**(2018-2019)**

**16 CS C31 – Internet of Things**

**Year: 3rd Semester: 6th**

**Lab Manual**

**Department Of Computer Science and Engineering**



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

**(Autonomous)**

**Gandipet - 500075**

**PREPARED BY: FACULTY INCHARGE:**

**B. Tamiz Ms. K. Dharani Tejaswini,**

**160116733177, Assistant Professor,**

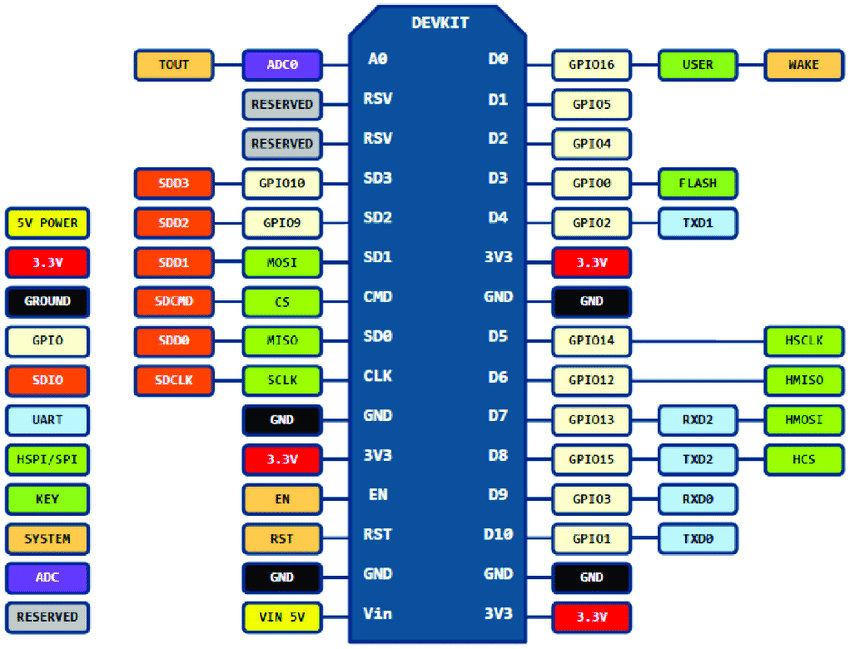
**B.E 3/4 – CSE 3, Department Of CSE,**

**2018 – 2019. C B I T.**

***Index***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Experiment Name** | **Date** | **Page No** | **Remark** |
| 1 | NodeMCU ESP8266 architecture |  | 3 |  |
| 2 | Write a program to implement LED blinking with NodeMCU. | 21/12/2018 | 12 |  |
| 3 | Write a program to implement rain water detection using rain sensor with NodeMCU. | 28/12/2018 | 16 |  |
| 4 | Write a program to implement encryption and decryption using Hill Cipher Technique. | 04/01/2019 | 18 |  |
| 5 | Write a program to implement encryption and decryption using DES Algorithm Technique. | 11/01/2019 | 24 |  |
| 6 | Write a program to implement encryption and decryption using Triple DES Algorithm Technique with 3 keys. | 18/01/2019 | 27 |  |
| 7 | Write a program to implement encryption and decryption using RSA Algorithm. | 25/01/2019 | 29 |  |
| 8 | Write a program to implement encryption and decryption using Blowfish Algorithm. | 15/02/2019 | 33 |  |
| 9 | Write a program to implement Secure Hash Algorithm. | 01/03/2019 | 36 |  |
| 10 | Write a program to implement Message Digest Algorithm. | 01/03/2019 | 39 |  |
| 11 | Write a program to implement MAC Algorithm. | 08/03/2019 | 43 |  |
| 12 | Write a program to implement Digital Certificate. | 15/03/2019 | 45 |  |
| 13 | Write a program to demonstrate Diffie-Hellman key exchange Algorithm. | 15/03/2019 | 48 |  |

***NodeMCU ESP8266 Architecture***

******

**NodeMCU** is Lua based firmware of ESP8266. It contains many pins.Some of them are:

1)**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.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.Hence 11 pins are used.2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC.Note that D0/GPIO16 pin can be only used as GPIO read/write, no special functions are supported on it.

1. **Analog to Digital Converter (ADC)** is used to convert analog signal into digital form.
2. ****Pulse Width Modulation (PWM)**** is a technique by which width of a pulse is varied while keeping the frequency of the wave constant.

p

1. **I2C (Inter-Integrated Circuit)** is serial bus interface connection protocol. It is also called as TWI (two wire interface) since it uses only two wires for communication. Those two wires are

SDA (serial data) and SCL (serial clock).I2C is acknowledgement based communication protocol.I2Cworks in two modes namely,

* Master mode
* Slave mode

1. **HTTP (Hypertext Transfer Protocol)** is a standard Application protocol which functions as request-response protocol between client and server.
2. The **Serial Peripheral Interface (SPI)** is a bus interface connection protocol.NodeMCU based ESP8266 has Hardware SPI with four pins available for SPI communication. With this SPI interface, we can connect any SPI enabled device with NodeMCU and make communication possible with it.

***Lab Experiment-1***

**I) Aim:** Blinking the ‘ON BOARD’ LED light.

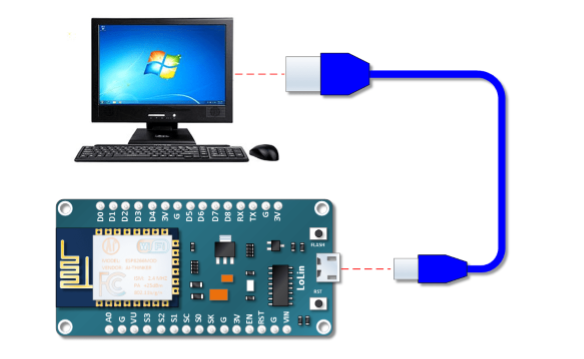
**Description:** To **‘ON’** the blue LED present on the NodeMCU board for one second and after one second delay **‘OFF’** the LED.This is repeated continuously.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU.

