Smart Transportation System

Government College of Engineering, Bodinayakanur

Computer Science and Engineering

Submitted by:

Gayathri R

Table of Contents

1. **Abstract**

This project involves creating a smart parking system using embedded systems with Keil software. It will utilize sensors to detect parking space occupancy and provide real-time updates to users via a mobile app or digital displays. The system aims to optimize parking space usage and enhance the overall parking experience.

2. **Introduction**

Urbanization has led to increased vehicle ownership, exacerbating the challenge of finding parking spaces in crowded cities. Traditional parking management systems often struggle to efficiently utilize parking resources, resulting in congestion and wasted time. To address these issues, this project proposes the development of a smart parking system using embedded systems technology with the Keil software platform. This system aims to optimize parking space utilization, reduce congestion, and enhance the overall parking experience for users by leveraging real-time data and intelligent decision-making algorithms.

3. **Methodology**

1. Requirements Analysis: Conduct a comprehensive analysis of the project requirements, including functional and non-functional aspects, user needs, and system constraints.

2. System Design: Design the architecture of the smart parking system, specifying components such as sensors, microcontrollers, control unit, and user interfaces.

3. Embedded Software Development: Develop embedded software using the Keil µVision IDE and C/C++ programming language to program microcontrollers for sensor interfacing, data processing, and communication with the control unit.

4. Sensor Deployment: Deploy parking space sensors in the target parking facility, ensuring optimal placement and installation methods for accurate vehicle occupancy detection.

5. Control Unit Implementation: Implement the centralized control unit to collect sensor data, aggregate parking occupancy information, and provide real-time updates to users through mobile apps or digital displays.

6. User Interface Development: Develop user interfaces such as mobile apps or digital displays to enable users to access parking availability information, navigate to available spaces, make payments, and reserve parking spots

4. **Existing Work**

Existing work in the field of smart parking systems involves various approaches such as sensor-based detection, data analytics, and user interfaces to optimize parking space utilization, reduce congestion, and improve the overall parking experience for users.

5. **Proposed Work**

The proposed work aims to develop a smart parking system using embedded systems technology with the Keil software platform. This system will utilize sensors to detect parking space occupancy, microcontrollers to process data, and a centralized control unit to provide real-time updates to users. Additionally, user interfaces such as mobile apps or digital displays will be implemented to enhance the parking experience.

6. **System Requirements**

- **Hardware Components**

  - 1. Microcontrollers: Select microcontrollers compatible with Keil µVision IDE, capable of interfacing with sensors and the centralized control unit.

  - 2. Parking Space Sensors: Sensors like ultrasonic, infrared, or magnetic sensors for detecting vehicle occupancy in real-time.

- 3. Communication Modules: Hardware components (e.g., Wi-Fi, Bluetooth, LoRa modules) to establish communication between sensors, microcontrollers, and the centralized control unit.

- **Software Components**

  - 1. Keil µVision IDE: A development environment for writing, compiling, and debugging embedded software using C/C++.

  - 2. RTOS (Real-time Operating System): If needed, an RTOS can manage tasks, scheduling, and resource allocation in the embedded system.

- 3. Communication Protocols Support: Software support for communication protocols such as UART, SPI, I2C, and CAN for interfacing with sensors and the centralized control unit.

7. **Implementation Details** :

a. Sensors and Cameras: Install sensors and cameras throughout the parking Lot to monitor occupancy and detect vehicles entering and exiting.

b. Data Collection and Processing: Gather data from sensors and cameras to track parking availability in real-time. Process this data to analyze parking patterns and optimize space utilization.

c. Mobile App or Website: Develop a user-friendly interface for drivers to check parking availability, reserve spots in advance, and navigate to available spaces.

d. Automated Payment System: Integrate a payment system that allows users to pay for parking digitally, either through the app or via automated kiosks at the parking lot.

e. Dynamic Signage: Install digital signs at key points in the parking lot to display real-time availability and guide drivers to available spaces

f. Data Analytics and Optimization: Implement data analytics algorithms to analyze parking usage patterns, optimize parking space allocation, and improve overall system efficiency. By analyzing historical data, the system can identify peak usage times, predict parking demand, and dynamically adjust pricing or allocate resources accordingly.

