**********7Segment**********

```
import time
import datetime
from lib import tm1637 as obj

Display = obj.TM1637(2,3,5)
Display.Clear()

while(True):
    now = datetime.datetime.now()
    hour = now.hour

    minute = now.minute
    second = now.second
    Display.Clear()
    val = [(int(hour / 10)), (hour % 10), (int(minute / 10)), (minute % 10) ]
    Display.Show(val)
    Display.ShowDoublepoint((second % 2))
    time.sleep(0.25)
```

**********8*8 Matrix**********

```
import re
import time
import argparse

from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT,
SINCLAIR_FONT, LCD_FONT
def demo(n, block_orientation, rotate, msg):
    # create matrix device
    serial = spi(port=0, device=0, gpio=noop())
    device = max7219(serial, cascaded=n or 1, block_orientation=block_orientation, rotate=rotate
or 0)
    show_message(device, msg, fill="white", font=proportional(LCD_FONT), scroll_delay=0.1)
    time.sleep(3)
    pass
if __name__ == "__main__":
    try:
        text_display = raw_input("Enter message to be display on 8x8 matrix = ")
        demo(1, 0, 0, text_display)
    except KeyboardInterrupt:
        pass
    finally:
        print ("Program exit ...")
```

```
**********Oscilloscope**********

import time
import matplotlib.pyplot as plt
from drawnow import *
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()

GAIN = 1
val = [ ]
cnt = 0
plt.ion()
# Start continuous ADC conversions on channel 0 using the previous gain value.
adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 channel 0')
#create the figure function

def makeFig():
    plt.ylim(-50000,50000)
    plt.title('Discover LAB Osciloscope')
    plt.grid(True)
    plt.ylabel('ADC outputs')
    plt.plot(val, 'ro-', label='Channel 0')
    plt.legend(loc='lower right')

while (True):
    # Read the last ADC conversion value and print it out.
```

```
value = adc.get_last_result()
print('Channel 0: {0}'.format(value))
# Sleep for half a second.
time.sleep(0.1)
val.append(int(value))
drawnow(makeFig)
plt.pause(.000001)
cnt = cnt+1
if(cnt>50):
    val.pop(0)
```

```
**********FingerPrint**********

import RPi.GPIO as GPIO
import time
from pyfingerprint.pyfingerprint import PyFingerprint

# GPIO configuration
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

# Initialize Fingerprint Module object
try:
    fingerprint_module = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)
    if not fingerprint_module.verifyPassword():
        raise ValueError('The given fingerprint sensor password is wrong!')
except:
    try:
        fingerprint_module = PyFingerprint('/dev/ttyUSB1', 57600, 0xFFFFFFFF, 0x00000000)
        if not fingerprint_module.verifyPassword():
            raise ValueError('The given fingerprint sensor password is wrong!')
    except:
        try:
            fingerprint_module = PyFingerprint('/dev/ttyUSB2', 57600, 0xFFFFFFFF, 0x00000000)
            if not fingerprint_module.verifyPassword():
                raise ValueError('The given fingerprint sensor password is wrong!')
        except:
            try:
```

```
            fingerprint_module = PyFingerprint('/dev/ttyUSB3', 57600, 0xFFFFFFFF,
0x00000000)
            if not fingerprint_module.verifyPassword():
                raise ValueError('The given fingerprint sensor password is wrong!')
        except Exception as e:
            print('Exception message: ' + str(e))
            exit(1)

# Function to enroll a fingerprint in the database
def enrollFingerInDB():
    print("Enroll your Finger into Fingerprint Database")
    try:
        print('Keep your finger...')

        while not fingerprint_module.readImage():
            pass

        fingerprint_module.convertImage(0x01)
        result = fingerprint_module.searchTemplate()
        positionNumber = result[0]

        if positionNumber >= 0:
            print('This fingerprint template already exists at position number = ' +
str(positionNumber))

        print('Remove your finger...')
        time.sleep(2)

        print('Keep your finger again...')