**Software Requirements:** Arduino IDE v1.1.8 Windows

**Board/Connection Diagram:**

****

**CODE:**

void setup()

{

pinMode(LED\_BUILTIN,OUTPUT);

}

void loop()

{

digitalWrite(LED\_BUILTIN,HIGH);

delay(1000);

digitalWrite(LED\_BUILTIN,LOW);

delay(1000);

}

**Result:** The IN\_BUILT LED blinks.

**II) Aim:** Blinking an ‘external’ LED light without a resistor.

**Description:** To **‘ON’** the external red LED present on the breadboard for five seconds and after two seconds delay **‘OFF’** the LED.This is repeated continuously.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

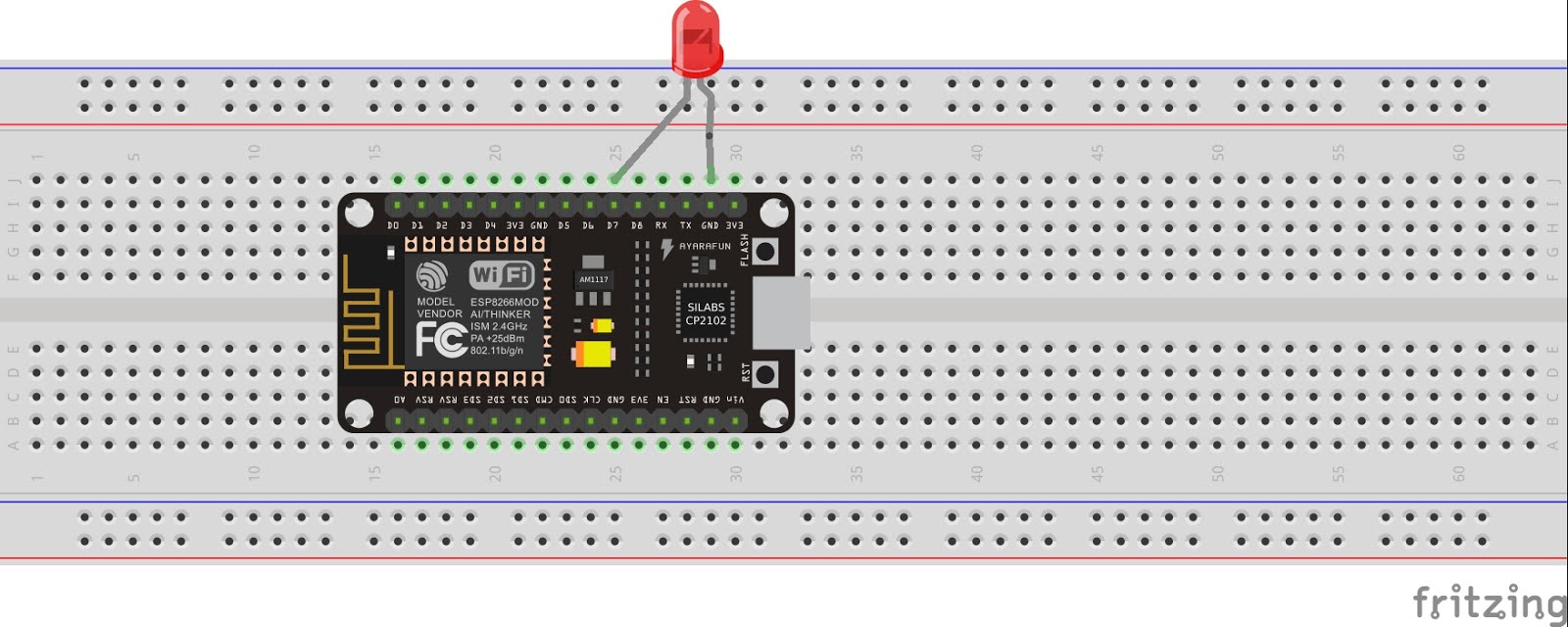
2)USB cable to connect NodeMCU Board to CPU

3) Breadboard

4)LED.

**Software Requirements:** Arduino IDE v1.1.8 Windows

**Board/Connection Diagram:**

****

**CODE:**

#define LED 13

void setup()

{

pinMode(LED,OUTPUT);

}

void loop()

{

digitalWrite(LED,HIGH);

delay(5000);

digitalWrite(LED,LOW);

delay(2000);

}

**Result:** The external LED light blinks.

**III) Aim:** Blinking an ‘external’ LED light with a resistor.

**Description:** To **‘ON’** the external red LED present on the breadboard for five seconds and after two seconds delay **‘OFF’** the LED.This is repeated continuously.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU

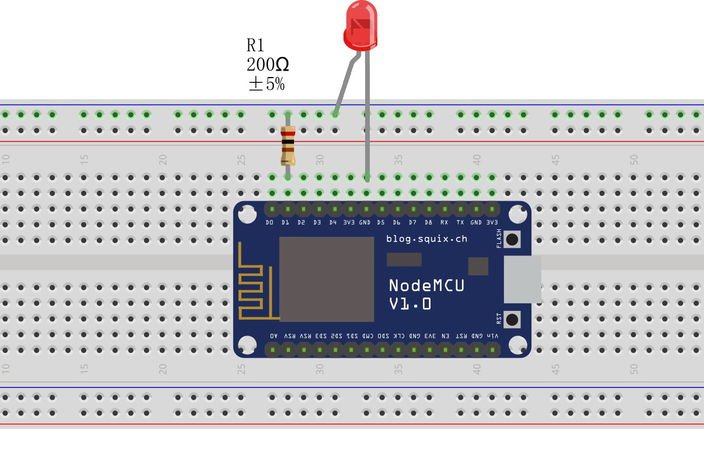
3) Breadboard

4)LED

5)Resistor 330 ohm.

