import time

import os

import sys

import RPi.GPIO as GPIO

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(17,GPIO.OUT) # trigger

GPIO.setup(27,GPIO.IN) # ECHO

GPIO.setup(23,GPIO.OUT)#sensing led1

GPIO.setup(20,GPIO.OUT)#sensing led2

GPIO.setup(21,GPIO.OUT)#sensing led3

GPIO.setup(24,GPIO.OUT)#buzzer

GPIO.setup(22,GPIO.OUT)# alert back led

GPIO.setup(26,GPIO.OUT)#safe back led

GPIO.setup(10,GPIO.IN) #switch

GPIO.setup(5,GPIO.OUT) # trigger

GPIO.setup(6,GPIO.IN) # ECHO

# Define GPIO to LCD mapping

LCD\_RS = 9

LCD\_E = 25

LCD\_D4 = 8

LCD\_D5 = 7

LCD\_D6 = 13

LCD\_D7 = 19

# Define some device constants

LCD\_WIDTH = 16 # Maximum characters per line

LCD\_CHR = True

LCD\_CMD = False

LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line

LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants

E\_PULSE = 0.0005

E\_DELAY = 0.0005

GPIO.setup(LCD\_E, GPIO.OUT) # E

GPIO.setup(LCD\_RS, GPIO.OUT) # RS

GPIO.setup(LCD\_D4, GPIO.OUT) # DB4

GPIO.setup(LCD\_D5, GPIO.OUT) # DB5

GPIO.setup(LCD\_D6, GPIO.OUT) # DB6

GPIO.setup(LCD\_D7, GPIO.OUT) # DB7

def lcd\_init():

# Initialise display

lcd\_byte(0x33,LCD\_CMD) # 110011 Initialise

lcd\_byte(0x32,LCD\_CMD) # 110010 Initialise

lcd\_byte(0x06,LCD\_CMD) # 000110 Cursor move direction

lcd\_byte(0x0C,LCD\_CMD) # 001100 Display On,Cursor Off, Blink Off

lcd\_byte(0x28,LCD\_CMD) # 101000 Data length, number of lines, font size

lcd\_byte(0x01,LCD\_CMD) # 000001 Clear display

time.sleep(E\_DELAY)

def lcd\_byte(bits, mode):

# Send byte to data pins

# bits = data

# mode = True for character

# False for command

GPIO.output(LCD\_RS, mode) # RS

# High bits

GPIO.output(LCD\_D4, False)

GPIO.output(LCD\_D5, False)

GPIO.output(LCD\_D6, False)

GPIO.output(LCD\_D7, False)

if bits&0x10==0x10:

GPIO.output(LCD\_D4, True)

if bits&0x20==0x20:

GPIO.output(LCD\_D5, True)

if bits&0x40==0x40:

GPIO.output(LCD\_D6, True)

if bits&0x80==0x80:

GPIO.output(LCD\_D7, True)

# Toggle 'Enable' pin

lcd\_toggle\_enable()

# Low bits

GPIO.output(LCD\_D4, False)

GPIO.output(LCD\_D5, False)

GPIO.output(LCD\_D6, False)

GPIO.output(LCD\_D7, False)

if bits&0x01==0x01:

GPIO.output(LCD\_D4, True)

if bits&0x02==0x02:

GPIO.output(LCD\_D5, True)

if bits&0x04==0x04:

GPIO.output(LCD\_D6, True)

if bits&0x08==0x08:

GPIO.output(LCD\_D7, True)

# Toggle 'Enable' pin

lcd\_toggle\_enable()

def lcd\_toggle\_enable():

# Toggle enable

time.sleep(E\_DELAY)

GPIO.output(LCD\_E, True)

time.sleep(E\_PULSE)

GPIO.output(LCD\_E, False)

time.sleep(E\_DELAY)

def lcd\_string(message,line):

# Send string to display

message = message.ljust(LCD\_WIDTH," ")

lcd\_byte(line, LCD\_CMD)

for i in range(LCD\_WIDTH):

lcd\_byte(ord(message[i]),LCD\_CHR)

# Initialise display

lcd\_init()

lcd\_byte(0x80, LCD\_CMD)

lcd\_string(" WELCOME",LCD\_LINE\_1)

time.sleep(2)

lcd\_byte(0x01, LCD\_CMD)

lcd\_string("HAVE SAFE DRIVE",LCD\_LINE\_1)

lcd\_byte(0x01, LCD\_CMD)

def distance():

GPIO.output(5,True)

time.sleep(0.00001)

GPIO.output(5,False)

start=time.time()

stop=time.time()

while GPIO.input(6)==0:

start=time.time()

while GPIO.input(6)==1:

stop=time.time()

timetaken=stop-start

d=(timetaken\*34300)/2

return d

led=GPIO.PWM(23,40)

led1=GPIO.PWM(20,40)

led2=GPIO.PWM(21,40)

buzz=GPIO.PWM(24,50)

def buzzsound(h):

buzz.start(100)

buzz.ChangeDutyCycle(100)

time.sleep(.1)

GPIO.output(24,GPIO.LOW)

buzz.ChangeDutyCycle(100)

time.sleep(.1)

GPIO.output(24,GPIO.LOW)

time.sleep(.1)

buzz.ChangeDutyCycle(100)

time.sleep(.1)

GPIO.output(24,GPIO.LOW)

time.sleep(.1)

buzz.ChangeDutyCycle(0)

def sensing1():

GPIO.output(20,GPIO.LOW)

GPIO.output(21,GPIO.LOW)

GPIO.output(23,GPIO.LOW)

lcd\_byte(0x80, LCD\_CMD)

for i in range (0,2):

GPIO.output(23,GPIO.HIGH)

time.sleep(.1)

GPIO.output(23,GPIO.LOW)

time.sleep(.1)

GPIO.output(23,GPIO.HIGH)

time.sleep(.1)

buzzsound(h=20)

lcd\_string("CRITICALLY ALERT ",LCD\_LINE\_1)

lcd\_byte(0xC0, LCD\_CMD)

m=round(dist,2)

text='Dist in cm: {}'.format(m)

buzzsound(h=40)

lcd\_string(text,LCD\_LINE\_2)

for i in range (0,2):

GPIO.output(23,GPIO.HIGH)

time.sleep(.1)

GPIO.output(23,GPIO.LOW)

time.sleep(.1)

GPIO.output(23,GPIO.HIGH)

time.sleep(.1)

buzzsound(h=20)

lcd\_byte(0x80, LCD\_CMD)

def sensing2():

GPIO.output(23,GPIO.LOW)

GPIO.output(21,GPIO.LOW)

GPIO.output(20,GPIO.LOW)

lcd\_byte(0x80, LCD\_CMD)

lcd\_string(" ALERT ",LCD\_LINE\_1)

lcd\_byte(0xC0, LCD\_CMD)

m=round(dist,2)

text='Dist in cm: {}'.format(m)

lcd\_string(text,LCD\_LINE\_2)

for i in range (0,2):

GPIO.output(20,GPIO.HIGH)

time.sleep(.1)

GPIO.output(20,GPIO.LOW)

time.sleep(.1)

GPIO.output(20,GPIO.HIGH)

buzzsound(h=20)

lcd\_byte(0x80, LCD\_CMD)

def dis():

GPIO.output(17,True)

time.sleep(0.00001)

GPIO.output(17,False)

start=time.time()

stop=time.time()

while GPIO.input(27)==0:

start=time.time()

while GPIO.input(27)==1:

stop=time.time()

timetaken=stop-start

d=(timetaken\*34300)/2

return d

def sensing3():

lcd\_byte(0x80, LCD\_CMD)

lcd\_string(" BE ALERT ",LCD\_LINE\_1)

lcd\_byte(0xC0, LCD\_CMD)

m=round(dist,2)

text='Dist in cm: {}'.format(m)

lcd\_string(text,LCD\_LINE\_2)

GPIO.output(21,GPIO.HIGH)

time.sleep(.1)

lcd\_byte(0x80, LCD\_CMD)

def switch():

#print("switch pressed")

dist=distance()

#print ("DISTANCE=%.2f cm"%dist)

if(dist>0 and dist<10):

GPIO.output(22,GPIO.HIGH)

GPIO.output(26,GPIO.LOW)

lcd\_byte(0x80, LCD\_CMD)

lcd\_byte(0x01, LCD\_CMD)

lcd\_string(" ALERT CALL",LCD\_LINE\_1)

buzzsound(h=20)

lcd\_byte(0x01, LCD\_CMD)

else:

GPIO.output(26,GPIO.HIGH)

GPIO.output(22,GPIO.LOW)

lcd\_byte(0x80, LCD\_CMD)

lcd\_byte(0x01, LCD\_CMD)

lcd\_string(" SAFE DRIVE ",LCD\_LINE\_1)

time.sleep(2)

lcd\_byte(0x01, LCD\_CMD)

try:

while True:

switch()

dist=dis()

#print ("DISTANCE=%.2f cm"%dist)

for i in range (0,2):

GPIO.output(23,GPIO.LOW)

GPIO.output(20,GPIO.LOW)

GPIO.output(21,GPIO.LOW)

if(dist>0 and dist<=2):

sensing1()

elif (dist>2 and dist<=3):

sensing2()

elif (dist>4 and dist<=6):

sensing3()

except KeyboardInterrupt:

print "EXIT"