Traffic management system problem

IOT Phase:03

INTRODUCTION:

* A smart traffic management system utilizing camera data, communication and automated algorithms is to be developed to keep traffic flowing more smoothly. The aim is to optimally control the duration of green or red light for a specific traffic light at an intersection.
* The traffic signals should not flash the same stretch of green or red all the time, but should depend on the number of vehicles present. When traffic is heavy in one direction, the green lights should stay on longer; less traffic should mean the red lights should be on for a longer time interval.

# System analysis:

## A.EXISTING SYSTEM :

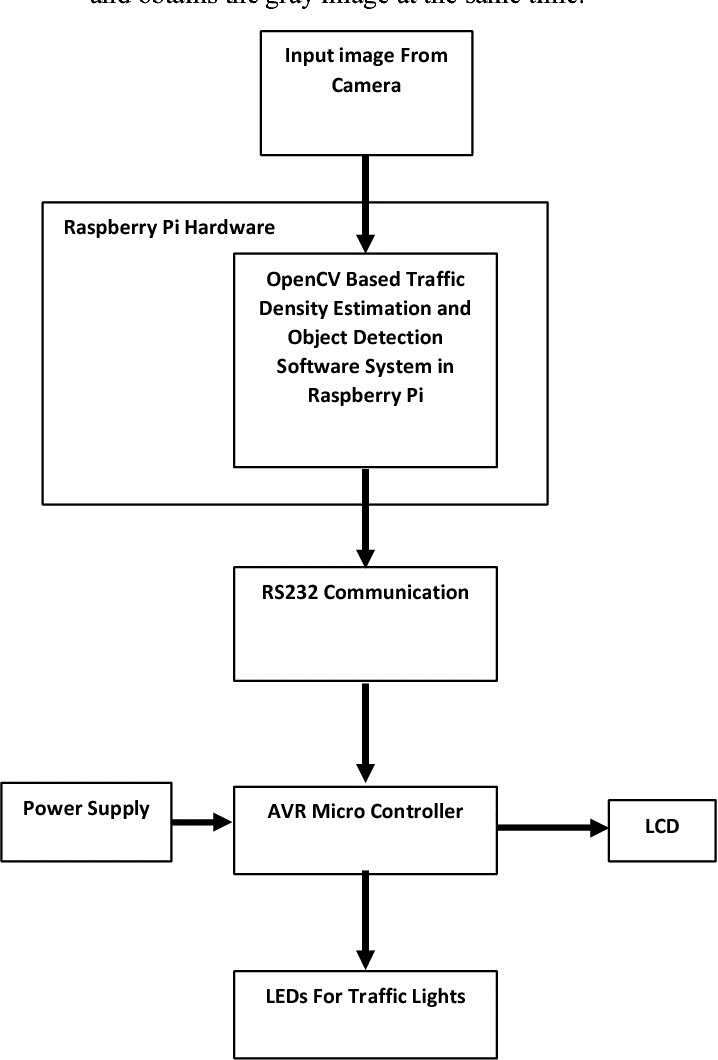
* The existing traffic system is generally controlled by the traffic police. The main drawback of this system controlled by the traffic police is that the system is not smart enough to deal with the traffic congestion. The traffic police official can either block a road for more time or let the vehicles on another road pass by i.e. the decision making may not be smart enough and it entirely depends on the official’s decision. Moreover, even if traffic lights are used, the time interval for which the vehicles will be shown a green or red signal is fixed.
* The database is used to train the module in order to better predict the changes in timings of the traffic light and its density. This communication is done using Wi-Fi. More specifically, the cloud server uses an equation that takes the data received (number of cars) as input then determines the time interval of LED’s needed for a smooth traffic flow. This calculated time is then compared to the current actual time of the LEDs (this data is saved in a database).

**B. PROPOSED** **SYSTEM :**

* The first and primary element of this system is the camera. The cameras interact with the physical environment, meaning vehicles presence or absence while the camera data is sent to the database for training the module for further prediction. The cameras transmit status based on the presence of vehicles near it.
* The camera transmits the data at specified time intervals to the processor (raspberry pi), it processes the data and sends the processed data to the controller. The computed data from Raspberry pi is then transmitted to the controller through Wi-Fi connectivity. The controller makes use of the collected data to perform the Intelligent Traffic routing.

