**MASTER OF COMPUTER APPLICATIONS**

**2021-23**



Affiliated to



Guru Gobind Singh Indraprastha University

GGSIPU, Golf Course Rd, Sector 16 C, Dwarka, Delhi, 110078

**M.C.A. 3rd Semester**

**IOT Lab Assignment**

**LAB FILE**

**Submitted to: Submitted by:**

**Anil Kumar**

**(02711804421)**

**Ms. Kiran Walia**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Experiment Number**. | **Topic** | **Signature** |
| 1 | Study of Arduino Architecture. |  |
| 2 | Simulation of Led Glow With Arduino Board using tinkercad |  |
| 3 | Simulation of Led Glow in series pattern with Arduino Board using tinkercad |  |
| 4 | Simulation of Led Glow in parallel pattern with Arduino Board using tinkercad |  |
| 5 | Simulation of Led Glow using push button with Arduino Board using tinkercad |  |
| 6 | Simulation of changing the intensity of led using potentiometer with Arduino Board using tinkercad |  |
| 7 | Study of RaspberryPi architecture and pin diagram |  |
| 8 | Setup of raspberry pi board |  |
| 9 | Led Connection using RaspberryPi |  |
| 10 | Interfacing of DHT sensor using raspberry pi |  |
| 11 | Interfacing of IR sensor using raspberry pi |  |
| 12 | Interfacing of flame sensor using raspberry pi |  |
| 13 | Visualization od DHT Sensor data on cloud (ThingSpeak) |  |

**Experiment Number 1:-**

**Aim: -**The goal of Arduino is to create an accessible way for software developers to enter the world of microcontroller programming.

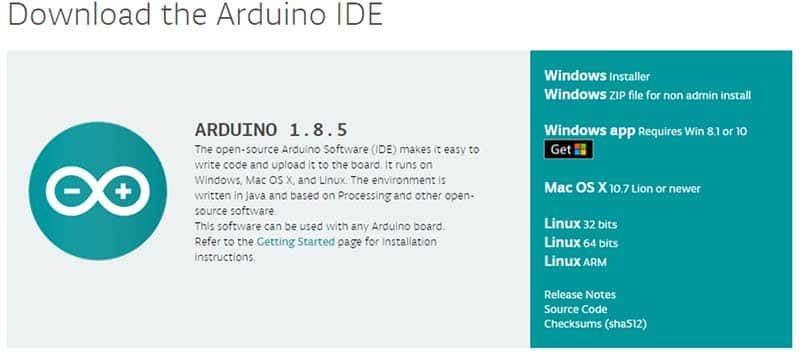
**Apparatus Used:**- Needle-nose Pliers, Wire Strippers,PrecisionScrewdriver Set,Flush Cutters, Fine Tip Straight Tweezers,Digital Multimeter,Soldering Iron , Pana vise Jr andSolder Sucker

**Theory: -** Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex.  This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world.   By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays.  Because of it’s flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects.

**Procedure: -**

**Step 1: Download and Install the IDE**

You can download the IDE from the official Arduino website.



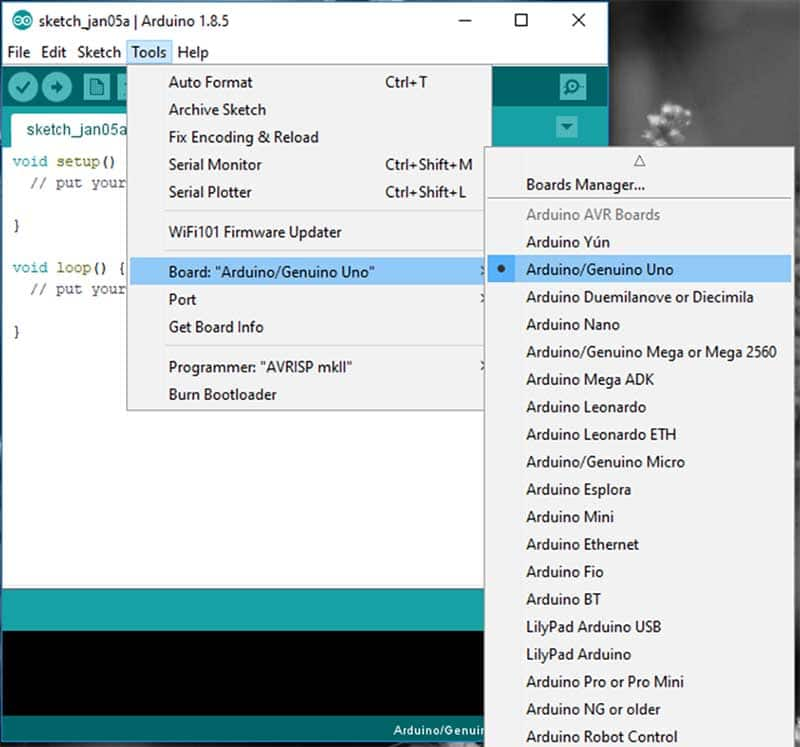
Once downloaded, install the IDE and ensure that you enable most (if not all) of the options, INCLUDING the drivers.

### Step 2: Get the Arduino COM Port Number

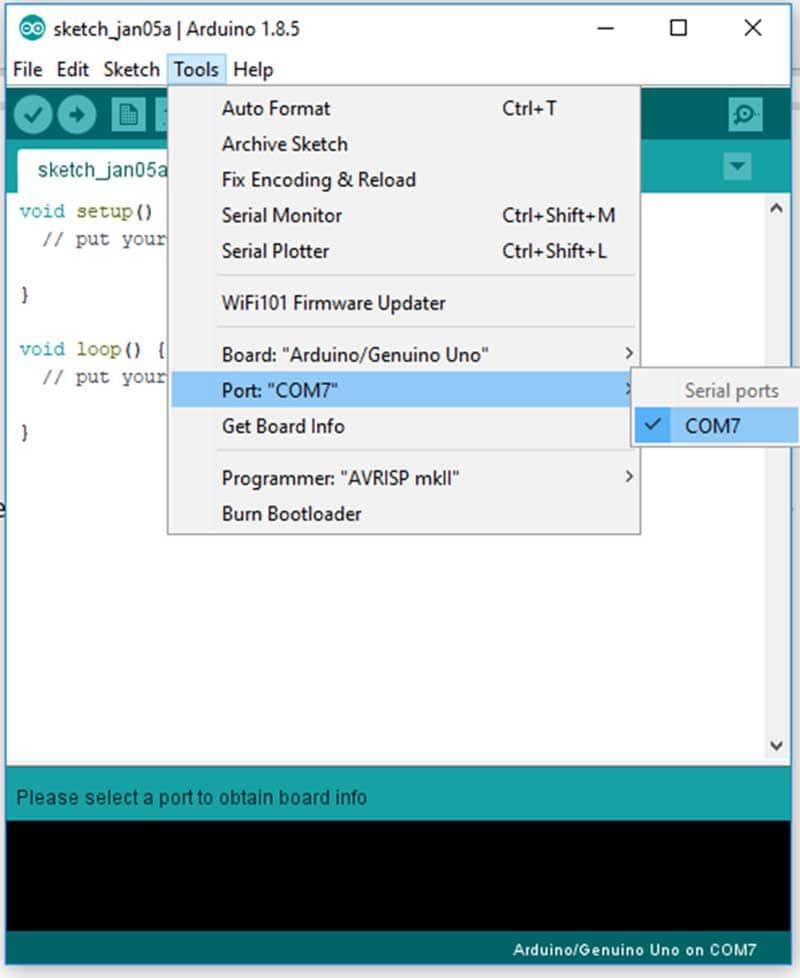
Next, you’ll need to connect the Arduino Uno board to the computer.This is done via a USB connection we do not need to provide power to the Arduino, as the USB provides 5V up to 2A. When the Arduino is connected, the operating system should recognize the board as a generic COM port (for example, my Arduino Uno uses a CH340G, which is an RS-232 serial to USB converter). Once it’s recognized, we will need to find out what port number it has been assigned. The easiest way to do this is to type “device manager” into Windows Search and select Device Manager when it shows.

### Step 3: Configure the IDE

Now that we have determined the COM port that the Arduino is on, it’s time to load the Arduino IDE and configure it to use the same device and port. Start by loading the IDE. Once it’s loaded, navigate to Tools > Board > Arduino Uno. However, if you are using a different board (i.e., not the Arduino Uno), you must select the proper board!

