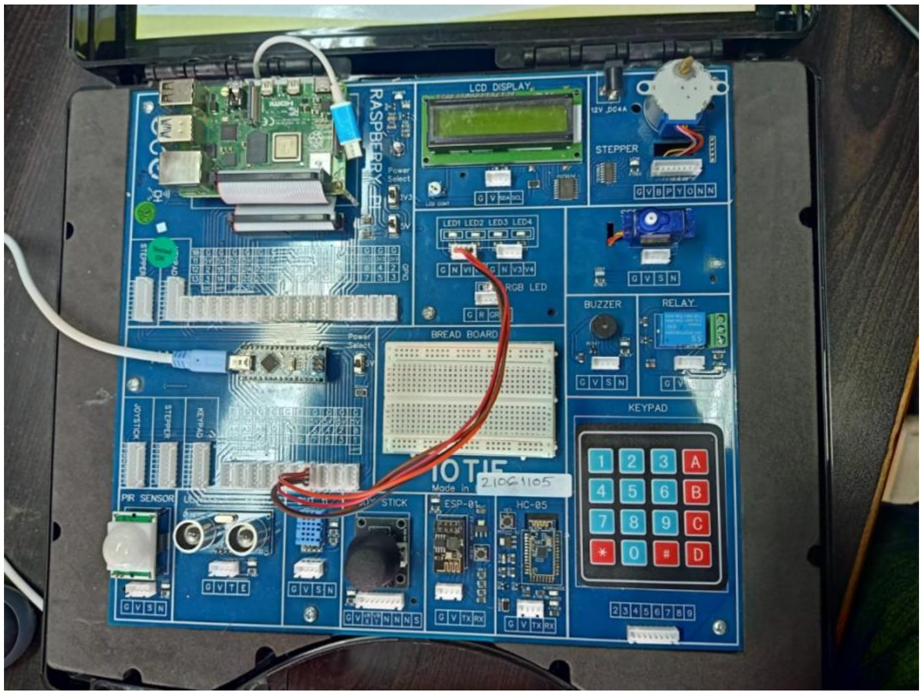
WEEK – 1

1. **AIM :** To install IDE of Arduino and write a program using Arduino IDE to blink LED
   COMPONENTS REQUIRED: Arduino board**,** LED's**,** Resistors, Breadboard



THEORY:

INSTALLATION OF IDE

- Search for Arduino IDE in google chrome.
- Download the Arduino IDE software into PC.
- Proceed with board specific instructions.
- Open the downloaded Arduino software and write the program code.
- Check the output and dump it into Arduino board using certain cable.

OR

1.Visit Tinkercad's website
2.Create an account or log into an existing one
3.Select "Circuits" on the left side of the screen
4.Select "Create new Circuit" on the next page
5.After building your circuit click the "Code" button in the top menu bar
6.Enter your code and start the simulation

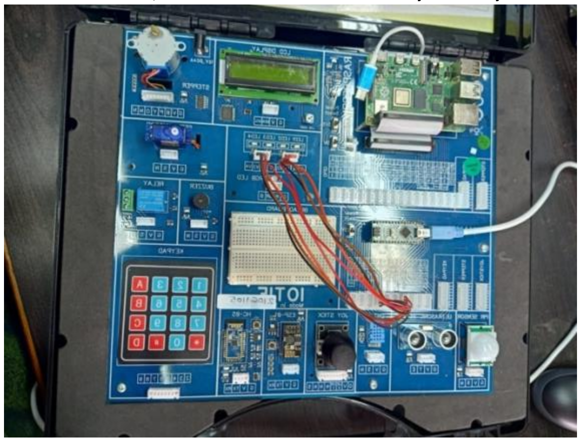
CODE: Program to Blink an LED With Arduino in Tinkercad

```
void setup()
{
pinMode(7, OUTPUT);
}
void loop()
{
digitalWrite(7, HIGH);
delay(5000); // Wait for 5000 millisecond(s)
digitalWrite(7, LOW);
delay(5000); // Wait for 5000 millisecond(s)
}
```

OUTPUT:

After starting the simulation, it sets the LED HIGH (Fig.2) i.e. turns ON the LED for 5000 milli seconds and after that it againsets the LED LOW (Fig.1) i.e turns OFF the LED for 5000 milli seconds. This process continues till we stop the simulation.

2. AIM : To write a program using Arduino IDE to blink two LED's.
COMPONENTS REQUIRED: Arduino board**,** LED's**,** Breadboard.



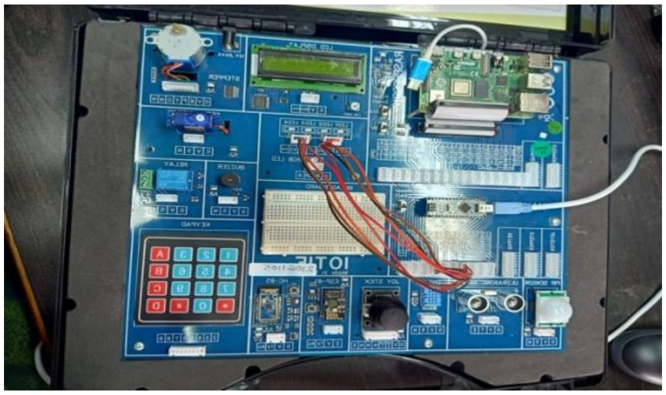
CODE:

```
void setup()
{
```

```
pinMode(7, OUTPUT);
pinMode(12, OUTPUT);
}
void loop()
{
digitalWrite(7, HIGH);
delay(5000); // Wait for 5000 millisecond(s)
digitalWrite(7, LOW);
delay(5000); // Wait for 5000 millisecond(s)
digitalWrite(12, HIGH);
delay(5000); // Wait for 5000 millisecond(s)
digitalWrite(12, LOW);
delay(5000); // Wait for 5000 millisecond(s)
}
```

OUTPUT :
After starting the simulation, It sets the LED1 (right LED) to HIGH i.e, turns ON for 5000 milli seconds and then sets the LED1 (right LED) to LOW i.e, turns OFF for 5000 milli seconds. Now it sets the LED2 (left LED) to HIGH i.e, turns ON for 5000 milli seconds and then sets the LED2 (left LED) to LOW i.e, turns OFF for 5000 milli seconds. This process continues till we stop the simulation.

3. AIM : To blink three LED's or Automatic Traffic light system
COMPONENTS REQUIRED: Arduino board**,** LED's **,** Breadboard



CODE:

```
void setup()

{
  pinMode(7, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
}
void loop()
{
```

```
digitalWrite(7, HIGH);
delay(10000); // Wait for 10000 millisecond(s)
digitalWrite(7, LOW);
delay(10000); // Wait for 10000 millisecond(s)

digitalWrite(10, HIGH);
delay(3000); // Wait for 3000 millisecond(s)
digitalWrite(10, LOW);
delay(3000); // Wait for 3000 millisecond(s)

digitalWrite(12, HIGH);
delay(10000); // Wait for 10000 millisecond(s)
digitalWrite(12, LOW);
delay(10000); // Wait for 10000 millisecond(s)
  }
```

OUTPUT:

After starting the simulation, it sets the LED1 (Green LED) to HIGH i.e, turns ON for 10000 milli seconds and then sets the LED2 (Yellow LED) to HIGH i.e, turns ON for 3000 milli seconds and then sets the LED3 (Red LED) to HIGH i.e, turns ON for 10000 milli seconds. This process continues till we stop the simulation.

WEEK – 2

1. AIM: To control LED with push button

COMPONENTS REQUIRED: Arduino board**,** LED's **,** Resistors**,** Breadboard**,** Push button

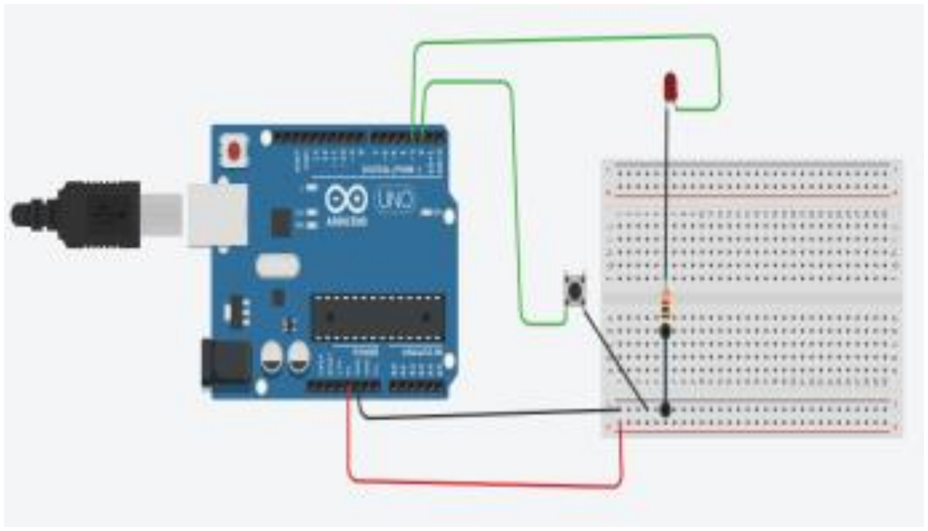


Fig.1: Circuit before starting the simulation

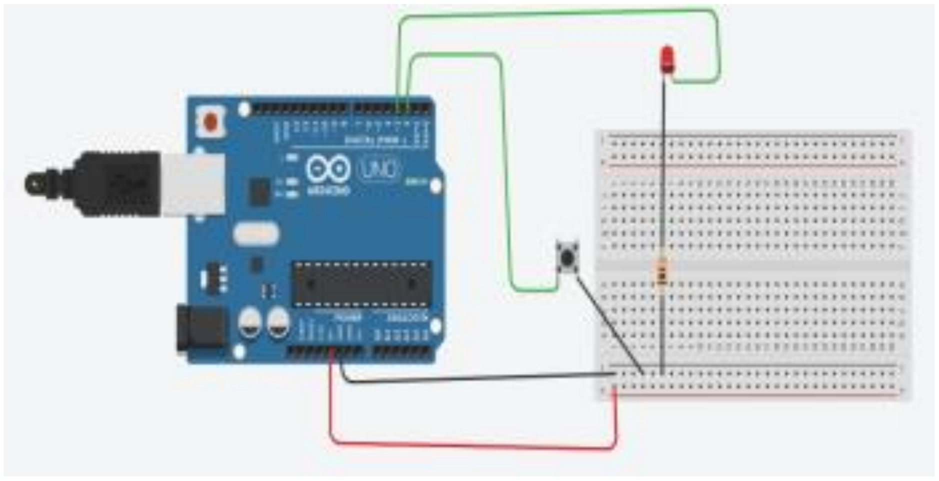


Fig.2: After starting the simulation

THEORY:

We set the Button1 variable as integer 2 and we connect the button at pin 2 on the Board. Then the LED is connected to pin 3 using the resistor in series with it. In this code we can turn on the LED when we press the button then turn it off when we press the button again**.**

CODE:

```
int LED=3;
int Button1=2;
void setup()
{
pinMode(Button1, INPUT);
digitalWrite(Button1, HIGH);
pinMode(LED, OUTPUT);
}
void loop()
```

```
{
if(digitalRead(Button1)==LOW)
 {
digitalWrite(LED, HIGH);
 }
    else
 {
digitalWrite(LED, LOW);
 }
  }
```

OUTPUT:

After starting the simulation, when the push button is pressed the LED will be ON whereas if the pushbutton is not pressed the LED will be OFF.

2. AIM: To control two LED's with two push buttons

COMPONENTS REQUIRED: Arduino board, LED's **,** Resistors**,** Breadboard**,** Push buttons

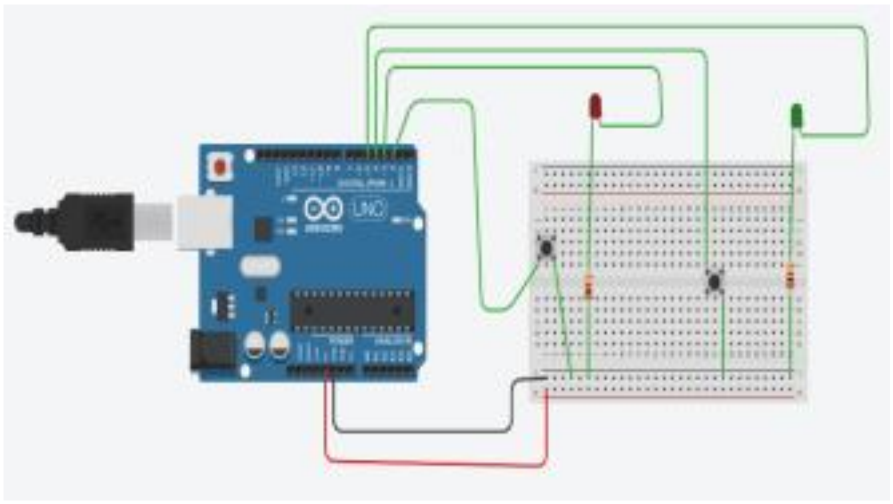


Fig.1: Circuit before starting the simulation

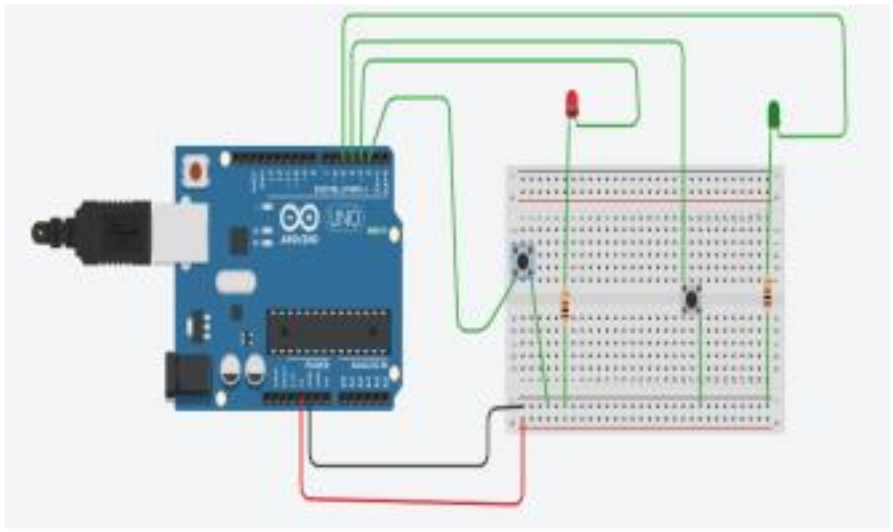


Fig.2: After starting the simulation, if the push button corresponding to the red LED is pressed then the Red LED will be ON and when the button is released Red LED will be OFF.