**Software Requirements:** Arduino IDE v1.1.8 Windows

**Board/Connection Diagram:**

****

**CODE:**

#define LED 13

void setup()

{

pinMode(LED,OUTPUT);

}

void loop()

{

digitalWrite(LED,HIGH);

delay(5000);

digitalWrite(LED,LOW);

delay(2000);

}

**Result:** The external LED with resistor blinks.

**IV) Aim:** Controlling an LED light using ‘**WiFi**’ and ‘**BLYNK APP**’.

**Description:** Since ESP8266 is a WIFI module, we can use it to operate the output devices (such as LED, motors etc.) wirelessly. To ON an external RED colour LED present on the breadboard for one second and after one second delay OFF the LED. This is repeated continuously.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU

3) Breadboard

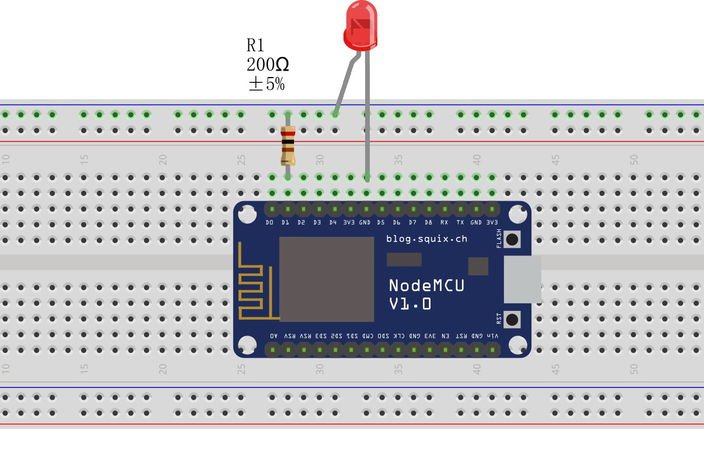
4)LED

5)Resistor 330 ohm.

**Software Requirements:** 1)Arduino IDE v1.1.8 Windows

2)Blynk App

**Board/Connection Diagram:**

****

**CODE:**

#define BLYNK\_PRINT Serial

#include <ESP8266Wifi.h>

#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "4e732d4bb5ea4bac8016f1955a5f82c6"; //CHANGE THISS.....enter the Auth token that has been sent to your mail ID by Blynk app of your mobile phone.

// Your WiFi credentials or Hotspot credentials (in case you do not have WI-FI in your surroundings).

// Set password to "" for open networks.

char ssid[] = "BSNL\_BB"; // CHANGE THISSS...the network name

char pass[] = "india0516"; //CHANGE THISSS... WiFi/ hotspot password

void setup() {

// Debug console Serial.begin(9600);

Blynk.begin(auth, ssid, pass);

// You can also specify server:

//Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);

//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);

}

void loop() {

Blynk.run();

}

**Result:** The LED blinks with the help of the Blynk App.

***Lab Experiment-2***

**I) Aim:** Connect the ‘**Rain Sensor**’ to NodeMCU and display output onto the serial monitor:

**0**-When there is rain.

**1**-When there is no rain.

**Description:** To display **‘0’** on the serial monitor when water is poured on to the rain shield and to display ‘**1**’ when the rain shield is dry.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU

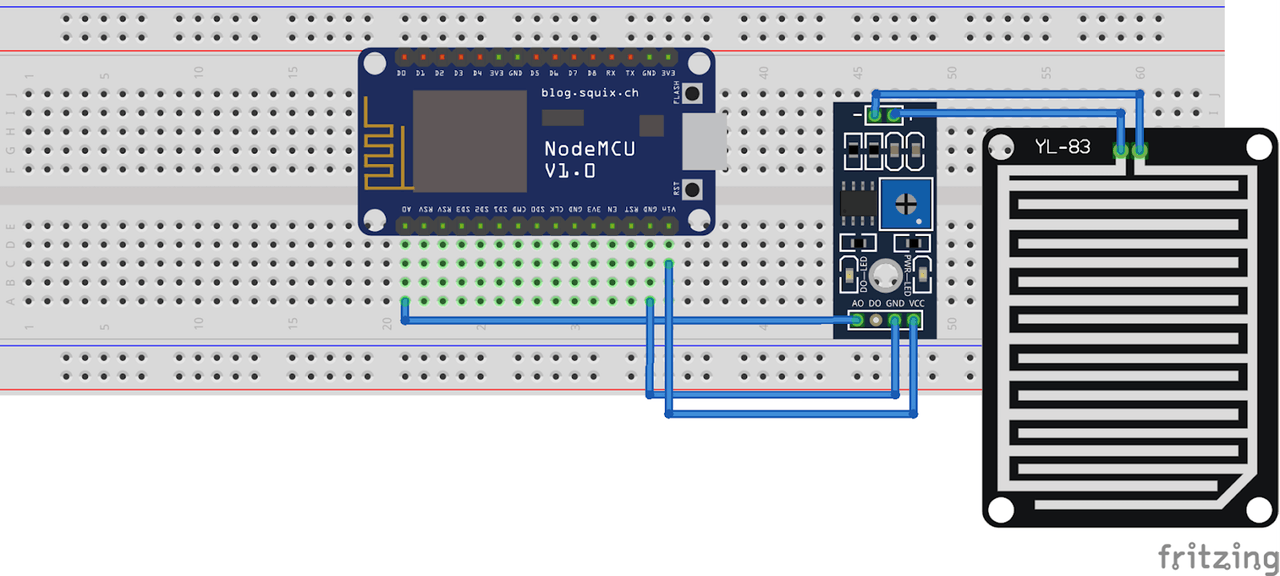
3) Breadboard

4)rain Sensor

5)Rain Shield.

**Software Requirements:** Arduino IDE v1.1.8 Windows

**Board/Connection Diagram:**

****

**CODE:**

void setup() {

Serial.begin(9600);

pinMode(13,INPUT);

}

void loop()

{

int sensorReading = digitalRead(D7);

if(sensorReading == 0)

Serial.println(“0”);

else

Serial.println(“1”);

delay(1000);

}

**Result:** It is not raining.

It is not raining.

It is raining. **[When water is poured and the shield is wet].**

**II) Aim:** Connect an ‘**LED**’ to the above circuit and let the LED glow when ever there is rain.

**Description:** To **‘ON’** the external red LED present on the breadboard when there is water and when there is no water **‘OFF’** the LED.

