**SMART PUBLIC RESTROOM WITH IoT**

**Manokar S Mohammed Riffath S**

**1. Introduction**

The "Smart Public Restroom with IoT" project is aimed at creating an intelligent public restroom that leverages Internet of Things (IoT) technology to enhance hygiene and user experience. This project utilizes various sensors to monitor restroom conditions and implement automated actions such as flushing and periodic cleaning. By doing so, it aims to provide a cleaner and more efficient public restroom experience.

**2. Data Input**

The project relies on various sensors to gather real-time data about the restroom environment. The following section details how data input is obtained:

**2.1 Occupancy Detection**

- The occupancy sensor, a Passive Infrared (PIR) motion sensor, is strategically placed at the entrance and key locations within the restroom.

- It continuously monitors the restroom for any motion.

- When a person enters the restroom, the occupancy sensor detects the motion and sends a signal to the control module, indicating occupancy**.**

**2.2 Temperature-Based Entry Detection**

- A temperature sensor (e.g., DHT22) placed in a central location within the restroom constantly measures the ambient temperature.

- The system is programmed to detect a rise in temperature of 2°C or more, which typically occurs when a person enters and the body heat increases the temperature.

- A temperature increase of this magnitude is considered indicative of someone entering the restroom.

**2.3 Humidity Sensor**

- A humidity sensor (e.g., DHT22) is also positioned centrally within the restroom to monitor humidity levels.

- The humidity sensor provides precise humidity readings.

- The system is programmed to trigger actions when the humidity level exceeds a predefined threshold (e.g., 60%).

**3) Data Processing**

Data from the sensors is processed in real-time to make informed decisions regarding restroom operations:

**3.1. Occupancy and Entry Detection**

- When the occupancy sensor detects motion, it sends an occupancy signal to the control module.

- If the temperature sensor records a temperature increase of 2°C or more around the same time, it corroborates the entry.

- The control module processes these signals and increments the occupancy counter.

**3.2. Automatic Flushing**

- The control module continuously monitors the occupancy counter.

- When occupancy is detected by the PIR sensor, the control module initiates the flushing mechanism, ensuring automatic flushing after restroom use**.**

**3.3. Periodic Cleaning**

**-** A counter keeps track of the number of occupants.

- When the predefined cleaning interval (e.g., every 3 persons) is reached, the control module initiates an automatic cleaning mechanism. This process resets the occupancy counter.

**3.4. Humidity-Based Flushing**

- The system's control module continuously monitors humidity levels.

- When the humidity sensor records a humidity level exceeding the predefined threshold (e.g., 60%), the control module triggers flushing**.**

**4. Code and Functionality**

The code for the project is developed in Python and includes the following functionalities:

**4.1 Sensor Data Reading**

The project's Python code is responsible for reading data from the occupancy, temperature, and humidity sensors. This is the foundational step for collecting information about the restroom environment.

**4.2 Flushing Logic**

One of the core functionalities is the automatic flushing system. It's implemented based on occupancy detection. When someone enters the restroom, the system initiates the flushing mechanism, ensuring a clean and ready-to-use environment for the next user.

**4.3 Automatic Cleaning**

To maintain a high standard of hygiene, the system is designed to initiate automatic cleaning after a predetermined number of occupants. This periodic cleaning ensures that the restroom remains in optimal condition throughout the day.

**4.4 Humidity-Based Flushing**

The system also includes the capability to trigger flushing based on humidity levels. When the humidity exceeds a predefined threshold, the system automatically flushes, ensuring a comfortable and pleasant restroom experience.

**4.5 MQTT Communication**

The project relies on MQTT (Message Queuing Telemetry Transport) for communication. The Python code handles the connection to an MQTT server, facilitating remote control and monitoring. This allows for real-time interaction with the restroom system.

**4.6 Sensor Data Display**

To keep users informed about the restroom's condition, the Python code also interfaces with a digital display. The code dynamically updates the display with sensor data, presenting users with vital information such as occupancy status, temperature, and humidity levels**.**

By combining these functionalities, the Python code forms the backbone of the "Smart Public Restroom with IoT" project, ensuring automated, efficient, and user-friendly restroom operations.

**5. Digital Display**

- A digital display, connected to the control module, presents users with current restroom status information.

- The display shows the following information:

- Current occupancy status (e.g., "Vacant" or "Occupied").

- Real-time temperature within the restroom (e.g., "Temperature: 25°C").

- Current humidity level (e.g., "Humidity: 55%").

This information is dynamically updated on the digital display to keep users informed about the status of the restroom, temperature, and humidity conditions.

