

Course Code	Course Title	Credits	Lectures /Week
USCSP404	IoT Technologies – Practical	1	3
1	Preparing Raspberry Pi: Hardware preparation and Installation		
2	Demonstrate Arduino Uno and its pins interfacing with IDE.		
3	GPIO: Light the LED with Python with/without a button using either Uno/Raspberry Pi.		
4	SPI: Camera Connection and capturing Images/Videos using SPI		
5	GPIO: LED Grid Module: Program the 8X8 Grid with Different Formulas		
6	Stepper Motor Control: PWM to manage stepper motor speed using Uno/Raspberry Pi.		
7	Node RED: Connect LED to Internet of Things		
8	Use different types of sensors (LDR, Temperature) with Raspberry Pi/Uno.		
9	Trigger a set of led GPIO on any IoT platform via any related web server		
10	Interface with any sensor and send its value over the internet to the server using any suitable protocol		

Practical 1:

Preparing Raspberry Pi : Hardware Preparation and Installation

Steps:

1. Attach all the peripheral to the raspberry pi as follows
 - a. In the USB port attach the keyboard and mouse.
 - b. Attach the LAN cable to the internet port.
 - c. Attach the monitor to the HCMII port
 - d. Attach the charger to micro USB power
 - e. If you want to attach any audio device you can attach it to audio port.
2. Download the raspberry pi operating system on the micro sd card.
3. Go to the [raspberrypi.org/downloads](https://www.raspberrypi.org/downloads)
4. Downloads noobs on your machine

5. Extract the file and copy those file on your sd card
6. Attach your sd card to raspberry pi
7. When you switch on the raspberry pi it will display a boot screen.
8. Click on the rasbian option and click it.
9. Install the OS on your sd card and will reboot.

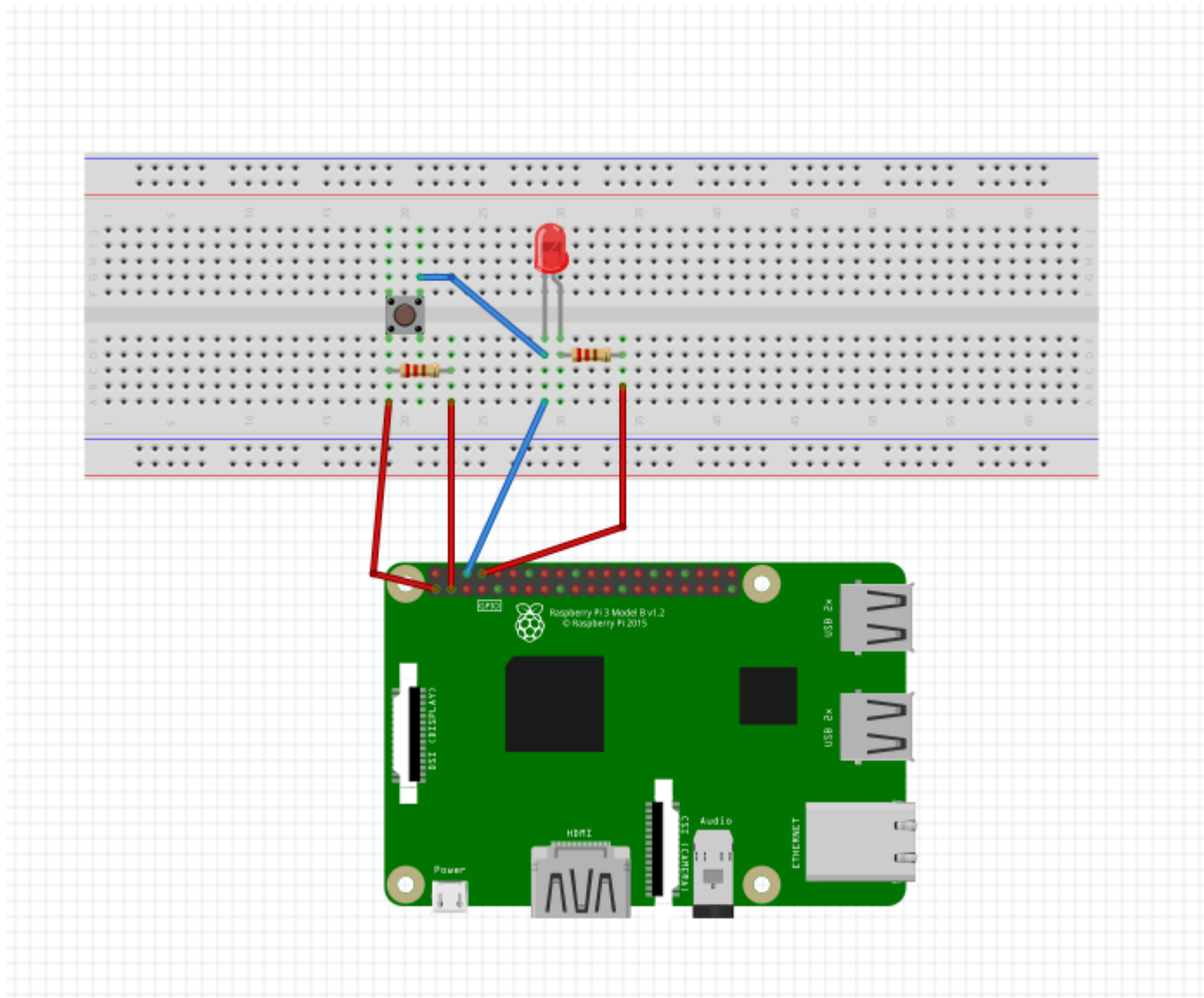
Practical 2:

Perform Linux Commands

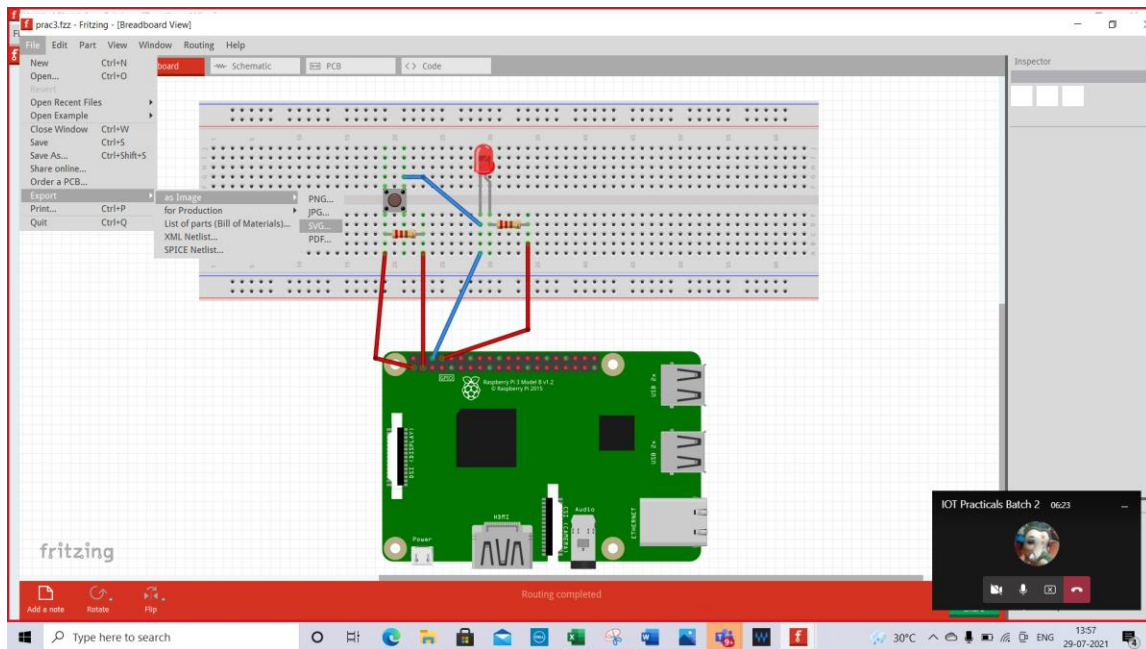
Practical 3 :

