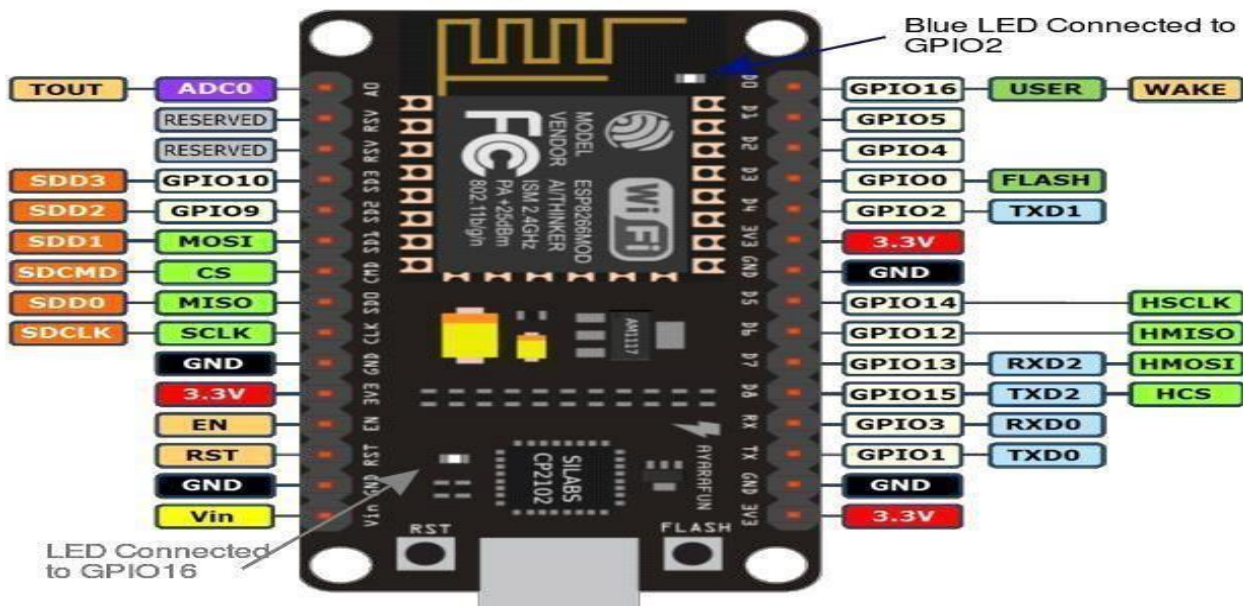


## NodeMCU - Experiments

### Experiment-1 Study of NodeMCU PIN diagram

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, Node MCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board. NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core widely used in IoT applications. NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then Node MCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to Node MCU project, enabling Node MCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs. Since NodeMCU is an open source platform, their hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consists of ESP8266 wifi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 WiFi Module. There is Version2 (V2) available for NodeMCU Dev Kit i.e. NodeMCU Development Board v1.0 (Version2), which usually comes in black colored PCB.



Nodemcu Dev Kit has Arduino like Analog (i.e. A0) and Digital (D0-D8) pins on its board. It supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time.

NodeMCU Development kit provides access to these GPIOs of ESP8266. The only thing to take care is that NodeMCU Dev kit pins are numbered differently than internal GPIO notations of ESP8266 as shown in below figure and table. For example, the D0 pin on the NodeMCU Dev kit is mapped to the internal GPIO pin 16 of ESP8266. The GPIO's shown in blue box (1, 3, 9, 10) are mostly not used for GPIO purpose on Dev Kit. ESP8266 is a system on a chip (SoC) design with components like the processor chip. The processor has around 16 GPIO lines, some of which are used internally to interface with other components of the SoC, like flash memory.

Since several lines are used internally within the ESP8266 SoC, we have about 11 GPIO pins remaining for GPIO purpose. Now again 2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC from which compiled object code is downloaded.

Hence finally, this leaves just 9 general purpose I/O pins i.e. D0 to D8.

As shown in above figure of NodeMCU Dev Kit. We can see RX, TX, SD2, SD3 pins are not mostly used as GPIOs since they are used for other internal process. But we can try with SD3 (D12) pin which mostly like to respond for GPIO/PWM/interrupt like functions. Note that D0/GPIO16 pin can be only used as GPIO read/write, no special functions are supported on it.

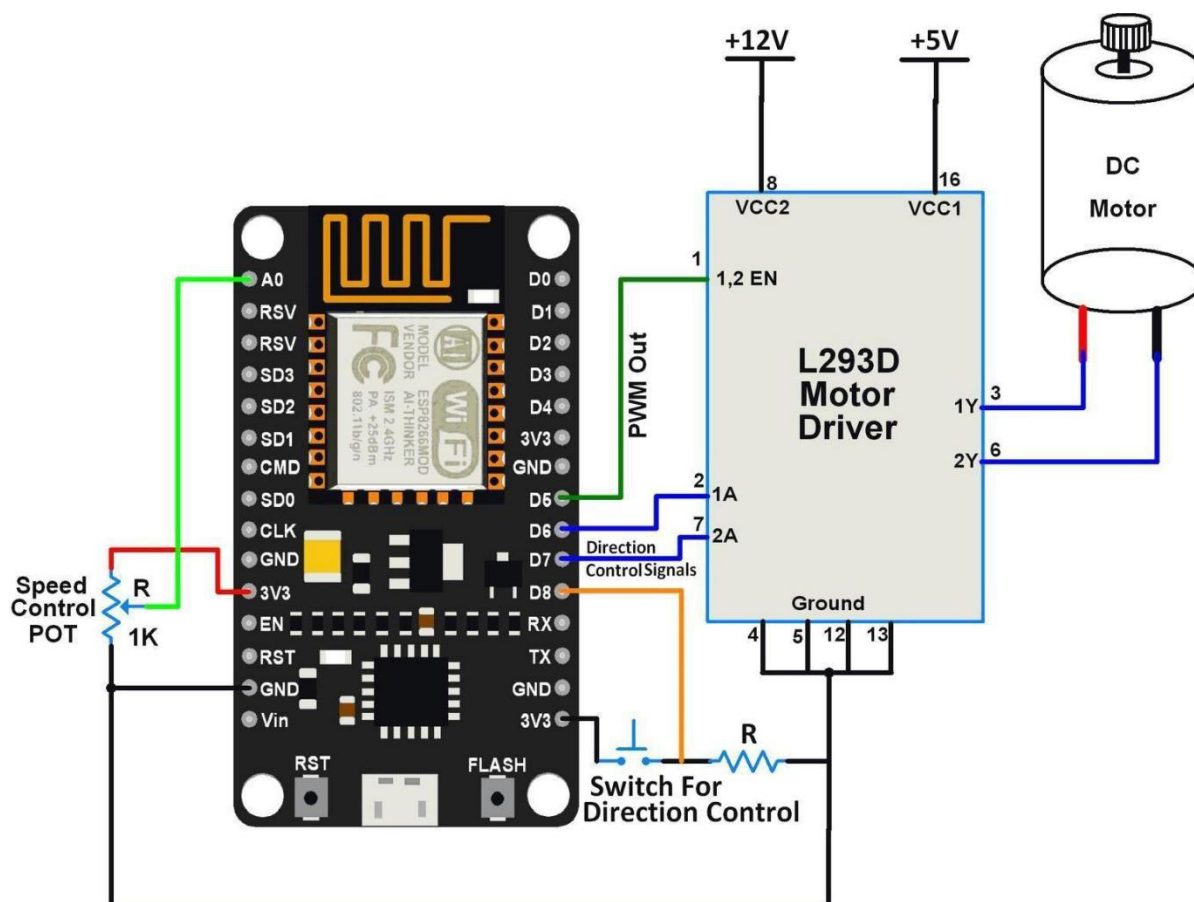
## PIN DESCRIPTION:

Pin Names on NodeMCU Development Kit	ESP8266 Internal GPIO Pin number
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD2	GPIO9
D12/SD3	GPIO10

## 2. Study of various sensors used in NodeMCU

### 2.1 DC motor

DC motor converts electrical energy in the form of Direct Current into mechanical energy in the form of rotational motion of the motor shaft.



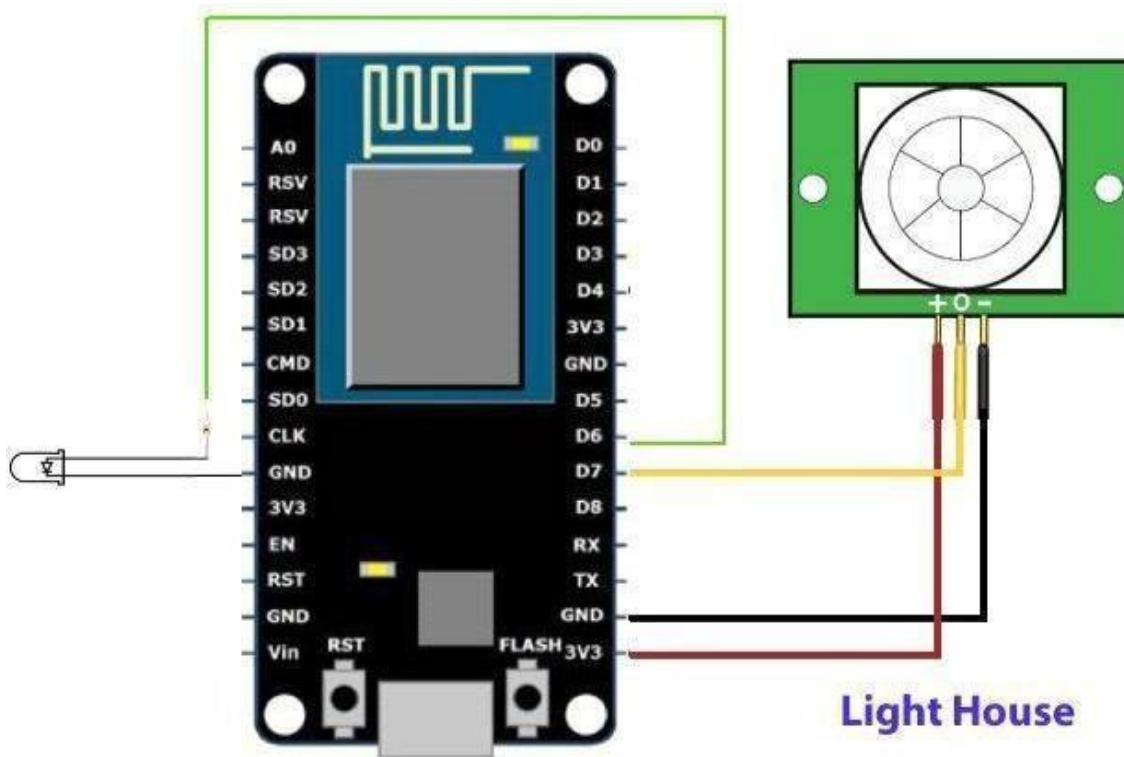
The DC motor speed can be controlled by applying varying DC voltage; whereas the direction of rotation of motor can be changed by reversing the direction of current through it.

For applying varying voltage, we can make use of PWM technique. For reversing the current, we can make use of H-Bridge circuit or motor driver IC's that employ the H-Bridge technique.

## 2.2 PIR Sensor

Pyroelectric / Passive Infra Red sensor : PIR sensors allow you to sense motion, generally used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out.

Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor. Sensitivity range: up to 20 feet (6 meters) 110 degrees x 70 degrees detection range.



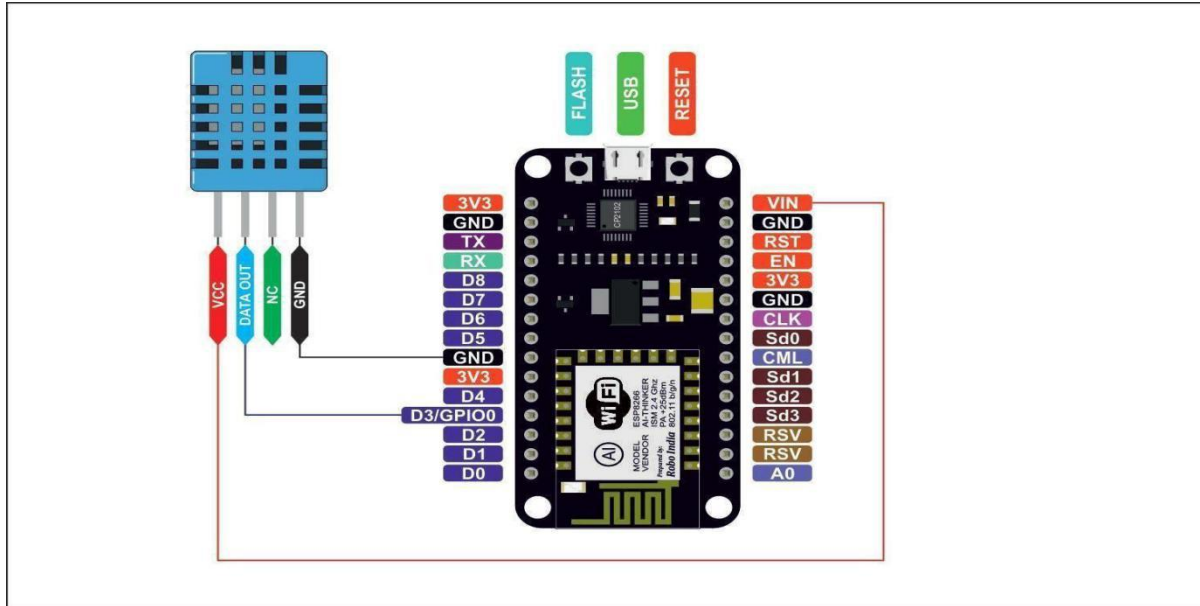
Connecting PIR sensors to a NodeMCU is really simple. The PIR acts as a digital output so all you need to do is operate the pin to flip high (detected) or low(not detected).Most PIR modules have a 3-pin connection at the side or bottom. The pinout may vary between modules so check the pinout carefully! Power is usually3-5v DC input.

