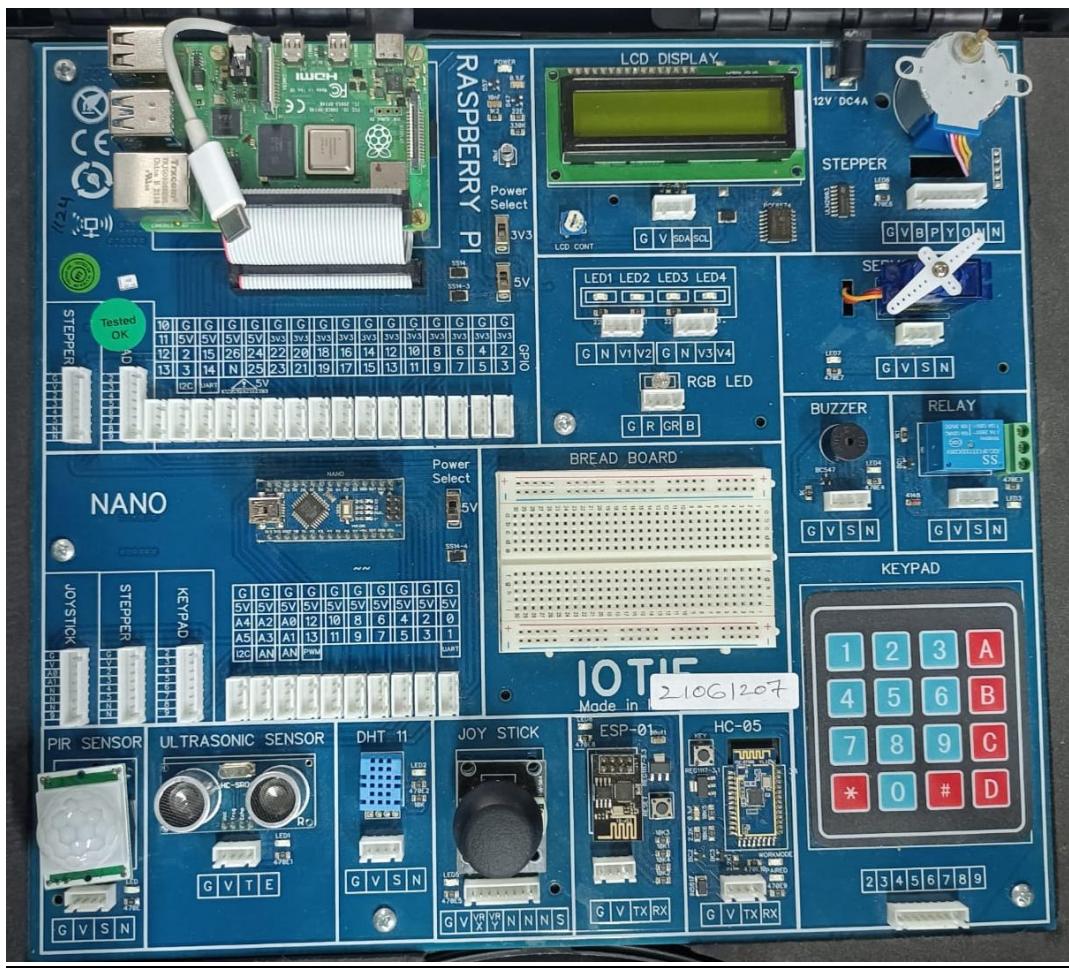


# IoT Lab

# Manual



# **INDEX**

<b>S.No</b>	<b>Title</b>	<b>Page No.</b>
1.	Install IDE of Arduino and write a program using Arduino IDE to blink LED.	3-6
2.	Interface LED and buzzer with Arduino to buzz for a period of time.	6-7
3.	Interface RGB LED with Aurdino to obtain different colours and brightness using PWM.	7-16
4.	a) Control a servo motor using Arduino with an input given through a push button (e.g: When the push button is pressed the servo motor has to rotate by 15 degrees). b) Rotate Stepper motor either clockwise or anti clockwise at 'n' number of steps using Arduino.	16-21
5.	Control any two actuators connected to the Arduino using Bluetooth/Wifi.	22-25
6.	Interface analog/digital sensors with Arduino and analyse the corresponding readings. (Sensors like temperature, alcohol, humidity, pressure, gas, sound pollution, level, weight, flow, proximity, LDR, PIR, pulse, vibration, sound etc..)	26-33
7.	Installation and setup of Raspberry Pi.	34-49
8.	Interface RGB LED with Raspberry Pi to obtain different colours and brightness using PWM.	49-54
9.	a) Interface an ultrasonic sensor with Raspberry pi to print distance readings on the monitor when the sensor changes its position. b) Reading the data from an analog sensor with Raspberry using Arduino serial port or ADC MCP3208 using SPI.	55-58
10.	Post/read the data to/from the cloud via MQTT broker with a Raspberry Pi.	59-60
11.	Send real-time sensor data to a smartphone using Raspberry Pi onboard Bluetooth	61-64
12.	Implement an intruder alert system that alerts through email	65-68

## WEEK-1

Install IDE of Arduino and write a program using Arduino IDE to blink LED.

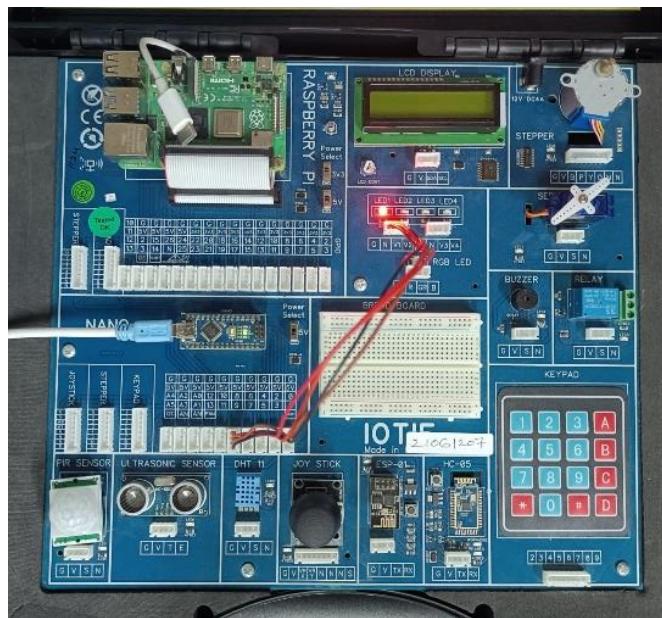
1.

**AIM:** To implement a program to blink LED every second.

**PROGRAM:**

```
void setup(){  
    pinMode(8,OUTPUT);  
}  
  
void loop(){  
    digitalWrite(8,1);  
    delay(1000);  
    digitalWrite(8,0);  
    delay(1000);  
}
```

**CIRCUIT:**



2.

**AIM:** To implement a program to blink 2 LEDs every second alternatively.

**PROGRAM:**

```
void setup(){  
    pinMode(8,OUTPUT);  
}
```

```

pinMode(9,OUTPUT);
}

void loop(){
    digitalWrite(8,1);
    digitalWrite(9,0);
    delay(1000);
    digitalWrite(8,0);
    digitalWrite(9,1);
    delay(1000);
}

```

### CIRCUIT:



### 3.

**AIM:** To implement a program to blink 2 LEDs every second alternatively for 10 times.

### PROGRAM:

```

void setup(){
    pinMode(8,OUTPUT);
    pinMode(9,OUTPUT);
    int i=0;
}

void loop(){
    while(i<10){
        digitalWrite(8,1);
        digitalWrite(9,0);
        delay(1000);
    }
}

```

```

digitalWrite(8,0);
digitalWrite(9,1);
delay(1000);
i++;
}
}

```

### CIRCUIT:



### 4.

**AIM:** To implement a program to change the intensity of LED.

### PROGRAM:

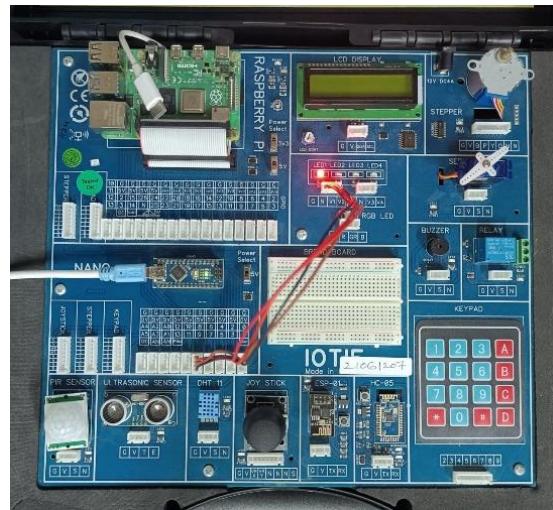
```

void setup()
{
    pinMode(8,OUTPUT);
}

void loop()
{
    for(int i=0;i<=255;i++)
    {
        analogWrite(8,i);
        delay(100);
    }
    for(int i=255;i>0;i--)
    {
        analogWrite(8,i);
        delay(100);
    }
}

```

## CIRCUIT:



## WEEK-2

Interface LED and buzzer with Arduino to buzz for a period of time.

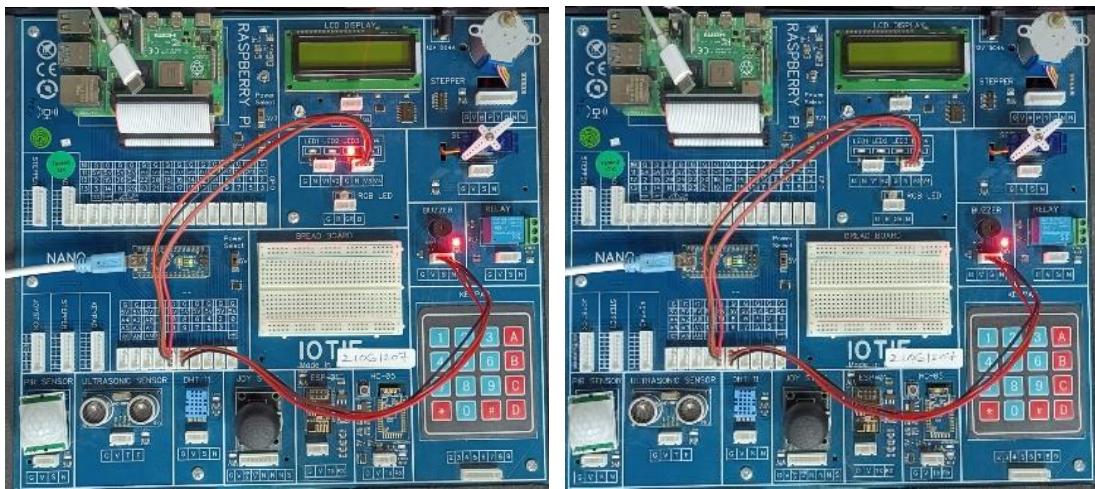
### 1.

**AIM:** To implement a program to switch LED and buzzer ON every second.

### PROGRAM:

```
void setup(){  
    pinMode(8,OUTPUT);  
    pinMode(9,OUTPUT);  
}  
  
void loop(){  
    digitalWrite(8,1);  
    digitalWrite(9,1);  
    delay(1000);  
    digitalWrite(8,0);  
    digitalWrite(9,0);  
    delay(1000);  
}
```

## CIRCUIT:



## WEEK-3

Interface RGB LED with Arduino to obtain different colours and brightness using PWM

### 1.

**AIM:** To implement a program to obtain red, blue and green colors of RGB LED with user input.

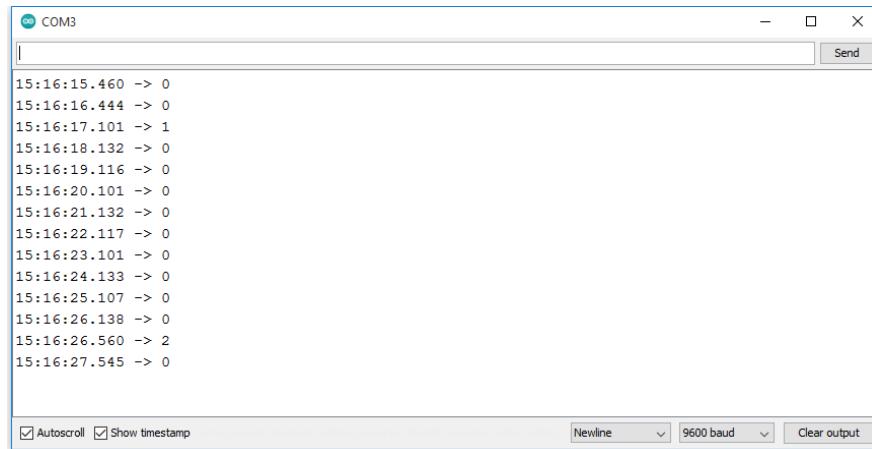
### PROGRAM:

```
int c;  
void setup() {  
    Serial.begin(9600);  
    pinMode(10,OUTPUT);  
    pinMode(8,OUTPUT);  
    pinMode(9,OUTPUT);  
}  
  
void loop()  
{  
    c = Serial.parseInt();  
    if(c==1) {  
        digitalWrite(8,1);  
        digitalWrite(9,0);  
        digitalWrite(10,0);  
    }  
    else if(c==2) {  
        digitalWrite(9,1);  
        digitalWrite(8,0);  
        digitalWrite(10,0);  
    }  
}
```

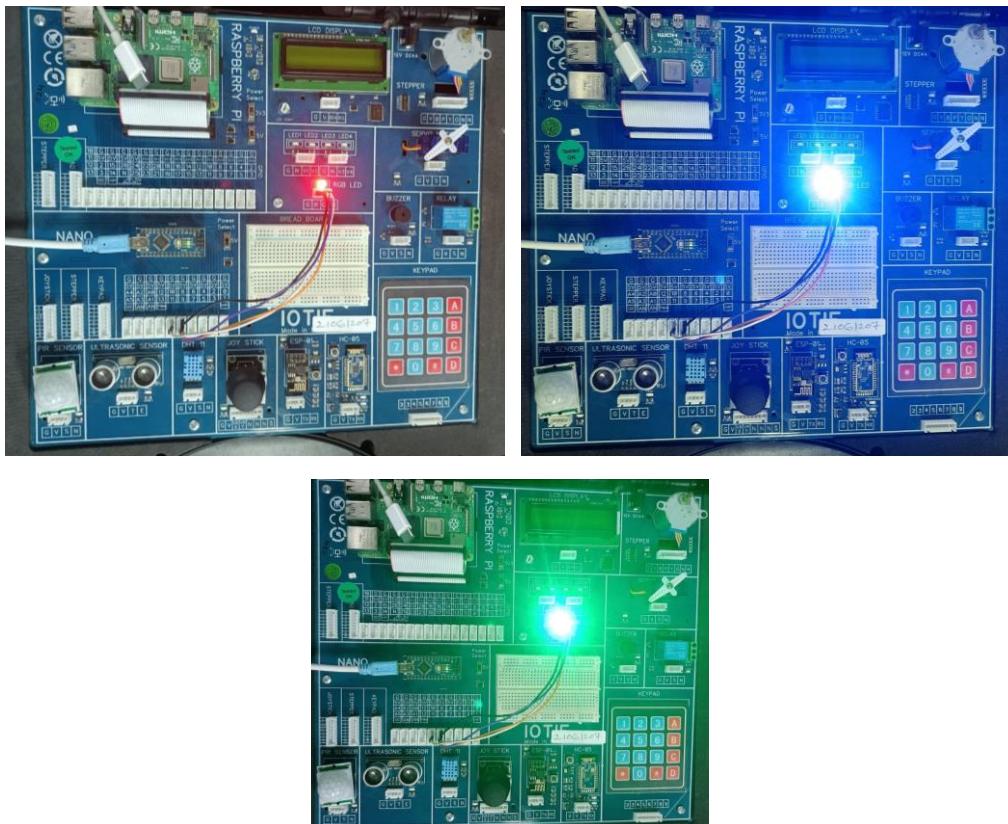
```

    }
else if(c==3) {
    digitalWrite(10,1);
    digitalWrite(8,0);
    digitalWrite(9,0);
}
Serial.println(c);
}

```



## CIRCUIT:



## 2.

**AIM:** To implement a program to obtain different colors from RGB LED through user input.

### PROGRAM:

```
String color = "";
void setup()
{
    Serial.begin(9600);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}
void loop()
{
    color = Serial.readString();
    color.trim();

    if(color == "red" || color == "RED") {
        rgb(255,0,0);
    }
    if(color == "green" || color == "GREEN") {
        rgb(0,0,255);
    }
    if(color == "blue" || color == "BLUE") {
        rgb(0,255,0);
    }
    if(color == "yellow" || color == "YELLOW") {
        rgb(255,0,255);
    }
    if(color == "cyan" || color == "CYAN") {
        rgb(0,255,255);
    }
    if(color == "magenta" || color == "MAGENTA") {
        rgb(255,255,0);
    }
    if(color == "purple" || color == "PURPLE") {
        rgb(128,128,0);
    }
    if(color == "navyblue" || color == "NAVYBLUE") {
        rgb(0,128,0);
    }
    if(color == "brown" || color == "BROWN") {
        rgb(165,42,42);
    }
    if(color == "sage" || color == "SAGE") {
        rgb(178,136,172);
    }
}
```

```

        }
    }

void rgb(int x, int y, int z)
{
    analogWrite(9,x);
    analogWrite(10,y);
    analogWrite(11,z);
}

```

## CIRCUIT:



### 3.

**AIM:** To implement a program to change the intensity of RGB LED.

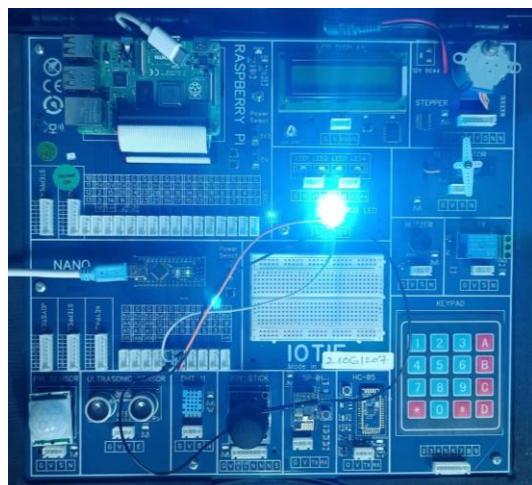
#### **PROGRAM:**

```
void setup()
{
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);

}

void loop()
{
    for(int i = 0; i <=255; i++)
    {
        analogWrite(9,i);
        analogWrite(10,i);
        analogWrite(11,i);
        delay(100);
    }
    for(int i = 255; i >=0; i--)
    {
        analogWrite(9,i);
        analogWrite(10,i);
        analogWrite(11,i);
        delay(100);
    }
}
```

#### **CIRCUIT:**



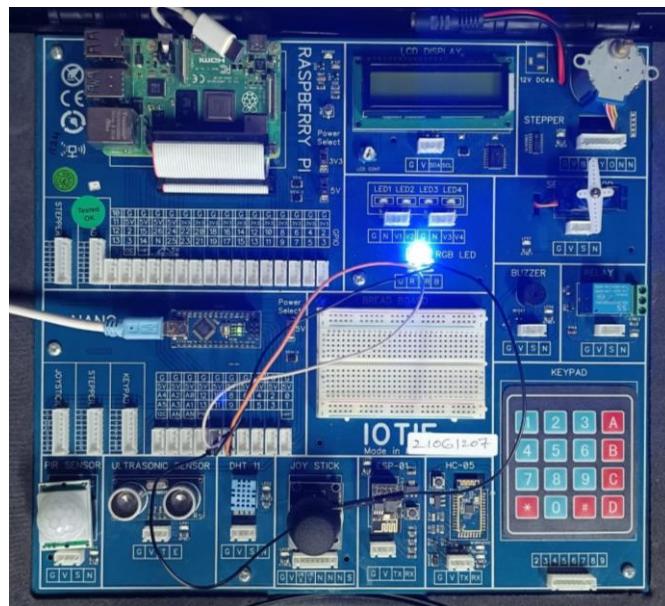
#### 4.

