TEAM MEMBER S.VIGNESH

Phase 5 – Document Submission 510421106057

Project – Public Transport Optimization

CONTENT

CHAPTER --------------------------------------------- I

CHAPTER --------------------------------------------- 2

CHAPTER --------------------------------------------- 3

CHAPTER --------------------------------------------- 4

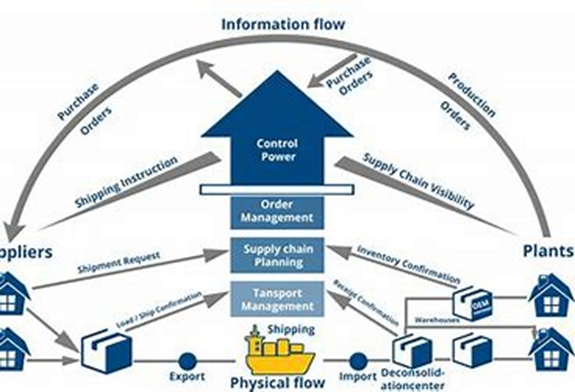
CHAPTER --------------------------------------------- 5

CHAPTER ----- 1

ABSTRACT

A significant amount of research work carried out on traffic management systems, but intelligent traffic monitoring is still an active research topic due to the emerging technologies such as the Internet of Things (IoT) and Artificial Intelligence (AI). The integration of these technologies will facilitate the techniques for better decision making and achieve urban growth. However, the existing traffic prediction methods mostly dedicated to highway and urban traffic management, and limited studies focused on collector roads and closed campuses. Besides, reaching out to the public, and establishing active connections to assist them in decision making is challenging when the users are not equipped with any smart devices.

PUBLIC TRANSPORT OPTIMIZATION



OBJECTIVE:

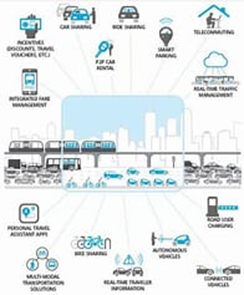
The objective of public transport optimization is to make the transportation network as efficient as possible by applying data and technology. It involves analyzing shipments, rates, lanes, and constraints to generate realistic load plans, trip plans, driver selection, and tour planning. The goal is to reduce costs, improve productivity, and increase customer satisfaction.

Phase 1: Public Transport Optimization

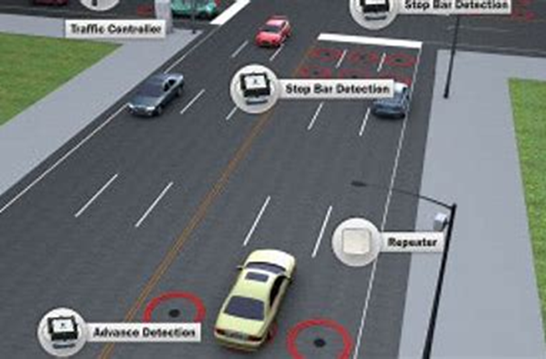
DEFINITION:

The public transportation sector in developing countries, in general, has faced several challenges and been affected by many problems. One of the most relevant problems in this context is the surplus or deficit of public transportation vehicles and trips for different routes due to the improper estimation of the passenger demand as well as the number and size of vehicles needed to meet this demand. Moreover, in some areas, public transportation services do not run according to a fixed timetable or even clear operating hours, with these hours changing irregularly and without notice.

Smart transportation systems improve traffic flow and safety, reducing travel times and fuel consumption. It is imperative to use IoT infrastructures more and seamlessly integrate information and communication technologies (ICT) to create a sustainable, intelligent transportation system.



IOT SENSOR DESIGN:



A device that provides a usable output in response to a specified measurement.

The sensor attains a physical parameter and converts it into a signal suitable for processing the characteristics of any device or material to detect the presence of a particular physical quantity.

Real time traffic:

Real-time traffic monitoring systems play a key role in the transition toward smart cities. A considerable amount of literature has been published on intelligent traffic management systems based on the IoT paradigm. Autonomous traffic sensing is at the heart of smart city infrastructures, wherein smart wireless sensors are used to measure traffic flow, predict congestion, and adaptively control traffic routes. Doing so effectively provides an awareness that enables more efficient use of resources and infrastructure.

•This research proposes an IoT based system model to collect, process, and store real-time traffic data.

•The objective is to provide real-time traffic updates on traffic congestion

•The early-warning messages will help citizens to save their time, especially during peak hours and unusual traffic incidents through roadside message units.