The circuit connections are made as follows :

- Vcc pin of the HC-SR501 is connected to +3v of the NodeMCU.
- Output Pin of the HC-SR501 is connected to Digital pinD7 of the NodeMCU. GND pin of the HC-SR501 is connected to Ground pin (GND) of the NodeMCU

## 2.3 DHT11 for Temperature and Humidity

The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes



Wiring the DHT11 to the NodeMCU is really easy, but the connections are different depending on which type you have either 3-pins or 4-pins.

The wiring connections are made as follows :

- Pin 1 of the DHT11 goes into +3v of the NodeMCU.
- Pin 2 of the DHT11 goes into Digital Pin D4 of the NodeMCU.
- Pin 3 of the DHT11 goes into Ground Pin (GND) of the NodeMCU.
- Wiring the DHT11 to the NodeMCU is really easy, but the connections are different depending on which type you have either 3-pins or 4-pins.

The wiring connections are made as follows :

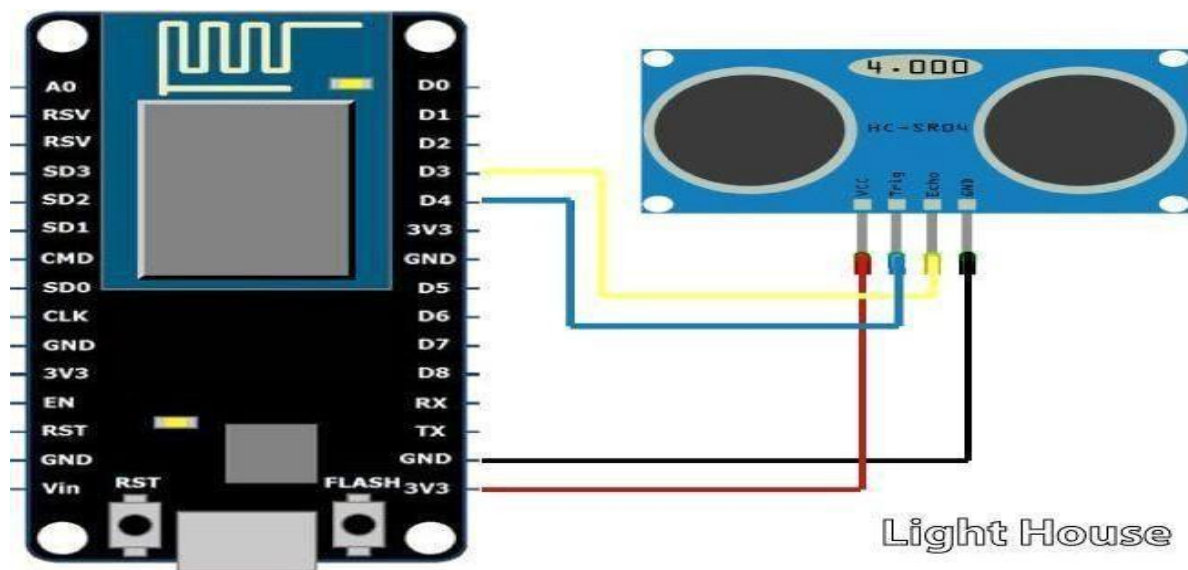
- Pin 1 of the DHT11 goes into +3v of the NodeMCU.
- Pin 2 of the DHT11 goes into Digital Pin D4 of the NodeMCU.
- Pin 3 of the DHT11 goes into Ground Pin (GND) of the NodeMCU.

## 2.4 Ultrasonic Sensor(HC-SR04)

An Ultrasonic sensor is a device that can measure the distance of an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and waits for that sound wave to bounce back. By recording the time taken between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sensor and the object.

### Specifications of HC-SR04

1. Power supply : 5v DC
  - Ranging distance : 2 cm – 500 cm
  - Ultrasonic Frequency : 40k Hz



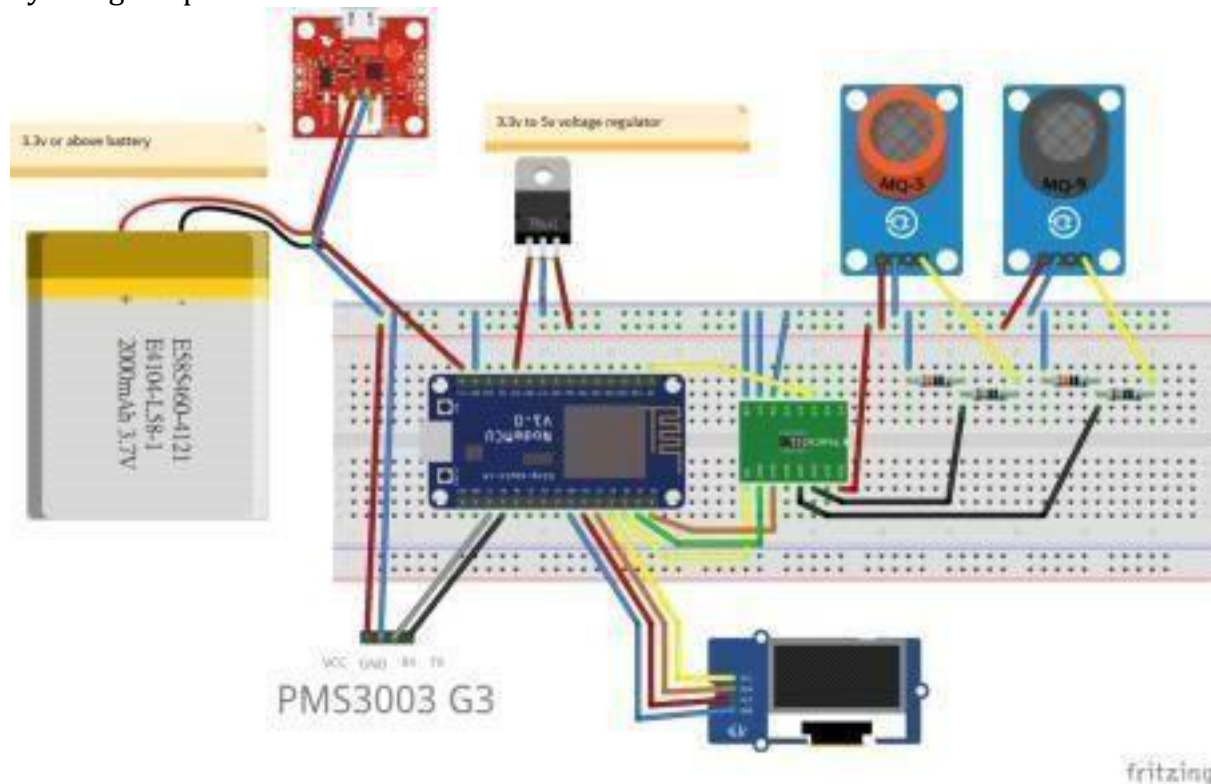
The circuit connections are made as follows:

The HC- SR04 sensor attach to the Breadboard. The sensor Vcc is connected to the NodeMCU +3.3v. The sensor GND is connected to the NodeMCU GND. The sensor Trigger Pin is connected to the NodeMCU Digital I/O D4 The sensor Echo Pin is connected to the NodeMCU Digital I/O



## 2.5 Gas Sensor

The Grove-Gas Sensor(MQ5)module is useful for gas leakage detection(in home and industry). It is suitable for detecting H<sub>2</sub>, LPG, CH<sub>4</sub>, CO, Alcohol. Due to its high sensitivity and fast response time, measurements can be taken as soon as possible. The sensitivity of the sensor can be adjusted by using the potentiometer.



MQ2 and MQ9 gas sensor both have four pins:

- VCC
- GND
- DO(Digital Control)
- AO(Analog Output)

The analog output of MQ2 and MQ9 is between 0V to 5V whereas the analog pin of nodemcu can only read between 0V to 3.3V. That means nodemcu cannot read the data if MQ2or MQ9 sensor output is above 3.3V. The data read is not accurate. Therefore, voltage is needed step down. So, Vin is connected to AO pin of MQ gas sensors. Vout is connected to channels of the multiplexer. Only three pins of each sensor are used:

- Vcc to 5V supply
- GND to nodemcu GND pin
- Vout of voltage divider to CD4051BE channel 1 and channel 2(pin 14 and 15) multiplexer

- Vdd (pin 16) to 5V supply
- INH, Vee, Vss (pin 6, 7, 8) to nodemcu GND pin
- common out/in (pin 3) to nodemcu A0 pin
- A, B, C (pin 11, 10, 9) to nodemcu D0, D1, D2

A, B, C (pin11, 10, 9) are used to select channel for output. A, B, C are Digital Input Which Means Only Read 0 and 1. 3-digit binary number is formed in the order of CBA. A sub Channel 1 and 2, denary number of 1 and 2 in 3-digit binary number are 001 and 010 respectively. Therefore, when we want output of channel 1, D0 output 1, D1 and D2 output 0. When we want output of channel 2, D0 output 0, D1 output 1 and D2 output 0.



## **Experiment - 2 Installation of Arduino and Home Automation System**

### **1. Installation of Arduino:**

- 1)Download the Arduino Software IDE from <https://www.arduino.cc>
- 2)Open the installer and allow the driver installation
- 3)Choose the components to install.
- 4)Choose the directory (keeping the default one is suggested).
- 5)Click on next, Arduino will be installed.

### **1.1 Configuration Steps (After Installation) :**

- 1) In the menu go to Files, then open preferences.
- 2) Enter the URL “ [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)” at the Additional Boards Manager URLs.
- 3) In the menu go to Tools, select Boards and open Board Manager.
- 4) Search for esp8266 and install the package.

### **1.2 WHAT IS ARDUINO?**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

### **1.3 WHY ARDUINO?**

Arduino boards are relatively inexpensive compared to other micro controller platforms. The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most micro controller systems are limited to Windows. The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. The Arduino software is published as open source tools, available for extension by experienced programmers.

## 2.Home Automation System

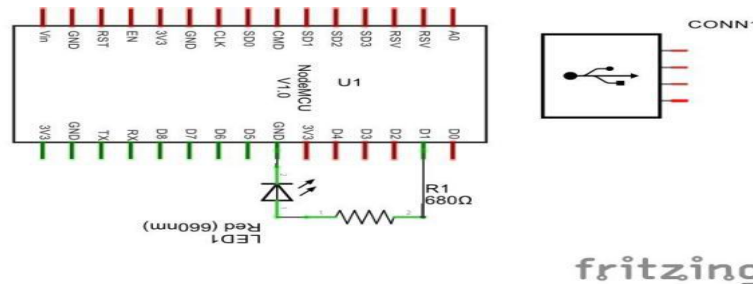
### 2.1 Experiment to make the LED ON and OFF using the NodeMCU

**Aim:** To make the LED ON and OFF using Node MCU.

**Description:** NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 WiFi SoC from Espressif, and hardware which is based on the ESP-12 module. The term "NodeMcu" by default refers to the firmware rather than the dev kits. The firmware ESP8266 uses the Lua scripting language. ESP8266 Wifi modules includes a CP2102 TTL to USB chip for programming and debugging, is breadboard-friendly, and can simply be powered via its micro USB port.

**Hardware Requirements:** NodeMCU, USB cable, LED, Resistor

**Circuit Diagram:**



**Program:**

```
void setup(){
// initialize digital pin on 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 delay(1000); //wait for a second
digitalWrite( LED_BUILTIN,LOW); //turn the LED off delay(1000); //wait for a second
}
```

**Steps:**

- Connect NodeMCU to computer using USB cable.
- In the menu go to tools, under port select the port through which NodeMCU is connected to computer.
- Compile the program and upload.

**Output:**

LED on the NodeMCU blinks.

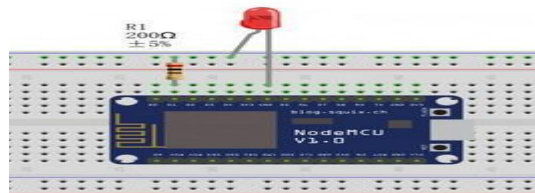
## 2.2 Experiment to make the external LED ON and OFF.

**Aim:** To make an external LED blink using NodeMCU.

**Description:** NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 WiFi SoC from Espressif, and hardware which is based on the ESP-12 module. The term "NodeMcu" by default refers to the firmware rather than the dev kits. The firmware ESP8266 uses the Lua scripting language. ESP8266 Wifi modules include a CP2102 TTL to USB chip for programming and debugging, is breadboard-friendly, and can simply be powered via its micro USB port.

**Hardware Requirements:** NodeMCU, USB cable, LED

**Circuit Diagram:**



**Program:**

```
void setup()
{ pinMode(5,OUTPUT);
  pinMode(LED_BUILTIN,OUTPUT);
}
void loop(){
  digitalWrite(5, HIGH); //turn the LED on digitalWrite(LED_BUILTIN, HIGH); //turn the LED on
  delay(1000); //wait for a second
  digitalWrite( 5,LOW); //turn the LED off digitalWrite(LED_BUILTIN,LOW); //turn the LED on
  delay(1000); //wait for a second
}
```

**Steps:**

- Connect NodeMCU to computer using USB cable.
- In the menu go to tools, under port select the port through which NodeMCU is connected to computer.
- Connect the negative pin (shorter pin) of LED to GND pin of NodeMCU and positive pin(longer pin) to the GPIO pin specified in the program.
- Compile the program and upload.

**Output:**

LED blinks and in-built LED blinks.

## 2.3 Experiment to make the LED ON and OFF by using Blynk app.

**Aim:** To make an external LED blink using NodeMCU and Blynk app on mobile.

**Description:** NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 WiFi SoC from Espressif, and hardware which is based on the ESP-12 module. The term "NodeMcu" by default refers to the firmware rather than the dev kits. The firmware ESP8266 uses the Lua scripting language. ESP8266 Wifi modules includes a CP2102 TTL to USB chip for programming and debugging, is breadboard-friendly, and can simply be powered via its micro USB port.

**Requirements:** NodeMCU, USB cable, LED, Blynk App (on mobile), Internet connection to mobile. Wi fi access to NodeMCU for internet.

