.

IoT in Healthcare

CHAN JUN QI JULIUS

1.0 Introduction

The fast expansion of Internet of Things (IoT) technology is transforming industries and communities throughout the world, opening up new potential for innovation and efficiency. Imagine a future in which commonplace things are networked, forming a network capable of addressing societal challenges.

This report proposes and develops a prototype that leverages the Internet of Things (IoT) technology to address healthcare difficulties. IoT has transformed the way industries work, providing increased efficiency, real-time monitoring, and data-driven decision-making. In healthcare, IoT applications may dramatically enhance patient care, operational efficiency, and safety, in line with global goals such as the United Nations' Sustainable Development Goals (SDGs).

The report offers a full overview of the project, beginning with background research on IoT's function in healthcare. It contains an analysis of numerous case studies that demonstrate the actual uses of IoT in fields like as remote patient monitoring, emergency response systems, and smart medical waste management.

Following that, the prototype's design and architecture are examined, with a focus on the conceptual model and technical architecture that support the system. The implementation section describes the development process, including simulations and sensor integration. Finally, the study provides an evaluation of the prototype's efficacy, covering use cases and potential operational and security concerns, as well as recommendations.

2.0 Background Study

2.1 IoT in Healthcare

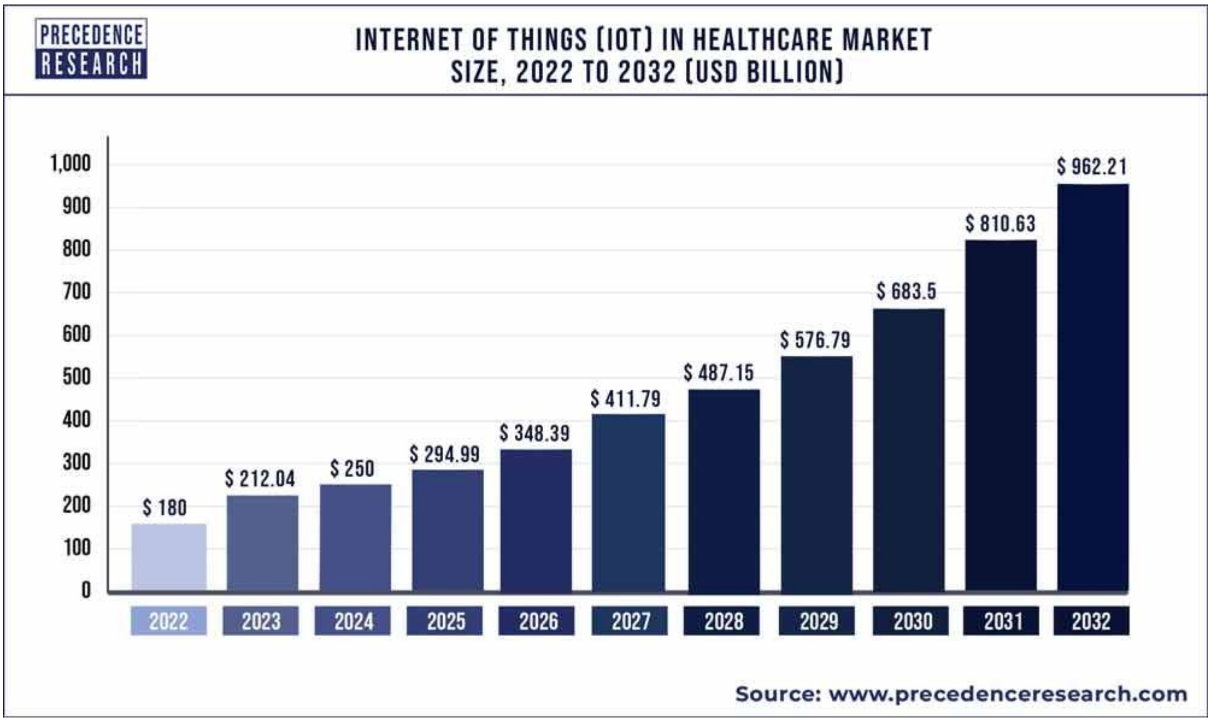


Figure 1: IoT in Healthcare Market, 2022 to 2032 (Precedence Research, 2023)

As shown in Figure 1, the global Internet of Things (IoT) in the healthcare market size was valued at USD 180 billion in 2022 and it is expected to reach around USD 962.21 billion by 2032 with a compound annual growth rate (CAGR) of 18.3% during the forecast period 2023 to 2032. (Precedence Research, 2023).

The healthcare domain has a large impact and aligns with numerous SDGs, particularly Goal 3: Good Health and Well-Being. The industry has a set of challenges that can be addressed where integrating Internet of Things (IoT) technologies in the healthcare industry represents a digital transformation with tangible benefits.

2.2 Case Studies

2.2.1 Case Study 1: Remote Patient Health Monitoring

Before, the only way for a patient to speak with a doctor was via appointments & through tele until IoT technology emerged. Hospitals & doctors lacked a consistent method for monitoring patients' well-being & providing them with appropriate guidance. (Medium, 2021). The key issues include restricted access to care in distant places and fragmented health data caused by several providers.

The implementation of Remote Patient Monitoring (RPM) has already addressed these issues by using physical sensors in IoT devices to read and communicate patient data such as heart rate to a hospital system. However, most RPM uses wireless and mobile networks to send data from patients to healthcare practitioners. Network connectivity issues, data loss, and transmission delays can all have an impact on RPM's efficacy.

A patient in a rural area with poor network coverage may experience intermittent data transmission, leading to gaps in monitoring and delayed responses to potential health issues. (Ddoccla, 2024).

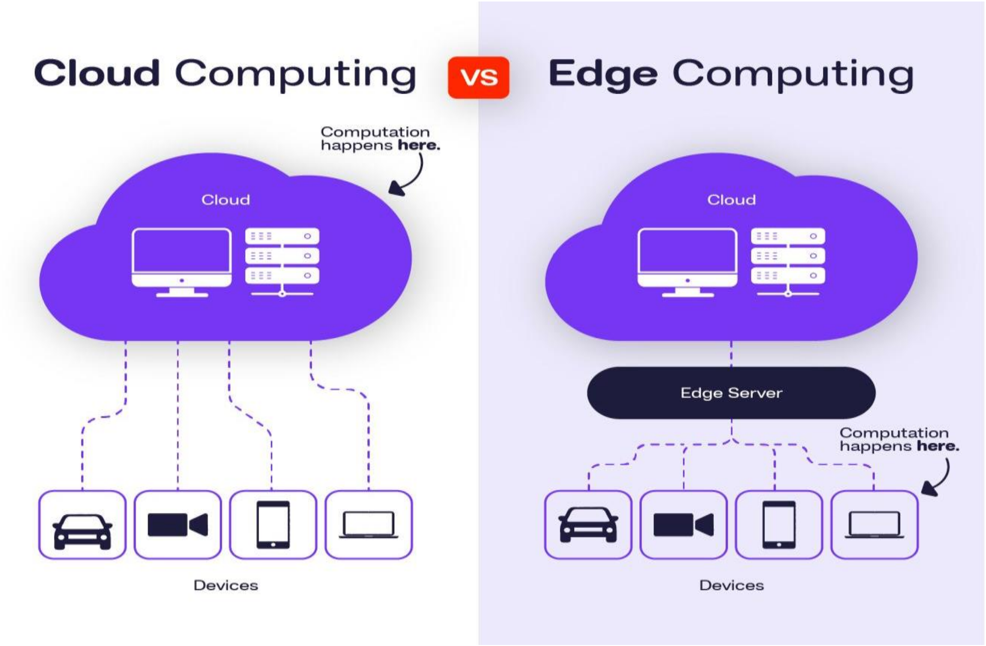


Figure 2: Cloud Computing vs Edge Computing (Aron Wagner, 2023)

A proposed improvement for assuring reliable data transfer in RPM is to integrate edge computing-supported devices such as Raspberry Pi with its sensor, which processes and stores health data locally, reducing reliance on constant network access. Edge computing also has scalability and versatility enabling continuous patient monitoring.

2.2.2 Case Study 2: Emergency Cardiac Response System

The occurrence of Sudden Cardiac Death (SCD) among adults in the general population varies between 40 and 100 out of every 100,000 individuals. SCD causes the deaths of approximately 4.25 million people worldwide yearly, and about 250,000 to 300,000 humans die each year in the United States because of SC. (J.R. Velázquez, 2021).

Sudden Cardiac Death (SCD) can be lethal in minutes if there are no paramedics or an emergency response system. Individuals who are often isolated or do not have an emergency contact are more vulnerable due to restricted access to urgent assistance.

An ideal IoT system would combine ECG sensors with a communication mechanism that uses GSM/LTE modules for real-time data transfer and cloud-based platforms for processing and sending alerts.

This design assures that emergency services are contacted as soon as SCD is expected, regardless of where the patient is. The modules provide dependable and extensive coverage, while cloud platforms allow interaction with emergency response systems, closing the gap in timely intervention.

2.2.3 Case Study 3: Smart Medical Waste Management

High-income countries generate on average up to 0.5 kg of hazardous waste per hospital bed per day; while low-income countries generate on average 0.2 kg. About 85% is general, non-hazardous waste comparable to domestic waste. The remaining 15% is considered hazardous material that may be infectious, chemical, or radioactive. (World Health Organization, 2018).

Medical waste is frequently toxic and infectious, posing environmental and health problems. Hospitals, clinics, and other medical institutions create a sheer volume of garbage, ranging from wasted syringes to infected bandages.

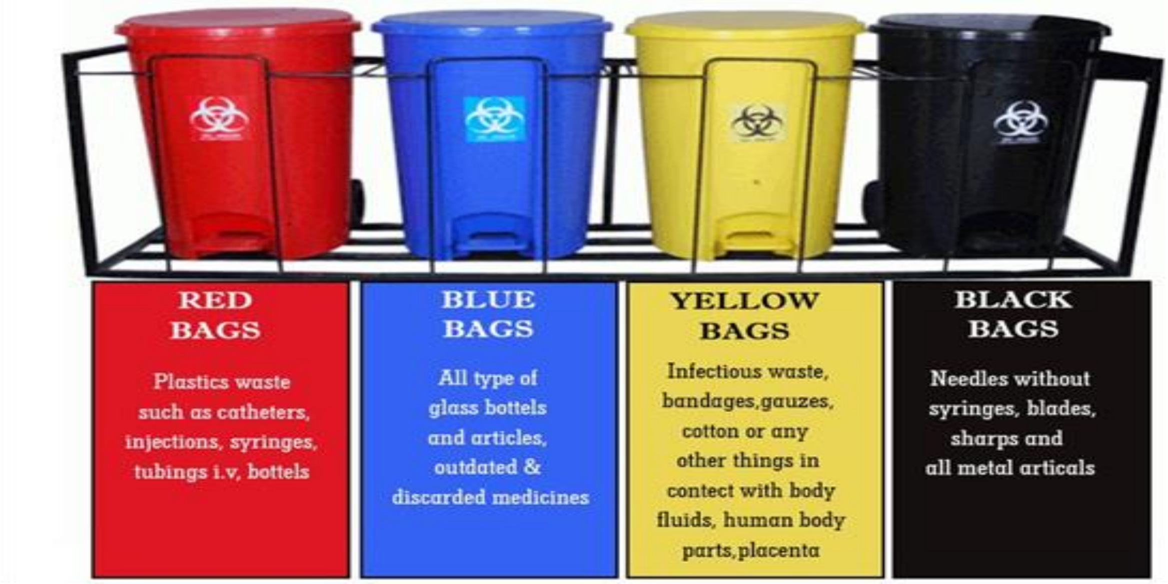

RED BAGS
Plastics waste such as catheters, injections, syringes, tubings i.v, bottels
BLUE BAGS
All type of glass bottels and articles, outdated & discarded medicines
YELLOW BAGS
Infectious waste, bandages,gauzes, cotton or any other things in contect with body fluids, human body parts,placenta
BLACK BAGS
Needles without syringes, blades, sharps and all metal articals

Figure 3: Medical Waste Management Bins (Medium, 2022)

Coloured bins shown in Figure 3 are still commonly used in healthcare facilities for the segregation and disposal of various types of medical waste, rather than more advanced systems, which introduces key issues including limited real-time monitoring, safety and hygiene concerns and even littering if the bins are filled.

IoT-enabled sensors, such as ultrasonic fill level sensors and MQ-135 gas sensors, may be integrated into garbage bins and disposal trucks. These sensors provide real-time data to a centralized system using wireless protocols such as Zigbee, or LoRA, which is powered by an IoT gateway such as the Raspberry Pi allowing hospital administrators to maximize waste management while reducing human labor.

The system may also leverage automation to make decisions like sending collection vehicles and warning about dangerous gasses to dashboards.

2.2.4 Case Study 4: Predictive Maintenance of Medical Equipment

Medical devices contain pumps, filters, and parts that have a certain life cycle and need to be replaced at intervals. Typically, hospital staff have technicians on-site to check these machines and their components to make sure they're working properly, but if an issue is missed and the machine breaks, it can cause downtime and disruption to patient treatments. (Hilbelink, 2021).

Cloud IoT technologies can offer a reliable solution for properly managing medical devices. Usage sensors measure operational hours and device performance, whilst environmental sensors monitor temperature and humidity to guarantee proper operation.

