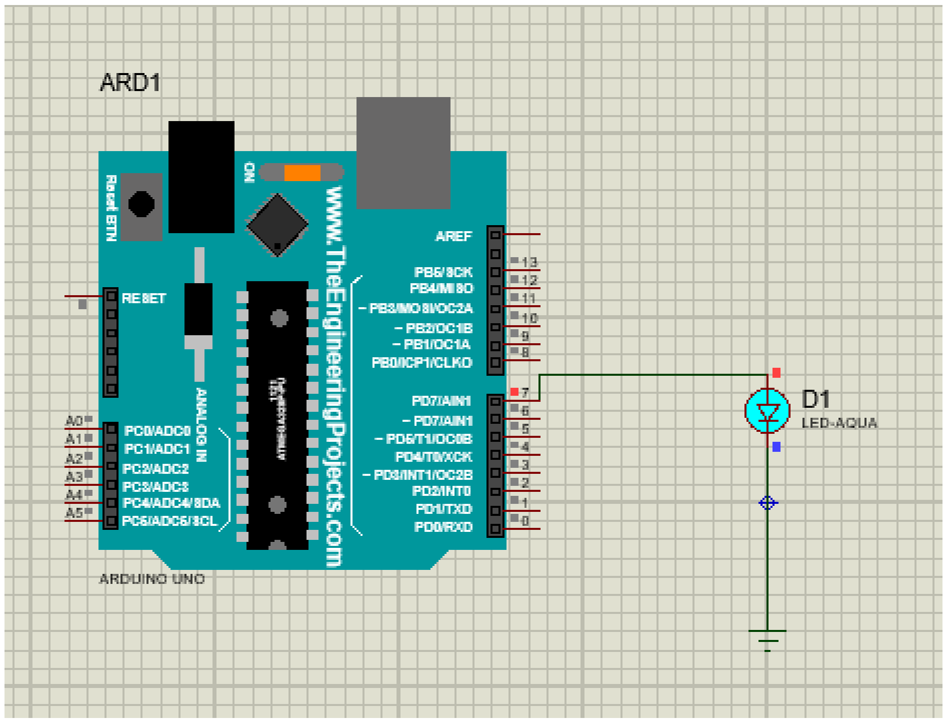
LAB (1 - 13)

Task No. 1: Write a sketch to interface Arduino with LED, LED should blink with a delay of 1 second.

Solution:

```
int LED = 7;
void setup() {
pinMode(LED, OUTPUT);
}

void loop() {
digitalWrite(LED, HIGH);
delay(1000);
digitalWrite(LED, LOW);
delay(1000);
}
```

Output:

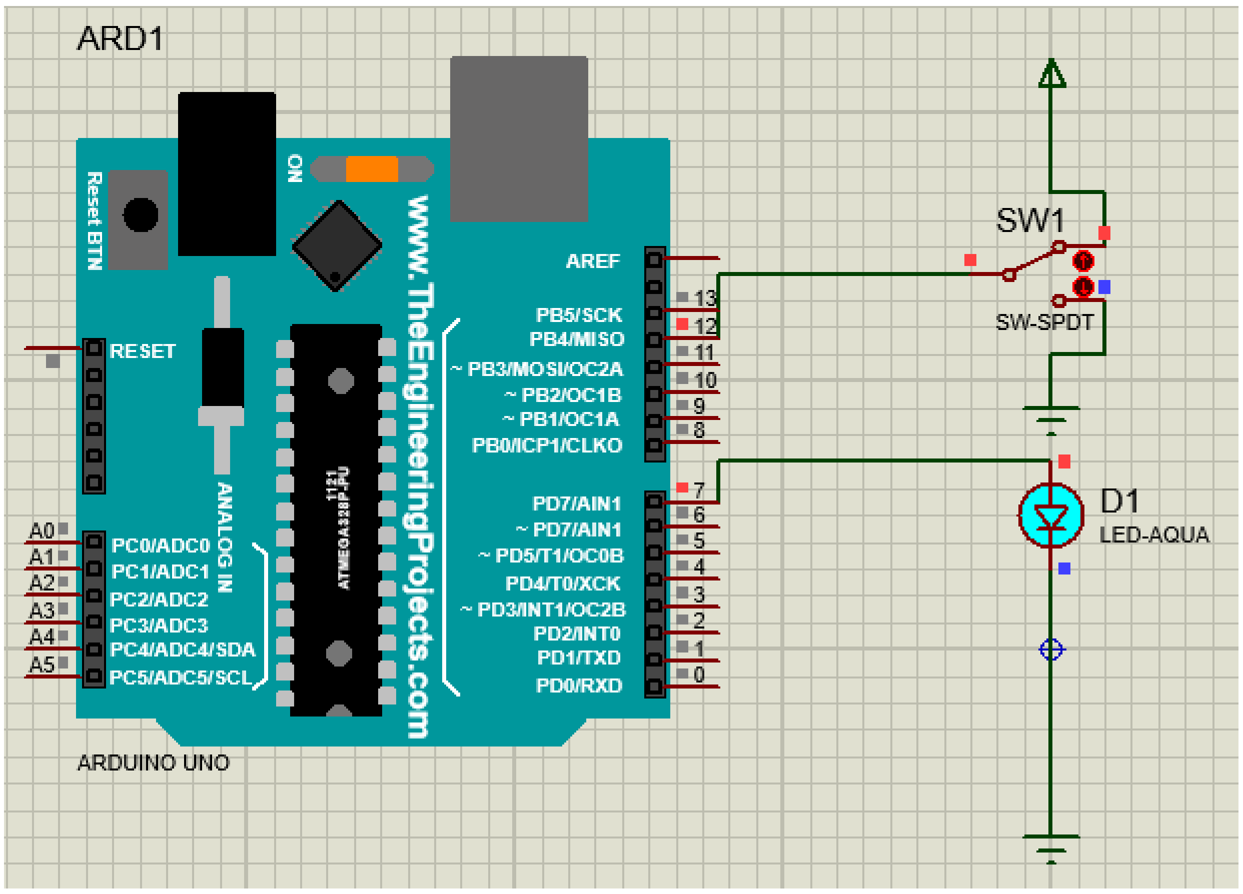


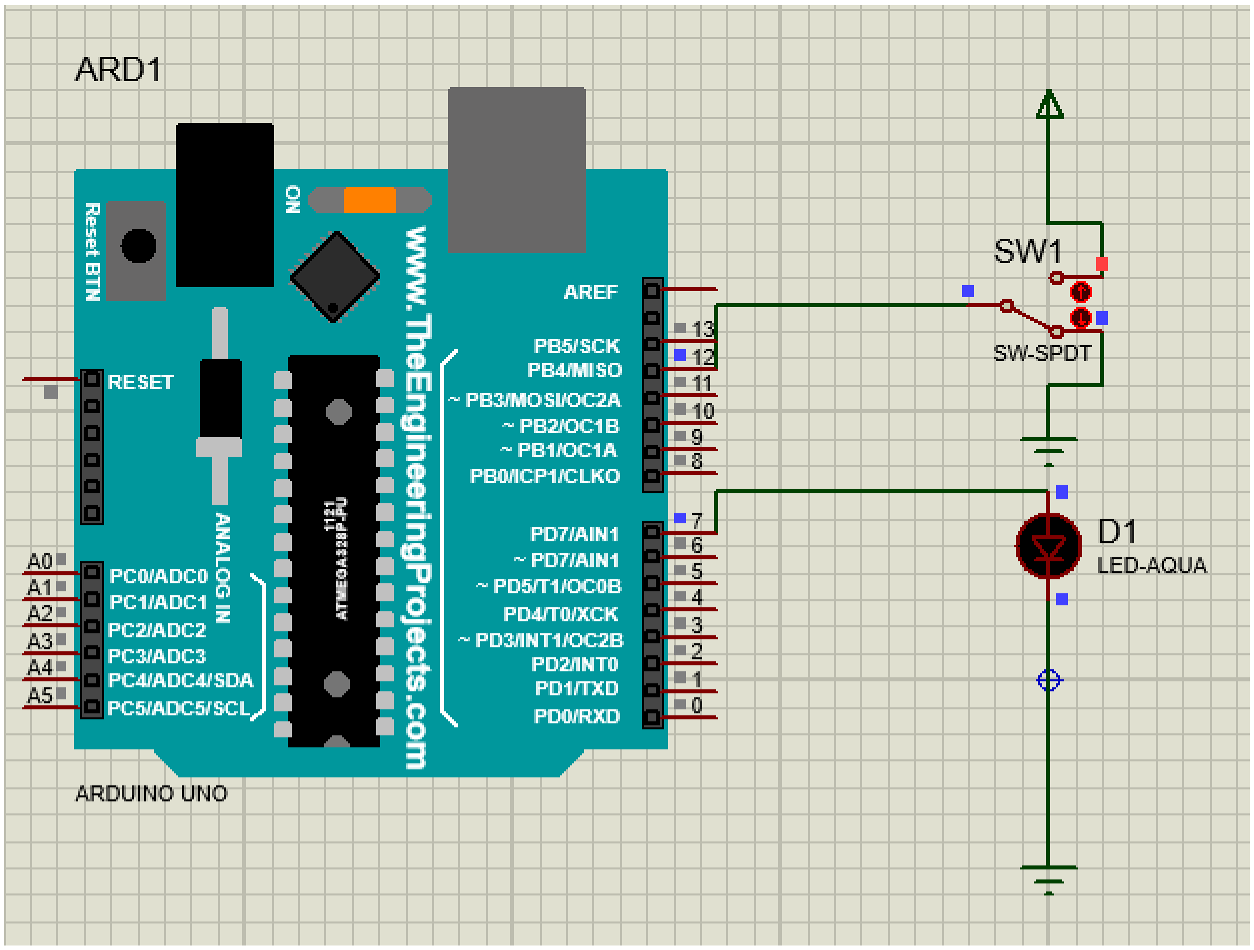
Task No. 2: Write a sketch to interface Arduino with SPDT switch & LED.

Solution:

```
int LED = 7;
int SW = 12;
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SW, INPUT);
}
void loop() {
  int buttonstate = digitalRead(SW);

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

Output:



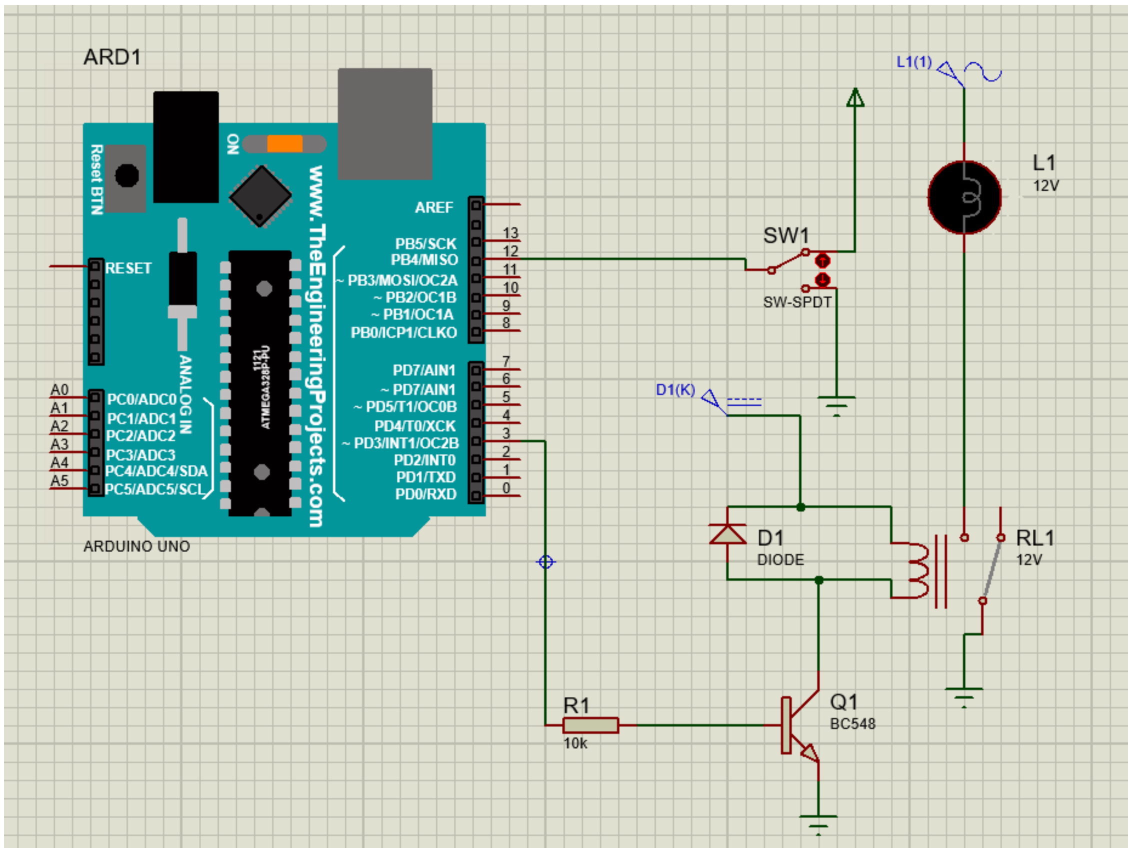

ARD1
Reset BTN
ON
www.TheEngineeringProjects.com
ATMEGA328P-PU
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
RESET
ANALOG IN
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ARDUINO UNO
SW1
SW-SPDT
D1
LED-AQUA

Task No. 1: Write a sketch to interface Arduino with the Relay. The Relay should be controlled by a SPDT Switch.

Solution:

```
int buttonPin = 13;
int relayPin = 3;
void setup() {
  pinMode(relayPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
  int buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH)
  {
    digitalWrite(relayPin, HIGH);
  }
  else {
    digitalWrite(relayPin, LOW);
  }
}
```

Output:



Task No. 2: Write a sketch to interface Arduino with Seven Segment Display. It should work as a decade counter. The Start / Stop of counting should be controlled through a SPDT Switch.

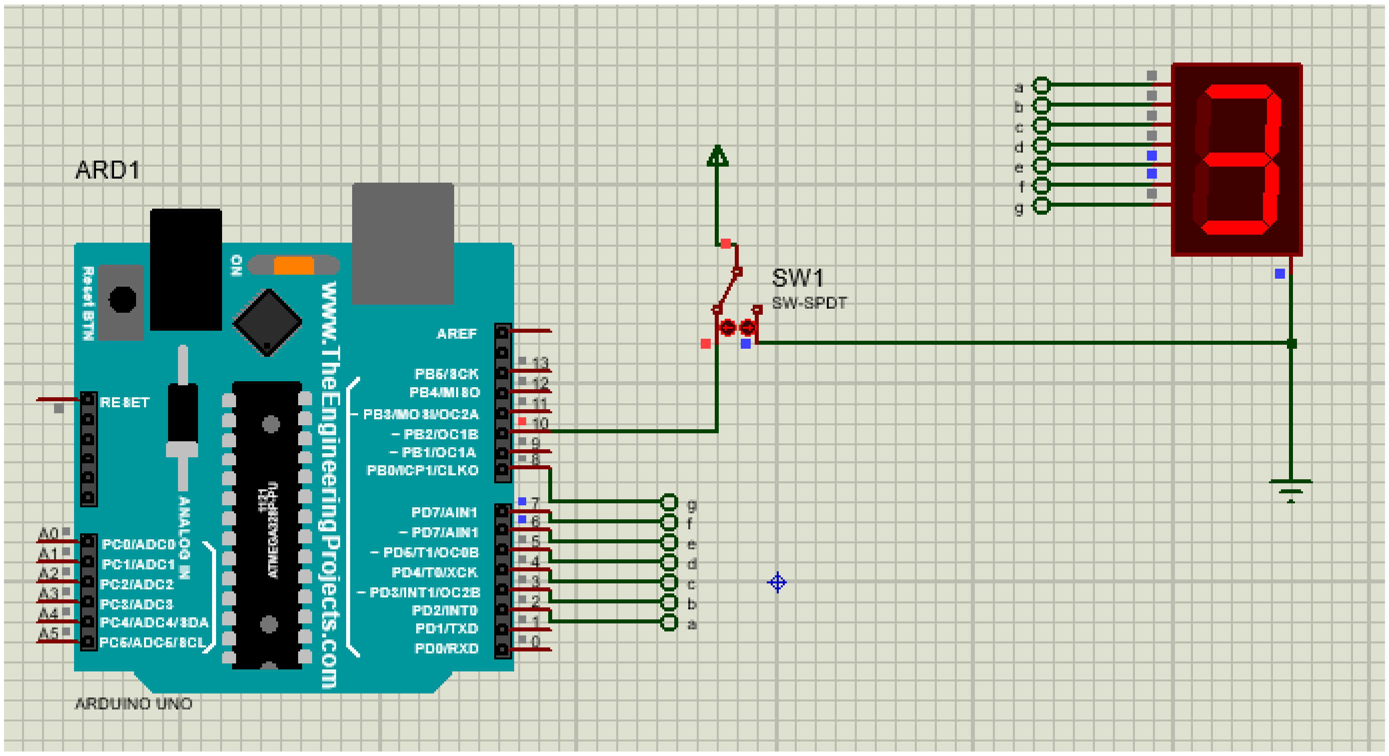
Solution:

```
#define segA 2
#define segB 3
#define segC 4
#define segD 5
#define segE 6
#define segF 7
#define segG 8
#define button 10
int COUNT = 0;
int ButtonState;
// Count Integer for 0-9 Increment.
// Variable Checking the State of the Button.
void setup() {
  for (int i = 2; i < 9; i++) {
    pinMode(i, OUTPUT);
  }
  pinMode(10, INPUT);
}
void loop() {
  ButtonState = digitalRead(button);
  if (ButtonState == HIGH) {
    switch (COUNT) {
      case 0:
        digitalWrite(segA, HIGH);
        digitalWrite(segB, HIGH);
        digitalWrite(segC, HIGH);
        digitalWrite(segD, HIGH);
        digitalWrite(segE, HIGH);
        digitalWrite(segF, HIGH);
        digitalWrite(segG, LOW);
        break;
      case 1:
        digitalWrite(segA, LOW);
        digitalWrite(segB, HIGH);
        digitalWrite(segC, HIGH);
        digitalWrite(segD, LOW);
        digitalWrite(segE, LOW);
        digitalWrite(segF, LOW);
        digitalWrite(segG, LOW);
        break;
      case 2:
        digitalWrite(segA, HIGH);
        digitalWrite(segB, HIGH);
        digitalWrite(segC, LOW);
        digitalWrite(segD, HIGH);
        digitalWrite(segE, HIGH);
        digitalWrite(segF, LOW);
        digitalWrite(segG, HIGH);
        break;
      case 3:
```

```
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, HIGH);
    break;
  case 4:
    digitalWrite(segA, LOW);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    break;
  case 5:
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    break;
  case 6:
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    break;
  case 7:
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    break;
  case 8:
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    break;
  case 9:
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
```

```
      digitalWrite(segE, LOW);
      digitalWrite(segF, HIGH);
      digitalWrite(segG, HIGH);
      break;
    }
    if (COUNT < 10) {
      COUNT++;
      delay(1000);
    }
    if (COUNT == 10) {
      COUNT = 0;
      delay(1000);
    }
  }
}
```

Output:

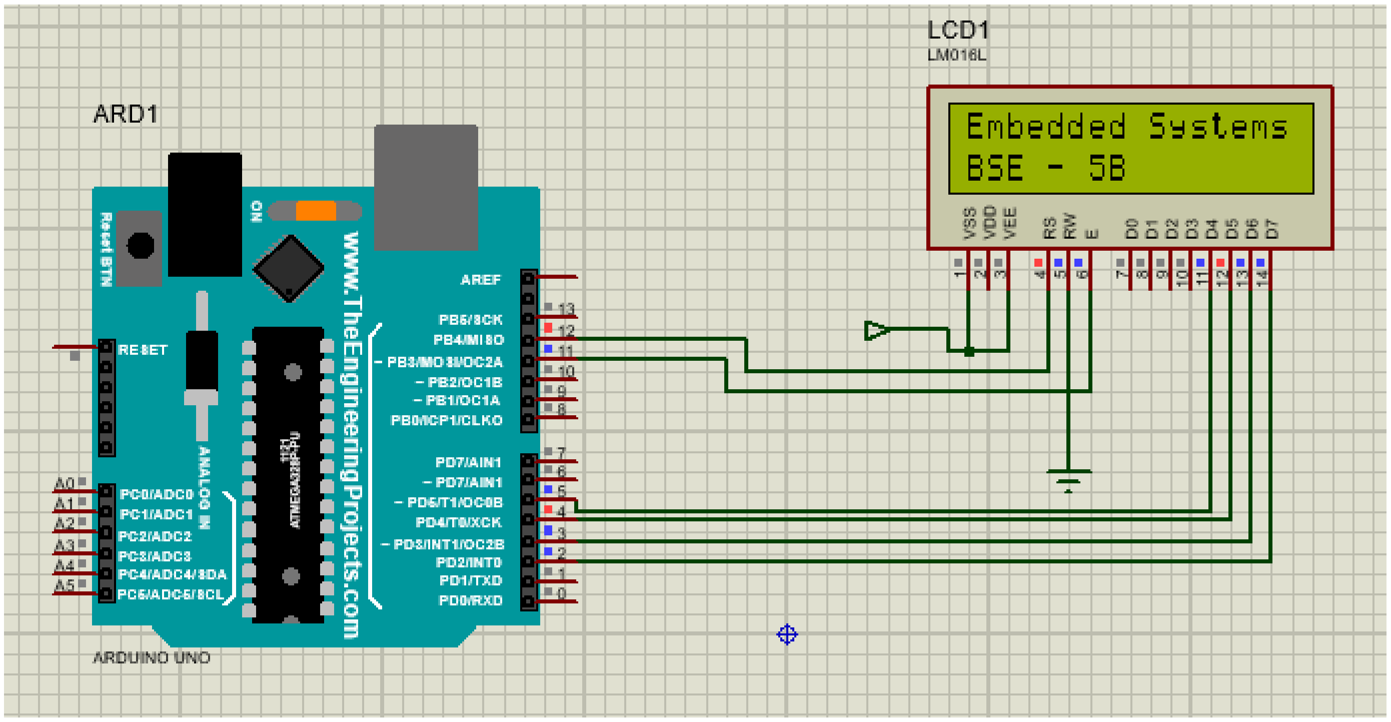


Task No. 1: Write a sketch to interface Arduino with 16x2 Liquid Crystal Display (LCD). Write the name of your course "Embedded Systems" in the 1st Line and your Section "BSE - 5B" in the 2nd Line of LCD. This Text should blink with a delay of 0.5 seconds.

Solution:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
  lcd.begin(16, 2);  // Setting Up the LCD No. of Rows & Columns
}
void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Embedded Systems");
  lcd.setCursor(0, 1);
  lcd.print("BSE - 5B");
  delay(500);
  lcd.clear();
  delay(5000);
}
```

Output:

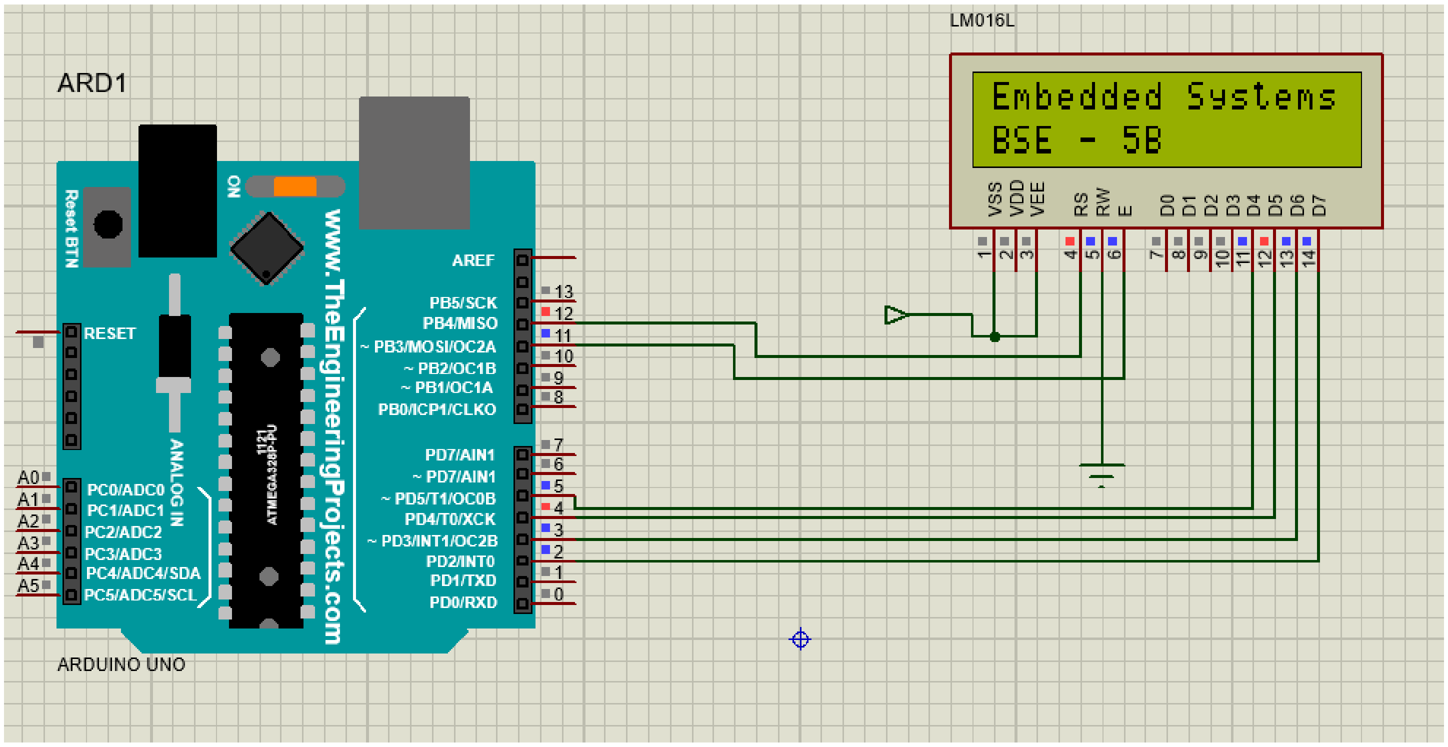

LCD1
LM016L
Embedded Systems
BSE - 5B
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
AREF
RESET
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7

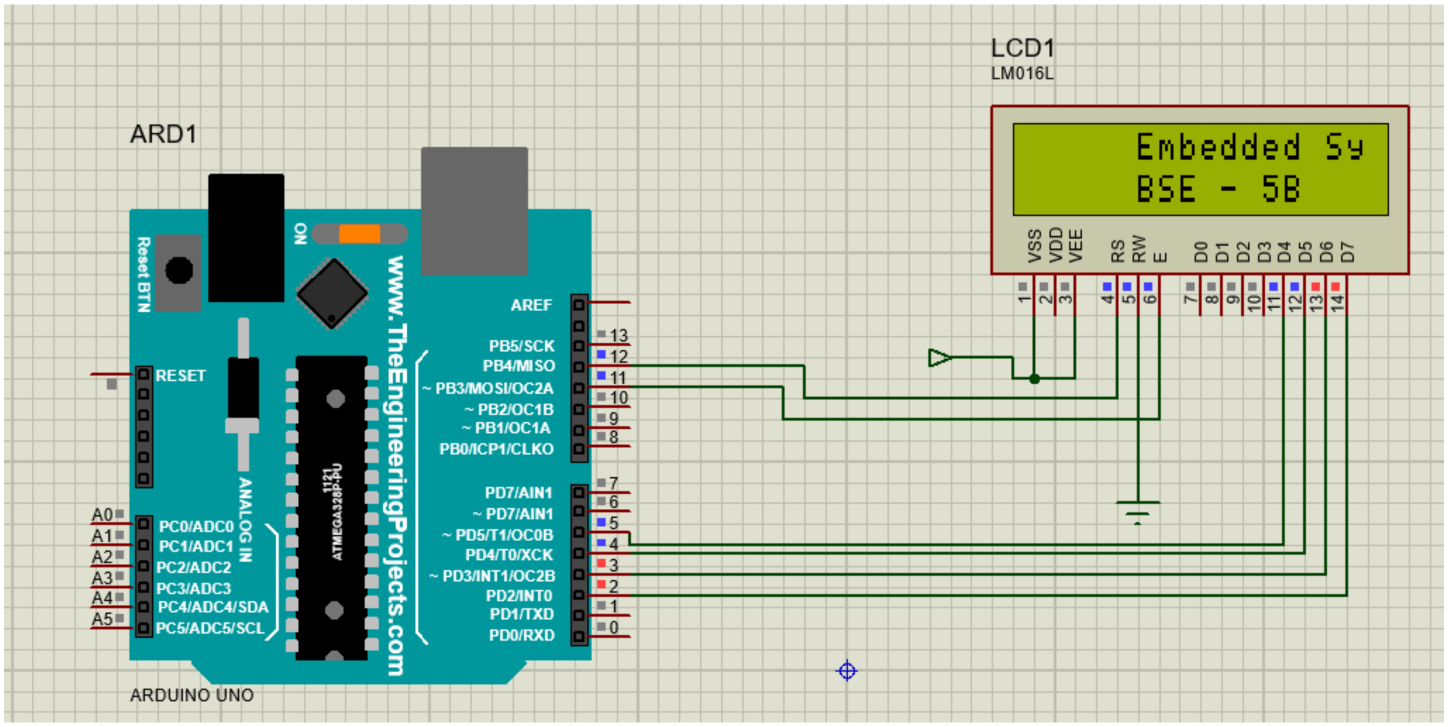
Task No. 2: Write a sketch to interface Arduino with 16x2 Liquid Crystal Display (LCD). First line of LCD should display your name, second line of LCD should display your registration number, and text in both line should keep moving from left to right.

Solution:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
  lcd.begin(16, 2);
}
void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Embedded Systems");
  lcd.setCursor(0, 1);
  lcd.print("BSE - 5B");
  delay(500);