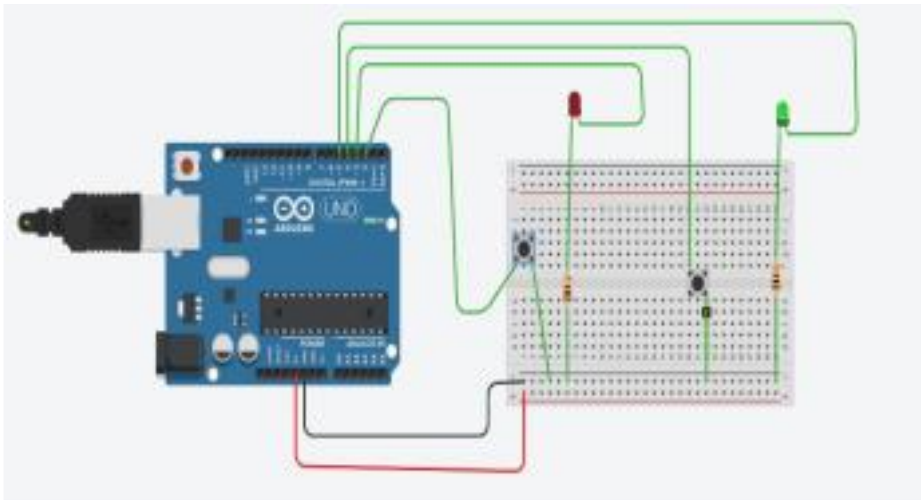


Fig.3: If the push button corresponding to the Green LED is pressed then the Green LED will be ON and when the button is released Green LED will be OFF.

THEORY:

We set the Button1 variable as integer 2 and we connect the button at pin 2 on the Board. We set the Button2 variable as integer 4 and we connect the button at pin 4 on the Board. Then the LED is connected to pin 3 using the resistor in series with it and the LED2 is connected to pin 5 using the resistor in series with it. In this code we can turn on the LED’s when we press the buttons then turn it off when we press the buttons again**.**

CODE:

```
int LED=3;
int Button1=2;
int LED2=5;
int Button2=4;
void setup()
```

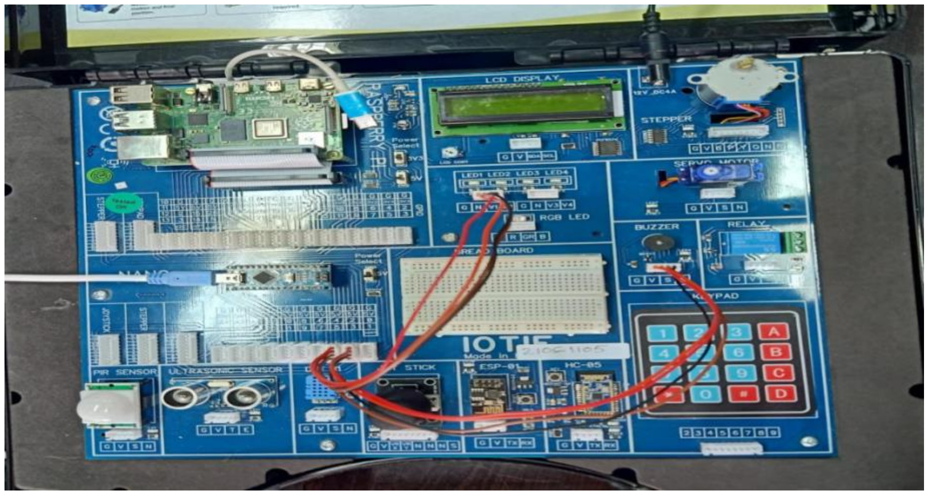
```
{
pinMode(Button1, INPUT);
digitalWrite(Button1, HIGH);
pinMode(LED, OUTPUT);
pinMode(Button2, INPUT);
digitalWrite(Button2, HIGH);
pinMode(LED2, OUTPUT);
}
void loop()
{
if(digitalRead(Button1)==LOW)
 {
digitalWrite(LED, HIGH);
 }
 else
 {
digitalWrite(LED, LOW);
 }
if(digitalRead(Button2)==LOW)
 {
digitalWrite(LED2, HIGH);
 }
 else
 {
digitalWrite(LED2, LOW);
 }
}
```

OUTPUT:

After starting the simulation, if the push button corresponding to the red LED is pressed then the Red LED will be ON and when the button is released, Red LED will be OFF. After that, If the push button corresponding to the Green LED is pressed then the Green LED will be ON and when the button is released then the Green LED will be OFF.

3. AIM: To interface LED and buzzer to buzz for a period of time

COMPONENTS REQUIRED : Arduino board, LED's **,**Resistors **,**Breadboard**,**Push buttons



THEORY:

In this code, the buzzer buzzes and LED will be ON for some time and the buzzer stops buzzing and LED will be OFF for some time

CODE:

```
void setup()
{
pinMode(2, OUTPUT);
pinMode(4, OUTPUT);
}
void loop()
{
digitalWrite(2, HIGH);
digitalWrite(4, HIGH);
delay(1000);
digitalWrite(2, LOW);
digitalWrite(4, LOW);
delay(1000);
}
```

OUTPUT:

After starting the simulation, the buzzer buzzes and LED will be ON for 1000 milli seconds and after that the buzzer stops buzzing and LED will be OFF for 1000 milli seconds. This process continues till we stop the simulation.

4. AIM: To control buzzer and LED with push button

COMPONENTS REQUIRED: Arduino board, LED's **,** Resistors**,** Breadboard**,** Push buttons

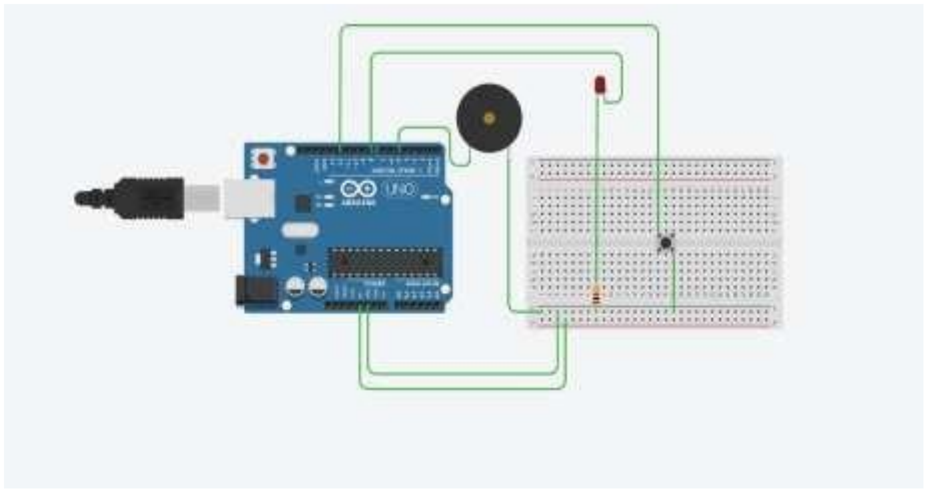


Fig.1: Circuit before starting the simulation

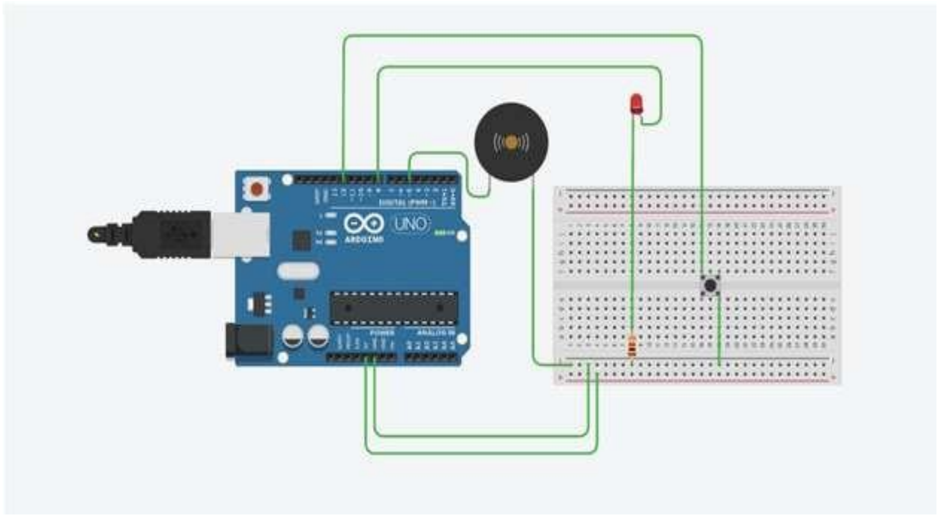


Fig 2: After starting the simulation, if the push button corresponding to the red LED is pressed then the Red LED will be ON and the buzzer starts buzzing.

THEORY:

In this code, if the push button corresponding to the LED is pressed then the LED will be ON and the buzzer starts buzzing. When the button is released then the LED will be OFF and the buzzer stops buzzing

CODE:

```
int buzzer=5;
int led=8;
int button=12;
void setup()
{
pinMode(buzzer, OUTPUT);
pinMode(led, OUTPUT);
pinMode(button, INPUT);
digitalWrite(button, HIGH);
}
void loop()
{
 if(digitalRead(button)==LOW)
 {
digitalWrite(buzzer,HIGH);
digitalWrite(led,HIGH);
 }
 else
 {
digitalWrite(buzzer,LOW);
digitalWrite(led,LOW);
 }
}
```

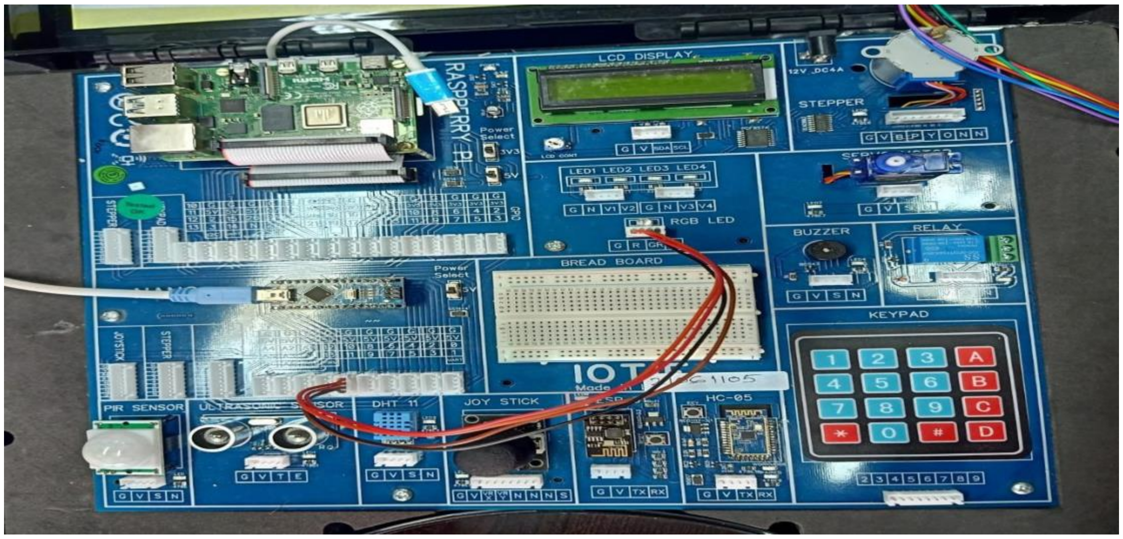
OUTPUT:

After starting the simulation, if the push button corresponding to the red LED is pressed then the Red LED will be ON and the buzzer starts buzzing. When the button is released then the LED will be OFF and the buzzer stops buzzing.

WEEK-3

1. **AIM:** To Interface RGB LED with Aurdino to obtain different colours and brightness using PWM.

COMPONENTS REQUIRED: Arduino board , LED, connecting wires, resistor



THEORY:

The RGB led consists of three different led's, from the name you can guess that these led's are red, green and blue. We can obtain many other colors by mixing up these colors. The Arduino has aanalog write function which will help us in obtaining different colors for Arduino RGB led.

