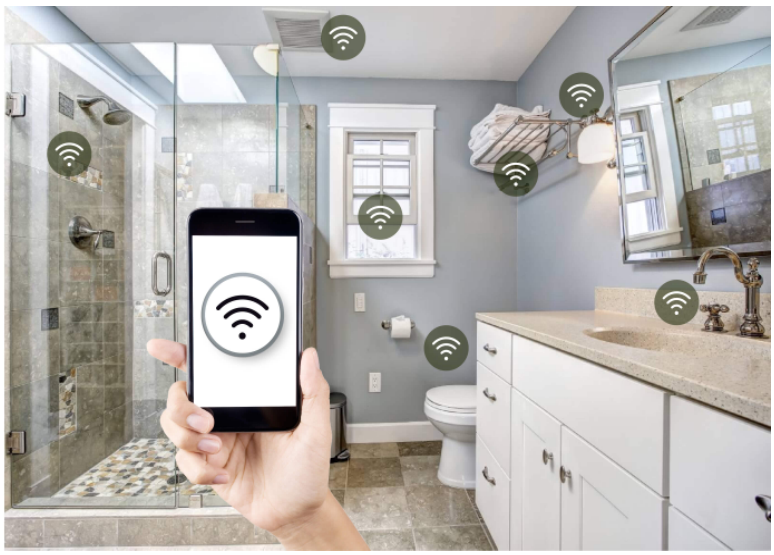
SMART PUBLIC RESTROOM USING RASPBERRY-Pi

INTRODUCTION:

Smart public restrooms are a modern and innovative approach to enhancing the public restroom experience. These facilities are equipped with advanced technology and features designed to improve hygiene, accessibility, and overall user convenience. Smart public restrooms often include amenities such as touchless fixtures, automatic flushing toilets, motion-activated sinks, and sensor-operated soap dispensers, all of which contribute to a more sanitary environment. Additionally, these restrooms may offer features like real-time occupancy tracking, cleanliness monitoring, and even mobile apps to help users find and access nearby facilities. Smart public restrooms are becoming increasingly common in urban areas, airports, shopping centers, and other high-traffic locations, aiming to provide a more comfortable and efficient restroom experience for the public.



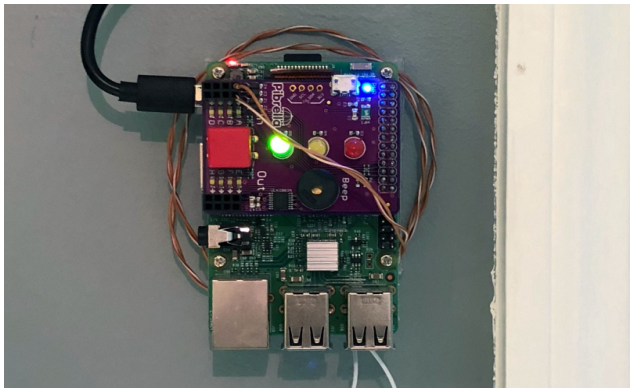
STEP1:

To attain the entire process of this project,need to read data from DHT11 or DHT22 Sensor using raspberry.pi.Here is a python code program using the adafruit DHT library to accomplish this steps mentioned in upcoming.

- First,you'll need to install the adafruit DHT library if you haven't

already.Open a terminal on your raspberry Pi and run the following commands .

CIRCUIT:



Preparing Raspberry Pi

Installing the OS:

1) **Download the latest Raspbian.**
   https://www.raspberrypi.com/software/

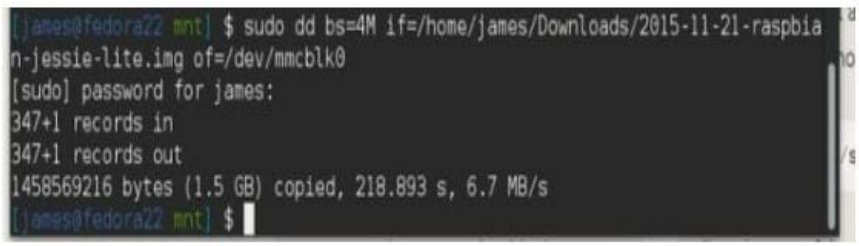
2) **Follow the instruction to install the Raspbian in your micro SD card.**