  for (int i = 0; i < 29; i++) {
    lcd.scrollDisplayRight();
    delay(100);
  }

  delay(1000);
  lcd.clear();
  delay(5000);
}
```

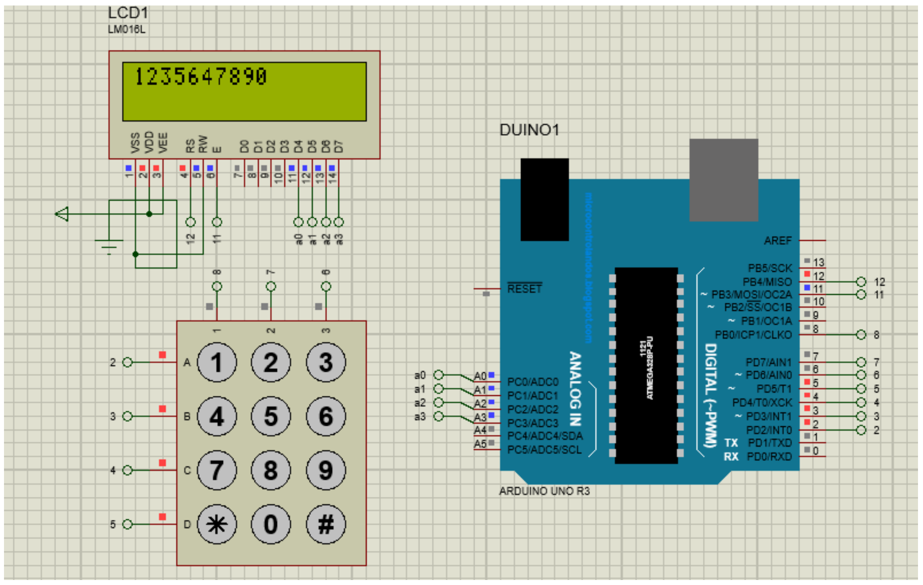
Output:




LCD1
LM016L
Embedded Sy
BSE - 5B
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
ATMEGA328P-PU
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
RESET
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ANALOG IN
Reset BTN
ON

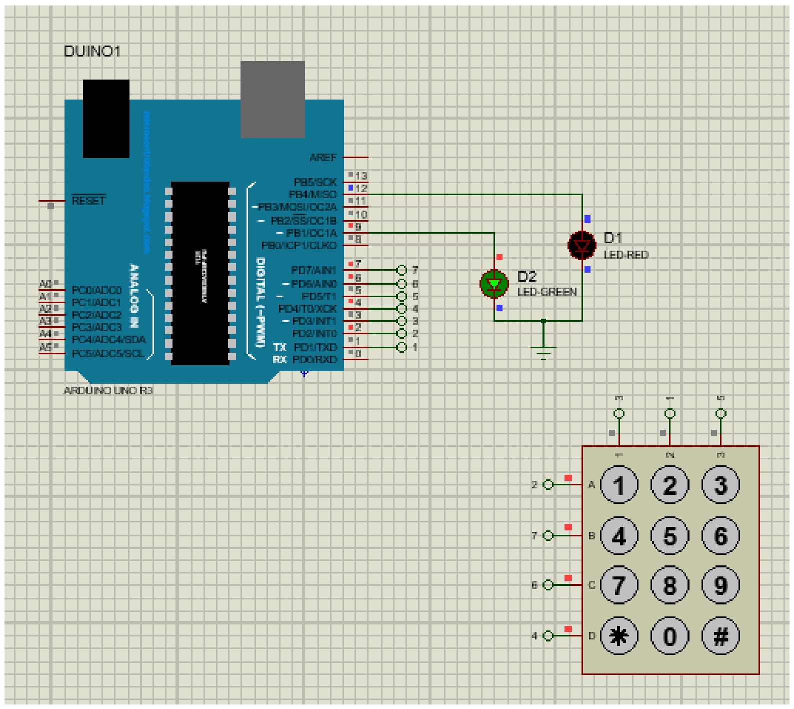
Task No. 1: Write a sketch to interface Arduino with a 3 x 4 Matrix Keypad. The display of the pressed key should be displayed on the LCD.

Solution:



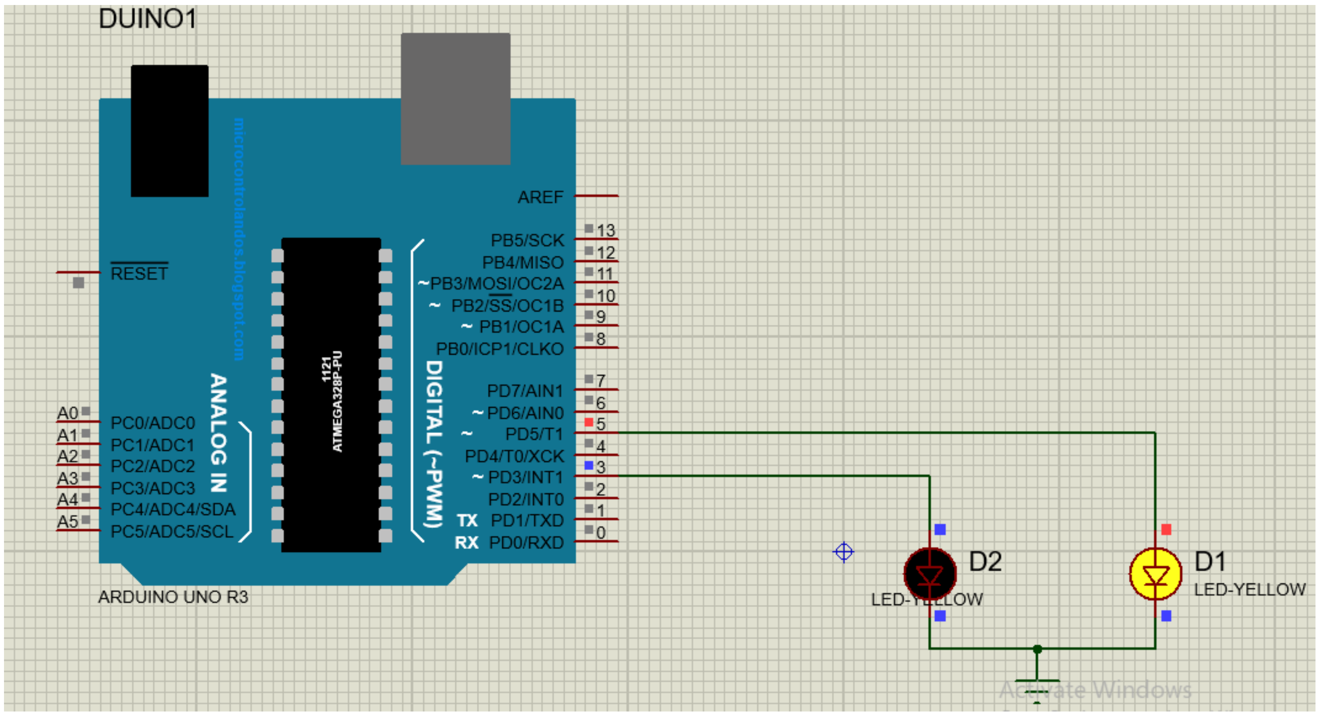
Task No. 2: Write a sketch that works as security keypad lock. Set any password. If the input password matches with the set password, Green LED should glow, otherwise Red LED will glow.

Solution:



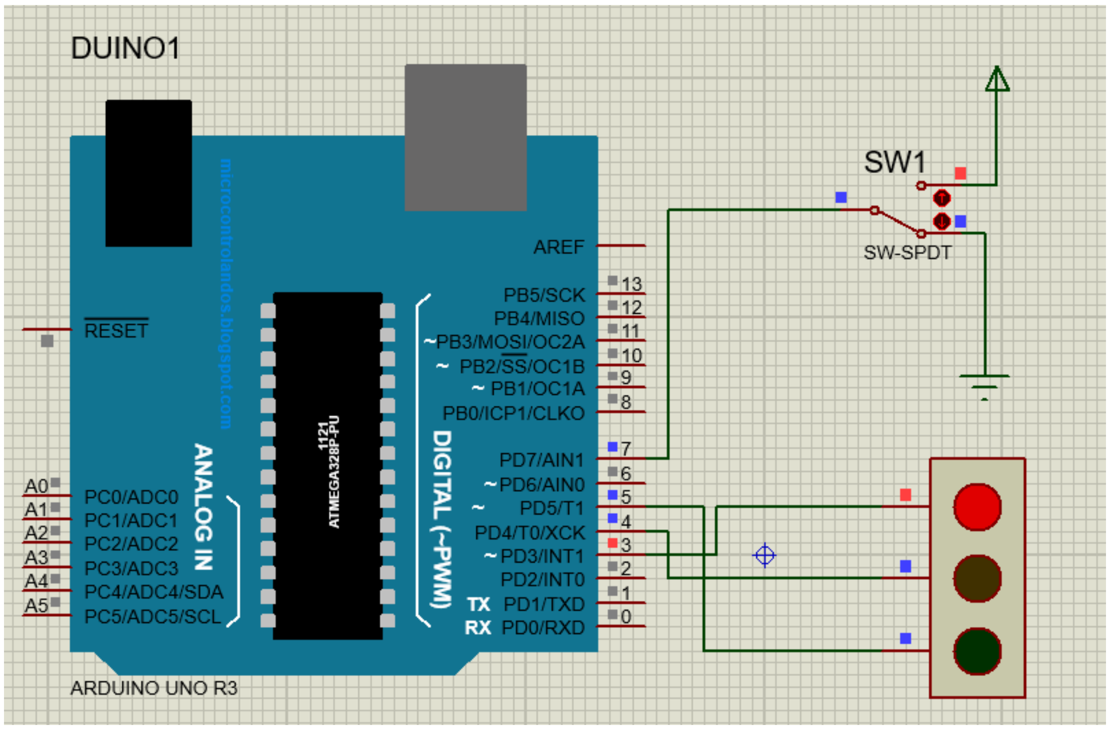
Task 1: Write a sketch to blink the 2 LEDs interfaced with Arduino at a different rate simultaneously. (i.e. "delay" function limits the designer to perform multitasking from the controller, so this sketch is implement without utilizing this function).

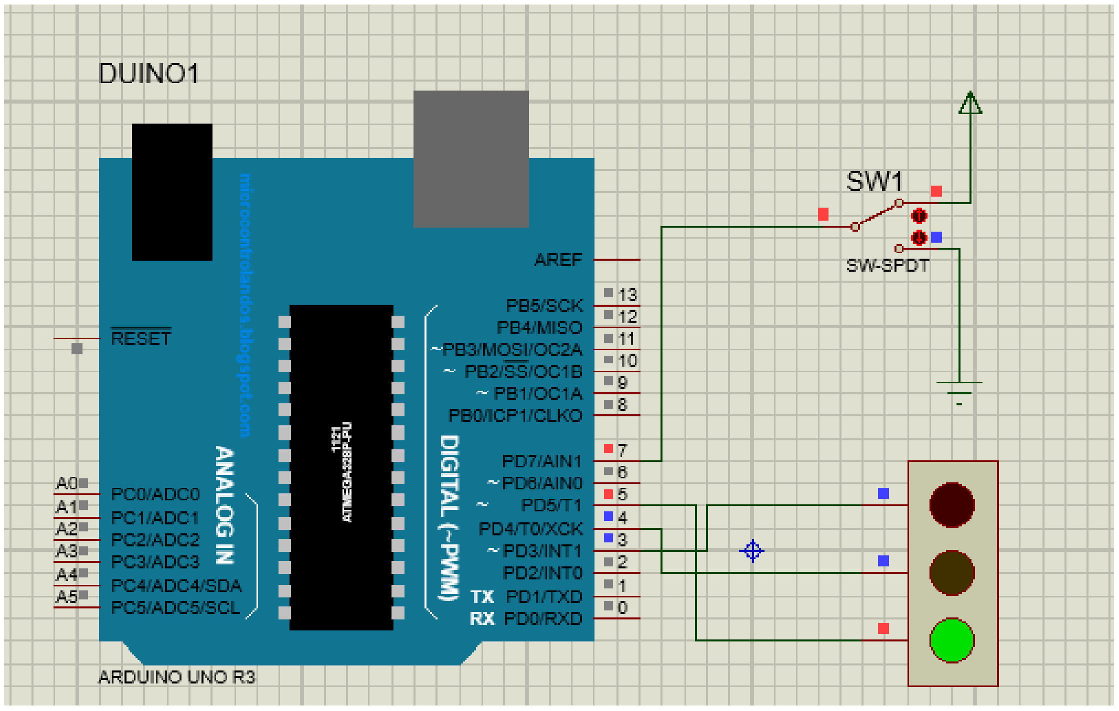
Simulation:

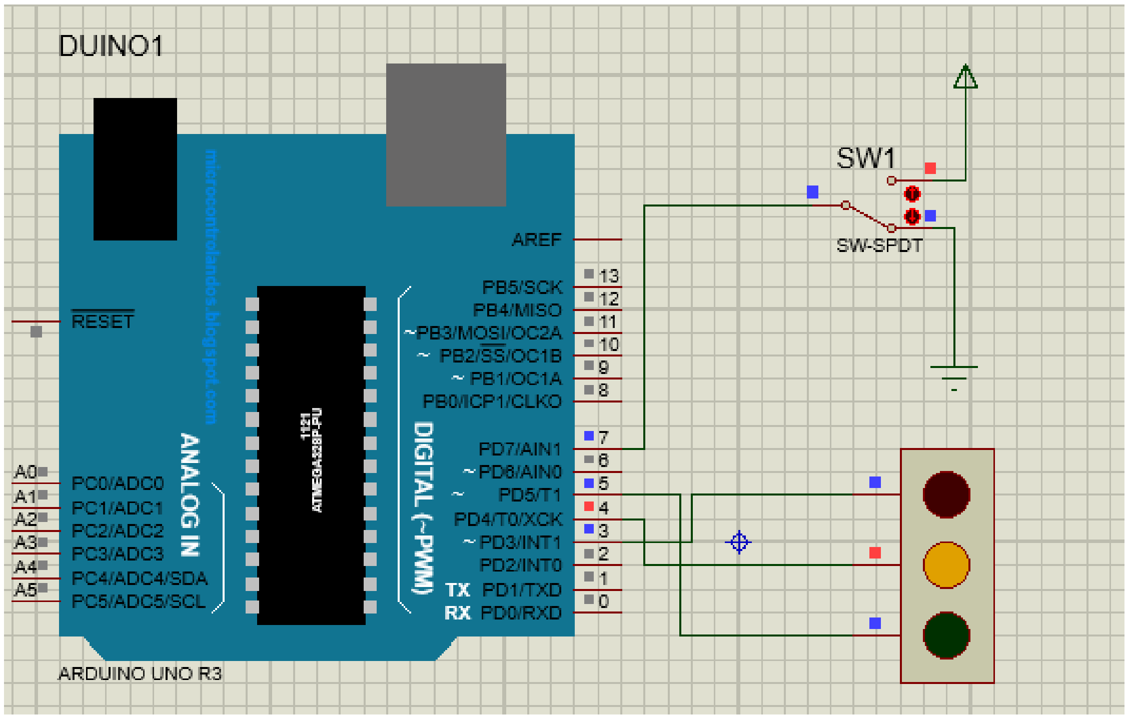


Task 2: Write a sketch to implement the one-way traffic light controller using FSM concepts. The sensor will work to sense the traffic on the road whose output will be the stimulus for the state transition.

Simulation:

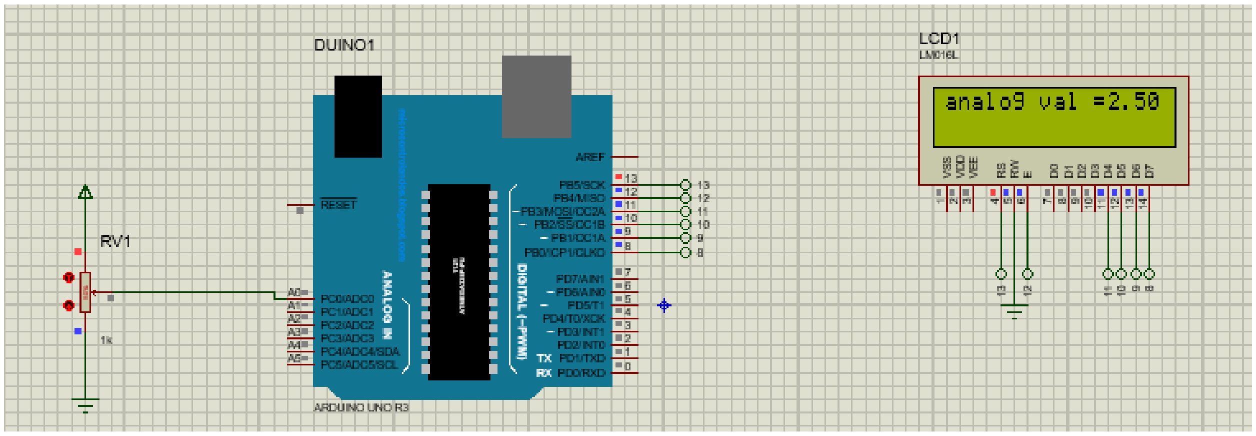





DUINO1
microcontrolandos.blogspot.com
ANALOG IN
ATMEGA328P-PU
DIGITAL (~PWM)
AREF
RESET
PB5/SCK
PB4/MISO
~PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~ PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
13
12
11
10
9
8
7
6
5
4
3
2
1
0
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ARDUINO UNO R3
SW1
SW-SPDT

