GOVERNMENT COLLEGE OF ENGINEERING, ERODE

அரசினர் பொறியியல் கல்லூரி, ஈரோடு
Government College of Engineering, Erode
(Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai)

B.E Electronics and Communication Engineering

SMART PARKING

Done By

TEAM LEADER: HARIKUMAR S	731121106305
TEAM MEMBER: SADAI ESWARAN M	731121106039
TEAM MEMBER: MEYYARASAN B	731121106307
TEAM MEMBER: SIVABALAN R	731121106308

Under the mentor of **Dr.M.SATHYAKALA**

Department of Information Technology (IT)

Department of Electronics and Communication Engineering.

Government College of Engineering

Erode ,PO ,near Vasavi College,TamilNadu-638316, Affiliated to Anna University ,Chennai.

SMART PARKING SYSTEM PHASE 4

IoT Smart Parking project, it can enhance the functionality and user experience by incorporating web development technologies. Here's how can integrate web technologies into various aspects of the project:

1. Web-based Dashboard for Administrators:

Create a web-based dashboard for administrators to monitor and manage the parking system. This dashboard should provide real-time information about parking spot occupancy, reservations, and transaction history. Use web development technologies like HTML, CSS, and JavaScript, and consider using a web framework for efficiency.

2. Mobile App:

Develop a mobile app to reserve parking spots, make payments, and receive notifications. Use cross-platform mobile app development frameworks like React Native or Flutter to streamline app development for both Android and iOS.

3. Online Reservation System:

Implement a web-based reservation system for students to check parking spot availability and make reservations. This system can be integrated with the mobile app and can be developed using standard web technologies.

4. Payment Gateway Integration:

If you include a payment system, you'll need to integrate a payment gateway into your web app for processing payments. Popular payment gateways often provide APIs for this purpose. Here's a simplified example using Python and Flask:

5. Real-time Updates:

Use web development technologies to ensure real-time updates on parking spot availability, reservation confirmation, and payment status. You can achieve this with technologies like WebSocket for real-time communication between the server and clients.

6. User Authentication and Management:

For user authentication and management, you can create user registration and login systems within the mobile app and web interface. Use web development technologies for user interfaces and backend logic

7. Data Analytics and Reporting:

Utilize web technologies to create data analytics and reporting features for administrators. You can use JavaScript libraries for data visualization and reporting tools.

PROGRAM:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change the HEX address

#include <Servo.h>
Servo myservo1;

int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 6;  // Digital pin for the smoke detector
int BuzzerPin = 7;         // Digital pin for the buzzer

int Slot = 4;  // Enter Total number of parking Slots

bool flag1 = false;
bool flag2 = false;

unsigned long lastLcdUpdate = 0;  // Variable to track the time of the last LCD
update
unsigned long lcdUpdateInterval = 1000;  // Update the LCD every 1000
milliseconds (1 second)

void setup() {
  lcd.begin(16, 2);  // Initialize LCD with 16 columns and 2 rows
  lcd.backlight();
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(SmokeDetectorPin, INPUT);
  pinMode(BuzzerPin, OUTPUT);

  myservo1.attach(3);
  myservo1.write(100);

  lcd.setCursor(0, 0);
  lcd.print("     ARDUINO     ");
  lcd.setCursor(0, 1);
  lcd.print(" PARKING SYSTEM ");
  delay(2000);
  lcd.clear();

  Serial.begin(9600);  // Start serial communication for debugging
}

void loop() {
  if (digitalRead(IR1) == LOW && !flag1) {
    if (Slot > 0) {
      flag1 = true;
      if (!flag2) {
        myservo1.write(0);
        Slot--;
      }
    } else {
      displayMessage("    SORRY :(    ", "  Parking Full  ");
    }
  }

  if (digitalRead(IR2) == LOW && !flag2) {
    flag2 = true;
    if (!flag1) {
      myservo1.write(0);
      Slot++;
    }
  }

  if (flag1 && flag2) {
    delay(1000);
    myservo1.write(100);
    Serial.println("Servo returned to initial position.");
    flag1 = false;
    flag2 = false;
  }

  // Update the LCD display with a delay
  if (millis() - lastLcdUpdate >= lcdUpdateInterval) {
    updateLcdDisplay();
    lastLcdUpdate = millis();
  }

  // ... (Rest of your code)
}

void updateLcdDisplay() {
  if (digitalRead(SmokeDetectorPin) == HIGH) {
    displayMessage("   WARNING!   ", " Smoke Detected ");
    digitalWrite(BuzzerPin, HIGH);  // Turn on the buzzer
  } else {
    displayMessage("    WELCOME!    ", "Slot Left: " + String(Slot));
    digitalWrite(BuzzerPin, LOW);   // Turn off the buzzer
  }
}

void displayMessage(const char *line1, const String &line2) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}
```

PRORAM DISCRIPTION:

This Arduino program appears to be controlling a parking system with certain features. Here's an explanation of each part of the program:

1. Library Inclusions:

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

These lines include necessary libraries for I2C communication with a Liquid Crystal Display (LCD) and for controlling a servo motor.

2. LiquidCrystal_I2C Initialization:

This initializes an LCD object with the I2C address 0x27 and specifies the display's dimensions (16 columns and 2 rows).

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

3. Variable Declarations:

These lines declare various variables for pins, flags, and timing used in the program.

```
int IR1 = 2;
int IR2 = 4;
int SmokeDetectorPin = 6;
int BuzzerPin = 7;
int Slot = 4;
bool flag1 = false;
bool flag2 = false;
unsigned long lastLcdUpdate = 0;
unsigned long lcdUpdateInterval = 1000;
```

4. Setup Function:

The setup function is called once when the Arduino board is powered on or reset. It initializes pins, the LCD, the servo motor, and sets up serial communication.

```
  void setup() {
    // Initialize LCD, set pins, and
configure Serial communication.
    // Also, it initializes the servo motor
and displays a welcome message.
  }
```

5. Loop Function:

The loop function runs repeatedly and is the core of the program. It checks the state of two infrared (IR) sensors (IR1 and IR2), updates the LCD, and triggers a buzzer if smoke is detected.

```
  void loop() {
    // Continuously checks the state of
IR sensors and updates the LCD.
    // If smoke is detected, it triggers
the buzzer.
  }
```

6. IR Sensor Logic:

The program checks the state of IR sensors (IR1 and IR2). When a car is detected by IR1 and IR2 is not active, it means a car is entering, and the program decreases the available parking slot count (Slot). If both IR sensors are triggered, indicating a car has passed through, it returns the servo to the initial position.

7. LCD Display Update:

The program updates the LCD display at a regular interval. If smoke is detected by a smoke detector, it displays a warning message and activates the buzzer. Otherwise, it displays the number of parking slots available.

8. Update Lcd Display Function:

This function is called to update the LCD display based on the status of the smoke detector and the parking slot count.

```
  void updateLcdDisplay() {
    // Checks smoke detector status and updates the LCD accordingly.
  }
```

9. Display Message Function:

This function clears the LCD and displays a message on it. It's used for showing different messages on the LCD screen.

```
  void displayMessage(const char *line1, const String &line2) {
    // Clears the LCD and displays a message on it.
```

Conclusion:

This program controls a parking system with features to detect and manage the entry and exit of cars, display the number of available parking slots on an LCD, and trigger a buzzer in case of smoke detection. It makes use of IR sensors, a servo motor, and a smoke detector to achieve these functionalities.