**IOT PHASE- 5**

**PROJECT TITLE:** SmartPublicRestroom

**PHASE 5 :** Project documentation

Presented by

* SNEHA K
* JENIFER TABITHA S
* GAYATHRI P
* JASMIN MARY J

**INRODUCTION:**

A cutting-edge initiative that seamlessly integrates IoT sensor technology and Raspberry Pi solutions to enhance the public restroom experience. Our comprehensive system includes a network of strategically placed sensors, which, in real-time, monitor restroom occupancy, cleanliness, and supply levels. This data is then visualized through an intuitive platform, providing users with up-to-date restroom information, ensuring a more convenient and hygienic experience. Explore our innovative approach in the diagram below, where technology meets convenience for the modern public space.



**The objectives of a Smart Public Restroom project :**

1. **Enhanced User Experience**: Improve the overall experience of restroom visitors by providing real-time information on restroom availability, cleanliness, and amenities.

2. **Efficient Resource** **Management**: Optimize the use of resources such as water, energy, and cleaning supplies through data-driven insights.

3**. Cost Savings**: Reduce operational costs by minimizing unnecessary maintenance and maximizing resource efficiency.

4. **Hygiene** **and** **Sanitation**: Ensure a cleaner and more sanitary environment by monitoring and maintaining cleanliness levels in real-time.

5. **Accessibility** : Make restrooms more accessible by providing information on nearby accessible restrooms for individuals with specific needs.

6. **Data Collection** **and** **Analysis**: Gather valuable data on restroom usage and conditions for continuous improvement and informed decision-making.

7. **Environmental Impact**: Minimize the environmental footprint through resource conservation and waste reduction.

Setting up IOT sensors for a public toilet involves a combination of hardware and software components to monitor various aspects of the restroom. Here's a basic overview of the sensor setup:

1. Occupancy Sensors:

* Install occupancy sensors at the entrance of each restroom stall.
* These sensors detect when a stall is in use or vacant.

2. Toilet Paper and Soap Dispenser Sensors:

* Attach sensors to toilet paper and soap dispensers to monitor supply levels.
* Sensors can trigger alerts when supplies are running low.

3. Water Usage Sensors:

* Integrate water flow sensors on faucets and flush mechanisms.
* Monitor water consumption and detect leaks or excessive usage.

4. Air Quality Sensors:

* Use air quality sensors to measure temperature and humidity.
* Ensure a comfortable and pleasant restroom environment.

5. Occupancy and Wait-Time Display:

* Display occupancy status and estimated wait times outside the restroom.
* LED indicators or digital displays can be used for this purpose.

6. Security Cameras:

* Install security cameras for security and monitoring purposes.
* Ensure privacy considerations are met and data is stored securely.

7. Microcontroller or Raspberry Pi Integration:

* Connect all sensors to a microcontroller or a Raspberry Pi for data collection.
* Use appropriate interfaces such as GPIO or USB to link sensors with the controller.

8. Data Processing and Communication:

* Develop software to process sensor data and transmit it to a central server.
* Use protocols like MQTT or HTTP for communication.

9. Centralized Dashboard:

* Create a web-based dashboard or mobile app to display real-time restroom data.
* Include information on occupancy, supply levels, and air quality.

10. Alerts and Notifications:

* Implement alert systems to notify maintenance staff or users when supplies are low or when cleaning is required.

11. Analytics and Reporting:

* Analyze data to identify usage patterns, optimize maintenance schedules, and enhance the user experience.

12. Privacy and Data Security:

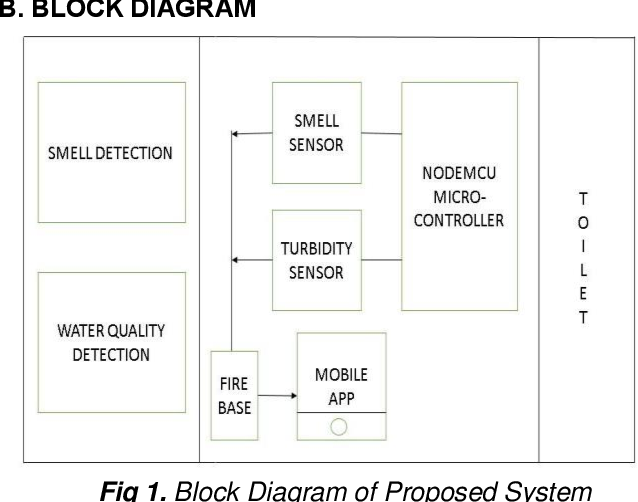
* Ensure data collected is handled with the utmost care and in compliance with data privacy regulations.

13. Regular Maintenance:

* Establish a maintenance schedule to monitor the functioning of sensors and replace batteries or sensors as needed.

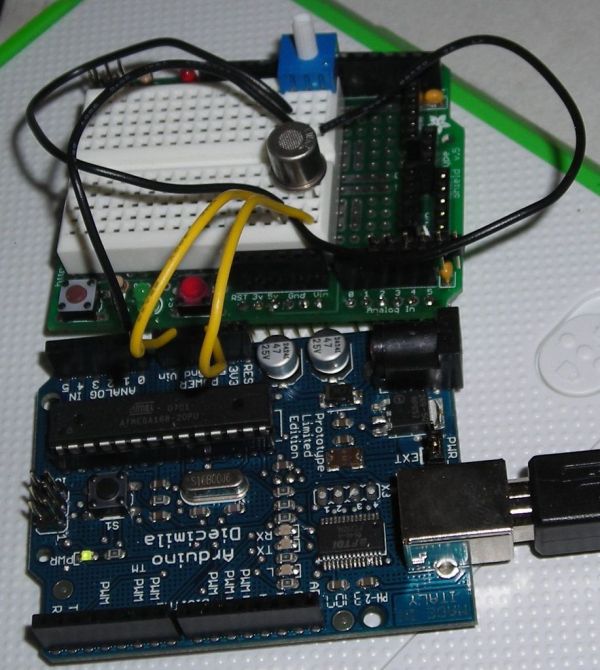
Here we includes diagram, schematics and screenshots of Iot sensors, restroom information platform, and mobile and interfaces:

**BLOCK DIAGRAM FOR SMART PUBLIC RESTROOM:**

****

**IOT SENSORS SETUP AND DEPLOYMENT OF SENSORS:**

**ODOR SENSORS:**

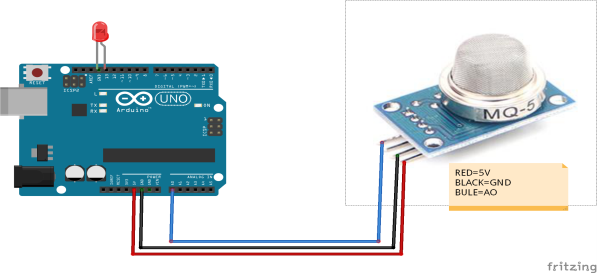
Gas Sensors (e.g., methane or ammonia detectors) : These sensors continuously monitor air quality in the toilet. If unusual odors are detected, the system can activate a ventilation system to improve air quality 

**FIG2**. ODOR SENSOR WITH ARDUINO

**WORKING PRINCIPLE:**

* A gas sensor in a smart public toilet, connected to an Arduino, operates by measuring the concentration of specific gases, like methane or hydrogen sulfide, in the air.
* It uses a gas-sensitive material that changes its electrical properties when exposed to these gases.
* The Arduino reads these changes and converts them into digital data.
* If gas levels exceed safe limits, the Arduino triggers an alarm or ventilation system, ensuring public safety.
* This real-time monitoring enhances hygiene and prevents accidents in public restrooms.

**PIN CONNECTION WITH ARDUINO:**



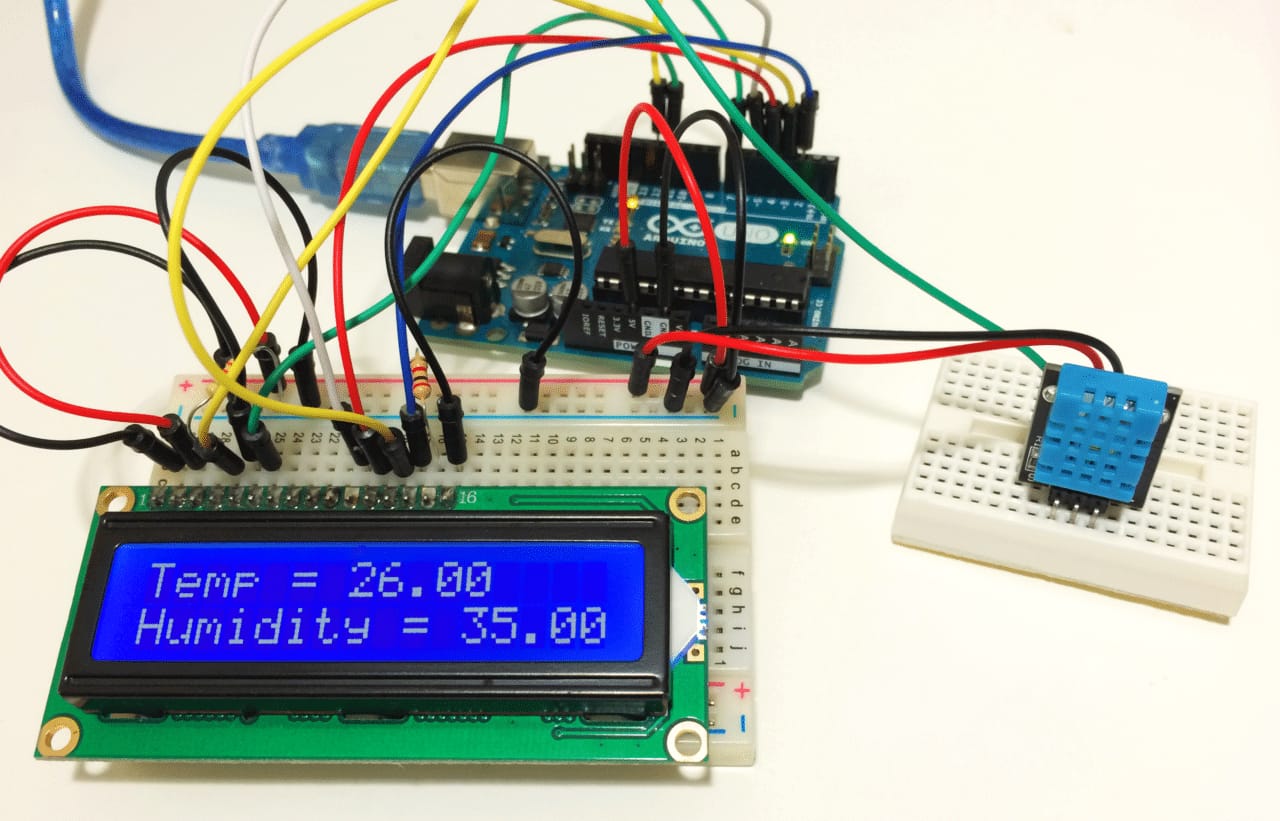
RED-11V

BLACK-GND,

BLUE-AD

**TEMPERATURE AND HUMIDITY SENSOR:**

* To monitor restroom conditions, which could be useful for maintenance or user comfort.

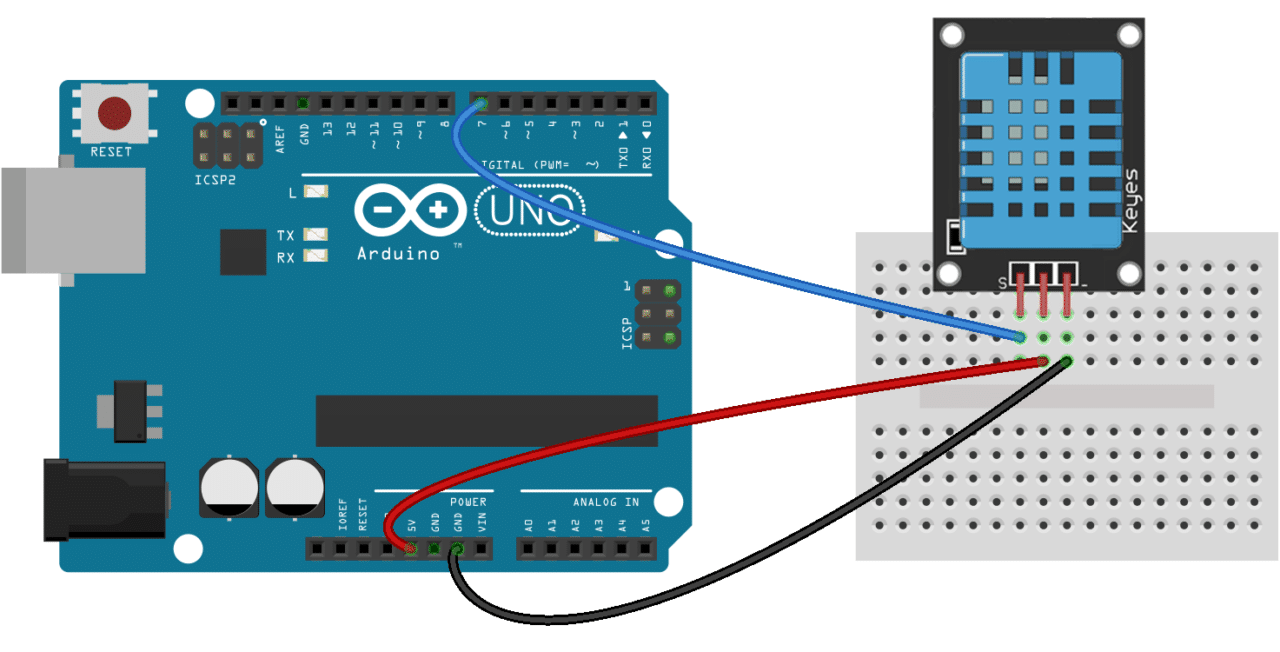


**FIG3.** Temperature and humidity sensor

**WORKING PRINCIPLE:**

* A temperature and humidity sensor, like the DHT22, is connected to an Arduino in a smart public toilet.
* The sensor measures ambient temperature and humidity levels.
* The Arduino reads data from the sensor through its digital pins and converts it into meaningful values. These values can be displayed on an LCD screen or sent to a central control system for monitoring and control.
* The system can trigger actions like activating fans or heaters based on these readings, ensuring user comfort and efficient energy use.
* This makes the smart public toilet more user-friendly and environmentally friendly.

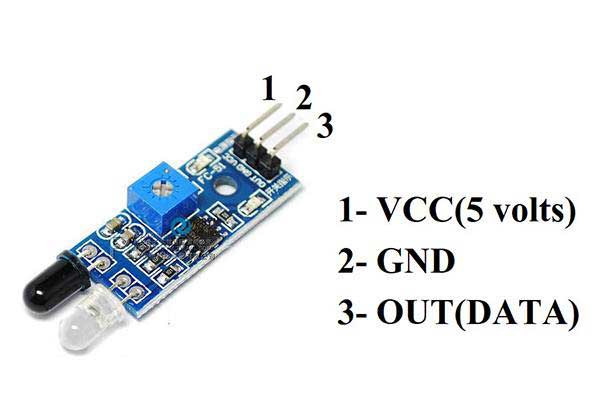
**PIN CONNECTION WITH ARDUINO:**



* Connect GND Pin of T&H sensor to GND on Arduino
* Connect Vcc Pin of T&H sensor to 11V on Arduino
* Connect Data Pin of T&H sensor to D3 on Arduino

**OCCUPANCY SENSOR:**

**Infrared** **Sensors**: These sensors can detect the presence of a user within the toilet stall. When someone enters, the sensor triggers the lighting and ventilation systems to turn on, ensuring a well-lit and comfortable environment.

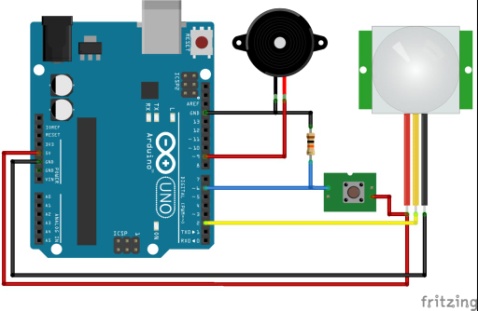


**FIG11.** INFRAREDSENSOR

**WORKING PRINCIPLE:**

* An infrared occupancy sensor in a smart public toilet operates by emitting infrared light.
* When a person enters, their body heat reflects this light back to the sensor.
* Arduino processes the signal, triggering actions like activating lights, flushing, or opening doors.
* The sensor continually monitors for changes in the infrared signal, allowing it to detect occupancy and respond accordingly.
* This efficient, non-contact method ensures an automated and hygienic user experience in the smart toilet facility.

**PIN CONNECTION WITH ARDUINO:**

****

* Connect the VCC pin of the infrared sensor to the 11V pin of the Arduino.
* Connect the GND pin of the infrared sensor to the GND pin of the Arduino.
* Connect the OUT pin of the infrared sensor to any digital pin of the Arduino.

**MOBILE APP DEVELOPMENT:**

**Find a toilet:**

The "Find a Toilet" is designed to help users locate nearby public restrooms efficiently. Its key features include:

1. GPS Integration: The app uses GPS to pinpoint the user's location.

2. Map Interface: It displays a map with restroom icons, highlighting nearby facilities.

3. Search Filters: Users can filter by cleanliness, accessibility, or ratings.

4. User Reviews: Allows users to leave reviews and ratings for restrooms.

5. Directions: Provides step-by-step directions to the chosen restroom.

6. Crowd sourcing: Users can add new restroom locations to the database.

7. Emergency Features: Some apps offer emergency options for urgent needs.

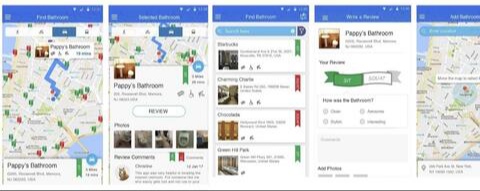
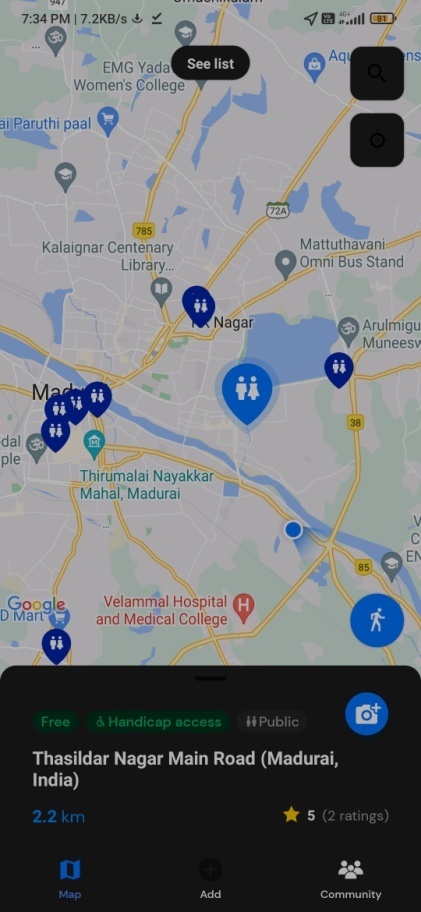
8. Offline Access: Some apps allow users to download maps for offline use.

9. Public Awareness: Promotes awareness of public restroom availability.

10. Community-Building: Connects users through a shared need for clean, accessible restrooms.

These design are invaluable for travelers, parents, and anyone in need of a public restroom, improving convenience and sanitation.

**Design:**

****

**Cleanliness:**

The "Find a Clean Public Toilet" is designed to help users locate nearby public restrooms and assess their cleanliness. It uses a mobile interface for easy access.

1. Frontend: Developed using Android Studio and Java, it provides a user-friendly interface.

2. Backend: Utilizes a cloud-based server, built with Node.js and MongoDB, to store restroom data.

3. Google Maps API: Displays toilet locations on a map.

4. User Authentication Package: Ensures secure user accounts.

5. Location Services: Tracks user's GPS coordinates.

6. Review and Rating System: Allows users to rate and review restrooms.

7. Notification Package: Sends alerts for nearby facilities.

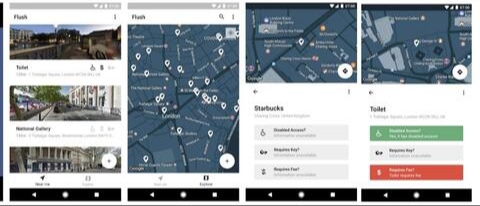
8. Data Analytics Tools: Analyzes user reviews and usage patterns.

9. Database Management: Stores toilet data, user reviews, and ratings.

10. Search and Filter Functions: Helps users find specific toilet features (e.g., cleanliness, accessibility).

The app's software design and packages work together to provide a practical solution for finding clean public toilets with user-generated feedback for a better restroom experience.

**DESIGN:**

****

A real-time restroom information system enhances user experience and restroom management in several ways:

**User Experience Enhancement:**

1. Improved Convenience: Users can check real-time restroom occupancy and availability before heading to the restroom, reducing wait times and increasing convenience.

2. Enhanced Cleanliness: Sensors can monitor cleanliness levels, and users can be informed about the restroom's recent cleaning status. This encourages users to choose cleaner restrooms.

3. Supply Awareness:Real-time data on soap and toilet paper supply levels ensures users are not caught off guard. They can choose restrooms with ample supplies, leading to a more comfortable experience.

4. Accessible Restrooms:The system can provide information on nearby accessible restrooms for individuals with specific needs, improving inclusivity.

5. Reduced Queue Frustration: Real-time occupancy data allows users to select less crowded restrooms, reducing frustration during busy times.

**Restroom Management Improvement:**

1. Optimized Cleaning Schedules: The system can trigger cleaning alerts based on usage patterns or when cleanliness falls below a set threshold. This ensures efficient resource allocation for maintenance.

2. Supply Replenishment Efficiency: Supply levels are monitored, and alerts are sent when they run low, leading to more efficient and timely restocking.

3. Water and Energy Conservation: Water usage data from sensors can help identify inefficient fixtures and detect leaks, leading to conservation and cost savings.

4. Data-Driven Decision Making: Collected data provides insights into restroom usage patterns, helping facility managers make informed decisions about restroom design and maintenance.

5. Cost Reduction: By optimizing cleaning and supply replenishment, operational costs are reduced. Resource efficiency minimizes waste, leading to cost savings.

6. Environmental Impact Reduction: Reduced resource consumption and waste contribute to a more eco-friendly operation, aligning with sustainability goals.

7. User Feedback and Engagement: Some systems allow users to provide feedback or ratings, giving facility managers valuable insights into user satisfaction and areas that need improvement.

**Wokwi Simulation:**

The project can be simulated by using wokwi simulation software.

//Code for the device to detect the status of the Waste bins

#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#define SOUND\_SPEED 0.034//define sound speed in cm/uS

#define LED 15

const int trigPin = 13;

const int echoPin = 12;

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "rr1qlb"//IBM ORGANITION ID

#define DEVICE\_TYPE "sensor\_devices"//Device type mentioned in ibm watson IOT Platform

#define DEVICE\_ID "swmsfmc\_D1"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "swmsfmc\_D1" //Token

String data3;

float distance;

long duration;

//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name

char publishTopic[] = "iot-2/evt/data/fmt/json";// topic name and type of event perform and format in which data to be send

char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING

char authMethod[] = "use-token-auth";// authentication method

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;//client id

//-----------------------------------------

WiFiClient wifiClient; // creating the instance for wificlient

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server id,portand wificredential

void setup()// configureing the ESP32

{

Serial.begin(115200); // Starts the serial communication

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

pinMode(LED,OUTPUT);

delay(10);

Serial.println();

}

void loop()// Recursive Function

{

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculate the distance

distance = duration \* SOUND\_SPEED/2;

if (distance<100)

{

digitalWrite(LED, HIGH);

Serial.print("Alert! Bin is filled.");

Serial.println();

delay(1000);

}

else

{

digitalWrite(LED, LOW);

Serial.print("Bin is not filled.");

Serial.println();

}

if(distance<100)

{

PublishData(distance);

}

delay(1000);

if (!client.loop()) {

mqttconnect();

}

}

/.....................................retrieving to Cloud.............................../

void PublishData(float distance) {

mqttconnect();//function call for connecting to ibm

/\*

creating the String in in form JSon to update the data to ibm cloud

\*/

String payload = "{\"Alert\":";

payload += distance;

payload += "}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(publishTopic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else it will print publish failed

} else {

Serial.println("Publish failed");

}

}

void mqttconnect() {

if (!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void wificonnect() //function defination for wificonnect

{

Serial.println();

Serial.print("Connecting to ");

WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void initManagedDevice() {

if (client.subscribe(subscribetopic)) {

Serial.println((subscribetopic));

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength)

{

Serial.print("callback invoked for topic: ");

Serial.println(subscribetopic);

for (int i = 0; i < payloadLength; i++) {

//Serial.print((char)payload[i]);

data3 += (char)payload[i];

}

Serial.println("data: "+ data3);

if(data3=="lighton")

{

Serial.println(data3);

digitalWrite(LED,HIGH);

}

else

{

Serial.println(data3);

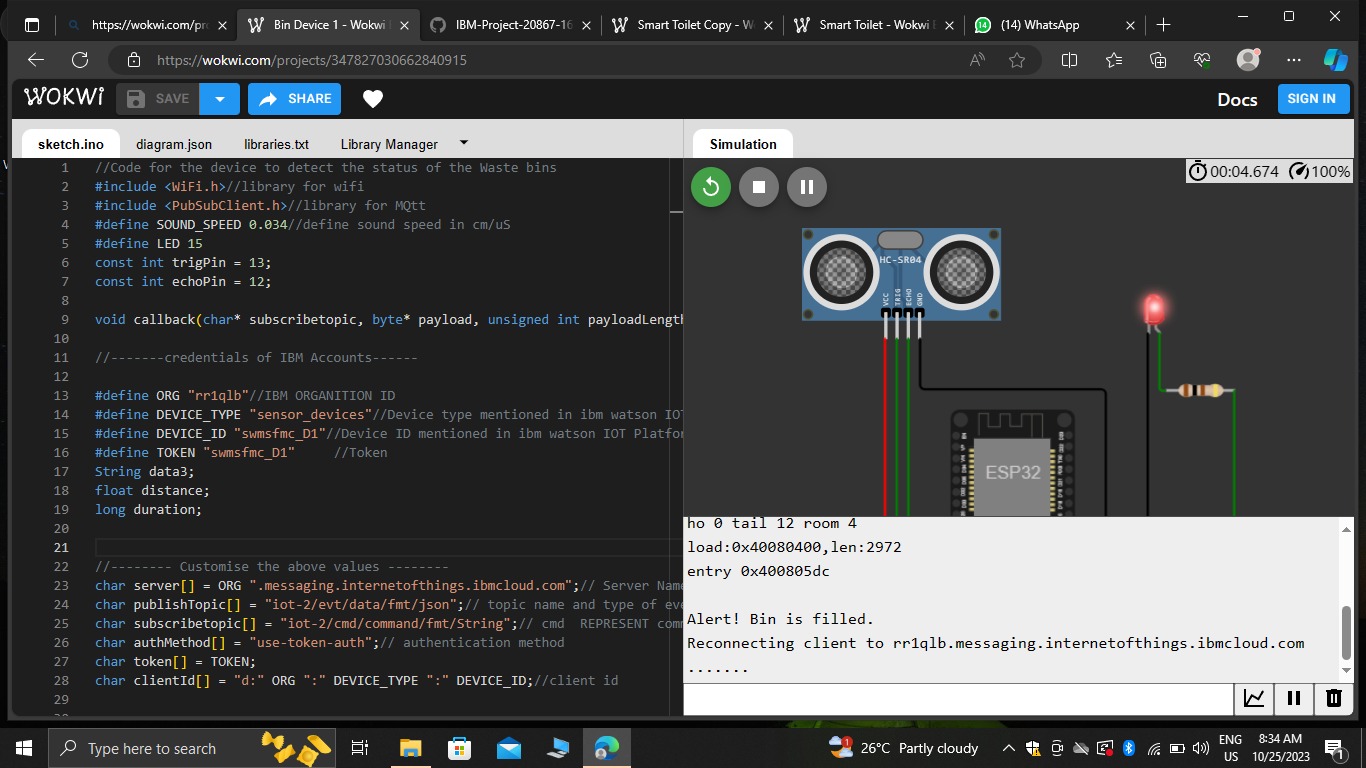
digitalWrite(LED,LOW);