**AIM:** To implement a program to increase the intensity of RGB LED.

#### PROGRAM:

```
void setup(){}
void loop()
{
    for(i=0;i<=255;i++)
    {
        for(j=0;j<=255;j++)
        {
            for(k=0;k<=255;k++)
            {
                RGB(i,j,k);
                delay(100);
            }
        }
    }
}
void RGB(int R, int G, int B)
{
    analogWrite(9, R);
    analogWrite(10,G);
    analogWrite(11,B);
}
```

#### CIRCUIT:



## 5.

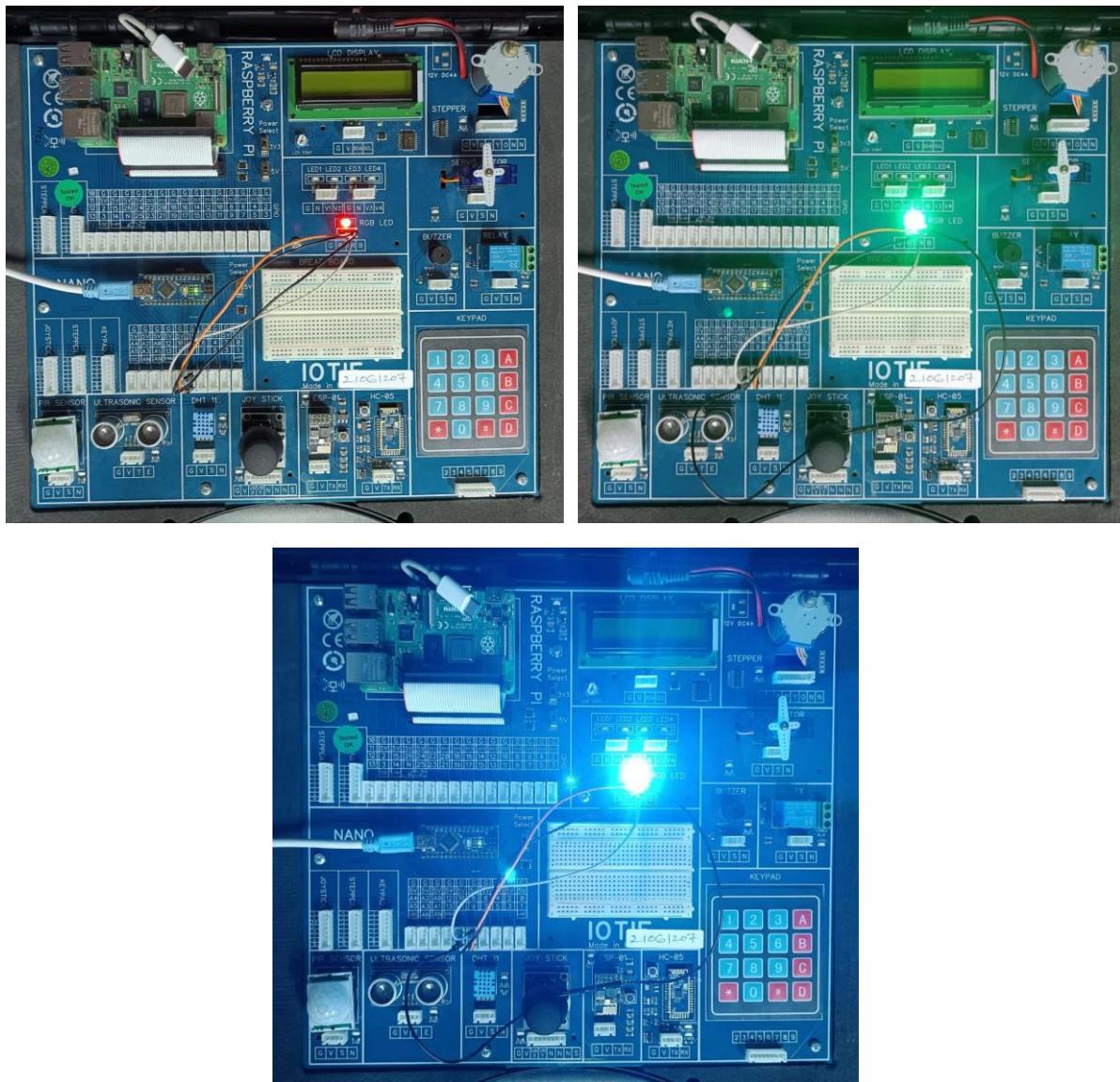
**AIM:** To implement a program to first increase the intensity of RGB LED in the order - green, red and blue.

### PROGRAM:

```
int i;
void setup() {}
void loop()
{
    for(int i=0;i<=255;i++)
    {
        RGB(0,i,0);
        delay(100);
    }
    for(int i=0;i<=255;i++)
    {
        RGB(i,0,0);
        delay(100);
    }
    for(int i=0;i<=255;i++)
    {
        RGB(0,0,i);
        delay(100);
    }
}

void RGB()
{
    analogWrite(9,R);
    analogWrite(10,G);
    analogWrite(11,B);
}
```

## CIRCUIT:



## 6.

**AIM:** To implement a program to switch ON any one of the 4 LEDs

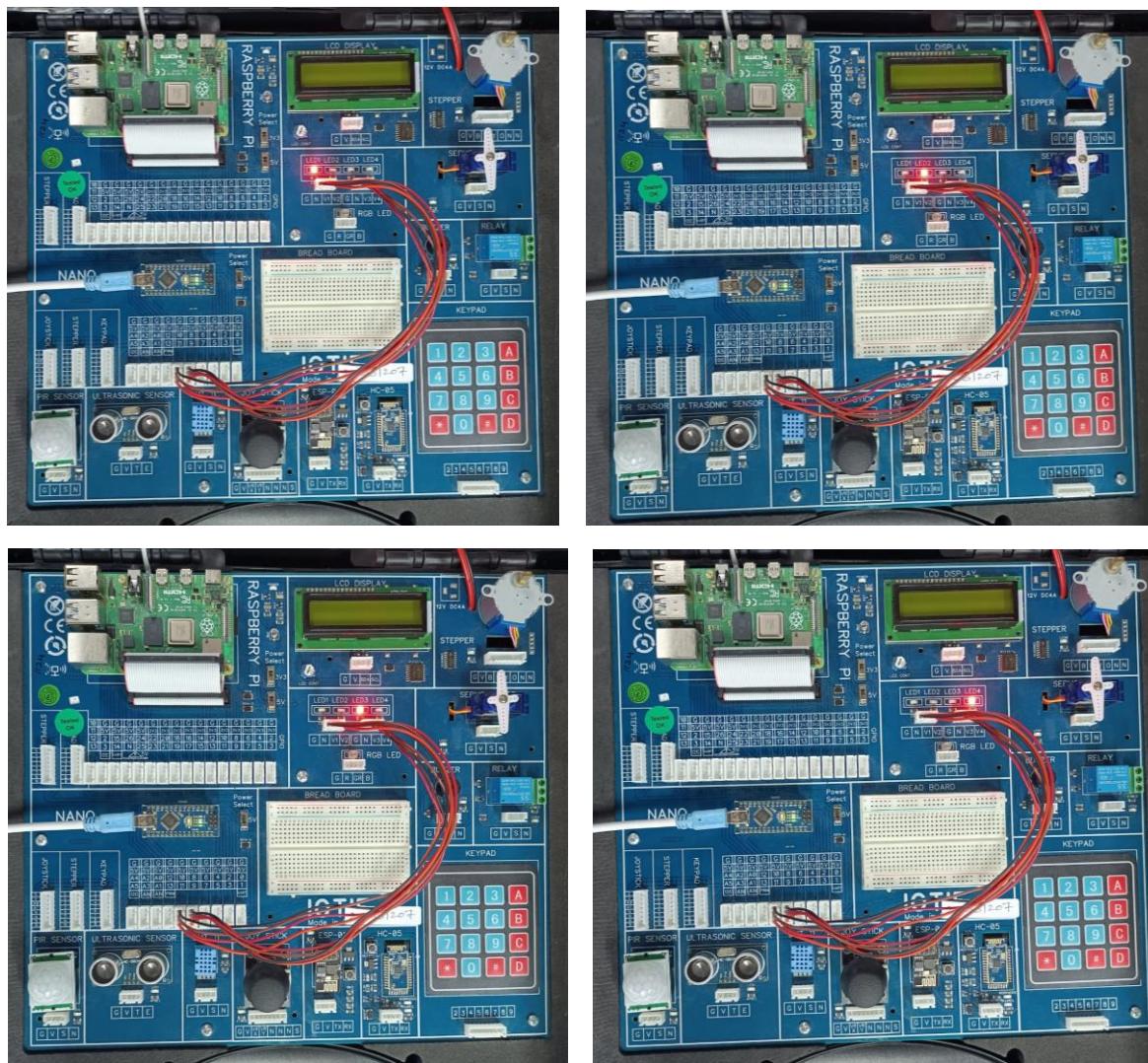
### PROGRAM:

```
int c;  
  
void setup(){  
    Serial.begin(9600);  
    pinMode(8,OUTPUT);  
    pinMode(9,OUTPUT);  
    pinMode(10,OUTPUT);  
    pinMode(11,OUTPUT);
```

```
}

void loop(){
    c=Serial.parseInt();
    if(c==1){
        digitalWrite(8,1);
        digitalWrite(9,0);
        digitalWrite(10,0);
        digitalWrite(11,0);
    }
    else if(c==2){
        digitalWrite(8,0);
        digitalWrite(9,1);
        digitalWrite(10,0);
        digitalWrite(11,0);
    }
    else if(c==3){
        digitalWrite(8,0);
        digitalWrite(9,0);
        digitalWrite(10,1);
        digitalWrite(11,0);
    }
    else if(c==4){
        digitalWrite(8,0);
        digitalWrite(9,0);
        digitalWrite(10,0);
        digitalWrite(11,1);
    }
}
```

## CIRCUIT:



## WEEK-4

- a) Control a servo motor using Arduino with an input given through a push button (e.g: When the push button is pressed the servo motor has to rotate by 15 degrees). b) Rotate Stepper motor either clockwise or anti clockwise at 'n' number of steps using Arduino.

1.

**AIM:** To implement a program to rotate servo motor  $90^0$  for every 500ms.

### **PROGRAM:**

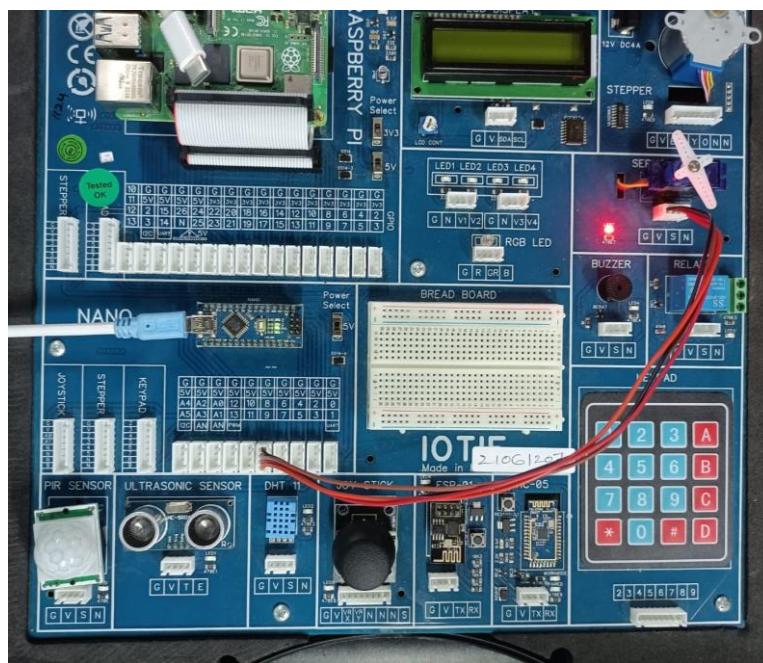
```
#include<Servo.h>
Servo s1;
int servopin=8;
```

```

void setup(){
    s1.attach(servopin);
}
void loop(){
    s1.write(90);
    delay(500);
}

```

## CIRCUIT:



## 2.

**AIM:** To implement a program to rotate servo motor  $180^0$  in clockwise direction.

### PROGRAM:

```

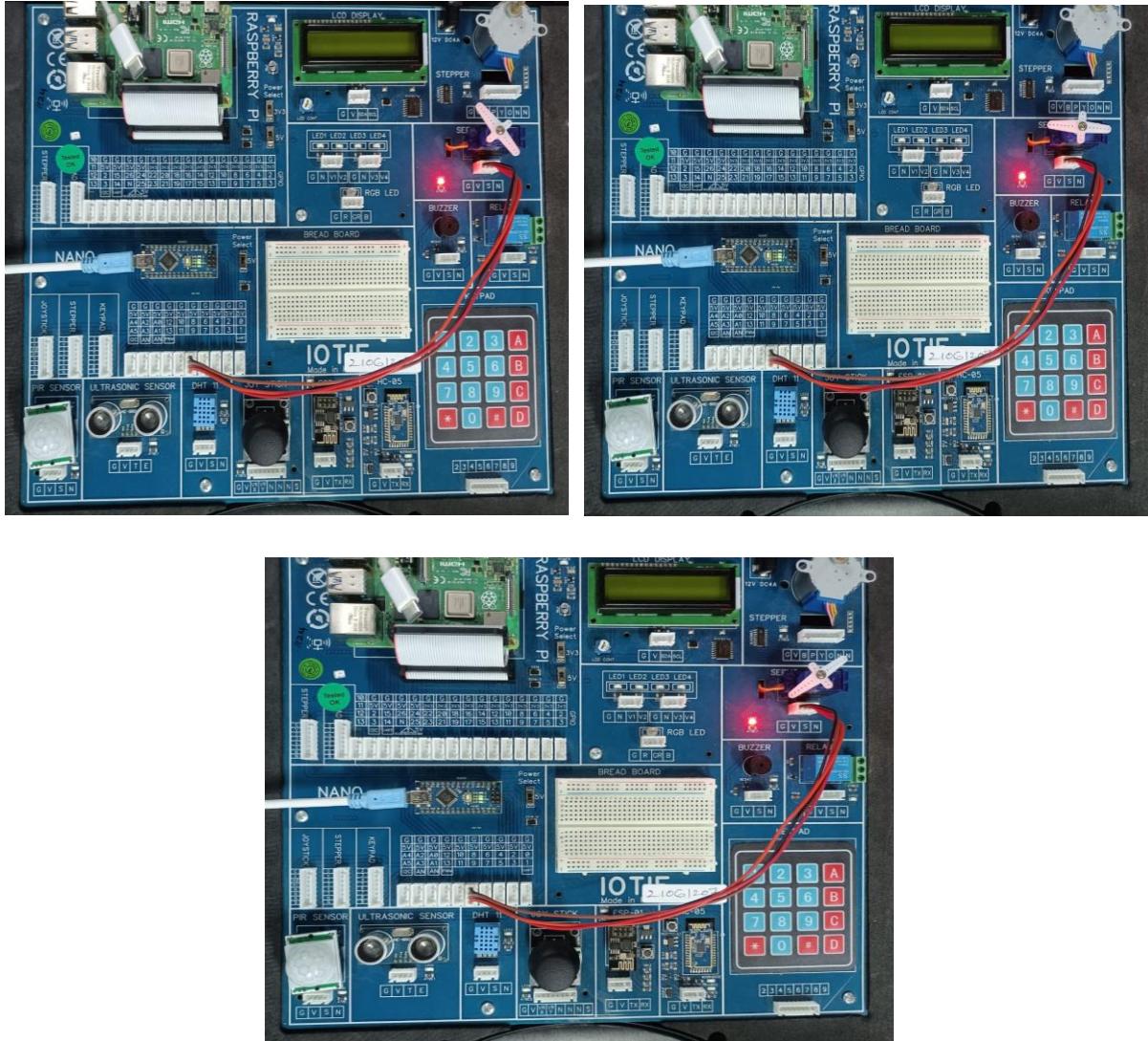
#include<Servo.h>
Servo s1;
int servopin=8;
void setup(){
    s1.attach(servopin);
}
void loop(){
    for(i=0;i<=180;i+=45){
        s1.write(i);
        delay(500);
    }
}

```

```
}
```

```
}
```

## CIRCUIT:



3.

**AIM:** To implement a program to rotating servo motor 180<sup>0</sup> in anti-clockwise direction.

### PROGRAM:

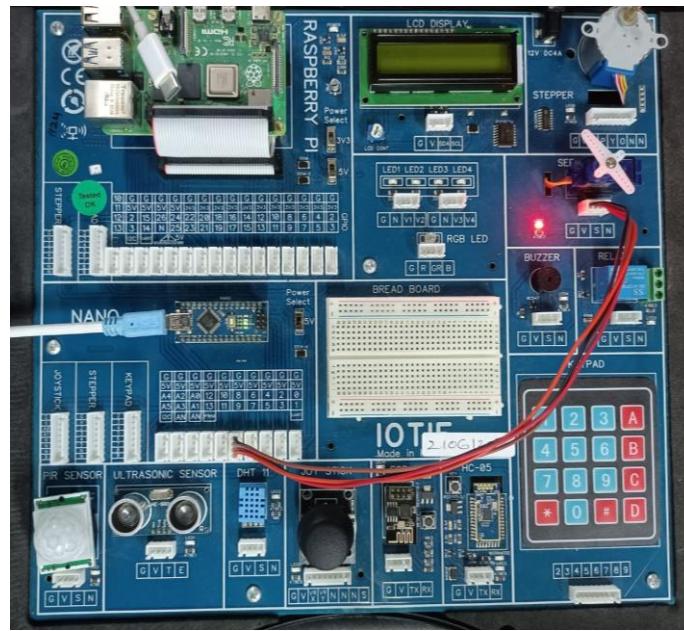
```
#include<Servo.h>
Servo s1;
int servopin=8;
void setup(){
    s1.attach(servopin);
}
```

```

void loop(){
for(i=180;i>=0;i--){
    s1.write(i);
    delay(500);
}
}

```

## CIRCUIT:



## 4.

**AIM:** To implement a program to rotate stepper motor  $360^0$  in clockwise and then in anti-clockwise direction.

### PROGRAM:

```

#include<Stepper.h>
int steps_per_rev=32;
int gear_reduction=64;
int steps_req=gear_reduction*steps_per_rev;
Stepper motor(steps_per_rev,2,4,3,5);
void setup(){
    Serial.begin(9600);
}
void loop(){
    Serial.println(steps_req);
}

```

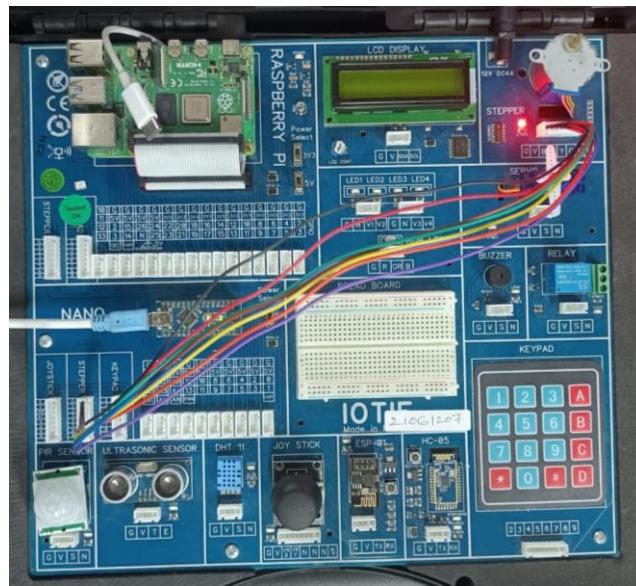
```

motor.setSpeed(900);
motor.step(steps_req);
delay(1000);
Serial.println(steps_req);
motor.setSpeed(900);
motor.step(-steps_req);
delay(1000);

}

```

## CIRCUIT:



## 5.

**AIM:** To implement a program to rotate stepper motor  $180^0$  in clockwise and then  $90^0$  in anti-clockwise direction.