CODE:

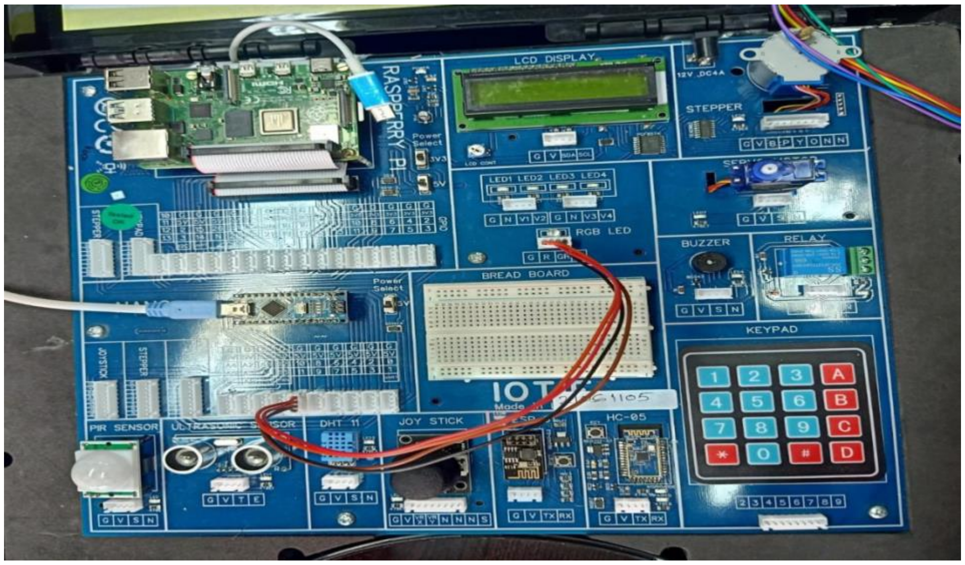
```
int red_light_pin=11;
int  green_light_pin=9;
int blue_light_pin=10;
void setup()
{
pinMode(red_light_pin, OUTPUT);
pinMode(green_light_pin, OUTPUT);
pinMode(blue_light_pin, OUTPUT);
}
void loop()
{
RGB_color(255,0,0); //Red
delay(1000);
RGB_color(0,255,0); //Green
delay(1000);
RGB_color(0,0,255); //Blue
delay(1000);
RGB_color(255,255,125); //Raspberry
delay(1000);
RGB_color(0,255,255); //cyan
delay(1000);
RGB_color(255,0,255); //Magenta
delay(1000);
RGB_color(255,255,0); //Yellow
delay(1000);
RGB_color(255,255,255); //White
delay(1000);
}
voidRGB_color(intred_light_value,intgreen_light_value,intblue_light_value) {
analogWrite(red_light_pin,red_light_value);
analogWrite(green_light_pin,green_light_value);
analogWrite(blue_light_pin,blue_light_value);
}
```

OUTPUT:

The RGB LED displays all the specific colors with a certain time delay.

2. AIM: To display specific color or whatever the color which is given by the user and to display that color.

COMPONENTS REQUIRED: Arduino board, LED, connecting wires , resistor



THEORY:

Inside the RGB led, there are three more led's. So by changing the brightness of these led's, we can obtain many other colors. To change brightness of RGB led, we can use the PWM pins of Arduino. The PWM pins will give signal different duty cycles to the RGB led to obtain different colors. The below RGB color will help us in selecting different colors for Arduino RGB led.

CODE:

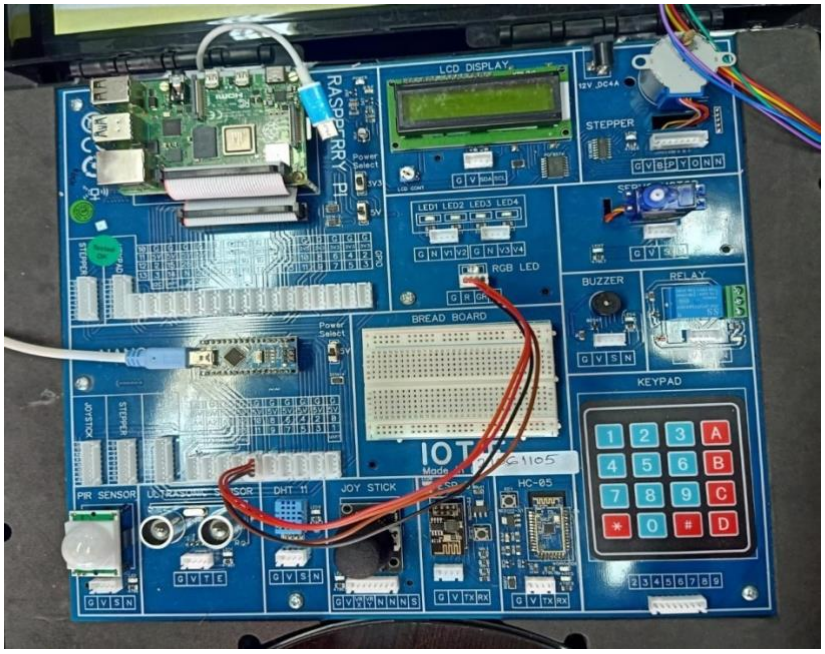
```
intred_light_pin=11;
intgreen_light_pin=9;
intblue_light_pin=10;
String col;
void setup()
{
pinMode(red_light_pin, OUTPUT);
pinMode(green_light_pin, OUTPUT);
pinMode(blue_light_pin, OUTPUT);
Serial.begin(9600); //baud rate
}
void loop()
{
 col=Serial.readString();
Serial.println(col);
 if(col=="RED")
 {
RGB_color(255,0,0);//Red
delay(1000);
 }
 else if(col=="GREEN")
 {
RGB_color(0,255,0);//Green
delay(1000);
 }
 else if(col=="BLUE")
 {
RGB_color(0,0,255);//Blue
delay(1000);
 }
 else if(col=="RASPBERRY")
 {
RGB_color(255,255,125);//Raspberry
delay(1000);
```

```
 }
 else if(col=="CYAN")
 {
RGB_color(0,255,255);//cyan
delay(1000);
 }
 else if(col=="MAGENTA")
 {
RGB_color(255,0,255);//Magenta
delay(1000);
 }
 else if(col=="YELLOW")
 {
RGB_color(255,255,0);//Yellow
delay(1000);
 }
 else
 {
RGB_color(255,255,255);//White
delay(1000);
 }
}
voidRGB_color(intred_light_value,intgreen_light_value,intblue_light_value) {
analogWrite(red_light_pin,red_light_value);
analogWrite(green_light_pin,green_light_value);
analogWrite(blue_light_pin,blue_light_value);
}
```

OUTPUT:

This code displays specific color which the user enters in the serial window

3.AIM: To write a code to display specified RGB colours using loops in Arduino
COMPONENTS REQUIRED: Arduino board , LED, connecting wires,resistor



THEORY:

Inside the RGB led, there are three more led’s. So by changing the brightness of these led’s, we can obtain many other colors. To change brightness of RGB led, we can use the PWM pins of Arduino. The PWM pins will give signal different duty cycles to the RGB led to obtain different colors. The below RGB color will help us in selecting different colors for Arduino RGB led.

CODE:

```
intred_light_pin= 2;
intgreen_light_pin = 3;
intblue_light_pin = 4;
void setup()
 {
pinMode(red_light_pin, OUTPUT);
pinMode(green_light_pin, OUTPUT);
pinMode(blue_light_pin, OUTPUT);
Serial.begin(9600); // baud rate
}
void loop()
{
```

```
for(inti=0;i<256;i=i+60)
 {
for(int j=0;j<256;j=j+60)
 {
for(int k=0;k<256;k=k+60)
 {
RGB_color(i, j, k);
delay(1000);
 }
 }
 }
}
voidRGB_color(intred_light_value, intgreen_light_value, intblue_light_value) {
analogWrite(red_light_pin, red_light_value);
analogWrite(green_light_pin, green_light_value);
analogWrite(blue_light_pin, blue_light_value);
}
```

OUTPUT:

This code displays specific color which the user enters in the serial window

WEEK-4

1) AIM: To control a servo motor using Arduino

COMPONENTS REQUIRED: arduino board, servo motor, connecting wires

THEORY:

Servo motors are a class of motors where you can tell the motor, what position it should move to or what speed it should move .The name servo motor is related to the term servomechanism, which means that the motor is constantly monitored to control its motion. Servo motors move, stop, and start conveyor belts carrying product along to various stages, for example, in product packaging/bottling, and labelling.

CODE:

```
#include <Servo.h>
Servo S1;
intServoPin= 9;
void setup()
{
S1 . attach (ServoPin);
}
void loop ( )
{
S1. write (0);
delay(1000);
S1. write (90);
delay(1000);
S1. write (180);
delay (1000);
}
```

OUTPUT:

The servo motor will rotate to 0 degrees, pause for one second, then rotate to 90 degrees, pause for one second, then rotate to 180 degrees, pause for one second, then start over.

USING FOR LOOP

```
#include <Servo.h>
Servo S1;
intServoPin= 9;
inti;
void setup()
{
S1 . attach (ServoPin);
}
void loop ( )
{
for(i=0;i<180;i=i+10)
{
S1. write (i);
delay (1000);
}
for(i=180;i>0;i=i-10)
{
S1. write (i);
delay (1000);
}
}
```

OUTPUT:

The servo motor will rotate from 0 to 180 degrees with an angle of 10 degrees and it will pause for everyone second when it rotates 10 degrees.

2) AIM: Rotate Stepper motor either clockwise or anti clockwise at ‘n’ number of steps using

Arduino.
COMPONENTS REQUIRED: Arduino board, stepper motor, connecting wires.

THEORY: It consist of 1 Stepper Motor of 200 steps 1.8° which is capable of rotate 200 steps per rotation this means that for each step it rotates approximately 1.8° (360/200).It will move over a certain number of steps in one direction or the other, and tell it the speed at which to step in that direction.

CODE:

```
#include <Stepper.h>
constintstepsPerRevolution = 200;
Stepper myStepper (stepsPer Revolution, 2, 4, 3, 5);
void setup()
{
myStepper.setSpeed (60);
Serial.begin (9600);
}
void loop ()
{
 Serial.println("clockwise");
myStepper.step (stepsPerRevolution);
delay(500);
Serial.printin ("counterclockwise");
mystepper.step (-stepsPerRevolution);
delay (500);
}
```

OUTPUT: The motor will take one revolution in one direction, then one revolution in the other direction

WEEK-5

1. AIM : To display LED using PIR sensor

COMPONENTS REQUIRED: PIR sensor, connecting wires , Arduino board

THEORY:

The PIR motion sensor is ideal to detect movement. PIR stand for “Passive Infrared”. Basically, the PIR motion sensor measures infrared light from objects in its field of view. So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range.

CODE:

```
intsensordata;
void setup()
{
pinMode (13,OUTPUT);
pinMode (8, INPUT);
Serial.begin(9600);
}
void loop ( )
{
sensordata = digitalRead (8);
if (sensordata = = HIGH)
{
digitalWrite(13, HIGH);
Serial.println (“sensor activated");
Serial.print ("motion detected at: ");
Serial.print(millis ()/1000);
Serial. println("sec'");
delay (50);
}
else
{
digital Write (13, LOW);
}
delay (10);
}
```

OUTPUT:

When motion is detected, the LED is turned on. If no motion is detected, the LED turns off.

2. **AIM :** To write a program to determine the temperature and humidity values in an environment
COMPONENTS REQUIRED: DHT sensor , connecting wires , arduino board

THEORY: The DHT11 is a commonly used Temperature and humidity sensor for prototypes monitoring the ambient temperature and humidity of a given area. The sensor can measure temperature from 0°C to 50°C with an accuracy of ±2°C and humidity from 20% to 90% with an accuracy of ±5% RH.

CODE:

```
#include <DHT.h>
DHT dht (8, DHT11);
floath,t;
void setup()
{
Serial.begin (9600);
dht.begin();
Serial.print In ("connect DHT to GV89”); Serial.print In ("starting DHT test"); delay
(2000); void loop()
{
h =dht.readHumidity();
t = dht. readTemperature();
if (isnan (h) || isnan (t))
{
Serial. println ("failed to read from dht"); }
```

```
else
{
Serial.print("Humidity ");
Serial.println (h);
Serial.print ("temperature");
Serial println (t);
}
```

OUTPUT:

DHT sensor placed in any environment detects the level of humidity and temperature values in that place and it is printed continuously in the serial monitor until the supply given to the sensor is halted

WEEK-6

1. AIM : To set up Bluetooth HC-05 with Arduino

COMPONENTS REQUIRED: Bluetooth HC-05 module, connecting wires ,Arduino board

THEORY:

As Bluetooth Module HC-05 works on serial communication. It receives the data from the app and sends it through TX pin of Bluetooth module to RX pin of Arduino. The uploaded code inside Arduino checks the data received. If the receive data is 1, the LED turns ON, and if the received data is 0 the LED turns OFF.

CODE:

```
#include <SoftwareSerial.h>
SoftwareSerialEEBlue (10,11);
void setup() {
Serial.begin (9600);
EEBlue.begin (9600);
Serial.println ("bluetooth gates are open");
}
void loop ()
{
if (EEBlue.available())
Serial.write (EEBlue.read ());
if (Serial.available () )
EEBlue.write (Serial read());
}
```

OUTPUT: When text is entered in the Bluetooth terminal in the mobile phone then it will be displayed in this serial monitor.In the same way if any text is entered in the serial monitor then it will be displayed in the Bluetooth terminal of the mobile phone

2. AIM : To write a program to control the Servo motor rotation which is connected to the arduino using Bluetooth

COMPONENTS REQUIRED: Bluetooth HC-05 module, connecting wires ,arduino board , servo motor

THEORY: The HC-05 is an expensive module that is compatible with wide range of devices including smartphone, laptops and tablets. The HC-05 is a popular module and it communicates with the help of USART at 9600 Baud rate hence it is easy to interface with any microcontroller that supports USART. The android app sends data packets to the Bluetooth module. The Bluetooth modules send this data packet to Arduino Uno through Serial Communication. Arduino Uno is programmed to generate control signal for the servo motor depending upon the value of the data packet.

CODE:

```
#include <SoftwareSerial.h>
SoftwareSerialEEBlue (10,11);
#include<Servo.h>
void setup() {
myservo.attach(6);
pinMode(13,OUTPUT);
Serial.begin (9600);
EEBlue.begin (9600);
Serial.println ("bluetooth gates are open");}
void loop () {
if (EEBlue.available()){
X=EEBlue.read();
        Serial.write (x);
        myservo.write(x);
        delay(1000)
        myservo.write(-x);
        delay(1000)
}
if (Serial.available () )
        x=Serial.read();
        EEBlue.write (x);
```

```
          myservo.write(x);
          delay(1000)
          myservo.write(-x);
          delay(1000)
}
```

OUTPUT:

Whether the input can be given through the serial monitor window or by using the Bluetooth terminal application based on the value or number given by the user the servomotor starts to rotate for that particular angle of rotation only in both clockwise and anticlockwise direction one after the other and they are having a certain delay after each operation.

