COLLEGE CODE: 5113

Batch Members:

1. HEMACHANDAR S K – 511321104029 - hemachandar7549@gmail.com
2. BHARATH RAJ R – 511321104011 – bharathraj0247@gmail.com
3. MANNU A Z – 511321104052 – mannu12a2@gmail.com
4. DHANUSH KUMAR K – 511321104022 - dhanushlifestyle007@gmail.com

INTERNET OF THINGS

Project No.7 - Smart Public Restrooms

Building the iot enabled smart restroom system:

Creating an IoT-enabled smart restroom system is a complex project that involves various hardware and software components. Below, I'll outline the basic steps and components you might need to build such a system. Please note that this is a high-level overview, and specific details may vary depending on your requirements.

Hardware Components:

1. **Sensors**:
   - Motion sensors for presence detection.
   - Proximity sensors for door lock control.
   - Environmental sensors for measuring cleanliness (e.g., air quality sensors).
   - Water usage sensors for monitoring water consumption.
   - Occupancy sensors for counting the number of people inside.
2. **Microcontrollers**:
   - Use microcontrollers (e.g., Arduino, Raspberry Pi) to interface with sensors, control actuators, and process data.
3. **Actuators**:
   - Locking mechanism for the restroom door.
   - Automated faucets, soap dispensers, and hand dryers.
   - Lighting control for energy efficiency.
   - Alerting mechanisms (e.g., LED displays or speakers) for providing information to users.
4. **Networking**:
   - Connect microcontrollers to the internet via Wi-Fi, Ethernet, or other suitable network interfaces.

Software Components:

1. **IoT Platform**:
   - Choose an IoT platform (e.g., AWS IoT, Azure IoT, or Google Cloud IoT) to manage devices, collect data, and enable remote monitoring and control.
2. **Firmware**:
   - Write firmware for microcontrollers to read sensor data, control actuators, and communicate with the IoT platform.
3. **Cloud Services**:
   - Implement cloud-based services to store and analyze data from the sensors.
   - Create a dashboard for remote monitoring and management.
4. **Mobile App/Web Interface**:
   - Develop a user interface for restroom visitors to access information and provide feedback (e.g., an app or a web portal).
5. **Data Analytics**:
   - Implement data analytics to gain insights from sensor data, such as restroom occupancy patterns, cleanliness, and resource usage.

Security and Privacy:

- Ensure data privacy and security by implementing appropriate encryption, authentication, and access control mechanisms.

Power Management:

- Implement power-saving mechanisms to optimize energy usage for battery-operated devices.

Integration:

- Integrate the smart restroom system with external services, such as maintenance systems for cleaning alerts or emergency services for safety concerns.

Compliance:

- Ensure that the system complies with relevant regulations and standards, such as ADA (Americans with Disabilities Act) for accessibility.

Testing and Maintenance:

- Thoroughly test the system and implement maintenance procedures to keep it running smoothly.

Remember that building a fully functional IoT-enabled smart restroom system is a significant undertaking, and you might require expertise in electronics, software development, networking, and data analytics. Additionally, you'll need to consider the specific needs and regulations applicable to your location or industry. It's often beneficial to work with a team of experts to ensure a successful implementation.

Deploying iot sensors in public restroom to collect data:

Deploying IoT sensors in public restrooms to collect data requires careful planning and consideration of several aspects. Here's a step-by-step guide to help you deploy IoT sensors in public restrooms:

1. **Define Objectives**:
   - Clearly define the goals of deploying IoT sensors. Are you collecting data for occupancy, cleanliness, or other purposes?
2. **Choose Sensor Types**:
   - Select appropriate sensors based on your objectives. For example, use occupancy sensors (motion or infrared sensors), cleanliness sensors (humidity or odor sensors), and other relevant sensors.
3. **Hardware Selection**:
   - Choose suitable hardware platforms for your sensors. Raspberry Pi, Arduino, or specialized IoT devices can be used.
4. **Data Communication**:
   - Determine the communication protocol and network connectivity. Common options include Wi-Fi, LoRa, Zigbee, or cellular networks.
5. **Power Supply**:
   - Ensure a reliable power supply for the sensors. This might involve batteries, power over Ethernet (PoE), or other power sources.