### PROGRAM:

```

#include<Stepper.h>
int steps_per_rev=32;
int gear_reduction=64;
int steps_req=gear_reduction*steps_per_rev;
Stepper motor(steps_per_rev,2,4,3,5);
void setup(){
    Serial.begin(9600);
}
void loop(){

```

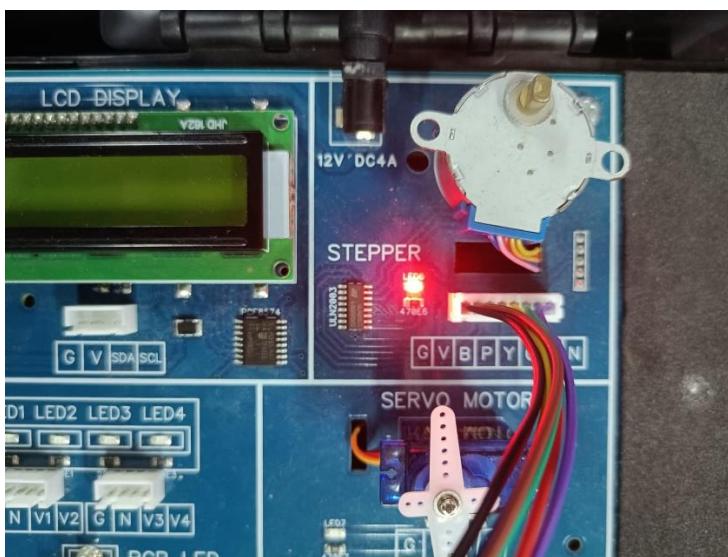
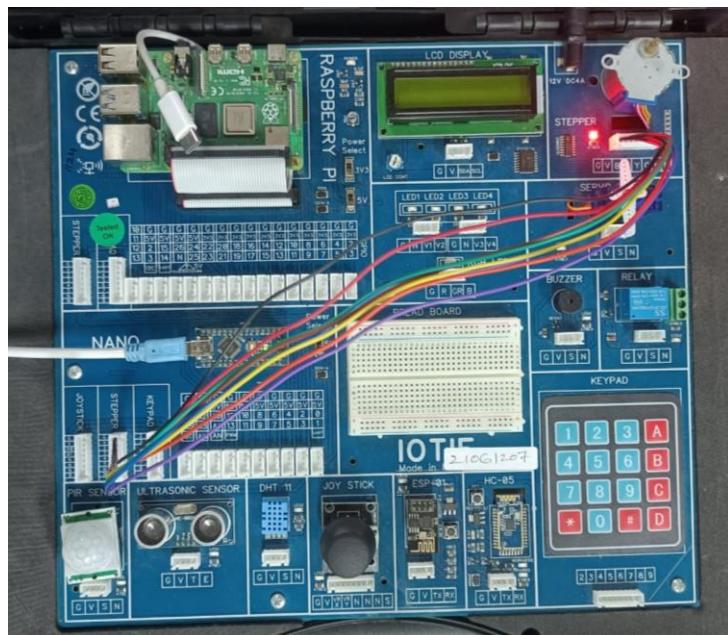
```

Serial.println(steps_req);
motor.setSpeed(900);
motor.step(steps_req/2);
delay(1000);
Serial.println(steps_req);
motor.setSpeed(900);
motor.step(-steps_req/4);
delay(1000);

}

```

## CIRCUIT:



## WEEK-5

Control any two actuators connected to the Arduino using Bluetooth/Wifi.

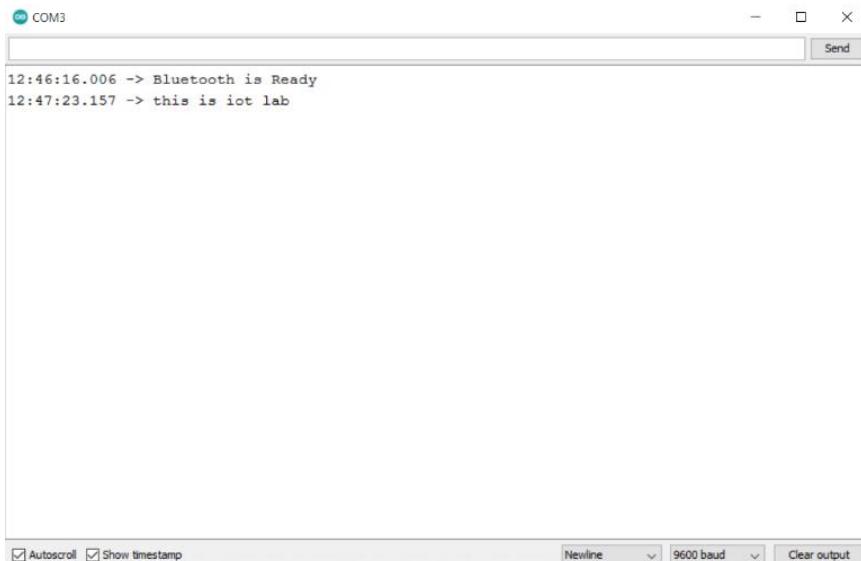
1.

**AIM:** To implement a program to send and receive messages using Bluetooth module with Arduino.

**PROGRAM:**

```
#include <SoftwareSerial.h>
SoftwareSerial EEBLUE(10,11);
void setup()
{
    Serial.begin(9600);
    EEBLUE.begin(9600);
    Serial.println("Bluetooth is ready ");
}
void loop()
{
    if(EEBLUE.available())
        Serial.write(EEBLUE.read());
    if(Serial.available())
        EEBLUE.write(Serial.read());
}
```

**OUTPUT:**



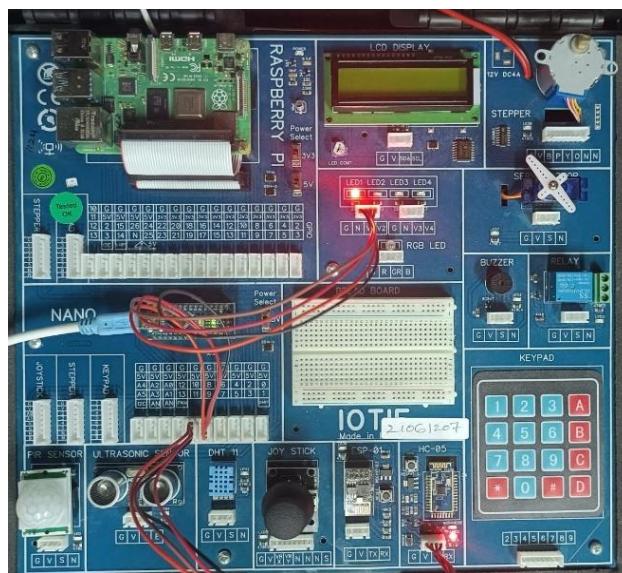
## 2.

**AIM:** To implement a program using Bluetooth to interface LED based on Bluetooth terminal input.

### PROGRAM:

```
#include <SoftwareSerial.h>
SoftwareSerial EEBLUE(10,11);
int input;
void setup()
{
    Serial.begin(9600);
    EEBLUE.begin(9600);
    Serial.println("BLUETOOTH IS READY");
    pinMode(8,OUTPUT);
}
void loop()
{
    if(EEBLUE.available()){
        input=EEBLUE.parseInt();
        if(input==1)
            digitalWrite(8,1);
        if(input==0)
            digitalWrite(8,0);
    }
}
```

### CIRCUIT:



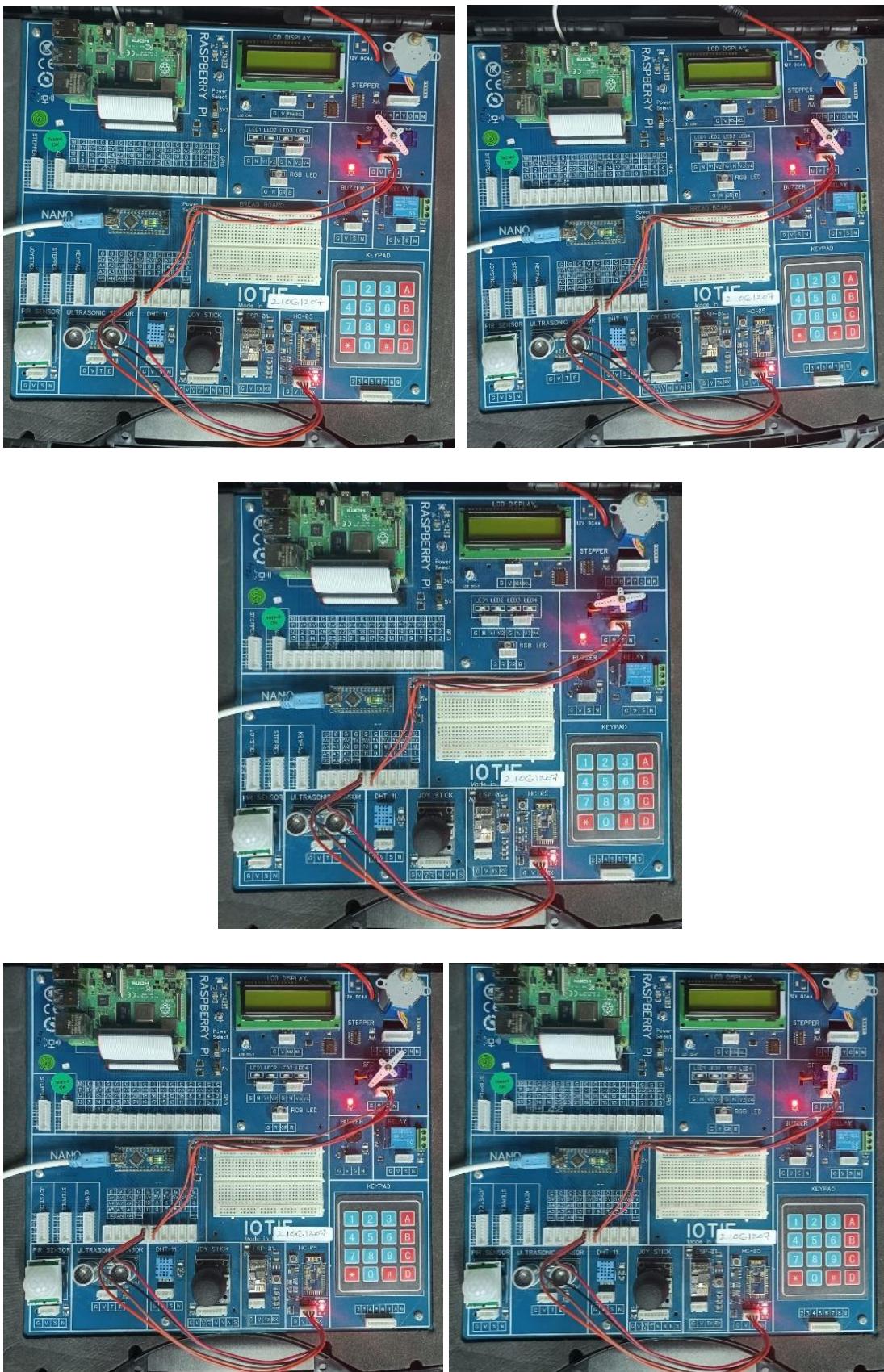
### 3.

**AIM:** To interface a servo motor using Bluetooth terminal.

#### **PROGRAM:**

```
#include <SoftwareSerial.h>
#include<Servo.h>
SoftwareSerial EEBLUE(10,11);
Servo s1;
int angle;
void setup()
{
    Serial.begin(9600);
    EEBLUE.begin(9600);
    Serial.println("BLUETOOTH IS READY");
    s1.attach(8);
}
void loop()
{
    if(EEBLUE.available()){
        angle=EEBLUE.parseInt();
        for(int i=0;i<=angle;i+=20){
            s1.write(i);
            delay(500);
        }
    }
}
```

## CIRCUIT:



## WEEK-6

Interface analog/digital sensors with Arduino and analyse the corresponding readings.  
(Sensors like temperature, alcohol, humidity, pressure, gas, sound pollution, level, weight, flow, proximity, LDR, PIR, pulse, vibration, sound etc..)

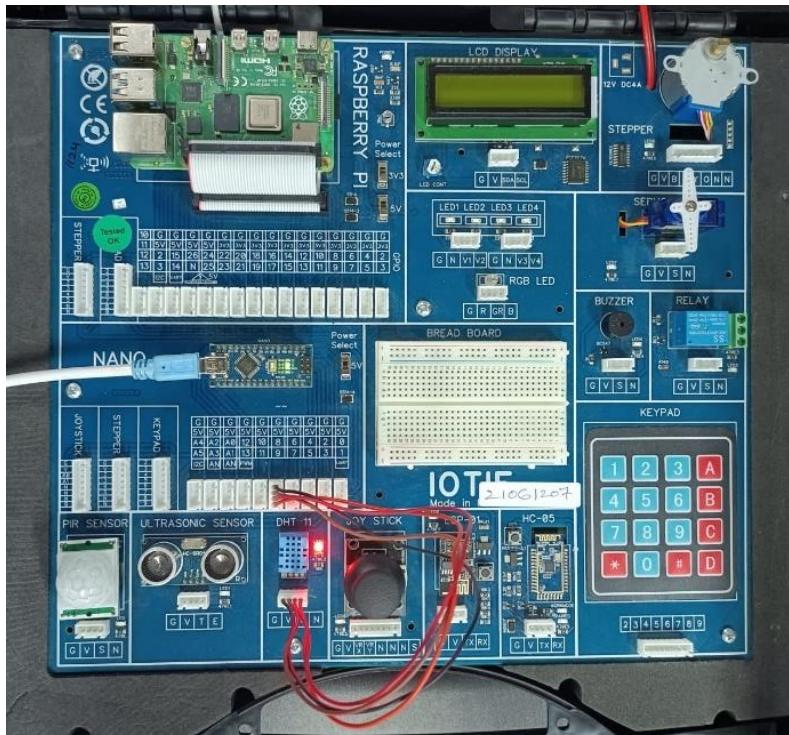
**1.**

**AIM:** To interface DHT11 sensor for measuring humidity and temperature.

**PROGRAM:**

```
#include<DHT.h>
DHT dht(8,DHT11)
float t,h;
void setup(){
    Serial.begin(9600);
    dht.begin();
    Serial.println("Starting DHT Test");
    delay(2000);
}
void loop(){
    h=dht.readHumidity();
    t=dht.readTemperature();
    if(isnan(h)||isnan(t)){
        Serial.println("Failed to read");
    }
    else{
        Serial.println(h);
        Serial.println(t);
    }
}
```

## CIRCUIT:



## OUTPUT:

```
15:03:27.542 -> 27.80
15:03:27.542 -> 63.00
15:03:27.589 -> 27.80
15:03:27.589 -> 63.00
15:03:27.589 -> 27.80
15:03:27.589 -> 63.00
15:03:27.589 -> 27.80
15:03:27.636 -> 63.00
15:03:27.636 -> 27.80
15:03:27.636 -> 63.00
15:03:27.636 -> 27.80
15:03:27.636 -> 63.00
15:03:27.636 -> 27.80
15:03:27.636 -> 63.00
15:03:27.636 -> 27.80
15:03:27.636 -> 63.00
```

2.

**AIM:** To interface PIR sensor for detecting motion of an object.

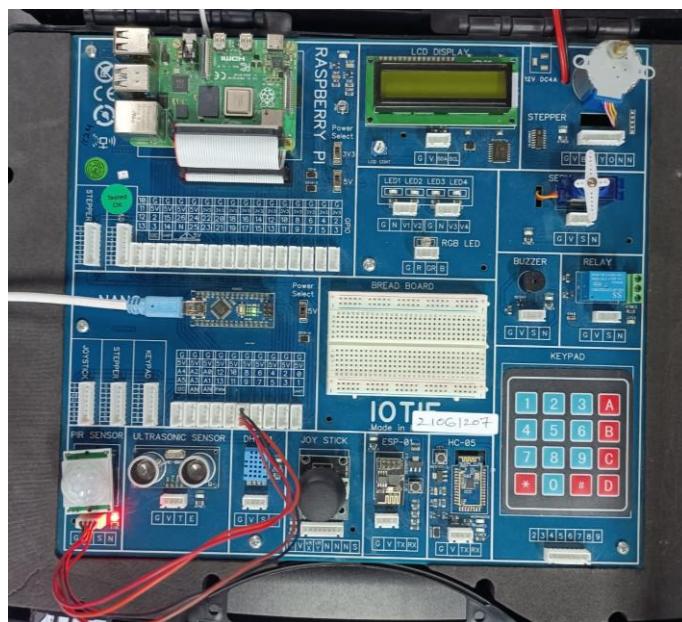
**PROGRAM:**

```

int sensordata;
void setup(){
    Serial.begin(9600);
    pinMode(13,OUTPUT);
    pinMode(8,INPUT);
}
void loop(){
    sesnsordata=digitalRead(8);
    if(sensordata==HIGH){
        digitalWrite(13,HIGH);
        Serial.println("Sensor Activated");
        Serial.print("Motion detected at");
        Serial.print(millis()/1000);
        Serial.println("secs");
        delay(50);
    }
    else{
        digitalWrite(13,LOW);
    }
    delay(50);
}

```

## CIRCUIT:



## OUTPUT:

```
15:06:46.697 -> Sensor Activated
15:06:46.697 -> Motion detected at 0secs
15:06:46.791 -> Sensor Activated
15:06:46.791 -> Motion detected at 0secs
15:06:46.885 -> Sensor Activated
15:06:46.885 -> Motion detected at 0secs
15:06:46.978 -> Sensor Activated
15:06:46.978 -> Motion detected at 0secs
15:06:47.072 -> Sensor Activated
15:06:47.072 -> Motion detected at 1secs
```

3.

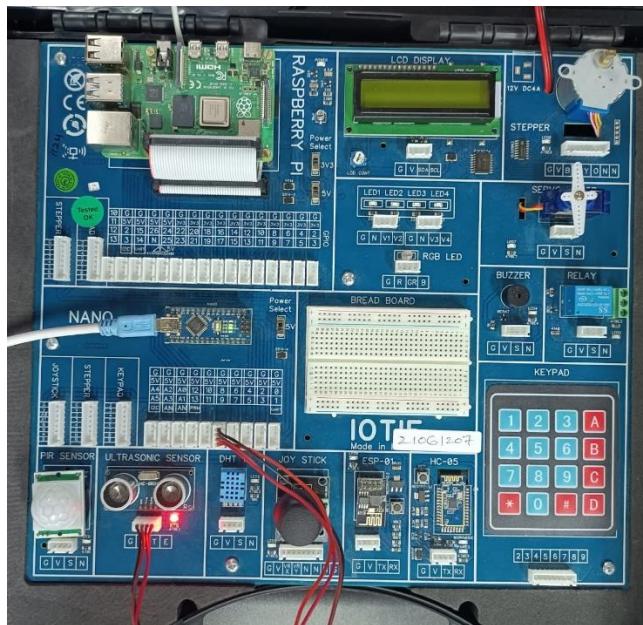
**AIM:** To interface ultrasonic sensor for detecting the presence and measuring the distance of an object.

## PROGRAM:

```
int distance,duration;
void setup(){
    Serial.begin(9600);
    pinMode(8,OUTPUT);
    pinMode(9,INPUT);
}
void loop(){
    digitalWrite(8,HIGH);
    delay(100);
    digitalWrite(8,LOW);
    duration=pulseIn(9,1);
    distance=duration*0.032/2;
    Serial.print("Distance");
    Serial.print(distance);
    Serial.println("cm");
```

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

## CIRCUIT:



## OUTPUT:

```
15:09:17.802 -> Distance147cm  
15:09:18.919 -> Distance147cm  
15:09:20.045 -> Distance148cm  
15:09:21.123 -> Distance147cm  
15:09:22.248 -> Distance18cm
```

4.

