## AIM:

To display time over 4 digit 7 segment display using raspberry pi.

## ALGORITHM:

**Step 1:** Use Python's datetime module to get the current time.

**Step 2:** Define a mapping of the digits 0-9 to their respective configurations on the 7-segment display.

**Step 3:** Continuously update the display with the current time in a loop. Split the time into hours and minutes.

**Step 4:** then convert each digit into its corresponding 7-segment display configuration using the mapping.

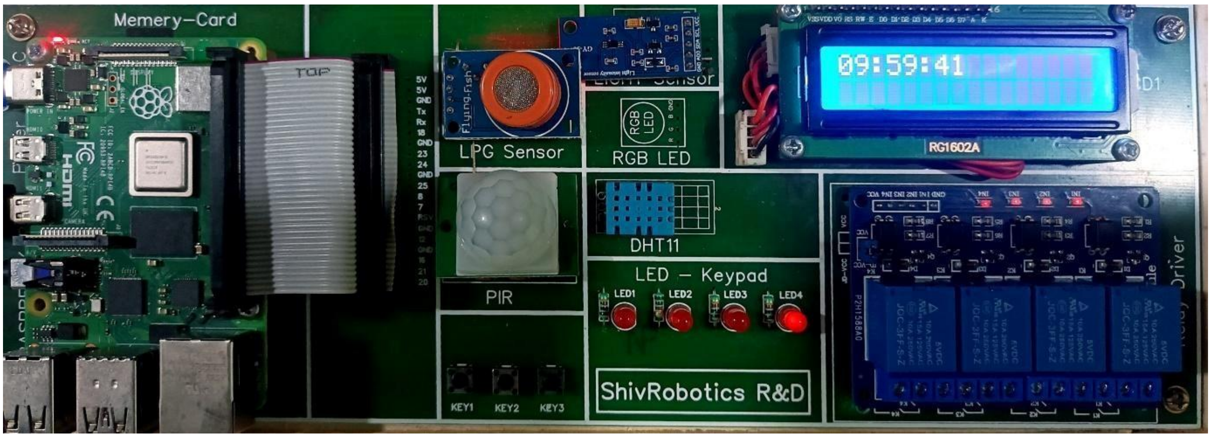
**Step 5:** Use the Raspberry Pi's GPIO pins to control the segments of the display.

**Step 6:** You'll need to set the GPIO pins to the appropriate state to light up the segments required to display each digit.

## SOURCE CODE:

```
import time
from rpi_lcd import LCD
# Create an instance of the LCD with the I2C address 0x27 lcd
= LCD(0x27)
try:     while
True:
        # Get the current time
        current_time = time.strftime("%H:%M:%S")
# Clear the LCD and display the current time
        lcd.clear()
lcd.text(current_time, 1)
        # Sleep for one second before updating the time
        time.sleep(1)
except KeyboardInterrupt:
    pass
# Clear the LCD before exiting
lcd.clear()
```

## OUTPUT:



## RESULT:

Thus to display time over 4 digit 7 segment display using raspberry pi was executed and verified successfully.

## AIM:

To make a working model of raspberry pi based oscilloscope.

## ALGORITHM:

**Step 1:** Connect the ADC to the Raspberry Pi according to its datasheet or instructions.

**Step 2:** Continuously read analog data from the ADC channels.

**Step 3:** Display the waveform as ASCII art in the terminal.

**Step 4:** Create a simple graphical interface using libraries like Tkinter or PyQt to plot the waveform in real-time.

**Step 5:** Display the waveform data on a web page using a web framework like Flask or Django.

**Step 6:** Implement triggering to stabilize the waveform display.

**Step 7:** Implement controls to adjust voltage and time scaling of the displayed waveform.

**SOURCE CODE:**

**ARDUINO CODE:**

```
// Arduino - Pi - Scope By Mike Cook int
buffer [512]; // 1K input buffer
int sample, lastSample; int
pot1, triggerVoltage;
int triggerTimeout = 1000; // time until auto trigger
unsigned long triggerStart; char triggerType = '2';
void setup(){   Serial.begin(115200);
pinMode(13,OUTPUT);
  // set up fast sampling mode
  ADCSRA = (ADCSRA & 0xf8) | 0x04; // set 16 times division
} void
loop(){
  if( triggerType != '2') trigger(); // get a trigger
digitalWrite(13,HIGH);// timing marker
for(int i=0; i<512 ; i++){     buffer[i] =
analogRead(0);
  }
  digitalWrite(13,LOW); // timing marker   pot1 =
analogRead(2); // switch channel to cursor pot   for(int
i=0; i<512 ; i++){
    Serial.write(buffer[i]>>8);
    Serial.write(buffer[i] & 0xff);
  }
  // send back pot values for cursors
pot1 = analogRead(2);   analogRead(3);
// next cursor pot
  Serial.write(pot1>>8);
Serial.write(pot1 & 0xff);   pot1
= analogRead(3);   triggerVoltage
= analogRead(4);
  Serial.write(pot1>>8);
Serial.write(pot1 & 0xff);
triggerVoltage = analogRead(4);
  pot1 = analogRead(0); // prepair for next sample run
  Serial.write(triggerVoltage>>8);
Serial.write(triggerVoltage & 0xff);
  while(Serial.available() == 0) { } // wait for next request
triggerType = Serial.read(); // see what trigger to use   while
(Serial.available() != 0) { // remove any other bytes in buffer
     Serial.read();
  } } void
trigger(){
  // trigger at rising zero crossing
triggerStart = millis();

  sample = analogRead(0);
  do {
  lastSample = sample;
sample = analogRead(0);
  }
  while(!(lastSample < triggerVoltage && sample >
triggerVoltage) && (millis() - triggerStart < triggerTimeout));
}
```

**PYTHON CODE**

```
# #!/usr/bin/env python3
# Scope - Pygame powered Oscilloscope
# By Mike Cook May 2018 import
serial, pygame, os, time
pygame.init()
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Arduino / Pi Oscilloscope")
pygame.event.set_allowed(None)pygame.event.set_allowed([pyg
ame.KEYDOWN, pygame.MOUSEBUTTONDOWN,
pygame.QUIT, pygame.MOUSEBUTTONUP])
textHeight=20 ; font = pygame.font.Font(None, textHeight)
screenWidth = 720 ; screenHight = 360 screen =
pygame.display.set_mode([screenWidth,screenHight],0,32)
display = pygame.Surface((512,256))

backCol = (150,150,100) ; black = (0,0,0) # background colours
pramCol = (200,200,150) # parameter colour logo =
pygame.image.load("images/PyLogo.png").convert_alpha()
sampleInput = serial.Serial("/dev/ttyUSB0",115200, timeout = 5)
# For Mega or nano
#sampleInput = serial.Serial("/dev/ttyACM0",115200, timeout =
5) # For Uno
displayWidth = 512 ; displayHight = 256 LedRect = [
pygame.Rect((0,0),(0,0))]*17 inBuf = [0]*512 # quick
way of getting a 512 long buffer chOff = displayHight//2
# Channel Offset run = [True,False,False,True,False] #
run controls expandT = 1 ; expandV = 1 # voltage &
time expansion sampleTime = 17 # uS for 58KHz
sample smples_cm = 32 * sampleTime volts_sample =
5/1024 # volts per sample
measureTime = False ; measureVolts = False;savedTime =
0;savedVoltage = 0
cursorT = 0; cursorV = 0; vMag = 1; svLed = False; stLed = False
triggerC = 512 ; savedVoltsC = -1 ; savedTimeC = -1 def main():
pygame.draw.rect(screen,backCol,(0,0,screenWidth,screenHight+
2),0)
   defineControls()
   drawControls()
time.sleep(0.1)
   sampleInput.flushInput() # empty any buffer contents
sampleInput.write(b'2') # tell Arduino to get a new buffer
while(1):
     time.sleep(0.001) # let other code have a look in
readArduino() # get buffer data       plotWave() #
draw waveform                 if measureTime or
measureVolts :          updateControls(True)
drawScope() # display new screen
checkForEvent()
     while run[4]: # if in hold mode wait here
checkForEvent()       if run[3]:
       sampleInput.write(b'1') # tell Arduino to get an other buffers
     else:
          sampleInput.write(b'2') # buffer but no trigger            def
drawGrid():
pygame.draw.rect(display,(240,240,240),(0,0,displayWidth,displa
yHight),0)
  for h in range(32,256,32): # draw horizontal
     pygame.draw.line(display,(120,120,120),(0,h),(512,h),1)
for v in range(32,512,32): # draw vertical
pygame.draw.line(display,(120,120,120),(v,0),(v,256),1)
pygame.draw.line(display,(0,0,0),(256,0),(256,256),1)
pygame.draw.line(display,(0,0,0),(0,128),(512,128),1) def
drawControls():
  drawWords("Time Magnify",10,300,black,backCol)
drawWords("Voltage Magnify",220,300,black,backCol)
drawWords("Measure",440,300,black,backCol)
drawWords("Time",440,320,black,backCol)
drawWords("Volts",486,320,black,backCol)
drawWords("Save",540,300,black,backCol)
drawWords("Time",540,320,black,backCol)
drawWords("Volts",586,320,black,backCol)
  drawWords("1/"+chr(0x394)+"Time",540,257,black,backCol)
drawWords(chr(0x394)+"Time",540,237,black,backCol)
drawWords("Saved Time",540,217,black,backCol)
drawWords("Time",540,197,black,backCol)
drawWords(chr(0x394)+"Voltage",540,167,black,backCol)
drawWords("Saved Voltage",540,147,black,backCol)
drawWords("Voltage",540,127,black,backCol)
drawWords("Run Single Freeze Trigger",540,77,black,backCol)
screen.blit(logo,(540,2))     updateControls(True)
def updateControls(blank):
  global vDisp
if blank:
   pygame.draw.rect(screen,backCol,resultsRect,0)
if expandT*smples_cm >= 1000:
      drawWords("Time "+str((expandT*smples_cm)//1000)+"mS
per division   ",10,280,black,backCol)
   else:
       drawWords("Time "+str(expandT*smples_cm)+"uS per
division    ",10,280,black,backCol)
   volts_cm = int(volts_sample*128*1000/expandV)
drawWords("Voltage "+str(volts_cm)+"mV per
division",220,280,black,backCol)     for n in
range(0,6): # time option LED
      drawWords("x"+str(1<<n),10+n*30,320,black,backCol)
drawLED(n,expandT == 1<<n)     for n in range(6,9): #
voltage options

drawWords("x"+str(1<<(n-6)),220+(n-6)*30,320,black,backCol)
drawLED(n,expandV == 1<<(n-6))
drawLED(9,measureTime)     drawLED(10,measureVolts)
drawLED(11,stLed)     drawLED(12,svLed)
   for n in range(13,17):
      drawLED(n,run[n-13])
if measureTime :
      t = (cursorT>>1)*sampleTime / expandT
drawWords(" "+trunk(t,5)+"
"+chr(0x3bc)+"S",640,197,black,pramCol) # current time
drawWords(" "+trunk(savedTime,5)+"
"+chr(0x3bc)+"S",640,217,black,pramCol)
drawWords(" "+trunk(t-savedTime,5)+"
"+chr(0x3bc)+"S",640,237,black,pramCol) # delta time
if t-savedTime != 0 :
        drawWords((trunk(1000000 / abs(t-savedTime),5))+"
Hz",640,257,black,pramCol)     if measureVolts :
     vDisp = (((1024-cursorV)>>2)-128)*volts_sample * vMag
delta = vDisp - savedVoltage
     drawWords(" "+trunk(delta,4)+" V",640,167,black,pramCol)
drawWords(" "+trunk(savedVoltage,4)+"
V",640,147,black,pramCol)
     drawWords(" "+trunk(vDisp,4)+" V",640,127,black,pramCol)
def trunk(value, place): # truncate a value string
v=str(value)+"000000"     if value>0:
     v = v[0:place]
   else:
     v = v[0:place+1] # extra place for the minus sign
   return v
def drawLED(n,state): # draw LED
   if state :
      pygame.draw.rect(screen,(240,0,0),LedRect[n],0)
   else :
      pygame.draw.rect(screen,(240,240,240),LedRect[n],0) def
defineControls():
  global LedRect, resultsRect
for n in range(0,6):
      LedRect[n] = pygame.Rect((10+n*30,336),(15,15))
for n in range(6,9):
      LedRect[n] = pygame.Rect((220+(n-6)*30,336),(15,15))
  LedRect[9] = pygame.Rect((440,336),(15,15))  # time
  LedRect[10] = pygame.Rect((486,336),(15,15)) # volts
  LedRect[11] = pygame.Rect((540,336),(15,15)) # save time
  LedRect[12] = pygame.Rect((586,336),(15,15)) # save volts
  LedRect[13] = pygame.Rect((545,100),(15,15)) # run
  LedRect[14] = pygame.Rect((580,100),(15,15)) # single
  LedRect[15] = pygame.Rect((628,100),(15,15)) # freeze
LedRect[16] = pygame.Rect((676,100),(15,15)) # trigger
resultsRect = pygame.Rect((639,125),(90,153))

def plotWave():     global vMag
lastX=0 ; lastY=0     vMag = 2 #
adjust voltage scale     if expandV
== 1:
      vMag = 4     if
expandV == 4:
      vMag =1     drawGrid()     s = 0 #
sample pointer     for n in range(0,
displayWidth, expandT):
        y = (512-inBuf[s])//vMag + chOff
        if n != 0:
            pygame.draw.line(display,(0,200,0),(lastX ,lastY), (n,y
,y),2)        lastX =
n        lastY = y
s += 1     if
measureTime :
      pygame.draw.line(display,(0,0,255),(cursorT>>1,0),
(cursorT>>1,256),1)
if savedTimeC != -1:
for n in range(0,256,12):
pygame.draw.line(display,(0,0,255),(savedTimeC,n),(savedTimeC
,n+6),1)     if
measureVolts :
      pygame.draw.line(display,(255,0,0),(0,cursorV>>2),
(512,cursorV>>2),1)
if savedVoltsC != -1:
        for n in range(0,512,12):
pygame.draw.line(display,(255,0,0),(n,savedVoltsC),(n+6,savedV
oltsC),1)
   if run[3] : # use trigger
      y = (triggerC-512)//vMag + chOff
for n in range(0,512,12):
         pygame.draw.line(display,(255,128,0),(n,y),(n+6,y),1)
def drawScope(): # put display onto scope controls
```

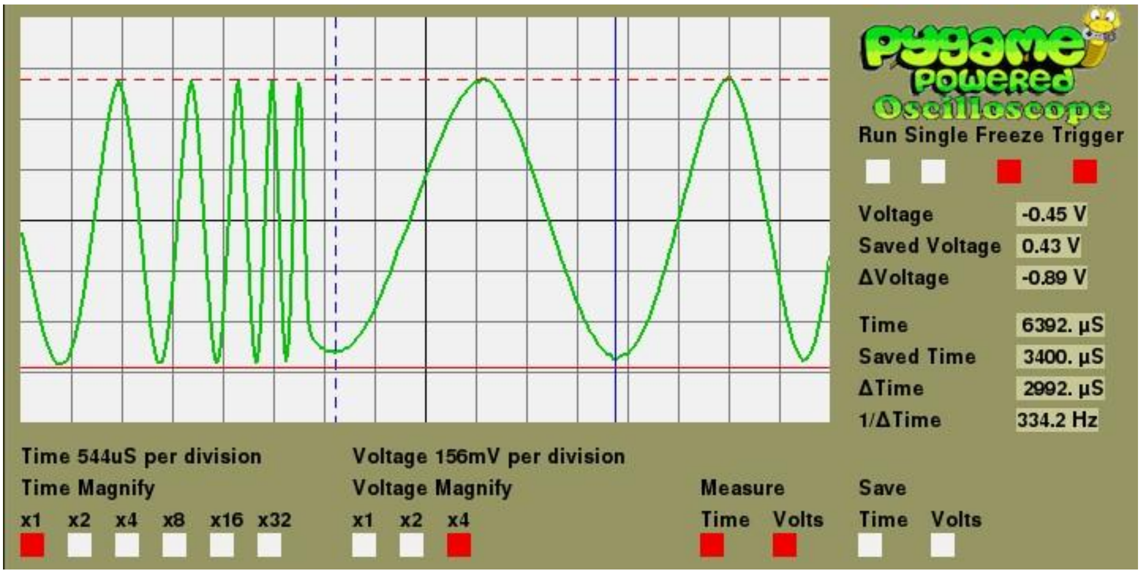
```
screen.blit(display,(10,10))     pygame.display.update() def
drawWords(words,x,y,col,backCol) :
   textSurface = font.render(words, True, col, backCol)
textRect = textSurface.get_rect()     textRect.left = x
textRect.top = y
   screen.blit(textSurface, textRect) def
readArduino(): # get buffer and controls
global cursorT, cursorV, triggerC, run     if
run[2] : #if in freeze mode funnel data into
junk        for i in range(0,1024):
        junk = sampleInput.read()
else: # otherwise read into the buffer
for i in range(0,512):
         inBuf[i] = ((ord(sampleInput.read())) << 8) |
ord(sampleInput.read())
   cursorT = ((ord(sampleInput.read())) << 8) |
ord(sampleInput.read())
   cursorV = 1024 - (((ord(sampleInput.read())) << 8) |
ord(sampleInput.read()))
   triggerC = 1024 - (((ord(sampleInput.read())) << 8) |
ord(sampleInput.read()))      if run[1]: #single sweep
requested       run[1] = False
     run[2] = True # put in freeze mode
updateControls(True)
```

```
def handleMouse(pos): # look at mouse down
  global
expandT,expandV,measureTime,measureVolts,svLed,stLed
global savedVoltsC, savedTimeC, run
  #print(pos)
  for n in range(0,6) :       if
LedRect[n].collidepoint(pos):
        expandT = 1<<n    for n in
range(6,9) :       if
LedRect[n].collidepoint(pos):
        expandV = 1<<(n-6)
  if LedRect[9].collidepoint(pos): #toggle time measurement
measureTime = not(measureTime)        if not measureTime :
savedTimeC = -1      if LedRect[10].collidepoint(pos):
      measureVolts = not(measureVolts) # toggle volts
measurement
      if not measureVolts :
savedVoltsC = -1
  if LedRect[11].collidepoint(pos) and measureTime: # save time
stLed = True        savedTimeC = cursorT>>1
  if LedRect[12].collidepoint(pos) and measureVolts: # save volts
svLed = True        savedVoltsC = cursorV>>2    # run controls
logic
```

```
  if LedRect[13].collidepoint(pos) and not run[1]: # run
     run[0] = not(run[0])
if not run[0]:
run[2] = True       else:
        run[2] = False
  if LedRect[14].collidepoint(pos): # single
run[1] = True         run[0] = False
run[2] = False         run[4] = True
updateControls(False)         drawScope()
  if LedRect[15].collidepoint(pos) and not run[1]: # freeze
run[2] = not(run[2])       if not run[2]:           run[0] =
True       else:
        run[0] = False
  if LedRect[16].collidepoint(pos): # trigger
run[3] = not(run[3])          updateControls(False)
def handleMouseUp(pos): # look at mouse up
global savedVoltage,savedTime, svLed, stLed, run
if LedRect[12].collidepoint(pos) and measureVolts:
     savedVoltage = vDisp       svLed = False
updateControls(False)    if
LedRect[11].collidepoint(pos) and measureTime:
savedTime = (cursorT>>1)*sampleTime / expandT
stLed = False       updateControls(False)    if
```

```
LedRect[14].collidepoint(pos): # single        run[4] =
False        updateControls(False) def terminate(): #
close down the program     pygame.quit() # close
pygame     os._exit(1)
def checkForEvent(): # see if we need to quit
event = pygame.event.poll()     if event.type
== pygame.QUIT :
        terminate()
   if event.type == pygame.KEYDOWN :
if event.key == pygame.K_ESCAPE :
         terminate()
      if event.key == pygame.K_s : # screen dump
os.system("scrot -u")

   if event.type == pygame.MOUSEBUTTONDOWN :
handleMouse(pygame.mouse.get_pos())     if
event.type == pygame.MOUSEBUTTONUP :
handleMouseUp(pygame.mouse.get_pos())      # Main
program logic: if _name_ == '_main_':
   main()
```

## OUTPUT



Run □ Single □ Freeze ■ Trigger ■

Voltage -0.45 V
Saved Voltage 0.43 V
ΔVoltage -0.89 V

Time 6392. µS
Saved Time 3400. µS
ΔTime 2992. µS
1/ΔTime 334.2 Hz

Time 544uS per division
Time Magnify
x1 ■ x2 □ x4 □ x8 □ x16 □ x32 □

Voltage 156mV per division
Voltage Magnify
x1 □ x2 □ x4 ■

Measure
Time ■ Volts ■

Save
Time □ Volts □

## RESULT

Thus the raspberry pi based oscilloscope was executed and verified successfully.

## AIM:

To setting up wirless access point using raspberry pi.

## ALGORITHM:

**Step 1:** Install Hostapd and dnsmasq.

**Step 2:** Configure Static IP for the Wireless Interface.

**Step 3:** Configure Hostapd.

**Step 4:** Update Hostapd Configuration File

**Step 5:** Configure dnsmasq.

**Step 6:** Enable IP Forwarding.

**Step 7:** Start Services and Enable at Boot.

**Step 8:** Reboot the Raspberry Pi.

## SOURCE CODE

```
from flask import Flask, render_template
import RPi.GPIO as GPI
app = Flask(_name_) #
Set up GPIO pins
relay_pin = 12 light_pin
= 6
GPIO.setmode(GPIO.BCM)  # Use BCM GPIO numbering
GPIO.setup(relay_pin, GPIO.OUT)
GPIO.setup(light_pin, GPIO.OUT)
# Define the initial status of the relay and light
relay_status = "Off" light_status = "Off"
# Home page to display control options
@app.route("/") def
index():
    return render_template("index.html", relay_status=relay_status,
light_status=light_status)
# Route to control the relay
@app.route("/control-relay/<action>")
def control_relay(action):     global
relay_status     if action == "On":
        GPIO.output(relay_pin, GPIO.HIGH)  # Turn on the relay
        relay_status = "On"
elif action == "Off":
        GPIO.output(relay_pin, GPIO.LOW)   # Turn off the relay
        relay_status = "Off"
```

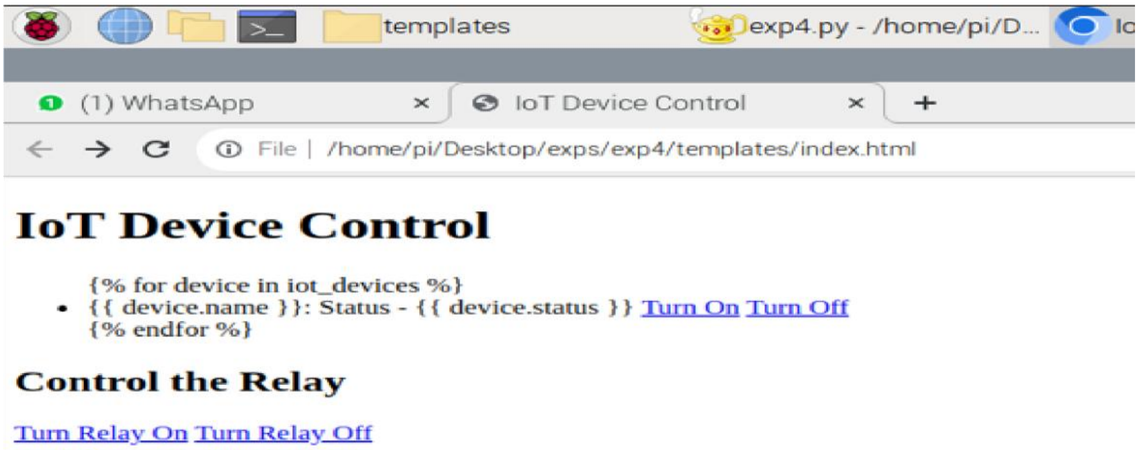
```
   return "OK"
# Route to control the light
@app.route("/control-light/<action>")
def control_light(action):     global
light_status     if action == "On":
        GPIO.output(light_pin, GPIO.HIGH)  # Turn on the light
        light_status = "On"
elif action == "Off":
        GPIO.output(light_pin, GPIO.LOW)   # Turn off the light
        light_status = "Off"
return "OK" if _name_
== "_main_":
   app.run(host="0.0.0.0", port=8000)
```