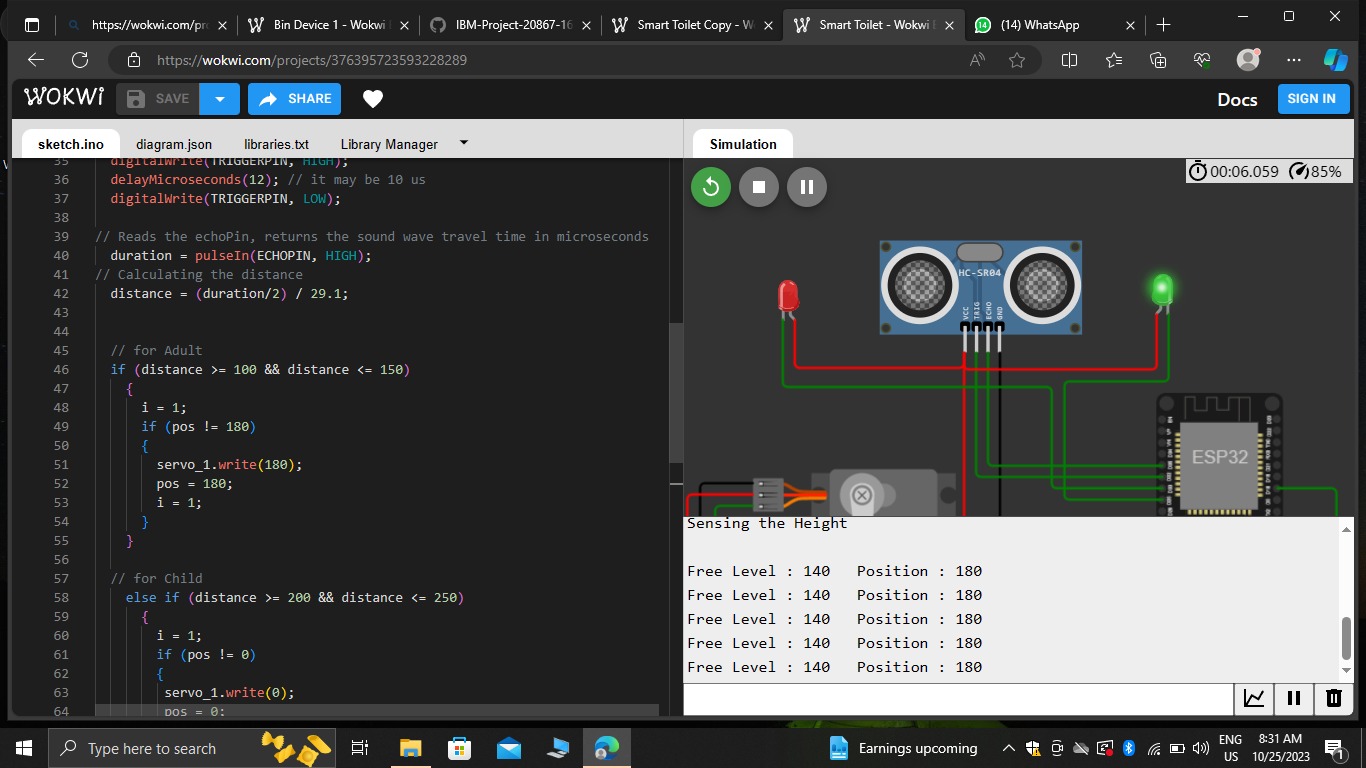
}

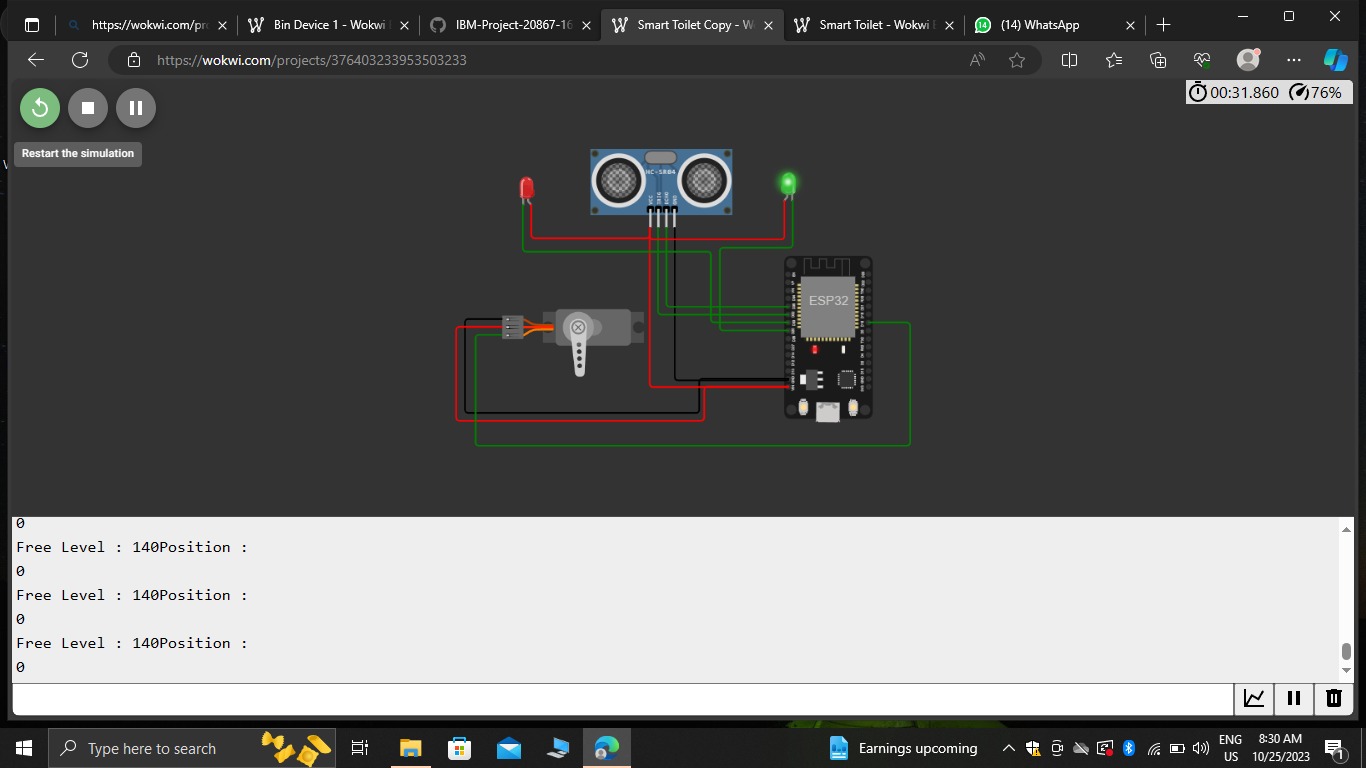
data3="";

}

**CODE IMPLEMENATION:**

****

****

****

**Develop a Python script on the loT sensors to send real-time occupancy and cleanliness data to the restroom information platform.**

Import serial

Import time

# Set the COM port where your Arduino is connected

Arduino\_port = ‘COM3’ # Change this to the appropriate port

# Set the baud rate to match your Arduino’s configuration

Baud\_rate = 9600

# Open a connection to the Arduino

Ser = serial.Serial(arduino\_port, baud\_rate)

Try:

While True:

# Read data from the Arduino

Arduino\_data = ser.readline().decode().strip()

# Split the data into sensor values

Sensor\_values = arduino\_data.split(‘,’)

If len(sensor\_values) == 4:

Mq2\_value, ir\_sensor\_value, temperature, humidity = sensor\_values

Print(f’MQ-2 Value: {mq2\_value}, IR Sensor Value: {ir\_sensor\_value}’)

Print(f’Temperature: {temperature}°C, Humidity: {humidity}%’)

Else:

Print(“Invalid data received from Arduino”)

# Delay for a moment before reading again

Time.sleep(1)

Except KeyboardInterrupt:

Print(“Exiting the program”)

Finally:

Ser.close()

**SAMPLE OUTPUT** :

MQ-2 Value: 123, IR Sensor Value: 456

Temperature: 25°C, Humidity: 40%

MQ-2 Value: 124, IR Sensor Value: 457

Temperature: 26°C, Humidity: 41%

MQ-2 Value: 125, IR Sensor Value: 458

Temperature: 27°C, Humidity: 42%

...

