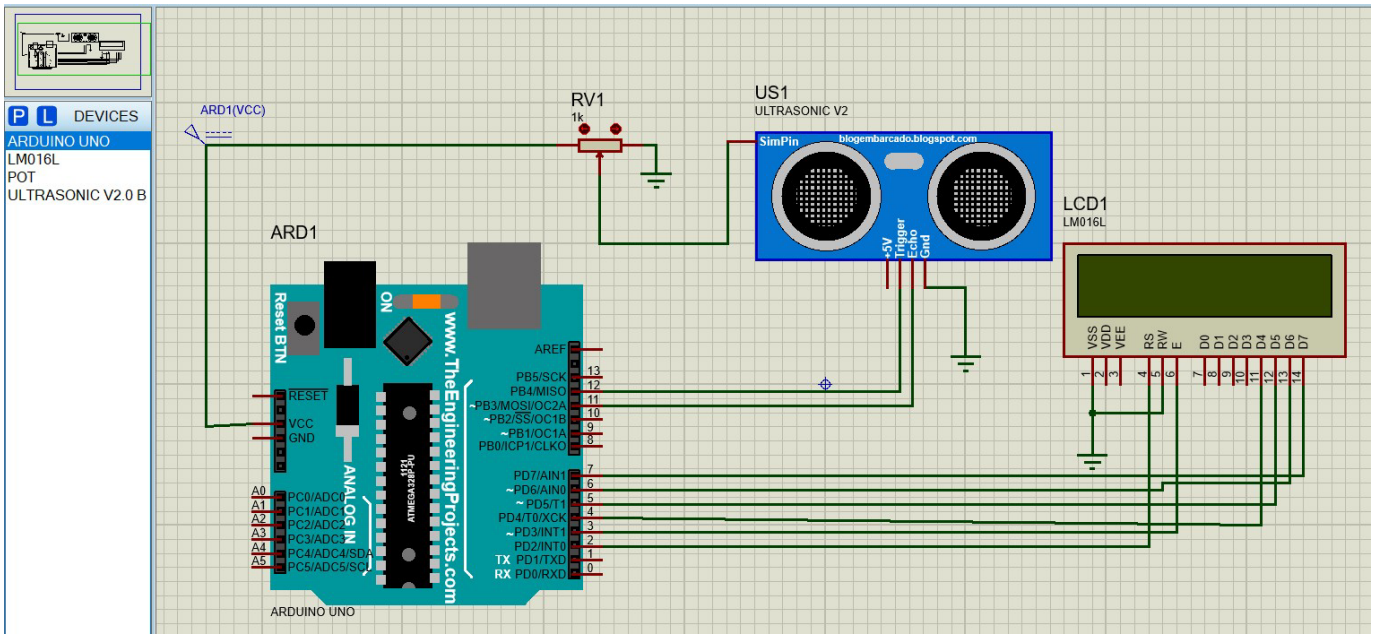
1) To write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.

1.Schematic:



2.Program: distance_measure.ino

```
// include the library code:
#include <LiquidCrystal.h> //library for LCD

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

// defines pins numbers
const int trigPin = 2;
const int echoPin = 3;

// defines
variables long
duration; int
distance;
```

```
void setup()
{
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  lcd.begin(20, 4); // set up the LCD's number of columns and
rows:   lcd.setCursor(0,0); // set the cursor position:
  lcd.print("  THE BRIGHT LIGHT    ");
lcd.setCursor(0,1);   lcd.print("DISTENCE
MEASUREMENT            ");
}
void loop()
{
  // Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro
seconds   digitalWrite(trigPin, HIGH);
delayMicroseconds(10);   digitalWrite(trigPin,
LOW);

  // Reads the echoPin, returns the sound wave travel time in
microseconds   duration = pulseIn(echoPin, HIGH);

  // Calculating the distance in cm
distance = duration*0.034/2;

  // Adust the distance Range (2cm to
```

400cm) int D = map(distance, 10, 1095, 2,

400);

// Prints the distance on the

LCD lcd.setCursor(0,2);

lcd.print(" Distance: ");

lcd.print(D); lcd.print("cm

"); lcd.setCursor(0,3);

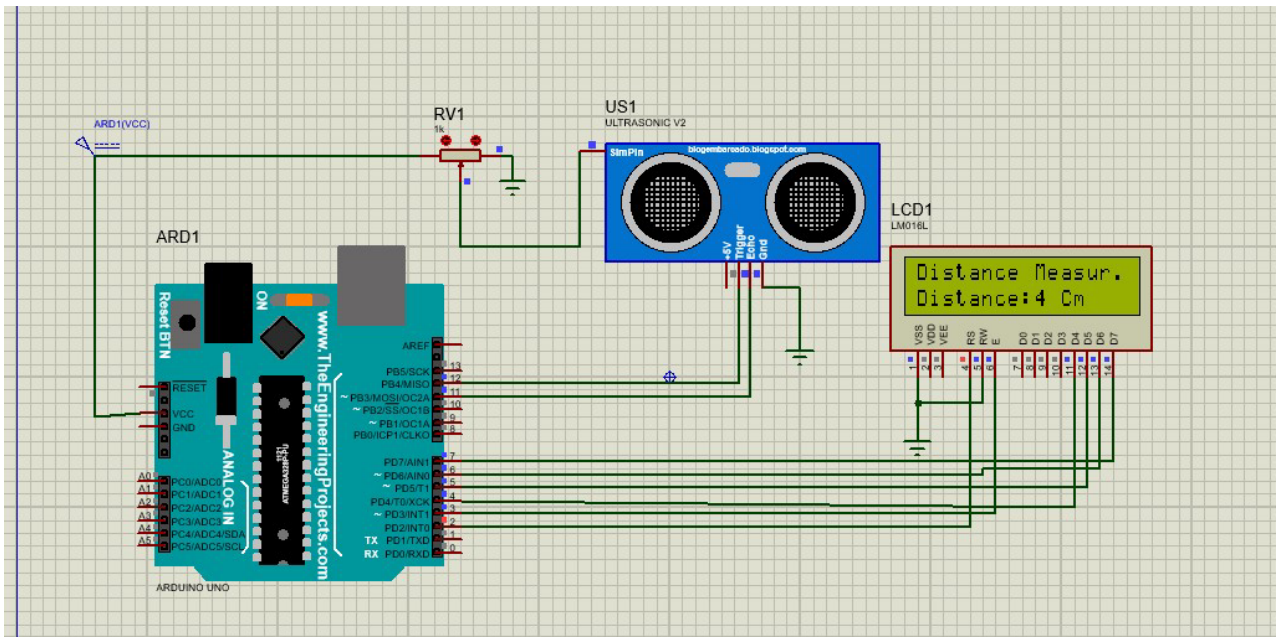
lcd.print(" Distance: ");

lcd.print(D/30.48);

lcd.print("ft ");

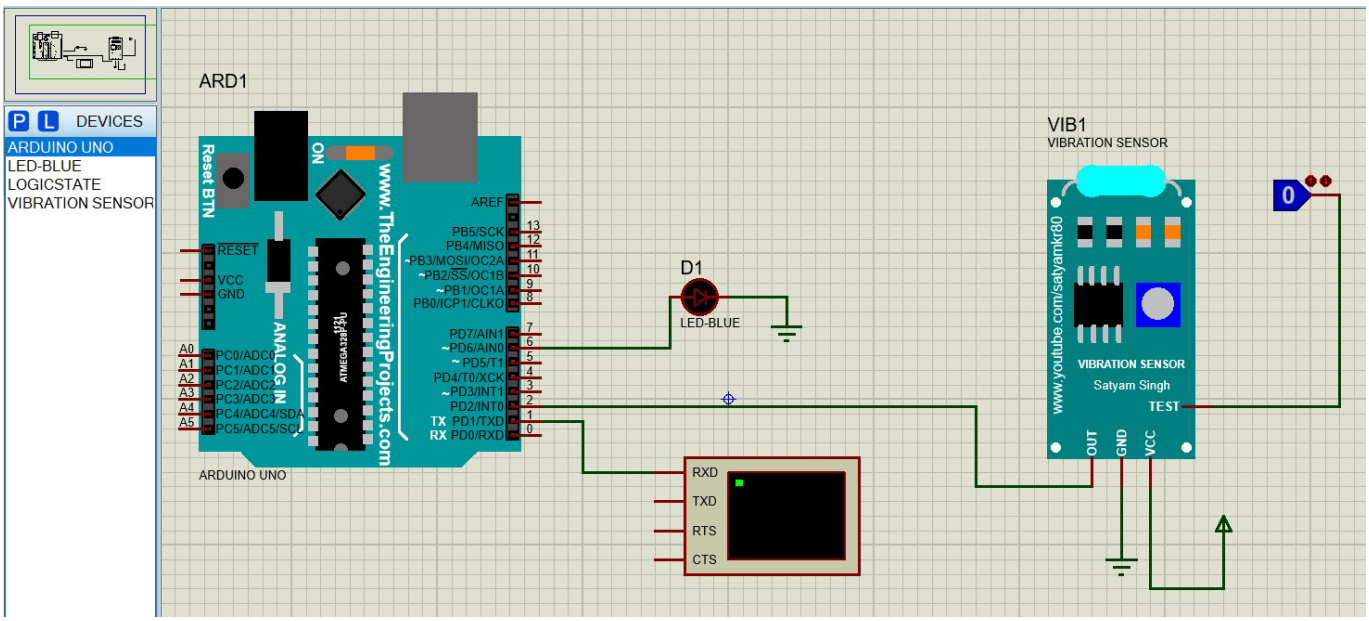
}

3.OUTPUT:



2) To write a program to detects the vibration of an object with sensor using Arduino.