If you are using Linux you can follow these steps to install Raspbian Lite on your micro SD card:

- **Check the device name for your micro SD by running:**

```
[james@fedora22 mnt] $ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        3.8G     0  3.8G   0% /dev
tmpfs           3.8G   79M  3.7G   3% /dev/shm
tmpfs           3.8G  1.5M  3.8G   1% /run
tmpfs           3.8G     0  3.8G   0% /sys/fs/cgroup
/dev/sda1       451G  175G  253G  41% /
tmpfs           3.8G  408K  3.8G   1% /tmp
tmpfs           769M  8.0K  769M   1% /run/user/42
tmpfs           769M   28K  769M   1% /run/user/1000
/dev/mmcblk0    7.4G  4.0K  7.4G   1% /run/media/james/3980-8C72
```

- **Unmmount your device with the following commands:**



Expand the File system and Enable:

- Login as user: pi password: raspberry
- Execute the command sudo raspi-config in the terminal
- Select Expand Filesystem and press Enter
- Select OK and you will return to the main menu
- Select Advanced Options
- Select I2C and press Enter
- Select Yes and press Enter
- Select OK and press Enter
- Select Yes and press Enter
- Select OK and you will return to the main menu
- Select Finish and press Enter
- Select Yes and press Enter to reboot the Raspberry pi.

STEP 2:

Create a Python script (e.g.,smartrestroom.py) and add the following code:

```
import RPi.GPIO as GPIO
import time
import Adafruit_DHT
import requests

# Define GPIO pins for sensors
TOILET_SENSOR_PIN = 17
DHT_SENSOR_PIN = 4

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

```
    # Function to check toilet seat occupancy
    def check_toilet_seat_occupancy():
        if GPIO.input(TOILET_SENSOR_PIN):
            return "Occupied"
        else:
            return "Vacant"

    # Function to read temperature and humidity
    def read_temperature_humidity():
        sensor = Adafruit_DHT.DHT22
        humidity, temperature = Adafruit_DHT.read_retry(sensor,
DHT_SENSOR_PIN)
        if humidity is not None and temperature is not None:
            return temperature, humidity
        else:
            return None, None

    # Function to monitor air quality (you would need an
appropriate sensor for this)
    def monitor_air_quality():
        # Implement air quality monitoring logic here
        return "Good"

    # Function to send data to a remote server
    def send_data_to_server(data):
        # Replace with your server's API endpoint
        server_url = "https://your-server-endpoint.com"
        try:
            response = requests.post(server_url, json=data)
            if response.status_code == 200:
                print("Data sent successfully")
            else:
                print("Failed to send data")
        except Exception as e:
            print("Error:", str(e))

    # Main loop
    while True:
        toilet_occupancy = check_toilet_seat_occupancy()
```

```
        temperature, humidity = read_temperature_humidity()
        air_quality = monitor_air_quality()

        data = {
            "toilet_occupancy": toilet_occupancy,
            "temperature": temperature,
            "humidity": humidity,
            "air_quality": air_quality
        }

        # Send data to a remote server
        send_data_to_server(data)

        # Sleep for a certain period (e.g., 1 minute)
        time.sleep(60)
```

CODE FUNCTION EXPALNATION:

1. Importing Libraries:

RPi.GPIO: This library allows the Raspberry Pi to interface with GPIO pins.

- time: Used for creating time delays with `time.sleep()`.

Adafruit_DHT: Used for interfacing with the DHT22 sensor to read temperature and humidity.

requests: Used for making HTTP requests to send data to a remote server.

2. GPIO Pin Definitions:

TOILET_SENSOR_PIN is set to 17, which corresponds to the GPIO pin connected to a sensor that detects toilet seat occupancy.

DHT_SENSOR_PIN is set to 4, which corresponds to the GPIO pin connected to the DHT22 temperature and humidity sensor.

3. GPIO Initialization:

GPIO.setmode(GPIO.BCM): This line sets the GPIO numbering mode to Broadcom SOC channel numbers.

GPIO.setup(TOILET_SENSOR_PIN, GPIO.IN): It sets up the TOILET_SENSOR_PIN as an input pin.

4. Functions:

check_toilet_seat_occupancy(): This function checks the state of the toilet seat occupancy sensor. If the sensor reads a high signal, it returns "Occupied"; otherwise, it returns "Vacant."

read_temperature_humidity(): This function reads temperature and humidity data from the DHT22 sensor. It uses the Adafruit DHT library and retries the reading if it fails. It returns the temperature and humidity values or `None` if the reading fails.

monitor_air_quality(): This function is a placeholder for monitoring air quality. You would need to implement the actual logic for air quality monitoring and use an appropriate sensor.

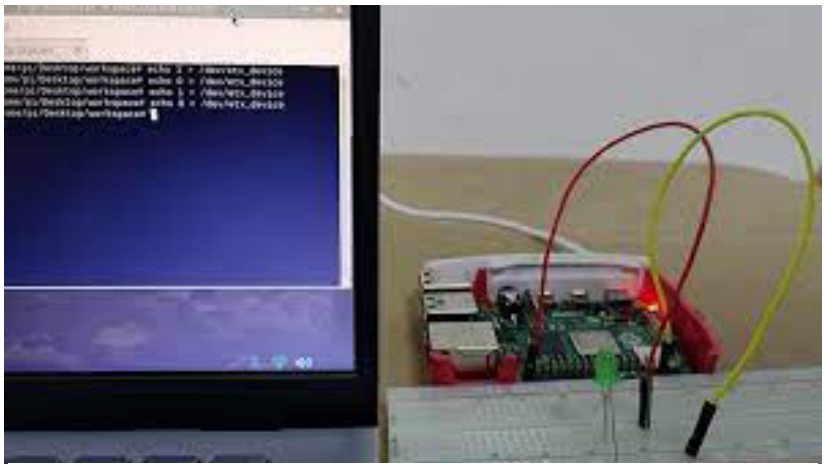
send_data_to_server(data): This function sends a dictionary (`data`) containing the collected data to a remote server. It sends an HTTP POST request to the specified server endpoint. It also checks the response status code and prints whether the data was sent successfully or not.

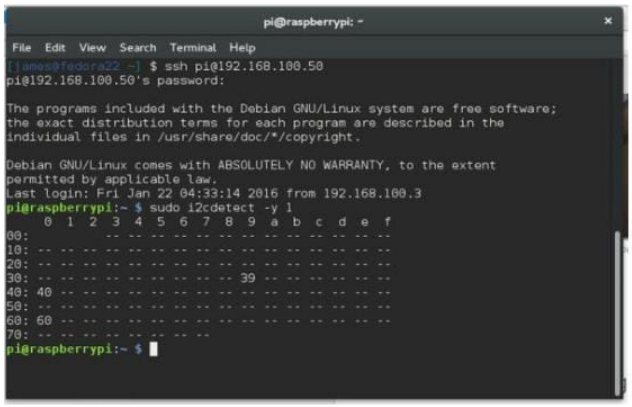
STEP 3:(Save the script and run it)

python tempThe script will continuously read the temperature and humidity values from the DHT sensor and print them to the terminal. Press Ctrl+C to stop the program.

Remember that the DHT sensors may have different pinouts and require different parameters, so make sure to adjust the code accordingly. Additionally, double-check your wiring and make sure you have the necessary permissions to access GPIO pins (usually, you need to run the script with superuser privileges or add your user to the group).

YOU SHOULD SEE THE FOLLOWING OUTPUT AS GIVEN





Once you finish the wiring and sensor placement your device should look something like this.