Smart Parking System Using NodeMCU and Cloud

Lingamaneni Devansh, CS21B2023

I. AIM

The aim of the smart car parking system project is to develop an efficient parking management solution solely utilizing ESP8266 microcontroller, IR sensor for entry and exit detection, and additional sensors for monitoring parking slot occupancy. Integration with a servo motor facilitates automated barrier control for entry and exit. The system aims to optimize parking space utilization, minimize congestion, and enhance user convenience by providing real-time information on parking slot availability via a web-based or mobile application interface. This solution contributes to reducing traffic congestion, improving urban mobility, and enhancing the overall parking experience for drivers, all within the constraints of the provided components.

II. Apparatus/Software Required

NodeMCU ESP8266 Microcontroller, IR Sensors, Servo Motor, Blynk Cloud Platform, Wi-Fi Connection, Jumper Wires,Breadboard

III. Procedure

A. Introduction

The growing problem of parking shortages has become a major worry for both drivers and urban planners in our fast changing society. In addition to wasting time, the never-ending search for a place to park also exacerbates traffic congestion, increases air pollution, and elevates passenger stress levels. Nonetheless, the use of technology in the shape of intelligent parking systems offers a glimmer of hope. By saving time and effort, parking occupancy monitoring systems assist in locating open spots and efficiently directing cars to empty lots. This creative approach ushers in a new era of urban transportation by automating the parking process through the use of sensors, data analysis, and the Internet of Things (IoT).The system's objectives are to improve parking efficiency, lessen the negative consequences of using conventional parking techniques, and improve overall user experience by utilizing smart parking technology.

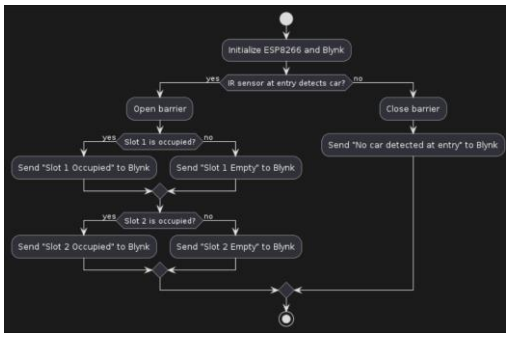


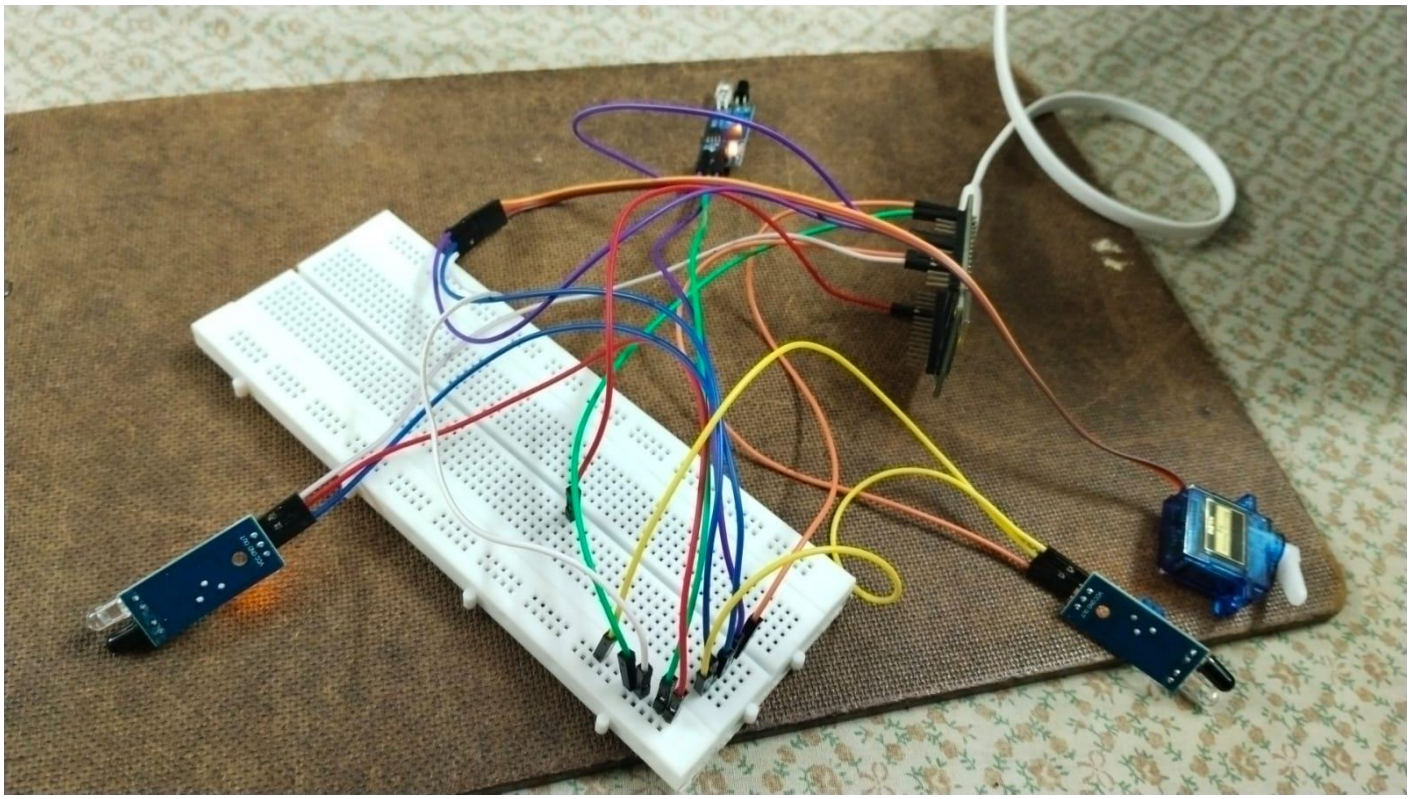
Fig. 1. Flowchart of project's implementation

B. Methodology/Problem Statement

This addresses the drawbacks of conventional parking management systems by utilizing IoT technology and the chosen components. Using IR sensors, a servo motor, and the NodeMCU ESP8266 microprocessor, a smart parking system with real-time car detection and barrier control may be created. By integrating with the Blynk cloud platform, stakeholders may view data visualization and remote monitoring capabilities, enabling them to obtain parking status updates from any location. By giving timely insights into parking slot availability, this system maximizes the use of available parking spaces, reduces traffic, and improves the entire parking experience for users.

C. Implementation

1) Hardware Design: The hardware setup consists of a NodeMCU ESP8266 microcontroller
- NodeMCU ESP8266 Microcontroller: Serves as the central control unit, managing the system and interfacing with sensors.
- IR Sensors: Detects vehicle presence in parking lots and communicates with the ESP8266. - IR Sensors: Detects vehicle presence in parking lots and communicates with the ESP8266.
- Connectivity: Achieved through jumper wires and breadboard for reliable connections.



2) Executable Code:

```
#define BLYNK_TEMPLATE_ID "TMPL3_s_hPGqs"
#define BLYNK_TEMPLATE_NAME "parking"
#define BLYNK_AUTH_TOKEN "3rbEF6h3S5ByWSBk5q1p_JjeHmACnU7I"

#include <Servo.h>
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

```
// WiFi credentials
char ssid[] = "D";
char password[] = "dev@n$h777";
char auth[] = BLYNK_AUTH_TOKEN;

// Blynk virtual pin assignments
#define VIRTUAL_PIN_SLOT1 V10
#define VIRTUAL_PIN_SLOT2 V11

// Define GPIO pins for IR sensors and servo motor
const int irSensorPin1 = D3;   // IR sensor 1 connected to D3 (GPIO 0)
const int irSensorPin2 = D6;   // IR sensor 2 connected to D6 (GPIO 12)
const int irSensorPin3 = D4;   // IR sensor 3 connected to D4 (GPIO 2)
const int servoPin = D5;       // Servo motor connected to D5 (GPIO 14)

int freeslots = 2;  // Initial number of free parking slots
Servo myServo;



void setup() {
  Serial.begin(9600);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");

  // Initialize Blynk with WiFi and authentication token
  Blynk.begin(auth, WiFi.SSID().c_str(), WiFi.psk().c_str());

  // Attach servo to the GPIO pin
  myServo.attach(servoPin);

  // Initialize IR sensor pins
  pinMode(irSensorPin1, INPUT);
  pinMode(irSensorPin2, INPUT);
  pinMode(irSensorPin3, INPUT);

}

void loop() {
  Blynk.run();  // Continuously run Blynk to maintain connection

  // Read IR sensor states
  int sensorState1 = digitalRead(irSensorPin1);
  int sensorState2 = digitalRead(irSensorPin2);
  int sensorState3 = digitalRead(irSensorPin3);

  // Determine overall parking status based on sensor states
```

```
if (sensorState1 == HIGH ) {
  // Vehicle detected in at least one parking slot, rotate servo to 180 degrees
  myServo.write(180);
} else {
  // No vehicle detected in any parking slot, rotate servo to 0 degrees
  myServo.write(0);
}

// Update free slots based on sensor readings
if (sensorState2 == HIGH ) {
  if (freeslots < 2) {
    freeslots++; // Increment freeSlots if a vehicle leaves a parking slot
    Blynk.virtualWrite(VIRTUAL_PIN_SLOT1, 0); // Turn off virtual bulb for slot 1
  }
} else {
  if (freeslots > 0) {
    freeslots--; // Decrement freeSlots if a vehicle occupies a parking slot
    Blynk.virtualWrite(VIRTUAL_PIN_SLOT1, 255); // Turn on virtual bulb for slot 1
  }
}

if (sensorState3 == HIGH ) {
  if (freeslots < 2) {
    freeslots++; // Increment freeSlots if a vehicle leaves a parking slot
    Blynk.virtualWrite(VIRTUAL_PIN_SLOT2, 0); // Turn off virtual bulb for slot 2
  }
} else {
  if (freeslots > 0) {
    freeslots--; // Decrement freeSlots if a vehicle occupies a parking slot
    Blynk.virtualWrite(VIRTUAL_PIN_SLOT2, 255); // Turn on virtual bulb for slot 2
  }
}



delay(1000); // Adjust delay for stability
  }
```

3) Software Implementation: Using the appropriate programming tools, the code for the NodeMCU ESP8266 microcontroller is created to function on that platform. Its primary function is to enable the communication between the sensors in the smart parking system and the Blynk cloud platform. Through this code, the microcontroller synchronizes with the sensors to collect necessary information that is sent out to the Blynk cloud for further processing and showing. Real-time monitoring of parking space occupancy provides actionable insights for effective parking management.

IV. RESULTS/OUTPUT

Smart Vehicle Parking

- Infrared (IR) sensors are being used by our project to detect the presence of vehicles in parking lots.
- Data is gathered and processed by the NodeMCU ESP8266 microcontroller from these sensors.
- The data is utilized to determine the real-time occupancy of parking lots.

Data Visualization

- A web-based dashboard on the Blynk cloud platform displays real-time parking occupancy data, including available and occupied parking spaces.

- By using web browsers or the Blynk mobile application, users can view the parking space information in this dashboard.
- The system is also designed to send notification alerts when parking spaces are full or if any unusual parking behaviors are detected, which enhances user awareness and helps manage the park more efficiently.

V. Conclusion and Future Directions

In conclusion, the development of the IoT-based smart vehicle parking system offers significant benefits in parking management efficiency and user experience. By integrating sensors with the NodeMCU ESP8266 microcontroller and leveraging the Blynk cloud platform, we've created a cost-effective and scalable solution for real-time parking space monitoring. The web-based dashboard empowers users with

actionable insights for informed decision-making and efficient parking space utilization.

Future Directions: Looking ahead, there are opportunities to enhance the system further. These include:

- Sensor Optimization: Continuously calibrating and optimizing sensor readings for improved accuracy.
- Additional Sensor Integration: Incorporating more sensors to enhance parking space detection and management.
- Machine Learning Algorithms: Implementing algorithms to analyze parking data trends and predict future demand.
- Mobile Application Development: Creating a mobile app for seamless access to real-time parking

REFERENCES

- Documentation for Blynk Cloud Code - https://docs.blynk.io/en