SMART PARKING

Hardware Setup:

1. Connect the ultrasonic sensor to the Raspberry Pi using GPIO pins. Typically, an ultrasonic sensor has four pins: VCC, Trig, Echo, and GND.

- Connect VCC to 5V on the Raspberry Pi.
- Connect Trig to a GPIO pin (e.g., GPIO17).
- Connect Echo to another GPIO pin (e.g., GPIO18).
- Connect GND to ground (GND) on the Raspberry Pi.

Python Script to Collect Sensor Data:

You can use Python to interface with the ultrasonic sensor and collect data. Here's a basic script to get you started:

CODE:

```
import RPi.GPIO as GPIO

import time

# Set GPIO mode and pins

GPIO.setmode(GPIO.BCM)

TRIG = 17

ECHO = 18

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)


def get_distance():

    # Trigger the ultrasonic sensor

    GPIO.output(TRIG, True)

    time.sleep(0.00001)


 GPIO.output(TRIG, False)

  # Record the start and end times
```

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

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

    # Calculate distance
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound (343 m/s) / 2
    distance = round(distance, 2)

    return distance
try:
    while True:
        distance = get_distance()
        print(f"Distance: {distance} cm")
        # Here, you can send 'distance' to the cloud or a mobile app server.
        time.sleep(1)

except KeyboardInterrupt:
    GPIO.cleanup()
```

This script continuously measures the distance using the ultrasonic sensor and prints the results. You can modify it to send the 'distance' data to your cloud platform.

Sending Data to the Cloud:

To send data to the cloud or a mobile app server, you'll need to set up an API endpoint on your server to receive and process the data. You can use libraries like `requests` in Python to make HTTP POST requests to your API.

In your script, replace the comment with code to send data to your server. Here's a simplified example:

CODE:

```
import requests

server_url = "http://your-server-endpoint.com/data"


# Inside the loop

data = {"distance": distance}

response = requests.post(server_url, json=data)
```

INSTALLATION CODE:

Step 1: Install Required Libraries

Open a terminal on your Raspberry Pi and make sure your system is up to date:

sudo apt-get update

sudo apt-get upgrade

Install the RPi.GPIO library for GPIO control and the requests library for making HTTP requests:

pip install RPi.GPIO

pip install requests

Step 2: Write the Python Script

Create a Python script with the ultrasonic sensor data collection code. You can use the Nano text editor to create the script:

nano parking_sensor.py

Paste the Python script from my previous response into this file.

Step 3: Save and Exit

To save the file in Nano, press `Ctrl + O`, confirm the filename, and press `Enter`. Then, exit Nano by pressing `Ctrl + X`.

Step 4: Run the Python Script

To run the Python script, use the following command:

python parking_sensor.py

This will start the script, which will continuously measure the distance using the ultrasonic sensor and print the results to the terminal.

CONCLUSION:

In conclusion, the IoT-based smart parking project offers real-time parking availability data, reducing traffic congestion, and improving urban mobility. It optimizes parking space utilization, increasing revenue potential for parking operators. User-friendly mobile access and secure data handling enhance the parking experience. With streamlined payment options, the system benefits users and operators alike. As cities grow, this technology is crucial for more efficient, sustainable, and livable urban environments, transforming the way we approach parking management.