1.Schematic:

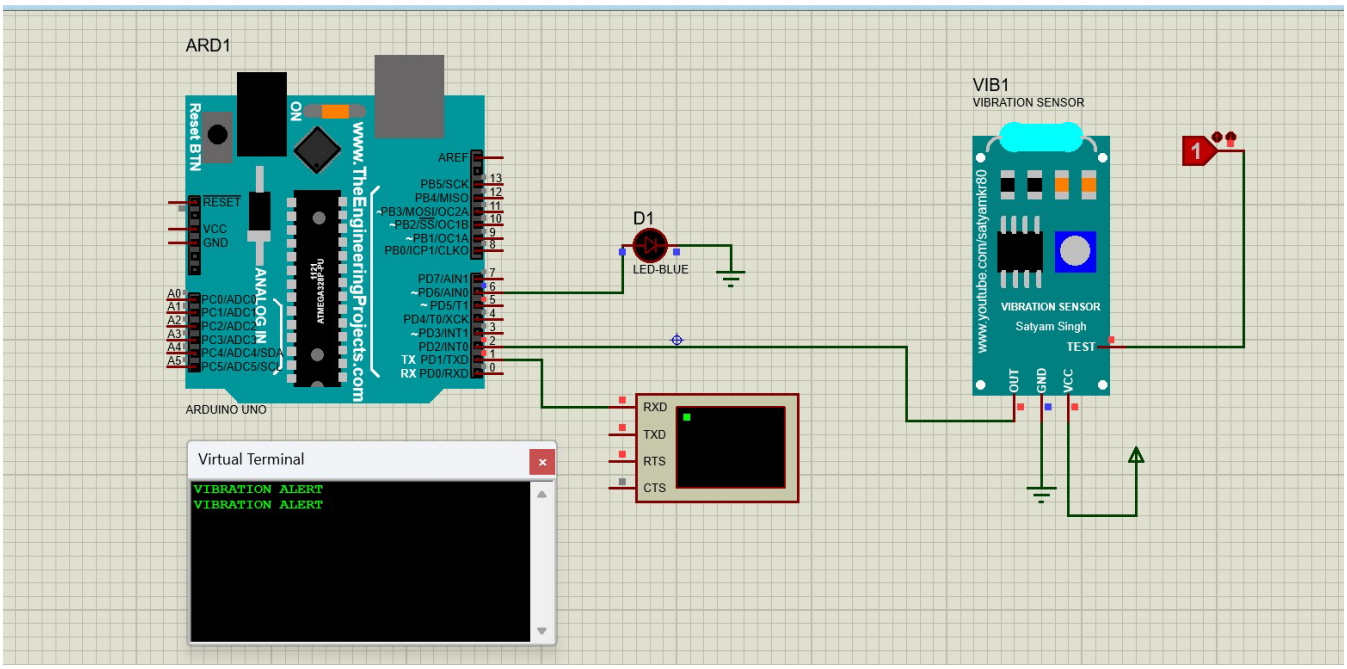


2.Program:

```
int b1 = 2; int
d1 = 5; int
cnt=0,cnt2;
int timer=0;
            // a maximum of eight servo objects can be
created  int pos = 0;    // variable to store the servo
position  void setup() {
 Serial.begin(9600);      //initialize serial
```

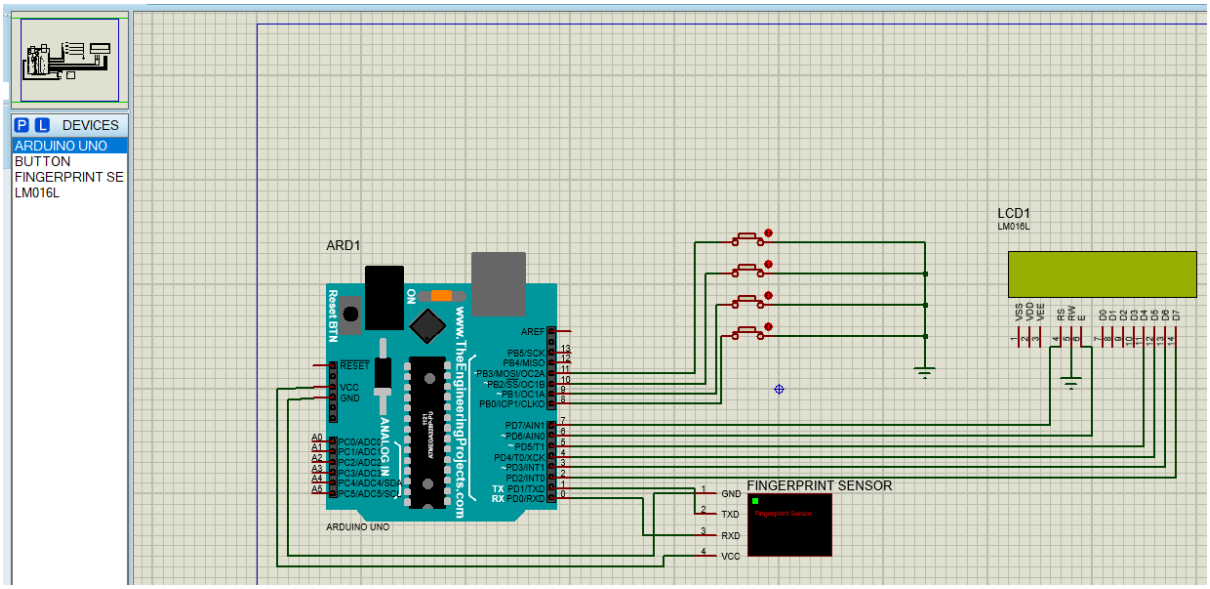
```
pinMode(b1, INPUT_PULLUP);
pinMode(d1, OUTPUT);
digitalWrite(d1, HIGH);
digitalWrite(d1,LOW);
  delay(300);              // wait for a second
cnt=0;
}
void loop() {
  if(digitalRead(b1) ==
HIGH){   Serial.println("VIBRATION ALERT");
digitalWrite(d1, HIGH);
  delay(300);              // wait for a second
  digitalWrite(d1, LOW);
  delay(300);              // wait for a second
digitalWrite(d1, HIGH);
  delay(300);              // wait for a second
  digitalWrite(d1, LOW);
  delay(300);              // wait for a second
digitalWrite(d1, HIGH);
  delay(300);              // wait for a second
  digitalWrite(d1, LOW);
  delay(300);              // wait for a second
  }
}
```

3.OUTPUT:



3) To write a program to sense a finger when it is placed on the board Arduino

1.Schematic:



2.Program:

```
#include <LiquidCrystal.h>
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
```

```
const int buttonPin8 = 8;     // the number of the pushbutton
pin const int buttonPin9 = 9;     // the number of the
pushbutton pin const int buttonPin10 = 10;    // the number of
the pushbutton pin const int buttonPin11 = 11;    // the number
of the pushbutton pin
//const int ledPin =  13;      // the number of the LED pin
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);  // initialize the library with the numbers of the
interface pins int buttonState8 = 0; int buttonState9 = 0; int buttonState10 = 0;
int buttonState11 =
0; uint8_t id =1;
uint8_t id_used =0;
int key= 0;

uint8_t getFingerprintEnroll();
//SoftwareSerial mySerial(11, 12);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial);

void setup() {
  pinMode(buttonPin8, INPUT_PULLUP);
pinMode(buttonPin9, INPUT_PULLUP);
pinMode(buttonPin10, INPUT_PULLUP);
pinMode(buttonPin11, INPUT_PULLUP);
  lcd.begin(16, 2);
lcd.print("
FINGERPRINT");
lcd.setCursor(0, 1);
  lcd.print("ATTENDANCE DEMO");
  delay(500);
finger.begin(57600);
```

```
}
void loop() {
  showmenu();
  while((key = getkey()) == 0);
  if (key == 1)
{           id =
getid();     if(id
== 0)
    {
       lcd.clear();
       lcd.setCursor(0, 0);
lcd.print("No space");
lcd.setCursor(0, 1);
lcd.print("Del templetes");
delay(1000);     }else{
      getFingerprintEnroll();
    }
    delay(1500);
  } else if (key == 2)
{       getFingerprintID();
delay(1500);
  } else if (key == 3) {
    showDelmenu();
    delay(1500);       }
else if (key == 4)
{          lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Credits:");
delay(500);
```

```
lcd.clear();
lcd.setCursor(0, 0);
      lcd.print(“");
      lcd.setCursor(0, 1);
lcd.print("BROTI");
delay(1000);
 }else {
   //lcd.clear();
 }
}
void showmenu()
{
 lcd.clear();
lcd.setCursor(0,          0);
lcd.print("1:Enroll   2:Scan
");           lcd.setCursor(0,
1);  lcd.print("3:Del  ");
}

void showDelmenu()
{
 lcd.clear();  lcd.setCursor(0, 0);  id_used =
getUsedIdnext();  lcd.print("ID
<");lcd.print(id_used);lcd.print(">");
lcd.print(" 1:Del");  lcd.setCursor(0, 1);
lcd.print("2:Up 3:Dn 4:Back");
while(getkey()!=4)
  {
    if(getkey()==2)
```

```
    {
      while(getkey()==2);                    id_used   =
getUsedIdnext();                  lcd.setCursor(0,   0);
lcd.print("ID <");lcd.print(id_used);lcd.print(">  1-Del");
    }else
if(getkey()==3){       while(getkey()==
3);       id_used = getUsedIdprev();
lcd.setCursor(0, 0);       lcd.print("ID
<");lcd.print(id_used);lcd.print(">  1-
Del");
    }else  if(getkey()==1){            while(getkey()==1);
deleteFingerprint(id_used);       unreserveId(id_used);
id_used = getUsedIdprev();         lcd.setCursor(0, 0);
lcd.print("ID <");lcd.print(id_used);lcd.print(">  1-Del");
    }
  }
}

int getkey()
{

if( digitalRead(buttonPin8)){
while( digitalRead(buttonPin8));
   return 1;
 }else
if(digitalRead(buttonPin9)){    while(
 digitalRead(buttonPin9));
   return 2;
 }else
```

```
if(digitalRead(buttonPin10)){     while(
 digitalRead(buttonPin10));
    return 3;
  }else
if(digitalRead(buttonPin11)){     while(
digitalRead(buttonPin11));
    return 4;
  }else{return 0;}
}
int getid(){   for(id = 1
 ;id <=160;id++)
  {
    if(EEPROM.read(id)==0)
    {
      return id;
    }
  }
  if(id>160)
  {
    return 0;
  }
}

int getUsedIdnext()
{
  for(;id_used<=160;)
  {
    id_used++;
if(EEPROM.read(id_used)==1)
```

```
    {
      return id_used;
    }
  }
  if(id_used>160)
  {
    return 0;
  }
 return id_used;
}

int getUsedIdprev()
{
  if(id_used>1)
for(;id_used>0;)
  {
    id_used--;
    if(EEPROM.read(id_used)==1)
    {
      return id_used;
    }
  }
  if(id_used>160)
  {
    return 0;
  }
 return id_used;
}
```

```
void reserveId(int
id){   EEPROM.write(id,1);
}

void unreserveId(int id){
  EEPROM.write(id,0);
}

uint8_t getFingerprintEnroll() {

  int p = -1;
lcd.print("Waiting for finger");
while (p !=
FINGERPRINT_OK) {
    p = finger.getImage();
switch (p) {
    case FINGERPRINT_OK:
      lcd.clear();
      lcd.print("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      lcd.clear();
lcd.print(".");
break;
    case FINGERPRINT_PACKETRECIEVEERR:
      lcd.clear();
      lcd.print("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      lcd.clear();
```

```
      lcd.print("Imaging error");
      break;
default:
lcd.clear();
      lcd.print("Unknown error");
      break;
    }

  p = finger.image2Tz(1);
switch (p) {
    case FINGERPRINT_OK:
      lcd.clear();
      lcd.print("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      lcd.clear();
      lcd.print("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      lcd.clear();
      lcd.print("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      lcd.clear();
      lcd.print("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      lcd.clear();
      lcd.print("Could not find fingerprint features");
      return p;
```

```
default:
lcd.clear();
      lcd.print("Unknown error");
      return p;
  }

  lcd.clear();
  lcd.print("Remove finger");
delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  lcd.clear();   lcd.print("ID
");lcd.print(id);   p = -1;
lcd.clear();
  lcd.print("Place same finger again");
while (p != FINGERPRINT_OK) {
    p = finger.getImage();
switch (p) {
    case FINGERPRINT_OK:
      lcd.clear();
      lcd.print("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      lcd.clear();
lcd.print(".");
break;
    case FINGERPRINT_PACKETRECIEVEERR:
```

```
      lcd.clear();
      lcd.print("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      lcd.clear();
      lcd.print("Imaging error");
      break;
default:
lcd.clear();
      lcd.print("Unknown error");
      break;
    }
}
  p = finger.image2Tz(2);
switch (p) {
    case FINGERPRINT_OK:
      lcd.clear();
      lcd.print("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      lcd.clear();
      lcd.print("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      lcd.clear();
      lcd.print("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      lcd.clear();
```

```
      lcd.print("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      lcd.clear();
      lcd.print("Could not find fingerprint features");
      return p;
    default:
lcd.clear();
      lcd.print("Unknown error");
      return p;
  }

  // OK converted!
  lcd.clear();
  lcd.print("Creating model #");lcd.print(id);

  p =
finger.createModel();   if
(p == FINGERPRINT_OK)
{
    lcd.clear();
     lcd.print("Prints matched!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    lcd.clear();
    lcd.print("Communication error");
    return p;
  } else if (p == FINGERPRINT_ENROLLMISMATCH) {
    lcd.clear();
    lcd.print("Fingerprints did not
```

```
match");     return p;   } else
{     lcd.clear();
    lcd.print("Unknown error");
    return p;
  }

  lcd.clear();
  lcd.print("ID
");lcd.print(id);   p =
finger.storeModel(id);   if
(p == FINGERPRINT_OK)
{
    lcd.clear();     lcd.print("Stored at
id ");lcd.print(id);     reserveId(id);
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    lcd.clear();
    lcd.print("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    lcd.clear();
    lcd.print("Could not store in that location");
return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    lcd.clear();     lcd.print("Error
writing to flash");     return p;   }
else {     lcd.clear();
    lcd.print("Unknown error");
    return p;
  }
```

