By,

1.Ariharasudan M - au721421106007

2.Amuruth R - au721421106006

3.Darshan M - au721421106009

4.Karthiban K - au721421106027

5.Kishore G J - au721421106033

Abstract

Final part of the project

TRAFFIC MANAGEMENT SYSTEM

USING IoT

# **OVERVIEW**:

Traffic management is the practice of controlling and regulating the movement of vehicles and pedestrians on roadways to ensure safe, efficient, and orderly traffic flow. It encompasses traffic signals, signs, rules, road design, enforcement, technology, and strategies to reduce congestion, accidents, and improve transportation efficiency.Top of Form

# **OBJECTIVE** **:**

* **Traffic Flow Optimization**: Predict congestion to implement traffic flow optimization strategies, such as adjusting traffic signal timings, managing lane configurations, and implementing dynamic speed limits to alleviate congestion.
* **Dynamic Routing**: Provide real-time traffic congestion information to drivers through navigation apps and electronic road signs, enabling them to choose alternative routes and reduce congestion on heavily congested roads.
* **Public Transportation Integration**: Integrate congestion prediction with public transportation systems to optimize bus or train schedules and routing to reduce congestion and improve transit efficiency.
* **Incident Management**: Detect and predict congestion caused by accidents, road closures, or other incidents, allowing for quicker incident response and diversion of traffic.
* **Intersection Control**: Use congestion prediction to optimize traffic signal timings at intersections, reducing wait times and minimizing congestion buildup.

**IoT SENSORS:**

* **TRAFFIC FLOW: The Inductive Loop Detector (ILD) sensor** is one of the most common sensors in traffic management. It is used for collecting traffic flow, vehicle's occupancy, length, and speed.
* **CONGESTION IN REAL TIME: The ultrasonic sensors** are used to detect vehicles, and the density levels of a given road are sent to an LCD, and the data sent to the server for later usage. In similar research, the authors proposed an ultrasonic sensor-based system model specifically for road intersections.

**Real-Time Transit Information Platforms:**

**Mobile Applications:**

* Mobile apps provide real-time traffic updates, alternative routes, and transit information to help commuters plan their journeys.

**Dynamic Message Boards:**

* Electronic message boards along highways display real-time information about traffic conditions and advisories.

**Web-based Traffic Portals:**

* Interactive websites offer live traffic maps, incident reports, and travel time estimations for informed decision-making.

**Integration Approach:**

* **Seamless Access**: Design a web-based platform and user-friendly mobile apps to provide the public with easy access to real-time traffic information anytime, anywhere.
* **Partner Integration:** Collaborate with transportation authorities and agencies to seamlessly integrate our traffic management system into existing infrastructure and information frameworks.
* **Sustainable Future:** Contribute to building smarter cities by leveraging our traffic management system to optimize transportation systems, reduce greenhouse gas emissions, and improve overall urban livability.

**Algorithm for congestion prediction:**

* Data Collection:

To predict traffic congestion, relevant data needs to be collected. We will explore various sources of data, such as traffic cameras, IoT devices, and GPS data. Additionally, we'll discuss the types of data collected, such as traffic volume, speed, and historical patterns. Preprocessing techniques will also be explored to ensure data accuracy and consistency.

* Feature Extraction:

Extracting meaningful features is essential for accurate congestion prediction. We will delve into the selection of relevant features from the collected data and explore different techniques used for extraction. Additionally, we'll discuss the process of feature engineering to enhance prediction performance.

* Model Selection:

Choosing the right model is critical for congestion prediction accuracy. We will explore different types of models, such as regression, classification, and time series forecasting. Evaluation criteria for model selection will be discussed, and a comparison of various models will be presented to determine the most suitable choice.

* Model Training and Validation:

Once the model is selected, it needs to be trained and validated. We will explore the process of splitting data into training and validation sets to ensure unbiased performance evaluation. The training process of the selected model, along with validation and fine-tuning techniques, will be discussed in detail.

* Congestion Prediction and Visualization:

Real-time congestion prediction is crucial for traffic management. We will discuss the techniques used for predicting congestion based on the trained model. Additionally, visualization techniques will be explored to present the prediction results in an intuitive and actionable manner. Integration of prediction with traffic management systems will also be highlighted.

# **CIRCUIT DIAGRAM:**

# **COMPONENTS:**

1. Arduino UNO R3 CH340
2. LCD display
3. Ultrasonic Sensor HC-SR04
4. Male to Male Jumper Wires

# **Code:**

import RPi.GPIO as GPIO

import time

echoPin = 12

trigPin = 13

GPIO.setmode(GPIO.BCM)

GPIO.setup(trigPin, GPIO.OUT)

GPIO.setup(echoPin, GPIO.IN)

def measure\_distance():

GPIO.output(trigPin, GPIO.LOW)

time.sleep(0.2)

GPIO.output(trigPin, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trigPin, GPIO.LOW)

while GPIO.input(echoPin) == 0:

pulse\_start = time.time()

while GPIO.input(echoPin) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance\_cm = pulse\_duration \* 34300 / 2

distance\_inch = pulse\_duration \* 13503.9 / 2

return distance\_cm, distance\_inch

try:

while True:

distance\_cm, distance\_inch = measure\_distance()

print("Distance: {} cm".format(distance\_cm))

print("Distance: {} inch".format(distance\_inch))

# lcd.print("Distance: {} cm".format(distance\_cm))

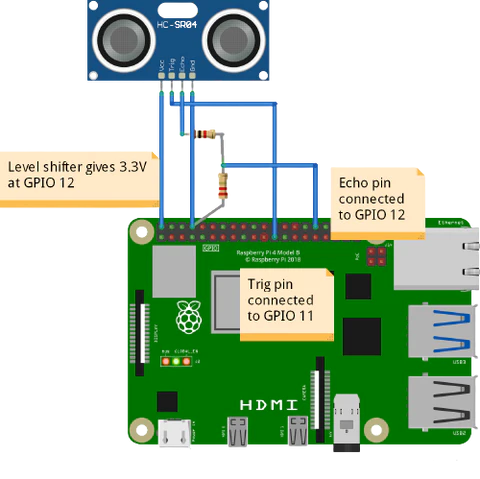
# lcd.print("Distance: {} inch".format(distance\_inch))

time.sleep(1)

except KeyboardInterrupt:

GPIO.cleanup()}

# RASPBERRY PI INTEGRATION:



CODE:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

TRIG\_PIN = 11

ECHO\_PIN = 12

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

GPIO.output(TRIG\_PIN, GPIO.LOW)

time.sleep(2)

GPIO.output(TRIG\_PIN, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, GPIO.LOW)

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = (pulse\_duration \* 34000) / 2

distance = round(distance, 2)

print("Object is at", distance, "cm from the ultrasonic sensor")

GPIO.cleanup()

**PLATFORM:**

* **Operating System Platforms:** Examples include Windows, macOS, and Linux, which provide the foundation for running various applications.
* **Cloud Computing Platforms:** Providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud offer cloud platforms that allow businesses to build, deploy, and scale applications in the cloud.
* **Mobile App Development Platforms:** iOS and Android are platforms for developing mobile applications, each with its own set of development tools and environments.
* **Web Development Platforms:** Web development platforms like WordPress, Drupal, and Ruby on Rails provide tools and frameworks for building websites and web applications.
* **E-commerce Platforms:** Platforms like Shopify and Magento are designed specifically for creating and managing online stores.
* **IoT (Internet of Things) Platforms:** These platforms help developers build and manage IoT applications and devices.

# **WEB DEVELOPMENT PLATFORM:**

# **Web Development Frameworks:** Use frameworks like Django, Ruby on Rails, or Express.js to expedite the development process.

# **Content Management Systems (CMS):** If your website will involve content management, consider using platforms like WordPress, Joomla, or Drupal.

# **Database Management Systems:** Tools like MySQL, PostgreSQL, and MongoDB are commonly used for database management.

# **Version Control Systems:** Git is essential for tracking changes in the codebase and collaborating with a team.

# **Development Environments:** IDEs (Integrated Development Environments) like Visual Studio Code, PyCharm, and Sublime Text are popular for web development.

**CODE FOR WEBSITE :**

1. **Server-Side Python Code:**

**from flask import Flask, render\_template, request**

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

**# Store received sensor data in memory for demonstration.**

**sensor\_data = []**

**@app.route('/')**

**def index():**

**return render\_template('index.html', sensor\_data=sensor\_data)**

**@app.route('/receive\_data', methods=['POST'])**

**def receive\_data():**

**data = request.get\_json()**

**sensor\_data.append(data)**

**return 'Data received and stored'**

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

**app.run(debug=True)**

1. **HTML and Template for Web Page:**

**<!-- templates/index.html -->**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Traffic Congestion Monitor</title>**

**</head>**

**<body>**

**<h1>Traffic Congestion Monitor</h1>**

**<div id="congestion-display">**

**<table>**

**<tr>**

**<th>Timestamp</th>**

**<th>Congestion Level</th>**

**</tr>**

**{% for data in sensor\_data %}**

**<tr>**

**<td>{{ data.timestamp }}</td>**

**<td>{{ data.congestion\_level }}</td>**

**</tr>**

**{% endfor %}**

**</table>**

**</div>**

**</body>**

**</html>**

1. **Python Code for Bluetooth (Bluetooth LE):**

**import pygatt**

**import json**

**def send\_data\_via\_bluetooth(data):**

**device\_address = 'your\_device\_address'**

**client = pygatt.GATTToolBackend()**

**client.start()**

**try:**

**device = client.connect(device\_address)**

**device.char\_write('your\_characteristic\_uuid', json.dumps(data).encode('utf-8'))**

**finally:**

**client.stop()**

**# Usage:**

**data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30}**

**send\_data\_via\_bluetooth(data)**

1. **Python Code for Wi-Fi (HTTP Post Request):**

**import requests**

**import json**

**def send\_data\_via\_wifi(data):**

**server\_url = 'http://your-server-ip-or-domain/receive\_data'**

**headers = {'Content-Type': 'application/json'}**

**response = requests.post(server\_url, data=json.dumps(data), headers=headers)**

**# Usage:**

**data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30}**

**send\_data\_via\_wifi(data)**

**THANK YOU**