**CODE:**

import paho.mqtt.client as mqtt

import RPi.GPIO as GPIO

import time

# Configure GPIO pins for actuators and display

GPIO.setmode(GPIO.BCM)

FLUSH\_PIN = 17 # Pin for flushing

CLEANING\_PIN = 18 # Pin for cleaning (e.g., UV lights or cleaning robot)

DIGITAL\_DISPLAY\_PIN = 19 # Pin for the digital display

GPIO.setup(FLUSH\_PIN, GPIO.OUT)

GPIO.setup(CLEANING\_PIN, GPIO.OUT)

GPIO.setup(DIGITAL\_DISPLAY\_PIN, GPIO.OUT)

# Set up MQTT client for IoT communication

MQTT\_BROKER = "your\_broker\_address"

MQTT\_TOPIC\_SENSORS = "restroom/sensors"

MQTT\_CLIENT\_ID = "restroom\_pi"

client = mqtt.Client(MQTT\_CLIENT\_ID)

client.connect(MQTT\_BROKER, 1883)

# Sensor reading functions (replace with actual sensor code)

def read\_temperature\_sensor():

# Replace with code to read temperature sensor

# Example: temperature = read\_actual\_temperature\_sensor()

temperature = 25.5 # Replace with actual sensor reading

return temperature

def read\_humidity\_sensor():

# Replace with code to read humidity sensor

# Example: humidity = read\_actual\_humidity\_sensor()

humidity = 50.5 # Replace with actual sensor reading

return humidity

def read\_occupancy\_sensor():

# Replace with code to read occupancy sensor

# Example: occupancy = read\_actual\_occupancy\_sensor()

occupancy = True # Replace with actual sensor reading (True for occupied, False for vacant)

return occupancy

# Flushing function

def flush\_toilet():

GPIO.output(FLUSH\_PIN, GPIO.HIGH) # Activate the flush mechanism

time.sleep(2) # Adjust as needed

GPIO.output(FLUSH\_PIN, GPIO.LOW) # Deactivate the flush mechanism

# Cleaning function (example for periodic cleaning)

def clean\_restroom():

# Start the cleaning mechanism here

print("Restroom is being cleaned.")

GPIO.output(CLEANING\_PIN, GPIO.HIGH) # Activate cleaning mechanism

time.sleep(5) # Adjust as needed

GPIO.output(CLEANING\_PIN, GPIO.LOW) # Deactivate cleaning mechanism

# Digital display function

def display\_message(message):

# Control the digital display to show the message

print("Display message:", message)

# Replace this with code to show the message on your specific display

# MQTT message handler

def on\_message(client, userdata, msg):

# Process incoming MQTT messages

payload = msg.payload.decode("utf-8")

# Implement your logic for remote commands here

pass

# Subscribe to MQTT commands

client.subscribe("restroom/commands")

client.on\_message = on\_message

client.loop\_start()

# Counters for occupancy and cleaning

occupancy\_counter = 0

cleaning\_interval = 3 # Clean automatically every 3 persons

try:

while True:

temperature = read\_temperature\_sensor()

humidity = read\_humidity\_sensor()

occupancy = read\_occupancy\_sensor()

# Detect person based on temperature rise

if temperature > 25:

occupancy\_counter += 1

# Check if it's time to trigger automatic cleaning

if occupancy\_counter >= cleaning\_interval:

clean\_restroom()

occupancy\_counter = 0

# Flushing logic based on humidity

if humidity > 60:

flush\_toilet()

# Publish sensor data to MQTT topic

sensor\_data = {

"temperature": temperature,

"humidity": humidity,

"occupancy": occupancy

}

client.publish(MQTT\_TOPIC\_SENSORS, str(sensor\_data))

# Display sensor data on the digital display

display\_message(f"Temp: {temperature}°C, Humidity: {humidity}%, Occupancy: {occupancy}")

time.sleep(10) # Adjust the interval as needed

except KeyboardInterrupt:

GPIO.cleanup()

client.disconnect()

client.loop\_stop()

**6. Conclusion**

The "Smart Public Restroom with IoT" project demonstrates the effective use of sensors for real-time data input, processing, and automation of restroom functions. The system uses occupancy, temperature, and humidity sensors to monitor conditions and act accordingly, ensuring a cleaner and more efficient public restroom experience. The digital display provides users with critical information about the restroom environment, enhancing their overall experience.

**7. References**

**- Paho MQTT Python Client - [https://pypi.org/project/paho-mqtt/](https://pypi.org/project/paho-mqtt/)**

**- Raspberry Pi GPIO -[https://pypi.org/project/RPi.GPIO/](https://pypi.org/project/RPi.GPIO/)**