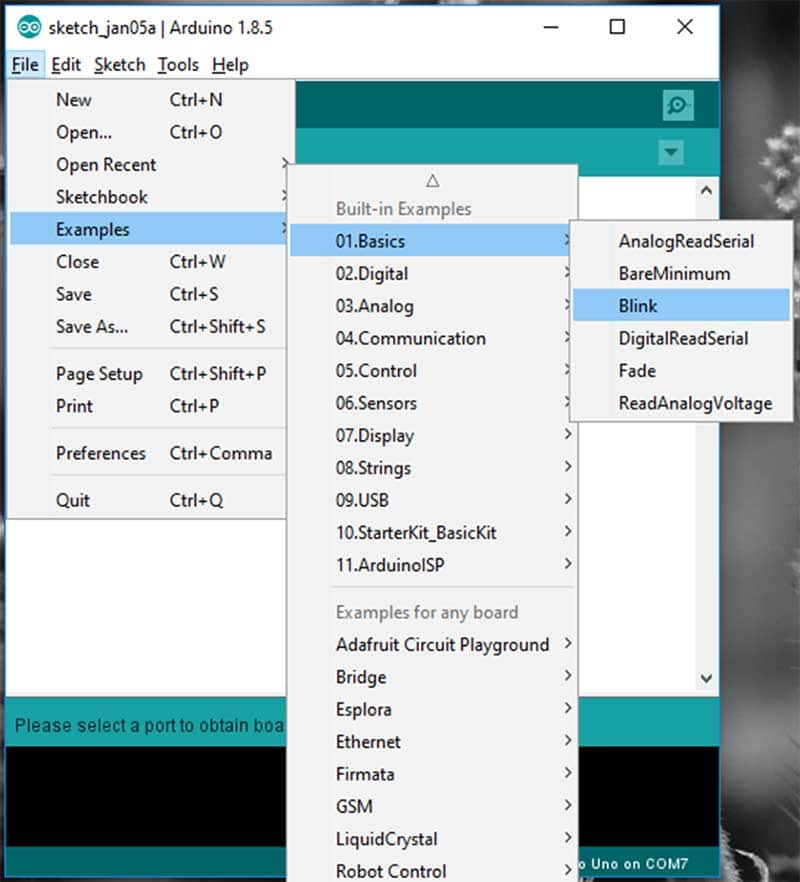


Next, you must tell the IDE which COM port the Arduino is on. To do this, navigate to Tools > Port > COM7. Obviously, if your Arduino is on a different port, select that port instead.



### Step 4: Loading a Basic Example

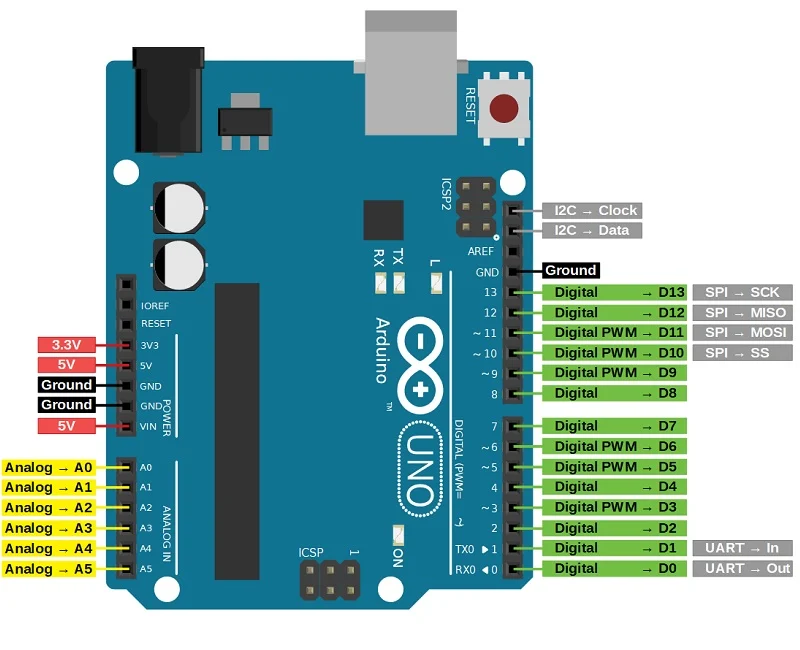
For the sake of simplicity, we will load an example project that the Arduino IDE comes with. This example will make the onboard LED blink for a second continuously. To load this example, click File > Examples > 01. Basics > Blink.



### Conclusion

The Arduino is a powerful prototyping tool for many reasons, including its lack of a dedicated programmer, its wide range of available libraries, and the simplicity of its IDE. While we only got a light to blink in this project, you can expect much more in the future. Try your hand at interfacing with displays, taking measurements, talking over the internet, and possibly even working with AI!

**Pin Diagram: -**

****

Experiment No. 2

**Aim :**Simulation of Led Glow With Arduino Board using tinkercad

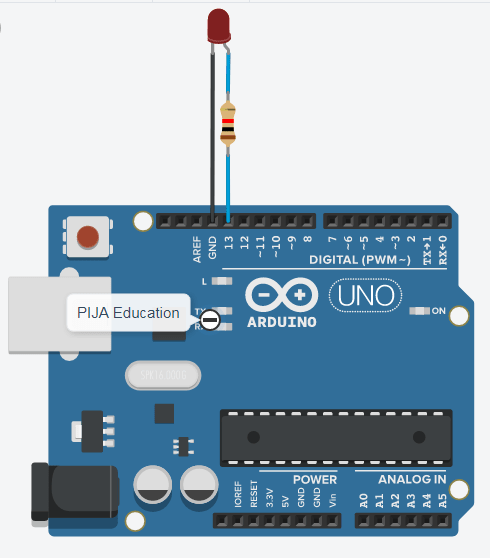
**Apparatus Used:**

**Software :**Tinkercad(Web App)

|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| Arduino UNO Board | 1 |
| Led | 1 |
| Resistor | 1 |
| Connecting Wires | Few |

**Theory:**In this section, we will learn LED blinking using Arduino in Tinkercad.

**Procedure:**



**CIRCUIT DIAGRAM**

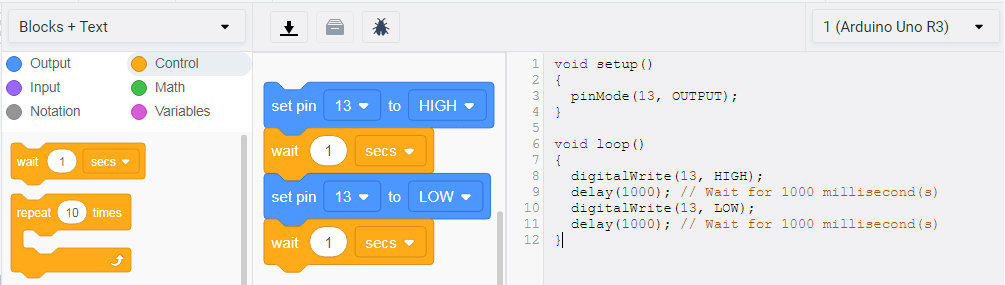
### **CONNECTION TABLE FOR LED BLINKING USING ARDUINO IN TINKERCAD**

|  |  |  |
| --- | --- | --- |
| Sno | Arduino | LED |
| 1 | 13 | Anode (+) |
| 2 | GND | Cathode(-) |

### **CIRCUIT EXPLANATION**

Here, The **anode (+) and cathode (-)** terminals of LED are connected to **pin 13 and ground (GND)**of Arduino Uno respectively and a resistor of 280 ohm is placed between the LED anode terminal and pin number 13 which help us to limit the current and prevent LED from burning.

If we do not connect the resistor here we will get a message in software during simulation that **“current through LED is 52.3 mA, while the recommended maximum is 20.0 mA”**. The usable lifetime of the LED may reduce. That’s why there is a need for a resistor to reduce the current.