.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU

3) Breadboard

4)rain Sensor

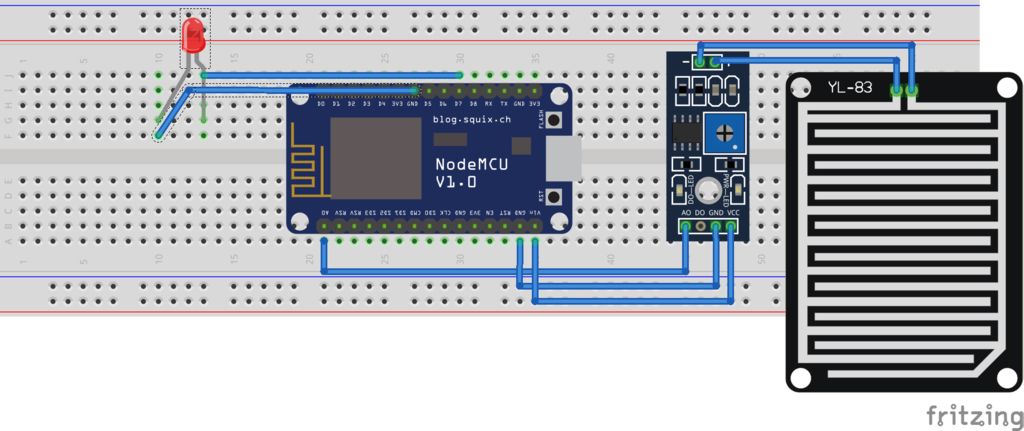
5)Rain Shield

6)Resistor 330 ohm

7)LED.

**Software Requirements:** Arduino IDE v1.1.8 Windows

**Board/Connection Diagram:**

****

**CODE:**

void setup()

{

pinMode(LED,OUTPUT);

}

void loop() {

int sensorReading = digitalRead(D5);

if(sensorReading == 0)

digitalWrite(LED,HIGH);

else

digitalWrite(LED,LOW);

delay(1000);

}

**Result:** It is not raining.

LED does not blink.

It is not raining.

LED does not blink.

It is raining. **[When water is poured and the shield is wet].**

LED **blinks.**

**III) Aim:** Send a notification to your ‘**Blynk App**’ when ever there is rain.

**Description:** Since ESP8266 is a WIFI module, we can use it to operate the output devices (such as LED, motors etc.) wirelessly. To ON an external RED colour LED present on the breadboard for one second and after one second delay OFF the LED. This is repeated continuously.

**Hardware Requirements:** 1) NodeMCU ESP8266 Board

2)USB cable to connect NodeMCU Board to CPU

3) Breadboard

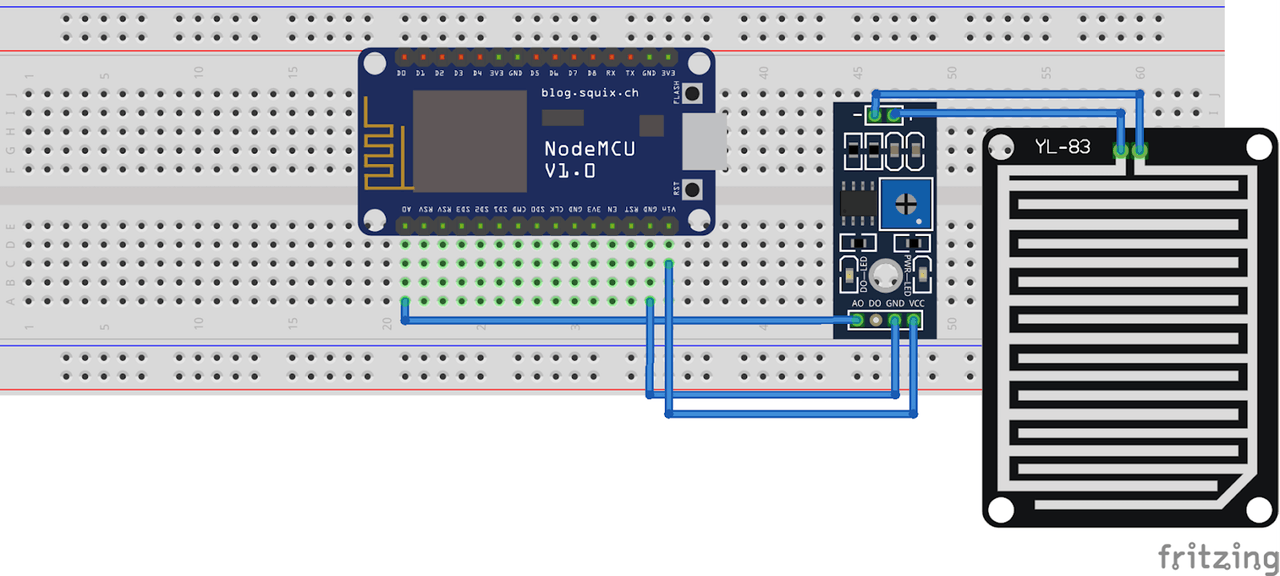
4)Rain Sensor

5)Rain Shield.

**Software Requirements:** 1)Arduino IDE v1.1.8 Windows

2)Blynk App.

**Board/Connection Diagram:**



**CODE:**

#define BLYNK\_PRINT Serial

#include <ESP8266Wifi.h>

#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "4e732d4bb5ea4bac8016f1955a5f82c6"; //CHANGE THISS.....enter the Auth token that has been sent to your mail ID by Blynk app of your mobile phone.

// Your WiFi credentials or Hotspot credentials (in case you do not have WI-FI in your surroundings).

// Set password to "" for open networks.

char ssid[] = "BSNL\_BB"; // CHANGE THISSS...the network name

char pass[] = "india0516"; //CHANGE THISSS... WiFi/ hotspot password

void setup() {

// Debug console Serial.begin(9600);

Blynk.begin(auth, ssid, pass);

// You can also specify server:

//Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);

//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);

}

void loop() {

Int sensorReading = digitalRead(D5);

Blynk.run();

if(sensorReading == 0)

Blynk.notify(“Raining”);

else

Blynk.notify(“ Not Raining”);

}

**Result:** When water is poured on the rain shield, a notification is sent to the app saying “Raining” and when it is dry the app notifies that it is “Not Raining”.

***Lab Experiment-3***

# I) Aim: To get the distance between obstacle and the sensor using ultrasonic sensor.

**Description:** Connect trigger pin to D1 of NODEMCU and echo pin to D2 of NODEMCU. The distance between the sensor and the obstacle is shown on the serial monitor.

