IOT PHASE 3

TOPIC:SMART PARKING

NAME RAUSHAN KUMAR

REGISTER NO:950321104040

INTRODUCTION

A car parking system using Arduino is a modern and efficient solution designed to manage and streamline the process of parking vehicles in various settings, such as commercial parking lots, residential complexes, and public parking areas. This system combines the power of Arduino microcontrollers and various sensors to automate parking space allocation, vehicle identification, and payment processing.

The primary objective of a car parking system with Arduino is to provide convenience to drivers while optimizing space utilization and improving overall management. This system typically employs a range of sensors, including ultrasonic or infrared sensors, to detect the presence of vehicles in parking spaces. It uses Arduino microcontrollers to process this data and make decisions based on the occupancy status of parking spots.

1 COMPONENT NEEDED

Creating a smart car parking system with Arduino requires a variety of components to detect, manage, and control parking spaces efficiently. Below is a list of essential components typically used in such a system:

1. Arduino Microcontroller: An Arduino board (e.g., Arduino Uno, Arduino Mega) serves as the brain of the system, processing data from various sensors and controlling actuators.

2. Parking Space Sensors:

- Ultrasonic Sensors: Used to detect the presence of vehicles in parking spaces.

- Infrared Sensors: Can be an alternative to ultrasonic sensors for vehicle detection.

3. Display and Indicators:

- LED Displays: Provide real-time information about parking space availability.

- LCD Screens: Display additional information and instructions for drivers.

4.Communication Module:

- Wi-Fi, Bluetooth, or GSM module: Used for communication with a central server or mobile app to relay parking space availability data to users.

5. Actuators:

- Servo Motors: Control barrier gates or entry/exit systems.

- Relays: Used for switching on and off various components, such as lights and alarms.

6. Power Supply:

- A stable power source, such as a 5V power supply for Arduino and various sensors and components.

7. Security and Surveillance:

- CCTV Cameras: Monitor the parking area for security and surveillance.

- Microphones and Speakers: Provide audio monitoring and announcements.

8. Payment Processing System:

- Payment gateways and card readers for automated payment processing.

9. Data Storage and Logging:

- SD Card or EEPROM: Store data related to parking duration, occupancy, and transactions.

10. Software:

- Develop the necessary code using Arduino IDE to control the system, process sensor data, and communicate with users and central servers.

11. Sensors for Entry/Exit:

- Additional sensors and RFID or barcode readers to identify vehicles when entering and exiting the parking area.

PROGAMMING THE ESP32

#include <Servo.h>

Servo barrierServo; // Create a servo object to control the barrier gate

int ultrasonicSensorPin = 2; // Pin for the ultrasonic sensor

int ledDisplayPin = 8; // Pin for the LED display

int barrierOpenAngle = 90; // Angle at which the barrier gate is open

int barrierClosedAngle = 0; // Angle at which the barrier gate is closed

void setup() {

barrierServo.attach(9); // Attach the servo to pin 9

pinMode(ultrasonicSensorPin, INPUT);

pinMode(ledDisplayPin, OUTPUT);

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

}

void loop() {

int distance = measureDistance(); // Read the distance from the ultrasonic sensor

if (distance < 30) {

openBarrier();

displaySpaceOccupied();

} else {

closeBarrier();

displaySpaceAvailable();

}

}

int measureDistance() {

// Use ultrasonic sensor to measure distance and return the result

// Implement the code to measure distance here

}

void openBarrier() {

barrierServo.write(barrierOpenAngle); // Open the barrier gate

// You can add additional actions like starting a timer for parking duration here

}

void closeBarrier() {

barrierServo.write(barrierClosedAngle); // Close the barrier gate

}

void displaySpaceOccupied() {

digitalWrite(ledDisplayPin, HIGH); // Turn on the LED display to show occupied status

// You can also send data to a central server or mobile app to update parking status

}

void displaySpaceAvailable() {

digitalWrite(ledDisplayPin, LOW); // Turn off the LED display to show available status

// You can send data to update parking status as well

}

DIGRAM