WEEK-7

AIM: Demonstration of setup & working of Raspberry Pi

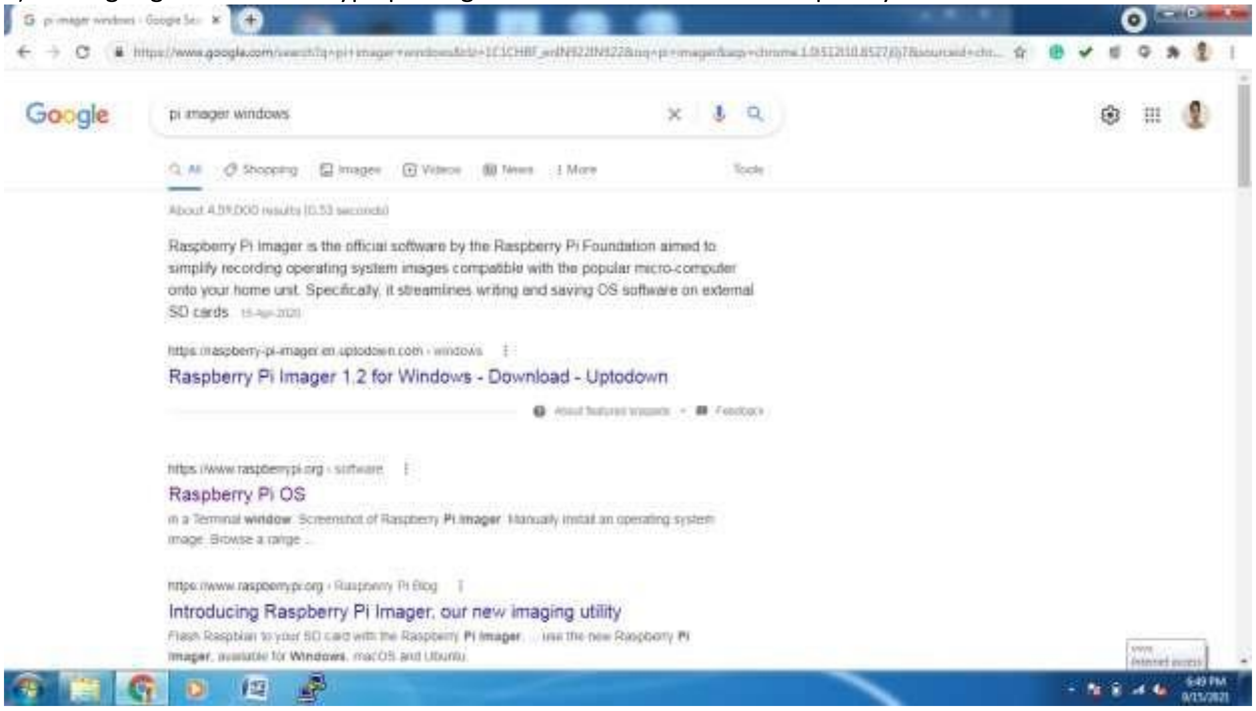
COMPONENTS REQUIRED: Raspberry pi

THEORY:

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries. The original model became more popular than anticipated, selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of HDMI and USB devices.

Steps to install RaspberryPi

1) Go to google search and type pi imager windows and click on 'Raspberry Pi OS'



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

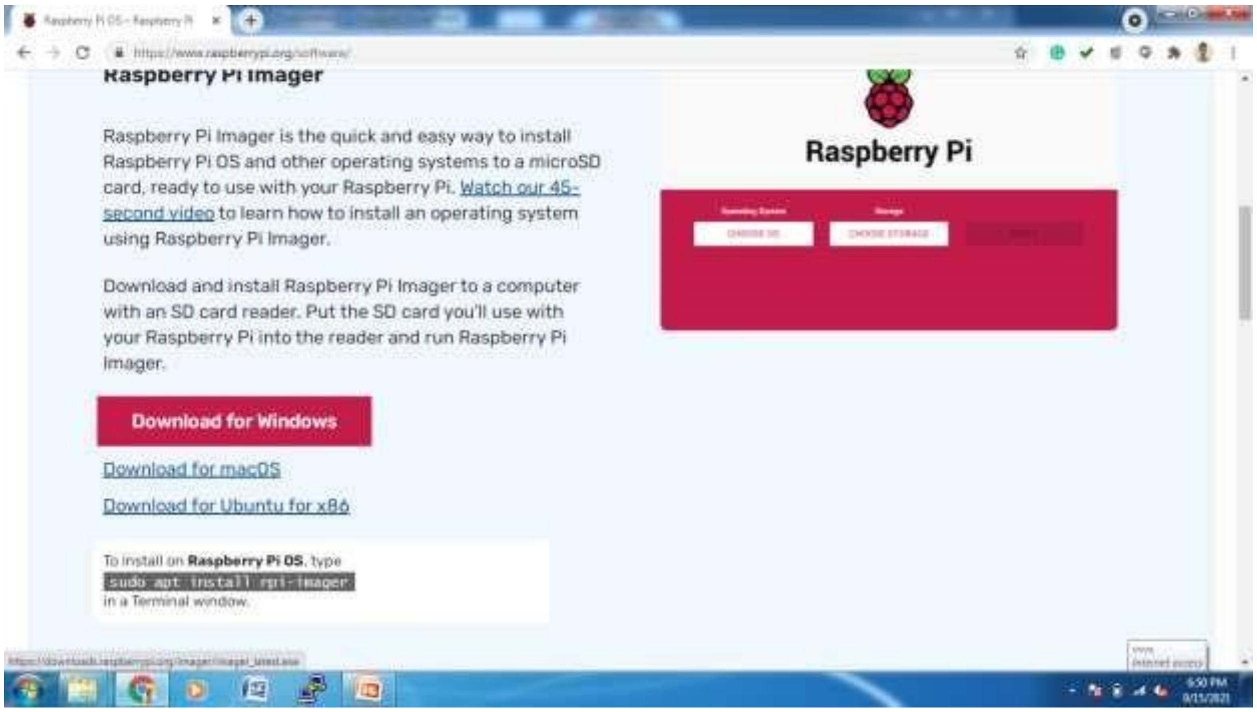
in a Terminal **window**. Screenshot of Raspberry **Pi Imager**. Manually install an operating system image. Browse a range ...

https://www.raspberrypi.org › Raspberry Pi Blog

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.

2) Inside the website click on 'Download for Windows'


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

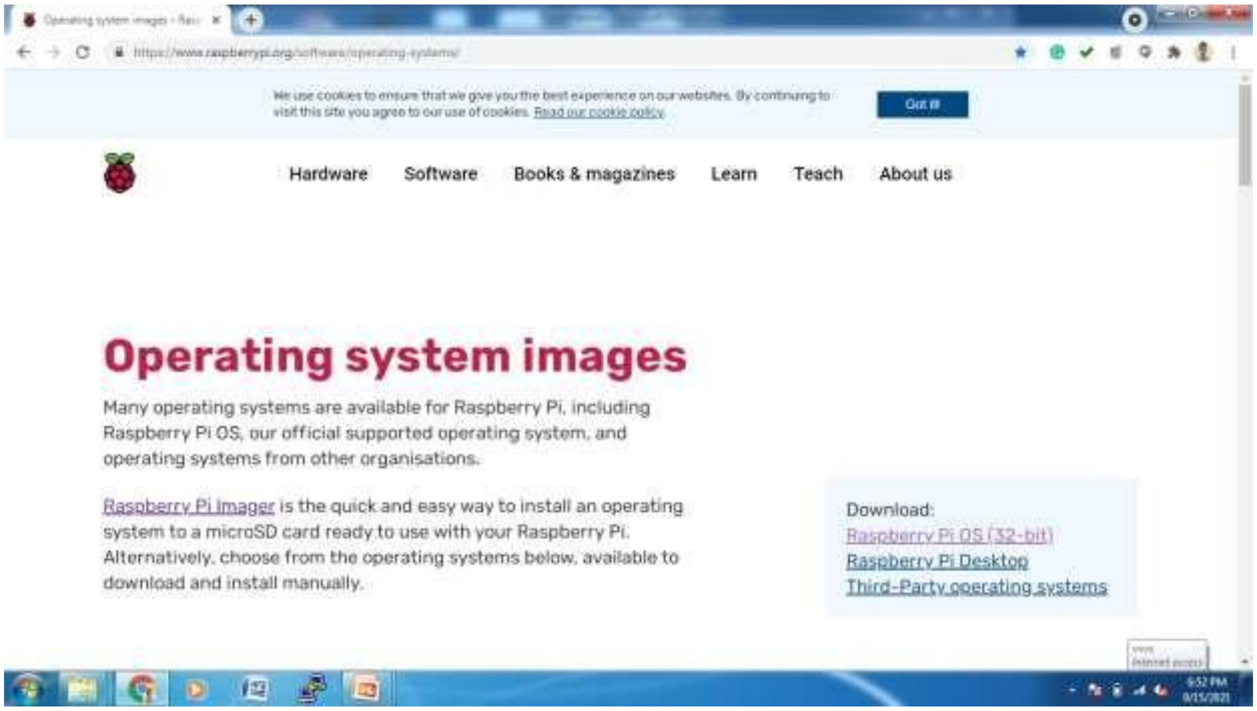
Download for Ubuntu for x86

To install on **Raspberry Pi OS**, type

```
sudo apt install rpi-imager
```

in a Terminal window.

3) Next click on Raspberry Pi OS(32-bit)


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.

Hardware Software Books & magazines Learn Teach About us

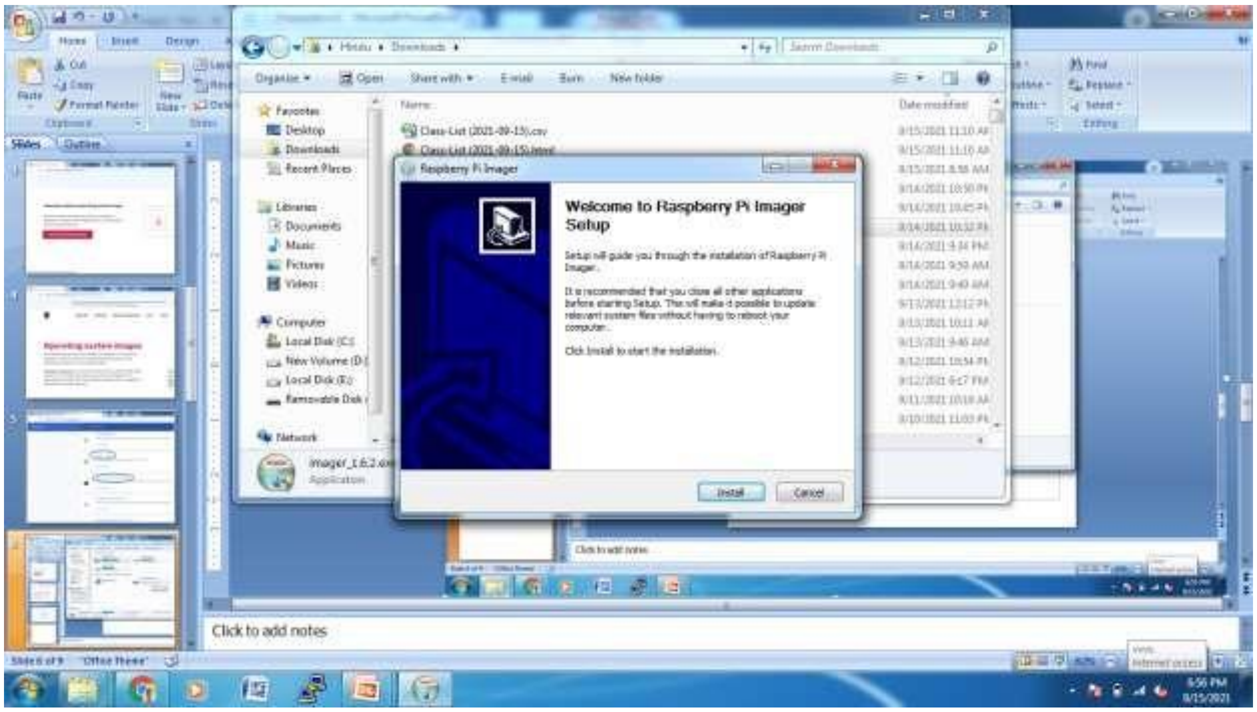
Operating system images

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

4) The below window will pop up showing ‘Welcome to RaspberryPi Imager Setup’ and asks to install. Click on install

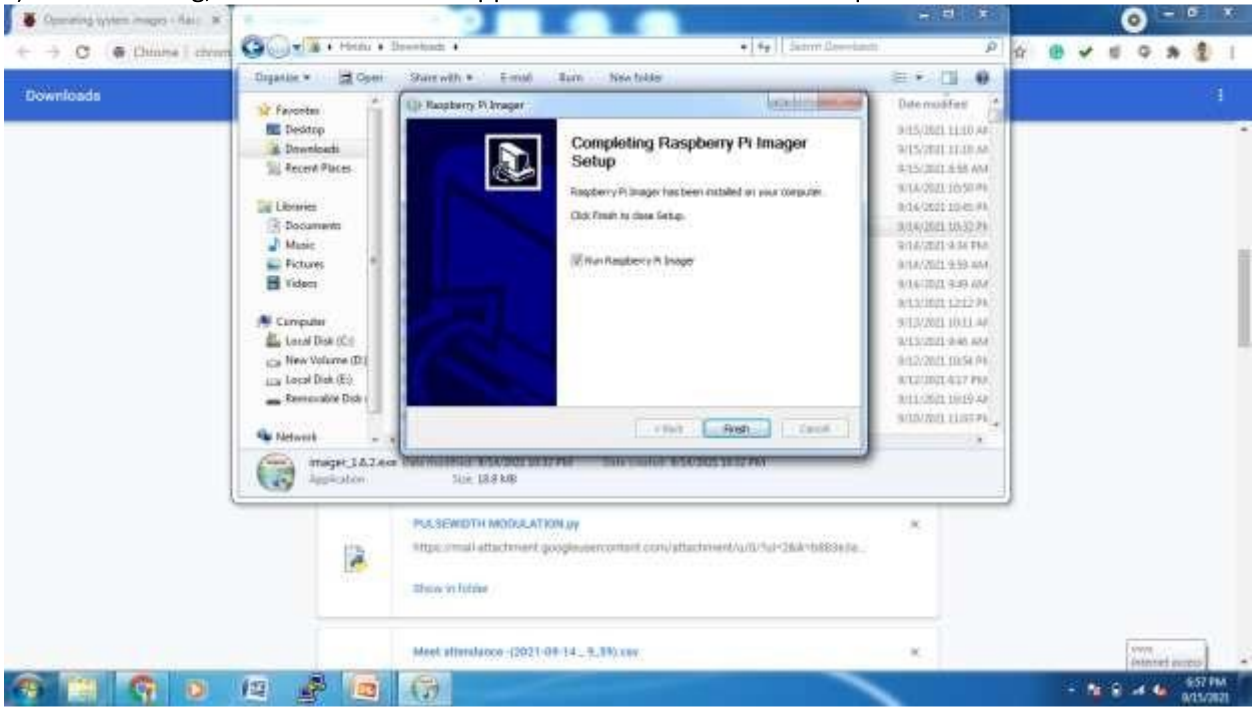

Welcome to Raspberry Pi Imager Setup

Setup will guide you through the installation of Raspberry Pi Imager.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Install to start the installation.

