PROJECT TITLE: SMART PARKING

PHASE-2

INTRODUCTION:

In an era of rapid urbanization and escalating vehicle ownership, the quest for efficient urban mobility solutions is more pressing than ever. Traffic congestion and the eternal search for parking spaces have become everyday nuisances, posing significant environmental, economic, and social challenges. In response, smart parking systems have emerged as beacons of hope, revolutionizing the way cities manage their transportation infrastructure. Harnessing technologies like IoT, data analytics, and AI, smart parking optimizes operations, reduces congestion, and enhances the urban living experience.

PROCEDURE:

Creating a smart parking system using an ESP32 and an HC-SR04 ultrasonic distance sensor is a great project idea. This

system can help you monitor and manage parking spaces efficiently. Here's a step-by-step guide on how to set up such a system:

Components Needed:

1. ESP32 development board
2. HC-SR04 ultrasonic distance sensor
3. Breadboard and jumper wires
4. Power source for the ESP32 (e.g., USB cable or battery)
5. Optional: LEDs and resistors for status indicators

Step 1: Circuit Setup

Connect the HC-SR04 sensor to the ESP32 as follows:

- HC-SR04 VCC to ESP32 5V or 3.3V
- HC-SR04 GND to ESP32 GND
- HC-SR04 Trig (Trigger) to an ESP32 GPIO pin (e.g., GPIO2)
- HC-SR04 Echo to an ESP32 GPIO pin (e.g., GPIO15)

You can also add LEDs to indicate parking space availability, connecting them to additional GPIO pins on the ESP32.

Step 2: Arduino IDE Setup

Make sure you have the Arduino IDE set up for ESP32 development. You'll need to install the ESP32 board in the Arduino IDE.

1. Go to "File" > "Preferences" in the Arduino IDE.

2. In the "Additional Boards Manager URLs" field, add this URL: `https://dl.espressif.com/dl/package_esp32_index.json`

3. Click "OK" and close the Preferences window.

4. Go to "Tools" > "Board" > "Boards Manager."

5. Search for "esp32" and install the "esp32" board by Espressif Systems.

6. Select your ESP32 board under "Tools" > "Board."

Step 3:Coding

```
#define TRIG_PIN1  26 // ESP32 pin GPIO26 connected to Ultrasonic Sensor 1's TRIG pin
#define ECHO_PIN1  25 // ESP32 pin GPIO25 connected to Ultrasonic Sensor 1's ECHO pin
#define LED_PIN1   18 // ESP32 pin GPIO18 connected to LED 1's pin

#define TRIG_PIN2  27 // ESP32 pin GPIO27 connected to Ultrasonic Sensor 2's TRIG pin
#define ECHO_PIN2  32 // ESP32 pin GPIO32 connected to Ultrasonic Sensor 2's ECHO pin
#define LED_PIN2   33 // ESP32 pin GPIO33 connected to LED 2's pin

#define TRIG_PIN3  14 // ESP32 pin GPIO14 connected to Ultrasonic Sensor 3's TRIG pin
#define ECHO_PIN3  12 // ESP32 pin GPIO12 connected to Ultrasonic Sensor 3's ECHO pin
#define LED_PIN3   13 // ESP32 pin GPIO13 connected to LED 3's pin

#define DISTANCE_THRESHOLD 50 // centimeters

// variables for sensor 1
float duration_us1, distance_cm1;

// variables for sensor 2
```

```
float duration_us2, distance_cm2;

// variables for sensor 3
float duration_us3, distance_cm3;

void setup() {
  Serial.begin(9600);      // initialize serial port

  // Sensor 1 setup
  pinMode(TRIG_PIN1, OUTPUT);
  pinMode(ECHO_PIN1, INPUT);
  pinMode(LED_PIN1, OUTPUT);

  // Sensor 2 setup
  pinMode(TRIG_PIN2, OUTPUT);
  pinMode(ECHO_PIN2, INPUT);
  pinMode(LED_PIN2, OUTPUT);

  // Sensor 3 setup
  pinMode(TRIG_PIN3, OUTPUT);
  pinMode(ECHO_PIN3, INPUT);
  pinMode(LED_PIN3, OUTPUT);
}

void loop() {
  // Sensor 1 measurements
  digitalWrite(TRIG_PIN1, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN1, LOW);
```

```
 duration_us1 = pulseIn(ECHO_PIN1, HIGH);
 distance_cm1 = 0.017 * duration_us1;

 // Sensor 2 measurements
 digitalWrite(TRIG_PIN2, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN2, LOW);
 duration_us2 = pulseIn(ECHO_PIN2, HIGH);
 distance_cm2 = 0.017 * duration_us2;

 // Sensor 3 measurements
 digitalWrite(TRIG_PIN3, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN3, LOW);
 duration_us3 = pulseIn(ECHO_PIN3, HIGH);
 distance_cm3 = 0.017 * duration_us3;

 // Control LED 1 based on sensor 1
 if (distance_cm1 < DISTANCE_THRESHOLD) {
   digitalWrite(LED_PIN1, HIGH);
 } else {
   digitalWrite(LED_PIN1, LOW);
 }

 // Control LED 2 based on sensor 2
 if (distance_cm2 < DISTANCE_THRESHOLD) {
   digitalWrite(LED_PIN2, HIGH);
 } else {
   digitalWrite(LED_PIN2, LOW);
```

```
 }

 // Control LED 3 based on sensor 3
 if (distance_cm3 < DISTANCE_THRESHOLD) {
   digitalWrite(LED_PIN3, HIGH);
 } else {
   digitalWrite(LED_PIN3, LOW);
 }

 // Print values to Serial Monitor
 Serial.print("Sensor 1 distance: ");
 Serial.print(distance_cm1);
 Serial.println(" cm");

 Serial.print("Sensor 2 distance: ");
 Serial.print(distance_cm2);
 Serial.println(" cm");

 Serial.print("Sensor 3 distance: ");
 Serial.print(distance_cm3);
 Serial.println(" cm");

 // Delay for a short time before repeating the measurement
 delay(100); // Adjust this delay as needed
}
```

Step 4: Testing

Upload the code to your ESP32 and monitor the serial output. It will display whether the parking space is occupied or vacant based on the distance measured by the HC-SR04 sensor.

Step 5: Integration and Scaling

To create a complete smart parking system, you can:

1. Add more HC-SR04 sensors for multiple parking spaces.

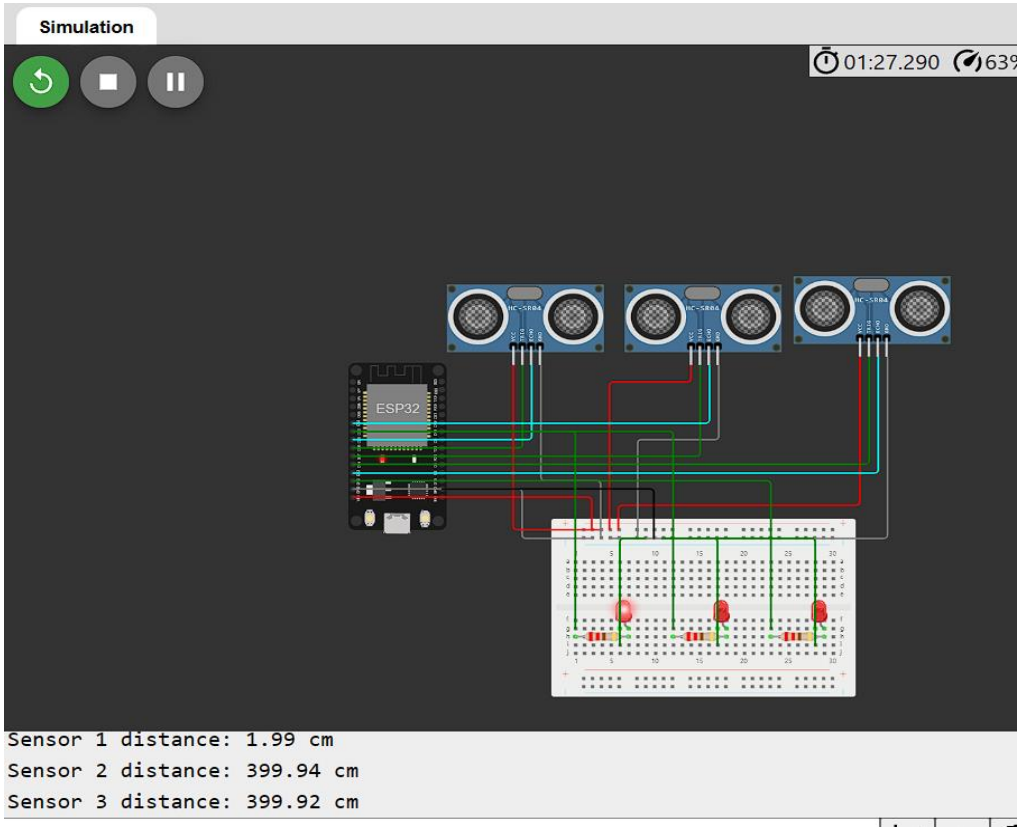
2. Use Wi-Fi or Bluetooth to send parking space status to a central server or app.

3. Create a user interface to display available parking spaces in real-time.

4. Implement notifications for users when a space becomes available.

5. Add a database to keep track of parking space occupancy history.

Circuit Diagram:



APPLICATIONS:

1. Public Parking Lots: Smart parking systems in public parking lots can help drivers quickly find available parking spaces. LED lights above each parking space can indicate whether it's vacant or occupied, making it easier for drivers to locate a spot.

2. Indoor Parking Garages: Indoor parking garages in shopping malls, airports, and office buildings can use ultrasonic sensors and LED lights to guide drivers to available spaces. This can reduce congestion and improve the overall parking experience.

3. Disabled Parking Spaces: Dedicated parking spaces for individuals with disabilities can be equipped with ultrasonic sensors and LED lights to ensure that these spots are reserved exclusively for those who need them.