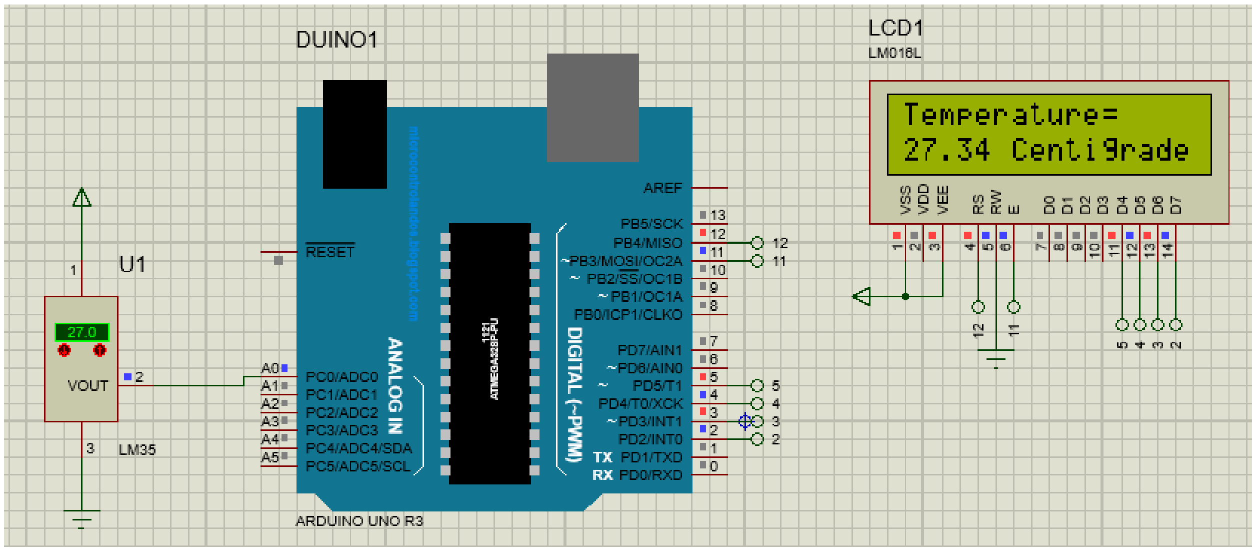
Task 1: Write a program to interface potentiometer with analog pin of Arduino Uno to read analog values and display it on LCD.

Simulation:



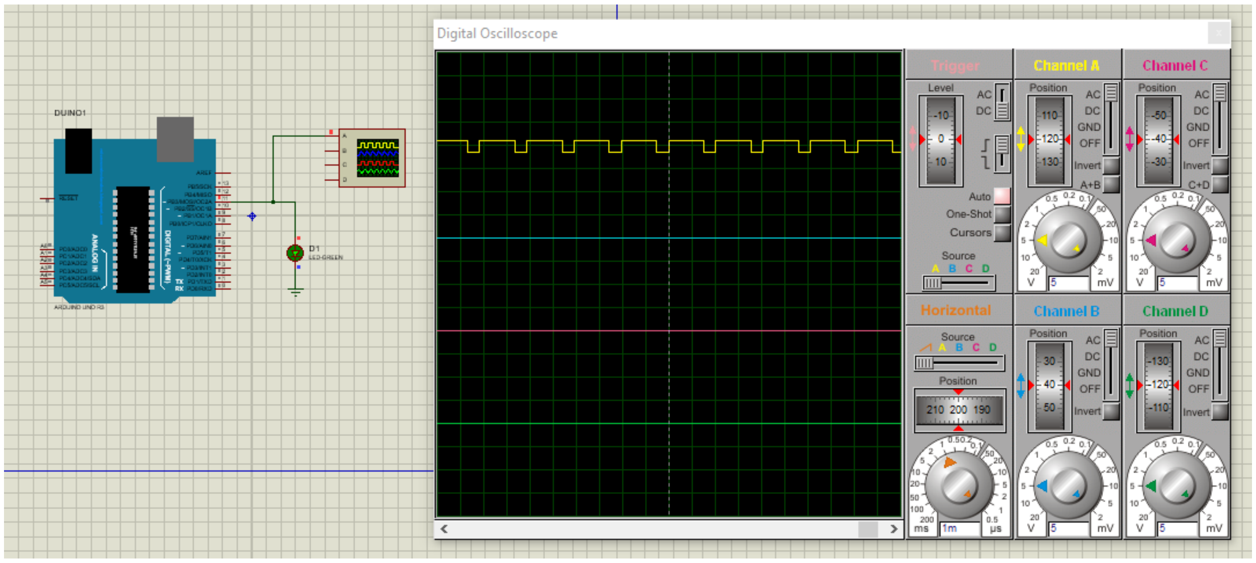
Task 2: Write a sketch to interface Arduino with the Temperature Sensor (LM35). The value of the Temperature should be displayed on the LCD.

Simulation:



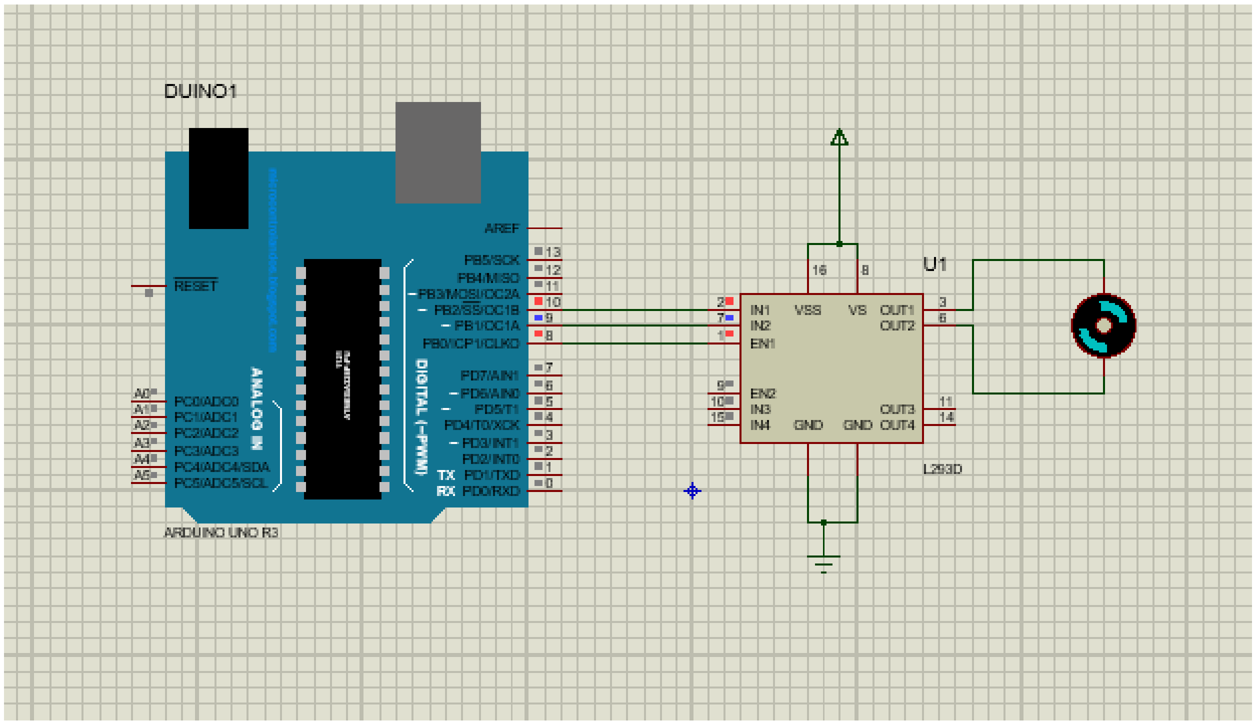
Task 1: Write a program to generate a PWM signal with duty cycles (25%, 50%, 75% & 100%).

Simulation:



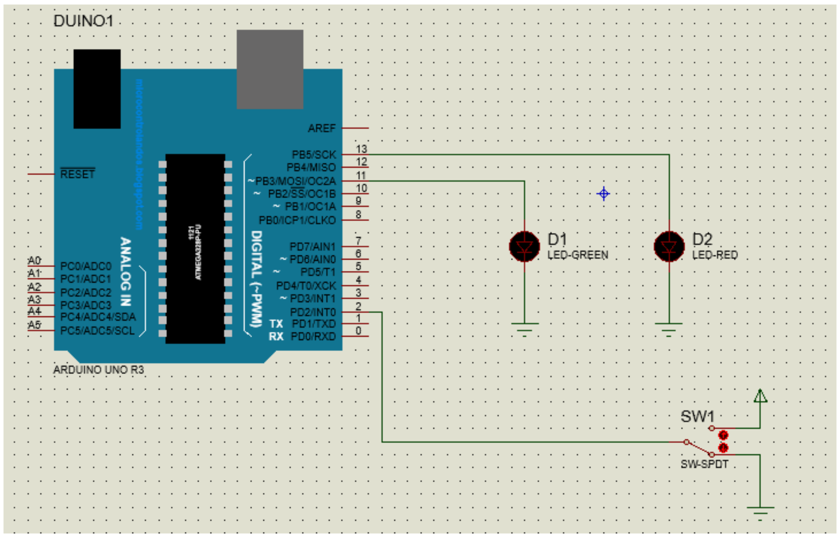
Task 2: Write a program to interface DC motor with Arduino Uno.

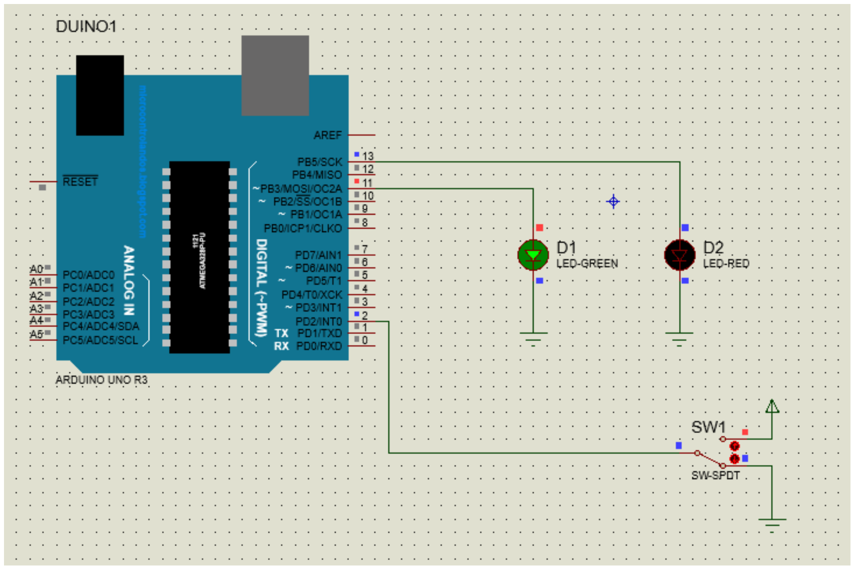
Simulation:

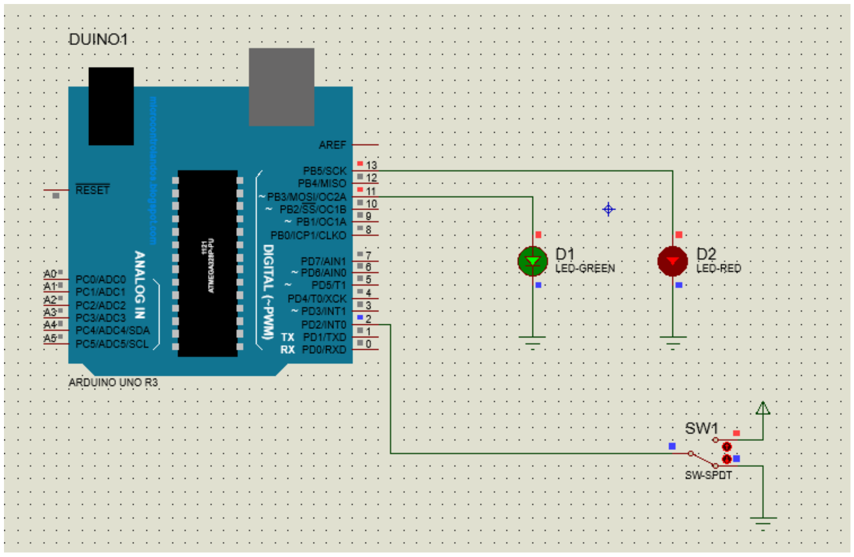
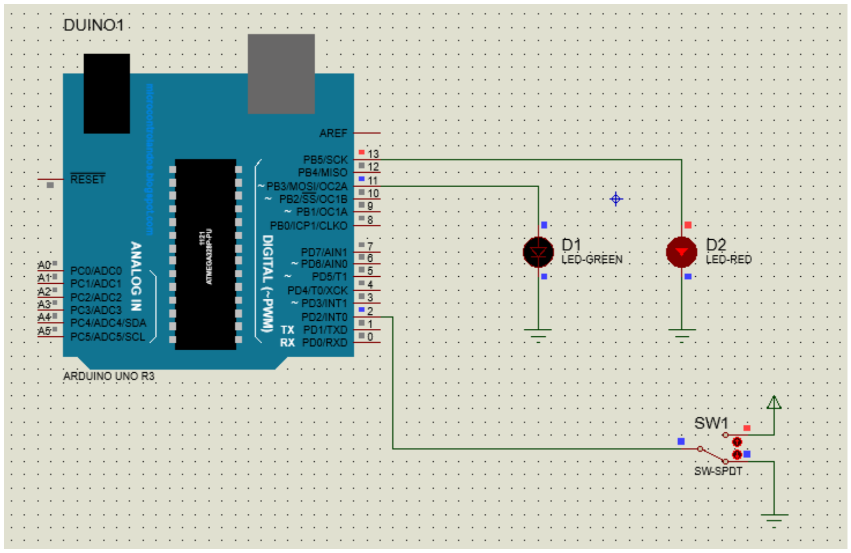


Task 1: In this lab, we will use external interrupt in Arduino UNO.

Simulation:

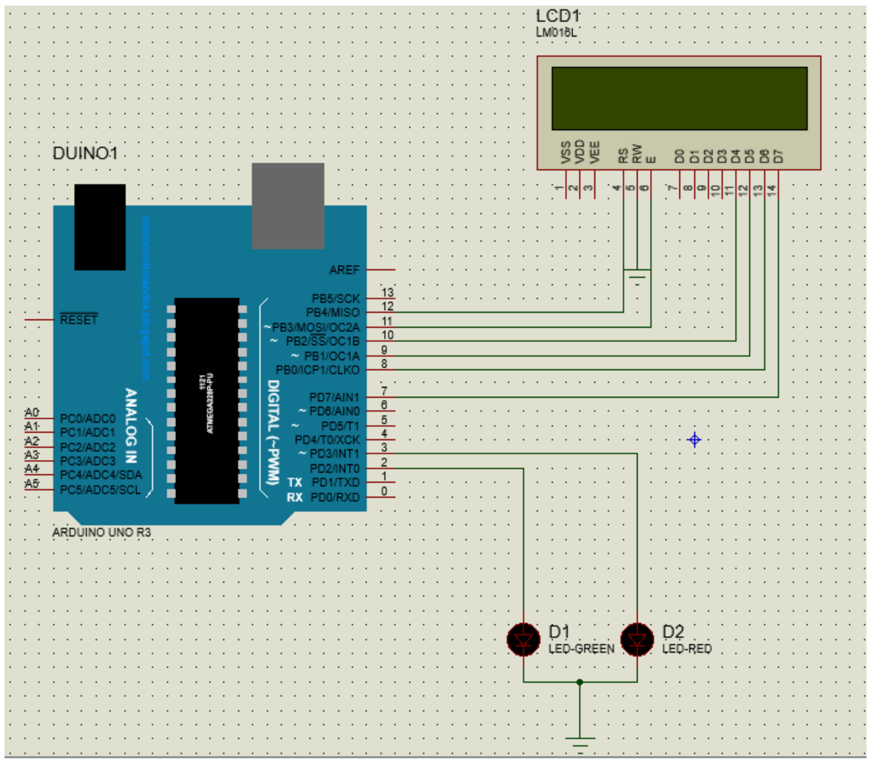


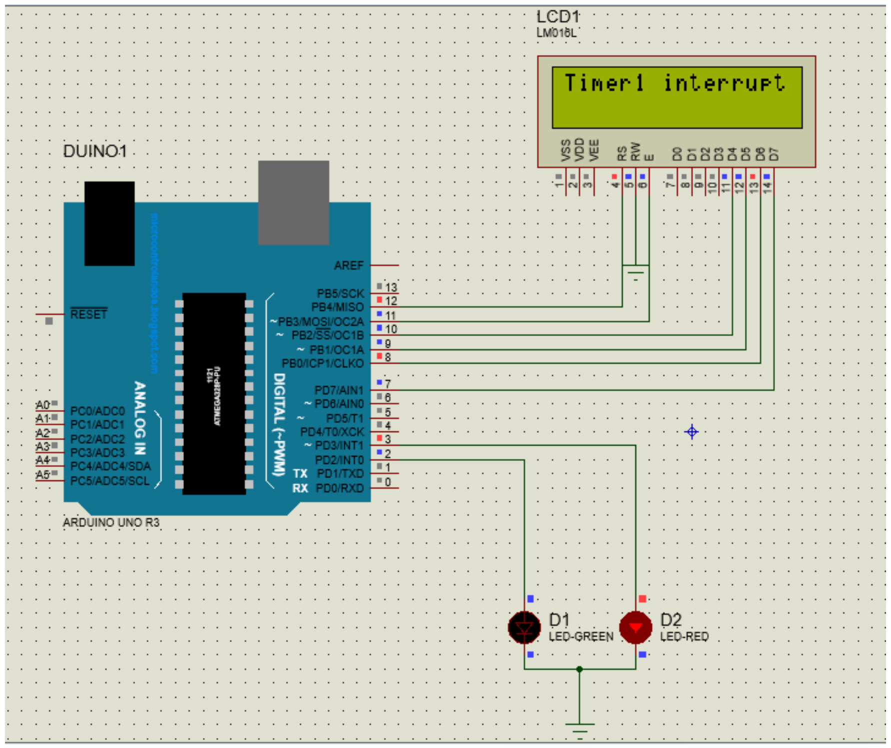


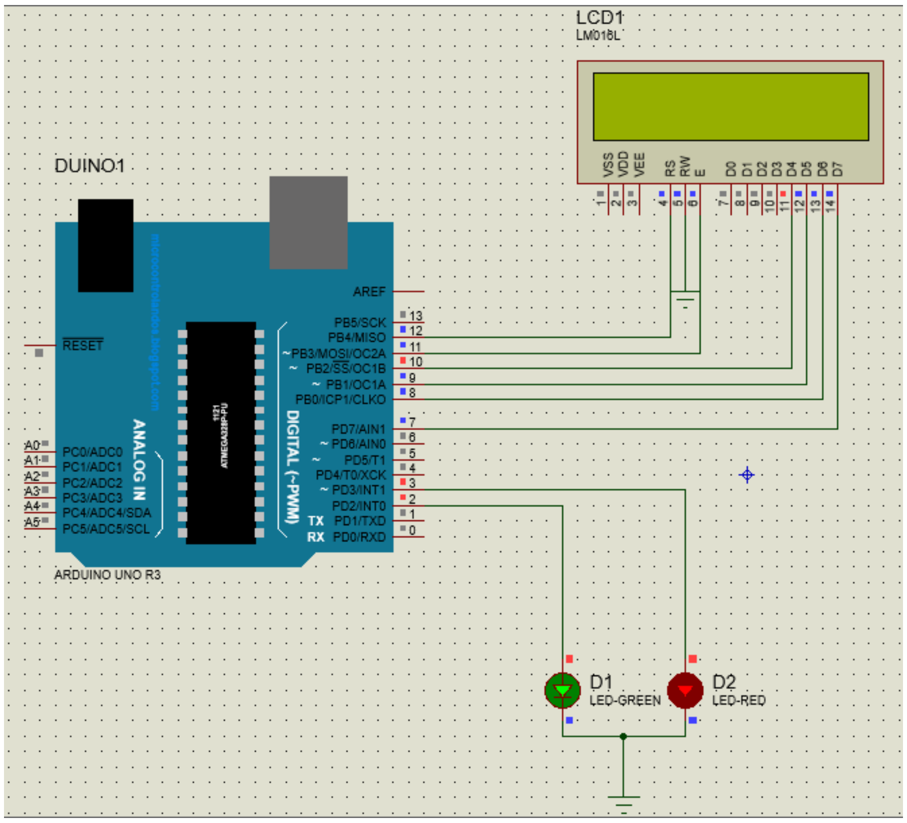

DUINO1
AREF
PB5/SCK
PB4/MISO
~PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~ PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ANALOG IN
DIGITAL (~PWM)
ATMEGA328P-PU
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT

DUINO1
AREF
PB5/SCK
PB4/MISO
~PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~ PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ANALOG IN
DIGITAL (~PWM)
ATMEGA328P-PU
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT

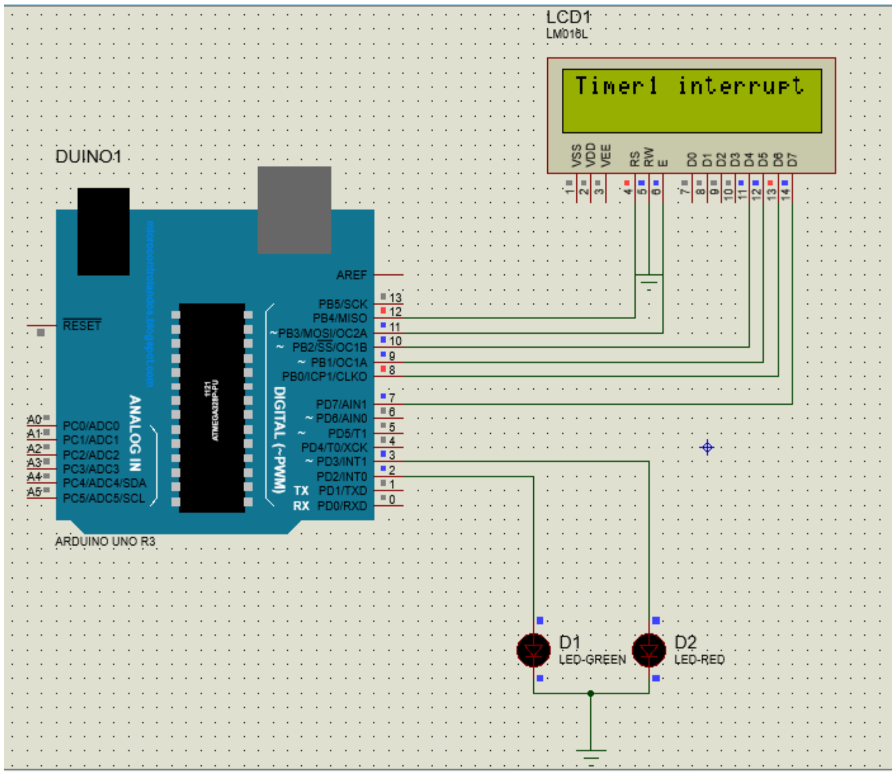
Task 2: Write a program to use timer interrupt in Arduino Uno.

Simulation:





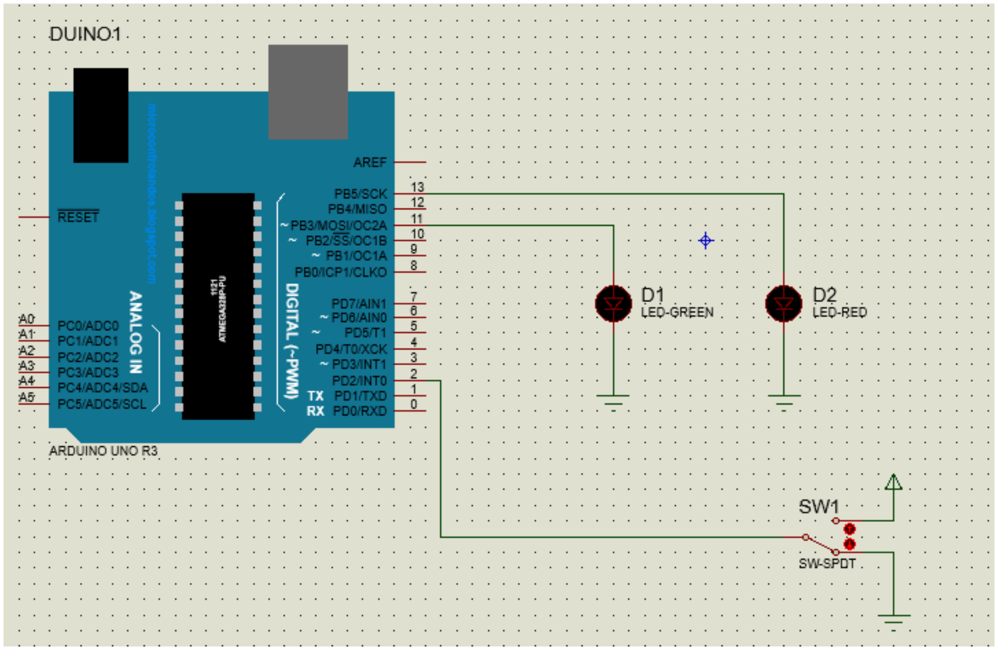


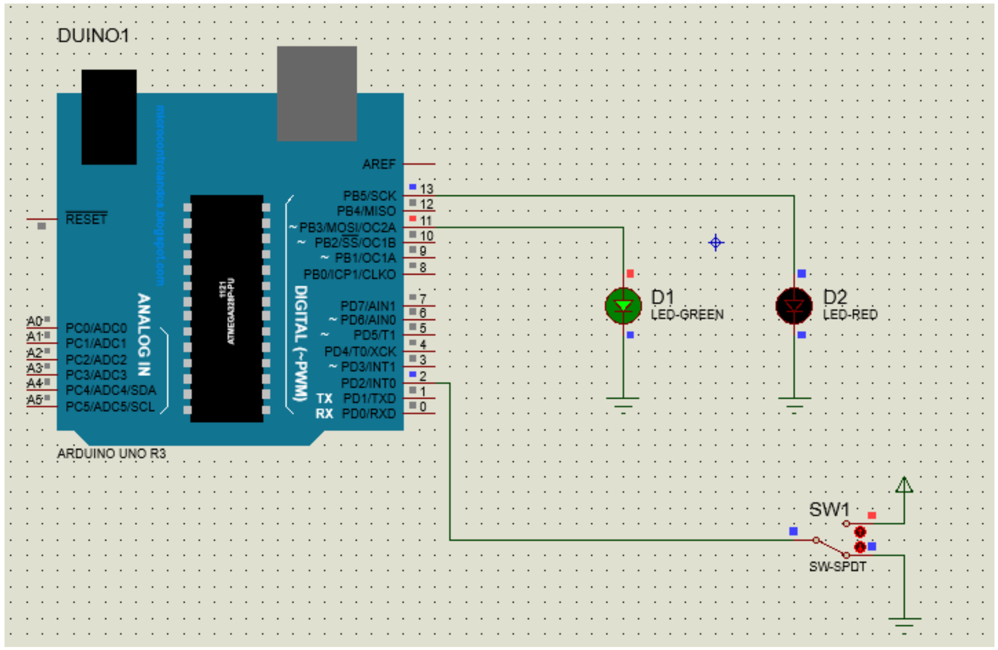

Timer1 interrupt

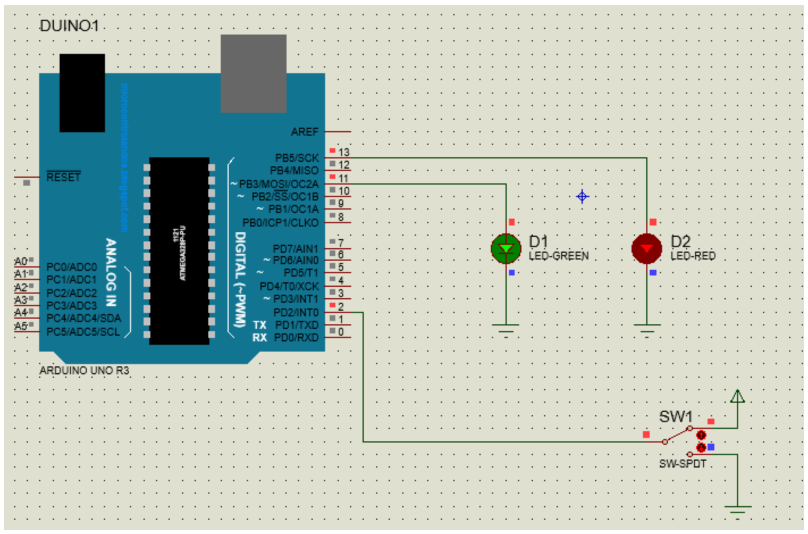
Task 3: Using the concept of interrupts you have learnt, develop a program that uses interrupt.

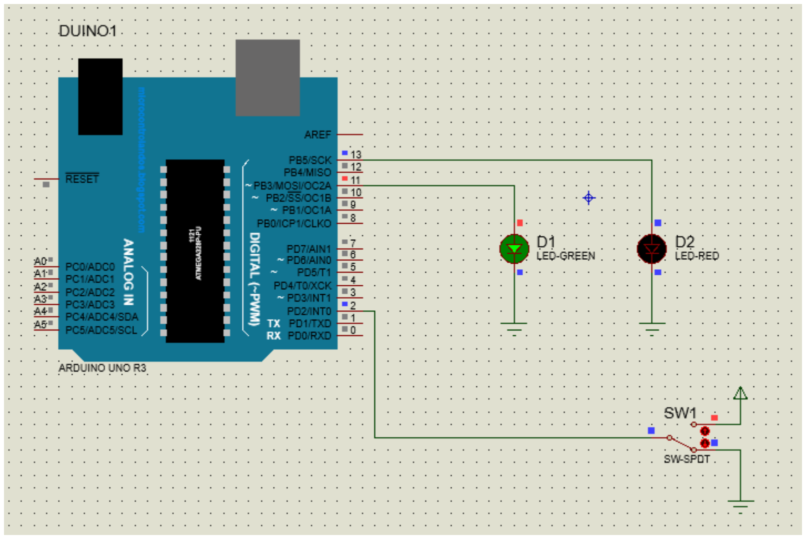
```
int i=0;

