IOT PROJECT—PHASE 3

PROJECT NAME—SMART PARKING SYSTEM

Start building the IoT sensor system and Raspberry Pi integration.

Configure IoT sensors (e.g., ultrasonic sensors) to detect parking space occupancy.

Write Python scripts on Raspberry Pi to collect data from sensors and send it to the cloud or mobile app server.

ChatGPT

Building an IoT sensor system with Raspberry Pi integration to detect parking space occupancy involves several steps:

Hardware Setup:

Connect ultrasonic sensors to the Raspberry Pi as per their datasheets.

Ensure power supply and grounding are appropriate.

Raspberry Pi Configuration:

Install a suitable operating system on your Raspberry Pi.

Set up an internet connection, either via Ethernet or Wi-Fi.

Python Script for Sensor Data Collection:

Write Python scripts to interface with the ultrasonic sensors.

Use libraries like RPi.GPIO to manage GPIO pins.

Collect data from the sensors, which may include distance measurements.

Data Processing:

Process the data as needed, like filtering out noise.

Convert distance measurements into occupancy information.

Cloud or Server Integration:

Set up your Raspberry Pi to send data to a cloud service or a mobile app server.

You can use libraries or SDKs for cloud providers like AWS IoT, Google Cloud IoT, or Azure IoT.

Ensure secure communication, possibly using MQTT or HTTPS.

Data Storage and Management:

Store the collected data in a database or cloud storage for historical analysis.

Implement mechanisms to manage and clean up the data periodically.

Mobile App Development (if needed):

If you intend to create a mobile app, start developing it.

Use appropriate programming languages (e.g., Java or Swift) and frameworks.

Configure the app to receive and display parking occupancy information.

Testing and Debugging:

Rigorously test the entire system to ensure data accuracy and reliability.

Debug any issues that arise during testing.

Scalability and Maintenance:

Plan for scalability if you intend to expand the system to cover more parking spaces.

Establish maintenance procedures and monitoring for long-term stability.

Remember to consult the documentation and resources for the specific sensors and cloud services you are using, as they may have specific requirements and recommendations

Setup Hardware:

Connect your sensors to the Raspberry Pi. This may require installing appropriate libraries or drivers for your specific sensors.

Install Required Libraries:

Install any necessary Python libraries for working with sensors. For example, you might use libraries like Adafruit\_Python\_DHT for DHT sensors, RPi.GPIO for GPIO pins, or smbus for I2C sensors. You can install these using pip:

Copy code

Pip install Adafruit\_Python\_DHT RPi.GPIO smbus

Collect Sensor Data:

Write Python code to read data from the sensors. For example, if you have a DHT22 temperature and humidity sensor, you can use the Adafruit\_Python\_DHT library to get the data.

Choose a Cloud Service:

Select a cloud service to store and manage your data. Options include AWS, Azure, Google Cloud, or platforms like IoT Core.

Send Data to the Cloud:

Use Python to send the collected sensor data to your chosen cloud service. For example, you can use MQTT, HTTP requests, or a specific API provided by your cloud service.

Set Up a Mobile App Server:

If you want to display the data in a mobile app, you’ll need a server to host the app’s backend. You can use frameworks like Flask or Django to create a server.

Create an API:

Develop an API on your server that the mobile app can call to fetch the sensor data.

Send Data to Mobile App Server:

Use Python on the Raspberry Pi to send the sensor data to your mobile app server using HTTP requests or another suitable method.

Here’s a basic code template for collecting data from a DHT22 sensor and sending it to a hypothetical cloud server:

Python

Copy code

Import Adafruit\_DHT

Import requests

# Sensor setup

Sensor = Adafruit\_DHT.DHT22

Pin = 4 # GPIO pin where the sensor is connected

# Cloud server endpoint

Server\_url = ‘https://your-cloud-server.com/upload’

While True:

Humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

If humidity is not None and temperature is not None:

# Send data to the cloud server

Data = {‘temperature’: temperature, ‘humidity’: humidity}

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

If response.status\_code == 200:

Print(‘Data sent to the cloud server successfully.’)

Else:

Print(‘Failed to send data to the cloud server.’)

Please note that the actual implementation will depend on your specific sensors, cloud service, and mobile app server. Additionally, you should consider security and error handling for real-world applications.

PROJECT BY

PUNITHKUMAR. BA

21EC131

923321106031

ECE 3RD YEAR