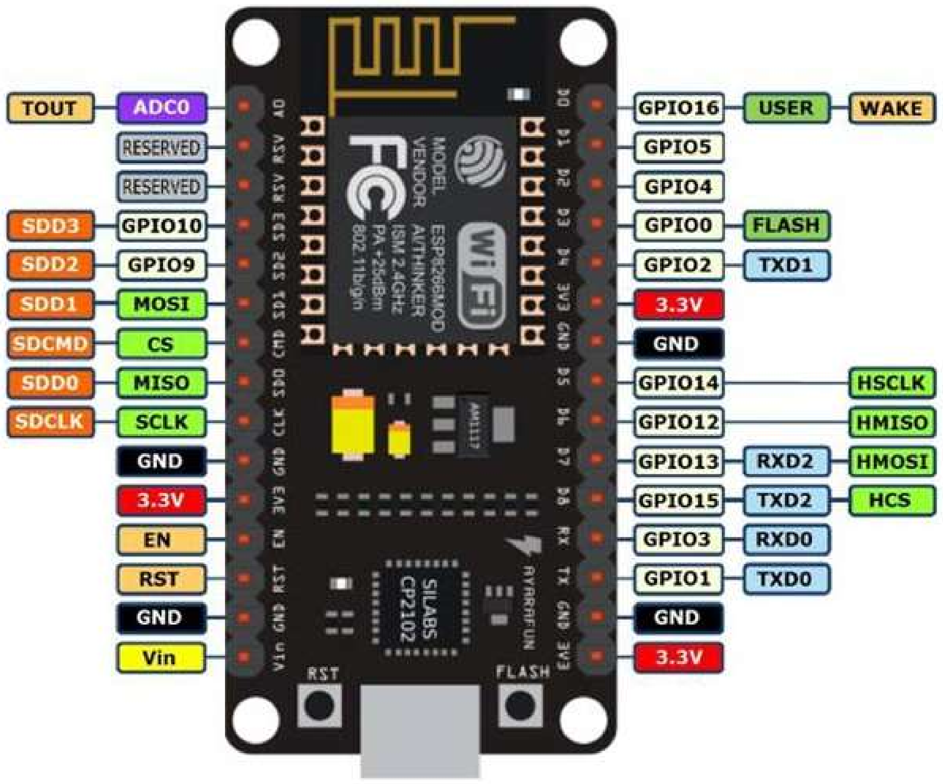
1. Introduction to IoT equipment's and perform necessary software installation

Aim: Introduction to IoT equipment's and perform necessary software installation and make the led blink.

Description:
NodeMCU (Node MicroController Unit) 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.

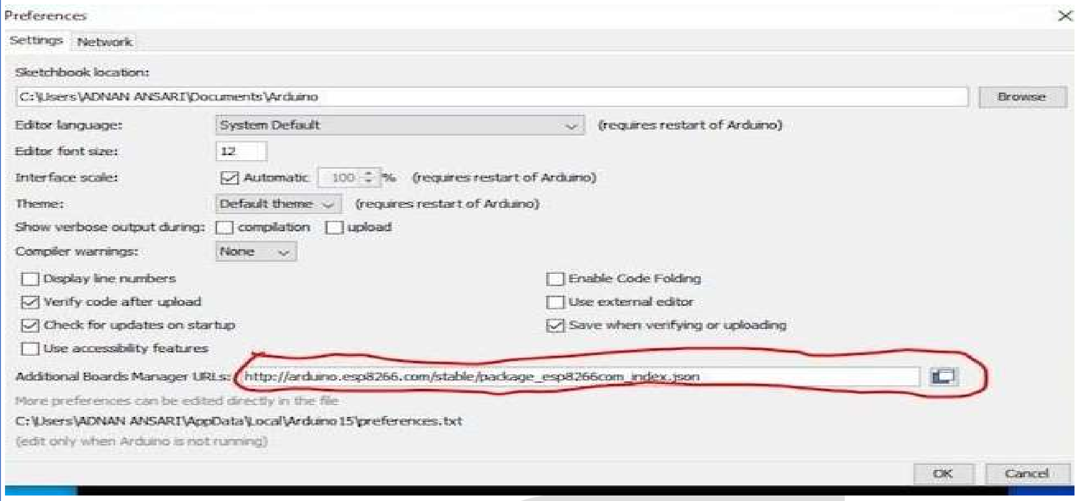
But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x 86 chips) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors.   The term NodeMCU usually refers to the firmware, while the board is called Devkit. NodeMCU Devkit 1.0 consists of an ESP-12E on a board, which facilitates its use. It also has a voltage regulator, a USB interface.



Step 1: Installing Arduino IDE Software
Install Arduino IDE software from the link http://www.arduino.cc/en/main/software After installing Arduino IDE icon is created on the Desktop as show in the figure.

Step 2: Open the software and click on the left upper corner "file" option then click on the preference. Click on the preference and paste the link that we are sharing in the next step.

Step 3: Paste the given link into the additional board manager URL.
http://arduino.esp8266.com/stable/package_esp8266com_index.json



Preferences

Settings Network

Sketchbook location:
C:\Users\ADNAN ANSARI\Documents\Arduino

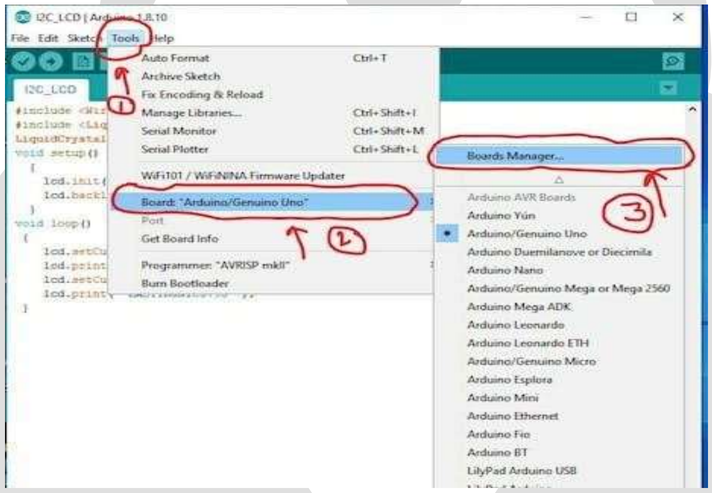
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None

- [ ] Display line numbers
- [✓] Verify code after upload
- [✓] Check for updates on startup
- [ ] Use accessibility features
- [ ] Enable Code Folding
- [ ] Use external editor
- [✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Users\ADNAN ANSARI\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

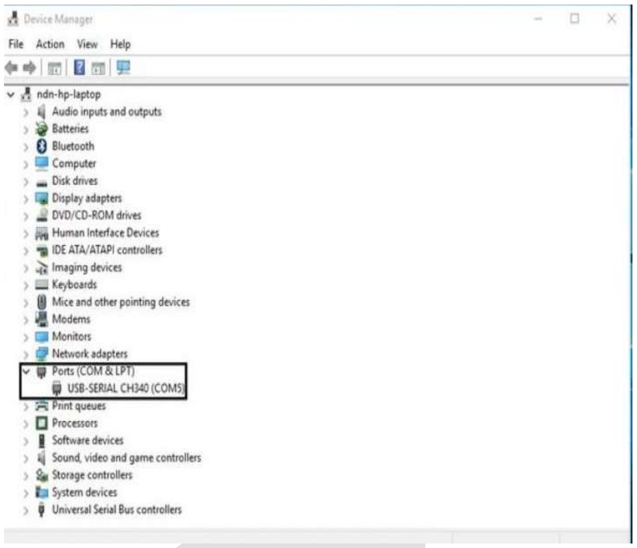
Step 4: This is the last step in which you have to go to the tools from the given option. Click on the board where arduino/genuino uno and select the board manager to add the board in App.

Step 5: Write esp8266 in the search and install the board to the software



Step 6: Selecting ESP8266 Arduino Board
To run the esp8266 with Arduino we have to select the Board: “Arduino/Genuino Uno” and then change it to NodeMCU 1.0 (ESP-12E Module) or other esp8266 modules depending on what you have.

Step 7: Connecting ESP8266 to the PC
Now let’s connect the ESP8266 module to your computer through USB cable as shown in the figure. When module is connected to the USB, COM port is detected eg: here COM5 is shown in the figure.



Device Manager

File Action View Help

ndn-hp-laptop
- Audio inputs and outputs
- Batteries
- Bluetooth
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Keyboards
- Mice and other pointing devices
- Modems
- Monitors
- Network adapters
- Ports (COM & LPT)
  - USB-SERIAL CH340 (COM5)
- Print queues
- Processors
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

Step 8:- Selecting COM Port
Click on tools to select the port: “COM” based on which esp8266 module is connected to your respected COM port of the computer. To select COM ports refer previous steps.

Step 9: Upload the program to your board.

Sketch − the first new terminology is the Arduino program called “sketch”.

Structure:
Arduino programs can be divided in three main parts: Structure, Values (variables and constants), and Functions.

Void setup ( ) {
}
PURPOSE − the setup () function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board

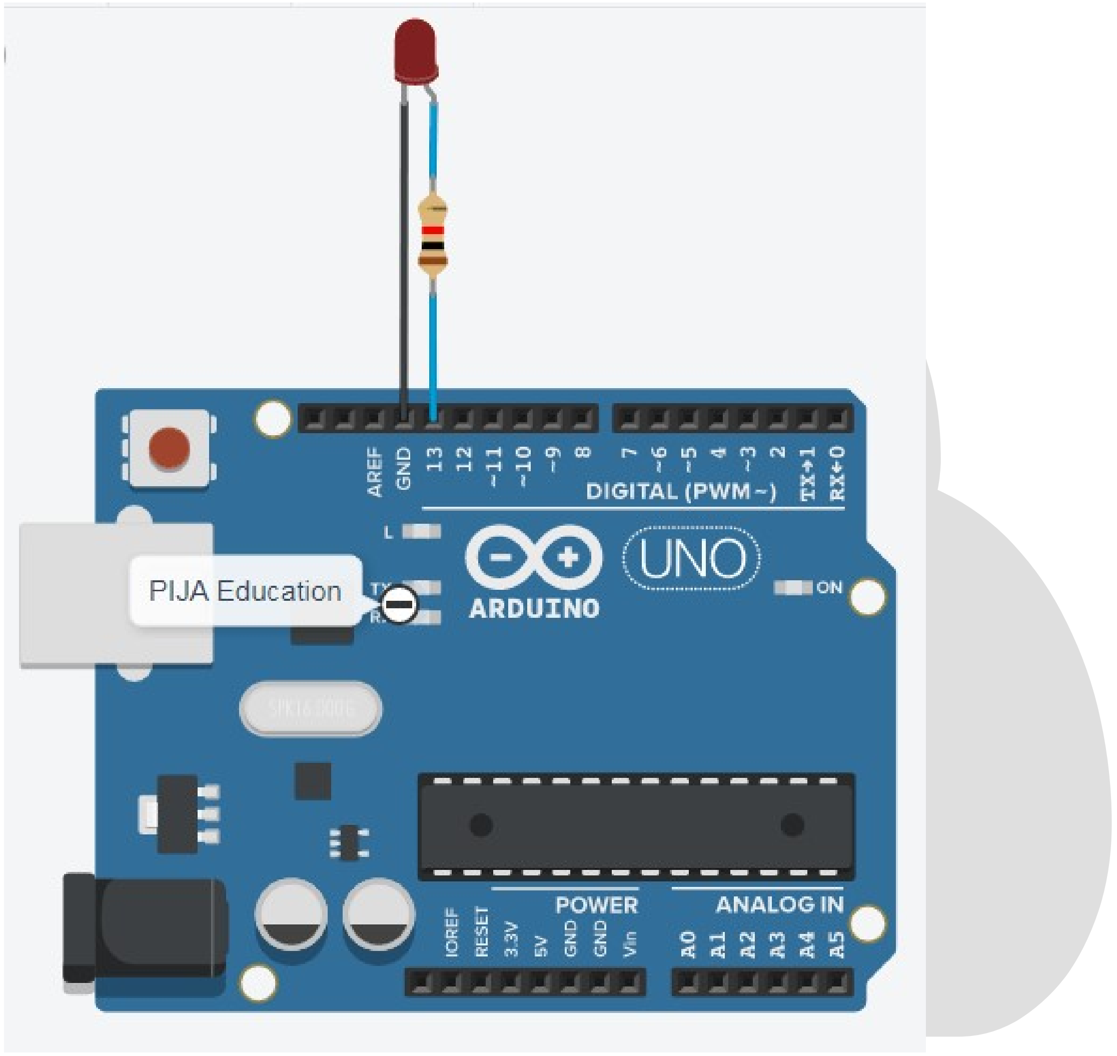
Void Loop ( ) {
}

PURPOSE − after creating a setup ( ) function, which initializes and sets the initial values, the loop () function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino devices.