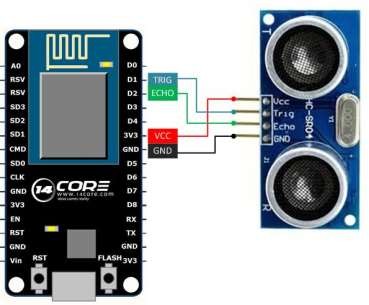
# Hardware Requirements:

* + NODEMCU ESP8266 Board
  + USB Cable to connect NODEMCU board to CPU
  + Breadboard
  + Ultrasonic sensor
  + Jumper wires

# Software Requirements:

* + Arduino IDE v1.8.8 Windows

# Board/ Connection Diagram



**Code:**

#define TRIGGER 5

#define ECHO 4

void setup()

{

Serial.begin (9600);

pinMode(TRIGGER, OUTPUT);

pinMode(ECHO, INPUT);

pinMode(BUILTIN\_LED, OUTPUT);

}

void loop() {

long duration, distance;

digitalWrite(TRIGGER, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER, LOW);

duration = pulseIn(ECHO, HIGH);

distance = (duration/2) / 29.1;

Serial.print(distance);

Serial.println("Centimeter:");

delay(1000);

}

**Result:** The distance between obstacle and sensor is shown on the serial monitor.

**II) Aim**: To display distance on the blynk app using lcd display

**Description:** Since ESP8266 is a WIFI module, we can use it to operate the output devices wirelessly. In this experiment, distance between obstacle and sensor is displayed on blynk app using lcd display.

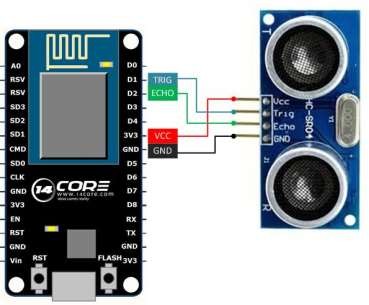
# Hardware Requirements:

* + NODEMCU ESP8266 Board
  + USB Cable to connect NODEMCU board to CPU
  + Breadboard
  + Ultrasonic sensor
  + Jumper wires

# Software Requirements:

* + Arduino IDE v1.8.8 Windows
  + Blynk App

**Connection Diagram:**



# Code:

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#define TRIGGERPIN D1

#define ECHOPIN D2

char auth[] = "cec8874b842241eb971b3e1c32a850ec";

char ssid[] = "Redmi";

char pass[] = "lavubhargu"; WidgetLCD lcd(V1);

void setup()

{

Serial.begin(9600);

pinMode(TRIGGERPIN, OUTPUT);

pinMode(ECHOPIN, INPUT);

Blynk.begin(auth, ssid, pass);

lcd.clear();

lcd.print(0, 0, "Distance in cm");

}

void loop()

{

lcd.clear();

lcd.print(0, 0, "Distance in cm");

long duration, distance;

digitalWrite(TRIGGERPIN, LOW);

delayMicroseconds(3);

digitalWrite(TRIGGERPIN, HIGH);

delayMicroseconds(12);

digitalWrite(TRIGGERPIN, LOW);

duration = pulseIn(ECHOPIN, HIGH);

distance = (duration/2) / 29.1;

Serial.print(distance);

Serial.println("Cm");

lcd.print(7, 1, distance);

Blynk.run();

delay(3500);

}

**Result** : The distance between obstacle and sensor is shown on the blynk app.

**III) Aim**: To locate the current position of GPS module and display it on the google maps in the blynk app.

**Description:** Connect TX of GPS module to D2 of NODEMCU, RX of GPS module to D1 of NODEMCU, GND of GPS module to GND of NODEMCU, VCC to 3V3 of NODEMCU

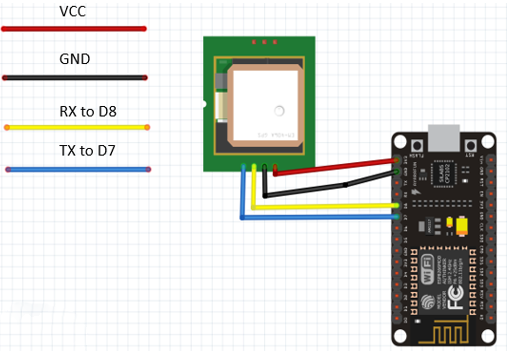
# Hardware Requirements:

* + NODEMCU ESP8266 Board
  + USB Cable to connect NODEMCU board to CPU
  + Breadboard
  + GPS module
  + Jumper wires

# Software Requirements:

* + Arduino IDE v1.8.8 Windows
  + Blynk App

**Connection Diagram:**

****

# Code:

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

static const int RXPin = 4, TXPin = 5;

static const uint32\_t GPSBaud = 9600;

TinyGPSPlus gps;

WidgetMap myMap(V0);

SoftwareSerial ss(RXPin, TXPin);

BlynkTimer timer;

float spd;

float sats;

String bearing;

char auth[] = "941b8474a77e4ff8b596f2ca2a652162";

char ssid[] = "Redmi";

char pass[] = "lavubhargu";

unsigned int move\_index = 1; // fixed location for now

void setup()

{

Serial.begin(115200);

Serial.println();

ss.begin(GPSBaud);

Blynk.begin(auth, ssid, pass);

timer.setInterval(5000L, checkGPS);

}

void checkGPS()

{

if (gps.charsProcessed() < 10)

{

Serial.println(F("No GPS detected: check wiring."));

Blynk.virtualWrite(V4, "GPS ERROR");

}

}

void loop()

{

while (ss.available() > 0)

{

if (gps.encode(ss.read())) displayInfo();

}

Blynk.run();

timer.run();

}

void displayInfo()

{

if (gps.location.isValid() )

{

float latitude = (gps.location.lat()); //Storing the Lat. and Lon.

float longitude = (gps.location.lng());

Serial.print("LAT: ");

Serial.println(latitude, 6); // float to x decimal places

Serial.print("LONG: ");

Serial.println(longitude, 6);

Blynk.virtualWrite(V1, String(latitude, 6));

Blynk.virtualWrite(V2, String(longitude, 6));

myMap.location(move\_index, latitude, longitude, "GPS\_Location");

spd = gps.speed.kmph(); //get speed

Blynk.virtualWrite(V3, spd);

sats = gps.satellites.value(); //get number of satellites

Blynk.virtualWrite(V4, sats);

bearing = TinyGPSPlus::cardinal(gps.course.value());

Blynk.virtualWrite(V5, bearing);

}

Serial.println();

}

**Result :** The current location of GPS module is shown on the blynk app.

**Output:**



**IV) Aim:** To use accelerometer and find x,y,z coordinates

# Description: Connect VCC of accelerometer to 3v3 pin of nodemcu .Connect x-out (or) y-out (or) z-out to A0 pin of the nodemcu . GND of accelerometer to GND of the nodemcu.