**AIM:** To implement a program to use joystick with serial monitor.

## PROGRAM:

```
int VRx = A0;  
int VRy = A1;
```

```

int xPosition = 0;
int yPosition = 0;
int SW_state = 0;
int mapX = 0;
int mapY = 0;

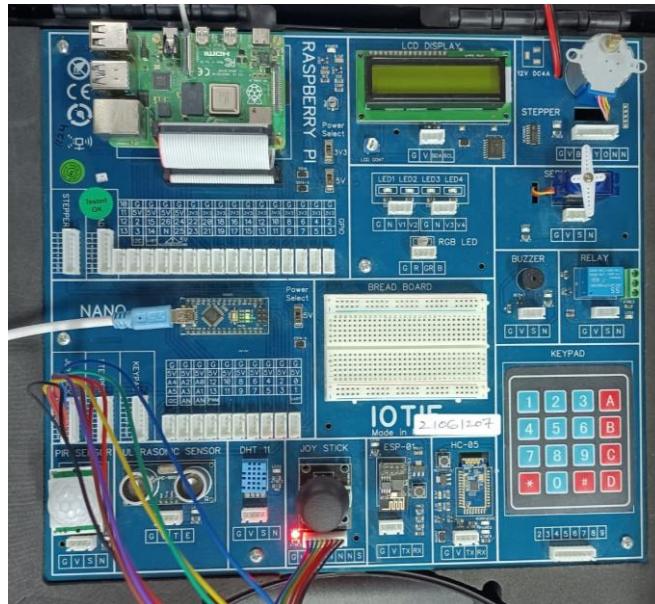
void setup() {
    Serial.begin(9600);
    pinMode(VRx, INPUT);
    pinMode(VRy, INPUT);
}

void loop() {
    xPosition = analogRead(VRx);
    yPosition = analogRead(VRy);
    mapX = map(xPosition, 0, 1023, -512, 512);
    mapY = map(yPosition, 0, 1023, -512, 512);

    Serial.print("X: ");
    Serial.print(mapX);
    Serial.print(" | Y: ");
    Serial.print(mapY);
    delay(100);
}

```

## CIRCUIT:



## OUTPUT:

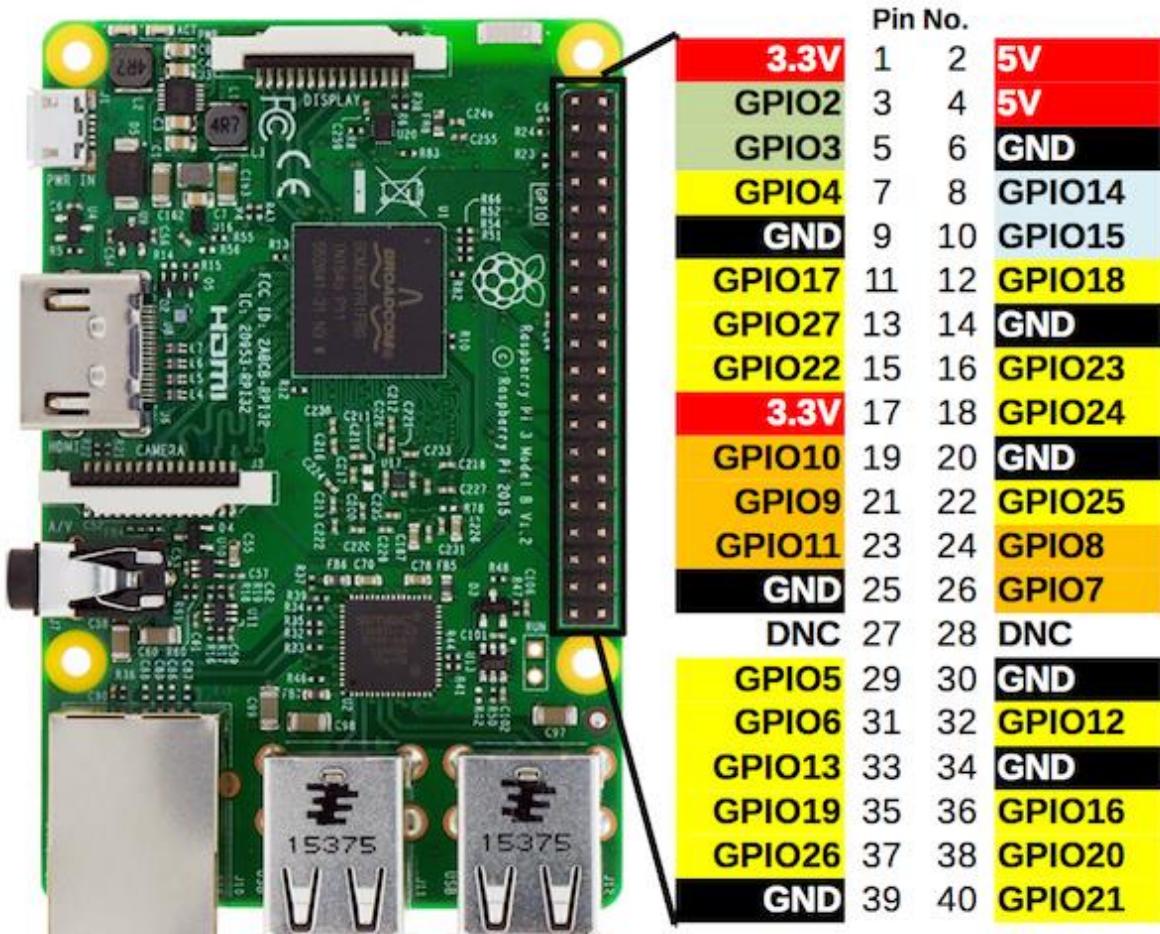
```

COM3
512 ->x 512 ->Y
336 ->x 512 ->Y
133 ->x 512 ->Y
111 ->x 512 ->Y
111 ->x 512 ->Y
112 ->x 512 ->Y
112 ->x 512 ->Y
112 ->x 512 ->Y
112

```

Autoscroll  Show timestamp    Newline    9600 baud    Clear output

## RASPBERRY PI



**AIM:** To implement a program to blink LED with raspberry pi.

**PROGRAM:**

```
import RPi.GPIO as GPIO
import time

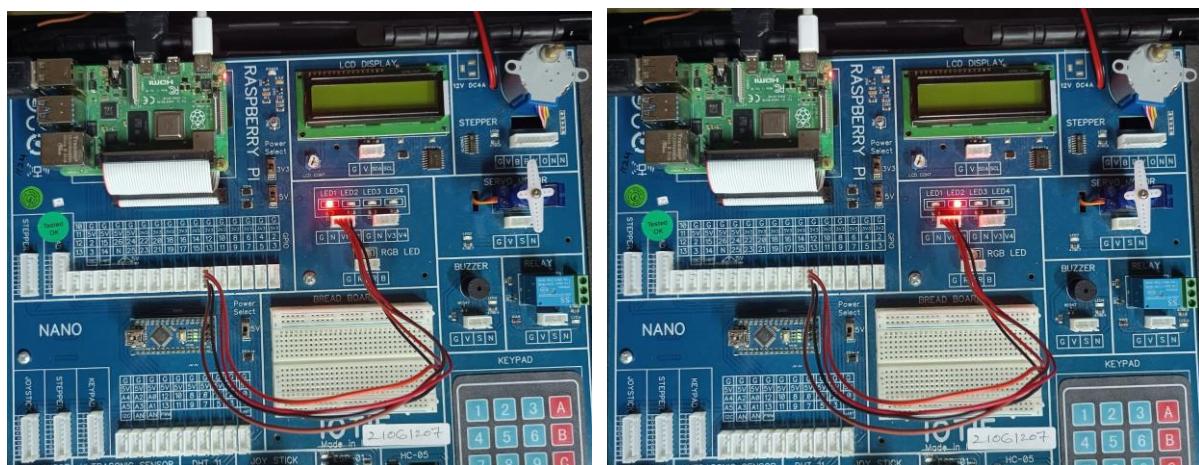
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

ledPinOne = 12
ledPinTwo = 13

GPIO.setup(ledPinOne, GPIO.OUT)
GPIO.setup(ledPinTwo, GPIO.OUT)

try:
    while True:
        GPIO.output(ledPinOne,HIGH)
        GPIO.output(ledPinTwo,LOW)
        time.sleep(100)
        GPIO.output(ledPinOne,LOW)
        GPIO.output(ledPinTwo,HIGH)
        time.sleep(100)
except KeyboardInterrupt:
    print("\nExiting Program\n")
    GPIO.cleanup()
    exit()
```

**CIRCUIT:**



## WEEK-7

### Installation and setup of Raspberry Pi.

pi imager windows - Google Search

pi imager windows

About 4,59,000 results (0.53 seconds)

Raspberry Pi Imager is the official software by the Raspberry Pi Foundation aimed to simplify recording operating system images compatible with the popular micro-computer onto your home unit. Specifically, it streamlines writing and saving OS software on external SD cards. 15-Apr-2020

<https://raspberry-pi-imager.en.uptodown.com/windows>

**Raspberry Pi Imager 1.2 for Windows - Download - Uptodown**

<https://www.raspberrypi.org/software>

**Raspberry Pi OS**

in a Terminal window. Screenshot of Raspberry Pi Imager. Manually install an operating system image. Browse a range ...

<https://www.raspberrypi.org/RaspberryPiBlog>

**Introducing Raspberry Pi Imager, our new imaging utility**

Flash Raspbian to your SD card with the Raspberry Pi Imager. ... use the new Raspberry Pi Imager, available for Windows, macOS and Ubuntu.

Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. [Watch our 45-second video](#) to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

**Download for Windows**

[Download for macOS](#)

[Download for Ubuntu for x86](#)

To install on **Raspberry Pi OS**, type  
`sudo apt install rpi-imager`  
in a Terminal window.

https://downloads.raspberrypi.org/imager/imager\_latest.exe

Raspberry Pi OS – Raspberry Pi

https://www.raspberrypi.org/software/

**Manually install an operating system image**

Browse a range of operating systems provided by Raspberry Pi and by other organisations, and download them to install manually.

**See all download options**

A diagram shows a microSD card with a red downward arrow pointing to it, indicating where the image should be written.

https://www.raspberrypi.org/software/operating-systems/

Internet access 6:51 PM 9/15/2021

Operating system images – Rasp

https://www.raspberrypi.org/software/operating-systems/

We use cookies to ensure that we give you the best experience on our websites. By continuing to visit this site you agree to our use of cookies. [Read our cookie policy.](#)

Got it!

Raspberry Pi logo

Hardware Software Books & magazines Learn Teach About us

## Operating system images

Many operating systems are available for Raspberry Pi, including Raspberry Pi OS, our official supported operating system, and operating systems from other organisations.

[Raspberry Pi Imager](#) is the quick and easy way to install an operating system to a microSD card ready to use with your Raspberry Pi. Alternatively, choose from the operating systems below, available to download and install manually.

Download:  
[Raspberry Pi OS \(32-bit\)](#)  
[Raspberry Pi Desktop](#)  
[Third-Party operating systems](#)

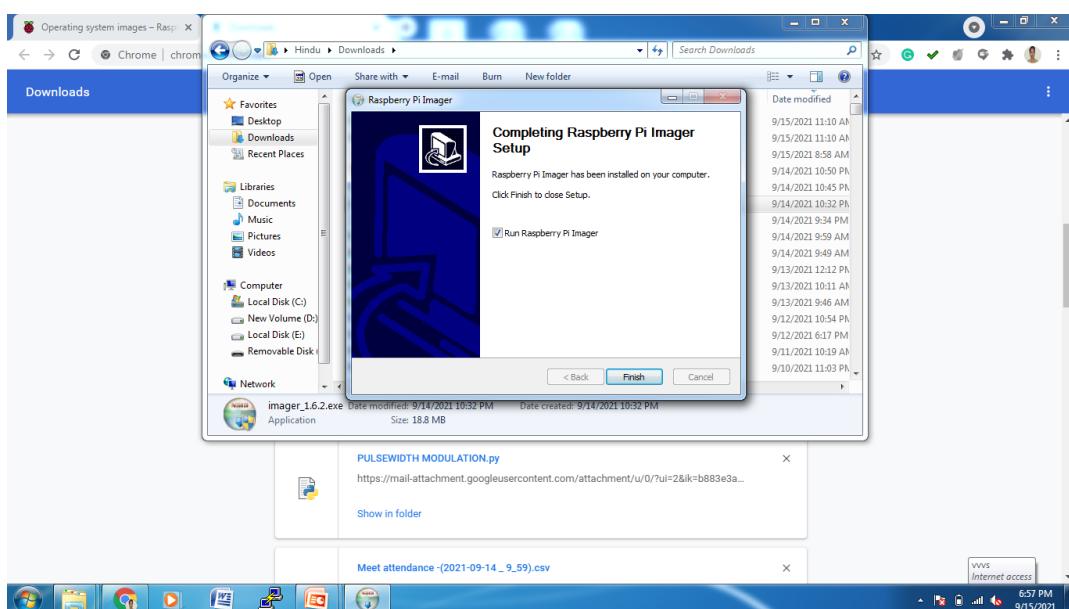
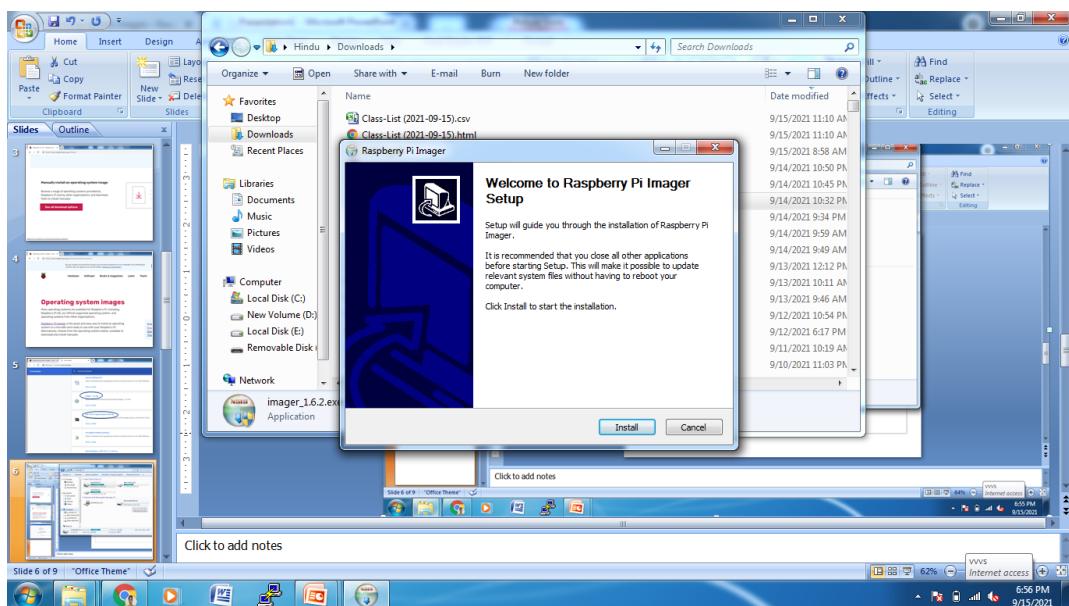
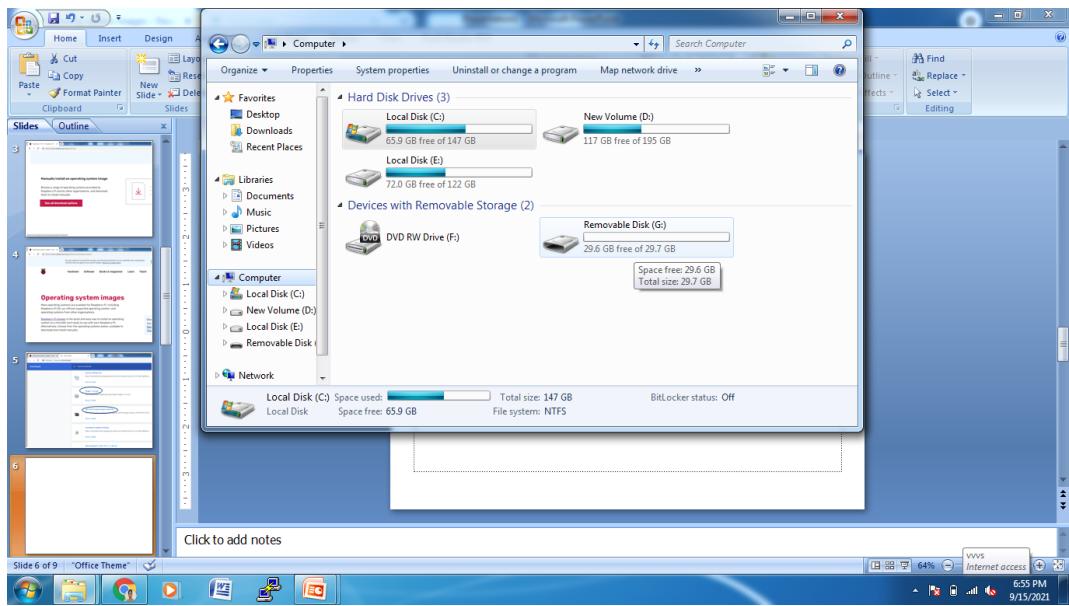
Operating system images – Rasp

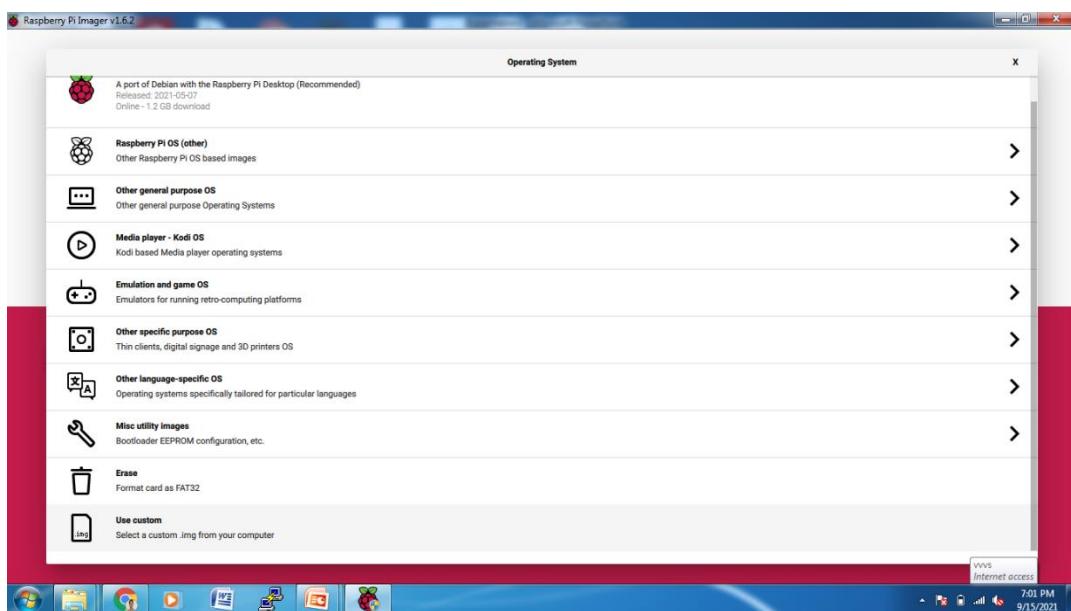
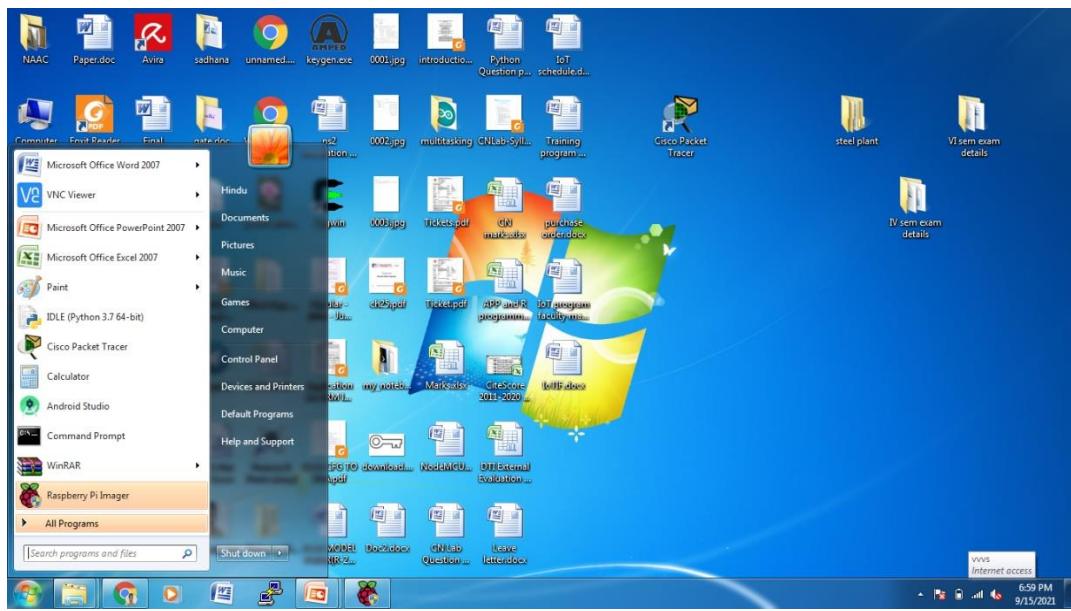
Downloads

