smart parking USING IOT

BATCH MEMBER

211121106305 : Banukumar S

PHASE 3 SUBMISSION DOCUMENT

PROJECT TITLE : smart parking using IOT.

PHASE 3 : Development Part 1

TOPIC : Creating a Smart Parking project using ESP32

Building the IoT-enabled to detect and manage parking space

Smart parking systems can employ a variety of sensors to efficiently manage parking spaces. These sensors can provide real-time data on parking space availability, occupancy, and even vehicle identification. Here's a general overview of the types of sensors commonly used for smart parking management:

1. *Ultrasonic Sensors (HC-SR04):* These sensors measure distance by emitting and receiving ultrasonic waves. They are commonly used to detect the presence of vehicles in parking spaces. When a vehicle enters or exits a space, the change in distance triggers occupancy data.

2. *Magnetic Sensors:* Magnetic sensors detect changes in the Earth's magnetic field caused by the presence of a vehicle. They are embedded in the ground and are suitable for detecting the occupancy of individual parking spaces.

3. *Infrared Sensors:* Infrared sensors can be used to detect the presence of a vehicle by measuring the heat emitted by the engine or the body of the vehicle. They are often placed at the entrance of parking spaces.

4. *Image and Video Cameras:* High-resolution cameras can capture images or video of parking areas, and computer vision algorithms can analyze the images to determine parking space occupancy. License plate recognition can also be implemented for identifying specific vehicles.

5. *Ground-Loop Sensors:* These are electromagnetic sensors embedded in the ground. When a vehicle passes over them, it disrupts the electromagnetic field, indicating an occupied space.

6. *Acoustic Sensors:* Acoustic sensors can detect sound or vibrations produced by vehicles. They are especially useful in underground or multi-level parking facilities.

7. *Lidar Sensors:* Lidar sensors use laser technology to create precise 3D maps of the environment. They can accurately detect the presence of vehicles and provide data for parking space management.

8. *Radar Sensors:* Radar sensors can detect moving objects, including vehicles. They are suitable for monitoring parking lot entrances and exits.

9. *Wireless Parking Sensors:* These sensors are placed on or near parking spaces and use wireless communication to relay data to a central management system. They are versatile and easy to install.

10. *Inductive Loop Sensors:* Inductive loops are embedded in the pavement and use electromagnetic fields to detect vehicles passing over them. They are often used at traffic lights and can be adapted for parking space management.

11. *Bluetooth and WiFi Sensors:* These sensors can detect the presence of smartphones or other devices in vehicles, allowing for the monitoring of parking space occupancy and user identification.

12. *Pressure Sensors:* Pressure-sensitive mats or sensors can be placed in parking spaces to detect the weight of vehicles, indicating whether the space is occupied.

13. *GPS and Geolocation Sensors:* These sensors use GPS technology to track vehicle locations and can provide data on available parking spaces in a larger area.

The choice of sensor depends on factors like cost, accuracy, environmental conditions, and the specific requirements of the smart parking system. Many systems combine multiple sensor types for redundancy and accuracy. Additionally, the integration of these sensors with a central management system and user interfaces is crucial for effective smart parking management.

ntegrating an IoT-enabled parking space management system using an ESP32 and HC-SR04 sensor into public spaces involves several steps. Here's an elaboration of the process:

For integrating sensors into public transportation vehicles, consider the following steps:

1. *Project Planning and Design:*

   - Define the scope and objectives of your IoT parking system.

   - Create a detailed system design, including the ESP32, HC-SR04 sensor, and any other components you plan to use.

   - Consider power sources, communication methods, and enclosure design for outdoor use.

2. *Regulatory Compliance:*

   - Ensure your device complies with local regulations and standards.

   - Obtain necessary permits and approvals for deploying IoT devices in public spaces.

3. *Hardware Assembly:*

   - Set up the ESP32 with the required peripherals and interfaces.

   - Connect the HC-SR04 sensor for accurate distance measurements.

   - Test the hardware in a controlled environment to ensure it functions correctly.

4. *Firmware Development:*

   - Write the firmware for the ESP32 to read data from the HC-SR04 sensor and communicate with your server or cloud platform.

   - Implement error handling and data storage capabilities to ensure reliability.

5. *Cloud Platform Setup:*

   - Choose a cloud platform to store and manage data from your devices.

   - Create a database to store parking space status, historical data, and user information.

   - Implement secure communication protocols to transmit data from the ESP32 to the cloud.

6. *User Interface Development:*

- Design a user-friendly web or mobile app to display parking space availability.

- Implement features for users to reserve or check available spaces.

- Ensure the interface is responsive and intuitive for the public.

7. *Power Management:*

- Design a power supply system that can operate the ESP32 and sensor continuously, considering potential solar or battery solutions for outdoor use.

8. *Network Connectivity:*

- Set up a reliable internet connection for the ESP32, which may include Wi-Fi or cellular networks.

- Ensure strong network coverage in the chosen public spaces.

9. *Testing and Calibration:*

- Calibrate the HC-SR04 sensor for accurate distance measurements.

- Test the entire system in real-world conditions to identify and resolve issues.

10. *Security Implementation:*

- Implement robust security measures to protect user data and the device from tampering or unauthorized access.

- Use encryption for data transmission and secure authentication methods.

11. *Data Analytics and Insights:*

- Analyze the data collected from the system to gain insights into parking space utilization.

- Use this information to optimize parking management and improve user experiences.

12. *Deployment:*

- Install the IoT devices in selected public spaces, ensuring they are weatherproof and securely mounted.

- Regularly monitor the devices to ensure they are functioning correctly.

13. *User Engagement and Support:*

- Promote your IoT parking system to the public.

- Provide user support and assistance, such as a helpline or online FAQ, to address any issues or inquiries.

14. *Maintenance and Updates:*

- Schedule regular maintenance to replace batteries, update firmware, and ensure device reliability.

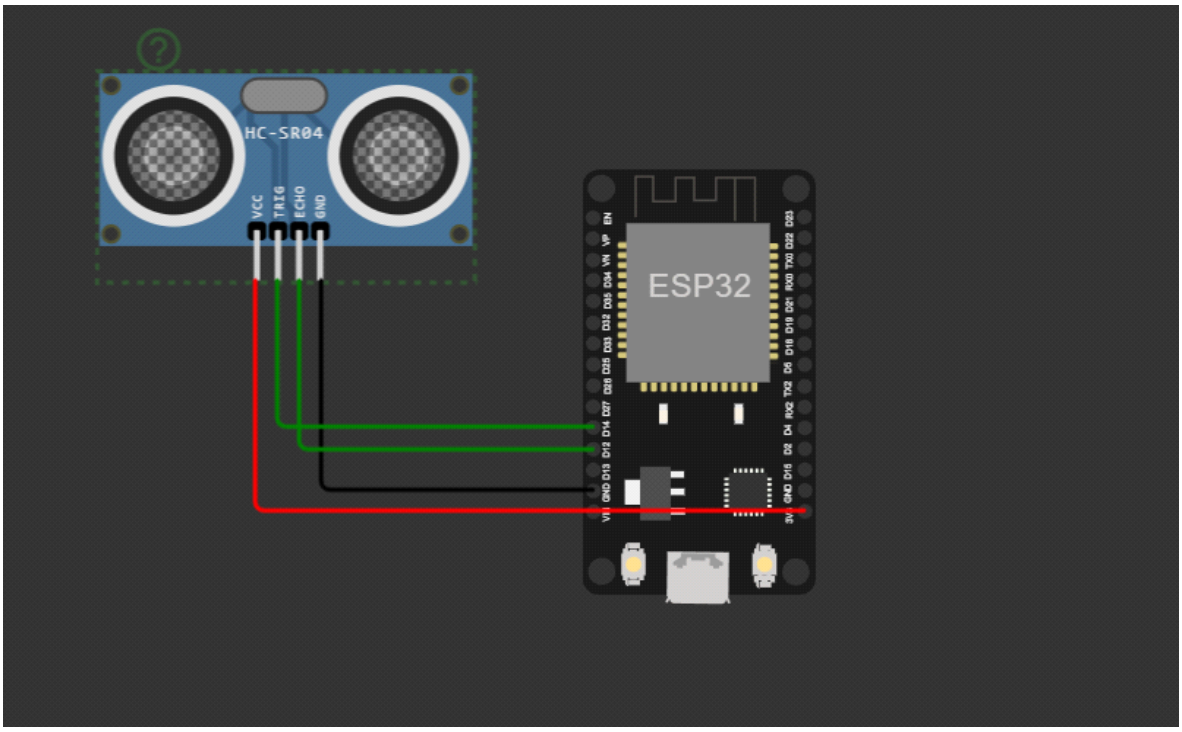
- Continuously improve the system based on user feedback and evolving needs.

15. *Scalability:*

- Plan for the scalability of your system to accommodate more public spaces as your project grows.

Remember that integrating IoT devices into public spaces comes with various challenges, including data privacy, security, and infrastructure requirements. It's important to collaborate with local authorities and stakeholders to ensure a smooth integration process and adherence to regulations.

STIMULATION:



CODING :

```
#include <Arduino.h>
int triggerPin = 5;
const int echoPin = 4;

void setup() {
  Serial.begin(115200);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

```
void loop() {
  long duration;
  float distance;

  // Send a 10us pulse to trigger the HC-SR04
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  // Read the echo pulse duration in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  // Check if a parking space is available
  if (distance > 5 && distance < 50) {
    Serial.println("Parking space occupied.");
  } else {
    Serial.println("Parking space vacant.");
  }

  delay(1000); // Wait for a moment before the next measurement
}
from gpiozero import DistanceSensorconst
from time import sleep

# Define GPIO pin numbers for trigger and echo pins
```

```
GPIO_TRIGGER1 = 17
GPIO_ECHO1 = 18
GPIO_TRIGGER2 = 23
GPIO_ECHO2 = 24

sensor1 = DistanceSensor(trigger=GPIO_TRIGGER1, echo=GPIO_ECHO1)
sensor2 = DistanceSensor(trigger=GPIO_TRIGGER2, echo=GPIO_ECHO2)
# Add more sensors if needed

def check_parking_spaces(distance1, distance2):
    # You can define your parking space logic here
    # For example, check if a car is within a certain distance threshold
    if distance1 < 0.2:
        print("Parking Space 1 is occupied")
    else:
        print("Parking Space 1 is available")

    if distance2 < 0.2:
        print("Parking Space 2 is occupied")
    else:
        print("Parking Space 2 is available")

while True:
    distance1 = sensor1.distance
    distance2 = sensor2.distance
    # Read distances from more sensors if needed

    # Process distance data and manage parking spaces here
    check_parking_spaces(distance1, distance2)
```

```
    sleep(1)  # Delay for better readability
```

.In conclusion, this smart parking system, utilizing the ESP32 and HC-SR04 sensor, showcases the potential of IoT technology to efficiently manage parking spaces, enhance user experiences, and contribute to more convenient and sustainable urban environments.