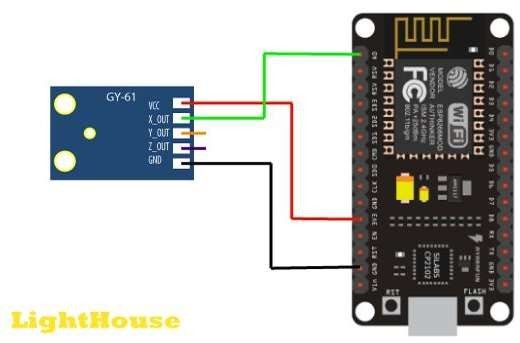
# Hardware requirements:

* + Nodemcu
  + Breadboard
  + Accelerometer
  + Connecting wires
  + USB cabe

# Software requirements:

Arduino IDE v1.8.8 windows

# Connection diagram:



**Code:**

const int xpin=A0;

void setup()

{

Serial.begin(9600);

}

void loop()

{

int x=analogRead(xpin);

delay(100);

Serial.print(“x-axis”);

Serial.print(x);

}

**Result:** x,y,z co-ordinates are displayed on the serial monitor.

***Lab Experiment-4***

**AIM:** To detect the smoke in the room using smoke sensor.

**DESCRIPTION:** Connect A0 pin of smoke sensor to A0 pin of NODEMCU and D0 pin remains disconnected. GND pin of smoke sensor is connected to GND of NODEMCU.

**HARDWARE REQUIREMENTS:**

•Nodemcu

•Breadboard

•Smoke sensor

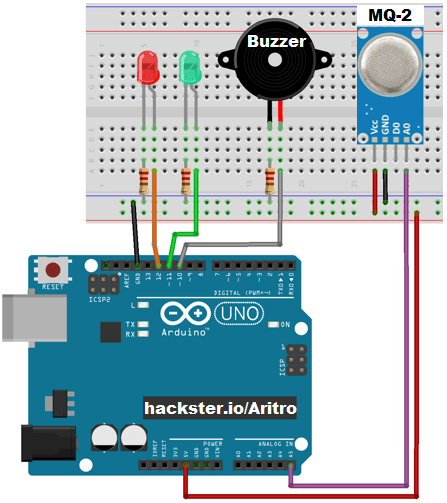
•Connecting wires

•USB cable

**SOFTWARE REQUIREMENTS:**

•Arduino IDE v 1.8.8 Ubuntu

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

int redLed = 12;

int buzzer = 4;

int smokeA0 = A0;

int sensorThres = 700;

void setup()

{

pinMode(redLed, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(smokeA0, INPUT);

Serial.begin(9600);

}

void loop()

{

int analogSensor = analogRead(smokeA0);

Serial.print("Pin A0: ");

Serial.println(analogSensor);

if (analogSensor < sensorThres)

{

digitalWrite(redLed, HIGH);

Serial.println("no smoke");

noTone(buzzer);

}

else

{

digitalWrite(redLed, LOW);

Serial.println("smoke detected");

tone(buzzer, 1000, 200);

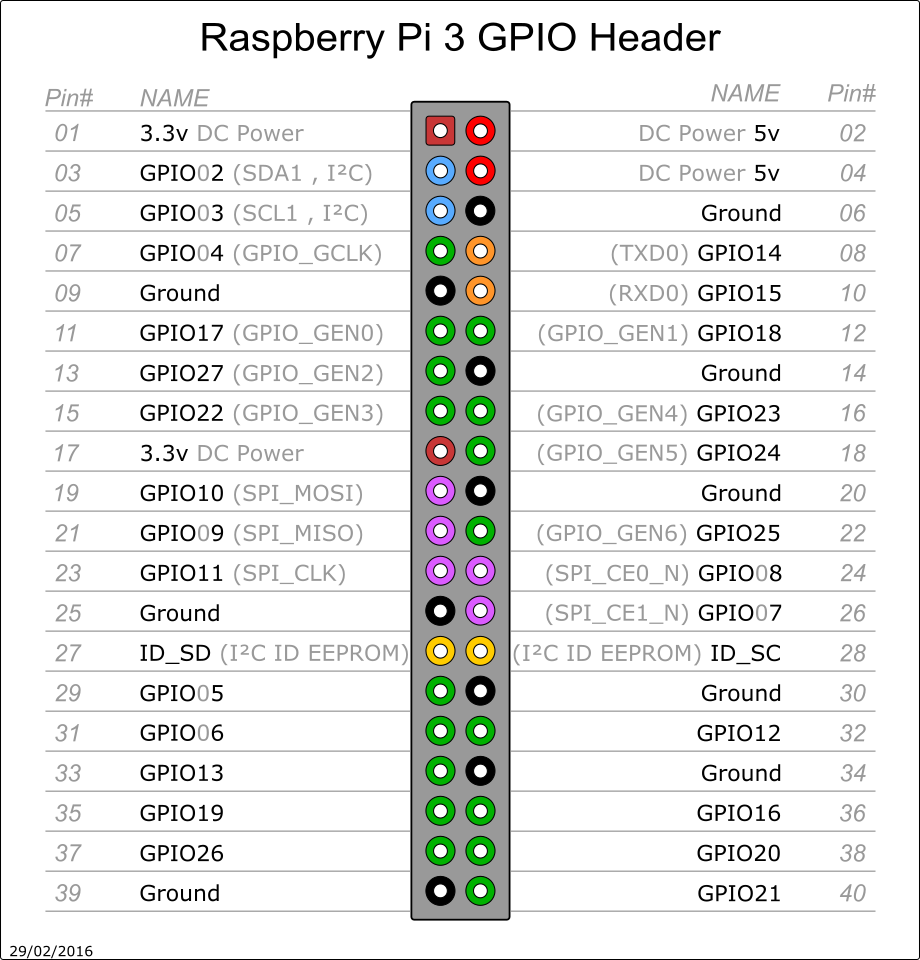
}

delay(100);

