**Phase 3 project**

**Project Title: SMART PARKING**

**Project ID:**  proj\_223739\_Team\_5

**College Code:** 6208

**College:** Gnanamani College of Technology

**Branch:** B.Tech-Information Technology

**Year:** IIIrd year

**Team Members & ID:**

ABISHEK R - (au620821205002)

SANJAY S - (au620821205047)

PRIYADHARSAN B -(au620821205303)

KARTHICKRAJA M -(au620821205025)

**SMART PARKING**

**Definition:**

An IoT-based parking system is a centralized management that enables drivers to search for and reserve a parking spot remotely through their smartphones. It offers a convenient arrangement for drivers to park their cars when they are looking to avoid potential traffic congestion.

The system’s hardware sensors detect available slots and communicate the information to the drivers in that area in real-time. IoT technology ensures that they do not have to worry about finding an available space again – allowing them to travel conveniently.

**PHASE 3**

Building an IoT sensor system to detect parking space occupancy using ultrasonic sensors and integrating it with a Raspberry Pi involves several steps. Here's a step-by-step guide:

**Hardware and Sensor Setup**

1. **Select IoT Hardware:**
   * Raspberry Pi (any model with available GPIO pins)
   * Ultrasonic sensors (e.g., HC-SR04)
2. **Wiring Ultrasonic Sensors:**
   * Connect the VCC (power) and GND (ground) pins of the ultrasonic sensor to the 5V and GND pins on the Raspberry Pi.
   * Connect the Echo pin of the sensor to a GPIO pin on the Raspberry Pi (e.g., GPIO 17).
   * Connect the Trigger pin of the sensor to another GPIO pin on the Raspberry Pi (e.g., GPIO 18).

**Raspberry Pi Python Script**

You'll need to write a Python script that reads data from the ultrasonic sensors and sends it to a cloud service or a mobile app server. Here's an example script using the GPIO Zero library and the **requests** library for sending data to a server:

from gpiozero import DistanceSensor

import requests

# Ultrasonic Sensor Configuration

sensor = DistanceSensor(echo=17, trigger=18, max\_distance=2) # Adjust max\_distance based on your setup

# Cloud/Mobile App Server Configuration

server\_url = 'https://your-server.com/api/parking'

headers = {'Content-Type': 'application/json'}

while True:

try:

# Read distance from the ultrasonic sensor

distance = sensor.distance

# Convert distance to occupancy status (you may need to calibrate this)

if distance < 0.2: # Adjust this threshold based on your parking space dimensions

occupancy\_status = 'Occupied'

else:

occupancy\_status = 'Vacant'

# Prepare data for sending to the server

data = {'parking\_space\_id': 'A101', 'occupancy\_status': occupancy\_status}

# Send data to the server

response = requests.post(server\_url, json=data, headers=headers)

if response.status\_code == 200:

print("Data sent successfully.")

else:

print("Failed to send data. Status code:", response.status\_code)

except Exception as e:

print("An error occurred:", e)

# Set an appropriate polling interval

time.sleep(5) # Adjust the interval as needed

**Server and Data Handling**

You need a server to receive and process the data sent by the Raspberry Pi. This server should expose an API endpoint to accept parking space occupancy data. Depending on your needs, you can implement the server using various technologies such as Flask, Django, Node.js, or any other web framework.

Ensure that your server handles the incoming data securely, stores it, and potentially provides an interface for accessing the occupancy data via a mobile app or web application.

Remember to handle security considerations, such as authentication and encryption, for both the Raspberry Pi and your server, to protect the data.