### Program:

```
#include <ESP8266WiFi.h> #include <BlynkSimpleEsp8266.h> char auth[] = "YourAuthToken";
char ssid[] = "YourNetworkName"; // Your WiFi credentials.
char pass[] = "YourPassword"; // Set password to "" for open networks. void setup()
{
  Serial.begin(115200); Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
}
void loop(){ Blynk.run();
}
```

### Steps:

- Connect NodeMCU to computer using USB cable.
- In the menu go to tools, under port select the port through which NodeMCU is connected to computer.
- Download the Blynk library and put tools and libraries folders of Blynk library in your sketchbook folder separately.
- Install Blynk app and create an account by giving email address.
- Open the Blynk app on mobile. Create new project. Add a new button in it. Select Digital and a GPIO pin for the button.
- Connect the negative pin (shorter pin) of LED to GND pin of NodeMCU and positive pin(longer pin) to the GPIO pin specified in the button of Blynk app.
- Compile the program and upload.

### Output:

The LED connected to NodeMCU will be on and off whenever the button in the Blynk app is pushed

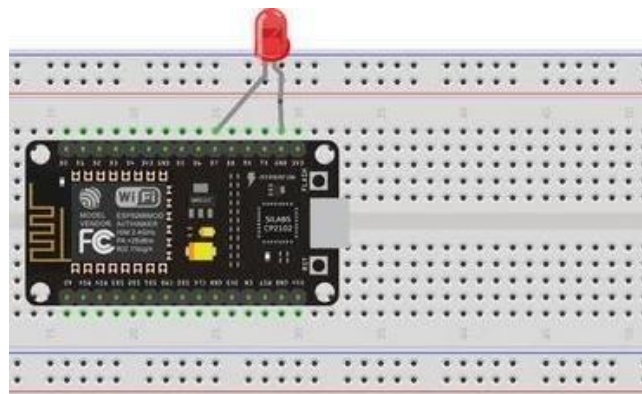
## 2.4 Experiment to make the LED ON and OFF by using the wifi module.

**Aim:** To create a model web page to control LED using NodeMCU connected on a local network.

**Requirements:** ESP8266 Node, MCU Breadboard, LED, Jumper Wires, Arduino IDE.

**Description:** NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 WiFi SoC from Espressif, and hardware which is based on the ESP-12 module. The term "NodeMcu" by default refers to the firmware rather than the dev kits. The firmware ESP8266 uses the Lua scripting language. ESP8266 Wifi modules includes a CP2102 TTL to USB chip for programming and debugging, is breadboard-friendly, and can simply be powered via its micro USB port.

### Circuit Diagram:



### Program:

```
#include <ESP8266WiFi.h>

const char* ssid = "AVATAR";
const char* password = "@1234567#";

int ledPin = 13; // GPIO13---D7 of NodeMCU WiFiServer server(80);

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

pinMode(ledPin, OUTPUT); digitalWrite(ledPin, LOW);
// Connect to WiFi network Serial.println(); Serial.println(); Serial.print("Connecting to ");
Serial.println(ssid); WiFi.begin(ssid, password);

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

```

// Start the server server.begin(); Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL to connect: "); Serial.print("http://"); Serial.print(WiFi.localIP());
Serial.println("/");

}

void loop() {
// Check if a client has connected WiFiClient client = server.available(); if (!client) {
return;
}

// Wait until the client sends some data Serial.println("new client"); while(!client.available()){
delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r'); Serial.println(request);
client.flush();
// Match the request int value = LOW;
if (request.indexOf("/LED=ON") != -1) { digitalWrite(ledPin, HIGH);
value = HIGH;
}
if (request.indexOf("/LED=OFF") != -1) { digitalWrite(ledPin, LOW);
value = LOW;
}

// Set ledPin according to the request
//digitalWrite(ledPin, value);

// Return the response client.println("HTTP/1.1 200 OK"); client.println("Content-Type:
text/html"); client.println(""); // do not forget this one client.println("<!DOCTYPE HTML>");
client.println("<html>");

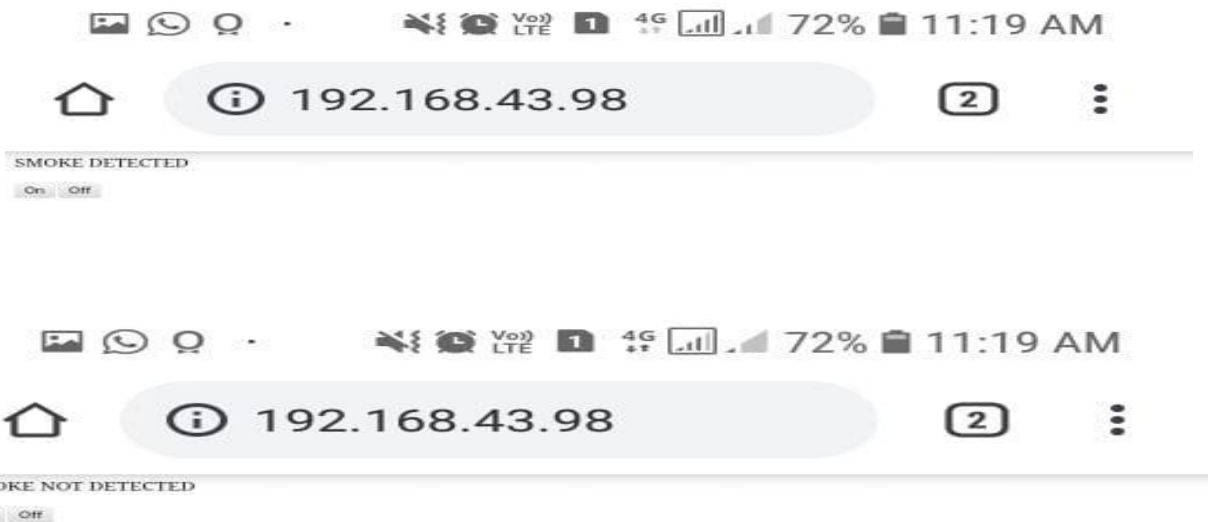
client.print("Led is now: ");

if(value == HIGH) { client.print("On");
} else { client.print("Off");
}
client.println("<br><br>");
client.println("<a href='\"/LED=ON\"'><button>On </button></a>"); client.println("<a
href='\"/LED=OFF\"'><button>Off </button></a><br />"); client.println("</html>");

```



**Output:**



## **Experiment-3 Experimenting Rain sensing automatic wiper system.**

### **3.1 Experiment Rain sensing with simple Blink LED ON and OFF.**

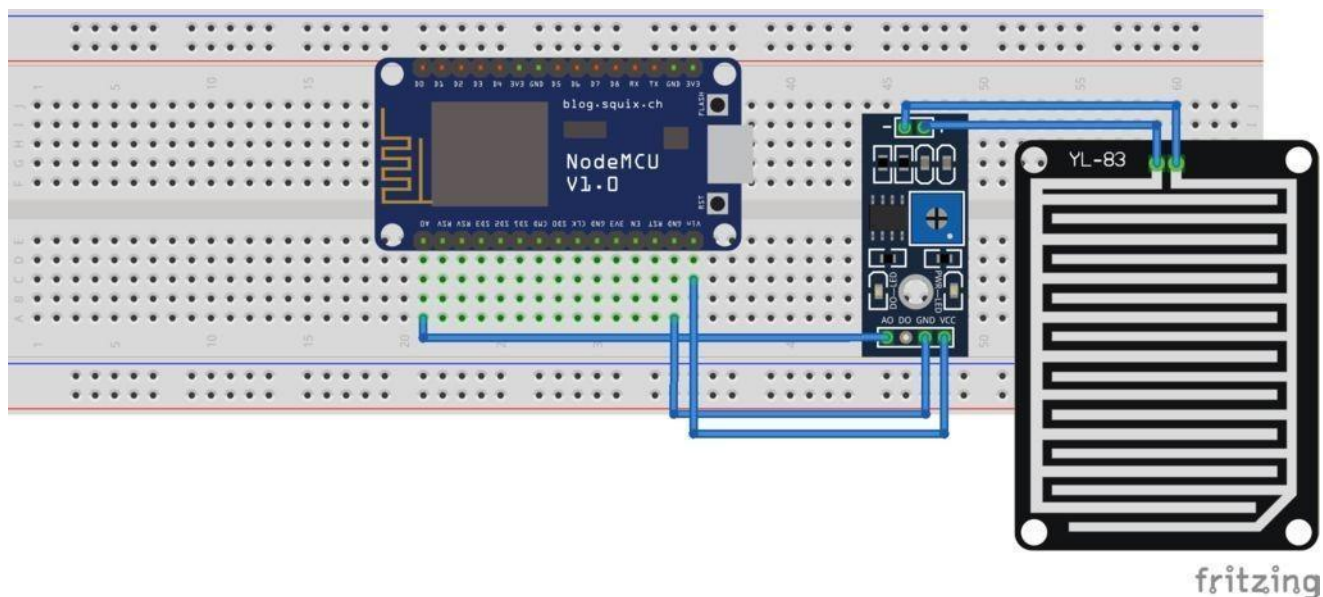
**Aim:** To demonstrate rain sensing automatic wiper system using NodeMCU.

**Description:** Water level sensor is basically a board on which nickel is coated in the form of lines. It works on the principle of resistance. When there is no rain drop on board. Resistance is high so we gets high voltage according to  $V=IR$ . When raindrop present it reduces the resistance because water is conductor of electricity and presence of water connects nickel lines in parallel so reduced resistance and reduced voltage drop across it.

The analog output is used in detection of drops in the amount of rainfall. Connected to 3,3V power supply, the LED will turn Off when induction board has no rain drop, and output is Low. When dropping a little amount water, output is High, the switch indicator will turn on. Brush off the water droplets, and when restored to the initial state, outputs high level. When no rain digital output is 1 and analog output gives 1023 max value. When rain is present digital output is 0 and analogue output is much less than 1023.

**Requirements:** NODE MCU, Water Level Sensor, USB Cable, LED, Jumper Cables

**Circuit Diagram:**



**Program:**

```
int sensorPin = A0; int enable2 = 13;
int sensorValue2 = 0; // variable to store the value coming from sensor Rain sensor

void setup() {

  OUTPUT: pinMode(enable2,OUTPUT); Serial.begin(9600);
  delay(10);

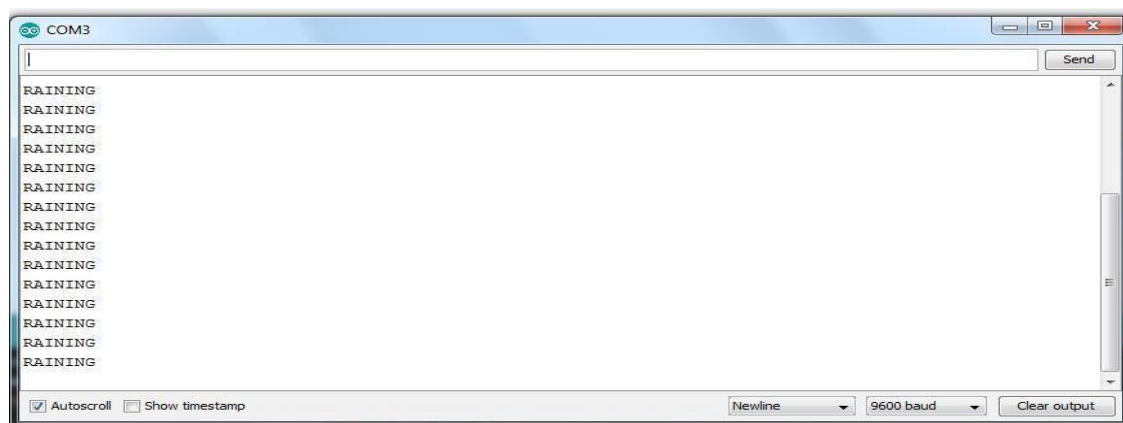
}

void loop() {

  delay(500);
  sensorValue2 = analogRead(sensorPin); sensorValue2 = constrain(sensorValue2, 150, 440);
                                     sensorValue2 =
  map(sensorValue2, 150, 440, 1023, 0);

  if (sensorValue2 >= 20)
  {
    Serial.print("raining"); digitalWrite(enable2, HIGH);
    Serial.print("Not Raining"); digitalWrite(enable2, LOW);
  }

  Serial.println(); delay(100);
}
```

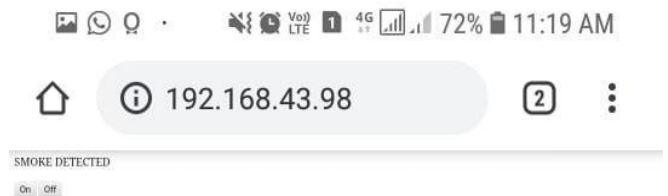
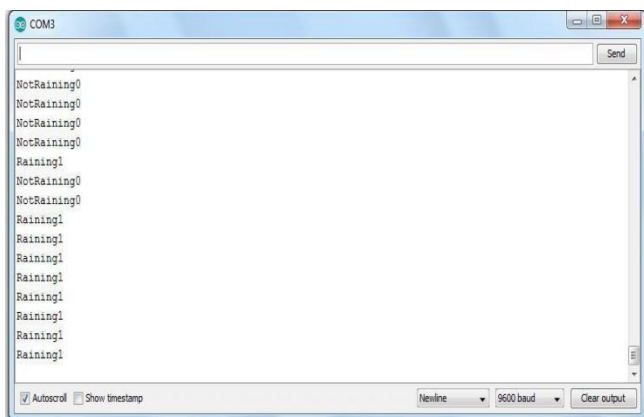
**OUTPUT:**

### 3.2 Experiment Rain sensing by using interrupts notify the state of weather.

#### Program: [USING INTERRUPTS]