}

**RESULT:** The smoke in the room is detected and buzzer will be switched on if there is smoke.

***Raspberry Pi 3 B+ Architecture***

******

***Lab Experiment-5***

**AIM:** To make an LED blink using Raspberry Pi.

**DESCRIPTION:** On and off an LED using Raspberry Pi module and Python programming.

**HARDWARE REQUIREMENTS:**

•Raspberry Pi 3 B+ Module

•Breadboard

•LED

•Connecting wires

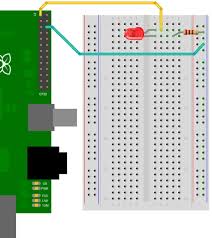
•Resistor

**SOFTWARE REQUIREMENTS:**

•Raspbian OS

•Python

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

import RPi.GPIO as GPIO

from time import sleep

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(8, GPIO.OUT, initial = GPIO.LOW)

While True:

GPIO.output(8, GPIO.HIGH)

sleep(1)

GPIO.output(8, GPIO.LOW)

sleep(1)

**RESULT:** The LED blinks.

***Lab Experiment-6***

**AIM:** To implement rain alert system using Raspberry Pi.

**DESCRIPTION:** When it is raining, the buzzer beeps and displays the alert and when its not raining no buzzer rings and it displays it is not raining.

**HARDWARE REQUIREMENTS:**

•Raspberry Pi 3 B+ Module

•Breadboard

•Rain sensor, shield

•Connecting wires

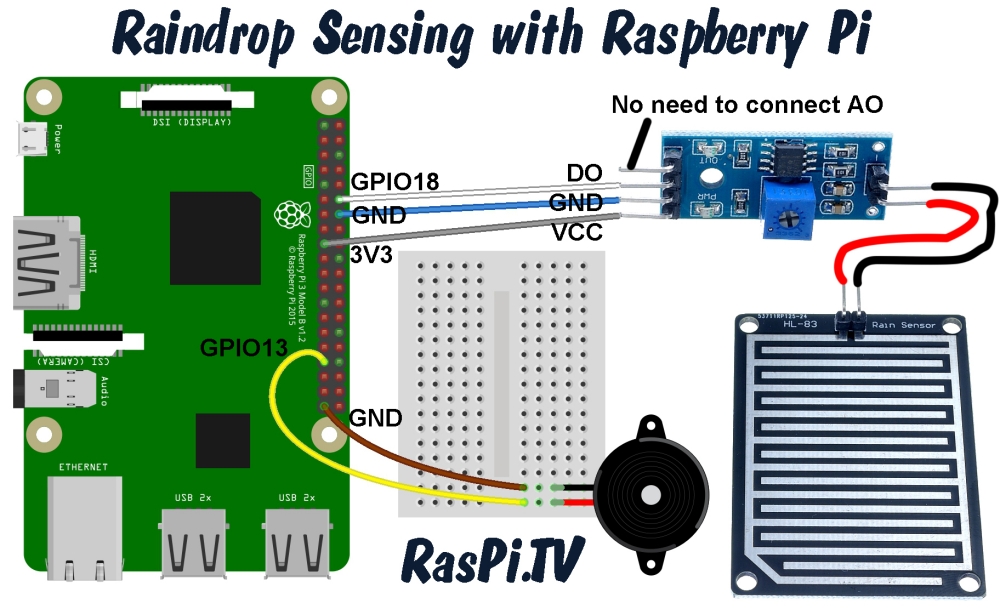
•Buzzer

**SOFTWARE REQUIREMENTS:**

•Raspbian OS

•Python

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

from gpiozero import Buzzer,InputDevice

from time import sleep

buzz = Buzzer(13)

no\_rain =InputDevice(18)

def buzz\_now(iterations):

for x in range(iterations):

buzz.on()

sleep(0.1)

buzz.off()

sleep(0.1)

while True:

if not no\_rain.is\_active:

print(“Its Raining”)

buzz\_now(5)

else:

print(“It is not Raining”)

sleep(1)

**RESULT:** The buzzer rings.

It is Raining.

***Lab Experiment-7***

**AIM:** To implement traffic signalling system with ultrasonic sensor using Raspberry Pi.

**DESCRIPTION:** Based on the distance read using the ultrasonic sensor, the traffic lights are operated.

**HARDWARE REQUIREMENTS:**

•Raspberry Pi 3 B+ Module

•Breadboard

•3 LEDs

•Connecting wires

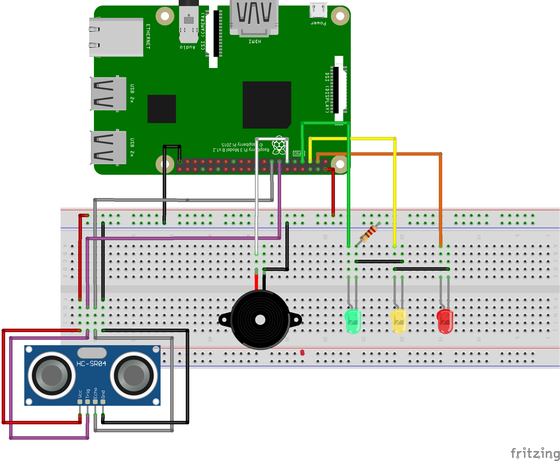
•Ultrasonic sensor

**SOFTWARE REQUIREMENTS:**

•Raspbian OS

•Python

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

#!/user//bin/env python

#libraries

import RPi.GPIO as GPIO

from gpiozero import TrafficLights

import time

GPIO.setmode(GPIO.BCM)

lights = TrafficLights(25, 8, 7)

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

GPIO.output(GPIO\_TRIGGER, True)

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

startTime = time.time()

stopTime = time.time()

while GPIO.input(GPIO\_ECHO) == 0:

startTime = time.time()

while GPIO.input(GPIO\_ECHO) == 1:

stopTime = time.time()

TimeElapsed = stopTime - startTime

distance(TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == “\_\_main\_\_”:

try:

while true:

dist = distance()

print(“Measured Distance = %lf cm” %dist)

if (dist<10):

lights.amber.on()

time.sleep(3)

lights.off()

lights.green.on()

time.sleep(3)

lights.off()

else:

lights.red.on()

time.sleep(3)

lights.off()

time.sleep(5)

except Keyboard Interrupt

print(“Measurement stopped by user”)

GPIO.cleanup()

**Result:** Traffic light system is demonstrated.

***Lab Experiment-8***

**AIM:** To implement smoke/flame detection using Raspberry Pi.

**DESCRIPTION:** When it is smoking in the surroundings, the smoke sensor detects the level of the smoke in the atmosphere and after reaching a certain threshold level it it prints that the smoke is detected.

