

Smart Fill-Level & Occupancy Monitoring

R26-IT-126

Project Proposal Report

Gamaariya C C – IT22202772

B.Sc. (Hons) Degree in Information Technology Specialized in
Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

March 2026

DECLARATION

I declare that this is my own work, and this proposal does not knowingly incorporate any material previously submitted for a degree or diploma at any other university or higher education institution, nor does it contain any material that has been previously published or written by another person apart from where proper acknowledgment is given in the text.

Name	Student ID	Signature
Gamaariya C. C.	IT22202772	

The above candidate is conducting research for their undergraduate Dissertation under my supervision.

Name	Date	Signature
Lididha D	20/03/20	

ABSTRACT

The conventional waste management systems in urban settings are based on the fixed route schedules that do not indicate the current state of waste bins, which leads to overflow, useless collection journeys, high operation expenses, and environmental risks. A novel gap present in practice is the lack of an integrated framework of data-driven methods that entail the integration of Internet of Things (IoT) sensing, the analytics of behaviors, and predictive intelligence.

This study presents a Smart Waste Management System that was designed as a multi-component system, which was created through collaboration. Personal contribution of this study is on Smart Fill-Level and Occupancy Monitoring module, which offers multi-level fill sensing (Low, Medium, High, and Critical) and lid-opening occurrence monitoring, and abnormal usage pattern, including illegal dumping, sudden fill-level peaks, and unreliable lid behavioral pattern. This module uses a distance sensor based on the time-of-flight measurement principles through the use of ultrasonic distance sensing and an ESP32 microcontroller that is used to measure volumetric fill percentage. An anomaly detection engine that uses rules on real-time event streams is used to issue context-based alerts in case overflow risks or behavioral anomalies are detected. The data is streamed to Microsoft Azure and Firebase Cloud as event logs and fill-level time series, which helps to build the data foundation of the application in the form of downstream machine learning and dynamic routing modules written by team members.

The proposed system will bridge the identified research gap in how it offers not only reactive fill detection but a behavior-sensitive monitoring paradigm that can explain the reasons behind overflow conditions and provide accountability and compliance evidence which may be especially relevant to a managed setting, including campus and office complexes. The technologies utilized are ESP32 microcontroller, HC-SR04 ultrasonic sensor, magnetic reed switches in tracking the lid, MQTT/HTTP protocol in transmitting to the cloud, and Firebase Realtime Database in event storage. The validation will go by simulation under the normal and abnormal use conditions controlled and the performance measured against the ability to detect the anomalies, precision of alert and the false-positives.

The anticipated results are the decrease in the number of cases of overflows, enhanced operational decision-making, and indicative improvement of the SDG 11 (Sustainable Cities and Communities) and SDG 12 (Responsible Consumption and Production).

Keywords: Smart Waste Management, IoT Fill-Level Monitoring, Anomaly Detection, Lid-Event Tracking, Overflow Prediction, ESP32, Cloud Analytic

ACKNOWLEDGEMENT

Sincere thanks to Mr. Uditha Dharmakeerthi and Mr. Kanishka Yapa, Supervisor and Co-Supervisor respectively, who guided, mentored and critically reviewed me during the course of the research proposal development. They have been skilled in the area of IoT systems and data analytics, hence their skills have been instrumental in the direction and quality of this work. The gratitude is also given to the Department of Information Technology in Sri Lanka Institute of Information Technology (SLIIT) because of the facilities in form of academic resources, laboratories and institutional support required to carry out this research. The researcher appreciates the spirit of teamwork that the project team members possess and whose complementary modules add to the generally architecture system. Lastly, the researcher owes his or her gratitude to his family and colleagues who remained very supportive in the process of preparing this proposal

TABLE OF CONTENT

Contents

DECLARATION	2
ABSTRACT (Maximum 300 words)	3
ACKNOWLEDGEMENT	4
TABLE OF CONTENT (Auto Generated)	5
LIST OF TABLES	7
LIST OF FIGURES	8
LIST OF ABBREVIATIONS	9
LIST OF APPENDICES	10
1.INTRODUCTION	11
1.1 Background and Context.....	11
1.2 Literature Review	12
1.3 Research Gap	13
2.OBJECTIVE	14
2.1 Main Objective.....	14
2.2 Specific Objectives	14
3.METHODOLOGY	15
3.1 Research Approach	15
3.2 Multi-Level Fill Sensing Design	15
3.3 Lid-Opening Event Tracking.....	16
3.4 Anomaly Detection Engine	16
3.5 Data Transmission and Cloud Integration	17
3.6 Data Collection and Ethical Considerations.....	17
3.7 Validation Strategy and Performance Metrics.....	17
4.HIGH-LEVEL SYSTEM ARCHITECTURE	18
4.1 System Overview	18
4.2 Smart Fill-Level & Occupancy Monitoring Module - Detailed Architecture.....	18
4.3 Data Flow: Edge Sensors to Analytics Modules.....	19
4.4 Scalability and Security Considerations	19
5.USER REQUIREMENTS	19
5.1 Stakeholder Identification	19
5.2 Functional Requirements	20
5.3 Non-Functional Requirements	21