## HTML CODE:

```
<!DOCTYPE html>
<html>
<head>
   <title>IoT Device Control</title>
</head>
<body>
   <h1>IoT Device Control</h1>
   <ul>
      {% for device in iot_devices %}
      <li>
         {{ device.name }}: Status - {{ device.status }}
```

```
            <a href="/control/{{ device.id }}/On">Turn On</a>
            <a href="/control/{{ device.id }}/Off">Turn Off</a>
        </li>
        {% endfor %}
    </ul>
    <h2>Control the Relay</h2>
    <a href="/control-relay/On">Turn Relay On</a>
    <a href="/control-relay/Off">Turn Relay Off</a>
    <h3>Control the light</h3>
    <a href="/control-light/On">Turn Light On</a>
    <a href="/control-light/Off">Turn Light Off</a>
</body> </html>
```

## OUTPUT

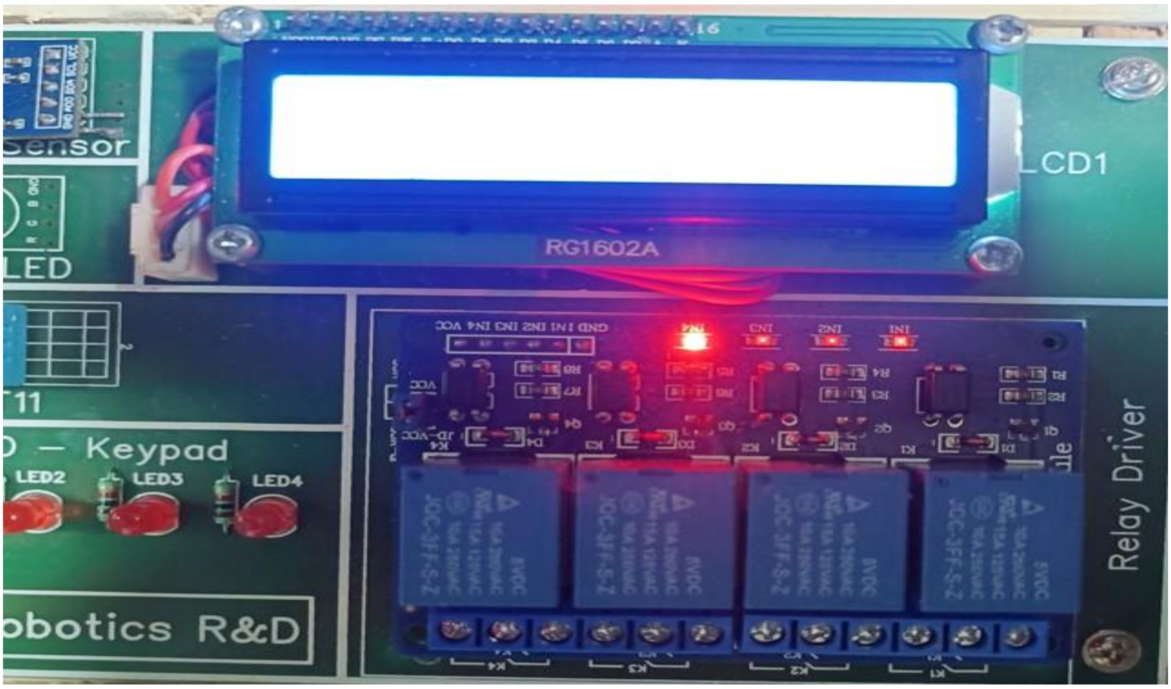


# IoT Device Control

{% for device in iot_devices %}
- {{ device.name }}: Status - {{ device.status }} Turn On Turn Off
{% endfor %}

## Control the Relay

Turn Relay On Turn Relay Off



## RESULT

Thus to setup wirless access point using raspberry pi was executed and verified successfully.

## AIM

To make a fingerprint sensor interfacing with raspberry pi.

## ALGORITHM

**Step 1:** Connect the fingerprint sensor module to the Raspberry Pi.

**Step 2:** Check the datasheet or documentation of the fingerprint sensor for specific wiring instructions.

**Step 3:** Search for and install Python libraries compatible with your fingerprint sensor.

**Step 4:** Use pip to install the required libraries.

**Step 5:** Import necessary libraries in your Python script.

**Step 6:** Implement functions to enroll fingerprints, verify fingerprints, and perform other actions supported by the sensor.

