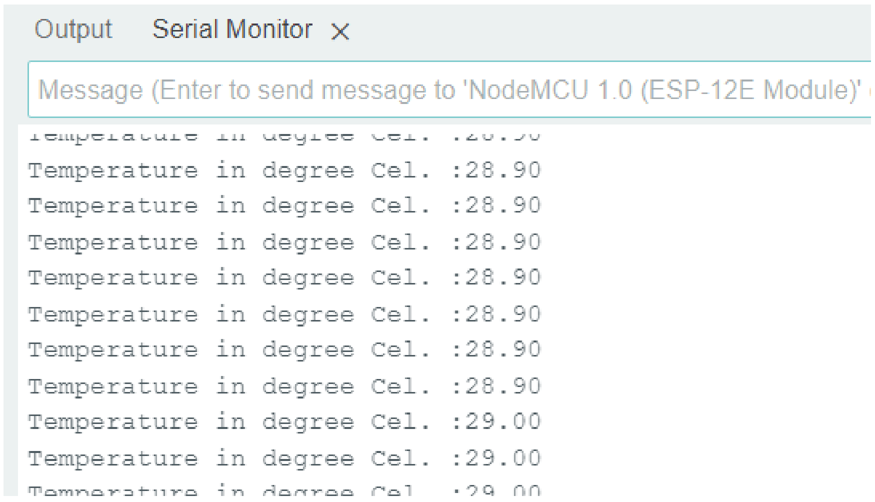
Python Code:

```
import socket

UDP_IP = "192.168.199.118"
UDP_PORT = 9000

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

while True:
    received_data, addr = sock.recvfrom(1024)
    print(received_data)
```

Output:



```
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.20'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'26.70'
b'27.10'
b'27.10'
b'27.10'
b'27.10'
```


Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)'

```
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :28.90
Temperature in degree Cel. :29.00
Temperature in degree Cel. :29.00
```

Result: Created UDP server on Arduino and respond with humidity data to UDP client when requested.

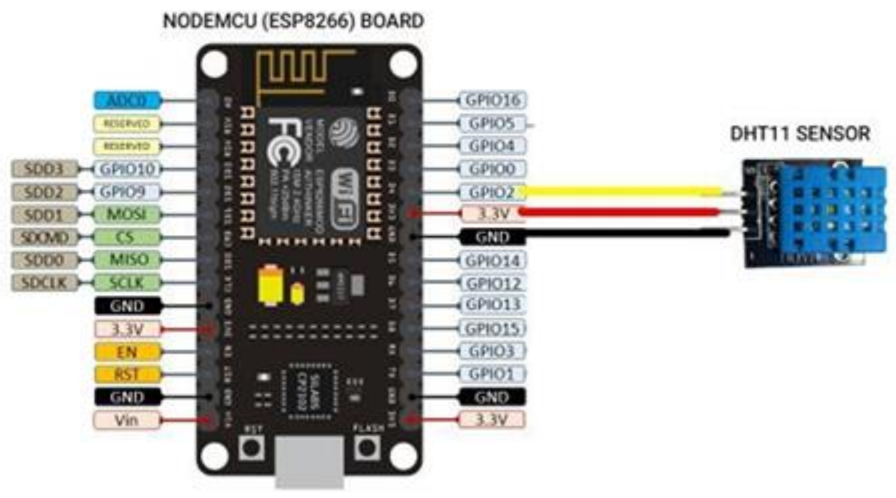
Experiment No: 8

Aim: Write a program to create TCP server on Arduino and respond with Temperature data to TCP client when requested.

Component Required:

Sl No	Components	Quantity
1	Node MCU ESP 8266	1
2	Temperature sensor DHT11	1
3	USB cable	1
4	Connecting wires	--

Circuit Connection:



NODE MCU	DHT11
GND	GND
3V3	VCC
D4	DATA

Theory :

The Transmission Control Protocol (TCP) is a widely used protocol in the Internet Protocol (IP) suite. It is a connection-oriented protocol that provides reliable, ordered, and error-checked delivery of data between applications running on hosts1. In IoT, TCP is used to establish connections between clients and servers, allowing devices to interact with each other and resolve common problems.

