Please note that “tabs” may be out of line due to copy paste into this word document…

Python code is dependent on tabs for reading.

import RPi.GPIO as GPIO                    #Import GPIO library

import time

from multiprocessing import Process

import sys

#light sensor

pir\_sensor = 18

BUZZ = 9

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(12, GPIO.OUT) # PIR light

GPIO.setup(25, GPIO.OUT) # ultra 1 light

GPIO.setup(24, GPIO.OUT) # ultra 2 valve

GPIO.setup(BUZZ, GPIO.OUT) # buzzer output

GPIO.setup(18, [GPIO.IN](http://gpio.in/)) # PIR sensor reading

current\_state = 0

# buzzer

successLength = 0.1

successTone = 0.0001

failLength = 0.2

failTone = 0.001

# ultrasonic

TRIG = 4

ECHO = 17

TRIG2 = 27

ECHO2 = 22

GPIO\_LIGHT = 25

VALVE= 24

print("process started")

GPIO.setup(TRIG, GPIO.OUT)                  #Set pin as GPIO out

GPIO.setup(ECHO, [GPIO.IN](http://gpio.in/))                   #Set pin as GPIO in

GPIO.setup(TRIG2, GPIO.OUT)                  #Set pin as GPIO out

GPIO.setup(ECHO2, [GPIO.IN](http://gpio.in/))                   #Set pin as GPIO in

def light():

    try:

        while True:

            time.sleep(0.1)

            current\_state = GPIO.input(18)

            if current\_state == 1:

                print("GPIO pin %s is %s" % (pir\_sensor, current\_state))

                GPIO.output(12,True) #turn on light outside pin as GPIO in

                time.sleep(5)

                GPIO.output(12,False)  #turn off light

                time.sleep(1)

    except KeyboardInterrupt:

        pass

    finally:

        GPIO.cleanup()

def ultra():

    while True:

      GPIO.output(TRIG, False)                 #Set TRIG as LOW

      print ("Waiting For Sensor To Settle")

      time.sleep(2)                            #Delay of 2 seconds

      GPIO.output(TRIG, True)                  #Set TRIG as HIGH

      time.sleep(0.00001)                      #Delay of 0.00001 seconds

      GPIO.output(TRIG, False)                 #Set TRIG as LOW

      while GPIO.input(ECHO)==0:               #Check whether the ECHO is LOW

        pulse\_start = time.time()              #Saves the last known time of LOW pulse

      while GPIO.input(ECHO)==1:               #Check whether the ECHO is HIGH

        pulse\_end = time.time()                #Saves the last known time of HIGH pulse

      pulse\_duration = pulse\_end - pulse\_start #Get pulse duration to a variable

      distance = pulse\_duration \* 17150        #Multiply pulse duration by 17150 to get distance

      distance = round(distance, 2)            #Round to two decimal points

      if distance > 2 and distance < 100:      #Check whether the distance is within range

        print ("Distance:",distance - 0.5, "cm")  #Print distance with 0.5 cm calibration

        GPIO.output(25,True)  #turn on light inside

        print ("light on!")

      else:

        print ("Out Of Range")                   #display out of range

        GPIO.output(25,False)

        print ("light OFFFFF!") #turn off light inside

def ultra2():

    while True:

      GPIO.output(TRIG2, False)                 #Set TRIG as LOW

      print ("Waitng For Sensor To Settle")

      time.sleep(2)                            #Delay of 2 seconds

      GPIO.output(TRIG2, True)                  #Set TRIG as HIGH

      time.sleep(0.00001)                      #Delay of 0.00001 seconds

      GPIO.output(TRIG2, False)                 #Set TRIG as LOW

      while GPIO.input(ECHO2)==0:               #Check whether the ECHO is LOW

        pulse\_start = time.time()              #Saves the last known time of LOW pulse

      while GPIO.input(ECHO2)==1:               #Check whether the ECHO is HIGH

        pulse\_end = time.time()                #Saves the last known time of HIGH pulse

      pulse\_duration = pulse\_end - pulse\_start #Get pulse duration to a variable

      distance = pulse\_duration \* 17150        #Multiply pulse duration by 17150 to get distance

      distance = round(distance, 2)            #Round to two decimal points

      if not (distance > 2 and distance < 10):      #Check whether the distance is within range

        print ("Distance:",distance - 0.5, "cm")  #Print distance with 0.5 cm calibration

        GPIO.output(24, GPIO.HIGH)  #turn on light inside

        print ("valve on!")

      else:

        print ("Out Of Range")                   #display out of range

        GPIO.output(24, GPIO.LOW)

        print ("valve OFFFFF!") #turn off light inside

def keypad():

    MATRIX = [['1', '2', '3', 'A'],

          ['4', '5', '6', 'B'],

          ['7', '8', '9', 'C'],

          ['\*', '0', '#', 'D']]

    ROW = [5, 6, 13, 19]

    COL = [26, 21, 20, 16]

    for j in range(4):

        GPIO.setup(COL[j], GPIO.OUT)

        GPIO.output(COL[j], 1)

    for i in range(4):

        GPIO.setup(ROW[i], [GPIO.IN](http://gpio.in/), pull\_up\_down=GPIO.PUD\_UP)

    password = "1234"

    attempt = ""

    try:

        while (True):

            for j in range(4):

                GPIO.output(COL[j], 0)

                for i in range(4):

                    if GPIO.input(ROW[i]) == 0:

                        time.sleep(0.01)

                        while (GPIO.input(ROW[i]) == 0):

                            pass

                        attempt += MATRIX[i][j]

                        if len(attempt) == len(password):

                            if attempt == password:

                                print ("Password OK")

                                success()

                                GPIO.setup(23, GPIO.OUT) # Door Lock 2

                                # This is where you unlock the door.

                                GPIO.output(23,False)  #turn on light inside

                                time.sleep(2)

                                GPIO.output(23,True)

                                attempt = ""

                            else:

                                print ("Password incorrect")

                                fail()

                                attempt = ""

                time.sleep(0.01)

                GPIO.output(COL[j], 1)

    except KeyboardInterrupt:

        GPIO.cleanup()

def success():

    successBeep(time.time() + successLength)

    time.sleep(0.1)

    successBeep(time.time() + successLength)

    time.sleep(0.1)

    successBeep(time.time() + successLength)

def fail():

    failBeep(time.time() + failLength)

    time.sleep(0.2)

    failBeep(time.time() + failLength)

    time.sleep(0.2)

    failBeep(time.time() + failLength)

def failBeep(end):

    while end > time.time():

        GPIO.output(BUZZ, 0)

        time.sleep(failTone)

        GPIO.output(BUZZ, 1)

        time.sleep(failTone)

def successBeep(end):

        while end > time.time():

            GPIO.output(BUZZ,0)

            time.sleep(successTone)

            GPIO.output(BUZZ,1)

            time.sleep(successTone)

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

     p1 = Process(target = light)

     p1.start()

     p2 = Process(target = ultra)

     p2.start()

     p3 = Process(target = ultra2)

     p3.start()

     p4 = Process(target = keypad)

     p4.start()