5) After installing,another window will apper as below.Click on finish to proceed further

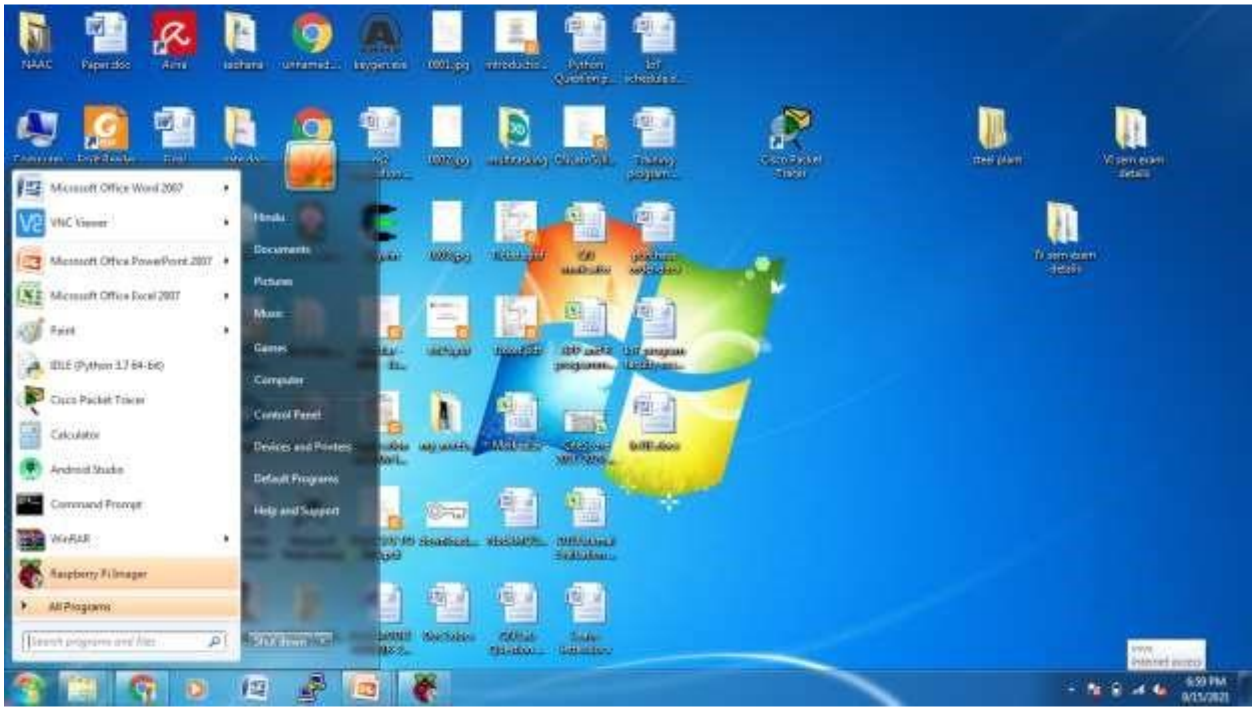

Completing Raspberry Pi Imager Setup

Raspberry Pi Imager has been installed on your computer.

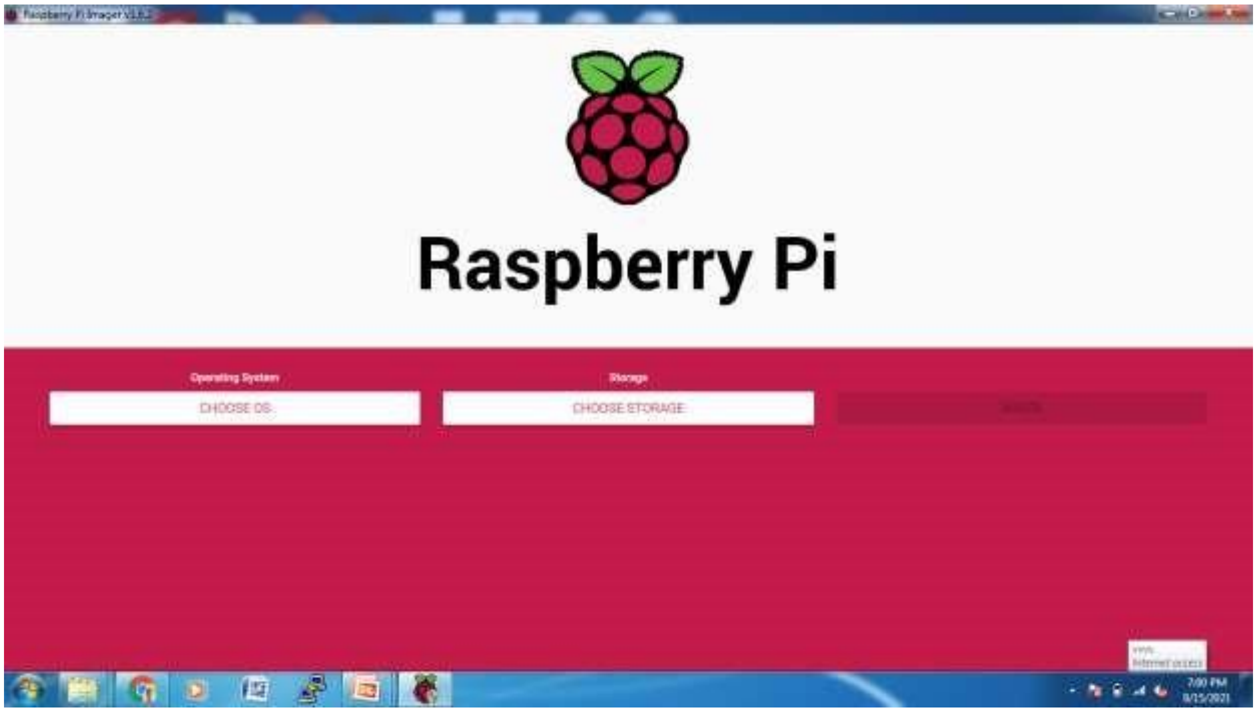
Click Finish to close Setup.

[✓] Run Raspberry Pi Imager

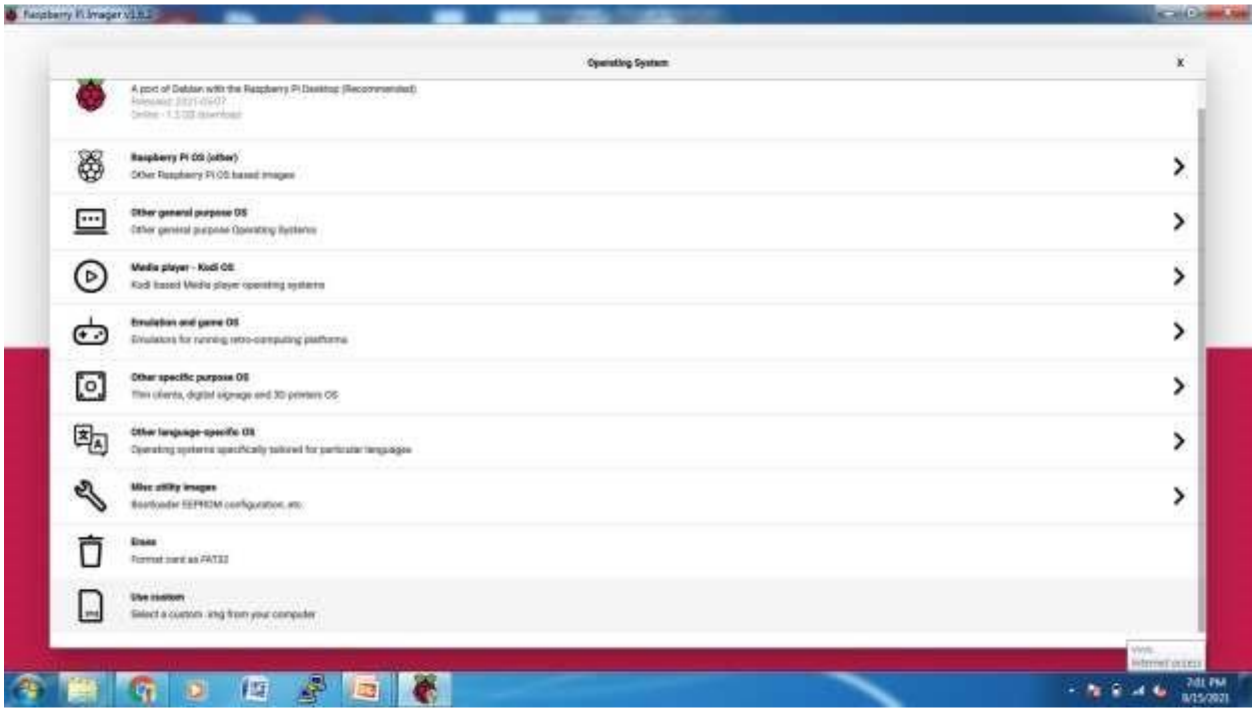
6) Raspberry Pi imager is installed as shown in the below figure.Now click on the the imager

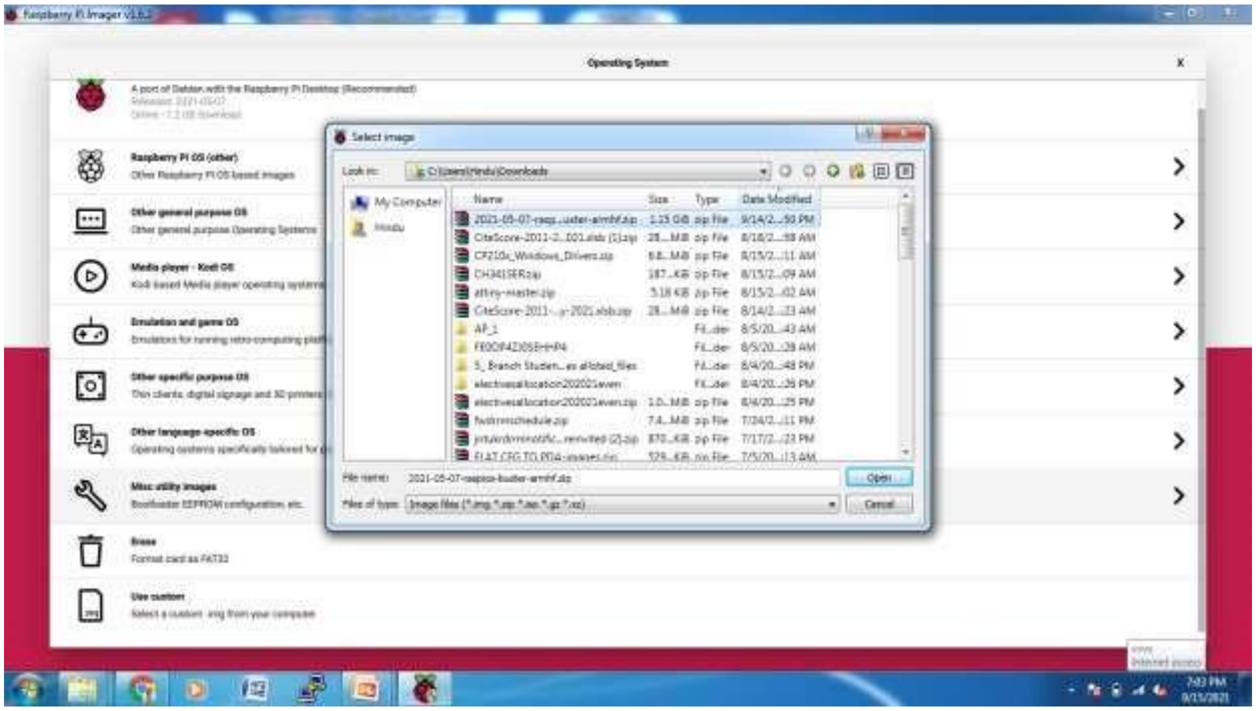


7) After opening the imager,the below window appears.Click on 'CHOOSE OS'