ARRANGEMENTS OF BLOCKS AND TEXT CODE PROGRAMMING FOR LED BLINKING

CODE BLOCKS AND TEXT CODE EXPLANATION

### **Block Explanation**

**Step 1:** Firstly select the **output block**. Drag **set pin**command from it and drop on scripting area (work area).

https://i0.wp.com/pijaeducation.com/wp-content/uploads/2020/12/setpin-min.png?ssl=1

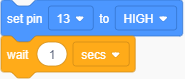
After that, click next to **set pin**and select **13**from the dropdown menu and then select HIGH and from the next dropdown menu which is a state, select HIGH means “1”, for digital pin 13. HIGH is required to glow an LED while LOW is required to turn it OFF.

https://i0.wp.com/pijaeducation.com/wp-content/uploads/2020/12/setpin-13-high-min.png?ssl=1

**Step 2:** Then, we will click on the control block and drag wait command block from it and drop on the scripting area(work area) below set pin command

https://i0.wp.com/pijaeducation.com/wp-content/uploads/2020/12/wait-block-tinkercad-min.png?ssl=1

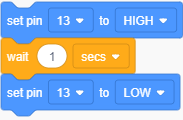
This means the LED will be ON for 1 second (1000 milliseconds).



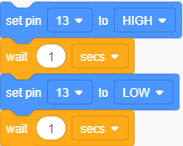
**Step 3:** Again, click on the output block. Drag the set pin command from it and drop on the work area below the wait block command

https://i0.wp.com/pijaeducation.com/wp-content/uploads/2020/12/setpin-min.png?ssl=1

Similar to step 1, select pin 13 and set the state of the pin to LOW (0).



**Step 4**: Repeat step 2, for wait or delay but you can change delay timings suck as 500 ms, 1 sec or 2 sec etc.



**Step 5**: All steps above execute infinitely and you will see continuously LED blinking using Arduino in Tinkercad.

**Text Code Explanation**

**Step 1**:

voidsetup(){

 pinMode(13,OUTPUT);

}

In this function, we have to pass two arguments –

1. First one is pin number and
2. Second is its mode, **pinmode(pinNumber, mode)**.

In this we set pin number 13 as output: **pinmode(13, OUTPUT)**.

**Step2:**

In loop function we are using two inbuilt functions that are **digitalWrite()** and **delay()**.

voidloop(){

 digitalWrite(13,HIGH);

 delay(1000);

 digitalWrite(13,LOW);

 delay(1000);

}

1. We set pin number 13 state to high
2. Give delay of 1 second
3. Then we again change pin number 13 state to low
4. Then delay by 1 second
5. And the loop will start repeating instructions again and again.

And the loop gets executed continuously and follows the same pattern.

**Experiment No. 3**

**Aim:**Simulation of led glow in series pattern with arduino board using tinkercad.

**Apparatus Used:**

**Software:**Tinkercad(Web App)

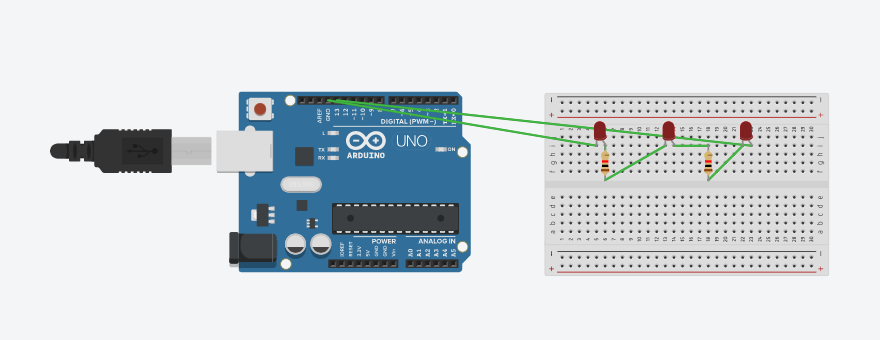
|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| Arduino UNO Board | 1 |
| Led | 3 |
| Resistor | 2 |
| Bread Board | 1 |
| Wire | 4 |

**Theory:** On starting the simulation the 3 led connected in series patter starts to glow.

**Procedure:**

1. Establish ground connection using wire on the negative (smaller pin) of one of the three LED and GND pin of Arduino .
2. Connect resistor to the positive (larger pin) of LED and attach other end of the resistor to the negative side of another led.
3. Connect another led in similar manner.
4. Establish connection using PIN 13 of Arduino and positive end of the third led in series.

**Circuit Diagram:**

****

**Code**

**void setup()**

**{**

**pinMode(LED\_BUILTIN, OUTPUT);**

**}**

**void loop()**

**{**

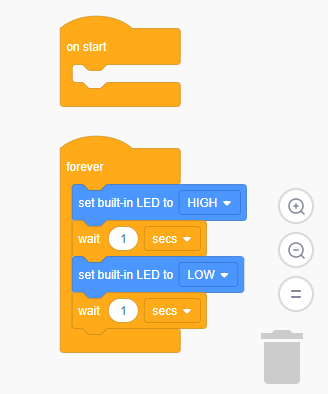
**digitalWrite(LED\_BUILTIN, HIGH);**

**digitalWrite(LED\_BUILTIN, HIGH);**

**delay(10); // Delay a little bit to improve simulation performance**

**}**

**Block Diagram:**

****

**Experiment No. 4**

**Aim:**Simulation of Led Glow in parallel pattern with Arduino Board using tinkercad.

**Apparatus Used:**

**Software:**Tinkercad(Web App)

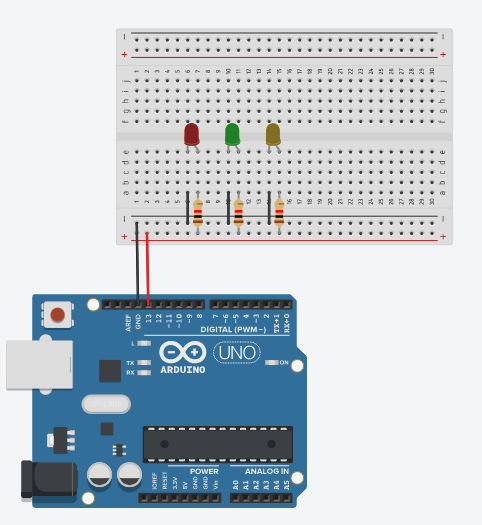
|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| Arduino UNO Board | 1 |
| Led | 3 |
| Resistor | 3 |
| Bread Board | 1 |
| Wire | 5 |

**Theory:** On starting the simulation the 3 led connected in parallel patter starts to glow.

**Procedure:**

1. Establish ground connection using wire on the negative (smaller pin) of one of the three LED and GND pin of Arduino .
2. Connect resistor to the positive (larger pin) of LED.
3. Connect all 3LEDs in the similar manner.
4. Position your LEDs so the legs go to two different rows of the breadboard.
5. Now connect ground to the negative of the first LED and PIN 13 to the last LED with resistor.
6. Establishing connection using PIN 13 of Arduino and positive end of the third LED with resistor will create aparallel connection of LEDs.

**Circuit Diagram:**

****

**Code**

void setup()

{

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop()

{

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000); // Wait for 1000 millisecond(s)

digitalWrite(LED\_BUILTIN, LOW);

delay(1000); // Wait for 1000 millisecond(s)

}

**Block Diagram:**

**Graphical user interface

Description automatically generated with medium confidence**

**Experiment Number 5:-**

**Aim:-** To show simulation of Led Glow using push button with Arduino Board using Tinkercad

**Apparatus Used:-** In Tinkercad (web app) – Arduino Uno, Breadboard, LED, Resistor, Push Button and Wires.

**Theory: -** Pushbuttons connect two points in a circuit when you press them.

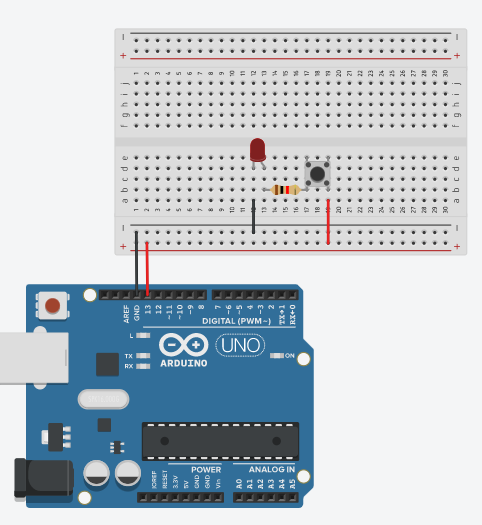
**Procedure:-**

-Establish ground connection using wire on the negative (smaller pin) of LED and GND pin of Arduino .

