***Building an IoT-based smart public restroom project involves several steps, including hardware setup, sensor integration, and Python script development. Here's a comprehensive guide to implementing this project:***

***Hardware Setup:***

**a. **Microcontroller:** Select Arduino to act as the central processing unit for the IoT system.**

**b. **Sensors:** Choose appropriate sensors based on the desired functionalities,s: -**

* ****Occupancy Sensor:** Detects restroom occupancy to indicate availability.**

***Deployment:***

import time

import board

import busio

import digitalio

# Define GPIO pin for occupancy sensor

occupancy\_pin = digitalio.DigitalInOut(board.D2)

occupancy\_pin.direction = digitalio.Direction.INPUT

while True:

# Read occupancy sensor status

occupancy\_status = occupancy\_pin.value

if occupancy\_status:

print("Occupancy detected")

else:

print("No occupancy detected")

# Delay between readings

time.sleep(1)

* *****Air Quality Sensor:*** Monitors air quality and triggers alerts for unpleasant odors. -**

**Deployment:**

import time

import board

import busio

import adafruit\_ccs811

# Initialize I2C communication

i2c = busio.I2C(board.SCL, board.SDA)

# Create CCS811 air quality sensor object

ccs811 = adafruit\_ccs811.CCS811(i2c)

while True:

# Check if sensor is ready

if not ccs811.data\_ready:

continue

# Read air quality data

eco2 = ccs811.eco2

tvoc = ccs811.tvoc

print("eCO2:", eco2, "ppm")

print("TVOC:", tvoc, "ppb")

# Delay between readings

time.sleep(1)

* *****Water Level Sensor*:** Tracks water levels in tanks and sends notifications when refills are needed. -**

**Deployment:**

import time

import board

import busio

import adafruit\_vl53l0x

# Initialize I2C communication

i2c = busio.I2C(board.SCL, board.SDA)

# Create VL53L0X ultrasonic sensor object

vl53l0x = adafruit\_vl53l0x.VL53L0X(i2c)

while True:

# Read distance in millimeters

distance = vl53l0x.range

# Convert distance to water level percentage

water\_level = (1 - distance / max\_distance) \* 100

print("Water level:", water\_level, "%")

# Delay between readings

time.sleep(1)

* ***Soap Dispenser Sensor:***Monitors soap usage and sends replenishment alerts.

**Deployment:**

import time

import board

import busio

import adafruit\_hx711

# Define GPIO pins for HX711 load cell amplifier

hx711\_data\_pin = board.D2

hx711\_clock\_pin = board.D3

# Create HX711 object

hx711 = adafruit\_hx711.HX711(hx711\_data\_pin, hx711\_clock\_pin)

# Calibrate the sensor with known weight

hx711.set\_reference\_unit(calibration\_factor)

# Initialize soap level

soap\_level = 100

while True:

# Read weight in grams

weight = hx711.get\_weight(5)

# Calculate soap level based on weight change

soap\_level = soap\_level - (weight / initial\_weight) \* 100

# Check if soap level is below threshold

if soap\_level < threshold:

print("Soap level low, refill required")

# Delay between readings

time.sleep(1)

***Sensor Integration:***

1. ****Connect Sensors:** Connect the selected sensors to the microcontroller board using appropriate wiring and communication protocols (e.g., I2C, SPI).**

import time

import board

import busio

import digitalio

import adafruit\_ccs811

import adafruit\_vl53l0x

import adafruit\_hx711

# Define GPIO pin for occupancy sensor

occupancy\_pin = digitalio.DigitalInOut(board.D2)

occupancy\_pin.direction = digitalio.Direction.INPUT

# Initialize I2C communication

i2c = busio.I2C(board.SCL, board.SDA)

# Create CCS811 air quality sensor object

ccs811 = adafruit\_ccs811.CCS811(i2c)

# Create VL53L0X ultrasonic sensor object

vl53l0x = adafruit\_vl53l0x.VL53L0X(i2c)

# Define GPIO pins for HX711 load cell amplifier

hx711\_data\_pin = board.D3

hx711\_clock\_pin = board.D4

# Create HX711 object

hx711 = adafruit\_hx711.HX711(hx711\_data\_pin, hx711\_clock\_pin)

# Calibrate the sensor with known weight

hx711.set\_reference\_unit(calibration\_factor)

while True:

# Read occupancy sensor status

occupancy\_status = occupancy\_pin.value

if occupancy\_status:

print("Occupancy detected")

else:

print("No occupancy detected")

# Check if air quality sensor is ready

if not ccs811.data\_ready:

continue

# Read air quality data

eco2 = ccs811.eco2

tvoc = ccs811.tvoc

print("eCO2:", eco2, "ppm")

print("TVOC:", tvoc, "ppb")

# Read distance from ultrasonic sensor

distance = vl53l0x.range

# Convert distance to water level percentage

water\_level = (1 - distance / max\_distance) \* 100

print("Water level:", water\_level, "%")

# Read weight from load cell sensor

weight = hx711.get\_weight(5)

# Calculate soap level based on weight change

soap\_level = soap\_level - (weight / initial\_weight) \* 100

print("Soap level:", soap\_level, "%")

# Delay between readings

time.sleep(1)

1. ****Calibrate Sensors:** Calibrate each sensor to ensure accurate readings and set appropriate thresholds for triggering alerts or actions.**

import time

import board

import busio

import digitalio

import adafruit\_hx711

# Define GPIO pins for HX711 load cell amplifier

hx711\_data\_pin = board.D3

hx711\_clock\_pin = board.D4

# Create HX711 object

hx711 = adafruit\_hx711.HX711(hx711\_data\_pin, hx711\_clock\_pin)

# Define known weight for calibration

known\_weight = 100 # grams

# Calibrate the sensor

hx711.set\_reference\_unit(known\_weight)

while True:

# Read weight from load cell sensor

weight = hx711.get\_weight(5)

# Print the measured weight

print("Measured weight:", weight, "grams")

# Delay between readings

time.sleep(1)

***Python Script Development:***

1. **Data Acquisition:** Develop Python code to read sensor data from the microcontroller board. Use libraries specific to the sensors and communication protocols.
2. b. **Data Processing:** Implement algorithms to process sensor data, such as calculating occupancy percentages, averaging air quality readings, and tracking water level changes.

c. **Alert Generation:** Set up logic to generate alerts or notifications based on sensor readings. For instance, send alerts when air quality deteriorates or when soap dispensers need refilling.

d. **Data Visualization:** Create a user interface or dashboard to visualize sensor data and alert status in real-time. Use libraries like Matplotlib or Plotly to generate graphs and charts.

import time

import board

import busio

import digitalio

import adafruit\_ccs811

import adafruit\_vl53l0x

import adafruit\_hx711

# Define GPIO pin for occupancy sensor

occupancy\_pin = digitalio.DigitalInOut(board.D2)

occupancy\_pin.direction = digitalio.Direction.INPUT

# Initialize I2C communication

i2c = busio.I2C(board.SCL, board.SDA)

# Create CCS811 air quality sensor object

ccs811 = adafruit\_ccs811.CCS811(i2c)

# Create VL53L0X ultrasonic sensor object

vl53l0x = adafruit\_vl53l0x.VL53L0X(i2c)

# Define GPIO pins for HX711 load cell amplifier

hx711\_data\_pin = board.D3

hx711\_clock\_pin = board.D4

# Create HX711 object

hx711 = adafruit\_hx711.HX711(hx711\_data\_pin, hx711\_clock\_pin)

# Calibrate the sensor with known weight

hx711.set\_reference\_unit(calibration\_factor)

# Define threshold values for alerts

eco2\_threshold = 1000 # ppm

tvoc\_threshold = 200 # ppb

water\_level\_threshold = 20 # %

soap\_level\_threshold = 20 # %

while True:

# Read occupancy sensor status

occupancy\_status = occupancy\_pin.value

if occupancy\_status:

print("Occupancy detected")

else:

print("No occupancy detected")

# Check if air quality sensor is ready

if not ccs811.data\_ready:

continue

# Read air quality data

eco2 = ccs811.eco2

tvoc = ccs811.tvoc

print("eCO2:", eco2, "ppm")

print("TVOC:", tvoc, "ppb")

# Check for air quality alerts

if eco2 > eco2\_threshold:

print("High eCO2 level detected")

if tvoc > tvoc\_threshold:

print("High TVOC level detected")

# Read distance from ultrasonic sensor

distance = vl53l0x.range

# Convert distance to water level percentage

water\_level = (1 - distance / max\_distance) \* 100

print("Water level:", water\_level, "%")

# Check for water level alert

if water\_level < water\_level\_threshold:

print("Low water level detected")

# Read weight from load cell sensor

weight = hx711.get\_weight(5)

# Calculate soap level based on weight change

soap\_level = soap\_level - (weight / initial\_weight) \* 100

print("Soap level:", soap\_level, "%")

# Check for soap level alert

if soap\_level < soap\_level\_threshold:

print("Low soap level detected")

# Delay between readings

time.sleep(1)

***Deployment and Testing:***

a. **Deploy Script:** Deploy the Python script onto the microcontroller board, ensuring compatibility with the board's operating system and libraries.

b. **System Testing:** Thoroughly test the entire system, including sensor readings, data processing, alert generation, and data visualization.

***Maintenance and Updates:***

a. **Regular Monitoring:** Monitor the system regularly to ensure proper functioning and address any issues promptly.

b. **Software Updates:** Update the Python script and libraries as needed to maintain compatibility and incorporate new features.

***Additional Considerations:***

a. **Networking:** If remote monitoring or control is desired, integrate the system with a network connection (e.g., Wi-Fi, Ethernet) and implement secure communication protocols.

b. **Data Storage:** Consider storing sensor data and alert logs in a database for historical analysis and trend identification.

c. **User Feedback:** Gather feedback from restroom users to improve the system's effectiveness and user experience.

By following these steps and considerations, you can successfully build an IoT-based smart public restroom system that enhances hygiene, efficiency, and user satisfaction.