1. Download Fritzing using the given link
2. Create circuit as shown below

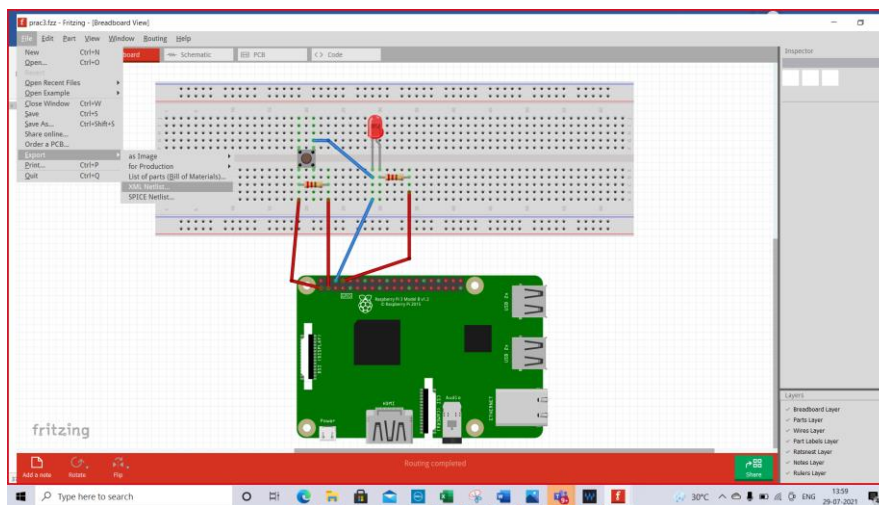
Circuit



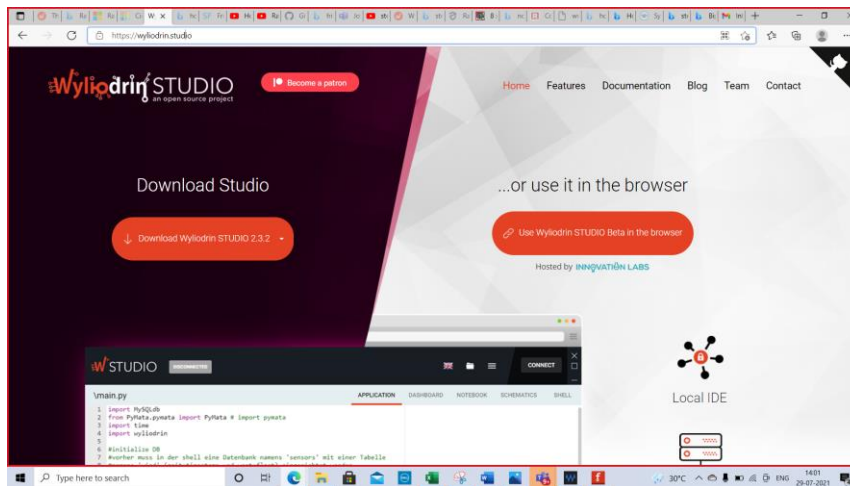
3. Export circuit as SVG file



4. Export again as XML file

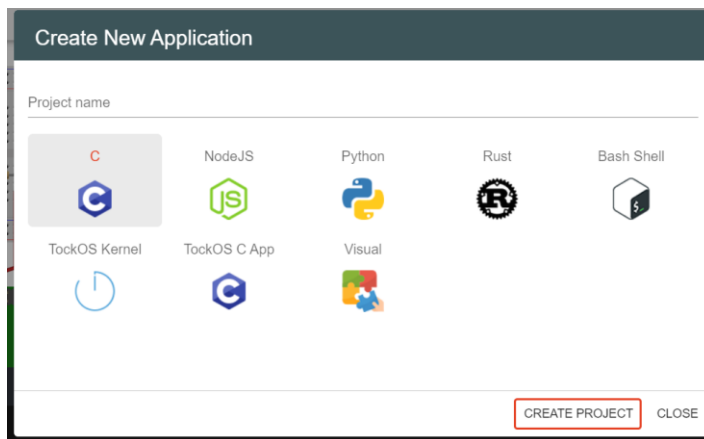


5. Download Wylidrin from the following link [WylidrinSTUDIO](https://wylidrin.com/)

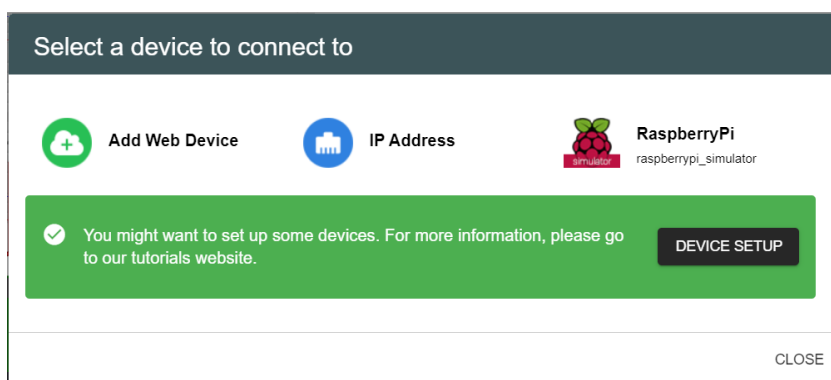


6. Open Wylodrin

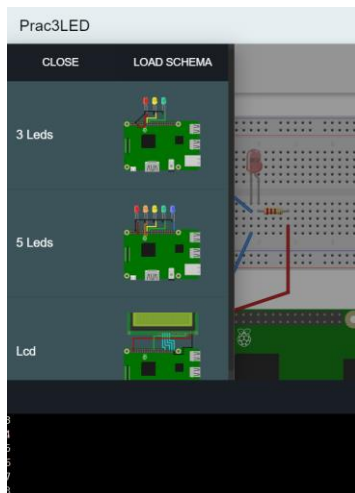
7. Create a new project, give proper name and choose Node.js as language



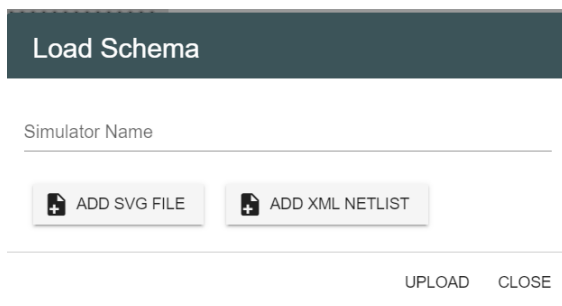
8. Click on connect button and choose Raspberry Pi simulator



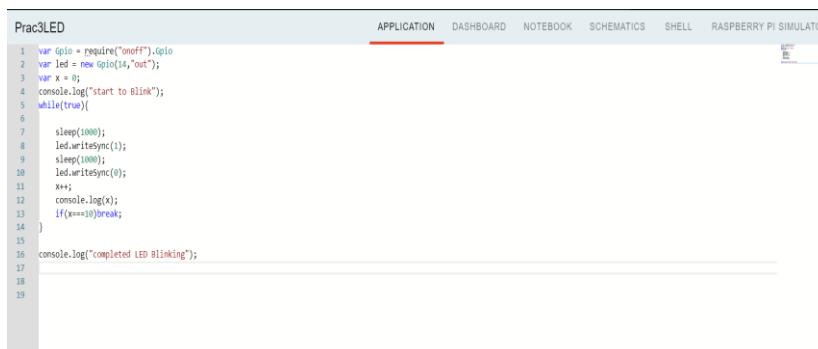
9. Click on load schema button



10. Select the .svg and .xml file created using fritzing



11. In the Application Tab write the following code



Code :

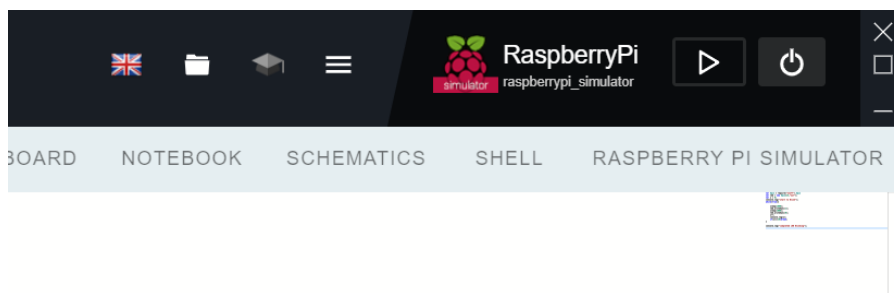
GPIO: Light the LED with Node.js

```
var Gpio = require("onoff").Gpio
var led = new Gpio(14,"out");
var x = 0;
console.log("start to Blink");
while(true){

    sleep(1000);
```

```
led.writeSync(1);  
sleep(1000);  
led.writeSync(0);  
x++;  
console.log(x);  
if(x===5)break;  
}  
  
console.log("completed LED Blinking");
```

12. Click on Run



13. Your LED will start blinking

GPIO: Light the LED with Python

Import RPi.GPIO as GPIO

Import time

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(18,GPIO.OUT)

print "LED on"

GPIO.output(GPIO.HIGH)

Time.sleep(1)

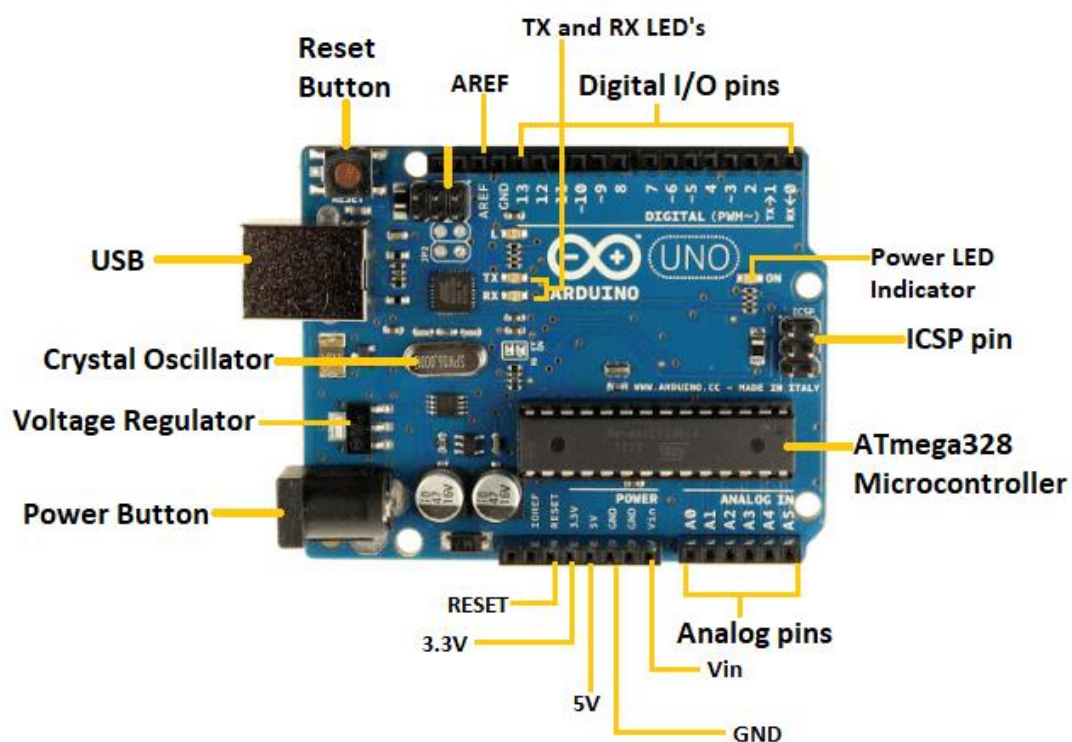
Print "LED off"

GPIO.output(18,GPIO.LOW)

Execute the code using *sudo python prac3.py*

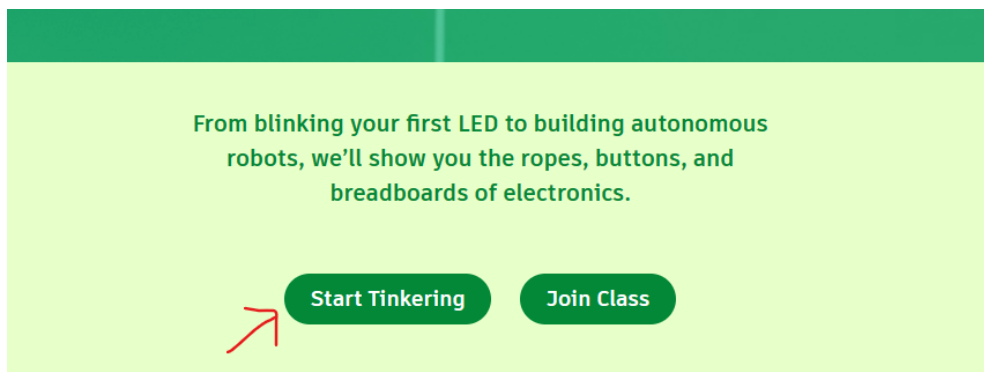
PRACTICAL 3

Aim : Demonstrate Arduino Uno and its pins interfacing with IDE.