```

```
        while not fingerprint_module.readImage():
            pass

        fingerprint_module.convertImage(0x02)

        if fingerprint_module.compareCharacteristics() == 0:
            raise Exception('Fingers do not match')

        fingerprint_module.createTemplate()
        positionNumber = fingerprint_module.storeTemplate()
        print('Finger enrolled successfully!')
        print('New template position #' + str(positionNumber))

    except Exception as e:
        print('Enrollment failed.')
        print('Error: ' + str(e))

# Function to search for a fingerprint in the database
def searchFingerInDB():
    print("Search your Finger into Fingerprint Database")
    try:
        print('Keep your finger...')

        while not fingerprint_module.readImage():
            pass

        fingerprint_module.convertImage(0x01)
```

```
        result = fingerprint_module.searchTemplate()
        positionNumber = result[0]
        accuracyScore = result[1]

        if positionNumber == -1:
            print('No fingerprint template match found.')
        else:
            print('Fingerprint found at position number = ' + str(positionNumber))

    except Exception as e:
        print('Searching failed.')
        print('Error: ' + str(e))

# Function to delete a fingerprint from the database
def deleteFingerInDB():
    print("Delete your Finger data from Fingerprint Database")
    try:
        positionNumber = input('Please enter the template position you want to delete (in the range
0 to 1000): ')
        positionNumber = int(positionNumber)

        if fingerprint_module.deleteTemplate(positionNumber):
            print('Template deleted successfully.')

    except Exception as e:
        print('Deletion failed.')
        print('Error: ' + str(e))

# Function to get a fingerprint image
```

```
def getFingerPrintImage():
    print("Generate fingerprint image")
    try:
        print('Keep your finger... and wait for seconds')

        while not fingerprint_module.readImage():
            pass

        fingerprint_module.downloadImage('fingerprint.bmp')
        print('The image is available in the current program directory.')

    except Exception as e:
        print('Image generation failed.')
        print('Error: ' + str(e))

# Function to get the fingerprint template count
def getFingerprintTemplateCount():
    print('Fingerprint template in database: ' + str(fingerprint_module.getTemplateCount()) + '/' +
str(fingerprint_module.getStorageCapacity()))

def main():
    while True:
        print("\nChoose an option in the menu below:")
        print("1. Enroll Fingerprint")
        print("2. Search Fingerprint")
        print("3. Delete Fingerprint")
        print("4. Get Fingerprint Image")
        print("5. Get Fingerprint Record Count in Database")
        print("6. Exit Program")
```

```
choice = input("Enter your choice: ")

if choice == "1":
    enrollFingerInDB()
elif choice == "2":
    searchFingerInDB()
elif choice == "3":
    deleteFingerInDB()
elif choice == "4":
    getFingerPrintImage()
elif choice == "5":
    getFingerprintTemplateCount()
elif choice == "6":
    exit(0)
else:
    print("Invalid choice. Please")
```

```
**********RFID**********
import RPi.GPIO as GPIO
import sys
from lib import SimpleMFRC522

# GPIO configuration
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)

def writeToRFIDTag():
    reader = SimpleMFRC522.SimpleMFRC522()

    try:
        text = raw_input('Enter new ID for your TAG:')
        print("Place your RFID TAG on the reader...")
        reader.write(text)
        print("Writing RFID TAG done")
    except Exception as e:
        print("Writing RFID TAG fail.")
        print("Error: " + str(e))

def readFromRFIDTag():
    reader = SimpleMFRC522.SimpleMFRC522()

    try:
        print("Place your RFID TAG on the reader...")
        id, text = reader.read()
```

```
        print(id)
        print(text)
    except Exception as e:
        print("Error reading RFID tag.")
        print("Error: " + str(e))

def deleteFromRFIDTag():
    reader = SimpleMFRC522.SimpleMFRC522()

    try:
        text = " " * 128  # Blank data
        print("Place your RFID TAG on the reader...")
        reader.write(text)
        print("Deleting ID from RFID TAG done")
    except Exception as e:
        print("Deleting ID from RFID TAG fail.")
        print("Error: " + str(e))

def main():
    clearscreen(25)

    choice = 0  # for choosing the option

    while True:
        clearscreen(25)
        print("Program to demonstrate use of RFID-RC522 module.")
        print("\nChoose an option in the menu below:")
        print("1. Write RFID tag.")