-Connect resistor to the positive (larger pin) of LED and attach one end of push button to the resistor’s other end.

-Establish connection using PIN 13 of Arduino and other end of Push button.

Circuit Diagram :

****

Code :

void setup()

{

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop()

{

digitalWrite(LED\_BUILTIN, HIGH);

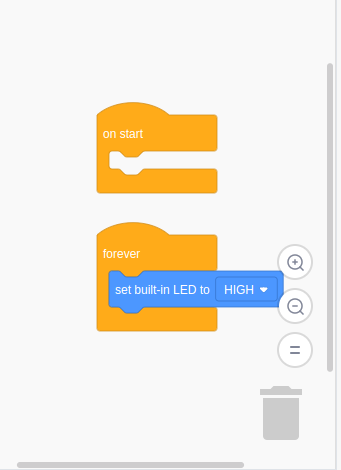
delay(10); // Delay a little bit to improve simulation performance

}

Block Diagram:

Graphical user interface, application

Description automatically generated



**Experiment No. 6**

**Aim :**Simulation of changing the intensity of led using potentiometer with Arduino Board using TinkerCad

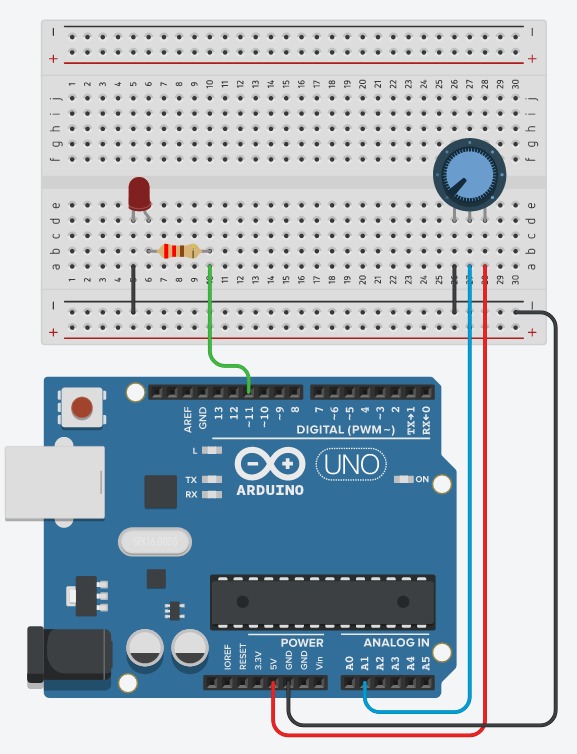
**Apparatus Used:**

**Software :**TinkerCad (Web App)

|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| Arduino UNO Board | 1 |
| Breadboard | 1 |
| 220 ohm Resistor | 1 |
| Potentiometer | 1 |
| LED – any color | 1 |
| Wire (male to female) | Few bunch |

**Theory:** In this section, we will see how to control brightness of LED using potentiometer with Arduino Board on TinkerCad.

**Procedure:**



### **CIRCUIT EXPLANATION**

* As a best practice we’ll start with the ground (GND). It’s very important to make a common ground for all components. To do this we will first plug a black wire (black is convention for GND) between a GND pin of the Arduino and the “minus” line on the breadboard. From this “minus” line, then we will be able to connect all other grounds, which will make things easier to manage.

## Add the LED:

## Connect the shorter leg of the LED to the ground. You can directly plug this leg into the “minus” line of the breadboard, or add a small black wire.

## Plug the other (longer) leg of the LED to an independent line on the breadboard.

## From this leg, add a 220 Ohm resistor to yet another line.

## Add a wire between the other side of the resistor and a PWM-compatible digital pin (so we can control the brightness). Here on Arduino Uno you can choose between pins 3, 5, 6, 9, 10, and 11 – you can recognize PWM compatibility with the “~” next to the pin number.

## Add the Potentiometer:

## Plug the 3 legs of the potentiometer to 3 different lines on the breadboard.

## Connect the extreme left (or right) leg to GND.

## Connect the other extreme leg to 5V on the Arduino.

## Add a wire between the middle pin and an analog pin.

Arduino code to control LED brightness with the potentiometer

## when we turn the knob up (for example turning clockwise), we want the LED brightness to increase.

## The minimum knob position will correspond to the minimum brightness – LED turned off. And the maximum position will correspond to the maximum brightness – LED with full intensity, same as if you used digitalWrite() with HIGH.

## Code:

#define LED\_PIN 11

#define POTENTIOMETER\_PIN A1

void setup()

{

pinMode(LED\_PIN, OUTPUT);

}

void loop()

{

int potentiometerValue = analogRead(POTENTIOMETER\_PIN);

int brightness = potentiometerValue / 4;

analogWrite(LED\_PIN, brightness);

}

Code explained step by step

#define LED\_PIN 11

#define POTENTIOMETER\_PIN A1

We start by a best practice, which is to create some defines (you could also create some const int variables) for the pins we are going to use in the program

void setup()

{

pinMode(LED\_PIN, OUTPUT);

}

In the void setup(), we need to initialize the mode for the pins we want to use:

* The LED is a component we control, so we use pinMode() with OUTPUT.
* The potentiometer is a component we read from, so its mode is INPUT. However, for analog pins, no need to use pinMode() as those pins are already in INPUT mode!

**Read potentiometer value**

void loop()

{

int potentiometerValue = analogRead(POTENTIOMETER\_PIN);

This number corresponds to the voltage we read on the analog pin. The higher the voltage, the higher this number. For 0V you get 0, and for 5V you get 1023.

**Compute LED brightness**

**int** brightness = potentiometerValue / 4;

With this line and the 5 arguments in the function, what we’re saying is: take the potentiometerValue from the range 0-1023, to the range 0-255.

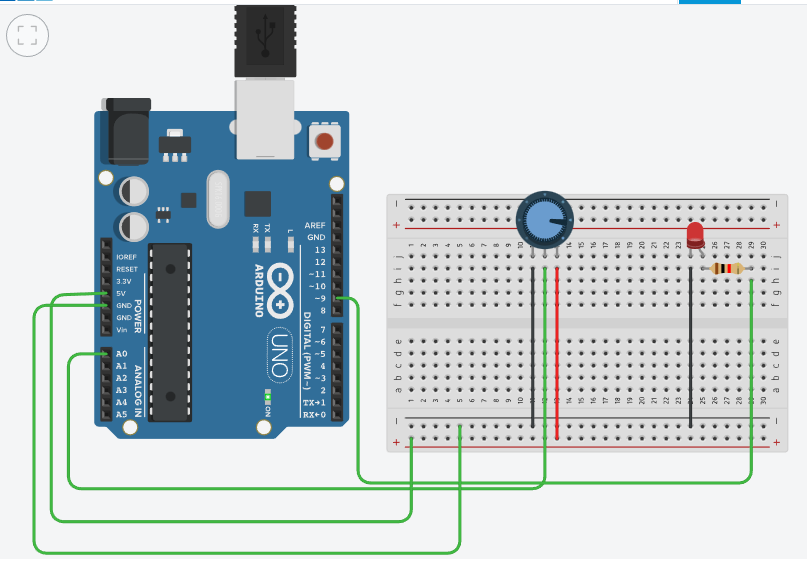
**Apply brightness setting to LED**

analogWrite(LED\_PIN, brightness);

}

So now we have computed the brightness to apply to the LED. This brightness variable is an integer number between 0 and 255. Exactly what we need for the analogWrite() function.

analogWrite() takes 2 arguments: the pin to apply the voltage to, and then a byte number between 0-255.