Devices Used:

NodeMCU, Breadboard, jumper wires and led

Circuit Diagram:



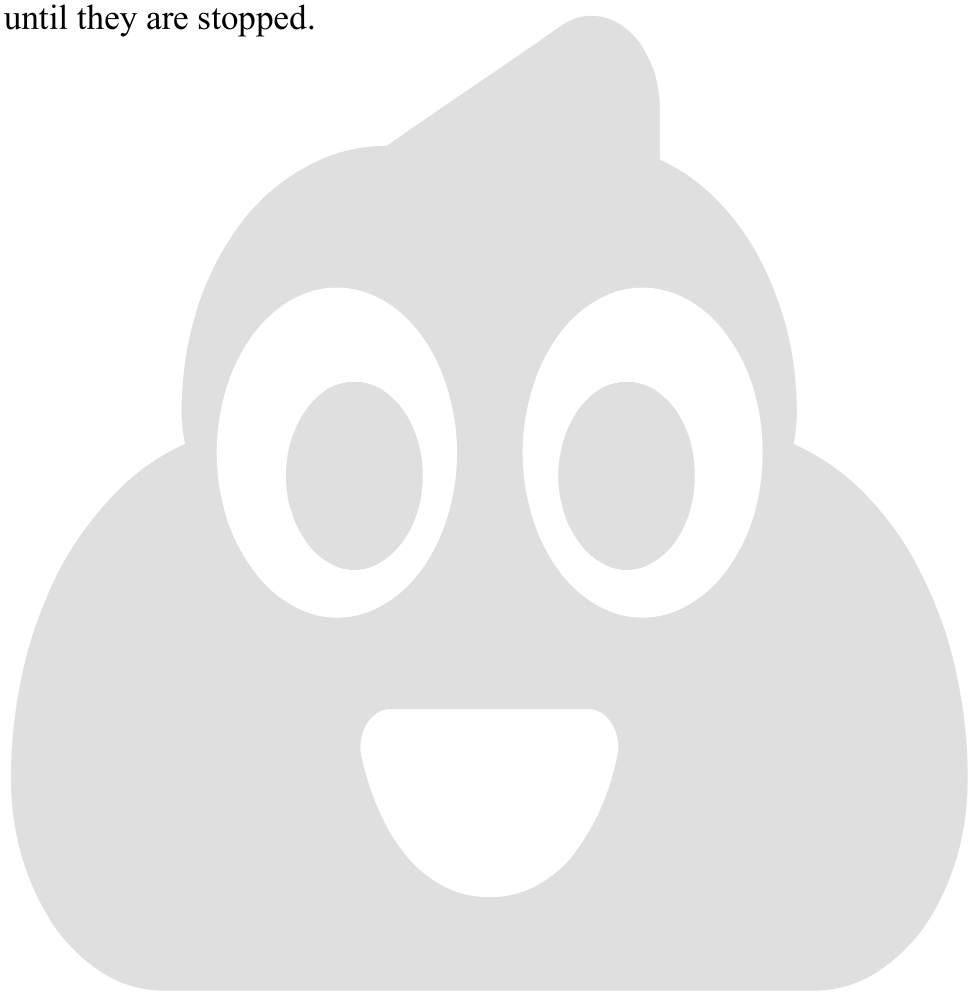
Code:

```
void setup( )
{
   // initialize digital pin LED_BUILTIN as an output.
   pinMode(D0, OUTPUT);
}
void loop( )
```

```
{
   digitalWrite(D0, HIGH);    // turn the LED on (HIGH is the voltage level)
   delay(random(1000));                              // wait for a second
   digitalWrite(D0, LOW);
  delay(random(1000));   // turn the LED off by making the voltage LOW
}
```

Output and result analysis:

The led is turned ON for a second and then it gets turned OFF for the next second. This continues until they are stopped.

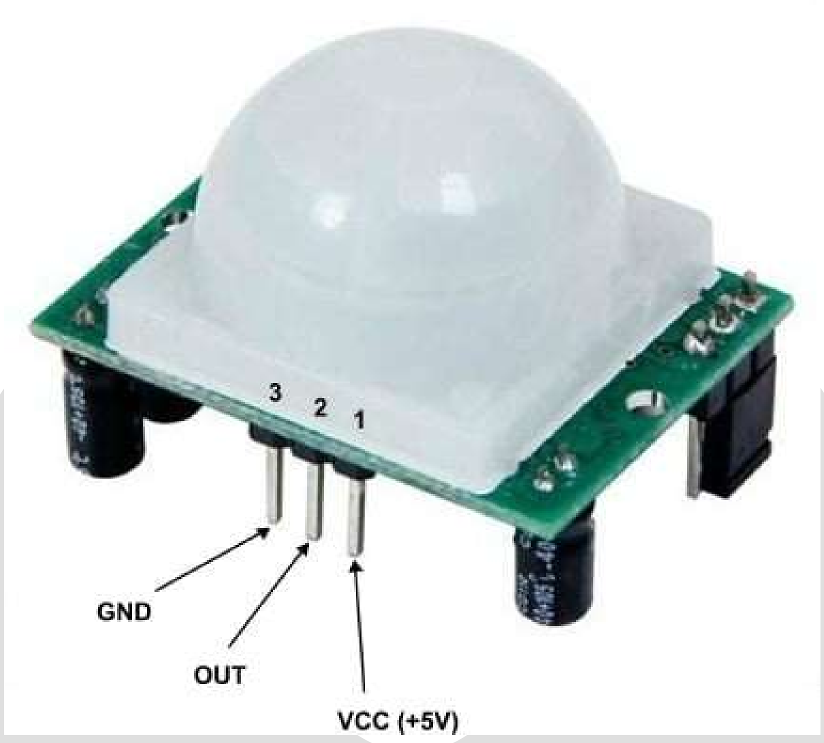


2. Write a program to interface PIR sensor with Arduino and turn ON LED when motion is detected.

Aim: To interface PIR sensor with Arduino and turn ON LED when motion is detected.

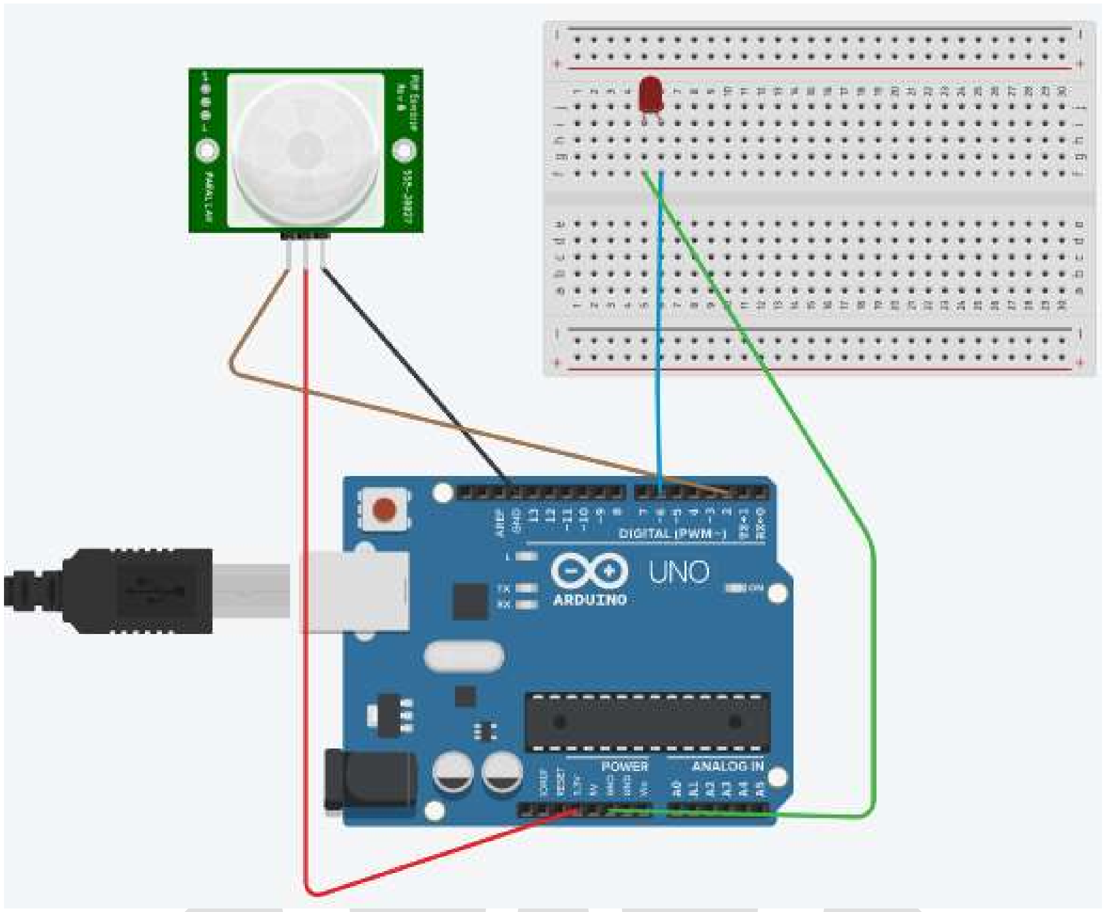
Devices Used:
PIR sensor, NodeMCU, Breadboard, jumper wires and led

Description:
PIR sensors allow you to sense motion. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses.



What is a PIR Sensor?
A passive infrared sensor is an electronic sensor that measures infrared light radiating from objects. PIR sensors mostly used in PIR-based motion detectors. Also, it used in security alarms and automatic lighting applications. The below image shows a typical pin configuration of the PIR sensor, which is quite simple to understand the pinouts. The PIR sensors consist of 3 pins. Pin1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC. Pin2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor. The pin 2 of the sensor carries the detected IR signal to an amplifier from the Pin3 of the sensor connected to the ground.