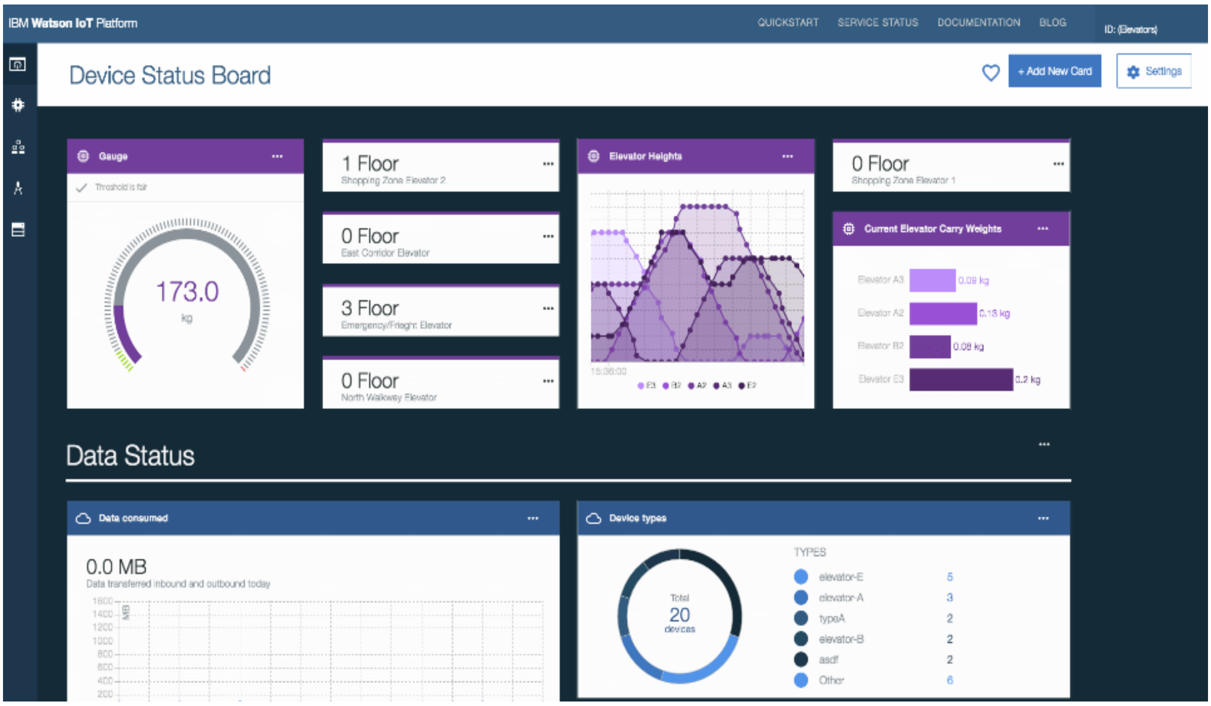


Figure 4: IoT Watson device monitoring dashboard (G2, n.d.)

Proximity sensors can detect both device status and usage. These sensors can send data to cloud platforms or dashboards as shown in Figure 4 for real-time analysis. The system forecasts when equipment will require maintenance or replacement, decreasing the need for manual checkups and preventing unexpected issues.

2.2.5 Case Study 5: Smart Medication Adherence

Medication nonadherence is a prevalent public health problem that compromises patients' health outcomes and increases healthcare expenditures. Studies in Singapore showed that 25.7%–38.9% of patients are nonadherent. Forgetfulness was a major reason for non-adherence. (Annals, 2024).

Emerging smart medication dispensing, and adherence products have already addressed the issue by offering the options of automated dispensing, reminders, and notifications. However, Elderly or less tech-savvy patients may struggle with smart medication systems due to complex interfaces as most smart pill systems can only be interacted with through a phone application.

Figure 5: Google Nest Hub Max (Rozette Rago, 2024)

An extra integration like an Adaptive Smart Medication System with voice activation, huge displays, and customized warnings can help to solve technological adoption difficulties. This system, which includes a smart pill dispenser connects to a home care system by Wi-Fi or Bluetooth, similar to Figure 5. It may include several interaction modes to promote simplicity of use, to increase drug adherence.

3.0 Prototype Design & Architecture

3.1 Conceptual model

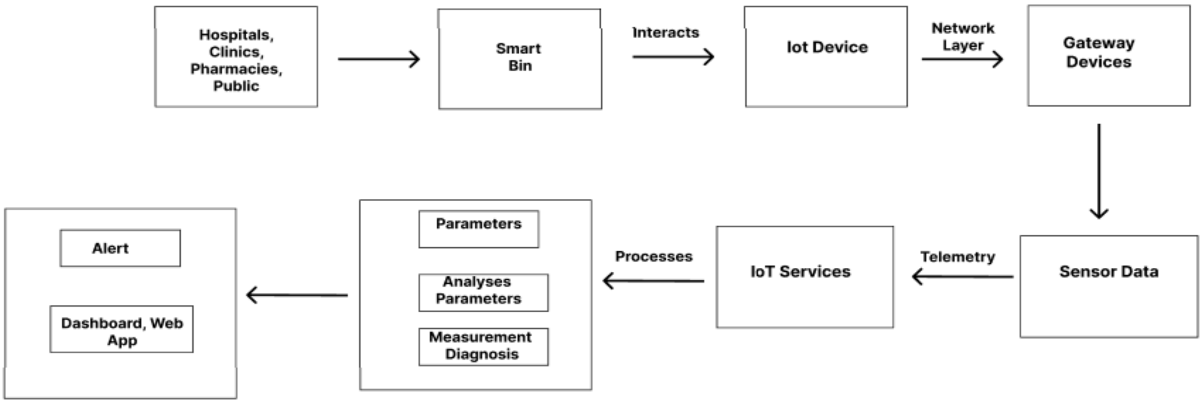


Figure 6: Conceptual Model of Prototype

Figure 6 shows the conceptual model of the system that monitors medical waste in public IoT-enabled smart bins. Data is collected in the bins through the sensor and delivered via the network to central servers and cloud services for processing. Alerts will be generated for situations like full bins and real-time data will be displayed on the web app.

3.1.1 Conceptual Framework

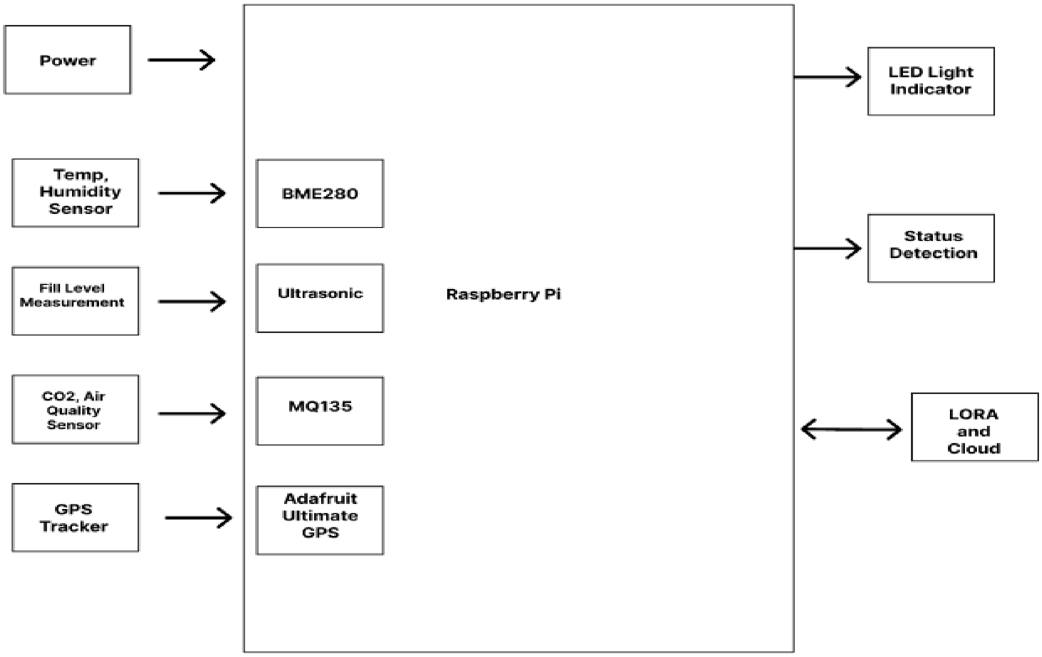


Figure 7: Conceptual Framework of Prototype

The Raspberry Pi serves as the central component of the medical smart waste management system, in Figure 7. The Raspberry Pi processes the data through the sensors and also operates an LED Light Indicator to display the power status. The Raspberry Pi then sends this data to the cloud via a LoRa module.

3.2 Technology Architecture

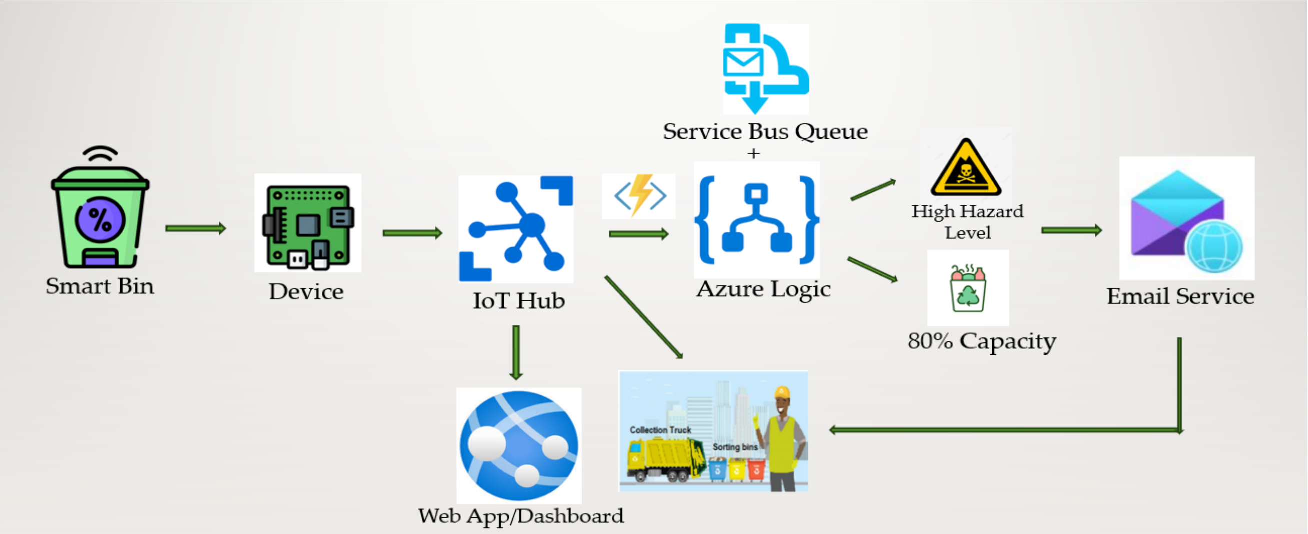


Figure 8: Technology Architecture

Figure 8 shows a potential solution that combines enabling technologies to produce an effective waste management system. The Raspberry Pi device collects data from sensors and sends it to the Azure IoT Hub via MQTT. The Azure IoT Hub manages and processes incoming data, which is subsequently displayed on the Web App/Dashboard for real-time monitoring. Azure Logic Apps delivers email notifications when certain conditions are met.