**Experiment No. 7**

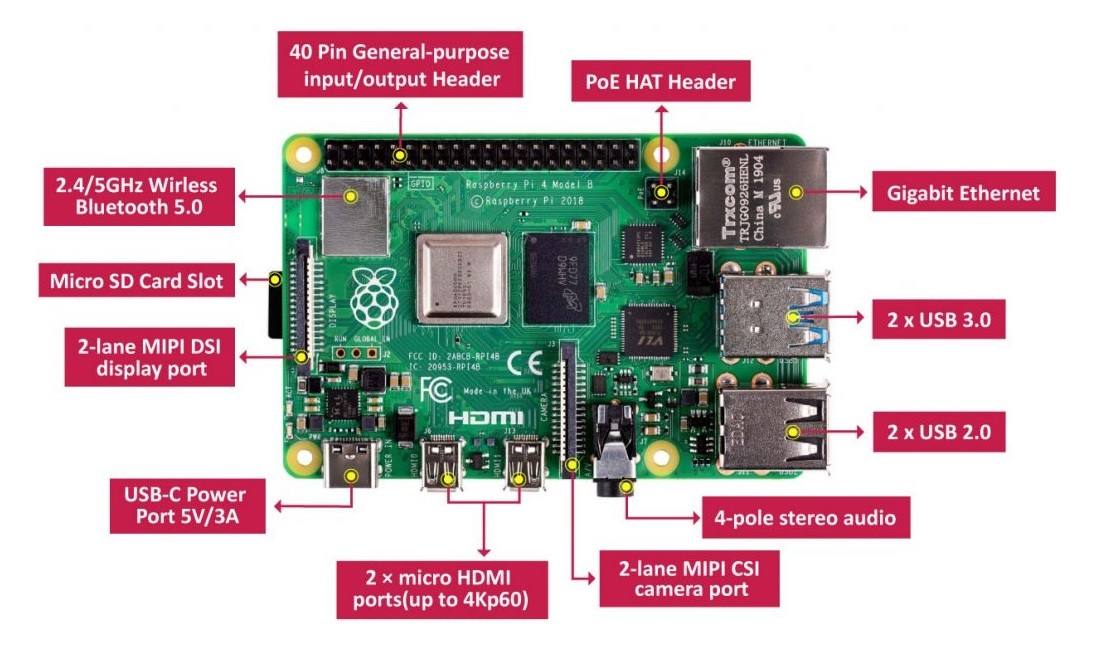
**Aim:**Study of Raspberry architecture and pin diagram

**Apparatus Used:**

**Software:**Raspberry

**Theory:**Study of Raspberry Pi architecture and pin diagram.

**Diagram:**



What is a Raspberry Pi?

Raspberry Pi is a credit-card sized single-board computer designed and manufactured by the Raspberry Pi foundation in the United Kingdom. Raspberry Pi has an ARMv6 700 MHz single-core processor, a VideoCore IV GPU and 512MB of RAM. it uses an SD card for its operating system and data storage. The Raspberry Pi officially supports Raspbian, a lightweight linux OS based on Debian. Back in 2006, while Eben Upton, his colleagues at University of Cambridge, in conjunction with Pete Lomas and David Braben, formed the Raspberry Pi Foundation. Early prototypes of the Raspberry Pi were based on the 8-bit Atmel ATMega644 in order to reduce cost. Following prototypes utilized an ARM processor similar to what was used in the release version of the Raspberry Pi. In 2012, the team started its first production run consisted of 10,000 Raspberry Pi unites manufactured by foundries in China and Taiwan. Unfortunately, there was a manufacturing issue where the ethernet jack on the Raspberry Pi. This incident caused some minor shipping delays It took the team six years of hardware development to create the Raspberry Pi makers and electronics enthusiasts adore today

**Model B**



In April 2012, the original Raspberry Pi, also known as the model B, was launched. It offered an ARMv6 700 MHz single-core processor, a VideoCore IV GPU and 512MB of RAM, all packed in a credit card size board. The model B supported 26-pin GPIO and a SD card slot for loading the operating system. Priced at $35 per unit.

**Model A**

In February 2013, the Raspberry Pi model A was launched, It packed the same ARM single-core processor and the GPU as model B with half of the RAM capacity at 256MB. Priced at $25 per unit, the model A could be regarded as the cheaper and less powerful version of model B.

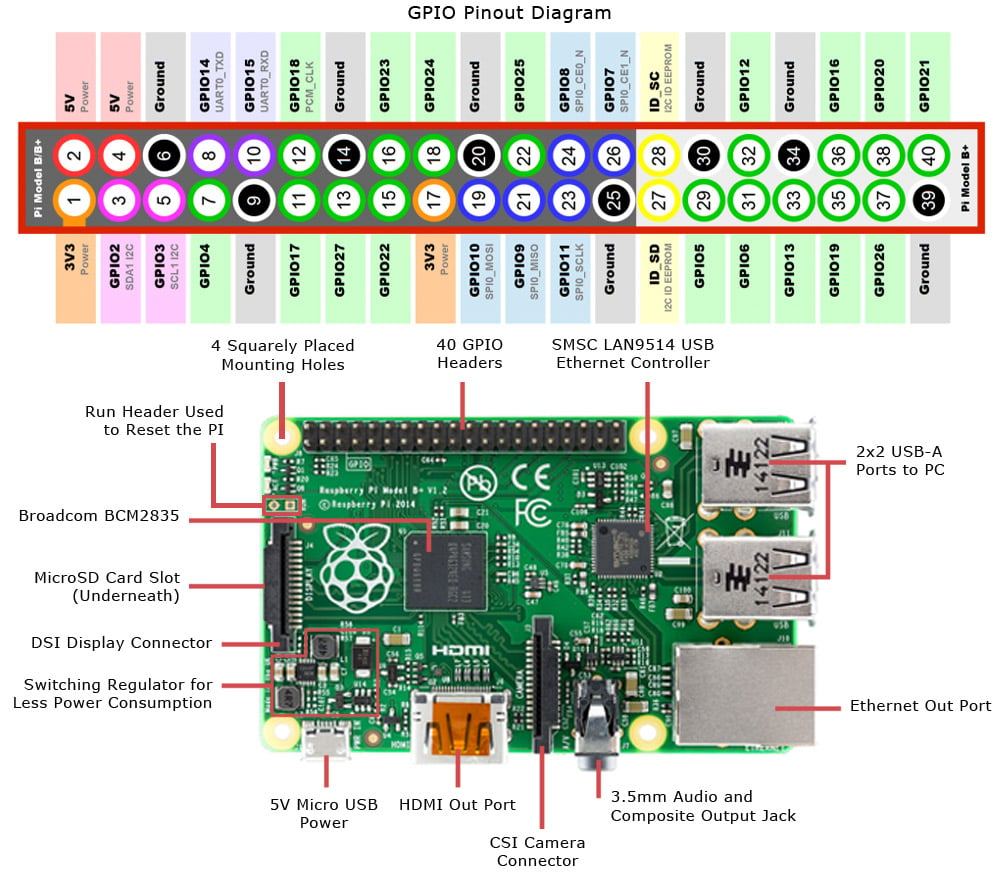
**Model B+**

In July 2014, the Raspberry Pi model B+ was launched, It was a worthy successor to the original Raspberry Pi. Packing the same processing unit as its predecessor. The model B+ received two more USB ports, an updated 40-pin GPIO layout, and a micro-SD card slot for loading the operating system. Priced at $30 per unit, the model B+ was cheaper and offered more connectivity than model B, Making it an Ideal choice for education.

**Model A+**



In November 2014, the Raspberry Pi model A+ was launched, It offered improved specifications over the model A. Packing the same processor as its predecessor, the model A+ upgraded RAM capacity from 256MB to 512MB. Smaller footprint, 40-pin GPIO, and micro-SD card. Priced at $20 per unit, the model A+ was the smallest and cheapest single-board computer at that time.