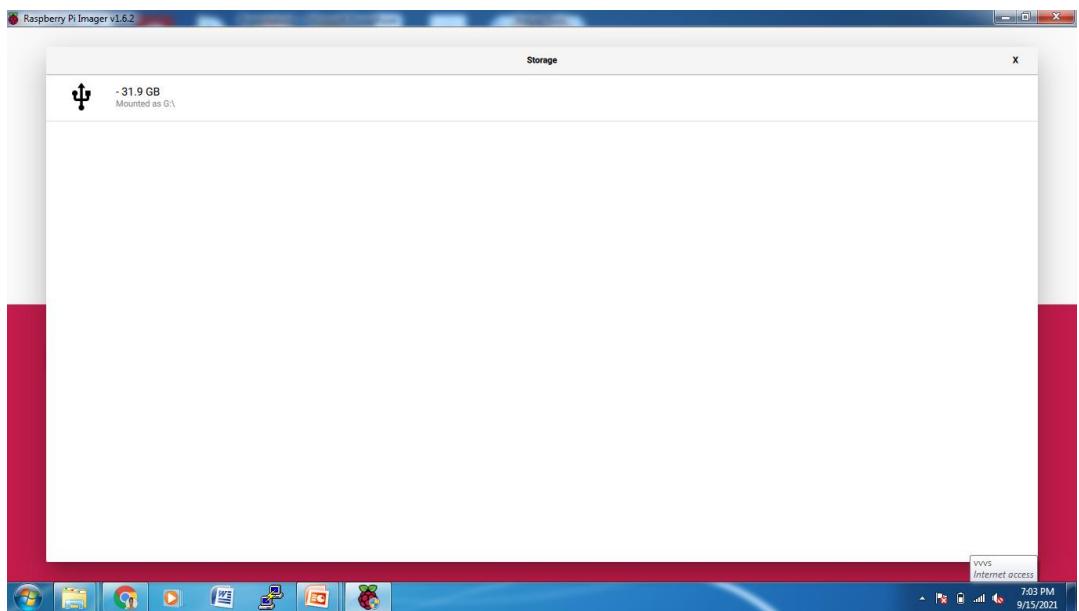
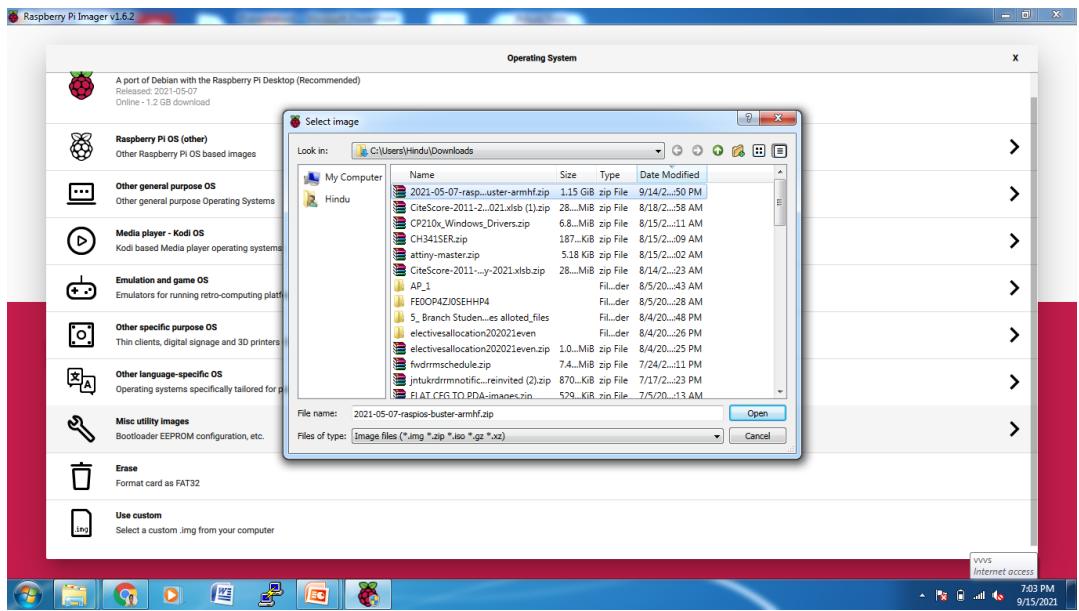
Search downloads

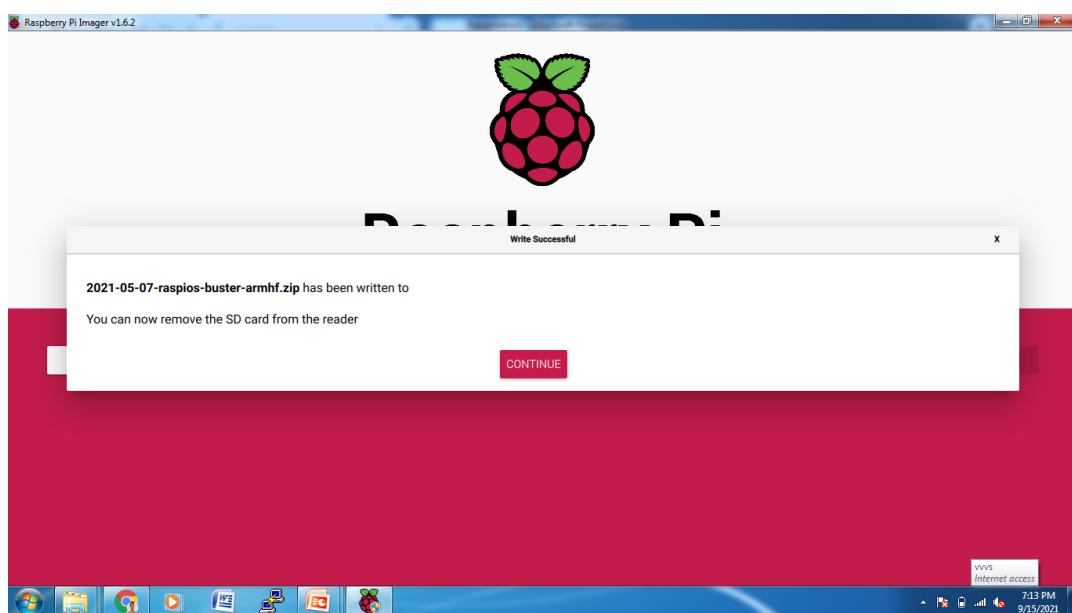
	pi proxy settings.docx https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=b883e3a...	X
	imager_1.6.2.exe https://downloads.raspberrypi.org/imager/imager_1.6.2.exe	X
	2021-05-07-raspios-buster-armhf.zip https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2021-05-28...	X
	PULSEWIDTH MODULATION.PY https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=b883e3a...	X
	Meet attendance -(2021-09-14 _ 9_59).csv	X

Internet access 6:52 PM 9/15/2021

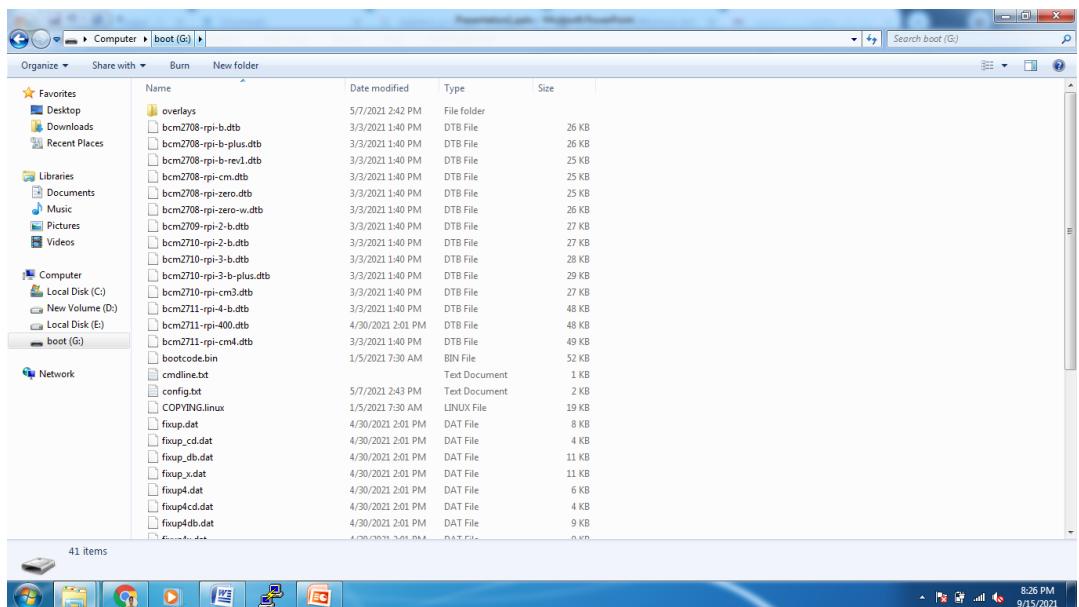
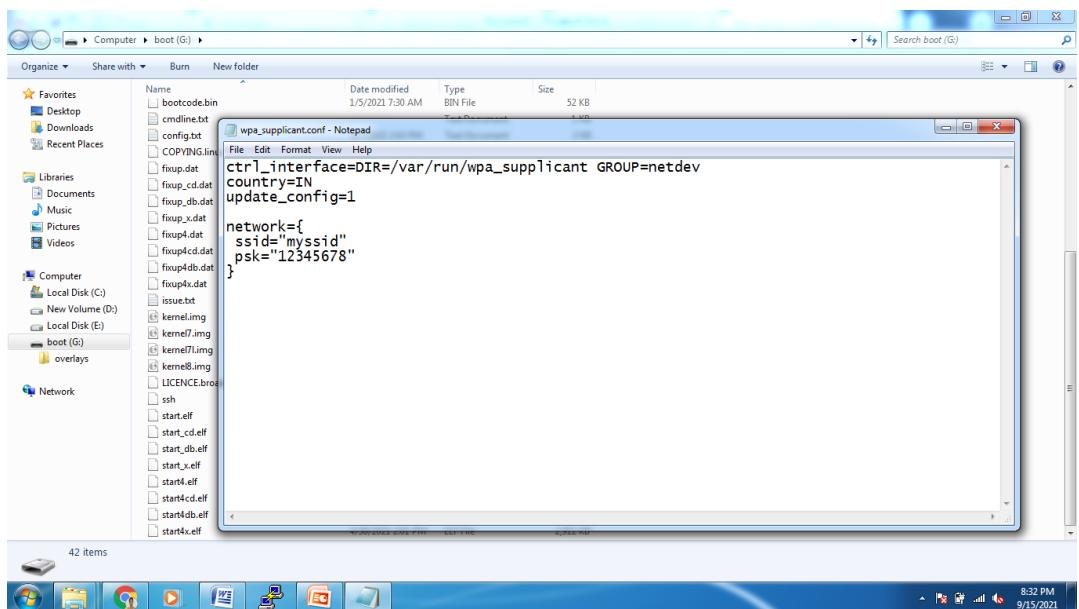
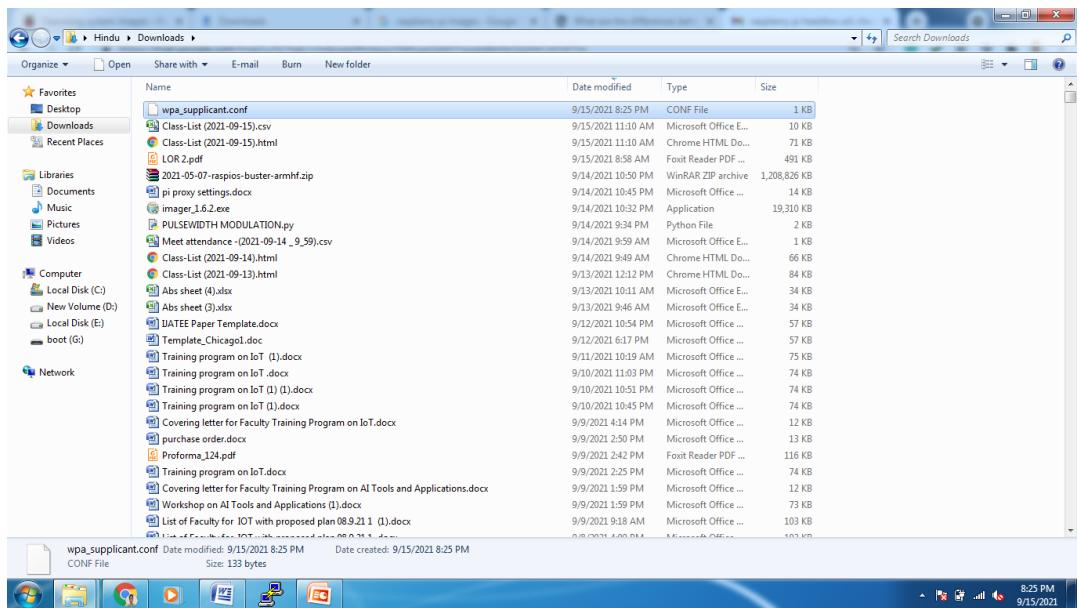


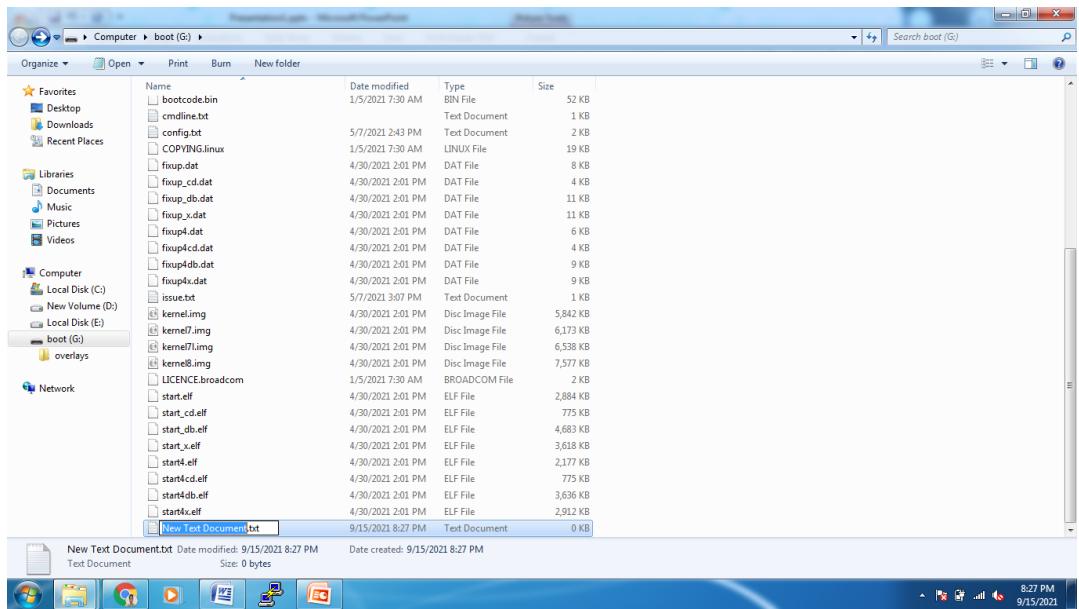
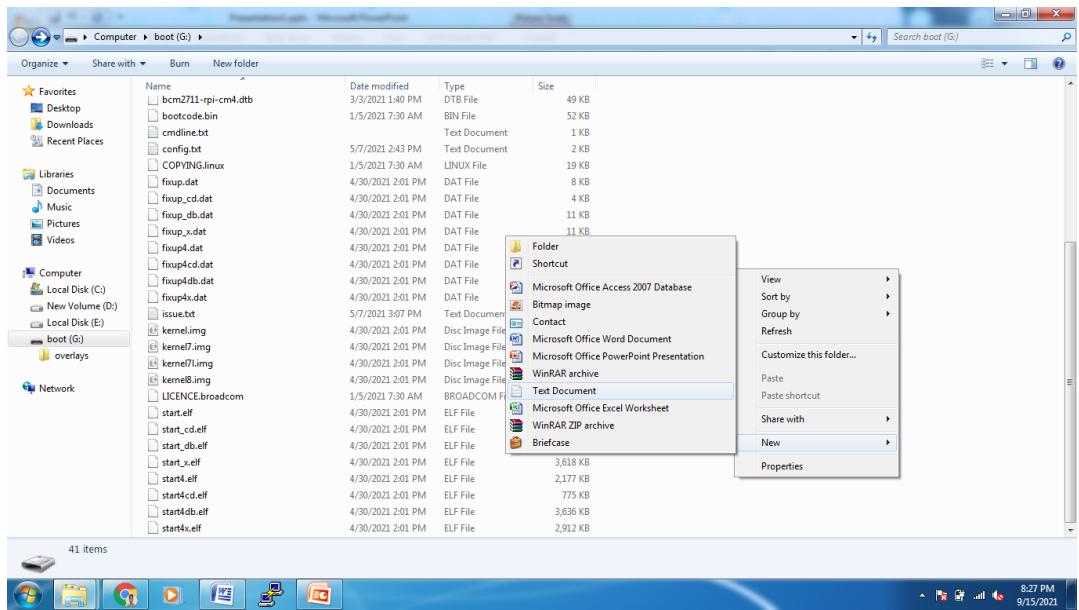
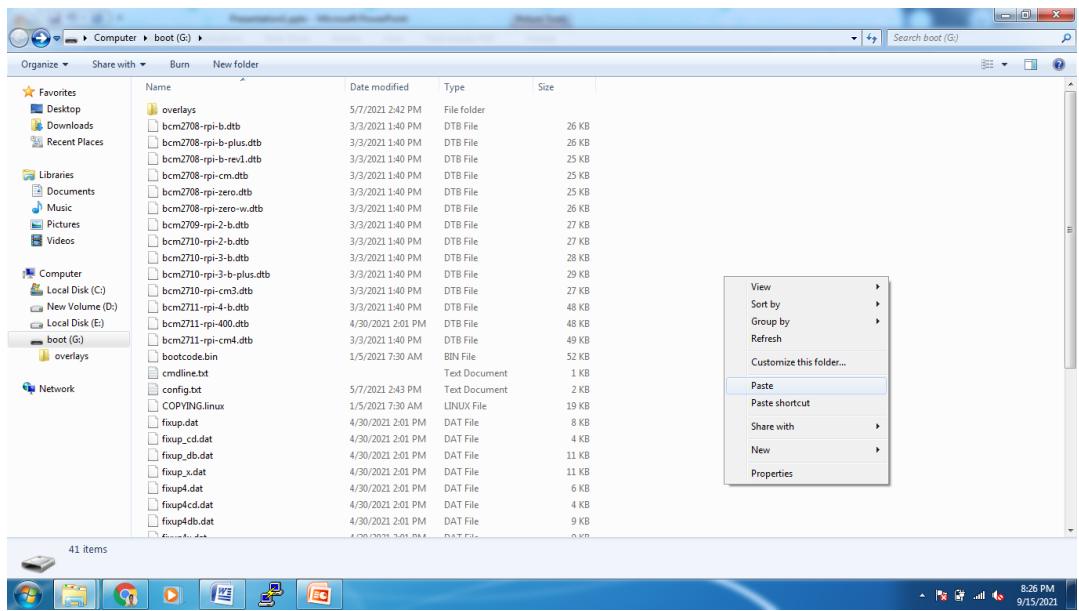


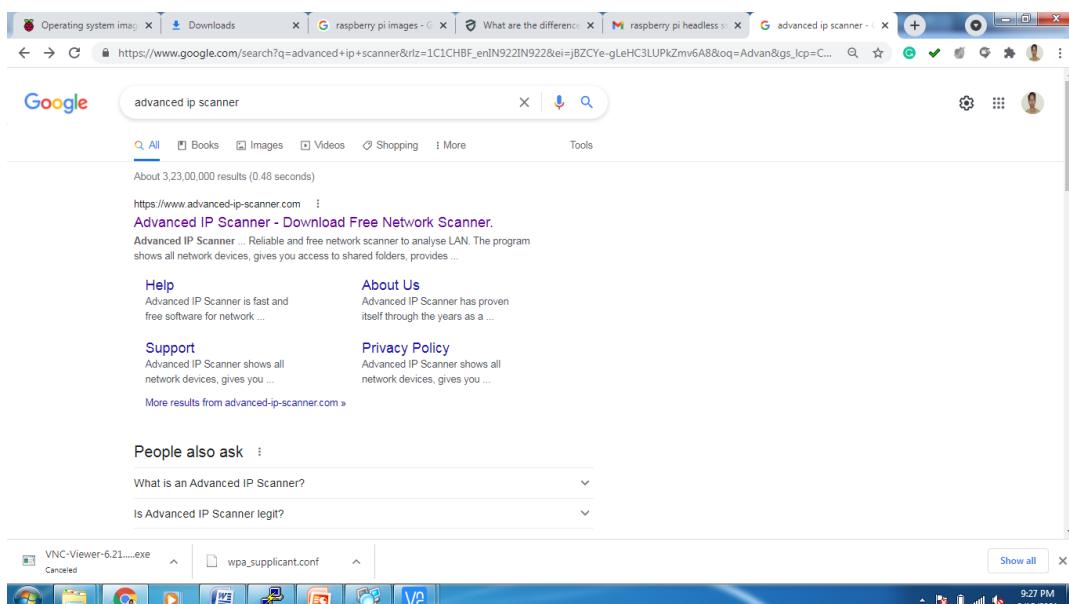
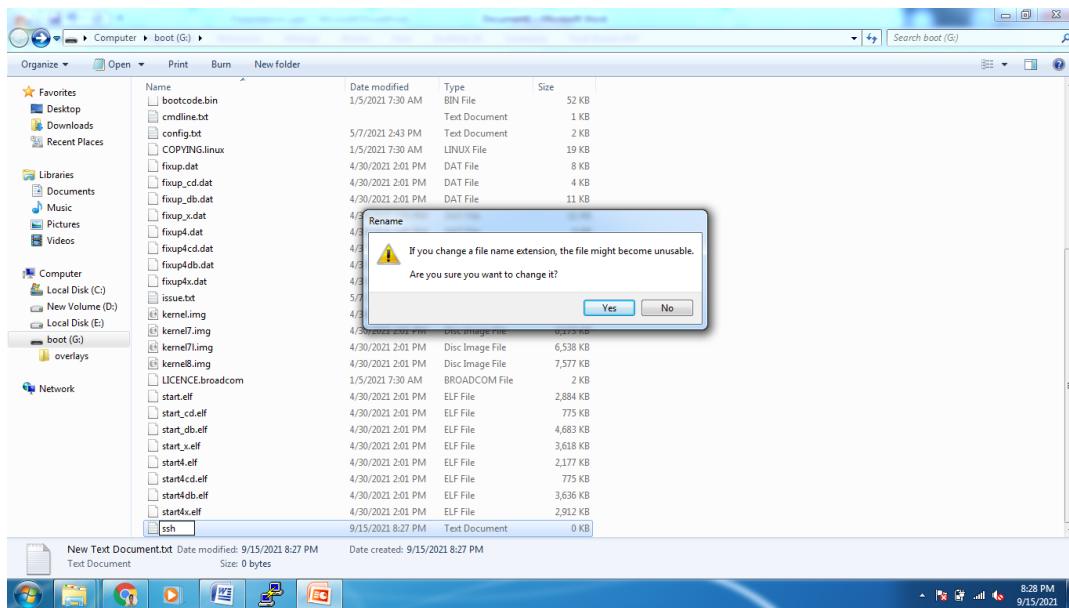