8. **Data Collection and Processing**

- Security Measurea

mplement security measures such as encryption, authentication, and access control to protect data transmitted over the communication protocol. This safeguards sensitive information and prevents unauthorized access or tampering.

- Scalability and Flexibility:

  Design the communication protocol to be scalable and flexible to to accommodate future expansions or enhancements to the parking systems.

  This includes support for additional sensors, gateways, and server nodes as the system grows.

- Error Handling and Recovery:

  Incorporate error handling mechanisms into the communication protocol to detect and recover from communication failures or data discrepancies.

9. **Communication Protocols**

1.Standardized Protocol: Adopt a standardized communication protocol such as

MQTT (Message Queuing Telemetry Transport) or CoAP (ConstrainedApplication Protocol) for IoT devices. These protocols are lightweight,

efficient, and suitable for the constrained environments typical of

parking systems.

2. Sensor-to-Gateway Communication: Enable sensors to

communicate with a central gateway using the chosen protocol. This

allows for real-time transmission of parking occupancy data from

sensors to the central system.

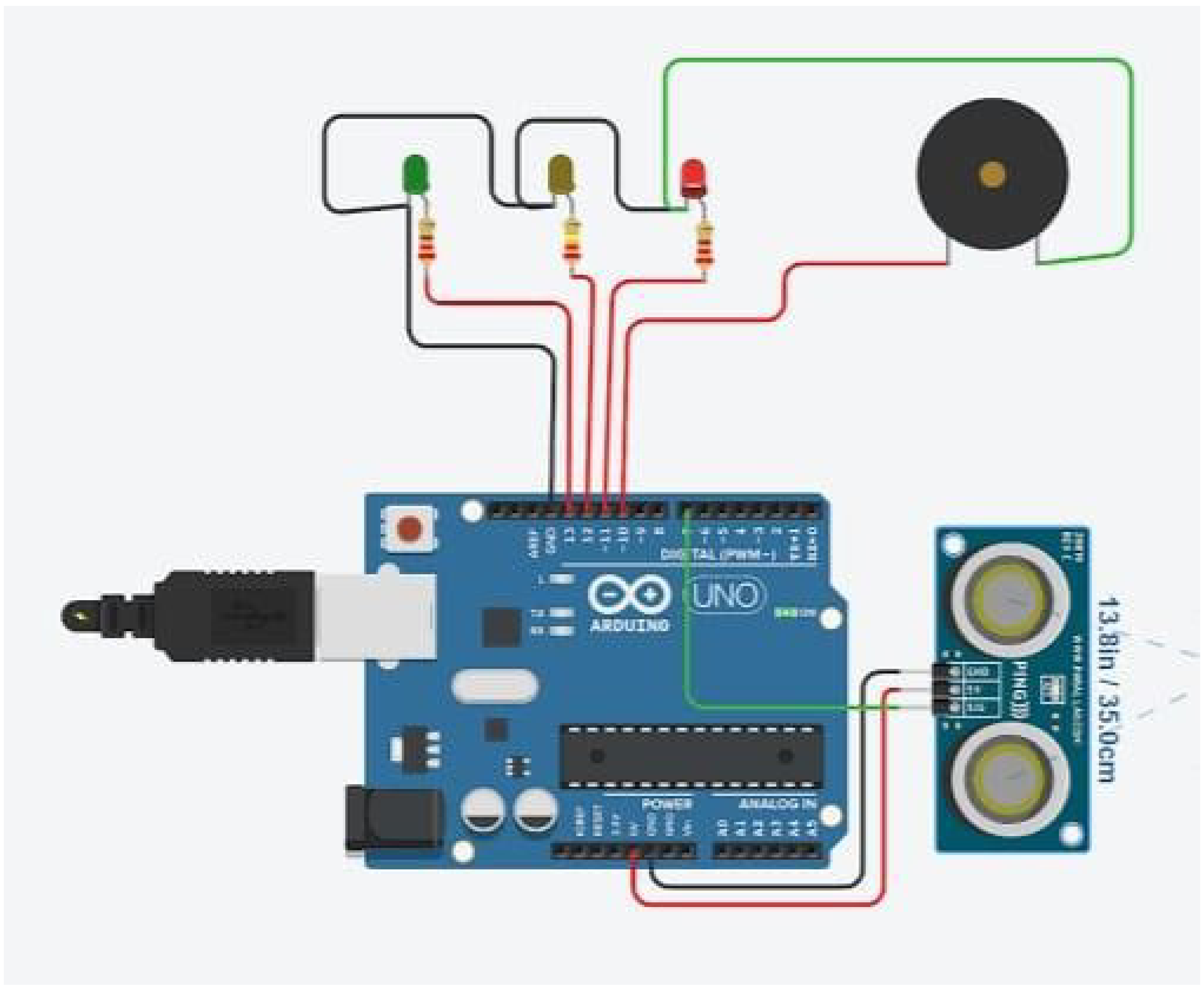
3. Gateway-to-Server Communication: Establish communication

between the gateway and the central server. The gateway aggregates data from multiple sensors and relays it to the server for processing and analysis.

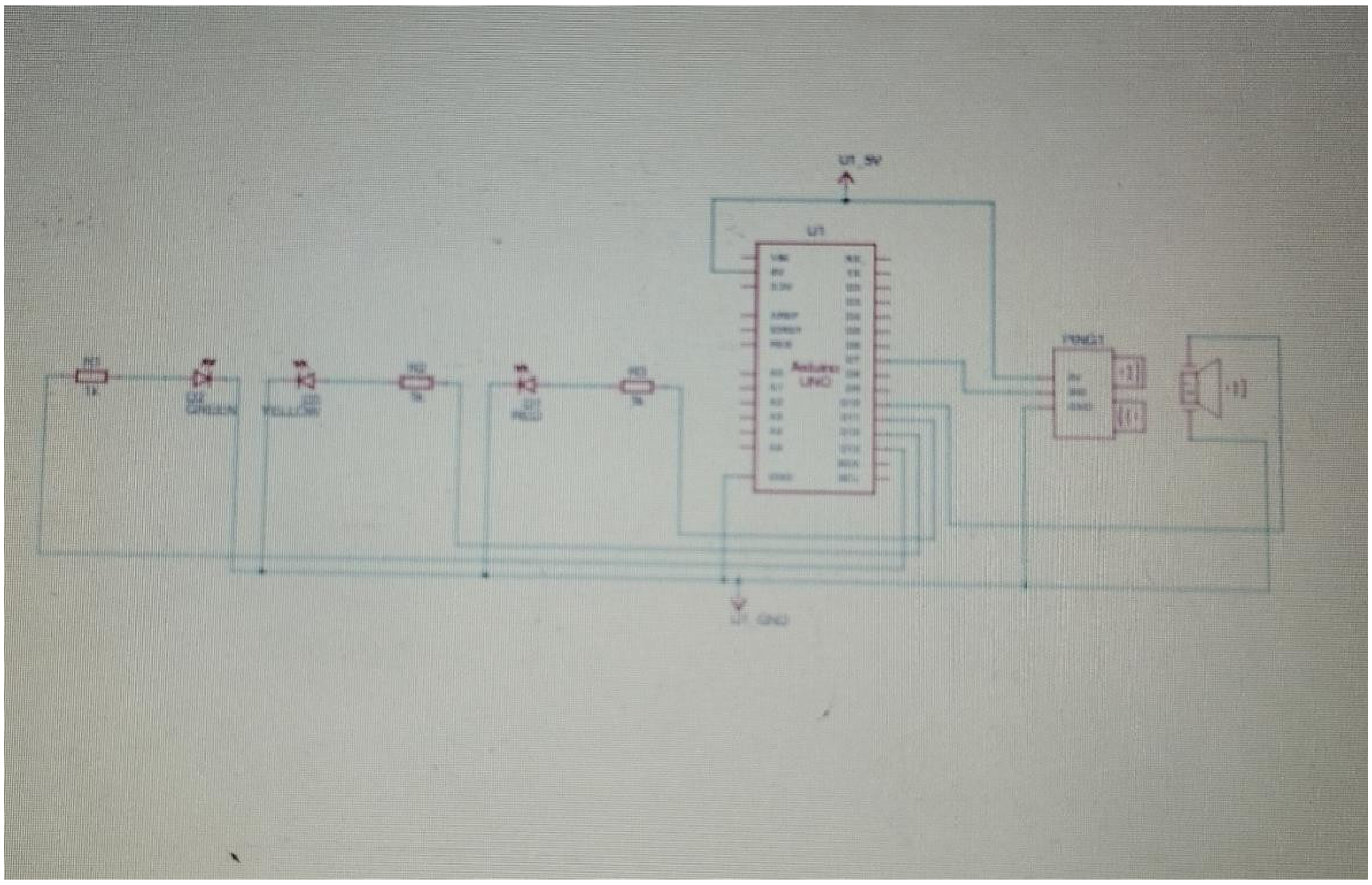
4. Server-to-User Communication: Develop APIs (Application Programming Interfaces) or web services on the server side to provide access to parking data for user-facing applications such as mobile apps or websites. These APIs should adhere to RESTful principles for easy integration and scalability.

5. Real-time Updates: Implement mechanisms for real-time updates between the server and user applications. This ensures that users receive up-to-date information on parking availability and other relevant data.

10. **Block Diagram**



11. **Circuit Diagram**



12. **Program Code**

(Embedded C) : #define green

```
13

#define yellow 12

#define red 11

#define buzzer 10

int inches = 0;

int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

 pinMode(triggerPin, OUTPUT); // Clear the trigger

 digitalWrite(triggerPin, LOW);
 delayMicroseconds(2);
 // Sets the trigger pin to HIGH state for 10
 microseconds digitalWrite(triggerPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(triggerPin, LOW);
 pinMode(echoPin, INPUT);
 // Reads the echo pin, and returns the sound wave travel
time in
microseconds
 return pulseIn(echoPin, HIGH);
}
void setup(){
 Serial.begin(9600);
 pinMode(green,OUTPUT);
 pinMode(yellow,OUTPUT);
 pinMode(red,OUTPUT);
 pinMode(buzzer,OUTPUT);
}
void loop()
{
 // measure the ping time in cm
 cm = 0.01723 * readUltrasonicDistance(7, 7);
```

```
// convert to inches by dividing by

2.54 inches = (cm / 2.54);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.println("cm");

delay(100); // Wait for 100 millisecond(s)


if(cm>=100)

{

digitalWrite(green,HIGH);

delay(500);

digitalWrite(green,LOW);

delay(500);

}


else if(cm>=50 && cm<100)

{

digitalWrite(yellow,HIGH);

delay(500);

digitalWrite(yellow,LOW);

delay(500);
```

```
 }
 else if(cm>=25 && cm<50)
 {
 digitalWrite(red,HIGH);
 delay(500);
 digitalWrite(red,LOW);
 delay(500);
}
 else if(cm<25)
 {
 digitalWrite(buzzer,HIGH);
 delay(500);
 digitalWrite(buzzer,LOW);
 delay(500);
 }
}
```

13. Simulation Output Link

https://www.tinkercad.com/things/e64kLEq0rMR-shiny-lappi

14. Conclusion

In summary, the smart parking system developed using embedded systems with Keil software offers an effective solution to urban parking challenges. Through real- time data, efficient processing, and user-friendly interfaces, it optimizes parking space use, reduces congestion, and improves the overall parking experience. Future enhancements could include integrating advanced technologies like machine learning and augmented reality for further optimization and integration with broader smart city infrastructure.