**Pin Diagram: -**

**Experiment No. 8**

**Aim :**Setup of raspberry pi board

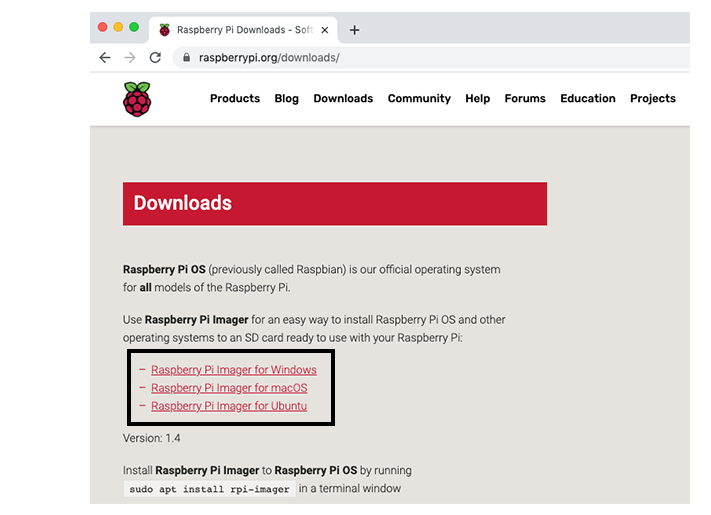
**What we need?**

* A power supply with at least 2.5 amps power
* A micro-SD card
* A keyboard and a mouse.
* A monitor
* HDMI

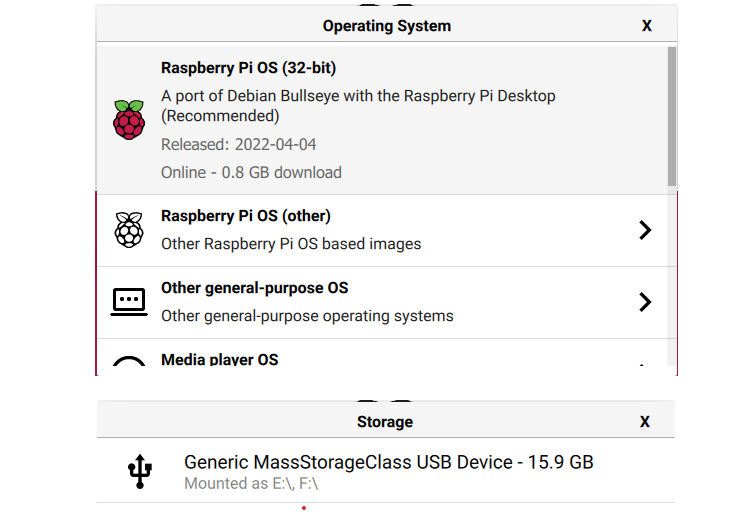
**Procedure:**

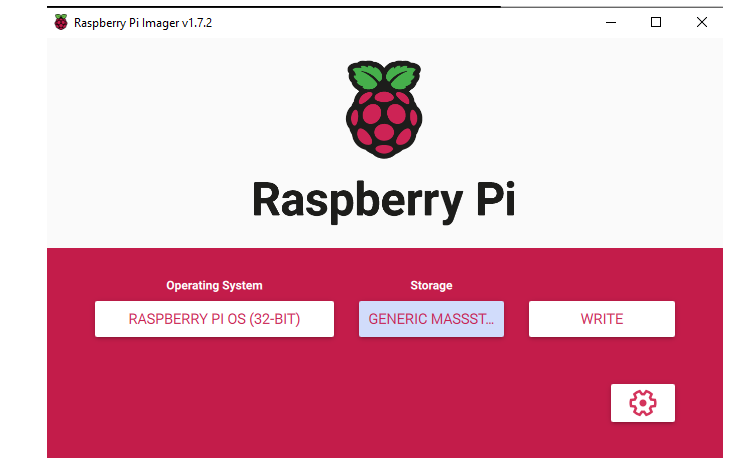
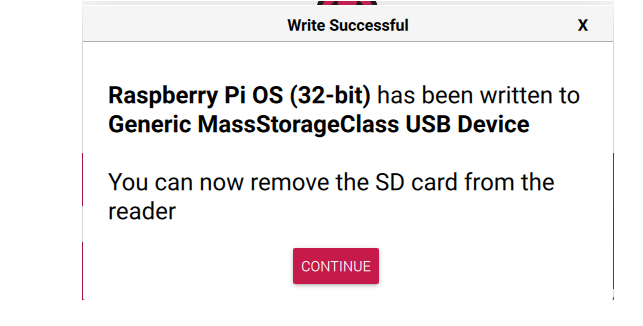
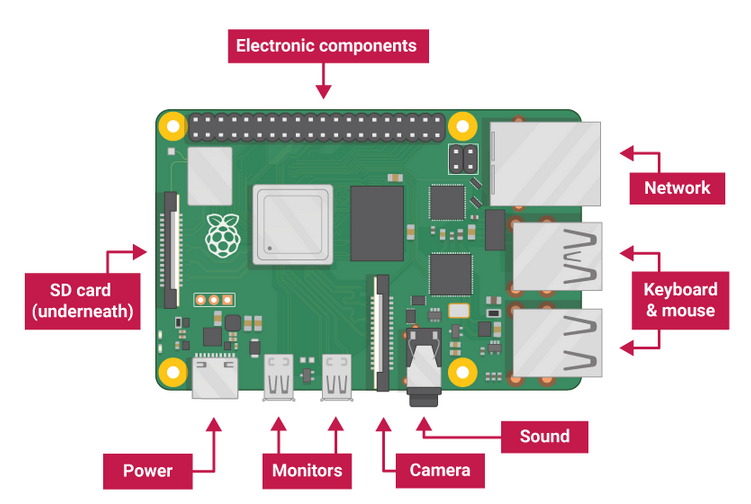
**Step1: Install the raspberry pi 3 to the SD card using the raspberry pi imager.**

* Download raspberry pi imager from the official website.



* Insert your SD card in the slot in the computer**.**
* Launch the imager. In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on.



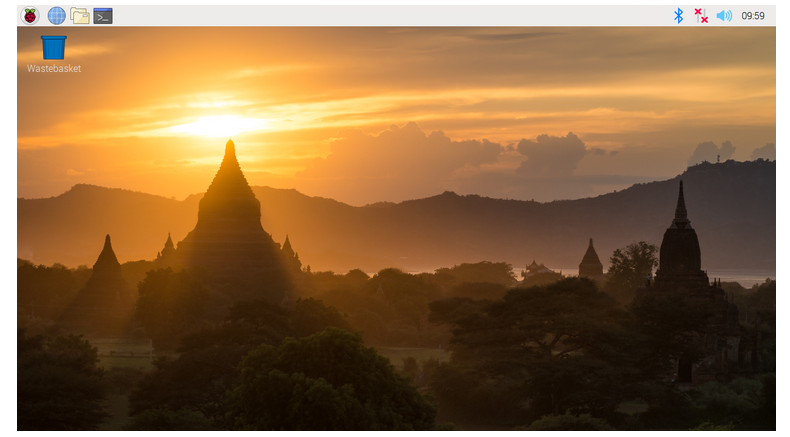
* Then click ‘write’
* Wait until the installation completes. When you get the following screen, eject your sd card.

**Step2: Connect all peripherals to the board.**

* Place the SD card in the right slot in the raspberry pi 3.
* Connect the mouse and keyboard.
* Connect the monitor.

**Step3: start up the raspberry pi.**

* Plug the power supply into a socket and connect it to your Raspberry Pi’s power port.
* Now, you see the raspberry pi welcome page on the screen.
* Here, you can set country, language, time zone, etc. You can create users. And the installation is finished.



**Experiment No. 9**

**Aim :**Led Connection using RaspberryPi

**Apparatus Used:**

**Software :**Tinkercad(Web App)

|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| Raspberry Pi | 1 |
| Led | 1 |
| Resistorbetween(330 ohm -1k ohm) | 1 |
| Bread Board | 1 |
| Male-Female jumper Wires | 2 |