6.COMMERCIALIZATION PLAN.....	22
6.1 Target Market and Customer Persona.....	22
6.2 Value Proposition	22
6.3 Revenue Model and Pricing Strategy.....	22
6.4 Competitive Advantage and Intellectual Property	22
7.BUDGET AND JUSTIFICATION	23
8.WORK BREAKDOWN STRUCTURE (WBS)	24
9.GANTT CHART	26
10.REFERENCES LIST.....	27
11.APPENDICES	28
Appendix A: Hardware Component Specifications	28
Appendix B: Firebase Realtime Database Schema	28
Appendix C: Risk Analysis Matrix.....	29

LIST OF TABLES

Table 1: Fill-Level Classification Thresholds and Corresponding Alert Actions

Table 2: Anomaly Detection Rule Set for Smart Fill-Level Module

Table 3: Functional Requirements of the Smart Fill-Level & Occupancy Monitoring Module

Table 4: Non-Functional Requirements

Table 5: Budget and Justification for Smart Fill-Level & Occupancy Monitoring Module

Table 6: Work Breakdown Structure – Gamaariya C C

LIST OF FIGURES

Figure 1: High-Level System Architecture of the Smart Waste Management System

Figure 2: Smart Fill-Level & Occupancy Monitoring Module – Detailed Component Diagram

Figure 3: Data Flow Diagram: Edge Sensor Layer to Cloud Analytics

Figure 4: Gantt chart – Project Timeline (Semester 1 & Semester 2)

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
ESP32	Espressif Systems 32-bit Microcontroller
HC-SR04	Ultrasonic Distance Sensor Module
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
MAE	Mean Absolute Error
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
RMSE	Root Mean Square Error
SDG	Sustainable Development Goal
TIM	Technology Integration and Management
WBS	Work Breakdown Structure

LIST OF APPENDICES

Appendix A: Hardware Component Specifications – ESP32 and HC-SR04

Appendix B: Anomaly Detection Rule Table – Detailed Logic

Appendix C: Firebase Realtime Database Schema – Event Log Structure

1. INTRODUCTION

1.1 Background and Context

Solid waste management in the fast becoming urbanized settings has come out to be one of the most urgent forms of operation as well as environmental concern in the twenty first century. Waste management services on municipal and institutional levels across the world are still based on fixed-time collection systems, which are route-based and reactive in nature. In this type of paradigms, the collection crews will travel along predefined paths without considering the actual bin occupancy, and these are accompanied by systemic inefficiencies, such as unnecessary vehicle movements, high fuel usage, high labor expenses, and frequent overflow problems, which create real hazards to the general health [1]. According to the estimates made by the World Bank, the municipal solid waste across the globe is expected to grow between 2.01 billion tons in 2016 and 3.40 billion tons in 2050, which puts a massive strain on the collection systems and the systems of urban governance [1].

In Sri Lanka, local authorities in the urban areas experience similar problems. Ineffective and rugged collection of waste is still a major contributor to environmental degradation, littering, and dissatisfaction among the community in cities. This is because the collection supervisors do not have actionable visibility of the bin status as real-time monitoring capabilities are not available, so they will not be able to do timely interventions when the bins are nearly full. This reactive stance, in addition to raising the cost of operation, compromises national hygiene and recycling policies.

To address these issues, the scientific community has become more interested in researching how the Internet of Things (IoT) technologies and artificial intelligence (AI) can be used to make waste management a more proactive and data-driven science. Smart bins can be IoT-powered with multi-parameter sensors to measure the fill levels, weight, temperature, humidity, and gas levels and send structured data streams to the cloud platforms to be analyzed in real-time and used to support decisions [2]. Such systems in combination with machine learning models and dynamic routing algorithms have shown the potential of making major improvements in their operations in the form of reduced frequency of collection, decrease in emissions and higher rates of recycling [8].