```
const byte interrupt Pin = 13;
volatile boolean checkInterrupt = false;
int number Of Interrupts = 0;
unsigned long debounceTime = 1000; unsigned long last Debounce=0;
void setup()
{
  Serial.begin(115200); pinMode(interrupt Pin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interrupt Pin), handle Interrupt, FALLING);
}
void handle Interrupt()
{
  check Interrupt= true;
}
void loop()
{
  if(check Interrupt == true && ( millis() - last Debounce) > debounceTime ))
  {
    last Debounce = millis(); check Interrupt = false; number Of Interrupts ++; Serial.print("Rain
    detected "); Serial.println(number Of Interrupts);
  }
  else checkInterrupt = false;
}
```

#### Output:



## **Experiment-4 Study of various sensors that are used in IOT.**

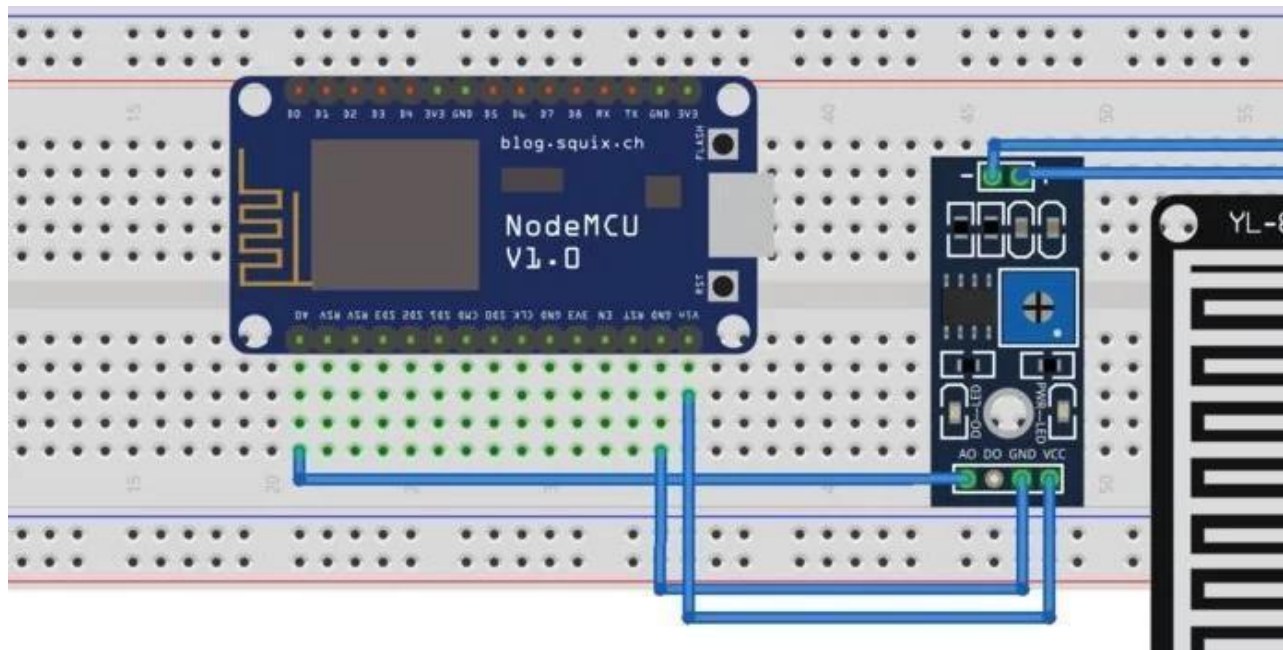
### **4.1 Experiment to make the interface of Rain Sensing:**

**Aim:** To create an interface of Water Level sensor with NodeMcu.

**Requirements:** ESP8266 Node, MCU Breadboard, Jumper Wires, Arduino IDE, Rain Sensor.

**Description:** Water Level sensor is basically a board on which nickel is coated in the form of lines. It works on the principle of resistance. When there is no rain drop on board. Resistance is high so we gets high voltage according to  $V=IR$ . When raindrop present it reduces the resistance because water is conductor of electricity and presence of water connects nickel lines in parallel so reduced resistance and reduced voltage drop across it.

#### **Circuit Diagram:**



#### **Program:**

```
#include <ESP8266WiFi.h> const char* ssid = "AVATAR";
const char* password = "@1234567#"; WiFiClient client;
int sensorPin = A0;    // input for LDR and rain sensor int enable2 = 13;    // enable reading
Rain sensor
int sensorValue2 = 0; // variable to store the value coming from sensor Rain sensor
WiFiServer server(80); void setup() {
pinMode(enable2, OUTPUT);
Serial.begin(115200); delay(10);
WiFi.begin(ssid, password); Serial.println(); Serial.println();
```

```

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

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

// Start the server server.begin(); Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL to connect: "); Serial.print("http://"); Serial.print(WiFi.localIP());
Serial.println("/");

}
void loop() {
// Check if a client has connected WiFiClient client = server.available(); if (!client) {
return;
}

// Wait until the client sends some data Serial.println("new client"); while(!client.available()){
delay(1);
}

// Return the response client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html"); client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>"); client.println("<html>");
client.println("<br><br>"); delay(500);
sensorValue2 = analogRead(sensorPin); sensorValue2 = constrain(sensorValue2, 150, 440);
sensorValue2 = map(sensorValue2, 150, 440, 1023, 0); if (sensorValue2<=20)
{
Serial.print("rain is detected");

client.print("rain is detected");

digitalWrite(enable2, HIGH);
}
else

{
Serial.print("rain not detected"); client.print("rain not detected"); digitalWrite(enable2, LOW);
}
//Serial.print("Rain value:   ");
//Serial.println(sensorValue2);
Serial.println();

```



```

delay(100); client.println("</html>");

delay(1);
Serial.println("Client disconnected"); Serial.println("");

}

```

## Output:

The screenshot shows the Arduino IDE interface. The sketch editor on the left contains the following code:

```

// webrain
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.println("<br><br>");
delay(500);
sensorValue2 = analogRead(sensorPin);
sensorValue2 = constrain(sensorValue2, 150, 440);
sensorValue2 = map(sensorValue2, 150, 440, 1023, 0);
if (sensorValue2 <= 20)
{
  Serial.print("rain is detected");
  client.print("rain is detected");
  digitalWrite(enable2, HIGH);
}
else
{
  Serial.print("rain not detected");
  client.print("rain not detected");
  digitalWrite(enable2, LOW);
}
//Serial.print("Rain value: ");
//Serial.println(sensorValue2);
Serial.println();

```

The serial monitor on the right shows the output of the sketch:

```

rain not detected
Client disconnected

new client
rain not detected
Client disconnected

new client
rain not detected
Client disconnected

new client
rain not detected
Client disconnected

new client
rain not detected
Client disconnected

new client
rain not detected
Client disconnected

new client
rain not detected
Client disconnected

```

The status bar at the bottom indicates: NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/ttyUSB0.

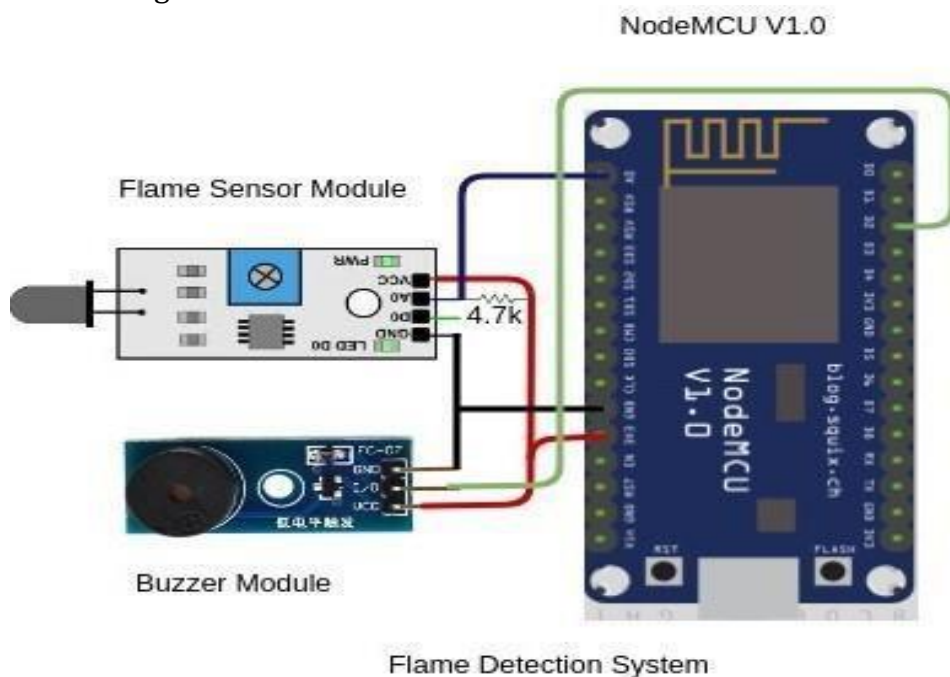
## 4.2 Experiment to make interface of Flame Sensing.

**Aim:** To create an interface for flame sensor to detect Flame using NodeMcu.

**Requirements:** ESP8266 Node, MCU Breadboard, Jumper Wires, Arduino IDE, Rain Sensor, Buzzer.

**Description:** A flame sensor detects the presence of fire or flames. In extremely hazardous environments, flame sensors work to minimize the risks associated with fire. There are several different types of flame sensor – some will raise an alarm while others may activate a fire suppression system or deactivate a combustible fuel line. Among the many different types of flame sensor, ultraviolet flame sensors, near IR array flame sensors, infrared flame sensors, and IR3 flame detection sensors are the most prominent.

Circuit Diagram:



**Program:**

```
#include<SoftwareSerial.h>
int sensorPin = A0; // select the input pin for the LDR
int sensorValue = 0; // variable to store the value coming from the sensor
int led = 4; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer
void setup() {
  // declare the ledPin and buzzer as an OUTPUT: pinMode(led, OUTPUT); pinMode(buzzer,OUTPUT);
  Serial.begin(9600);
}

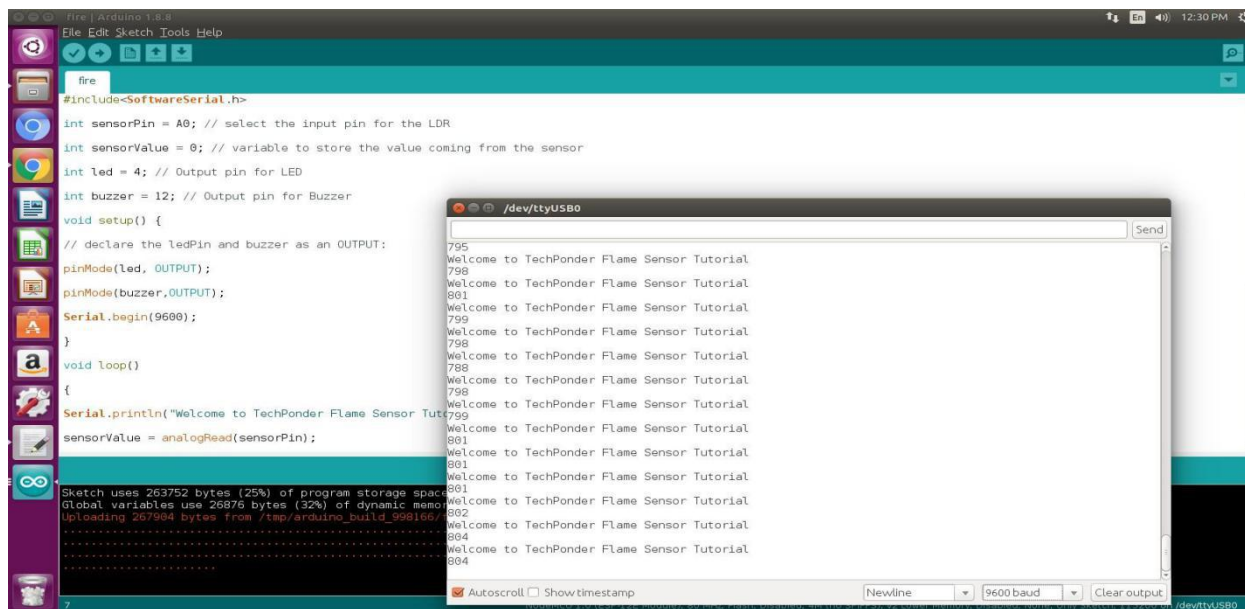
void loop()
{
```

```

sensorValue = analogRead(sensorPin); Serial.println(sensorValue);
if (sensorValue > 500)
{
  Serial.println("Fire Detected"); Serial.println("LED on"); digitalWrite(led,HIGH);
  digitalWrite(buzzer,HIGH); delay(1000);
}
digitalWrite(led,LOW); digitalWrite(buzzer,LOW); delay(sensorValue);
}

```

## Output:



The screenshot shows the Arduino IDE interface. The code editor on the left contains the following code:

```

#include<SoftwareSerial.h>

int sensorPin = A0; // select the input pin for the LDR
int sensorValue = 0; // variable to store the value coming from the sensor

int led = 4; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer

void setup() {
  // declare the ledPin and buzzer as an OUTPUT:
  pinMode(led, OUTPUT);
  pinMode(buzzer,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Welcome to TechPonder Flame Sensor Tutorial");
  sensorValue = analogRead(sensorPin);

```

The serial monitor on the right, titled "/dev/ttyUSB0", shows the output of the program. It displays the message "Welcome to TechPonder Flame Sensor Tutorial" repeatedly, with line numbers 795 through 804 visible on the left side of the monitor. The monitor also includes a "Send" button, a "Newline" dropdown menu, a "9600 baud" rate selector, and a "Clear output" button.

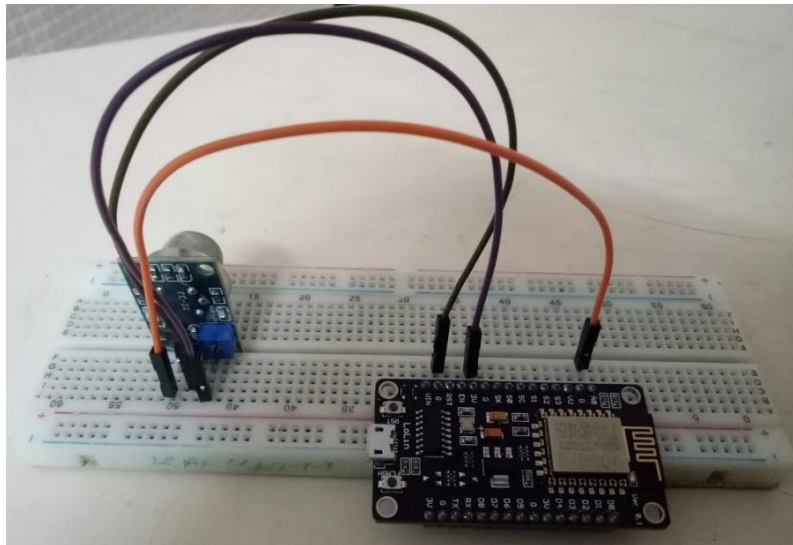
### 4.3 Experiment to make a interface for Smoke Detection:

**Aim:** In order to create an interface for smoke detection using NodeMcu

**Requirements:** ESP8266 Node, MCU Breadboard, Jumper Wires, Arduino IDE, Smoke Sensor, buzzer.

**Description:** Based on the result, the fire warning detector will send notification to user through ESP8266 when temperature detected is greater than room temperature OR MQ2 detected smoke or combustion gases. The buzzer will sound if both sensors is triggered. It also can send signal when low battery, and control by using smartphone's apps for testing buzzer.

**Circuit Diagram:**



**Program:**

```
#include <ESP8266WiFi.h>
const char* ssid = "AVATAR";
const char* password = "@1234567#"; int buzzer_pin = 13;
int smoke_sensor_pin = A0; WiFiServer server(80); void setup()

{

Serial.begin(115200);
// Set the baudrate according to your esp8266 pinMode(buzzer_pin, OUTPUT);
pinMode(smoke_sensor_pin, INPUT);
// Connect to WiFi network Serial.println(); Serial.println(); Serial.print("Connecting to ");
```

```

Serial.println(ssid);

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}
Serial.println(""); Serial.println("WiFi connected");

// Start the server server.begin(); Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL to connect: "); Serial.print("http://"); Serial.print(WiFi.localIP());
Serial.println("/");

}
void loop()
{
// Check if a client has connected WiFiClient client = server.available(); if (!client) {
return;
}
// Wait until the client sends some data Serial.println("new client"); while(!client.available()){
delay(1);
}
// Read the first line of the request
String request = client.readStringUntil('\r'); Serial.println(request);
client.flush();
// Return the response client.println("HTTP/1.1 200 OK"); client.println("Content-Type:
text/html"); client.println(""); // do not forget this one client.println("<!DOCTYPE HTML>");
client.println("<html>");

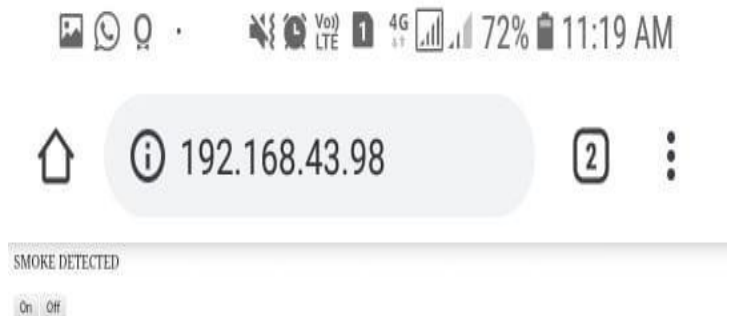
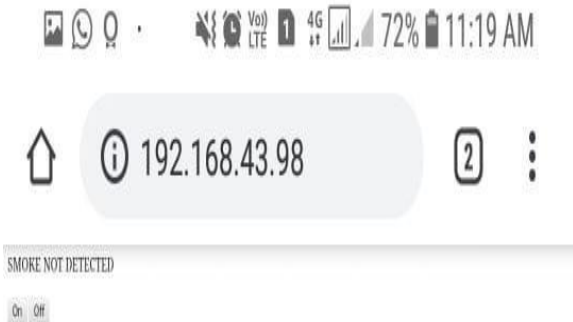
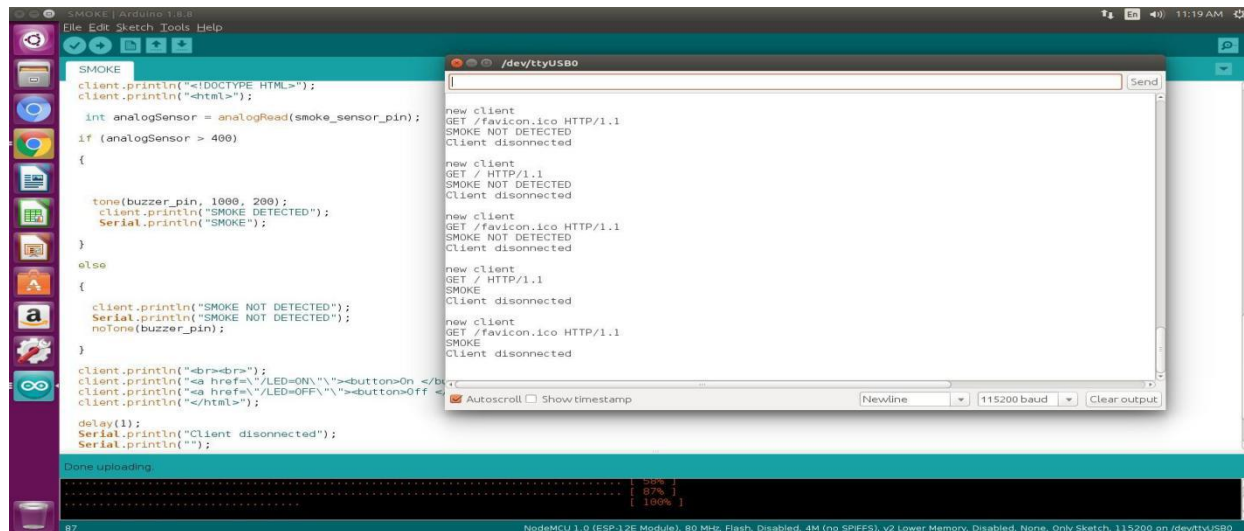
int analogSensor = analogRead(smoke_sensor_pin); if (analogSensor > 400)
{
tone(buzzer_pin, 1000, 200); client.println("SMOKE DETECTED"); Serial.println("SMOKE");

}
Else
{
client.println("SMOKE NOT DETECTED"); Serial.println("SMOKE NOT DETECTED");
noTone(buzzer_pin);
}
client.println("<br><br>");
client.println("<a href=\\\"/LED=ON\\\"><button>On </button></a>"); client.println("<a
href=\\\"/LED=OFF\\\"><button> client.println("</html>");
delay(1);

```

```
Serial.println("Client disonnected"); Serial.println("");
}
```

## Output:





#### 4.4 Experiment to make the interface for Ultrasonic sensor.

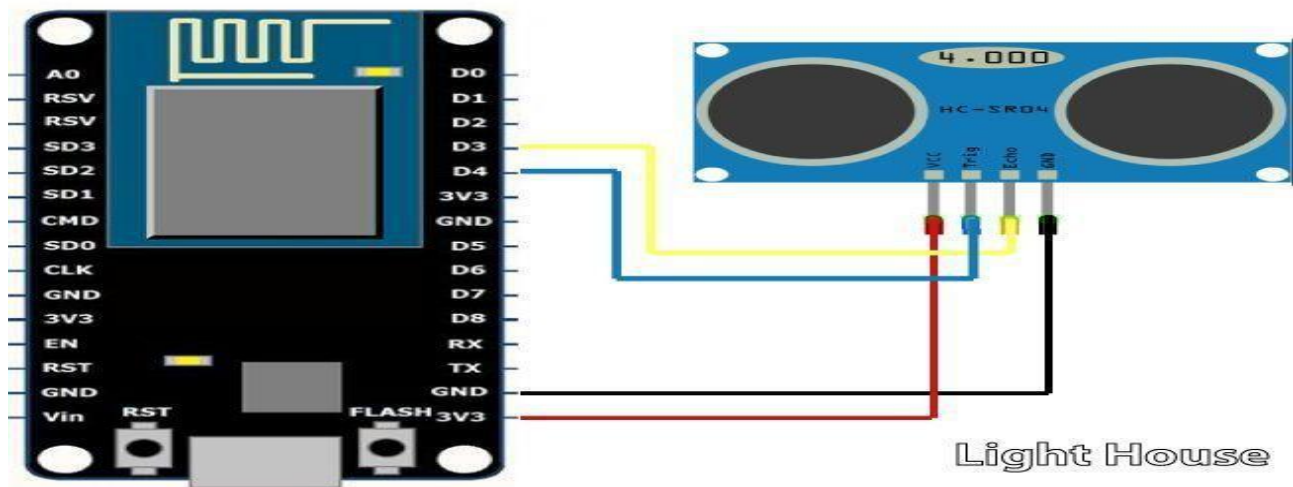
**Aim:** To create an interface for ultrasonic sensor to detect Distance using NodeMcu.

**Requirements:**

- ESP8266
- NodeMCU
- Breadboard
- Jumper Wires
- Arduino IDE
- HC-SR04, (Ultrasonic-Sensor)

**Description:** An Ultrasonic sensor is a device that can measure the distance of an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and waits for that sound wave to bounce back. By recording the time taken between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sensor and the object.

**Circuit Diagram:**



**Program:**

```
// defines pins numbers const int trigPin = 2; //D4 const int echoPin = 0; //D3
// defines variables long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output pinMode(echoPin, INPUT); // Sets the
  echoPin as an Input Serial.begin(9600); // Starts the serial communication
}

void loop() {
  // Clears the trigPin 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);

  // Calculating the distance distance= duration*0.034/2;
  // Prints the distance on the Serial Monitor Serial.print("Distance: "); Serial.println(distance);
  delay(2000);
}
```

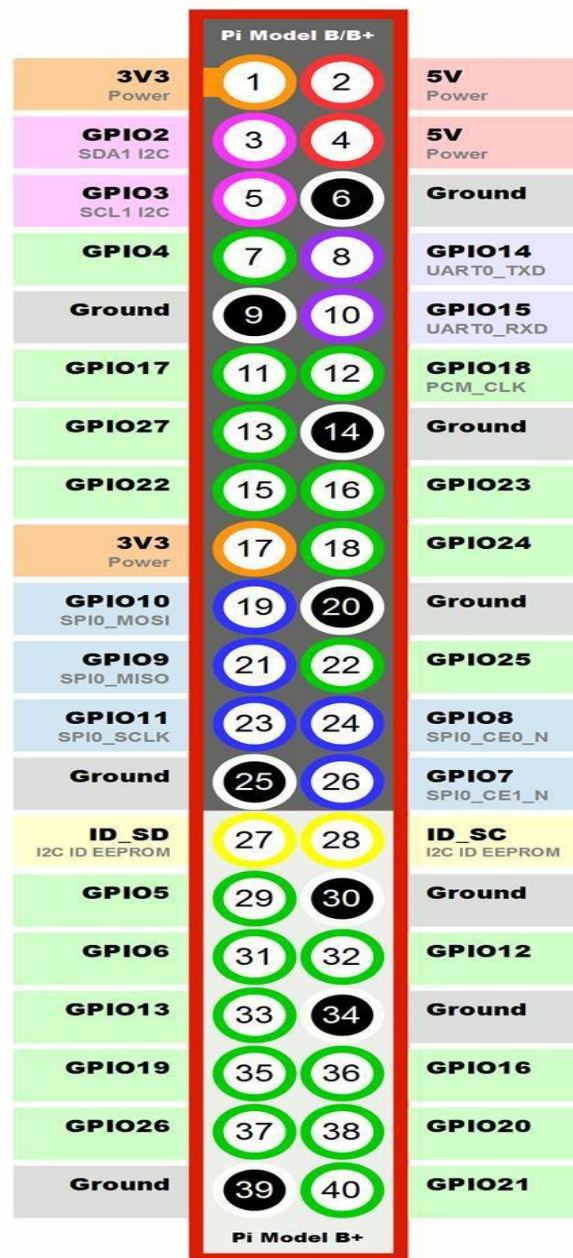
**Output:**

```
Distance: 12
Distance: 67
Distance: 2369
Distance: 397
Distance: 3
Distance: 8
Distance: 74
Distance: 2367
Distance: 2367
Distance: 2366
Distance: 2366
Distance: 8
Distance: 7
Distance: 6
Distance: 4
Distance: 760
Distance: 2367
Distance: 7
Distance: 2366
Distance: 32
```

## Raspberry pi -Experiments

### Experiment-5 Raspberry pi pin diagram and installation procedure.

#### 5.1 Raspberry pi pin diagram and its description.



www.raspberrypi-spy.co.uk

<b>Pin</b>	<b>Particle</b>	<b>Description</b>
GPIO0	ID_SD	I2C data line used to identify Pi Hats (RESERVED FOR SYSTEM)
GPIO1	ID_SC	I2C clock line used to identify Pi Hats (RESERVED FOR SYSTEM)
GPIO2	SDA	I2C data line [2]
GPIO3	SCL	I2C clock line [2]
GPIO4	D0	Digital IO
GPIO5	D4	Digital IO
GPIO6	D5	Digital IO
GPIO7	CE1	SPI chip enable 1, digital IO
GPIO8	CE0	SPI chip enable 0, digital IO
GPIO9	MISO	SPI master-in slave-out [3]
GPIO10	MOSI	SPI master-out slave-in [3]
GPIO11	SCK	SPI clock [3]
GPIO12	D13/A4	Digital IO
GPIO13	D6	PWM-capable digital IO
GPIO14	TX	UART hardware serial transmit [1]
GPIO15	RX	UART hardware serial receive [1]
GPIO16	D14/A5	PWM-capable digital IO
GPIO17	D1	Digital IO
GPIO18	D9/A0	PWM-capable digital IO
GPIO19	D7	PWM-capable digital IO
GPIO20	D15/A6	Digital IO
GPIO21	D16/A7	Digital IO
GPIO22	D3	Digital IO
GPIO23	D10/A1	Digital IO
GPIO24	D11/A2	Digital IO
GPIO25	D12/A3	Digital IO
GPIO26	D8	Digital IO
GPIO27	D2	Digital IO

## 1.2 Raspberry pi installation procedure

- Download raspbian from the below site: <https://www.raspberrypi.org/downloads/>
- Unzip the downloaded folder. You will find an image (.img file) in it.
- We need to write the image file to an SD card using Etcher software. This SD card will be plugged into the raspberry pi module.
- So first, to install the image file onto the SD card plug the SD card into a card reader and format it.
- Download Etcher software from the below site and install it in on your PC  
<https://www.balena.io/etcher/>
- the OS image file that was downloaded and unzipped earlier.
- Add select Flash.
- This will install the Raspbian OS onto the SD card.
- Plug the SD card into the raspberry pi module, connect the HDMI cable of monitor to pi module and connect the power cable.

### 1.3 Experiment to make Sample led program execution on raspberry pi using python.

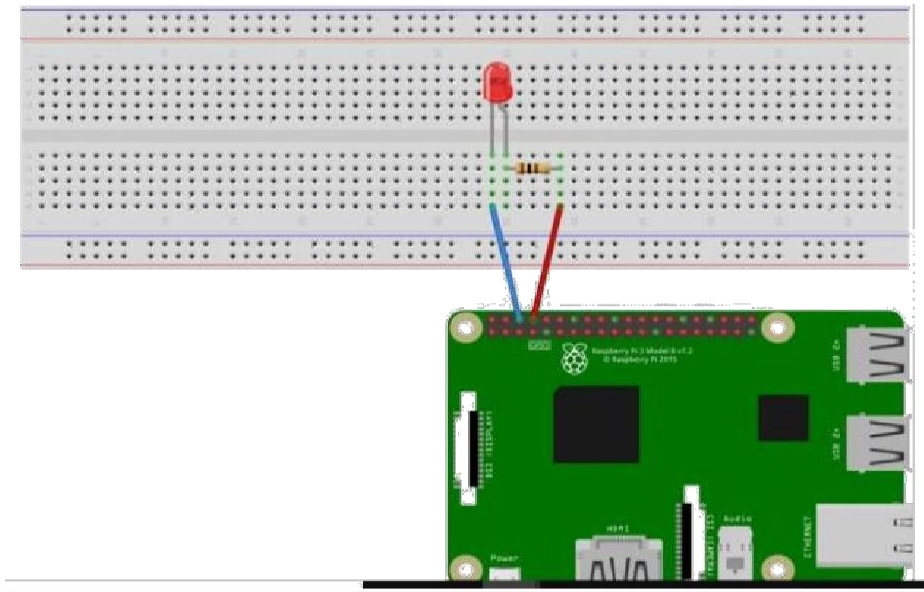
**Aim:** Make an LED blink using the Raspberry Pi.

**Description:** On/OFF an LED using Raspberry Pi module and Python Programming

**Hardware Requirements:** Raspberry Pi 3, B+ module LED, Resistor Breadboard

**Software Requirements:** Raspbian OS Python

**Circuit diagram:**



**Program:**

```
import RPi.GPIO as GPIO
# Import Raspberry Pi GPIO library from time import sleep
# Import the sleep function from the time module GPIO.setwarnings(False)
# Ignore warning for now GPIO.setmode(GPIO.BOARD)
# Use physical pin numbering

GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)
# Set pin 8 to be an output pin and set initial value to low (off)

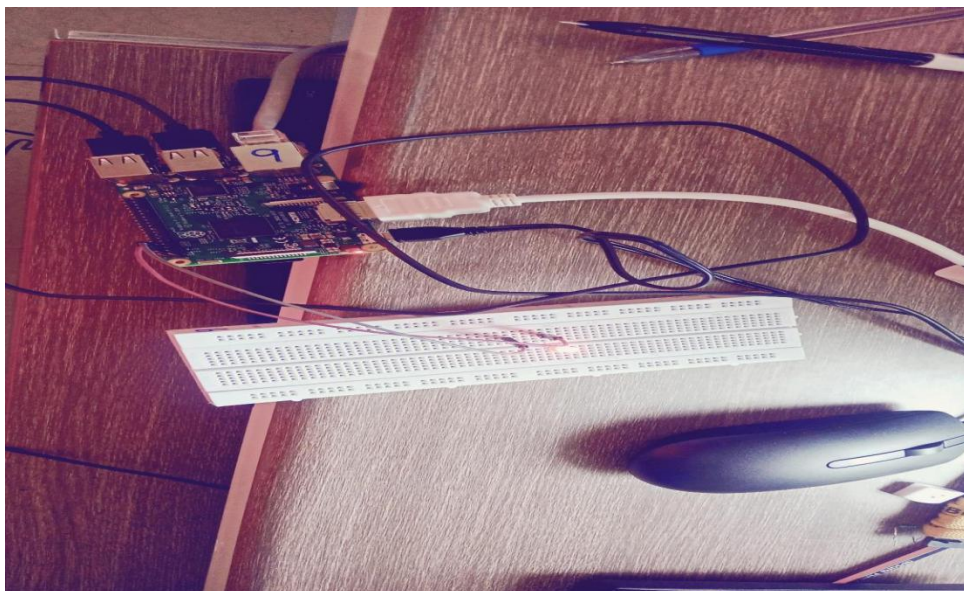
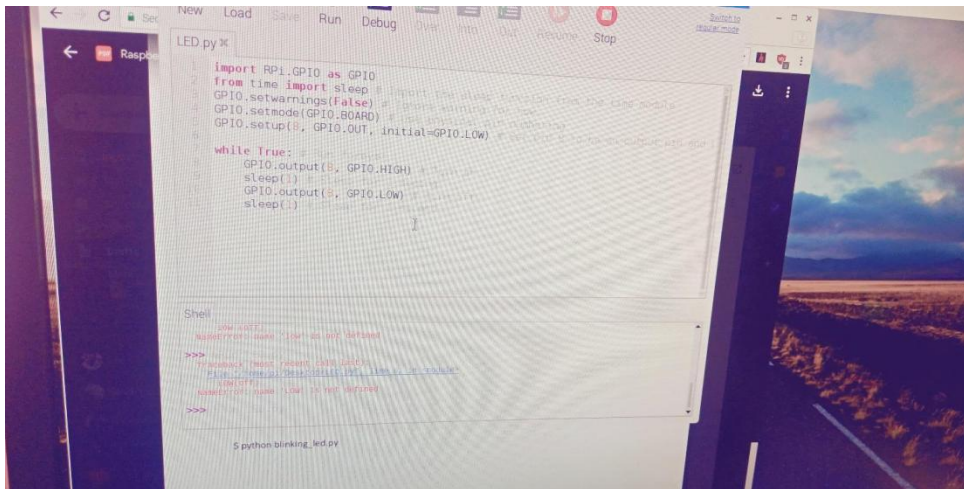
while True:
    # Run forever GPIO.output(8, GPIO.HIGH)
    # Turn on sleep(1)
```



```
# Sleep for 1 second GPIO.output(8, GPIO.LOW)
# Turn off sleep(1)
# Sleep for 1 second
```

## Output:

python blinking\_led.py



## Experiment-6 Study of Various Sensors using Raspberry pi.

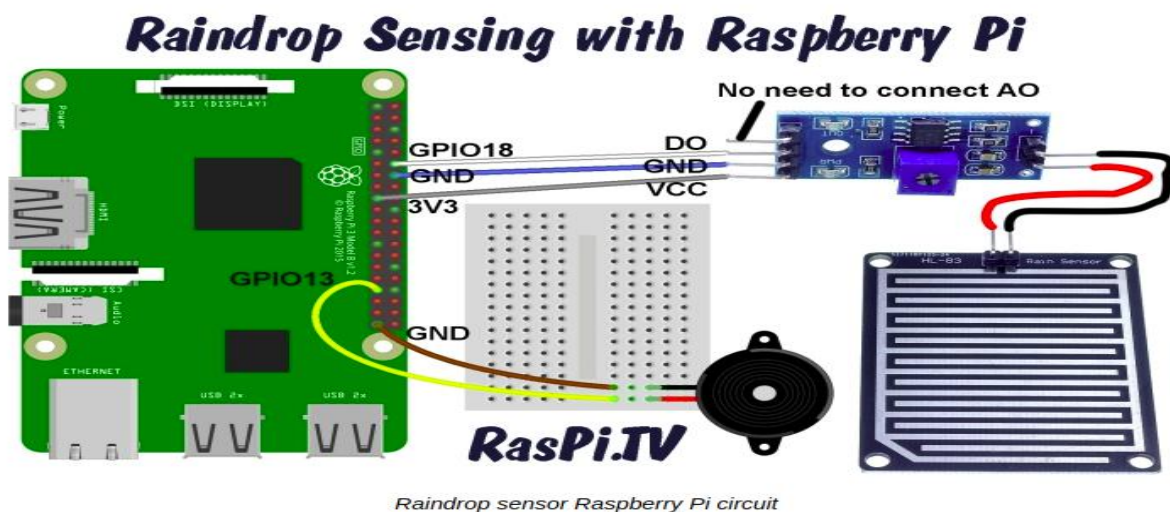
### 6.1 Experiment to detect the water level using the water level sensor.

**Aim:** To detect rain with water level sensor using Raspberry Pi.

**Requirements:** Raspberry pi, Water Level Sensor, Buzzer, Jumper Cables

**Description:** We're going to use a simple water level sensor, a buzzer and a Pi to alert you when it rains. To get an alert when it starts to rain, we're going to rig this sensor up to a Raspberry Pi and trigger a buzzer. You could also have it blink LEDs, send you an SMS or tweet a photo of the washing line and the sky. But we're going with a buzzer for now.

**Circuit Diagram:**



**Program:**

```
# raindrop sensor DO connected to GPIO18
# HIGH = no rain, LOW = rain detected
# Buzzer on GPIO13
#import RPi.GPIO as GPIO
#import time
from time import sleep
from gpiozero import Buzzer, InputDevice

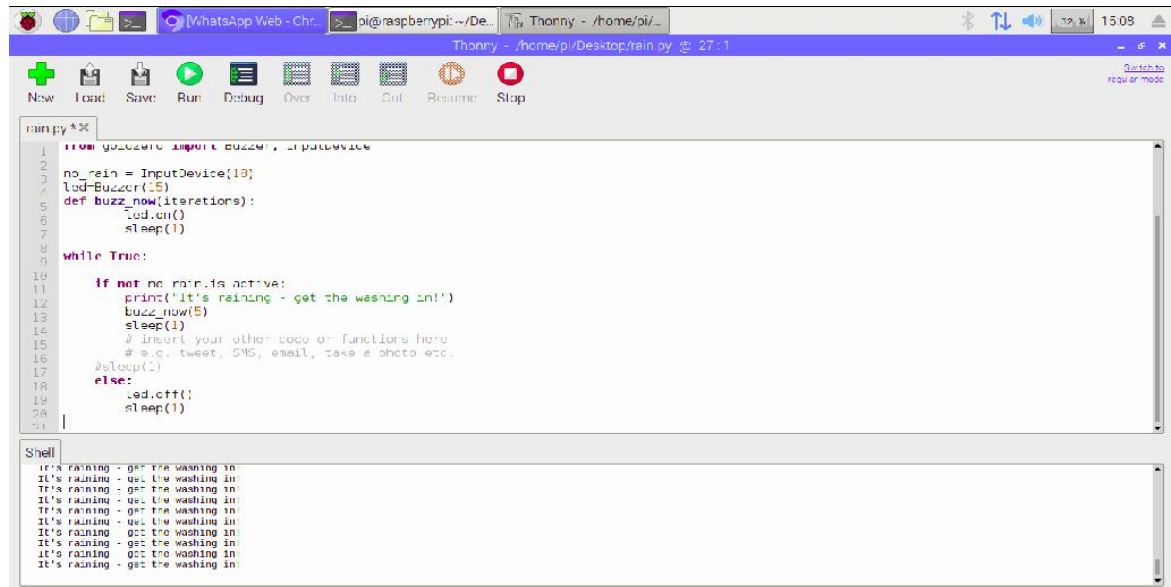
no_rain = InputDevice(18)
led=Buzzer(15)
```

```

def buzz_now(iterations):
    led.on()
    sleep(1)
while True:
    if not no_rain.is_active:
        print("It's raining - get the washing in!")
        buzz_now(5)
        sleep(1)
        # insert your other code or functions here
        # e.g. tweet, SMS, email, take a photo etc.
    #sleep(1)
else:
    led.off()
    sleep(1)

```

## Output:



```

Thonny - /home/pi/Desktop/rain.py @ 27:1
New Load Save Run Debug Over Info Out Resume Stop
rain.py ***
1 from gpiozero import Buzzer, InputDevice
2 no_rain = InputDevice(18)
3 led = Buzz(25)
4
5 def buzz_now(iterations):
6     led.on()
7     sleep(1)
8
9 while True:
10
11     if not no_rain.is_active:
12         print('It's raining - get the washing in!')
13         buzz_now(5)
14         sleep(1)
15         # insert your other code or functions here
16         # e.g. tweet, SMS, email, take a photo etc.
17     #sleep(1)
18 else:
19     led.off()
20     sleep(1)
21
Shell
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!
It's raining - get the washing in!

```

## 2. Experiment to detect the Accident alert system by using:

### 2.1 Experiment to detect the fire detection by using flame detector and make the LED ON and OFF.

**Aim:** Here in we are going to see how simple it is to come up with your own Fire detecting device using flame sensor and Raspberry pi . We can use a buzzer or turn on water sprinklers or even send an automatic email to fire department. I'm going to use an LED which blinks on fire detection.

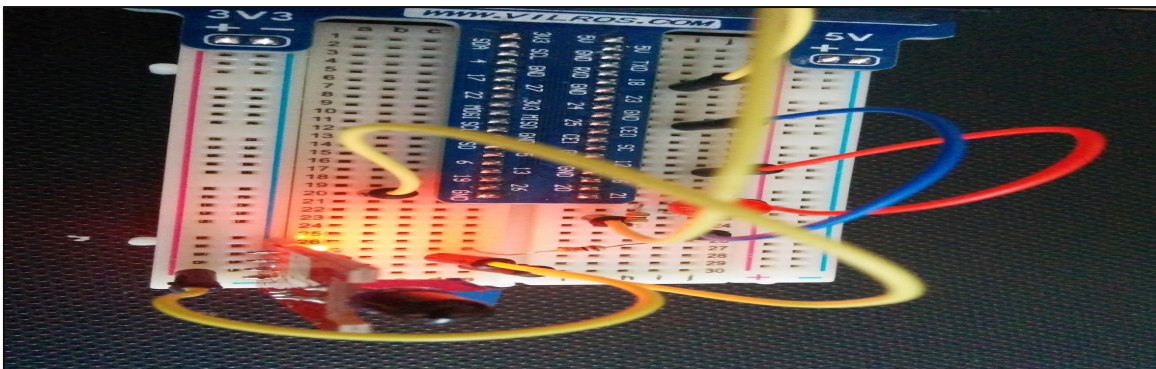
**Requirements :**Raspberry Pi,Flame detector,LED , 330 Ohm resistor

**Description:**Fire alarm system is a real-time monitoring system that detects the presence of smoke in the air The key feature of the system is the ability to remotely send an alert when a fire is detected. When the presence of smoke is detected,The advantage of using this system is it will reduce the possibility of false alert reported to the Firefighter.

#### Flame Sensor Interface:

- VCC :- 3.3V-5V voltage
- GND :- GND
- DO :- board digital output interface (0 and 1)
- AO :- board analog output interface

#### Circuit/ Connection Images:



#### Connection Steps:

- 1) Connect Ground on sensor to ground on Raspberry Pi.
- 2) Supply + 3.3V to power( + symbol on sensor).
- 3) Connect Digital out pin of sensor to GPIO input using a resistor (256 0).
- 4) Connect GPIO out to LED .

**Program:**