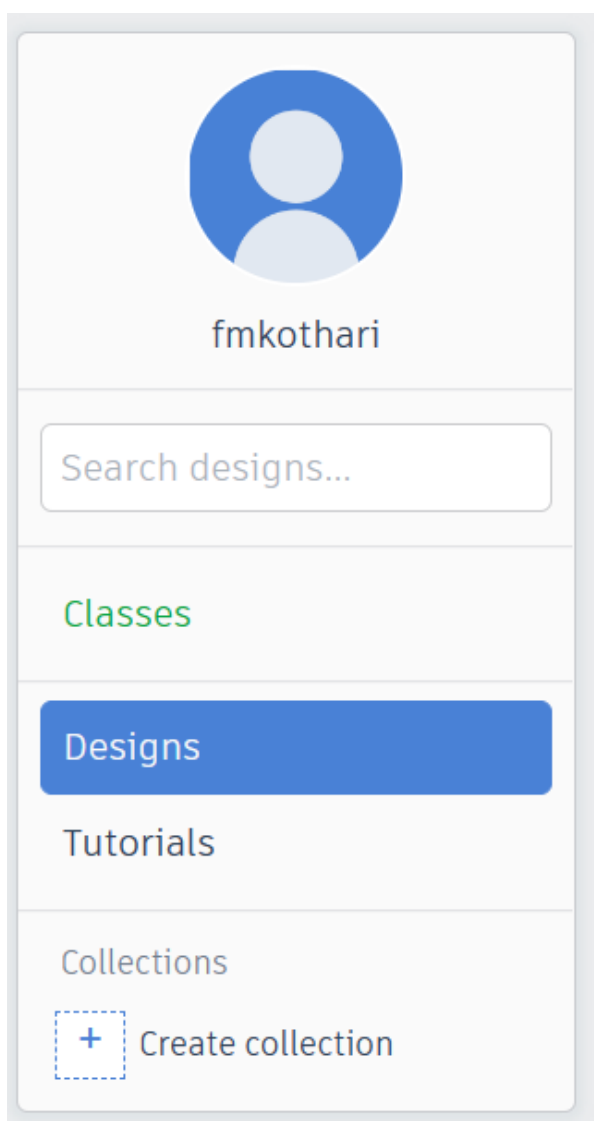


Step1 : Go to <https://www.tinkercad.com/circuits>

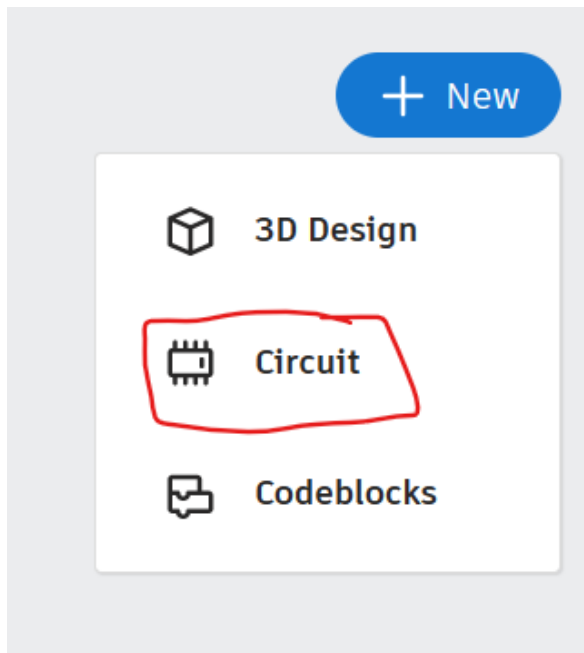
Step 2 : Click on start tinkering



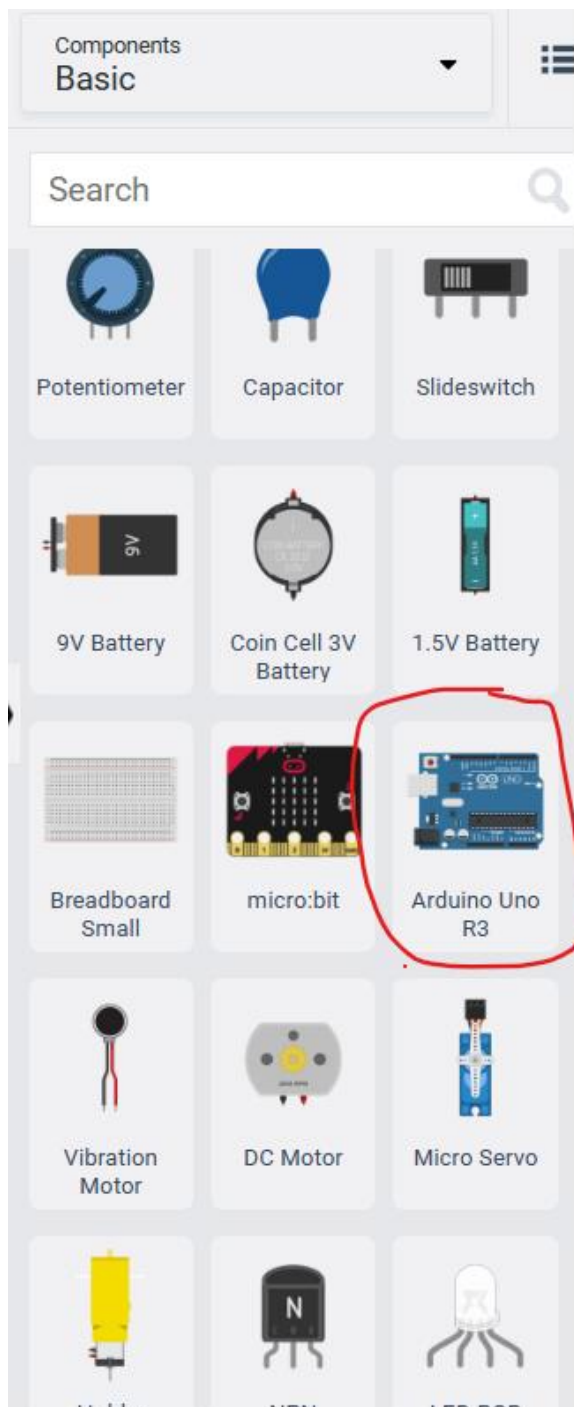
Step 3 : Click on Design in the left pan



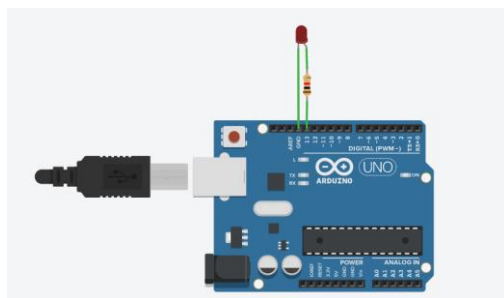
Step4 : Click on New and choose circuit



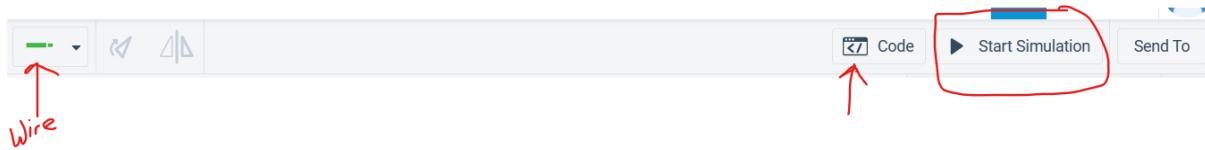
Step 5 : Now choose Arduino from the right pan drag and drop



