ABSTRACT:

The objective of an SeatSense Smart Bus Seat Monitoring Systems is to accurately and efficiently track the number of passengers boarding and alighting public transportation vehicles such as buses. The primary goals of implementing such a system typically include:

- **Accurate Passenger Counting:** Providing precise data on the number of passengers using public transportation services at various times and locations.
- **Data for Planning and optimization:** Generating data that can be used by transportation authorities and operators to optimize routes, schedules, and resources based on passenger demand patterns.
- **Resource Allocation:** Helping transportation agencies allocate resources effectively by understanding passenger flow and demand on different routes and at different times of the day.
- **Performance Monitoring:** Enabling operators to monitor and assess the performance of their services, including factors like occupancy rates, service reliability, and adherence to schedules.
- **Service Improvement:** Identifying areas where service improvements are needed based on passenger volume data, such as adding more vehicles during peak times or adjusting routes to better serve high-demand areas.

The aim of SeatSense Smart Bus Seat Monitoring Systems is to provide a reliable and accurate method for counting passengers boarding and alighting buses without the need for manual intervention. The specific aims of implementing an SeatSense Smart Bus Seat Monitoring Systems include:

- **Accuracy:** Ensuring that the system can accurately count the number of passengers entering and exiting the vehicle, minimizing errors and discrepancies.
- **Efficiency:** Automating the passenger counting process to reduce the need for manual counting by drivers or personnel, thereby saving time and resources.
- **Real-Time Data:** Providing real-time data on passenger occupancy and demand, enabling transportation authorities and operators to make informed decisions promptly.
- **Cost-Effectiveness:** Offering a cost-effective solution for gathering passenger count data compared to manual counting methods, which may be labor-intensive and prone to errors.
- **Integration:** Integrating seamlessly with other transportation management systems, such as fare collection, scheduling, and dispatching systems, to streamline operations and improve overall efficiency.

- **Improved Service Quality:** Contributing to the enhancement of service quality by providing insights into passenger demand patterns and enabling operators to optimize services accordingly.

The outreach of an SeatSense Smart Bus Seat Monitoring Systems can be broadened through several means:

- **Scalability:** The SeatSense Smart Bus Seat Monitoring Systems should be scalable to accommodate transportation networks of various sizes, ranging from small local bus routes to large regional or national transit systems.
- **Geographical Coverage:** The SeatSense Smart Bus Seat Monitoring Systems should be deployable in urban, suburban, and rural areas, catering to the needs of passengers across different geographical locations.
- **Accessibility:** The SeatSense Smart Bus Seat Monitoring Systems should be accessible to transportation agencies and operators of all sizes, including public transit authorities, private transportation companies, and community-based transit services. This ensures that the benefits of SeatSense Smart Bus Seat Monitoring Systems are available to a wide range of stakeholders.

INTRODUCTION:

In today's dynamic transportation landscape, the efficient management of public transportation services is critical for meeting the evolving needs of passengers and optimizing operational resources. SeatSense Smart Bus Seat Monitoring Systems have emerged as indispensable tools for transit agencies and operators seeking to enhance the effectiveness, reliability, and sustainability of their services.

SeatSense Smart Bus Seat Monitoring Systems utilize advanced sensor technologies and data processing algorithms to accurately track the number of passengers boarding and alighting public transportation vehicles, such as buses, trains, trams, and ferries. By automating the passenger counting process, SeatSense Smart Bus Seat Monitoring Systems offer numerous benefits to transportation stakeholders, including improved operational efficiency, enhanced service planning, optimized resource allocation, and better fare revenue management.

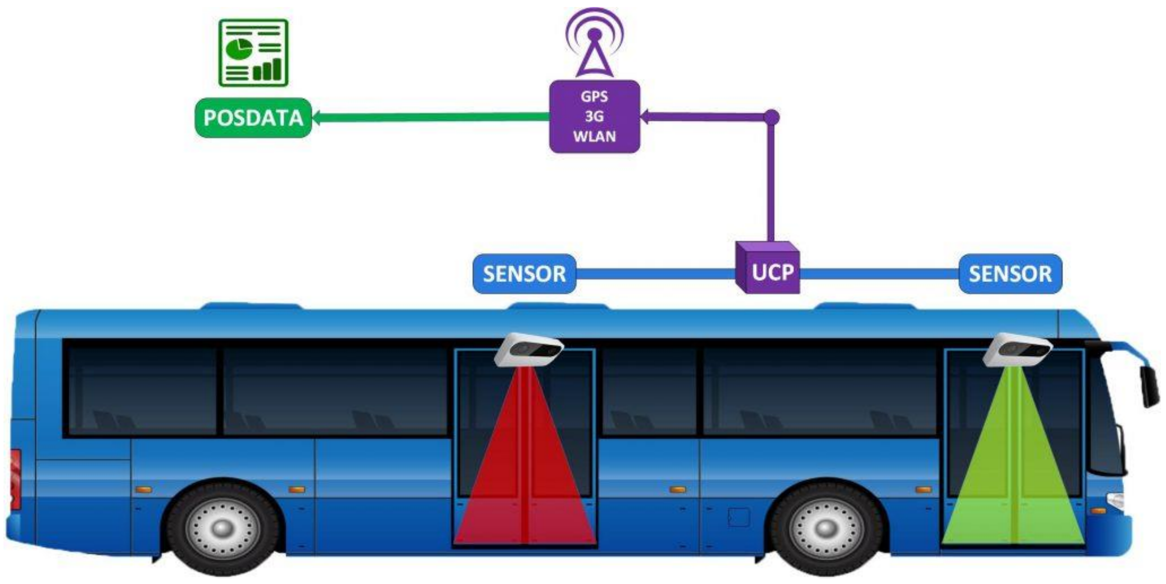
The deployment of SeatSense Smart Bus Seat Monitoring Systems represents a paradigm shift in the way public transportation services are managed and optimized. Traditionally, manual passenger counting methods, such as visual counts or ticket validations, were prone to inaccuracies, time-consuming, and often relied on subjective judgments. In contrast, SeatSense Smart Bus Seat Monitoring Systems provide precise and real-time passenger count data, enabling transit agencies to make informed decisions based on empirical evidence rather than estimates.

One of the key advantages of SeatSense Smart Bus Seat Monitoring Systems is their ability to capture comprehensive data on passenger demand patterns, allowing transit operators to optimize service frequencies, deploy vehicles strategically, and tailor services to meet fluctuating demand levels. By analyzing passenger count data, transit agencies can identify high-demand routes and time periods, adjust schedules accordingly, and improve overall service reliability and efficiency.

Furthermore, SeatSense Smart Bus Seat Monitoring Systems facilitate more effective fare revenue management by correlating passenger counts with fare transactions, ensuring accurate fare collection and minimizing revenue leakage. This not only enhances financial accountability but also supports sustainable funding for public transportation services.

In addition to operational benefits, SeatSense Smart Bus Seat Monitoring Systems contribute to a better passenger experience by providing real-time information on vehicle occupancy levels, estimated wait times, and service disruptions. This improves passenger satisfaction and encourages increased ridership, ultimately contributing to the overall sustainability of public transportation systems.

As public transportation networks continue to evolve and adapt to changing mobility trends and urbanization, the role of SeatSense Smart Bus Seat Monitoring Systems in optimizing service delivery and enhancing passenger experience will become increasingly vital. By leveraging advanced technologies and data-driven insights, SeatSense Smart Bus Seat Monitoring Systems empower transit agencies and operators to meet the evolving needs of passengers while ensuring the efficiency, reliability, and sustainability of public transportation services.



VISION AND SIGHT:

The vision of the SeatSense project is to develop a smart bus seat monitoring system that provides real-time information about vacant seats in a bus. This project aims to integrate software solutions with hardware components to create an innovative, data-driven approach to managing passenger flows and optimizing transportation services. Below are the main features of this project and their impact:

1. Enhanced Efficiency:

- **Real-Time Data Collection and Analysis:** Sensors placed within the bus collect data on seat occupancy, which is then transmitted in real-time to a central server. This enables continuous monitoring of passenger flows and allows operators to adjust services based on current demand.
- **Streamlined Operations:** By leveraging real-time data, transportation authorities can make immediate decisions to optimize routes and schedules, reducing wait times for passengers and improving punctuality.
- **Service Reliability:** The system enables transportation operators to better plan and allocate resources based on live data, leading to more reliable services and improved passenger satisfaction.

2. Accurate Insights:

- **Precise Passenger Counts:** Accurate counting of passengers provides valuable insights into ridership patterns, including peak times, preferred routes, and travel frequencies.
- **Demand Forecasting:** Historical data combined with current trends allows transportation planners to forecast demand more accurately and adjust services accordingly.
- **Data-Driven Decision-Making:** Operators can make informed decisions for route optimization, service planning, and resource allocation based on concrete data rather than relying on assumptions.

3. Cost Savings:

- **Reduction in Manual Labor Costs:** Automating the passenger counting process reduces the need for manual counts, saving time and labor costs.
- **Improved Capacity Utilization:** Real-time monitoring allows operators to adjust bus services to meet demand more effectively, leading to better capacity utilization.
- **Fare Optimization:** Data insights can inform fare strategies, such as dynamic pricing based on demand, maximizing revenue potential.

4. **<u>Enhanced Passenger Experience</u>:**

- **<u>Reduced Overcrowding</u>:** Real-time monitoring helps identify and alleviate overcrowding on buses, leading to a more comfortable ride for passengers.
- **<u>Improved Onboard Comfort</u>:** By adjusting services based on data, transportation authorities can ensure that buses are not overloaded and that seating arrangements are optimized.
- **<u>Tailored Services</u>:** Detailed passenger data allows operators to customize services to meet passenger demographics and preferences, improving overall satisfaction.

5. **<u>Integrated Solutions</u>:**

- **<u>Seamless Integration</u>:** The SeatSense system can integrate with other transportation management systems to provide a comprehensive solution for urban mobility.
- **<u>Interoperability with Smart City Initiatives</u>:** By linking with smart city infrastructure, such as traffic management and parking systems, the project contributes to holistic urban mobility solutions.
- **<u>Collaboration with Other Services</u>:** Seamless integration with other public transportation options like trains, trams, and ferries can create a unified network, offering passengers a more cohesive travel experience.

NOVELTY:

The innovative aspect of the SeatSense Smart Bus Seat Monitoring System lies in its sophisticated integration of hardware components and advanced software functionalities. The utilization of IR (Infrared) sensors in combination with an Arduino Uno microcontroller and a 16*2 LCD display ensures precise and reliable seat occupancy detection. This amalgamation of technologies forms the basis for the real-time monitoring system.

The inclusion of a Bluetooth module facilitates seamless communication between the hardware and the SeatSense App. This app, with its user-friendly interface, not only displays real-time seat availability but also employs Bluetooth integration for data transmission, enabling users to make informed decisions before boarding the bus. The LED light indicator serves as a visual cue for seat availability, enhancing the user experience further.

The Arduino code, governing the functionality of the system, is a critical element in ensuring accurate data interpretation and transmission. Simulator testing is conducted to validate the system's performance in a controlled environment, followed by real-world testing to confirm its reliability in practical scenarios.

In summary, the novelty of the SeatSense Smart Bus Seat Monitoring System lies in its intricate combination of hardware components, from IR sensors to Arduino Uno, and the sophisticated integration with the SeatSense App, providing passengers with a comprehensive and technologically advanced solution to the common issue of uncertainty in seat availability during bus commutes.

FUTURE MODIFICATIONS IN SEATSENSE PROJECT:

The SeatSense Smart Bus Seat Monitoring System opens avenues for future modifications and enhancements. One potential avenue is the incorporation of machine learning algorithms to optimize seat occupancy predictions, adapting to varying passenger behaviours and patterns. This could further refine the accuracy of real-time updates.

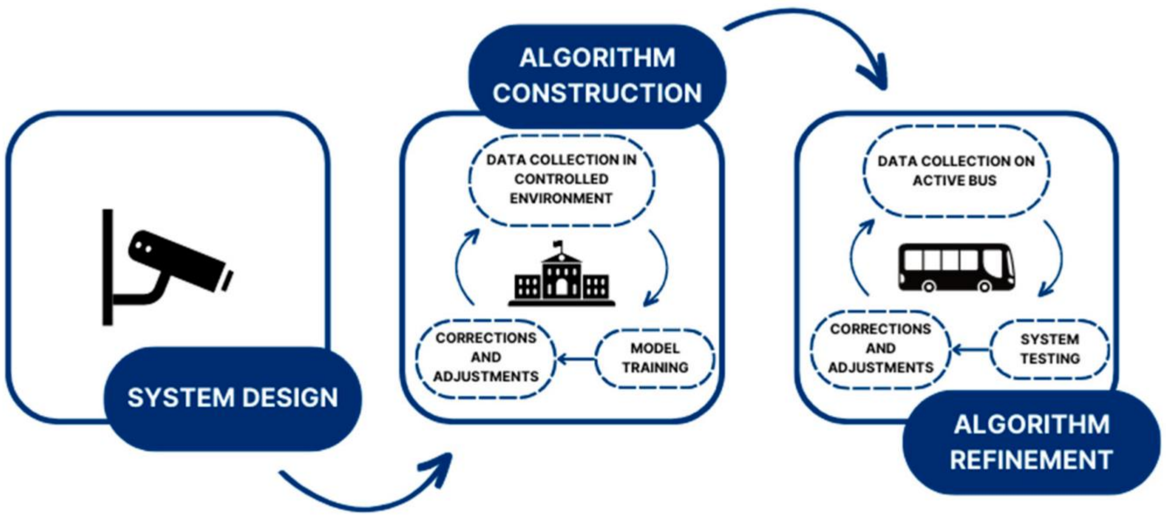
Additionally, exploring advanced sensor technologies, such as computer vision or pressure sensors, could provide even more granular data on seat occupancy. These enhancements might contribute to a more nuanced understanding of the bus's interior dynamics, improving the system's overall reliability.

Integration with smart city initiatives could be another future modification, allowing the SeatSense Smart Bus Seat Monitoring Systems to interface with broader urban transportation networks, providing passengers with comprehensive information beyond just seat availability, such as bus schedules, routes, and real-time traffic updates.

Considering the growing focus on sustainability, incorporating energy-efficient components or exploring alternative power sources could enhance the system's environmental friendliness and reduce its carbon footprint.

Lastly, continuous updates to the SeatSense App, incorporating user feedback and evolving technological trends, can ensure the system stays at the forefront of providing an optimal and user-friendly commuting experience. These future modifications would contribute to the ongoing evolution of the SeatSense Smart Bus Seat Monitoring System.

METHODOLOGY:


SYSTEM DESIGN
ALGORITHM CONSTRUCTION
DATA COLLECTION IN CONTROLLED ENVIRONMENT
MODEL TRAINING
CORRECTIONS AND ADJUSTMENTS
DATA COLLECTION ON ACTIVE BUS
SYSTEM TESTING
CORRECTIONS AND ADJUSTMENTS
ALGORITHM REFINEMENT

Methodology of SeatSense Smart Bus Seat Monitoring Systems involves several key components and processes to accurately count passengers boarding and alighting public transportation vehicles. The general methodology typically includes the following steps:

1)Sensor Development: SeatSense Smart Bus Seat Monitoring Systems use IR sensors to detect and count passengers as they board and alight the bus. A single IR sensor strategically placed at the bus entrance/exit accurately detects passenger movement, ensuring unobstructed monitoring.

2)Data Collection: The sensors collect data on passenger entries and exits in real-time as they interact with the vehicle. This data is typically captured at each door or entrance/exit point of the vehicle to ensure comprehensive coverage.

3)Data Processing: Raw data collected by the sensors is processed and analyzed to distinguish between valid passenger counts and other movements or objects that may trigger the sensors (e.g., luggage, strollers). Algorithms are often used to filter out noise and accurately count passengers.

4) Validation and Calibration: SeatSense Smart Bus Seat Monitoring Systems undergo validation and calibration processes to ensure accuracy and reliability. This may involve comparing SeatSense Smart Bus Seat Monitoring Systems counts with manual counts or other validation methods to verify the system's performance under various conditions.

5)Integration with vehicle systems: SeatSense Smart Bus Seat Monitoring Systems are often integrated with other onboard systems, such as vehicle location systems (e.g., GPS) and fare collection systems, to correlate passenger counts with vehicle locations and fare transactions.

6)Data Transmission and Storage: Processed passenger count data is stored onboard the vehicle or transmitted in real-time to a central server for further analysis and reporting. Data transmission may occur wirelessly or through physical connections, depending on the system's design.

7)Analysis and Reporting: Passenger count data is analyzed to generate insights into passenger demand patterns, service performance, and other key metrics. Reports and visualizations may be generated for transportation operators, planners, and stakeholders to inform decision-making. The real-time data of the availability of seats in a bus will be displayed in the app, website and the lcd display screen on the bus.

8)Maintenance and Monitoring: SeatSense Smart Bus Seat Monitoring Systems require regular maintenance and monitoring to ensure optimal performance. This includes routine checks, software updates, and troubleshooting to address any issues that may arise.

9)Compliance and Privacy: SeatSense Smart Bus Seat Monitoring Systems must comply with regulations and privacy guidelines governing the collection and use of passenger data. Measures are implemented to protect passenger privacy and ensure data security throughout the data lifecycle.

10)Continuous Improvement: SeatSense Smart Bus Seat Monitoring Systems undergo continuous improvement processes based on feedback from operators, passengers, and stakeholders. This may involve refining algorithms, upgrading hardware, or implementing new technologies to enhance accuracy and reliability.

PROJECT DESIGN AND IMPLEMENTATION:

1. LCD Display:

- **Real-Time Updates:** The LCD display provides real-time seat occupancy updates to passengers and staff as they enter the bus, allowing for informed decisions on seating.
- **Strategic Placement:** The display is typically placed near the entrance or above the driver's compartment, ensuring easy visibility for passengers and staff at eye level.
- **Hardware Choice:** A standard 16x2 or 20x4 character LCD display is commonly used for its reliability and ease of integration with Arduino.
- **Data Display:** The LCD shows key information such as available seats, bus capacity, and alerts (e.g., bus full or near capacity) based on the sensor data processed by the Arduino.

2. Wiring and Cable Management:

- **Simplified Wiring:** With only two sensors initially, the wiring setup is straightforward. However, the system may expand to include additional sensors for improved coverage.
- **Cable Routing:** Cables from sensors to the Arduino and Wi-Fi module are routed neatly along the bus's interior to prevent interference and potential hazards.
- **Cable Management:** Proper cable management, such as bundling and securing cables, ensures neatness and longevity of the system.
- **Interference Prevention:** Attention is paid to minimizing electromagnetic interference (EMI) through shielding and routing cables away from high-power lines.

3. Software Integration:

- **Android App:** The Android app, developed using Android Studio, provides a user-friendly interface for displaying seat availability and other relevant information to passengers.
- **Server Setup:** A server is set up using WAMP to store and process seat occupancy data. The server is managed through PuTTY, which allows secure remote access for monitoring and adjustments.

- **Data Communication:** The app communicates with the server to fetch and display real-time data, and the server interfaces with the Arduino and ESP8266 to receive and process data.

4. Role of Arduino Uno:

- **Central Processing:** Arduino Uno is responsible for processing data from sensors and controlling the overall system.
- **Central Location:** The Arduino is centrally located within the bus for easy access during maintenance and troubleshooting.
- **Sensor Data Processing:** Arduino reads data from IR sensors and interprets it to determine seat occupancy.
- **Display Control:** The Arduino controls the LCD display, updating it in real-time with seat occupancy data.
- **Integration:** Arduino integrates with ESP8266 to transmit data to the server for storage and further analysis.

5. Role of ESP8266 Wi-Fi Module:

- **Stable Connection:** The ESP8266 Wi-Fi module is placed near the Arduino for stable and reliable communication, ensuring optimal signal strength.
- **Data Transmission:** The ESP8266 transmits data from the Arduino to the server over Wi-Fi, facilitating real-time data updates for the app.
- **Integration with Sensors:** The ESP8266 integrates with IR sensors for seamless data collection and transmission to the server.
- **Network Configuration:** The module is configured with network credentials and error handling to maintain consistent connectivity.

SEATSENSE APP:

The development of an app for an SeatSense Smart Bus Seat Monitoring System involves creating a user-friendly platform that provides transportation authorities and operators with essential data and insights in real-time. The app serves as a central hub for monitoring passenger flow, analyzing data, and managing resources efficiently. Below, the key components of the app development process are outlined in detail:

1. **User Interface:**

The user interface (UI) is designed with ease of use and clarity in mind. Key aspects include:

- **Dashboard:** The main screen displays key metrics such as current passenger counts, occupancy rates, and historical trends. Visual elements such as graphs and charts make data interpretation straightforward.
- **Navigation:** The app should have intuitive navigation, with clear labels and menus for accessing different features like data analysis, route management, and settings.
- **Customization:** Allow users to customize the UI by selecting specific data views, themes, and layout preferences according to their needs.
- **Accessibility:** Ensure the app is accessible to all users, including those with disabilities, by adhering to web accessibility standards.

2. **Wi-Fi Integration:**

Wi-Fi integration plays a crucial role in enabling data communication and connectivity:

- **Network Connectivity:** The app should support reliable Wi-Fi connectivity to facilitate communication with on-board passenger counting devices.
- **Automatic Connection:** Enable the app to automatically connect to available and secured Wi-Fi networks to maintain uninterrupted data flow.
- **Wi-Fi Security:** Implement strong security protocols, such as WPA3, to protect data transmitted over the network.

3. **Data Transmission:**

Efficient data transmission ensures real-time data is received and processed quickly:

- **Data Compression:** Utilize data compression techniques to optimize data transmission, minimizing network load and latency.
- **Secure Transmission:** Implement encryption protocols (e.g., SSL/TLS) to secure data during transmission between devices and the app.
- **Error Handling:** Design the app to handle transmission errors gracefully, such as lost connections, by providing retries and fallback options.

4. **Real-Time Updates:**

Real-time updates are essential for monitoring passenger data and making quick decisions:

- **Live Monitoring:** Enable continuous, real-time monitoring of passenger counts and other metrics for each vehicle or route.
- **Data Refresh Rate:** Set appropriate data refresh rates to balance the need for timely updates with system performance and network bandwidth.

- **Alerting:** Configure real-time alerts for specific conditions, such as overcrowding or low passenger numbers, to enable prompt intervention.

5. Notifications:

Notifications keep users informed of important events or changes in passenger data:

- **Customizable Alerts:** Allow users to configure notifications based on their preferences and operational requirements.
- **Push Notifications:** Provide real-time push notifications for mobile devices, informing users of significant events like passenger surges or schedule changes.
- **Email and SMS Notifications:** Offer additional notification methods for users who prefer different channels of communication.
- **Notification Management:** Include features for managing notification settings, including enabling or disabling specific types of alerts and setting notification frequency.

In summary, the app development for an SeatSense Smart Bus Seat Monitoring System focuses on providing a seamless and efficient experience for transportation authorities and operators. Through a user-friendly interface, reliable data transmission, real-time updates, and customized notifications, the app enhances operational oversight and supports data-driven decision-making.

ARDUINO INTEGRATION WITH APP:

Integrating Arduino with the app for an SeatSense Smart Bus Seat Monitoring System involves leveraging Arduino's capabilities to collect data from sensors and transmit it wirelessly to the app. This provides real-time monitoring and data analysis for transportation authorities. Below are the key aspects of Arduino integration with the app:

1. Wi-Fi Module:

Arduino uses Wi-Fi modules to facilitate wireless data transmission:

- **Module Selection:** Commonly used modules include the ESP8266 or ESP32, which offer Wi-Fi connectivity and are compatible with Arduino boards.
- **Network Configuration:** Configure the Wi-Fi module to connect to a secured Wi-Fi network by providing the necessary credentials (SSID and password) in the Arduino code.
- **Data Transmission:** Implement code to send passenger count data from Arduino to the app's server using HTTP requests (e.g., POST requests) or other appropriate protocols.

- **Error Handling:** Include error handling to manage Wi-Fi connectivity issues, such as retries and connection verification.

2. Sensor Integration:

Sensors play a critical role in counting passengers:

- **Sensor Types:** Common sensors for passenger counting include infrared (IR) sensors, ultrasonic sensors, and time-of-flight (ToF) sensors.
- **Connection to Arduino:** Connect sensors to the Arduino board through digital or analog input/output pins as appropriate.
- **Sensor Calibration:** Calibrate sensors to ensure accurate passenger counting and minimize false positives or negatives.
- **Data Processing:** Implement code to process sensor data and translate it into passenger count information for transmission to the app.

3. Arduino Code:

Arduino code is essential for controlling the sensors and transmitting data:

- **Sensor Reading:** Write code to read data from the sensors at a specified interval and process it to determine passenger counts.
- **Data Formatting:** Format data appropriately (e.g., JSON or CSV) for transmission to the app.
- **Wi-Fi Communication:** Integrate libraries such as WiFiClient or ESP8266WiFi to facilitate data transmission over Wi-Fi.
- **Error Handling:** Include error handling and recovery mechanisms for scenarios such as sensor malfunctions or data transmission failures.

4. Power Supply:

A stable power supply is necessary for reliable operation:

- **Power Source:** Provide an appropriate power source for the Arduino board and connected sensors, such as a battery pack or onboard vehicle power.
- **Voltage Regulation:** Ensure the power supply provides the correct voltage and current for the Arduino and sensors.
- **Power Management:** Implement power-saving techniques (e.g., sleep mode) in the code to prolong battery life if using a battery-powered system.

5. Simulator Testing:

Simulator testing helps validate the system before real-world deployment:

- **Simulated Inputs:** Use simulated data or inputs to test the system's functionality and accuracy.

- **Testing Environments:** Employ software simulators to mimic the behavior of sensors and the Arduino board.
- **Debugging Tools:** Utilize debugging tools and logs to troubleshoot issues during testing.

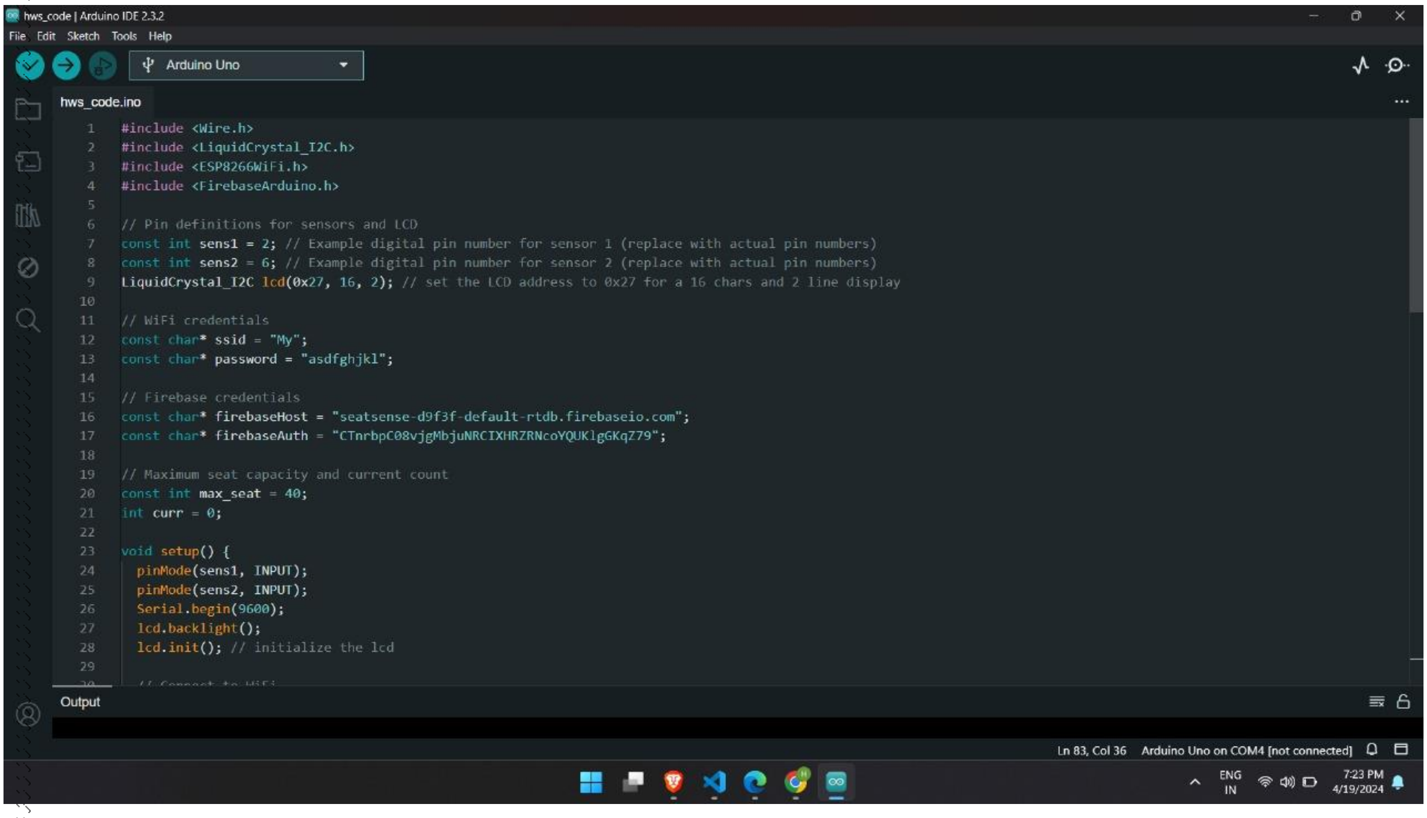
6. Real-World Testing:

Real-world testing ensures the system performs as expected in actual use:

- **Test Scenarios:** Conduct tests in different environments and conditions to evaluate the system's performance and robustness.
- **Accuracy Assessment:** Measure the system's accuracy by comparing it to manual counts or other reliable sources.
- **Feedback Collection:** Gather feedback from operators and users to identify potential improvements or issues.
- **Continuous Monitoring:** Continuously monitor the system's performance and make necessary adjustments to the code and configuration.

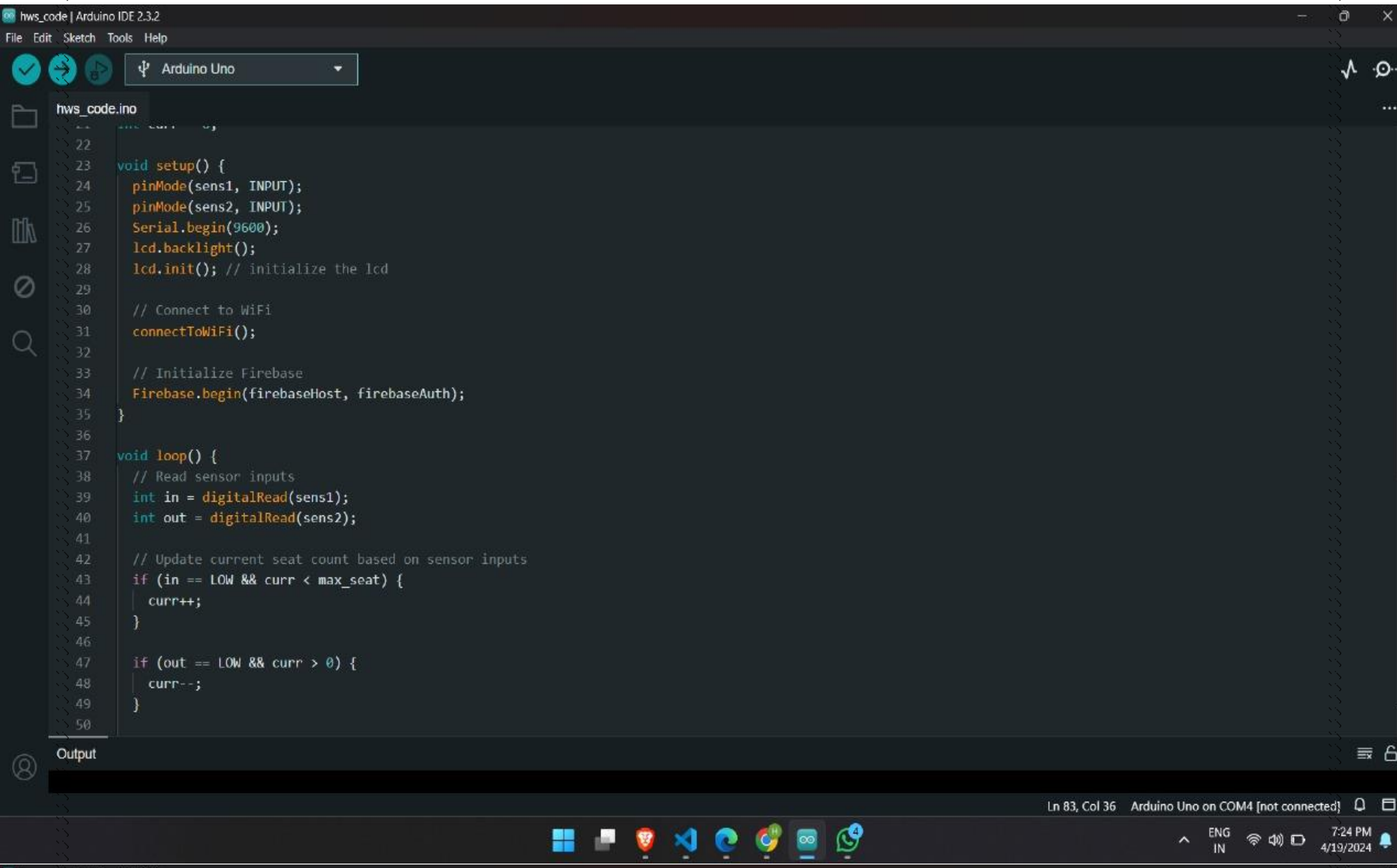
In summary, integrating Arduino with the app for an SeatSense Smart Bus Seat Monitoring System involves utilizing Wi-Fi modules for data transmission, connecting sensors for passenger counting, and writing Arduino code for processing and transmitting data. Thorough simulator and real-world testing are essential to ensure the system's reliability and accuracy.

Arduino Code:

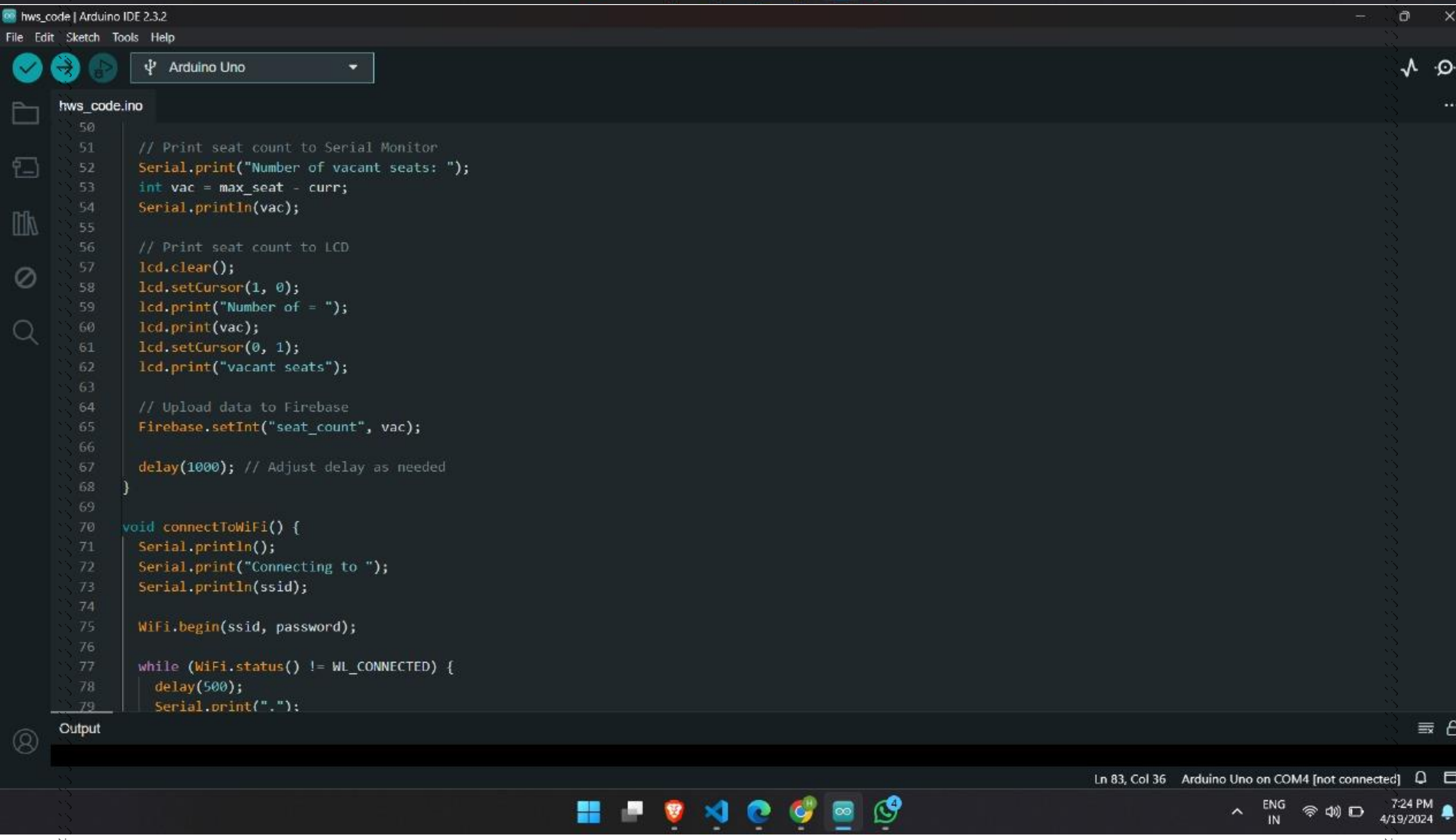


```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

// Pin definitions for sensors and LCD
const int sens1 = 2; // Example digital pin number for sensor 1 (replace with actual pin numbers)
const int sens2 = 6; // Example digital pin number for sensor 2 (replace with actual pin numbers)
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

// WiFi credentials
const char* ssid = "My";
const char* password = "asdfghjkl";

// Firebase credentials
const char* firebaseHost = "seatsense-d9f3f-default-rtdb.firebaseio.com";
const char* firebaseAuth = "CTnrbpC08vjgMbjuNRCIXHRZRNcoYQUKlgGKqZ79";

// Maximum seat capacity and current count
const int max_seat = 40;
int curr = 0;

void setup() {
  pinMode(sens1, INPUT);
  pinMode(sens2, INPUT);
  Serial.begin(9600);
  lcd.backlight();
  lcd.init(); // initialize the lcd

```

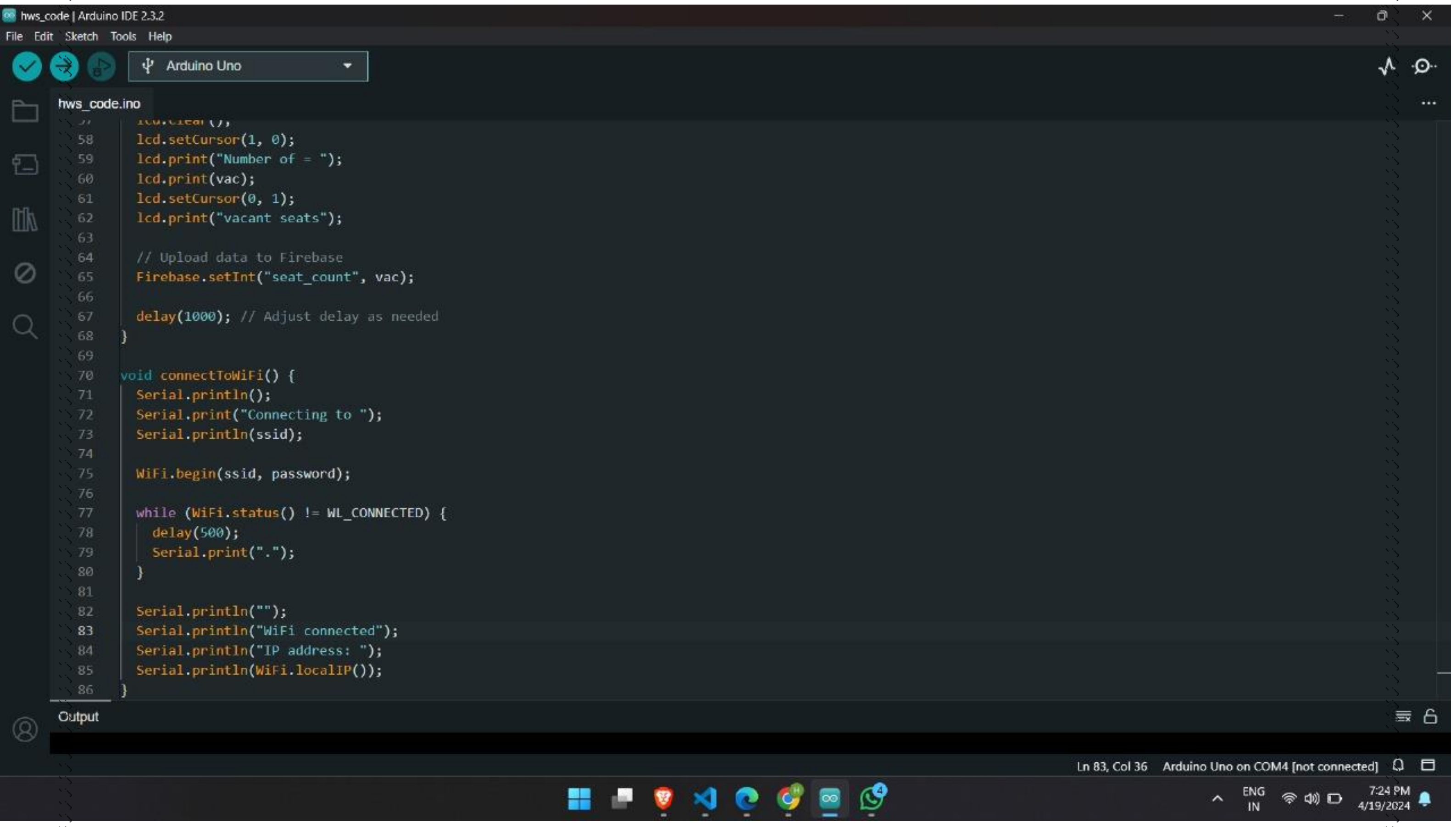
Ln 83, Col 36 Arduino Uno on COM4 [not connected]


hws_code | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
hws_code.ino
Output
Ln 83, Col 36 Arduino Uno on COM4 [not connected]
ENG IN
7:24 PM 4/19/2024

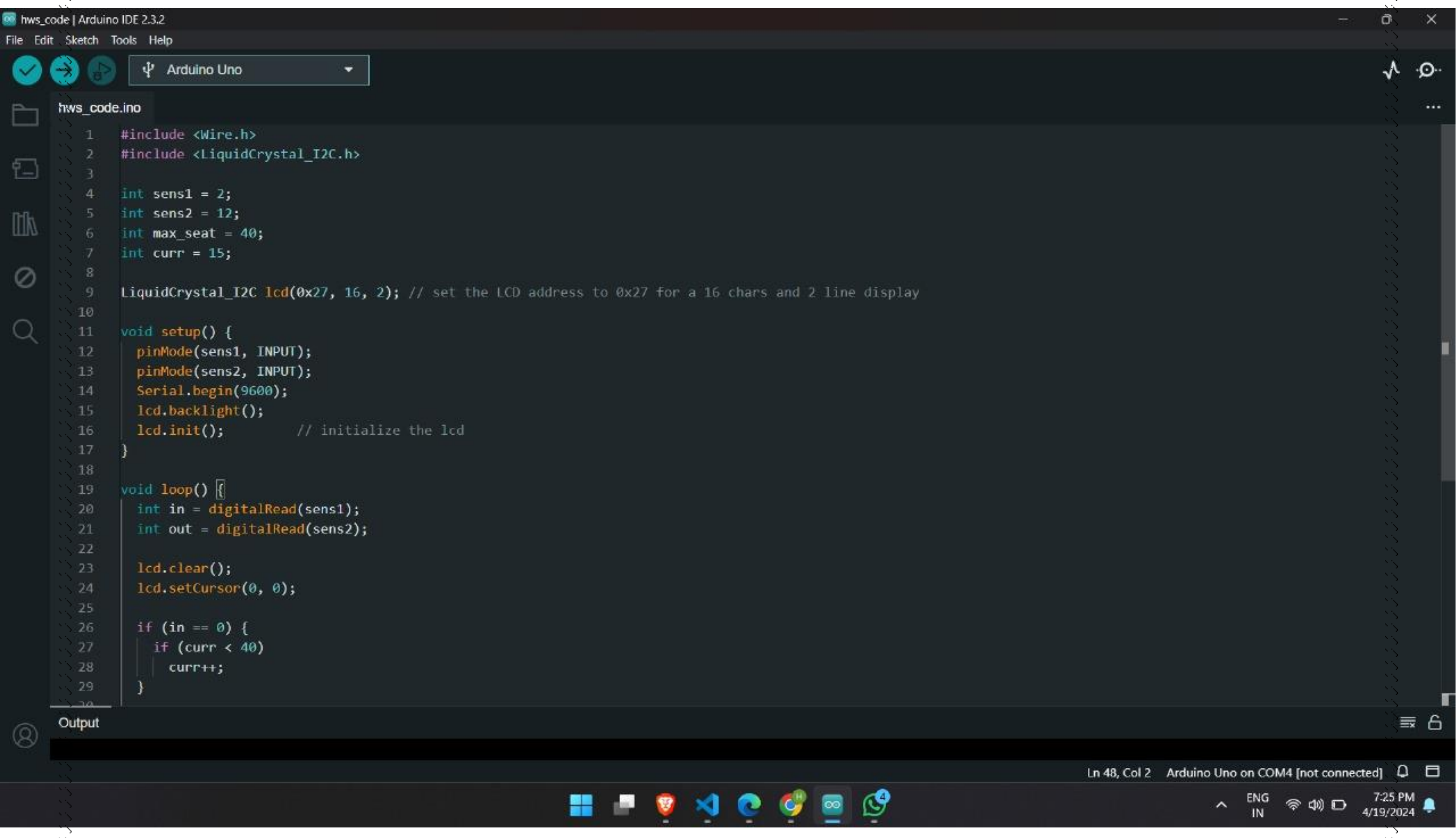
```
void setup() {
  pinMode(sens1, INPUT);
  pinMode(sens2, INPUT);
  Serial.begin(9600);
  lcd.backlight();
  lcd.init(); // initialize the lcd

  // Connect to WiFi
  connectToWiFi();

  // Initialize Firebase
  Firebase.begin(firebaseHost, firebaseAuth);
}

void loop() {
  // Read sensor inputs
  int in = digitalRead(sens1);
  int out = digitalRead(sens2);

  // Update current seat count based on sensor inputs
  if (in == LOW && curr < max_seat) {
    curr++;
  }

  if (out == LOW && curr > 0) {
    curr--;
  }

```


hws_code | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
hws_code.ino
Output
Ln 83, Col 36 Arduino Uno on COM4 [not connected]
ENG IN
7:24 PM 4/19/2024

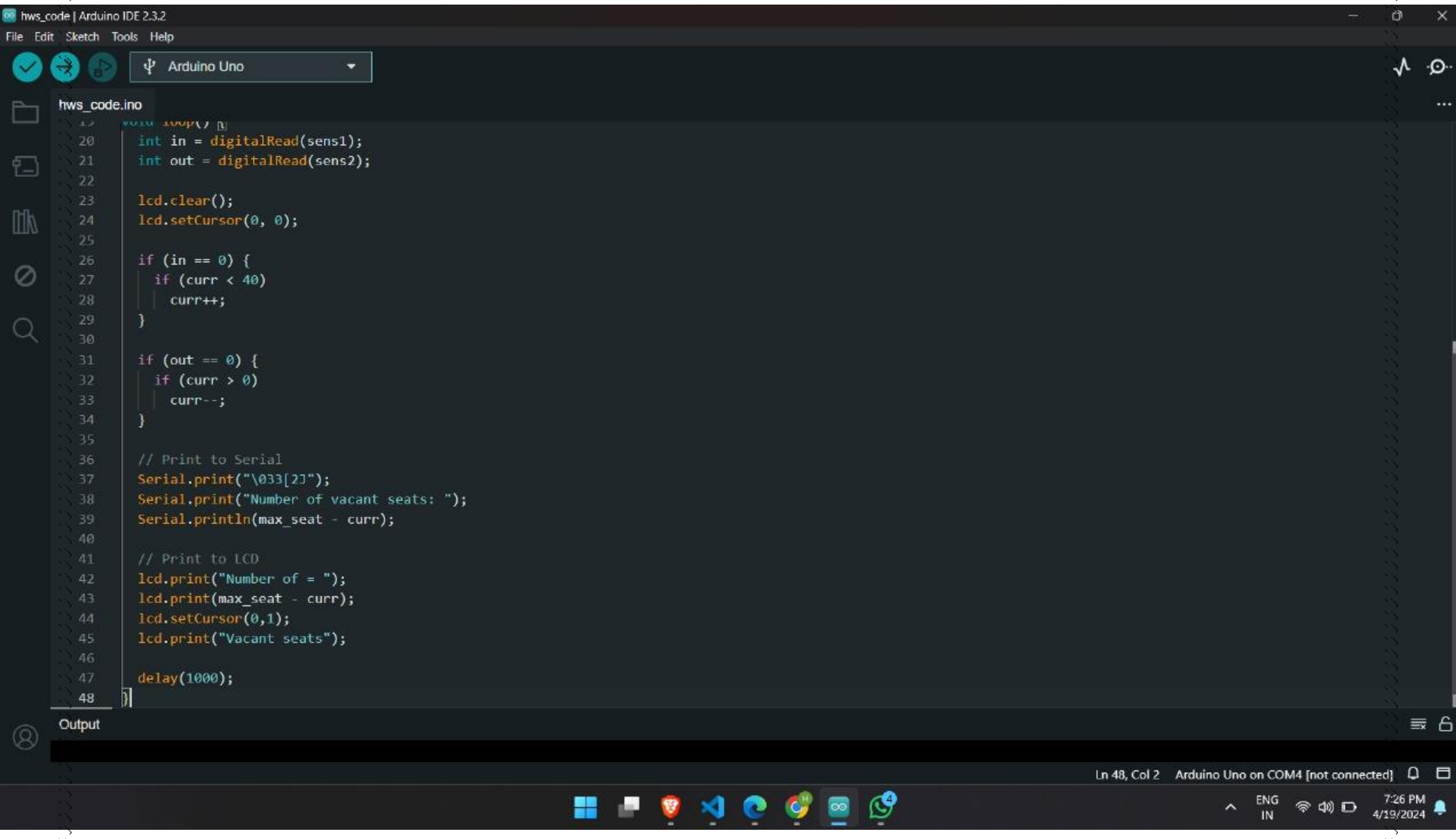
```

  // Print seat count to Serial Monitor
  Serial.print("Number of vacant seats: ");
  int vac = max_seat - curr;
  Serial.println(vac);

  // Print seat count to LCD
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("Number of = ");
  lcd.print(vac);
  lcd.setCursor(0, 1);
  lcd.print("vacant seats");

  // Upload data to Firebase
  Firebase.setInt("seat_count", vac);

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

void connectToWiFi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```


hws_code | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
hws_code.ino
Output
Ln 83, Col 36 Arduino Uno on COM4 [not connected]
ENG IN
7:24 PM 4/19/2024

```
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("Number of = ");
  lcd.print(vac);
  lcd.setCursor(0, 1);
  lcd.print("vacant seats");

  // Upload data to Firebase
  Firebase.setInt("seat_count", vac);

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

void connectToWiFi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```


hws_code | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
hws_code.ino
Output
Ln 48, Col 2 Arduino Uno on COM4 [not connected]
ENG IN
7:25 PM 4/19/2024

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

int sens1 = 2;
int sens2 = 12;
int max_seat = 40;
int curr = 15;

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup() {
  pinMode(sens1, INPUT);
  pinMode(sens2, INPUT);
  Serial.begin(9600);
  lcd.backlight();
  lcd.init();         // initialize the lcd
}

void loop() {
  int in = digitalRead(sens1);
  int out = digitalRead(sens2);

  lcd.clear();
  lcd.setCursor(0, 0);

  if (in == 0) {
    if (curr < 40)
      curr++;
  }
```



hws_code | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
hws_code.ino
Output
Ln 48, Col 2 Arduino Uno on COM4 [not connected]

```
  int in = digitalRead(sens1);
  int out = digitalRead(sens2);

  lcd.clear();
  lcd.setCursor(0, 0);

  if (in == 0) {
    if (curr < 40)
      curr++;
  }

  if (out == 0) {
    if (curr > 0)
      curr--;
  }

  // Print to Serial
  Serial.print("\033[2J");
  Serial.print("Number of vacant seats: ");
  Serial.println(max_seat - curr);

  // Print to LCD
  lcd.print("Number of = ");
  lcd.print(max_seat - curr);
  lcd.setCursor(0,1);
  lcd.print("Vacant seats");

  delay(1000);
}
```

SEATSENSE WEBSITE:

1. Server Setup:

- **WAMP Environment:** The local server is typically set up using a WAMP (Windows, Apache, MySQL, PHP) stack, which provides the necessary infrastructure for running a web server, database, and PHP scripts.
- **Server Management:** Tools such as PuTTY may be used to manage the server remotely, allowing administrators to access and control server operations securely.

2. Data Processing and Storage:

- **Data Collection:** Real-time data from the bus, including seat occupancy information collected by the Arduino and ESP8266, is transmitted to the server.
- **Database Management:** Data is stored in a MySQL database for efficient access and retrieval. This database may contain tables for storing seat occupancy data, bus routes, schedules, and other relevant information.
- **Data Processing:** PHP scripts on the server process incoming data, converting raw data from the bus into meaningful information, such as seat availability updates.

3. Website Features:

- **User Interface:** The website provides a clean, intuitive user interface for accessing real-time data and historical data. It may include dashboards, charts, and tables to visualize data.
- **Real-Time Monitoring:** The website offers real-time monitoring of seat availability and passenger flow on each bus, allowing operators to make quick decisions.
- **Historical Data Access:** Users can access historical data for analysis, trend identification, and decision-making related to route planning and resource allocation.

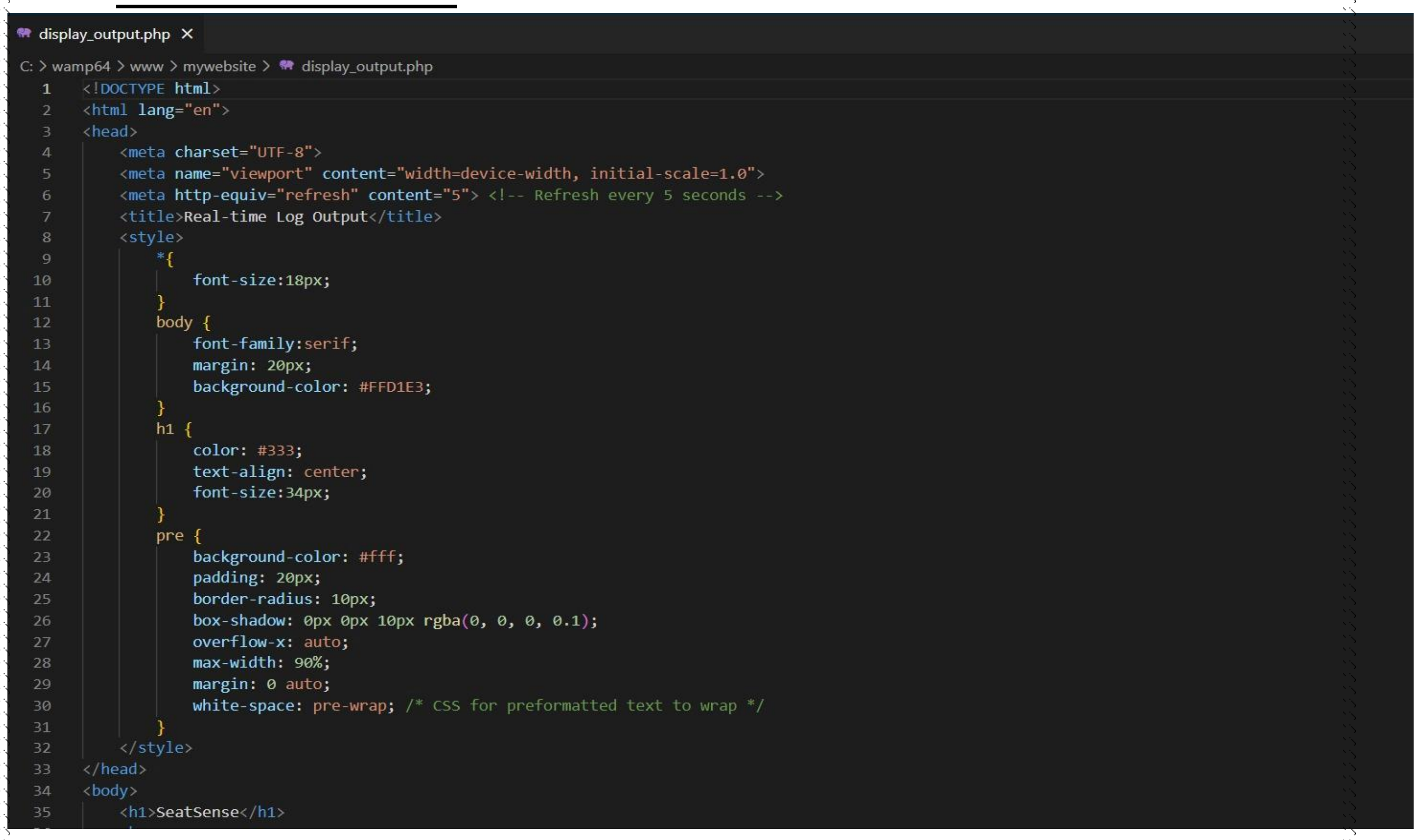
4. Communication with Clients:

- **API Integration:** The server exposes APIs for communication with other applications such as the mobile app and other transportation management systems, allowing them to access seat availability data.
- **Web Sockets:** For instant data updates, Web Sockets can be used to push real-time updates from the server to the client-side applications.

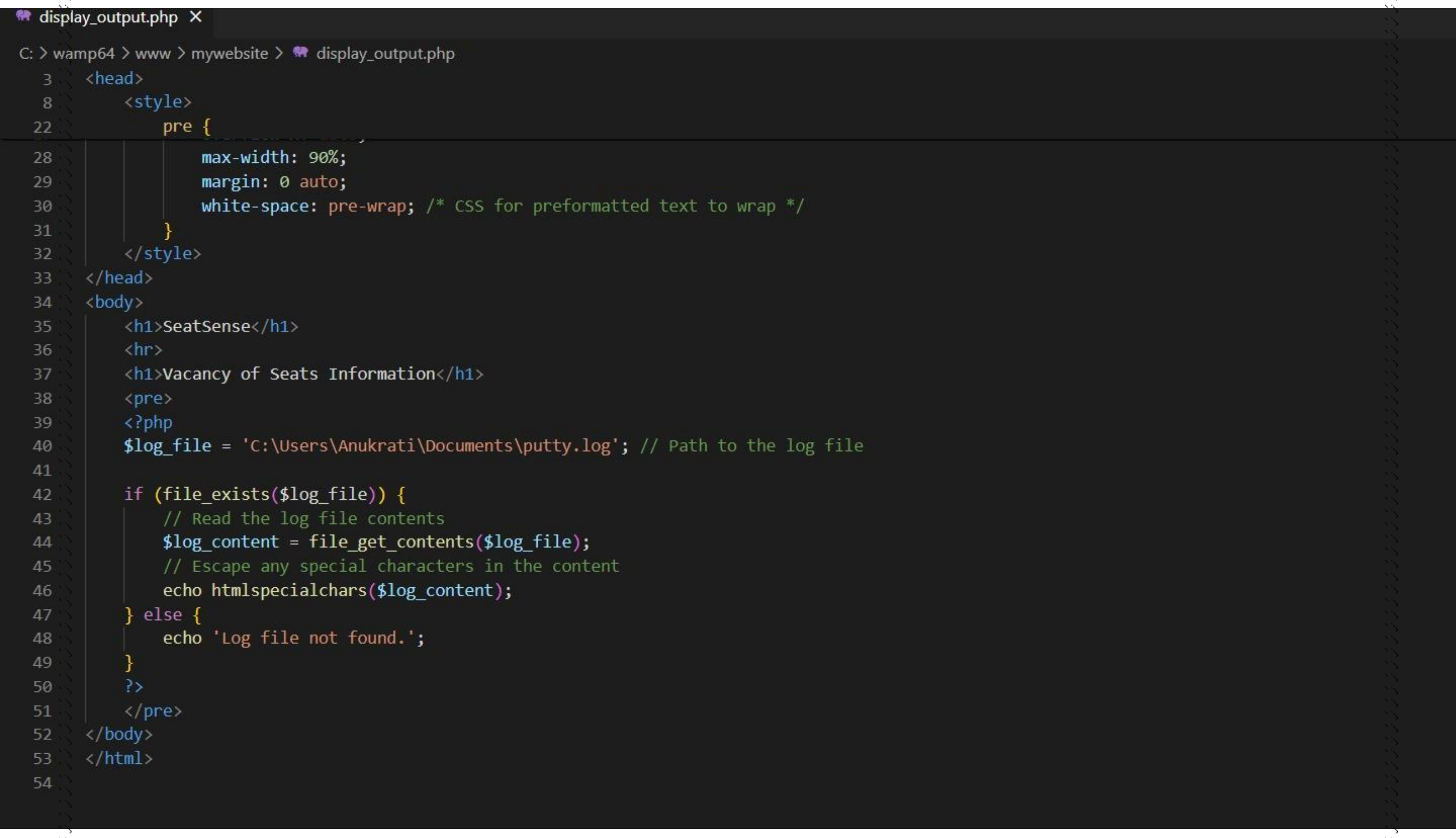
5. Customization and Expansion:

- **Customization: The server website can be customized to fit the specific needs of different transportation authorities, such as different data views and user roles.**
- **Scalability: The server is designed to handle varying levels of traffic and data volume, supporting potential expansion of the SeatSense project to multiple buses and routes.**

Code for website:

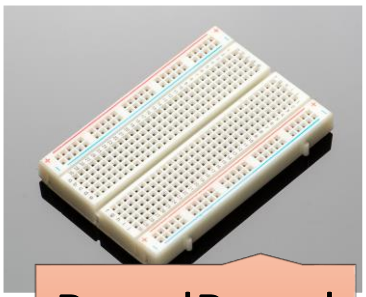


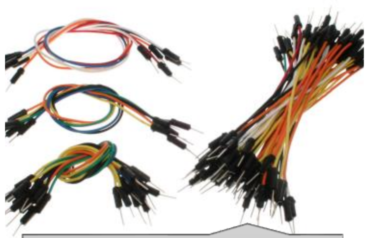
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="refresh" content="5"> <!-- Refresh every 5 seconds -->
    <title>Real-time Log Output</title>
    <style>
        *{
            font-size:18px;
        }
        body {
            font-family:serif;
            margin: 20px;
            background-color: #FFD1E3;
        }
        h1 {
            color: #333;
            text-align: center;
            font-size:34px;
        }
        pre {
            background-color: #fff;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
            overflow-x: auto;
            max-width: 90%;
            margin: 0 auto;
            white-space: pre-wrap; /* CSS for preformatted text to wrap */
        }
    </style>
</head>
<body>
    <h1>SeatSense</h1>
```



```
<head>
    <style>
        pre {
            max-width: 90%;
            margin: 0 auto;
            white-space: pre-wrap; /* CSS for preformatted text to wrap */
        }
    </style>
</head>
<body>
    <h1>SeatSense</h1>
    <hr>
    <h1>Vacancy of Seats Information</h1>
    <pre>
    <?php
    $log_file = 'C:\Users\Anukrati\Documents\putty.log'; // Path to the log file

    if (file_exists($log_file)) {
        // Read the log file contents
        $log_content = file_get_contents($log_file);
        // Escape any special characters in the content
        echo htmlspecialchars($log_content);
    } else {
        echo 'Log file not found.';
    }
    ?>
    </pre>
</body>
</html>
```

COMPONENTS USED:

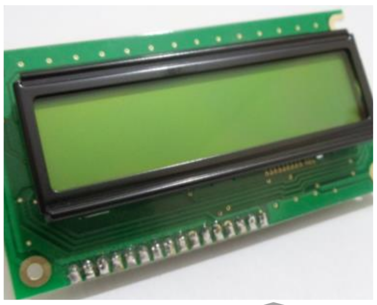

BreadBoard


Jumpwires

9V battery

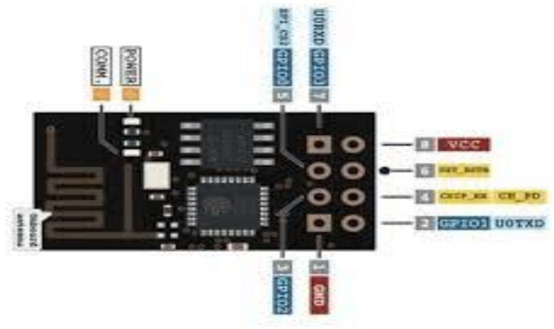
Arduino Uno

1R sensors-3


16*2mm LCD

16*2 LCD I2C

Interface Adapter



ESP01 ESP8266 Wireless

Transceiver Module

APPROACH TO MAKE THIS PROJECT:

1)Project Scope: Define the scope of the SeatSense Smart Bus Seat Monitoring System project and deployment locations (e.g., urban, suburban) the system will cover.

2)Components selection and Research: Research and select appropriate components considering factors such as application, usage, cost, compatibility and limitations.

3)System Design: Design the SeatSense Smart Bus Seat Monitoring Systems architecture, including hardware components (e.g., sensors), software algorithms for data processing, and integration with existing infrastructure.

4)Prototype Development: Develop a prototype of the SeatSense Smart Bus Seat Monitoring Systems, including building or acquiring hardware components, developing software algorithms, and designing a user interface for system monitoring.

5)Testing and Validation: Conduct field testing of the SeatSense Smart Bus Seat Monitoring Systems to validate accuracy and reliability under various conditions. Compare SeatSense Smart Bus Seat Monitoring Systems counts with manual counts or other validation methods.

6)Deployment and Integration: Implement the SeatSense Smart Bus Seat Monitoring Systems on selected vehicles or transportation routes, ensuring proper installation of hardware components and integration with existing infrastructure.

7)Data Collection and Analysis: Collect passenger count data from the deployed SeatSense Smart Bus Seat Monitoring Systems, analyze data to generate insights into passenger demand patterns, service performance, and other key metrics.

8)Monitoring and Maintenance: Monitor the performance of the SeatSense Smart Bus Seat Monitoring Systems continuously, conduct regular maintenance checks, software updates, and troubleshooting to address any issues.

9)Evaluation and Optimisation: Evaluate the effectiveness of the SeatSense Smart Bus Seat Monitoring Systems in achieving project objectives. Identify areas for optimization and improvement based on stakeholder feedback and data analysis.

10)Documentation and Training: Document the SeatSense Smart Bus Seat Monitoring Systems design, development process, and operational procedures. Provide training to relevant stakeholders involved in operating and maintaining the SeatSense Smart Bus Seat Monitoring Systems.

CHALLENGES FACED IN MAKING THE PROJECT:

Overcoming hurdles in the development process, we encountered several challenges that shaped our journey towards creating the Smart Bus Seat Monitoring System.

1. Transitioning from ESP01 to ESP8266 for better integration with the application:

- **Challenge:** Initially using ESP01 limited connectivity and capabilities, particularly when handling complex tasks and data transmissions for the app. This affected integration with the application.
- **Solution:** Switched to ESP8266, which provided more features and better performance. ESP8266 has more GPIO (General Purpose Input/Output) pins, greater memory, and a built-in microcontroller, which allowed for better compatibility with the application.
- **Implementation:** Rewrote the existing code to leverage ESP8266's features, including its enhanced Wi-Fi capabilities and additional I/O pins. This allowed for more efficient data transmission and a smoother integration process.

2. Switching from Blynk to Android Studio due to ESP01 unresponsiveness:

- **Challenge:** Using Blynk with ESP01 led to unresponsiveness issues, as ESP01's limited capabilities and unstable connection caused delays and data loss.
- **Solution:** Switched to Android Studio to develop a custom app. This switch provided greater flexibility in designing a user-friendly app tailored to the project's needs.
- **Implementation:** Designed and developed a new app using Android Studio, allowing for greater control over features and better optimization for ESP8266. This improved the overall responsiveness and stability of the system.

3. Adapting to using four IR sensors instead of two for Arduino and ESP8266 integration:

- **Challenge:** The initial setup with two IR sensors was insufficient for precise passenger counting, leading to potential inaccuracies.
- **Solution:** Added two more IR sensors, making a total of four. This increase improved the coverage and accuracy of passenger detection.
- **Implementation:** Updated the Arduino code to handle four IR sensors, adjusting for the increased number of inputs. This involved coding to interpret the sensor data accurately and adjusting logic for counting passengers.

4. **Identifying and rectifying an infinite loop issue in the Arduino and PHP code for website development on WAMP:**

- **Challenge:** An infinite loop issue in the Arduino and PHP code caused system crashes and disrupted communication between the app and the backend.
- **Solution:** Traced the loop issue through debugging techniques such as logging and error checking. Identified the root cause as a flawed loop structure or a missing exit condition.
- **Implementation:** Corrected the loop structure in both Arduino and PHP code to include proper exit conditions and error handling mechanisms. This ensured smooth data flow and prevented system crashes.

FUTURE SCALABILITY OF THE PROJECT:

1. **Predictive Analytics Integration:**

- **Forecasting Seat Availability:** By leveraging historical data, predictive analytics can be used to forecast seat availability, identifying peak and off-peak times, as well as potential demand fluctuations.
- **Passenger Experience Optimization:** This forecasting can guide transportation operators in adjusting schedules and routes to better meet demand, reducing overcrowding and wait times.
- **Machine Learning Models:** Develop machine learning models that continuously improve as more data is collected, providing increasingly accurate predictions over time.

2. **Expansion to Other Transportation Modes:**

- **Multimodal Coverage:** Extend the system beyond buses to include trains, trams, ferries, and other forms of public transportation, creating a unified platform for monitoring and optimizing passenger flow.
- **Common Infrastructure:** Maintain a consistent architecture and data protocols across different transportation modes to facilitate seamless expansion.
- **Integration with Transportation Networks:** Work with existing transportation networks and authorities to integrate SeatSense, improving overall system efficiency and passenger satisfaction.

3. **API Development for Third-Party Integration:**

- **Fostering Innovation:** Develop APIs that allow third-party developers to access seat occupancy data, enabling them to create applications and services that enhance the transportation ecosystem.
- **Collaboration Opportunities:** Collaborate with partners in mobility, logistics, and other sectors to offer new and innovative solutions for passengers and operators.
- **Data Security and Privacy:** Implement robust security and privacy measures in API development to protect sensitive data.

4. **Integration with Smart Payment Systems:**

- **Seamless Fare Collection:** Integrate with smart payment methods such as contactless cards, mobile payments, or digital wallets to streamline fare collection.
- **Unified User Experience:** Combine seat availability data with payment systems for a unified user experience, allowing passengers to book seats and pay fares in one place.
- **Data-Driven Pricing:** Utilize the data to implement dynamic pricing based on demand, providing incentives for off-peak travel and optimizing revenue.

5. **Continuous Feedback Loop:**

- **Passenger Feedback Collection:** Establish mechanisms to gather feedback from passengers, such as surveys or app-based reporting features.
- **Continuous Improvement:** Use feedback to identify areas for improvement and implement changes to enhance the passenger experience.
- **Engagement and Trust:** Engage passengers by acting on their feedback and building trust in the system's responsiveness and user-friendliness.

6. **Dynamic Route Optimization:**

- **Real-Time Data Utilization:** Develop algorithms that leverage real-time data on passenger demand and traffic conditions to optimize routes dynamically.
- **Efficiency Gains:** Optimize routes to reduce travel times and improve efficiency, saving fuel and lowering operational costs.
- **Passenger Convenience:** Adjust routes in response to demand changes, offering more convenient travel options for passengers.

RESULTS AND ACHIEVEMENTS:

1)Real-Time Seat Availability:

- **Instant Updates:** The system delivers real-time updates on seat availability through various channels: hardware LCD displays within the bus, an Android application for mobile users, and a website for broader access.
- **Improved Passenger Experience:** Real-time information allows passengers to make informed decisions regarding their commute, such as which bus to take or where to sit, enhancing their overall travel experience.
- **Multiple Access Points:** By offering information through multiple channels, the system caters to diverse user preferences and accessibility needs, ensuring that passengers have options to receive updates in the manner most convenient for them.

2)Seamless Integration:

- **Harmonized Components:** The successful integration of hardware and software components ensures smooth functioning across all platforms, creating a cohesive user experience.
- **Interconnected Systems:** The combination of Arduino, ESP8266, sensors, and the software backend results in a well-coordinated system that efficiently processes data and communicates it to the app, website, and LCD displays.
- **Ease of Use:** The seamless integration of various components simplifies the system's operation and maintenance, providing a user-friendly interface for both transportation operators and passengers.

3)Improved Efficiency:

- **Data Synchronization:** Synchronized data presentation across LCD displays, mobile applications, and websites ensures consistent and accurate information is provided to all users.
- **Informed Decision-Making:** Passengers can make informed decisions about their travel plans, such as which bus to take based on seat availability, thus improving overall journey efficiency.
- **Optimized Bus Operations:** Transportation operators benefit from the real-time data, enabling them to optimize routes and schedules for improved punctuality and resource utilization.

4)Overcoming Challenges:

- **Technical Hurdles:** The project faced several challenges, such as hardware integration, data transmission issues, and software development hurdles, which were addressed and resolved.

- **<u>Code Debugging</u>:** Challenges in the Arduino and PHP code were identified and rectified, ensuring smooth functioning and preventing system crashes.
- **<u>Infrastructure Improvements</u>:** Wiring and cable management were optimized to avoid interference and ensure reliable data transmission throughout the system.
- **<u>Continuous Innovation</u>:** By overcoming these challenges, the project has set a solid foundation for future growth and improvements, paving the way for continuous innovation in seat monitoring and public transportation efficiency.