NAME	**M.ARUN**
DEPT	**ECE –III YEAR**
REG NO	**420121106007**
COLLEGE CODE	**4201**
GROUP	**IBM-GROUP 5**

Smart Parking

Phase 3: Development Part 1

Hardware Setup:

Connect the ultrasonic sensor to the Raspberry Pi. Ultrasonic sensors typically have 4 pins: VCC, GND, Trig (trigger), and Echo. Connect VCC and GND to the appropriate GPIO pins and Trig and Echo to two other GPIO pins on your Raspberry Pi.

2.Install Required Libraries:

Install the RPi.GPIO library for controlling GPIO pins and, if necessary, any other libraries related to your specific ultrasonic sensor. You can install RPi.GPIO with the following command:

bash Copy code

```
pip install RPi.GPIO
```

Collect Sensor Data:

Write a Python script to read data from the ultrasonic sensor. The script will measure the time it takes for a sound wave to bounce off an object and return. This will give you the distance to the object.

python Copy code

```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

TRIG = 23  # GPIO pin for the trigger
ECHO = 24  # GPIO pin for the echo

GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    pulse_start = time.time()
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    pulse_end = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
```

```
    pulse_end = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound = 343m/s, distance
    return distance


try:
    while True:
        dist = distance()
        print(f'Distance: {dist:.2f} cm')
        # Implement logic to send the distance data to the server or update
        time.sleep(1)  # Adjust the delay as needed

except KeyboardInterrupt:
    GPIO.cleanup()
```

<u>Send Data to the Server:</u>

Adapt the script to send data to your cloud server or mobile app server. You can use the same HTTP request method as in the previous example, or you can use a different protocol such as MQTT for real-time updates.

Server-Side Implementation:

Implement an endpoint on your server to receive the data. The server can analyze the distance data to determine parking space occupancy.

Security, Error Handling, Logging, and Automation:

Follow the same security, error handling, logging, and automation guidelines as mentioned in the previous example.

Data Analysis:

On the server side, implement logic to analyze the distance data to determine if a parking space is occupied or vacant. You can set a threshold distance value to make this determination.

Security:

Ensure your data transfer is secure. For example, use HTTPS for web requests and consider implementing authentication and authorization mechanisms.

Error Handling:

Implement proper error handling to account for network issues and other potential problems.

Logging and Monitoring:

Implement logging and monitoring to track the status of data collection and transmission.

Remember to adapt the script and server-side implementation to your specific requirements, including the type of ultrasonic sensor used and the data format expected by your cloud or mobile app server.