```
from time import sleep
import RPi.GPIO as GPIO

#GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
LedOut = 18
Flamein = 25

GPIO.setup(Flamein, GPIO.IN)

GPIO.setup(LedOut, GPIO.OUT)
count = 0

while True:
    try:
        if (GPIO.input(25) == True):
            print 'fire in your house'
            GPIO.output(LedOut, 1)
            sleep(0.1)
            GPIO.output(LedOut, 0)
            sleep(0.1)
            GPIO.output(LedOut, 1)
        else:
            print 'you are safe'
            GPIO.output(LedOut, 0)
    except KeyboardInterrupt:
        exit()
GPIO.cleanup()
```

## 2.2 Experiment to detect the distance where the object is moving by using the ultrasonic sensor.

**Aim:** To find the distance of an object using ultrasonic sensor with Raspberry Pi.

**Requirements:** Raspberry pi, ultrasonic sensor, jumper wires.

**Description:** Ultrasonic distance sensors are designed to measure distance between the source and the target using ultrasonic waves. We use ultrasonic waves because they are relatively accurate across short distances and don't cause disturbances as they are inaudible to human ear.

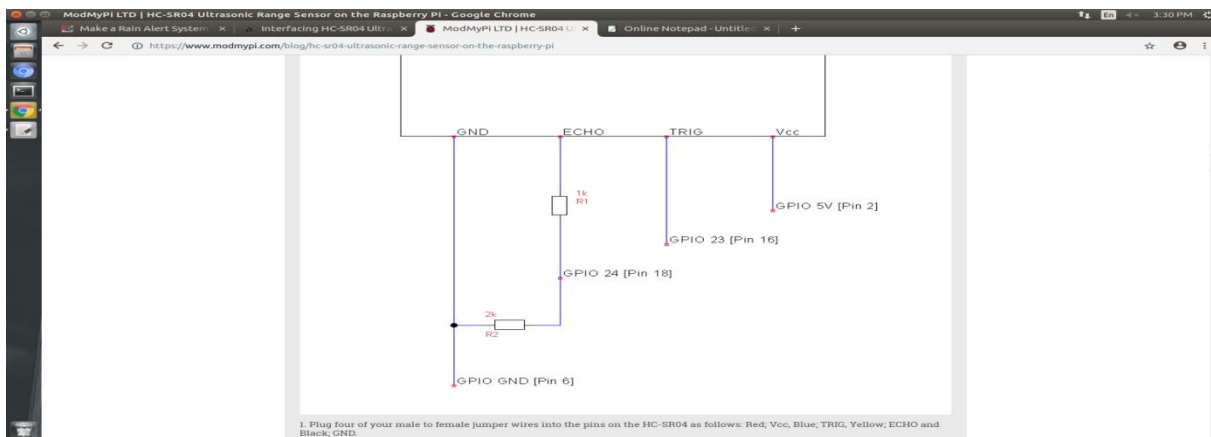
Time taken by pulse is actually for to and fro travel of ultrasonic signals, while we need only half of this. Therefore Time is taken as Time/2.

Distance = Speed \* Time/2

Speed of sound at sea level = 343 m/s or 34300 cm/s

Thus, Distance = 17150 \* Time (unit cm)

### Circuit Diagram:



### Program:

```
import RPi.GPIO as GPIO          #Import GPIO library
import time                      #Import time library
GPIO.setmode(GPIO.BCM)          #Set GPIO pin numbering
TRIG = 23                       #Associate pin 23 to TRIG
ECHO = 24                       #Associate pin 24 to ECHO
print ("Distance measurement in progress")
GPIO.setup(TRIG,GPIO.OUT)        #Set pin as GPIO out
GPIO.setup(ECHO,GPIO.IN)         #Set pin as GPIO in
while True:
    GPIO.output(TRIG, False)     #Set TRIG as LOW
    print ("Waiting For Sensor To Settle")
    time.sleep(2)                #Delay of 2 seconds
```



```

GPIO.output(TRIG, True)          #Set TRIG as HIGH
time.sleep(0.00001)              #Delay of 0.00001 seconds
GPIO.output(TRIG, False)         #Set TRIG as LOW
while GPIO.input(ECHO)==0:       #Check whether the ECHO is LOW
    pulse_start = time.time()    #Saves the last known time of LOW pulse
while GPIO.input(ECHO)==1:       #Check whether the ECHO is HIGH
    pulse_end = time.time()      #Saves the last known time of HIGH pulse
pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable
distance = pulse_duration * 17150 #Multiply pulse duration by 17150 to get distance
distance = round(distance, 2)    #Round to two decimal points
print(distance)
if distance >=5 and distance<1000: #Check whether the distance is within range
    print ("Distance:",distance - 0.5,"cm") #Print distance with 0.5 cm calibration
elif distance >=1000 and distance<2000:
    print ("Accident")

```

## OUTPUT:

The screenshot shows the Thonny IDE interface. The top toolbar includes buttons for New, Load, Save, Run, Debug, Over, Info, Quit, Resume, and Stop. The main editor window displays the Python code for the ultrasonic sensor. The shell window at the bottom shows the execution output, including 'Waiting For Sensor To Settle', '2317.87', 'Accident', 'Waiting For Sensor To Settle', '36.58', 'Distance: 36.58 cm', 'Waiting For Sensor To Settle', '47.31', 'Distance: 46.81 cm', 'Waiting For Sensor To Settle', and '55.89'.

```

17 time.sleep(2) #Delay of 2 seconds
18
19 GPIO.output(TRIG, True) #Set TRIG as HIGH
20 time.sleep(0.00001) #Delay of 0.00001 seconds
21 GPIO.output(TRIG, False) #Set TRIG as LOW
22
23 while GPIO.input(ECHO)==0: #Check whether the ECHO is LOW
24     pulse_start = time.time() #Saves the last known time of LOW pulse
25
26 while GPIO.input(ECHO)==1: #Check whether the ECHO is HIGH
27     pulse_end = time.time() #Saves the last known time of HIGH pulse
28
29 pulse_duration = pulse_end - pulse_start #Get pulse duration to a variable
30
31 distance = pulse_duration * 17150 #Multiply pulse duration by 17150 to get distance
32 distance = round(distance, 2) #Round to two decimal points
33 print(distance)
34 if distance >=5 and distance<1000: #Check whether the distance is within range
35     print ("Distance:",distance - 0.5,"cm") #Print distance with 0.5 cm calibration
36 elif distance >=1000 and distance<2000:

```

```

Shell
Waiting For Sensor To Settle
2317.87
Accident
Waiting For Sensor To Settle
36.58
Distance: 36.58 cm
Waiting For Sensor To Settle
47.31
Distance: 46.81 cm
Waiting For Sensor To Settle
55.89

```

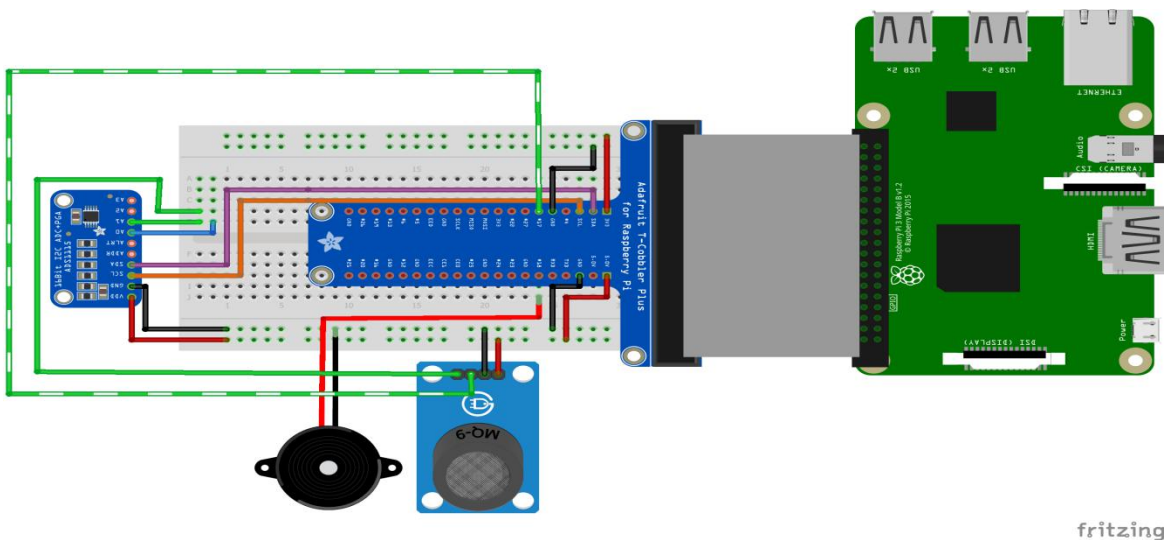
## 2.3 Experiment to detect the smoke by using the smoke sensor.

**Aim:** To set up the smoke sensor using Raspberry Pi.

**Requirements:** Raspberry Pi board, Smoke Sensor, Jumper cables.

**Description:** The smoke sensor we will use is the MQ-2. This is a sensor that is not only sensitive to smoke, but also to flammable gas. The MQ-2 smoke sensor reports smoke by the voltage level that it outputs. The more smoke there is, the greater the voltage that it outputs. Conversely, the less smoke that it is exposed to, the less voltage it outputs. The MQ-2 also has a built-in potentiometer to adjust the sensitivity to smoke. By adjusting the potentiometer, you can change how sensitive it is to smoke, so it's a form of calibrating it to adjust how much voltage it will put out in relation to the smoke it is exposed to. We will wire the MQ-2 to a Raspberry Pi so that the Raspberry Pi can read the amount of voltage output by the sensor and output to us if smoke has been detected if the sensor outputs a voltage above a certain threshold. This way, we will know that the sensor has, in fact, detected smoke.

### Circuit Diagram:



### Program:

```
#####SimpsonsAlarm.py#####
import RPi.GPIO as GPIO
import time
import math

# Board Setup
```



```

GPIO.setmode(GPIO.BCM)

# Pin Setup
gas = 17
buzzer = 18

GPIO.setup(gas, GPIO.IN)
GPIO.setup(buzzer, GPIO.OUT)
global Buzz
GPIO.PWM
Buzz = GPIO.PWM(buzzer, 440)

def loop():
    while True:
        lvls = "Gas: " +str(gaslvl)+", Fire: "+str(firetmp)
        digGas = GPIO.input(gas) # Gas sensor digital output

        if digGas == 0: # fire or gas levels are too high
            warning = 'DOH! Something is not okay!!\nLevels: '+lvls #Simpson's Reference
            print(warning)
            time_end = time.time() + 30 # setup for 30 second while loop
            while time.time() < time_end: # flash warnings for 30 seconds
                clearLCD()
                print(0, 0, '*****Danger*****')
                print(0, 1, ' Gas:'+str(gaslvl))
                time.sleep(3)
            else: # otherwise
                print (lvls) # display fire and gas levels in command line
                print(0, 0, 'Gas: '+str(gaslvl))
                print(0, 0, 'Everything Is Ok')
                print(0, 1, '*****')
                Buzz.start(50) # make the buzzer make noise
                time.sleep(2)
                Buzz.stop() # stop the buzzer
                time.sleep()

```

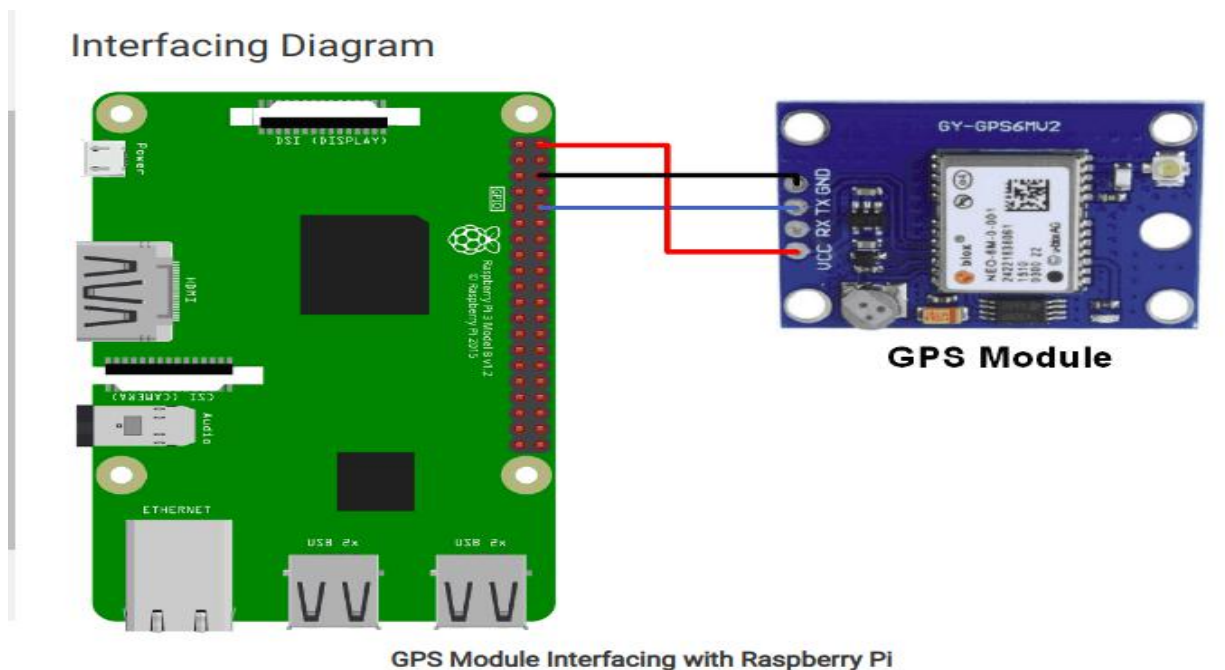
## 2.4 Experiment to detect the location by using the GPS module.

**Aim:** To detect the location by GPS with Raspberry Pi.

**Requirements:** raspberry pi, GPS Module, jumperwires.

**Description:** GPS stands for Global Positioning System and used to detect the Latitude and Longitude of any location on the Earth, with exact UTC time (Universal Time Coordinated). GPS module is the main component in our vehicle tracking system project. This device receives the coordinates from the satellite for each and every second, with time and date.

**Circuit Diagram:**



**Program:**

GPS Interfacing with Raspberry Pi using Python

<http://www.electronicwings.com>

'''

