**Phase 3:Development Part 1**

**1. Setting Up IoT Sensors:**

**Ultrasonic Sensors:**

* Ultrasonic sensors can measure distance using sound waves. Set up the ultrasonic sensors at each parking space you want to monitor.
* Connect the sensors to the Raspberry Pi using GPIO pins. Ultrasonic sensors usually have 4 pins: VCC, GND, TRIG, and ECHO. Connect VCC to 5V, GND to GND, TRIG to a GPIO pin (like GPIO17), and ECHO to another GPIO pin (like GPIO18).

**2. Writing Python Scripts on Raspberry Pi:**

**Prerequisites:**

**Python Libraries:** You need libraries like RPi.GPIO for GPIO control and potentially libraries like requests for sending data to the cloud.

python code

import RPi.GPIO as GPIO

import time

import requests

# GPIO Setup

TRIG = 17

ECHO = 18

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)

# Function to measure distance from ultrasonic sensor

def measure\_distance():

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

pulse\_start, pulse\_end = 0, 0

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 \* 17150 # Speed of sound = 343 m/s

distance = round(distance, 2)

return distance

# Main function

def main():

try:

while True:

distance = measure\_distance()

print(f"Distance: {distance} cm")

# Send data to the cloud or mobile app server

# Example: Use HTTP POST request to send data

# Change the URL to your server endpoint

# requests. post('http://your-server-endpoint', data={'distance': distance})

time. sleep(2) # Wait for 2 seconds before taking the next measurement

except Keyboard Interrupt:

print("Measurement stopped by user")

GPIO.cleanup()

if \_\_name\_\_ == '\_\_main\_\_':

main()

In this script:

measure\_distance() function measures the distance using the ultrasonic sensor.

Data is collected and can be sent to your cloud server or mobile app server. Ensure to modify the URL in the requests.post line to your server's endpoint.

The script runs an infinite loop, taking measurements every 2 seconds. You can adjust this interval based on your requirements.

**3. Cloud or Mobile App Integration:**

* On your cloud server or mobile app, set up an endpoint to receive data from the Raspberry Pi. Process the incoming data and update the parking space occupancy status accordingly.
* Ensure that your server has proper security measures, like HTTPS and authentication, to protect the data transmission.

**4. Testing:**

* Run the Python script on your Raspberry Pi by executing python example.py in the terminal.
* Monitor the output to ensure the correct distance measurements are being obtained.
* Verify that data is being sent to your cloud server or mobile app and that the occupancy status is being updated correctly.

Depending on our project requirements, we might need to add more sensors, implement error handling, or optimize the code for better performance and reliability. Additionally, security considerations are crucial, especially if we are dealing with sensitive data.