Circuit Diagram:

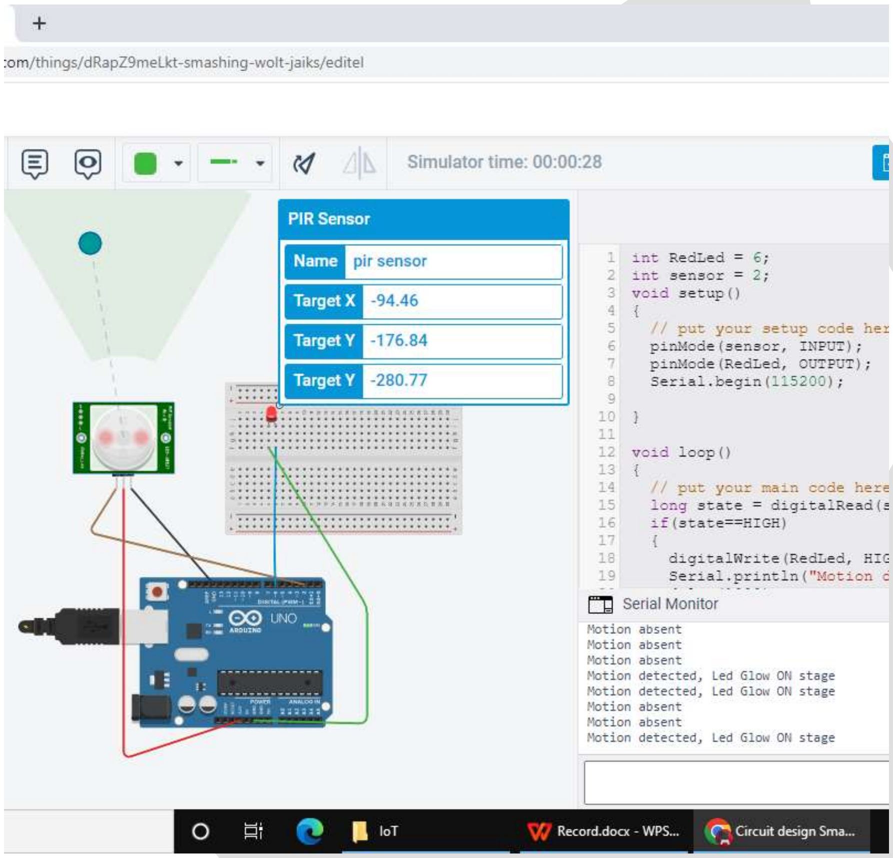


Code:

```
int RedLed = 12; //pin D6
int sensor = 5; //pin D1
void setup( )
{
   pinMode(sensor, INPUT);
   pinMode(RedLed, OUTPUT);
   Serial.begin(115200);
}

void loop( )
{
   // put your main code here, to run repeatedly:
   long state = digitalRead(sensor);
   if(state==HIGH)
   {
     digitalWrite(RedLed, HIGH);
     Serial.println("Motion detected, Led Glow ON stage");
```

```
      delay(1000);
    }
    else
    {
      digitalWrite(RedLed, LOW);
      Serial.println("Motion absent");
      delay(1000);
    }
}
```

Output and result analysis:



:om/things/dRapZ9meLkt-smashing-wolt-jaiks/editel

Simulator time: 00:00:28

PIR Sensor

Name	pir sensor
Target X	-94.46
Target Y	-176.84
Target Y	-280.77

```
int RedLed = 6;
int sensor = 2;
void setup()
{
  // put your setup code her
  pinMode(sensor, INPUT);
  pinMode(RedLed, OUTPUT);
  Serial.begin(115200);

}

void loop()
{
  // put your main code here
  long state = digitalRead(s
  if(state==HIGH)
  {
    digitalWrite(RedLed, HIG
    Serial.println("Motion c
```

Serial Monitor

```
Motion absent
Motion absent
Motion absent
Motion detected, Led Glow ON stage
Motion detected, Led Glow ON stage
Motion absent
Motion absent
Motion detected, Led Glow ON stage
```

When motion is detected the LED is turned ON and the same is printed in the serial monitor. The led is off when there is no motion detected.

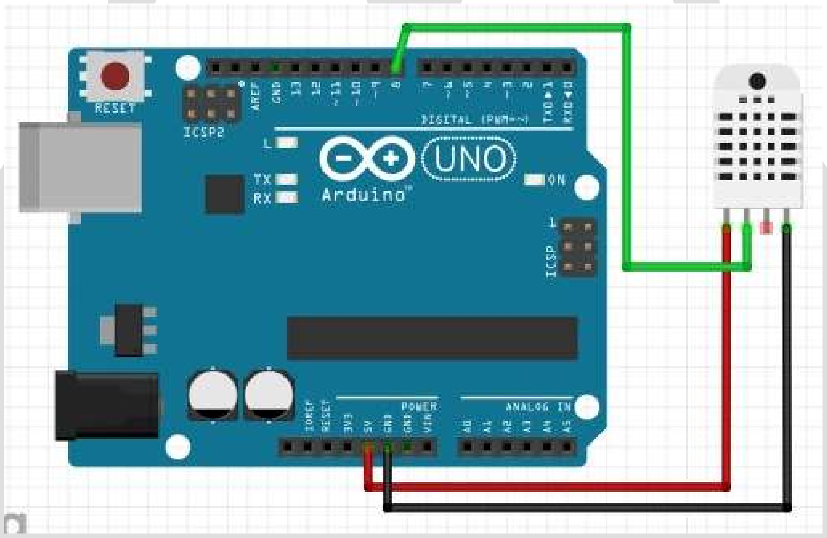
3. Write a program to interface DHT22 sensor with Arduino and display temperature and humidity readings.

Aim: Write a program to interface DHT22 sensor with Arduino and display temperature and humidity readings.

Devices Used:
DHT22 sensor, NodeMCU, Breadboard and jumper wires

Description:
The DHT22 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an internal circuit to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers. The sensor can measure temperature from -40°C to 80°C and humidity from 0% to 100% with an accuracy of ±1°C and ±1%.

Circuit Diagram:



Connect your DHT22 to your Node MCU 1.0 as follows:
1. Connect 3V3 (3.3V pin) or Vin (5V pin) of the Node MCU to the Vcc pin of the DHT22.
2. Digital pin D1 (GPIO 5) to the second pin of the DHT22.
3. Connect a 5K resistor data pin and VCC pin of the DHT22.
4. Leave the third DHT22 pin unconnected.
5. Connect the fourth pin of the DHT22 to ground of ESP8266.

Code:

```
#include <DHT.h>
int DHTPIN = 8;
#define DHTTYPE DHT22
DHT dht(DHTPIN,DHTTYPE);
float temp, hum;
void setup()
{
  Serial.begin(9600);
  dht.begin();
}
void loop( )
{
  hum = dht.readHumidity();
  temp = dht.readTemperature();
  Serial.print("Humidity : ");
  Serial.print(hum);
  Serial.println("%");
  Serial.print("Temperature : ");
  Serial.print(temp);
  Serial.println("C");
  Serial.println("-----------------");
  delay(4000);
}
```

Output and result analysis:

The DHT22 sensor captures both humidity and teemperature whose values as shown below:

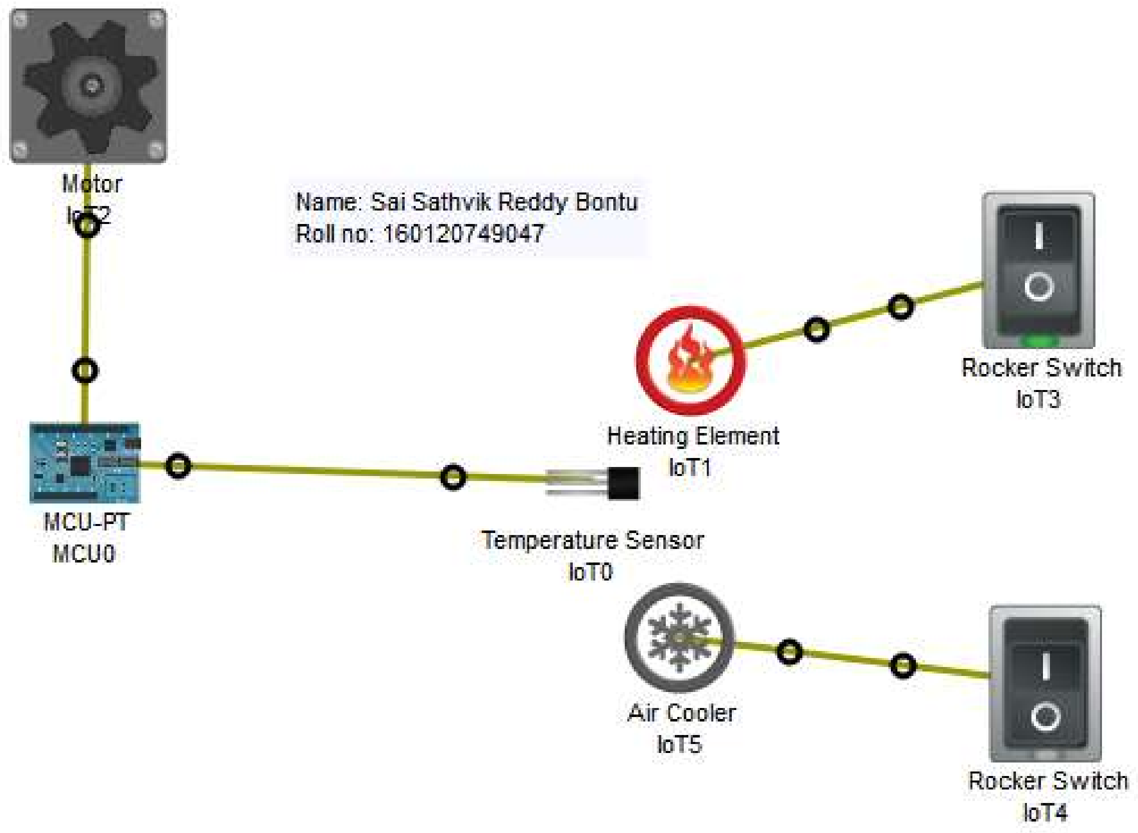
```
Humidity : 70.50%
Temperature : 22.60C
---------------------------
Humidity : 58.00%
Temperature : 25.60C
---------------------------
Humidity : 73.00%
Temperature : 25.60C
---------------------------
```

4. Write a program to interface motor with Raspberry Pi. Turn ON motor when the temperature is high.

Aim: To interface motor with Raspberry Pi. Turn ON motor when the temperature is high.

Devices Used:
Temperature sensor, NodeMCU, Breadboard, motor and jumper wires

Circuit Diagram:



Code:

```
function setup( )
{
        pinMode(A0, INPUT);
        pinMode(0, OUTPUT);
}

function loop( )
{
        var temp = analogRead(A0);
        temp = Math.floor(map(temp,0,1023,-100,100));
```

```
	Serial.println(temp);
	if(temp > 10)
	{
		digitalWrite(0, HIGH);
	}
	if(temp < 10)
	{
		digitalWrite(0, LOW);
	}
}
```

Output and result analysis:

When the temperature crosses 10C or whatever value is set in the program, the motor starts rotating.

Once the temperature falls below 10C the motor is turned off.

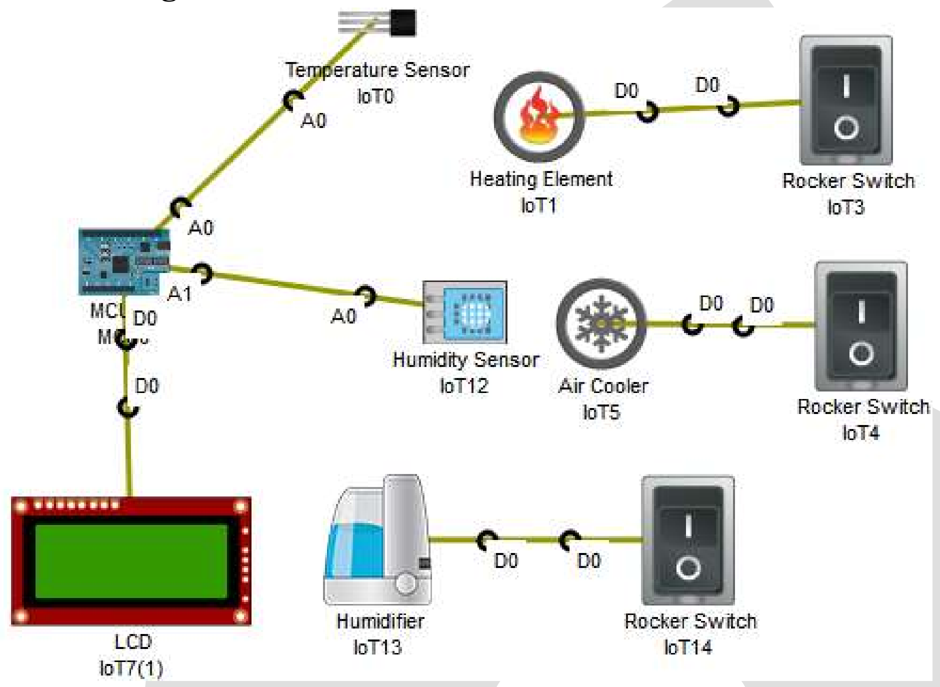
```
Temperatue = 18
Motor is running
Temperatue = 18
Motor is running
Temperatue = 18
Motor is running
Temperatue = 19
Motor is running
Temperatue = 19
Motor is running
```

5. Write a program to interface LCD with Raspberry Pi and print temperature and humidity readings on it.

Aim: To interface LCD with Raspberry Pi and print temperature and humidity readings on it.

Devices Used:
Temperature sensor, Humidity sensor, LCD screen, RaspberryPi, Heating element, Humidifier, Breadboard and jumper wires

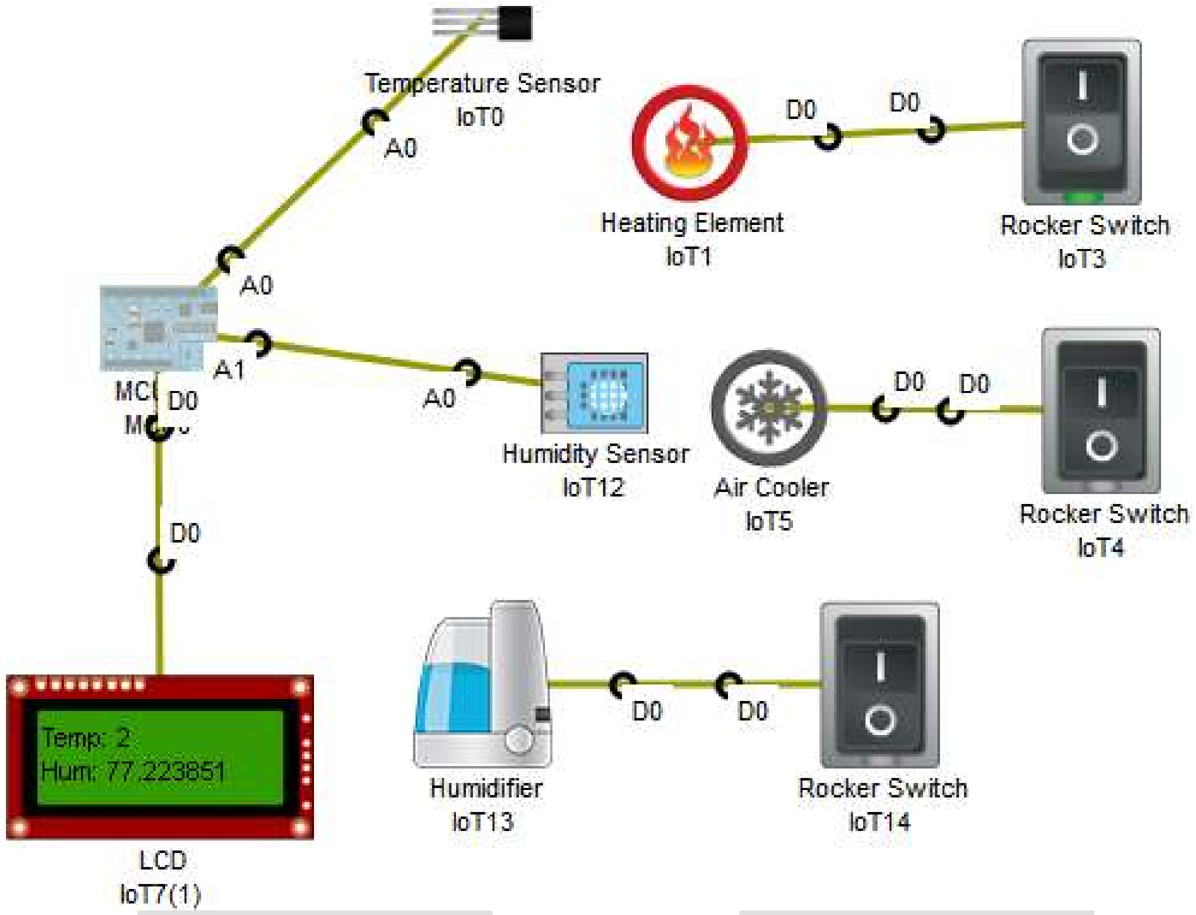
Circuit Diagram:



Code:

```
function setup( )
{
        pinMode(A0, INPUT);
        pinMode(A1, INPUT);
        pinMode(0, OUTPUT);
}
function loop( )
{
        var temp=analogRead(A0);
        temp=Math.floor(map(temp,0,1023,-100,100));
        Serial.println(temp);
        var humidity=analogRead(A1);
        humidity=map(humidity,0,1023,0,100);
        Serial.println("Temperature: "+temp+ "\n"+ "Humidity: " +humidity);
        var ans =    "Temp: "+temp+ "\n"+ "Hum: "+humidity;
        customWrite(0, ans);
}
```

Output and result analysis:



Output on serial monitor:

```
Temperature: 9
Humidity: 76.05083088954056
Temperature: 9
Humidity: 76.05083088954056
Temperature: 10
Humidity: 76.05083088954056
Temperature: 10
Humidity: 76.05083088954056
Temperature: 11
Humidity: 76.05083088954056
```

Temperature and humidity are controlled bu heating element and Humidifier which are displayed on a LCD screen.

6. Write a program to send sensor data to smart phone using Bluetooth.

Aim: To write a program to send sensor data to smart phone using Bluetooth.

Devices Used:

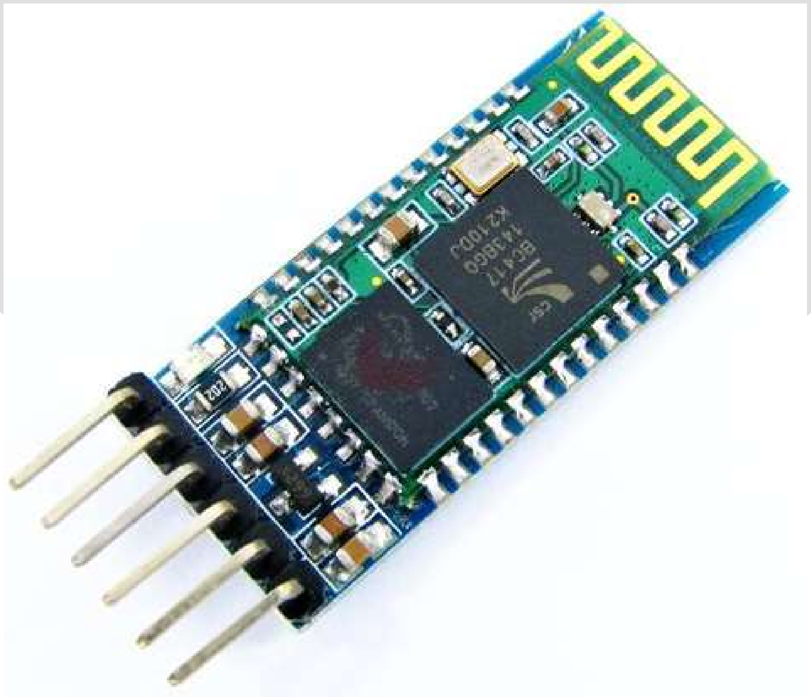
- NodeMCU
- Bluetooth-Module (HC-05, HC-06, ...)
- Android-Device
- App "Arduino Bluetooth Data"

Description:

Bluetooth Module HC-05:

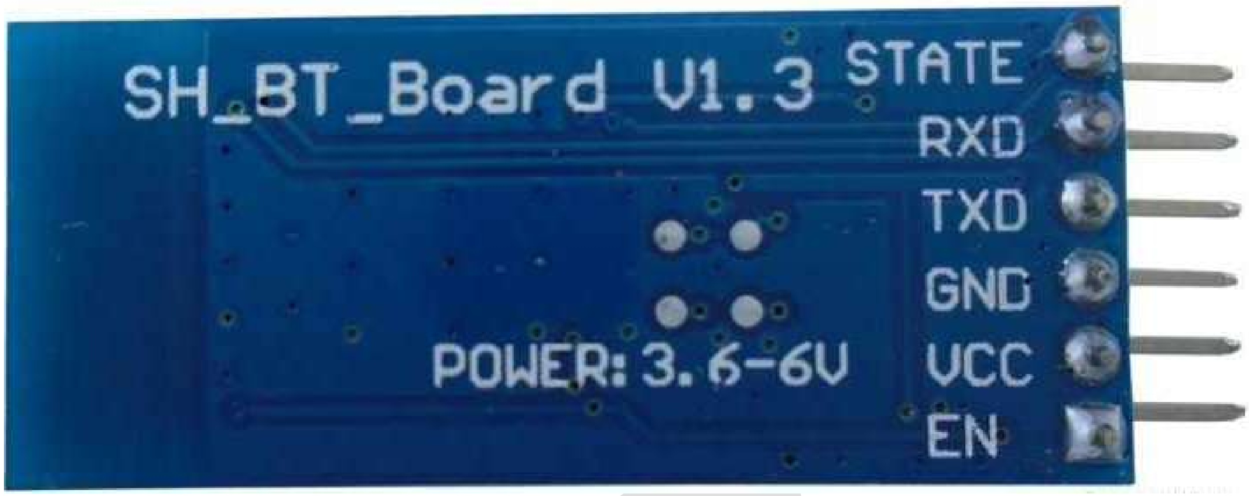
- It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications.
- It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions.
- It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.
- It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

HC-05 is a Bluetooth module which is designed for wireless comunication. This module can be used in a master or slave configuration.



HC-05 Bluetooth Module

Pin Description:



Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.. It has 6 pins,

1. Key/EN: It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

1. Data mode: Exchange of data between devices.
2. Command mode: It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

2. VCC: Connect 5 V or 3.3 V to this Pin.
3. GND: Ground Pin of module.
4. TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)
5. RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).
6. State: It tells whether module is connected or not.

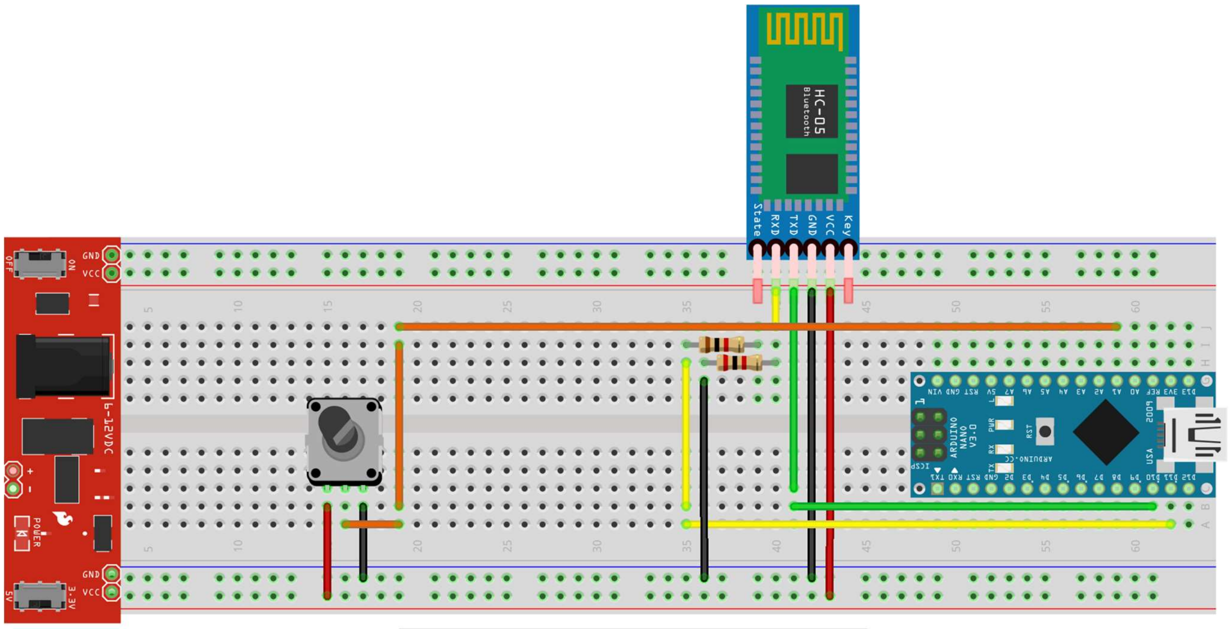
HC-05 module Information

HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.

This module works on 3.3 V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.

As HC-05 Bluetooth module has 3.3 V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.

Circuit Diagram:



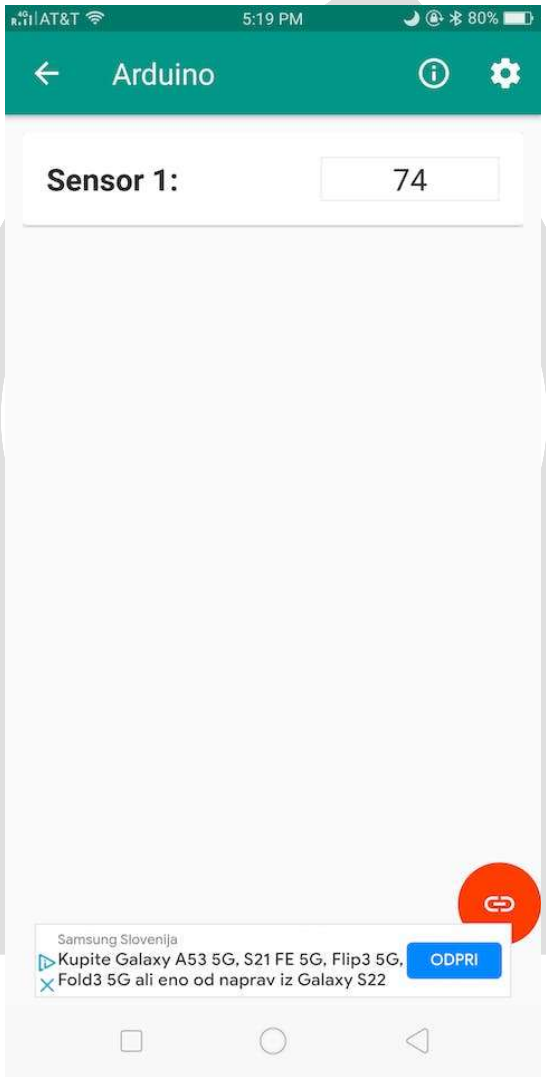
Code:

```
#include <SoftwareSerial.h>
SoftwareSerial bluetooth(10, 11); //RX, TX
int input;
int device = A1;
void setup( )
{
   Serial.begin(9600);
   bluetooth.begin(9600);
   pinMode(device, INPUT);
}

void loop( )
{
   input = analogRead(device);
   input = map(input, 0, 1023, 0, 180);
   bluetooth.print(input);
   bluetooth.print(";");
   Serial.println(input);
   delay(20);
}
```

Output and result analysis:

Application Set Up

1. Power the Arduino.
2. Now, the hc-05 module should blink rapidly.
3. Next, Open the app.
4. Allow it to access Bluetooth settings.
5. In the list, select hc-05.
6. Select receiver mode.
7. Now the module should blink once every 2 seconds.
8. Here, click the link
9. It will load for some time now.
10. Afterwards, this is what we will see

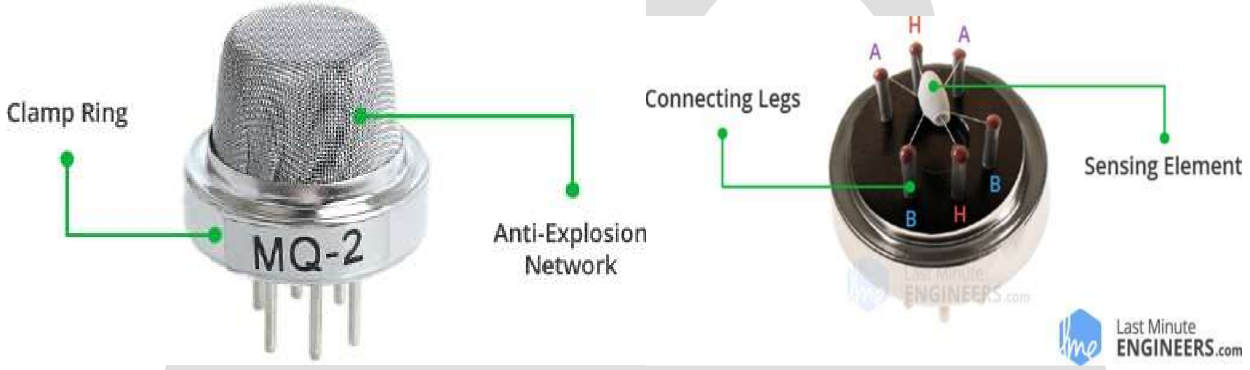

AT&T
5:19 PM
80%
Arduino
Sensor 1:
74
Samsung Slovenija
Kupite Galaxy A53 5G, S21 FE 5G, Flip3 5G, Fold3 5G ali eno od naprav iz Galaxy S22
ODPRI

7. Write a program to interface flame/smoke sensor with Arduino /Raspberry Pi and give an alert messagewhen flame/smoke is detected.

Aim: To write a program to interface flame/smoke sensor with Arduino /Raspberry Pi and give an alert message when flame/smoke is detected.

Devices Used:
NodeMCU, Alarm, Smoke Sensor, LED, Jumper wires

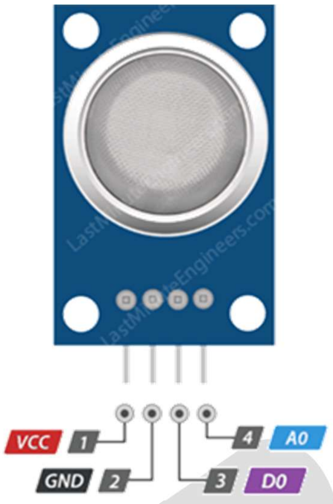
Description:
The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as Chemiresistors because sensing is based on the change in resistance of the sensing material when exposed to gasses.
The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW. It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm.

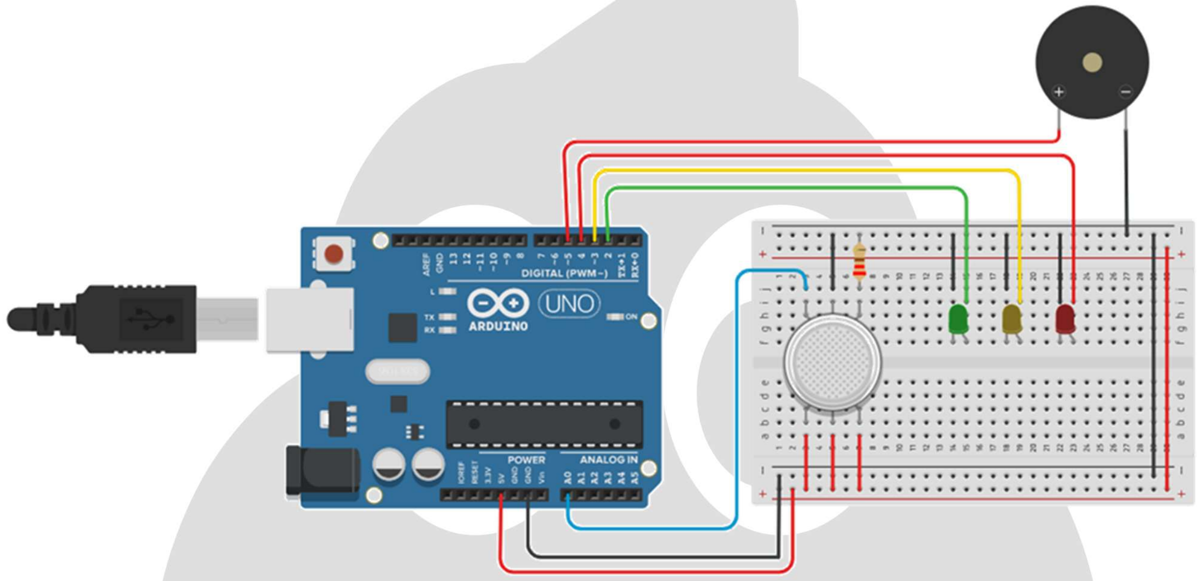


This sensor contains a sensing element, mainly aluminium-oxide based ceramic, coated with Tin dioxide, enclosed in a stainless steel mesh. Sensing element has six connecting legs attached to it. Two leads are responsible for heating the sensing element, the other four are used for output signals.
Oxygen gets adsorbed on the surface of sensing material when it is heated in air at high temperature. Then donor electrons present in tin oxide are attracted towards this oxygen, thus preventing the current flow.