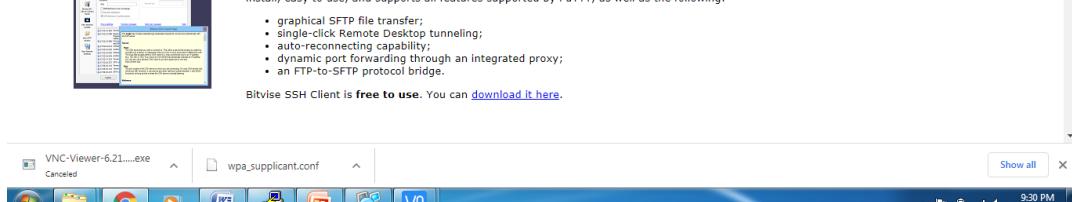
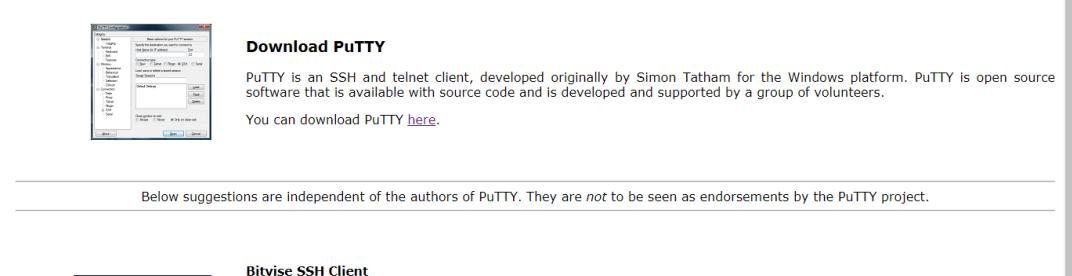
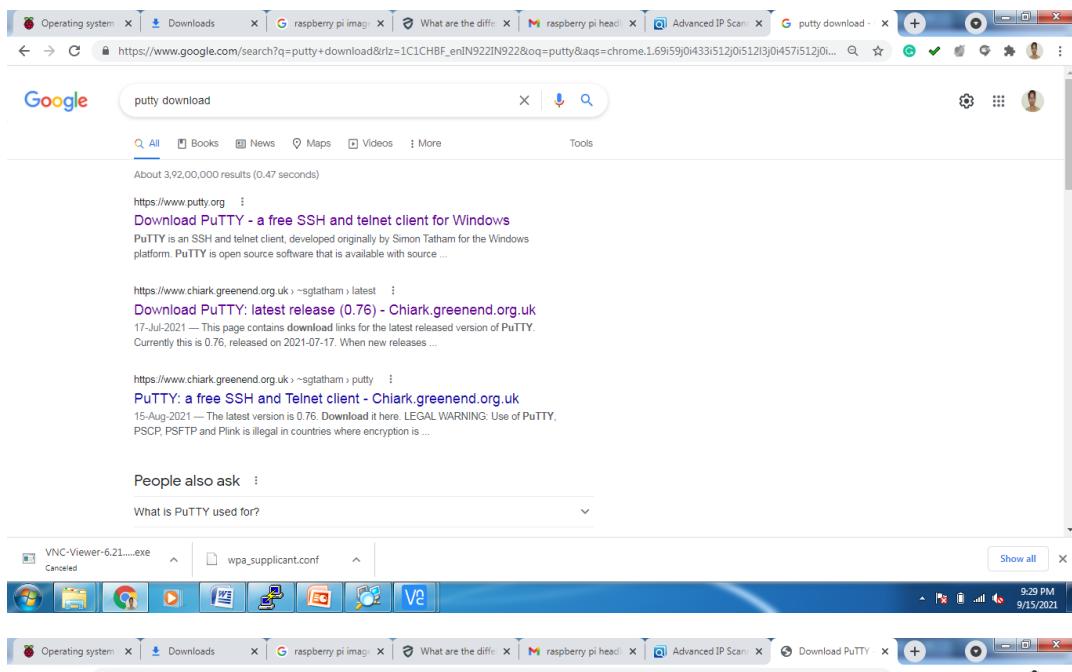
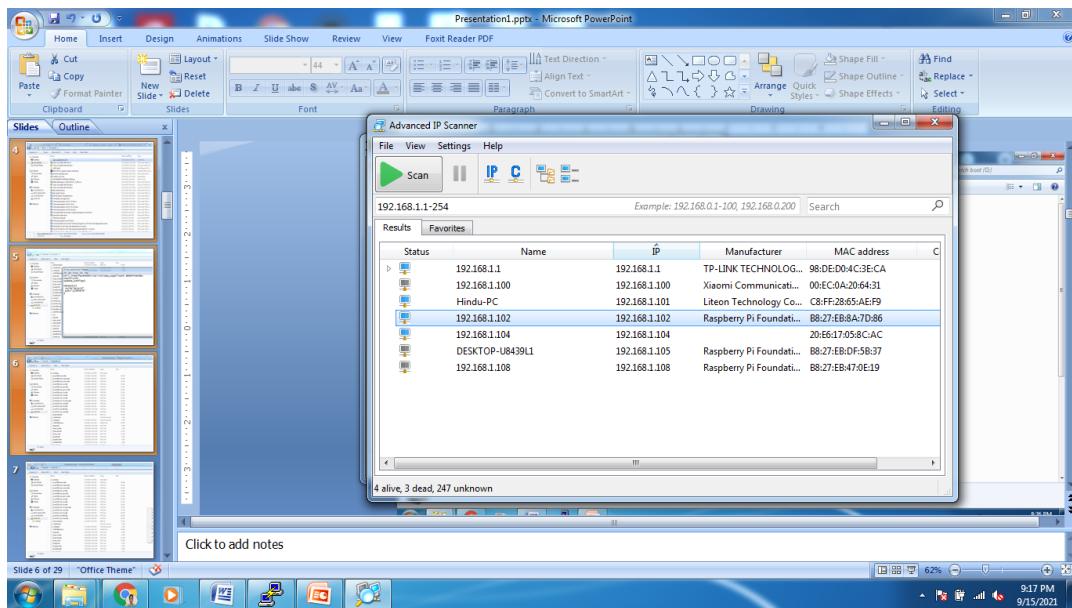


## Connecting to WiFi









This page contains download links for the latest released version of PuTTY. Currently this is 0.76, released on 2021-07-17.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a [permanent link to the 0.76 release](#).

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the [development snapshots](#), to see if the problem has already been fixed in those versions.

### Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

**MSI ('Windows Installer')**

64-bit x86:	<a href="#">putty-64bit-0.76-installer.msi</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>
64-bit Arm:	<a href="#">putty-arm64-0.76-installer.msi</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>
32-bit x86:	<a href="#">putty-0.76-installer.msi</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>

**Unix source archive**

.tar.gz:	<a href="#">putty-0.76.tar.gz</a>	(or by <a href="#">FTP</a> )	<a href="#">(signature)</a>
----------	-----------------------------------	------------------------------	-----------------------------

### Alternative binary files

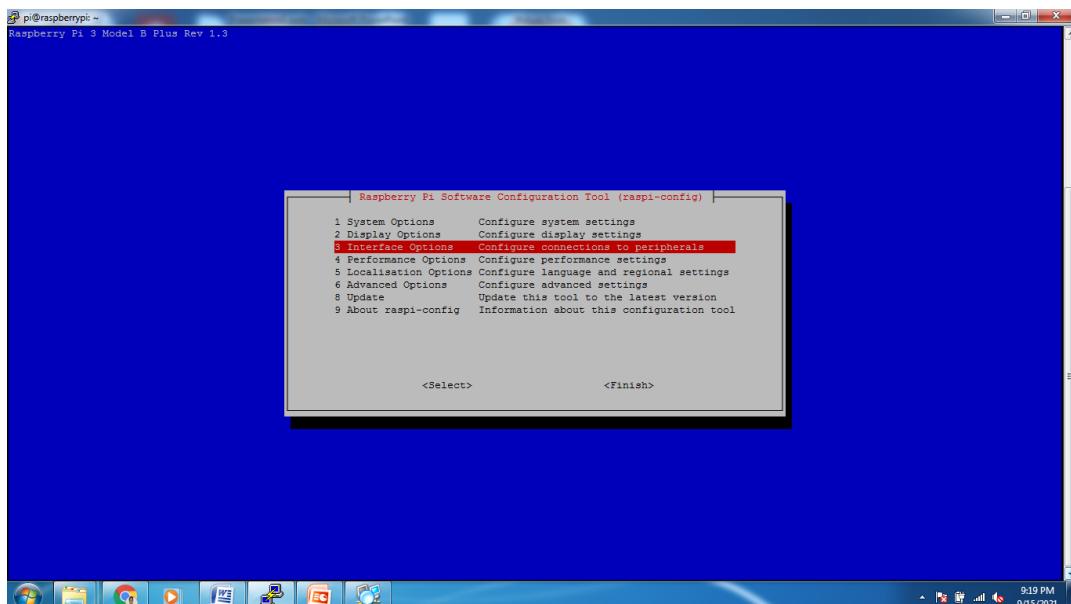
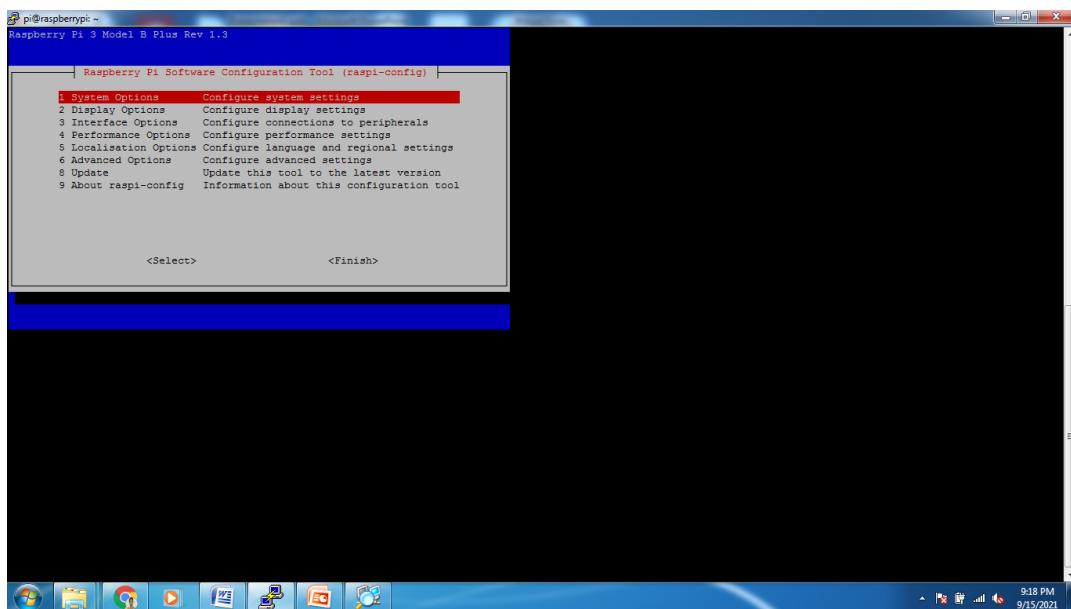
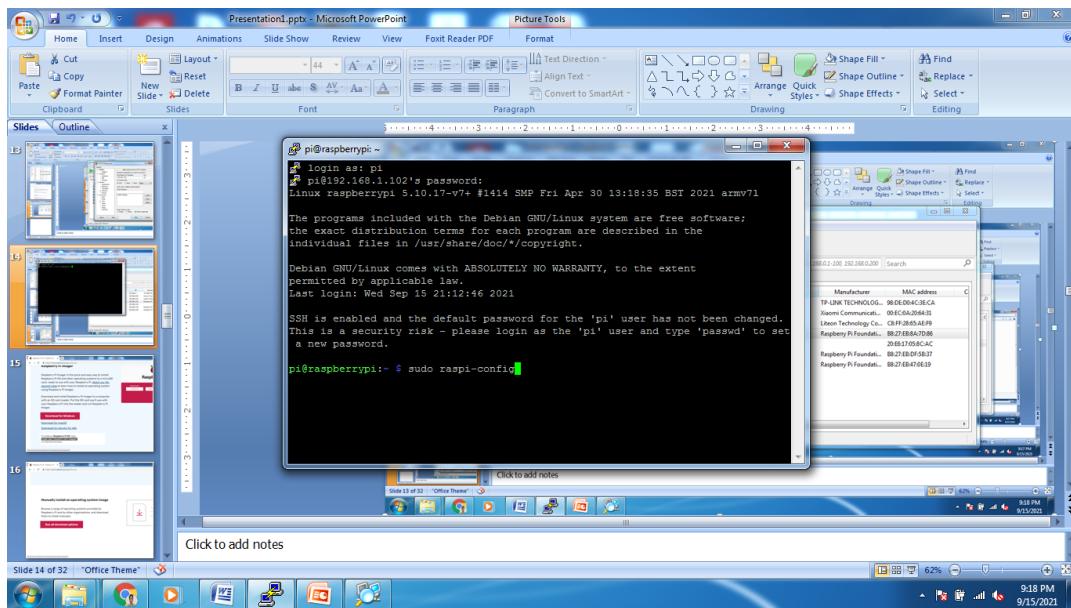
The installer packages above will provide versions of all of these (except PuTTYtel), but you can download standalone binaries one by one if you prefer.

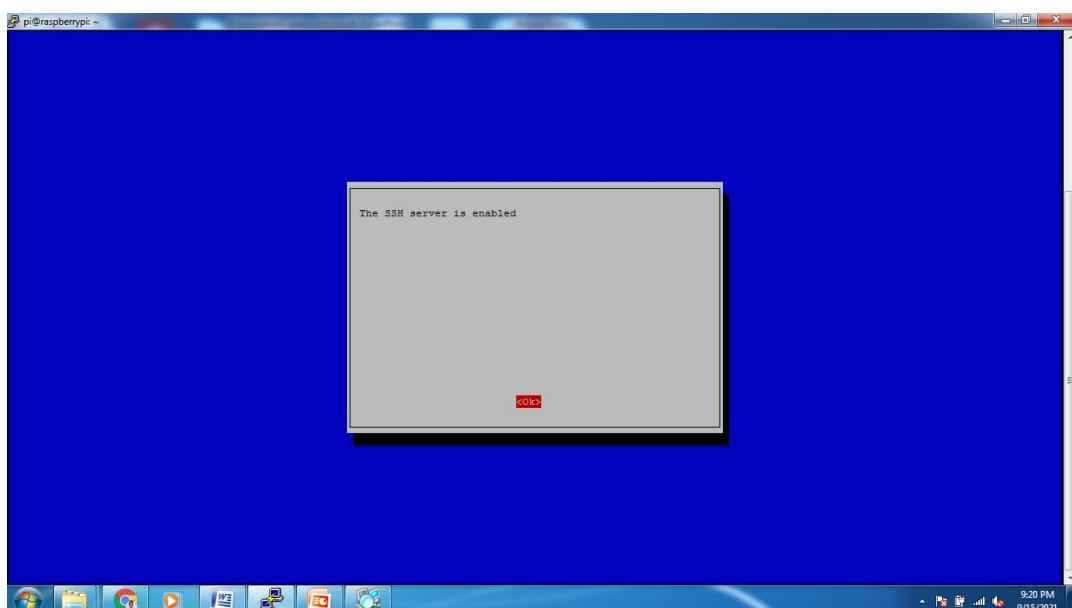
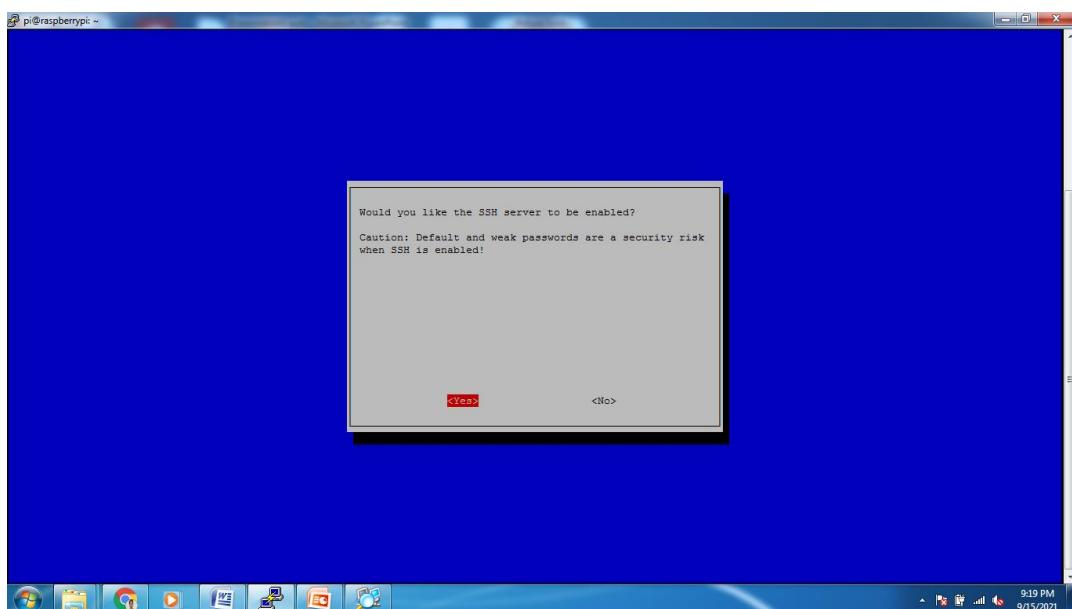
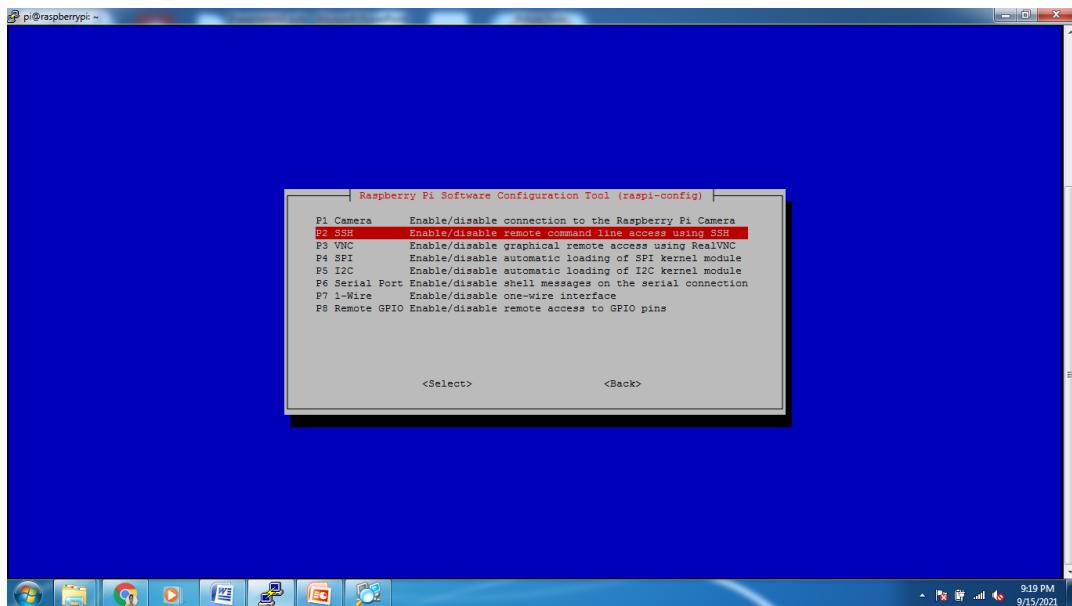
VNC-Viewer-6.21....exe Canceled wpa\_supplicant.conf Show all