```

```
        print("2. Read RFID tag.")
        print("3. Clear RFID tag.")
        print("4. Exit program.")
        print("\nEnter choice: ")
        choice = int(raw_input())

        if choice == 1:
            writeToRFIDTag()
        elif choice == 2:
            readFromRFIDTag()
        elif choice == 3:
            deleteFromRFIDTag()
        elif choice == 4:
            exit(0)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()
        print("\nProgram terminated.")
```

**********Picamera**********

```
import argparse
import datetime
import imutils
import json
import cv2
import time
import dropbox
from picamera.array import PiRGBArray
from picamera import PiCamera

show_video = True
min_upload_seconds = 3.0
min_motion_frames = 8
camera_warmup_time = 2.5
delta_thresh = 5
resolution = [640, 480]
fps = 16
min_area = 5000
use_dropbox = False

camera = PiCamera()
camera.resolution = tuple(resolution)
camera.framerate = fps
rawCapture = PiRGBArray(camera, size=tuple(resolution))
# allow the camera to warm up
```

```
print("*   warming up...")
time.sleep(camera_warmup_time)
avg = None
lastUploaded = datetime.datetime.now()
motionCounter = 0
for f in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    frame = f.array
    timestamp = datetime.datetime.now()
    text = "Not Detected"

    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    if avg is None:
        print("*   starting monitoring system...")
        avg = gray.copy().astype("float")
        rawCapture.truncate(0)
        continue
    cv2.accumulateWeighted(gray, avg, 0.5)
    frameDelta = cv2.absdiff(gray, cv2.convertScaleAbs(avg))

    thresh = cv2.threshold(frameDelta, delta_thresh, 255,
        cv2.THRESH_BINARY)[1]
    thresh = cv2.dilate(thresh, None, iterations=2)
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
```

```
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
for c in cnts:
    if cv2.contourArea(c) < min_area:
        continue
    (x, y, w, h) = cv2.boundingRect(c)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    text = "Detected"
ts = timestamp.strftime("%A %d %B %Y %I:%M:%S%p")
cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
cv2.putText(frame, ts, (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX,
    0.35, (0, 0, 255), 1)
if text == "Detected":
    if (timestamp - lastUploaded).seconds >= min_upload_seconds:
        motionCounter += 1
        if motionCounter >= min_motion_frames:
          if use_dropbox:
           cv2.imwrite("images/" + str(ts) + ".jpg", frame)
           print ("Image saved with name = " + "images/" + str(ts) + ".jpg")
           lastUploaded = timestamp
           motionCounter = 0
else:
    motionCounter = 0
if show_video:
    cv2.imshow("Home monitoring System", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
```

```
    rawCapture.truncate(0)
**********TeleGram**********

import sys
import time
import random
import datetime
import telepot
import RPi.GPIO as GPIO

RELAY1 = 20
RELAY2 = 16

FAN   = RELAY1
LIGHT = RELAY2

GPIO.setwarnings(False)
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
# set up GPIO output channel
GPIO.setup(RELAY1, GPIO.OUT)
GPIO.setup(RELAY2, GPIO.OUT)

#Your Telegram token key variable.
telegramBotToken = '689381833:AAG8l0Lw4rnUWB4nYlihZSR2RHf3ovMkDqs'

#function to on and off devices
```

```
def on(pin):
        GPIO.output(pin,GPIO.HIGH)

        return "on"

def off(pin):
        GPIO.output(pin,GPIO.LOW)
        return "off"

def handle(msg):
    chat_id = msg['chat']['id']
    print (str(chat_id))
    command = str(msg['text'])

    print ('Receive message from Telegram: %s' % command)

    if 'Fan' in command or 'fan' in command:
            if 'on' in command:
                    bot.sendMessage(chat_id, str( "Fan " + on(FAN) ))
            elif 'off' in command:
                    bot.sendMessage(chat_id, str( "Fan " + off(FAN) ))

    elif 'Light' in command or 'light' in command:
            if 'on' in command:
                    bot.sendMessage(chat_id, str( "Light " + on(LIGHT) ))
            elif 'off' in command:
                    bot.sendMessage(chat_id, str("Light " + off(LIGHT) ))
```

```
bot = telepot.Bot(telegramBotToken)
bot.message_loop(handle)
print ('I am listening...')

while 1:
    time.sleep(10)
```