When reducing gases are present, these oxygen atoms react with the reducing gases thereby decreasing the surface density of the adsorbed oxygen. Now current can flow through the sensor, which generated analog voltage values.
These voltage values are measured to know the concentration of gas. Voltage values are higher when the concentration of gas is high.

Circuit Diagram:



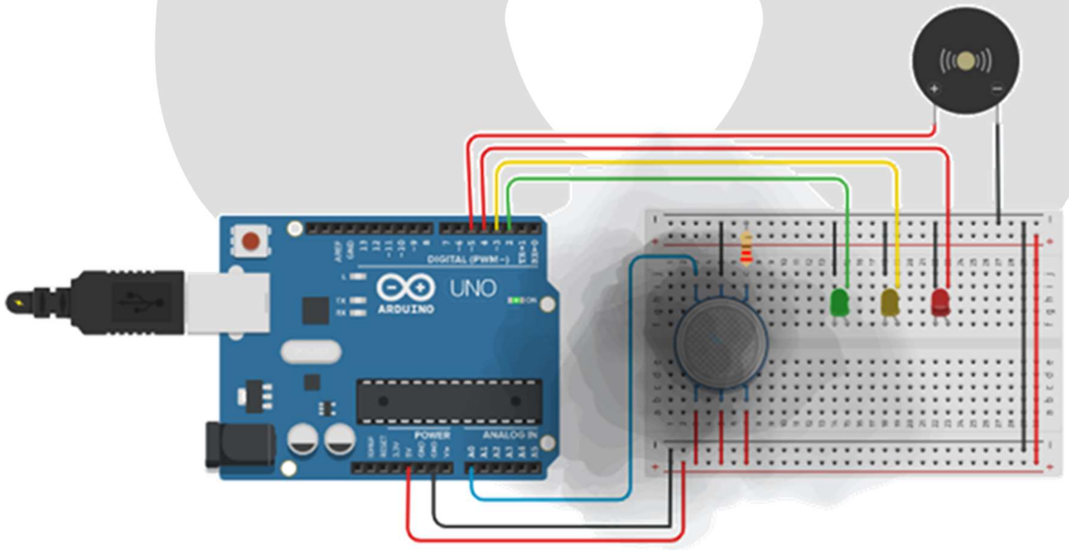


Code:

```
# define GREEN 2
# define ORANGE 3
# define RED 4
# define Buzzer 5
# define Gas_SENSOR A0
void setup( )
{
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   Serial.begin(9600);
   }
```

```
void loop( )
{
   digitalWrite(2, LOW);
   digitalWrite(3, LOW);
   digitalWrite(4, LOW);
   digitalWrite(5, LOW);

   int sensor_In = analogRead(A0);
   Serial.println(sensor_In);
   delay(500);
   if(sensor_In>=45)
   {
           digitalWrite(RED, HIGH);
         digitalWrite(Buzzer, HIGH);
   }
   else if(sensor_In>=30)
   {
           digitalWrite(ORANGE, HIGH);
   }
   else{
           digitalWrite(GREEN, HIGH);
   }

}
```

Output and result analysis:



Alarm is sounded when dense smoke is detected yellow light is on when there is less smoke and green light is turned on when there is no smoke

8. Perform experiment with GPS module by interfacing with Arduino/Raspberry Pi.

Aim: To Perform experiment with GPS module by interfacing with Arduino/Raspberry Pi.

Devices Used:
NodeMCU, NEO-6M GPS module , Jumper wires

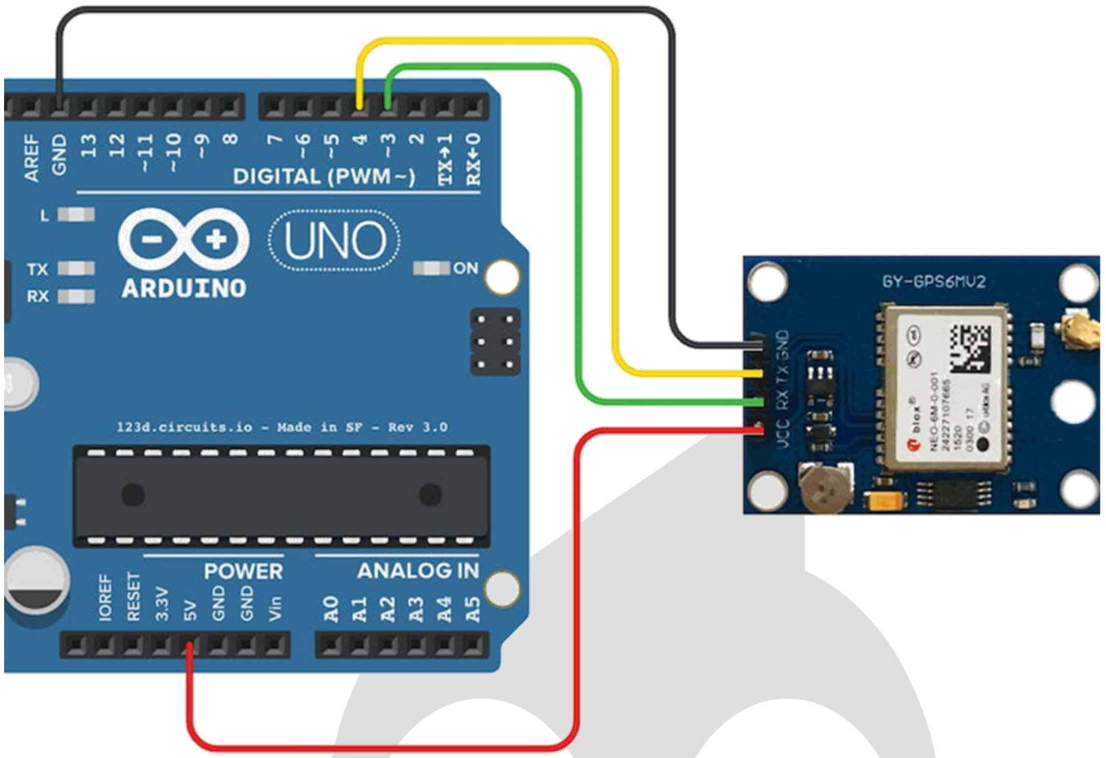
Description:
NEO-6M GPS Module: GPS stands for global positioning system and can be used to determine position, time and speed if you are travelling.

The NEO-6M GPS module has four pins: VCC, RX, TX, and GND. The module communicates with the Arduino via serial communication using the TX and RX pins, so the wiring couldn't be simpler:

NEO-6M GPS Module	Wiring to Arduino UNO
VCC	VIN
RX	TX pin defined in the software serial
TX	RX pin defined in the software serial
GND	GND

Download and install required libraries for GPS to work in Arduino IDE
(i) SoftwareSerial library
(ii) TinyGPS library

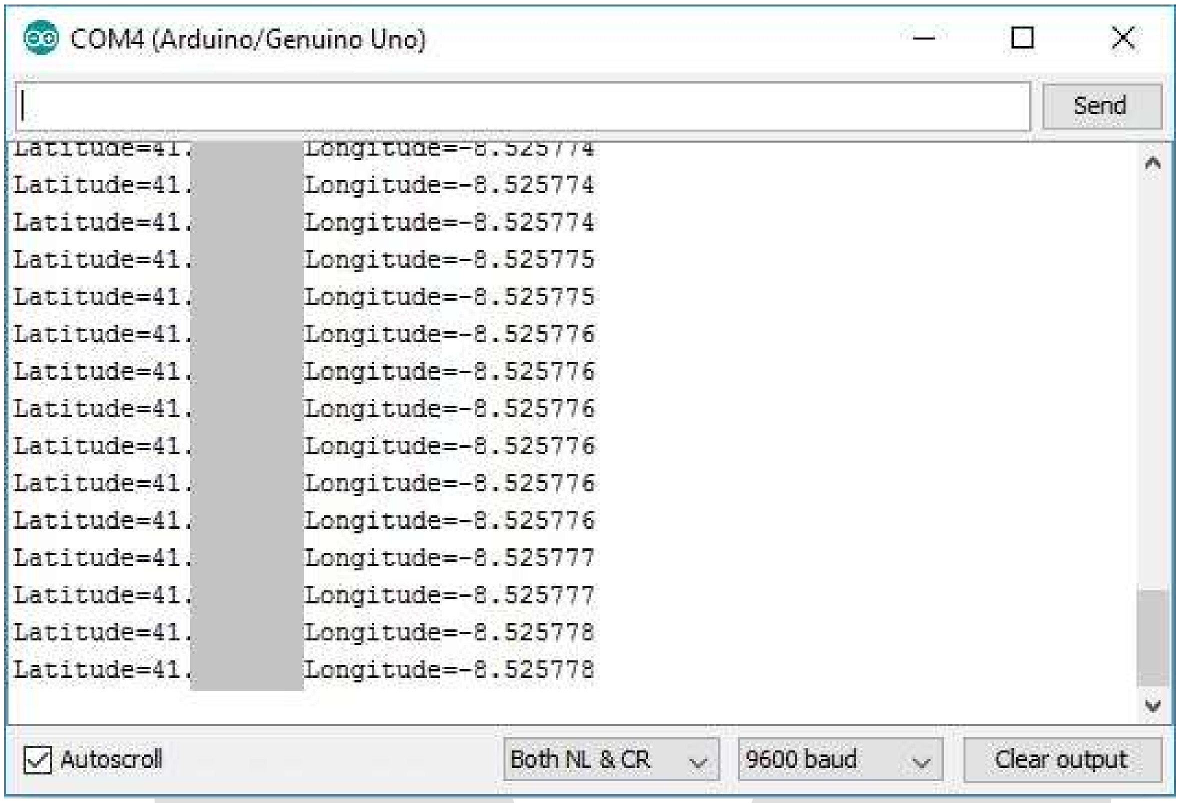
Circuit Diagram:



Code:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;    // The TinyGPS++ object
SoftwareSerial ss(RXPin, TXPin);    // The serial connection to the GPS device
void setup( )
{
   Serial.begin(9600);
   ss.begin(GPSBaud);
}
void loop( )
{
   // This sketch displays information every time a new sentence is correctly encoded.
   while (ss.available( ) > 0)
   {
      gps.encode(ss.read( );
      if (gps.location.isUpdated( ))
      {
            Serial.print("Latitude= ");
            Serial.print(gps.location.lat( ), 6);
            Serial.print(" Longitude= ");
            Serial.println(gps.location.lng( ), 6);
        }
    }
}
```

Output and result analysis:



COM4 (Arduino/Genuino Uno)