Step 6 : Now select the led and create the circuit as shown in the diagram

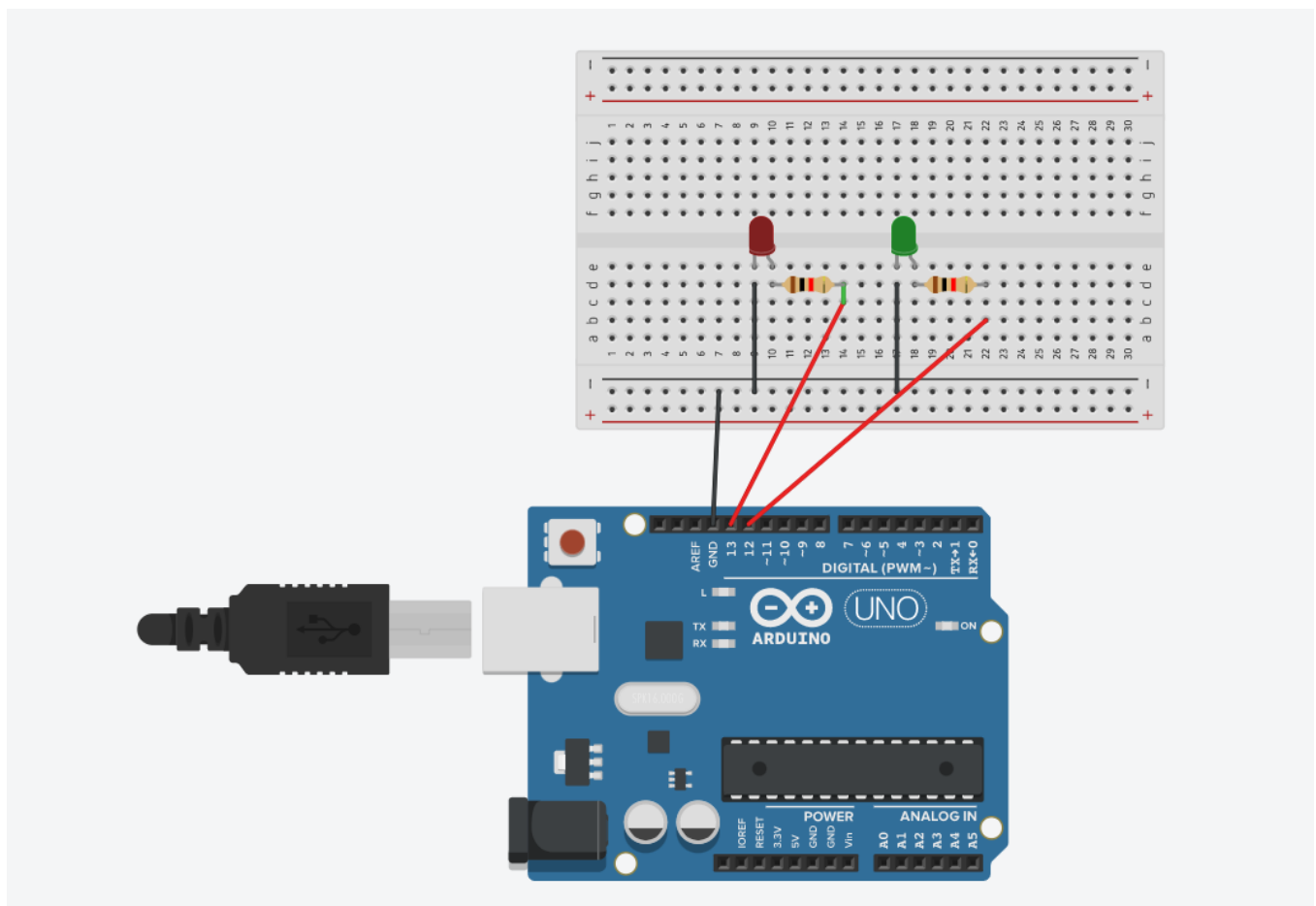


Step 7 : Click on the green line for wires. To start the led click on start simulation button. Code button is used to view the C++ code.



Code :

Light LED using breadboard



```
// C++ code

int ledLight1 = 13;

void setup()
{  pinMode(ledLight1, OUTPUT);
   }

void loop()
{
    digitalWrite(ledLight1, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(ledLight1, LOW);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Light two LED using breadboard

```
// C++ code

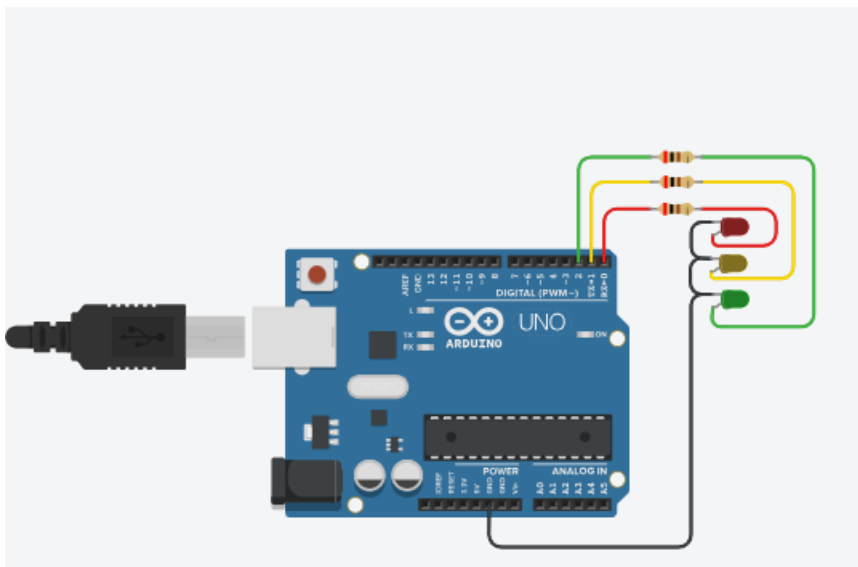
//

int ledLight1 = 13;
int ledLight2 = 12;

void setup()
{
    pinMode(ledLight1, OUTPUT);
    pinMode(ledLight2, OUTPUT);
}
```

```
void loop()
{
    digitalWrite(ledLight1, HIGH);
    digitalWrite(ledLight2, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(ledLight1, LOW);
    digitalWrite(ledLight2, LOWS);
    delay(1000); // Wait for 1000 millisecond(s)
}
```

Create Traffic Signal using LED



Code :

```
int led_red = 0; // the red LED is connected to Pin 0 of the Arduino
int led_yellow = 1; // the yellow LED is connected to Pin 1 of the Arduino
int led_green = 2; // the green LED is connected to Pin 2 of the Arduino
```

```
void setup() {  
    // set up all the LEDs as OUTPUT  
    pinMode(led_red, OUTPUT);  
    pinMode(led_yellow, OUTPUT);  
    pinMode(led_green, OUTPUT);  
}  
  
void loop() {  
    // turn the green LED on and the other LEDs off  
    digitalWrite(led_red, LOW);  
    digitalWrite(led_yellow, LOW);  
    digitalWrite(led_green, HIGH);  
    delay(2000); // wait 2 seconds  
  
    // turn the yellow LED on and the other LEDs off  
    digitalWrite(led_red, LOW);  
    digitalWrite(led_yellow, HIGH);  
    digitalWrite(led_green, LOW);  
    delay(1000); // wait 1 second  
  
    // turn the red LED on and the other LEDs off  
    digitalWrite(led_red, HIGH);  
    digitalWrite(led_yellow, LOW);
```

```
digitalWrite(led_green, LOW);
```

```
delay(3000); // wait 3 seconds
```

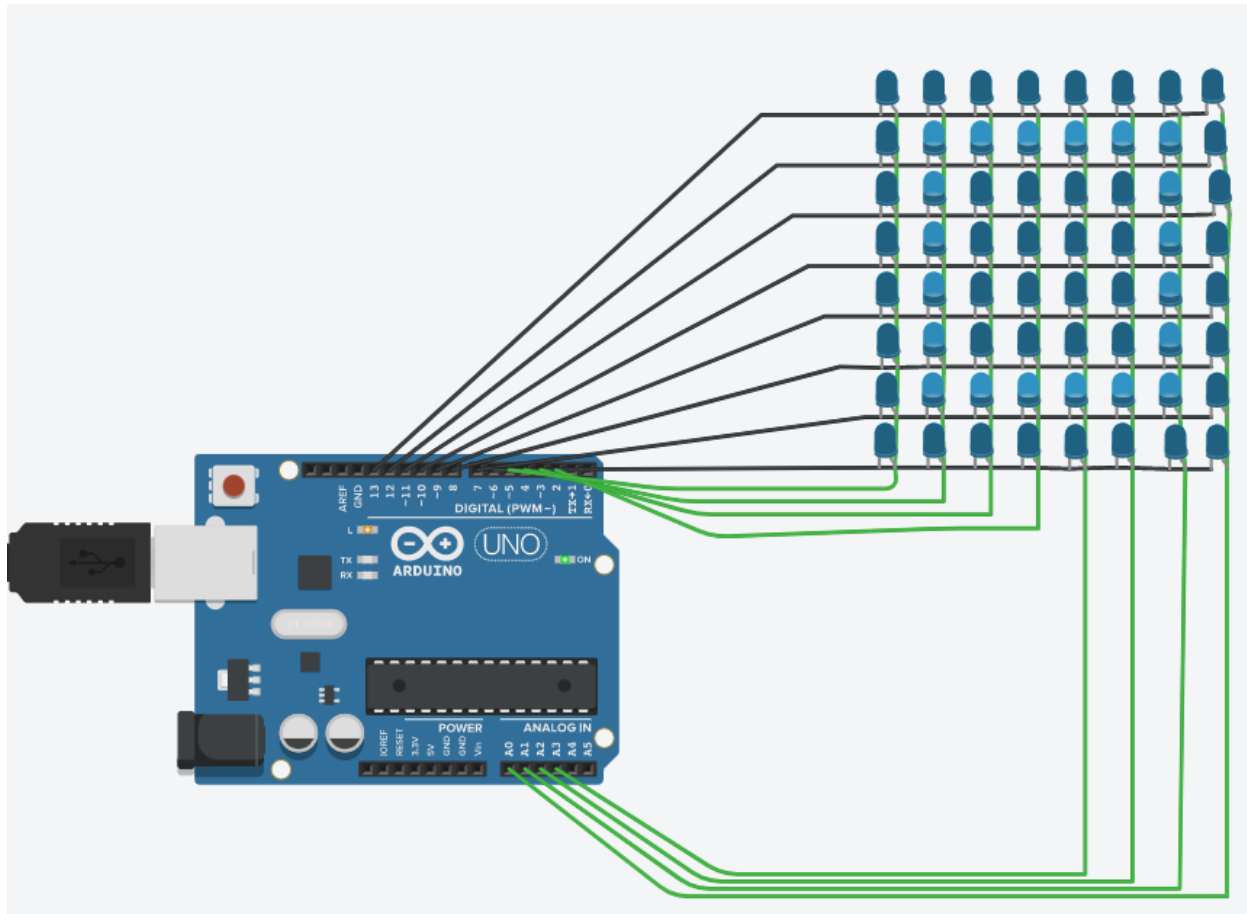
PRACTICAL 5

Aim : GPIO LED Grid Module : Program the 8x8 grid with Different Formulas

Steps :

1. Build the following circuit using www.tinkercad.com

2.



3. Use Arduino for building the circuit



4. In the  tab add the following C++ code.

