**Public Transport Optimization using Internet of Things (IoT)**

**Phase 5**

|  |  |
| --- | --- |
| **Date** | 01-11-2023 |
| **Team Id** | **Proj\_223982\_team** |
| **Project Name** | Public Transport Optimization |
| **Students with**  **Id** | Saiprasanna P  Rajasuriya L K  Naveen Kumar K  Udhayakumar P  Vignesh V |

**Introduction**

Public transport optimization is a crucial field aimed at improving efficiency, accessibility, and sustainability of urban transportation systems. It involves route planning, scheduling, technology integration, environmental sustainability, accessibility, and data analysis. Collaboration between planners, authorities, and technology companies ensures liveable, sustainable, and well-connected urban environments.

**Ideas**

* A sensor that can detect the passenger count of the vehicle.
* A micro control that can help you processing the data.
* A cloud platform will restore the data.
* To avoid the congestion while travelling.

**SENSORS USED IN SMART PARKING SYSTEM:**

* **ESP32 Microcontroller:** The ESP32 is a versatile microcontroller that provides Wi-Fi connectivity, making it suitable for IoT applications. In this project, it acts as the central processing unit.



* **UltrasonicSensors:** The integration of ultrasonic sensors, which measure the distance between objects using ultrasonic waves, allows for precise parking. However, such devices have one drawback — the sensor might get blocked with dirt.



**Code**

#include <Ultrasonic.h>

#include <WiFi.h>

#include <ThingSpeak.h>

Ultrasonic entranceSensor(2, 4);  // Trigger (D2) and Echo (D4) pins for the entrance sensor

Ultrasonic exitSensor(5, 16);     // Trigger (D5) and Echo (D16) pins for the exit sensor

int incomingPassengers = 0;

int outgoingPassengers = 0;

bool entranceDetected = false;

bool exitDetected = false;

const char\* ssid = "Wokwi-GUEST"; // Replace with your Wi-Fi SSID

const char\* password = ""; // Replace with your Wi-Fi password

const char\* server = "api.thingspeak.com";

const unsigned long channelID = 2325427; // Your ThingSpeak Channel ID

const char\* writeAPIKey = "FIAPRTHYVSVZKM2X"; // Your ThingSpeak Write API Key

WiFiClient client;

void setup() {

**Serial**.begin(115200);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

**Serial**.println("Connecting to WiFi...");

  }

**Serial**.println("Connected to WiFi");

}

void loop() {

  long entranceDistance = entranceSensor.read();

  long exitDistance = exitSensor.read();

**Serial**.print("Entrance Distance: ");

**Serial**.print(entranceDistance);

**Serial**.print(" cm, Exit Distance: ");

**Serial**.print(exitDistance);

**Serial**.println(" cm");

  if (entranceDistance < 30 && !entranceDetected) {

    incomingPassengers++;

    entranceDetected = true;

**Serial**.println("Passenger entered: " + String(incomingPassengers));

  } else if (entranceDistance >= 30) {

    entranceDetected = false;

  }

  if (exitDistance < 30 && !exitDetected) {

    outgoingPassengers++;

    exitDetected = true;

**Serial**.println("Passenger exited: " + String(outgoingPassengers));

  } else if (exitDistance >= 30) {

    exitDetected = false;

  }

  // Send passenger counts to ThingSpeak

  ThingSpeak.begin(client);

  ThingSpeak.setField(1, incomingPassengers);

  ThingSpeak.setField(2, outgoingPassengers);

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

  if (status == 200) {

**Serial**.println("Data sent to ThingSpeak successfully");

  } else {

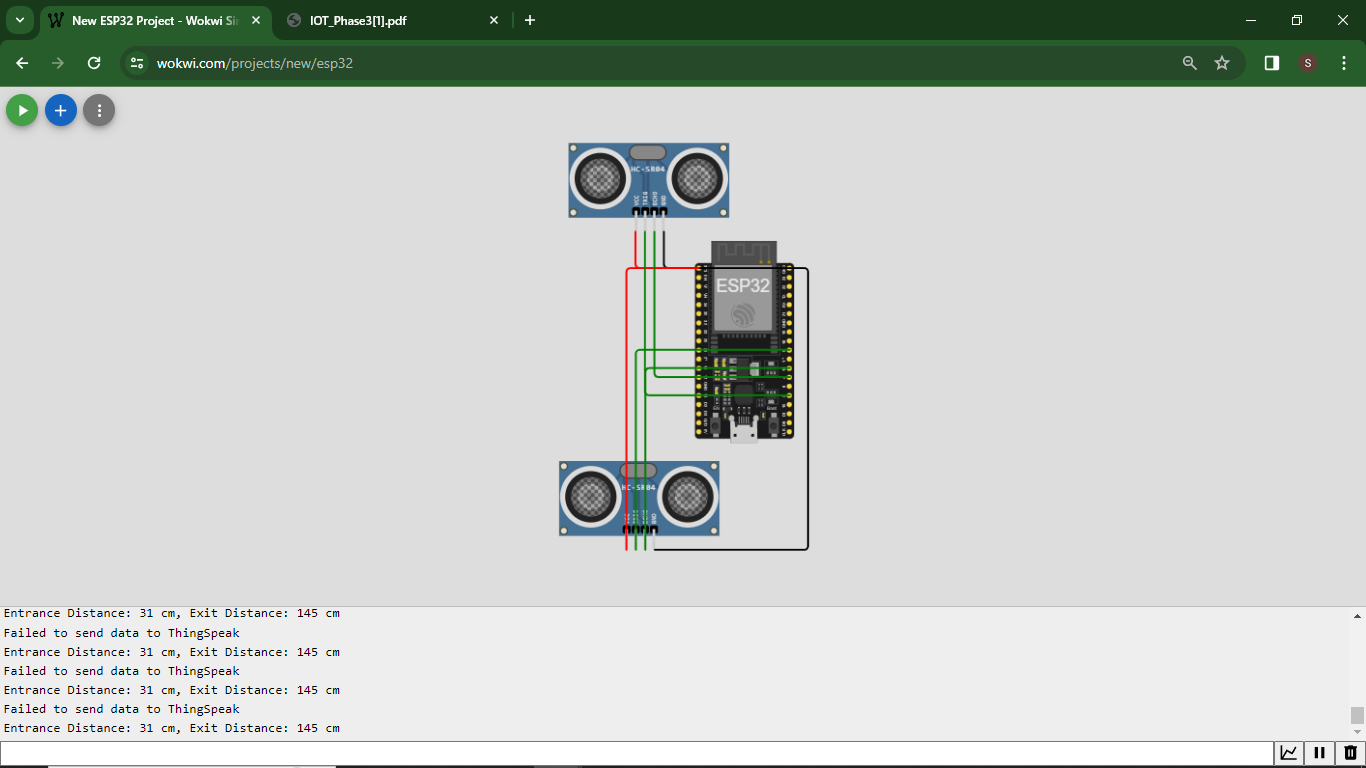
**Serial**.println("Failed to send data to ThingSpeak");

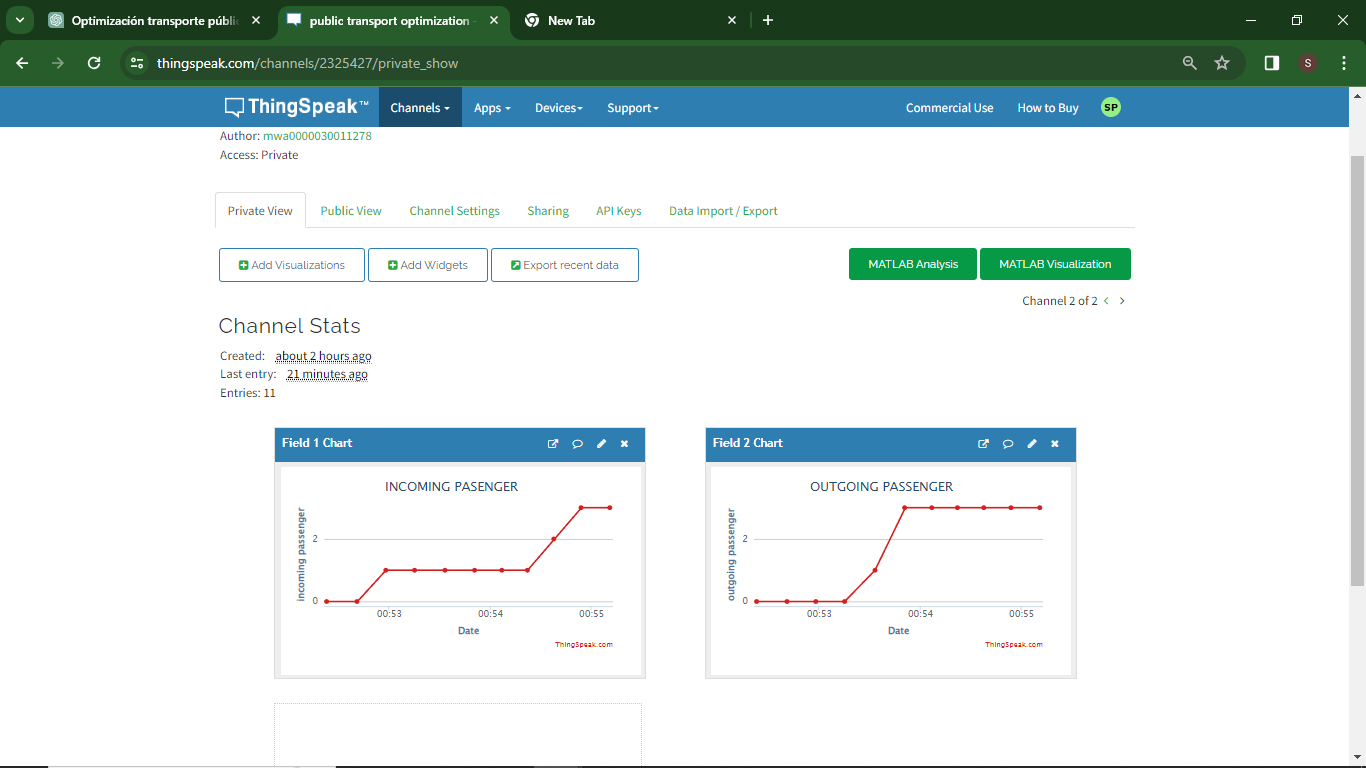
  }

  delay(1000);  // Adjust the delay as needed

}

**Simulation Circuit**



**In Think Speak**

**HTML: For structuring the webpage.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Public Transport Optimization</title>

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

</head>

<body>

<h1>Public Transport Optimization</h1>

<div class="input-container">

<label for="start">Starting Point:</label>

<select id="start">

<option value="Dharapuram">Dharapuram</option>

<option value="salem">salem</option>

<option value="coorg">coorg</option>

</select>

</div>

<div class="input-container">

<label for="end">Ending Point:</label>

<select id="end">

<option value="Dharapuram">Dharapuram</option>

<option value="salem">Salem</option>

<option value="coorg">coorg</option>

</select>

</div>

<button id="findRoute">Find Route</button>

<div id="result"></div>

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

</body>

</html>

**CSS code: For styling and layout.**

body {

font-family: Arial, sans-serif;

text-align: center;

}

.input-container {

margin: 20px;

}

select {

padding: 5px;

width: 200px;

}

button {

padding: 10px 20px;

background-color: #007BFF;

color: #fff;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#result {

margin: 20px;

}

**Java script: For dynamic content, real-time updates, and data analysis.**

**Chart.js: A JavaScript library for creating interactive charts.**

document.getElementById("findRoute").addEventListener("click", function() {

const start = document.getElementById("start").value;

const end = document.getElementById("end").value;

// You can implement your public transport optimization logic here

// This is a placeholder example

const route = findOptimalRoute(start, end);

const resultElement = document.getElementById("result");

resultElement.innerHTML = `Optimal route from ${start} to ${end}: ${route}`;

});

function findOptimalRoute(start, end) {

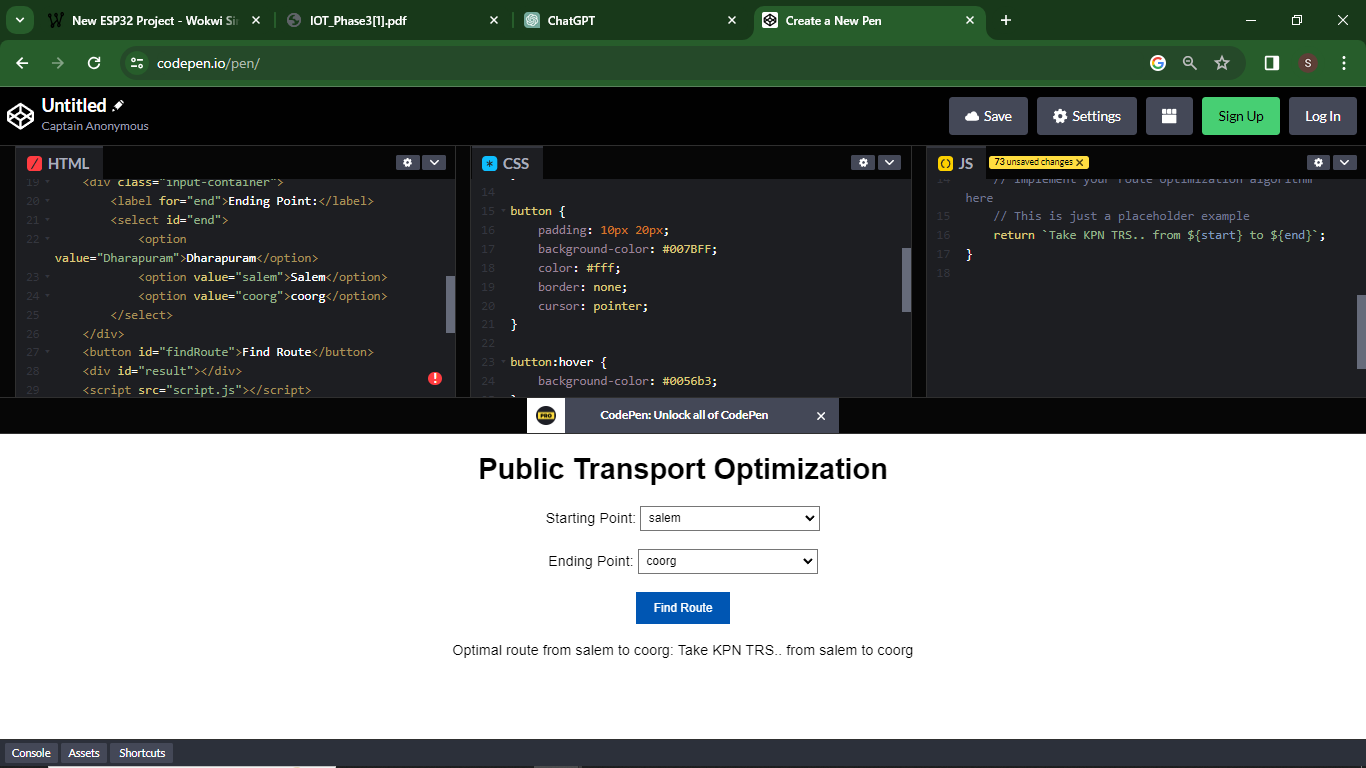
// Implement your route optimization algorithm here

// This is just a placeholder example

return `Take KPN TRS.. from ${start} to ${end}`;

}

**Output**



**GitHub Account**

Sai Prasanna P (Leader) - <https://github.com/saiprasanna011/PTO_PHASE1.git>

**Application**

* The application uses real-time route optimization algorithms, multi-modal transportation, and capacity optimization to meet demand and avoid overcrowding.
* The user interface includes a mobile app or website, map integration, alerts, notifications, and payment options.
* Collaboration with transit agencies and revenue models are considered, and the application is continuously improved based on user feedback and changing transportation conditions.

**Conclusion**

The public transport optimization system aims to create a user-friendly web application that helps users find efficient and convenient public transport routes. It should consider factors like distance, travel time, number of transfers, and cost, be accessible to all users, and handle errors. The system should be scalable, test thoroughly, collect user feedback, and be deployed to a web server for internet accessibility.