A TCP server is a program that listens for incoming connections from clients and responds to their requests. When a client connects to the server, it sends a request for data. The server then sends back the requested data to the client. In IoT, TCP servers are used to provide access to data from sensors and other devices
In the context of the Internet of Things (IoT), a TCP server refers to a network service running on a device or gateway that listens for incoming TCP connections from other IoT devices or clients

In the context of IoT, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two protocols of the Transport Layer that are used to transmit data between devices over a network.

TCP is a connection-oriented protocol that provides reliable delivery services by keeping track of the segments being transmitted or received by assigning numbers to every single one of them. It also implements an error control mechanism for reliable data transfer and takes into account the level of congestion in the network .

On the other hand, UDP is a connectionless protocol that is used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control. It is a suitable protocol for multicasting as UDP supports packet switching. Normally used for real-time applications which cannot tolerate uneven delays between sections of a received message.

In summary, TCP is more reliable but slower than UDP, while UDP is faster but less reliable than TCP

Procedure

To run Python code for UDP Server

1) Install python software (Ignore if already installed)
2) Open python IDLE
3) In python IDLE go to **File** → **New File** (it opens new script windows) → **Type the code**
4) Click on **Run** button and **Save** the program from script window
5) See the output from IDLE shell (command prompt)

Arduino Code to create TCP server on Arduino and respond with Temperature data to TCP client when requested.

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

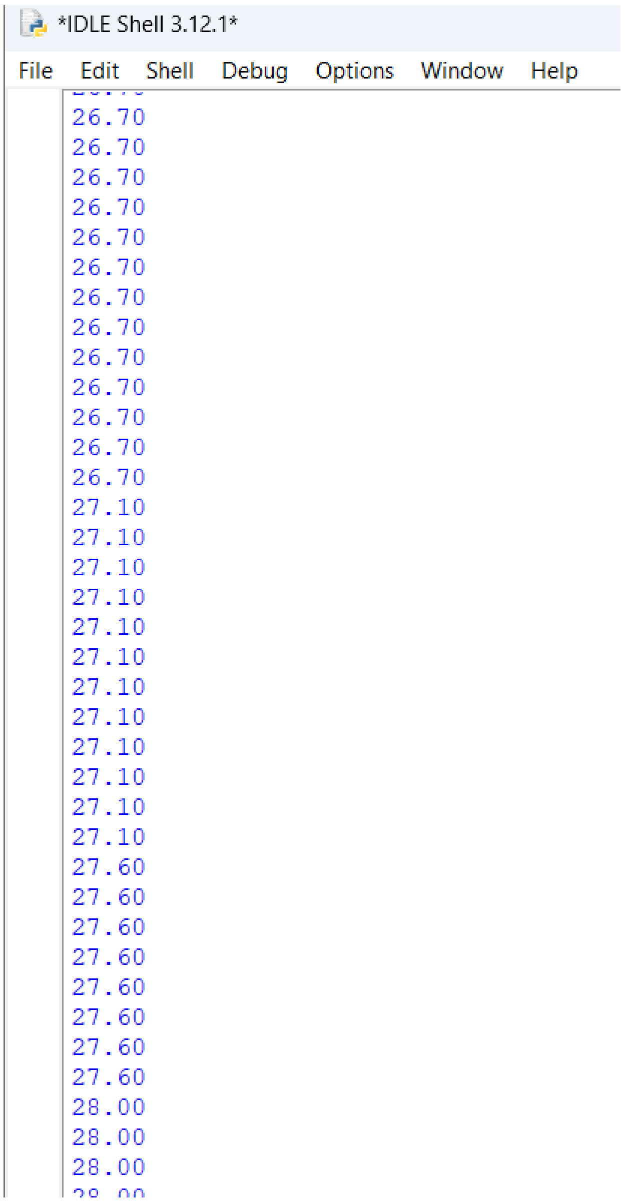
```
#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

float temp;
const char* ssid = "Vaibhav";
const char* password = "123456789";
const char* tcpServerIP = "192.168.199.118";  // Replace <IP>
with the actual IP address of your computer
unsigned int port = 80; //most commonly used protocol port in
TCP
WiFiClient client;  // Create a TCP client object

void setup() {
  Serial.begin(9600);
  dht.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected successfully");
}
void loop() {
  client.connect(tcpServerIP, port);
    // Read temperature from DHT sensor and convert it to a
string
  temp = dht.readTemperature();
    // Confirm value in serial monitor
  Serial.print("Temperature in degree Celsius: ");
  Serial.println(temp);
    // Send the temperature over TCP as a string
  client.print(temp);
  delay(500);

}
```

Python code for UDP Server
`import socket`

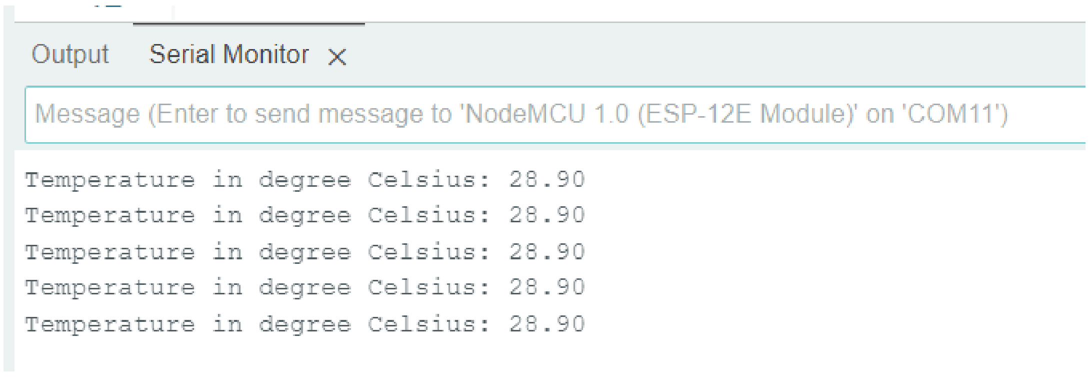
```
mysock =socket.socket(socket.AF_INET,  socket.SOCK_STREAM)
TCP_IP = '192.168.1.6' # Get local machine IP from LAN or WIFI setting of PC
TCP_PORT=80; #most commonly used protocol in TCP

mysock.bind((TCP_IP, TCP_PORT))

mysock.listen(5)  # Now wait for client connection.
while True:
    received_data, addr=mysock.accept() # Establish connection with client.

    data = received_data.recv(1024).decode()
    #we are extracting 1024 bytes of data at a time.
    #(One can use one’s convenient number here).
    print(data)
```

Result: Created TCP server on Arduino and respond with Temperature data to TCP client when requested.

Output:



```
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
26.70
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.10
27.60
27.60
27.60
27.60
27.60
27.60
27.60
27.60
28.00
28.00
28.00
```


Output
Serial Monitor ×
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM11')

```
Temperature in degree Celsius: 28.90
Temperature in degree Celsius: 28.90
Temperature in degree Celsius: 28.90
Temperature in degree Celsius: 28.90
Temperature in degree Celsius: 28.90
```

Result: Python Program used to create TCP server and Arduino program used to respond with Temperature data to TCP client when requested is written and verified.

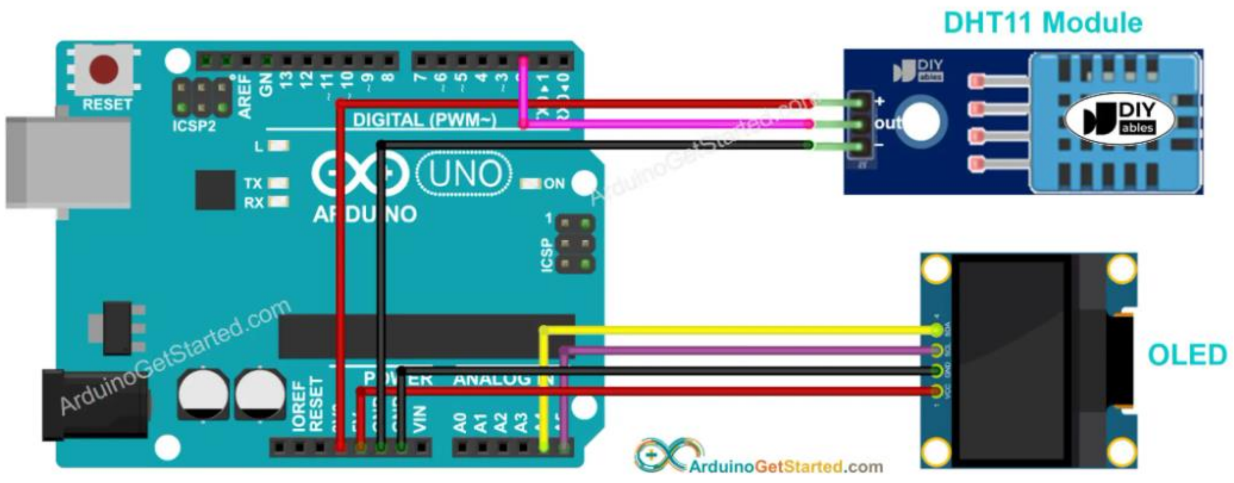
Experiment No.: 9

Aim: To interface OLED with Arduino and write a program to print temperature and humidity readings on it.

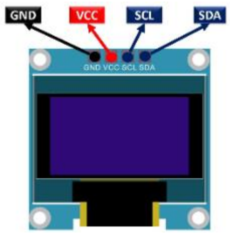
Component Required:

Sl No	Components	Quantity
1	Arduino UNO	1
2	DHT11 Temperature sensor	1
3	USB cable for Arduino UNO	1
4	Connecting wires	--
5	OLED Display 128×64, SSD1306 I2C	1

Circuit Connection:



Theory :



The SSD1306 OLED I2C 128X64 OLED Display module is a small monochrome organic light-emitting diode (OLED) display that is controlled through an I2C interface. It has a display resolution of 128×64 pixels, and the SSD1306 is the controller chip that manages the display. It's commonly used for display purposes in various electronics projects and is compact low power, and easily readable in low light conditions.

Specification of OLED	
Size	0.96 inch
Terminals	4
Pixels	128×64
Communication	I2C only
VCC	3.3V-5V
Operating Temperature	-40°C to +80°C

To control the OLED display you need the adafruit_SSD1306.h and the adafruit_GFX.h libraries. Follow the next instructions to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.
2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.
3. Click install button to install library
4. If ask click on install all button to install library dependencies
5. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.
6. If ask click on install all button to install library dependencies
7. After installing the libraries, restart your Arduino IDE
8. Find the **Adafruit_SSD1306.h** file in the Arduino Library folder. Generally, it is located at **Documents\Arduino\libraries** on windows systems. There you will find the Adafruit_SSD1306.h file inside the Adafruit_SSD1306 folder.

Arduino Program to display temperature and humidity on OLED

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

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

#define DHTPIN 2 // pin connected to DHT11 sensor
#define DHTTYPE DHT11

Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
// create SSD1306 display object connected to I2C
DHT dht(DHTPIN, DHTTYPE);

String displayString;

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

  // initialize OLED display with address 0x3C for 128x64
  if (!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    while (true);
  }

  delay(2000);         // wait for initializing
  oled.clearDisplay(); // clear display

  oled.setTextSize(2);      // text size
  oled.setTextColor(WHITE); // text color
  oled.setCursor(0, 10);    // position to display

  dht.begin();              // initialize DHT11 the temperature
and humidity sensor
  displayString.reserve(10); // to avoid fragmenting memory when
using String
}

void loop() {
  float humi  = dht.readHumidity();    // read humidity
  float tempC = dht.readTemperature(); // read temperature

  // check if any reads failed
  if (isnan(humi) || isnan(tempC)) {
    displayString = "Failed";
  } else {
    displayString  = String(tempC, 1); // one decimal places
    displayString += "°C";
    displayString += String(humi, 1); // one decimal places
    displayString += "%";
  }

  Serial.println(displayString);    // print the temperature in
Celsius to Serial Monitor
  oledDisplayCenter(displayString); // display temperature and
humidity on OLED
}

void oledDisplayCenter(String text) {
  int16_t x1;
  int16_t y1;
  uint16_t width;
  uint16_t height;

  oled.getTextBounds(text, 0, 0, &x1, &y1, &width, &height);

  // display on horizontal and vertical center
  oled.clearDisplay(); // clear display
  oled.setCursor((SCREEN_WIDTH - width) / 2, (SCREEN_HEIGHT -
height) / 2);
  oled.println(text); // text to display
  oled.display();
}
```

Result: Temperature and humidity values are seen on OLED.