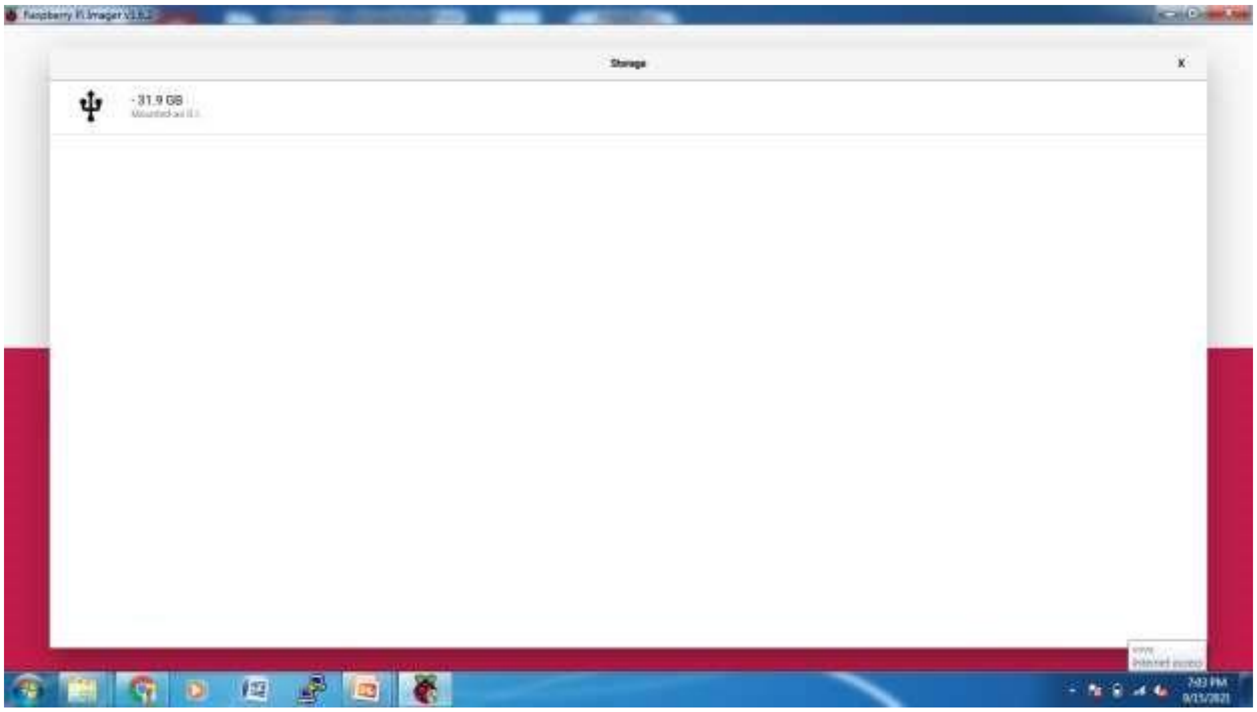


8) Click on Use Custom and choose the operating system.

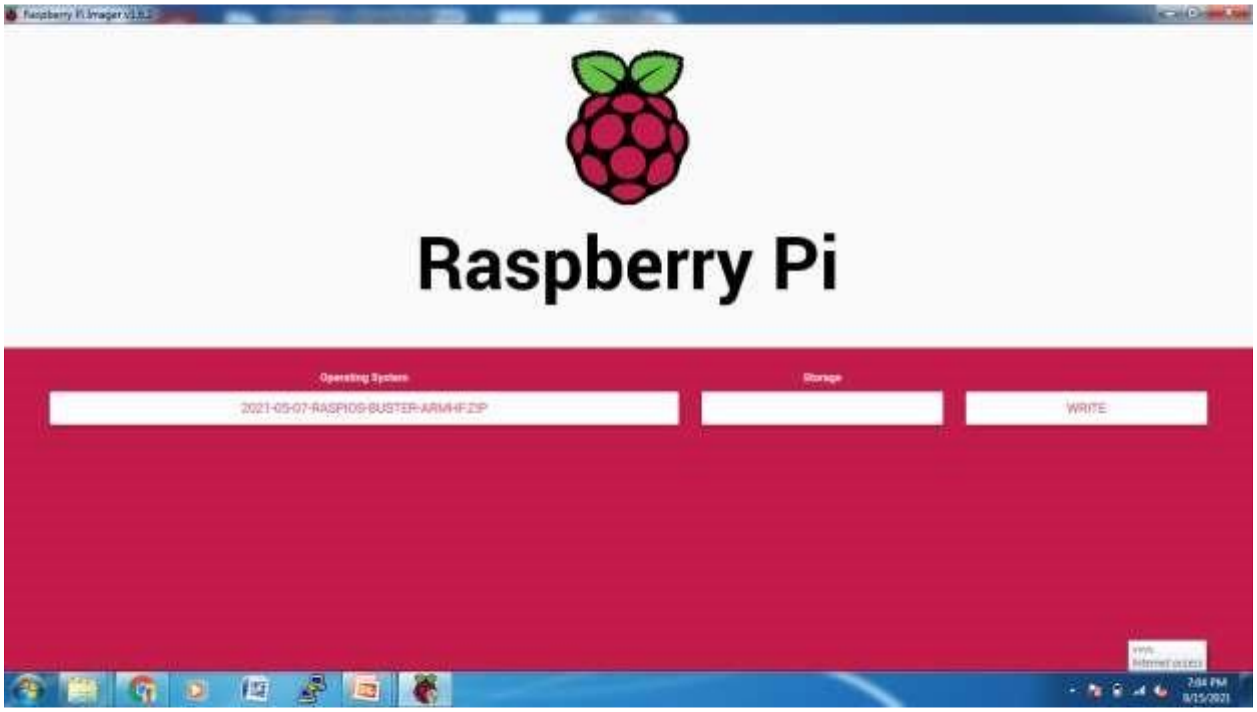

Raspberry Pi Imager v1.6.2
Operating System
A port of Debian with the Raspberry Pi Desktop (Recommended)
Raspberry Pi OS (other)
Other Raspberry Pi OS based images
Other general purpose OS
Other general purpose Operating Systems
Media player - Kodi OS
Kodi based Media player operating systems
Emulation and game OS
Emulators for running retro-computing platforms
Other specific purpose OS
Thin clients, digital signage and 3D printers OS
Other language-specific OS
Operating systems specifically tailored for particular languages
Misc utility images
Bootloader EEPROM configuration, etc.
Erase
Format card as FAT32
Use custom
Select a custom .img from your computer


Raspberry Pi Imager v1.6.2
Operating System
Select image
Look in: C:\Users\Hindu\Downloads
My Computer
Hindu
Name
Size
Type
Date Modified
2021-05-07-raspios-buster-armhf.zip
File name: 2021-05-07-raspios-buster-armhf.zip
Files of type: Image files (*.img *.zip *.iso *.gz *.xz)
Open
Cancel

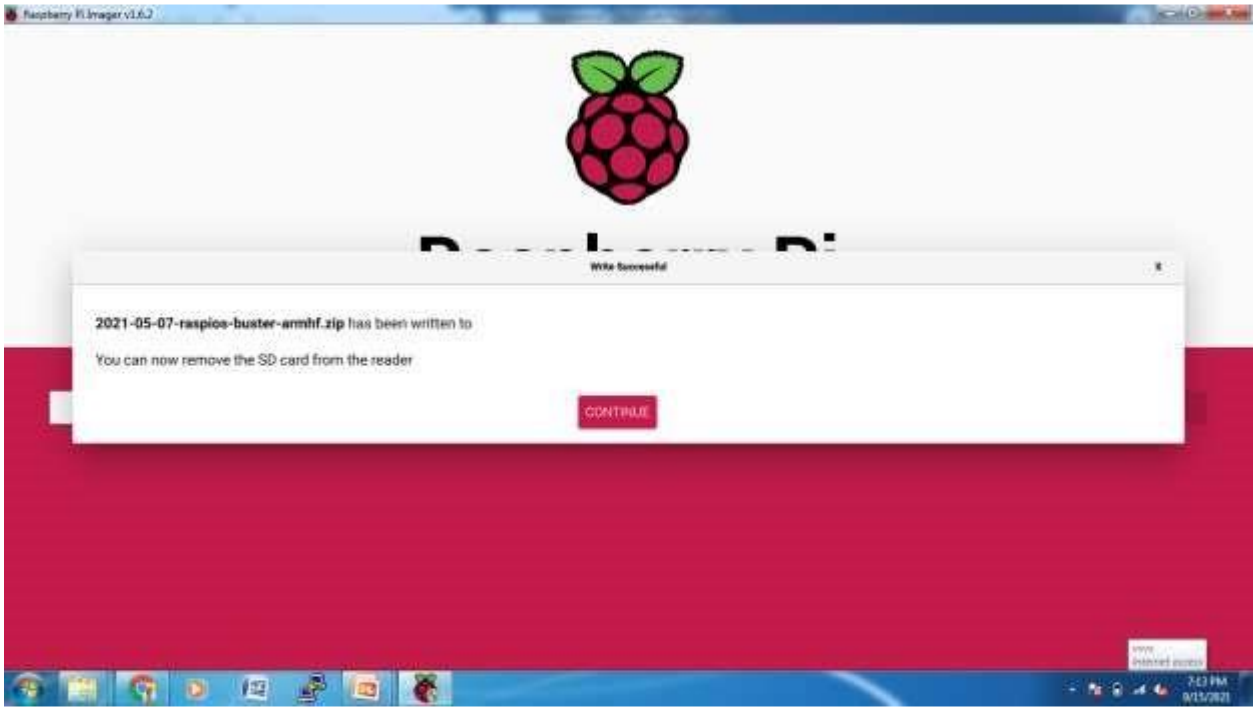
9) Now come back and ‘Choose Storage’



10) And lastly click on Write as shown in the below figure



11) The below window appears and thus Raspberry Pi is installed and Setup



2021-05-07-raspios-buster-armhf.zip has been written to

You can now remove the SD card from the reader

OUTPUT:

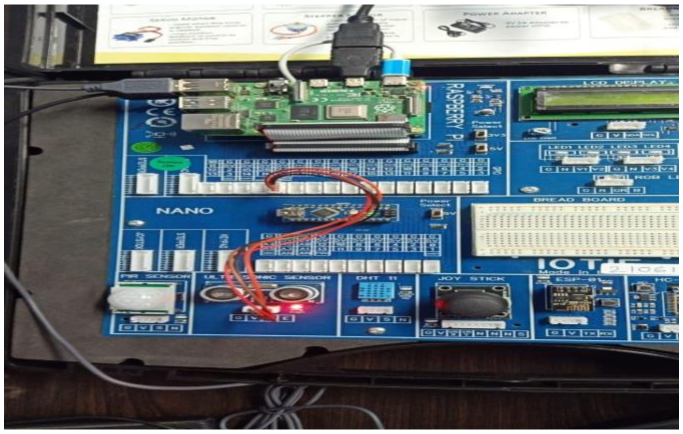
Demonstrated the set up and working of Raspberry Pi.

WEEK-8

1) AIM:

Interface an ultrasonic sensor with Raspberry pi to print distance readings on the monitor when the sensor changes its position.

COMPONENTS REQUIRED: Raspberry Pi kit, monitor, connecting probes



THEORY:

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.

CODE:

```
import RPi.GPIO as GPIO
import time
print("=== ULTRASONIC TEST ===")
print("Connect G 5V 24 25 to G V T E of Ultrasonic")
def distance(trigPin, echoPin):
        GPIO.output(trigPin, True)
time.sleep(0.00001)
        GPIO.output(trigPin, False)
        while GPIO.input(echoPin) == 0:
                pulse_start = time.time()
        while GPIO.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

GPIO.setmode(GPIO.BCM)
# Ultrasonic Pins
trigPin = 24
echoPin = 25
GPIO.setup(trigPin, GPIO.OUT)
GPIO.setup(echoPin, GPIO.IN)
while True:
        dist = distance(trigPin, echoPin)
        print("Measured Distance = {} cm".format(dist))
        time.sleep(0.25)
```

OUTPUT:

The distance is measured continously and is displayed in the serial window until interrupted by the keyboard.

WEEK-9

1) AIM: Interface RGB LED with Raspberry Pi to obtain different colours.

COMPONENTS REQUIRED: Raspberry Pi kit, system, connecting probes.

CIRCUIT DIAGRAM:

THEORY:

Based on the high value that was send to a pin those corresponding color will be emitted out by RGB LED. Here we will observe the Green, Red, Blue, Pink colours simultaneously one after another. These colours will be emitted by the RGB LED continuously until the supply given to the LED is turned off. When any two or three pins are high, we will get another color because of combination of red, green, blue lights. here the Pi board behaves like a CPU to the monitor and it can perform operations related to CPU.