**HARDWARE REQUIREMENTS:**

•Raspberry Pi 3 B+ Module

•Breadboard

•Smoke sensor

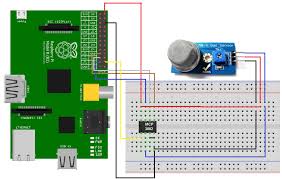
•Connecting wires

**SOFTWARE REQUIREMENTS:**

•Raspbian OS

•Python

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

import time  
import botbook\_mcp3002 as mcp #  
  
smokeLevel= 0  
  
def readSmokeLevel():  
 global smokeLevel  
 smokeLevel= mcp.readAnalog()  
  
def main():  
 while True: #  
 readSmokeLevel() #  
 print ("Current smoke level is %i " % smokeLevel) #  
 if smokeLevel > 120:  
 print("Smoke detected")  
 time.sleep(0.5) # s  
  
if\_name\_=="\_main\_":  
main()

**RESULT:** Current smoke level is 600.

Smoke detected.

***Lab Experiment-9***

**AIM:** To implement soil quality with soil sensor using Raspberry Pi.

**DESCRIPTION:** When the moisture level in the soil is less, the soil sensor does not detect water ans when the soil is wet it detects water.

**HARDWARE REQUIREMENTS:**

•Raspberry Pi 3 B+ Module

•Breadboard

•Soil sensor

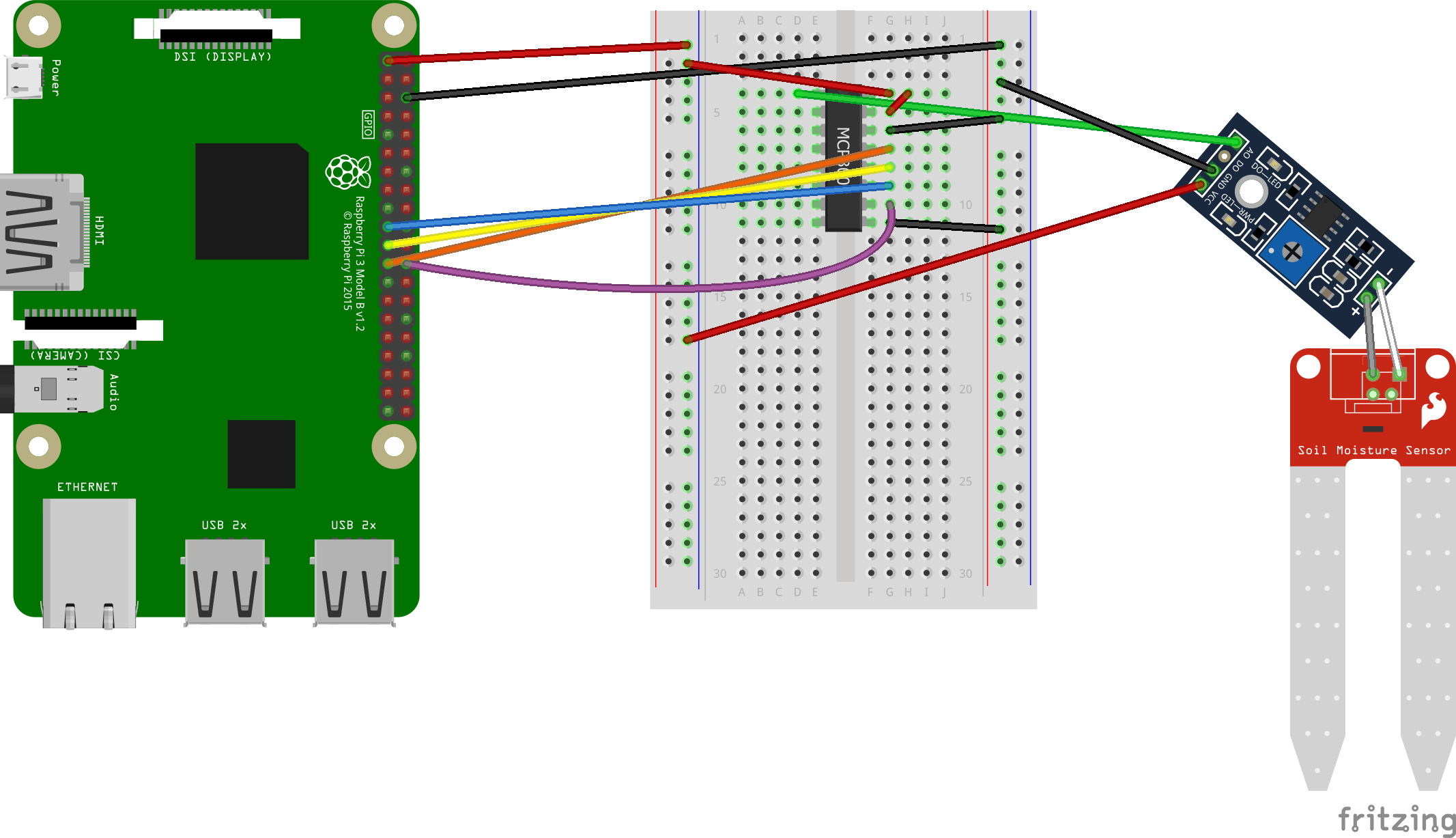
•Connecting wires

**SOFTWARE REQUIREMENTS:**

•Raspbian OS

•Python

**BOARD/CONNECTION DIAGRAM:**

****

**CODE:**

import RPi.GPIO as GPIO

from gpiozero import InputDevice

from time import sleep

GPIO.setmode(GPIO.BCM)

inp = InputDevice(7)

while True:

if(inp.is\_active):

print(“Water not detected”)

sleep(1)

else:

print(“Water detected”)

sleep(1)

**RESULT:** Water not detected.

Water not detected.

Water detected. **[When water is added to dry soil and soil sensor is inserted.]**

Water detected.