```
import RPi.GPIO as GPIO
import time
button = 16
led    = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led, GPIO.OUT)
while True:
    button_state = GPIO.input(button)
    if  button_state == True:
        GPIO.output(led, True)
    else:
        GPIO.output(led, False)
```

```
import RPi.GPIO as GPIO
import time
button = 16
led    = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led, GPIO.OUT)
while True:
    button_state = GPIO.input(button)
    if  button_state == True:
        GPIO.output(led, True)
    else:
        GPIO.output(led, False)
```

```
# 50 Hz , 2.5 msec to 12.5 msec
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
servoPIN = int(12)
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)
servo = GPIO.PWM(servoPIN, 50) # GPIO 12 for PWM with 50Hz
servo.start(2.5) # Initialization
# 0 Dgrees
servo.ChangeDutyCycle(2.5)
print("0")
sleep(1)
#45 Degrees
servo.ChangeDutyCycle(5)
print("45")
sleep(1)
#90 Degrees
servo.ChangeDutyCycle(7.5)
print("90")
sleep(1)
#135 Degree
servo.ChangeDutyCycle(10)
print("135")
sleep(1)
#180 Degree
servo.ChangeDutyCycle(12.5)
print("180")
sleep(1)
#135 Degree
servo.ChangeDutyCycle(10)
print("135")
```

```
sleep(1)
#90 Degrees
servo.ChangeDutyCycle(7.5)
print("90")
sleep(1)
#45 Degrees
servo.ChangeDutyCycle(5)
print("45")
sleep(1)
# 0 Dgrees
servo.ChangeDutyCycle(2.5)
print("0")
sleep(1)
GPIO.cleanup()
```

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
servoPIN = int(12)
IR=int(25)
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)
servo = GPIO.PWM(servoPIN, 50) # GPIO 12 for PWM with 50Hz
servo.start(2.5) # Initialization
#Pushbutton as pullup
GPIO.setup(IR,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
ang=float(0)
while True:
    if GPIO.input(IR)==False:
        servo.ChangeDutyCycle(12.5)
        time.sleep(0.3)
        print("Sensor is sensed")
    else:
        servo.ChangeDutyCycle(2.5)
```

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(8, GPIO.OUT) # Set pin 8 to be an output pin and set initial value to low (off)
while True: # Run forever
GPIO.output(8, GPIO.HIGH) # Turn on
 sleep(1) # Sleep for 1 second
 GPIO.output(8, GPIO.LOW) # Turn off
 sleep(1) # Sleep for 1 second
```

```
import RPi.GPIO as GPIO
from time import sleep
ledpin = 12                             # PWM pin connected to LED
GPIO.setwarnings(False)                       #disable warnings
GPIO.setmode(GPIO.BOARD)                      #set pin numbering system
GPIO.setup(ledpin,GPIO.OUT)
pi_pwm = GPIO.PWM(ledpin,1000)                #create PWM instance with frequency
pi_pwm.start(0)                               #start PWM of required Duty Cycle
while True:
    for duty in range(0,101,1):
        pi_pwm.ChangeDutyCycle(duty) #provide duty cycle in the range 0-100
        sleep(0.01)
    sleep(0.5)
```

```
 for duty in range(100,-1,-1):
        pi_pwm.ChangeDutyCycle(duty)
        sleep(0.01)
    sleep(0.5)
```

```
import tkinter as tk
from tkinter import font as tkFont
win=tk.Tk()
win.title("Welcome!")
win.geometry("300x400")
helv36=tkFont.Font(family='Helvitica',size=15)
def B1():
    label1=tk.Label(win,text="red",height="5",width="5")
    label1.grid(row=0,column=2)
    label1['font']=helv36

def B2():
    label2=tk.Label(win,text="green",height="5",width="5")
    label2.grid(row=1,column=2)
    label2['font']=helv36

def B3():
    label3=tk.Label(win,text="blue",height="5",width="5")
    label3.grid(row=2,column=2)
    label3['font']=helv36

button1=tk.Button(win,text="B1",height="5",width="4",bg="red",command=B1)
button1.grid(row=0,column=0)
button1['font']=helv36
```

```
button2=tk.Button(win,text="B2",height="5",width="4",bg="green",command=B2)
button2.grid(row=1,column=0)
button2['font']=helv36

button3=tk.Button(win,text="B3",height="5",width="4",bg="blue",command=B3)
button3.grid(row=2,column=0)
button3['font']=helv36
```

```
import serial
if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.reset_input_buffer()
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            print(line)
```

```
from picamera import PiCamera
from time import sleep
import cv2
camera=PiCamera()
camera.start_preview()
camera.capture("Desktop/imag1.jpg")
sleep(2)
camera.stop_preview()
sleep(2)
camera.close()
```