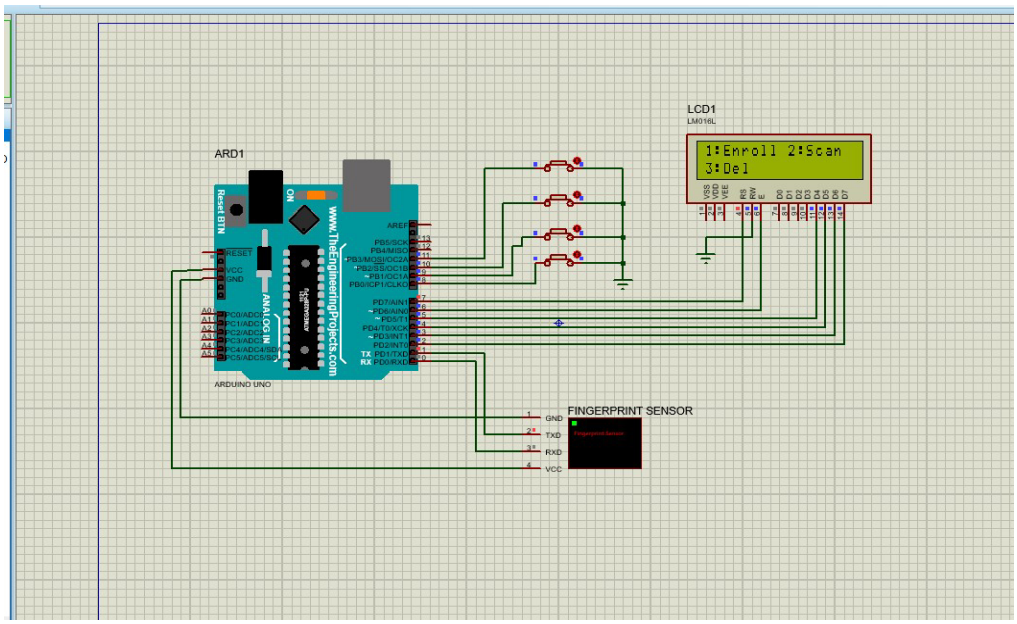
```
}
uint8_t
getFingerprintID()
{   uint8_t p = 0;
lcd.clear();
lcd.print("Place Finger");
delay(1500);

 p = finger.getImage();


 switch (p) {
   case FINGERPRINT_OK:
     lcd.clear();
     lcd.print("Image taken");
     break;
   case FINGERPRINT_NOFINGER:
     lcd.clear();
     lcd.print("No finger detected");
     return p;
   case FINGERPRINT_PACKETRECIEVEERR:
     lcd.clear();
     lcd.print("Communication error");
     return p;
   case FINGERPRINT_IMAGEFAIL:
     lcd.clear();
     lcd.print("Imaging
error");       return p;
default:       lcd.clear();
     lcd.print("Unknown error");
     return p;
```

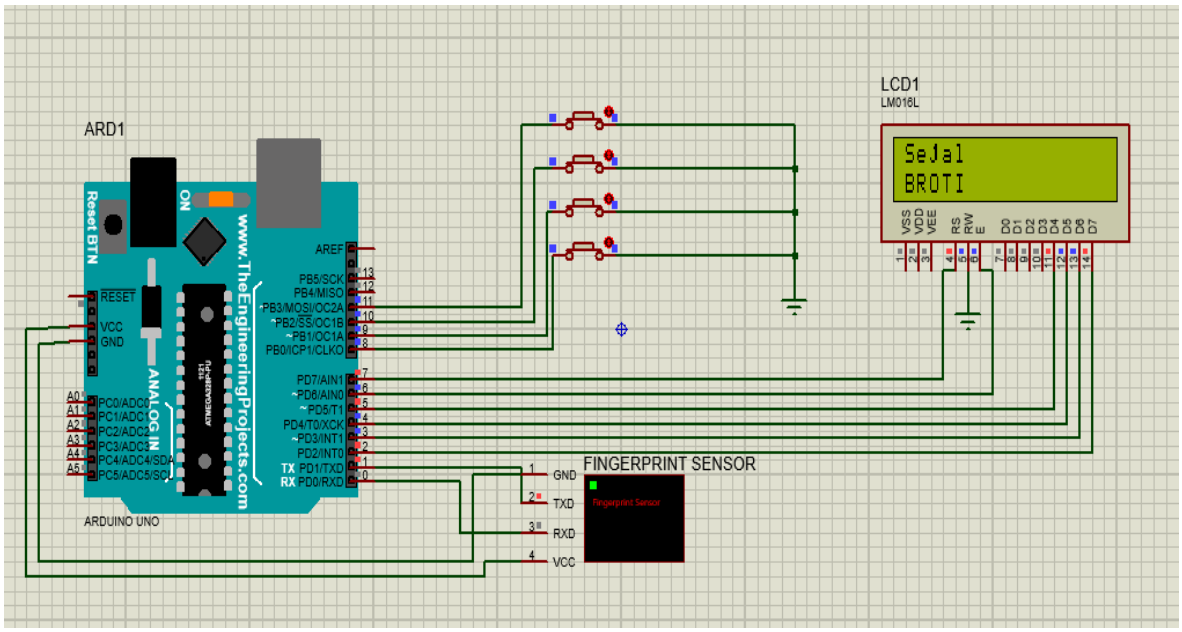
```
  }
  p = finger.image2Tz();
switch (p) {
    case FINGERPRINT_OK:
      lcd.clear();
      lcd.print("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      lcd.clear();
      lcd.print("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      lcd.clear();
      lcd.print("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      lcd.clear();
      lcd.print("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      lcd.clear();
      lcd.print("Could not find fingerprint features");
      return p;
default:
lcd.clear();
      lcd.print("Unknown error");
      return p;
  }
  p =
```

```
finger.fingerFastSearch();
if (p == FINGERPRINT_OK)
{
   lcd.clear();
   lcd.print("Found a print match!");   } else if
(p == FINGERPRINT_PACKETRECIEVEERR) {
   lcd.clear();
   lcd.print("Communication error");
   return p;
 } else if (p == FINGERPRINT_NOTFOUND) {
   lcd.clear();
   lcd.print("Did not find a
match");     return p;   } else
{     lcd.clear();
   lcd.print("Unknown error");
   return p;
 }

 lcd.clear();     lcd.print("Found ID #");
lcd.print(finger.fingerID);
}
uint8_t deleteFingerprint(uint8_t id)
{   uint8_t p = -1;
  p = finger.deleteModel(id);
   if (p == FINGERPRINT_OK) {
   Serial.println("Deleted!");
 } else if (p == FINGERPRINT_PACKETRECIEVEERR)
{     Serial.println("Communication error");
   return p;
```

```
  } else if (p == FINGERPRINT_BADLOCATION) {
   Serial.println("Could not delete in that location");
return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.print("Unknown error: 0x"); Serial.println(p, HEX);
    return p;
  }
}
```

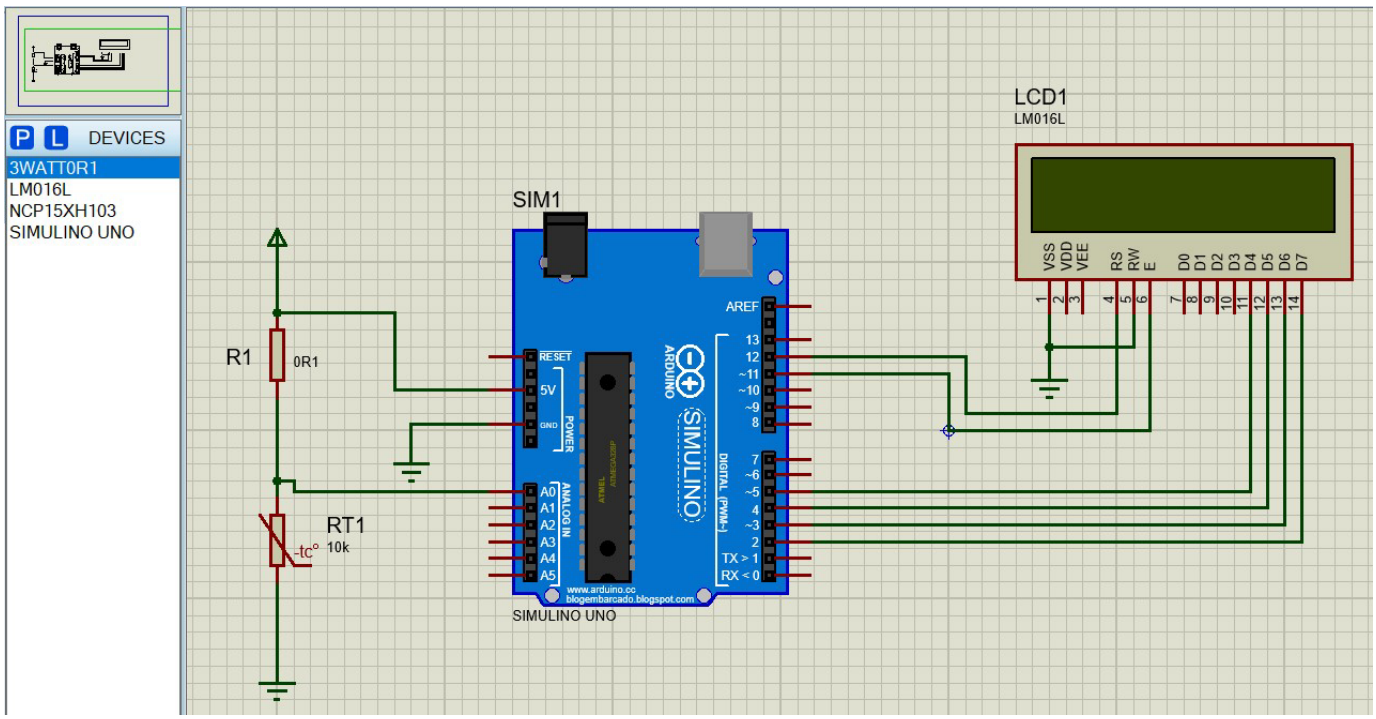
OUTPUT:





4) To write a program to get temperature notification using Arduino.

1)Schematic:



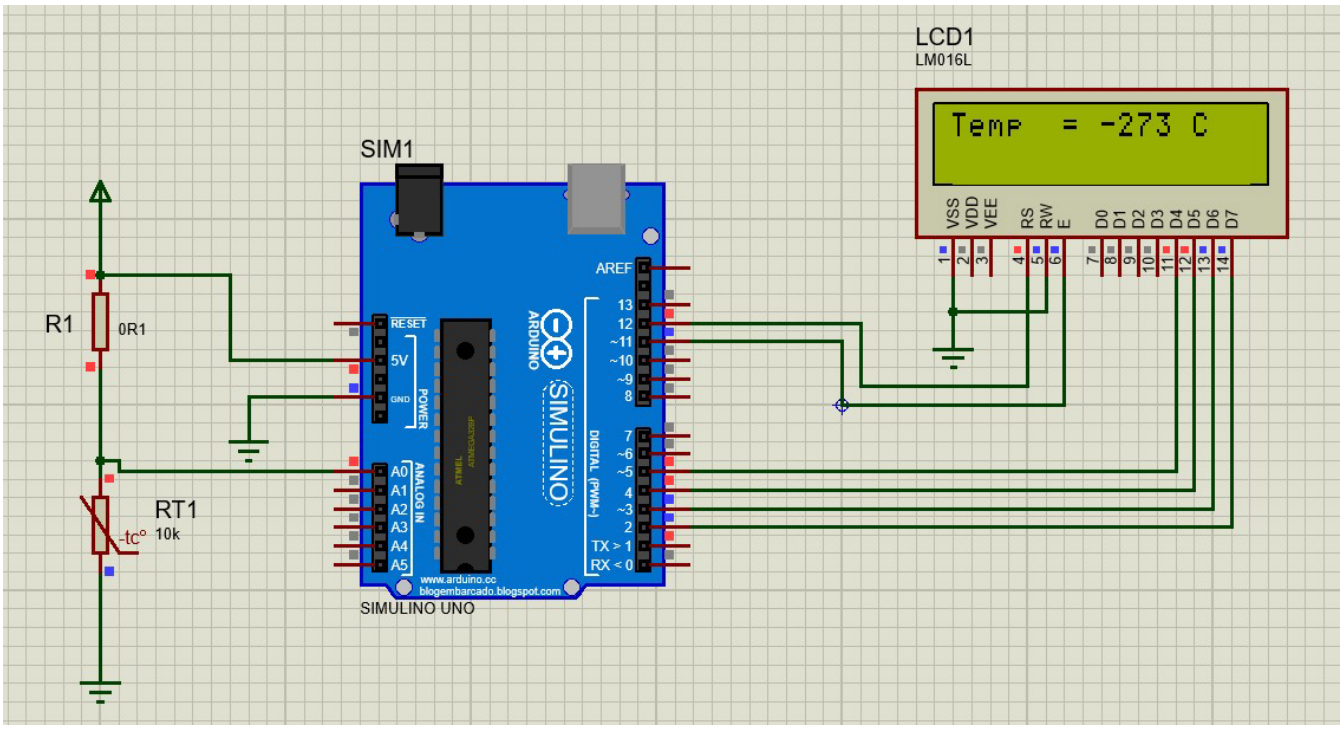
2)Program:

```
#include <LiquidCrystal.h> int
THERMISTORPIN = 0, BCOEFFICIENT =
3380;
float THERMISTORNOMINAL = 10000, TEMPERATURENOMINAL = 25,
```

```
SERIESRESISTOR = 10000;
LiquidCrystal lcd(12, 11, 5, 4, 3,
2); int sample[5]; void setup()
{   Serial.begin(9600);
lcd.begin(16, 2);
}
void loop() {
  int i;
  float average;
  // Take N samples in a row, with a slight delay
  for (i = 0; i < 5; i++) {
    sample[i] = analogRead(THERMISTORPIN);
    delay(10);
  }
  // Average all the samples out
average = 0;
  for (i = 0; i < 5; i++) {
    average += sample[i];
  }
  average /= 5;
  // Convert the value to resistance   average = 1023.0 /
average - 1; // Use 1023.0 for floating point division   average =
SERIESRESISTOR / average;
  // Calculate temperature using the Steinhart-Hart equation
  float steinhart;
  steinhart = average / THERMISTORNOMINAL; // (R/Ro)
  steinhart = log(steinhart);              // ln(R/Ro)
steinhart /= BCOEFFICIENT;               // 1/B *
ln(R/Ro)
```

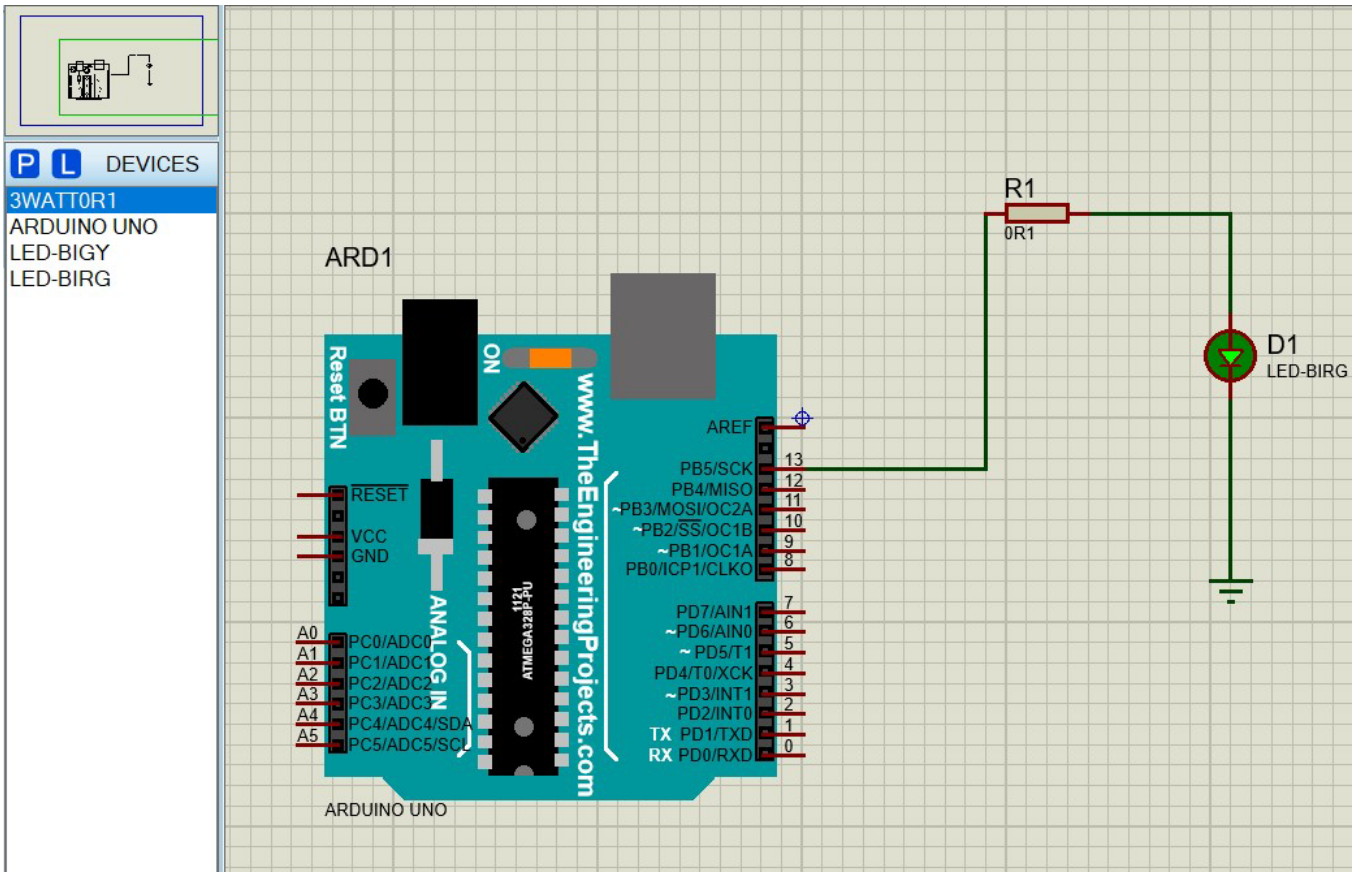
```
  steinhart += 1.0 / (TEMPERATURENOMINAL + 273.15); // +
(1/To)
  steinhart = 1.0 / steinhart;              // Invert
steinhart -= 273.15;                      // Convert to
Celsius
  // Print debugging information
  Serial.print("Analog Value: ");
  Serial.println(analogRead(THERMISTORPIN));
  Serial.print("Average Resistance: ");
  Serial.println(average);
  Serial.print("Temperature: ");
  Serial.println(steinhart);
  // Display on LCD
  lcd.setCursor(0,    0);
lcd.print("Temp    =  ");
lcd.print((int)steinhart)
;            lcd.print("  C");
delay(500);
lcd.clear();
}
```

3)OUTPUT:



5)To write a program for LDR to vary the light intensity of LED using Arduino

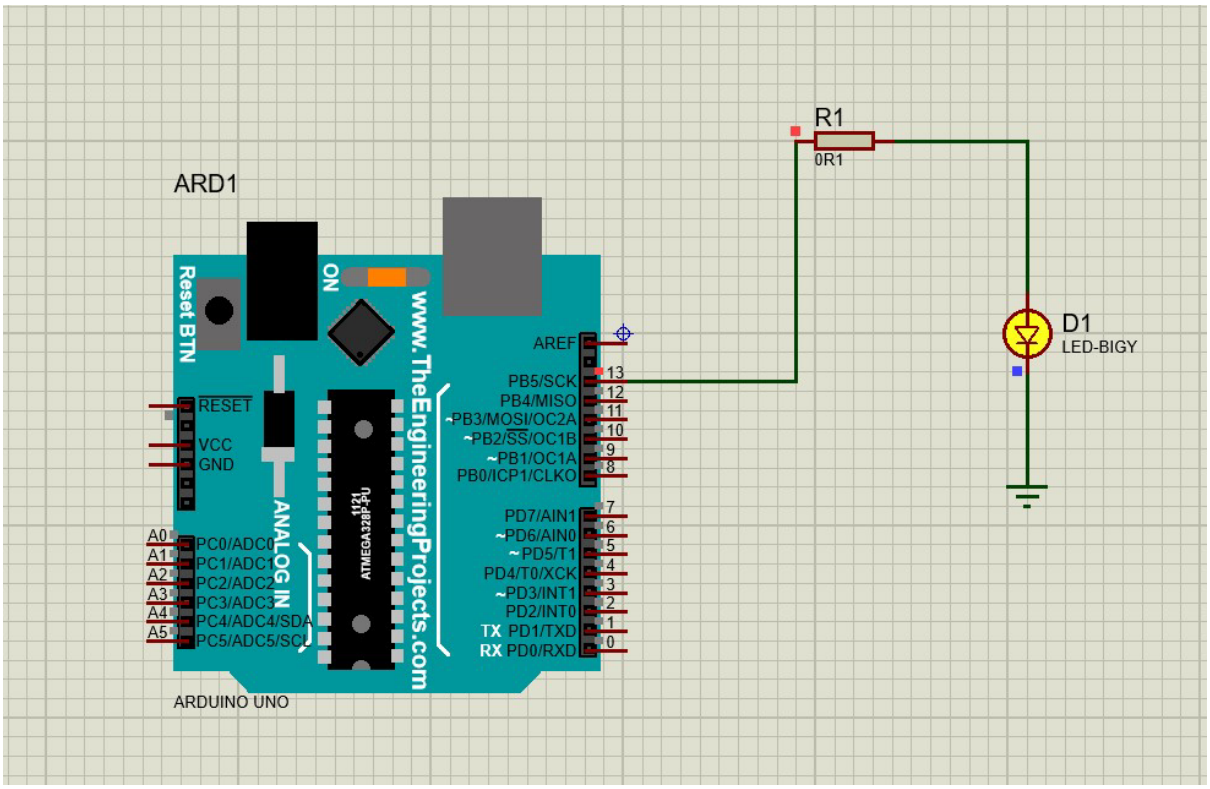
1.schematic

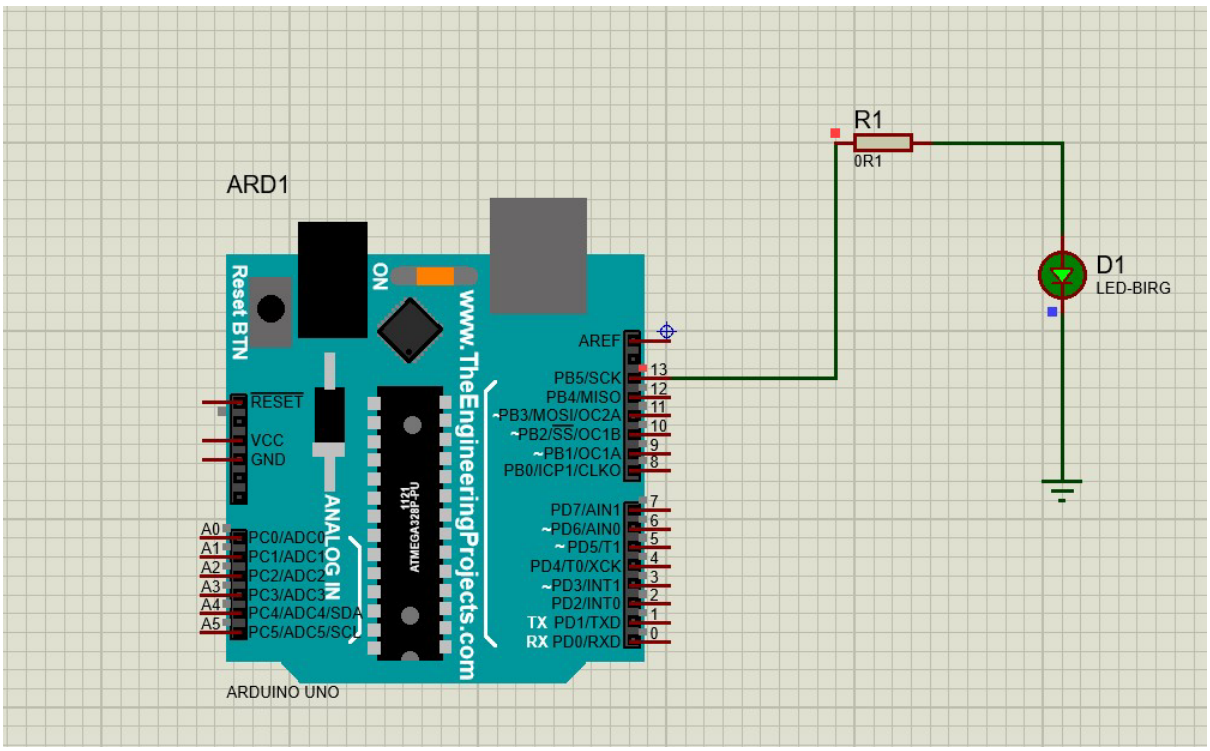


2.Program : LED.ino

```
void setup() {   pinMode(13, OUTPUT); // pin 13 - change value if
you have LED on diff pin
}
void loop() {   digitalWrite(13, HIGH);  // set pin 13 to high
voltage, turning LED on   delay(1000);            // wait 1000
milliseconds, or one second.   digitalWrite(13, LOW);   // set
pin 13 to low voltage, or zero. LED off.   delay(1000);
//wait one second before starting the loop again.
}
```

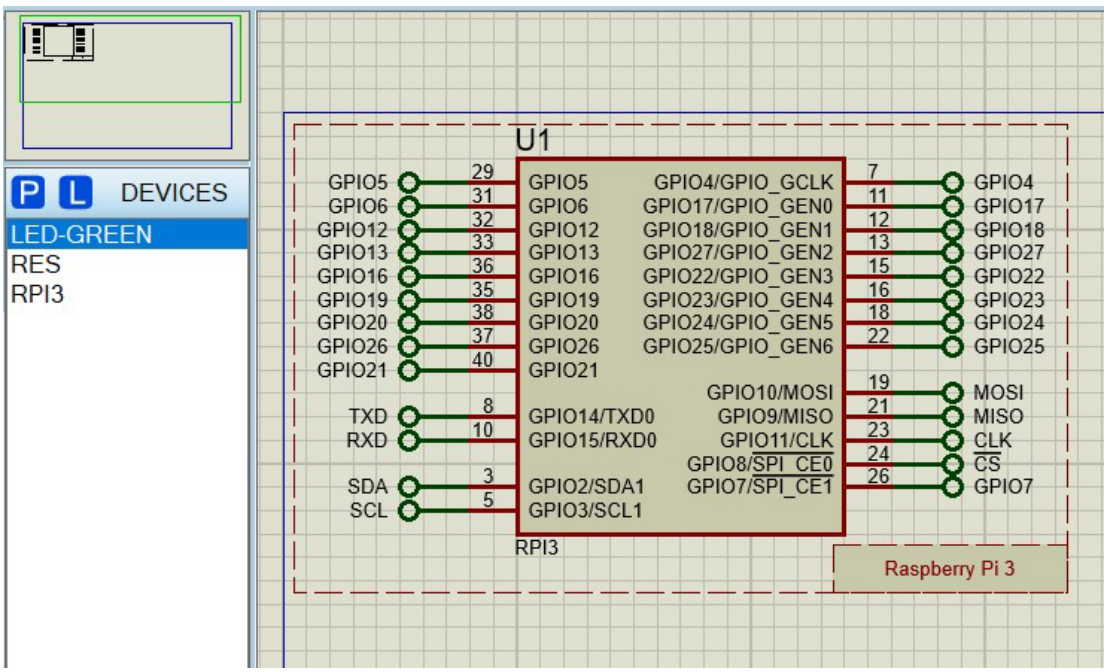
3)Output:





6) Run some python programs on Pi like:

Schematic:



a) Read your name and print Hello message with

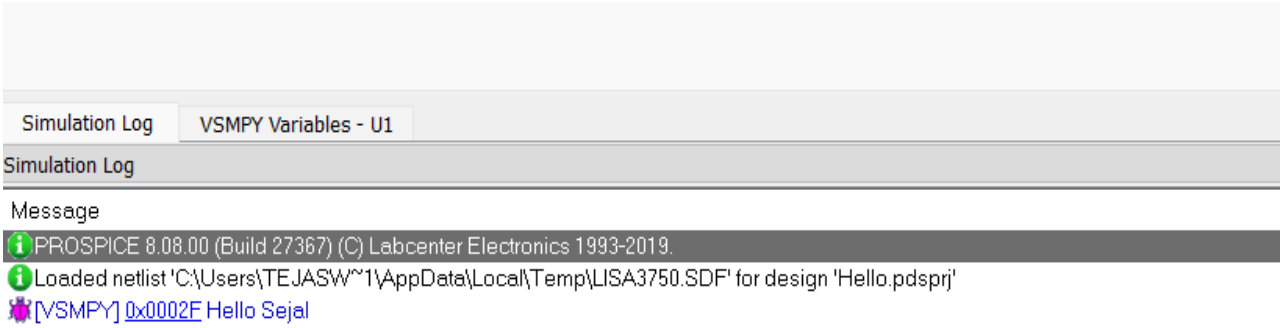
name

Program :

```
from goto import * import time import var import pio
import resource import cpu import FileStore import
VFP


def peripheral_setup () : #
Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileSto
re ()
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)

# Install interrupt handlers
```

```
def peripheral_loop () :
 pass

def main () : #
Setup
peripheral_setup
()
 name="Sejal"
print("Hello "+name)
# Infinite loop  while 1 :
  peripheral_loop()
pass
# Command line
execution if __name__ ==
'__main__' :
   main()
```

OUTPUT:



b)Read two numbers and print their sum, difference, product and division.

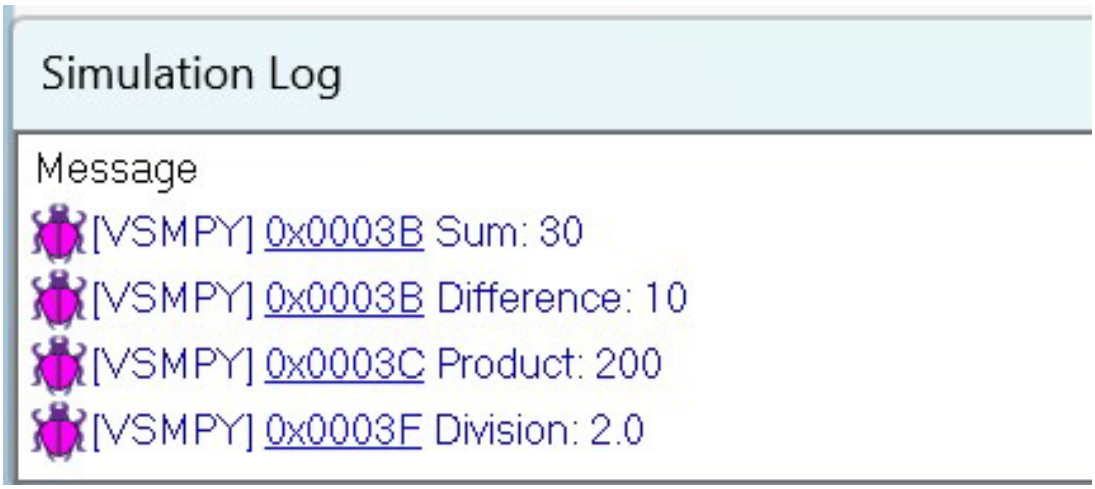
Program :

```
# Modules from
goto import *
import time
import var
import pio
import resource
import cpu
import FileStore
import VFP

def peripheral_setup () : #
Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileSto
re ()
pio.server=VFP.VfpServer ()
```

```
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers

def peripheral_loop () :
 pass

#---CONFIG_END---

# Main function
def main () : #
Setup
peripheral_setup
()  num1 = 20
num2 = 10

# Calculate sum, difference, product, and
division  sum_result = num1 + num2
difference_result = num1 - num2
product_result = num1 * num2
# Print the results  print(f"Sum:
{sum_result}")  print(f"Difference:
{difference_result}")
print(f"Product: {product_result}")
print(f"Division: {division_result}")

# Infinite loop
while 1 :
peripheral_loop()
pass
# Command line
```

```
execution if __name__ ==
'__main__' :
  main()
```

OUTPUT:



c) Word and character count of a given string.

Program:

```
 from goto import * import time import var
import pio import resource import cpu import
FileStore import VFP


def peripheral_setup () : #
Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileSto
re ()
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers


def peripheral_loop () :
```

```
 pass


#---CONFIG_END---


# Main function def main
() : # Setup
peripheral_setup()
str="Hello World"
character_count =
len(str)  word_count =
len(str.split())
print("Character count:
",character_count)
print("word count:
",word_count)
# Infinite loop  while 1 :
  peripheral_loop()
pass
# Command line
execution if __name__ ==
'__main__' :
   main()
```

OUTPUT:

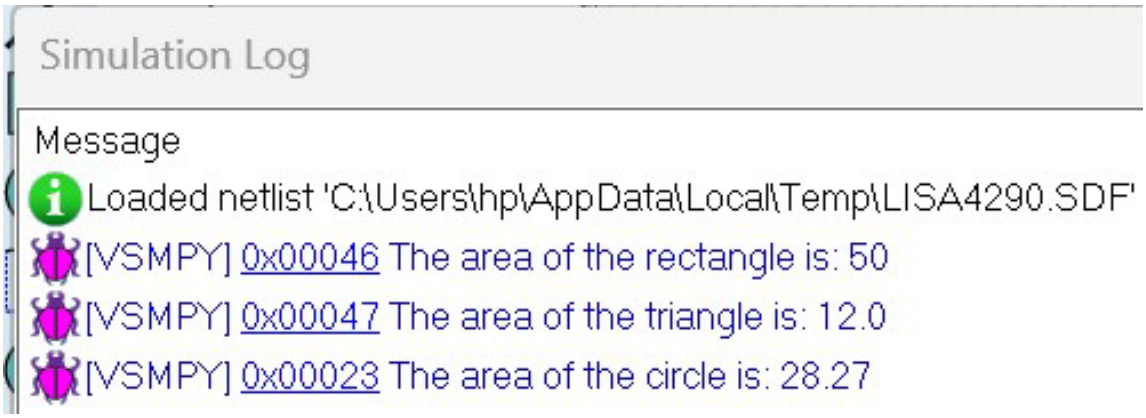
d)Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

Program:

```
 from goto import * import time import var import pio import resource import math
import cpu import FileStore import VFP def peripheral_setup () : # Peripheral
Constructors  pio.cpu=cpu.CPU ()  pio.storage=FileStore.FileStore ()
pio.server=VFP.VfpServer ()  pio.storage.begin ()  pio.server.begin (0)


def peripheral_loop () :
 pass def
main () :
peripheral_setup()  def
calculate_rectangle_area(width, height):
   return width * height  def
```

```
calculate_triangle_area(base, height):
    return 0.5 * base * height
def
calculate_circle_area(radius):
return math.pi * radius ** 2

 rectangle_width = 5
rectangle_height = 10

 triangle_base = 6
triangle_height = 4

 circle_radius = 3

# Calculate areas  rectangle_area =
calculate_rectangle_area(rectangle_width, rectangle_height)
triangle_area = calculate_triangle_area(triangle_base, triangle_height)
circle_area = calculate_circle_area(circle_radius)
# Print the results  print(f"The area of the
rectangle is: {rectangle_area}")  print(f"The area
of the triangle is: {triangle_area}")  print(f"The
area of the circle is: {circle_area:.2f}")

while 1 :
peripheral_loop()
pass
# Command line
execution if __name__ ==
'__main__' :
   main()
```

2. OUTPUT:



7) Run some python programs on Pi like:

a) Handle Divided by Zero Exception.

1.Program :

Modules from goto import * import time import var import pio import resource import math import cpu import FileStore import VFP

def peripheral_setup () : #

Peripheral Constructors

pio.cpu=cpu.CPU ()

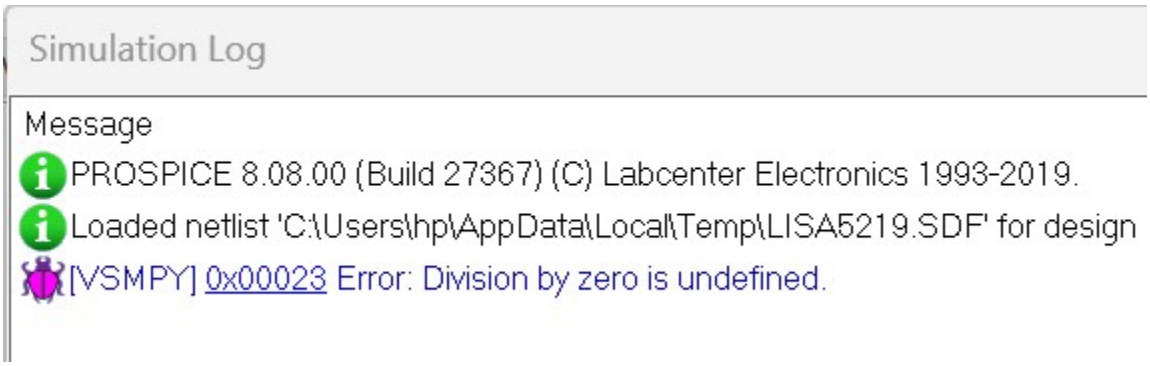
pio.storage=FileStore.FileSto

re ()

```
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers

def peripheral_loop () :
 pass

#---CONFIG_END---

# Main function
def main () : #
Setup
peripheral_setup
() numerator =
10 denominator
= 0  # Change
this to a non-
zero value to see
a successful
division

# Attempt to perform the division
 try:
   result = numerator / denominator     print(f"The result
of {numerator} / {denominator} is: {result}")  except
ZeroDivisionError:
   print("Error: Division by zero is undefined.")

# Infinite loop
while 1 :
```

```
peripheral_loop()
pass
# Command line
execution if __name__ ==
'__main__' :
  main()
```

2.OUTPUT:



b) Print current time for 10 times with an interval of10seconds.

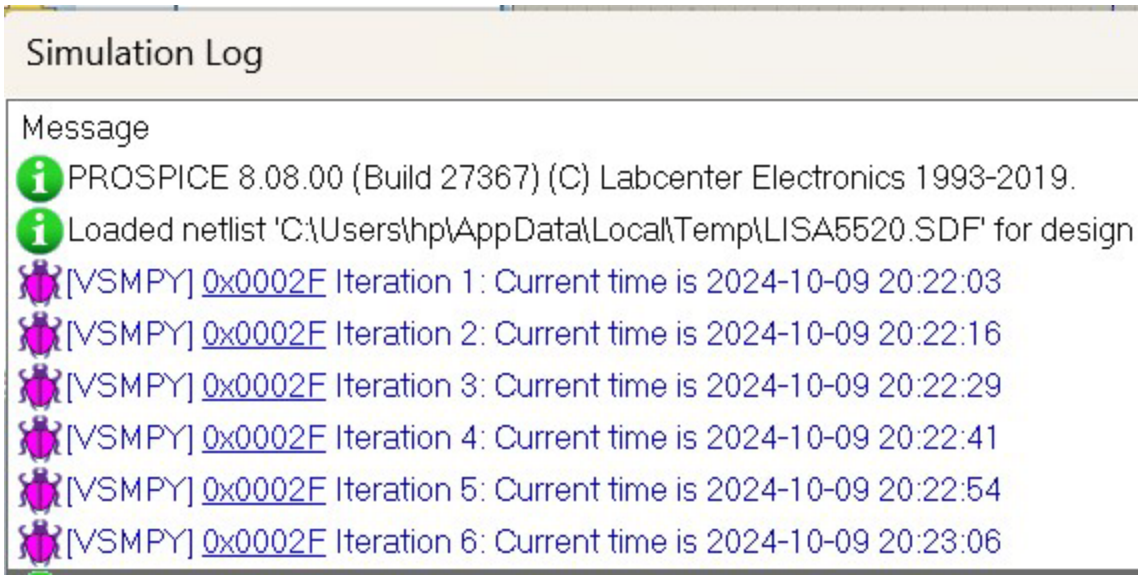
1. Program:

```
 from goto import * import time import var import pio import resource


# Peripheral Configuration Code (do not edit)
#---
CONFIG_BEGIN---
import cpu import
```

```
FileStore import
VFP

def peripheral_setup () : #
Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileSto
re ()
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers

def peripheral_loop () :
 pass

#---CONFIG_END---

# Main function
def main():    #
Setup
peripheral_setup()
# Print current
time 10 times with
an interval of 10
seconds    for i in
range(10):
      current_time = time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime())       print(f"Iteration {i + 1}: Current time is
{current_time}")       time.sleep(10)  # Wait for 10 seconds
```

```
    # Infinite loop
while 1:
peripheral_loop()
        pass


# Command line
execution if _name_ ==
'_main_':
    main()
```

2.OUTPUT:



c) Read a fileline byline and print the word count of each line

1. Program:

```
# Modules from
goto import *
import time
import var
import pio
import resource
import math

# Peripheral Configuration Code (do not edit)
#---
CONFIG_BEGIN---
import cpu import
FileStore import
VFP

def peripheral_setup () : #
Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileSto
re ()
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers

def peripheral_loop () :
 pass # Main
function
def main () : # Setup  peripheral_setup() filename =
r'C:\Users\hp\OneDrive\Desktop\arduino pract\Example.txt' try:
with open(filename, 'r') as file:        for line_number, line in
```

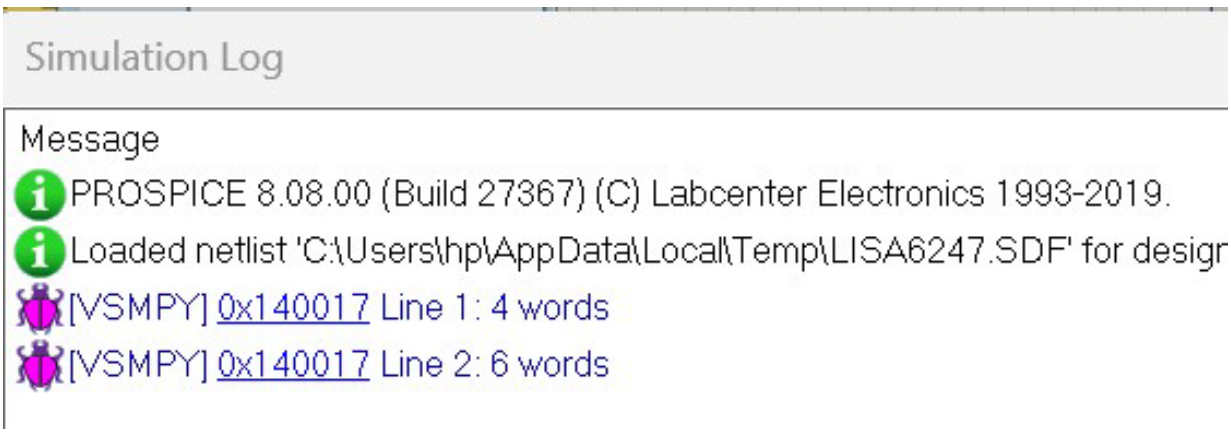
```
enumerate(file, start=1):            # Count the number of words in
the line            word_count = len(line.split())            print(f"Line
{line_number}: {word_count} words")  except FileNotFoundError:
    print(f"Error: The file '{filename}' does not exist.")


# Infinite loop
while 1 :
peripheral_loop()
pass
# Command line
execution if __name__ ==
'__main__' :
  main()
```

Example.txt

Welcome to our college.

Our college is providing different courses.

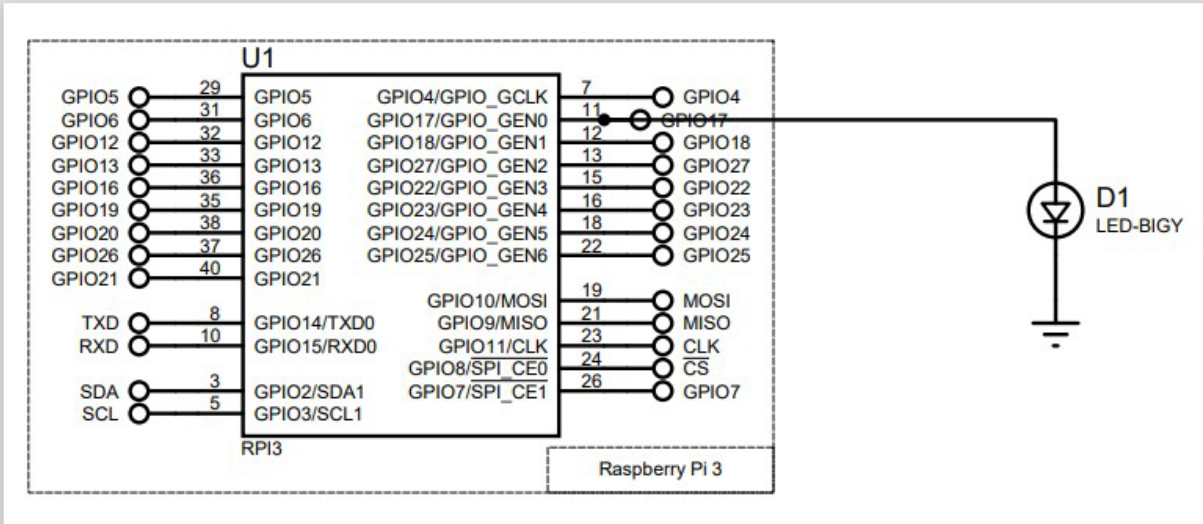
2. OUTPUT:



8) Run some python programs on Pi like

a) Light an LED through Python program

1.Schematic:

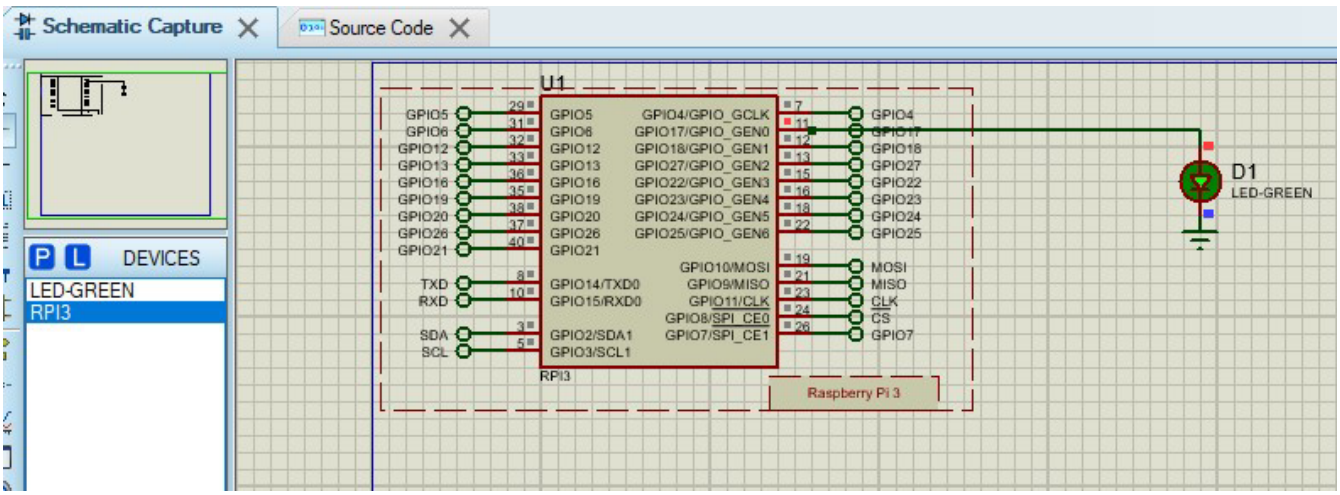


2.Program:

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

# GPIO setup
LED_PIN = 17

def peripheral_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(LED_PIN, GPIO.OUT)

def peripheral_loop():
```

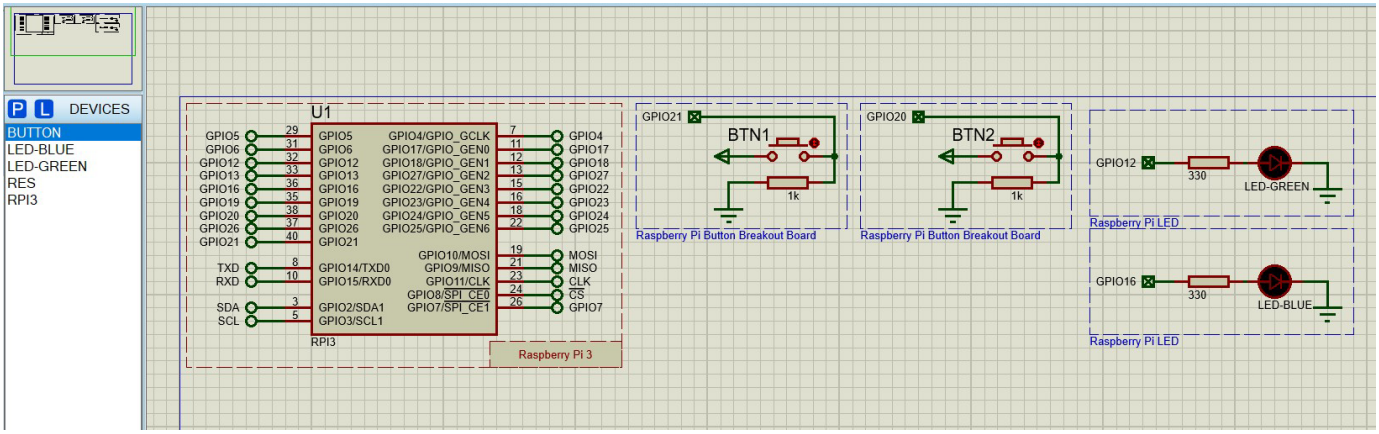
```
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(1)
def main():
    try:
        peripheral_setup()
while True:
            peripheral_loop()
except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

3.Output:



b) Get input from two switches and switch on corresponding

LEDs 1.Schematic:

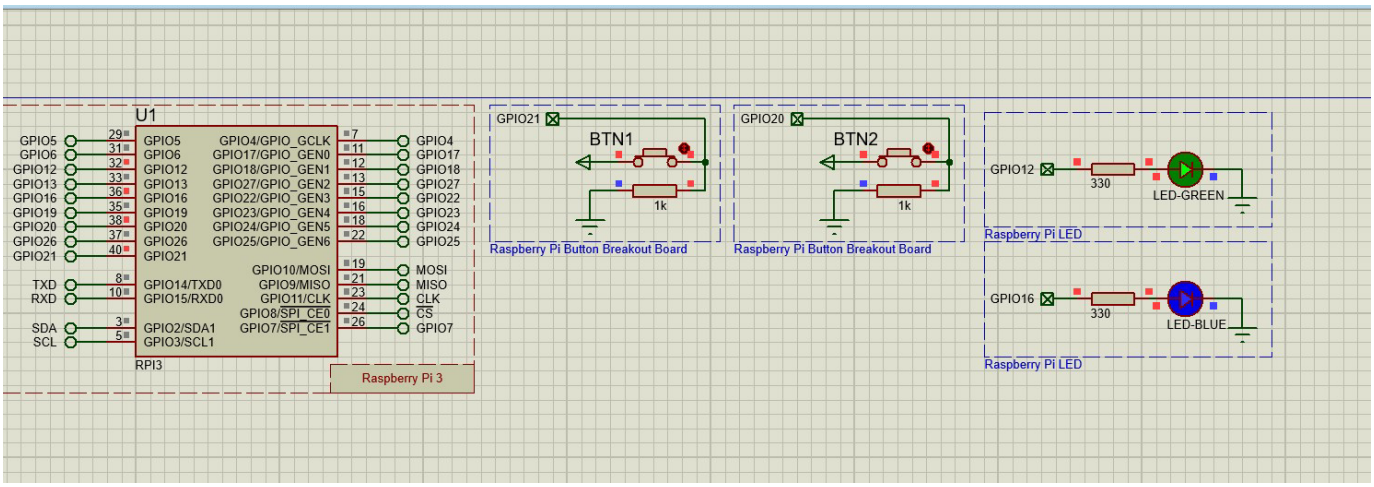


2.Program:

```
 import RPi.GPIO as
GPIO import time


# Set the mode to BCM
GPIO.setmode(GPIO.BCM)


# Pin configuration
switch1_pin = 21  # Switch 1
switch2_pin = 20  # Switch
2 led1_pin = 16     # LED 1
(Green) led2_pin = 12     #
LED 2


# Setup pins for switches as input with pull-down resistors
```

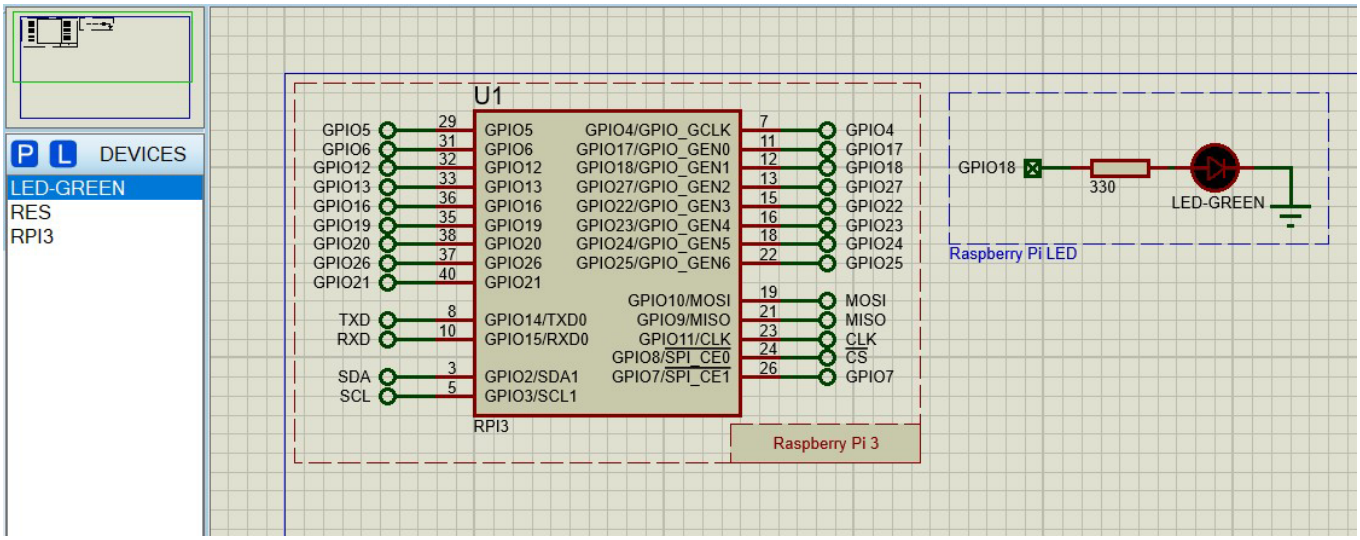
```
GPIO.setup(switch1_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(switch2_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Setup pins for LEDs as output
GPIO.setup(led1_pin, GPIO.OUT)
GPIO.setup(led2_pin, GPIO.OUT)
try:    while
True:
      # Read input from switches       if
GPIO.input(switch1_pin) == GPIO.HIGH:
         GPIO.output(led1_pin, GPIO.HIGH)  # Turn on LED 1
      else:
         GPIO.output(led1_pin, GPIO.LOW)   # Turn off LED 1

      if GPIO.input(switch2_pin) == GPIO.HIGH:
         GPIO.output(led2_pin, GPIO.HIGH)  # Turn on LED 2
      else:
         GPIO.output(led2_pin, GPIO.LOW)   # Turn off LED 2

      time.sleep(0.1)  # Add a small delay to prevent bouncing issues

except KeyboardInterrupt:
   # Clean up GPIO when the program is terminated
   GPIO.cleanup()
```

3.OUTPUT:



c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.

1. Schematic :



2.Program: Main.py

import RPi.GPIO as

GPIO import time

```
# Set up GPIO
LED_PIN = 18  # Replace with your actual pin number
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

# Function to read configuration from
file def read_config(filename):
   config = {}
try:
      with open(filename, 'r') as file:
         for line in file:
            key, value = line.strip().split('=')
config[key] = float(value)     except
FileNotFoundError:
      print(f"Error: The file '{filename}' does not exist.")
return config

# Main function
def main():
   config = read_config('led_config.txt')

   on_time = config.get('on_time', 1)  # Default to 1 second if not found
off_time = config.get('off_time', 1)  # Default to 1 second if not found
    try:
while True:
         GPIO.output(LED_PIN, GPIO.HIGH)  # Turn
LED on           time.sleep(on_time)  # Wait for the
on time           GPIO.output(LED_PIN, GPIO.LOW)
# Turn LED off           time.sleep(off_time)  # Wait
for the off time     except KeyboardInterrupt:
```

```
        print("Program interrupted. Cleaning up...")
    finally:
        GPIO.cleanup()  # Clean up GPIO settings

if __name__ == "__main__":
    main()
```

led_config.txt:

```
on_time=1  # Time in seconds to keep the LED on
off_time=0.5  # Time in seconds to keep the LED off
```

OUTPUT: