1. INTRODUCTION

The rapid urbanization and increasing number of vehicles on the road have led to significant challenges in managing parking spaces efficiently. Traditional parking systems often rely on manual processes, which can be time-consuming and prone to errors. The "Smart Car Parking System" addresses these issues by integrating modern technology to automate and streamline the parking process.

1.1 Overview

The Smart Car Parking System is a comprehensive solution designed to enhance the parking experience for both users and parking lot managers. The system utilizes a combination of hardware components, such as IR sensors, ESP32 microcontroller, LEDs, and servo motors, to monitor and control parking spaces. Additionally, a user-friendly mobile application facilitates the reservation and payment process, ensuring a seamless interaction for the user.

Key features of the system include:

- ➔ *Real-time Parking Space Monitoring*: IR sensors detect the presence of vehicles in parking slots, providing real-time information on available and occupied spaces.
- ➔ *Automated Barriers*: Servo motors control the entry and exit barriers, reducing the need for human intervention and ensuring secure access to the parking lot.
- ➔ *User-Friendly Mobile Application*: The app allows users to search for available parking spots, reserve a spot, navigate to the location, and make payments, all from their mobile device.
- ➔ *LED Indicators*: LEDs indicate the status of parking slots, with green LEDs showing available spaces and red LEDs indicating occupied ones.

1.2 Motivation

The motivation behind the Smart Car Parking System stems from the need to address common problems faced by drivers and parking lot operators. These problems include the difficulty in finding available parking spaces, inefficient use of parking resources, and the time-consuming process of manually managing parking lots. The system aims to:

- ➔ *Reduce Search Time:* By providing real-time information on available parking spaces, the system helps drivers quickly find a spot, reducing the time spent searching and lowering traffic congestion in parking areas.
- ➔ *Enhance User Convenience:* The mobile application simplifies the process of finding and reserving a parking spot, making it more convenient for users.
- ➔ *Improve Parking Management:* Automated monitoring and control of parking spaces improve the efficiency of parking lot management, reducing the need for manual intervention and minimizing errors.
- ➔ *Increase Revenue:* By optimizing the use of parking spaces and reducing unauthorized parking, the system can help parking lot operators increase their revenue.

Overall, the Smart Car Parking System aims to create a more efficient, user-friendly, and secure parking experience, leveraging modern technology to solve everyday challenges in urban environments. The following sections of this report will delve into the specific requirements, design, implementation, and results of the project, demonstrating how these objectives were achieved.

2. SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

2.1 Hardware Requirements

1. **ESP32 Development Board**
   - ESP32 DevKit V1
2. **Jumper Wires**
   - Male-to-male and male-to-female jumper wires for connecting components
3. **Breadboard**
   - For prototyping and connecting components
4. **MicroUSB Cable**
   - For programming and powering the ESP32 board
5. **Computer with USB Port**
   - For programming the ESP32 and monitoring serial output
6. **Infrared Sensors and Servo Motors**
   - Connected to the ESP32

2.2 Software Requirements

1. **Development Environment:**

- *Arduino IDE:*
  - Version: 1.8.19 or newer
- *ESP32 Board Support Package:*
  - Version: 2.0.3 or newer

2. **Libraries for Functionality:**

- *Blynk Library*
  - For integrating the ESP32 with the Blynk app:
    - Version: 1.0.1 or newer
- *Wifi Library*
  - For the ESP32 to connect to a Wifi Network:
    - Pre-installed with the ESP32 Board

3. Libraries for ESP32 Power Management:

- *ESP32 Power Management Libraries*
    - ESP32-specific power management functions:
    - Included with the ESP32 Board

4. Programming and Uploading Code:

- *Operating System to Burn Code to ESP32:*
    - Windows, Linux
- *Arduino IDE's built-in uploader*
- *USB drivers for ESP32*
    - Usually included in the board support package

5. Frontend Technologies for Blynk:

- *Blynk Frontend Technologies*
    - Blynk App (Mobile)
        - Available on iOS and Android
    - Blynk Cloud
        - Provides backend services and dashboard for the Blynk app
    - Blynk HTTP(S) API
        - For sending and receiving data from the ESP32 to Blynk cloud

6. Cloud Services and Server:

- *Cloud and Server*
    - Blynk Cloud
        - Default server used by Blynk for handling IoT devices

3. DESIGN

The design of the Smart Car Parking System involves the integration of various hardware components as shown in the hardware model diagram. The ESP32 microcontroller serves as the central unit, interfacing with IR sensors, LEDs, and servo motors.

3.1 Model of Hardware

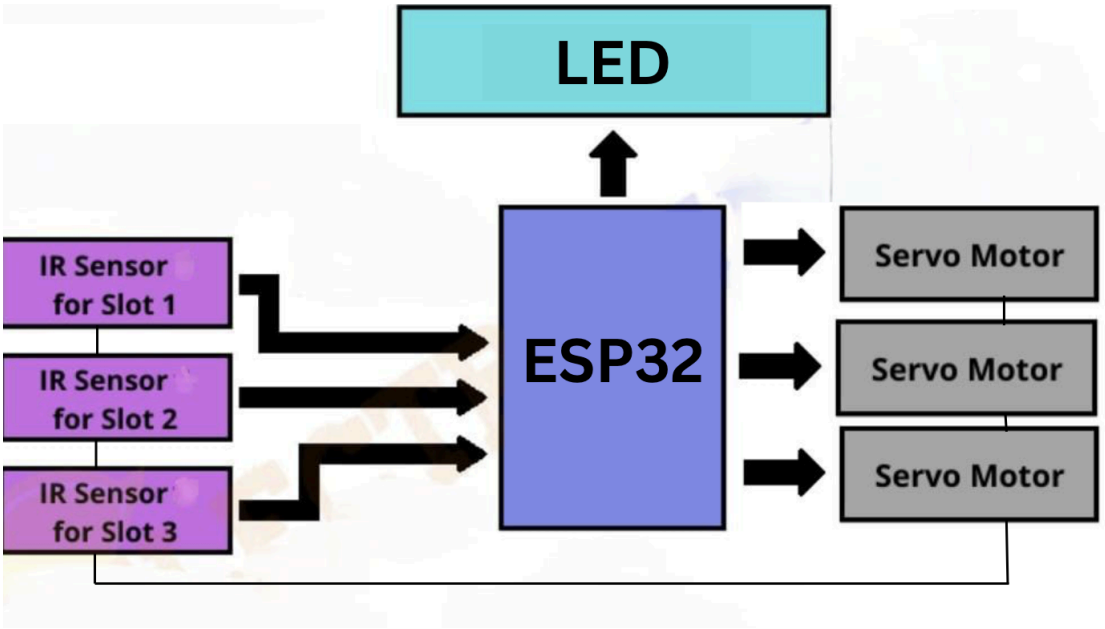

Fig 3.1.1 Hardware model

The hardware model for the Smart Car Parking System is designed to efficiently manage and monitor parking spaces using a combination of sensors, controllers, and actuators. The key components and their interactions are depicted in the hardware model diagram.

3.2 Blueprint of Hardware

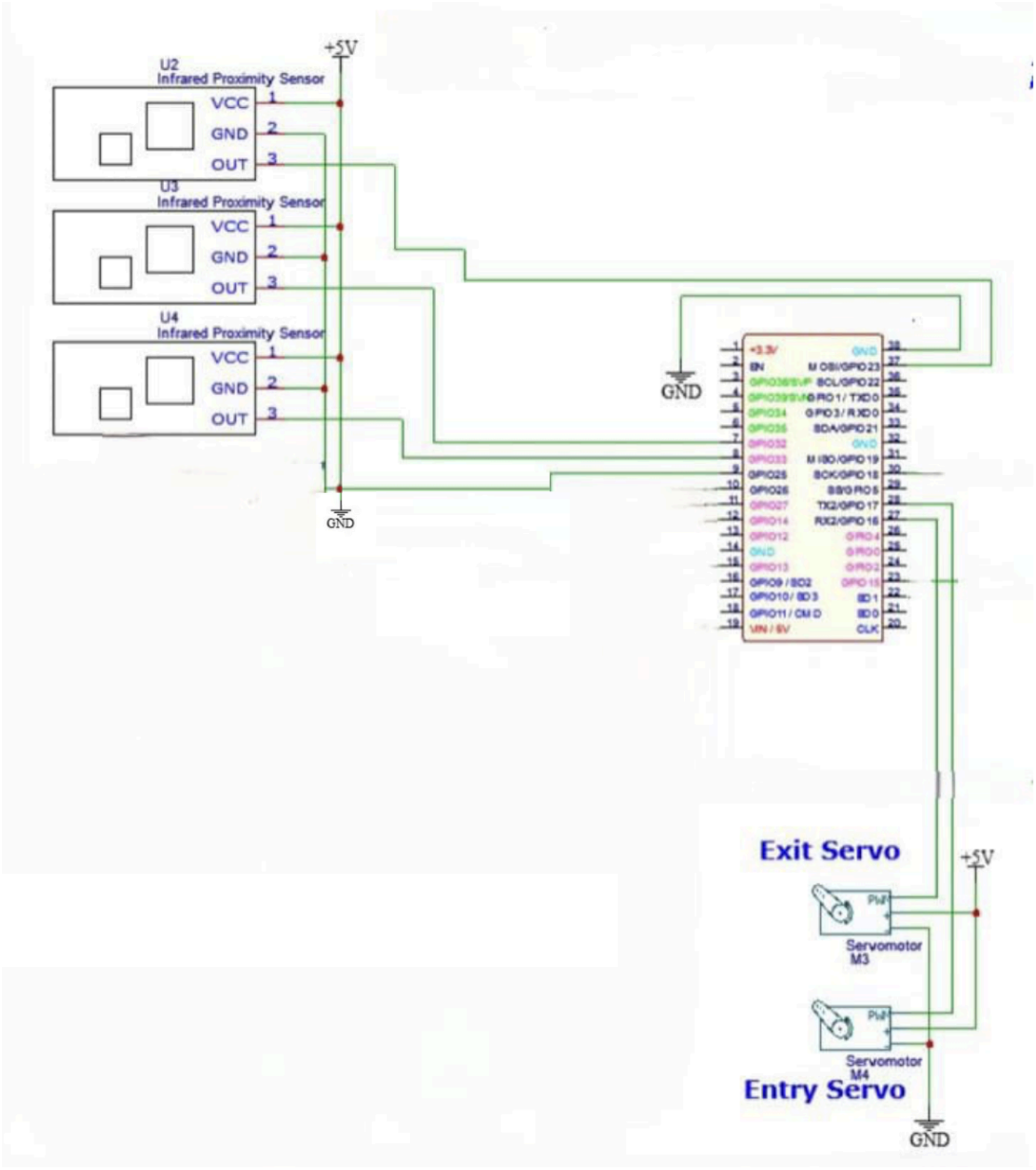

Fig 3.2.1 Hardware Blueprint

This blueprint provides a comprehensive view of how the various components are interconnected, ensuring the smooth operation of the smart parking system. The layout is organized to ensure clarity, with all connections clearly labeled and color-coded, specifically highlighting the integration of various components with the ESP32 development board.

4. IMPLEMENTATION

The implementation of the Smart Car Parking System includes the following steps:

1. Setting up the ESP32: Configure the ESP32 with the Arduino IDE and install the necessary libraries.
2. *Connecting the IR Sensors*: Attach the IR sensors to detect the presence of vehicles in different slots.
3. Controlling the LEDs: Use LEDs to indicate the availability of parking slots. Green LEDs represent available slots, while red LEDs indicate occupied slots.
4. Operating the Servo Motors: Servo motors control the barriers at the entry and exit points of the parking lot. They are operated based on the input from IR sensors.
5. Programming the Logic: Write the code to integrate all components, ensuring smooth communication and operation based on sensor inputs and predefined logic.

4.1 User Interaction:

The user interaction with the Smart Parking System is designed to be seamless and user-friendly. Below is the flow of user interaction:

1. **Start**: The user initiates the process.
2. **Download & Install App**: The user downloads and installs the Smart Parking System app on their mobile device.
3. **User Registration/Login**: The user is prompted to either sign up or log in.
    - **Sign Up**: New users enter their details to create an account.
    - **Login**: Returning users verify their credentials to log in.
4. **Home Screen**: After successful login, the user is taken to the home screen.
5. **Search for Parking Spot**: The user searches for available parking spots.
6. **View Available Spots**: The app displays the available parking spots.
7. **Select Spot**: The user selects a desired parking spot.
8. **Reserve Spot**: The user reserves the selected parking spot.
9. **Navigate to Spot**: The app provides navigation instructions to the reserved parking spot.
10. **Park Vehicle**: The user parks the vehicle in the spot.
11. **End**: The process is completed, and the user can exit the app.

4.2 Implementation Code:

```
#include <ESP32Servo.h>
// IR sensor pins
#define IR_SENSOR_PIN_1 4
#define IR_SENSOR_PIN_2 5
#define IR_SENSOR_PIN_3 6
// Servo motor pins
#define SERVO_PIN_1 2
#define SERVO_PIN_2 14
#define SERVO_PIN_3 15
Servo servoMotor1;
Servo servoMotor2;
Servo servoMotor3;
unsigned long timestamp1 = 0; // Variable to store timestamp for IR sensor 1
unsigned long timestamp2 = 0; // Variable to store timestamp for IR sensor 2
unsigned long timestamp3 = 0; // Variable to store timestamp for IR sensor 3
void setup() {
 Serial.begin(9600);
 // Initialize servo motors
 servoMotor1.attach(SERVO_PIN_1);
 servoMotor2.attach(SERVO_PIN_2);
 servoMotor3.attach(SERVO_PIN_3);
 // IR sensor pin modes
 pinMode(IR_SENSOR_PIN_1, INPUT);
 pinMode(IR_SENSOR_PIN_2, INPUT);
 pinMode(IR_SENSOR_PIN_3, INPUT);
}
void loop() {
 // Read IR sensor values
 int irValue1 = digitalRead(IR_SENSOR_PIN_1);
 int irValue2 = digitalRead(IR_SENSOR_PIN_2);
 int irValue3 = digitalRead(IR_SENSOR_PIN_3);
 // Record timestamp for IR sensor 1
 if (irValue1 == LOW && timestamp1 == 0) {
 timestamp1 = millis(); // Record timestamp when motion detected
 Serial.print("Motion detected by Sensor 1 at: ");
 Serial.println(timestamp1);
 servoMotor1.write(0); // Rotate servo 1 to 0 degrees
 } else if (irValue1 == HIGH) {
 timestamp1 = 0; // Reset timestamp if no motion
 servoMotor1.write(90); // Rotate servo 1 to 90 degrees
 }
 // Record timestamp for IR sensor 2
 if (irValue2 == LOW && timestamp2 == 0) {
 timestamp2 = millis(); // Record timestamp when motion detected
 Serial.print("Motion detected by Sensor 2 at: ");
 Serial.println(timestamp2);
 servoMotor2.write(0); // Rotate servo 2 to 0 degrees
```

```
} else if (irValue2 == HIGH) {
timestamp2 = 0; // Reset timestamp if no motion
servoMotor2.write(90); // Rotate servo 2 to 90 degrees
}
// Record timestamp for IR sensor 3
if (irValue3 == LOW && timestamp3 == 0) {
timestamp3 = millis(); // Record timestamp when motion detected
Serial.print("Motion detected by Sensor 3 at: ");
Serial.println(timestamp3);
servoMotor3.write(0); // Rotate servo 3 to 0 degrees
} else if (irValue3 == HIGH) {
timestamp3 = 0; // Reset timestamp if no motion
servoMotor3.write(90); // Rotate servo 3 to 90 degrees
}
delay(100); // Delay for stability
```

4.3 Hardware Implementation

- **ESP32 Development Board:**
  - Role: Acts as the central processing unit, managing data from the GPS module and coordinating communication with the Blynk app.
  - Connection: Interfaces with all other components via GPIO pins and serial ports.
- **Breadboard and Jumper Wires:**
  - Role: Facilitate the prototyping and connection of various components.
  - Usage: Allows for flexible and temporary connections during development.
- **Computer with USB Port:**
  - Role: Used for programming the ESP32, uploading code, and monitoring serial output.
  - Software: The Arduino IDE is utilized for code development and deployment.
- **LED Indicators:**
  - A single LED is used to indicate the availability of parking slots. When a slot is available, the LED will light up in green. When a slot is occupied, the LED will light up in red.
- **Servo Motors:**
  - Three servo motors are used to control barriers for the three parking slots. These motors are connected to the ESP32, which sends commands to raise or lower the barriers based on the occupancy status of the slots.

4.4 Working Mechanism:

- **IR Sensors Detection:**
  - When a vehicle enters a parking slot, the IR sensor for that slot detects its presence and sends a signal to the ESP32 microcontroller.
- **Data Processing by ESP32:**
  - The ESP32 processes the input from the IR sensors to determine which slots are occupied and which are available. Based on this information, the ESP32 updates the status of the LED indicators.
- **LED Indicators:**
  - The LED indicators reflect the status of each parking slot. If a slot is available, the corresponding LED will be green. If a slot is occupied, the LED will be red.
- **Servo Motor Control:**
  - The ESP32 sends commands to the servo motors to raise or lower the barriers based on the occupancy status. When a vehicle is detected in a slot, the corresponding barrier will be lowered to prevent another vehicle from entering. When the slot becomes available, the barrier will be raised.

This hardware model ensures an efficient and automated system for managing parking spaces, reducing the need for manual intervention and enhancing the overall user experience.

5. RESULT

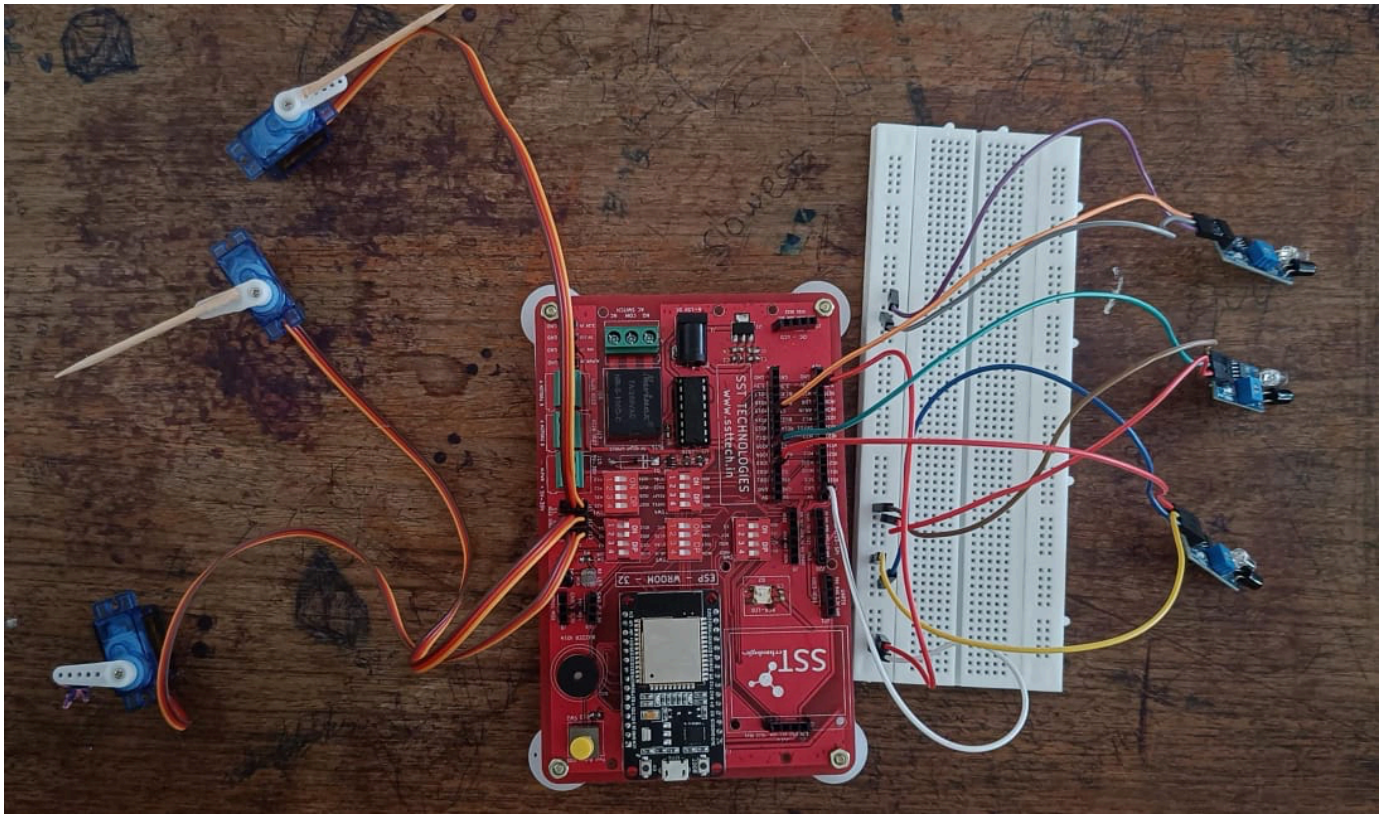

Fig 5.1: Connections on Actual Hardware

This image showcases the physical setup of the smart parking system, illustrating the hardware connections required for the system to function. The setup includes an ESP32 development board, which acts as the central controller. Various sensors, likely infrared (IR) sensors, are connected to the ESP32 to detect the presence of vehicles in parking spaces. The wiring setup is clearly visible, with connections running from the ESP32 to each sensor

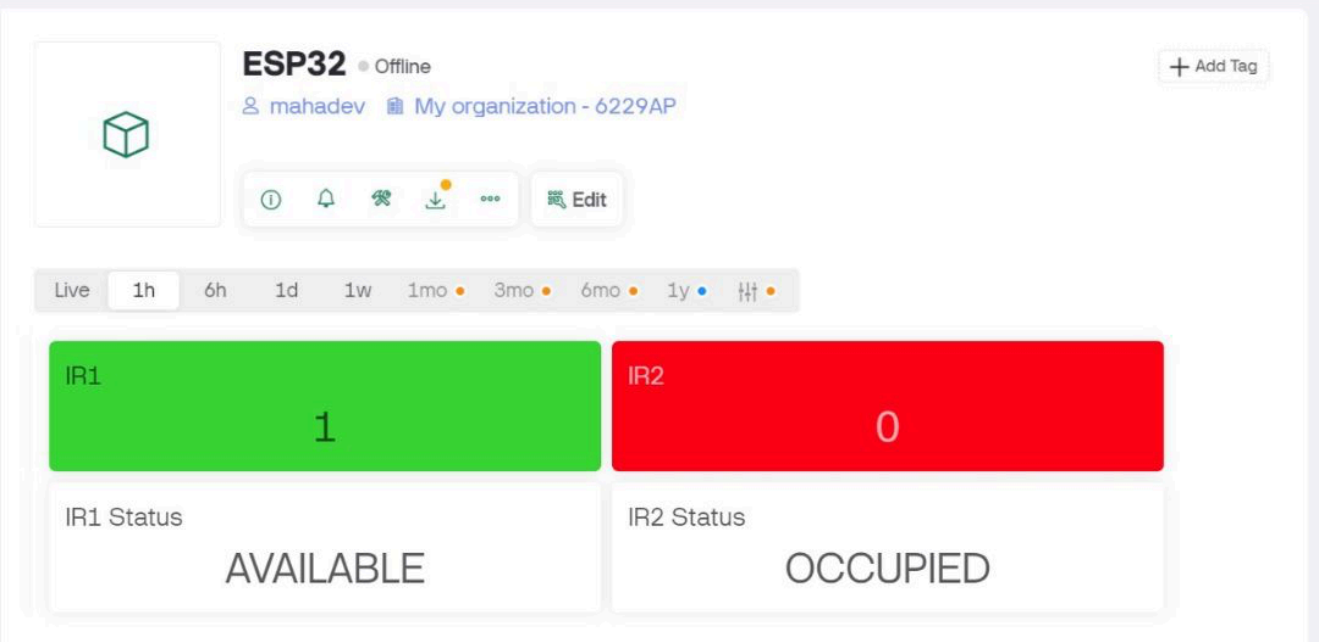

Fig 5.2: Blynk output

The Blynk output image shows a user interface for monitoring the status of two infrared (IR) sensors in a smart parking system. The interface indicates that IR1 shows "AVAILABLE" with a green background, meaning the corresponding parking space is free. IR2 shows "OCCUPIED" with a red background, indicating that the parking space is currently occupied. This user-friendly display provides a real-time view of the parking space availability, accessible remotely through the Blynk application.

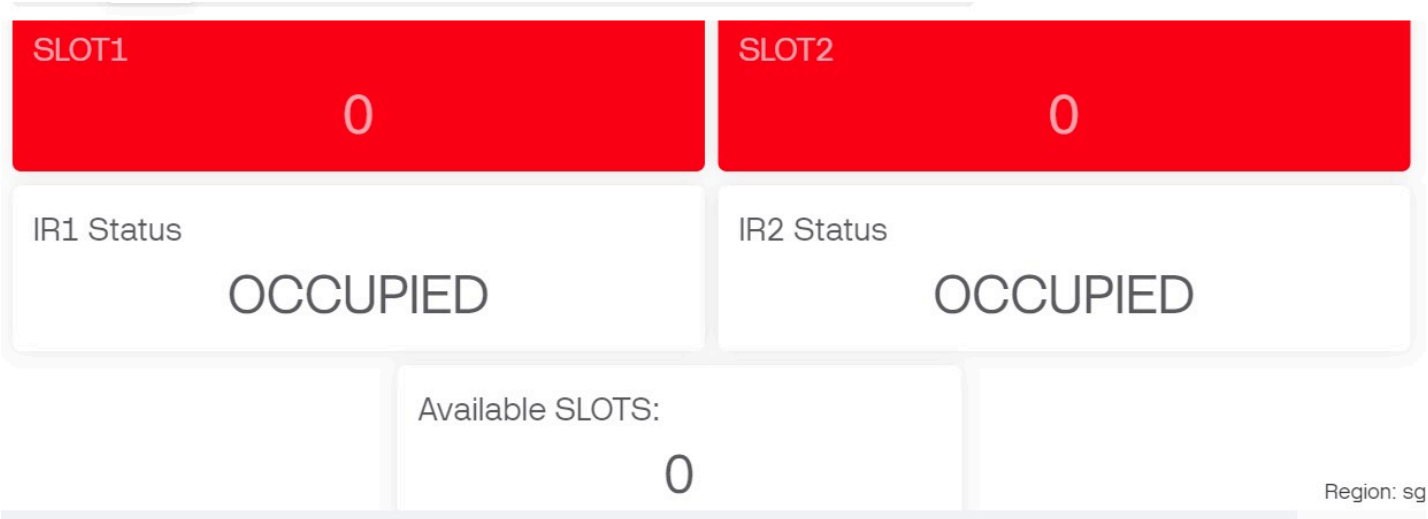

Fig 5.3: Available Slots

The image displays the available slots free for parking at any given moment. Currently, there are no available slots due to both slots being occupied.

Fig 5.4: Serial Monitor Output

This image displays the output from a Serial Monitor, which is used to interact with an ESP32 development module. The monitor shows the status of different sensors (sensor1, sensor2) connected to the system. The numbers following the sensor labels indicate the sensor's status, likely representing the detection of a vehicle (1 for detected and 0 for not detected). This output helps in debugging and confirming that the sensors are correctly identifying the presence of vehicles in the parking slots.

6. CONCLUSION

The Smart Car Parking System is a technologically advanced solution designed to address the common issues associated with traditional parking management systems. By integrating modern hardware components such as the ESP32 microcontroller, IR sensors, LED indicators, and servo motors, the system offers a robust and efficient mechanism to monitor and control parking slots.

The implementation of this system has numerous benefits:

1. **Efficiency:**
    - The automated nature of the Smart Car Parking System significantly enhances the efficiency of parking management. Real-time data processing ensures that the status of each parking slot is continuously updated, allowing users to quickly identify available spaces without manual intervention.
2. **User Convenience:**
    - The system provides a seamless user experience by simplifying the process of finding and reserving a parking spot. The mobile app integration allows users to search for available spots, reserve them in advance, and receive navigation assistance to the selected spot, thereby reducing the time spent searching for parking.
3. **Reduced Congestion:**
    - By streamlining the parking process, the system helps in reducing congestion within parking areas. Users are directed to available slots efficiently, minimizing traffic buildup and improving the overall flow of vehicles within the parking facility.
4. **Enhanced Security:**
    - The use of IR sensors and servo motors ensures that each parking slot is monitored and controlled effectively. Unauthorized access to reserved or occupied slots is prevented, enhancing the security of parked vehicles.
5. **Cost-Effectiveness:**
    - Although the initial setup of the system involves a cost for hardware and installation, the long-term benefits outweigh these expenses. Reduced labor costs, efficient space utilization, and improved user satisfaction contribute to the overall cost-effectiveness of the system.

6. **Scalability:**
    - The modular design of the system allows for easy scalability. Additional sensors and slots can be integrated without significant modifications to the existing setup, making it suitable for parking facilities of varying sizes.
7. **Environmental Impact:**
    - By reducing the time vehicles spend idling and searching for parking, the system contributes to lower emissions and a reduced environmental footprint. Efficient parking management supports sustainable urban development.

In conclusion, the Smart Car Parking System represents a significant advancement in the domain of parking management. Its integration of cutting-edge technology not only enhances the user experience but also contributes to the overall efficiency and security of parking operations. The successful implementation of this system can serve as a model for future developments in smart city infrastructure, paving the way for more intelligent and sustainable urban environments.

REFERENCES

1. Arduino Documentation: https://www.arduino.cc/en/Guide/HomePage
2. ESP32 Datasheet: https://www.espressif.com/en/products/socs/esp32
3. Servo Motor Specifications: https://www.towerpro.com.tw/product/servo-motor/
4. IR Sensor Information: https://components101.com/sensors/ir-sensor-module