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
servoPIN = int(12)
IR=int(25)
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)
servo = GPIO.PWM(servoPIN, 50) # GPIO 12 for PWM with 50Hz
servo.start(2.5) # Initialization
user=input("Enter Character:")
while True:
    if (user=='P'):
        servo.ChangeDutyCycle(2.5)
        print("0 Degrees")
        print("Character Received=",user)
    elif (user=='Q'):
        servo.ChangeDutyCycle(5)
        print("45 Degrees")
        print("Character Received=",user)
    elif (user=='R'):
        servo.ChangeDutyCycle(7.5)
        print("90 Degrees")
        print("Character Received=",user)
    elif (user=='S'):
        servo.ChangeDutyCycle(10)
        print("135 Degrees")
        print("Character Received=",user)
    elif (user=='T'):
        servo.ChangeDutyCycle(12.5)
        print("180 Degrees")
        print("Character Received=",user)
```

```
 import serial
if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.reset_input_buffer()
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            print(line)
```

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
TRIG = int(23)
ECHO = int(24)
red=int(12)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(red,GPIO.OUT)
P = GPIO.PWM(red,100)
P.start(0)

while True:
    GPIO.output(TRIG, False)
    # 2 micro Seconds Low pulse
    time.sleep(0.000002)
    # 10 microseconds High Pulse
```

```
GPIO.output(TRIG, True)
time.sleep(0.000010)
# Low pulse
GPIO.output(TRIG, False)

StartTime = time.time()
StopTime = time.time()

while GPIO.input(ECHO) == 0:
    StartTime = time.time()

while GPIO.input(ECHO) == 1:
    StopTime = time.time()

TimeElapsed = StopTime - StartTime
distance = (TimeElapsed * 34300) / 2
distance= int (distance)
print(distance)
if (distance>=2 and distance<=20):
    P.ChangeDutyCycle(100)
elif (distance>=21 and distance<=30):
    P.ChangeDutyCycle(60)
elif (distance>=31 and distance<=40):
    P.ChangeDutyCycle(30)
else:
    P.ChangeDutyCycle(0)
time.sleep(0.1)
```

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
TRIG = int(23)
ECHO = int(24)
red=int(16)
green=int(20)
blue=int(21)
GPIO.cleanup()

GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)


while True:
   GPIO.output(TRIG, False)
   time.sleep(0.000002)
   GPIO.output(TRIG, True)
   time.sleep(0.000010)
   GPIO.output(TRIG, False)

   StartTime = time.time()
   StopTime = time.time()

   while GPIO.input(ECHO) == 0:
      StartTime = time.time()
```

```
    while GPIO.input(ECHO) == 1:
        StopTime = time.time()

    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    distance= int (distance)
    print(distance)

    if (distance>=2 and distance<=20):

        GPIO.output(red,True);
        GPIO.output(green, True)
        GPIO.output(blue, False)

    elif (distance>=21 and distance<=30):

        GPIO.output(red,False);
        GPIO.output(green, True)
        GPIO.output(blue, True)

    elif (distance>=31 and distance<=40):

        GPIO.output(red,True);
        GPIO.output(green, False)
        GPIO.output(blue, True)
else:
        GPIO.output(red,False);
        GPIO.output(green, False)
        GPIO.output(blue, False)