INTRODUCTION:-

In an era marked by rapid urbanization and the pressing need for sustainable solutions, the optimization of public transportation systems has become paramount. Leveraging the power of the Internet of Things IOT holds immense promise in revolutionizing how we design, manage, and experience public transportation. IOT, with its interconnected network of devices and sensors, offers an unprecedented opportunity to enhance the efficiency, safety, and accessibility of public transport systems. This introduction explores the potential of IOT-driven solutions in optimizing public transportation, from real-time tracking and predictive maintenance to data-driven decision-making, ultimately paving the way for smarter and more sustainable urban mobility.

Public transportation plays a pivotal role in shaping modern cities and communities, providing a lifeline for millions of people daily. However, the efficiency and effectiveness of these systems are often marred by challenges such as congestion, delays, and operational inefficiencies. In today's data-driven world, the integration of the Internet of Things IOT has emerged as a transformative force, promising to revolutionize public transport optimization. By embedding sensors, connectivity, and real-time data analytics into transportation networks, IOT offers the potential to enhance the reliability, accessibility, and sustainability of public transit systems. This introduction delves into the compelling realm of IOT -driven solutions, exploring how they can re-imagine and elevate the future of public transportation for the benefit of commuters and cities alike.

Public transport optimization is a vital endeavor in today's urban landscape, where efficient and sustainable mobility is a key driver of economic development and quality of life. This pursuit revolves around the strategic improvement of public transportation systems, encompassing buses, trains, trams, and more, to ensure they operate seamlessly, punctually, and eco-consciously.

# ALGORITHM:

Step 1: Define Objectives and Constraints.

Determine the goals of your public transport optimization, such as minimizing travel time, reducing congestion, or maximizing efficiency.

Identify constraints, including budget limitations, available resources, and regulatory requirements.

Step 2: Data Collection and Preparation.

Gather data about the transportation network, including routes, stops, schedules, and passenger demand.

Ensure that data is accurate, up-to-date, and properly formatted.

Step 3: Network Modeling.

Create a network representation of the transportation system, including nodes for stops and edges for routes between them.

Assign attributes to nodes and edges, such as travel times, capacities, and costs.

Step 4: Algorithm Selection.

Choose optimization algorithms that align with your objectives and constraints Machine Learning (e.g., neural networks): For predicting demand and optimizing schedules dynamically.

Step 5: Implementation

Develop the program using a programming language of your choice the chosen algorithms and data structures.

Create user interfaces if necessary for input and output.

Step 6: Testing and Validation

Test the program with real or simulated data to ensure it produces meaningful and efficient results.

Validate the results against known benchmarks or industry standards.

Step 7: Optimization and Fine-Tuning

Continuously improve the program by fine-tuning parameters, algorithms, and data sources.

Optimize for scalability and real-time updates if required.

Step 8: Deployment and Integration

Deploy the program in the operational environment, whether it's a city's transportation department or a private company.

Integrate it with existing transportation systems and databases.

Step 9: Monitoring and Maintenance

Implement monitoring systems to track the performance of the public transport optimization program

DATA PROCESSING:

Analyze the collected data to determine factors like vehicle location, occupancy, and route

# Sample code to process GPS data

def process\_gps\_data(gps\_data):

# Parse GPS data and determine vehicle location

# Calculate estimated time of arrival (ETA) at stops

# Detect any anomalies or issues

CHAPTER ------ 2

PROCEDURE:-

1. IoT Devices and Data Collection:

Utilize IoT devices like GPS trackers, sensors, or simulated data sources for a student project.

Simulate data transmission from these devices to a central server.

2. Data Processing and Analysis:

Develop a server-side application using web development technologies to process and analyze the data.

3. Web Development:

Create a web-based platform for students, mimicking the real-world application.

Use a combination of HTML, CSS, and JavaScript to build the web user interface.

4. User Interfaces:

Develop simple web pages with intuitive interfaces that students can interact with.

Use HTML forms and JavaScript for data input and display.

5. Data Visualization:

Implement basic data visualization using JavaScript libraries like Chart.js to show simulated data trends.

6. Alerts and Notifications:

Simulate alerting and notification mechanisms within the web application.

7. User Authentication and Security:

For a basic student project, you can skip user authentication, but implement basic security practices for data handling.

8. Database Management:

Use a simplified database or data storage system (e.g., local storage) to mimic data storage.

9.Testing and Quality Assurance:

Ensure that the web application is bug-free and functions as expected for the student project.