```
Latitude=41.[redacted]  Longitude=-8.525774
Latitude=41.            Longitude=-8.525774
Latitude=41.            Longitude=-8.525774
Latitude=41.            Longitude=-8.525775
Latitude=41.            Longitude=-8.525775
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525776
Latitude=41.            Longitude=-8.525777
Latitude=41.            Longitude=-8.525777
Latitude=41.            Longitude=-8.525778
Latitude=41.            Longitude=-8.525778
```

Autoscroll ✓ | Both NL & CR | 9600 baud

Coordinates of the recorded location are displayed on the serial monitor.

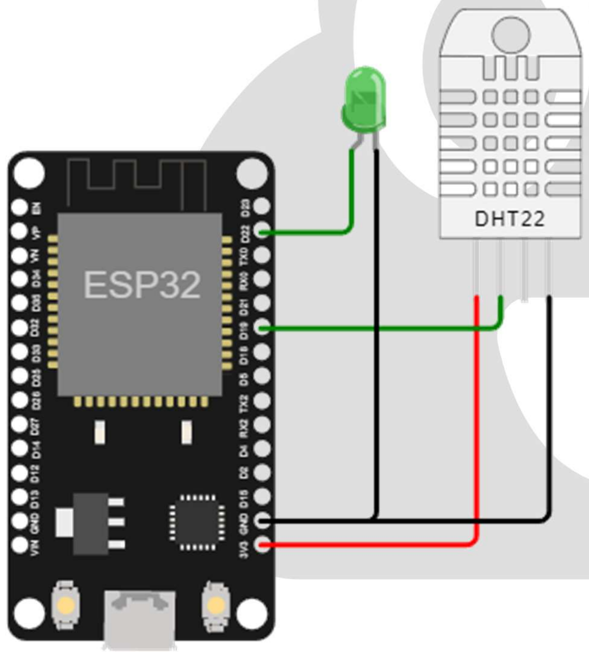
9. Write a program to send and receive messages using MQTT protocol

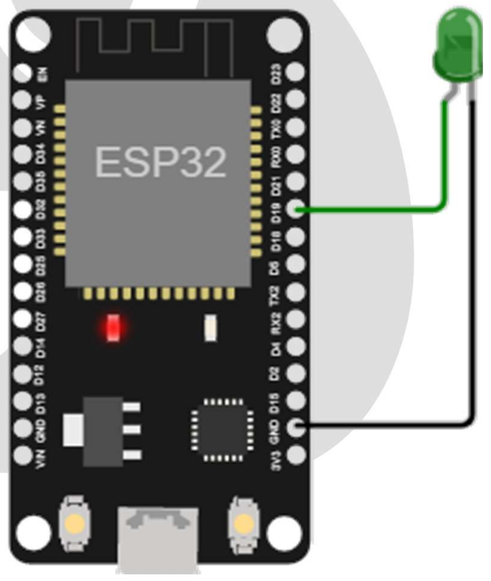
Aim: To send and receive messages using MQTT protocol.

Devices Used:
NodeMCU, Adafruit library and platform, mosquitto platform, DHT22 or tempearature and humidity sensors

Description:
MQTT (originally an initialism of MQ Telemetry Transport) is a lightweight, publish-subscribe, machine to machine network protocol for Message queue/Message queuing service. The sender (Publisher) and the receiver (Subscriber) communicate via Topics and are decoupled from each other. The connection between them is handled by the MQTT broker. The MQTT broker filters all incoming messages and distributes them correctly to the Subscribers.

Circuit Diagram:
Publisher: **Subscriber:**





Code:
Publisher:
/* In this program, we will send temp and humidity values
/* from one board (publisher) to another board (subscriber)
/* using MQTT protocol */

```
#include <WiFi.h>
#include <ArduinoMqttClient.h>
#include <DHTesp.h>
// DHTesp.h is named as DHT library for ESPx in Wokwi's library manager
// thus use that name if you want to add this library
// this library works with Arduino boards as well, not just ESP

// ****** Simulated WiFi Credintials for Wokwi ******
const char ssid[] = "Wokwi-GUEST";
const char pass[] = "";

// ****** Creating Instances from Classes (OOP) ******
WiFiClient wifiClient;   // provided by WiFi.h
MqttClient mqttClient(wifiClient); // by ArduinoMqttClient.h
DHTesp dht; // by DHTesp.h

// ****** Initialising MQTT Variables ******
const char broker[] = "test.mosquitto.org"; // broker's url
const int port = 1883; // broker's port
// mosquitto.org provides a free MQTT broker,
// thus we're using it for now

const char topic0[] = "c5tempValue";
const char topic1[] = "c5humValue";
// for each datapoint you want to send, you create a topic
const int dhtPin = 19;
const int led = 22;

void setup(){
  Serial.begin(9600);
  pinMode(led, OUTPUT);

  //****** Setting up DHT22 ******
  pinMode(dhtPin, INPUT);
  dht.setup(dhtPin, DHTesp::DHT22);
  // since DHT has many models, dht.setup() tells which model we're using

  //****** Connecting to Simulated WiFi ******
  Serial.print("Connecting to WiFi:");
  WiFi.begin(ssid, pass, 6); // 6 means WiFi Channel 6
  while ( WiFi.status() != WL_CONNECTED){
    Serial.print('.'); // prints dots till it connects to wifi
    delay(2000);
  }
  Serial.println("Connected!");

  //****** Connecting to MQTT Broker ******
  Serial.println("Connecting to MQTT Broker...");
  if( !mqttClient.connect(broker, port)){
    Serial.print("Connection to broker failed!: ");
    Serial.println(mqttClient.connectError());
  }
  else
    Serial.println("Connected to MQTT Broker!");
}

void loop( ){
  // ****** Keeping MQTT Connection alive ******
  mqttClient.poll( );

  // ****** Reading values from DHT22 ******
  int temp = dht.getTemperature( );
  int hum = dht.getHumidity( );

  // ****** Sending temp value thru topic0 ******
  mqttClient.beginMessage(topic0);
  mqttClient.print(temp);
  mqttClient.endMessage( );

  // ****** Sending hum value thru topic1 ******
  mqttClient.beginMessage(topic1);
  mqttClient.print(hum);
  mqttClient.endMessage( );
  Serial.printf("Sent values: %d'C     %d%% \n", temp, hum);
  delay(2000);
  // below is optional, it just blinks that green LED upon a successful send
  digitalWrite(led, HIGH);
  delay(600);
  digitalWrite(led, LOW);
}
```

Subscriber:

```
#include <WiFi.h>
#include <ArduinoMqttClient.h>

// ****** Simulated WiFi Credintials ******
const char ssid[] = "Wokwi-GUEST";
const char pass[] = "";

// ****** Creating Instances from Classes (OOP) ******
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

// ****** Initialising MQTT Variables ******
const char broker[ ] = "test.mosquitto.org";
const int port = 1883;
// mosquitto.org provides a free MQTT broker, thus we're using it
const char topic0[ ] = "c5tempValue";
const char topic1[ ] = "c5humValue";
// for each datapoint you want to receive, you make a topic
const int led = 19;
void setup(){
  Serial.begin(9600);
  pinMode(led, OUTPUT);

  //****** Connecting to Simulated WiFi ******
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, pass, 6);
  while ( WiFi.status() != WL_CONNECTED){
    Serial.print('.');
    delay(2427);
  }
  Serial.println("Connected!");

  //****** Connecting to MQTT Broker ******
  if ( !mqttClient.connect(broker, port)){
    Serial.print("Error connecting to broker: ");
    Serial.println(mqttClient.connectError());
  }
  else
    Serial.println("Connected to Broker!");
  // ****** Subscribing to Topics provided by broker ******
  mqttClient.onMessage(onMqttMessage);
  // invokes onMqttMessage( ) function when a msg is received
  mqttClient.subscribe(topic0);
  mqttClient.subscribe(topic1);
  Serial.printf("Subscribed to: %s   %s \n", topic0, topic1);
}
void loop( ){
  // ****** Keeping MQTT Connection Alive ******
  mqttClient.poll( );
}

// ****** Function invoked when new message received ******
void onMqttMessage(int messageSize) {
  // ****** Printing Topic Name ******
  Serial.printf("Topic %s has sent: ",mqttClient.messageTopic());

  // ****** Printing Topic's data ******
  while (mqttClient.available())
    Serial.print((char)mqttClient.read());
  /* above mqttClient.read() fn returns the value sent
  /* by our publisher BYTE BY BYTE.
  /* Thus, the while loop casts every byte to a char
  /* and prints them one by one */
  Serial.println();
  // below is optional, it just blinks that green LED upon a successful receive
  digitalWrite(led, HIGH);
  delay(600);
  digitalWrite(led, LOW);
}
```

Using Adafruit:

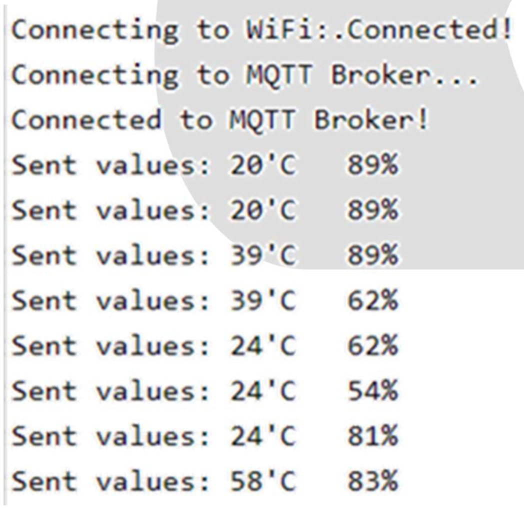
```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/******************** WiFi Access Point **************************/
#define WLAN_SSID           "rd"
#define WLAN_PASS           "1234567890"
```

```
/************************* Adafruit.io Setup *****************************/
#define AIO_SERVER          "io.adafruit.com"
#define AIO_SERVERPORT    1883                          // use 8883 for SSL
#define AIO_USERNAME       "my_username"
#define AIO_KEY              "my_key"
/************* Setup *******************************/

/************ Global State (you don't need to change this!) ******************/
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiFlientSecure for SSL
//WiFiClientSecure 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);
/******************************* Feeds *******************************/
// Setup a feed called 'distance' & 'feed1' for publishing & Subscribe.

Adafruit_MQTT_Publish distance = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/mydist");
Adafruit_MQTT_Subscribe feed1 = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/feed1");
/*************************** Sketch Code *********************************/

const int trigPin = 5;//D1
const int echoPin = 4;//D2

//define sound velocity in cm/uS
//#define soundbuzzer 14

#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

//int sound =500;
long duration;
float distanceCm;
float distanceInch;

void MQTT_connect();
void setup() {
   Serial.begin(115200);
   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
   //pinMode(soundbuzzer, OUTPUT);
   delay(10);
   Serial.println(F("Adafruit MQTT demo"));
   // Connect to WiFi access point.
   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(WLAN_SSID);
   WiFi.begin(WLAN_SSID, WLAN_PASS);
   while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.println("try to connecting please wait.........");
   }
   Serial.println();
   Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
   mqtt.subscribe(&feed1);
}
uint32_t x=0;
uint8_t slider1=0;
void loop() {
   // Ensure the connection to the MQTT server is alive (this will make the first
   // connection and automatically reconnect when disconnected).    See the
MQTT_connect
   // function definition further below.
   MQTT_connect();

   //********************Subscription********************
   Adafruit_MQTT_Subscribe *subscription;
   while ((subscription = mqtt.readSubscription(5000))) {
      if (subscription == &feed1) {
            Serial.println((char *)feed1.lastread);

         }
      }

   //***************Publishing***************************

  digitalWrite(trigPin, LOW);
   delayMicroseconds(2);
   // Sets the trigPin on HIGH state for 10 micro seconds
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
     // Reads the echoPin, returns the sound wave travel time in microseconds
   duration = pulseIn(echoPin, HIGH);

   // Calculate the distance
   distanceCm = duration * SOUND_VELOCITY/2;

   // Convert to inches
   distanceInch = distanceCm * CM_TO_INCH;

