SMART PARKING SYSTEM

Processing and Data

1. **Hardware Setup:**

   - Acquire IoT devices like ultrasonic sensors or cameras for detecting parking spaces.

   - Set up a Raspberry Pi or Arduino board to act as the IoT device to collect data.

   - Establish a network connection for the IoT device to send data.

2. **Data Collection:**

   - Use the IoT devices to collect data on parking space occupancy.

   - Ultrasonic sensors can detect the presence of a vehicle in a parking space, and cameras can capture images for further analysis.

3. **Data Processing:**

   - Develop a Python script to process the data from the IoT devices.

   - Analyze the sensor data or images to determine parking space availability.

4. **Data Communication:**

   - Set up a communication protocol to send the parking space data to a central server or a cloud platform.

   - Use technologies like MQTT or HTTP for data transmission.

5. **User Interface:**

   - Create a user interface, possibly a web or mobile app, to display real-time parking space availability.

   - Users can check the app to find vacant parking spots.

6. **Document and Assessment:**

   - Document the project comprehensively, including hardware setup, data processing, and user interface.

   - Explain the Python script and the technologies used in detail.

   - Share the documentation with your instructor or assessors for evaluation.

Sensors

1. **Ultrasonic Sensors:**

- Ultrasonic sensors are commonly used to measure the distance between the sensor and an object. They can be placed at each parking space to detect the presence of a vehicle. When a vehicle enters or leaves a parking space, changes in distance are detected, indicating occupancy.

2. **Infrared Sensors:**

- Infrared (IR) sensors can detect the presence of a vehicle by emitting and receiving infrared light. These sensors can be positioned to cover parking spaces and trigger when a vehicle obstructs the light path.

3. **Magnetic Sensors:**

- Magnetic sensors can be embedded in the ground at each parking space. They detect changes in the Earth's magnetic field caused by the presence of a vehicle. When a vehicle parks or leaves, the magnetic field changes, indicating occupancy.

4. **Camera Sensors:**

- IP cameras or CCTV cameras can capture images of parking spaces. These images can be processed using computer vision techniques to determine the availability of parking spaces. Cameras provide visual data for more accurate detection.

5. **LIDAR Sensors:**

- LIDAR (Light Detection and Ranging) sensors use laser beams to measure distances and create detailed 3D maps of the environment. They are used in advanced smart parking systems to detect not only vehicle presence but also the size and type of vehicles.

Payment and Reservation

Implementing parking reservation and payment features in Smart Parking IoT project can greatly enhance its functionality. Here's how to integrate these features:

1. **Reservation System:**

- Develop a reservation system that allows users to book parking spaces in advance. This can be part of your user interface, whether it's a mobile app or a web application.

- Users can select the desired parking spot, date, and time for their reservation.

- Implement a database to store reservation information, including user details and reservation times.

2. **Real-time Availability Updates:**

- Ensure that the parking space availability information is updated in real time. Users should see which spaces are available for immediate use and which are reserved.

3. **Payment Integration:**

- Integrate a payment gateway into your app or web interface to process parking fees.

- Allow users to make payments securely using credit cards, mobile wallets, or other payment methods.

- Calculate parking fees based on reservation time or actual parking duration.

4. **QR Code or RFID Access:**

- Implement a system for access control to the parking space. This can involve QR codes or RFID cards given to users upon reservation.

- Users can scan their codes at entry and exit points to access the parking area.

5. **Notifications:**

- Send confirmation emails or notifications to users upon successful reservation and payment.

- Notify users when their reservation time is about to expire.

6. **Cancellations and Refunds:**

- Define a policy for reservation cancellations and refunds. Allow users to cancel reservations within a certain time frame and provide refunds accordingly.

7. **Security:**

- Implement strong security measures to protect user data and payment information.

- Use encryption and follow best practices for securing payment transactions.

8. **Documentation and Support:**

- Document the reservation and payment processes clearly for users.

- Offer customer support for any issues or inquiries related to reservations and payments.

Python script for this Project:

```
Import serial
Import requests
Import time

# Set your ThingSpeak API key and ThingSpeak channel URL
Api_key = “your_api_key”
Thing_speak_url = https://api.thingspeak.com/update

# Set the COM port for your Arduino Uno
Arduino_port = ‘COM3’

# Initialize serial communication with Arduino Uno
Try:
   Arduino = serial.Serial(arduino_port, 9600)
Except serial.SerialException:
   Print(f”Failed to connect to {arduino_port}. Check your COM port and connections.”)
   Exit()

Def read_sensor_data():
   Try:
      # Read data from the Arduino via serial communication
      Arduino.write(b’R’)  # Signal Arduino to send sensor data
      Sensor_data = arduino.readline().decode().strip()
      Return sensor_data
   Except Exception as e:
      Print(f”Failed to read sensor data: {str€}”)
      Return None
```

```
Def send_to_thingspeak(data):
    Payload = {“api_key”: api_key, “field1”: data}
    Try:
        Response = requests.post(thing_speak_url, params=payload)
        If response.status_code == 200:
            Print(“Data sent to ThingSpeak successfully”)
        Else:
            Print(“Failed to send data to ThingSpeak”)
    Except Exception as e:
        Print(f”Error sending data to ThingSpeak: {str€}”)


If __name__ == “__main__”:
    Try:
        While True:
            Sensor_data = read_sensor_data()
            If sensor_data is not None:
                Print(“Sensor Data:”, sensor_data)
                Send_to_thingspeak(sensor_data)
                Time.sleep(60)  # Adjust the interval as needed
    Except KeyboardInterrupt:
        Print(“Exiting the program.”)
        Arduino.close()
```