```
#define ROW1 13
```

```
#define ROW2 12
```

```
#define ROW3 11
```

```
#define ROW4 10
```

```
#define ROW5 9
```

```
#define ROW6 8
```

```
#define ROW7 7
```

```
#define ROW8 6
```

```
#define COL1 5
```

```
#define COL2 4
```

```
#define COL3 3
```

```
#define COL4 2
```

```
#define COL5 A3
```

```
#define COL6 A2
```

```
#define COL7 A1
```

```
#define COL8 A0
```

```
const int row[] = {ROW1, ROW2, ROW3, ROW4, ROW5, ROW6, ROW7,  
ROW8};
```

```
const int col[] = {COL1,COL2, COL3, COL4, COL5, COL6, COL7, COL8};
```

```
int KARE1[8][8] = { {0,0,0,0,0,0,0,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,1,1,1,1,1,1,0},  
                    {0,0,0,0,0,0,0,0} };
```

```
int KARE2[8][8] = { {1,1,1,1,1,1,1,1},  
                    {1,0,0,0,0,0,0,1},  
                    {1,0,1,1,1,1,0,1},  
                    {1,0,1,1,1,1,0,1},  
                    {1,0,1,1,1,1,0,1},  
                    {1,0,1,1,1,1,0,1},  
                    {1,0,0,0,0,0,0,1},  
                    {1,1,1,1,1,1,1,1} };
```

```
int KARE3[8][8] = { {1,1,1,1,1,1,1,1},  
                    {1,1,1,1,1,1,1,1},  
                    {1,1,0,0,0,0,1,1},  
                    {1,1,0,1,1,0,1,1},  
                    {1,1,0,1,1,0,1,1},  
                    {1,1,0,0,0,0,1,1},  
                    {1,1,0,0,0,0,1,1},  
                    {1,1,0,0,0,0,1,1} };
```

```
        {1,1,1,1,1,1,1,1},  
        {1,1,1,1,1,1,1,1}};
```

```
int KARE4[8][8]={{1,1,1,1,1,1,1,1},  
                 {1,1,1,1,1,1,1,1},  
                 {1,1,1,1,1,1,1,1},  
                 {1,1,1,0,0,1,1,1},  
                 {1,1,1,0,0,1,1,1},  
                 {1,1,1,1,1,1,1,1},  
                 {1,1,1,1,1,1,1,1},  
                 {1,1,1,1,1,1,1,1}};
```

```
void setup() {  
    Serial.begin(9600);  
    for (int i = 2; i <= 13; i++) {  
        pinMode(i, OUTPUT);  
        digitalWrite(i, LOW);  
    }  
    pinMode(A0, OUTPUT);  
    digitalWrite(A0, LOW);  
    pinMode(A1, OUTPUT);  
    digitalWrite(A1, LOW);  
    pinMode(A2, OUTPUT);  
    digitalWrite(A2, LOW);  
    pinMode(A3, OUTPUT);
```

```
    digitalWrite(A3, LOW);  
}  
  
void loop() {  
  
    /* for (int c=0; c<22; c++){  
        drawScreen(KARE2);  
  
    }  
    delay(444);  
    for (int c=0; c<22; c++){  
        drawScreen(KARE3);  
  
    }  
    delay(444);  
    for (int c=0; c<22; c++){  
        drawScreen(KARE4); */  
  
    for (int c=0; c<22; c++){  
        drawScreen(KARE1);  
  
    }  
    delay(444);  
    for (int c=0; c<22; c++){
```

```
drawScreen(KARE2);

}

delay(444);
for (int c=0; c<22; c++){
drawScreen(KARE3);

}

delay(444);
    for (int c=0; c<22; c++){
drawScreen(KARE4);

}

delay(444);
    for (int c=0; c<22; c++){
drawScreen(KARE3);

}

delay(444);
    for (int c=0; c<22; c++){
drawScreen(KARE2);

}

delay(444);
```

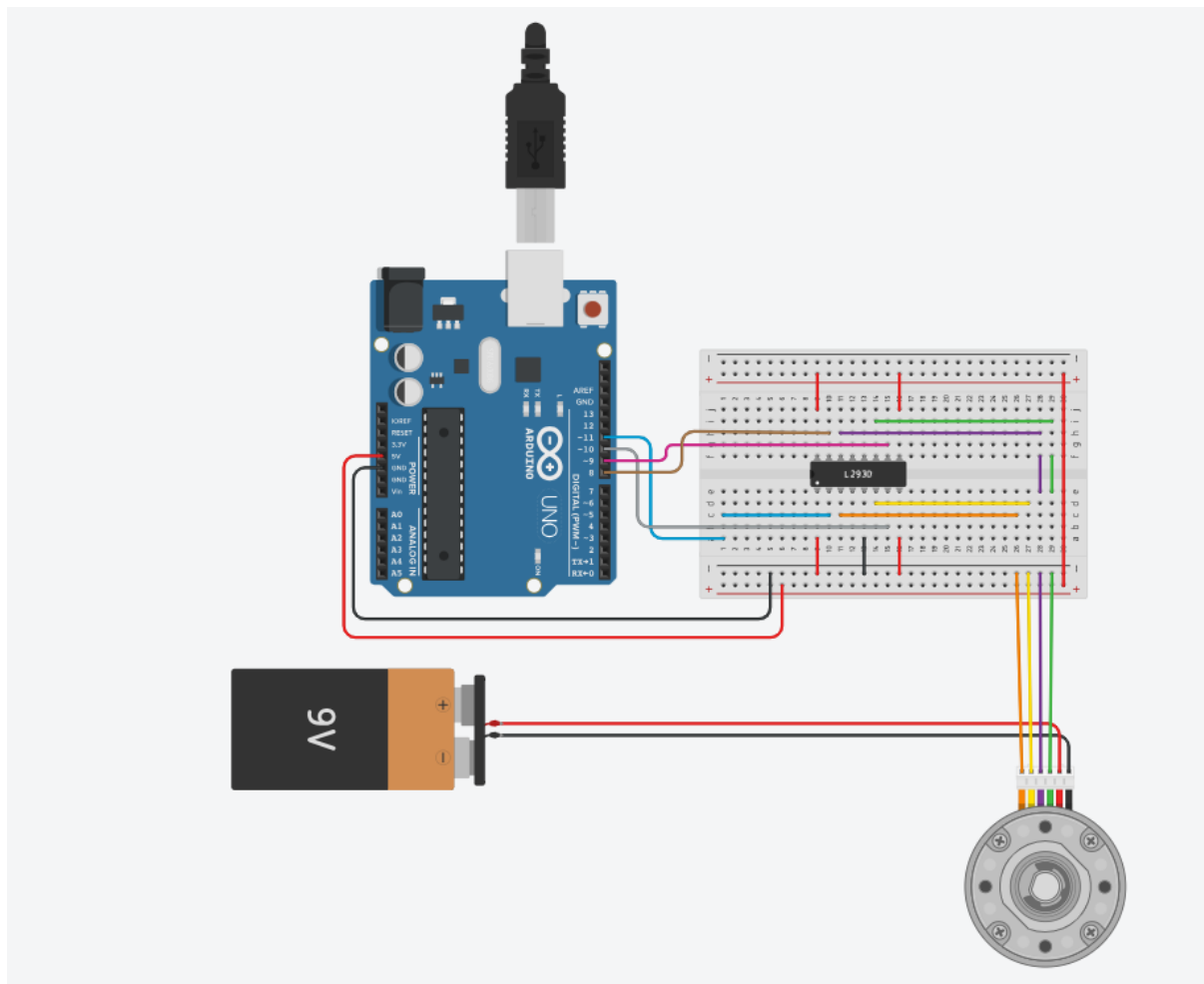
```
}
```

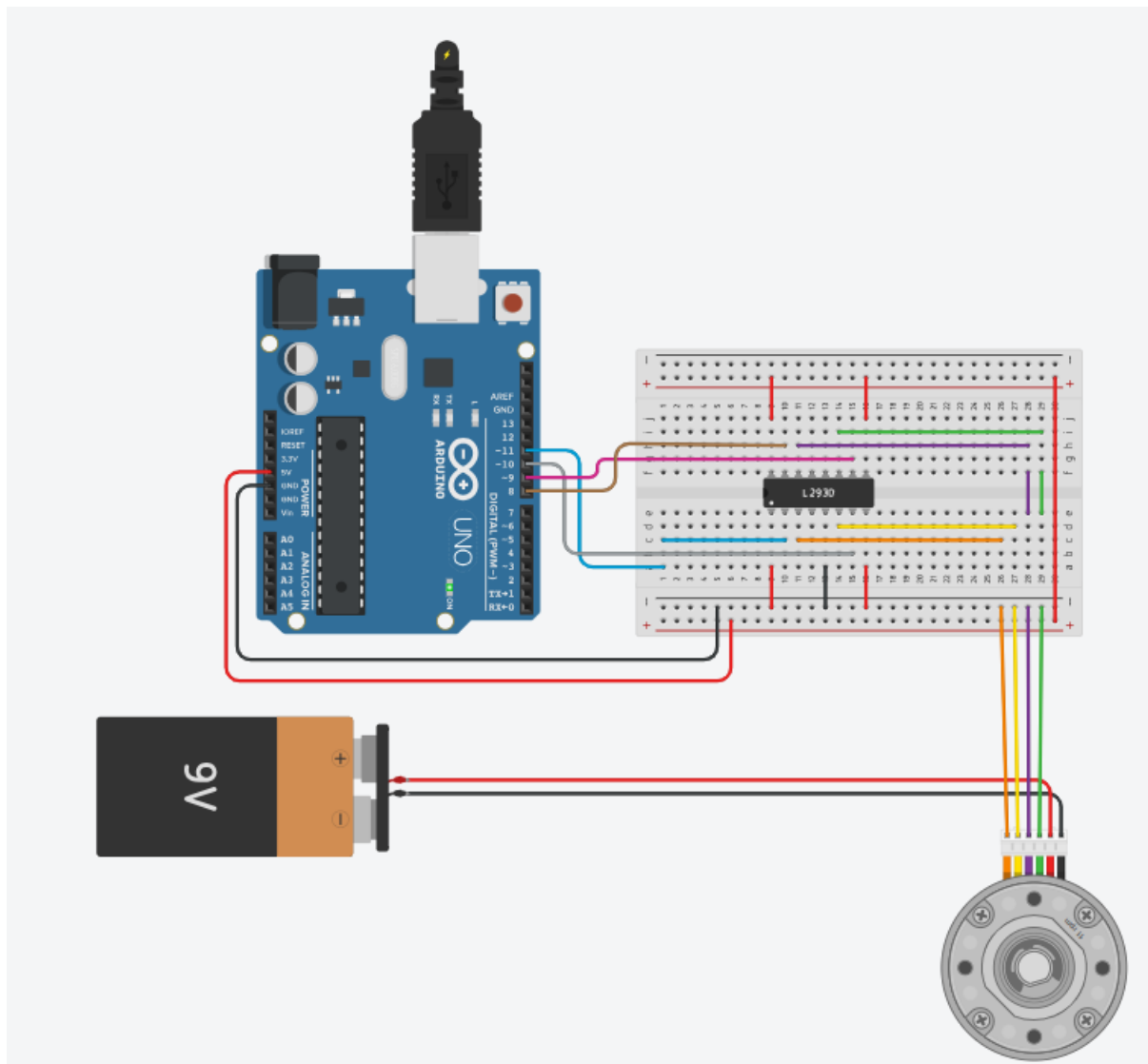
```
void drawScreen(int letter[8][8]){  
    for (int c=0; c<8; c++){  
  