int x=0;

void setup() {

 // put your setup code here, to run once:

pinMode(11,OUTPUT);

pinMode(13,OUTPUT);

pinMode(2,INPUT);

attachInterrupt(digitalPinToInterrupt(2),routine,CHANGE);

}


void loop() {


 // put your main code here, to run repeatedly:

digitalWrite(11,HIGH);

delay(1000);

digitalWrite(11,LOW);

delay(1000);

}

void routine()

{

  x= ~i;

  i=x;

   digitalWrite(13,i);

}
```

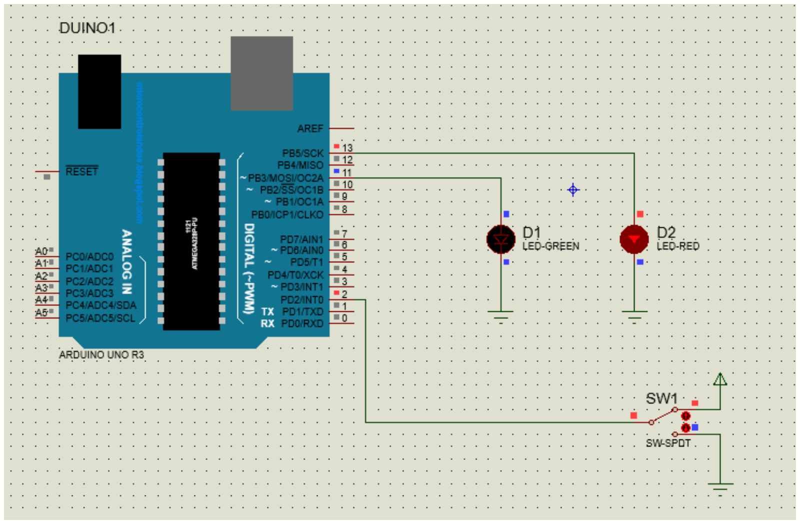
Simulation:


DUINO1
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~ PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
DIGITAL (~PWM)
ANALOG IN
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT


DUINO1
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/SS/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD6/AIN0
~ PD5/T1
PD4/T0/XCK
~ PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD
DIGITAL (~PWM)
ANALOG IN
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT


DUINO1
AREF
PB5/SCK 13
PB4/MISO 12
~PB3/MOSI/OC2A 11
~ PB2/SS/OC1B 10
~ PB1/OC1A 9
PB0/ICP1/CLKO 8
PD7/AIN1 7
~ PD6/AIN0 6
~ PD5/T1 5
PD4/T0/XCK 4
~ PD3/INT1 3
PD2/INT0 2
TX PD1/TXD 1
RX PD0/RXD 0
DIGITAL (~PWM)
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ANALOG IN
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT


DUINO1
AREF
PB5/SCK 13
PB4/MISO 12
~PB3/MOSI/OC2A 11
~ PB2/SS/OC1B 10
~ PB1/OC1A 9
PB0/ICP1/CLKO 8
PD7/AIN1 7
~ PD6/AIN0 6
~ PD5/T1 5
PD4/T0/XCK 4
~ PD3/INT1 3
PD2/INT0 2
TX PD1/TXD 1
RX PD0/RXD 0
DIGITAL (~PWM)
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ANALOG IN
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT


DUINO1
AREF
PB5/SCK 13
PB4/MISO 12
~PB3/MOSI/OC2A 11
~ PB2/SS/OC1B 10
~ PB1/OC1A 9
PB0/ICP1/CLKO 8
PD7/AIN1 7
~ PD6/AIN0 6
~ PD5/T1 5
PD4/T0/XCK 4
~ PD3/INT1 3
PD2/INT0 2
TX PD1/TXD 1
RX PD0/RXD 0
DIGITAL (~PWM)
RESET
A0 PC0/ADC0
A1 PC1/ADC1
A2 PC2/ADC2
A3 PC3/ADC3
A4 PC4/ADC4/SDA
A5 PC5/ADC5/SCL
ANALOG IN
ARDUINO UNO R3
D1
LED-GREEN
D2
LED-RED
SW1
SW-SPDT

Task 1: Using the Concept of SPI communication, write a program to control a LED on slave arduino by a SPDT switch on master arduino. Attach proteus simulation results and arduino code.

Code Master:

```
#include <SPI.h>
int pin = 9;
void setup() {
  // put your setup code here, to run once:
digitalWrite(SS,HIGH);
SPI.begin();
SPI.setClockDivider(SPI_CLOCK_DIV2);
pinMode(pin,INPUT);
}
int state;
int oldstate = 0;
void loop() {
  // put your main code here, to run repeatedly:
int sw= digitalRead(pin);
 if(sw==HIGH)
 {
int c;
state =! oldstate;
digitalWrite(SS,LOW);
c=state;
SPI.transfer(c);
digitalWrite(SS,HIGH);
oldstate = state;
delay(500);
 }
}
```

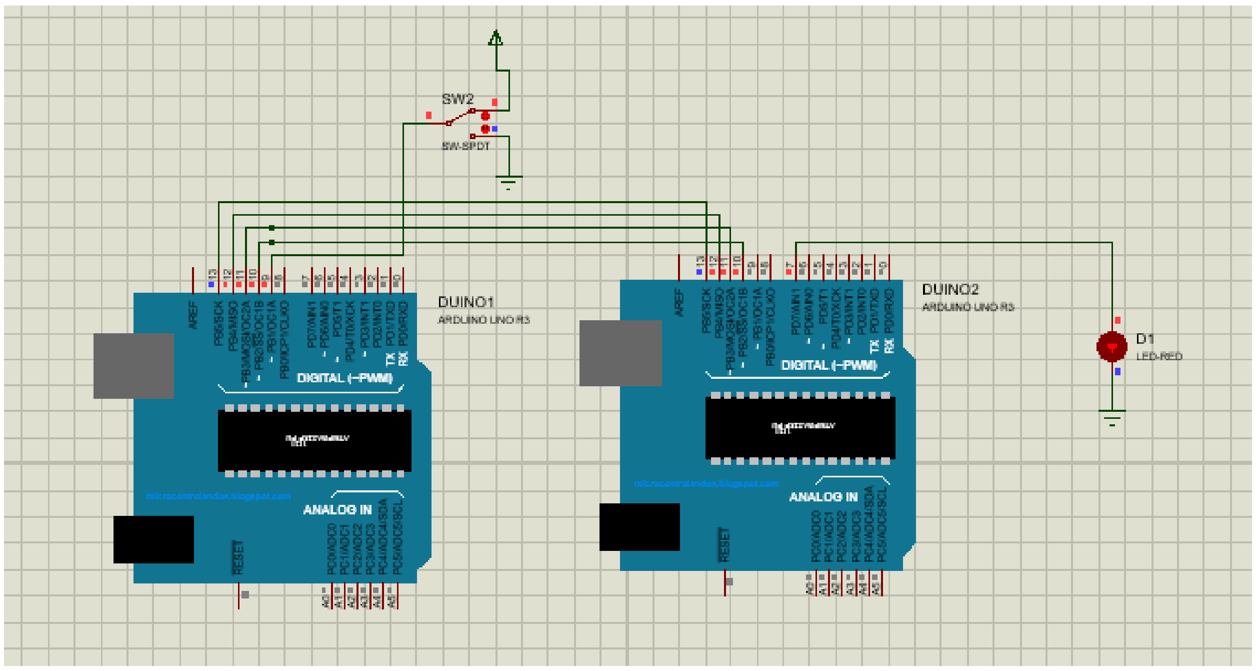
Code Slave:

```
#include <SPI.h>
int buff;
volatile boolean process;
//SLAVE
void setup() {
  // put your setup code here, to run once:
pinMode(MISO,OUTPUT);
SPCR = _BV(SPE);
process = false;
SPI.attachInterrupt();
pinMode(7,OUTPUT);
}
ISR(SPI_STC_vect)
```

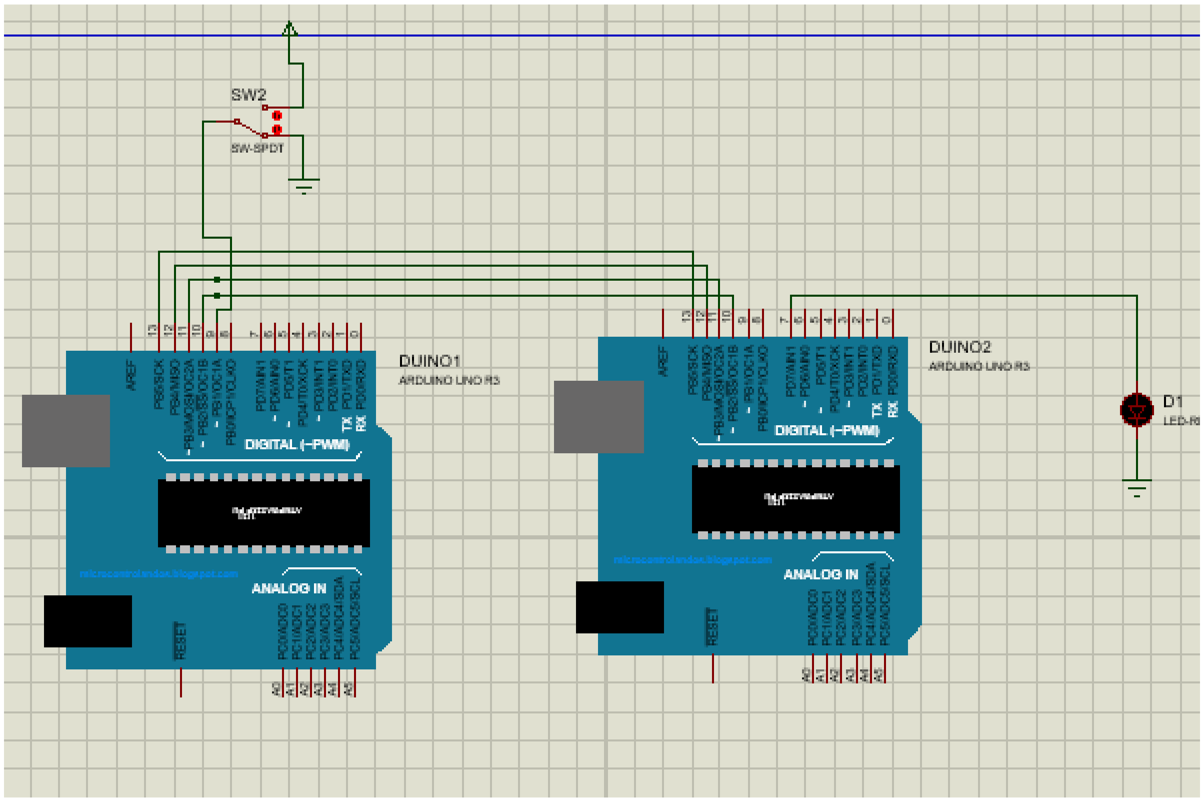
```
{
  int c = SPDR;
  buff = c;
  process =true;
}

void loop() {
  // put your main code here, to run repeatedly:

if(process)
{
  process=false;
  digitalWrite(7,buff);
}
}
```

OUTPUT:

Switch on:

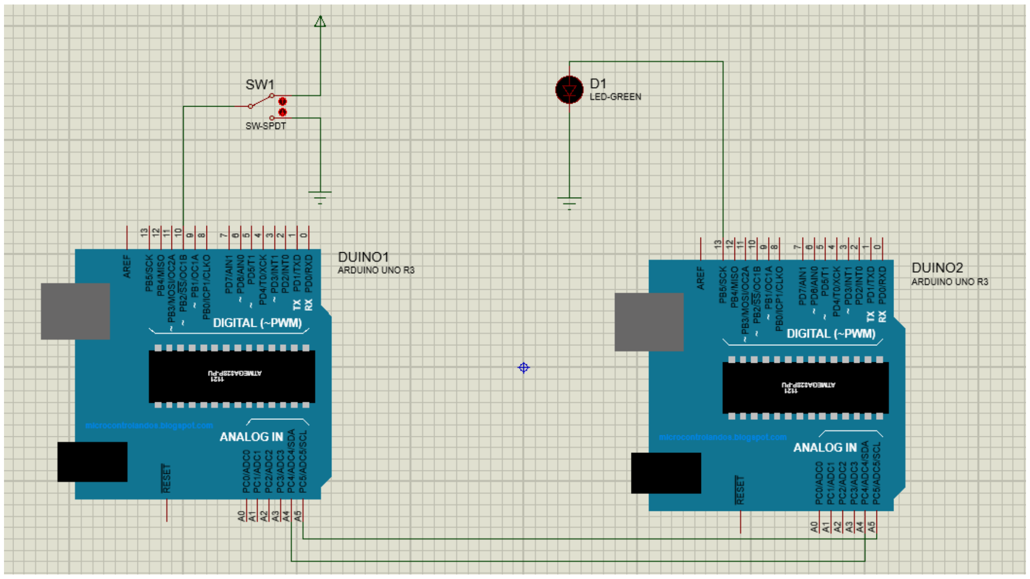


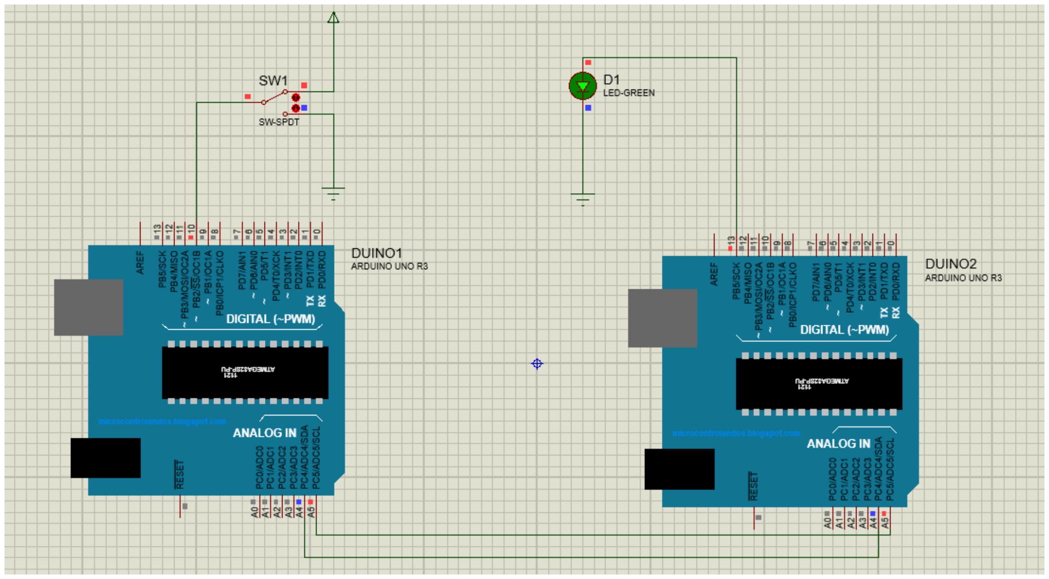
Switch off:


SW2
SW-SPDT
DUINO1
ARDUINO UNO R3
DIGITAL (~PWM)
ANALOG IN
RESET
DUINO2
ARDUINO UNO R3
DIGITAL (~PWM)
ANALOG IN
RESET
D1
LED-RE

Task 1: In this task we will control the LED on Slave Arduino using SPDT switch on Master Arduino through I2C communication.

Simulation:





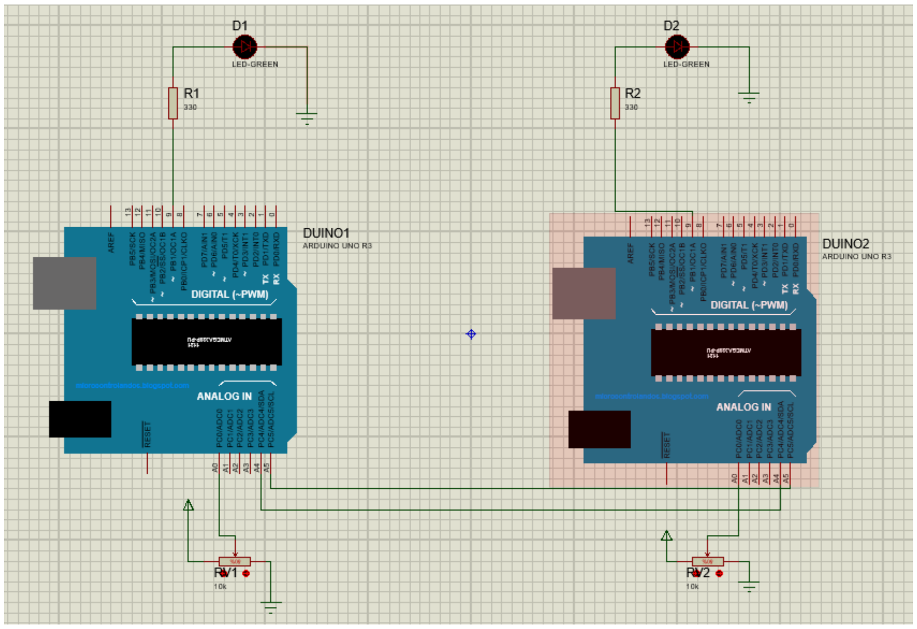
Task 2: Develop a Program which provide the use of Inter integrated 12C Interface programming.

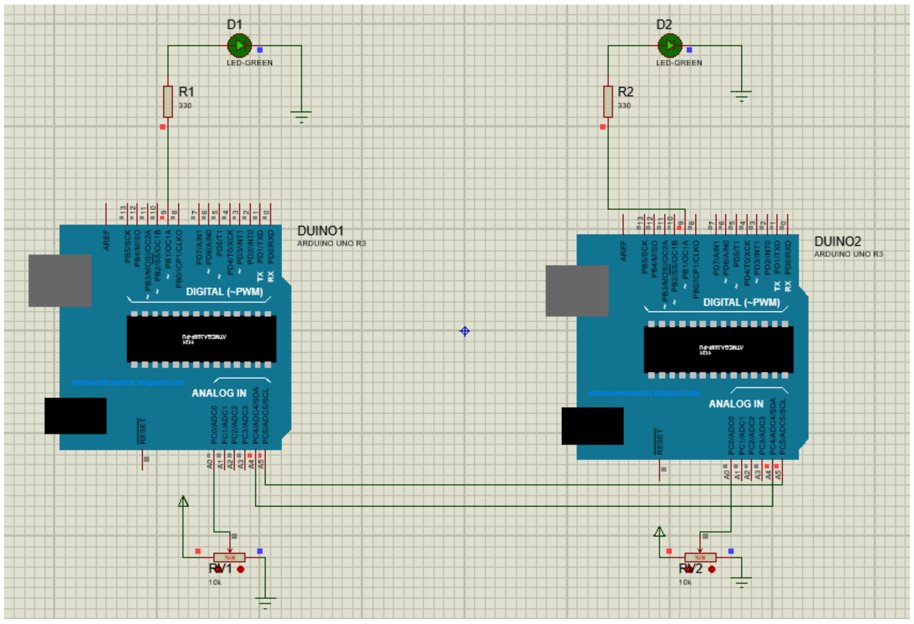
Master:

```
#include <Wire.h>

#define ledPin 9

byte rcvData;

int potValue;

void setup()

{

  Wire.begin();

  rcvData = 255;

  pinMode(ledPin, OUTPUT);

}

void loop()

{

  potValue = analogRead(A0);

  potValue = map(potValue, 0, 1023, 0, 255);


  Wire.beginTransmission(0x14);

  Wire.write(potValue);

  Wire.endTransmission();

  Wire.requestFrom(0x14, 1);

  if(Wire.available())

  {

    rcvData = Wire.read();

  }

  analogWrite(ledPin, rcvData);

}
```

Slave:

```
#include <Wire.h>

#define ledPin 9

byte rcvData;

int potValue;

void setup()

{

  Wire.begin(0x14);

  /*Event Handlers*/

  Wire.onReceive(DataReceive);

  Wire.onRequest(DataRequest);

  rcvData = 255;

  pinMode(ledPin, OUTPUT);

}

void loop()

{

  potValue = analogRead(A0);

  potValue = map(potValue, 0, 1023, 0, 255);

  analogWrite(ledPin, rcvData);

}

void DataReceive(int numBytes)

{

  while(Wire.available())

  {

    rcvData = Wire.read();

  }

}

void DataRequest()

{
  Wire.write(potValue);

}
```





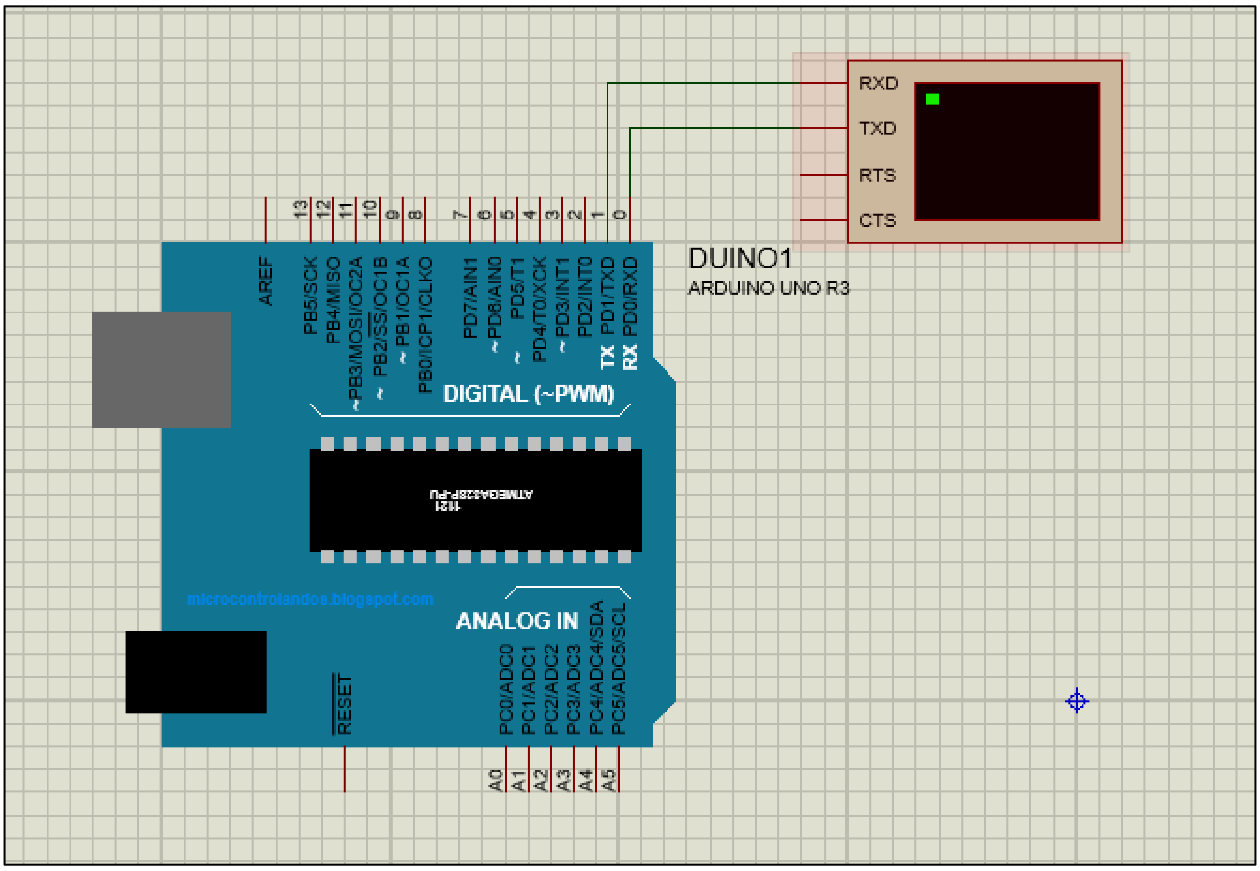
Lab Tasks/Practical Work

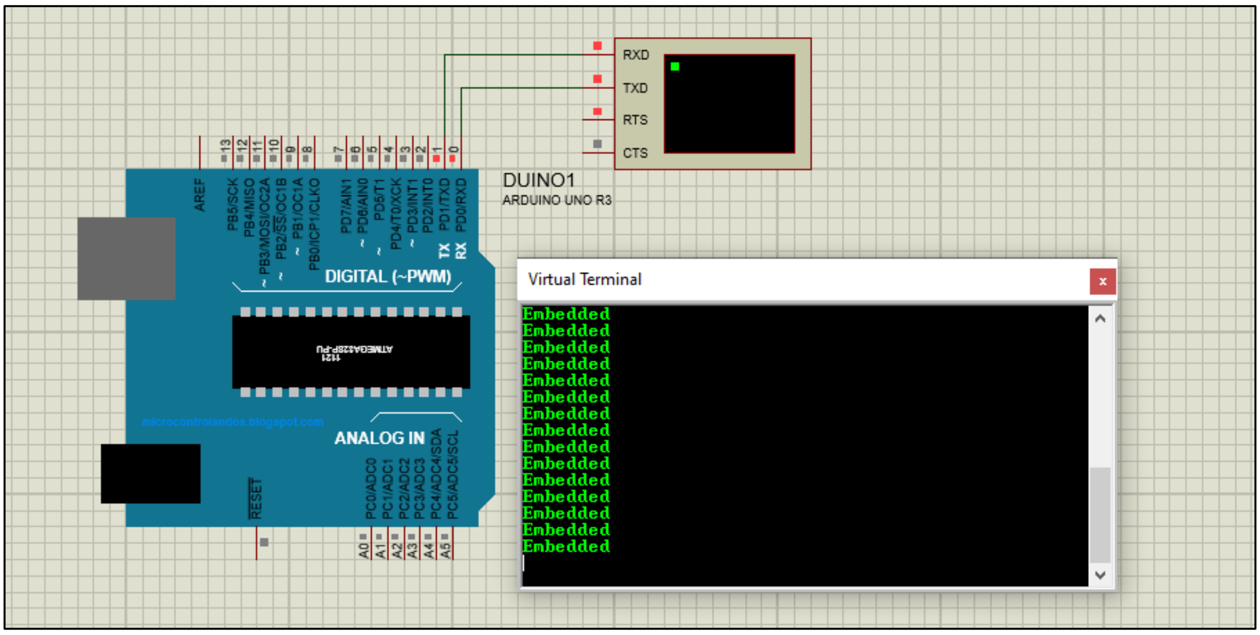
Task1:

In this task, we will write a program to print a string on serial monitor.

Solution:

Output:





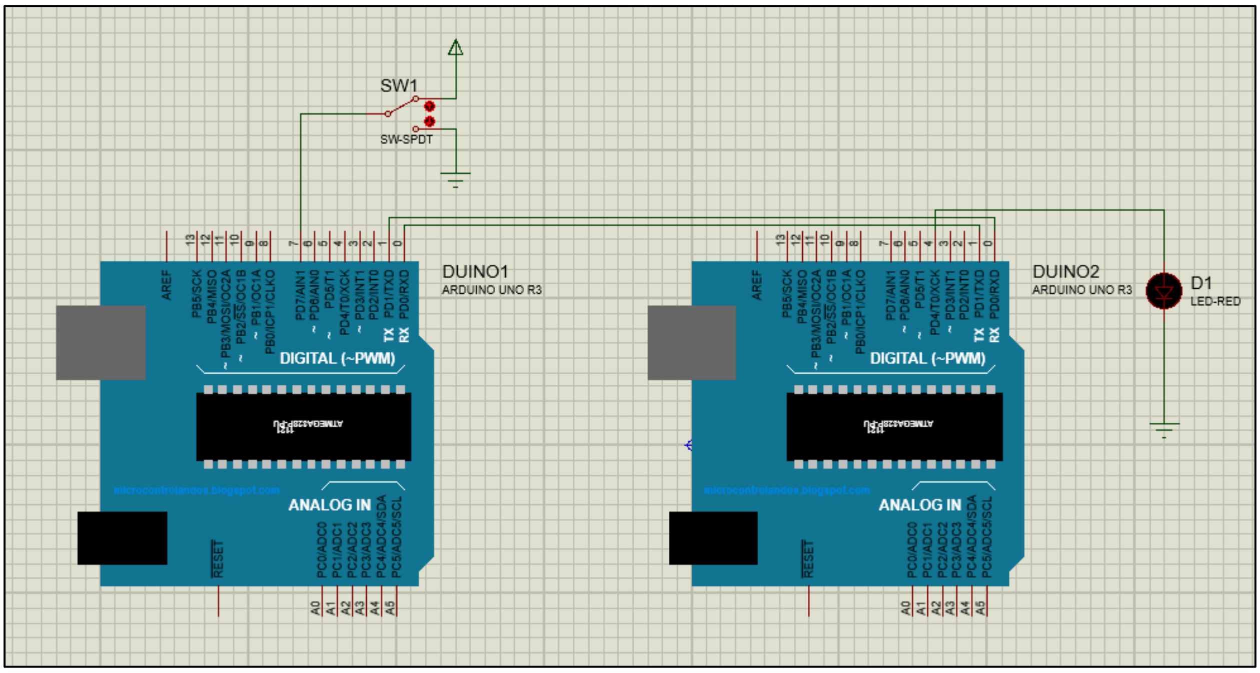
```
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
Embedded
```

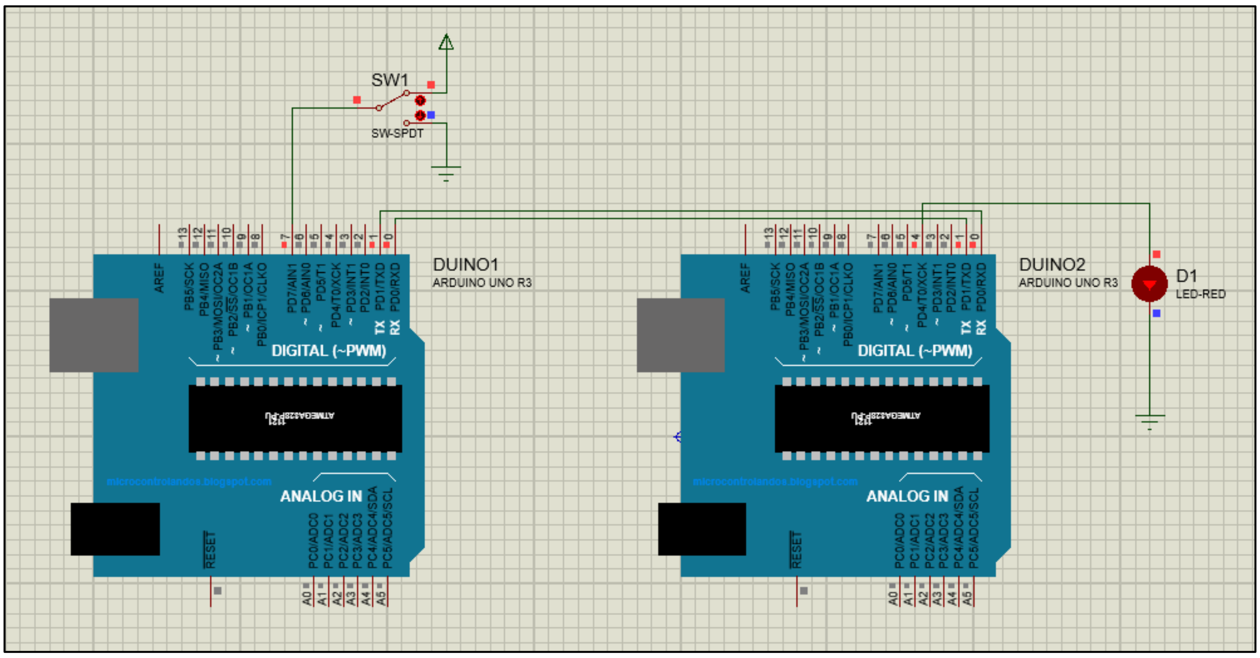
Task 2:

In this task we will control the LED on “Arduino 2” using SPDT switch on “Arduino 1” through UART communication.

Solution:

Output:





Task 3: Develop a Program which provide serial usart interfacing programming using Arduino UNO board.

Solution:

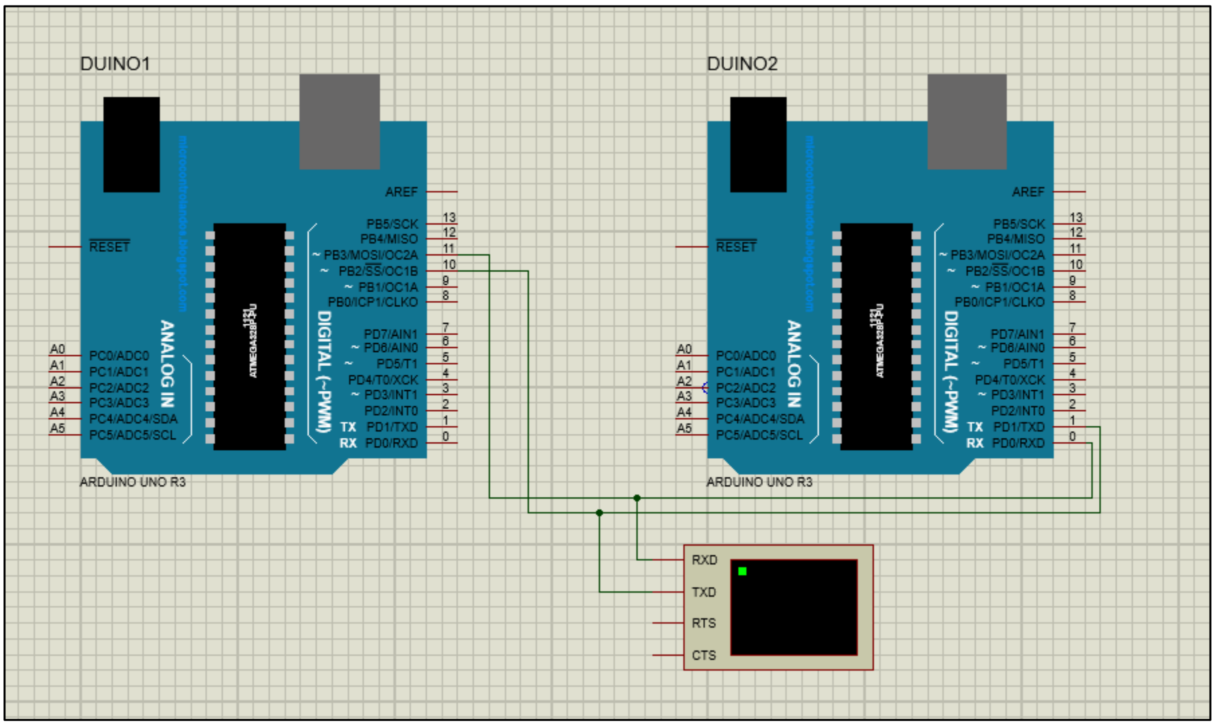
Master:

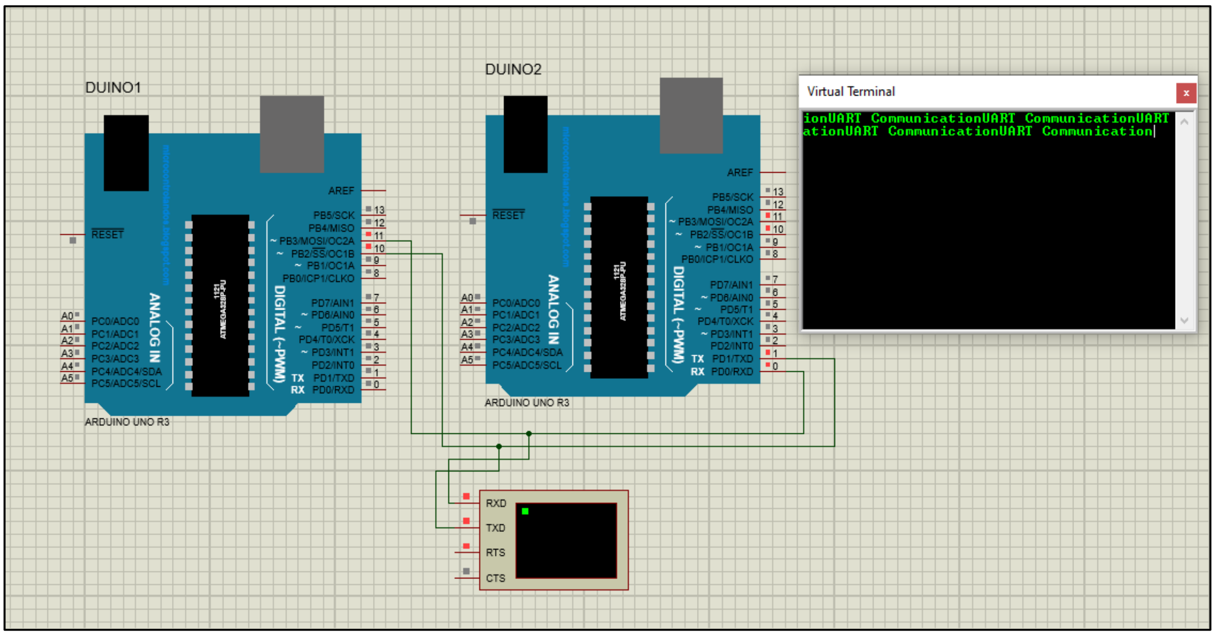
```
#include <SoftwareSerial.h>
SoftwareSerial softSerial(10, 11);


void setup()
{
  softSerial.begin(9600);
}
void loop()
{
  softSerial.write("UART Communication");
  delay (100);
}
```

Slave:

```
#include <SoftwareSerial.h>
SoftwareSerial softSerial(10, 11);


char ip;
void setup()
{
  softSerial.begin(9600);
}
void loop()
{
  if (softSerial.available())
  {
    ip=softSerial.read();
    softSerial.print(ip);
  }
}
```

Output:

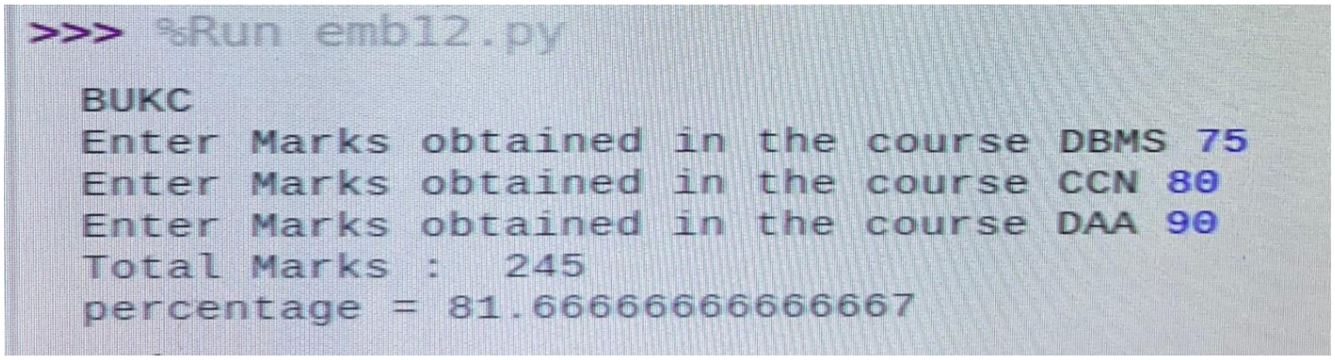