6.Sensor Placement:

- Strategically place the sensors within the restroom to ensure accurate data collection. For example, install occupancy sensors near entrance/exits and cleanliness sensors near trash cans or sinks.

6. **Data Processing**:
   - Decide whether data processing will be performed on the sensors or on a central server. This depends on the complexity of the analysis.
7. **Data Storage**:
   - Set up a data storage solution to store the collected data. Options include local storage, cloud storage, or a combination of both.
8. **Data Security**:
   - Implement robust security measures to protect sensor data, including encryption, authentication, and access control.
9. **Software Development**:
   - Develop the software for your IoT sensors. This includes code for data collection, data transmission, and data analysis.
10. **User Interface**:
    - Create a user interface for monitoring and managing the data. This can be a web dashboard or a mobile app for administrators.
11. **Testing and Calibration**:
    - Test your sensors in a real-world restroom environment and calibrate them as needed for accurate data collection.
12. **Integration with Restroom Information Platform**:
    - Ensure that your IoT sensors can communicate with your restroom information platform through an API or other data transfer methods.
13. **Data Visualization**:
    - Create meaningful data visualizations and reports to make the collected data useful and actionable.
14. **Maintenance Plan**:
    - Establish a maintenance plan for sensor upkeep, battery replacement, and software updates.
15. **Compliance and Privacy**:
    - Be aware of data privacy regulations and ensure compliance with local laws, especially when collecting data in public spaces.
16. **Scalability**:
    - Design your system to be scalable so that you can easily add more sensors or expand to other locations.
17. **Monitoring and Alerts**:
    - Implement monitoring and alerting systems to notify you of any issues with the sensors or data collection.
18. **User Education**:
    - If applicable, educate restroom users about the presence of IoT sensors and how their data is being used.
19. **Data Analysis**:
    - Use the collected data to gain insights, optimize restroom operations, and make data-driven decisions.

python script on the iot sensor to send real time occupancy and cleanliness data to the restroom information platform

Creating a Python script for an IoT sensor to send real-time occupancy and cleanliness data to a restroom information platform typically involves using sensors and a communication protocol (e.g., MQTT) to transmit data. Below is a simple example using a Raspberry Pi and a PIR motion sensor for occupancy and a digital sensor for cleanliness. This script sends data to a hypothetical MQTT broker.

Here's a Python script for the IoT sensor:

```
import time
import requests
import RPi.GPIO as GPIO  # If using Raspberry Pi and GPIO

# Define GPIO pins for sensors
MOTION_SENSOR_PIN = 17  # Replace with the actual pin number
CLEANLINESS_SENSOR_PIN = 18  # Replace with the actual pin number

# URL of the restroom information platform
PLATFORM_URL = "http://your-restroom-platform.com/api/data"

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(MOTION_SENSOR_PIN, GPIO.IN)
GPIO.setup(CLEANLINESS_SENSOR_PIN, GPIO.IN)
```

```
        # Send data to the platform
        send_data_to_platform(occupancy, cleanliness)

        # Update every 5 seconds (adjust as needed)
        time.sleep(5)
except KeyboardInterrupt:
    print("Script terminated by user.")
finally:
    GPIO.cleanup()  # Cleanup GPIO on script exit
```

Output:

```
Data sent successfully.
Data sent successfully.
Data sent successfully.
Data sent successfully.
Data sent successfully.
...
```

In this script:

1. We import the necessary libraries (RPi.GPIO for GPIO control, time for timing, json for creating JSON payloads, and paho-mqtt for MQTT communication).
2. Define the sensor pins, MQTT broker settings, and the MQTT topic where the data will be published.
3. Initialize the GPIO pins and MQTT client.
4. Define functions to read the occupancy and cleanliness status from the sensors.
5. In the main loop, we continuously read sensor data and create a JSON payload with the occupancy and cleanliness status.
6. The data is published to the MQTT broker, and a message is printed to the console.
7. The loop repeats with a delay, and you can adjust the delay to control how often data is sent.