```
import serial          #import serial package
```

```
from time import sleep
```

```
import webbrowser      #import package for opening link in browser
```

```
import sys             #import system package
```

```
def GPS_Info():
```

```

global NMEA_buff
global lat_in_degrees
global long_in_degrees
nmea_time = []
nmea_latitude = []
nmea_longitude = []
nmea_time = NMEA_buff[0]          #extract time from GPGGA string
nmea_latitude = NMEA_buff[1]       #extract latitude from GPGGA string
nmea_longitude = NMEA_buff[3]      #extract longitude from GPGGA string

print("NMEA Time: ", nmea_time,'\n')
print ("NMEA Latitude:", nmea_latitude,"NMEA Longitude:", nmea_longitude,'\n')

lat = float(nmea_latitude)          #convert string into float for calculation
longi = float(nmea_longitude)       #convert string into float for calculation

lat_in_degrees = convert_to_degrees(lat) #get latitude in degree decimal format
long_in_degrees = convert_to_degrees(longi) #get longitude in degree decimal format

#convert raw NMEA string into degree decimal format
def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value - int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = "%.4f" %(position)
    return position

gpgga_info = "$GPGGA,"
ser = serial.Serial ("/dev/ttyS0", baudrate=9600)          #Open port with baud rate
GPGGA_buffer = 0
NMEA_buff = 0
lat_in_degrees = 0
long_in_degrees = 0
try:
    while True:
        received_data = (str)(ser.readline())              #read NMEA string received
        GPGGA_data_available = received_data.find(gpgga_info) #check for NMEA GPGGA string
        #print(received_data)
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split("$GPGGA,",1)
            NMEA_buff = (GPGGA_buffer.split(','))
            print(NMEA_buff)
            GPS_Info()                                     #get time, latitude, longitude

```

```
print("lat in degrees:", lat_in_degrees," long in degree: ", long_in_degrees, '\n')
```

```
map_link = 'http://maps.google.com/?q=' + lat_in_degrees + ',' + long_in_degrees
```

```
print("<<<<<<<press ctrl+c to plot location on google maps>>>>>>\n")
```

```
print("-----\n")
```

except KeyboardInterrupt:

```
webbrowser.open(map_link)    #open current position information in google map
```

```
sys.exit(0)
```

**Output:**

```

40
41 gpgga_info = "$GPGGA,"
42 ser = serial.Serial ("/dev/ttyS0", baudrate=9600)          #Open port with baud rate
43 GPGGA_buffer = 0
44 NMEA_buff = 0
45 lat_in_degrees = 0
46 long_in_degrees = 0
47
48 try:
49     while True:
50         received_data = (str)(ser.readline())              #read NMEA string received
51         GPGGA_data_available = received_data.find(gpgga_info)  #check for NMEA GPGGA string
52         #print(received_data)
53         if (GPGGA_data_available>0):
54             GPGGA_buffer = received_data.split("$GPGGA,")[1]  #store data coming after "$GPGGA," string
55             NMEA_buff = (GPGGA_buffer.split(','))              #store comma separated data in buffer
56             print(NMEA_buff)
57             GPS_info()                                         #get time, latitude, longitude
58
59             print("lat in degrees:", lat_in_degrees," long in degree: ", long_in_degrees, '\n')
60             map_link = 'http://maps.google.com/?q=' + lat_in_degrees + ',' + long_in_degrees
61             webbrowser.open(map_link)
62
63 except KeyboardInterrupt:
64     webbrowser.open(map_link)
65     sys.exit(0)

```

```

[091912.00', '1723.50805', 'N', '07819.17330', 'E', '1', '05', '4.49', '494.8', 'M', '-74.2', 'N', '', '*71\\r\\n"]
NMEA Time: 091912.00

NMEA Latitude: 1723.50805 NMEA Longitude: 07819.17330

lat in degrees: 17.3918 long in degree: 78.3196

<<<<<<<press ctrl+c to plot location on google maps>>>>>>
-----

```

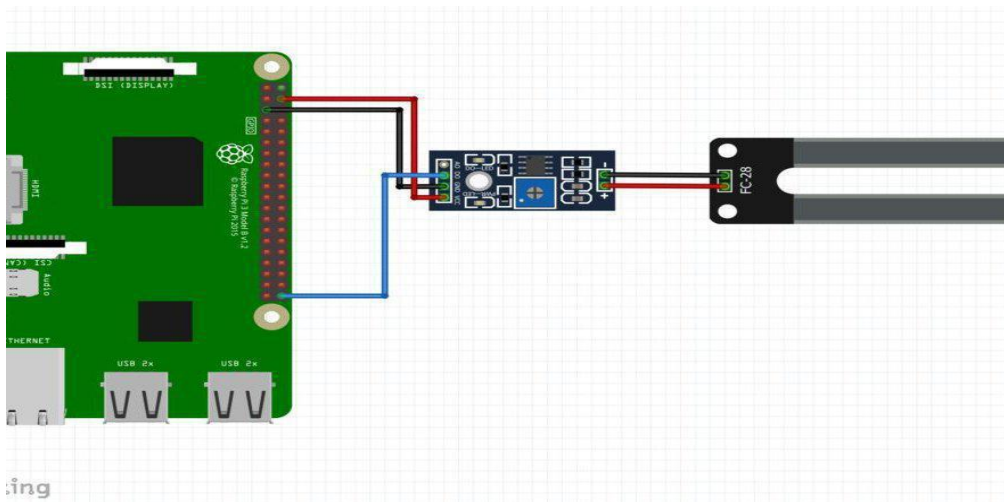
### 3. Experiment to detect the moisture level of the soil by using the soil moisture.

**Aim:** To detect moisture in soil with Soil Moisture Sensor.

**Requirements:** Raspberry pi, Soil Moisture, jumper wires.

**Description:** The sensor measures the volumetric content of water in soil and presents the moisture value in voltage. It also provides both analog and digital outputs, but for this project we'll be solely utilizing the analog output. To enable the sensor analog mode, we'll need an analog pin to produce output. Which poses an issue, since the Raspberry Pi doesn't include an analog pin.

**Circuit Diagram:**



**Program:**

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

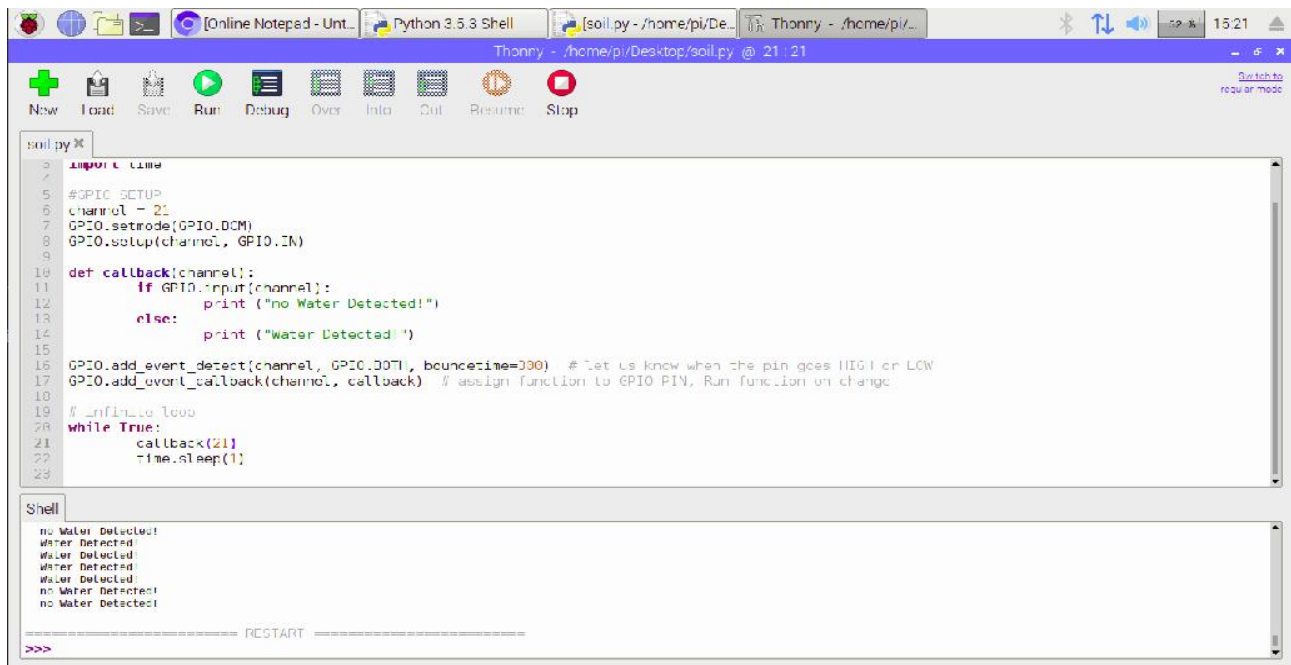
#GPIO SETUP
channel = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
    if GPIO.input(channel):
        print ("no moisture Detected!")
    else:
        print ("moisture Detected!")
```

```
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300) # let us know when the pin goes
HIGH or LOW
GPIO.add_event_callback(channel, callback) # assign function to GPIO PIN, Run function on change
```

```
# infinite loop
while True:
    callback(21)
    time.sleep(1)
```

## Output:



```
soil.py
1 import RPi.GPIO as GPIO
2
3 # GPIO SETUP
4 channel = 21
5 GPIO.setmode(GPIO.BOARD)
6 GPIO.setup(channel, GPIO.IN)
7
8 def callback(channel):
9     if GPIO.input(channel):
10         print ("no Water Detected!")
11     else:
12         print ("Water Detected!")
13
14 GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300) # let us know when the pin goes HIGH or LOW
15 GPIO.add_event_callback(channel, callback) # assign function to GPIO PIN, Run function on change
16
17 # infinite loop
18 while True:
19     callback(21)
20     time.sleep(1)
21
22
23
```

```
Shell
no Water Detected!
Water Detected!
Water Detected!
Water Detected!
Water Detected!
no Water Detected!
no Water Detected!
RESTART
>>>
```

## 4. Experiment to detect the temperature by using Humidity sensor

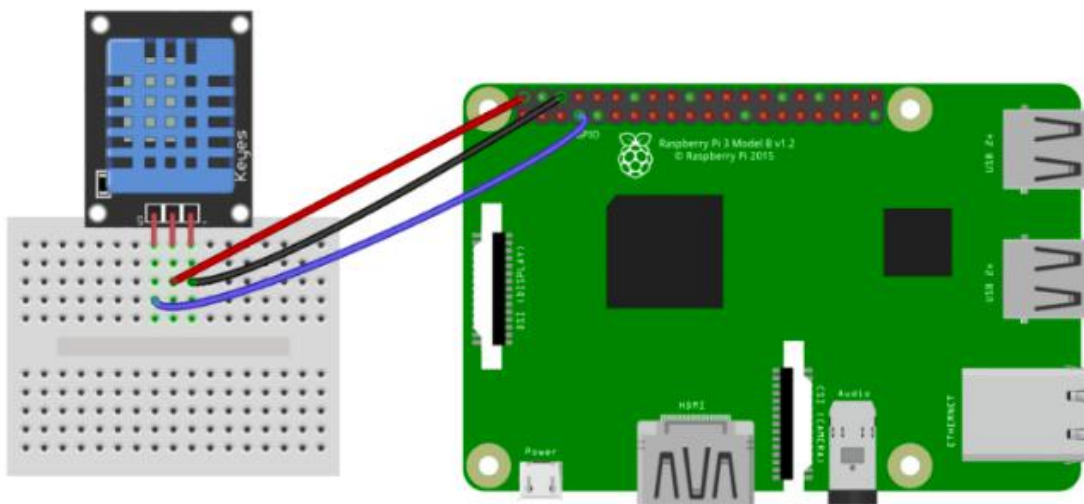
**Aim:** To detect temperature and humidity using temperature and humidity sensor.

**Requirements:** Raspberry pi, Humidity Sensor, jumper wires.

**Description:** Digital temperature sensors like the DS18B20 differ from analog thermistors in several important ways. In thermistors, changes in temperature cause changes in the resistance of a ceramic or polymer semiconducting material. Usually, the thermistor is set up in a voltage divider, and the voltage is measured between the thermistor and a known resistor. The voltage measurement is converted to resistance and then converted to a temperature value by the microcontroller.

Digital temperature sensors are typically silicon based integrated circuits. Most contain the temperature sensor, an analog to digital converter (ADC), memory to temporarily store the temperature readings, and an interface that allows communication between the sensor and a microcontroller. Unlike analog temperature sensors, calculations are performed by the sensor, and the output is an actual temperature value.

**Circuit Diagram:**



Steps  
to be  
followed

**before executing the code:**

First of all, some packages have to be installed:

```
sudo apt-get update
```

```
sudo apt-get install build-essential python-dev python-openssl git
```

Now the library for the sensors can be loaded. I use a pre-built Adafruit library that supports a variety of sensors:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git && cd Adafruit_Python_DHT
```

```
sudo python setup.py install
```

This creates a Python library that we can easily integrate into our projects.