# **SYSTEM DESIGN:**

* The Architecture system consists of six modules:
* 1) Raspberry Pi
* 2) LED lights which are used for the purpose of signal.
* 3) Traffic cameras which are used for monitoring traffice
* 4)Micro controllers.



**SYSTEM IMPLEMENTATION:**

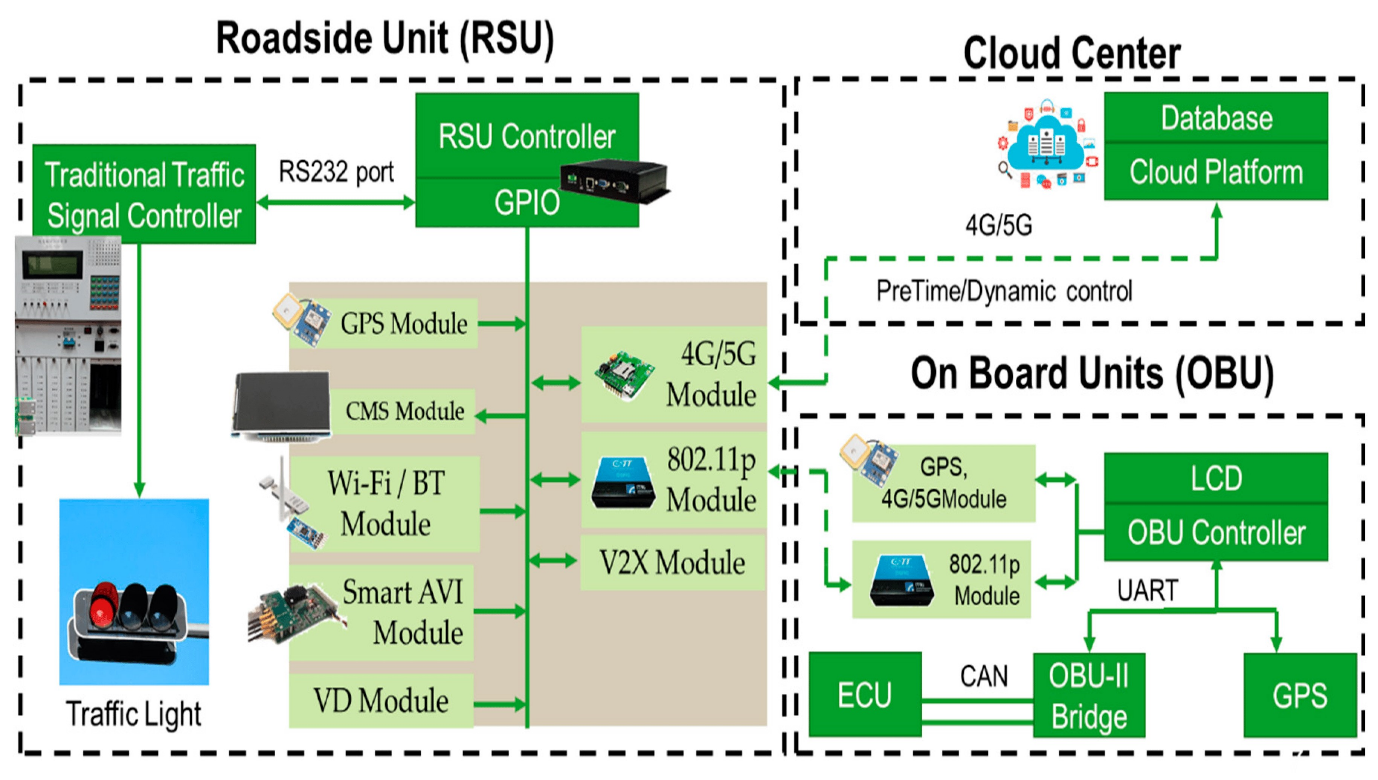
1. Camera: Continuously record traffic video.