3.3 Wireframe

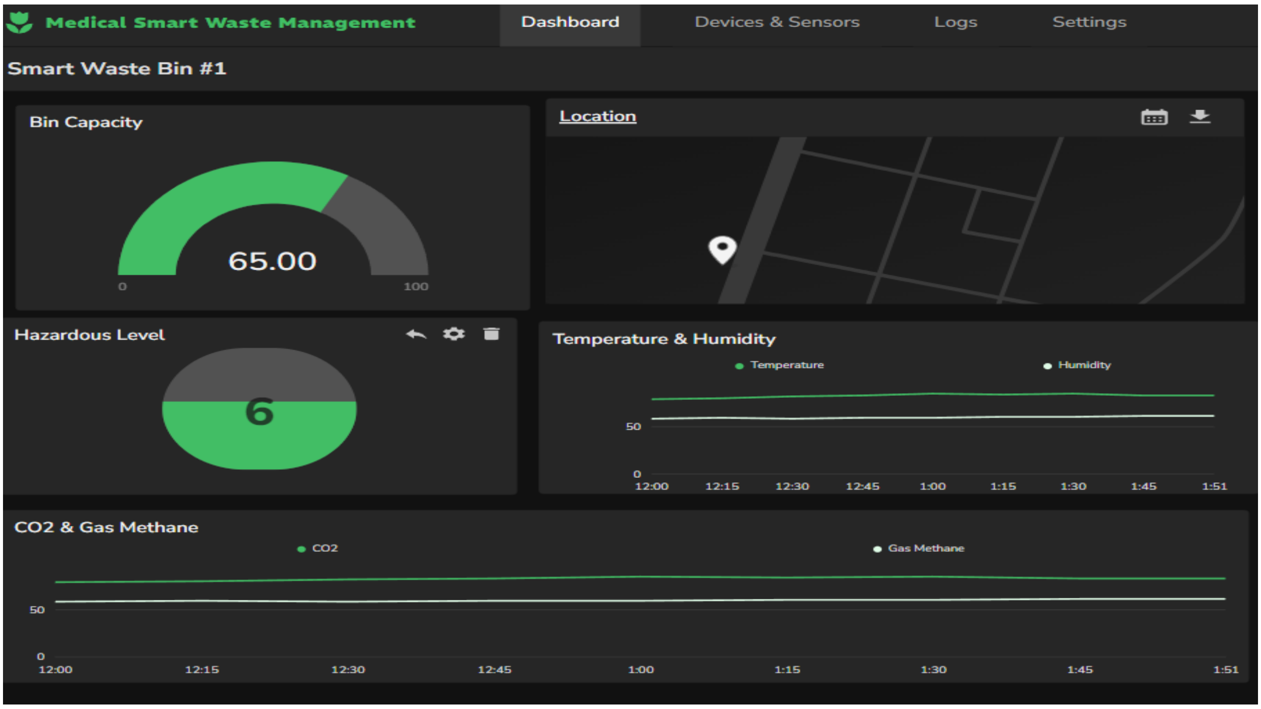
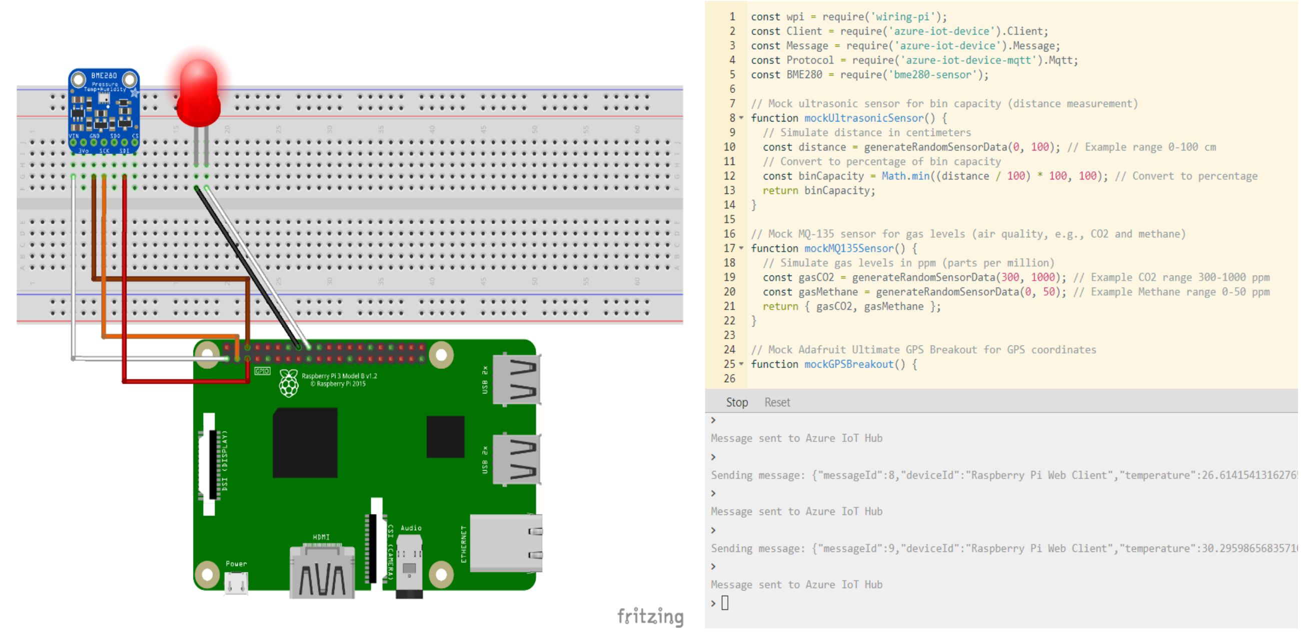


Figure 9: Wireframe of Prototype UI

4.0 Prototype Implementation

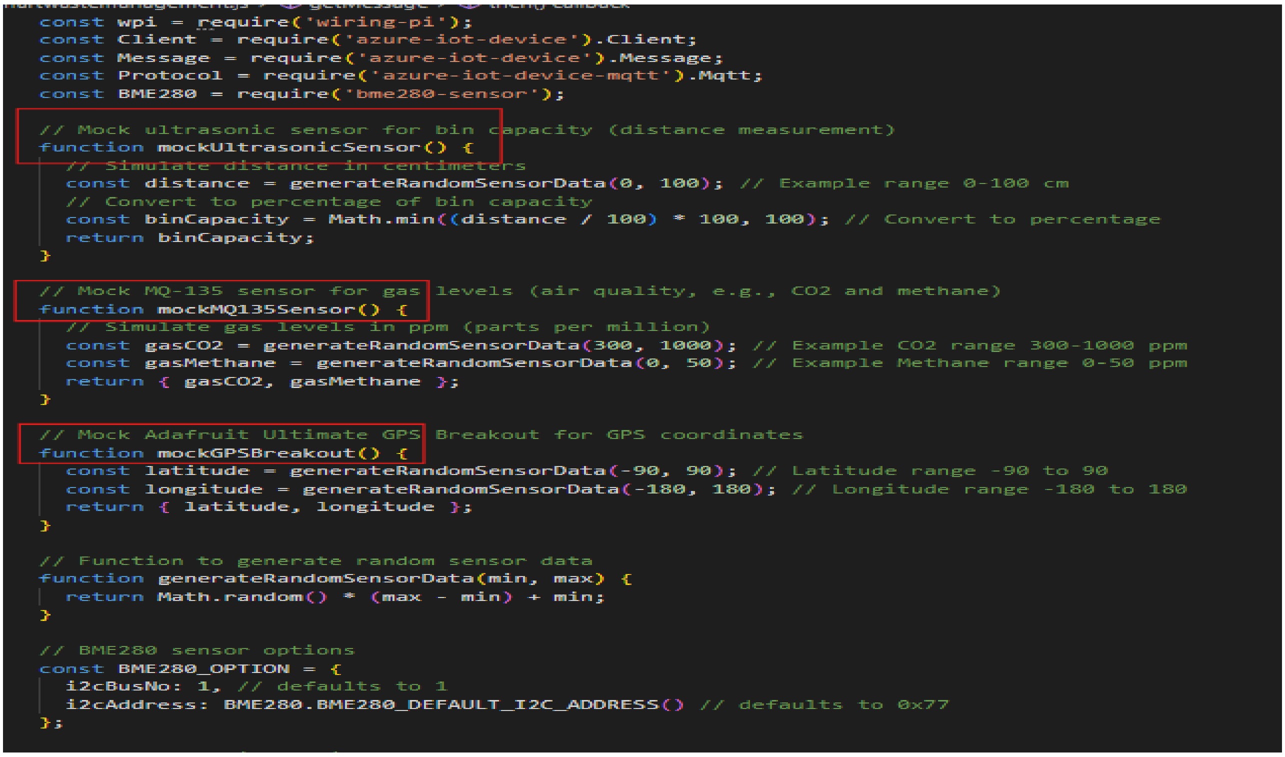


```
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const BME280 = require('bme280-sensor');

// Mock ultrasonic sensor for bin capacity (distance measurement)
function mockUltrasonicSensor() {
  // Simulate distance in centimeters
  const distance = generateRandomSensorData(0, 100); // Example range 0-100 cm
  // Convert to percentage of bin capacity
  const binCapacity = Math.min((distance / 100) * 100, 100); // Convert to percentage
  return binCapacity;
}

// Mock MQ-135 sensor for gas levels (air quality, e.g., CO2 and methane)
function mockMQ135Sensor() {
  // Simulate gas levels in ppm (parts per million)
  const gasCO2 = generateRandomSensorData(300, 1000); // Example CO2 range 300-1000 ppm
  const gasMethane = generateRandomSensorData(0, 50); // Example Methane range 0-50 ppm
  return { gasCO2, gasMethane };
}

// Mock Adafruit Ultimate GPS Breakout for GPS coordinates
function mockGPSBreakout() {

```

```
>
Message sent to Azure IoT Hub
>
Sending message: {"messageId":8,"deviceId":"Raspberry Pi Web Client","temperature":26.6141541316276
>
Message sent to Azure IoT Hub
>
Sending message: {"messageId":9,"deviceId":"Raspberry Pi Web Client","temperature":30.2959865683571
>
Message sent to Azure IoT Hub
>
```

Figure 10: Raspberry Pi Azure IoT Online Simulator

The prototype was created with an online Raspberry Pi simulator with JavaScript, which sent sensor data to Azure IoT Hub via a connection string and MQTT protocol. The simulator's key feature includes the ability to send telemetry messages to Azure IoT Hub as shown in Figure 10 and provide real-time data to the dashboard web app in Appendix 1 – 3.



```
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const BME280 = require('bme280-sensor');

// Mock ultrasonic sensor for bin capacity (distance measurement)
function mockUltrasonicSensor() {
  // Simulate distance in centimeters
  const distance = generateRandomSensorData(0, 100); // Example range 0-100 cm
  // Convert to percentage of bin capacity
  const binCapacity = Math.min((distance / 100) * 100, 100); // Convert to percentage
  return binCapacity;
}

// Mock MQ-135 sensor for gas levels (air quality, e.g., CO2 and methane)
function mockMQ135Sensor() {
  // Simulate gas levels in ppm (parts per million)
  const gasCO2 = generateRandomSensorData(300, 1000); // Example CO2 range 300-1000 ppm
  const gasMethane = generateRandomSensorData(0, 50); // Example Methane range 0-50 ppm
  return { gasCO2, gasMethane };
}

// Mock Adafruit Ultimate GPS Breakout for GPS coordinates
function mockGPSBreakout() {
  const latitude = generateRandomSensorData(-90, 90); // Latitude range -90 to 90
  const longitude = generateRandomSensorData(-180, 180); // Longitude range -180 to 180
  return { latitude, longitude };
}

// Function to generate random sensor data
function generateRandomSensorData(min, max) {
  return Math.random() * (max - min) + min;
}

// BME280 sensor options
const BME280_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: BME280.BME280_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};
```

Figure 11: Mock Sensors of Prototype

The provided code in Figure 11 contains mock sensors that replicate sensor data best utilized for a waste management system prototype and also incorporated into Raspberry PI Device. These IoT components simulate genuine sensors' behaviour through a randomly generated value.

BME280 Sensor

- Temperature Monitoring: Tracks bin or surrounding area temperature to detect issues like fire risks.
- Humidity Monitoring: Measures humidity to monitor waste conditions and prevent Mold or bacterial growth.

Ultrasonic Sensor

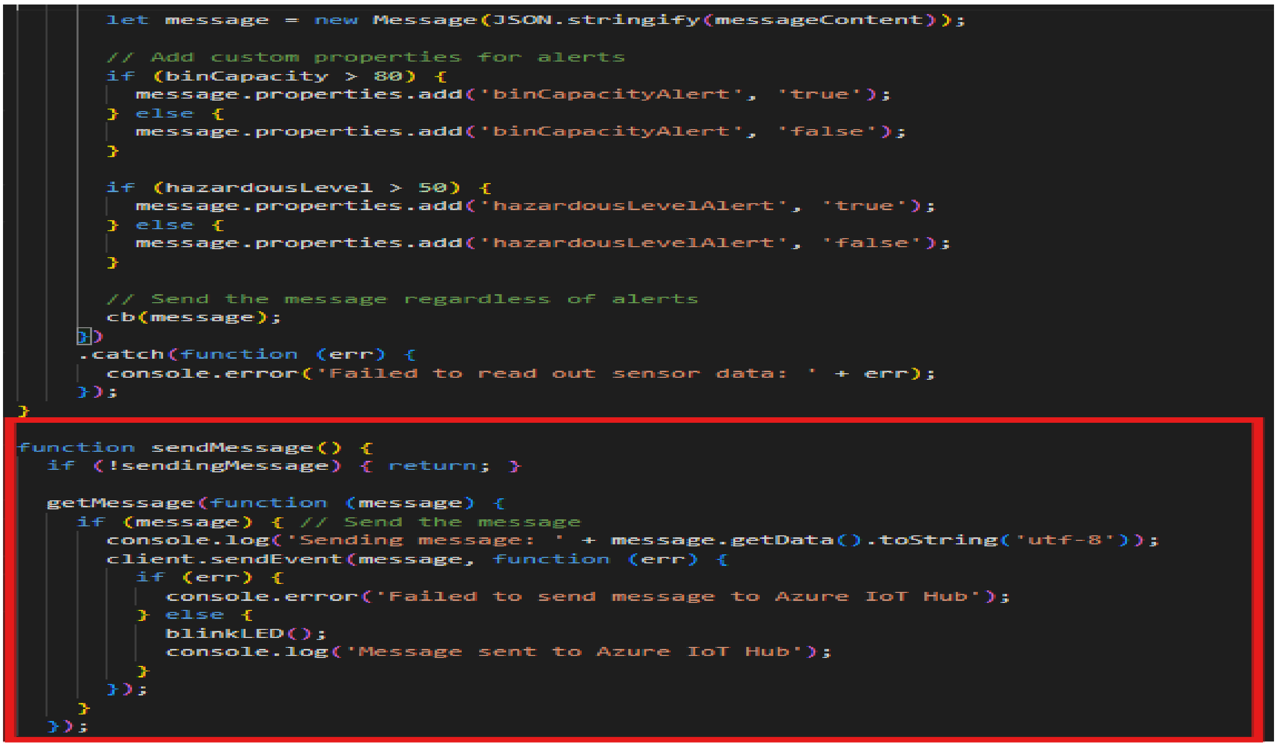
- Bin Capacity: Determines the distance from the sensor to the waste level.
- Detect Overflows: Helps in identifying when a bin is nearing or has reached, reducing overflow issues.

MQ135 Sensor

- Measure Air Quality: Detects levels of gases such as CO2, which can indicate the presence of hazardous or decomposing waste.
- Monitor Gas Concentrations: Provides data on air pollution and potential safety hazards within or around the waste bin.

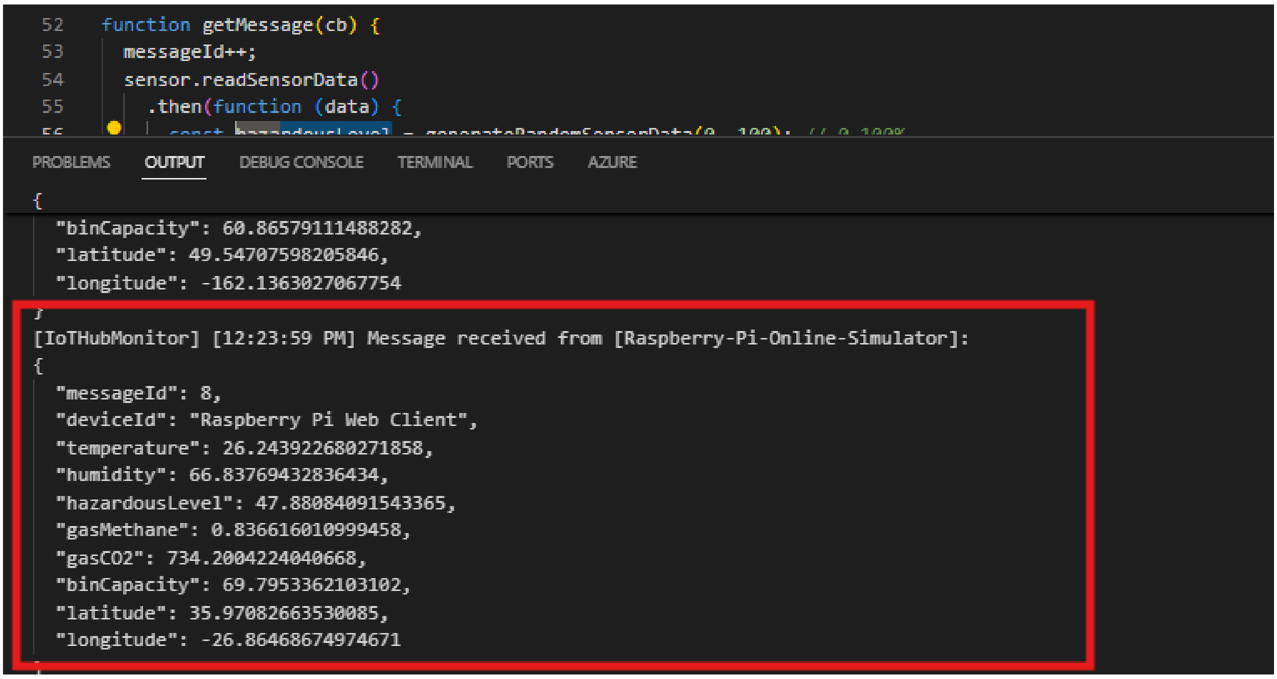
Adafruit Ultimate GPS Sensor

- GPS tracking: Provides accurate latitude, longitude for location and real-time tracking.



```
      let message = new Message(JSON.stringify(messageContent));

      // Add custom properties for alerts
      if (binCapacity > 80) {
        message.properties.add('binCapacityAlert', 'true');
      } else {
        message.properties.add('binCapacityAlert', 'false');
      }

      if (hazardousLevel > 50) {
        message.properties.add('hazardousLevelAlert', 'true');
      } else {
        message.properties.add('hazardousLevelAlert', 'false');
      }

      // Send the message regardless of alerts
      cb(message);
    })
    .catch(function (err) {
      console.error('Failed to read out sensor data: ' + err);
    });
}

function sendMessage() {
  if (!sendingMessage) { return; }

  getMessage(function (message) {
    if (message) { // Send the message
      console.log('Sending message: ' + message.getData().toString('utf-8'));
      client.sendEvent(message, function (err) {
        if (err) {
          console.error('Failed to send message to Azure IoT Hub');
        } else {
          blinkLED();
          console.log('Message sent to Azure IoT Hub');
        }
      });
    }
  });
```

Figure 12: Telemetry Data Sent to Azure IoT Hub

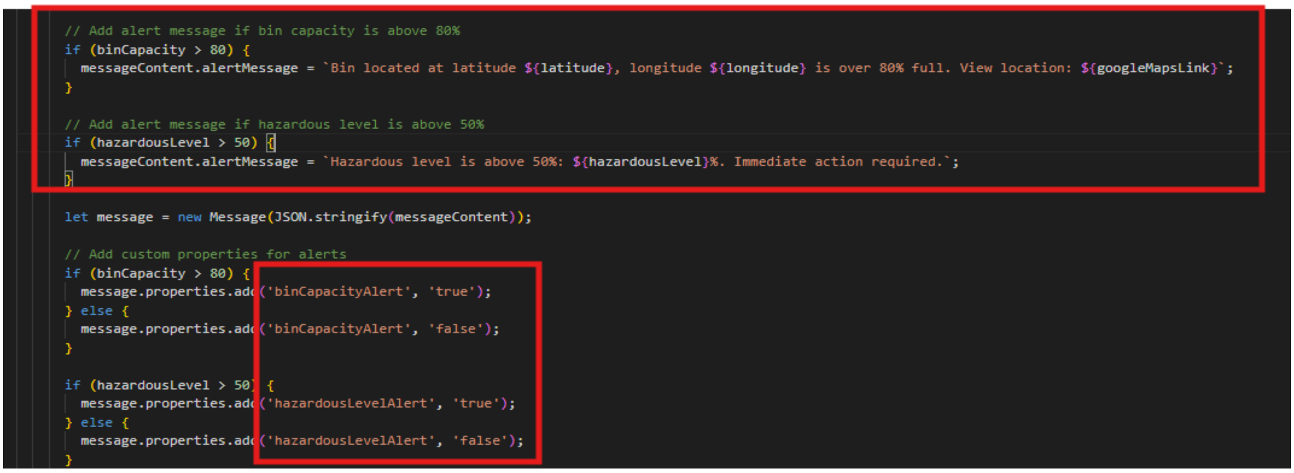


```
function getMessage(cb) {
  messageId++;
  sensor.readSensorData()
    .then(function (data) {
      const hazardousLevel = generateRandomSensorData(0, 100); // 0-100%
```

```
{
  "binCapacity": 60.86579111488282,
  "latitude": 49.54707598205846,
  "longitude": -162.1363027067754
}
[IoTHubMonitor] [12:23:59 PM] Message received from [Raspberry-Pi-Online-Simulator]:
{
  "messageId": 8,
  "deviceId": "Raspberry Pi Web Client",
  "temperature": 26.243922680271858,
  "humidity": 66.83769432836434,
  "hazardousLevel": 47.88084091543365,
  "gasMethane": 0.836616010999458,
  "gasCO2": 734.2004224040668,
  "binCapacity": 69.7953362103102,
  "latitude": 35.97082663530085,
  "longitude": -26.86468674974671
```

Figure 13: Telemetry Data Parameters

The code in Figure 12 uses ‘client.sendEvent’ to send the message to Azure IoT Hub. A success message is logged and the blinkLED feature is activated for visual confirmation if the message is transmitted successfully. message obtained from the simulator will include the metrics seen on IoT Hub in Appendix 4 together with the sensor readings of the parameters shown in Figure 13.



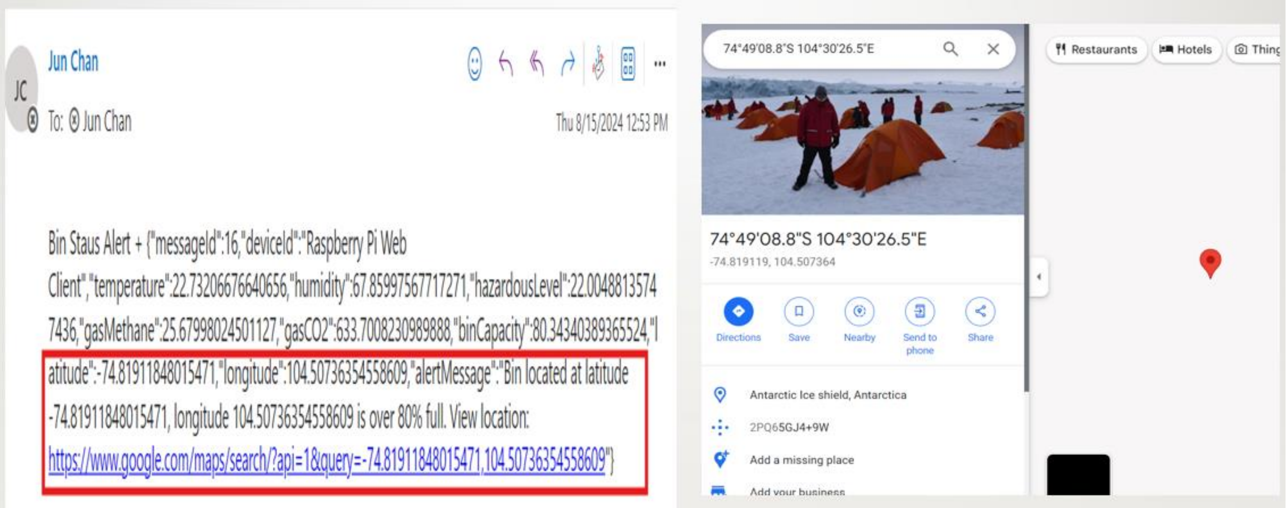
```
// Add alert message if bin capacity is above 80%
if (binCapacity > 80) {
  messageContent.alertMessage = `Bin located at latitude ${latitude}, longitude ${longitude} is over 80% full. View location: ${googleMapsLink}`;
}

// Add alert message if hazardous level is above 50%
if (hazardousLevel > 50) {
  messageContent.alertMessage = `Hazardous level is above 50%: ${hazardousLevel}%. Immediate action required.`;
}

let message = new Message(JSON.stringify(messageContent));

// Add custom properties for alerts
if (binCapacity > 80) {
  message.properties.add('binCapacityAlert', 'true');
} else {
  message.properties.add('binCapacityAlert', 'false');
}

if (hazardousLevel > 50) {
  message.properties.add('hazardousLevelAlert', 'true');
} else {
  message.properties.add('hazardousLevelAlert', 'false');
}
```

Figure 14: Alert Messages

A function of alert messages based on sensor data for dangerous levels and bin capacity is shown in Figure 14. An email notice with the location of the bin and a Google Maps will be sent if the dangerous level is above 50% or the bin capacity surpasses 80%, indicating that prompt action is necessary.

The binCapacityAlert property is set to 'true' when the bin capacity exceeds 80%, and the hazardousLevelAlert property is set to 'true' when the hazardous level goes beyond 50% to indicate specific conditions for the trigger.

Service Bus Queue was utilized in Appendix 5, for handling message queuing and routing. Azure Logic in Appendix 6 triggers the alerts based on the custom properties.



Jun Chan

JC

To: Jun Chan

Thu 8/15/2024 12:53 PM

Bin Staus Alert + {"messageId":16,"deviceId":"Raspberry Pi Web Client","temperature":22.73206676640656,"humidity":67.85997567717271,"hazardousLevel":22.00488135747436,"gasMethane":25.67998024501127,"gasCO2":633.7008230989888,"binCapacity":80.34340389365524,"latitude":-74.81911848015471,"longitude":104.50736354558609,"alertMessage":"Bin located at latitude -74.81911848015471, longitude 104.50736354558609 is over 80% full. View location: https://www.google.com/maps/search/?api=1&query=-74.81911848015471,104.50736354558609"}

74°49'08.8"S 104°30'26.5"E

74°49'08.8"S 104°30'26.5"E

-74.819119, 104.507364

Antarctic Ice shield, Antarctica

2PQ65GJ4+9W

Add a missing place

Add your business

Figure 15: Bin's Location (Simulation)

When triggered, an email notification is sent to alert the personnel about the bin status. This notification in Figure 15 includes details about the bin's location requesting immediate action, such as medical waste disposal.

5.0 Prototype Evaluation

5.1 Use Cases

5.1.1 Timely Waste Disposal in Hospitals

Hospital employees, such as janitors and facilities managers, may rely on the medical smart waste management system to keep trash containers from overflowing. The system uses ultrasonic sensors to monitor bin capacity and sends an email notice when it reaches 80% full, assuring timely disposal and hygiene in important locations.

5.1.2 Protecting Health Workers from Hazardous Gases

In medical facilities, the smart waste management system checks the air quality around waste bins. If dangerous gas levels, such as CO2, exceed safe limits, the system notifies safety officers via email, allowing for prompt action to protect workers and patients.

5.1.3 Preventing Environmental Risks

Facility managers may use the system to monitor temperature and humidity levels around waste bins, preventing circumstances that may lead to mold development or fires. If environmental conditions become dangerous, alerts are issued, allowing for prompt action.

5.2 Key Measurement Metrics

The effectiveness of the prototype was measured using the following key metrics:

- Accuracy: The precision of sensor readings, including bin capacity, hazardous gas levels, and location of the device.

- Response Time: The time taken for the system to detect an issue and send an alert.

- Alert Functionality: The reliability of the alert system, email notifications, and the accuracy of the alert conditions.

- System Integration: The seamless integration of Raspberry Pi with Cloud Services and the web dashboard.

5.3 Outcome

The prototype effectively solved issues in medical waste management with sensors accurately identifying the correct parameters of the values generated and issuing alerts as required. When the threshold was exceeded, the system efficiently notified hospital personnel via email aligning with its expected outcome.

6.0 Conclusion

The project aims to improve medical waste management by integrating IoT technologies and a Raspberry Pi-based system. The main issue is the inefficient and unsafe handling of medical waste due to limited real-time monitoring and ineffective segregation methods.

The proposed solution, which uses IoT-enabled sensors, enables real-time data collection and alert notifications for full bins and hazardous conditions.

However, challenges such as compatibility limitations with Raspberry Pi sensors and Azure's limited database management resources led to the need for a more robust database solution and hardware improvements.

Future work should focus on integrating a database for enhanced data logging and analysis and developing a physical prototype with a reliable power supply for long-term deployment. Security and privacy measures are crucial, including encryption of data in transit and at rest, and adhering to best cybersecurity practices.

Overall, this project aligns with the United Nations' Goal 3: Good Health and Well-Being by improving medical waste management and contributing to safer and more efficient healthcare environments.

References

Cloud Computing vs. Edge Computing: Which Is Best for Your Business? (n.d.). American Cloud. https://americancloud.com/blog/cloud-computing-vs-edge-computing-which-is-best-for-your-business

Hilbelink, L. (2022, September 20). *Remote and predictive maintenance of medical equipment in the era of connected healthcare*. Eurotech. https://www.eurotech.com/blog/remote-and-predictive-maintenance-of-medical-equipment-in-the-era-of-connected-healthcare/

Hood, C., Sikka, N., Van, C. M., & Mossburg, S. (2023, March 15). *Remote patient monitoring*. PSNet. https://psnet.ahrq.gov/perspective/remote-patient-monitoring#:~:text=RPM%20allows%20you%20to%20better,the%20data%20that%20are%20transmitted.

Insights | The Challenges with Remote Patient Monitoring (RPM) | General. (n.d.). https://www.doccla.com/post/challenges-with-remote-patient-monitoring#:~:text=Ensuring%20consistent%20and%20reliable%20data,compromise%20the%20effectiveness%20of%20RPM.

Ordr. (2024, July 25). *10 Ways Internet of Things(IoT) Impacts Healthcare Security - Ordr*. https://ordr.net/article/iot-healthcare-examples

Patient's degree of adherence, challenges & preferences towards medicine taking (PACT) in Singapore. (2024, March 27). Annals. https://annals.edu.sg/patients-degree-of-adherence-challenges-preferences-towards-medicine-taking-pact-in-singapore/

Peranzo, P. (n.d.). 8 Sectors That Can Benefit the Most from IoT Development in 2024. *Imaginovation | Top Web & Mobile App Development Company Raleigh*. https://imaginovation.net/blog/sectors-benefit-from-iot-development/

Rehman, S. U., Sadek, I., Huang, B., Manickam, S., & Mahmoud, L. N. (2024a). IoT-based emergency cardiac death risk rescue alert system. *MethodsX*, *13*, 102834. https://doi.org/10.1016/j.mex.2024.102834

Rehman, S. U., Sadek, I., Huang, B., Manickam, S., & Mahmoud, L. N. (2024b). IoT-based emergency cardiac death risk rescue alert system. *MethodsX*, *13*, 102834. https://doi.org/10.1016/j.mex.2024.102834

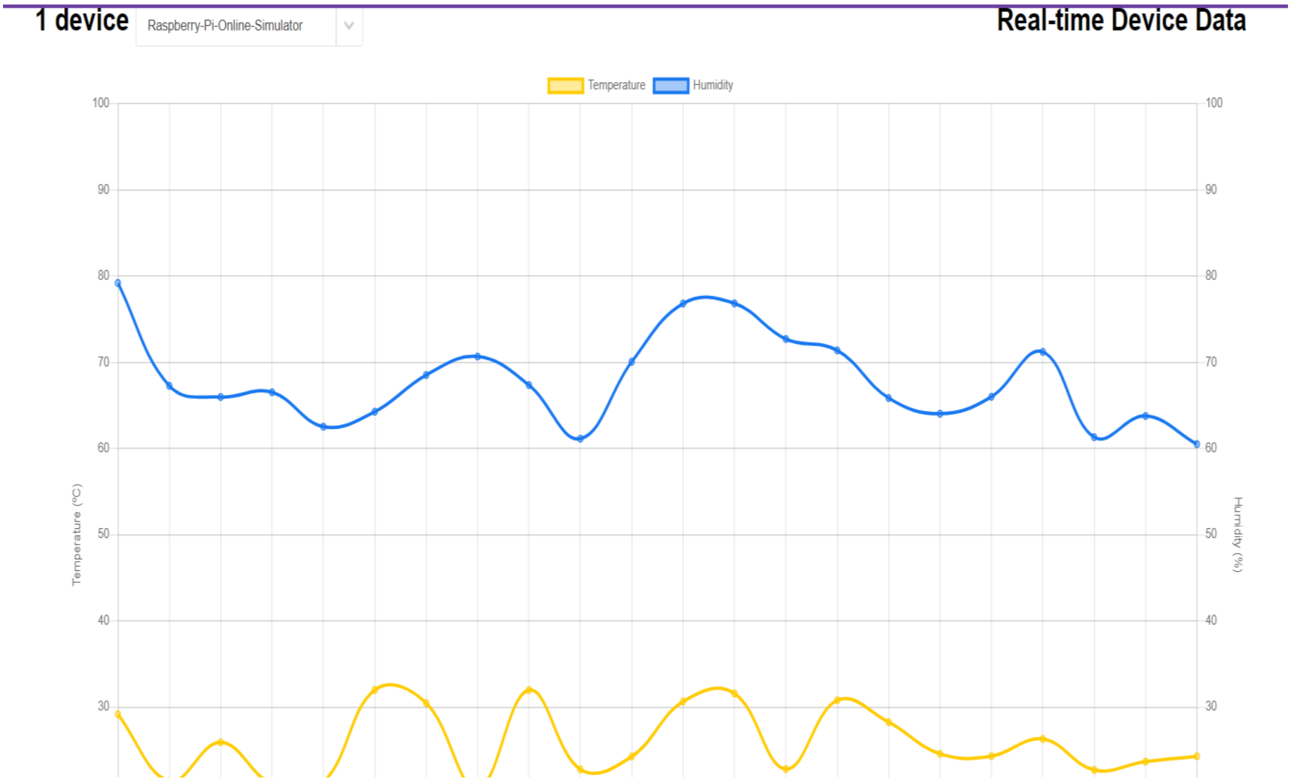
Research, P. (2023, August 25). *Internet of Things (IOT) in Healthcare Market Report 2023-2032*. https://www.precedenceresearch.com/internet-of-things-in-healthcare-market

Sonnier, W. (2023, November 30). *Leveraging IoT in medical waste management*. MedTech Intelligence. https://medtechintelligence.com/column/leveraging-iot-in-medical-waste-management/

Stfalcon.com. (2023, December 15). The impact of the internet of things in healthcare - Stfalcon.com - medium. *Medium*. https://stfalconcom.medium.com/the-impact-of-the-internet-of-things-in-healthcare-c068fee4bd09#:~:text=Before%2C%20the%20only%20way%20for,providing%20them%20with%20appropriate%20guidance.

World Health Organization: WHO. (2018, February 8). *Health-care waste*. https://www.who.int/news-room/fact-sheets/detail/health-care-waste

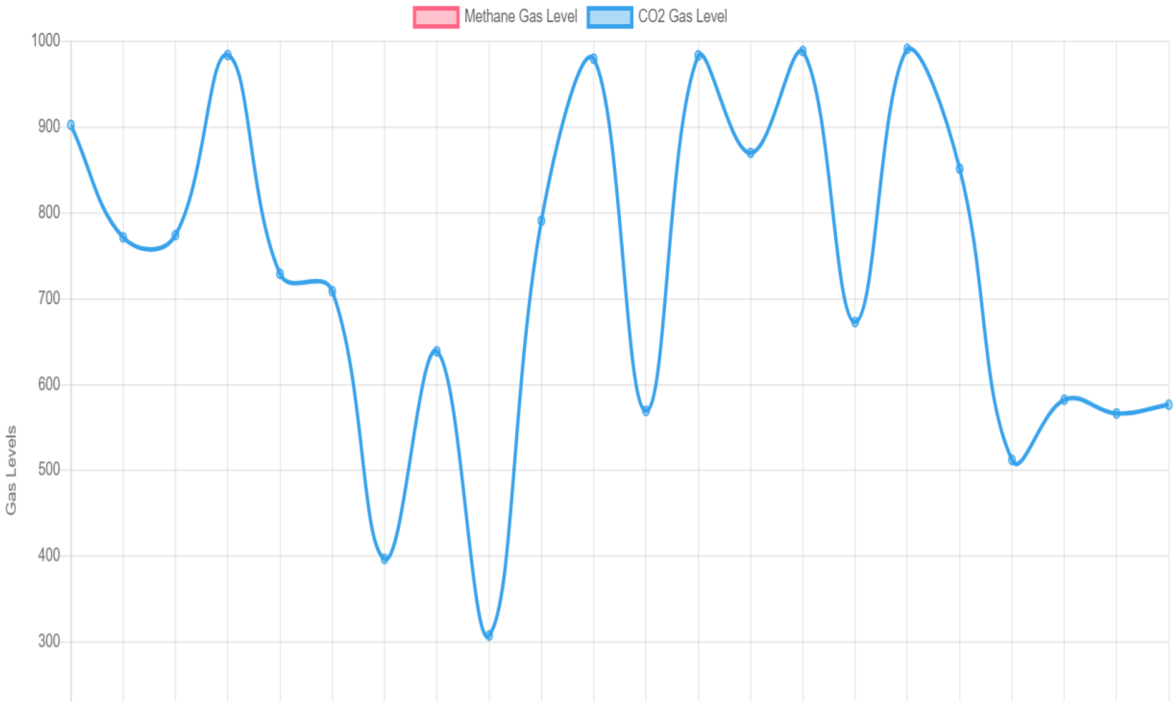
Appendix



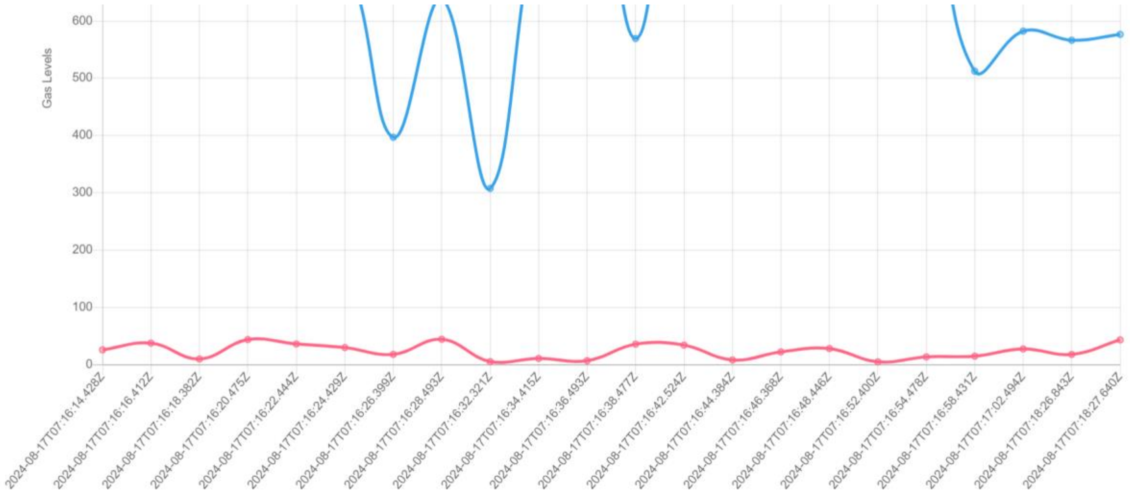
1 device Raspberry-Pi-Online-Simulator

Real-time Device Data

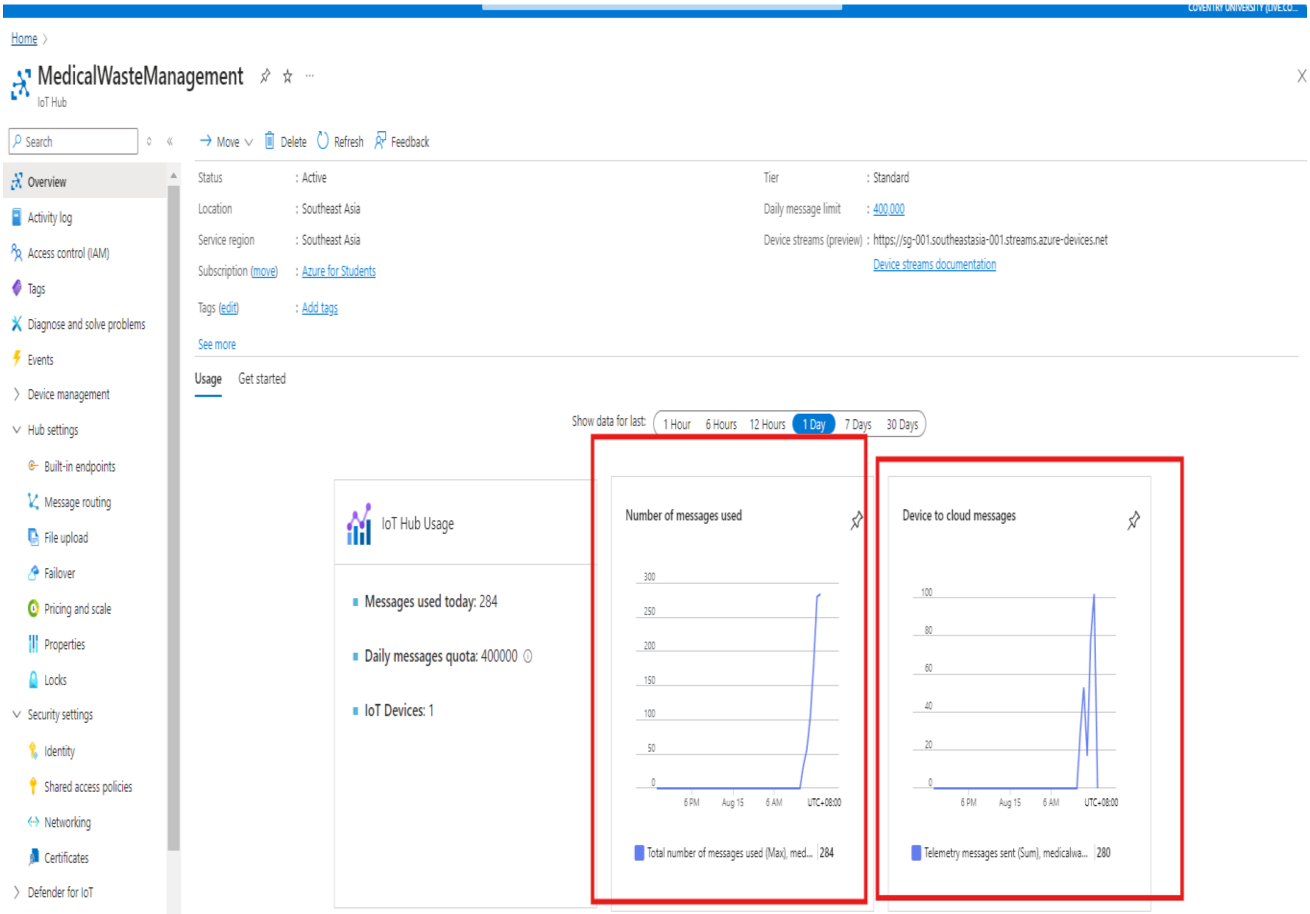
Appendix 1: Real-Time Data Analytics Dashboard



Appendix 2: Real-Time Data Dashboard (CO2 Gas Level)



Appendix 3: Real-Time Data Dashboard (Menthane Gas Level)



MedicalWasteManagement

IoT Hub

Status : Active

Location : Southeast Asia

Service region : Southeast Asia

Subscription (move) : Azure for Students

Tags (edit) : Add tags

Tier : Standard

Daily message limit : 400,000

Device streams (preview) : https://sg-001.southeastasia-001.streams.azure-devices.net

Device streams documentation

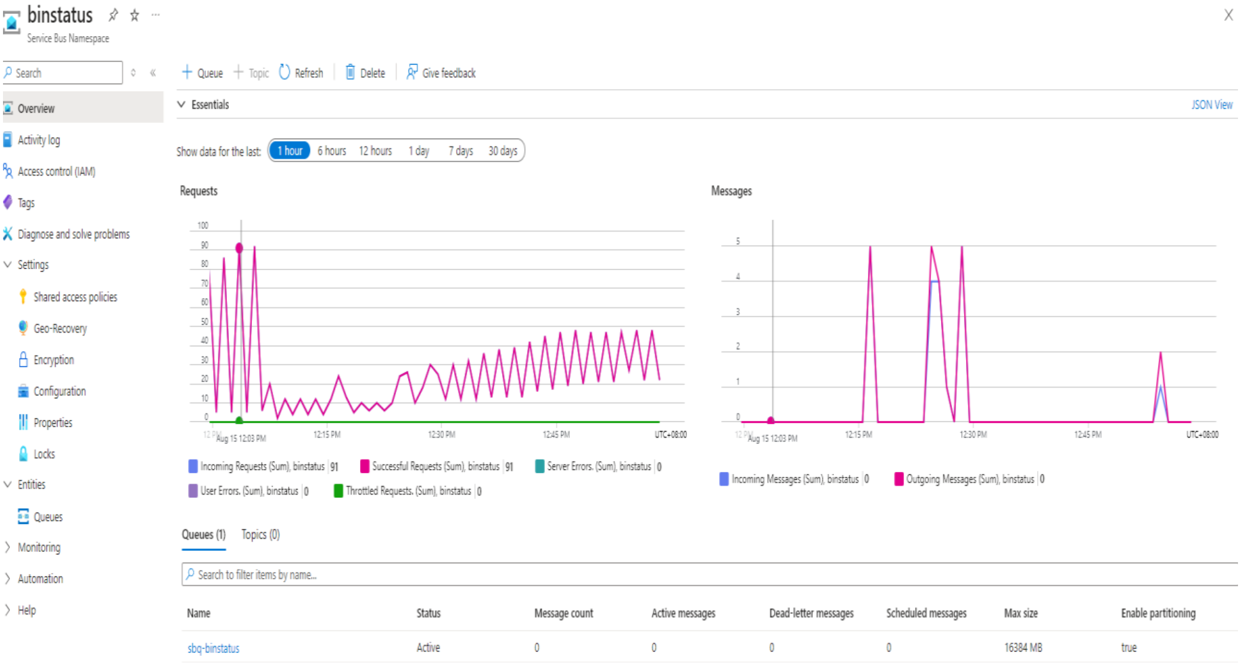
IoT Hub Usage

- Messages used today: 284
- Daily messages quota: 400000
- IoT Devices: 1

Total number of messages used (Max), med... 284

Telemetry messages sent (Sum), medicalwa... 280

Appendix 4: Azure IoT Hub



binstatus

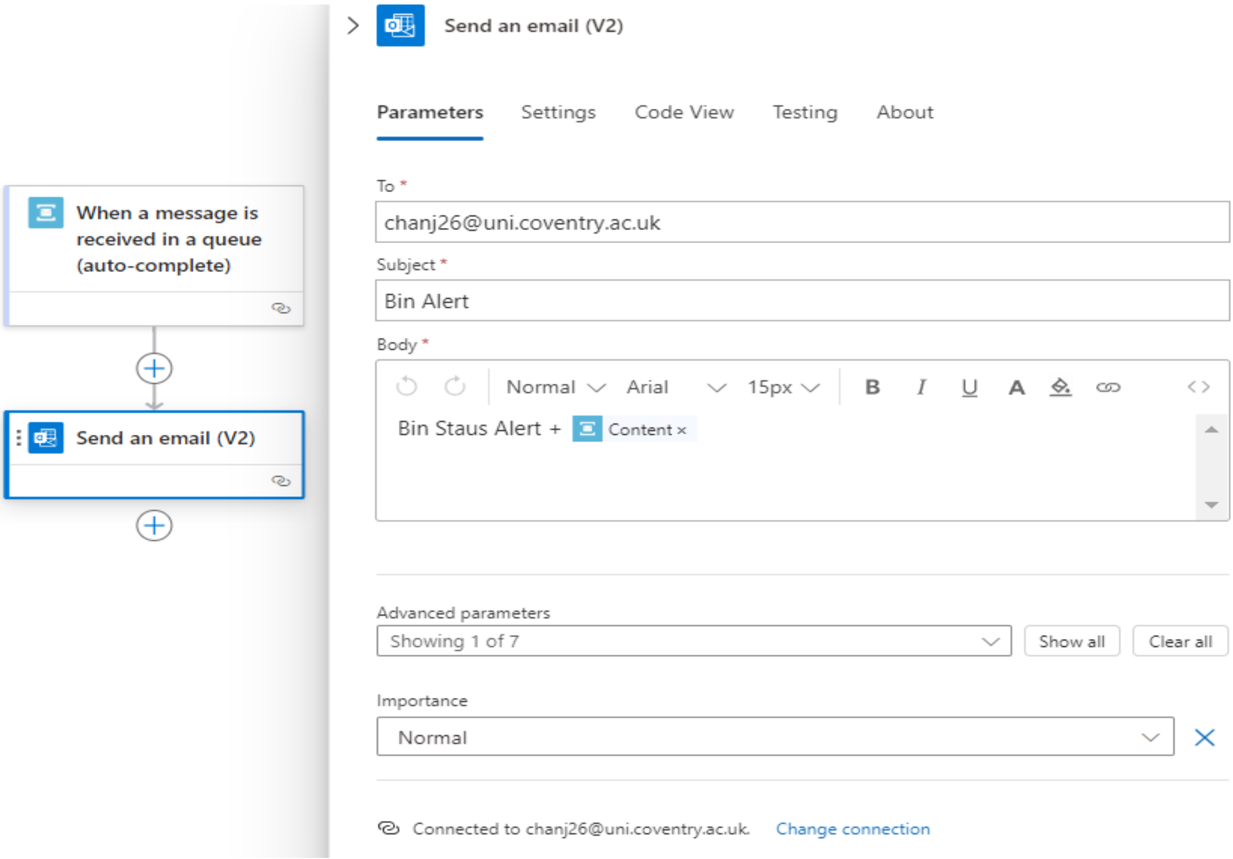
Service Bus Namespace

Incoming Requests (Sum), binstatus 91 | Successful Requests (Sum), binstatus 91 | Server Errors. (Sum), binstatus 0 | User Errors. (Sum), binstatus 0 | Throttled Requests. (Sum), binstatus 0

Incoming Messages (Sum), binstatus 0 | Outgoing Messages (Sum), binstatus 0

Name	Status	Message count	Active messages	Dead-letter messages	Scheduled messages	Max size	Enable partitioning
sbq-binstatus	Active	0	0	0	0	16384 MB	true

Appendix 5: Service Bus Queue



Send an email (V2)

To *
chanj26@uni.coventry.ac.uk

Subject *
Bin Alert

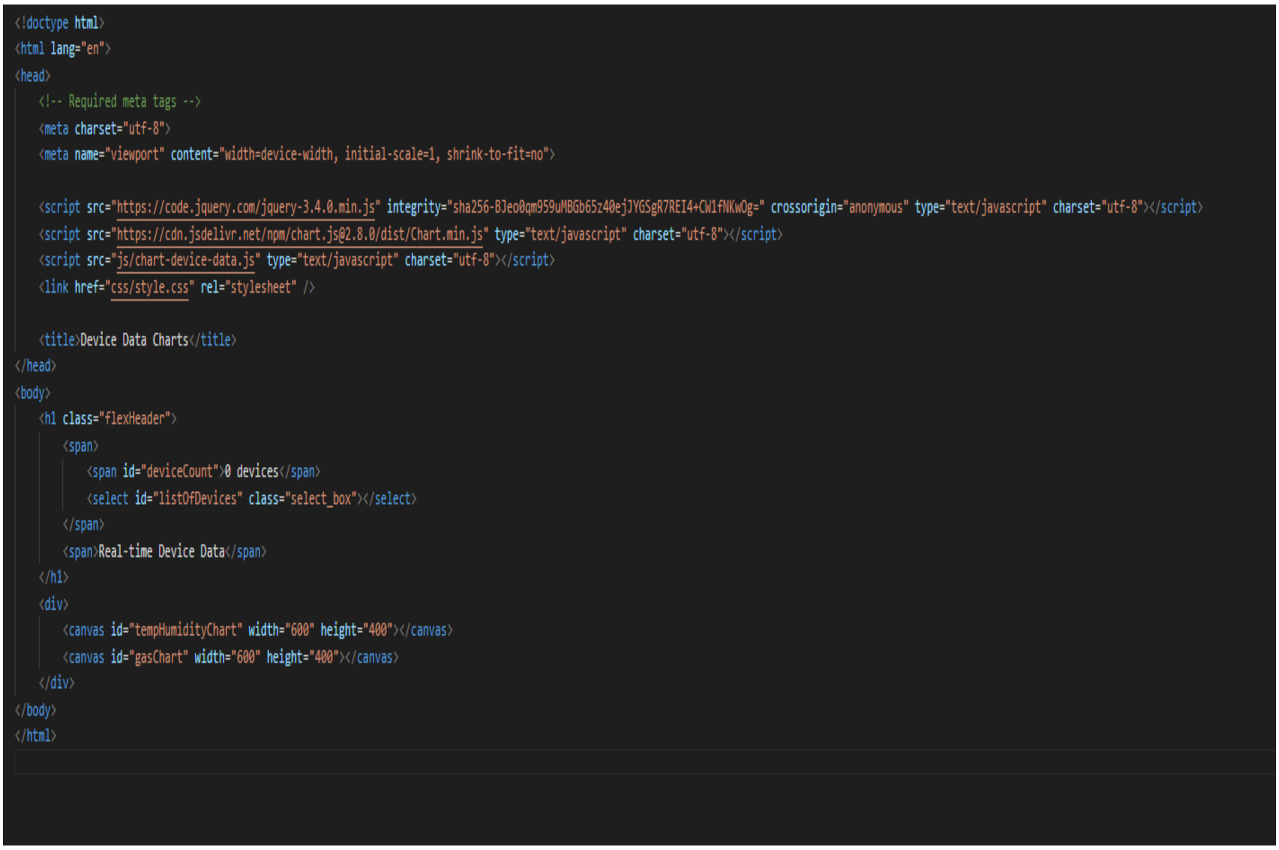
Body *
Bin Staus Alert + Content ×

Advanced parameters
Showing 1 of 7

Importance
Normal

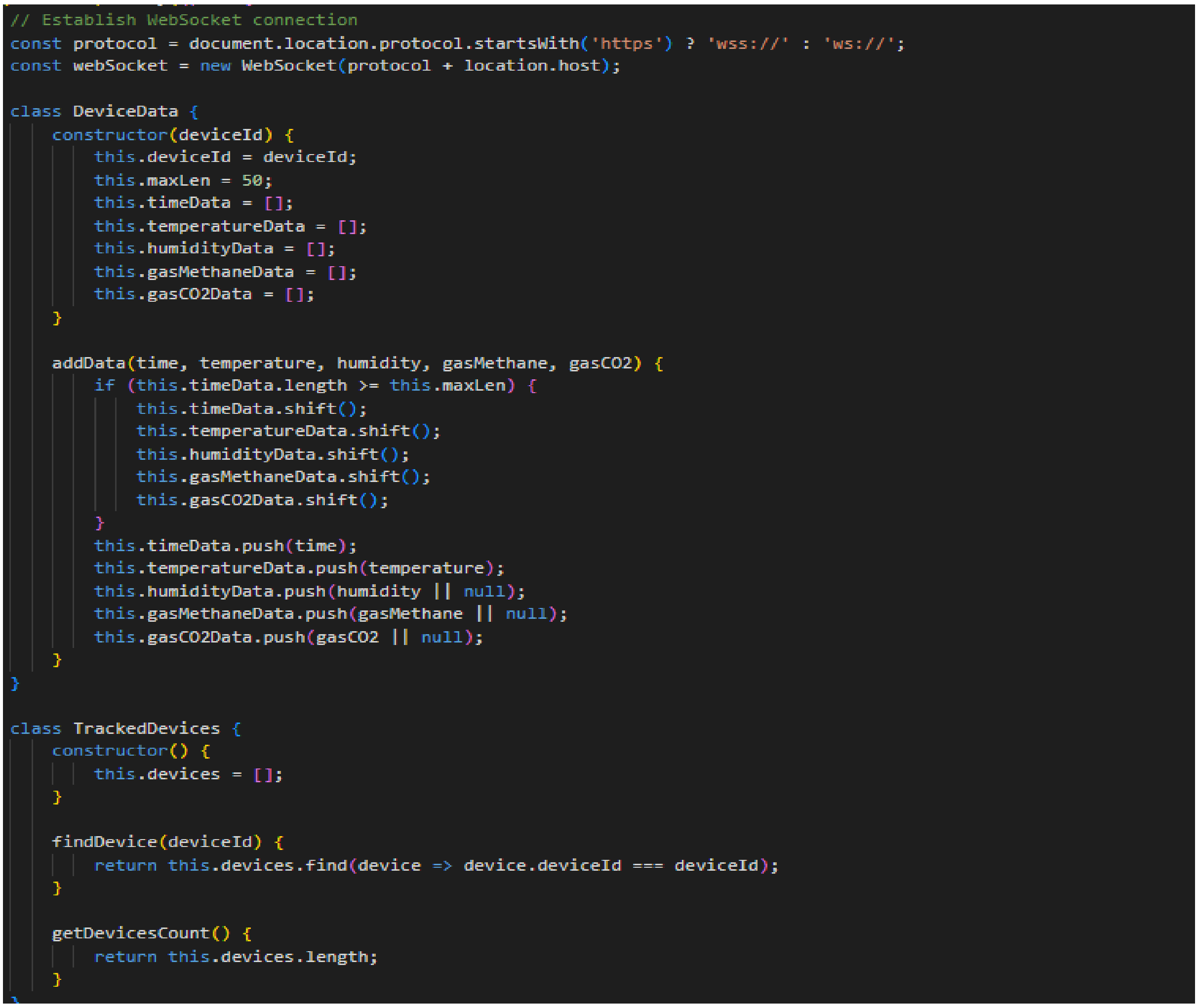
Connected to chanj26@uni.coventry.ac.uk.

Appendix 6: Service Bus Queue



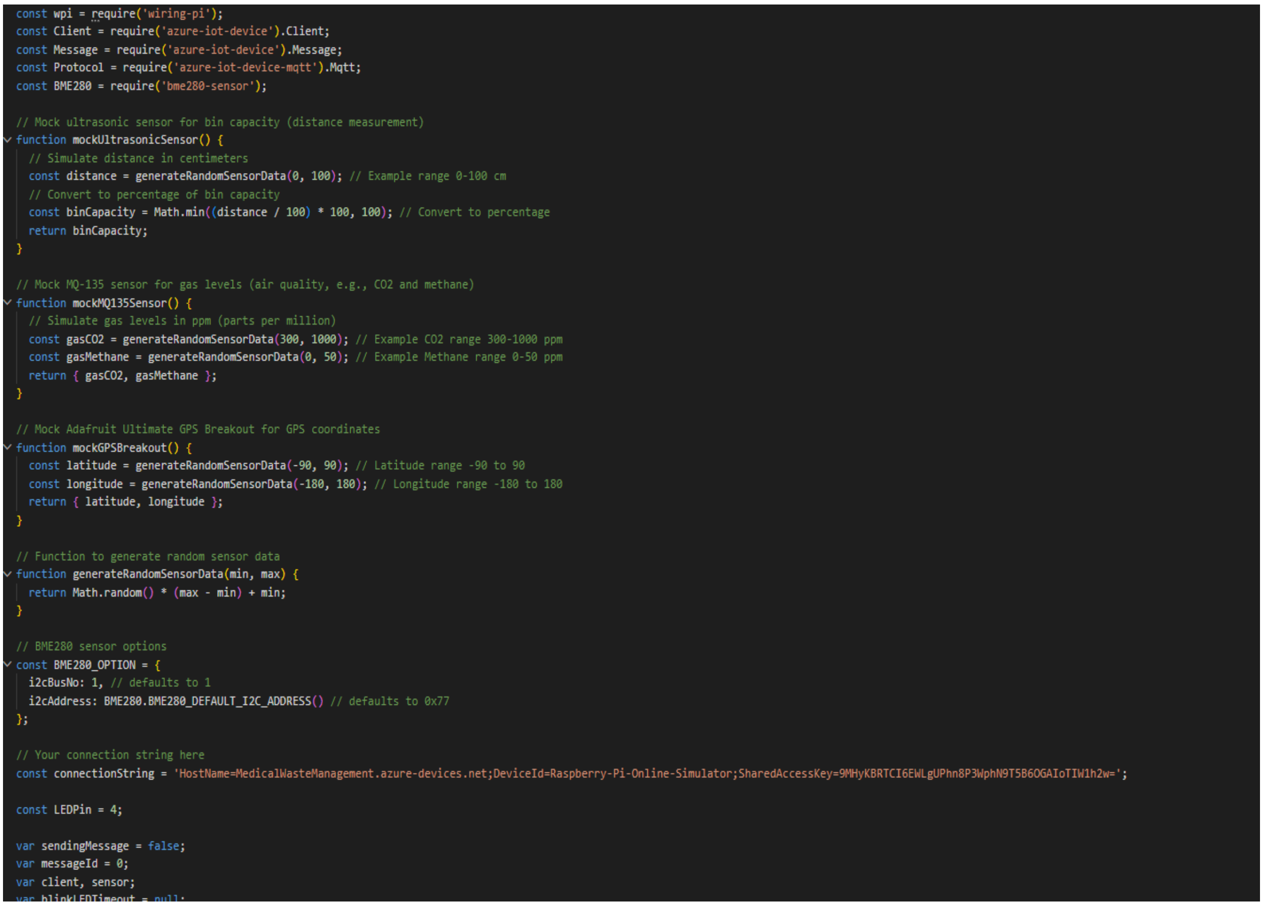
```html
<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <script src="https://code.jquery.com/jquery-3.4.0.min.js" integrity="sha256-BJeo0qm959uMBGb65z40ejJYGSgR7REI4+CW1fNKwOg=" crossorigin="anonymous" type="text/javascript" charset="utf-8"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0/dist/Chart.min.js" type="text/javascript" charset="utf-8"></script>
    <script src="js/chart-device-data.js" type="text/javascript" charset="utf-8"></script>
    <link href="css/style.css" rel="stylesheet" />

    <title>Device Data Charts</title>
</head>
<body>
    <h1 class="flexHeader">
        <span>
            <span id="deviceCount">0 devices</span>
            <select id="listOfDevices" class="select_box"></select>
        </span>
        <span>Real-time Device Data</span>
    </h1>
    <div>
        <canvas id="tempHumidityChart" width="600" height="400"></canvas>
        <canvas id="gasChart" width="600" height="400"></canvas>
    </div>
</body>
</html>
```

Appendix 8: index.html of web app



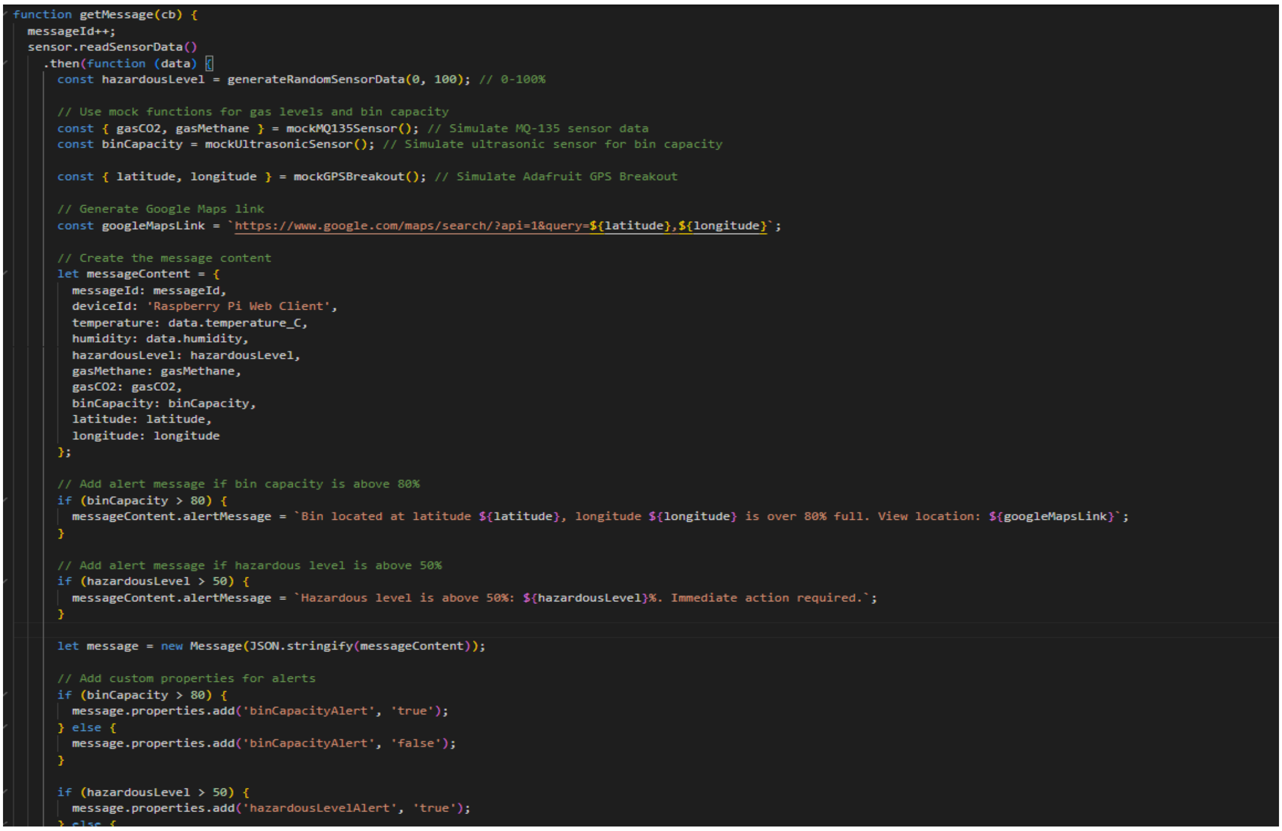
```
// Establish WebSocket connection
const protocol = document.location.protocol.startsWith('https') ? 'wss://' : 'ws://';
const webSocket = new WebSocket(protocol + location.host);

class DeviceData {
    constructor(deviceId) {
        this.deviceId = deviceId;
        this.maxLen = 50;
        this.timeData = [];
        this.temperatureData = [];
        this.humidityData = [];
        this.gasMethaneData = [];
        this.gasCO2Data = [];
    }

    addData(time, temperature, humidity, gasMethane, gasCO2) {
        if (this.timeData.length >= this.maxLen) {
            this.timeData.shift();
            this.temperatureData.shift();
            this.humidityData.shift();
            this.gasMethaneData.shift();
            this.gasCO2Data.shift();
        }
        this.timeData.push(time);
        this.temperatureData.push(temperature);
        this.humidityData.push(humidity || null);
        this.gasMethaneData.push(gasMethane || null);
        this.gasCO2Data.push(gasCO2 || null);
    }
}

class TrackedDevices {
    constructor() {
        this.devices = [];
    }

    findDevice(deviceId) {
        return this.devices.find(device => device.deviceId === deviceId);
    }

    getDevicesCount() {
        return this.devices.length;
    }
}
```

Appendix 9: chart.device.js of web app



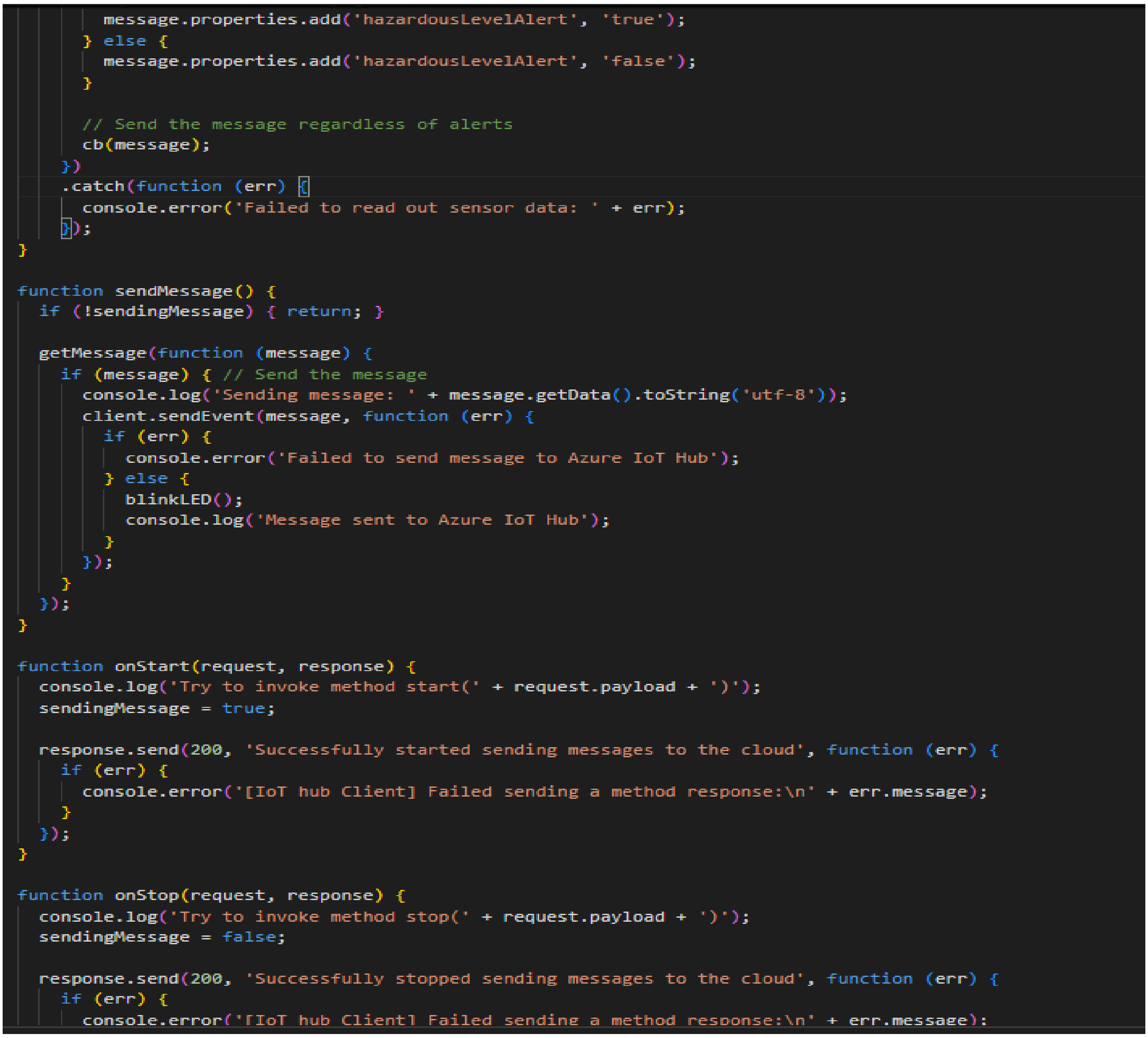
```javascript
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const BME280 = require('bme280-sensor');

// Mock ultrasonic sensor for bin capacity (distance measurement)
function mockUltrasonicSensor() {
  // Simulate distance in centimeters
  const distance = generateRandomSensorData(0, 100); // Example range 0-100 cm
  // Convert to percentage of bin capacity
  const binCapacity = Math.min((distance / 100) * 100, 100); // Convert to percentage
  return binCapacity;
}

// Mock MQ-135 sensor for gas levels (air quality, e.g., CO2 and methane)
function mockMQ135Sensor() {
  // Simulate gas levels in ppm (parts per million)
  const gasCO2 = generateRandomSensorData(300, 1000); // Example CO2 range 300-1000 ppm
  const gasMethane = generateRandomSensorData(0, 50); // Example Methane range 0-50 ppm
  return { gasCO2, gasMethane };
}

// Mock Adafruit Ultimate GPS Breakout for GPS coordinates
function mockGPSBreakout() {
  const latitude = generateRandomSensorData(-90, 90); // Latitude range -90 to 90
  const longitude = generateRandomSensorData(-180, 180); // Longitude range -180 to 180
  return { latitude, longitude };
}

// Function to generate random sensor data
function generateRandomSensorData(min, max) {
  return Math.random() * (max - min) + min;
}

// BME280 sensor options
const BME280_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: BME280.BME280_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};

// Your connection string here
const connectionString = 'HostName=MedicalWasteManagement.azure-devices.net;DeviceId=Raspberry-Pi-Online-Simulator;SharedAccessKey=9MHyKBRTCI6EWLgUPhn8P3WphN9T5B6OGAIoTIW1h2w=';

const LEDPin = 4;

var sendingMessage = false;
var messageId = 0;
var client, sensor;
var blinkLEDTimeout = null;
```

Appendix 10: Full Code Snippet #1 of Prototype



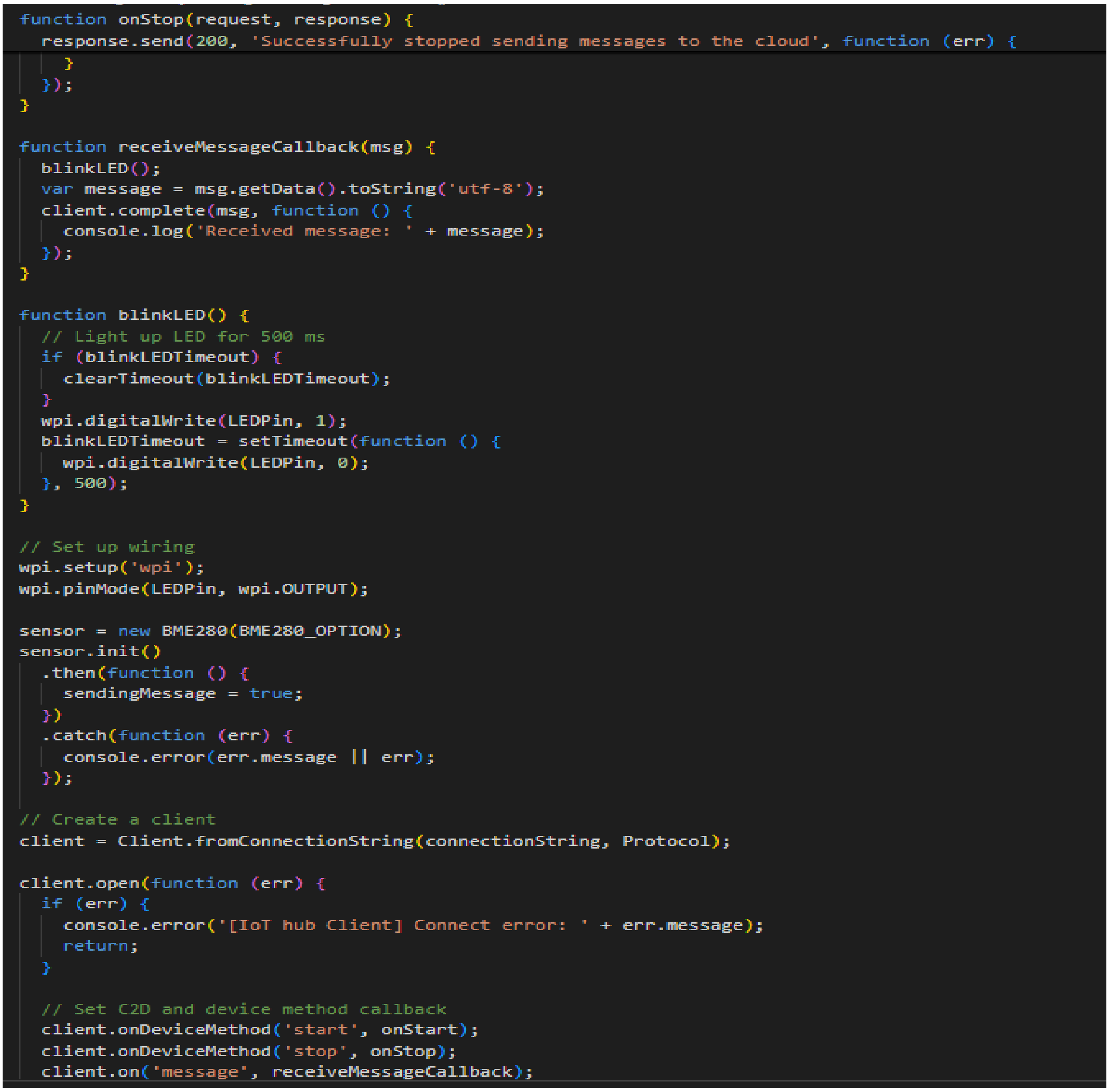
```javascript
function getMessage(cb) {
  messageId++;
  sensor.readSensorData()
    .then(function (data) {
      const hazardousLevel = generateRandomSensorData(0, 100); // 0-100%

      // Use mock functions for gas levels and bin capacity
      const { gasCO2, gasMethane } = mockMQ135Sensor(); // Simulate MQ-135 sensor data
      const binCapacity = mockUltrasonicSensor(); // Simulate ultrasonic sensor for bin capacity

      const { latitude, longitude } = mockGPSBreakout(); // Simulate Adafruit GPS Breakout

      // Generate Google Maps link
      const googleMapsLink = `https://www.google.com/maps/search/?api=1&query=${latitude},${longitude}`;

      // Create the message content
      let messageContent = {
        messageId: messageId,
        deviceId: 'Raspberry Pi Web Client',
        temperature: data.temperature_C,
        humidity: data.humidity,
        hazardousLevel: hazardousLevel,
        gasMethane: gasMethane,
        gasCO2: gasCO2,
        binCapacity: binCapacity,
        latitude: latitude,
        longitude: longitude
      };

      // Add alert message if bin capacity is above 80%
      if (binCapacity > 80) {
        messageContent.alertMessage = `Bin located at latitude ${latitude}, longitude ${longitude} is over 80% full. View location: ${googleMapsLink}`;
      }

      // Add alert message if hazardous level is above 50%
      if (hazardousLevel > 50) {
        messageContent.alertMessage = `Hazardous level is above 50%: ${hazardousLevel}%. Immediate action required.`;
      }

      let message = new Message(JSON.stringify(messageContent));

      // Add custom properties for alerts
      if (binCapacity > 80) {
        message.properties.add('binCapacityAlert', 'true');
      } else {
        message.properties.add('binCapacityAlert', 'false');
      }

      if (hazardousLevel > 50) {
        message.properties.add('hazardousLevelAlert', 'true');
      } else {
```

Appendix 11: Full Code Snippet #2 of Prototype



```
        message.properties.add('hazardousLevelAlert', 'true');
      } else {
        message.properties.add('hazardousLevelAlert', 'false');
      }

      // Send the message regardless of alerts
      cb(message);
    })
    .catch(function (err) {
      console.error('Failed to read out sensor data: ' + err);
    });
}

function sendMessage() {
  if (!sendingMessage) { return; }

  getMessage(function (message) {
    if (message) { // Send the message
      console.log('Sending message: ' + message.getData().toString('utf-8'));
      client.sendEvent(message, function (err) {
        if (err) {
          console.error('Failed to send message to Azure IoT Hub');
        } else {
          blinkLED();
          console.log('Message sent to Azure IoT Hub');
        }
      });
    }
  });
}

function onStart(request, response) {
  console.log('Try to invoke method start(' + request.payload + ')');
  sendingMessage = true;

  response.send(200, 'Successfully started sending messages to the cloud', function (err) {
    if (err) {
      console.error('[IoT hub Client] Failed sending a method response:\n' + err.message);
    }
  });
}

function onStop(request, response) {
  console.log('Try to invoke method stop(' + request.payload + ')');
  sendingMessage = false;

  response.send(200, 'Successfully stopped sending messages to the cloud', function (err) {
    if (err) {
      console.error('[IoT hub Client] Failed sending a method response:\n' + err.message);
```

Appendix 12: Full Code Snippet #3 of Prototype



```
function onStop(request, response) {
  response.send(200, 'Successfully stopped sending messages to the cloud', function (err) {
    }
  });
}

function receiveMessageCallback(msg) {
  blinkLED();
  var message = msg.getData().toString('utf-8');
  client.complete(msg, function () {
    console.log('Received message: ' + message);
  });
}

function blinkLED() {
  // Light up LED for 500 ms
  if (blinkLEDTimeout) {
    clearTimeout(blinkLEDTimeout);
  }
  wpi.digitalWrite(LEDPin, 1);
  blinkLEDTimeout = setTimeout(function () {
    wpi.digitalWrite(LEDPin, 0);
  }, 500);
}

// Set up wiring
wpi.setup('wpi');
wpi.pinMode(LEDPin, wpi.OUTPUT);

sensor = new BME280(BME280_OPTION);
sensor.init()
  .then(function () {
    sendingMessage = true;
  })
  .catch(function (err) {
    console.error(err.message || err);
  });

// Create a client
client = Client.fromConnectionString(connectionString, Protocol);

client.open(function (err) {
  if (err) {
    console.error('[IoT hub Client] Connect error: ' + err.message);
    return;
  }

  // Set C2D and device method callback
  client.onDeviceMethod('start', onStart);
  client.onDeviceMethod('stop', onStop);
  client.on('message', receiveMessageCallback);
```

Appendix 13: Full Code Snippet #4 of Prototype