**SMART PARKING**

**Hardware Setup:**

1. **Raspberry Pi Setup**:
   * Get a Raspberry Pi board (e.g., Raspberry Pi 4).
   * Install an operating system (e.g., Raspberry Pi OS) on the Pi.
   * Make sure your Pi is connected to the internet, either via Wi-Fi or Ethernet.
2. **Ultrasonic Sensors**:
   * Connect your ultrasonic sensors to the Raspberry Pi using GPIO pins.
   * Ensure you have a ground (GND), echo, and trigger pin connection for each sensor.
3. **Power Supply**:
   * Power the Raspberry Pi and sensors using an appropriate power supply.

**Software Setup:**

1. **Raspberry Pi Python Code**:
   * You will need the **RPi.GPIO** library for controlling GPIO pins on the Raspberry Pi. Install it if it's not already installed.

**CODE**

pip install RPi.GPIO

**PYTHON SCRIPT:**

import RPi.GPIO as GPIO

import time

import requests

# Configure GPIO pins

TRIG\_PIN = 18

ECHO\_PIN = 24

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

def get\_distance():

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = (pulse\_duration \* 34300) / 2 # Speed of sound = 343 m/s

return distance

try:

while True:

distance = get\_distance()

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

# Send data to a cloud server or mobile app server

# Replace 'YOUR\_SERVER\_URL' with the actual server URL

server\_url = 'YOUR\_SERVER\_URL'

data = {'distance': distance}

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

time.sleep(1) # Adjust the sampling interval as needed

except KeyboardInterrupt:

GPIO.cleanup()

1. **Cloud or Mobile App Server**:
   * Set up a cloud server or a mobile app server to receive and store the data from the Raspberry Pi. You can use platforms like AWS, Google Cloud, or a self-hosted server.
2. **Data Storage**:
   * Decide how you want to store and analyze the data. You can use databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB.
3. **Mobile App** :
   * If you want to build a mobile app to display parking space occupancy, you'll need to develop the app and configure it to communicate with the server.
4. **Notifications** :
   * Implement notifications to inform users of parking space availability in real-time.
5. **Security**:
   * Ensure the security of your IoT system, especially when handling data transmission and storage.
6. **Testing**:

* Thoroughly test your setup to ensure it works as expected.