CODE:

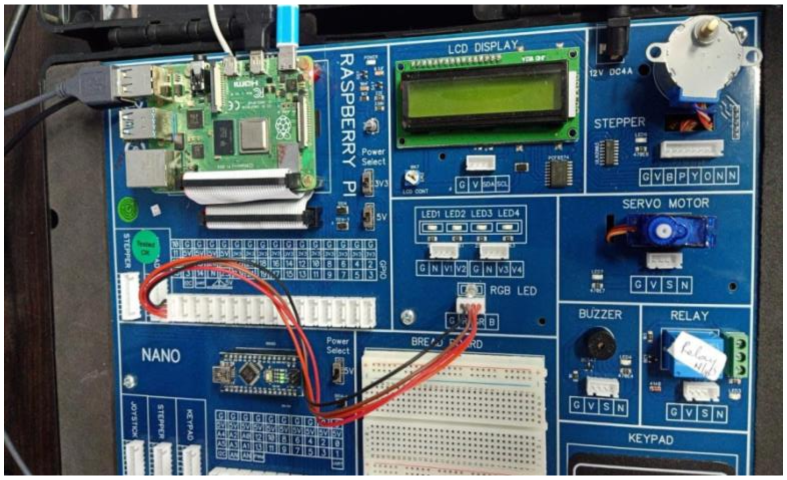
```
import RPi.GPIO as GPIO
import time

print("=== RGB LED TEST ===")
print("Connect 10 11 12 13 to G R Gr B of RGB LED")
GPIO.setmode(GPIO.BCM)
buzzer = 14
commonCathode = 10
red = 11
green = 12
blue = 13

defledColour(colour="none"):
for x in range(10, 14):
```

```
GPIO.setup(x, GPIO.OUT)
GPIO.output(x, GPIO.LOW)
if(colour == "red"):
GPIO.output(red, GPIO.HIGH)
elif(colour == "green"):
GPIO.output(green, GPIO.HIGH)
elif(colour == "blue"):
GPIO.output(blue, GPIO.HIGH)
elif(colour=="pink"):
GPIO.output(red, GPIO.HIGH)
GPIO.output(green, GPIO.HIGH)

while True:
ledColour("green")
time.sleep(2)
ledColour("pink")
time.sleep(2)
ledColour("red")
time.sleep(2)
ledColour("blue")
time.sleep(2)
```

OUTPUT:

Hence a program to obtain different colours by using RGB LED which is being interfaced with the Raspberry Pi board is successfully executed and practically observed**.**

2. AIM: Interface RGB LED with Raspberry Pi to obtain different brightness using PWM. **COMPONENTS REQUIRED:** Raspberry Pi kit, system, connecting probes.



THEORY:

The Raspberry Pi's GPIO gives either 3.3 volts as maximum or zero volts, so the output is a square wave and in order to vary the brightness, we cannot get the voltage between 0-3.3 volts.Therefore, here we use the concept of duty cycle that is the on and off time of the signal and vary the brightness accordingly. The thing which is to be noted here is that with the variation of the duty cycle the brightness of a led can be varied with the help of Pulse Width Modulation(PWM).constantly changing the duty cycle changes the T-on and T-off time and thus we can change the brightness of the led.

CODE:

```
import RPi.GPIO as IO
import time
IO.setwarnings(False)
IO.setmode(IO.BOARD)
IO.setup(19,IO.OUT)
IO.setup(23,IO.OUT)
IO.setup(32,IO.OUT)
IO.setup(33,IO.OUT)
IO.output(19,IO.LOW)
p=IO.PWM(23,100)
p.start(0)
q=IO.PWM(32,100)
r=IO.PWM(33,100)
q.start(10)
r.start(50)
while 1:
for x in range(50):
p.ChangeDutyCycle(x)
q.ChangeDutyCycle(x)
```

```
r.ChangeDutyCycle(x)
time.sleep(0.05)
for x in range(50):
p.ChangeDutyCycle(100-x)
q.ChangeDutyCycle(100-x)
r.ChangeDutyCycle(100-x)
time.sleep(0.05)
```

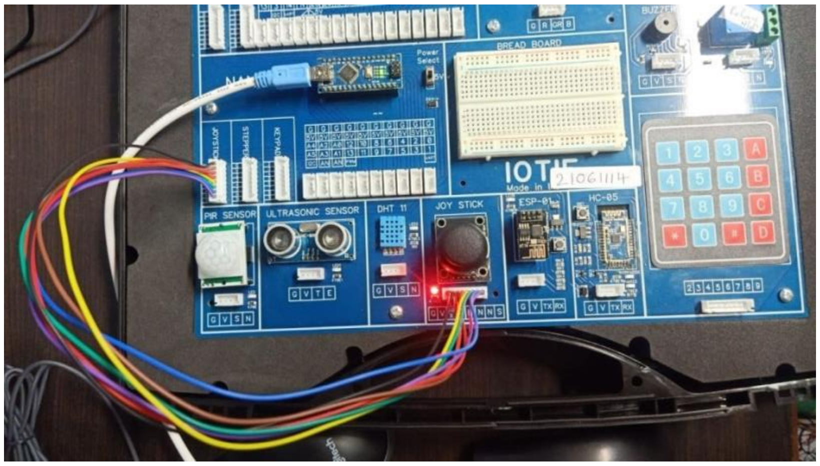
OUTPUT:

Hence a program to obtain different brightnessof RGB LED using PWM , which is being interfaced with the Raspberry Pi board is successfully executed and practically observed.

WEEK-10

1) AIM: Joystick using Arduino board

COMPONENTS REQUIRED: Arduino board, Ioystick, connecting probes



THEORY:

"**Joystick**" is simply a game controller. Because everybody has seen it while playing video games. But, apart from gaming, it can wide range of applications in electronics. Actually, the joystick is the combination of two potentiometers for the X and Y Plane. It gives analog data to the Arduino by reading the voltage from the potentiometer. This analog value get changes as we move the joystick shaft.

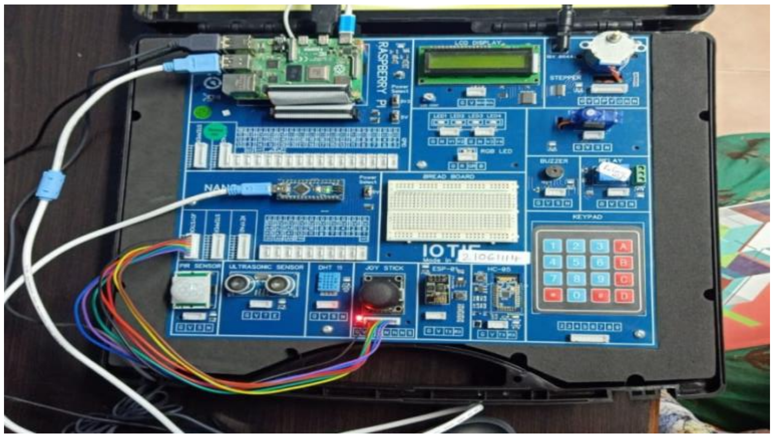
CODE:

```
intVRx = A0;
intVRy = A1;
int SW = 2;

intxPosition = 0;
intyPosition = 0;

intmapX = 0;
intmapY = 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.println(mapY);
  delay(100);
}
```

OUTPUT: Thus Joystick measured and displayed the X-axis,Y-axis and Z-axis direction.

2) AIM: Reading the data from an analog sensor with Raspberry using Arduino serial port or ADC MCP3208 using SPI.

COMPONENTS REQUIRED: Raspberry Pi kit, monitor, connecting probes



THEORY:

"**Joystick**" is simply a game controller. Because everybody has seen it while playing video games. But, apart from gaming, it can wide range of applications in electronics. Actually, the joystick is the combination of two potentiometers for the X and Y Plane. It gives analog data to the Arduino by reading the voltage from the potentiometer and it is given to raspberry pi. So that we can get the readings of joystick by using raspberry pi.We cannot give the joystick directly to raspberry pi as it won't allow analogdata to run.Thisanalog value get changes as we move the joystick shaft.

CODE:

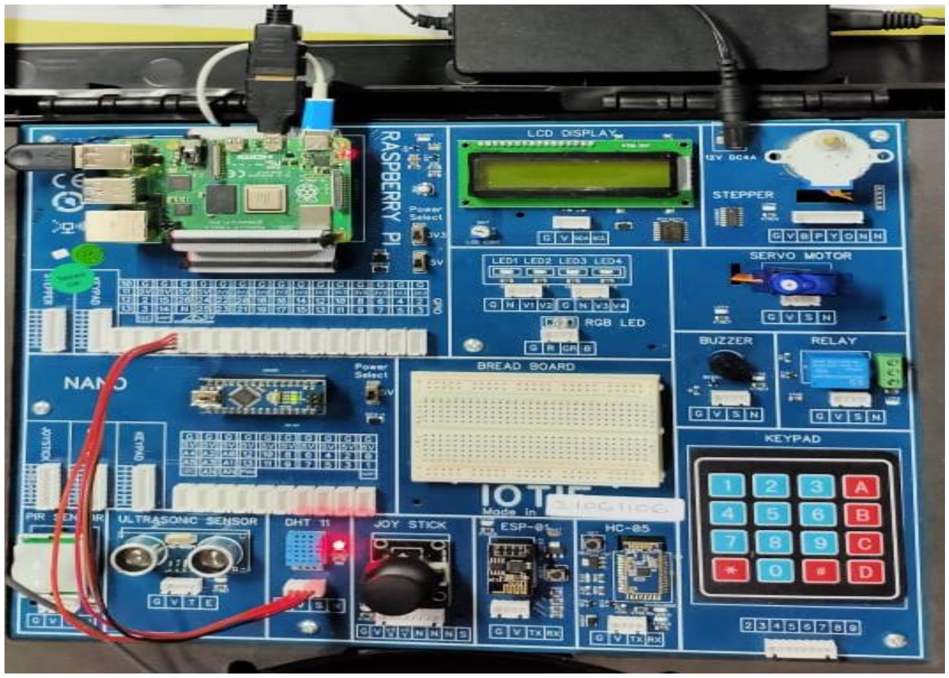
```
import serial
if __name__=='__main__':
    ser = serial.Serial('/dev/ttyUSB0', 9600, timeout =1)
    ser.reset_input_buffer()
    while True:
        if ser.in_waiting>0:
            X = ser.readline().decode('utf-8').rstrip()
            print(X)
```

OUTPUT:

Thus read the data from an analog sensor with Raspberry using Arduino serial port or ADC MCP3208 using SPI and displayed the X-axis,Y-axis and Z-axis direction.

WEEK-11

1) AIM: To Write a program that post/read the data from the think speak webserver via MQTT broker with a Raspberry Pi.



COMPONENTS REQUIRED: Raspberry Pi kit, monitor, connecting probes

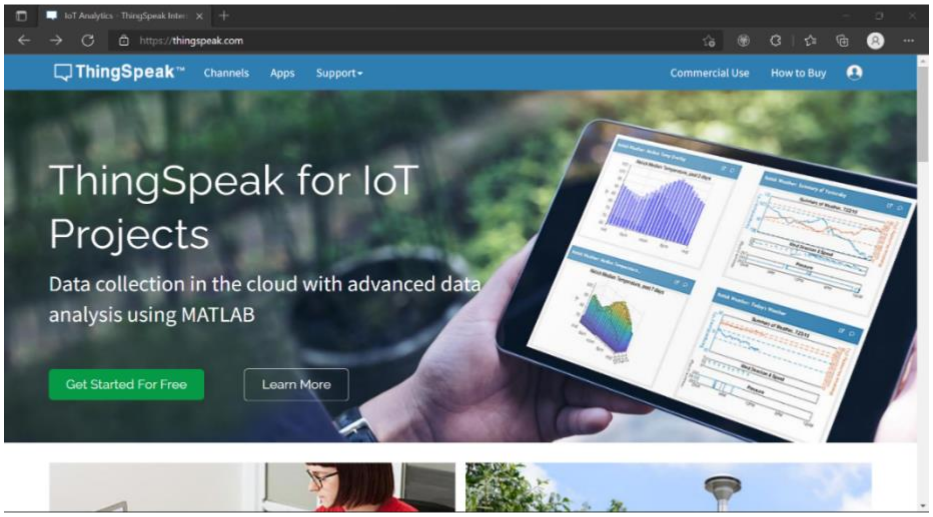
THEORY:

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. DHT11 is a Digital Sensor consisting of a Negative Temperature Coefficient Temperature Sensor, a Resistive-type Humidity Sensor and an 8-bit Microcontroller to convert the analog signals from these sensors and produce a Digital Output. The output from the DHT11 Sensor is Digital.

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data Streams in the cloud. Data can be sent to ThingSpeak from devices, create instant visualization of live data, and send alerts.

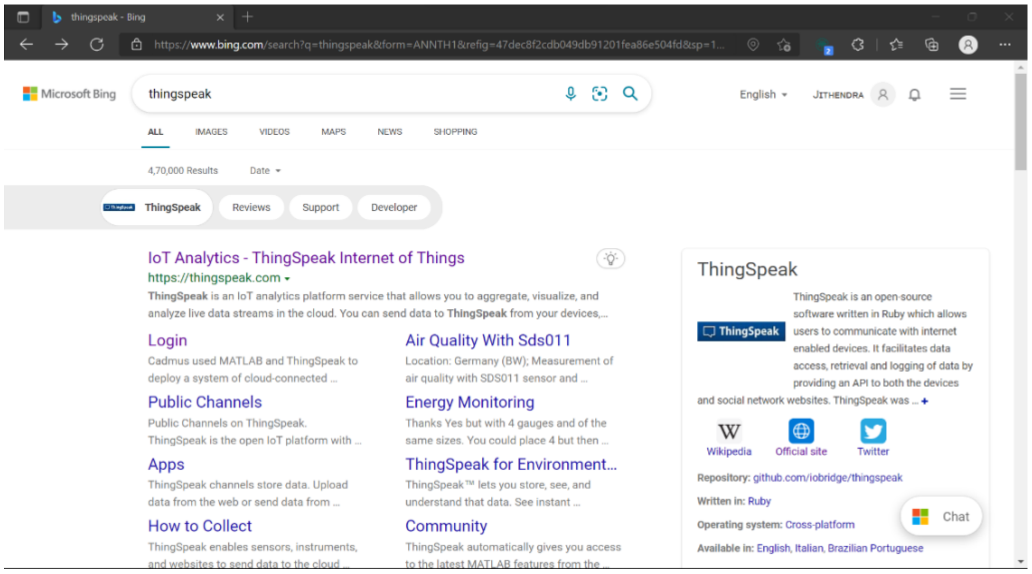
Creating a channel in ThingSpeak Cloud:

Go to the search engine of your personal computer and search for the official website of ThingSpeak


IoT Analytics - ThingSpeak Inter
https://thingspeak.com
ThingSpeak™
Channels
Apps
Support
Commercial Use
How to Buy
ThingSpeak for IoT Projects
Data collection in the cloud with advanced data analysis using MATLAB
Get Started For Free
Learn More

.

Click on the website and signup for an account by clicking on the top right corner of the page as shown in below figure.



