\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define LED1 10

#define LED2 11

#define LED3 12

#define LED4 13

void setup() {

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

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

}

void loop() {

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

  digitalWrite(LED1, HIGH);

  delay(200);

  digitalWrite(LED1, LOW);

  delay(200);

  digitalWrite(LED2, HIGH);

  delay(200);

  digitalWrite(LED2, LOW);

  delay(200);

  digitalWrite(LED3, HIGH);

  delay(200);

  digitalWrite(LED3, LOW);

  delay(200);

  digitalWrite(LED4, HIGH);

  delay(200);

  digitalWrite(LED4, LOW);

  delay(200);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RGB LED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define redpin 3

#define greenpin 5

#define bluepin 6

int r = 255;

int g = 0;

int b = 0;

void setup() {

  pinMode(redpin, OUTPUT);

  pinMode(greenpin, OUTPUT);

  pinMode(bluepin, OUTPUT);

}

void rgb(int r, int g, int b) {

  analogWrite(redpin, r);

  analogWrite(greenpin, g);

  analogWrite(bluepin, b);

}

void loop() {

  for (int i = 0; i < 255; i++) {

    rgb(r, g, b);

    g++;

    delay(10);

  }

  for (int i = 0; i < 255; i++) {

    rgb(r, g, b);

    r--;

    delay(10);

  }

  for (int i = 0; i < 255; i++) {

    rgb(r, g, b);

    b++;

    delay(10);

  }

  for (int i = 0; i < 255; i++) {

    rgb(r, g, b);

    g--;

    delay(10);

  }

  for (int i = 0; i < 255; i++) {

    rgb(r, g, b);

    r++;

    delay(10);

  }

  for (int i = 0; i < 255; i++) {

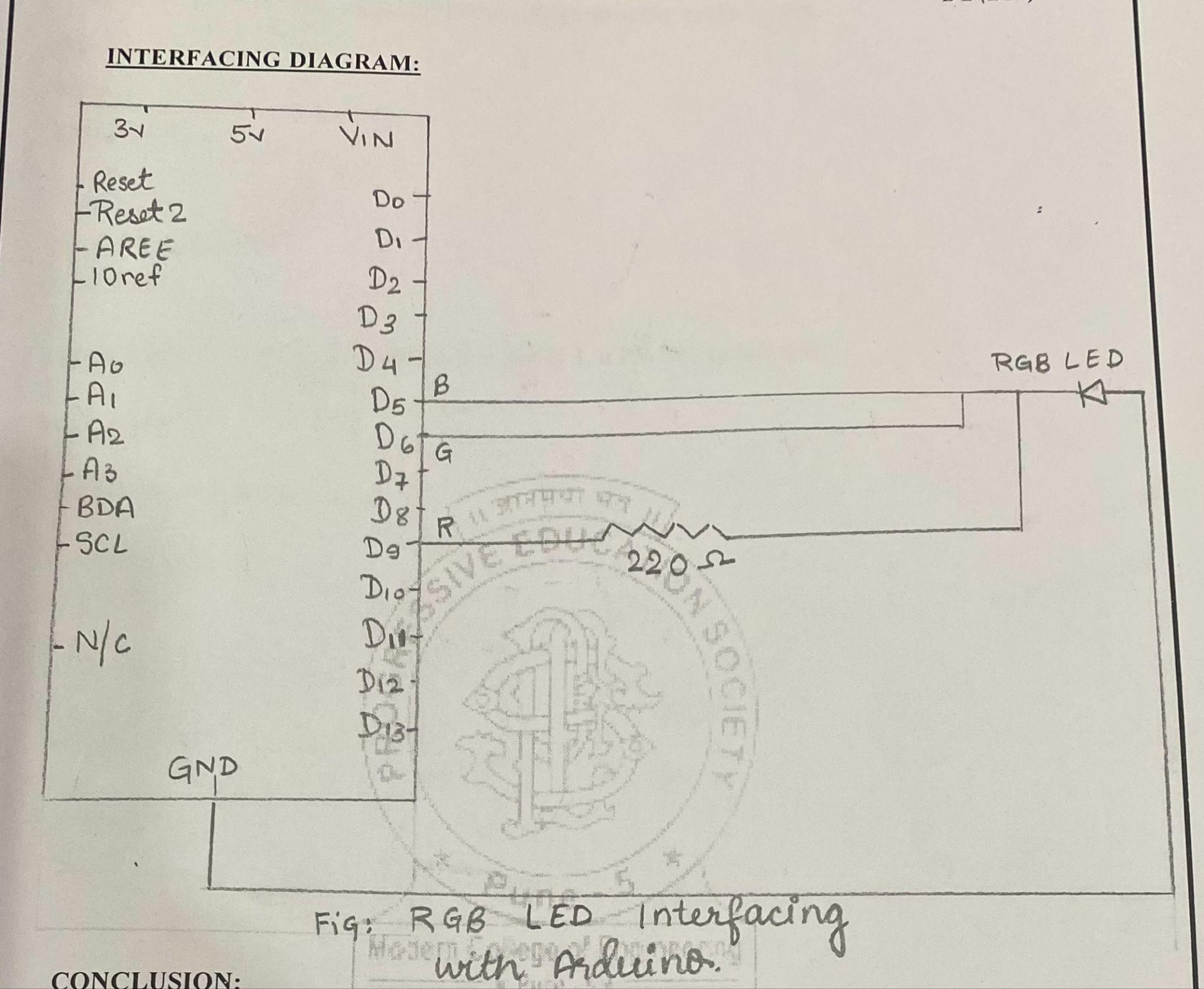
    rgb(r, g, b);

    b--;

    delay(10);

  }

}



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LDR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int ldrPin = A4; // Analog input pin for LDR int ldrValue; // Variable to store LDR value

void setup()

{

pinMode(ldrPin,INPUT);

Serial.begin(9600); // Initialize serial communication for debugging

}

void loop()

{

int readValue;

float realValue;

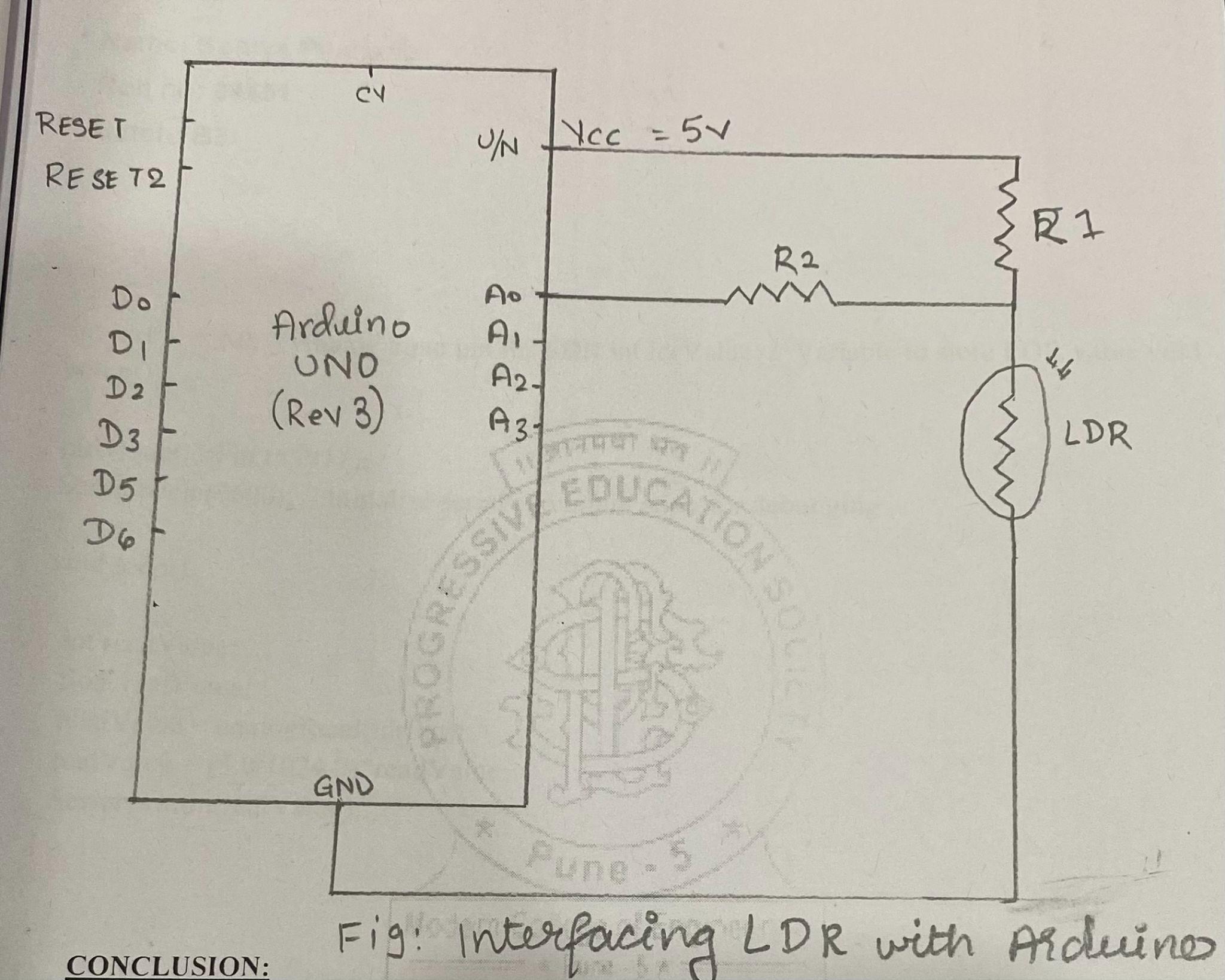
readValue = analogRead(ldrPin);

realValue = (5.0/1024.0)\*readValue;

Serial.println(realValue);

delay(1000);

}



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <LiquidCrystal.h>

LiquidCrystal lcd(6,7,2,3,4,5);   //lcd object

void setup() {

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

  lcd.begin(16, 2);

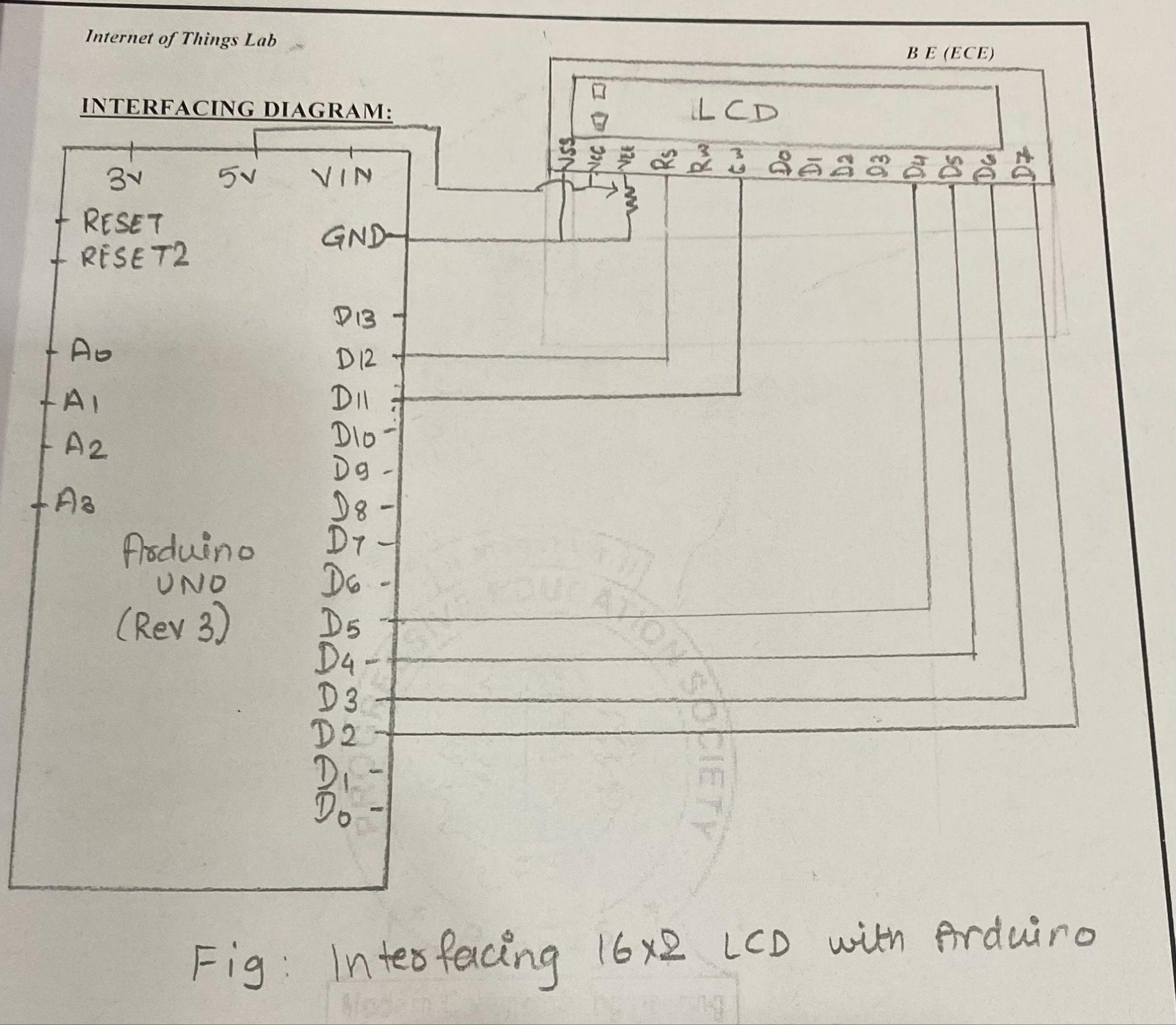
  lcd.print("BE ECE");

}

void loop() {

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

}



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LM35 TEMPERATURE SENSOR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup() {

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

Serial.begin(9600);

}

void loop() {

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

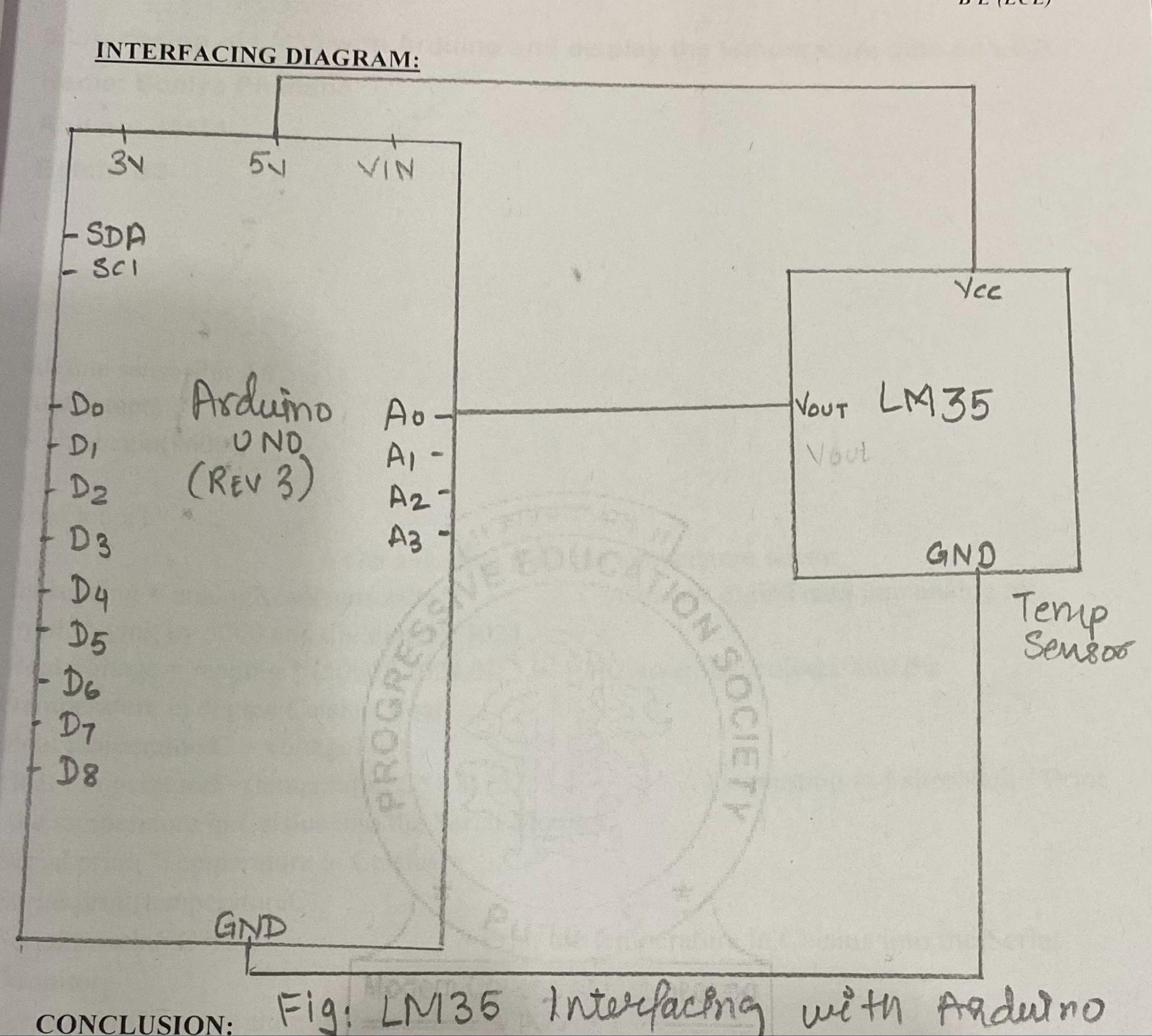
 unsigned int temp = analogRead(A0);

  temp = map(temp,0,1023,0,500);    //10mV = 1'C

  Serial.println(temp);

  delay(2000);

}



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RPi IR SENSOR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import time

import RPi.GPIO as GPIO

RUNNING = True

HIGH = 1

LOW = 0

DetectPin = 4

led = 8

def InitSystem():

GPIO.setmode(GPIO.BCM)

GPIO.setup(DetectPin,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(led,GPIO.OUT)

return

def DetectPerson():

while True:

input\_state = GPIO.input(DetectPin)

time.sleep(0.3)

if input\_state == 0:

return LOW

else:

return HIGH

try:

print ("\nCounting using IR LED\n")

print ("-----------------------------------------------\n")

InitSystem()

count =0;

while RUNNING:

state = DetectPerson()

if state == LOW:

count+=1

print ("person count =%d" %count)

GPIO.output(led,LOW)

time.sleep(1)

GPIO.output(led,HIGH)

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

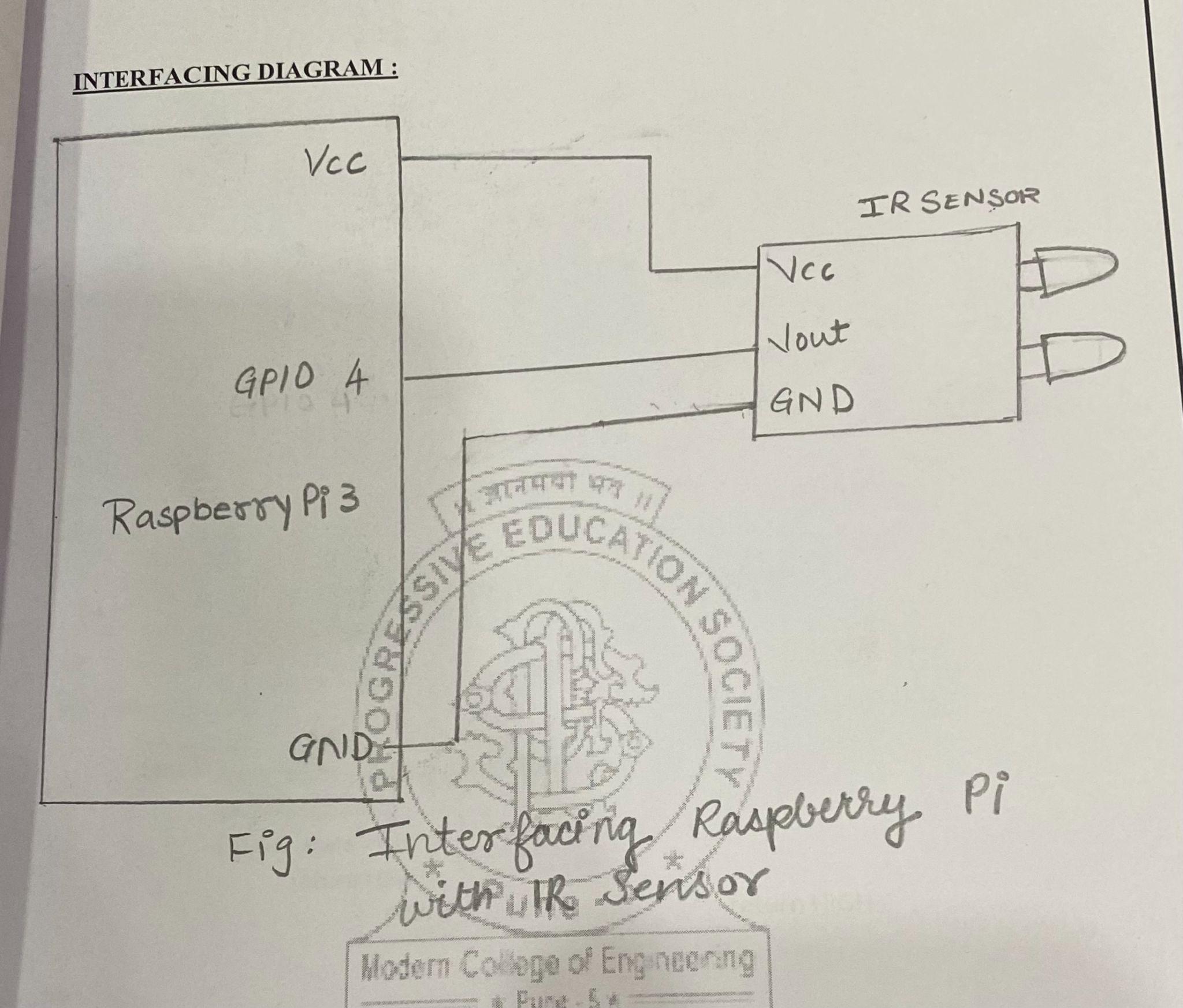
# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Rpi LED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import time

from gpiozero import LED

led1 = LED(8)

led2 = LED(10)

led3 = LED(9)

led4 = LED(11)

while True:

try:

led1.off()

time.sleep(0.5)

led1.on()

led2.off()

time.sleep(0.5)

led2.on()

led3.off()

time.sleep(0.5)

led3.on()

led4.off()

time.sleep(0.5)

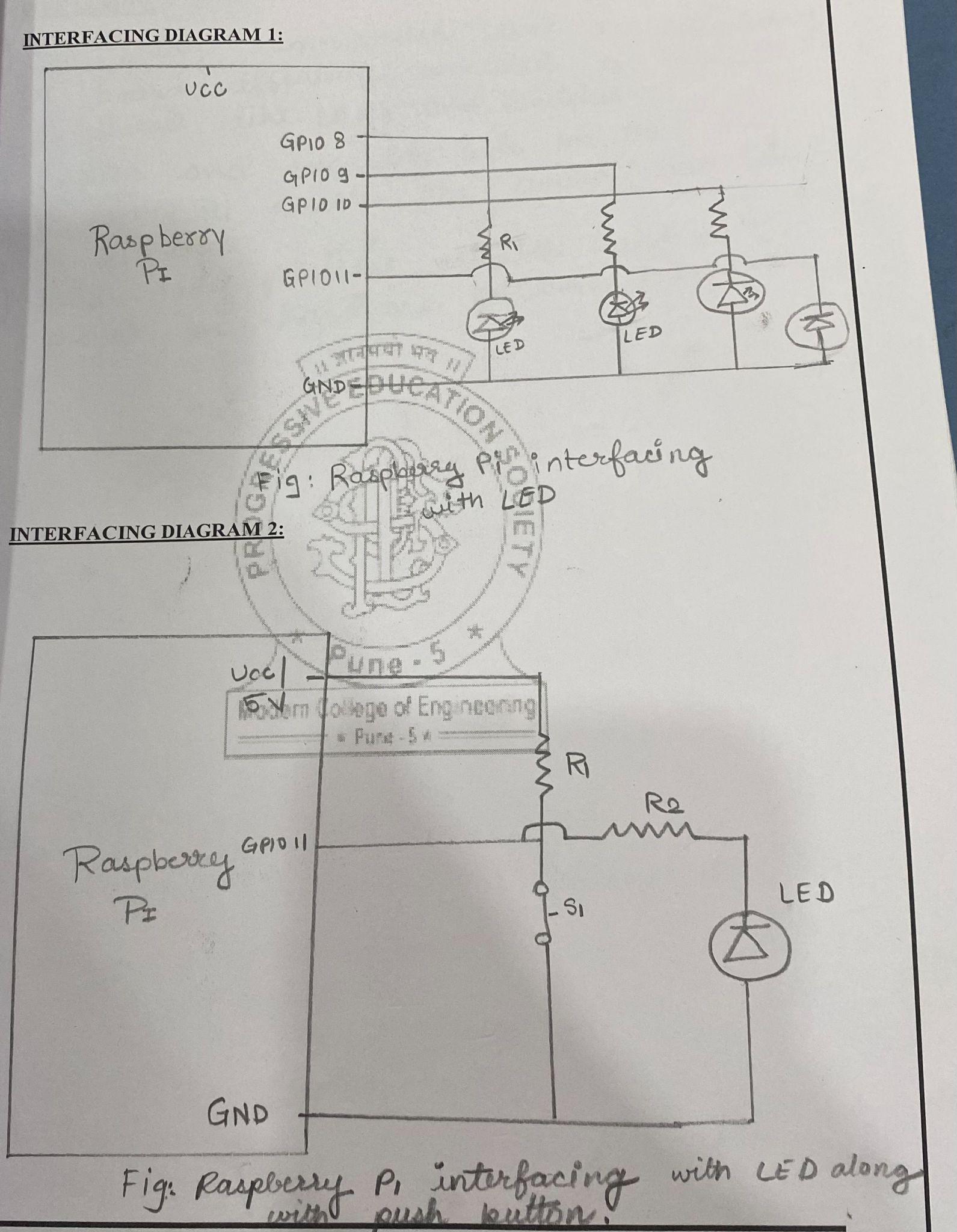
led4.on()

time.sleep(0.5)

except KeyboardInterrupt:

print("closing")

exit()



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RPI BUZZER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import time

import RPi.GPIO as GPIO

TRUE = 1

buzzer = 4

GPIO.setmode(GPIO.BCM)

GPIO.setup(buzzer,GPIO.OUT)

def buzzerState(val):

GPIO.output(buzzer,val)

try:

while TRUE:

buzzerState(1)

time.sleep(1)

buzzerState(0)

time.sleep(1)

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

print "\Quitting"

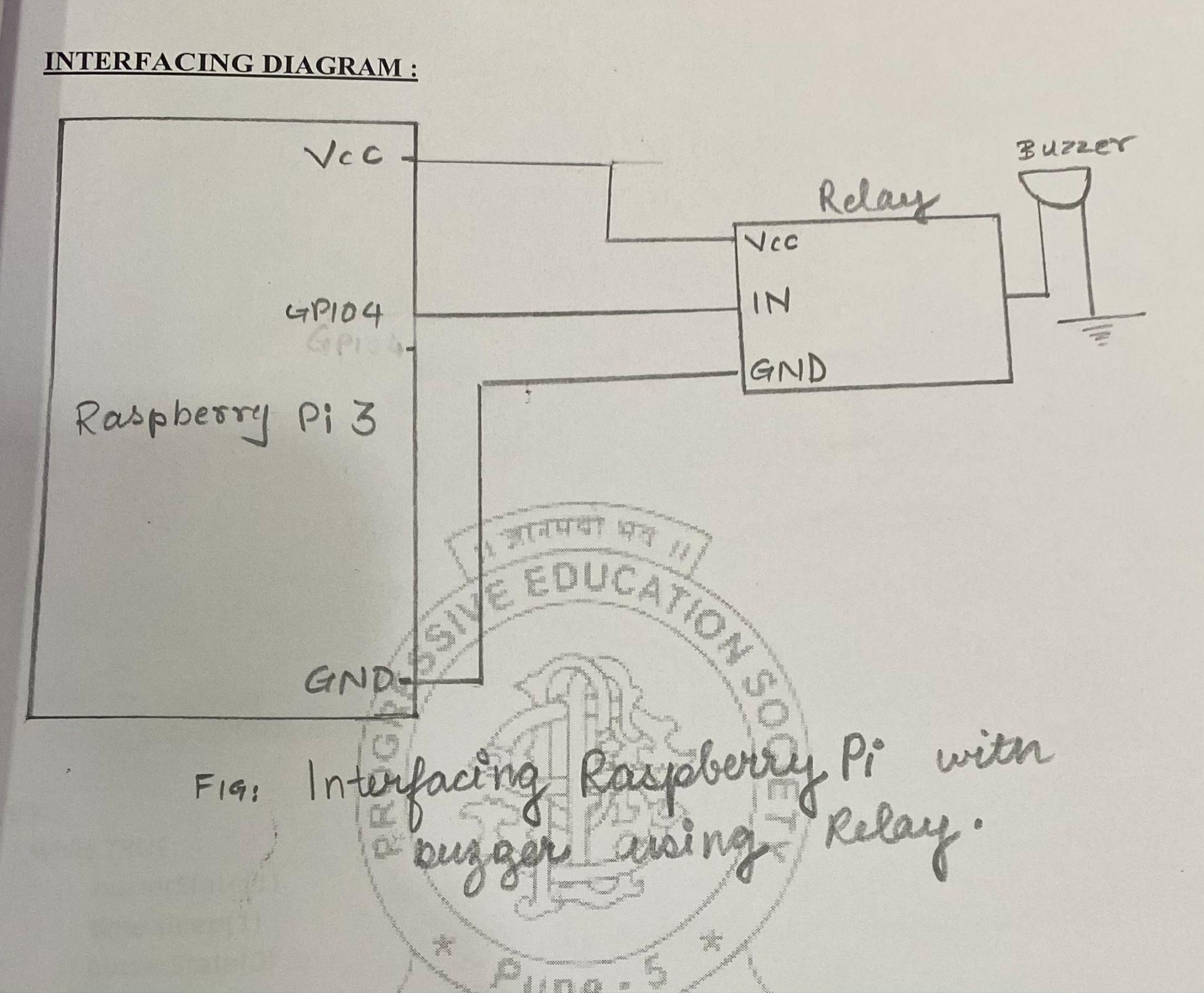
# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RPI ULTRASONIC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import RPi.GPIO as GPIO

import time

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 27

GPIO\_ECHO = 18

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