**Step 7:** Run the Python script on your Raspberry Pi.

## SOURCE CODE

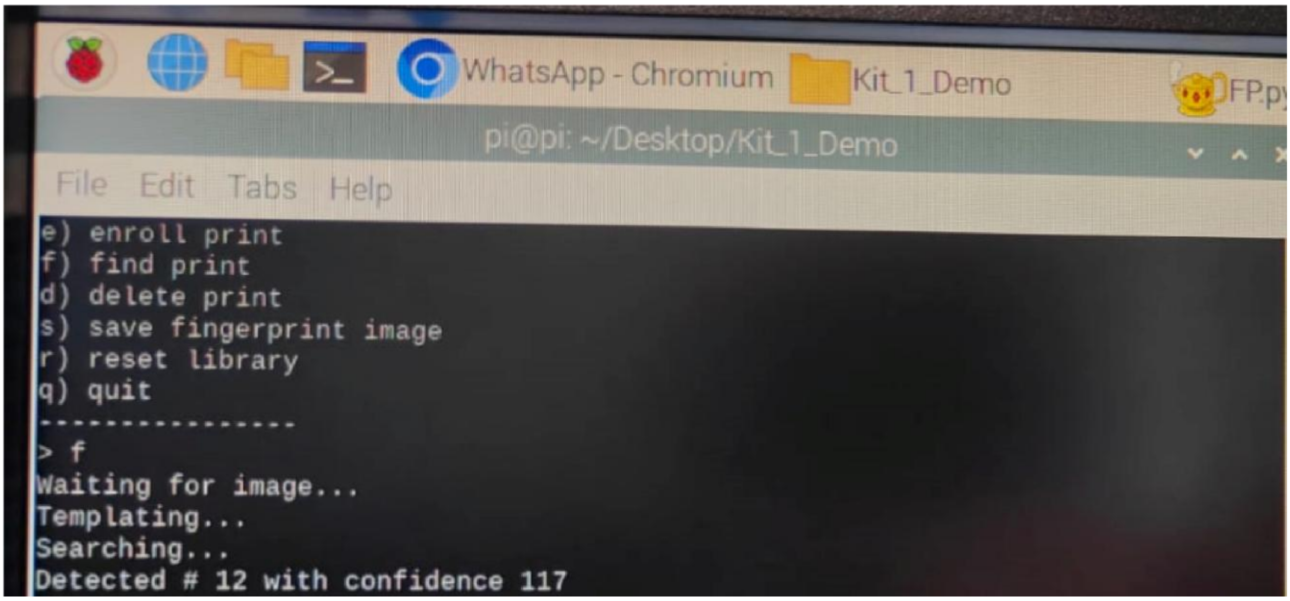
```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
import time import serial import
adafruit_fingerprint from rpi_lcd
import LCD lcd = LCD(0x27)
# import board
# uart = busio.UART(board.TX, board.RX, baudrate=57600)
# If using with a computer such as Linux/RaspberryPi, Mac, Windows with
USB/serial converter:
#uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)
# If using with Linux/Raspberry Pi and hardware UART: uart
= serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
# If using with Linux/Raspberry Pi 3 with pi3-disable-bt
# uart = serial.Serial("/dev/ttyAMA0", baudrate=57600, timeout=1)
finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)
lcd.text("FingerPrint Test", 1) lcd.text("Use Console", 2)
##################################################
def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
print("Waiting for image...")     while finger.get_image() !=
adafruit_fingerprint.OK:
        pass
    print("Templating...")     if finger.image_2_tz(1) !=
adafruit_fingerprint.OK:
        return False     print("Searching...")     if
finger.finger_search() != adafruit_fingerprint.OK:
        return False
return True
# pylint: disable=too-many-branches def
get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
print("Getting image...", end="")     i = finger.get_image()     if i ==
adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")         elif i ==
adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
return False


    print("Templating...", end="")
i = finger.image_2_tz(1)     if i ==
adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")         elif i ==
adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
return False
print("Searching...", end="")     i
= finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
if i == adafruit_fingerprint.OK:         print("Found
fingerprint!")         return True     else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
        else:
            print("Other error")
return False
# pylint: disable=too-many-statements def
enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
for fingerimg in range(1, 3):         if fingerimg == 1:
            print("Place finger on sensor...", end="")
        else:
            print("Place same finger again...", end="")
while True:

            i = finger.get_image()             if
i == adafruit_fingerprint.OK:
                print("Image taken")
                break
```

```
          if i == adafruit_fingerprint.NOFINGER:
print(".", end="")
          elif i == adafruit_fingerprint.IMAGEFAIL:
             print("Imaging error")
return False            else:
             print("Other error")
return False
      print("Templating...", end="")
i = finger.image_2_tz(fingerimg)
if i == adafruit_fingerprint.OK:
          print("Templated")
      else:
          if i == adafruit_fingerprint.IMAGEMESS:
             print("Image too messy")            elif i ==
adafruit_fingerprint.FEATUREFAIL:
             print("Could not identify features")
          elif i == adafruit_fingerprint.INVALIDIMAGE:
             print("Image invalid")
          else:
             print("Other error")
          return False        if
fingerimg == 1:
print("Remove finger")
time.sleep(1)
          while i != adafruit_fingerprint.NOFINGER:
             i = finger.get_image()
```

```
    print("Creating model...", end="")
i = finger.create_model()     if i ==
adafruit_fingerprint.OK:
        print("Created")
    else:
        if i == adafruit_fingerprint.ENROLLMISMATCH:
            print("Prints did not match")
        else:
            print("Other error")
return False
    print("Storing model #%d..." % location, end="")
i = finger.store_model(location)     if i ==
adafruit_fingerprint.OK:
        print("Stored")
else:
        if i == adafruit_fingerprint.BADLOCATION:
print("Bad storage location")         elif i ==
adafruit_fingerprint.FLASHERR:
            print("Flash storage error")
        else:
            print("Other error")
return False     return True
def save_fingerprint_image(filename):
    """Scan fingerprint then save image to filename."""
while finger.get_image():
        pass
    # let PIL take care of the image headers and file structure      from
PIL import Image  # pylint: disable=import-outside-toplevel      img
= Image.new("L", (256, 288), "white")      pixeldata = img.load()
mask = 0b00001111      result =
finger.get_fpdata(sensorbuffer="image")


    # this block "unpacks" the data received from the fingerprint
    #   module then copies the image data to the image placeholder "img"
    #   pixel by pixel.  please refer to section 4.2.1 of the manual for
    #   more details.  thanks to Bastian Raschke and Danylo Esterman.
    # pylint: disable=invalid-name
x = 0
    # pylint: disable=invalid-name
y = 0
    # pylint: disable=consider-using-enumerate
    for i in range(len(result)):
        pixeldata[x, y] = (int(result[i]) >> 4) * 17
        x += 1          pixeldata[x, y] = (int(result[i])
& mask) * 17          if x == 255:            x = 0
y += 1          else:
            x += 1
    if not img.save(filename):
        return True
return False
```

```
########################################################## def
get_num(max_number):
   """Use input() to get a valid number from 0 to the maximum size
   of the library. Retry till success!"""
   i = -1
   while (i > max_number - 1) or (i < 0):
      try:
         i = int(input("Enter ID # from 0-{}: ".format(max_number - 1)))
except ValueError:
         pass
return i
while True:
   print("----------------")
   if finger.read_templates() != adafruit_fingerprint.OK:
      raise RuntimeError("Failed to read templates")
print("Fingerprint templates: ", finger.templates)     if
finger.count_templates() != adafruit_fingerprint.OK:
      raise RuntimeError("Failed to read templates")
print("Number of templates found: ", finger.template_count)
if finger.read_sysparam() != adafruit_fingerprint.OK:
raise RuntimeError("Failed to get system parameters")
print("Size of template library: ", finger.library_size)
print("e) enroll print")     print("f) find print")     print("d)
delete print")     print("s) save fingerprint image")     print("r)
reset library")     print("q) quit")     print("----------------")     c
= input("> ")     if c == "e":
    enroll_finger(get_num(finger.library_size))
  if c == "f":        if
get_fingerprint():
      print("Detected #", finger.finger_id, "with confidence",
finger.confidence)
    else:           print("Finger
not found")    if c == "d":

    if finger.delete_model(get_num(finger.library_size)) ==
adafruit_fingerprint.OK:
print("Deleted!")
    else:
      print("Failed to delete")    if c == "s":
if save_fingerprint_image("fingerprint.png"):
      print("Fingerprint image saved")
    else:           print("Failed to save fingerprint image")
if c == "r":        if finger.empty_library() ==
adafruit_fingerprint.OK:
      print("Library empty!")
    else:           print("Failed to
empty library")    if c == "q":
    print("Exiting fingerprint example program")
raise SystemExit
```

## OUTPUT



```
e) enroll print
f) find print
d) delete print
s) save fingerprint image
r) reset library
q) quit
----------------
> f
Waiting for image...
Templating...
Searching...
Detected # 12 with confidence 117
```

## RESULT

Thus to make a fingerprint sensor interfacing with raspberry pi was executed and verified successfully.

## AIM

To interfacing raspberry pi gps module.

## ALGORITHM

**Step 1:** Connect the GPS module to the Raspberry Pi.

**Step 2:** Enable the serial port on the Raspberry Pi using raspi-config

**Step 3:** Install Python libraries compatible with your GPS module.

**Step 4:** Import necessary libraries in your Python script.

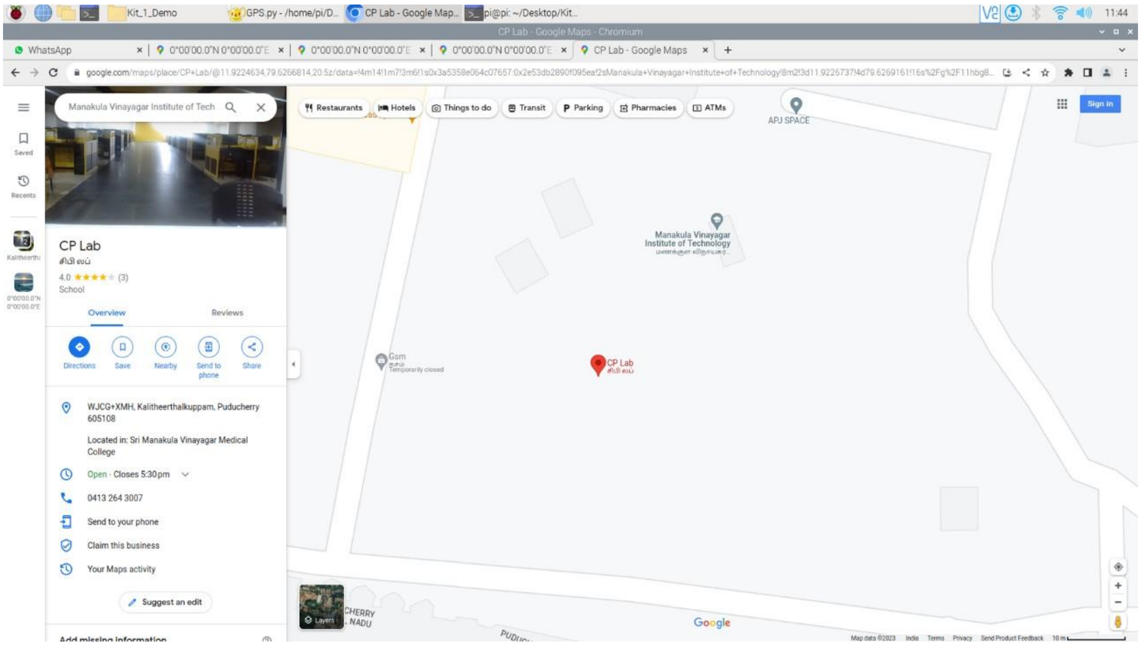
**Step 5:** Initialize the GPS module object and configure it according to the documentation.