        digitalWrite(col[c], HIGH);  
        for (int r = 0; r < 8; r++){  
            digitalWrite(row[r], 255*letter[r][c]);  
            delayMicroseconds(100);  
        }  
        for (int r = 0; r < 8; r++){  
            digitalWrite(row[r], HIGH);  
            delayMicroseconds(100);  
        }  
  
        digitalWrite(col[c], LOW);  
    }  
}
```

PRACTICAL 6

Aim : Stepper Motor Control: PWM to manage stepper motor speed using Uno/Raspberry Pi.

Simulator :





Code for simulator :

```
#include <Stepper.h>
```

```
const int stepsPerRevolution = 200; // change this to fit the number of steps per
revolution
```

```
// for your motor
```



```
// initialize the stepper library on pins 8 through 11:

Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);


int stepCount = 0; // number of steps the motor has taken


void setup() {

    // nothing to do inside the setup

}


void loop() {

    // read the sensor value:

    int sensorReading = analogRead(A0);

    Serial.print("10 bit number : " );

    Serial.print(sensorReading);

    //WriteLine(sensorReading);

    // map it to a range from 0 to 100:

    int motorSpeed = map(sensorReading, 0, 1023, 0, 250);

    // set the motor speed:

    if (motorSpeed > 0) {

        myStepper.setSpeed(motorSpeed);
```

```
// step 1/100 of a revolution:
```

```
myStepper.step(stepsPerRevolution / 100);
```

```
}
```

```
}
```

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameters

`value`: the number to map.

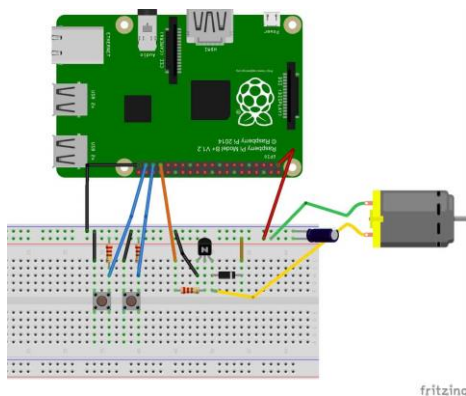
`fromLow`: the lower bound of the value's current range.

`fromHigh`: the upper bound of the value's current range.

`toLow`: the lower bound of the value's target range.

`toHigh`: the upper bound of the value's target range.

Using Kit :



Python Code :

```
#!/usr/bin/env python
```

```
# import required libs
```

```
import time
```

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

```
GPIO.cleanup() #cleaning up in case GPIOs have been preactivated
```

```
# Use BCM GPIO references
```

```
# instead of physical pin numbers
```

```
GPIO.setmode(GPIO.BCM)
```

```
# be sure you are setting pins accordingly
```

```
# GPIO17,GPIO18,GPIO22,GPI23
```

```
StepPins = [17,18,22,23]
```

```
# Set all pins as output
```

```
for pin in StepPins:
```

```
    GPIO.setup(pin,GPIO.OUT)
```

```
    GPIO.output(pin, False)
```

```
#wait some time to start
```

```
time.sleep(0.5)
```

```
# Define some settings
```

```
StepCounter = 0
```

```
WaitTime = 0.0015
```

Define simple sequence

StepCount1 = 4

Seq1 = []

Seq1 = range(0, StepCount1)

Seq1[0] = [1,0,0,0]

Seq1[1] = [0,1,0,0]

Seq1[2] = [0,0,1,0]

Seq1[3] = [0,0,0,1]

Define advanced sequence

as shown in manufacturers datasheet

StepCount2 = 8

Seq2 = []

Seq2 = range(0, StepCount2)

Seq2[0] = [1,0,0,0]

Seq2[1] = [1,1,0,0]

Seq2[2] = [0,1,0,0]

Seq2[3] = [0,1,1,0]

Seq2[4] = [0,0,1,0]

Seq2[5] = [0,0,1,1]

Seq2[6] = [0,0,0,1]

Seq2[7] = [1,0,0,1]

```
#Full torque
```

```
StepCount3 = 4
```

```
Seq3 = []
```

```
Seq3 = [3,2,1,0]
```

```
Seq3[0] = [0,0,1,1]
```

```
Seq3[1] = [1,0,0,1]
```

```
Seq3[2] = [1,1,0,0]
```

```
Seq3[3] = [0,1,1,0]
```

```
# set
```

```
Seq = Seq2
```

```
StepCount = StepCount2
```

```
# Start main loop
```

```
try:
```

```
    while 1==1:
```

```
        for pin in range(0, 4):
```

```
            xpin = StepPins[pin]
```

```
            if Seq[StepCounter][pin]!=0:
```

```
                #print " Step %i Enable %i" %(StepCounter,xpin)
```

```
                GPIO.output(xpin, True)
```

```
            else:
```

```
                GPIO.output(xpin, False)
```

```
        StepCounter += 1
```

```
# If we reach the end of the sequence
# start again

if (StepCounter==StepCount):
    StepCounter = 0

if (StepCounter<0):
    StepCounter = StepCount

# Wait before moving on

time.sleep(WaitTime)

except:

    GPIO.cleanup();

finally: #cleaning up and setting pins to low again (motors can get hot if you
wont)

    GPIO.cleanup();

    for pin in StepPins:

        GPIO.setup(pin,GPIO.OUT)