IoT Analytics - ThingSpeak Internet of Things
https://thingspeak.com
ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to **ThingSpeak** from your devices,...

Login
Cadmus used MATLAB and ThingSpeak to deploy a system of cloud-connected ...

Air Quality With Sds011
Location: Germany (BW); Measurement of air quality with SDS011 sensor and ...

Public Channels
Public Channels on ThingSpeak. ThingSpeak is the open IoT platform with ...

Energy Monitoring
Thanks Yes but with 4 gauges and of the same sizes. You could place 4 but then ...

Apps
ThingSpeak channels store data. Upload data from the web or send data from ...

ThingSpeak for Environment...
ThingSpeak™ lets you store, see, and understand that data. See instant ...

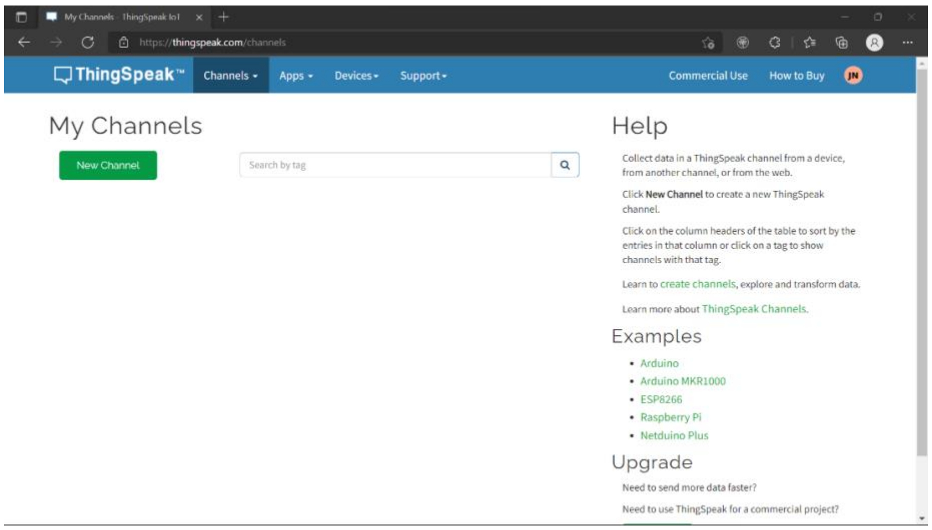
How to Collect
ThingSpeak enables sensors, instruments, and websites to send data to the cloud ...

Community
ThingSpeak automatically gives you access to the latest MATLAB features from the ...

ThingSpeak
ThingSpeak is an open-source software written in Ruby which allows users to communicate with internet enabled devices. It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites. ThingSpeak was ... +
Wikipedia Official site Twitter
Repository: github.com/iobridge/thingspeak
Written in: Ruby
Operating system: Cross-platform
Available in: English, Italian, Brazilian Portuguese

After signing up, click New Channel to create a new ThingSpeak channel.

Give the name of the channel, its description and field name as Temperature and Humidity as shown in below figure and save the channel.



My Channels

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

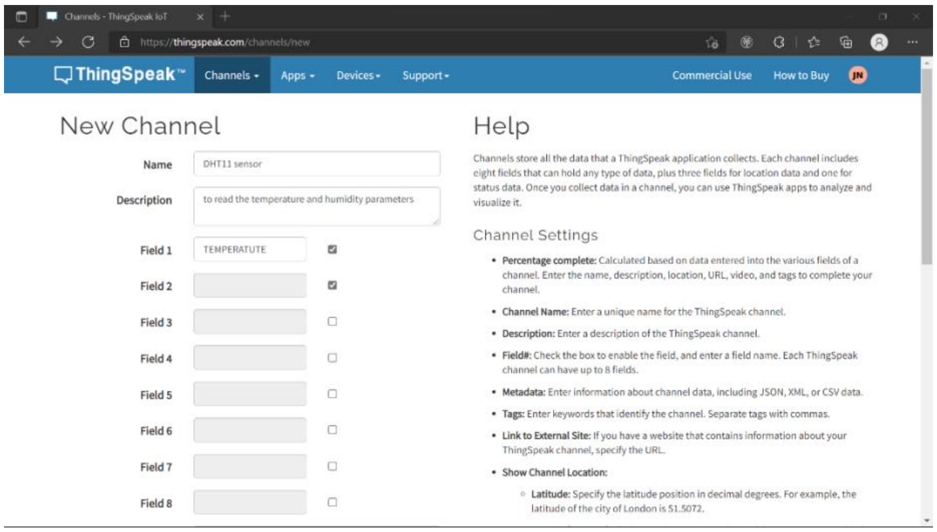
Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?



New Channel

Name	DHT11 sensor	
Description	to read the temperature and humidity parameters	
Field 1	TEMPERATUTE	☑
Field 2		☑
Field 3		☐
Field 4		☐
Field 5		☐
Field 6		☐
Field 7		☐
Field 8		☐

Help

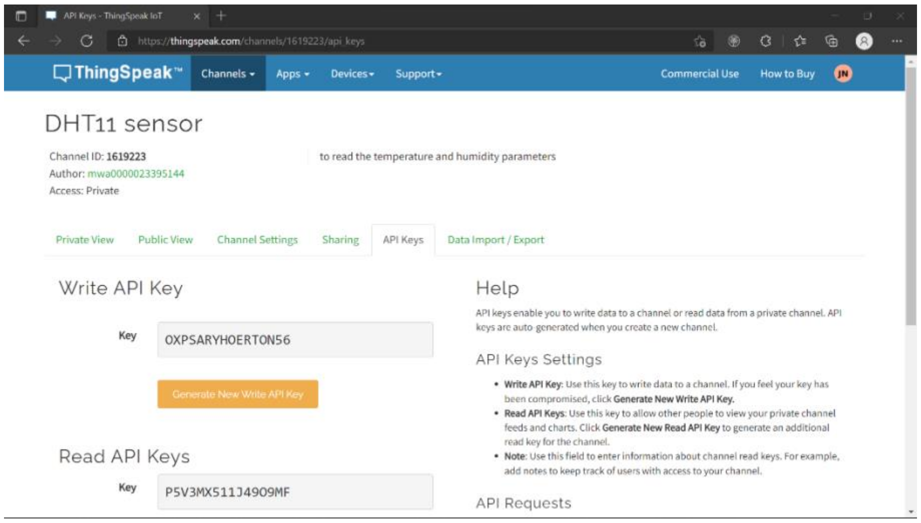
Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.

After saving the channel go to the API Keys menu which enables to write data to a channel or read data from a private channel.

Copy the Write API Key as the data is written to the channel


API Keys - ThingSpeak IoT
https://thingspeak.com/channels/1619223/api_keys
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
JN

DHT11 sensor

Channel ID: **1619223**
Author: mwa0000023395144
Access: Private

to read the temperature and humidity parameters

Write API Key

Key: OXPSARYHOERTON56

Read API Keys

Key: P5V3MX511J4909MF

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

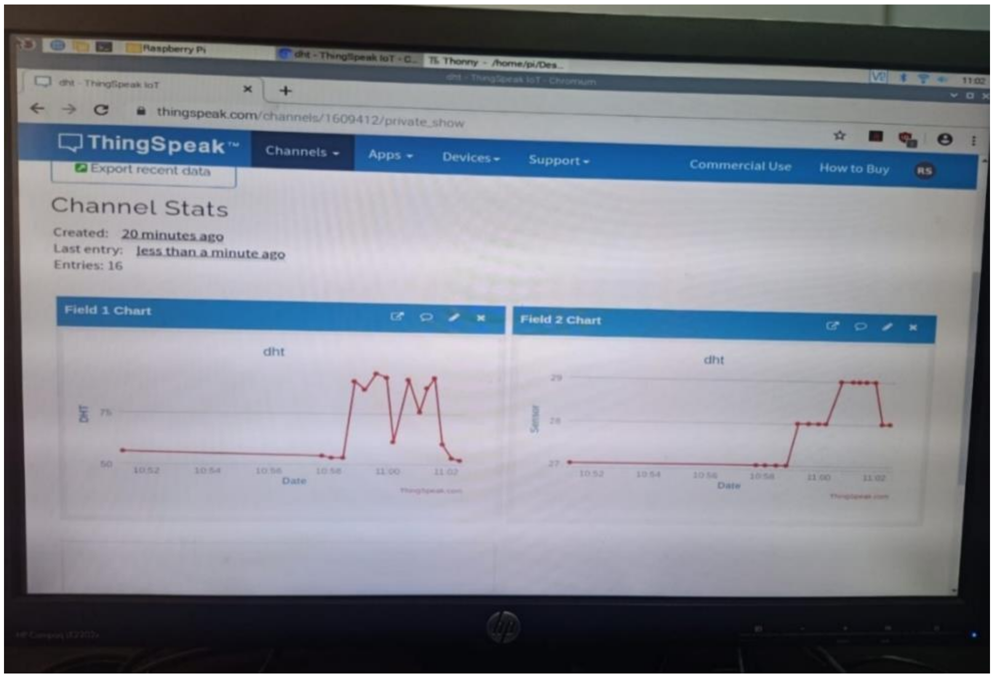
CODE:

```
importurllib.request
import time
import RPi.GPIO as GPIO
import Adafruit_DHT

writeAPIKey = "8CMZMVX6CQELK7KG" # enter write API key here
baseURL =
"https://api.thingspeak.com/update?api_key={}".format(writeAPIKey) sensor =
Adafruit_DHT.DHT11
sensorPin = 26
GPIO.setmode(GPIO.BCM) # Using BCM numbering

try:
 while True:
humidity, temperature = Adafruit_DHT.read_retry(sensor, sensorPin)
 if humidity is not None and temperature is not None:
 # Format the readings to two decimal places
 humidity = '%.2f' % humidity
 temperature = '%.2f' % temperature
 # Sending the readings to thingspeak
conn = urllib.request.urlopen(baseURL + '&field1={}&field2={}'.format(humidity, temperature))
print(conn.read())

 # Closing the connection
```

```
conn.close()

time.sleep(20) # wait 20 seconds before uploading next reading except
 KeyboardInterrupt:
GPIO.cleanup()
exit()
```

OUTPUT:

The DHT11 sensor is interfaced with Raspberry Pi and the readings of Temperature and Humidity are successfully posted on the ThingSpeak cloud and also observed the variations of DHT11 sensor in the field charts as shown in below figure.

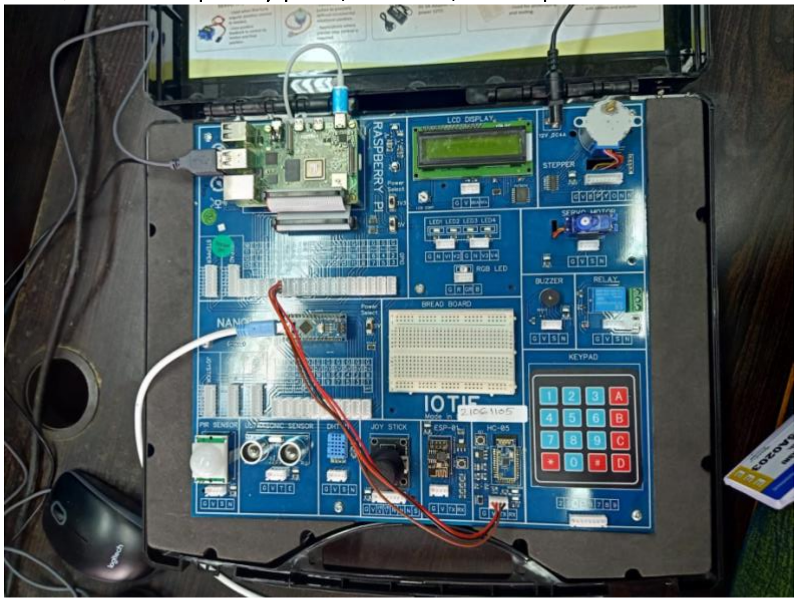


Channel Stats

Created: 20 minutes ago
Last entry: less than a minute ago
Entries: 16

WEEK-12

AIM: Send real-time sensor data to a smartphone using RaspberryPi on board Bluetooth
COMPONENTS REQUIRED: Raspberry pi kit, monitor, smart phone with Bluetooth app installed.



THEORY:

On the raspberryPi board monitor we are having a bluetooth connection symbol at the bottom right corner click on 'add a device' and check for the device or mobile bluetooth device name in which mobile bluetooth is already switched ON. Next pair the bluetooth devices by clicking on the 'allow' and 'pair' button/options. After these steps the Pi board and the mobile device are connected to the bluetooth and they can transfer data in between them. BY placing an object in front of an ultrasonic sensor it can calculate the distance corresponding to it and it will be printed on the monitor screen and the mobile device screen which is connected through bluetooth. On the mobile the distance measured by the sensor can be observed by opening 'Bluetooth terminal' connection.

CODE:

```
import bluetooth
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
trigPin = 24
echoPin = 25
GPIO.setup(trigPin, GPIO.OUT)
GPIO.setup(echoPin, GPIO.IN)

host =""
```

```
port=1
server = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
try:
server.bind((host,port))
print("bluetooth binding successful")
except:
print("bluetooth binding unsuccessful")
server.listen(1) # enables the server to listen for incoming connections client, address=
server.accept()
print("connection received from",address)
print("Client:", client)
print("=== ULTRASONIC TEST ===")
print("Connect G 5V 24 25 to G V T E of Ultrasonic")

def distance(trigPin, echoPin):
GPIO.output(trigPin, True)
time.sleep(0.00001)
GPIO.output(trigPin, False)

 while GPIO.input(echoPin) == 0:
pulse_start = time.time()

 while GPIO.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( "\nMeasured Distance : ")
client.send(str(dist))
client.send("cm")
time.sleep(2)
```

OUTPUT:

Hence a program for sending the real-time sensor data values to a smartphone using Raspberry Pi on boardbluetooth has been successfully executed.