```
ionUART CommunicationUART CommunicationUART
ationUART CommunicationUART Communication
```

Task 1: In this task we will write a simple Python script in IDLE to calculate the Percentage by taking the Marks of the courses as input.

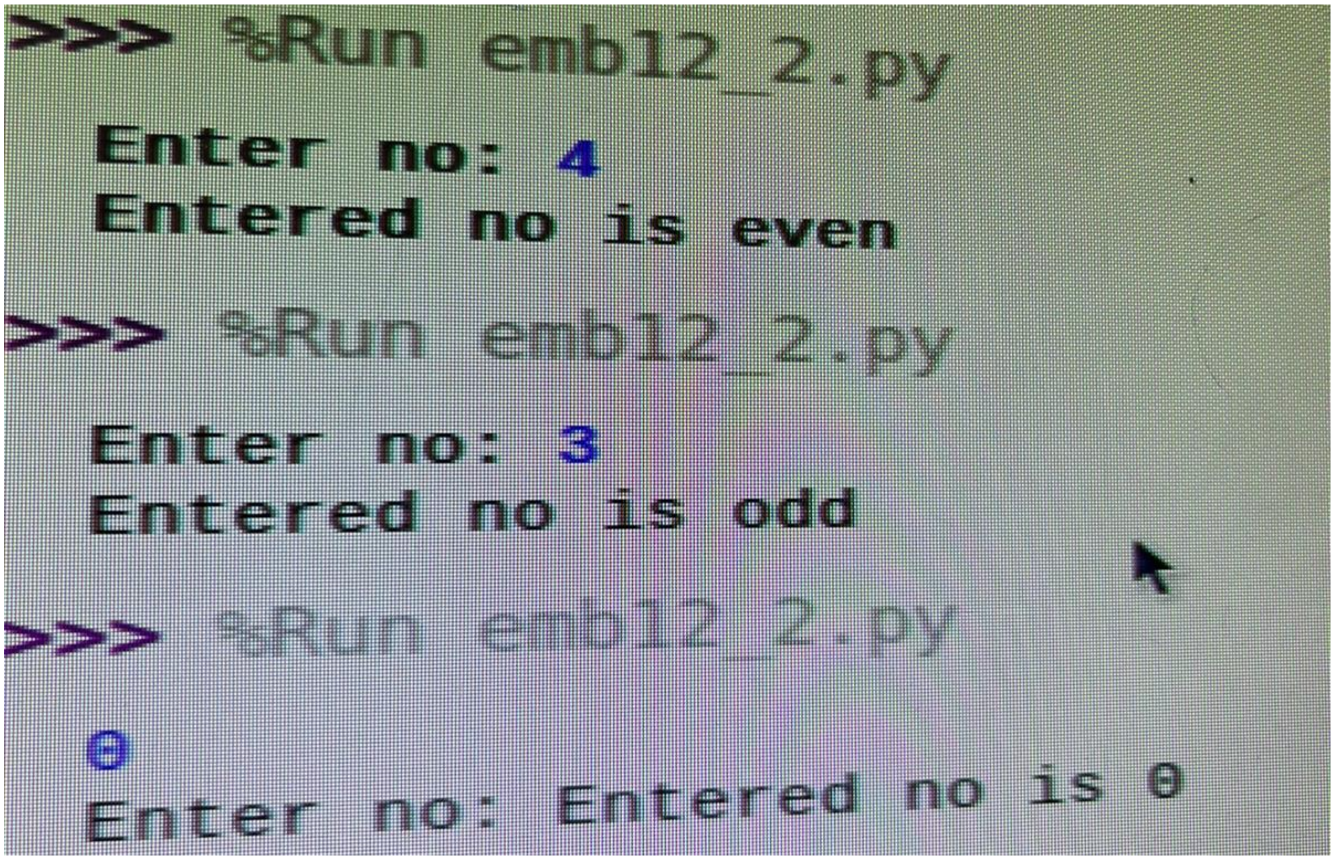
Simulation:



```
>>> %Run emb12.py

  BUKC
  Enter Marks obtained in the course DBMS 75
  Enter Marks obtained in the course CCN 80
  Enter Marks obtained in the course DAA 90
  Total Marks :  245
  percentage = 81.66666666666667
```

Task 2: In this task we will write a simple Python script to check that the input number is either even or odd.

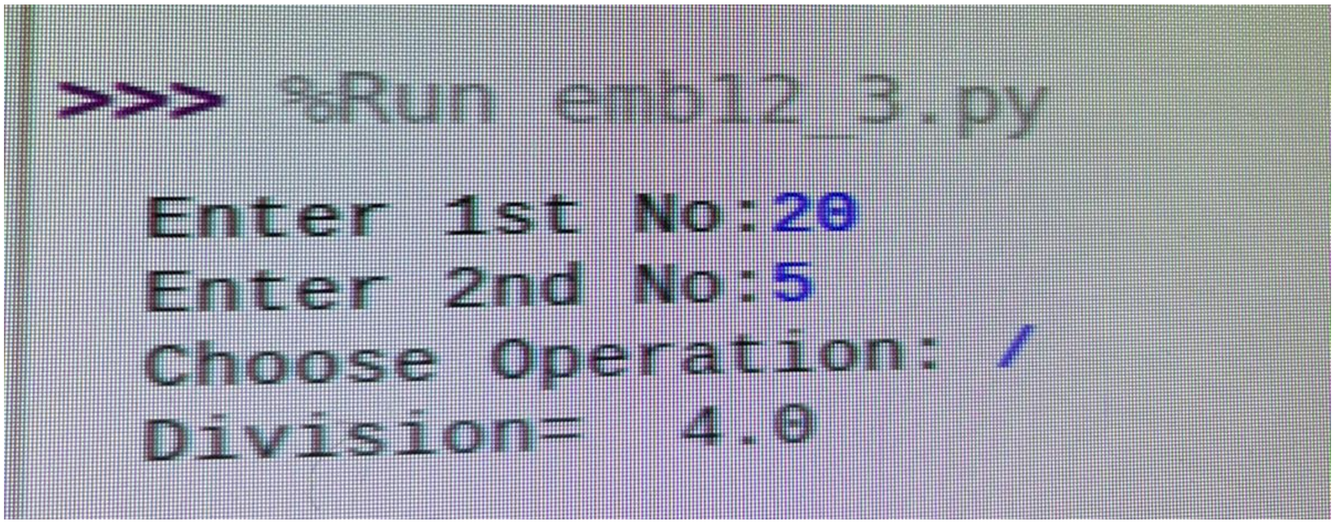
Simulation:



```
>>> %Run emb12_2.py

  Enter no: 4
  Entered no is even

>>> %Run emb12_2.py

  Enter no: 3
  Entered no is odd

>>> %Run emb12_2.py

  0
  Enter no: Entered no is 0
```

Task 3: Write Python Script for Calculator which can perform simple operations of Addition, Subtraction, Multiplication and Division.

Simulation:



```
>>> %Run emb12_3.py

  Enter 1st No:20
  Enter 2nd No:5
  Choose Operation: /
  Division=  4.0
```

LAB # 13

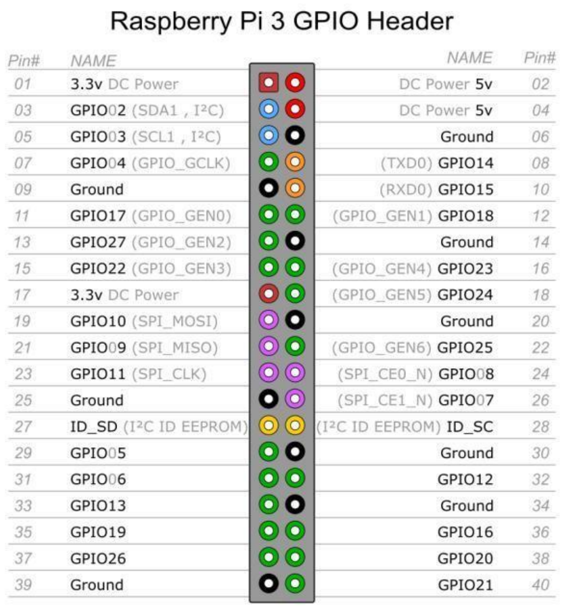
Raspberry PI Hardware Applications

Introduction

Raspberry – Pi GPIO Controller:

- The R-Pi GPIO connector actually has a number of different types of connection on them. There are:
- True GPIO (General Purpose Input Output) pins that you can use to turn LEDs on and off etc.
- I2C interface pins that allow you to connect hardware modules with just two control pins
- SPI interface with SPI devices, a similar concept to I2C but a different standard
- Serial Rx and Tx pins for communication with serial peripherals
- In addition, some of the pins can be used for PWM (pulse Width Modulation) for power control and another type of pulse generation for controlling servo motors called PPM

(Pulse Position Modulation). Fig. 9.1: GPIO Inputs



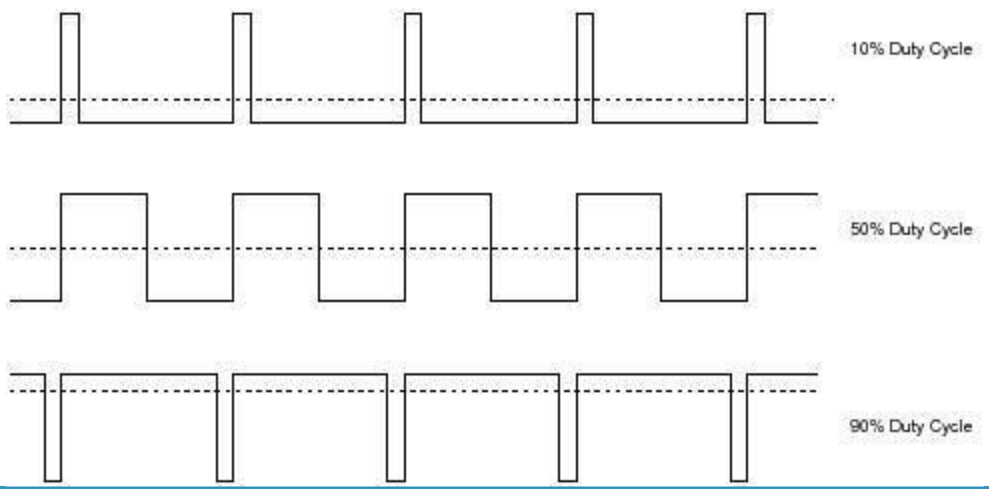
Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

PWM:

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a microprocessor's digital outputs.

Time



Boxing

Activity Name	Activity Time	Total Time
Login Systems + Setting up Raspberry PI Environment	3 mints + 5 mints	8 mints
Walk through Theory & Tasks	60 mints	60 mints
Implement Tasks	80 mints	80 mints
Evaluation Time	30 mints	30 mints

Total Duration	178 mints

Objectives

1. Design and Interface Raspberry-PI Hardware Applications.
2. Design Traffic Lights and Pulse Width Modulation (PWM) circuits.

Lab Tasks/Practical Work

1. In this task we will write a simple Python script in LXTerminal to glow LED using GPIO 7 of Raspberry-Pi.

```
import RPi.GPIO as GPIO
GPIO.setmode (GPIO.BOARD)
GPIO.setup (7, GPIO.OUT)
GPIO.output (7, True)
```

2. In this task we will write a simple Python script in LXTerminal to blink single LED using a GPIO 7 of Raspberry-Pi.

```
 import RPi.GPIO as GPIO
import time
GPIO.setmode (GPIO.BOARD)
GPIO.setup    (7,    GPIO.OUT)
While True:
    GPIO.output (7, True)
    time.sleep (0.5)
    GPIO.output (7, False)
    time.sleep (0.5)
```

3. In this task we will write a simple Python script to control LED using GPIOs as PWM application.

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT) pwm_led
= GPIO.PWM(7, 500)
pwm_led.start(100) while
True:
    duty_s = input("Enter Brightness (0 to 100):")
    duty = int(duty_s)
    pwm_led.ChangeDutyCycle(duty)
```

4. In this task we will write a simple Python script for Traffic control signals using various GPIOs.

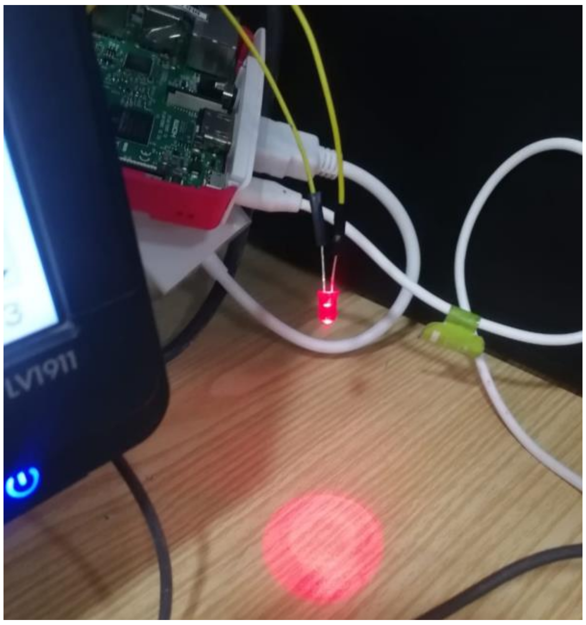
```
import RPi.GPIO as GPIO
 import time GPIO.cleanup()



GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(3,GPIO.OUT)
GPIO.setup(5,GPIO.OUT)
GPIO.setup(7,GPIO.OUT)
GPIO.setwarnings(False) while
True:
         GPIO.output(7,GPIO.HIGH)
time.sleep(5)
         GPIO.output(5,GPIO.HIGH)
         time.sleep(2)
         GPIO.output(7,GPIO.LOW)
         GPIO.output(5,GPIO.LOW)
         GPIO.output(3,GPIO.HIGH)
         time.sleep(10)
         GPIO.output(3,GPIO.LOW)
         GPIO.output(5,GPIO.HIGH)
```

4. Write a simple Python script to drive dc motors in both directions.

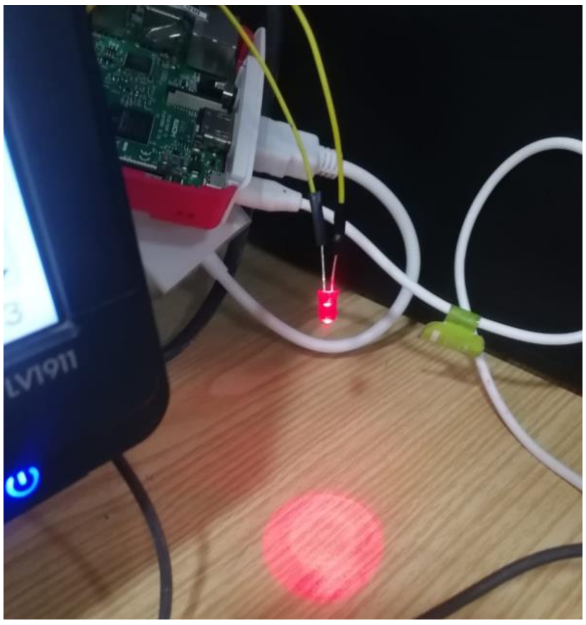
Task 1: In this task we will write a simple Python script in LXTerminal to glow LED using GPIO 7 of Raspberry-Pi.

Simulation:



Task 2: In this task we will write a simple Python script in LXTerminal to blink single LED using a GPIO 7 of Raspberry-Pi.

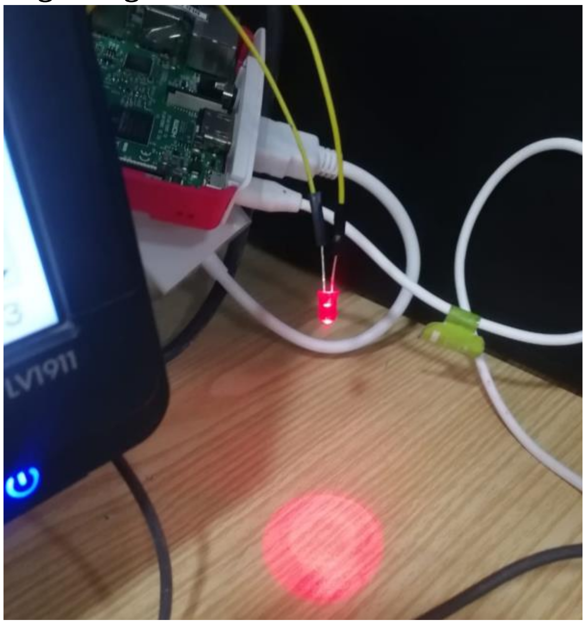
Simulation:



Task 3: In this task we will write a simple Python script to control LED using GPIOs as PWM application.

Simulation:

High Brightness:



Low Brightness: