import RPi.GPIO as GPIO

import dht11

import time

import Adafruit\_DHT

import requests

import threading

from time import sleep

import I2C\_LCD\_driver

# Temperature and humidity sensor configuration

sensor = Adafruit\_DHT.AM2302

pin = 21

TEMP\_THRESHOLD = 10

myUploadAPI = "YC37QCCEP2UGSJBM"

baseURL = "https://api.thingspeak.com/update?api\_key=%s" % myUploadAPI

# Set up GPIO

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

# DHT11 instance setup

instance = dht11.DHT11(pin=21)

GPIO.setup(18, GPIO.OUT) # Set GPIO 18 as output for the buzzer

# Buzzer functions

def buzzer\_onoff():

GPIO.output(18, 1) # Output logic high/'1'

sleep(1)

GPIO.output(18, 0)

sleep(1)

# Telegram function

def send\_message\_to\_telegram(bot\_token, chat\_id, message):

base\_url = f'https://api.telegram.org/bot{bot\_token}/sendMessage'

data = {

'chat\_id': chat\_id,

'text': message,

}

try:

response = requests.post(base\_url, data=data)

if response.status\_code == 200:

print("Message sent successfully!")

else:

print(f"Failed to send message. Status code: {response.status\_code}")

except requests.RequestException as e:

print(f"An error occurred: {e}")

# Keypad thread

def keypad\_thread():

try:

LCD = I2C\_LCD\_driver.lcd()

LCD.backlight(0) # Turn backlight off

sleep(0.5)

LCD.backlight(1) # Turn backlight on

while True:

LCD.lcd\_clear()

LCD.lcd\_display\_string("Enter Password:", 1)

password = get\_password(LCD)

if password == "4550":

LCD.lcd\_clear()

LCD.lcd\_display\_string("Access Granted!", 1)

sleep(2)

else:

LCD.lcd\_clear()

LCD.lcd\_display\_string("Access Denied!", 1)

send\_message\_to\_telegram('6340765560:AAEqmtqOv2rKfn\_IlIhByxPDMW-iMf1uR00', '-1001720740162', 'Wrong Password!')

sleep(2)

except KeyboardInterrupt:

pass

# PIR, moisture, servo thread

def pir\_moisture\_servo\_thread():

GPIO.setup(17, GPIO.IN) # set GPIO 17 as input (PIR)

GPIO.setup(24, GPIO.OUT) # set GPIO 24 as output (LED)

GPIO.setup(4, GPIO.IN) # set GPIO 4 as input (Moisture sensor)

GPIO.setup(26, GPIO.OUT) # set GPIO 26 as output (servo motor)

sleep(2) # to allow sensor time to stabilize

PIR\_state = 0 # use this, so that only a change in state is reported

PWM = GPIO.PWM(26, 50) # set 50Hz PWM output at GPIO26

try:

while True:

# PIR Motion Sensor

if GPIO.input(17): # read a HIGH i.e. motion is detected

if PIR\_state == 0:

print('detected LOW i.e. no motion detected')

GPIO.output(24, 0)

PIR\_state = 1

else: # read a LOW i.e. no motion is detected

if PIR\_state == 1:

print('detected HIGH i.e. motion detected')

GPIO.output(24, 1)

sleep(2)

PIR\_state = 0

GPIO.output(24, 0)

# Moisture Sensor

if GPIO.input(4): # if read a high at GPIO 4, moisture present

print('detected HIGH i.e. moisture window close')

PWM.start(3) # 3% duty cycle

print('duty cycle: 3%', 3) # 3 o'clock position

sleep(2) # allow time for movement

else: # otherwise (i.e. read a low) at GPIO 4, no moisture

print('detected LOW i.e. no moisture window open')

PWM.start(12) # 13% duty cycle

print('duty cycle: 12%', 12) # 9 o'clock position

sleep(2) # allow time for movement

sleep(0.5) # to limit print() frequency

except KeyboardInterrupt:

pass

finally:

GPIO.cleanup()

# Temperature and buzzer thread

def temperature\_buzzer\_thread():

try:

while True:

temperature = Adafruit\_DHT.read\_retry(sensor, pin)[1]

if temperature is not None:

print('Temp={0:0.1f}\*C'.format(temperature))

if temperature > TEMP\_THRESHOLD:

buzzer\_onoff()

data={

'field1': temperature}

resp = requests.get(baseURL, params=data) # Upload to Thingspeak

else:

print('Failed to get reading. Try again!')

sleep(20) # Limit to upload rate, for free channel

except KeyboardInterrupt:

pass

MATRIX = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9],

['\*', 0, '#']

]

ROW = [6, 20, 19, 13]

COL = [12, 5, 16]

for i in range(3):

GPIO.setup(COL[i], GPIO.OUT)

GPIO.output(COL[i], 1)

for j in range(4):

GPIO.setup(ROW[j], GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

def get\_key():

for i in range(3):

GPIO.output(COL[i], 0)

for j in range(4):

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

return MATRIX[j][i]

GPIO.output(COL[i], 1)

return None

def get\_password(LCD):

password = ""

while len(password) < 4:

key = get\_key()

if key is not None:

password += str(key)

LCD.lcd\_display\_string("\*", 2, len(password))

sleep(0.2)

return password

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

keypad\_t = threading.Thread(target=keypad\_thread)

pir\_moisture\_servo\_t = threading.Thread(target=pir\_moisture\_servo\_thread)

temperature\_buzzer\_t = threading.Thread(target=temperature\_buzzer\_thread)

keypad\_t.start()

pir\_moisture\_servo\_t.start()

temperature\_buzzer\_t.start()

keypad\_t.join()

pir\_moisture\_servo\_t.join()

temperature\_buzzer\_t.join()