Smart Parking

TEAM MEMBERS

912221104018 - K.K.Hari Prakash

912221104016 - P.Gurudeep Swasanekar

912221104302 - J.Gnana Prakasha Martin

912221104006 - D.Aravind

912221104306 - P.Muneeswaran

Phase-3 Development Part 1

Subject: Smart Parking

Step 1:

Gather the Hardware and Software

Hardware:

- Raspberry Pi

- Ultrasonic sensors

- Jumper wires

- Breadboard

- Power source for Raspberry Pi

- Internet connection (Wi-Fi or Ethernet).

Software:

- Raspbian OS installed on your Raspberry Pi.
- Python 3 installed on the Raspberry Pi.
- Access to a cloud server or a mobile app with an API to receive data (e.g., RESTful API).

Step 2:

Connect Ultrasonic Sensors

Connect the ultrasonic sensors to the Raspberry Pi. The HC-SR04 sensor typically has four pins:

VCC (5V)

Trig (Trigger)

Echo

GND (Ground)

Connect VCC to a 5V pin on the Raspberry Pi, GND to a GND pin, Trig to a GPIO pin (e.g., GPIO17), and Echo to another GPIO pin (e.g., GPIO18). Use jumper wires and a breadboard if needed.

Step 3:

Install Required Libraries

Install any Python libraries required for your sensors. For ultrasonic sensors, you might need libraries such as RPi.GPIO for GPIO access.

```
pip install RPi.GPIO
```

Step 4:

Write Python Scripts

You'll need Python scripts to interface with the ultrasonic sensors and send data to the cloud or mobile app server. You can use libraries like RPi.GPIO for GPIO control and requests for making HTTP requests.

Program:

```
import RPi.GPIO as GPIO
import time
import requests
# Set GPIO pins
TRIG = 17
ECHO = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
# Function to measure distance
def measure_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
```

```
    pulse_start = time.time()
    pulse_end = time.time()
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()
    pulse_duration = pulse_end - pulse_start
    distance = (pulse_duration * 34300) / 2
    return distance
try:
    while True:
        distance = measure_distance()
        print(f"Distance: {distance} cm")
        # Send data to the cloud or mobile app server
        # Replace URL with your server's endpoint
        data = {'distance': distance}
        response = requests.post('https://your-api-endpoint.com/data', json=data)
        print(response.status_code)
        time.sleep(1)  # Adjust the interval as needed
except KeyboardInterrupt:
    GPIO.cleanup()
```

This script measures the distance using the ultrasonic sensor and sends it to a server using HTTP POST.

Step 5:

Cloud or Mobile App Integration

Set up your cloud or mobile app to receive data from the Raspberry Pi. You'll need to implement an API on your server to accept POST requests and process the data.

Step 6:

Test and Expand

Test your setup and ensure the data is being sent and received correctly. You can also expand on this project by adding more sensors to monitor multiple parking spaces and create a user-friendly mobile app or web interface to display the occupancy status of the parking lot.

Output:

Distance: 45.0 cm

200

Distance: 43.5 cm

200

Distance: 42.0 cm

200

Conclusion:

To create an IoT parking space occupancy system with a Raspberry Pi and ultrasonic sensors, gather hardware, connect sensors, write Python scripts for distance measurement, send data to a server, and test the system. Consider expanding its features for a comprehensive solution.