2. Read Image: Read frames of the traffic image.

3. Grayscale Image Conversion: It converts colour image to grayscale image.This method is based on different colour transforms. According to the R, G, B value in the image, it calculates the value of grayscales and converts the image into a grayscale image.

4. Image Binarization: Grayscale image is converted into black and white image.

5. Traffic Signal Control: Based on vehicle count signal timings are changed and the respective LED glows.



Steps for controlling traffic light 2:

1. Initialize System

2. Configure ESP 8266 module for multi access point through AT commands

3. Connect WI-FI module to WI-FI network

4. Start UDP local port in WI-FI module

5. Establish UDP connection to Raspberry pi

6. Wait for data

7. Change traffic light signal 2 depending upon their received data from rasberry Pi.

## Data Sources:

Various data sources are used to monitor real-time traffic, including:

* **Traffic Cameras:**  Cameras are placed at strategic locations to capture real-time images and videos of road conditions.
* **Inductive Loop Sensors:** These sensors are embedded in the road surface to detect the presence and movement of vehicles.
* **GPS Data:** Many modern vehicles are equipped with GPS devices that transmit real-time location and speed data.
* **Mobile Apps:** Crowdsourced data from navigation apps like Waze and Google Maps provide valuable real-time traffic information.
* **Weather Sensors:** Weather conditions can impact traffic, so data from weather stations is also considered.
* **Data Collection and Analysis:** The data collected from these sources is processed and analyzed to provide insights into traffic conditions. This involves identifying traffic congestion, accidents, road closures, and other relevant information.
* **Traffic Management:** Transportation authorities and traffic management centers use this data to make real-time decisions, such as adjusting traffic signals, implementing detours, or dispatching emergency services in the case of accidents.
* **Public Information:**  Real-time traffic data is often made available to the public through websites, mobile apps, and digital signs on the road. This helps drivers plan their routes and avoid congested areas.Set up your sensors to collect real-time data.
* **Data Processing and Analysis:** Use Python libraries for data processing, such as OpenCV for image processing or NumPy and Pandas for data analysis.Implement algorithms to detect traffic conditions, congestion, accidents, etc., from the sensor data.
* **Data Transmission:** Establish a connection to a central server or cloud platform using Python libraries like MQTT, HTTP, or other IoT communication protocols.Send the processed traffic data to the server for real-time analysis.
* **Real-Time Visualization and Monitoring:** Set up a dashboard or web application to display real-time traffic information using a web framework like Flask or Django.
* **Control Logic:** Implement control logic in your Python script to make real-time decisions based on the analyzed traffic data. For example, you might change traffic signal timings or alert authorities in case of accidents.
* **Alerts and Notifications:** Use Python libraries to send alerts and notifications to relevant authorities or traffic management systems if abnormal situations are detected.
* **Logging and Storage:** Implement data logging and storage mechanisms to keep historical traffic data for analysis and reporting.
* **Security and Authentication:** Ensure that your IoT device and data transmission are secure. Use appropriate authentication and encryption methods.
* **Testing and Deployment:** Test your Python script thoroughly in a controlled environment before deploying it to a live traffic management system. Test individual components or units of your code in isolation to ensure they work as expected. Python has several testing frameworks like unittest, pytest, and nose to facilitate unit testing.
* **Maintenance and Updates:** Regularly update and maintain your Python script to ensure that it continues to operate efficiently and adapt to changing traffic conditions and requirements.

**Python script for send an real time traffic management to traffic problem:**

import paho.mqtt.client as mqtt

import json

import time

import random

mqtt\_broker = "mqtt.example.com" # Replace with your MQTT broker address

mqtt\_topic = "traffic\_data"

def generate\_traffic\_data():

vehicle\_count = random.randint(0, 100)

average\_speed = random.uniform(20, 80)

return {

"vehicle\_count": vehicle\_count,

"average\_speed": average\_speed,

"timestamp": int(time.time())

}

client = mqtt.Client("TrafficDevice")

client.connect(mqtt\_broker)

while True:

traffic\_data = generate\_traffic\_data(

payload = json.dumps(traffic\_data)

client.publish(mqtt\_topic, payload)

print(f"Published data: {payload}"

time.sleep(30)

client.disconnect()