   // Prints the distance on the Serial Monitor
   Serial.print("Distance (cm): ");
   //For Cloud Monitoring data
   distance.publish(distanceCm);
   // For Serial monitor data
   Serial.println(distanceCm);
   Serial.print("Distance (inch): ");
   Serial.println(distanceInch);
/*   if(distanceCm >5)
   {
        tone(soundbuzzer, sound);
        delay(1000);
        noTone(soundbuzzer);
        delay(1000);
        }
     else
     {
      noTone(soundbuzzer);
     }*/
   delay(3000);
}
```

```
// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
  int8_t ret;
  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }
  Serial.print("Connecting to MQTT... ");
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
        Serial.println(mqtt.connectErrorString(ret));
        Serial.println("Retrying MQTT connection in 5 seconds...");
        mqtt.disconnect();
        delay(5000);   // wait 5 seconds
        retries--;
        if (retries == 0) {
          // basically die and wait for WDT to reset me
          while (1);
        }
  }
  Serial.println("MQTT Connected!");
}
```

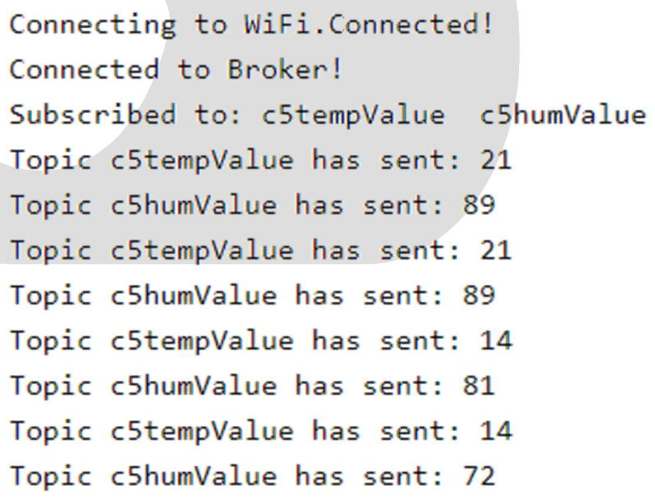
Output and result analysis:

Publisher:



```
Connecting to WiFi:.Connected!
Connecting to MQTT Broker...
Connected to MQTT Broker!
Sent values: 20'C   89%
Sent values: 20'C   89%
Sent values: 39'C   89%
Sent values: 39'C   62%
Sent values: 24'C   62%
Sent values: 24'C   54%
Sent values: 24'C   81%
Sent values: 58'C   83%
```

Subscriber:



```
Connecting to WiFi.Connected!
Connected to Broker!
Subscribed to: c5tempValue  c5humValue
Topic c5tempValue has sent: 21
Topic c5humValue has sent: 89
Topic c5tempValue has sent: 21
Topic c5humValue has sent: 89
Topic c5tempValue has sent: 14
Topic c5humValue has sent: 81
Topic c5tempValue has sent: 14
Topic c5humValue has sent: 72
```

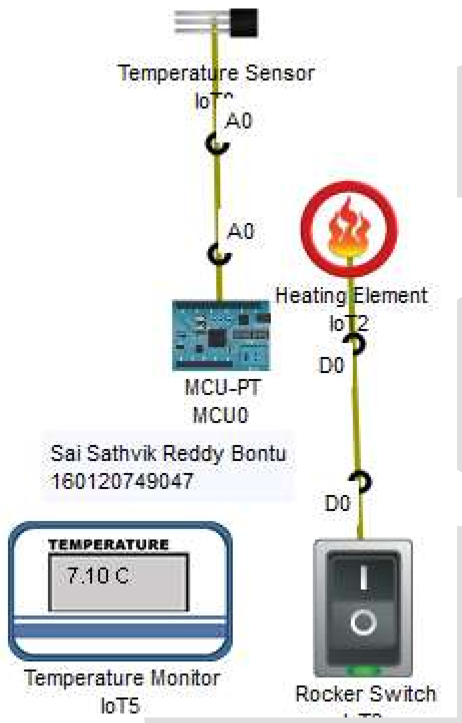
10.Write a program to upload sensor data to local/cloud server using wifi

Aim: To upload sensor data to local/cloud server using wifi

Devices Used:
NodeMCU, Jumper wires, tempearature and humidity sensors

Description:
ESP8266 has WiFi module using which the data read by sensors is send to webhost database tables by using the below arduino code, PHP and SQL.

Circuit Diagram:



Code:

```
var url = "https://sathvik2017.000webhostapp.com/dbinsert.php?data=";
function setup( ) {
        pinMode(0, INPUT);
        pinMode(1, OUTPUT);
}

function loop( ) {
        var temp=analogRead(A0);
        temp=Math.floor(map(temp,0,1023,-100,100));
        Serial.println(temp);
        var http = new RealHTTPClient();
        http.onDone = function(status, data)
        {
                Serial.println("status: " + status);
                Serial.println("data: " + data);
        };
        http.get(url+temp);
        delay(10000);
}
```

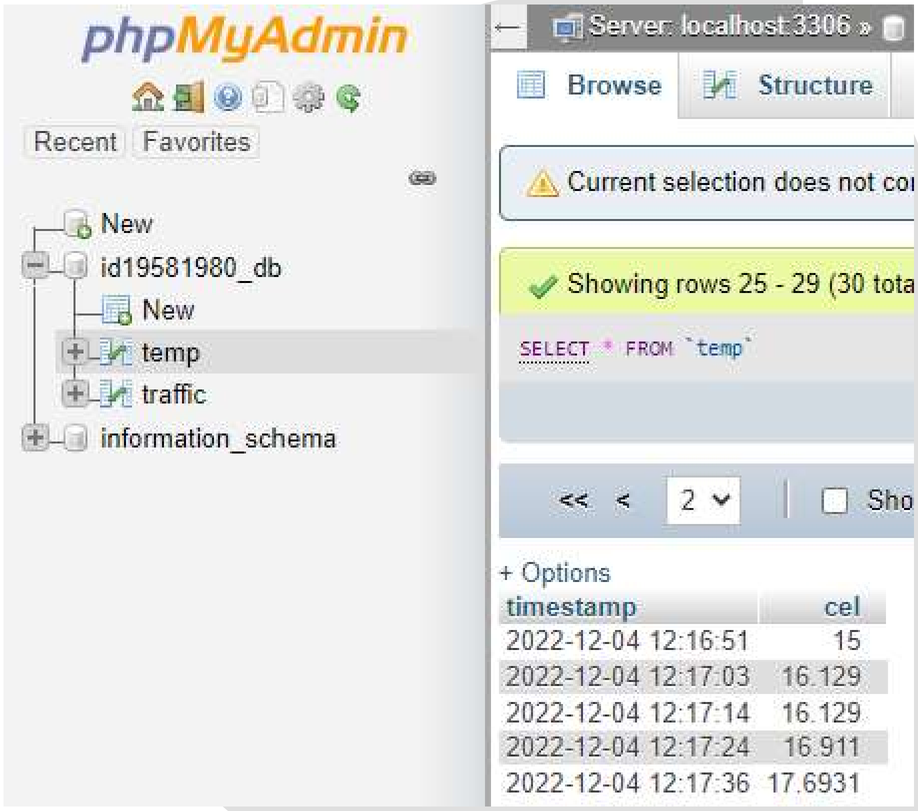
dbinsert.php:

```
<?php
//if($_GET['key']!="54321") die("Invalid key");
$servername = "localhost";
$username = "user";
$password = "my_password";
$dbname = "my_db";

// Create connection
$conn = mysqli_connect($servername, $username, $password,$dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";

$sql = "INSERT INTO temp (cel) VALUES (".$_GET['data'].")";

if (mysqli_query($conn, $sql)) {
   echo "<br>New record created successfully";
} else {
   echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}


mysqli_close($conn);

?>
```

Output and result analysis:

```
status: 200
data: Connected successfully<br>New record created successfully
16.129032258064512
status: 200
data: Connected successfully<br>New record created successfully
16.911045943304003
status: 200
data: Connected successfully<br>New record created successfully
17.693059628543494
status: 200
data: Connected successfully<br>New record created successfully
```

Table in the database has the values as shown below:



SELECT * FROM `temp`

timestamp	cel
2022-12-04 12:16:51	15
2022-12-04 12:17:03	16.129
2022-12-04 12:17:14	16.129
2022-12-04 12:17:24	16.911
2022-12-04 12:17:36	17.6931

11. Write a program to retrieve sensor data from local/cloud server

Aim: To retrieve sensor data from local/cloud server

Devices Used:
NodeMCU, Jumper wires, tempearature and humidity sensors

Description:
ESP8266 has WiFi module using which the data read by sensors is send to webhost database tables by using the below arduino code, PHP and SQL. Then the same data is read using the same technologies.

Code:
```
var url = "https://sathvik2017.000webhostapp.com/dbselect.php";
function setup( ) {
        pinMode(0, INPUT);
        pinMode(1, OUTPUT);
}

function loop( ) {
        var temp=analogRead(A0);
        temp= map(temp,0,1023,-100,100);
        Serial.println(temp);
        var http = new RealHTTPClient( );
        http.onDone = function(status, data)
        {
                Serial.println("status: " + status);
                Serial.println("data: " + data);
        };
        http.get(url);
        delay(10000);
}
```

dbselect.php:
```
<?php
$servername = "localhost";
$username = "user";
$password = "my_password";
$dbname = "my_db";
// Create connection
```

```
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
$sql = "SELECT timestamp, cel FROM temp";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
   // output data of each row
   while($row = mysqli_fetch_assoc($result)) {
      echo "Timestamp: " . $row["timestamp"]. " - Temperature(C): " . $row["cel"]."\n";
   }
} else {
   echo "0 results";
}
mysqli_close($conn);
?>
```

Output and result analysis:

```
status: 200
data: Timestamp: 2022-10-01 07:26:10 - Temperature(C): 26.5
Timestamp: 2022-10-01 07:31:18 - Temperature(C): 26.5
Timestamp: 2022-10-01 07:40:04 - Temperature(C): 47.2
Timestamp: 2022-10-01 07:46:45 - Temperature(C): 6
Timestamp: 2022-10-01 07:46:56 - Temperature(C): 7
Timestamp: 2022-10-01 07:47:07 - Temperature(C): 8
Timestamp: 2022-10-01 09:06:35 - Temperature(C): 98
Timestamp: 2022-10-01 09:17:58 - Temperature(C): 80
Timestamp: 2022-10-01 09:18:09 - Temperature(C): 80
Timestamp: 2022-10-01 09:18:20 - Temperature(C): 81
Timestamp: 2022-12-01 08:08:43 - Temperature(C): 5.18084
Timestamp: 2022-12-01 08:08:54 - Temperature(C): 5.96285
Timestamp: 2022-12-01 08:10:54 - Temperature(C): 52.45
Timestamp: 2022-12-04 12:16:30 - Temperature(C): 14
Timestamp: 2022-12-04 12:16:40 - Temperature(C): 14
Timestamp: 2022-12-04 12:16:51 - Temperature(C): 15
Timestamp: 2022-12-04 12:17:03 - Temperature(C): 16.129
Timestamp: 2022-12-04 12:17:14 - Temperature(C): 16.129
Timestamp: 2022-12-04 12:17:24 - Temperature(C): 16.911
Timestamp: 2022-12-04 12:17:36 - Temperature(C): 17.6931
```