**RASPBERRY PI CODING:**

Import RPi.GPIO as GPIO

Import time

Import picamera

From flask import Flask, render\_template, request

App = Flask(\_\_name\_\_)

# Define GPIO pin numbers

Occupancy\_pin = 17

Flush\_pin = 18

Light\_pin = 27

# Initialize GPIO pins

GPIO.setmode(GPIO.BCM)

GPIO.setup(occupancy\_pin, GPIO.IN)

GPIO.setup(flush\_pin, GPIO.OUT)

GPIO.setup(light\_pin, GPIO.OUT)

# Camera setup

Camera = picamera.PiCamera()

# Define global variables

Is\_occupied = False

# Web interface

@app.route(‘/’)

Def index():

Return render\_template(‘index.html’, is\_occupied=is\_occupied)

@app.route(‘/flush’, methods=[‘POST’])

Def flush():

GPIO.output(flush\_pin, GPIO.HIGH)

Time.sleep(2) # Flush for 2 seconds

GPIO.output(flush\_pin, GPIO.LOW)

Return render\_template(‘index.html’, is\_occupied=is\_occupied)

@app.route(‘/light’, methods=[‘POST’])

Def light():

GPIO.output(light\_pin, not GPIO.input(light\_pin))

Return render\_template(‘index.html’, is\_occupied=is\_occupied)

# Sensor monitoring

Def monitor\_occupancy():

Global is\_occupied

While True:

If GPIO.input(occupancy\_pin):

Is\_occupied = True

Else:

Is\_occupied = False

Time.sleep(1) # Check occupancy every second

If \_\_name\_\_ == ‘\_\_main\_\_’:

Try:

# Start occupancy monitoring thread

Occupancy\_thread = threading.Thread(target=monitor\_occupancy)

Occupancy\_thread.start()

App.run(host=’0.0.0.0’, port=80) # Run the web interface

Except KeyboardInterrupt:

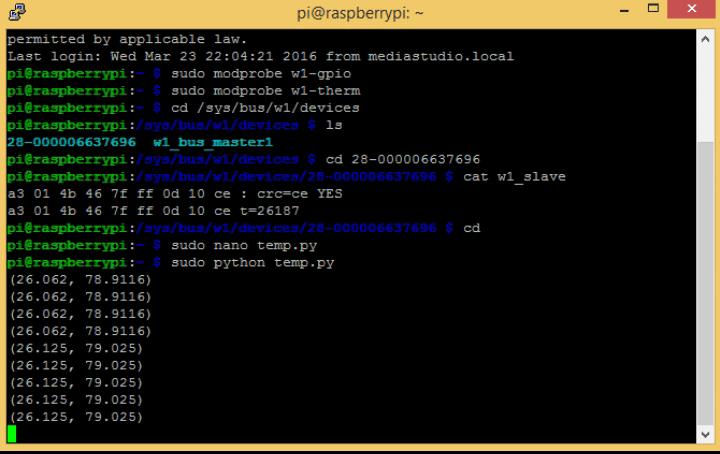
Pass

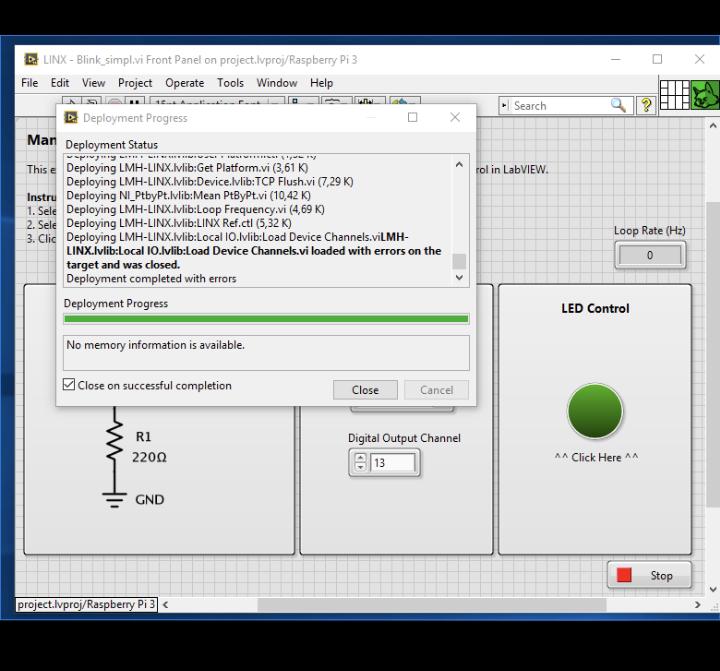
Finally:

# Clean up GPIO on program exit

GPIO.cleanup()

**SAMPLE OUTPUT:**

****

****

**CONCLUSION:**

In conclusion, the implementation of a Smart Public Restroom, combining IoT sensors with a robust software platform, has demonstrated its potential to revolutionize public restroom management. The integration of sensor hardware has enabled real-time monitoring of cleanliness and availability, enhancing user experience. The software platform's user-friendly interface empowers users to access vital restroom information conveniently. This project underlines the significance of data-driven decision-making and user-centric design in enhancing public facilities. Furthermore, the IoT-sensor-driven approach optimizes resource allocation and maintenance efforts, ultimately leading to more efficient and cleaner public restrooms. This innovation has the potential to elevate public hygiene standards and offer a model for smarter, more responsive public infrastructure in the future.

File naming convention: IOT\_Phase 5