PHASE 5

IOT-PUBLIC TRANSPORT OPTIMIZATION

INTRODUCTION

IoT Public Transport Optimization involves using smart technology to enhance public transportation systems. By deploying sensors on vehicles and at transit stops, it collects real-time data, helping predict passenger demand, optimize routes, and provide passengers with accurate, personalized travel information. This innovation streamlines public transportation, making it more convenient for passengers and eco-friendly. IoT optimization reduces congestion, emissions, and enhances urban mobility. In this project, we explore how IoT technology is transforming public transport for a more efficient, user-centric, and sustainable future.

Project Overview:

The "Transit Sense" project aims to revolutionize urban public transportation by leveraging IoT technologies to optimize routes, enhance passenger experiences, and provide real-time transit information. It combines hardware and software components to create a seamless, user-centric system for tracking passenger counts, distances, and delivering precise transit information. The project consists of multiple phases, each contributing to its ultimate goal.

Empathize:

- Understand the needs, pain points, and preferences of passengers by conducting interviews, surveys, and observations at transport hubs.

- Collaborate with transit authorities, operators, and local communities to gather insights and perspectives on public transport challenges.

Define:

- Define the specific challenges within the public transport system, such as overcrowding, delays, or accessibility issues, based on the insights gathered during the empathy phase.

- Create user personas representing different passenger segments to guide solution development.

Ideate:

- Organize ideation workshops with a diverse group of stakeholders to generate creative ideas for addressing the identified problems.
- Focus on features that enhance the passenger experience, such as real-time information, accessibility features, and sustainable transportation options.

Prototype:

- Develop low-fidelity prototypes of IOT-based solutions, including mobile apps, sensor networks, or data dashboards, to visualize how they would work.
- Gather feedback from passengers and stakeholders by conducting usability testing with the prototypes to refine the design.

Test:

- Implement a small-scale pilot of the IOT solution on a specific public transport route or mode to assess its effectiveness in a real-world setting.
- Collect data on passenger satisfaction, system performance, and environmental impact during the pilot phase.

Implement:

- If the pilot is successful, scale up the IOT solution to cover a larger portion of the public transport network.
- Collaborate closely with transit authorities and operators to integrate the IOT solution into existing systems.

Evaluate and Iterate:

- Continuously collect user feedback and monitor system performance to identify areas for improvement.
- Use iterative development cycles to refine and enhance the IOT-based solution based on ongoing feedback and changing needs.

IOT SENSOR DEPLOYMENT PHASE:

This is the step-by-step process of incorporating machine learning algorithms into the "Transit Sense" project to improve the accuracy of arrival time predictions. This enhancement will provide passengers with more reliable and timely information, ultimately improving the overall transit experience.

1: Data Collection and Preparation

Gather Historical Data:

 Detailed explanation of collecting historical data on public transport routes, including departure times, actual arrival times, traffic conditions, and weather data.

Data Preprocessing:

 How to clean and preprocess the dataset, handling missing values, outliers, and data cleaning techniques.

Real-Time Data Integration:

 Developing a real-time data pipeline to provide current traffic conditions, weather updates, and vehicle locations to the machine learning model.

2: Machine Learning Model Development

Model Selection:

 Explanation of selecting an appropriate machine learning algorithm. (e.g., Random Forest, Gradient Boosting, LSTM) for arrival time prediction.

Model Training:

 How to split the dataset into training and testing sets and train the machine learning model using historical data.

Model Evaluation:

 Metrics and methods for evaluating the model's accuracy and performance, including MAE and RMSE.

Step 3: Real-Time Integration and Passenger Access

Real-Time Data Integration:

 Implementing a real-time data pipeline to feed current traffic conditions, weather updates, and vehicle locations to the machine learning model.

Integration with Passenger Information Systems:

 How to incorporate the machine learning model into passenger information systems, making predictions accessible to passengers through mobile apps and information displays.

Step 4: Feedback Loop and Continuous Improvement

Feedback Mechanism:

 Creating a feedback mechanism for passengers to report inaccuracies in arrival time predictions.

Model Update and Iteration:

 How to use passenger feedback to continuously update and refine the machine learning model, ensuring its accuracy and responsiveness.

CODE IMPLEMENTATION

This phase of the project aims to optimize public transport using Arduino and IoT technologies.

It utilizes components such as the ESP32 Development Board, Ultrasonic Sensors, LED, Resistors, and the Blynk App to create a system for object detection and counting.

```
#define BLYNK_TEMPLATE_ID "TMPL26V4fGv5q"
#define BLYNK_TEMPLATE_NAME "Test"
#define BLYNK_AUTH_TOKEN "XEHxNF_Ur1Nt2p7wB5B20dNI1ZUwj34P"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

int duration1 = 0;
int distance1 = 0;
int duration2 = 0;
int distance2 = 0;
int dis1 = 0;
int dis2 = 0;
int dis_new1 = 0;
int dis_new2 = 0;
int entered = 0;
int left = 0;
int inside = 0;
#define LED 2
#define PIN_TRIG1 15
#define PIN_ECHO1 14
#define PIN_TRIG2 13
#define PIN_ECHO2 12
BlynkTimer timer;

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";    // your network SSID (name)
char pass[] = "";
#define BLYNK_PRINT Serial
```

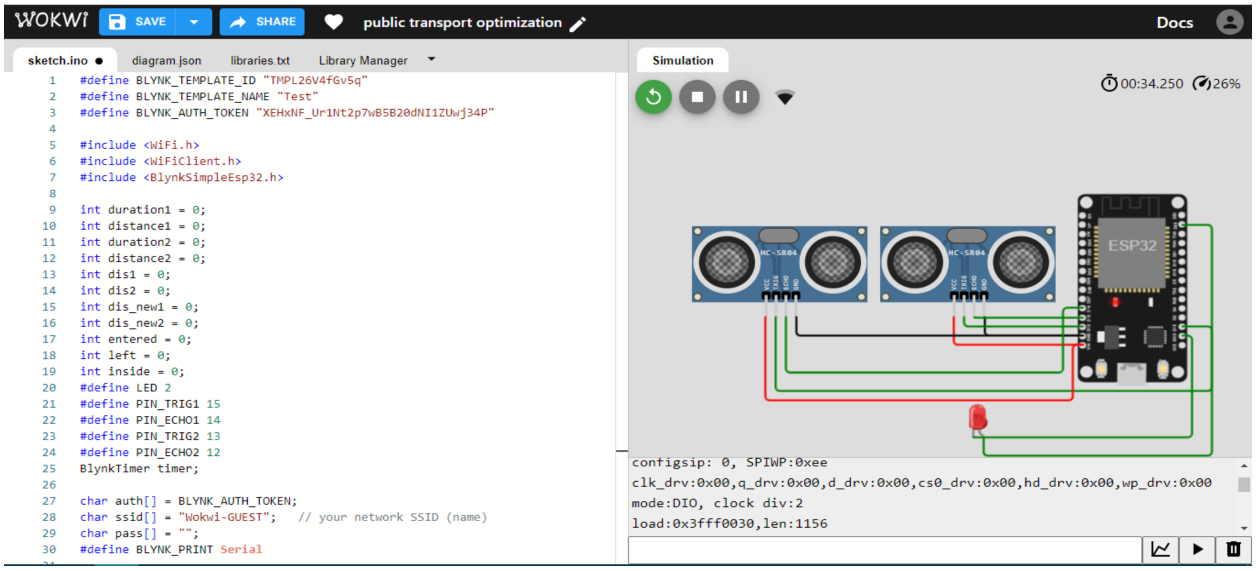
```
long get_distance1() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG1, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG1, LOW);

  // Read the result:
  duration1 = pulseIn(PIN_ECHO1, HIGH);
  distance1 = duration1 / 58;
  return distance1;
}

long get_distance2() {
  // Start a new measurement:
  digitalWrite(PIN_TRIG2, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG2, LOW);

  // Read the result:
  duration2 = pulseIn(PIN_ECHO2, HIGH);
  distance2 = duration2 / 58;
  return distance2;
}

void myTimer() {
  Serial.println("100");
  dis_new1 = get_distance1();
  dis_new2 = get_distance2();
  if (dis1 != dis_new1 || dis2 != dis_new2){
    Serial.println("200");
    if (dis1 < dis2){
      Serial.println("Enter loop");
      entered = entered + 1;
      inside = inside + 1;
      digitalWrite(LED, HIGH);
      Blynk.virtualWrite(V0, entered);
      Blynk.virtualWrite(V2, inside);
      dis1 = dis_new1;
      delay(1000);
      digitalWrite(LED, LOW);
    }
    if (dis1 > dis2){
      Serial.println("Leave loop");
      left = left + 1;
      inside = inside - 1;
      Blynk.virtualWrite(V1, left);
      Blynk.virtualWrite(V2, inside);
      dis2 = dis_new2;
      delay(1000);
    }

  }


}

 void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  pinMode(PIN_TRIG1, OUTPUT);
  pinMode(PIN_ECHO1, INPUT);
  pinMode(PIN_TRIG2, OUTPUT);
  pinMode(PIN_ECHO2, INPUT);
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 8080);
  timer.setInterval(1000L, myTimer);

}

void loop() {
  Blynk.run();
  timer.run();
}
```

Components:

ESP32 Development Board: A powerful microcontroller board.

Ultrasonic Sensors (2x): Used for distance measurement.

LED: A visual indicator.



```
#define BLYNK_TEMPLATE_ID "TMPL26V4fGv5q"
#define BLYNK_TEMPLATE_NAME "Test"
#define BLYNK_AUTH_TOKEN "XEHxNF_Ur1Nt2p7wB5B20dNI1ZUwj34P"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

int duration1 = 0;
int distance1 = 0;
int duration2 = 0;
int distance2 = 0;
int dis1 = 0;
int dis2 = 0;
int dis_new1 = 0;
int dis_new2 = 0;
int entered = 0;
int left = 0;
int inside = 0;
#define LED 2
#define PIN_TRIG1 15
#define PIN_ECHO1 14
#define PIN_TRIG2 13
#define PIN_ECHO2 12
BlynkTimer timer;

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";   // your network SSID (name)
char pass[] = "";
#define BLYNK_PRINT Serial
```

```
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
```

PLATFORM DEVELOPMENT

Begin by creating a web application aimed at optimizing public transport routes. The foundation involves structuring an HTML document, defining transit information, and creating a visually appealing layout using CSS. Enhance user experience by designing an intuitive interface. A well-designed interface ensures users find the application easy to navigate. Incorporate JavaScript to add interactivity. Implement event listeners to capture form submissions and extract user input effectively. This step involves the real -time transit information.

Step 1: Set Up the HTML Structure

Create an HTML file and set up the basic structure including the necessary HTML elements like **`<html>`**, **`<head>`**, and **`<body>`**. Include the form elements for Sensor1Distance,Sensor 2 Distance, People Entered, People Left, People Inside , and a **`<div>`** to display the transit information.

Step 2: Add CSS Styling (styles.css)

Create a separate CSS file (styles.css) to style the HTML elements. Add styles to format the header, form, input fields for a visually appealing layout.

Step 3: Link CSS File

Inside the **`<head>`** section of your HTML file, link the styles.css file using the **`<link>`** tag to apply the defined styles to your HTML elements.

Step 4: Write JavaScript Code (script.js)

Create a JavaScript file (script.js) or add the script directly inside **`<script>`** tags within the **`<body>`** section of your HTML file. In the script, add event listeners to handle form submission and create the **`Real-Time Transit Information`** function to process the input values and generate the transit information.

Step 5: Display Real Time Transit Information

This code will display the real time transit information using IOT sensors. You can use `text Content` or `inner HTML` to update the content of the element with the calculated route.

Program

HTML

```
<!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>
  <header>
     <h1>Public Transport Optimization</h1>
  </header>
    <section class="main-content">
    <h2> Real-Time Transit Information</h2>
   <form id="sensorForm">
  <label for="sensor1">Sensor 1 Distance:</label>
  <span id="sensor1Distance">----</span> cm<br>

  <label for="sensor2">Sensor 2 Distance:</label>
  <span id="sensor2Distance">----</span> cm<br>

  <label for="enterCount">People Entered:</label>
  <span id="enterCount">----</span><br>

  <label for="leaveCount">People Left:</label>
  <span id="leaveCount">----</span><br>

  <label for="insideCount">People Inside:</label>
  <span id="insideCount">---</span><br>
</form>
</section>
```

```
</body>

</html>
```

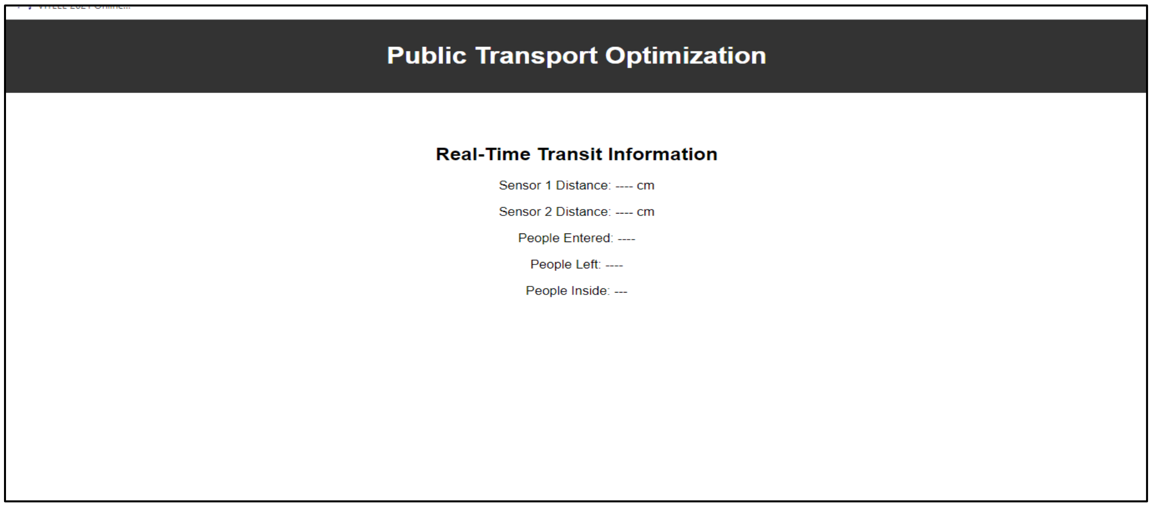
CSS

```
<style>
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
}

header {
   background-color: #333;
   color: #fff;
   padding: 10px 0;
   text-align: center;
}

.main-content {
   text-align: center;
   padding: 50px;
}

form {
   margin-bottom: 20px;
}



#Real-Time Transit Information {
   font-size: 18px;
   font-weight: bold;
}
</style>
```

JavaScript

```
<script>
```

```
document.getElementById("sensorForm").addEventListener("submit",
function(event) {
   event.preventDefault(); // Prevent the form from submitting and reloading the
page

   // Make a GET request to Arduino to fetch sensor data
   fetch("/sensorData")
      .then(response => response.json())
      .then(data => {
         // Update the UI with sensor data
         document.getElementById("sensor1Distance").textContent = data.sensor1
+ " cm";
         document.getElementById("sensor2Distance").textContent = data.sensor2
+ " cm";
         document.getElementById("enterCount").textContent = data.enterCount;
         document.getElementById("leaveCount").textContent = data.leaveCount;
         document.getElementById("insideCount").textContent = data.insideCount;
      })
      .catch(error => {
         console.error("Error:", error);
      });
});
</script>
```


Public Transport Optimization

Real-Time Transit Information

Sensor 1 Distance: ---- cm

Sensor 2 Distance: ---- cm

People Entered: ----

People Left: ----

People Inside: ---

REAL-TIME TRANSIT INFORMATION IMPROVEMENT

A real-time transit information system can significantly improve public transportation services and enhance the passenger experience in several ways:

1. Accurate Arrival and Departure Information:

Passengers can receive real-time updates about the arrival and departure times of buses, trains, or other public transportation vehicles. This accuracy reduces waiting times and allows passengers to plan their journeys more efficiently.

2. Dynamic Routing and Traffic Updates:

- Real-time systems can adjust routes based on current traffic conditions. This dynamic routing helps vehicles avoid traffic jams and take alternative routes, ensuring timely arrivals and minimizing delays.

3.Improved Passenger Safety:

- Real-time tracking allows transport authorities and passengers to know the exact location of vehicles. In case of emergencies or accidents, authorities can respond quickly. Additionally, passengers can avoid poorly lit or unsafe stops by checking the real-time location of vehicles.

4. Optimized Capacity Management:

   - Public transportation providers can use real-time data to monitor passenger loads. By knowing which routes and vehicles are crowded, they can optimize services, add more vehicles when needed, and distribute passengers evenly to enhance comfort and safety.

5. Enhanced User Experience through Apps:

   - Dedicated mobile apps can provide real-time transit information, including arrival times, route planning, and service disruptions. These apps can also offer features like mobile ticketing, allowing passengers to purchase tickets directly from their smartphones.

6. Increased Public Transportation Usage:

   - Providing real-time information can attract more passengers to public transportation by making the system more user-friendly, reliable, and efficient. Passengers are more likely to choose public transit when they can rely on accurate and timely information.

7. Better Operational Efficiency:

- Transit authorities can optimize their schedules and operations based on real-time data. This includes adjusting the frequency of services, deploying backup vehicles during peak times, and ensuring that resources are allocated where they are most needed.

8. Environmental Benefits:

- By encouraging more people to use public transportation through improved services, cities can reduce the number of private vehicles on the road. This leads to reduced traffic congestion and lower emissions, contributing to a cleaner and greener environment.

9. Community Engagement:

- Real-time transit systems can engage with the community by providing updates through social media, community websites, or public announcements. This fosters a sense of transparency and keeps passengers informed about any changes or disruptions.

A real-time transit information system can transform public transportation into a more convenient, efficient, and reliable option for commuters. It not only improves the operational aspects for transit authorities but also enhances the overall passenger experience, leading to increased usage of public transportation services.

Conclusion:

The "Transit Sense" project combines IoT sensors, machine learning, Arduino, and web applications to enhance public transportation services and passenger experiences. By collecting real-time data and delivering accurate information, it optimizes routes, reduces congestion, and provides a sustainable, user-centric approach to urban mobility.

In conclusion, public transport optimization, driven by IoT and advanced technology, represents a profound transformation in urban mobility. Through a series of project phases, we've harnessed real-time data and predictive analytics to enhance the passenger experience. Accurate arrival time predictions, eco-friendly options, and improved accessibility are now a reality. The advantages extend to cities and transit authorities with reduced congestion, emissions, and increased operational efficiency. This project showcases the immense potential of IoT and data-driven solutions, offering not just enhanced urban living but a more sustainable, connected, and environmentally friendly .