Journal of Multidisciplinary Research and Technology
Volume 1 Nomor 2 | April 2025
e-ISSN: 3089-3372; p-ISSN: 3089-3380, Hal 51-66
DOI: https://doi.org/10.63891/j-mart.v1i2.19
Available online at: https://journal.iset.or.id/index.php/J-MART/article/view/61

Design and Development of an Internet of Things-Based Trash Bin Capacity and Location Monitoring System

Bagas Naufal Insani[1], Nur Sultan Salahuddin[2*], Abdul Muchlis[3], Ario Geraldi[4]
[1,2,3,4] Gunadarma University, Indonesia

Abstract. *Waste is unwanted residual material after the end of a process. One of the problems regarding rubbish which is a threat from urban society today is that there are many trash cans that are full but not transported by cleaners. Therefore, we need a system that can monitor the capacity and location of the trash can from a distance, one method is by using Internet of Things. Internet of Things is a concept where an object has the ability to transfer data through a network without requiring human-to-human or human-computer interaction. By using IoT, activities that should be done directly can be done remotely via the internet, including in monitoring the level of trash bin fullness. This study aims to create a Internet of Things based trash monitoring system. An application on a smartphone is made so that it can know the capacity of several trash bins scattered in several locations, then provide notification to the officer. This can be done remotely via a smartphone connected to the internet.*

Keywords: *Cloud, IoT, Monitoring System, Smartphone, Trash Bin*

INTRODUCTION

Urban waste management presents a significant and growing challenge globally. As urban populations increase, so does the volume of solid waste generated. For instance, in DKI Jakarta, Indonesia, waste generation saw an appreciable increase of approximately 15%, from 6,016.30 tons in 2016 to 6,872.18 tons in 2017. Improperly managed waste leads to severe public health risks and environmental degradation. A critical operational issue in many urban centers is the inefficiency of conventional waste collection systems. These systems often rely on fixed schedules, which do not account for the actual fill level of waste receptacles.

This static approach leads to two primary problems, bins overflow before collection, creating unsanitary conditions, unpleasant odors, and the proliferation of disease vectors; and resources are wasted emptying bins that are still empty or only partially full. This inefficiency stems directly from a lack of real-time data; sanitation departments often have no information on the current status of individual bins. Furthermore, this absence of data makes it difficult to monitor the performance and accountability of sanitation services, as it is challenging to verify if and when full bins have been serviced.

Concurrently, rapid advancements in information and communication technology, particularly the Internet of Things (IoT), offer promising solutions to these logistical

Received: February 7, 2025; Accepted: April 16, 2025; Published: April 17, 2025
*Corresponding author, sultan@staff.gunadarma.ac.id

challenges. The IoT paradigm involves a network of physical objects embedded with sensors, software, and other technologies to connect and exchange data over the internet, often without direct human intervention. This technology enables the remote, real-time monitoring and control of physical assets, creating significant opportunities for optimization in sectors such as waste management.

By leveraging IoT, the problems associated with inefficient waste collection can be substantially mitigated. This paper proposes the design and development of an IoT-based smart waste monitoring system. The core of this system involves equipping waste bins with sensors (e.g., ultrasonic) to detect their fill levels in real-time. This fill-level data, along with the bin's geospatial location, is transmitted via a network to a central cloud service.

To make this data actionable, a mobile application for the Android platform was developed. This application provides sanitation workers and managers with a real-time dashboard, visualizing the status and location of all monitored bins. This allows for the dynamic optimization of collection routes, enabling crews to prioritize and service only those bins that are full or nearing capacity.

Therefore, the primary objective of this study is to design, build, and implement an end-to-end system for monitoring waste bin capacity and location using IoT technology. This system aims to provide real-time, remote monitoring of waste bin fill levels via a cloud service and an Android smartphone application, optimize waste collection logistics to reduce operational costs and prevent bin overflowm and establish a data-driven mechanism for monitoring the efficiency and responsiveness of sanitation services.

METHODOLOGY

The development of the IoT-Based Waste Bin Capacity and Location Monitoring System described in this paper is bifurcated into two core components: a hardware subsystem and a software subsystem. The hardware subsystem encompasses the physical components responsible for detecting the waste fill level and transmitting this data to the cloud. The software subsystem involves programming the microcontroller, configuring the cloud services for data storage and location tracking, and developing a smartphone application for remote monitoring.

System Architecture

The high-level architecture of the system, as depicted in the block diagram (Figure 3.1), comprises three primary blocks:

1. Smart Bin Monitoring Unit (ESP32): This is the core hardware unit attached to each waste bin. It is responsible for detecting the bin's fill level using input from ultrasonic sensors and subsequently transmitting the processed data to the cloud platform.
2. Cloud Platform (ThingsBoard & ThingSpeak): This block functions as the central data repository and processing hub. It stores the fill-level values and location data from multiple smart bins. ThingsBoard is used to provide a comprehensive web-based dashboard for visualization, while ThingSpeak is utilized for a specific alert mechanism.
3. Monitoring Application (Smartphone): This is the user-facing client application. It displays a map populated with the real-time status and location of all bins, provides graphical representations of fill-level data, and delivers push notifications to the user when a bin is full.

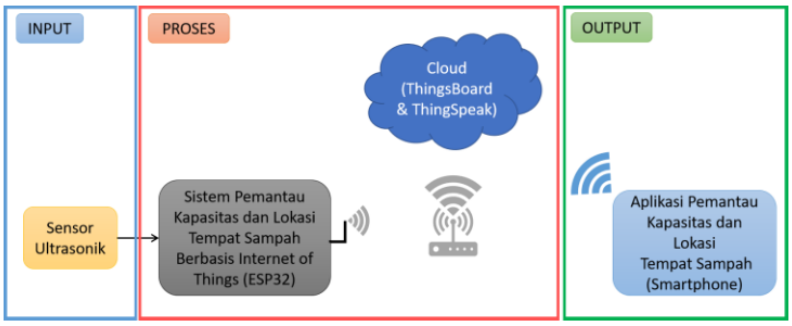

Figure 3.1 Block Diagram of the IoT-Based Waste Bin Capacity and Location Monitoring System

Hardware Subsystem Design

The electronic circuit for the monitoring unit is detailed in Figure 3.2. To mitigate measurement errors caused by unevenly distributed waste (e.g., trash piling up on one side), two HC-SR04 ultrasonic sensors are employed. The sensors are interfaced with an ESP32 microcontroller as follows:

1. Sensor 1: The Trig (Trigger) pin is connected to pin 25 on the ESP32, and the Echo pin is connected to pin 26.
2. Sensor 2: The Trig (Trigger) pin is connected to pin 32 on the ESP32, and the Echo pin is connected to pin 33.
3. Power: The VCC pins of both sensors are connected to the Vin pin of the ESP32, which provides a voltage level equivalent to the ESP32's power source. The GND pins of both sensors are connected to a common GND pin on the ESP32.

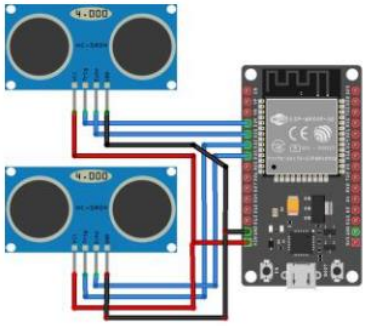

Figure 3.2 Electronic Circuit Diagram of the Monitoring System

Physical Design and Prototype

The physical placement of the components is crucial for accurate sensing and system durability. As illustrated in the design (Figure 3.3), the two ultrasonic sensors are mounted on the inner top lid of the bin, positioned on opposite sides. They are aimed downwards to measure the distance to the top layer of the waste. This dual-sensor placement strategy provides a more reliable fill-level reading by averaging or comparing the two measurements, reducing inaccuracies from irregular waste accumulation.

The microcontroller and battery pack are housed in a protective enclosure mounted on the exterior of the bin. This placement prevents interference with the processes of waste disposal and collection. The final implemented prototype is shown in Figure 3.4.

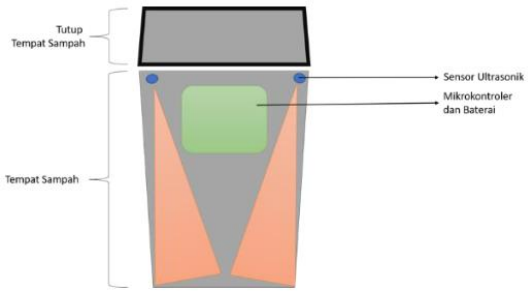

Figure 3.3 Physical Design of the Monitoring System Components on the Bin

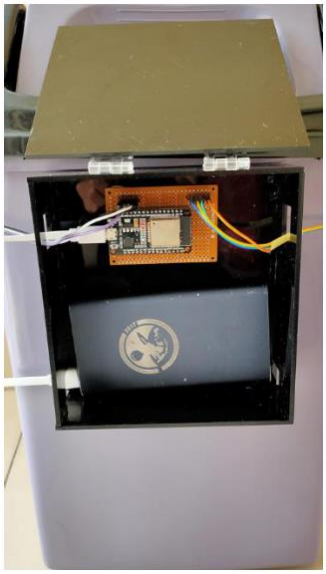

Figure 3.4 The Implemented Prototype

Software Subsystem Implementation

The software subsystem is composed of the microcontroller firmware, the cloud platform configuration, and the mobile application.

Microcontroller Firmware

Sensor Reading: The sensor operation (Figure 3.5) is initiated by sending a 10-microsecond HIGH pulse to the Trig pin. The sensor then emits an ultrasonic burst. The Echo pin outputs a HIGH signal for a duration equivalent to the time taken for the wave to travel to the object and return. The distance is calculated from this duration based on the speed of sound. This process is executed for both ultrasonic sensors to obtain two distance readings.

SmartGarbage_System_2Ping

```
void readCapacity() {
  //SENSOR 1
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin1, INPUT);
  duration1 = pulseIn(echoPin1, HIGH);

  // Convert the time into a distance
  cm1 = (duration1 / 2) / 29.1;   // Divide by 29.1 or multiply by 0.0343
  if (cm1 > trashHeight) {
    cm1 = trashHeight;
  }
```

Figure 3.5 Program Snippet for Reading the Ultrasonic Sensor

Data Transmission to ThingsBoard: Data is sent to the ThingsBoard platform using the tb.sendTelemetryFloat function (Figure 3.6). Three distinct values are transmitted: the raw distance from sensor 1, the raw distance from sensor 2, and the final calculated bin capacity (e.g., as a percentage). While all three values are sent, only the final capacity value is used for the primary dashboard visualization.

SmartGarbage_System_2Ping

```
void connectToThingsboard() {
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }

  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }

  Serial.println("Sending data to Thingsboard...");
  tb.sendTelemetryFloat("distance1", cm1);
  tb.sendTelemetryFloat("distance2", cm2);
  tb.sendTelemetryFloat("capacity", capacity);
  tb.loop();
}
```

Figure 3.6 Program Snippet for Sending Data to ThingsBoard Cloud

Data Transmission to ThingSpeak: The ThingSpeak platform is used for the alert system (Figure 3.7). In this implementation, data from all smart bins is aggregated into a single channel. To differentiate the data source, the data is formatted as a character string: BinName:BinCapacity (e.g., "BinA:95"). This format allows the mobile application, which listens to this channel, to parse the string and identify exactly which bin has triggered a "full" alert.

SmartGarbage_System_2Ping

```
void connectToThingspeak() {
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }

  char capacity_buf[5];
  dtostrf(capacity, 0, 1, capacity_buf);

  strcpy (mergedData, nameTS);
  strcat (mergedData, ":");
  strcat (mergedData, capacity_buf);

  if (processCount >= 16) {
    Serial.println("Sending data to Thingspeak...");
    ThingSpeak.writeField(myChannelNumber, 1, mergedData, myWriteAPIKey);
    processCount = 0;
  }

}
```

Figure 3.7 Program Snippet for Sending Data to ThingSpeak Cloud

Cloud Platform Configuration

ThingsBoard: The ThingsBoard dashboard (Figure 3.8) serves as the primary monitoring interface for system administrators. The main page displays a map with geolocated markers for each bin. Clicking a marker reveals detailed information, including capacity, address, and coordinates (latitude, longitude). A table is also provided, listing all registered bins in the system with their current status. For this research, four smart bin devices were configured and displayed.

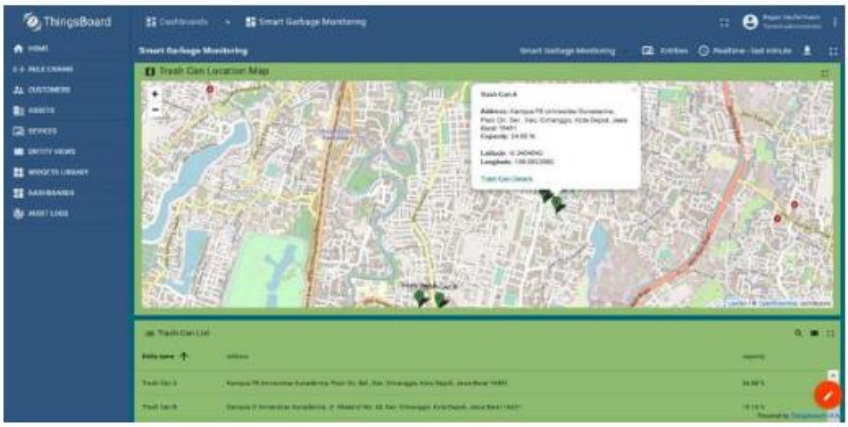

Figure 3.8 ThingsBoard Dashboard with Map and Bin List

ThingSpeak: The ThingSpeak platform is configured specifically for the real-time alert mechanism. A channel named 'Smart Garbage Monitoring' was created containing a single field: 'Capacity Alarm' (Figure 3.9). This channel is configured to receive data from any microcontroller that reports a capacity *above* a set threshold (e.g., >66%). The mobile application monitors this specific channel to generate alerts.

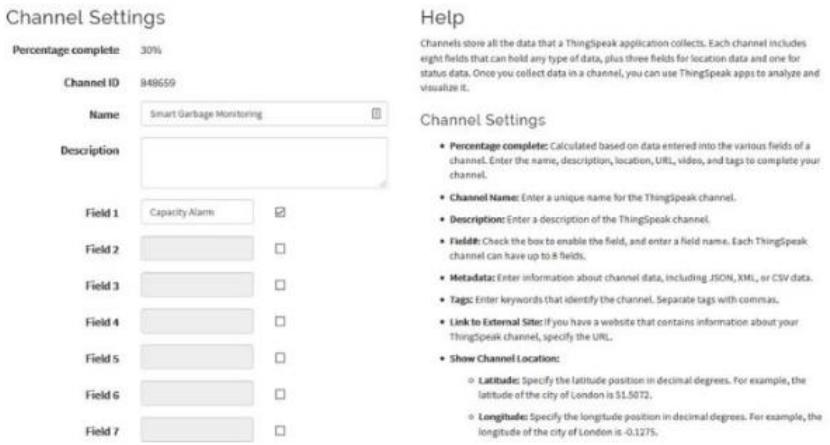

Figure 3.9 ThingSpeak Channel Configuration for Alerts

Mobile Application

The Android mobile application provides on-the-go monitoring for sanitation workers. The application essentially embeds and displays the dashboard created on the ThingsBoard cloud.

1. Main Page (Figure 3.10): This layout displays the main dashboard, which includes the map view and the list of all monitored bins.
2. Details Page (Figure 3.11): When a user selects a specific bin, this layout is shown, displaying the detailed view for that bin, including its current capacity value and a historical data graph.\

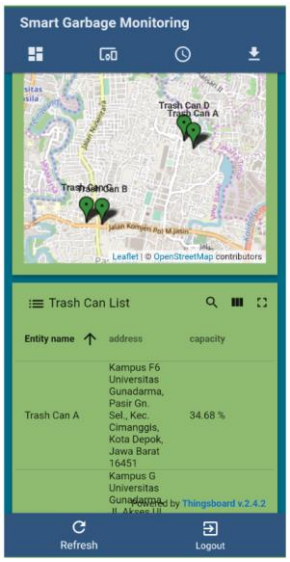

Figure 3.10 Application Layout for the Main Dashboard Activity

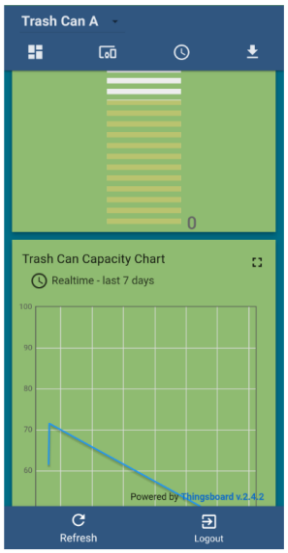

Figure 3.11 Application Layout for the Bin Details Activity

RESULTS

Ultrasonic Sensor Accuracy Test

This test was conducted to evaluate the accuracy of the distance readings provided by the ultrasonic sensor.

Table 4.1 Test Results for Ultrasonic Sensor Accuracy

No.	Object Distance	Measured Result	Difference	Percentage Error
1	5 cm	5.27 cm	0.27 cm	5.40 %
2	10 cm	10.34 cm	0.34 cm	3.40 %
3	15 cm	15.91 cm	0.91 cm	6.06 %
4	20 cm	19.66 cm	0.34 cm	1.70 %
5	25 cm	24.98 cm	0.02 cm	0.08 %
6	30 cm	30.02 cm	0.02 cm	0.06 %
Average			0.316 cm	2.78 %

Based on the test results in Table 4.1, it can be concluded that the ultrasonic sensor functions effectively. The average difference in distance measurement was only 0.316 cm, corresponding to an average error percentage of just 2.78%.

Bin Capacity Calculation Test

This test was performed to verify the correctness of the waste bin capacity calculation algorithm.

Table 4.2 Test Results for Bin Capacity Calculation

No.	Distance from Sensor 1	Distance from Sensor 2	Calculated Capacity	Status
1	30.33 cm	27.80 cm	11.93 %	Calculation correct
2	24.53 cm	25.67 cm	23.94 %	Calculation correct
3	20.31 cm	18.78 cm	40.78 %	Calculation correct
4	16.34 cm	13.10 cm	55.40 %	Calculation correct
5	9.11 cm	9.47 cm	71.85 %	Calculation correct
6	5.60 cm	6.12 cm	82.25 %	Calculation correct

As shown in Table 4.2, the resulting bin capacity calculations are consistent with the formula implemented on the microcontroller. In all six test cases, the calculations were performed successfully and yielded the correct values.

ThingsBoard Cloud Data Reception Test

This test was conducted to verify that data sent by the ESP32 microcontrollers was correctly received by the ThingsBoard cloud, particularly when handling data from multiple sources.

Table 4.3 Test Results for Data Reception by ThingsBoard Cloud

No.	Data Sent from Microcontroller ESP32				Data Received by ThingsBoard Cloud
	Microcontroller 1 (Bin A)	Microcontroller 2 (Bin B)	Microcontroller 3 (Bin C)	Microcontroller 4 (Bin D)	
1	30.33 %	-	-	-	Trash Can A: 30.33 % Trash Can B: - Trash Can C: Trash Can D: -
2	67.12 %	-	-	-	Trash Can A: 67.12 % Trash Can B: - Trash Can C: Trash Can D: -
3	8.34 %	53.51 %	-	-	Trash Can A: 8.34 % Trash Can B: 53.51 % Trash Can C: Trash Can D: -
4	49.06 %	71.65 %	-	-	Trash Can A: 49.06 % Trash Can B: 71.65 % Trash Can C: Trash Can D: -
5	11.99 %	26.42 %	44.78 %	-	Trash Can A: 11.99 % Trash Can B: 26.42 % Trash Can C: 44.78 % Trash Can D: -
6	50.10 %	85.85 %	90.77 %	-	Trash Can A: 50.10 % Trash Can B: 85.85 % Trash Can C: 90.77 % Trash Can D: -
7	12.19 %	73.71 %	81.63 %	86.56 %	Trash Can A: 12.19 % Trash Can B: 73.71 % Trash Can C: 81.63 % Trash Can D: 86.56 %
8	44.71 %	83.96 %	28.89 %	10.02 %	Trash Can A: 44.71 % Trash Can B: 83.96 % Trash Can C: 28.89 % Trash Can D: 10.02 %

Based on the results in Table 4.3, the ThingsBoard cloud platform successfully received all transmitted data. Data was correctly acquired and parsed, starting from a single microcontroller transmission, and scaling up to simultaneous transmissions from two, three, and finally all four microcontrollers. This confirms that the ThingsBoard platform can reliably receive concurrent data from multiple devices.

ThingSpeak Cloud Data Reception Test

This test was conducted to verify that data sent by the ESP32 microcontrollers was correctly received by the ThingSpeak cloud.

Table 4.4 Test Results for Data Reception by ThingSpeak Cloud

No.	Data Sent from Microcontroller ESP32	Data Received by ThingSpeak Cloud	Status
1	A:13.63	A:13.63	Data matches
2	A:70.12	A:70.12	Data matches
3	B:68.47	B:68.47	Data matches
4	B:39.82	B:39.82	Data matches
5	C:9.69	C:9.69	Data matches
6	C:98.51	C:98.51	Data matches
7	D:66.53	D:66.53	Data matches
8	D:27.13	D:27.13	Data matches

The test results in Table 4.4 show that the ThingSpeak cloud platform correctly received the data sent by the ESP32 microcontrollers. In all eight test cases, the received data perfectly matched the transmitted data.

Application Data Reading Test (ThingsBoard)

This test was conducted to ensure that the data displayed in the mobile application accurately reflected the data present on the ThingsBoard cloud.

Table 4.5 Test Results for Application Data Reading from ThingsBoard

No.	Data on ThingsBoard Cloud	Data Read by Application	Status
1	Trash Can A: 30.33 % Trash Can B: - Trash Can C: - Trash Can D: -	Trash Can A: 30.33 % Trash Can B: - Trash Can C: - Trash Can D: -	Data matches
2	Trash Can A: 67.12 % Trash Can B: - Trash Can C: - Trash Can D: -	Trash Can A: 67.12 % Trash Can B: - Trash Can C: - Trash Can D: -	Data matches
3	Trash Can A: 8.34 % Trash Can B: 53.51 % Trash Can C: - Trash Can D: -	Trash Can A: 8.34 % Trash Can B: 53.51 % Trash Can C: - Trash Can D: -	Data matches
4	Trash Can A: 49.06 % Trash Can B: 71.65 % Trash Can C: - Trash Can D: -	Trash Can A: 49.06 % Trash Can B: 71.65 % Trash Can C: - Trash Can D: -	Data matches
5	Trash Can A: 11.99 % Trash Can B: 26.42 % Trash Can C: 44.78 % Trash Can D: -	Trash Can A: 11.99 % Trash Can B: 26.42 % Trash Can C: 44.78 % Trash Can D: -	Data matches
6	Trash Can A: 50.10 % Trash Can B: 85.85 % Trash Can C: 90.77 %	Trash Can A: 50.10 % Trash Can B: 85.85 % Trash Can C: 90.77 %	Data matches

Based on the results in Table 4.5, the mobile application successfully read and displayed the data from the ThingsBoard cloud. In all eight test cases, the data shown in the application was identical to the data stored on the cloud platform. (Note: Data in table abbreviated for clarity, full data matches Table 4.3).

Application Notification Test (ThingSpeak)

This test was conducted to verify that the mobile application could correctly read data from the ThingSpeak alert channel and trigger the appropriate user notifications.

Table 4.6 Test Results for Application Notification based on ThingSpeak Data

No.	Data on ThingsBoard Cloud	Data Read by Application	Notification Displayed by Application	Keterangan
1	A:13.63	A:13.63	-	Data matches
2	A:70.12	A:70.12	"Trash Can A is full! Contains 70.12 % capacity. Please transport the garbage."	Data matches
3	B:68.47	B:68.47	"Trash Can B is full! Contains 68.47 % capacity. Please transport the garbage."	Data matches
4	B:39.82	B:39.82	-	Data matches
5	C:9.69	C:9.69	-	Data matches
6	C:98.51	C:98.51	"Trash Can C is full! Contains 98.51 % capacity. Please transport the garbage."	Data matches
7	D:66.53	D:66.53	"Trash Can D is full! Contains 66.53 % capacity. Please transport the garbage."	Data matches
8	D:27.13	D:27.13	-	Data matches

As shown in Table 4.6, the application successfully read the data from the ThingSpeak cloud channel. All data was read correctly across the eight tests. Furthermore, the notification system functioned as designed, triggering an alert only when the received capacity data exceeded the predefined threshold (e.g., >66%).

CONCLUSION

Based on the research and implementation conducted, the objectives of this study were successfully achieved through the creation of an Internet of Things (IoT) based waste bin monitoring system. This system effectively utilizes sensors to detect the fill level of waste bins and is complemented by an Android application that allows users to monitor the status and location of the bins. Furthermore, all sensor data is accessible via the Android smartphone application through integrated cloud services.

The developed system, leveraging IoT and sensor technology, demonstrated its capability to accurately detect waste fill levels. The test results confirmed that the data generated by the sensors was reliably transmitted to the cloud platforms (ThingsBoard and ThingSpeak) and accurately displayed on the user's Android smartphone application.

The implementation of this system has the potential to significantly improve the operational efficiency of waste collection, reducing operational and environmental costs associated with inefficient routes. It can also contribute to raising public awareness about the importance of proper waste management.

However, some limitations were identified that provide direction for future enhancements. Further improvements are needed in refining sensor accuracy for detecting various types of waste and optimizing the physical placement of sensors to ensure they do not inconvenience users during waste disposal.

Overall, the IoT-based waste bin capacity and location monitoring system developed in this study successfully met its established goals. It is hoped that these findings can provide a guide and inspiration for further research and development in the pursuit of efficient and sustainable urban waste management.

Based on the results and limitations of this study, the following areas are recommended for future development:

1. Integration of Weight Sensors: Enhancing the system by adding weight sensors to provide a more comprehensive capacity calculation, complementing the volume-based data from the ultrasonic sensors.
2. Automatic Route Optimization: Developing the application further by integrating the Google Maps API to automatically generate the most efficient collection routes based

3. User-Configurable Notifications: Implementing a feature within the mobile application that allows users (sanitation workers or administrators) to set their own minimum capacity threshold for triggering "full" notifications.

LIMITATION

It is inevitable that your research will have some limitations, and this is normal. However, it is critically important to strive to minimize the scope of these limitations throughout the research process. Additionally, you need to acknowledge your research limitations honestly in the conclusions chapter.

Identifying and acknowledging the shortcomings of your work is preferable to having them pointed out by your final work assessor. While discussing your research limitations, do not merely list and describe them. It is also crucial to explain how these limitations have impacted your research findings.

Your research may have multiple limitations, but you should discuss only those that directly relate to your research problems. For example, if conducting a meta-analysis of secondary data was not stated as your research objective, there is no need to mention it as a limitation of your research.

REFERENCES

A. Anitha. (2017). *Garbage Monitoring System using IoT*. IOP Conference Series: Materials Science and Engineering.

Abishek. (2017). "XML in Android: Basics And Different XML Files Used In Android". *http://abhiandroid.com/ui/xml*.

Ahmed, Elmustafa. (2019). *Internet of things in Smart Environment: Concept, Applications, Challenges, and Future Directions*. Sudan: World Scientific News.

Anonim. (1992). *Tata Cara Pengolahan Teknik Sampah Perkotaan*. Bandung: Yayasan LPMB.

Aslamia, Suhaybatul. (2015). *Robot Pendeteksi Manusia Sebagai Sistem Keamanan Ruangan Menggunakan Sensor PIR Dengan Media Komunikasi XBee Berbasis Arduino Leonardo*. Palembang: Politeknik Negeri Sriwijaya.

Azzola, Francesco. (2015). MQTT Protocol Tutorial: Step by step guide, Mosquitto and MQTT Security. https://www.hivemq.com/blog/mqttessentials-part-3-client-broker-connection *establishment*, 24 April 2018

Bitencourt, Eduardo N. (2018). *IoT Centralization and Management Applying ThingsBoard Platform*. Finland: Hame University of Applied Sciences.

Gabbrielli, Maurizio. (2010). *Programming Languages: Principles and Paradigms*. New York: University of Bologna.

Gosling J, Joy B, Steele G, Bracha G, Buckley A. (2015). *The Java Language Specification*. California: Oracle Corporation.

G2 Crowd. (2018). *Best IoT Management Software*. URL: https://www.g2crowd.com/categories/iot-management 26 April 2018

Hassan Qusay. (2011). *Demystifiying Cloud Computing*. Dakahlia: The Journal of Defense Software Engineering.

HiveMQ. (2015). *MQTT Essentials Part 3: Client, Broker and Connection Establishment*. URL: https://www.hivemq.com/blog/mqtt-essentials-part-3client-broker-connection-establishment, 24 April 2018

Marhaeni, Harmawati. et al. (2018). *Statistik Lingkungan Hidup Indonesia 2018*. Jakarta: Badan Pusat Statistik Indonesia

Maulana, Adnan. (2013). *Perancangan Media Informasi Mengenai Masalah Sampah di Lingkungan Pasar*. Bandung: Universitas Komputer Indonesia.

Miglani, Gaurav. "How to parse JSON in Java". http://www.geeksforgeeks.org/parse-json-java, 15 September 2017.

Morgan, Elijah. (2014). *HC-SR04 Ultrasonic Sensor*. Vietnam: Kysungheo.

MQTT. (2014). *MQTT Specifications*. URL : http://mqtt.org/documentation, 24 April 2018

N., Hendra. (2015). *Perancangan Penunjuk Rute Pada Kendaraan Pribadi Menggunakan Aplikasi Mobile GIS Berbasis Android Yang Terintegrasi Pada Google Maps.* Manado: Universitas Sam Ratulangi.

Owens, Michael. (2006). *The Definitive Guide to SQLite*. New York: Apress.

Patel K, Patel S. (2016). *Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges*. Gujarat: International Journal of Engineering Science and Computing.

Ramadhanu, Aras R. (2017). *Sistem Pemantau Lokasi Pengangkut Sampah dengan Pemetaan GPS Berbasis Web Server dan SMS*. Jakarta: Universitas Gunadarma.\

S., Navghane. (2016). *IoT Based Smart Garbage and Waste Collection Bin*. International Journal of Advanced Research in Electronics and Communication Engineering.

T., George. (1993). Integrated Solid Waste management. New York: McGraw- Hill. ThingsBoard. ThingsBoard Documentation.

T. Kellog. (2014). Why HTTP wonâA˘Zt´ work for IoT. URL : https://www.edn.com/electronics-blogs/eye-on-iot-/4437056/Why-HTTPWon-t-Work-for-IoT, 7 May 2018

Vermesan O, Friess P. (2013). *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. Norway: River publishers' series in communications.

Vermesan O, Friess P. (2014). *Internet of Things-From Research and Innovation to Market Deployment.* Norway: River publishers' series in communications.

Y., Bahar. 1986. *Teknologi Penanganan dan Pemanfaatan Sampah*. Jakarta: PT Waca Utama Pramaesti.

Zerynth. (2019). DOIT Esp32 DevKitv1. URL : https://docs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html, 30 Oktober 2019.