**Step 6:** Run the Python script on your Raspberry Pi.

## SOURCE CODE

```
import time
import serial
import pynmea2
import webbrowser
port = "/dev/ttyS0" ser = serial.Serial(port,
baudrate=9600, timeout=0.5)
dataout = pynmea2.NMEAStreamReader()
# Initialize the Google Maps URL map_url
= "https://www.google.com/maps" while
True:
    newdata = ser.readline().decode('utf-8')
if newdata.startswith('$GPGGA'):
        try:
            newmsg = pynmea2.parse(newdata)             newlat
= newmsg.latitude             newlong = newmsg.longitude
print(f"Latitude: {newlat}, Longitude: {newlong}")
# Update the Google Maps URL and refresh the tab
updated_map_url =
f"{map_url}/place/{newlat},{newlong}"
webbrowser.open(updated_map_url, new=2)         except
pynmea2.ParseError as e:
            print(f"Error parsing NMEA sentence: {e}")
time.sleep(3)
```

# OUTPUT



CP Lab
சிபி லப்
4.0 ★★★★★ (3)
School

WJCG+XMH, Kalitheerthalkuppam, Puducherry 605108

Located in: Sri Manakula Vinayagar Medical College

Open · Closes 5:30 pm

0413 264 3007

Send to your phone

Claim this business

Your Maps activity

Suggest an edit

Add missing information

# RESULT

Thus To interfacing raspberry pi gps module was executed and verified successfully.

## AIM

To IoT basaed web controlled home automation using raspberrry pi.

## ALGORITHM

**Step 1:** Decide on the devices you want to control and the sensors you want to use.

**Step 2:** Connect the sensors and actuators to the Raspberry Pi GPIO pins or using suitable interfaces.

**Step 3:** Use Flask or Django to create a web application that serves as the user interface for controlling home devices.

**Step 4:** Write Python scripts to read data from sensors connected to the Raspberry Pi.

**Step 5:** Use GPIO control or any suitable method to switch relays or trigger devices based on user commands.

**Step 6:** Implement security measures for authentication and authorization to prevent unauthorized access to the control system.

**Step 7:** Deploy the system in your home environment and continually improve based on feedback and new requirements.

## SOURCE CODE

## ARDUINO CODE:

```
const express = require('express');

const app = express(); const Gpio
```

```
= require('onoff').Gpio; const
RELAY_PIN = 13; const port =
3000;
const relay = new Gpio(RELAY_PIN, 'out');
app.use(express.static('public'));
app.get('/toggle_relay', (req, res) => {
    try {
        const currentValue = relay.readSync();
const newValue = currentValue === 0 ? 1 : 0;
relay.writeSync(newValue);
res.send(newValue.toString());
    } catch (error) {
        console.error('Error toggling relay:', error);
res.status(500).send('Internal Server Error');
    }
});
app.listen(port, () => {
    console.log(Server is running on port ${port});
});

process.on('SIGINT', () => {
relay.unexport();
process.exit();
});
```

**HTML CODE:**

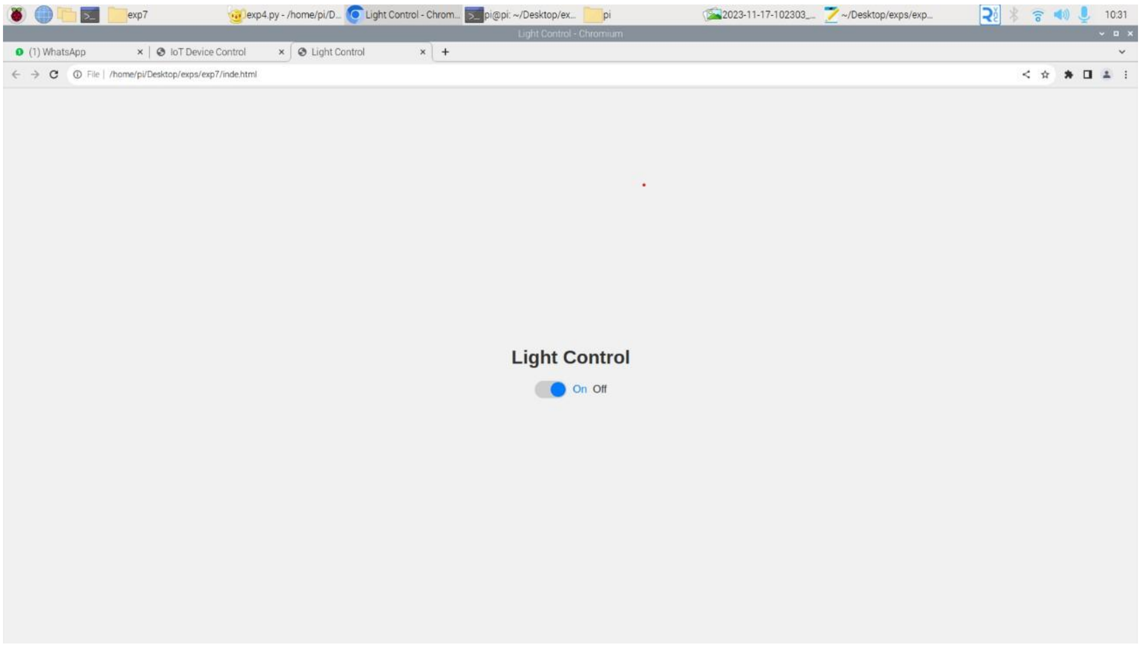
```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Light Control</title>
   <style>
body {
          font-family: Arial, sans-serif;
background-color: #f0f0f0;
margin: 0;           padding: 0;
          display: flex;           flex-
direction: column;           justify-
content: center;           align-
items: center;           height:
100vh;
       }
        h1 {
color: #333;
       }
       .toggle-container {
display: flex;           align-
items: center;
```

```
        }
        .toggle-button {
width: 50px;            height:
25px;            border: 2px solid
#ccc;            border-radius:
25px;            background-
color: #ccc;            display:
flex;            align-items:
center;            cursor: pointer;
        }
        .toggle-switch {            width: 25px;
height: 25px;            border-radius: 50%;
background-color: #007bff;
transition: transform 0.3s ease-in-out;
        }
        .on-text, .off-text {
font-size: 18px;            margin-
left: 10px;
        }
    </style>
</head>
<body>
    <h1>Light Control</h1>
```

```
  <div class="toggle-container">
    <div class="toggle-button" id="toggleButton"
onclick="toggleRelay()">
      <div class="toggle-switch" id="toggleSwitch"></div>
    </div>
    <span class="on-text" id="onText">On</span>
    <span class="off-text" id="offText">Off</span>
  </div>
<script>
    let relayState = 0; // Initial state is off
function toggleRelay() {
      relayState = 1 - relayState; // Toggle relay state
updateToggleUI();
      fetch(http://localhost:3000/toggle_relay)
        .then(response => {
if (!response.ok) {
            throw new Error('Network response was not
ok');
          }
          return response.text();
        })
        .then(data => {
          const relayStatus = data === '1' ? 'On' : 'Off';
alert(Relay is now ${relayStatus});
```

```
            })
            .catch(error => {
                alert('Error: ' + error.message);
            });
        }
        function updateToggleUI() {
            const toggleSwitch =
document.getElementById('toggleSwitch');
            const onText = document.getElementById('onText');
const offText = document.getElementById('offText');

            if (relayState === 1) {
                toggleSwitch.style.transform =
'translateX(25px)';                onText.style.color = '#007bff';
offText.style.color = '#333';
            } else {
                toggleSwitch.style.transform =
'translateX(0)';                onText.style.color = '#333';
offText.style.color = '#007bff';
            }
        }
    </script>
</body>
</html>
```

## OUTPUT



## RESULT

Thus, IoT basaed web controlled home automation using raspberrry pi was executed and verified successfully.

## AIM

To visit monitoring with raspberry pi and pi camera.

## ALGORITHM

**Step 1:** Connect the Pi Camera module to the Raspberry Pi's camera port.

**Step 2:** Enable the camera interface using raspi-config or by editing /boot/config.txt.

**Step 3:** Install the picamera library for interacting with the Pi Camera. If you plan to use image processing, consider installing OpenCV or other relevant libraries.

**Step 4:** Write Python scripts to capture images or video using the Pi Camera.

**Step 5:** Use the picamera library to control the camera settings, capture images or video streams, and save them to the Raspberry Pi's storage.

**Step 6:** If needed, implement image or video processing using libraries like OpenCV to analyze captured data. This could involve object detection, motion tracking, or any other analysis.

**Step 7:** Test the monitoring system by capturing images or video and checking if the system functions as expected.

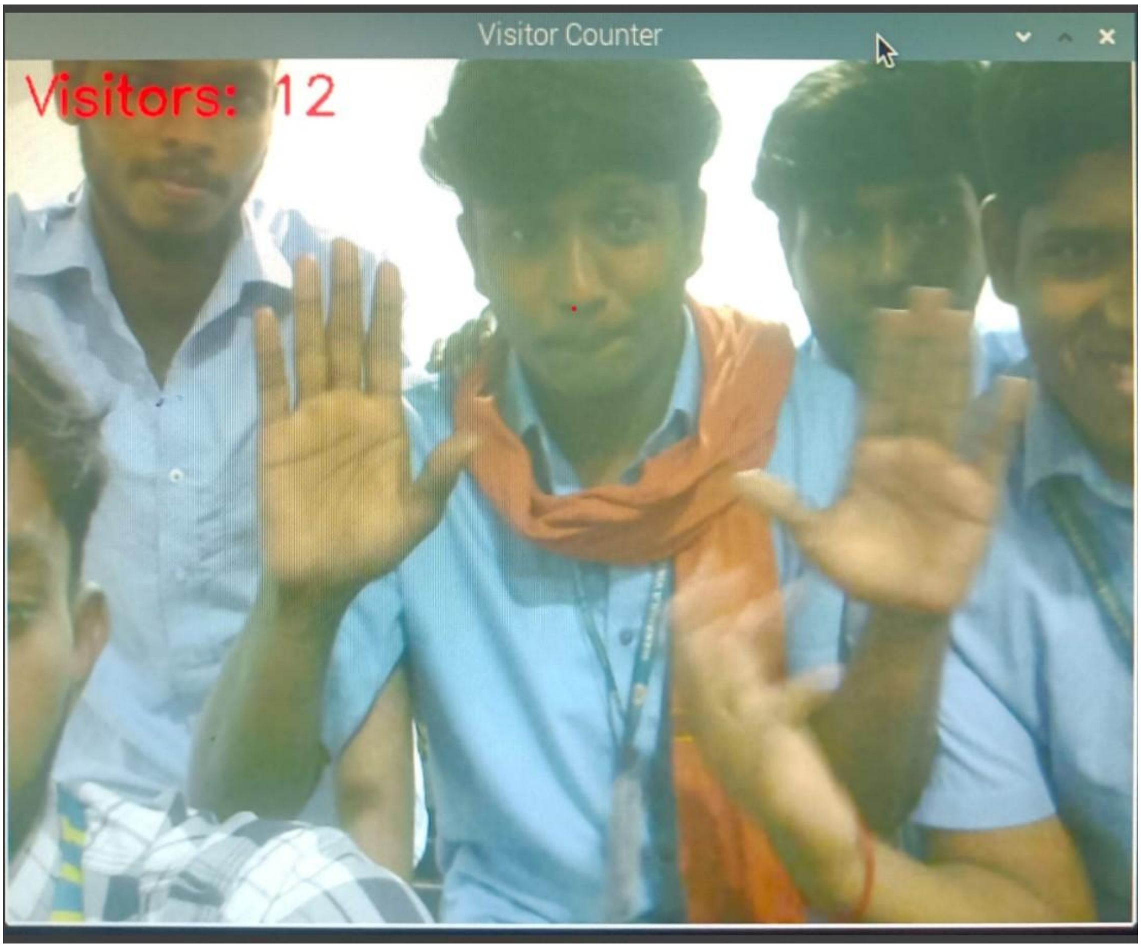
## SOURCE CODE

```
import cv2

# Initialize variables visitor_count

= 0 previous_detection = False

# Create a Haar Cascade classifier for face detection
```

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
# Open the webcam cap =
cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
break
    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect faces in the frame
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
    # Draw rectangles around detected faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
# If faces are detected, increment the visitor count     if
len(faces) > 0 and not previous_detection:
        visitor_count += 1
previous_detection = True     elif
len(faces) == 0:
        previous_detection = False
    # Display the current visitor count on the frame
    cv2.putText(frame, f"Visitors: {visitor_count}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

```
   # Display the frame
cv2.imshow('Visitor Counter', frame)     #
Exit the loop if the 'q' key is pressed     if
cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release the webcam and close OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

## OUTPUT



## RESULT

Thus, To visit monitoring with raspberry pi and pi camera was executed and verified successfully.

## AIM:

To interface Raspberry pi with **RFID.**

## ALGORITHM:

**Step 1:** Connect the RFID reader module to the appropriate GPIO pins on the Raspberry Pi. Usually, RFID readers use SPI or UART communication.

**Step 2:** Enable the SPI interface on the Raspberry Pi using raspi-config or by editing /boot/config.txt.

**Step 3:** Install necessary Python libraries for interfacing with the RFID reader. For example, spidev for SPI communication or any other library specific to your RFID module.

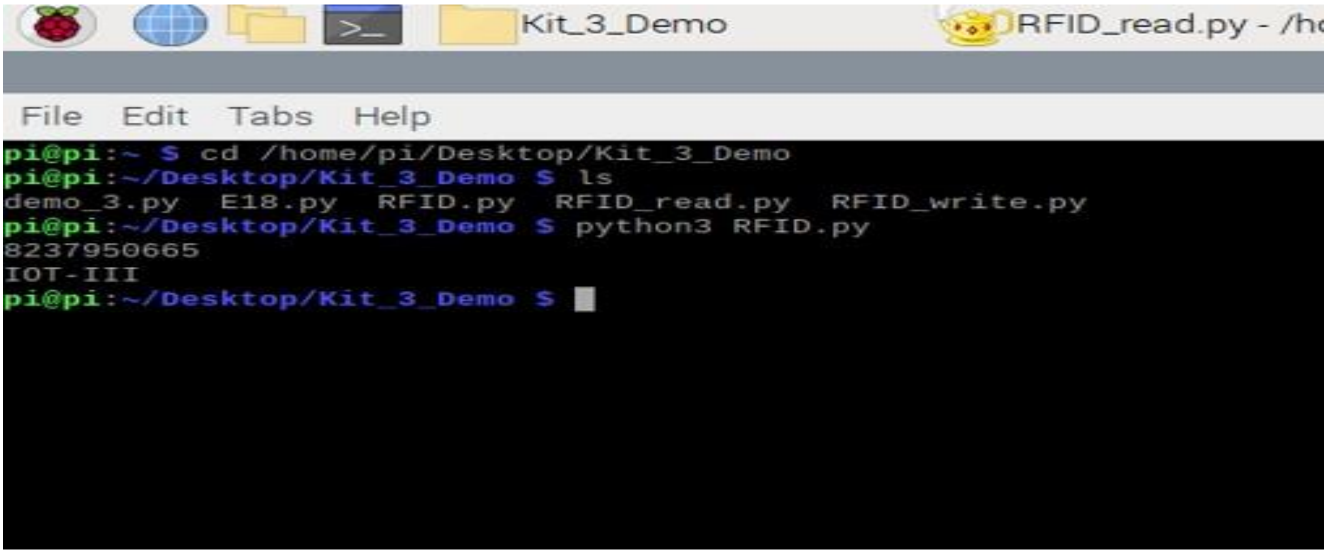
**Step 4:** Import necessary libraries in your Python script.

**Step 5:** Expand the code to handle different RFID functionalities like authentication, storing tag data, or integrating it with databases or other systems.

## SOURCE CODE:

```
#!/usr/bin/env python import RPi.GPIO
as GPIO from mfrc522 import
SimpleMFRC522 reader =
SimpleMFRC522()
try:
    text = input('New data:')
print("Now place your tag to write")
reader.write(text)        print("Written")
finally:
    GPIO.cleanup()
#!/usr/bin/env python import
RPi.GPIO as GPIO from
mfrc522 import
SimpleMFRC522 reader =
SimpleMFRC522()
try:
    id, text = reader.read()
print(id)        print(text)
finally:
    GPIO.cleanup()
```

## OUTPUT



```
pi@pi:~ $ cd /home/pi/Desktop/Kit_3_Demo
pi@pi:~/Desktop/Kit_3_Demo $ ls
demo_3.py  E18.py  RFID.py  RFID_read.py  RFID_write.py
pi@pi:~/Desktop/Kit_3_Demo $ python3 RFID.py
8237950665
IOT-III
pi@pi:~/Desktop/Kit_3_Demo $ 
```

## RESULT

Thus, To interface Raspberry pi with **RFID** was exected and verified successfully.

## AIM:

To build google assistant with raspberry pi.

## ALGORITHM:

**Step 1:** Decide on the devices you want to control and the sensors you want to use.

**Step 2:** Connect the sensors and actuators to the Raspberry Pi GPIO pins or using suitable interfaces.

**Step 3:** Use Flask or Django to create a web application that serves as the user interface for controlling home devices.

**Step 4:** Write Python scripts to read data from sensors connected to the Raspberry Pi.

**Step 5:** Use GPIO control or any suitable method to switch relays or trigger devices based on user commands.

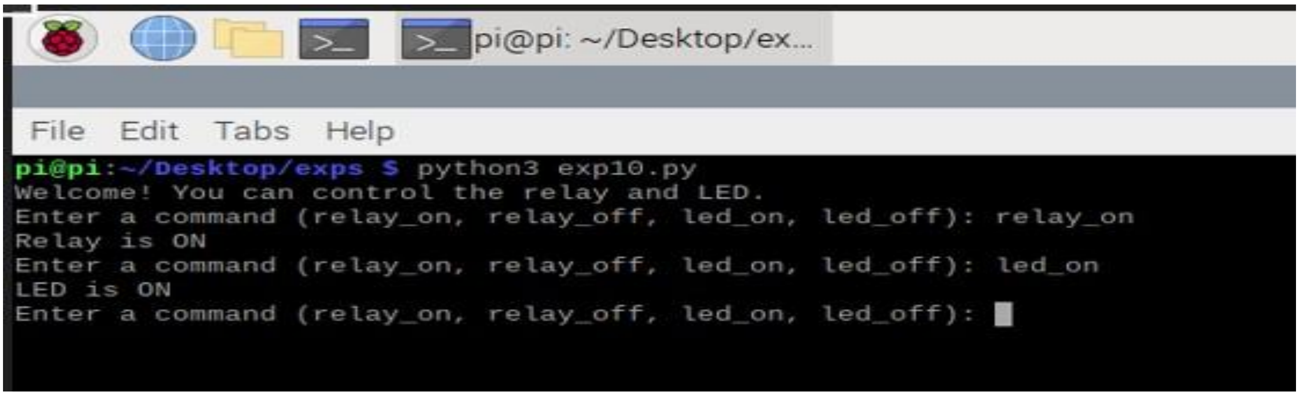
**Step 6:** Implement security measures for authentication and authorization to prevent unauthorized access to the control system.

**Step 7:** Deploy the system in your home environment and continually improve based on feedback and new requirements.

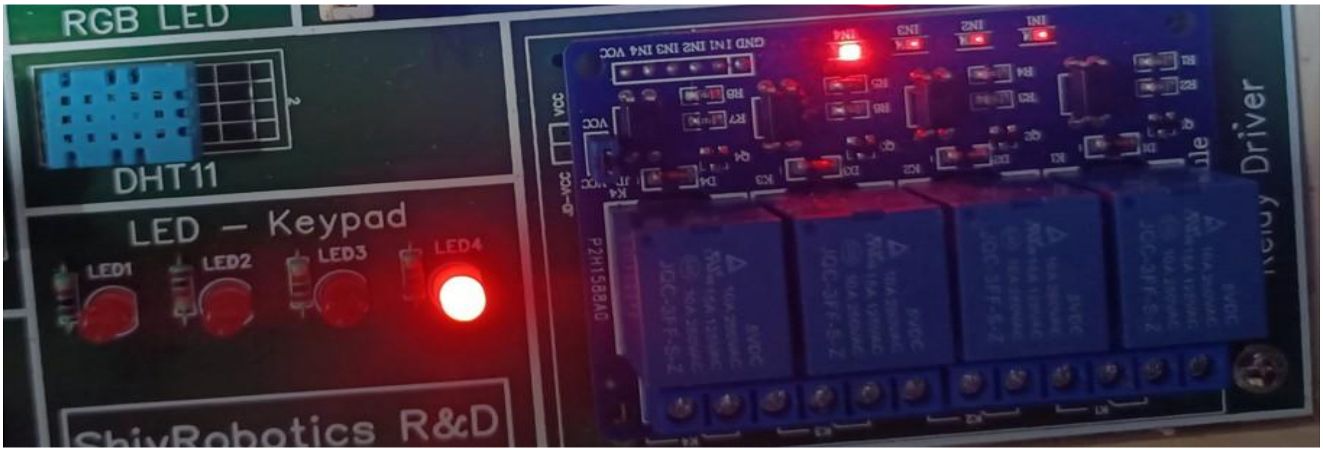
## SOURCE CODE

```
import RPi.GPIO as GPIO import
time
# Set the GPIO mode
GPIO.setmode(GPIO.BCM)
# Define the pin numbers for the relay and LED
relay_pin = 12 led_pin = 6
# Initialize the pins
GPIO.setup(relay_pin, GPIO.OUT)
GPIO.setup(led_pin, GPIO.OUT)
# Function to turn on the relay def
relay_on():
    GPIO.output(relay_pin, GPIO.LOW)  # Reverse logic to turn
ON
    print("Relay is ON")
# Function to turn off the relay def
relay_off():
    GPIO.output(relay_pin, GPIO.HIGH)  # Reverse logic to turn
OFF
    print("Relay is OFF") #
Function to turn on the LED
def led_on():
    GPIO.output(led_pin, GPIO.HIGH)
print("LED is ON") # Function to turn
off the LED def led_off():
    GPIO.output(led_pin, GPIO.LOW)
print("LED is OFF")
try:
    print("Welcome! You can control the relay and LED.")
while True:
        command = input("Enter a command (relay_on, relay_off,
led_on, led_off): ")             if command == "relay_on":
            relay_on()         elif
command == "relay_off":
            relay_off()         elif
command == "led_on":
            led_on()
        elif command == "led_off":
            led_off()
else:
            print("Invalid command. Try again.")
except KeyboardInterrupt:     print("Exiting
the program.")
    GPIO.cleanup()
```

## OUTPUT



```
pi@pi:~/Desktop/exps $ python3 exp10.py
Welcome! You can control the relay and LED.
Enter a command (relay_on, relay_off, led_on, led_off): relay_on
Relay is ON
Enter a command (relay_on, relay_off, led_on, led_off): led_on
LED is ON
Enter a command (relay_on, relay_off, led_on, led_off):
```



## RESULT

Thus, To build google assistant with raspberry pi was executed and verified successfully.