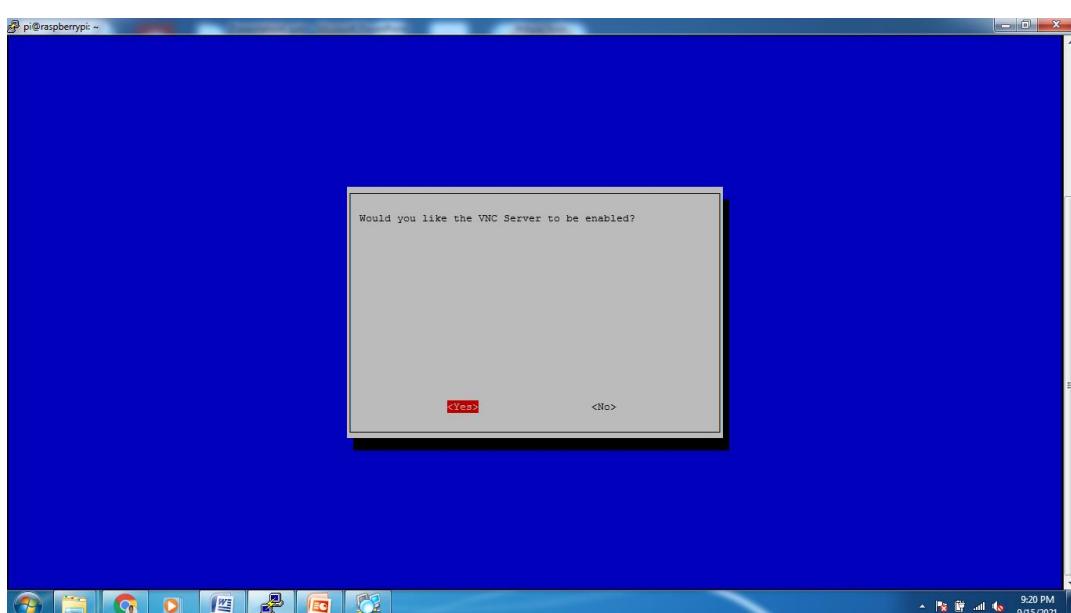
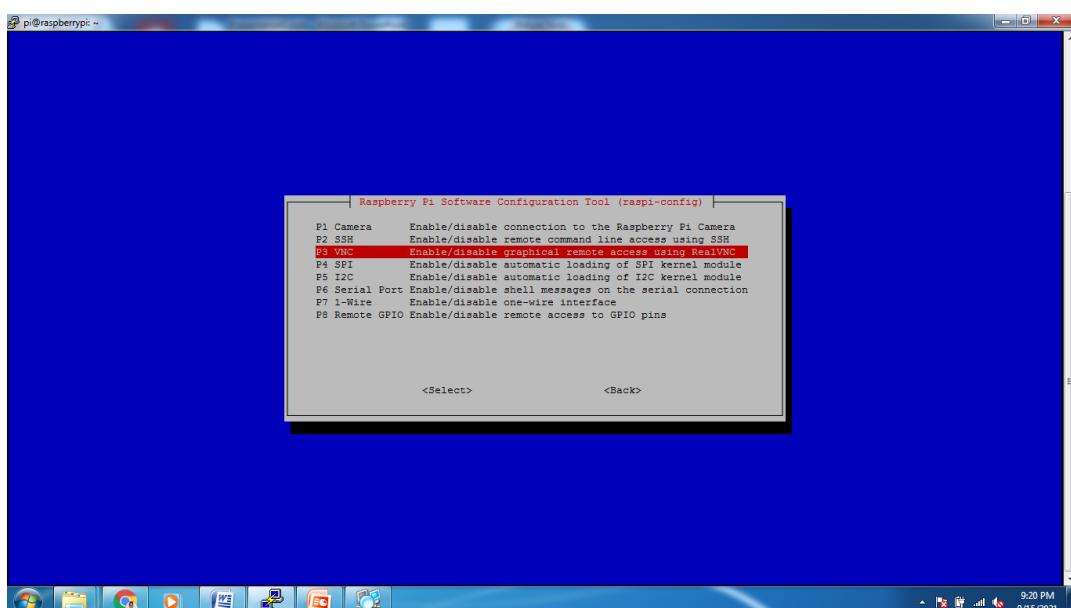
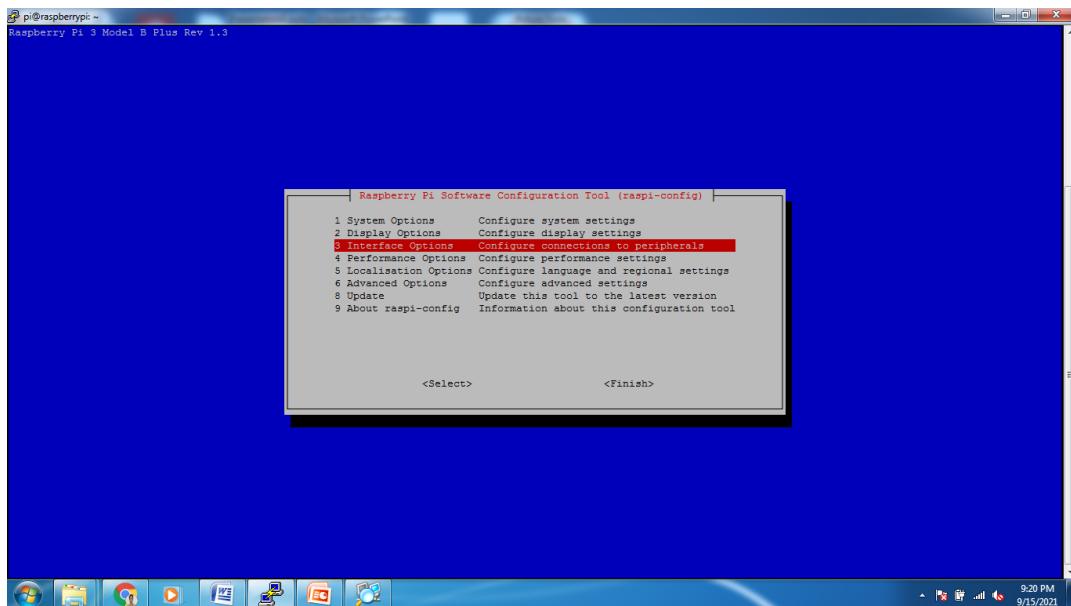
9:32 PM 9/15/2021

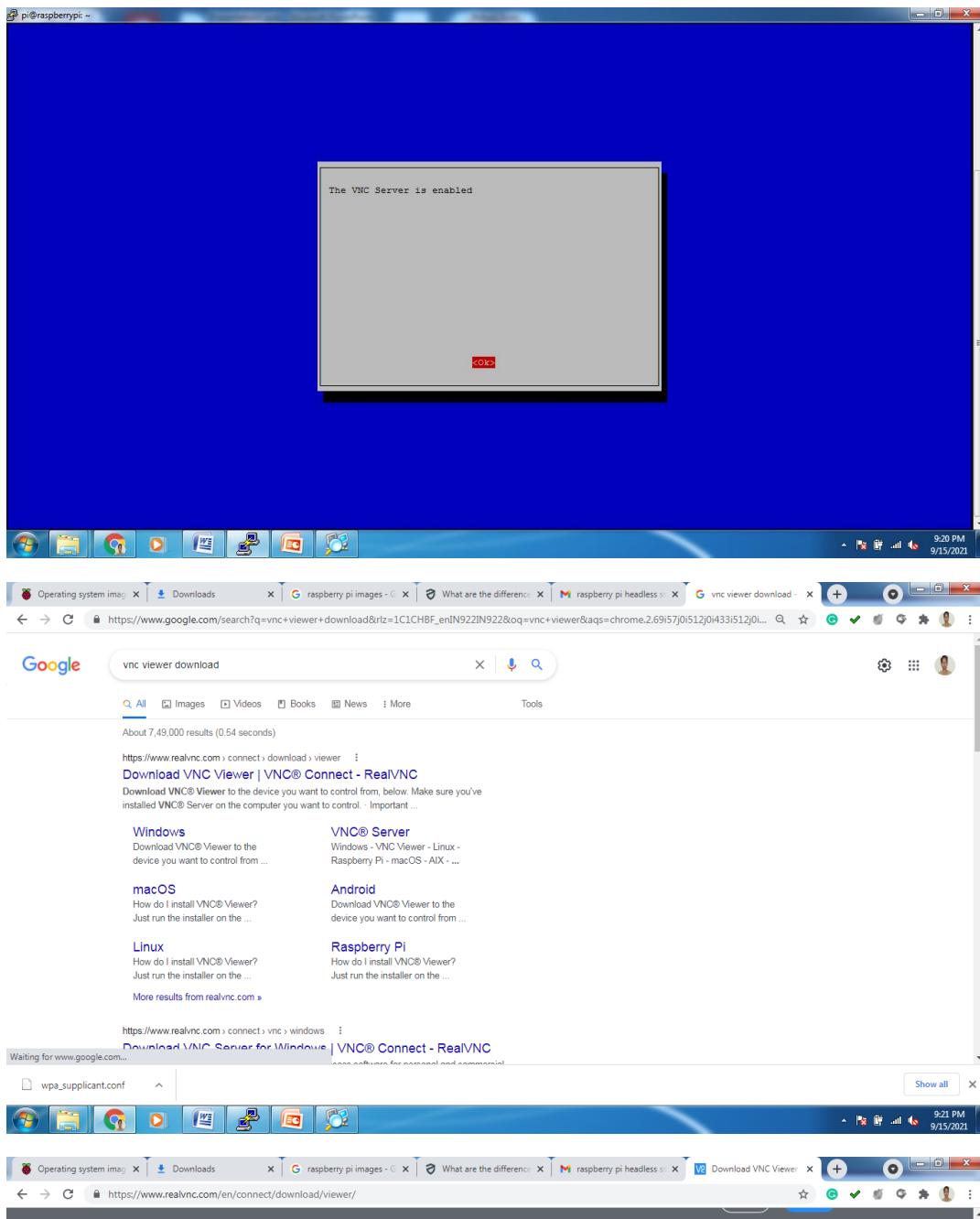
9:20 PM 9/15/2021

9:17 PM 9/15/2021









## VNC® Connect consists of VNC® Viewer and VNC® Server

Download VNC® Viewer to the device you want to control from, below. Make sure you've [installed VNC® Server](#) on the computer you want to control.

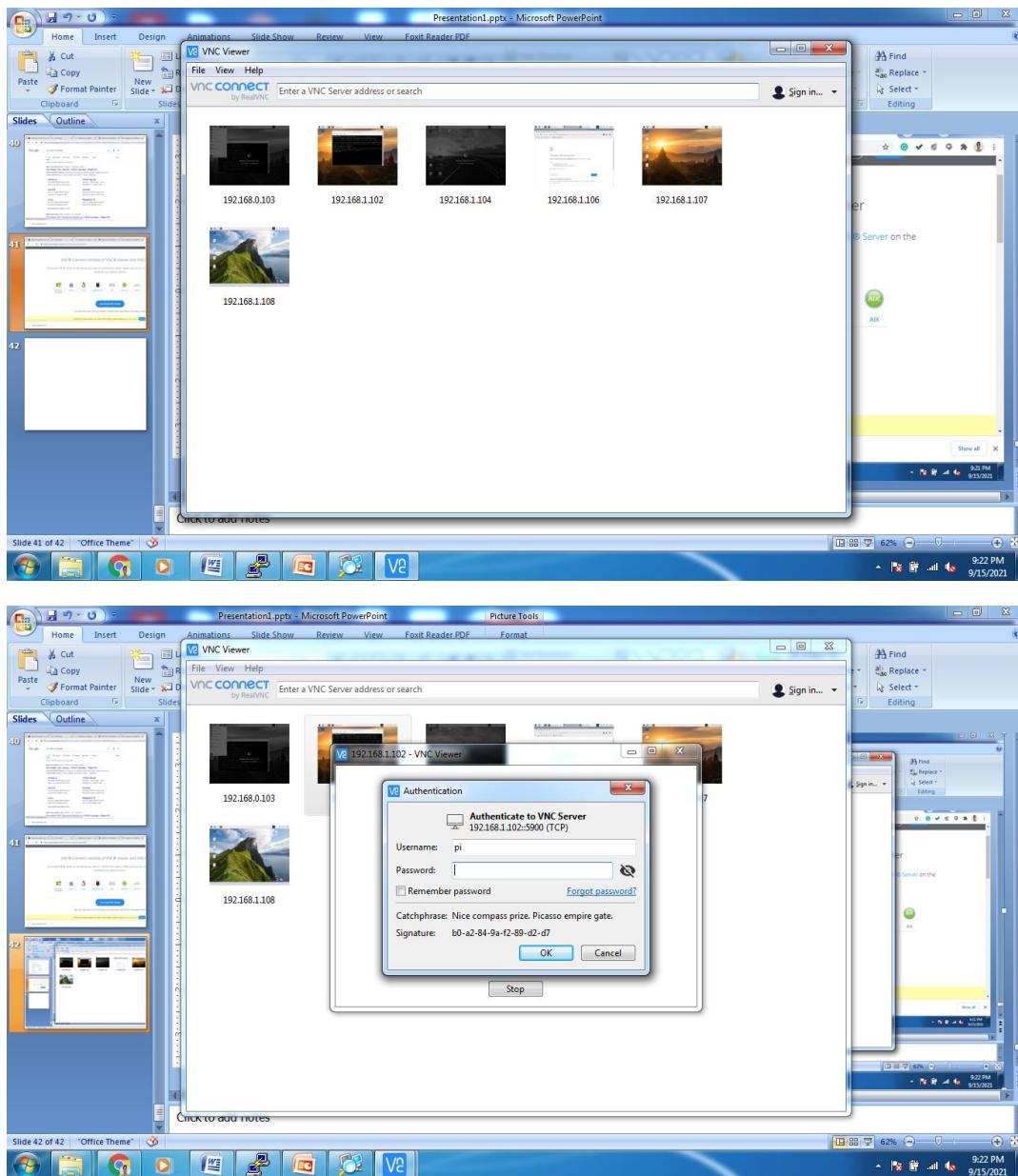


[Download VNC Viewer](#)

SHA-256: 6d2637db19c0c57d9375ddd15d24dd72e461a45a47fd5c017d529b8b7135599

RealVNC® uses cookies. For more information, please read our [privacy policy](#). [Got it](#)





## WEEK-8

Interface RGB LED with Raspberry Pi to obtain different colours and brightness using PWM.

### 1.

**AIM:** To interface RGB LED brightness using PWM pins with Raspberry Pi.

**PROGRAM:**

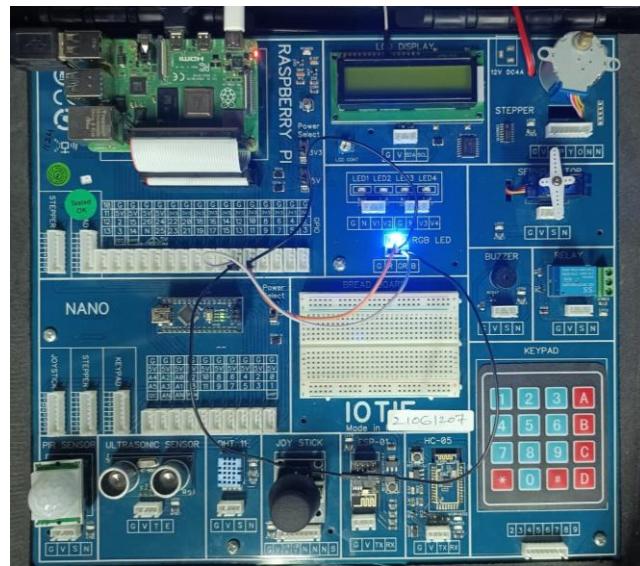
```
import RPi.GPIO as g
```

```
import time
```

```
g.setmode(g.BCM)
```

```
g.setwarnings(False)
ground = 10
red = 11
green = 12
blue = 13
g.setup(red, g.OUT)
g.setup(green, g.OUT)
g.setup(blue, g.OUT)
g.setup(ground, g.OUT)
g.OUTPUT(ground, g.LOW)
p = g.PWM(11,100)
q = g.PWM(12,100)
r = g.PWM(13,100)
p.start(0)
q.start(10)
r.start(50)
while True:
    for i in range(50):
        p.ChangeDutyCycle(i)
        q.ChangeDutyCycle(i)
        r.ChangeDutyCycle(i)
        time.sleep(0.5)
    for i in range(50):
        p.ChangeDutyCycle(100-i)
        q.ChangeDutyCycle(100-i)
        r.ChangeDutyCycle(100-i)
        time.sleep(0.5)
```

## CIRCUIT:



2.

**AIM:** To interface RGB LED with Raspberry Pi to obtain different colors through user input.

**PROGRAM:**

```
import RPi.GPIO as g  
import time  
  
g.setmode(g.BCM)  
g.setwarnings(False)  
ledpinone = 11  
ledpintwo = 12  
ledpinthree = 13  
  
g.setup(10,g.LOW)  
g.setup(ledpinone,g.OUT)  
g.setup(ledpintwo,g.OUT)  
g.setup(ledpinthree,g.OUT)  
  
while True:  
    color = input()  
    if color == 'red':
```

```

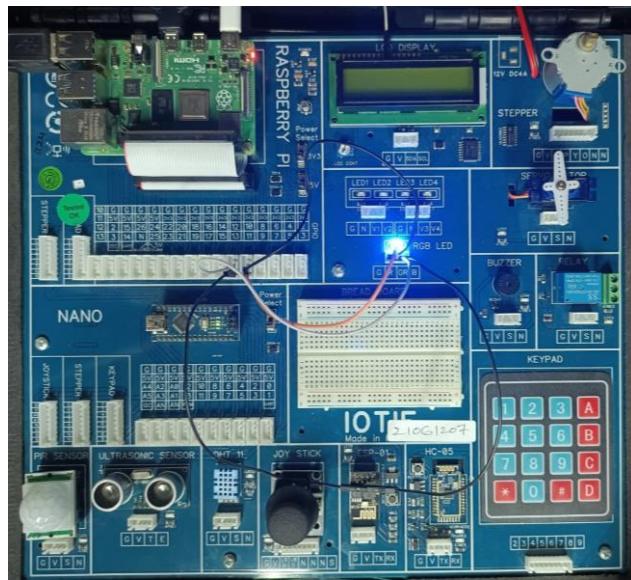
g.OUTPUT(ledpinone, g.HIGH)
g.OUTPUT(ledpintwo, g.LOW)
g.OUTPUT(ledpinthree, g.LOW)

elif color == 'green':
    g.OUTPUT(ledpinone, g.LOW)
    g.OUTPUT(ledpintwo, g.HIGH)
    g.OUTPUT(ledpinthree, g.LOW)

elif color == 'blue':
    g.OUTPUT(ledpinone, g.LOW)
    g.OUTPUT(ledpintwo, g.LOW)
    g.OUTPUT(ledpinthree, g.HIGH)

```

### CIRCUIT:



### 3.

**AIM:** To interface RGB led individual brightness using PWM pins with user input.

**PROGRAM:**

```
import RPi.GPIO as g
```

```
import time
```

```
g.setmode(g.BCM)
```

```

g.setwarnings(False)

ground = 10
red = 11
green = 12
blue = 13

g.setup(red, g.OUT)
g.setup(green, g.OUT)
g.setup(blue, g.OUT)
g.setup(ground, g.OUT)
g.OUTPUT(ground, g.LOW)

p = g.PWM(11,100)
q = g.PWM(12,100)
r = g.PWM(13,100)

p.start(0)
q.start(0)
r.start(0)

while True:
    color = input()
    if color == 'red':
        for x in range(100):
            p.ChangeDutyCycle(x)
            time.sleep(0.05)
        for x in range(100):
            p.ChangeDutyCycle(100-x)
            time.sleep(0.05)
    elif color == 'green':
        for x in range(100):

```

```

q.ChangeDutyCycle(x)
time.sleep(0.05)

for x in range(100):
    q.ChangeDutyCycle(100-x)
    time.sleep(0.05)

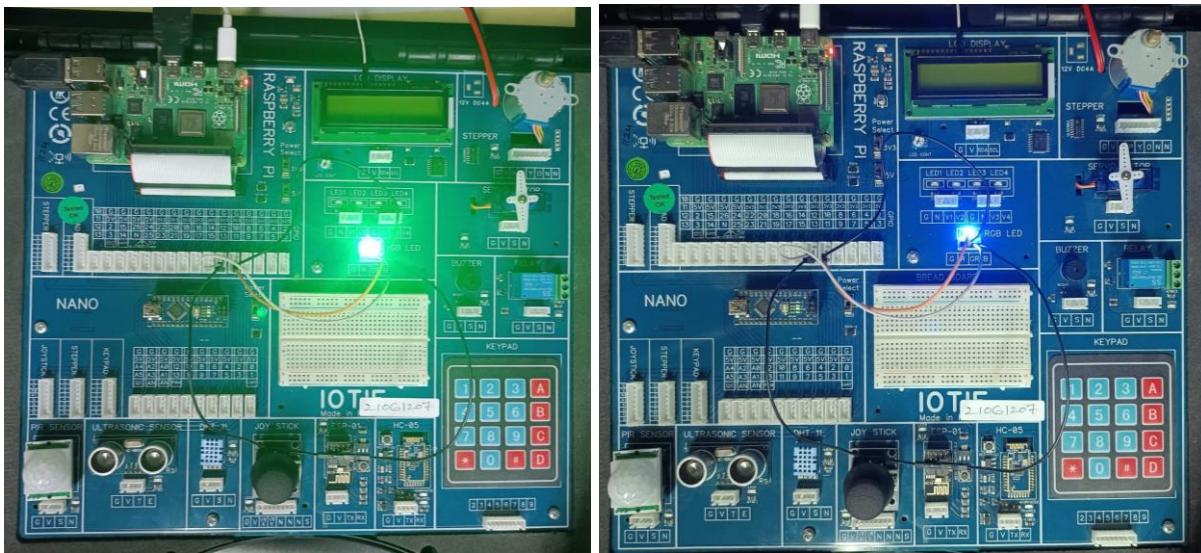
elif color == 'blue':

    for x in range(100):
        r.ChangeDutyCycle(x)
        time.sleep(0.05)

    for x in range(100):
        r.ChangeDutyCycle(100-x)
        time.sleep(0.05)

```

## CIRCUIT:



## **WEEK-9**

- a) Interface an ultrasonic sensor with Raspberry pi to print distance readings on the monitor when the sensor changes its position.
- b) Reading the data from an analog sensor with Raspberry using Arduino serial port or ADC MCP3208 using SPI.