    time.sleep(0.1)
```

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
TRIG = int(23)
ECHO = int(24)

servoPIN = int(12)

GPIO.setmode(GPIO.BCM)

GPIO.setup(servoPIN, GPIO.OUT)


servo = GPIO.PWM(servoPIN, 50) # GPIO 12 for PWM with 50Hz
servo.start(7.5) # Initialization

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def zero():
   servo.ChangeDutyCycle(2.5) # 0 Degrees

def fortyfive():
   servo.ChangeDutyCycle(5) # 45 Degree

def ninety():
   servo.ChangeDutyCycle(7.5) # 90 Degree
```

```
def one():
    servo.ChangeDutyCycle(12.5) # 180 Degrees

while True:
    GPIO.output(TRIG, False)
    time.sleep(0.000002)
    GPIO.output(TRIG, True)
    time.sleep(0.000010)
    GPIO.output(TRIG, False)
    StartTime = time.time()
    StopTime = time.time()
 while GPIO.input(ECHO) == 0:
        StartTime = time.time()

    while GPIO.input(ECHO) == 1:
        StopTime = time.time()
   TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    distance= int (distance)
    print(distance)

    if(distance >=2 and distance <=20):
        zero()
    elif (distance >=21 and distance <=30):
        fortyfive()
    elif (distance >=31 and distance <=40):
        ninety()
    else:
        one()



    time.sleep(0.1)
 import time
import smbus
import Adafruit_ADS1x15
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
adc = Adafruit_ADS1x15.ADS1115()
GAIN = int(1)
red = int(12)
blue= int(13)

GPIO.setmode(GPIO.BCM)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)
R = GPIO.PWM(red, 100) # GPIO 12 for PWM with 50Hz
B = GPIO.PWM(blue,100)

R.start(0) # Initialization
B.start(0)

while True:
    POT_1 = adc.read_adc(0,gain=GAIN)
    POT_2 = adc.read_adc(1,gain=GAIN)
    POT_1=int(POT_1)
    POT_2=int(POT_2)
    bright_red= int((POT_1*100)/32786)
    bright_blue= int((POT_2*100)/32786)
    R.ChangeDutyCycle(bright_red)
    B.ChangeDutyCycle(bright_blue)

    time.sleep(0.2)
```

```
import time
import smbus
import Adafruit_ADS1x15
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
adc = Adafruit_ADS1x15.ADS1115()
GAIN = int(1)
red = int(12)
blue= int(13)
GPIO.setmode(GPIO.BCM)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)

R = GPIO.PWM(red, 100) # GPIO 12 for PWM with 50Hz
B = GPIO.PWM(blue,100)

R.start(0) # Initialization
B.start(0)

while True:
   POT_1 = adc.read_adc(0,gain=GAIN)
   POT_2 = adc.read_adc(1,gain=GAIN)
   POT_1=int(POT_1)
   POT_2=int(POT_2)
   bright_red= int((POT_1*100)/32786)
   bright_blue= int((POT_2*100)/32786)
   R.ChangeDutyCycle(bright_red)
   B.ChangeDutyCycle(bright_blue)

   time.sleep(0.2)
```

```
import time
import smbus
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()GAIN = int(1)

while True:
   LDR = adc.read_adc(0,gain=GAIN)
   LDR=int(LDR)
   voltage= float((LDR*5.0)/32786)
   print(voltage)
   time.sleep(0.2)
```

```
import time
import smbus
import Adafruit_ADS1x15
import RPi.GPIO as GPIO

adc = Adafruit_ADS1x15.ADS1115()
GPIO.setwarnings(False)
GPIO.cleanup()
GAIN = int(1)

red = int(12)
green=int(13)

GPIO.setmode(GPIO.BCM)
GPIO.setup(red, GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
```

```
while True:
    value = adc.read_adc(0,gain=GAIN)
    voltage= float((value*5.0)/32786)
    temp=float(voltage*100)
    if (temp>=28):
        print("High Temp")
        GPIO.output(red,1)
        GPIO.output(green,0)

    else:
        print("Low Temp")
        GPIO.output(red,0)
        GPIO.output(green,1)
    print("Temperature:",temp)
    time.sleep(0.2)
```

```
from gpiozero import CPUTemperature
from time import sleep
import RPi.GPIO as GPIO
from RPLCD import i2c

lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'

address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()

while True:
    lcd.cursor_pos=(0,0)
    lcd.write_string("Temp:")
    cpu = CPUTemperature()
    temp=cpu.temperature
    if temp>=48:
        lcd.clear()
        lcd.cursor_pos=(0,0)
        lcd.write_string("High Temp")
    lcd.cursor_pos=(0,5)
    lcd.write_string(str(temp))
    print(temp)
    sleep(1)
```

```
import time
import smbus
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()
GAIN = int(1)

while True:
   X_pos = adc.read_adc(0,gain=GAIN)
   Y_pos = adc.read_adc(1,gain=GAIN)
   X_pos=int(X_pos)
   Y_pos=int(Y_pos)
   print("X:",X_pos,"Y:",Y_pos)
   time.sleep(0.2)
```

```
import RPi.GPIO as GPIO
from time import sleep
import smbus
import Adafruit_ADS1x15

GPIO.setwarnings(False)

adc = Adafruit_ADS1x15.ADS1115()
GAIN = int(1)

servoPIN = int(12)
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)

servo = GPIO.PWM(servoPIN, 50) # GPIO 12 for PWM with 50Hz
servo.start(2.5) # Initialization

while True:
    X_pos = adc.read_adc(0,gain=GAIN)
    X_pos=int(X_pos)

    if X_pos>=30000:
        servo.ChangeDutyCycle(2.5)
        sleep(0.2)
    elif X_pos<=100:
        servo.ChangeDutyCycle(12.5)
        sleep(0.2)
    else:
        servo.ChangeDutyCycle(7.5)

print("X:",X_pos)
sleep(0.2)
```

```
import time
import board
import adafruit_mpu6050

i2c = board.I2C()
mpu = adafruit_mpu6050.MPU6050(i2c)
while True:
    X=mpu.gyro[0]*(180/3.14)
    Y=mpu.gyro[1]*(180/3.14)
    Z=mpu.gyro[2]*(180/3.14)
    print("X:%3d"%X,"Y:%3d"%Y,"Z:%3d"%Z)
#     print("Gyro X:%.2f, Y: %.2f, Z: %.2f rad/s" % (mpu.gyro))
    #print(X)
    time.sleep(0.2)
```

```
import time
import board
import adafruit_mpu6050

i2c = board.I2C()
mpu = adafruit_mpu6050.MPU6050(i2c)

while True:
    X=mpu.gyro[0]*(180/3.14)
    Y=mpu.gyro[1]*(180/3.14)
    Z=mpu.gyro[2]*(180/3.14)
    print("X:%3d"%X,"Y:%3d"%Y,"Z:%3d"%Z)
#     print("Gyro X:%.2f, Y: %.2f, Z: %.2f rad/s" % (mpu.gyro))
    #print(X)
    time.sleep(0.2)
```

```
import RPi.GPIO as GPIO
from RPLCD import i2c
from time import sleep

GPIO.setwarnings(False)

lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'

address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()

msg=input("Enter Message:")
col=int(input("Enter col:"))
row=int(input("Enter row:"))


while True:

   lcd.cursor_pos=(row,col)
   lcd.write_string(msg)

#lcd.close(clear=True)
#lcd.write_string("LearnVern")
```

```
import RPi.GPIO as GPIO
from RPLCD import i2c
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

ir=int(22)
lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'
address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()



GPIO.setup(ir,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

number=input("Enter Number:")
lcd.cursor_pos=(0,0)
lcd.write_string("Number:")

while True:
    lcd.cursor_pos=(0,7)
    lcd.write_string(str(number))
    sleep(0.25)

#lcd.close(clear=True)
```

```
import RPi.GPIO as GPIO
from RPLCD import i2c
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

ir=int(22)



lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'
address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()
GPIO.setup(ir,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

number=input("Enter Number:")
lcd.cursor_pos=(0,0)
lcd.write_string("Number:")
while True:
    lcd.cursor_pos=(0,7)
    lcd.write_string(str(number))
    sleep(0.25)

#lcd.close(clear=True)
```

```
import RPi.GPIO as GPIO
from RPLCD import i2c
from time import sleep
import smbus
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()

GAIN = int(1)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'

address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()

while True:
   lcd.clear()
   value = adc.read_adc(0,gain=GAIN)
   value=int(value)
   lcd.cursor_pos=(0,0)
   lcd.write_string(str(value))
   sleep(0.25)

#lcd.close(clear=True)
```

```
import RPi.GPIO as GPIO
import time

from RPLCD import i2c

GPIO.setwarnings(False)

lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'

address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()

# row
L1 = 16
L2 = 20
L3 = 21
L4 = 26
# col
C1 = 19
C2 = 13
C3 = 5

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(L1, GPIO.OUT)
GPIO.setup(L2, GPIO.OUT)
GPIO.setup(L3, GPIO.OUT)
GPIO.setup(L4, GPIO.OUT)

GPIO.setup(C1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C3, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)


def readLine(line, characters):
    GPIO.output(line, GPIO.HIGH)
    if(GPIO.input(C1) == 1):

        lcd.write_string(characters[0])
    if(GPIO.input(C2) == 1):

        lcd.write_string(characters[1])
    if(GPIO.input(C3) == 1):

        lcd.write_string(characters[2])
    GPIO.output(line, GPIO.LOW)

while True:
    readLine(L1, ["P","Q","R"])
    readLine(L2, ["4","5","6"])
    readLine(L3, ["7","8","9"])
    readLine(L4, ["*","0","#"])
    time.sleep(0.25)
```

```
import RPi.GPIO as GPIO
import time

from RPLCD import i2c

GPIO.setwarnings(False)

lcdmode='i2c'
cols=16
rows=2
charmap='A00'
i2c_expander ='PCF8574'

address=0X3f
port=1
lcd=i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,cols=cols, rows=rows)
lcd.clear()

# row
L1 = 16
L2 = 20
L3 = 21
L4 = 26
# col
C1 = 19
C2 = 13
C3 = 5

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(L1, GPIO.OUT)
GPIO.setup(L2, GPIO.OUT)
GPIO.setup(L3, GPIO.OUT)
GPIO.setup(L4, GPIO.OUT)

GPIO.setup(C1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C3, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)



def readLine(line, characters):
    GPIO.output(line, GPIO.HIGH)
    if(GPIO.input(C1) == 1):

        lcd.write_string(characters[0])
    if(GPIO.input(C2) == 1):

        lcd.write_string(characters[1])
    if(GPIO.input(C3) == 1):

        lcd.write_string(characters[2])
    GPIO.output(line, GPIO.LOW)

while True:
    readLine(L1, ["P","Q","R"])
    readLine(L2, ["4","5","6"])
    readLine(L3, ["7","8","9"])
    readLine(L4, ["*","0","#"])
    time.sleep(0.25)
```

```
import RPi.GPIO as GPIO
import time



GPIO.setwarnings(False)

# row
L1 = 16
L2 = 20
L3 = 21
L4 = 26
# col
C1 = 19
C2 = 13
C3 = 5

red=int(18)
green=int(24)
blue=int(23)

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(L1, GPIO.OUT)
GPIO.setup(L2, GPIO.OUT)
GPIO.setup(L3, GPIO.OUT)
GPIO.setup(L4, GPIO.OUT)

GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)
GPIO.setup(C1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(C3, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

def row1():
    GPIO.output(L1, GPIO.HIGH)
    if(GPIO.input(C1)==1):
        print("red")
        GPIO.output(red,True)
        GPIO.output(green,False)
        GPIO.output(blue,False)

    elif(GPIO.input(C2)==1):
        print("green")
        GPIO.output(green,True)
        GPIO.output(red,False)
        GPIO.output(blue,False)

    elif(GPIO.input(C3)==1):
        print("blue")
        GPIO.output(blue,True)
        GPIO.output(green,False)
        GPIO.output(red,False)

    GPIO.output(L1, GPIO.LOW)

def row2():
    GPIO.output(L2, GPIO.HIGH)
    if(GPIO.input(C1)==1):
        print("Yellow")
        GPIO.output(red,True)
        GPIO.output(green,True)
        GPIO.output(blue,False)
    elif(GPIO.input(C2)==1):
        print("Cyan")
        GPIO.output(red,False)
        GPIO.output(green,True)
        GPIO.output(blue,True)
    elif(GPIO.input(C3)==1):
        print("Magenta")
        GPIO.output(red,True)
        GPIO.output(green,False)
        GPIO.output(blue,True)
    GPIO.output(L2, GPIO.LOW)

def row3():
    GPIO.output(L3, GPIO.HIGH)
    if(GPIO.input(C1)==1):
        print("7")
    elif(GPIO.input(C2)==1):
        print("All off")
        GPIO.output(red,False)
        GPIO.output(green,False)
        GPIO.output(blue,False)
    elif(GPIO.input(C3)==1):
        print("White")
        GPIO.output(red,True)
        GPIO.output(green,True)
        GPIO.output(blue,True)
    GPIO.output(L3, GPIO.LOW)
```

```
while True:
    row1()
    row2()
    row3()
    time.sleep(0.25)


 #L298d Motor Driver Programming
import RPi.GPIO as GPIO
from time import sleep

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
in1=int(23)
in2=int(24)
en=int(12)

GPIO.setup(en,GPIO.OUT)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
p=GPIO.PWM(en,1000)
p.start(0)
GPIO.cleanup()

#     p.ChangeDutyCycle(100)
#     GPIO.output(in1,True)
#     GPIO.output(in2,False)
#     sleep(1)
#     p.ChangeDutyCycle(0)
#     GPIO.output(in1,True)
#     GPIO.output(in2,False)
#     sleep(1)
#    p.ChangeDutyCycle(50)
#    GPIO.output(in2,True)
#    GPIO.output(in1,False)
#    sleep(1)


 import RPi.GPIO as GPIO
from time import sleep

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
in1=int(23)
in2=int(24)
en=int(12)

GPIO.setup(en,GPIO.OUT)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)

p=GPIO.PWM(en,1000)
p.start(0)

def forward():
    p.ChangeDutyCycle(speed)
    GPIO.output(in1,True)
    GPIO.output(in2,False)
def backward():
    p.ChangeDutyCycle(speed)
    GPIO.output(in2,True)
    GPIO.output(in1,False)

while True:
    X=input('Enter X:')
    speed=int(input("Enter Speed:"))
    if X=='F':
        forward()

    elif X=='B':
        backward()

    elif X=='S':
        p.ChangeDutyCycle(0)
        GPIO.output(in1,True)
        GPIO.output(in2,False)
```

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

bulb1=int(23)
bulb2=int(24)

GPIO.setup(bulb1,GPIO.OUT)
GPIO.setup(bulb2,GPIO.OUT)

GPIO.output(bulb1,True)
GPIO.output(bulb2,True)
while True:
    X=int(input('Enter No:'))
    if X==1:
        GPIO.output(bulb1,False)
        GPIO.output(bulb2,True)
    elif X==2:
        GPIO.output(bulb2,False)
        GPIO.output(bulb1,True)
    elif X==3:
        GPIO.output(bulb2,False)
        GPIO.output(bulb1,False)
    elif X==4:
        GPIO.output(bulb2,True)
        GPIO.output(bulb1,True)



from tkinter import *
from tkinter import font as tkFont
import time
import smbus
import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()
GAIN = int(1)

win=Tk()
win.title("Potentiometer")
win.geometry("240x350")

helv36=tkFont.Font(family='Helvitica',size=25)

def pressed():
    value = adc.read_adc(0,gain=GAIN)
    value=int(value)
    label1=Label(win,text=str(value),height='5',width='5')
    label1.grid(row=5,column=5)
    label1['font']=helv36
    print(value)
    time.sleep(0.2)

button=Button(win,text="Press",height="5",width="4",bg="blue",command=pressed)
button.grid(row=0,column=0)
import tkinter as tk
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

led=int(16)
GPIO.setup(led,GPIO.OUT)

win=tk.Tk()
win.title("learnvern")
win.geometry("300x400")


def first():
    GPIO.output(led,True)
    label1=tk.Label(win,text="ON",height="5",width="5")
    label1.grid(row=1,column=0)
    print("First Button Pressed")

def second():
    GPIO.output(led,False)
    print("Second button Pressed")

button1=tk.Button(win,text="B1",height="10",width="10",bg="yellow",command=first)
```

```
button1.grid(row=0,column=0)

button2=tk.Button(win,text="B2",bg="blue",command=second)
button2.grid(row=0,column=1)
win.mainloop()
```

```
from guizero import App,TextBox,Text, PushButton
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

led=int(16)
GPIO.setup(led,GPIO.OUT)

def ledon():
   GPIO.output(led,True)

def ledoff():
   GPIO.output(led,False)
app=App("Hello World")

welcome_message = Text(app, text="Welcome to my app",size=40)
my_name=TextBox(app)
button1=Pushbutton(app,command=ledon,text="ON")
button2=Pushbutton(app,command=ledoff,text="OFF")
app.display()
```

```
import tkinter as tk
from tkinter import font as tkFont

win=tk.Tk()
win.title("Welcome!")
win.geometry("300x400")
helv36=tkFont.Font(family='Helvitica',size=15)
def B1():
    label1=tk.Label(win,text="red",height="5",width="5")
    label1.grid(row=0,column=2)
    label1['font']=helv36

def B2():
    label2=tk.Label(win,text="green",height="5",width="5")
    label2.grid(row=1,column=2)
    label2['font']=helv36

def B3():
    label3=tk.Label(win,text="blue",height="5",width="5")
    label3.grid(row=2,column=2)
    label3['font']=helv36

button1=tk.Button(win,text="B1",height="5",width="4",bg="red",command=B1)
button1.grid(row=0,column=0)
button1['font']=helv36
button2=tk.Button(win,text="B2",height="5",width="4",bg="green",command=B2)
button2.grid(row=1,column=0)
button2['font']=helv36
button3=tk.Button(win,text="B3",height="5",width="4",bg="blue",command=B3)
button3.grid(row=2,column=0)
button3['font']=helv36
```

```
import tkinter as tk
from tkinter import font as tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

red=int(16)
green=int(20)
blue=int(21)

GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)

win=tk.Tk()
win.title("Led control Panel!")
win.geometry("300x400")

helv36=tkFont.Font(family='Helvitica',size=15)

def RED():
    GPIO.output(red,1)
    GPIO.output(green,0)
    GPIO.output(blue,0)

    label1=tk.Label(win,text="red",height="5",width="5")
    label1.grid(row=1,column=0)
    label1['font']=helv36

    label2=tk.Label(win,text="off",height="5",width="5")
    label2.grid(row=1,column=1)
    label2['font']=helv36

    label3=tk.Label(win,text="off",height="5",width="5")
    label3.grid(row=1,column=2)
    label3['font']=helv36


def GREEN():
    GPIO.output(red,0)
    GPIO.output(green,1)
    GPIO.output(blue,0)

    label1=tk.Label(win,text="off",height="5",width="5")
    label1.grid(row=1,column=0)
    label1['font']=helv36

    label2=tk.Label(win,text="green",height="5",width="5")
    label2.grid(row=1,column=1)
    label2['font']=helv36

    label3=tk.Label(win,text="off",height="5",width="5")
    label3.grid(row=1,column=2)
    label3['font']=helv36



def BLUE():
    GPIO.output(red,0)
    GPIO.output(green,0)
    GPIO.output(blue,1)
    label1=tk.Label(win,text="off",height="5",width="5")
    label1.grid(row=1,column=0)
    label1['font']=helv36

    label2=tk.Label(win,text="off",height="5",width="5")
    label2.grid(row=1,column=1)
    label2['font']=helv36

    label3=tk.Label(win,text="blue",height="5",width="5")
    label3.grid(row=1,column=2)
    label3['font']=helv36

def OFF():
    GPIO.output(red,0)
    GPIO.output(green,0)
    GPIO.output(blue,0)

    label1=tk.Label(win,text="off",height="5",width="5")
    label1.grid(row=1,column=0)
    label1['font']=helv36

    label2=tk.Label(win,text="off",height="5",width="5")
    label2.grid(row=1,column=1)
    label2['font']=helv36

    label3=tk.Label(win,text="off",height="5",width="5")
    label3.grid(row=1,column=2)
    label3['font']=helv36



button1=tk.Button(win,text="RED",height="5",width="4",bg="red",command=RED)
button1.grid(row=0,column=0)
```

```
button1['font']=helv36

button1['background']='#90EE90'


button2=tk.Button(win,text="GREEN",height="5",width="4",bg="green",command=GREEN)

button2.grid(row=0,column=1)

button2['font']=helv36


button3=tk.Button(win,text="BLUE",height="5",width="4",bg="blue",command=BLUE)

button3.grid(row=0,column=2)

button3['font']=helv36

button4=tk.Button(win,text="OFF",height="5",width="4",bg="white",command=OFF)

button4.grid(row=0,column=3)

button4['font']=helv36
```

```
from tkinter import *
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
red=int(12)
GPIO.setup(red,GPIO.OUT)
R = GPIO.PWM(red, 100)
R.start(0)

master = Tk()

def show_values():
   print(w1.get())
   R.ChangeDutyCycle(w1.get())

w1 = Scale(master,from_=0, to=100)
w1.grid(row=0,column=0)
w1.pack()
Button(master, text='Show', command=show_values).pack()
mainloop()

from tkinter import *
from tkinter import font as tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

red=int(12)
green=int(13)
blue=int(18)
```

```
GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)

R = GPIO.PWM(red, 100)
G = GPIO.PWM(green, 100)
B = GPIO.PWM(blue, 100)

R.start(0)
G.start(0)
B.start(0)

frame=Tk()
frame.title("Basic Slider")
frame.geometry("240x350")

helv36=tkFont.Font(family='Helvitica',size=15)

def first():
   print(w1.get())
   G.ChangeDutyCycle(0)
   B.ChangeDutyCycle(0)
   R.ChangeDutyCycle(w1.get())

def second():
   print(w2.get())
   G.ChangeDutyCycle(w2.get())
   R.ChangeDutyCycle(0)
   B.ChangeDutyCycle(0)

def third():
   print(w3.get())
```

```
    B.ChangeDutyCycle(w3.get())
    R.ChangeDutyCycle(0)
    G.ChangeDutyCycle(0)


w1 = Scale(frame,from_=0, to=100,label='A',bg='red',orient=HORIZONTAL)
w1['font']=helv36
w1.pack() # slider update
Button(frame, text='RUN',command=first).pack()


