Public Transportation Optimization

Project Description

The project involves integrating IoT sensors into public transportation vehicles to monitor ridership, track locations, and predict arrival times. The goal is to provide real-time transit information to the public through a public platform, enhancing the efficiency and quality of public transportation services. This project includes defining objectives, designing the IoT sensor system, developing the real-time transit information platform, and integrating them using IoT technology and Python.

Hardware Components

- ➢ GPS Module
- ➢ GSM Module
- ➢ RFID Reader
- ➢ Camera
- ➢ Microcontroller
- ➢ Power Supply Components
- ➢ Enclosures and Mounting Hardware
- ➢ Sensors and Peripherals
- ➢ Networking Hardware
- ➢ Storage Media
- ➢ Antennas
- ➢ Cabling and Wiring

Setup of Hardware

The hardware connections are first configured in order to build this project. Later on, the software component will be covered.

Step 1: Design IoT Sensor System

Identify and procure the necessary components and sensors:

- ➢ GPS Module: Select a reliable GPS module to track real-time bus locations.

- GSM Module: Choose a GSM module for data transmission and SMS alerts.
- RFID Reader: Install RFID readers on buses for student identification.
- Camera: Select cameras capable of capturing photos and videos.
- Microcontroller: Decide on a microcontroller platform (e.g., Arduino, Raspberry Pi) for data collection and processing.

Step 2: Develop Real-Time Transit Information Platform

Choose the technology stack:

- Use IoT technology for data connectivity and communication.
- Develop the platform using Python for data processing and storage.
- Design the platform architecture for real-time data processing and storage.

Step 3: Data Collection

- Install GPS modules on buses to continuously collect GPS coordinates and timestamps.
- Deploy RFID readers on buses to scan RFID cards during boarding and disembarking.
- Attach cameras inside buses to capture time stamped photos periodically.

Step 4: Data Processing

- Utilize the selected microcontroller to process data from sensors.
- Implement algorithms to:
- Check for route compliance by comparing GPS data to predefined routes.
- Verify public disembarkation by cross-referencing RFID data.
- Securely store processed data in local or cloud storage.

Step 5: Alerts and Communication

Program the system to:

- ➢ Trigger SMS alerts for deviations, unauthorized stops, RFID registration issues, and unusual events.
- ➢
- ➢ Integrate with Telegram for real-time monitoring by transportation authorities.

Step 6: Data Storage

- ➢ Set up secure data storage solutions:
- ➢ Choose between local storage on buses and cloud storage.
- ➢ Ensure data encryption to protect public information during transmission and storage.

Step 7: User Interface

- ➢ Develop a user-friendly web-based dashboard for real-time bus tracking, accessible by the public.

Step 8: Power Supply and Security

- ➢ Ensure reliable power sources for the IoT components, considering bus batteries and external power supplies.
- ➢ Implement strong encryption protocols to secure student data during both transmission and storage.

Step 9: Compliance and Maintenance

- ➢ Comply with regulations and best practices regarding the handling of public data to protect privacy.
- ➢ Establish a regular maintenance schedule for the entire system, including sensor and software maintenance.
- ➢ Provide effective training to relevant personnel for system management and maintenance.

Software components

- ➢ Python 3.7 IDLE
- ➢ Database Management System(SQL)
- ➢ Web Development Tools(Django,Flask)

- SMS Gateway Services or APIs
- Telegram Bot API

Software Development

Step 1: Software Architecture and Design

- Create a high-level architecture for the software, identifying key components, modules, and their interactions.
- Design the database schema to store collected data securely.
- Develop a software design document detailing how each component will be implemented.

Step 2: Select Development Tools and Frameworks

- Choose the development tools, frameworks, and libraries based on the project requirements. For example:
- Use Python for backend development.
- Select a web framework like Django or Flask for the web-based dashboard.
- Choose appropriate libraries for IoT communication, data storage, and geofencing.

Step 3: Real-Time Transit Information Platform

- Develop the real-time transit information platform, which processes and integrates data from GPS, RFID, and cameras.
- Implement algorithms for real-time data synchronisation and validation.
- Ensure that the platform can predict arrival times based on GPS data.

Step 4: User Interface Development

- ➢ Create the web-based dashboard for real-time bus tracking
- ➢ Design the user interface to be intuitive and user-friendly.
- ➢ Implement features such as route visualization, live tracking, and alerts.

Step 6: Data Storage and Security

- ➢ Develop data storage components to securely store collected data, either locally or in the cloud.
- ➢ Implement encryption protocols to protect student data during transmission and storage.
- ➢ Ensure data access controls and user authentication for security.

Step 7: Communication and Alerts

- ➢ Integrate communication protocols (e.g., SMS and Telegram) for sending alerts to parents and school authorities.
- ➢ Set up automated alerts for deviations, unauthorized stops, and unusual bus events.
- ➢ Establish real-time monitoring through the Telegram integration.

Demonstration

In this project, IoT sensors integrated into buses collect real-time data, including GPS coordinates, RFID data, and photos. A web-based dashboard and mobile app provide parents with real-time tracking and administrators with fleet management tools. The system triggers SMS alerts and integrates with Telegram for communication. Data is securely stored, and encryption is applied. Compliance with regulations and scheduled maintenance ensure a reliable and efficient public transportation system.