import threading

from flask import Flask, request, send\_from\_directory

import os

import subprocess, sys

import threading

import time

import json

from prometheus\_client import start\_http\_server, Summary, Gauge, start\_http\_server, Counter

import random

import time

import statistics

"""

import logging

logging.basicConfig(level=logging.DEBUG)

"""

instructionCode=''

numLines=0

globalData=1

tempDataPoints = []

garageOpen = False

ardStatus = 0

class garageLogicThread(threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

garageLogic()

def garageLogic():

global garageOpen

while True:

if len(tempDataPoints) > 10:

if statistics.mean(tempDataPoints[-2:]) < 62:

garageOpen = True

elif statistics.mean(tempDataPoints[-2:]) > 68:

garageOpen = False

"""

global tempDataPoints

global garageOpen

drasticChange = False

for val in tempDataPoints:

if (val-tempDataPoints[tempDataPoints.index(val)+1] >6):

drasticChange=True

last15 = tempDataPoints[-15:]

mean = statistics.mean(last15)

stdev = statistics.stdev(last15)

if (statistics.mean(tempDataPoints[0:15])>66):

if(mean<63 and drasticChange == True):

garageOpen = True

elif(mean<63):

garageOpen = True

else:

garageOpen = False

"""

class emailCheck (threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

dbCheck()

def dbCheck():

print('starting dbcheck')

class webserverThread(threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

webservice()

def webservice():

print('starting webserver')

app1 = Flask(\_\_name\_\_, static\_url\_path='/static')

@app1.route("/device", methods=['GET', 'POST'])

def device():

global numLines

global instructionCode

global globalData

global tempDataPoints

global ardStatus

ardStatus = 1

numVals = 0

if request.method == 'POST':

data = request.data.decode('ascii')

print("Data from Sensor is: "+data)

data = data[39:42]

globalData = float(data)

print ('globalData: '+str(globalData))

print("Distance Value is: "+data)

tempDataPoints.append(int(data))

for item in tempDataPoints:

numVals+=1

print(numVals)

if (numVals>=30):

del tempDataPoints[0]

"""

f = open("data.txt", "a+")

f.write(data + "\r\n")

f.close()

f = open("data.txt", "r")

for line in f:

numLines += 1

print(numLines)

if (numLines >= 5):

print('Database Full')

f.close()

if (numLines >= 5):

open('data.txt', 'w').close()

"""

return (instructionCode)

#@app1.route("/firmwareupdate", methods=['GET', 'POST'])

@app1.route("/firmwareupdate", methods=['GET', 'POST'])

def firmwareupdate():

print('Returning bin file to client')

#return app.send\_static\_file('GarageDoor.ino.generic.bin')

return send\_from\_directory('/home/pranaavn','GarageDoor.ino.generic.bin')

"""

return (action="/action\_page.php">

<input type="file" id="myFile" name="filename">

<input type="submit">

</form>)

"""

app1.run(host='0.0.0.0', port = 80, threaded = True, debug = False)

class instructionThread(threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

instructionThreadLoop()

def instructionThreadLoop():

global instructionCode

print('starting instructionThread')

while True:

instructionCode=input()

print("Instruction Code has been change to: "+instructionCode)

class exposeDataThread(threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

exposeData()

def exposeDataTemp():

app2 = Flask(\_\_name\_\_, static\_url\_path='/static')

@app2.route('/metrics', methods=['GET', 'POST'])

def metrics():

#return json.loads('{"GarageStatus":[{"status":"5"}]}')

#return '{"GarageStatus":[{"status":"5"}]}'

return 'metric\_name ["{" label\_name "=" `"` label\_value `"` { "," label\_name "=" `"` label\_value `"` } [ "," ] "}"] value [ timestamp ]'

print('running dbApp')

app2.run(host='0.0.0.0', port=9091, threaded=True, debug=False)

def exposeData():

global globalData

proxData = Gauge('ProximityData', 'Description of gauge')

proxData.set\_to\_current\_time()

proxData.set(globalData)

gStat = Gauge('GarageStatus', 'Description of gauge')

gStat.set\_to\_current\_time()

gStat.set(globalData)

flaskStat = Gauge('ArduinoStatus', 'Description of gauge')

flaskStat.set\_to\_current\_time()

flaskStat.set(0)

start\_http\_server(9091)

while True:

proxData.set(globalData)

if (garageOpen == False):

gStat.set(0)

elif (garageOpen == True):

gStat.set(1)

flaskStat.set(ardStatus)

time.sleep(10)

class statusCheckThread(threading.Thread):

def \_\_init\_\_(self, threadID, threadName):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.threadName = threadName

def run(self):

statusCheck()

def statusCheck():

while True:

if ardStatus == 1:

ardStatustatus = 0

time.sleep(20)

thread1 = webserverThread(1, 'webServerThread')

thread1.start()

thread2 = emailCheck(2, 'emailCheck')

thread2.start()

thread3 = instructionThread(3, 'instructionThread')

thread3.start()

thread4 = exposeDataThread(4, 'exposeDataThread')

thread4.start()

thread5 = garageLogicThread(5, 'garageLogicThread')

thread5.start()

thread6 = statusCheckThread(6, 'statusCheckThread')

thread6.start()