w2 = Scale(frame,from_=0, to=100,label='B',bg='green',orient=HORIZONTAL)
w2['font']=helv36
w2.pack() # slider update
Button(frame, text='RUN',command=second).pack()


w3 = Scale(frame,from_=0, to=100,label='C',bg='blue',orient=HORIZONTAL)
w3['font']=helv36
w3.pack() # slider update
Button(frame,text='RUN',command=third).pack()

mainloop() # continous

# Slider step and slider size
# transfer values to PWM LEds
# transfer values tp PWM servo motor
```

```
Import tkinter as tk
from tkinter import font as tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

win=tk.Tk()
win.title("Led control Panel!")
win.geometry("240x350")

helv36=tkFont.Font(family='Helvitica',size=12)

red=int(12)
green=int(13)
blue=int(18)

GPIO.setup(red,GPIO.OUT)
GPIO.setup(green,GPIO.OUT)
GPIO.setup(blue,GPIO.OUT)

R = GPIO.PWM(red, 100)
G = GPIO.PWM(green, 100)
B = GPIO.PWM(blue, 100)

R.start(0)
G.start(0)
B.start(0)



def F_RED():
    R.ChangeDutyCycle(100)
def M_RED():
    R.ChangeDutyCycle(60)
def L_RED():
    R.ChangeDutyCycle(25)

def F_GREEN():
    G.ChangeDutyCycle(100)
def M_GREEN():
    G.ChangeDutyCycle(60)
def L_GREEN():
    G.ChangeDutyCycle(25)

def F_BLUE():
    B.ChangeDutyCycle(100)
def M_BLUE():
    B.ChangeDutyCycle(60)
def L_BLUE():
    B.ChangeDutyCycle(25)

def alloff():
    R.ChangeDutyCycle(0)
    G.ChangeDutyCycle(0)
    B.ChangeDutyCycle(0)


button1=tk.Button(win,text="F_RED",height="5",width="7",command=F_RED)
button1.grid(row=0,column=0)
button1['font']=helv36
button1['background']='#FF0000'

button2=tk.Button(win,text="M_RED",height="5",width="7",command=M_RED)
```

```
button2.grid(row=0,column=1)
button2['font']=helv36
button2['background']='#801b1b'

button3=tk.Button(win,text="L_RED",height="5",width="7",command=L_RED)
button3.grid(row=0,column=2)
button3['font']=helv36
button3['background']='#FFCCCB'

button4=tk.Button(win,text="F_GREEN",height="5",width="7",command=F_GREEN)
button4.grid(row=1,column=0)
button4['font']=helv36
button4['background']='#00FF00'

button5=tk.Button(win,text="M_GREEN",height="5",width="7",command=M_GREEN)
button5.grid(row=1,column=1)
button5['font']=helv36
button5['background']='#3CB371'

button6=tk.Button(win,text="L_GREEN",height="5",width="7",command=L_GREEN)
button6.grid(row=1,column=2)
button6['font']=helv36
button6['background']='#90EE90'


button7=tk.Button(win,text="F_BLUE",height="5",width="7",command=F_BLUE)
button7.grid(row=2,column=0)
button7['font']=helv36
button7['background']='#0000FF'

button8=tk.Button(win,text="M_BLUE",height="5",width="7",command=M_BLUE)
button8.grid(row=2,column=1)
```

```
button8['font']=helv36
button8['background']='#0000CD'

button9=tk.Button(win,text="L_BLUE",height="5",width="7",command=L_BLUE)
button9.grid(row=2,column=2)
button9['font']=helv36
button9['background']='#ADD8E6'

button10=tk.Button(win,text="All off",bg='white',height="5",width="7",command=alloff)
button10.grid(row=3,column=1)
button10['font']=helv36
```

```
from tkinter import *
from tkinter import font as tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
servo=int(12)

GPIO.setup(servo,GPIO.OUT)

ang = GPIO.PWM(servo,50) # 50 Hz start
ang.start(2.5) # O degrres 2.5= 0 degrees

frame=Tk()
frame.title("Basic Slider")
frame.geometry("100x150")

helv36=tkFont.Font(family='Helvitica',size=15)

def rotate():
   angle=w1.get()
   print((angle*180)/12.5)
   ang.ChangeDutyCycle(w1.get())

w1 = Scale(frame,from_=2.5,
to=12.5,resolution=1,label='Servo',bg='yellow',orient=HORIZONTAL)
w1['font']=helv36
w1.pack() # slider update
Button(frame, text='Rotate',command=rotate).pack()

mainloop() # continous
```