        GPIO.output(pin, False)
```

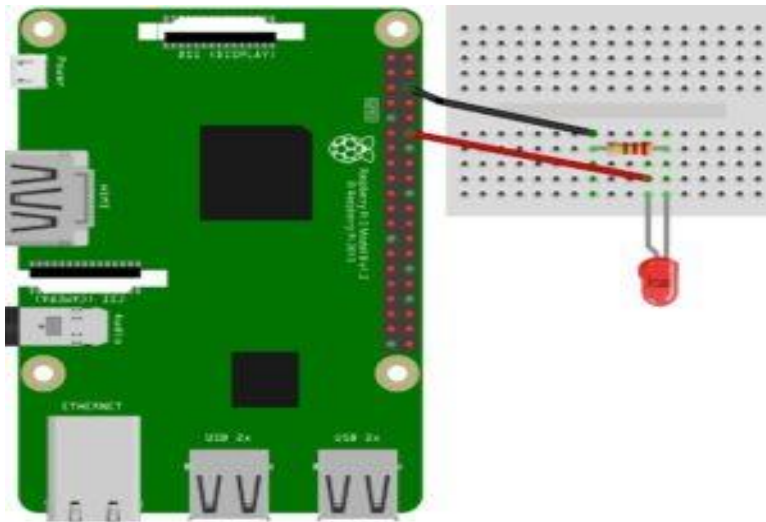
PRACTICAL 7

AIM: Node RED: Connect LED to Internet of Things

NODE Red: Connect LED to Internet of Things.

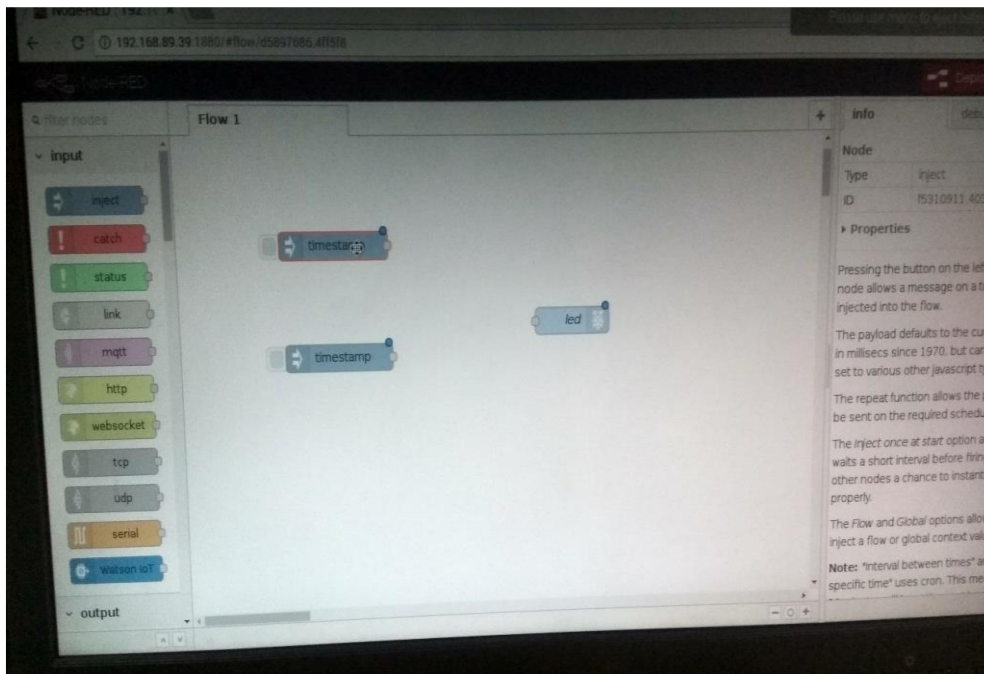
Hardware:

In this example, the LED is connected to GPIO-18 of the Raspberry Pi module.



Creating the flow:

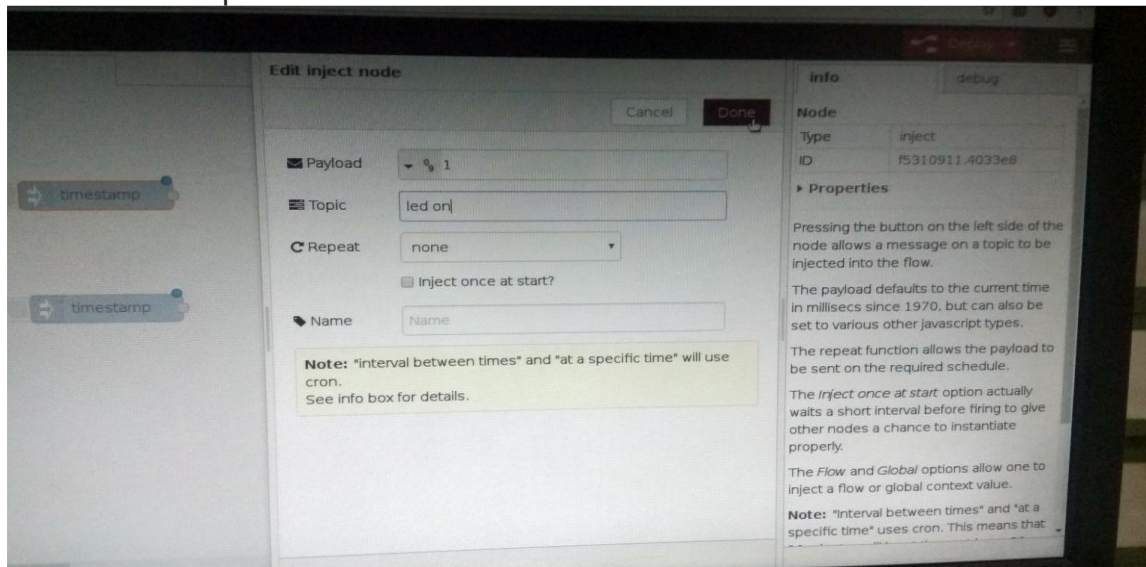
for this example we need the following nodes:



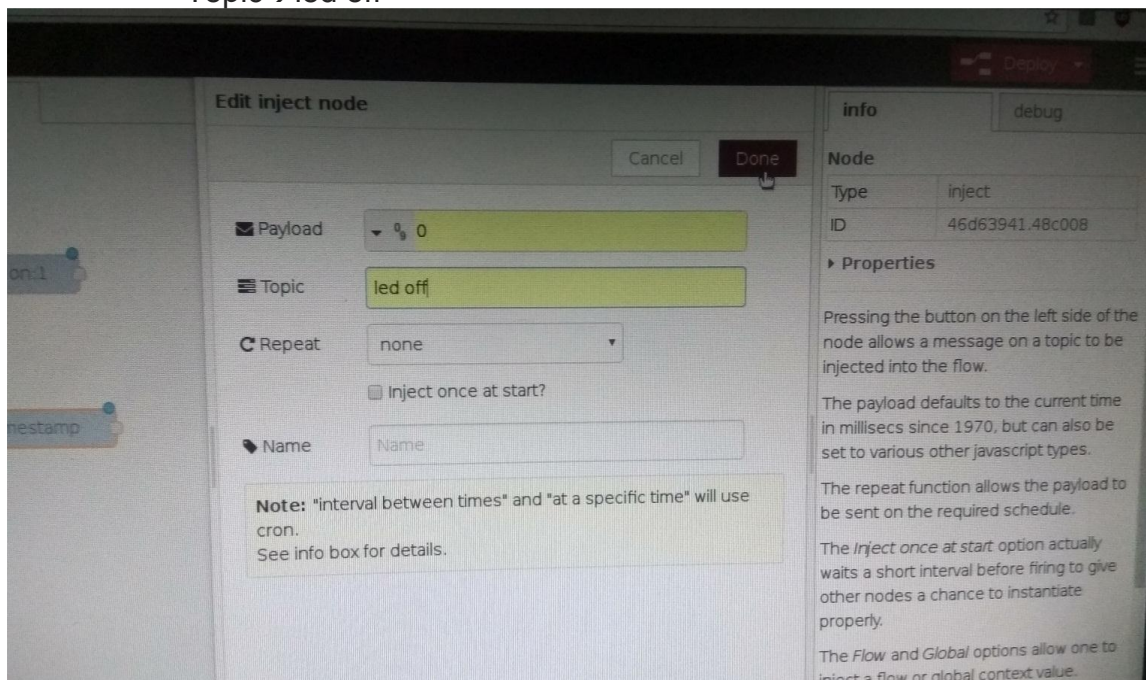
- *Inject.*
- *rpi-gpio out*

Node Properties:

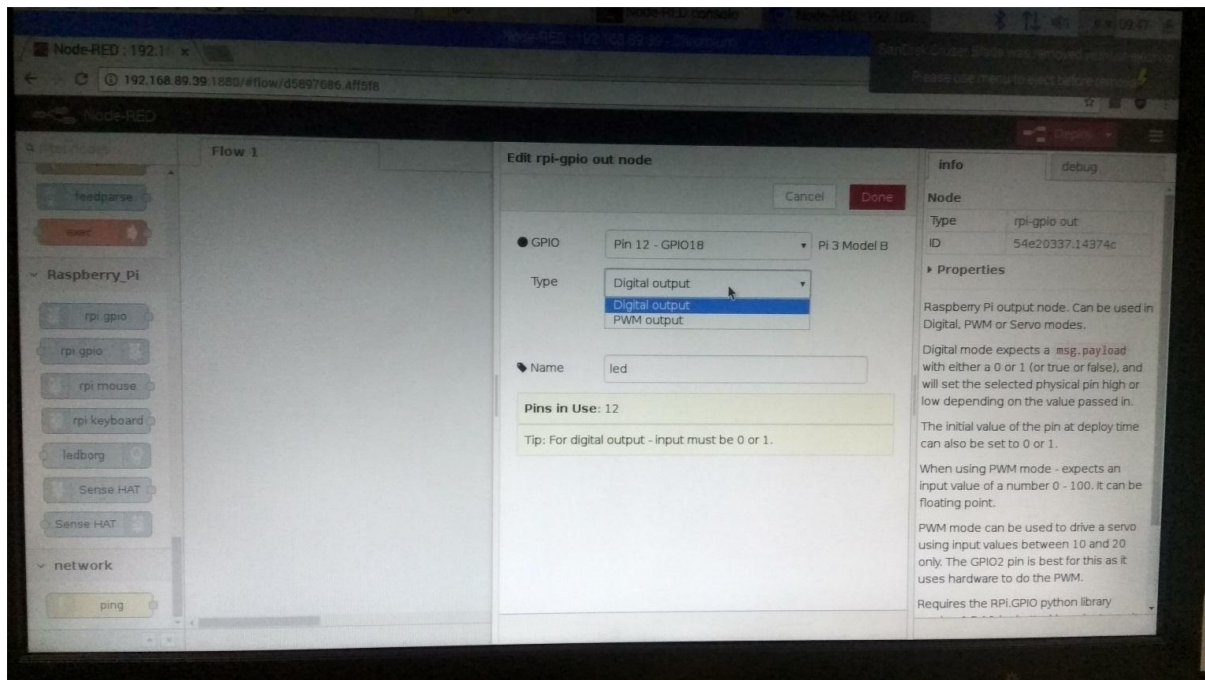
- *Inject:*
Payload → 1
Topic → led on



Payload → 0
Topic → led off

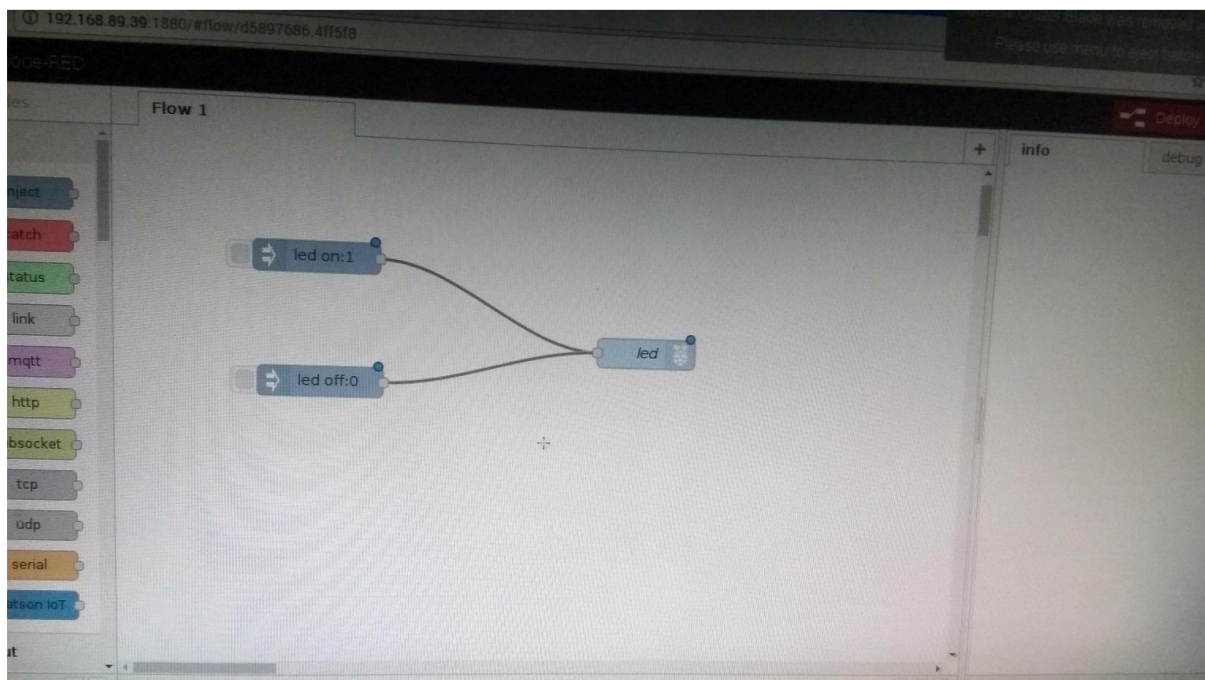


- *rpi-gpio out*
 - Pin → 12-GPIO18.
 - Type → Digital output.



Connection:

The output of the **Inject** is connected to the input of the **rpi-gpio out** node.



Output:

You need to click on deploy button on successful deployment led will go high on clicking inject with 1 input and will go low on clicking inject with 0.

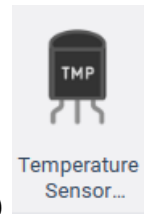
PRACTICAL 8

AIM: Use different types of sensors (LDR, Temperature) with Raspberry Pi/Uno.

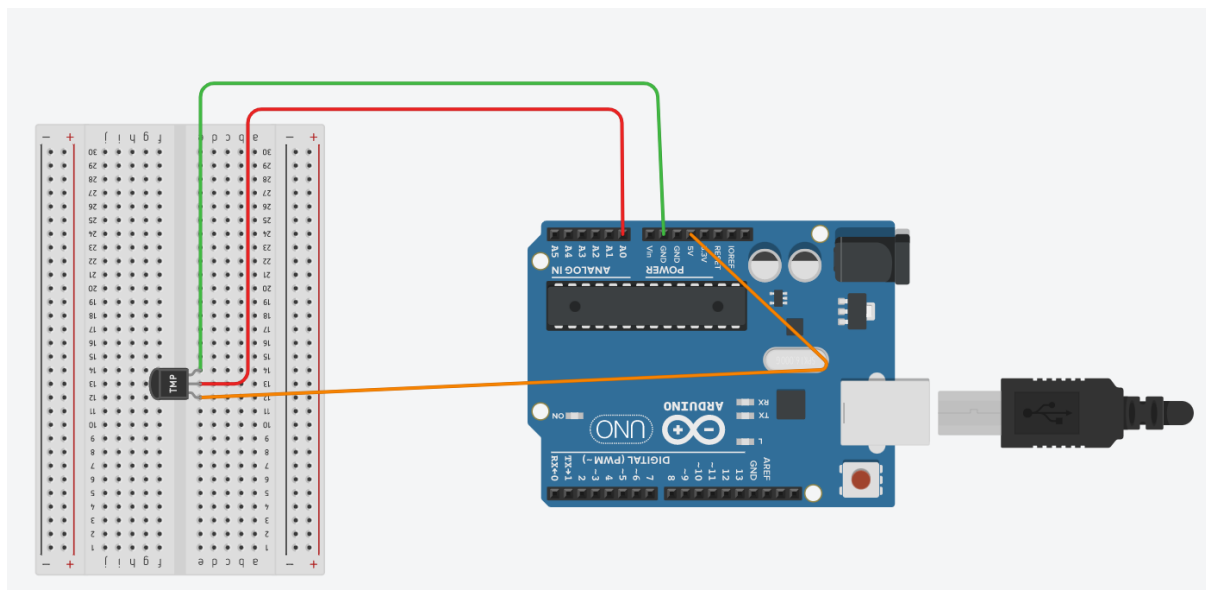
Temperature Sensor

Component needed

1. Arduino
2. Temperature sensor (tmp26)
3. Breadboard



Circuit



Code :