**Using web development technologies to create a platform that displays the real time traffic information:**

<!DOCTYPE html>

<html>

<head>

<title>Real-Time Traffic Monitoring System</title>

<link rel="stylesheet" href="style.css">

<script src="script.js"></script>

</head>

<body>

<div id="map"></div>

<script>

var map = new google.maps.Map(document.getElementById('map'), {

zoom: 12,

center: {lat: 37.7833, lng: -122.4167}

});

function updateTraffic() {

var trafficData = getTrafficData();

map.trafficLayer.setTrafficData(trafficData);

}

setInterval(updateTraffic, 5000); // Update the traffic every 5 seconds

</script>

</body>

</html>

import com.google.maps.GeoApiContext;

import com.google.maps.TrafficData;

import com.google.maps.TrafficLayer;

public class RealTimeTrafficMonitoringSystem {

public static void main(String[] args) {

GeoApiContext context = new GeoApiContext().setApiKey("YOUR\_API\_KEY");

TrafficLayer trafficLayer = new TrafficLayer(context);

map.addOverlay(trafficLayer);

new Timer().scheduleAtFixedRate(new TimerTask() {

public void run() {

TrafficData trafficData = trafficLayer.getTrafficData();

trafficLayer.setTrafficData(trafficData);

}

}, 0, 5000);

}

}

**Python script for updates android IOS and mobile apps for real time traffic monitering and route configuration:**

import android

import requests

import json

class MainActivity(android.widget.Activity):

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

super().\_\_init\_\_()

self.map = android.widget.MapView(self)

self.setContentView(self.map)

self.traffic\_layer = android.maps.TrafficLayer()

self.map.addOverlay(self.traffic\_layer)

self.update\_traffic\_data()

def update\_traffic\_data(self):

traffic\_data = requests.get('https://maps.googleapis.com/maps/api/traffic/json?parameters=sensor=true&key=YOUR\_API\_KEY').json()

self.traffic\_layer.setTrafficData(traffic\_data)

# Schedule the traffic data to be updated every 5 seconds

android.os.Handler().postDelayed(self.update\_traffic\_data, 5000)

def on\_search\_query(self, query):

# Get the user's destination address

destination\_address = query

directions\_api\_request = android.maps.DirectionsApiRequest()

directions\_api\_request.setOrigin(self.map.getMyLocation())

directions\_api\_request.setDestination(destination\_address)

directions\_task = android.maps.DirectionsTask()

directions\_task.setDirectionsApiRequest(directions\_api\_request)

directions\_task.execute()

def on\_directions\_task\_completed(self, directions\_result):

self.map.addPolyline(directions\_result.get\_routes()[0].get\_geometry())

self.map.move\_camera(android.maps.CameraUpdateFactory.new\_latLng\_zoom(directions\_result.get\_routes()[0].get\_legs()[0].get\_steps()[0].get\_start\_location(), 15))

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

android.app.ActivityThread.main(lambda: MainActivity())

# CONCLUSION:

* Smart Traffic Management System has been developed by using multiple features of hardware components in IoT. Traffic optimization is achieved using IOT platform for efficient utilizing allocating varying time to all traffic signal according to available vehicles count in road path. Smart Traffic Management System is implemented to deal efficiently with problem of congestion and perform re-routing at intersections on a road.
* This research presents an effective solution for rapid growth of traffic flow particularly in big cities which is increasing day by day and traditional systems have some limitations as they fail to manage current traffic effectively. Keeping in view the state-of-the-art approach for traffic management systems, a smart traffic management system is proposed to control road traffic situations more efficiently and effectively.
* It changes the signal timing intelligently according to traffic density on the particular roadside and regulates traffic flow by communicating with local server more effectively than ever before. The decentralized approach makes it optimized and effective as the system works even if a local server or centralized server has crashed. The system also provides useful information to higher authorities that can be used in road planning which helps in optimal usage of resource.

**DONE BY,**

**SIVAHARI.S**

**712521104309**