from flask import Flask, request, jsonify

import RPi.GPIO as GPIO

import torch

import cv2

import numpy as np

import logging

from werkzeug.serving import WSGIRequestHandler

# Constants

DEFAULT\_GREEN\_TIME = 30

BUFFER\_TIME = 7

MIN\_GREEN\_TIME = 15

MAX\_GREEN\_TIME = 90

VEHICLES\_PER\_SECOND = 4

# GPIO setup

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

# GPIO pin configuration

TRAFFIC\_LIGHTS = {

0: {'red': 11, 'yellow': 9, 'green': 10},

1: {'red': 12, 'yellow': 7, 'green': 8},

2: {'red': 16, 'yellow': 21, 'green': 20},

3: {'red': 19, 'yellow': 26, 'green': 13}

}

class TrafficController:

def \_\_init\_\_(self):

self.current\_signal = 0

self.is\_running = True

self.model = self.load\_model()

# Initialize GPIO

for signal in TRAFFIC\_LIGHTS.values():

for pin in signal.values():

GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, GPIO.LOW)

def load\_model(self):

try:

return torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

except Exception as e:

logging.error(f"Error loading YOLOv5 model: {e}")

return None

def update\_lights(self, states):

for signal\_id, state in enumerate(states):

pins = TRAFFIC\_LIGHTS[signal\_id]

if state == "green":

GPIO.output(pins['red'], GPIO.LOW)

GPIO.output(pins['yellow'], GPIO.LOW)

GPIO.output(pins['green'], GPIO.HIGH)

elif state == "yellow":

GPIO.output(pins['red'], GPIO.LOW)

GPIO.output(pins['yellow'], GPIO.HIGH)

GPIO.output(pins['green'], GPIO.LOW)

else: # red or inactive

GPIO.output(pins['red'], GPIO.HIGH)

GPIO.output(pins['yellow'], GPIO.LOW)

GPIO.output(pins['green'], GPIO.LOW)

def process\_image(self, image):

if self.model is None:

return 0

try:

results = model(image)

vehicle\_classes = [2, 3, 5, 7] # car, motorcycle, bus, truck

return sum([int(results.pred[0][:, -1].eq(class\_id).sum()) for class\_id in vehicle\_classes])

except Exception as e:

logging.error(f"Error processing image: {e}")

return 0

def cleanup(self):

GPIO.cleanup()

controller = TrafficController()

app = Flask(\_\_name\_\_)

@app.route('/process\_image', methods=['POST'])

def process\_image():

try:

if 'image' not in request.files:

return jsonify({'error': 'No image provided'}), 400

image\_file = request.files['image']

image\_bytes = image\_file.read()

nparr = np.frombuffer(image\_bytes, np.uint8)

image = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

vehicle\_count = controller.process\_image(image)

return jsonify({

'vehicle\_count': vehicle\_count,

'green\_time': max(MIN\_GREEN\_TIME, min(MAX\_GREEN\_TIME, (vehicle\_count // VEHICLES\_PER\_SECOND) \* 5))

})

except Exception as e:

return jsonify({'error': str(e)}), 500

@app.route('/update', methods=['POST'])

def update\_signals():

try:

data = request.json

controller.current\_signal = data['current\_signal']

controller.update\_lights(data['states'])

return jsonify({"status": "success"})

except Exception as e:

return jsonify({'error': str(e)}), 500

@app.route('/stop', methods=['GET'])

def stop\_signals():

try:

controller.is\_running = False

for pins in TRAFFIC\_LIGHTS.values():

for pin in pins.values():

GPIO.output(pin, GPIO.LOW)

return jsonify({"status": "success"})

except Exception as e:

return jsonify({'error': str(e)}), 500

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

try:

WSGIRequestHandler.protocol\_version = "HTTP/1.1"

app.run(host='0.0.0.0', port=5000, threaded=True)

finally:

controller.cleanup()