**Theory:** To stablish the connection to glow LED. We need all the things that are mention on above.

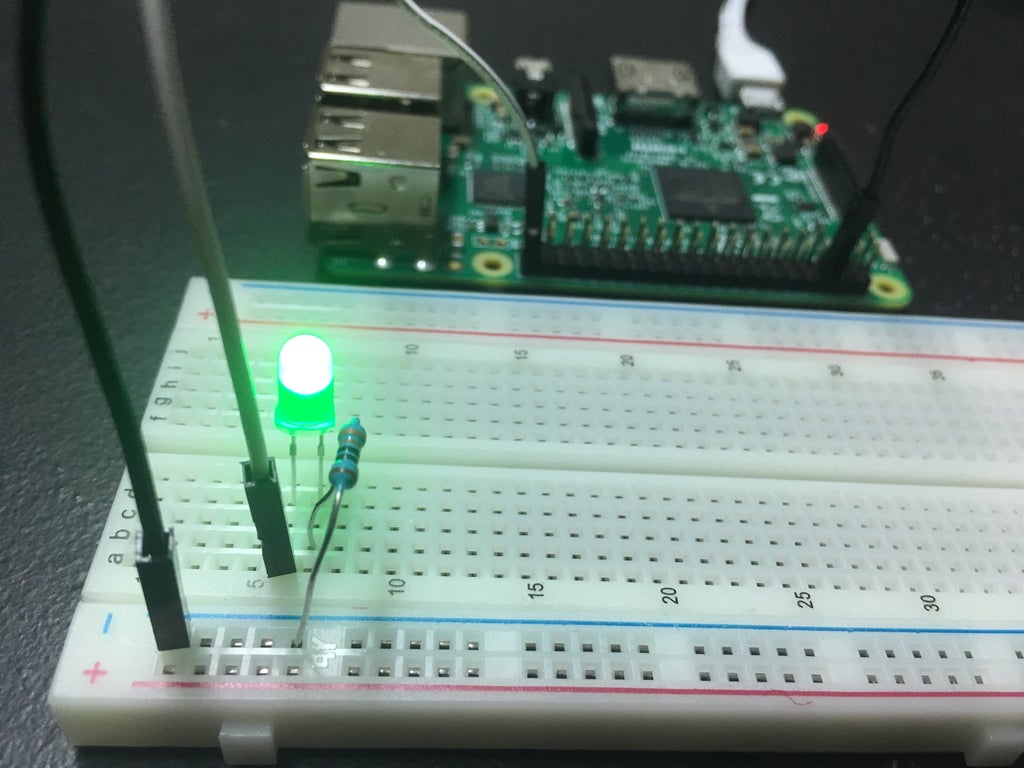
Text

Description automatically generated

**Procedure:**

1. **Connect one wire between one GND (ground) pin of the Raspberry Pi and the blue line of the breadboard.**
2. **Take the LED and check the 2 legs. You will see that one is shorter than the other. Plug the shorter leg to the blue line (now connected to GND), and the longer to any other connector. You can either directly connect the shorter leg to the blue line, or add an additional short male-to-male connector the result is the same.**
3. **Plug one leg of the resistor to the same line as the longer leg of the LED, and the other leg of the resistor to a different line.**
4. **Finally, to close the circuit plug one wire between the same line as the other leg of the resistor, and the GPIO number 17. This is the 6th pin on the GPIO header, starting from the left, on the inside side.**

**Circuit Diagram:**



**Experiment No.10**

### Aim-Interfacing of DHT sensor using Raspberry Pi

**Apparatus Used**-

* Raspberry pi Model 3 B+,
* Jumper Wires
* DHT sensor
* Breadboard
* Adapter
* Connecting Wires
* VGA Cable

**Theory:-**

The DHT-11 Digital Temperature And Humidity Sensor is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).

**Procedure**:-

1. Place DHT11 sensor on the breadboard.
2. Connect wire between Pin 1 of the DHT11 and Physical Pin 1 (3v3) of the Raspberry Pi.
3. Connect wire between Pin 2 of the DHT11 and Physical Pin 11 (GPIO4) of the Raspberry Pi
4. Connect wire between Pin 3 of the DHT11 and Physical Pin 6 (GND) of the Raspberry Pi

Code:-

import sys

import Adafruit\_DHT

while True:

humidity, temperature = Adafruit\_DHT. read\_retry(11, 4)

print('Temp: {0:0.1f} с Humidity: {1:0.1f} %'.format (temperature, humidity))

Text

Description automatically generated

**Experiment No.11**

**Aim:-**Interfacing of IR sensor using Raspberry Pi.

**Apparatus Used**-

* Raspberry pi Model 3 B+,
* Jumper Wires
* IR sensor
* Breadboard
* Adapter
* Connecting Wires
* VGA Cable

**Theory:-**

An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm … 50 µm. IR sensors are now widely used in motion detectors, which are used in building services to switch on lamps or in alarm systems to detect unwelcome guests.

**Procedure**:-

1. Place IR11 sensor on the breadboard.
2. Connect wire between Pin 1 of the IR11 and Physical Pin 1 (3v3) of the Raspberry Pi.
3. Connect wire between Pin 2 of the IR11 and Physical Pin 11 (GPIO4) of the Raspberry Pi
4. Connect wire between Pin 3 of the IR11 and Physical Pin 6 (GND) of the Raspberry Pi

Code: -

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(17,GPIO.IN)

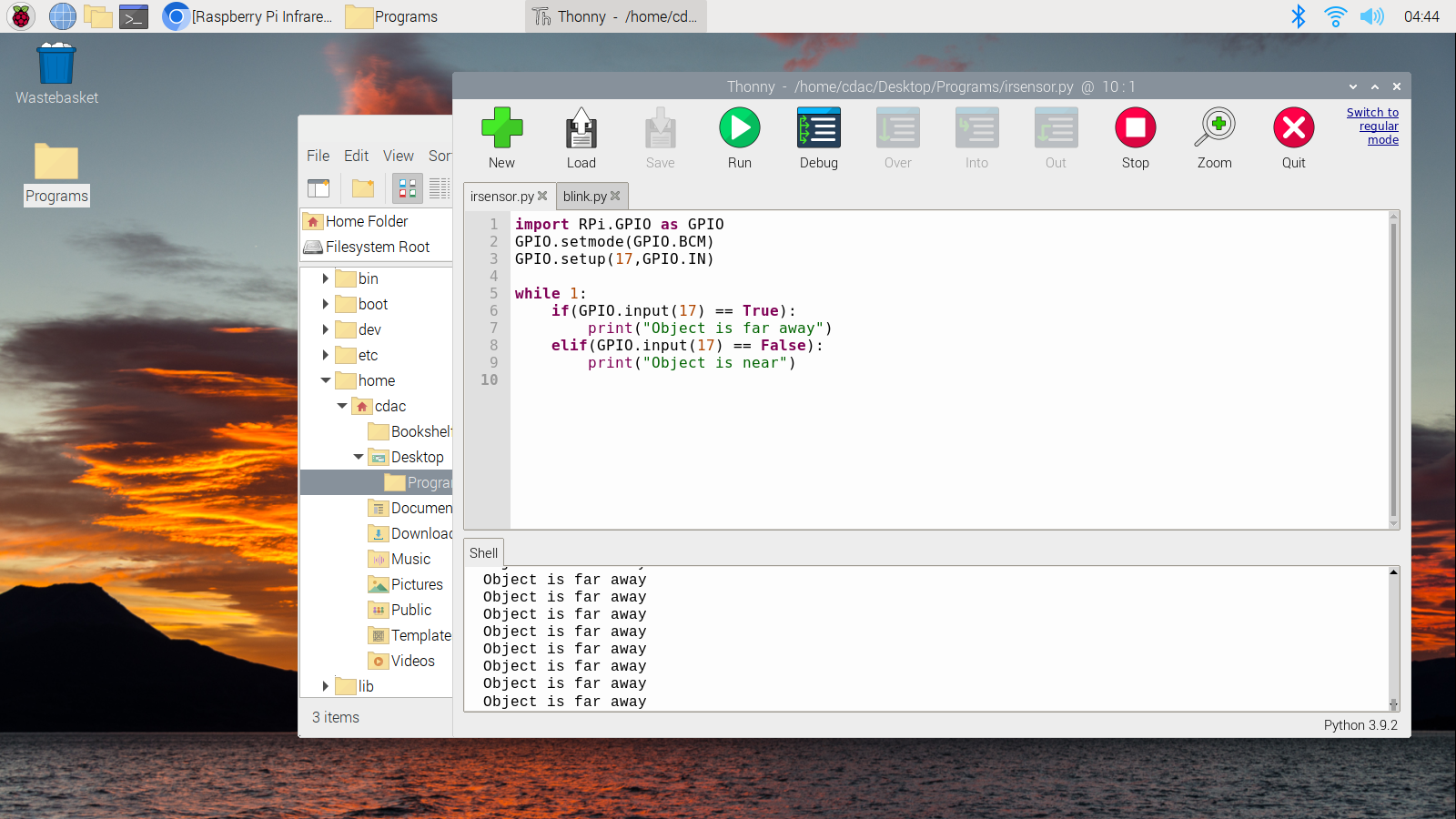
while 1:

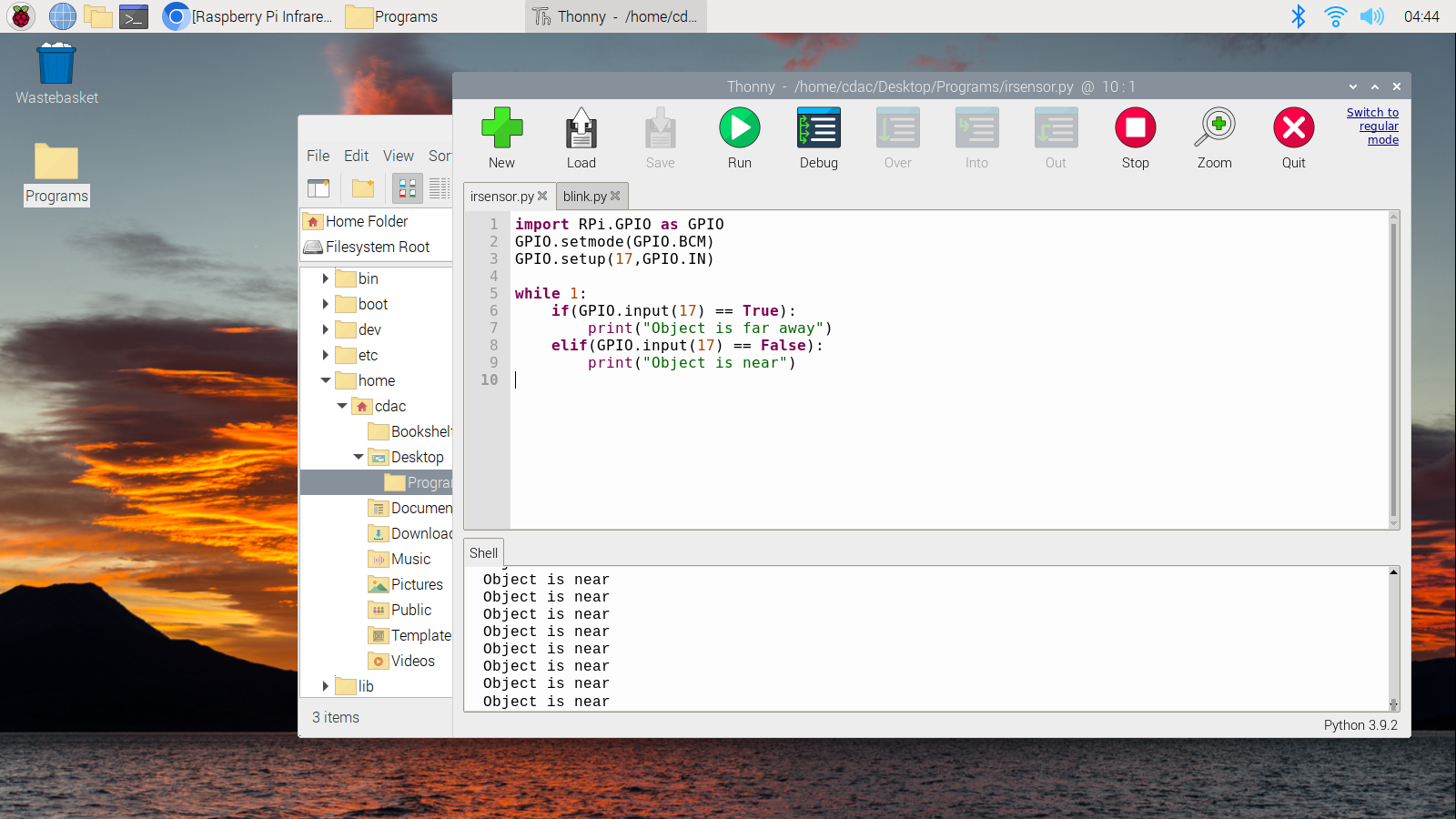
if(GPIO.input(17) == True):

print("Object is far away")

elif(GPIO.input(17) == False):

print("Object is near")





**Experiment No. 12**

**Aim :** Interfacing of flame sensor using raspberry pi

**Apparatus Used:**

**Software :**Thonny ide

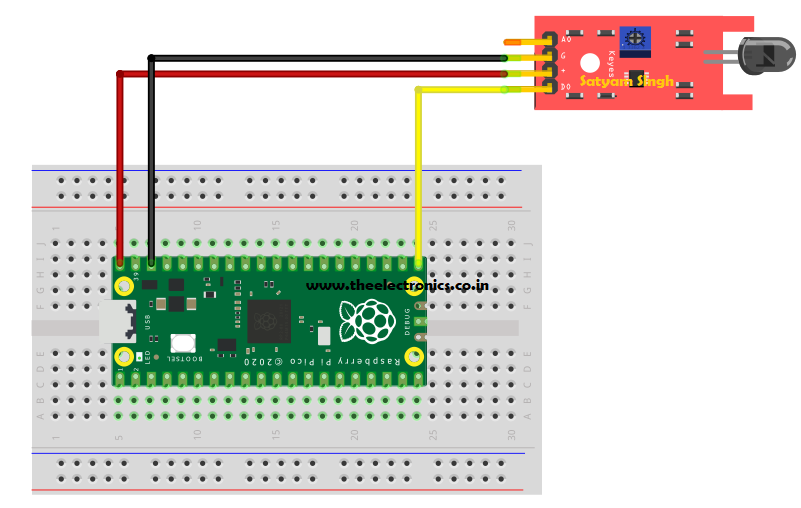
|  |  |
| --- | --- |
| **Apparatus** | **Quantity** |
| RPI 3 | 1 |
| Amp Power Adapter | 1 |
| 16 GB micro SD | 1 |
| 120 pcs jumper cable | 1 |
| IR Flame Sensor | 1 |

**Theory:** when sensor come in contact with flame the bulb automatically glows.

**SETUP:**

1. VCC -> 5V
2. GND -> GND
3. DO -> GPIO 21

**Circuit Diagram:**



**Code**

#!/usr/bin/python  
import RPi.GPIO as GPIO  
import time  
  
#GPIO SETUP  
channel = 21  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(channel, GPIO.IN)  
  
def callback(channel):  
 print("flame detected")  
  
GPIO.add\_event\_detect(channel, GPIO.BOTH, bouncetime=300) # let us know when the pin goes HIGH or LOW  
GPIO.add\_event\_callback(channel, callback) # assign function to GPIO PIN, Run function on change  
  
# infinite loop  
while True:  
time.sleep(1)

**Experiment No. 13**

**Aim:-**Visualization od DHT Sensor data on cloud (ThingSpeak)

**Apparatus Used**-

**Hardware requirements:**

* Any board based on an ESP8266 chip (for instance, NodeMCU)
* A DHT11 or DHT22 sensor

**Software and services**:

* DHT sensor library by Adafruit — v1.3.7
* Adafruit Unified Sensor — v1.0.3
* cloud4rpi-esp-arduino — v0.1.0
* Cloud4RPI — Cloud control panel for IoT devices PlatformIO IDE for VSCode

**Theory:-**

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use but requires careful timing to grab data

**Procedure**:-

1. Place IR11 sensor on the breadboard.
2. Connect wire between Pin 1 of the DHT11 and Physical Pin 1 (3v3) of the Raspberry Pi.
3. Connect wire between Pin 2 of the DHT11 and Physical Pin 11 (GPIO4) of the Raspberry Pi
4. Connect wire between Pin 3 of the DHT11 and Physical Pin 6 (GND) of the Raspberry Pi

Code:-

import thingspeak

import time

import Adafruit\_DHT

channel\_id = 1975689

write\_key = '21UJ010DZQM1FMWC'

pin = 4

sensor = Adafruit\_DHT.DHT11

def measure(channel):

try:

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

if humidity is not None and temperature is not None:

print('Temperature = {0:0.1f}\*C Humidity = {1:0.1f}%'.format(temperature, humidity) )

else:

print('Did not receive any reading from sensor. Please check!')

# update the Value

response = channel.update({'fieldl': temperature, ‘'field2': humidity})

except:

print("connection failure")

if name\_ == "\_ main\_":

channel = = thingspeak. Channel(id=channel\_id, api\_key=write\_key)

while True:

measure(channel)

#free account has a limitation of 15 sec between the updates

time.sleep(15)