**1.**

**AIM:** To interface an ultrasonic sensor with Raspberry pi to print distance readings on the monitor when the sensor changes its position.

**PROGRAM:**

```
import RPi.GPIO as g
import time

def distance(trigpin, echopin):
    g.output(trigpin, True)
    time.sleep(0.0001)
    g.output(trigpin, False)
    while g.input(echopin) == 0:
        pulse_start = time.time()
    while g.input(echopin) == 1:
        pulse_end = time.time()
    try:
        pulse_duration = pulse_end - pulse_start
    except:
        print('Calibrating')
        return -1
    distance = pulse_duration * 17150
    distance = round(distance + 1.15, 2)
    return distance

g.setmode(g.BCM)
trigpin = 24
```

```
echopin = 25
```

```
g.setup(trigpin, g.OUT)
```

```
g.setup(echopin, g.IN)
```

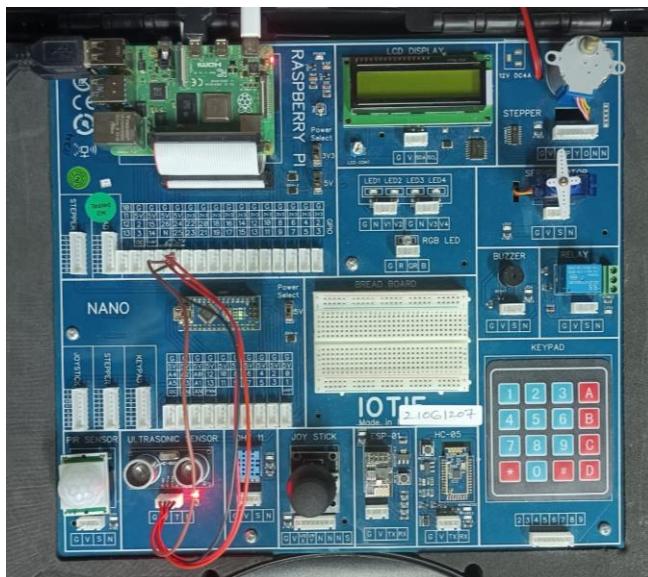
```
while True:
```

```
    dist = distance(trigpin, echopin)
```

```
    print('Measured distance = {}cm'.format(dist))
```

```
    time.sleep(0.01)
```

## CIRCUIT:



## OUTPUT:

```
measured disatnce=159.68cm
measured disatnce=159.23cm
measured disatnce=159.69cm
measured disatnce=160.11cm
measured disatnce=159.17cm
measured disatnce=159.27cm
measured disatnce=159.68cm
measured disatnce=159.19cm
measured disatnce=159.23cm
measured disatnce=159.67cm
measured disatnce=158.81cm
measured disatnce=159.23cm
measured disatnce=159.28cm
measured disatnce=159.26cm
measured disatnce=159.68cm
measured disatnce=159.68cm
measured disatnce=159.27cm
measured disatnce=159.32cm
measured disatnce=159.27cm
measured disatnce=159.66cm
measured disatnce=159.28cm
measured disatnce=159.65cm
measured disatnce=161.0cm
measured disatnce=159.3cm
measured disatnce=159.72cm
measured disatnce=159.35cm
measured disatnce=160.15cm
measured disatnce=158.87cm
measured disatnce=159.3cm
measured disatnce=159.29cm
measured disatnce=159.68cm
measured disatnce=159.68cm
```

## 2.

**AIM:** To read the data from an analog sensor (joystick) with Raspberry using Arduino serial port.  
**PROGRAM:**

```
# ## On Arduino
```

```
int VRx = A0;
```

```
int VRy = A1;
```

```
int xposition = 0;
```

```
int yposition = 0;
```

```
int mapx = 0
```

```
int mapy = 0;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    pinMode(VRx, INPUT);
```

```
    pinMode(VRy, INPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
    xposition = analogRead(VRx);
```

```
    yposition = analogRead(VRy);
```

```
    mapx = map(xposition, 0, 1023, -512, 512);
```

```
    mapy = map(yposition, 0, 1023, -512, 512);
```

```
    Serial.print("X:");
```

```
    Serial.print(mapx);
```

```
    Serial.print("Y:");
```

```
    Serial.print(mapy);
```

```
    delay(100);
```

```
}
```

```
# ON Raspberry pi
```

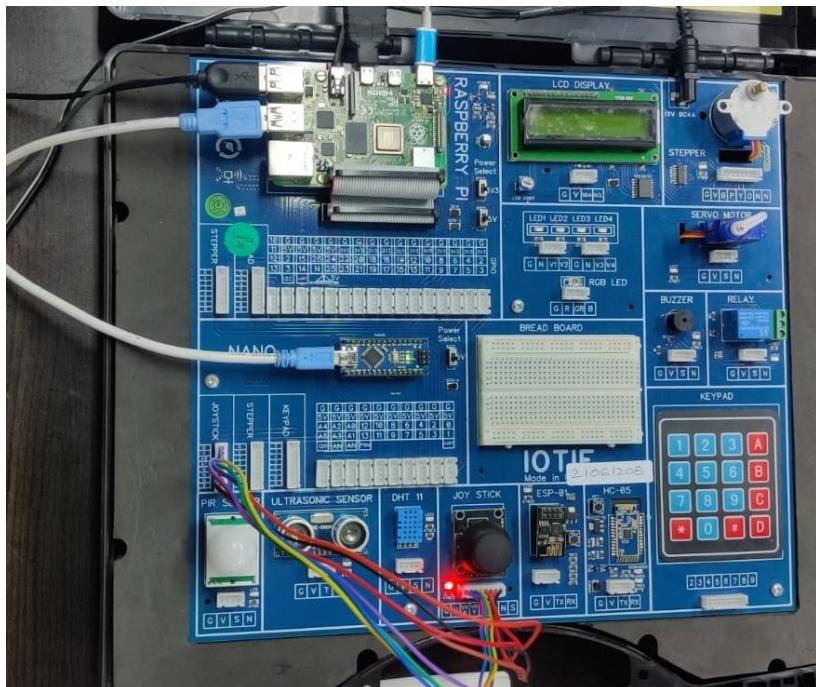
```
import serial
```

```
import time  
  
if __name__ == "__main__":  
  
    ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  
  
    ser.reset_input_buffer()
```

while True:

```
    if ser.in_waiting>0:  
  
        line = ser.readline().decode('utf-8').rstrip()  
  
        print(line)
```

## CIRCUIT:



## OUTPUT: (will look similar to this)

```
| Y:58X:73  
| Y:55X:67  
| Y:63X:59  
| Y:58X:73  
| Y:56X:64  
| Y:67X:73
```

## WEEK-10

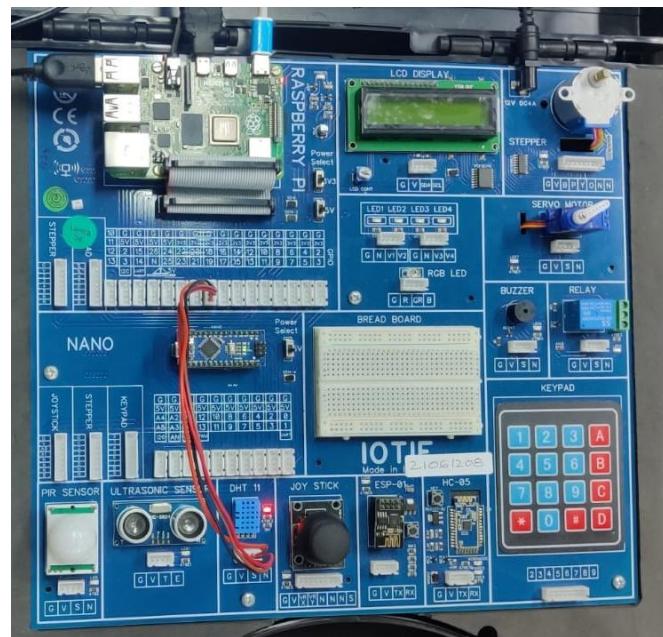
Post/read the data to/from the cloud via MQTT broker with a Raspberry Pi.

**AIM:** To post the data to the cloud via MQTT broker with a Raspberry Pi.

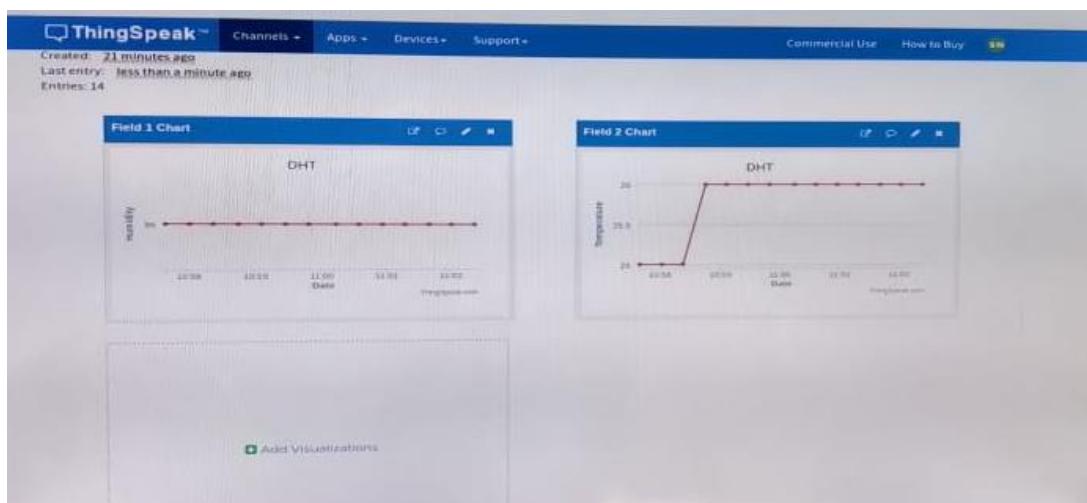
**PROGRAM:**

```
import urllib.request  
import time  
import RPi.GPIO as GPIO  
import Adafruit_DHT  
  
writeAPIKey = 'FNAIGO0VH990KRLW'  
baseURL = "https://api.thingspeak.com/update?api_key={}".format(writeAPIKey)  
sensor = Adafruit_DHT.DHT11  
sensorPin = 18  
GPIO.setmode(GPIO.BCM)  
try:  
    while True:  
        humidity,temperature = Adafruit_DHT.read_retry(sensor,sensorPin)  
        if humidity is not None and temperature is not None:  
            humidity = '%.2f' % humidity  
            temperature = '%.2f' % temperature  
            conn = urllib.request.urlopen(baseURL +  
                '&field1={}&field2={}'.format(humidity,temperature))  
            print(conn.read())  
            conn.close()  
        time.sleep(20)  
except KeyboardInterrupt:  
    GPIO.cleanup()  
    exit()
```

## CIRCUIT:



## OUTPUT:



## WEEK-11

Send real-time sensor data to a smartphone using Raspberry Pi onboard Bluetooth.

### **Here's what you'll need to do to install it:**

- Open Terminal and type **sudo apt-get install bluetooth bluez blueman**. Press enter.
- Once the packages have downloaded and installed, type **sudo reboot**.
- To access this menu, on the Raspberry Pi desktop click Menu, located in the upper left corner of the screen, scroll down to **Preferences** with your cursor and click **Bluetooth Manager**. From there, you can pair to any nearby devices, or you can make your Raspberry Pi discoverable so you can pair your phone to it from your phone's Bluetooth settings.

### **How to setup Bluetooth on Raspberry Pi:**

- From the Raspberry Pi desktop, open a new Terminal window.
- Type **sudo bluetoothctl** then press enter and input the administrator password (the default password is **raspberry**).
- Next, enter **agent on** and press enter. Then type **default-agent** and press enter.
- Type **scan on** and press enter one more time. The unique addresses of all the Bluetooth devices around the Raspberry Pi will appear and look something like an alphanumeric **XX:XX:XX:XX:XX:XX**. If you make the device you want to pair discoverable (or put it into pairing mode), the device nickname may appear to the right of the address. If not, you will have to do a little trial and error or waiting to find the correct device.
- To pair the device, type **pair [device Bluetooth address]**. The command will look something like **pair XX:XX:XX:XX:XX:XX**.
- If you're pairing a keyboard, you will need to enter a six-digit string of numbers. You will see that the device has been paired, but it may not have connected. To connect the device, type **connect XX:XX:XX:XX:XX:XX**.

1.

**AIM:** To implement a program to control sensor through smart phone using Raspberry Pi onboard Bluetooth.

### **PROGRAM:**

```
import bluetooth
```

```
import RPi.GPIO as GPIO
```

```
ledpin = 18
```

```

buzzerpin = 10

GPIO.setmode(GPIO.BCM)

GPIO.setup(ledpin, GPIO.OUT)

host = " "

port = 1

server = bluetooth.BluetoothSocket(bluetooth.RFCOMM)

try:

    server.bind((host, port))

    print("Bluetooth binding succesful")

except:

    print('Bluetooth binding failed')

server.listen(1)

client, address = sensor.accept()

print('connection received from', address)

print('client: ', client)

try:

    while True:

        data = client.recv(1024)

        data = data.decode('utf-8').strip().upper()

        print('client sent:', data)

        if data =='L1':

            GPIO.output(ledpin, True)

            send_data = 'lights on'

        elif data =='L0':

            GPIO.output(ledpin, False)

            send_data = 'lights off'

        elif data =='B1':

            GPIO.output(buzzerpin, True)

            send_data = 'Buzzer on'

```

```

        elif data =='B0':
            GPIO.output(buzzerpin, Flase)
            send_data = 'Buzzer off'

        else:
            send_data = 'supported elements are: L0 L1 B0 B1'

    except:
        client.send('\n Exiting the Program')
        GPIO.cleanup()
        client.close()
        server.close()
        exit()

```

## 2.

**AIM:** To implement a program to send real-time sensor data to smart phone using Raspberry Pi onboard Bluetooth.

### **PROGRAM:**

```

import bluetooth
import time
import RPi.GPIO as g

g.setmode(g.BCM)
trigpin = 24
echopin = 25
g.setup(trigpin, g.OUT)
g.setup(echopin, g.IN)
host = " "
port = 1
server = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
try:
    server.bind((host, port))
    print("bluetooth binding succesful")

```

```
except:  
    print('Bluetooth binding unsuccesful')  
    server.listen(1)  
    client, address = server.accept()  
    print('connection received from', address)  
    print('client', client)
```

```
def distance(trigpin, echopin):  
    g.output(trigpin, True)  
    time.sleep(0.0001)  
    g.output(trigpin, False)  
    while g.input(echopin) == 0:  
        pulse_start = time.time()  
    while g.input(echopin) == 1:  
        pulse_end = time.time()  
    try:  
        pulse_duration = pulse_end - pulse_start  
    except:  
        print('Calibrating')  
        return -1  
    distance = pulse_duration * 17150  
    distance = round(distance + 1.15, 2)  
    return distance
```

```
while True:  
    dist = distance(trigpin, echopin)  
    print('measured distance = {} cm'.format(dist))  
    client.send('\n measured distance')  
    client.speed(str(dist))  
    time.sleep(2)
```

## WEEK-12

Implement an intruder alert system that alerts through email

1.

**AIM:** To implement a program to send an email with Raspberry Pi.

**PROGRAM:**

```
import smtplib  
from time import sleep  
import RPi.GPIO as GPIO  
from sys import exit  
  
from_email = '<my-email>'  
recipients_list = ['<recipient-email>']  
cc_list = []  
subject = 'Hello: Message from Raspberry Pi'  
message = 'Input given by the user'  
username = '<Gmail-username>'  
password = '<password>'  
server = 'smtp.gmail.com:587'  
  
inp=int(input())  
  
def sendmail(from_addr, to_addr_list, cc_addr_list, subject, message, login, password, smtpserver):  
    header = 'From: %s \n' % from_addr  
    header += 'To: %s \n' % ','.join(to_addr_list)  
    header += 'Cc: %s \n' % ','.join(cc_addr_list)  
    header += 'Subject: %s \n \n' % subject  
    message = header + message  
  
    server = smtplib.SMTP(smtpserver)  
    server.starttls()
```

```
server.login(login, password)
problems = server.sendmail(from_addr, to_addr_list, message)
server.quit()
```

while True:

try:

```
if (inp == 1):
```

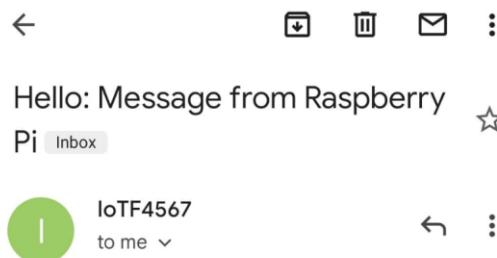
```
    sendmail(from_email, recipients_list, cc_list, subject, message, username, password,
server)
```

```
    sleep(.01)
```

except KeyboardInterrupt:

```
    exit()
```

## OUTPUT:



↳ Reply

« Reply all

↗ Forward

## 2.

**AIM:** To implement an intruder alert system that alerts through email.

### PROGRAM:

```
import smtplib
import time
from time import sleep
import RPi.GPIO as GPIO
from sys import exit

from_email = '<my-email>'
recipients_list = ['<recipient-email>']
cc_list = []
subject = 'Hello'
message = 'PIR with Raspberry Pi'
username = '<Gmail-username>'
password = '<password>'
server = 'smtp.gmail.com:587'

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(11, GPIO.IN)

def sendmail(from_addr, to_addr_list, cc_addr_list, subject, message, login, password,
smtpserver):
    header = 'From: %s \n' % from_addr
    header += 'To: %s \n' % ',' .join(to_addr_list)
    header += 'Cc: %s \n' % ',' .join(cc_addr_list)
    header += 'Subject: %s \n \n' % subject
    message = header + message

    server = smtplib.SMTP(smtpserver)
    server.starttls()
    server.login(login, password)
    problems = server.sendmail(from_addr, to_addr_list, message)
    server.quit()

while True:
    try:
        if (GPIO.input(11) == 1):
            sendmail(from_email, recipients_list, cc_list, subject, message, username, password,
server)
            sleep(.01)
```

```
except KeyboardInterrupt:  
    exit()
```

## OUTPUT:

