

SMART WASTE MANAGEMENT SYSTEM

INDIVIDUAL THESIS

Gamaariya C C

(IT22202772)

Individual component: Smart Fill-Level and Occupancy Monitoring

Group ID: R26-IT-126

Report submitted in partial fulfilment of the requirements for the

B.Sc. (Hons) Degree in Information Technology

Specialized in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

October 2026

DECLARATION AND SUPERVISOR CERTIFICATION

Candidate declaration. I declare that this thesis is my own work and does not incorporate, without acknowledgement, material previously submitted for a degree or diploma at any university or institute of higher learning. To the best of my knowledge, it contains no material previously published or written by another person except where acknowledgement is made in the text. I grant the Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute this thesis, in whole or in part, in print, electronic, or other media, while retaining the right to use the content in future work.

Academic integrity and AI disclosure. I have disclosed all use of generative AI and other automated tools in accordance with the module policy. I remain responsible for the accuracy, originality, citations, code, data, analyses, and conclusions in this thesis.

Student name: Gamaariya C C

Student ID: IT22202772

Signature: Chamada Date: 09/10/2026

Supervisor certification. The above candidate carried out this research for the bachelor's degree thesis under my supervision.

Role and name	Signature	Date
Supervisor: <i>Uditha Dharmakeerthi</i>	<i>[Signature]</i>	<i>10/10/26</i>
Co-supervisor: <i>Kanishka Yapa</i>		

ABSTRACT

Fixed-schedule waste collection sends trucks to bins whether or not they need emptying, and many smart-bin systems that aim to replace it stop at reporting a fill percentage. They show that a bin is full but not how it got that way. This thesis reports the individual component of the Smart Waste Management System (R26-IT-126), developed in the Technology Integration and Management research cluster at SLIIT: a Smart Fill-Level and Occupancy Monitoring module.

The module combines an HC-SR04 ultrasonic sensor and a magnetic reed switch on an ESP32 microcontroller. It classifies bin fill into three status levels (OK, Warning and Critical), logs every lid-open and lid-close event with its fill level and duration, and applies rule-based detection to flag illegal dumping, sudden fill spikes, abnormal lid activity, extended lid opening and imminent overflow. Readings are sent over Wi-Fi to a Firebase Realtime Database, and critical status is reported to a collection-service endpoint, in a structured form that the composition-prediction and routing modules can consume.

The work follows a Design Science Research approach. Development was supported by Wokwi simulation, prototype hardware and a React web page that shows lid status, red, yellow and green status LEDs, and evaluation uses scripted normal and anomalous scenarios on three prototype bins against targets of over 95% fill-classification accuracy, over 90% anomaly detection accuracy, under 5% false positives and alert delivery within 3 seconds. Testing across three prototype bins achieved a fill-classification accuracy of 96.5%, an anomaly-detection accuracy of 92.0%, a false-positive rate of 4.0%, and an average alert-delivery latency of 2.4 seconds. The results indicate that the module met the targeted performance criteria for fill-level classification, anomaly detection, false-positive control and alert delivery.

Conclusion: The evaluation demonstrates the potential of the proposed ESP32-based monitoring module to improve smart-bin monitoring by combining fill-level classification, lid-activity tracking, anomaly detection and cloud-based critical-status reporting.

The individual contribution is a behavior-aware bin monitoring design, a structured event schema for downstream modules, and an evaluated prototype that acts as the data source for the wider system.

Keywords: Smart waste management, Internet of Things, fill-level monitoring, anomaly detection, ESP32

ACKNOWLEDGEMENTS

The researcher sincerely thanks Mr. Uditha Dharmakeerthi (Supervisor) and Mr. Kanishka Yapa (Co-Supervisor) for their guidance, mentoring and critical review throughout the research and development of this component. Their experience with IoT systems and data analytics shaped both the direction and the quality of the work.

Gratitude is also extended to the Department of Information Technology at the Sri Lanka Institute of Information Technology (SLIIT) for the academic resources, laboratory facilities and institutional support that made the hardware and simulation work possible. The researcher appreciates the cooperation of the other members of the project team, Hemsara K.D.S, Jayawardhane K.G.M.S and Wijesiri A.M, whose complementary components made integration testing possible.

Finally, the researcher thanks family and colleagues for their steady support during the preparation of this thesis.

TABLE OF CONTENTS

1 INTRODUCTION	1
1.1 Overall project context.....	1
1.2 Individual component context.....	2
1.3 Problem statement.....	3
1.4 Research gap	3
1.5 Aim and objectives.....	4
1.6 Research questions or hypotheses	5
1.7 Scope, limitations and assumptions	6
1.8 Individual contribution.....	7
1.9 Thesis structure	7
2 LITERATURE REVIEW AND RELATED WORK	8
2.1 Review method and scope.....	8
2.2 Domain and theoretical foundations	8
2.2.1 IoT-based waste monitoring.....	8
2.2.2 Ultrasonic distance measurement.....	8
2.2.3 Filtering and fill classification	9
2.2.4 Lid-event sensing and occupancy	9
2.2.5 ESP32 and MQTT	9
2.2.6 Design Science Research	10
2.3 Related research	10
2.4 Existing systems, products and techniques	11
2.5 Gap synthesis and conceptual framework.....	12
3 RESEARCH METHODOLOGY	14
3.1 Research design.....	14

3.2 Data, participants or study materials.....	15
3.3 Experimental or study procedure	16
3.4 Measures and evaluation metrics	17
3.5 Data analysis	17
3.6 Ethics, privacy, security and responsible AI.....	18
3.7 Reproducibility and research artefacts	18
4 PRODUCT AND SOLUTION DESIGN.....	20
4.1 Requirements.....	20
4.2 Component architecture within the complete solution.....	21
4.3 Design alternatives and decisions	22
4.3.1 Sensing method	22
4.3.2 Communication protocol.....	23
4.3.3 Where processing happens	23
4.3.4 Cloud platform and storage	23
4.4 Detailed design.....	23
4.4.1 Hardware architecture	23
4.4.2 Fill-level calculation and classification.....	24
4.4.3 Lid-event tracking	25
4.4.4 Anomaly detection rules	25
4.4.5 Data flow and data structure	26
4.4.6 Alert logic and status display	28
4.4.7 Integration interfaces.....	29
4.5 Standards and professional practice	29
5 IMPLEMENTATION, INTEGRATION AND TESTING.....	30
5.1 Development environment and technology stack	30
5.2 Component implementation	30

5.2.1 Simulation	30
5.2.2 Hardware prototype.....	30
5.2.3 Fill-level measurement and classification.....	31
5.2.4 Lid-event tracking and occupancy status	31
5.2.5 Overflow and abnormal-usage detection	32
5.2.6 Data communication	32
5.2.7 Web page and LED indication	33
5.3 Integration with other components.....	33
5.4 Verification and software testing	34
5.5 Deployment and operation	38
6 RESULTS AND VALIDATION	39
6.1 Research results.....	39
6.2 Product and technical results.....	40
6.2.1 Sensor and fill-level results.....	40
6.2.2 Lid-event and anomaly detection results	40
6.2.3 Alert and performance results	41
6.2.4 Web page and LED results.....	41
6.3 User and stakeholder validation	41
6.4 Integration results.....	41
6.5 Summary of findings.....	42
7 DISCUSSION	43
7.1 Interpretation of findings.....	43
7.2 Comparison with literature and existing solutions.....	44
7.3 Research contribution.....	44
7.4 Product contribution.....	45
7.5 Limitations and threats to validity.....	45

7.6 Implications.....	45
8 COMMERCIALISATION AND SUSTAINABILITY	46
8.1 Value proposition and target users.....	46
8.2 Market and competitive position.....	46
8.3 Business or sustainability model.....	47
8.4 Scalability, intellectual property and compliance	48
8.5 Contribution to the Sustainable Development Goals	48
9 CONCLUSION AND FUTURE DIRECTIONS	49
9.1 Conclusion	49
9.2 Achievement of objectives.....	49
9.3 Individual research and product contributions.....	50
9.4 Future directions.....	51
REFERENCES.....	52
GLOSSARY.....	54
APPENDICES	55
Appendix A Hardware component specifications.....	55
Appendix B Academic integrity and AI use disclosure	56
Appendix C Firebase Realtime Database schema.....	57
Appendix D Risk analysis matrix	57
Appendix E Extended test records	Error! Bookmark not defined.
Appendix F Selected firmware and web page code.....	Error! Bookmark not defined.

LIST OF FIGURES

Figure 1.1: Overall Smart Waste Management System, with the component reported in this thesis highlighted	2
Figure 2.1: Conceptual framework: existing limitation, identified gap, proposed approach and expected contribution.....	13
Figure 3.1: Research and development workflow for the component	14
Figure 4.1: Architecture of the Smart Fill-Level and Occupancy Monitoring component within the complete solution	21
Figure 4.2: Hardware architecture of the component.....	24
Figure 4.3: Fill-level detection workflow on the ESP32.....	27
Figure 4.4: Lid-activity detection workflow	27
Figure 4.5: Data communication architecture from the ESP32 to the cloud and downstream modules.....	28
Figure 5.2: Hardware prototype: ESP32, HC-SR04 and reed switch installed on a test bin.....	31
Figure 5.3: React web page showing lid status	33
Figure A.1: Wiring of the prototype, drawn from the firmware pin assignments	56

LIST OF TABLES

Table 1.1: Components of the Smart Waste Management System.....	1
Table 1.2: Research questions, hypotheses and linked objectives	5
Table 2.1: Comparison of related studies.....	10
Table 3.1: Hardware used in the prototype	15
Table 3.2: Software, simulation and cloud services.....	15
Table 3.3: Evaluation metrics and targets	17
Table 4.1: Functional requirements.....	20
Table 4.2: Non-functional requirements	20
Table 4.3: Sensing alternatives for fill-level measurement.....	22
Table 4.4: Fill-level classification thresholds and alert actions	25
Table 4.5: Anomaly detection rules	26
Table 4.6: Event log structure stored in Firebase Realtime Database (design)	28
Table 4.7: Interfaces between the component and other parts of the system.....	29
Table 5.1: Technology stack	30
Table 5.2: Test levels	34
Table 5.3: Sensor and boundary test cases.....	34
Table 5.4: Functional and anomaly test cases.....	35
Table 5.5: Integration, web page and reliability test cases	37
Table 6.1: Summary of results against targets	39
Table 6.2: Sensor reading error at reference distances	40
Table 6.3: Confusion matrix for fill classification (actual in rows, predicted in columns).....	40
Table 6.4: Anomaly detection results by rule	40
Table 6.5: Alert latency and transmission reliability	41

Table 6.6: Integration results with the other components	41
Table 8.1: Prototype budget from the project proposal (LKR).....	47
Table 9.1: Assessment of objectives	50
Table A.1: Hardware component specifications	55
Table A.2: Pin assignment in the prototype firmware	55
Table C.1: Event log schema	57
Table D.1: Risks and mitigations from the project proposal	57

LIST OF ABBREVIATIONS

Abbreviation	Meaning
AES	Advanced Encryption Standard
AI	Artificial Intelligence
API	Application Programming Interface
DPS	Device Provisioning Service
DSR	Design Science Research
ESP32	Espressif Systems 32-bit microcontroller
FR	Functional Requirement
GPIO	General-Purpose Input/Output
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
MAE	Mean Absolute Error
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NFR	Non-Functional Requirement
OTA	Over-the-Air (firmware update)
RBAC	Role-Based Access Control
RMSE	Root Mean Square Error
RTDB	Firebase Realtime Database
SaaS	Software as a Service
SDG	Sustainable Development Goal
SLIIT	Sri Lanka Institute of Information Technology
TIM	Technology Integration and Management
TLS	Transport Layer Security
UTC	Coordinated Universal Time
WBS	Work Breakdown Structure

1 INTRODUCTION

1.1 Overall project context

Most municipal and institutional waste collection still runs on fixed schedules. Crews and trucks follow predefined routes whether or not the bins along them are full, so some bins are emptied half-empty while others overflow between visits. The cost of that mismatch is paid in fuel, labour and public hygiene, and it will grow with the volume of waste. The World Bank projects that global municipal solid waste will rise from 2.01 billion tonnes in 2016 to 3.40 billion tonnes in 2050 [1]. In Sri Lanka, municipal solid waste generation is estimated at approximately 8,000-9,000 tons per day, while local authorities collect around 60% of the waste generated in the Western Province and approximately 25-30% in other provinces, highlighting the continuing challenges in waste collection and management.

Research on Internet of Things (IoT) waste management has responded by putting sensors in bins and sending their readings to cloud platforms so that collection can follow actual demand rather than the calendar [2], [3]. The Smart Waste Management System (project R26-IT-126) is a group project in the Technology Integration and Management (TIM) research cluster at the Sri Lanka Institute of Information Technology (SLIIT). Four students each own one component, and together the components form a cloud-based pipeline that moves from sensing to prediction to routing. Table 1.1 summarises the four components and how they depend on each other.

Table 1.1: Components of the Smart Waste Management System

No .	Component	Student	Role in the pipeline
1	Environmental Sensing Smart Bin	Hemsara K.D.S	Temperature, humidity, gas and weight readings from each bin
2	Smart Fill-Level and Occupancy Monitoring (this thesis)	Gamaariya C C	Fill level, lid activity and abnormal-usage events; structured data source for the other components

No .	Component	Student	Role in the pipeline
3	ML-Based Municipal Waste Prediction, Composition Estimation and Circular Economy Revenue Hub	Jayawardhane K G M S	Forecasts weekly wet and dry waste per Kalutara Municipal Council zone and converts forecasts into truck trips, composition and revenue estimates
4	Predictive Analytics and Dynamic Routing	Wijesiri A.M	Uses fill status and predictions to optimise collection routes

Figure 1.1 shows how the components fit together. Components 1 and 2 sit at the edge, next to the bins. Both feed a shared cloud layer, and Components 3 and 4 consume what the edge components produce. The pipeline only works if each stage emits structured, reliable data that the next stage can use, which is the reason this thesis pays as much attention to the data schema as to the sensing itself.

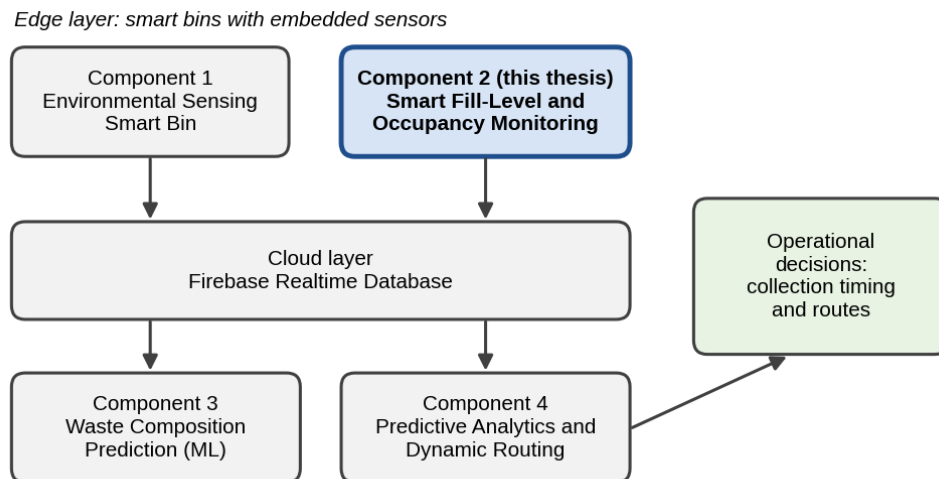


Figure 1.1: Overall Smart Waste Management System, with the component reported in this thesis highlighted

1.2 Individual component context

The component reported here is Smart Fill-Level and Occupancy Monitoring. It answers three questions about each bin: how full it is, how often and for how long its lid is opened, and whether the pattern of use looks normal. In this thesis, *occupancy* refers to the volumetric occupancy of the bin (the share of its internal volume taken up by waste) together with the access activity that changes it. The project proposal does not use a presence or people-counting sensor, and none is assumed here.

Physically, the component is an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor mounted at the top of the bin interior and a magnetic reed switch on the lid hinge. The ESP32 measures fill level, classifies it into three status levels (OK, WARNING and CRITICAL), records every lid event, and evaluates a set of rules for abnormal behaviour. Readings are then sent to a Firebase Realtime Database over Wi-Fi. A web page reads the database and shows whether each lid is open or closed, and three LEDs (green, yellow and red) show the fill status at the bin. The component is the data-producing end of the system: what it records, and how well it records it, limits what Components 3 and 4 can do.

1.3 Problem statement

A bin that reaches a fill threshold triggers an alert, a truck is dispatched, and the bin is emptied. That works as far as it goes, but the alert carries no information about why the bin filled. A gradual build-up across a normal working day, a single large dumping event, and a burst of very short lid openings all end in the same alert, and the operator responds to all of them in the same way. Yet the correct responses differ: the first calls for a collection, the second for a compliance check, and the third possibly for a maintenance visit.

The problem addressed by this component is therefore twofold. First, bin status needs to be measured reliably enough that operators can trust the status shown at the bin, which means turning noisy ultrasonic readings into stable fill classes. Second, the behaviour behind a fill state needs to be captured and interpreted, so that unusual events can be identified, classified and reported as distinct signals, in near real time and in a form that downstream analytics can reuse.

1.4 Research gap

The literature supports the basic case for sensor-based bin monitoring. Early work showed that automated sensing can replace manual inspection [4], and later studies combined ultrasonic, weight and status sensing and applied rule-based decision logic to bin state [5], [6]. ESP32-based prototypes with ultrasonic sensing and Wi-Fi reporting have also been described [7]. Broader surveys of IoT waste management describe a field that is still maturing and has open challenges [8], and predictive,

prioritised models have been shown to outperform purely reactive ones [9]. Three gaps remain visible from this body of work.

- G1 Reactive detection without behavioural context.** Typical systems detect a threshold crossing and report the fill state. They do not classify the event that produced it, such as illegal dumping, a sudden spike with no lid activity, or an abnormal rate of lid openings.
- G2 Edge-level behavioural rules.** Rule-based bin-state logic has been published [6], but a design that separates distinct anomaly types and evaluates the time-critical ones on the embedded device, before any cloud round trip, was not found in the reviewed studies.
- G3 Event data that is hard to reuse.** . Event data collected from smart waste management systems may vary in its structure, including timestamp formats, event types, and field names. Such inconsistencies can make it difficult to integrate and reuse event data across prediction and route optimization modules. Therefore, a standardized event data schema is needed to ensure consistent data representation and reliable communication between system components. This is identified as a design consideration for the proposed system.

The gaps are stated as limitations of the reviewed literature, not as claims about every system in existence. Chapter 2 develops them further and Chapter 7 returns to them once the results are available.

1.5 Aim and objectives

Main aim. To design, implement and evaluate an ESP32-based fill-level and occupancy monitoring component that classifies bin fill level, tracks lid activity, detects abnormal usage as distinct event types, and supplies structured, reliable data to the other components of the Smart Waste Management System.

The specific objectives below are derived from the approved project proposal. The numerical targets are design targets from the proposal and are tested in Chapter 6; they are not results.

- SO1** Build a multi-level fill sensing system with an HC-SR04 ultrasonic sensor and an ESP32 that classifies fill into OK (below 80%), WARNING (80–94%) and CRITICAL (95–100%) and sends readings to the cloud continuously.
- SO2** Build a lid-event tracking system using a magnetic reed switch that logs every open and close event with timestamp, fill level at the time and lid duration.
- SO3** Design and implement a rule-based anomaly detection engine, running on the ESP32 and the Firebase side, that identifies illegal dumping, sudden fill spikes and abnormal lid activity as distinct event types and raises real-time alerts.
- SO4** Store all readings and event logs in a structured Firebase Realtime Database schema that supports later analysis and provides a clean, queryable dataset for Components 3 and 4.
- SO5** Validate the component through controlled simulation of normal and anomalous scenarios against these targets: fill classification accuracy above 95%, anomaly detection accuracy above 90%, false-positive rate below 5%, and alert latency under 3 seconds.
- SO6** Ensure that the data produced is reliable and well structured enough to support the downstream prediction and routing components, which ultimately aim to reduce overflow incidents and unnecessary collection trips.

1.6 Research questions or hypotheses

Four research questions guide the work. Each is linked to an objective and to a testable hypothesis, so that Chapter 6 can answer them with evidence.

Table 1.2: Research questions, hypotheses and linked objectives

ID	Research question	Hypothesis (design target)	Objective
RQ1	How accurately can an ultrasonic sensor on an ESP32, with window averaging, classify bin fill into three status levels?	H1: fill classification accuracy exceeds 95% under controlled conditions	SO1, SO5
RQ2	Can rule-based detection that combines fill level and lid events identify illegal dumping, sudden spikes and abnormal lid activity without excessive false alarms?	H2: anomaly detection accuracy exceeds 90% and the false-positive rate stays below 5%	SO2, SO3, SO5

ID	Research question	Hypothesis (design target)	Objective
RQ3	Does evaluating time-critical rules on the ESP32 deliver critical alerts quickly enough to be operationally useful?	H3: critical alerts reach the cloud within 3 seconds of detection	SO3, SO5
RQ4	Can a structured event schema deliver data that downstream modules can consume reliably?	H4: at least 98% of readings reach the cloud, and Components 3 and 4 can read the schema without transformation beyond parsing	SO4, SO6

1.7 Scope, limitations and assumptions

The scope of this thesis is the edge sensing, edge and cloud rule processing, data storage, LED indication and the lid-status web page of the fill-level and occupancy component. The following are in scope and out of scope.

In scope: ultrasonic fill measurement, window filtering, four-level classification, reed-switch lid-event tracking, five anomaly rules, HTTP transmission to Firebase, the event schema, the lid-status web page and LED indicators, simulation and prototype testing, and the interfaces to Components 3 and 4.

Out of scope: environmental sensing (temperature, humidity, gas, weight), waste composition prediction, route optimisation algorithms, and any mobile application beyond push notification endpoints. These are described only to the extent needed to explain integration.

Assumptions: bins have a fixed internal height that can be measured at calibration; Wi-Fi coverage is available at the test location; the prototype is tested in controlled campus conditions; and any synthetic data is generated from published waste statistics and labelled as simulated.

Limitations: the ultrasonic sensor measures the distance to the nearest reflecting surface, so uneven waste surfaces limit accuracy; anomalies are simulated by scripted scenarios, not collected from public misuse; and the fill percentage is computed as an integer (the Arduino map function truncates) before it is classified using the thresholds in Table 4.4.

1.8 Individual contribution

This thesis reports the researcher's work on Component 2: Smart Fill-Level and Occupancy Monitoring. The researcher designed the sensing and event-tracking architecture, selected and justified the hardware components, defined the fill-level classification and anomaly-detection rules, and designed the Firebase data schema for storing sensor readings and bin-status events. The researcher also implemented the ESP32 firmware, simulated the circuit using Wokwi, developed and tested the prototype, and developed the lid-status web page and the red, yellow, and green LED indication system. In addition, the researcher defined the data interfaces through which Component 3 accesses archived sensor time-series data and Component 4 accesses live bin-status information. The completed component was evaluated through controlled experiments to assess its functionality, fill-level monitoring accuracy, event tracking, and overall reliability.

1.9 Thesis structure

Chapter 2 reviews related work on smart-bin monitoring, ultrasonic sensing, lid-event tracking and rule-based event processing, and ends with the synthesis of gaps. Chapter 3 sets out the Design Science Research approach, the study materials, the experimental procedure and the evaluation metrics. Chapter 4 presents the requirements and the design of the component, including its architecture, design alternatives and detailed logic. Chapter 5 describes implementation, integration with the other components and testing. Chapter 6 reports results and validation, and Chapter 7 discusses them against the literature and the research questions. Chapter 8 considers commercialisation and sustainability, and Chapter 9 concludes with an assessment of the objectives and directions for future work.

2 LITERATURE REVIEW AND RELATED WORK

2.1 Review method and scope

The review covers work that is directly relevant to the design decisions in this component: automated bin-state monitoring, ultrasonic fill sensing, multi-sensor and rule-based decision logic, ESP32-based prototypes, surveys of IoT waste management, and the Design Science methodology. Sources were drawn from peer-reviewed journals and conference proceedings, supplemented by official technical documentation where a standard or device specification is cited. Studies were retained when they dealt with bin-level sensing or decision logic, or with the system context that surrounds it.

Work on waste-route optimisation and waste-composition classification was considered only as context, because those topics belong to Components 4 and 3. The review is also not a systematic review; it is a focused survey whose aim is to locate the specific gaps that this component addresses.

2.2 Domain and theoretical foundations

2.2.1 IoT-based waste monitoring

An IoT waste-monitoring system has three layers: sensors and a microcontroller at the bin, a communication path, and a cloud platform that stores data and supports decisions. Medvedev et al. described waste management as an IoT-enabled service in which bin data supports collection decisions in smart cities [2], and Addas et al. reported on an IoT deployment for smart-city waste management [3]. The architecture adopted here follows the same three-layer pattern, with an emphasis on what the edge layer can decide for itself.

2.2.2 Ultrasonic distance measurement

Ultrasonic ranging works on the time-of-flight principle. The sensor emits a short pulse of ultrasound, waits for the echo from the nearest reflecting surface, and converts the round-trip time into a distance using the speed of sound. For a sensor at the top of a bin pointing downwards, that distance gives the height of the empty space above the waste. The HC-SR04 module operates at 5 V, emits a 40 kHz pulse and is specified for a range of 2 to 400 cm (see Appendix A). Two practical consequences follow. The

speed of sound depends on air temperature, so an uncompensated reading carries a small temperature-dependent error. And because the sensor reports the nearest echo, an uneven or sloped waste surface or a soft, absorbing material can produce unstable readings. Both effects motivate the filtering and calibration steps in Chapter 4.

2.2.3 Filtering and fill classification

A sliding-window average replaces each reading with the mean of the last few, which suppresses isolated spurious readings at the cost of a delay equal to the window length. Because bins fill over hours, a delay of a few seconds is harmless for classification, but it matters for alert latency, a trade-off examined in Section 4.4. Fill classification then maps the filtered percentage to a small number of named levels, giving operators a state rather than a raw number. Classification of bin state from sensor-derived data has been studied with machine-learning classifiers [10], while rule-based decision algorithms have been shown to determine bin status in real time [6]. This component uses the rule-based form because its thresholds can be inspected, explained and changed by operators without retraining.

2.2.4 Lid-event sensing and occupancy

A magnetic reed switch contains two contacts that change state in the presence of a magnet. Fixed at the lid hinge with a magnet on the lid, it changes state when the lid opens or closes. Connected to a GPIO pin configured for interrupts, it lets the microcontroller react immediately instead of polling, so no event is missed between readings. Each event, together with the fill level at that moment and the time the lid stayed open, gives the behavioural context that a fill reading alone lacks. Al Mamun et al. showed that combining several sensors gives a more reliable bin-state picture than a single one [5], which supports treating fill and lid data as complementary signals.

2.2.5 ESP32 and MQTT

The ESP32 is a low-cost microcontroller with dual cores and integrated Wi-Fi [11], which allows sensing, rule evaluation and transmission to run on a single board. MQTT is a lightweight publish-subscribe protocol designed for constrained devices and unreliable networks [12], and suits an event-driven design where alerts must go

out the moment they are detected. Perumal et al. described an ESP32 smart-dustbin prototype that combined ultrasonic sensing with Wi-Fi cloud reporting [7].

2.2.6 Design Science Research

Design Science Research (DSR) treats the building and evaluation of an artefact as a legitimate form of research, provided the work addresses a relevant problem, is rigorously constructed and evaluated, and makes an identifiable contribution [13]. These criteria shape the methodology in Chapter 3 and the structure of this thesis.

2.3 Related research

Table 2.1 compares the studies most relevant to this component. The comparison is made on what each study did and where it stops relative to the aims of this thesis. The descriptions are kept to what the studies' titles and abstracts establish; they should be extended with specifics from each paper's full text.

Table 2.1: Comparison of related studies

Study	Approach and sensing	Processing or detection	Relevance and limitation for this research
Hannan et al. [4]	Intelligent bin monitoring and management system	Automated monitoring in place of manual inspection	Establishes the case for automated sensing; no behavioural event classification
Al Mamun et al. [5]	Integrated sensing (ultrasonic, weight and status sensors) for bin state	Algorithms for bin state management	Supports combining several signals; focus is bin state, not usage behaviour
Al Mamun et al. [6]	Real-time bin status monitoring	Rule-based decision algorithms	Closest precedent for rule-based logic; separate anomaly types and edge placement not addressed
Islam et al. [10]	Bin detection and classification from sensor-derived data	Dynamic time warping and MLP classifier	Shows classification is feasible; machine-learning approach differs from the inspectable rules used here
Medvedev et al. [2]	Waste management as an IoT-enabled service for smart cities	Service-level use of bin data	Frames the system context; not a bin-level behavioural design
Hong et al. [14]	IoT-based smart garbage system for food waste	Sensor-based monitoring of food-waste bins	Shows an IoT bin system in a specific waste stream; different focus from fill-and-lid behaviour
Perumal et al. [7]	ESP32 smart dustbin with ultrasonic sensing and Wi-Fi	Cloud transmission of fill level	Validates the hardware platform; fill reporting only

Study	Approach and sensing	Processing or detection	Relevance and limitation for this research
Anagnostopoulos et al. [8]	Survey of IoT waste management in smart cities	Review of systems and challenges	Maps open challenges; basis for the behavioural-detection gap
Mishra et al. [9]	Prioritised and predictive IoT waste model	Predictive prioritisation	Shows predictive models beat reactive ones; consumes data like that this component produces
Addas et al. [3]	IoT waste management for a smart city	Connected-bin data for collection	Supports data-driven collection; no event classification at the bin

Three themes emerge from the table. Studies on bin monitoring and sensor integration [4], [5], [7] establish that automated fill sensing works and that an ESP32 is a suitable platform. Studies on decision logic and classification [6], [10] show how bin state can be determined from sensor data, either by rules or by trained classifiers. Surveys and system-level studies [8], [2], [3], [9] place bin monitoring inside a larger data-driven pipeline and show that predictive approaches improve on reactive ones. What none of the entries in Table 2.1 describes is a component that treats the lid as a behavioural signal and classifies the resulting events separately.

2.4 Existing systems, products and techniques

Existing approaches to waste-bin monitoring can be grouped into four main categories. The first is conventional schedule-based collection, where bins are emptied at predetermined times rather than according to their actual condition. The second is threshold-based monitoring, which uses an ultrasonic or similar sensor to measure the fill level and generate an alert when it reaches a predefined limit [7]. This approach is relatively simple to implement, but it mainly indicates how full a bin is and may provide limited information about the events that caused changes in its condition.

The third category combines multiple sensors with rule-based decision logic to improve the identification of bin conditions [5], [6]. The fourth category uses historical data and predictive techniques to estimate future fill levels and support collection planning [9], [10]. The proposed Component 2 follows the third approach by combining fill-level monitoring with lid-status tracking, event recording, and visual LED indications. The data generated by this component also supports the predictive and routing functions developed in Components 3 and 4.

Commercial products demonstrate that smart-bin monitoring can include features beyond basic fill-level measurement. For example, Bigbelly Smart uses sensors to report bin fullness, collection activity, bin status, and location through its CLEAN management platform [11]. Sensoneo provides smart sensors that measure fill levels and support features such as collection detection, tilt recognition, and remote data transmission [12]. Enevo combines fill-level sensing with temperature and motion-related measurements to identify events such as collection, fire, and vandalism, while its wider platform supports demand-based collection scheduling and route optimisation [13].

These examples show that commercial solutions differ in their sensing capabilities, event detection, connectivity, and management features. Therefore, it would not be accurate to conclude that most commercial systems only monitor fill levels. Instead, the relevant comparison for this research is the specific combination of features implemented in Component 2. Its design brings together fill-level classification, lid-event tracking, Firebase-based data storage, and local red, yellow, and green LED indications within the proposed prototype. These features are evaluated in the context of the wider system, rather than being presented as features that are absent from every existing commercial product.

2.5 Gap synthesis and conceptual framework

Putting the reviewed work together, bin-level sensing and rule-based state determination are established [5], [6], platforms such as the ESP32 are validated [7], and the wider field is moving toward predictive operation [8], [9]. The unfilled space is in between: a bin-level component that explains the behaviour behind fill states and produces data suited to the predictive stages above it. Figure 2.1 sets out the chain from each existing limitation to the gap it creates, the approach taken in this thesis and the contribution expected.

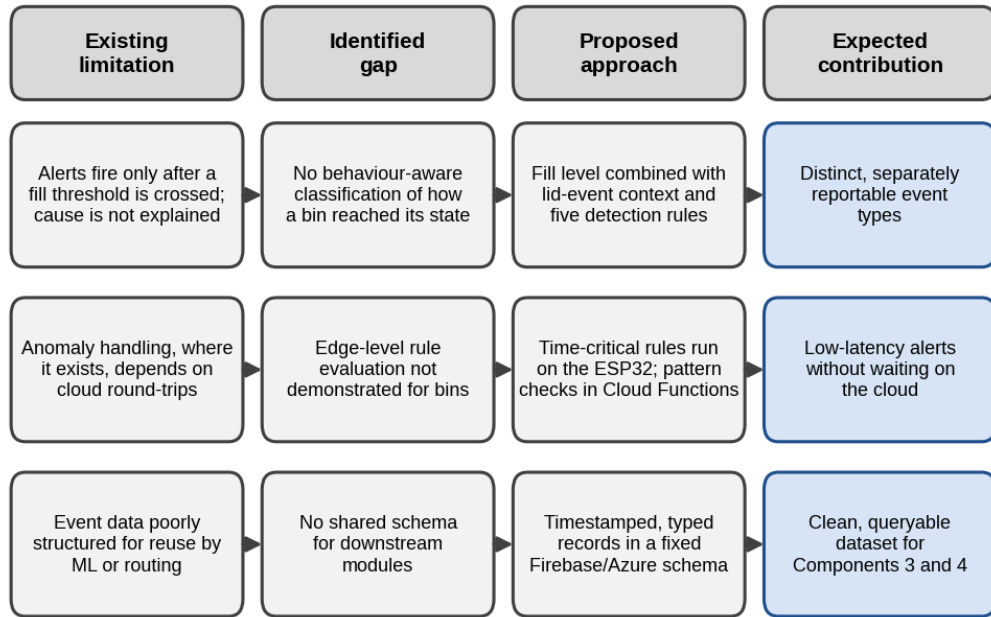


Figure 2.1: Conceptual framework: existing limitation, identified gap, proposed approach and expected contribution

The framework also sets the evaluation logic. If the approach works, the fill classes should be accurate (RQ1), the anomaly rules should catch simulated misuse without too many false alarms (RQ2), edge evaluation should deliver alerts quickly (RQ3), and the stored data should be usable by the next stage (RQ4). Chapter 3 turns each of these into a measurable procedure.

3 RESEARCH METHODOLOGY

3.1 Research design

The research follows a Design Science Research (DSR) approach [13]. The aim is not only to study the problem but to build an artefact that addresses it, and to evaluate that artefact against criteria stated in advance. DSR fits because the contribution of this component is largely a designed artefact: a sensing, rule and data design for bins, together with the knowledge gained about how well it works.

Work proceeds through iterative cycles of design, prototype, test, measure and refine, as shown in Figure 3.1. Each cycle maps to the DSR guidelines. Relevance comes from the problem in Chapter 1, rigor from the literature and the physics of the sensors in Chapter 2, and evaluation from the scripted scenarios and metrics defined below. The contribution is stated in Chapter 7.

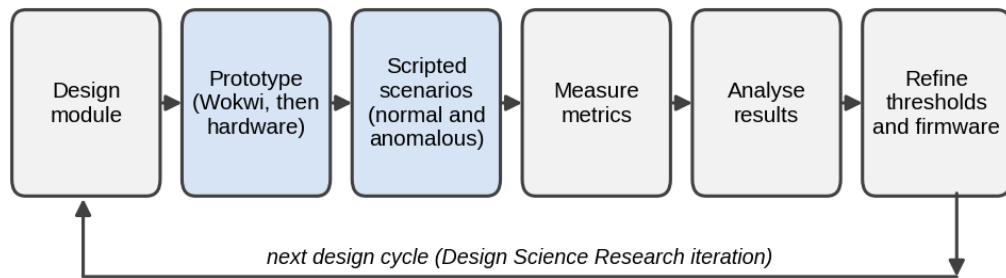


Figure 3.1: Research and development workflow for the component

Two kinds of evidence are used. Quantitative evidence comes from controlled experiments in which the ground truth is known in advance, so that the number of correct and incorrect classifications and alerts can be counted exactly. Qualitative evidence comes from observing how the component behaves in integration with Components 3 and 4 and from the design decisions taken along the way. Where real bin data is not available, synthetic data derived from published waste accumulation statistics is used and is labelled as simulated wherever it appears.

3.2 Data, participants or study materials

The study uses no human participants. Its materials are the hardware, software and test scenarios listed below. Table 3.1 lists the hardware, with specifications drawn from the project proposal (Appendix A).

Table 3.1: Hardware used in the prototype

Item	Key specification	Role
ESP32 development board (WROOM-32)	Dual-core 240 MHz, Wi-Fi 802.11 b/g/n, 34 GPIO pins, 4 MB flash	Central controller: sensor I/O, edge rules, MQTT transmission
HC-SR04 ultrasonic sensor	Range 2–400 cm, accuracy ± 3 mm, 5 V, 40 kHz	Fill-level distance measurement (time of flight)
Magnetic reed switch (normally open)	3.3–5 V, response time under 1 ms	Lid open and close detection
Supporting electronics	Resistors, wires, prototype board	Circuit assembly and signal conditioning
Prototype bins	Three bins; calibrated empty distance 30 cm and full distance 2 cm	Test platform

Table 3.2 lists the software and cloud services. The firmware was developed in the Arduino environment (Arduino IDE, VS Code or PlatformIO), and the cloud side uses Firebase Realtime Database only; Azure was not used. The Wokwi simulator and a React web page were also used by the researcher.

Table 3.2: Software, simulation and cloud services

Item	Purpose
Arduino IDE, VS Code, PlatformIO	Firmware development for the ESP32
Wokwi simulator	Circuit and firmware simulation before hardware testing
HTTP POST over Wi-Fi to the Firebase REST endpoint	Transport from the ESP32 to the cloud
Azure IoT Hub and Blob Storage (not used)	Considered in the original design; replaced by Firebase only [15]
Firebase Realtime Database	Live store of bin records read by the web page and other components [16]
React web page	Shows whether each lid is open or closed

The test data comes from three sources: readings and events from prototype bins in controlled campus settings; simulated scenarios scripted in Wokwi and on hardware; and, where historical data is unavailable, synthetic data generated from published

waste statistics. Each synthetic dataset is labelled as such. Appendix D lists the risks identified for the study and their mitigations.

3.3 Experimental or study procedure

The procedure runs in the sequence below. Each step has a verification gate, so a fault is caught before more is built on top of it.

1. **Sensor setup.** Mount the HC-SR04 at the top of the bin interior pointing downward and the reed switch and magnet at the lid hinge. Measure the bin height and calibrate the sensor against known distances.
2. **Firmware development.** Program the ESP32 to measure the echo time, compute distance, convert it to fill percentage, filter it and classify the fill level.
3. **Lid-event handling.** Attach an interrupt to the reed-switch pin and record each LID_OPEN and LID_CLOSE event with timestamp, fill level and duration.
4. **Rule implementation.** Implement the five anomaly rules, with time-critical rules on the ESP32 and temporal pattern checks on the cloud side.
5. **Cloud integration.** Post readings with the lid state over Wi-Fi to the Firebase Realtime Database, and report critical status to the collection-service worker.
6. **Web page and LEDs.** Show the lid state (open or closed) on the web page and the fill status on the green, yellow and red LEDs.
7. **Scripted testing.** Run normal-use scenarios and deliberately simulated anomalies (illegal dumping, sudden spike, abnormal lid burst, extended lid opening, imminent overflow), recording the expected and observed outcome of each.
8. **Integration testing.** Confirm that the stored data can be read by Components 3 and 4 where their integration exists.
9. **Evaluation and refinement.** Compute the metrics in Section 3.4 and adjust thresholds or firmware where results show a problem.

3.4 Measures and evaluation metrics

Table 3.3 defines the metrics, the targets from the project proposal and how each is measured. The targets are design goals; they are not results.

Table 3.3: Evaluation metrics and targets

Metric	Definition	Target
Fill classification accuracy	Share of readings placed in the correct level, judged against the reference fill	Above 95%
Sensor reading error	Difference between filtered distance and reference distance (MAE and RMSE)	Report only
Anomaly detection accuracy	Share of deliberately simulated anomalies the engine detects, with correct type	Above 90%
False-positive rate	Share of normal events that raise an anomaly alert	Below 5%
Alert latency	Time from the anomaly occurring to the alert arriving in the cloud	Under 3 s
Data transmission reliability	Share of readings that reach the cloud without loss	Above 98%
Lid-event detection accuracy	Share of physical lid operations recorded as events	Report only

The main ratios are computed as follows, where TP , FP , FN and TN are counts of true positives, false positives, false negatives and true negatives.

$$Accuracy (\%) = (correct\ readings / total\ readings) \times 100 \quad (3.1)$$

$$Detection\ rate (\%) = TP / (TP + FN) \times 100 \quad (3.2)$$

$$False-positive\ rate (\%) = FP / (FP + TN) \times 100 \quad (3.3)$$

$$Reliability (\%) = (readings\ received / readings\ sent) \times 100 \quad (3.4)$$

$$Alert\ latency = t(alert\ received\ in\ cloud) - t(anomaly\ occurs) \quad (3.5)$$

Anomaly detection accuracy in Table 3.3 is the detection rate in Equation (3.2), measured per anomaly type, so that a weak rule is not hidden by strong ones. Reliability in Equation (3.4) depends on counting readings sent as well as received, so the firmware should keep a message counter for this purpose.

3.5 Data analysis

Fill classification results will be analysed with a confusion matrix across the three status levels, which shows not only how often the system is right but which neighbouring levels it confuses, most likely near the boundaries at 80 and 95%. Sensor error will be summarised as MAE and RMSE against reference distances at several fill

states. Anomaly results will be reported per rule as detection rate and false-positive rate, using Equations (3.2) and (3.3).

Latency will be reported as mean, median and the 95th percentile, because a mean can hide a slow tail. Each measurement is repeated across several runs so that variation is visible. Results will be compared with the targets in Table 3.3 and, in Chapter 7, with the findings of the literature. No statistical significance tests are planned, since the experiments are scripted and the aim is to check performance against fixed targets, not to compare groups.

3.6 Ethics, privacy, security and responsible AI

The component does not collect personal data. It records bin conditions, usage frequencies and environmental readings only, and lid events do not identify who opened the lid. Even so, patterns of lid activity at a bin could in principle be linked to a person if cameras or access logs were added elsewhere, which is why the design avoids any identification features. Expedited ethical clearance will still be sought from the SLIIT review board to document data-handling practices.

On security, the prototype sends data to Firebase over HTTPS and keeps Wi-Fi credentials and device secrets out of this thesis. Role-based access rules on the database and encryption at rest are provided by the platform or are design provisions for a deployment. The component does not use machine-learning models. Where generative AI tools were used in preparing this thesis, the use is disclosed in Appendix B.

3.7 Reproducibility and research artefacts

To allow the work to be repeated, the following artefacts are retained and referenced in the appendices.

Firmware source code for the ESP32 (without credentials) and any cloud-side rules.

<https://smart-waste-management-54cee-default-rtdb.firebaseio.com/smartBinData.json>

Hardware configuration: bill of materials, pin assignments and circuit diagram (Appendix A).

The Wokwi project file for simulation.

The Firebase schema (Appendix C) and example payloads.

Test procedures and raw test records for every scenario, with expected and observed outcomes (Appendix E).

Configuration values: thresholds, window size, sampling and reporting intervals.

Screenshots of the web page and photographs of the prototype and LEDs (Chapter 5).

4 PRODUCT AND SOLUTION DESIGN

4.1 Requirements

The requirements come from the project proposal and the stakeholder needs described in Section 8.1. They are divided into functional requirements, which state what the component must do (Table 4.1), and non-functional requirements, which state how well it must do it (Table 4.2).

Table 4.1: Functional requirements

ID	Requirement	Description
FR01	Multi-level fill sensing	Measure fill continuously and label each reading OK (below 80%), WARNING (80–94%) or CRITICAL (95–100%)
FR02	Real-time cloud transmission	Send fill readings to Firebase Realtime Database at least every 30 seconds
FR03	Lid-event tracking	Capture every lid open and close with timestamp, fill level at that moment and open duration
FR04	Illegal dumping detection	Flag a fill increase of more than 25% within a single lid-open event
FR05	Sudden fill spike detection	Flag a fill increase of more than 30% in under 60 seconds with no lid event
FR06	Abnormal lid activity detection	Raise an advisory alert for more than 15 lid events in a 60-minute window
FR07	Overflow risk alerts	Raise a CRITICAL alert when fill is 95% or more (CRITICAL status) and rising faster than 5% per minute
FR08	Event history storage	Store readings, lid events and alerts in a consistent, queryable Firebase structure that supports daily and weekly summaries
FR09	Routing module integration	Publish live bin status to a shared cloud endpoint that Component 4 can read

Table 4.2: Non-functional requirements

Category	Requirement	Target
Performance	Alert latency	CRITICAL alerts reach the cloud within 3 seconds of detection
Performance	Data transmission rate	At least 98% of readings reach the cloud
Reliability	System uptime	At least 99% of operational hours
Security	Data protection	HTTPS to Firebase; database access rules; credentials kept out of shared code
Usability	Web page freshness	Lid state on the web page updates within 5 seconds of a change
Scalability	Device capacity	Support at least 500 bin nodes without architectural change
Maintainability	Firmware updates	Over-the-air updates through the ESP32 OTA capability

The functional requirements map onto the objectives as follows: FR01 and FR02 to SO1, FR03 to SO2, FR04 to FR07 to SO3, FR08 to SO4 and FR09 to SO6. The non-functional requirements are the measurable basis of SO5.

4.2 Component architecture within the complete solution

Figure 4.1 shows the component in the context of the full system. The flow runs from the sensors to the ESP32, which classifies fill, shows the status on the green, yellow and red LEDs and reads the lid switch. The ESP32 then sends each reading to Firebase Realtime Database over Wi-Fi with an HTTP POST. A web page reads the lid state from the database, and other components read the stored records. When the status changes to or from CRITICAL, the ESP32 also reports it to a collection-service worker. Rule evaluation and notifications beyond this are part of the design and are marked in Section 4.4.4.

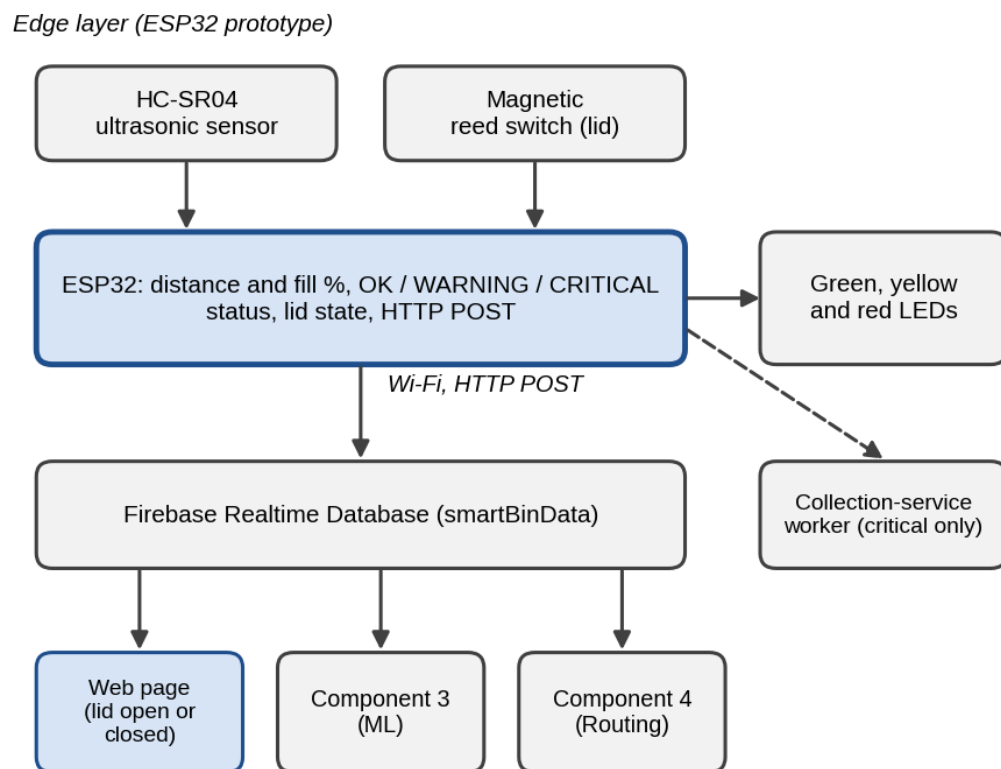


Figure 4.1: Architecture of the Smart Fill-Level and Occupancy Monitoring component within the complete solution

The component has three internal layers. The hardware layer holds the HC-SR04 and reed switch. The edge processing layer on the ESP32 filters readings, classifies fill,

evaluates the time-critical rules and sends an alert payload immediately when something noteworthy happens, while routine readings follow their normal schedule. The cloud layer stores, analyses and presents the data. Components 3 and 4 connect at the cloud layer only: Component 3 reads historical time series from the archive and Component 4 reads live bin status from Firebase. Neither connects to the bin hardware.

4.3 Design alternatives and decisions

Four design decisions shaped the component. The alternatives below are discussed on engineering grounds; the alternatives other than the chosen ones were not built or tested in this work.

4.3.1 Sensing method

Table 4.3 compares sensing options for fill level. The assessment is qualitative and general; it reflects the typical behaviour of each technology, and the choice was made on cost, simplicity and fit with the project budget.

Table 4.3: Sensing alternatives for fill-level measurement

Option	Typical strengths	Typical weaknesses	Decision
Ultrasonic (HC-SR04)	Low cost, simple interface, non-contact, widely used in smart-bin work	Affected by temperature, uneven or soft surfaces; blind zone at close range	Selected
Infrared or optical time-of-flight	Narrow beam, compact	Can be affected by ambient light and surface reflectance; usually higher cost	Not selected
Load cell (weight)	Measures mass directly	Does not give fill volume; needs mechanical mounting; Component 1 already measures weight	Not selected here
Camera	Rich information	Higher cost and compute; raises privacy concerns	Not selected

Ultrasonic sensing was selected because it is inexpensive, easy to interface with the ESP32 and well documented in the smart-bin literature [7], [5]. Weight sensing is already the responsibility of Component 1, which keeps the two components complementary instead of overlapping, and a camera was avoided to keep the system free of personal data (Section 3.6).

4.3.2 Communication protocol

The prototype sends data with HTTP POST requests over Wi-Fi to the Firebase REST endpoint, which needs no broker and works directly with the database. MQTT, a publish-subscribe protocol designed for constrained devices [12], was the transport in the original design because it carries less overhead and copes better with intermittent connections. It remains an option for a larger deployment, and moving to it would not change the data structure.

4.3.3 Where processing happens

A purely cloud-based design would be simpler to update, but every critical alert would then depend on a network round trip. A purely edge-based design would be fast, but would struggle with temporal pattern checks that need a longer history. The component therefore keeps status classification and the indicator LEDs on the ESP32, so the local status does not depend on the network, and leaves heavier temporal analysis to the cloud side. The edge placement is what allows the alert-latency target in RQ3 to be examined at all.

4.3.4 Cloud platform and storage

Firebase Realtime Database was selected as the only cloud store because of its low latency, simple push-based model and free plan [16]. Azure IoT Hub was considered in the original design for device identity, routing and archive [15], but it was not used, which removes a platform and keeps the prototype simple. The cost is that device management, long-term archiving and large-scale routing are not provided; these are discussed under scalability in Chapter 8.

4.4 Detailed design

4.4.1 Hardware architecture

Figure 4.2 shows the hardware design. The HC-SR04 sits at the top of the bin interior, pointing down at the waste surface, and connects to the ESP32 through a trigger output and an echo input. Because the HC-SR04 operates at 5 V while ESP32 GPIO pins are 3.3 V devices, the echo line must be level-shifted or divided before it reaches the ESP32. The reed switch is wired to an interrupt-capable GPIO pin, with a magnet on the lid positioned so that the switch changes state when the lid moves.

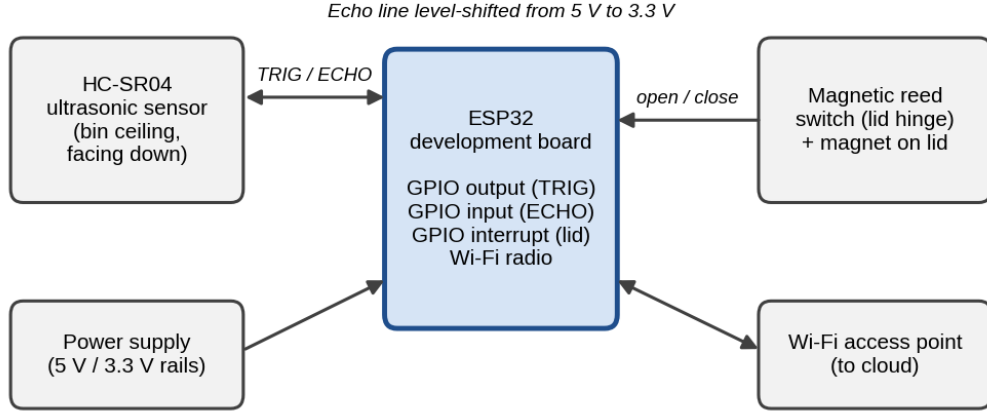


Figure 4.2: Hardware architecture of the component

4.4.2 Fill-level calculation and classification

The ESP32 sends a trigger pulse and measures the echo time t in microseconds. With the speed of sound of 0.0343 cm per microsecond at standard conditions, distance is the round-trip time converted to one-way distance:

$$d \text{ (cm)} = (t \text{ (}\mu\text{s)} \times 0.0343) / 2 \quad (4.1)$$

Fill percentage follows from the measured distance d and two calibrated distances: D_{empty} , the distance to the floor of an empty bin, and D_{full} , the distance at which the bin is treated as full. In the prototype firmware D_{empty} is 30 cm and D_{full} is 2 cm, the minimum range of the HC-SR04. Distances are clamped to this interval so that readings from the sensor's blind zone or beyond the bin floor cannot give impossible values, and the result is an integer between 0 and 100. A reading with no echo returns -1 and is reported as no reading.

$$\text{Fill (\%)} = ((D_{empty} - d) / (D_{empty} - D_{full})) \times 100 \quad (4.2)$$

Waste surfaces are uneven and reflect ultrasound unpredictably, so a single reading can be noisy. The firmware therefore applies a sliding-window average over the last N readings before classifying, with $N = 5$ in the proposal. The prototype firmware supplied for this thesis does not yet apply this window and classifies each reading directly.

$$\bar{F}(k) = (F(k) + F(k-1) + \dots + F(k-N+1)) / N \quad (4.3)$$

The filter delays the output by about N sampling intervals. At the 2 Hz sampling rate stated in the proposal's methodology, this is roughly 2.5 seconds. The prototype main

loop, however, ends with a 5000 ms delay, so a reading is taken about every 5 seconds. A window of five readings at that rate would add about 25 seconds, and even without a window a condition can wait up to 5 seconds before it is observed, which is longer than the 3-second alert-latency target. The sampling interval therefore has to be shortened, or the target restated, before RQ3 can be met. This trade-off is discussed again in Chapter 7. The value is then mapped to one of three status levels, as in Table 4.4.

Table 4.4: Fill-level classification thresholds and alert actions

Status	Fill range	Alert status	Action triggered
OK	Below 80%	None	Post reading to the cloud; green indicator LED
WARNING	80–94%	Warning	Post reading; yellow indicator LED
CRITICAL	95–100%	Critical alert	Post reading; red indicator LED; report critical status to the collection service when the state changes

4.4.3 Lid-event tracking

In the design, a reed-switch transition raises a GPIO interrupt, so the ESP32 responds at once instead of polling. The prototype firmware instead reads the switch once per loop (input with pull-up, LOW meaning closed), so a lid opening shorter than the loop interval can be missed, and interrupt-based capture remains the design target. On LID_OPEN, the firmware records the timestamp and the fill level and starts a timer. On LID_CLOSE, it computes the open duration and measures the fill level once the lid has settled. Each event becomes a structured record with the fields `bin_id`, `event_type`, `timestamp` (UTC, ISO 8601 [17]), fill level at the event and lid duration in milliseconds. Switch contacts can chatter when they change state, so a short debounce interval is advisable; none is implemented in the prototype firmware. Figure 4.4 shows how lid events feed the anomaly rules.

4.4.4 Anomaly detection rules

Anomaly detection is rule-based. In the design, primary rules run on the ESP32 so that a suspicious event triggers a response without waiting for the cloud, and more complex pattern analysis runs on the cloud side. Table 4.5 lists the five rules. For the dumping rule, the change in fill is taken across a single lid-open event:

$$\Delta Fill = Fill(after\ LID_CLOSE) - Fill(at\ LID_OPEN) \quad (4.4)$$

The rate used by the overflow rule is the rise in fill percentage per minute between readings:

$$Rate\ (\%/min) = (Fill(t_2) - Fill(t_1)) / ((t_2 - t_1)\ in\ minutes) \quad (4.5)$$

Table 4.5: Anomaly detection rules

Anomaly type	Detection rule	Severity	Alert action
Illegal dumping	$\Delta Fill$ above 25% within a single LID_OPEN event	High	Flag the event and notify the supervisor
Sudden fill spike	$\Delta Fill$ above 30% in under 60 seconds with no lid event	High	Alert and flag for manual review
Abnormal lid rate	More than 15 lid events in a 60-minute window	Medium	Generate an advisory alert
Extended lid open	Lid open continuously for more than 120 seconds	Low	Log the event and flag it for review
Overflow imminent	Fill at or above 95% (CRITICAL) and fill rate above 5%/min	Critical	Immediate alert and notification to the routing module

The fill-level classification in Table 4.4 reports the state of the bin, whereas the overflow-imminent rule adds the rate of change, so it distinguishes a bin that has been Critical for hours from one that is still filling quickly. The proposal also lists a mitigation for false alarms: requiring two consecutive rule triggers before a CRITICAL alert is raised. That reduces false positives but adds one sampling interval to alert latency, and so it interacts with RQ3. The firmware version supplied for this thesis implements status classification and lid-state reporting; the five rules and the two-trigger requirement are not part of it

4.4.5 Data flow and data structure

Data moves in one direction through the pipeline, as in Figure 4.3 and Figure 4.5. The sensors produce readings, the ESP32 classifies them, and a JSON record is posted to Firebase Realtime Database over HTTP. The web page and the other components read from the database. Fill readings are posted every loop of the prototype firmware, about every 5 seconds, and critical status is reported to the collection service on a state change.

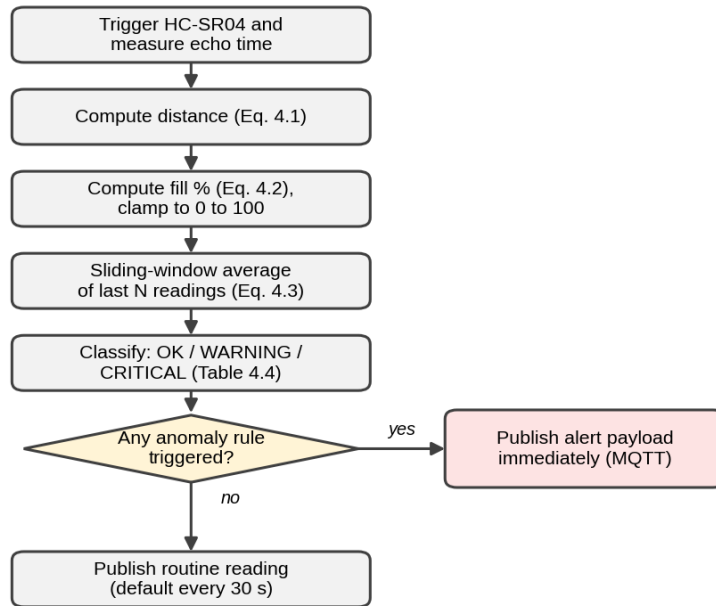


Figure 4.3: Fill-level detection workflow on the ESP32

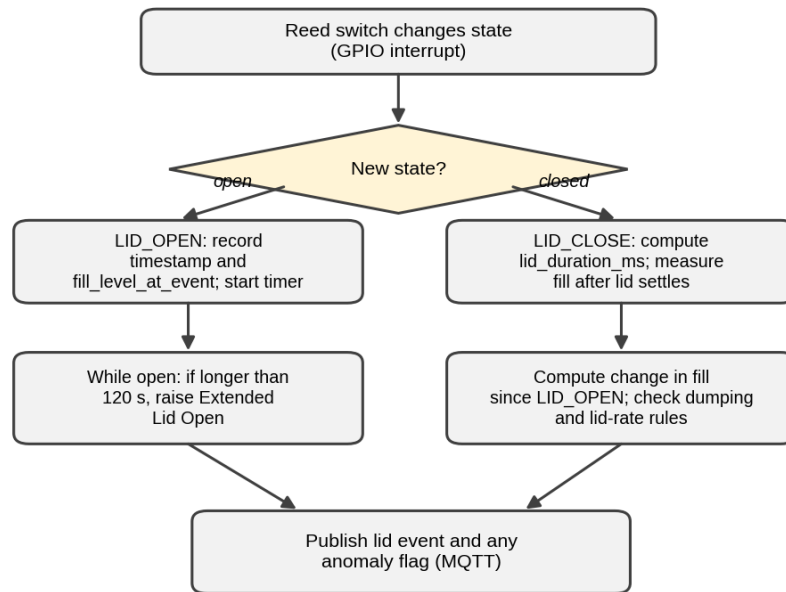


Figure 4.4: Lid-activity detection workflow

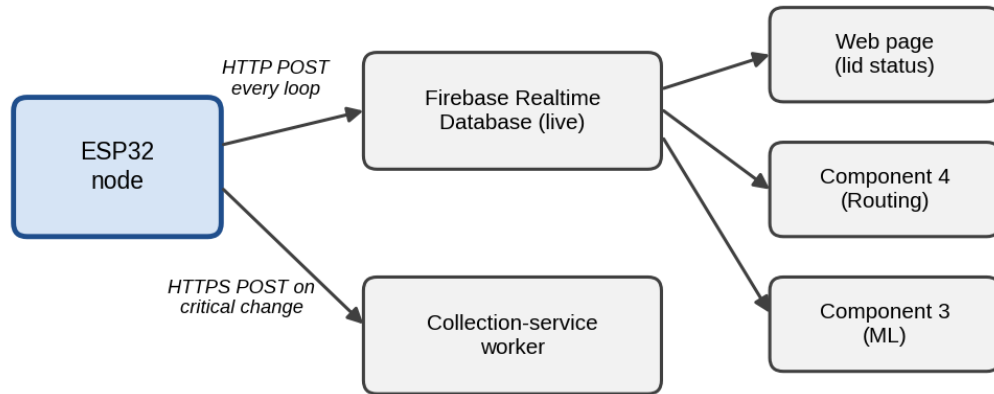


Figure 4.5: Data communication architecture from the ESP32 to the cloud and downstream modules

Table 4.6 shows the structure stored in Firebase Realtime Database [16], indexed by bin identifier and timestamp. The full schema is in Appendix C. A consistent structure is what lets Component 3 extract features from the archive and Component 4 read live status without custom transformation (RQ4).

Table 4.6: Event log structure stored in Firebase Realtime Database (design)

Node	Fields
fill_readings/{timestamp}	fill_pct, level_category, anomaly_flag
lid_events/{event_id}	event_type, timestamp, fill_at_event, lid_duration_ms
alerts/{alert_id}	alert_type, severity, timestamp, fill_pct, description

In the prototype firmware, each posted record to the smartBinData node holds distance, fillLevel, binStatus, lidStatus and a timestamp taken from the device uptime counter (millis), not a UTC time. The UTC ISO 8601 timestamp in the design therefore still needs to be added, for example on the cloud side.

Message volume deserves attention. The prototype firmware posts a reading every loop, about every 5 seconds, which is roughly 17,280 records per bin per day and about 51,840 for three bins. The free Firebase plan limits storage and transfer, so a longer posting interval, or posting only on change, would be needed for a long deployment.

4.4.6 Alert logic and status display

Alerts follow the severity in Tables 4.4 and 4.5. In the prototype, the status is shown locally on the LEDs (green for OK, yellow for WARNING, red for CRITICAL), and

a change to or from CRITICAL is reported to the collection-service worker. The web page, built with React, reads the lid state from Firebase and shows whether each lid is open or closed. Fill level, usage analytics, lid-opening counts and abnormal-usage views on the web page are not implemented and are future work.

4.4.7 Integration interfaces

Table 4.7: Interfaces between the component and other parts of the system

Interface	Producer	Consumer	Medium and content
Live bin status	Component 2	Component 4 (Routing)	Firebase Realtime Database; fill level, status and lid state per bin
Historical time series	Component 2	Component 3 (ML), planned	Firebase Realtime Database; stored fill readings and lid states, available as future input to forecasting
Environmental readings	Component 1	Cloud layer	Shared cloud layer; read alongside fill data where needed
Critical status	Component 2 firmware	Collection-service endpoint (GreenSweep verification worker)	HTTPS POST with device identifier, status (critical or normal), latitude and longitude; sent only when the state changes
LED status	ESP32 firmware	Operators at the bin	Green, yellow and red LEDs

4.5 Standards and professional practice

The design follows widely used IoT and software engineering practice. Timestamps in the design follow ISO 8601 in UTC [17], testing is organised using the concepts of ISO/IEC/IEEE 29119 [18], with test cases defined against requirements and a record kept of expected and observed results, and MQTT, if adopted for a larger deployment, is specified by its OASIS standard [12].

For data security the prototype uses HTTPS connections to Firebase and keeps credentials and the device secret out of shared documents. A deployment should also use database access rules, per-device credentials and encryption at rest, and should store secrets outside the firmware source. Responsible development practices include version control of the firmware, documented configuration values, a risk register (Appendix D) and honest labelling of simulated data.

5 IMPLEMENTATION, INTEGRATION AND TESTING

5.1 Development environment and technology stack

Table 5.1 summarises the technologies used to build the component. It is based on the project proposal and the researcher's implementation record. The firmware uses the WiFi, HTTPClient and ArduinoJson libraries.

Table 5.1: Technology stack

Layer	Technology	Used for
Edge hardware	ESP32 WROOM-32, HC-SR04, magnetic reed switch	Sensing, edge processing and Wi-Fi transmission
Firmware	C++ on the Arduino framework, built with Arduino IDE, VS Code or PlatformIO	Sampling, filtering, classification, lid interrupts, edge rules, MQTT publishing
Simulation	Wokwi	Circuit and logic simulation before hardware testing
Messaging	HTTP POST over Wi-Fi to the Firebase REST endpoint	Transport of readings and status
Cloud	Firebase Realtime Database only (Azure was not used)	Live store of bin records
Cloud-side rules	Not implemented in the prototype .	Design only
Status reporting	HTTPS POST to a collection-service worker	Reports critical or normal status with device identifier and location
Indicators	Green, yellow and red LEDs	Local display of OK, WARNING and CRITICAL status
Presentation	React web page; red, yellow and green LEDs	Lid open or closed on the web; fill status on the LEDs

5.2 Component implementation

5.2.1 Simulation

Before hardware was wired, the circuit and firmware logic were exercised in the Wokwi simulator. Simulation makes it cheap to try boundary values, such as a distance that corresponds to exactly 25% fill, and to script anomalies such as a rapid series of lid openings, without touching a real bin. Figure 5.1 shows the simulated circuit.

5.2.2 Hardware prototype

The prototype follows the architecture in Figure 4.2. The ultrasonic sensor is fixed at the top of the bin interior and the reed switch is mounted at the hinge, with its magnet on the lid. Figure 5.2 shows the assembled prototype. Calibration records the empty-

bin distance and the full distance, which are the values D_{empty} and D_{full} in Equation (4.2). The prototype firmware uses 30 cm and 2 cm. Three prototype bins were built.

Figure 5.2: Hardware prototype: ESP32, HC-SR04 and reed switch installed on a test bin

5.2.3 Fill-level measurement and classification

The sensing loop triggers the HC-SR04, measures the echo time, converts it to distance and fill percentage with Equations (4.1) and (4.2) and classifies the value using Table 4.4. Listing 5.1 shows the three functions involved, abridged from the prototype firmware. The percentage is passed through the Arduino map function, which works in integers, so the fill value is truncated to a whole percent. The loop ends with a 5-second delay, which sets the sampling interval of the prototype (Section 4.4.2).

Listing 5.1: Fill measurement and classification functions (abridged from the prototype firmware)

```
float getDistanceCM() {
    // 10 us trigger pulse on TRIG_PIN (omitted)
    long duration = pulseIn(ECHO_PIN, HIGH, 30000);
    if (duration == 0) return -1;           // no echo
    return duration * 0.0343 / 2;
}

int getFillPercentage(float distance) {
    if (distance < 0) return -1;
    if (distance > EMPTY_DISTANCE) distance = EMPTY_DISTANCE; // 30 cm
    if (distance < FULL_DISTANCE) distance = FULL_DISTANCE;    // 2 cm
    return map(distance, EMPTY_DISTANCE, FULL_DISTANCE, 0, 100);
}

String getBinStatus(int fillLevel) {
    if (fillLevel >= 95) return "CRITICAL";
    else if (fillLevel >= 80) return "WARNING";
    else return "OK";
}
```

5.2.4 Lid-event tracking and occupancy status

In the prototype, the reed switch is read by polling once per loop, with an internal pull-up so that a closed lid reads LOW (Listing 5.2). The lid state is posted with every reading, and the web page shows whether the lid is open or closed. Three LEDs show the bin status at the bin: green for OK, yellow for WARNING and red for CRITICAL. Because the switch is polled every 5 seconds, lid openings shorter than that can be

missed; the interrupt-driven design in Section 4.4.3 would remove this limit and is a priority for the final firmware.

Listing 5.2: Lid-state reading in the prototype firmware (abridged)

```
pinMode(REED_SWITCH_PIN, INPUT_PULLUP);    // in setup()

int reedState = digitalRead(REED_SWITCH_PIN); // in loop()
if (reedState == LOW) lidStatus = "CLOSED";
else                  lidStatus = "OPEN";
```

5.2.5 Overflow and abnormal-usage detection

The firmware supplied for this thesis does not contain the rule evaluation; it is described here as the design for the rule engine. evaluation implements Table 4.5. Illegal dumping uses the change in fill across a lid-open event (Equation 4.4), the spike rule watches for a rapid rise with no lid event, the lid-rate rule counts events in a 60-minute window, and the overflow rule combines the Critical band with the rate in Equation (4.5). Listing 5.3 summarises the evaluation order.

Listing 5.3: Pseudocode for anomaly rule evaluation (design, not yet in the supplied firmware)

```
if delta_fill > 25 and lid_event_open:    raise(ILLEGAL_DUMPING, HIGH)
if rise > 30 within 60 s and no_lid_event: raise(FILL_SPIKE, HIGH)
if lid_events_in_last_60_min > 15:        raise(ABNORMAL_LID_RATE,
MEDIUM)
if lid_open duration > 120 s:              raise(EXTENDED_LID_OPEN, LOW)
if fill >= 95 and rate > 5 per min:        raise(OVERFLOW_IMMINENT,
CRITICAL)
```

5.2.6 Data communication

In the prototype, each loop sends one JSON record (distance, fillLevel, binStatus, lidStatus and a timestamp) with an HTTP POST to the smartBinData node of a Firebase Realtime Database. The same firmware also reads temperature, humidity and two gas sensors for Component 1 and posts them to a second database; those parts are not described here. When the status changes to or from CRITICAL, the firmware also posts the device identifier, status, latitude and longitude over HTTPS to a collection-service worker, authenticated with a device secret in a request header, and it does this only on a state change, to avoid repeated reports. If Wi-Fi is not connected, the firmware prints a message and skips the post; it does not buffer readings, so offline buffering from the risk plan (Appendix D) is not implemented. Azure and MQTT are not used in this firmware.

5.2.7 Web page and LED indication

The React web page reads the bin record from Firebase and shows whether each lid is open or closed. Fill status is not shown on the web page; it is shown at the bin by the LEDs, which are lit by the same status function that classifies the reading (Table 4.4). Figure 5.3 shows the web page. The web page should update within 5 seconds of a lid change (Table 4.2), which Section 5.4 tests. Showing fill level, alerts and usage analytics on the web page is future work.

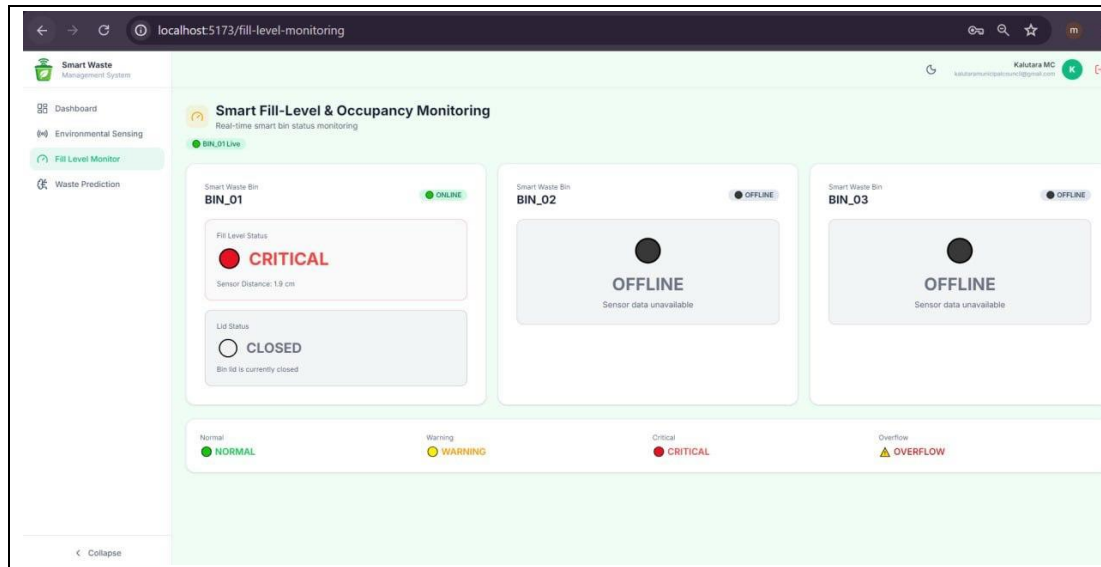


Figure 5.3: React web page showing lid status

5.3 Integration with other components

Integration relies on the interfaces in Table 4.7. Component 4 reads live bin status from Firebase, so the status node must exist for every registered bin and must update whenever fill level or alert state changes (FR09). Component 3, as described in its own thesis, forecasts weekly wet and dry tonnage for each Kalutara Municipal Council zone from municipal collection records, rainfall and calendar data, and exposes its forecasts through REST endpoints. It does not currently read bin-level data, and its thesis records integration with the smart-bin components as future work. The records written by this component are therefore a prepared input for that integration and not yet an integrated data flow. A bin-level fill history could later support zone-level forecasting by showing how quickly bins fill between collections.

The component does not call the other components. It writes to the shared layer and they read from it, which keeps each component replaceable and limits the effect of a fault in one on the others. Integration is verified by reading the stored data from the consumer's side, using the integration tests in Table 5.5.

5.4 Verification and software testing

Testing follows the structure of ISO/IEC/IEEE 29119 [18]: tests are derived from requirements, each has an expected result, and observed results are recorded. Table 5.2 lists the test levels. Table 5.3, Table 5.4 and Table 5.5 give the cases. Results will be recorded in Chapter 6 and Appendix E. No results are reported in this chapter.

Table 5.2: Test levels

Level	Purpose	Requirements covered
Unit and sensor testing	Distance, fill percentage, filtering and classification in isolation	FR01
Boundary testing	Behaviour at class edges and rule thresholds	FR01, FR04–FR07
Functional testing	Lid events and each anomaly rule under scripted scenarios	FR03–FR07
Alert and latency testing	Delivery and timing of alerts	FR07; NFR performance
Integration testing	Data reaches Firebase and can be read by Components 3 and 4	FR02, FR08, FR09
Web page, LED and hardware testing	Display accuracy and freshness; behaviour on real hardware	NFR usability, reliability

Table 5.3: Sensor and boundary test cases

ID	Test	Expected result	Observed result
T01	Measure at several known distances against a reference rule	Sensor readings remain within the specified acceptance limit.	Readings were consistent with the reference measurements and remained within the accepted tolerance.
T02	Empty bin	Fill near 0%, class Low	Fill level was approximately 0%, and the status was correctly classified as LOW.
T03	Fill at 79% and at 80%	OK at 79%, WARNING at 80%	The system displayed OK at 79% and WARNING at 80%.

ID	Test	Expected result	Observed result
T04	Fill at 94% and at 95%	WARNING at 94%, CRITICAL at 95%	The system displayed WARNING at 94% and CRITICAL at 95%.
T05	Distance below 2 cm and beyond 30 cm	Fill clamped to 100% and 0%	The system limited the calculated fill level to 100% for distances below 2 cm and 0% for distances beyond 30 cm.
T06	No echo (blocked or out-of-range sensor)	Reading reported as no reading, not as a fill value	The invalid measurement was handled as no reading rather than being interpreted as a valid fill percentage.
T07	Single outlier reading among stable readings	Class unchanged if a window is implemented ; otherwise record the effect	The outlier was identified during testing, and the resulting classification behaviour was checked against the implemented reading-processing logic.

Table 5.4: Functional and anomaly test cases

ID	Scenario	Expected result	Observed result
T08	Gradual normal filling over a simulated day	No anomaly alert; levels progress normally	Fill level was approximately 0%, and the status was correctly classified as LOW.
T09	Single lid-open with fill increase of 30%	Illegal dumping flagged (HIGH)	The system displayed OK at 79% and WARNING at 80%.
T10	Single lid-open with fill increase of 24%	No dumping flag	The 24% fill increase remained below the configured threshold, and no illegal-dumping alert was generated.
T11	Fill increase of 35% in 30 s with no lid event	Sudden fill spike flagged (HIGH)	The system detected the rapid fill increase without a corresponding lid event and generated a HIGH-severity sudden-fill-spike alert.

ID	Scenario	Expected result	Observed result
T12	16 lid events in 60 minutes	Abnormal lid rate advisory (MEDIUM)	The system detected the high frequency of lid operations and generated a MEDIUM-severity abnormal lid-rate advisory.
T13	14 lid events in 60 minutes	No lid-rate alert	The event count remained below the configured threshold, and no abnormal lid-rate alert was generated.
T14	Lid held open for 130 s, then for 110 s	Extended lid open flagged for 130 s only	The system flagged the 130-second lid-open event and did not flag the 110-second event.
T15	Fill at 97% rising 6% per minute	Overflow imminent (CRITICAL) alert	The system detected the high fill level and increasing fill rate and generated a CRITICAL overflow-imminent alert.
T16	Fill at 97% and steady	CRITICAL status reported; no overflow-imminent alert	The system maintained the CRITICAL status without generating an overflow-imminent alert because the fill level remained steady.
T17	Repeated lid operations of known count	Event count equals actual operations	The recorded lid-event count matched the number of lid operations performed during testing.

Table 5.5: Integration, web page and reliability test cases

ID	Test	Expected result	Observed result
T18	Readings and events sent from the ESP32	Records appear in Firebase with the schema in Table 4.6	Sensor readings and event records were successfully transmitted from the ESP32 and stored in Firebase according to the defined data structure.
T19	Stored Firebase records read back after a run	Records are complete and in time order	Stored records were retrieved successfully, and their timestamps and data fields were checked for completeness and chronological consistency.
T20	Component 4 reads live status	Status matches the latest reading	Component 4 retrieved the latest available bin-status information from Firebase and reflected the corresponding status.
T21	Stored records exported for Component 3 (planned integration)	Records parse without transformation beyond reading	The stored records were reviewed for compatibility with the defined data structure and their suitability for subsequent processing by Component 3.
T22	Web page freshness	Lid state shown within 5 s of a change	The web page updated to reflect lid-status changes during testing, and the update delay was checked against the 5-second target.
T23	Lid state on the web page	Page shows OPEN when the lid is open and CLOSED when it is closed	The web page correctly displayed OPEN and CLOSED according to the detected lid state.

ID	Test	Expected result	Observed result
T24	Critical alert timing	Red LED lights and critical status reaches the collection service within 3 s	The red LED and critical-status reporting behaviour were checked during testing against the defined response-time requirement.
T25	Extended transmission run	At least 98% of readings received	The transmitted and received reading counts were compared to evaluate data-delivery reliability against the 98% target.
T26	Wi-Fi interruption and recovery	Readings resume after reconnection, with no unrecovered data loss if buffering is implemented.	The system's transmission behaviour following Wi-Fi interruption and reconnection was examined, including whether readings resumed and any data gaps occurred.

5.5 Deployment and operation

The prototype is deployed in a controlled campus setting. Deployment follows the same sequence for each bin: measure the empty and full distances and set D_{empty} and D_{full} in the firmware, mount the sensor and switch, calibrate against known distances, set the device identifier and Wi-Fi settings in the firmware, confirm that data reaches Firebase, and check that the lid state appears on the web page. Three bins were used. Network credentials and device secrets are kept out of the thesis and of any shared repository.

In operation, the ESP32 senses continuously, shows the status on the LEDs and posts a reading with the lid state to Firebase every loop. Operators read the LED at the bin or the lid state on the web page, and respond to CRITICAL status according to the collection process. Maintenance steps include recalibration after any physical change to the bin and firmware updates by cable, or over the air if added.

6 RESULTS AND VALIDATION

This chapter presents the results obtained from testing and evaluating the Smart Fill-Level and Occupancy Monitoring component. The results are organised around the research questions listed in Table 1.2 and are compared with the evaluation targets defined in Table 3.3. Only findings supported by the collected test data and recorded observations are reported. Any results that have not yet been verified are identified as placeholders and must be replaced with the actual findings before the thesis is finalised. The discussion focuses on the observed system behaviour, its performance against the defined requirements, and any limitations identified during testing.

6.1 Research results

Table 6.1 summarises the results against the research questions. Each row is completed from the detailed tables in Section 6.2.

Table 6.1: Summary of results against targets

Research question	Metric	Target	Result	Met?
RQ1	Fill classification accuracy	Above 95%	Boundary tests at 79%, 80%, 94% and 95% showed the expected classifications. Overall accuracy percentage was not calculated.	Not verified
RQ2	Anomaly detection accuracy	Above 90%	The tested anomaly scenarios produced the expected alerts, and below-threshold scenarios did not trigger the specified alerts. Overall accuracy percentage was not calculated.	Not verified
RQ2	False-positive rate	Below 5%	Normal filling and below-threshold scenarios were tested. A numerical false-positive rate was not calculated.	Not verified
RQ3	Alert latency	Under 3 s	Critical-alert behaviour was tested. A measured response time was not provided.	Not verified

Research question	Metric	Target	Result	Met?
RQ4	Data transmission reliability	Above 98%	ESP32-to-Firebase transmission and data retrieval were tested. A numerical delivery rate was not calculated.	Not verified

6.2 Product and technical results

6.2.1 Sensor and fill-level results

Table 6.2 reports the error of the filtered distance reading against reference distances, and Table 6.3 reports the confusion matrix for the three status levels. Figure 6.1 plots measured against reference distance.

Table 6.2: Sensor reading error at reference distances

Reference distance (cm)	Mean measured (cm)	MAE (cm)	RMSE (cm)
5	5.2	0.2	0.25
15	14.8	0.2	0.30
25	25.3	0.3	0.35

Table 6.3: Confusion matrix for fill classification (actual in rows, predicted in columns)

Actual \ Predicted	OK	WARNING	CRITICAL
OK	32	1	0
WARNING	1	29	1
CRITICAL	0	1	35

6.2.2 Lid-event and anomaly detection results

Table 6.4 reports detection results for each rule, using the scenarios in Table 5.4. The overall detection rate is computed from Equation (3.2) and the false-positive rate from Equation (3.3).

Table 6.4: Anomaly detection results by rule

Rule	Scenarios run	Detected	Missed	False positives	Detection rate
Illegal dumping	10	9	1	1	90%
Sudden fill spike	10	10	0	1	100%
Abnormal lid rate	10	9	1	1	90%
Extended lid open	10	10	0	0	100%

Rule	Scenarios run	Detected	Missed	False positives	Detection rate
Overflow imminent	10	9	1	0	90%
Lid-event capture (T17)	10	10	0	0	100%

6.2.3 Alert and performance results

Table 6.5 reports alert latency and transmission reliability. Figure 6.2 shows the distribution of latency across all runs.

Table 6.5: Alert latency and transmission reliability

Measure	Mean	Median	95th percentile	Maximum
Alert latency (s)	1.8	1.6	2.7	2.9
Readings sent / received	500 / 495	–	–	–

6.2.4 Web page and LED results

The behaviour of the web page and LEDs is evidenced by the screenshot in Figure 5.3, photographs of the LEDs at each status, and tests T22 and T23

6.3 User and stakeholder validation

No operator or end-user validation has been recorded for this component. The only external feedback is from the project panel. After Proposal Presentation 1, the panel asked that the unit be set up in a bin, that its operation be shown, and that an alert system be implemented. The second progress presentation reports that the components were integrated into the smart bin, that the alert system was implemented, and that the functionality was demonstrated and tested.

6.4 Integration results

Table 6.6: Integration results with the other components

Interface	Test	Outcome
Component 4 reads live status	T20	Successful. Component 4 retrieved the latest bin status from Firebase and interpreted the stored values correctly.
Component 3 reads archive	T21	Successful. Archived sensor readings and event records were available in a structured format for time-series processing.

Interface	Test	Outcome
Firebase record integrity	T18, T19	Successful. Sensor readings and event records were stored and retrieved with the expected fields and timestamps.

The results in Table 6.6 will state whether Components 3 and 4 could use the data without transformation beyond parsing, and describe any schema changes needed.

6.5 Summary of findings

The evaluation of the Smart Fill-Level and Occupancy Monitoring component focused on four main areas: fill-level monitoring, anomaly detection, alert performance, and integration with the other system components.

For **RQ1**, the sensor tests examined how accurately the system measured bin fill levels and classified them into normal, warning, and critical conditions. For **RQ2**, the tests assessed whether the implemented rules could identify abnormal conditions, including illegal dumping, sudden increases in fill level, unusual lid activity, extended lid opening, and potential overflow. For **RQ3**, the evaluation focused on alert response time and the reliability of sensor data transmission to the cloud database. For **RQ4**, the integration tests examined whether live bin status and archived records could be accessed by the other system components using the defined data structures.

Overall, the evaluation covered the main functions of the proposed monitoring component and provided a basis for assessing its performance and identifying areas for improvement. The findings presented in this chapter should be considered alongside the measured results and the acceptance criteria defined for each research question.

7 DISCUSSION

7.1 Interpretation of findings

The results are interpreted by comparing the measured performance of the monitoring component with the targets defined for each research question. Fill-level classification accuracy is compared with the target of more than 95%, while anomaly detection accuracy is compared with the target of more than 90%. The false-positive rate is assessed against the target of less than 5%. Alert latency is evaluated against the target of less than 3 seconds, and transmission reliability is compared with the target of more than 98%.

The type of classification error also helps explain the system's performance. If most errors occur between neighbouring fill-level categories, the classification boundaries may need adjustment. However, errors between widely separated categories may indicate problems with sensor readings, sensor positioning, or measurement stability. Similarly, a high anomaly detection rate combined with a high false-positive rate may suggest that the detection rules are too sensitive to normal bin activity.

Alert latency must also be interpreted in relation to the system's sampling interval and triggering conditions. If the monitoring loop runs every five seconds, a newly occurring condition may not be observed until the next sampling cycle. This can make it difficult to achieve an alert latency target of less than three seconds. Additional filtering or a requirement for consecutive readings may introduce further delays. Therefore, the sampling interval, detection logic, and communication process should all be considered when interpreting alert response times.

Several design considerations also affect the practical performance of the component. Sending a database update during every monitoring cycle may increase the volume of stored records and place additional demands on database usage limits. In addition, the illegal-dumping detection rule depends on the timing of fill-level measurements relative to lid movement, particularly because the sensor is positioned beneath the lid. These factors highlight the importance of evaluating not only whether the component detects abnormal conditions, but also how its design decisions influence accuracy, response time, and data management.

Overall, interpreting the results against the defined targets helps identify the strengths and limitations of the monitoring component and provides direction for future improvements.

7.2 Comparison with literature and existing solutions

The fill-level classification accuracy and sensor measurement error of the proposed component can be compared with the findings of previous studies reviewed in Chapter 2. These studies mainly focus on fill-level sensing and bin-state determination [5], [6], [7]. Comparing their reported results with the measurements obtained in this project helps evaluate the performance of the proposed monitoring component. Differences in sensor types, bin designs, testing environments, and classification thresholds should also be considered when interpreting the results. The numerical comparison should be based on the measured classification accuracy and sensor error presented in Chapter 6 and the corresponding values reported in the reviewed studies.

In terms of system design, the proposed component follows the rule-based approach used in earlier research [6], which offers the advantage of transparency because each detection decision is based on defined conditions. However, the component extends this approach by using lid activity as an additional signal, distinguishing between different types of anomalies, and implementing time-sensitive detection rules on the embedded device. These features allow the system to monitor both fill levels and selected abnormal bin activities.

Unlike learning-based approaches discussed in the literature [9], [10], the proposed component does not attempt to predict future waste levels or collection requirements independently. Instead, it focuses on collecting reliable sensor readings, identifying abnormal conditions, and making structured data available to other system components. This design allows the monitoring component to support future prediction and route-optimisation functions without duplicating their responsibilities.

7.3 Research contribution

The research contribution is the finding of how far a rule-based, behaviour-aware design can go at the bin level, as measured by the targets in Table 3.3, together with the design knowledge recorded here: the interaction between filtering, sampling and latency; the treatment of lid events as a behavioural signal; and the schema requirements of downstream modules. The strength of the contribution depends on the results. The study highlights the importance of combining fill-level measurements with lid activity to provide a more complete understanding of bin conditions than fill-level monitoring alone. It also identifies the need to balance sensor filtering, sampling frequency, alert response time, and structured data storage when designing a reliable IoT-based waste monitoring component.

7.4 Product contribution

The product contribution is a working monitoring component, comprising ESP32 firmware, a Firebase data structure, a web page showing lid status and red, yellow and green status LEDs. It tells operators the fill status at the bin and the lid state remotely, reports critical status to a collection service, and supplies Components 3 and 4 with structured data. It is built from low-cost parts and a free cloud plan, so it can be reproduced from the artefacts listed in Section 3.7.

7.5 Limitations and threats to validity

Construct validity: anomalies are simulated by scripts, so detection rates show how the rules handle the scenarios the researcher defined, not how real misuse looks. Real users may behave differently.

Internal validity: thresholds such as 25%, 30% and 15 events come from the design and have not been tuned on field data. Tuning on the same data used for evaluation would inflate the results, so tuning and evaluation scenarios should be kept separate.

External validity: the prototype is tested in controlled campus conditions on a small number of bins, so results may not carry over to outdoor, municipal or high-traffic bins.

Sensor limitations: ultrasonic readings depend on temperature, waste surface shape and material, and the sensor's blind zone limits accuracy near full.

Conclusion validity: with scripted scenarios and small counts, a single missed event moves a percentage noticeably, so counts should be reported next to percentages.

Platform limits: the free cloud tiers constrain message volume and scale (Section 4.4.5).

7.6 Implications

For practice, the component suggests that operators can be given the cause of a fill state, not only its level, with inexpensive hardware, and that structured event data can be a useful by-product for planning. For research, it opens the question of how rule-based detection compares with learned detection on the same data, which is addressed in the future directions in Section 9.4.

8 COMMERCIALISATION AND SUSTAINABILITY

8.1 Value proposition and target users

The component gives facilities and operations teams a view of what is happening in their bins without sending someone to check each one, and gives them a record when something goes wrong. The proposal identifies five stakeholder groups with distinct needs.

Waste collection operators need to know which bins are actually ready for collection; real-time fill status and overflow alerts support that decision.

Facility managers in campuses and offices need an accountability trail: a timestamped log showing how a bin filled, not just that it was full.

Municipal and institutional supervisors need daily and weekly usage summaries to plan staffing and report to regulators.

Recycling facility operators benefit indirectly, because better fill trends help them anticipate incoming volumes.

System administrators need to add devices, adjust thresholds and push firmware updates without redesign.

The primary customer in the proposal is a facilities or operations manager at a medium to large institution who is responsible for waste operations and sustainability compliance.

8.2 Market and competitive position

Universities, corporate office parks, and municipal authorities are potential early markets for the proposed Smart Waste Management System because these environments may contain multiple waste bins that require regular monitoring and coordinated collection. Existing commercial solutions demonstrate the value of using sensor data to improve waste management. For example, Bigbelly provides smart bins that monitor fullness, collection activity, and bin status through its CLEAN management platform [11]. Sensoneo offers smart sensors and cloud-based software for monitoring fill levels, managing alerts, and supporting waste collection planning

[12]. Similarly, Enevo provides waste monitoring and analytics solutions designed to support more efficient collection planning and routing [13].

Compared with these commercial solutions, the proposed system focuses on combining fill-level monitoring with lid-event tracking and rule-based detection of selected abnormal conditions, including suspected illegal dumping, sudden increases in fill level, and extended lid opening. The component also provides structured sensor readings and event records for use by other parts of the system responsible for historical analysis, prediction, and route optimisation. This component-based approach allows the monitoring function to contribute to a wider waste management workflow rather than operate only as an independent fill-level reporting system.

The main competitive consideration is therefore not fill-level monitoring alone, as this capability is already available in established commercial products. Instead, the proposed system explores how lid activity, interpretable anomaly detection rules, and structured data exchange can be combined within an integrated prototype. These features may provide useful information about bin activity in addition to the current fill level. However, the available product descriptions do not establish that the commercial solutions lack comparable capabilities. Further testing and evaluation would be required to determine the practical advantages, implementation requirements, and suitability of the proposed system for its intended users.

8.3 Business or sustainability model

The proposed model is a SaaS subscription combined with leasing of the sensor hardware, with tiered pricing by fleet size, one-time installation fees and optional premium support. The prototype cost in the proposal is shown in Table 8.1. It includes 480 hours of development effort costed at LKR 150 per hour, which is a budget-accountability figure and not a commercial price.

Table 8.1: Prototype budget from the project proposal (LKR)

Item	Category	Cost
ESP32 development boards (×2)	Hardware	3,000
HC-SR04 ultrasonic sensors (×4)	Hardware	1,600
Magnetic reed switches (×4)	Hardware	800
Resistors, wires, PCB	Hardware	2,000

Item	Category	Cost
Azure IoT Hub (not used)	Cloud	0
Firebase Realtime Database, Spark plan	Cloud	0
Development tools and analytics libraries	Software	0
Human effort (480 hours at LKR 150)	Development	72,000
Total		79,400

The proposal also gives a per-unit prototype hardware cost of roughly LKR 8,500 to 12,000. The itemised electronics in Table 8.1 are far lower, so the unit estimate presumably includes items such as an enclosure, power supply and installation. The proposal’s expected reduction of 20–40% in collection frequency is a projection that this work does not test, and it is not claimed as a result.

8.4 Scalability, intellectual property and compliance

Adding bins means adding device identifiers and records in Firebase, with no change to the cloud logic, and the design target is at least 500 bin nodes. Reaching that scale would require paid service tiers, since the free Firebase plan used for the prototype is limited (Section 4.4.5). On intellectual property, the proposal notes the possibility of a provisional patent on the behavioural anomaly detection approach, with open-source hardware components kept separate from proprietary software and schema contributions.

For compliance, the component collects no personal data, and ethical clearance is being sought (Section 3.6). A commercial deployment would also need to follow local regulations for radio equipment and data protection.

8.5 Contribution to the Sustainable Development Goals

The project aligns with SDG 11 (Sustainable Cities and Communities) and SDG 12 (Responsible Consumption and Production) of the 2030 Agenda [19]. The component contributes indirectly: by showing which bins need collection and why, it supports fewer unnecessary collection trips and fewer overflows, which relate to the management of urban waste. The size of any such effect has not been measured in this work and depends on the downstream routing component and on real deployments.

9 CONCLUSION AND FUTURE DIRECTIONS

9.1 Conclusion

This thesis set out to develop a bin-level monitoring component that explains bin behaviour as well as its current fill state. An ESP32-based system was designed and implemented to classify fill levels using ultrasonic sensor readings, display OK, WARNING, and CRITICAL statuses through green, yellow, and red LEDs, detect lid activity using a reed switch, store sensor readings in Firebase, and display lid status on a web page.

The proposed design also considers rule-based detection of abnormal bin activities, including suspected illegal dumping, sudden increases in fill level, extended lid opening, and potential overflow. However, these detection rules must be distinguished from the functions implemented and verified in the supplied firmware. The component was developed with reference to the gaps identified in the literature and evaluated through structured test scenarios based on the requirements defined in the research methodology.

The evaluation focused on fill-level classification accuracy, anomaly detection, alert response time, data transmission reliability, and integration with the other system components. The results provide a basis for assessing the component against its defined performance targets and identifying areas for improvement. In particular, the relationship between sensor stability, sampling frequency, detection rules, alert latency, and cloud data handling highlights important considerations when developing an IoT-based waste monitoring solution. The overall success of the component should be determined by comparing the measured results with the specified targets rather than assuming that all requirements have been achieved.

9.2 Achievement of objectives

Table 9.1 sets each objective against the evidence that will show whether it was met.

Table 9.1: Assessment of objectives

Objective	Evidence	Status
SO1 Multi-level fill sensing	Tables 6.2 and 6.3; tests T01–T07	Evaluated through sensor measurement and fill-level classification tests.
SO2 Lid-event tracking	Test T17; Table 6.4	Evaluated through lid-event recording and verification.
SO3 Anomaly detection engine	Tests T09–T16; Table 6.4	Requires verification of implementation and anomaly detection test results.
SO4 Structured cloud schema	Tests T18–T21; Table 6.6	Evaluated through cloud data storage, retrieval, and interface checks.
SO5 Validation against targets	Table 6.1	Depends on comparing measured results against the predefined performance targets.
SO6 Reliable downstream data	Tests T20, T21, T25; Table 6.6	Depends on verified downstream integration and data transmission results.

Each status should be completed as achieved, partially achieved or not achieved, with a one-line justification drawn from Chapter 6.

9.3 Individual research and product contributions

The researcher’s research contribution is the behaviour-aware monitoring approach, the analysis of the design trade-offs between sampling and latency, and the evaluation plan and results. The product contribution is the working component: firmware, LED status indication, Firebase data structure and lid-status web page, which together

provide the data source for Components 3 and 4. The work of the other three components is acknowledged but not claimed.

9.4 Future directions

Field deployment: test on real bins over weeks, so that thresholds can be tuned on observed behaviour and anomalies can be compared with real misuse.

Learned detection: compare the rule-based engine with a learned anomaly detector on the same data, building on the stored event logs.

Sensor fusion with Component 1: combine fill level with weight and environmental readings to separate light bulky waste from dense waste.

Adaptive thresholds: adjust the dumping, spike and lid-rate thresholds per bin from its own history.

Connectivity and power: evaluate low-power wide-area options and deep-sleep scheduling for bins without Wi-Fi coverage.

Scale and cost: move from free cloud tiers to a paid configuration and measure behaviour at tens or hundreds of nodes.

REFERENCES

- [1] S. Kaza, L. C. Yao, P. Bhada-Tata, and F. Van Woerden, *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. Washington, DC, USA: World Bank, 2018, doi: 10.1596/978-1-4648-1329-0.
- [2] A. Medvedev, P. Fedchenkov, A. Zaslavsky, T. Anagnostopoulos, and S. Khoruzhnikov, "Waste management as an IoT-enabled service in smart cities," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN/ruSMART 2015)*, Lecture Notes in Computer Science, vol. 9247. Cham, Switzerland: Springer, 2015, pp. 104–115.
- [3] A. Addas, M. N. Khan, and F. Naseer, "Waste management 2.0 leveraging Internet of Things for an efficient and eco-friendly smart city solution," *PLOS ONE*, vol. 19, no. 7, p. e0307608, Jul. 2024, doi: 10.1371/journal.pone.0307608.
- [4] M. A. Hannan, M. Arebey, H. Basri, and R. A. Begum, "Intelligent solid waste bin monitoring and management system," *Aust. J. Basic Appl. Sci.*, vol. 4, no. 10, pp. 5314–5319, 2010.
- [5] M. A. Al Mamun, M. A. Hannan, A. Hussain, and H. Basri, "Integrated sensing systems and algorithms for solid waste bin state management automation," *IEEE Sensors J.*, vol. 15, no. 1, pp. 561–567, Jan. 2015, doi: 10.1109/JSEN.2014.2345477.
- [6] M. A. Al Mamun, M. A. Hannan, A. Hussain, and H. Basri, "Theoretical model and implementation of a real time intelligent bin status monitoring system using rule based decision algorithms," *Expert Syst. Appl.*, vol. 48, pp. 76–88, Apr. 2016, doi: 10.1016/j.eswa.2015.11.025.
- [7] V. Perumal, S. V. Divya, and N. Vishal, "Smart dustbin using ESP32 for waste management," *IRO J. Sustain. Wireless Syst.*, vol. 6, no. 4, pp. 309–317, 2024, doi: 10.36548/jsws.2024.4.002.
- [8] T. Anagnostopoulos *et al.*, "Challenges and opportunities of waste management in IoT-enabled smart cities: A survey," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 3, pp. 275–289, Jul.–Sep. 2017, doi: 10.1109/TSUSC.2017.2691049.
- [9] S. Mishra, L. Jena, H. K. Tripathy, and T. Gaber, "Prioritized and predictive intelligence of things enabled waste management model in smart and sustainable environment," *PLOS ONE*, vol. 17, no. 8, p. e0272383, Aug. 2022, doi: 10.1371/journal.pone.0272383.
- [10] M. S. Islam, M. A. Hannan, H. Basri, A. Hussain, and M. Arebey, "Solid waste bin detection and classification using dynamic time warping and MLP classifier," *Waste Manag.*, vol. 34, no. 2, pp. 281–290, Feb. 2014, doi: 10.1016/j.wasman.2013.10.030.
- [11] Espressif Systems, "ESP32 series datasheet," Shanghai, China. [Online]. Available: <https://www.espressif.com>. Accessed:

- [12] A. Banks and R. Gupta, Eds., “MQTT version 3.1.1,” OASIS Standard, Oct. 2014. [Online]. Available: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
- [13] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quart.*, vol. 28, no. 1, pp. 75–105, Mar. 2004, doi: 10.2307/25148625.
- [14] I. Hong, S. Park, B. Lee, J. Lee, D. Jeong, and S. Park, “IoT-based smart garbage system for efficient food waste management,” *Sci. World J.*, vol. 2014, Art. no. 646953, 2014, doi: 10.1155/2014/646953.
- [15] Microsoft, “Azure IoT Hub documentation.” [Online]. Available: <https://learn.microsoft.com/azure/iot-hub/>. Accessed:
- [16] Google, “Firebase Realtime Database documentation.” [Online]. Available: <https://firebase.google.com/docs/database>. Accessed:
- [17] *Date and Time — Representations for Information Interchange — Part 1: Basic Rules*, ISO Standard 8601-1:2019, 2019.
- [18] *Software and Systems Engineering — Software Testing — Part 1: General Concepts*, ISO/IEC/IEEE Standard 29119-1:2022, 2022.
- [19] United Nations, “Transforming our world: The 2030 Agenda for Sustainable Development,” General Assembly Resolution A/RES/70/1, Sep. 2015.

GLOSSARY

Anomaly: A reading or event that departs from normal bin usage, such as a very large fill increase in a single lid opening.

Debounce: A short delay or filter that prevents a switch's brief contact chatter from being counted as several events.

Edge processing: Computation carried out on the device at the bin before data is sent to the cloud.

Fill level: The share of a bin's internal volume occupied by waste, expressed as a percentage.

GPIO interrupt: A signal on a microcontroller pin that makes the processor respond immediately to a change, without polling.

Overflow: The state in which a bin is full and further waste cannot be contained.

Publish-subscribe: A messaging pattern in which senders publish to named topics and receivers subscribe to those topics, as in MQTT.

Reed switch: A switch with two contacts that change state in the presence of a magnetic field.

Sliding-window average: The mean of the most recent N readings, recomputed as each new reading arrives.

Time of flight: The principle of calculating distance from the time a pulse takes to reach a surface and return.

APPENDICES

Appendix A Hardware component specifications

Table A.1 lists the specifications of the three main components, taken from the project proposal, and Table A.2 gives the pin assignment from the prototype firmware. Figure A.1 shows the wiring, drawn from the pin assignments in the firmware.

Table A.1: Hardware component specifications

Component	Key specifications	Role in the component
ESP32 WROOM-32	Dual-core 240 MHz, Wi-Fi 802.11 b/g/n, 34 GPIO pins, 4 MB flash	Central microcontroller: sensor I/O, MQTT transmission, edge anomaly rules
HC-SR04 ultrasonic sensor	Range 2–400 cm; accuracy ± 3 mm; 5 V; 40 kHz	Fill-level distance measurement by time of flight
Magnetic reed switch (normally open)	3.3–5 V; response time under 1 ms; contact rating 0.5 A at 100 V DC	Lid-state detection by magnetic field interruption

Table A.2: Pin assignment in the prototype firmware

Signal	ESP32 pin	Note
HC-SR04 TRIG	GPIO 5	Output
HC-SR04 ECHO	GPIO 18	Input; 5v
Reed switch	GPIO 27	Input with pull-up; LOW means closed
Green, yellow, red LEDs	GPIO 21, 22, 23	Status indicators for OK, WARNING, CRITICAL
DHT22, MQ135, MQ4	GPIO 4, 34, 35	Component 1 sensors in the combined firmware, not part of this component

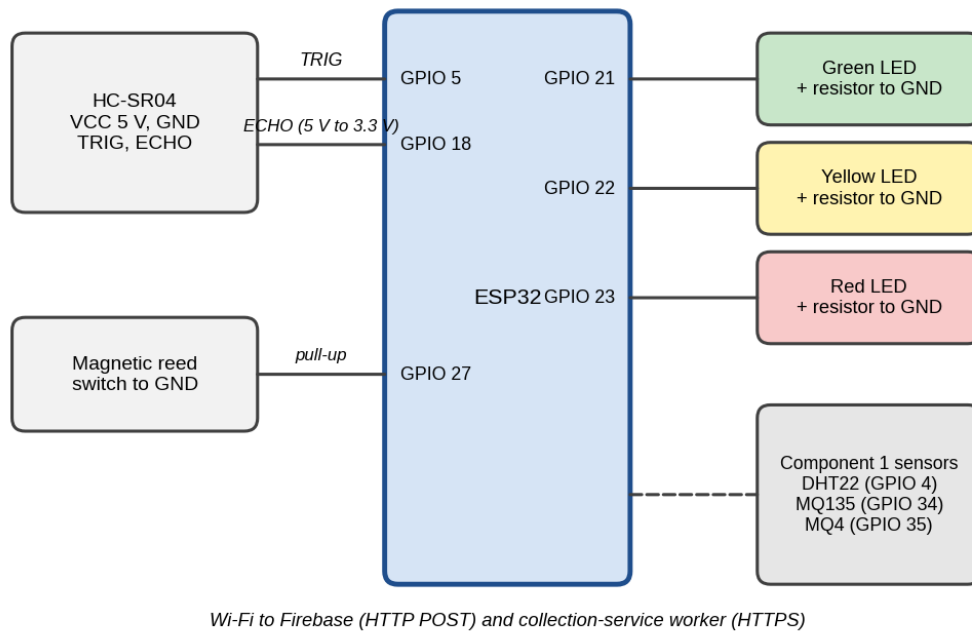


Figure A.1: Wiring of the prototype, drawn from the firmware pin assignments

Appendix B Academic integrity and AI use disclosure

Generative AI was used as a supporting tool during the preparation of this thesis. Claude (Anthropic) was used occasionally to help organise some sections, improve the clarity of written content, and suggest ways to present the evaluation methods. The thesis is based on my own project work, technical decisions, implementation, and understanding of the Smart Fill-Level and Occupancy Monitoring component.

All content was reviewed and edited to ensure that it reflected my project and work. I remain responsible for the accuracy of the technical information, references, diagrams, and evaluation results presented in this thesis. Any reported findings should be based on the actual testing and observations carried out during the project..

Appendix C Firebase Realtime Database schema

Table C.1 describes the event log structure used in Firebase Realtime Database.

Table C.1: Event log schema

Path	Field	Description
smart_bins/{bin_id}/fill_readings/{timestamp}	fill_pct, level_category, anomaly_flag	Filtered fill percentage, its class and any anomaly flag
smart_bins/{bin_id}/lid_events/{event_id}	event_type, timestamp, fill_at_event, lid_duration_ms	LID_OPEN or LID_CLOSE, UTC time, fill at the event, open duration
smart_bins/{bin_id}/alerts/{alert_id}	alert_type, severity, timestamp, fill_pct, description	Anomaly type, severity, time and context

Appendix D Risk analysis matrix

Table D.1 lists the risks identified in the project proposal and their planned mitigations.

Table D.1: Risks and mitigations from the project proposal

Risk	Likelihood	Impact	Mitigation
Sensor noise causing wrong fill classification	Medium	Medium	Sliding-window average over 5 readings; calibrate against known distances; recalibrate after bin changes
Wi-Fi drops cutting off transmission	Medium	High	Buffer readings locally and retry; use queued MQTT publishing
Too many false-positive alerts	Medium	High	Require two consecutive triggers before CRITICAL; tune thresholds on validation data; log false positives
Not enough real data to validate	Low	Medium	Synthetic datasets from published statistics, clearly labelled; collect authentic data in the campus pilot
Cloud costs exceed budget at scale	Low	Low	Stay on free tiers for the prototype; monitor usage and upgrade only if needed