Connecting Mobile app with Public Transport Optimization:

Connecting a mobile app to a Public Transport Optimization IoT project involves setting up a communication pathway between the mobile app and the IoT devices or backend server. Here's a high-level overview of the steps to achieve this connection:

1.Define App Requirements:

Determine the specific functionalities and features you want to offer in the mobile app. These could include real-time tracking, route information, alerts, and notifications.

2.Choose Development Platforms:

Decide whether you want to develop native apps for specific platforms (e.g., iOS and Android) or use cross-platform frameworks like React Native, Flutter, or Xamarin to build the app for multiple platforms simultaneously.

3.Select Development Tools:

Choose the development tools and integrated development environments (IDEs) suitable for the selected platform and framework.

4.Develop Mobile App:

Create the mobile app using the chosen platform and development tools. Integrate user interfaces, real-time tracking, and any other relevant features.

5.Implement Communication:

To connect the app with IoT devices or the backend server:

1.APIs: Develop RESTful or WebSocket APIs on the backend server to expose data and functionality to the app.

2.Mobile App Client: Implement communication within the app using libraries like fetch (for HTTP requests), Web Sockets, or specialized IoT communication protocols (e.g., MQTT).

6.Authentication and Security:

Implement user authentication mechanisms to ensure secure access to the app.

Ensure data security by using encryption and authentication methods, especially when dealing with sensitive data.

7.Real-Time Data Retrieval:

Enable the app to request and display real-time data from the IoT devices, such as vehicle location, passenger count, and alerts.

8.User-Friendly Interfaces:

Create user-friendly interfaces within the app to display real-time information and allow users to interact with the Public Transport Optimization system.

9.Push Notifications:

Implement push notification services to send real-time alerts and updates to the mobile app users. This could be for service delays, route changes, or other relevant information.

10.Testing:

Thoroughly test the app's functionality, performance, and user experience to ensure it works seamlessly with the IoT system.

11.Deployment:

Deploy the mobile app to app stores (e.g., Apple App Store, Google Play Store) for public or limited access.

12.Maintenance and Updates:

Continuously monitor the app's performance and user feedback. Address issues, release updates, and add new features as needed.

Objective:

The objective of this project is to optimize public transportation systems by leveraging the Internet of Things (IoT) technology. The project aims to enhance the efficiency, reliability, and overall user experience of public transportation services by collecting and analyzing real-time data from IoT devices deployed in vehicles, infrastructure, and passenger hubs.

IoT Device Setup:

Vehicle Sensors: IoT sensors will be installed in public transport vehicles, including buses, trams, and trains. These sensors will collect data such as vehicle location, speed, fuel consumption, passenger count, and engine health.

Passenger Counters: IoT cameras or sensors will be placed at vehicle entry and exit points to track passenger counts and occupancy in real-time.

Infrastructure Sensors: IoT sensors will be deployed at bus stops, train stations, and other passenger hubs to collect data on passenger foot traffic, environmental conditions (e.g., temperature, humidity), and vehicle arrival times.

Platform Development:

The IoT data collected from the devices will be transmitted to a centralized platform for analysis and decision-making. The platform will consist of the following components:

IoT Gateway: Data from the IoT devices will be sent to a central gateway for aggregation and forwarding. This gateway will be responsible for data preprocessing and transmission.

Cloud Data Storage: The preprocessed data will be stored in a cloud-based database for real-time and historical analysis.

Data Analysis and Prediction: Machine learning models will be developed to analyze the data and make predictions about vehicle schedules, maintenance needs, and passenger demand.

Dashboard: A user-friendly dashboard will be created for transportation authorities and commuters to access real-time information about vehicle locations, passenger counts, and estimated arrival times.

Code Implementation:

The code for this project will be written in various programming languages depending on the component:

IoT Device Code: Each IoT device will have its own code to read sensor data and transmit it to the IoT gateway. This code may be written in C, Python, or other suitable languages.

IoT Gateway Code: The gateway will have code to aggregate data, perform preprocessing, and securely transmit it to the cloud. It may use MQTT, HTTP, or other communication protocols.

Cloud Data Storage Code: This involves setting up and managing databases, which can be done using platforms like AWS, Azure, or Google Cloud.

Data Analysis and Prediction Code: Machine learning models will be implemented using Python, along with libraries like TensorFlow or scikit-learn.

Dashboard Code: Web-based dashboards can be developed using HTML, CSS, and JavaScript, along with frameworks like React or Angular.

Explanation in Detail:

The system will collect real-time data on vehicle locations, passenger counts, and environmental conditions. This data will be analyzed to optimize transportation routes, schedules, and vehicle maintenance. Passengers will have access to accurate arrival time predictions, reducing wait times and improving their experience. Transportation authorities will have tools to make data-driven decisions to enhance the efficiency and reliability of their services.

The project's success will lead to more sustainable, convenient, and user-friendly public transportation systems, benefiting both commuters and the environment.

URL for Tinkercad Project of Passenger Counting :

https://www.tinkercad.com/things/dAuGJH1r0NS-copy-of-auto-counting-of-passenger/editel?tenant=circuits

OVERVIEW OF THE OPTIMIZATION:-

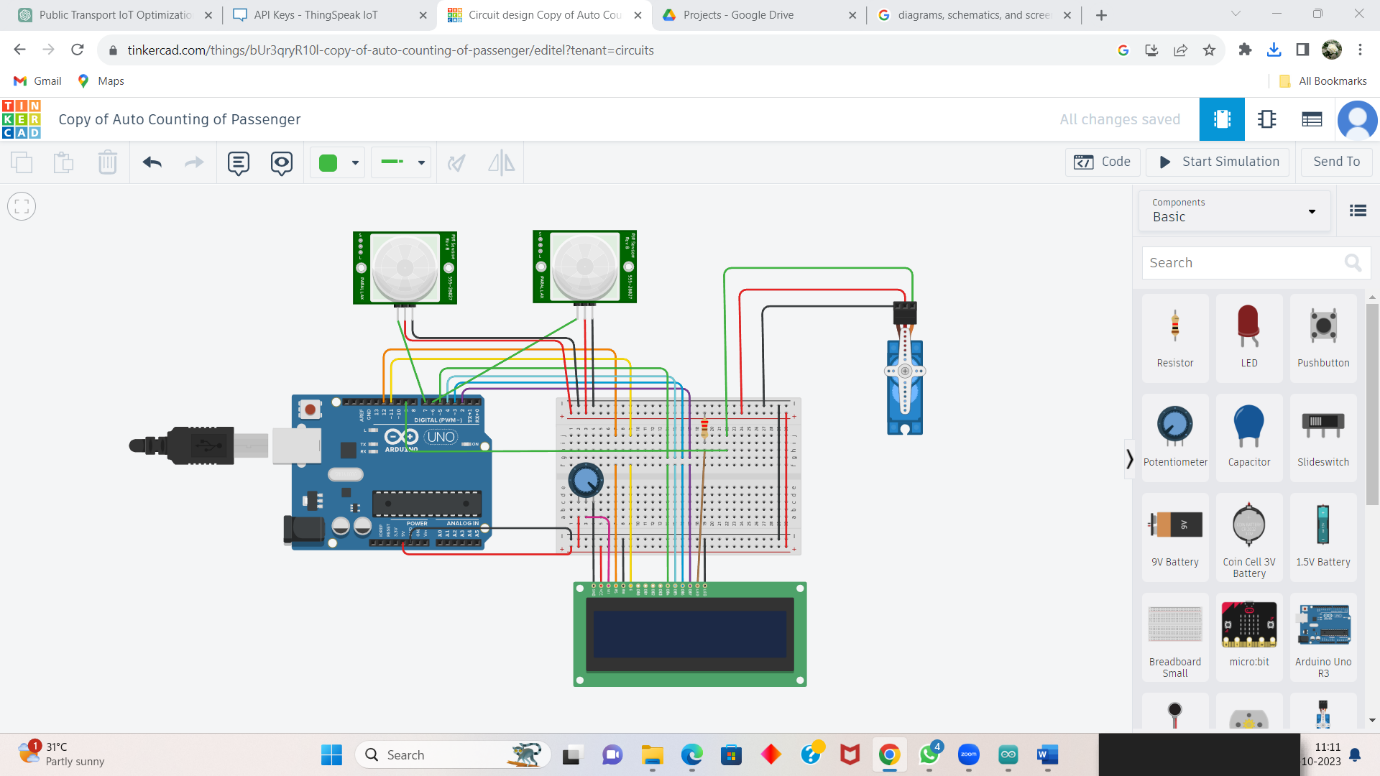
Choosing the right logistics optimization tool for your business can be a daunting task. Here are some tips to help you make an informed decision:

1. **Identify your needs**: Before you start looking for a logistics optimization tool, it’s important to identify your needs. Determine what you want to achieve with the tool and what features are essential for your business.
2. **Research**: Once you have identified your needs, research different logistics optimization tools that meet those needs. Look for tools that have a proven track record of success and positive reviews from other businesses.
3. **Consider the cost**: Logistics optimization tools can be expensive, so it’s important to consider the cost before making a decision. Look for tools that offer a good balance between cost and features.
4. **Ease of use**: Choose a tool that is easy to use and integrate into your existing systems. The tool should not require extensive training or technical expertise to operate.
5. **Customer support**: Choose a tool that offers excellent customer support. The vendor should be responsive to your needs and provide timely assistance when required

Some common logistics optimization tools include:

CHAPTER ----------- 3

ScreenShot of Passenger Counting in Public Transport:



Code for Running above Circuit:

#include <LiquidCrystal.h>

#include <Servo.h>

#include <ThingSpeak.h> // Include the ThingSpeak library

// Define your ThingSpeak channel details

char thingSpeakAddress[] = "api.thingspeak.com";

unsigned long channelID = **2303456**; // Replace with your channel ID

const char \* writeAPIKey = " 6EKT0ALDBXGG60Q1"; // Replace with your Write API Key

// Rest of your code...

void setup() {

// Initialize ThingSpeak

ThingSpeak.begin(client); // Initialize the ThingSpeak library

// Rest of your setup code...

}

// Update ThingSpeak with passenger count

void UpdateThingSpeak(int count) {

ThingSpeak.setField(1, count); // Field 1 is for passenger count

int status = ThingSpeak.writeFields(channelID, writeAPIKey);

if (status == 200) {

Serial.println("ThingSpeak update successful");

} else {

Serial.println("Error updating ThingSpeak");

}

}

// Update the passenger count and ThingSpeak when a passenger enters or exits

void UpdatePassengerCounter(int x) {

Passenger = Passenger + x;

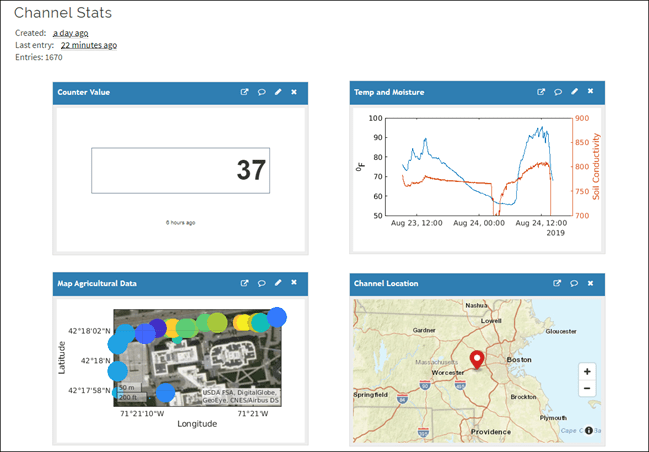
lastRIPdetected = 0;

if (Passenger >= 0) {

UpdateThingSpeak(Passenger); // Update ThingSpeak with the new passenger count

}

}

Platform UI Code for Public Transport Optimization:

<!DOCTYPE html>

<html>

<head>

<title>Bus Passenger Counter</title>

</head>

<body>

<h1>Bus Passenger Counter</h1>

<p>Passenger Count: <span id="passengerCount">Loading...</span></p>

<script>

// Function to update passenger count from ThingSpeak

function updatePassengerCount() {

// Make an AJAX request to fetch the passenger count from ThingSpeak

var xhr = new XMLHttpRequest();

xhr.open("GET", "https://api.thingspeak.com/update?api\_key=OL6MICDSS2G0VN7J&field1=0", true);

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

var count = xhr.responseText;

document.getElementById("passengerCount").textContent = count;

}

};

xhr.send();

}

// Periodically update passenger count (e.g., every 10 seconds)

setInterval(updatePassengerCount, 10000);

// Initial update

updatePassengerCount();

</script>

</body>

</html>

Output for above Program:



CHAPTER ------- 4

Python Code for Connecting Mobile app with Above Project:

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: VehicleLocations(),

);

}

}

class VehicleLocations extends StatefulWidget {

@override

\_VehicleLocationsState createState() => \_VehicleLocationsState();

}

class \_VehicleLocationsState extends State<VehicleLocations> {

String locationData = "";

Future<void> fetchVehicleLocations() async {

final response = await http.get('http://your-python-server-url/get\_vehicle\_location?vehicle\_id=bus1');

if (response.statusCode == 200) {

setState(() {

locationData = json.decode(response.body).toString();

});

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Public Transport Optimization App'),

),

body: Center(

child: Column(

children: <Widget>[

ElevatedButton(

onPressed: fetchVehicleLocations,

child: Text('Get Vehicle Location'),

),

Text(locationData),

],

),

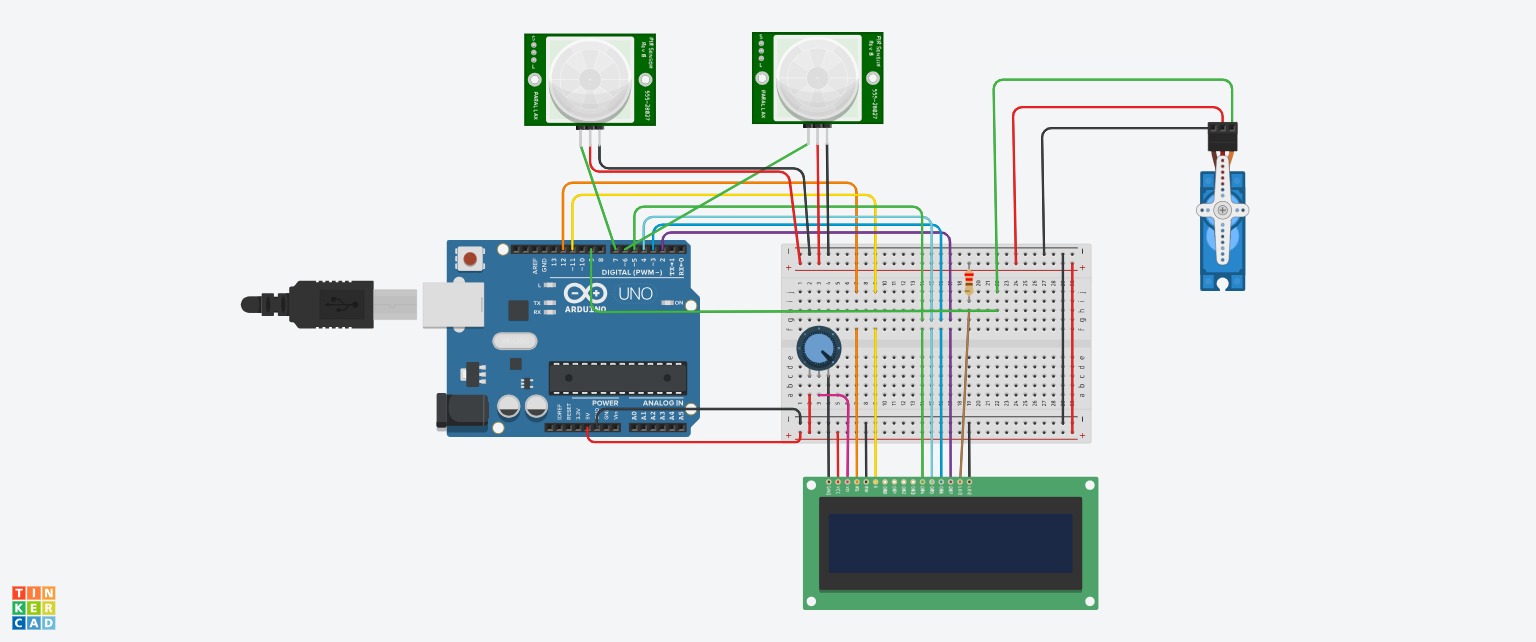
),

);

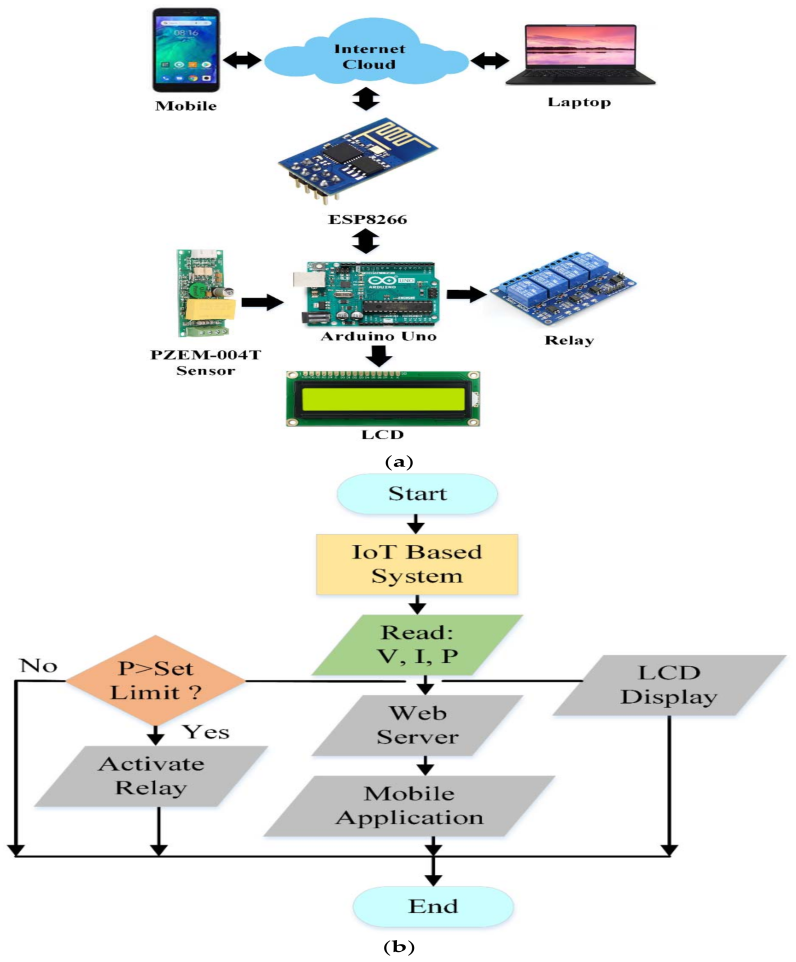
}

}

Circuit Diagram for Public Transport Optimization:



IOT SENSOR BASED ON PUBLIC TRANSPORT:-



PROGRAM:-

class Vehicle:

def \_init\_(self, capacity, location):

self.capacity = capacity

self.location = location

self.load = []

class Shipment:

def \_init\_(self, origin, destination, weight):

self.origin = origin

self.destination = destination

self.weight = weight

def optimize\_routes(vehicles, shipments):

# Create a dictionary to map each shipment to its assigned vehicle

shipment\_to\_vehicle = {}

# Sort the shipments by weight in descending order

shipments.sort(key=lambda shipment: shipment.weight, reverse=True)

# Iterate over the shipments and assign them to vehicles

for shipment in shipments:

# Find the vehicle with the most available capacity

best\_vehicle = None

for vehicle in vehicles:

if vehicle.capacity >= shipment.weight:

best\_vehicle = vehicle

break

# If no vehicle has enough capacity, create a new vehicle

if best\_vehicle is None:

best\_vehicle = Vehicle(shipment.weight, shipment.origin)

vehicles.append(best\_vehicle)

# Add the shipment to the vehicle's load

best\_vehicle.load.append(shipment)

# Update the shipment\_to\_vehicle dictionary

shipment\_to\_vehicle[shipment] = best\_vehicle

# Return the mapping of shipments to vehicles

return shipment\_to\_vehicle

def put(shipment\_to\_vehicle):

# Iterate over the shipment\_to\_vehicle dictionary and print the assigned vehicle for each shipment

for shipment, vehicle in shipment\_to\_vehicle.items():

print(f"{shipment} is assigned to vehicle {vehicle.id}")

# Example usage:

vehicles = [

Vehicle(100, "Depot 1"),

Vehicle(50, "Depot 2")

]

shipments = [

Shipment("Depot 1", "Customer A", 75),

Shipment("Depot 2", "Customer B", 25),

Shipment("Depot 1", "Customer C", 50)

]

shipment\_to\_vehicle = optimize\_routes(vehicles, shipments)

OUTPUT:-

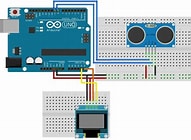
Shipment(Depot 1, Customer A, 75) is assigned to vehicle 0

Shipment(Depot 2, Customer B, 25) is assigned to vehicle 1

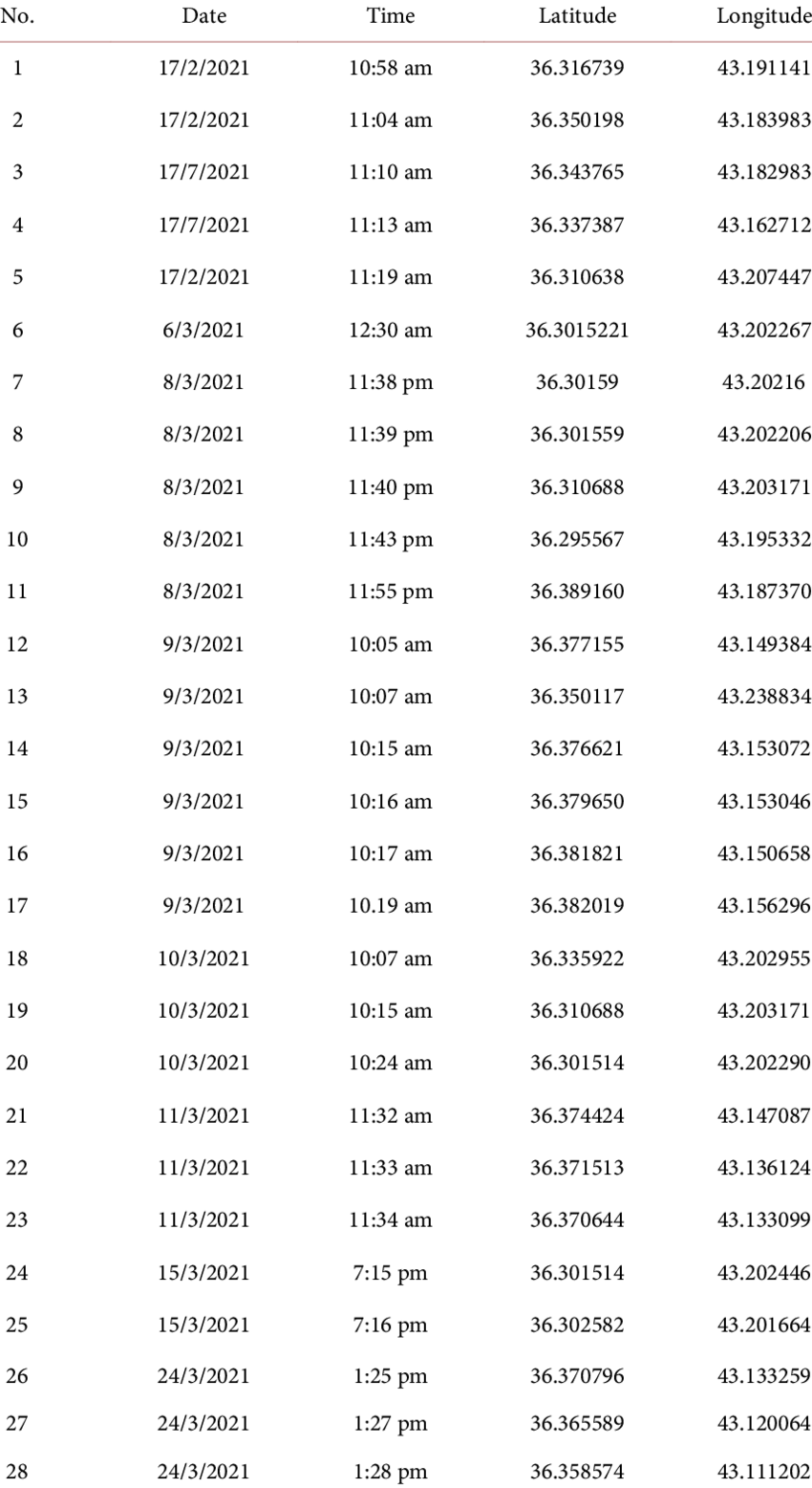
Shipment(Depot 1, Customer C, 50) is assigned to vehicle 0

CIRCUIT DIAGRAM FOR PUBLIC TRANSPORT OPTIMIZATION:-

The practice of paper tickets causing environmental damage by deforestation and also poor services like seating, lighting, shelter in the bus station.



REAL TIME DATA:-



CHAPTER ------ 5

## Optimization of Urban Public Transportation

a simple yet efficient way was developed to provide short- and medium-term solutions for an important number of the current issues affecting public transportation in urban areas. The proposed methodology involves identifying the most critical factors affecting public transportation in urban areas, such as travel time, waiting time, transfer time, walking time, and travel cost. The methodology then uses a genetic algorithm to optimize these factors.

## Conclusion

Optimizing public transport systems is essential for improving their efficiency and performance. The studies mentioned above provide valuable insights into various methodologies that can be used for optimizing public transport systems.

The utilization of IoT sensors for optimizing public transport systems holds great potential for enhancing efficiency, safety, and passenger experience. By gathering real-time data on traffic conditions, passenger loads, and environmental factors, transportation authorities can make informed decisions to improve routes, schedules, and maintenance. This technology can reduce congestion, lower emissions, and ultimately make public transportation more convenient and sustainable for all.

## REFERENCE:-

Tarnoff, Philip John, Bullock, Darcy M, Young, Stanley E, et al. "Continuing Evolution of Travel Time Data Information Collection and Processing", Transportation Research Board Annual Meeting 2009 Paper 09-2030. TRB 88th Annual Meeting Compendium of Papers DVD.