```
char degree=176;
```

```
void setup()
```

```
{
```

```
  pinMode(A0,INPUT);
```

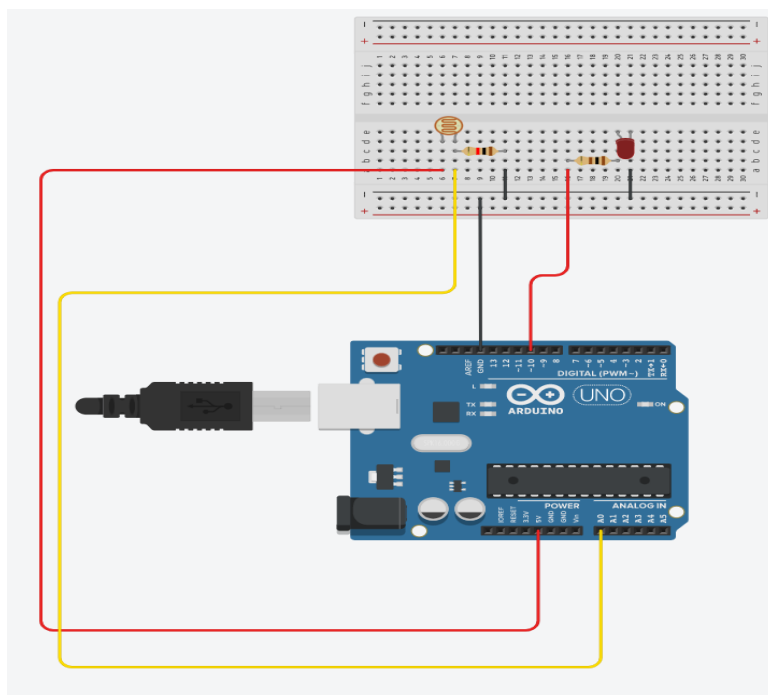
Fatema Kothari

Asst. Prof, Department of Computer Science, SIES(Nerul)

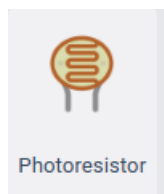
```
Serial.begin(9600);  
  
}  
  
void loop()  
{  
    int temp = analogRead(A0);  
    float voltage = (temp*5)/1024;  
    float milivolt = voltage*1000;  
    float celsius = (milivolt-500)/10;  
    float fahrenheit = (celsius*9/5)+32;  
    Serial.print("10 bit number : " );  
    Serial.println(temp);  
  
    Serial.print("voltage : " );  
    Serial.print(voltage);  
  
    Serial.print("milivolt : " );  
    Serial.print(milivolt);  
    Serial.println("V");  
  
    Serial.print("celsius : " );  
    Serial.print(celsius);  
    Serial.println("degree");
```

```
Serial.print("fahrenheit : " );  
Serial.print(fahrenheit);  
delay(1000);  
}
```

LDR Sensor



Component



- Photoresistor
- LED
- 2 resistors
- Breadboard

- Arduino

Code

```
const int ledPin = 10;
```

```
const int ldrPin = A0;
```

```
void setup() {
```

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

```
    pinMode(ledPin, OUTPUT);
```

```
    pinMode(ldrPin, INPUT);
```

```
}
```

```
void loop() {
```

```
    int ldrStatus = analogRead(ldrPin);
```

```
    Serial.println(ldrStatus);
```

```
    if (ldrStatus <=600) {
```

```
        digitalWrite(ledPin, HIGH);
```

```
}  
else {  
    digitalWrite(ledPin, LOW);  
} }
```

PRACTICAL 9

AIM: Trigger a set of led GPIO on any IoT platform via any related web server

Steps :

Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages.

On its own, Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

INSTALL APACHE

First install the apache2 package by typing the following command in to the Terminal:


```
sudo apt-get update
```

```
sudo apt-get install apache2 -y
```

TEST THE WEB SERVER

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to ***http://localhost/*** on the Pi itself, or ***http://192.168.1.10*** (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type ***hostname -I*** at the command line (or read more about finding your [IP address](#)).

Browse to the default web page either on the Pi or from another computer on the network and you should see the following:



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

This means you have Apache working!

CHANGING THE DEFAULT WEB PAGE

This default web page is just a HTML file on the filesystem. It is located at **`/var/www/html/index.html`**.

Note: The directory was `/var/www` in Raspbian Wheezy but is now `/var/www/html` in Raspbian Jessie

Navigate to this directory in a terminal window and have a look at what's inside:

`cd /var/www/html`

ls -al

This will show you:

```
total 12
```

```
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .
```

```
drwxr-xr-x 12 root root 4096 Jan  8 01:28 ..
```

```
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

This shows that by default there is one file in /var/www/html/ called index.html and it is owned by the root user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default pi user is assumed here) using

sudo chown pi: index.html.

You can now try editing this file and then refreshing the browser to see the web page change.

YOUR OWN WEBSITE

If you know HTML you can put your own HTML files and other assets in this directory and serve them as a website on your local network.

PRACTICAL 10

AIM: Interface with any sensor and send its value over the internet to the server using any suitable protocol