The current research project, which is called Smart Waste Management System is a group work by a team of four undergraduate students in the Technology Integration and Management (TIM) research cluster at the Sri Lanka Institute of Information Technology (SLIIT). The system combines four complementary modules, namely: (1) Environmental Sensing Smart Bin, (2) Smart Fill-Level and Occupancy Monitoring, (3) Waste Composition Prediction using Machine Learning and (4) Predictive Analytics and Dynamic Routing. Collectively, these elements will form a cloud-based and end-to-end smart system that is meant to transform the waste collection processes to a predictive paradigm rather than a reactive one.

The offered proposal will specifically deal with the contribution at the individual research level of Gamaariya C C : the Smart Fill-Level and Occupancy Monitoring module. It has multi-level fill detection, lid-opening detection, and behavior aware anomaly detection that make the basis of the data layer that subsequent analytics and routing decisions rely on. The innovative aspect of the given component is its ability to tell the behavioral settings of how and why that particular fill state was acquired in the first place and actively identify the presence of misuse and illegal dumping, as well as the exceptional usage patterns that could not be identified through simple threshold-based systems.

The study is in line with the United Nations sustainable development goals, i.e. SDG 11 (Sustainable Cities and Communities) and SDG 12 (Responsible Consumption and Production) as it encourages smart and efficient city garbage infrastructures. The study group congruency of the TIM specialization indicates the adoption of the emerging technologies - embedded systems, cloud computing, and data analytics - into an effective, scalable management solution.

1.2 Literature Review

There is a considerable amount of academic literature addressing the use of IoT and sensor-based technologies to smart waste management that has offered theoretical and empirical content of the current research. In this section, several important contributions to the areas of fill-level monitoring, lid-event sensing, anomaly detection, embedded systems integration, and intelligent bin management are reviewed.

According to World Bank estimates [1], the volume of municipal solid waste generated is expected to grow from 2.01 billion tons per year currently to 3.40 billion tons by 2050. This establishes the case for moving to data-driven waste collection as opposed to fixed-route waste collection. Addas et al. [2] corroborated this need in a case study on the deployment of IoT-enabled waste management, which shows that smart city systems leveraging connected bin sensors generate measurable gains in collection efficiency and less unnecessary vehicle trips than schedule-based models.

Some of the earliest serious engineering work on automated bin monitoring came from Hannan et al. [9], who proposed an intelligent bin monitoring and management system that used sensor arrays to track bin occupancy automatically rather than relying on manual inspection. The key insight from that work was simple but important: continuous automated sensing outperforms periodic manual checks for triggering timely collections. Al Mamun et al. [4] extended this further, combining ultrasonic, weight, and status sensors into an integrated sensing platform. Their multi-sensor approach was more reliable than any single sensor alone a finding directly relevant to the way I have designed my module to treat fill level and lid-event data as complementary rather than independent signals. The same team later published a rule-based decision algorithm framework [5] that showed threshold-based event logic can distinguish normal filling behavior from anomalous states with high accuracy. That paper is essentially the theoretical ancestor of the anomaly detection engine I am building.

On the hardware side, Perumal et al. [7] specifically validated the ESP32 as a platform for smart waste bin applications. Their system combined ultrasonic fill sensing with real-time Wi-Fi cloud transmission and found the ESP32's integrated connectivity and processing power well-suited to the task which is exactly the platform this module uses. Islam et al. [10] approached the classification problem from the ML side, showing that machine learning models trained on sensor-derived bin state data can reliably classify bin conditions. That classification output is important for my module because the event logs I generate will be one of the primary feature inputs for the ML composition prediction module my teammate is building.

The broadest survey of where this whole field stands comes from Anagnostopoulos et al. [3], whose comprehensive review of IoT-enabled smart city waste management systems identified a consistent gap: almost no existing system does behavioral anomaly detection. Systems detect overflow after it happens. They do not try to identify the patterns an unusually fast fill rate, an unusual cluster of lid events at 2am that signal something is wrong before overflow occurs, or that suggest the fill pattern is not normal usage at all. The authors explicitly called for rule-based event processing engines as a next

step in the field. That call is what this module directly responds to. Mishra et al. [8] reinforced this from a different angle, demonstrating that predictive and prioritized IoT waste management models consistently outperform purely reactive systems in both efficiency and environmental impact terms.

Finally, the research methodology adopted here is grounded in the design science framework of Hevner et al. [6], which sets out clear criteria for what counts as rigorous IT artefact research relevance, evaluation, contribution, and rigor. Those criteria shape how this proposal is structured and how validation will be designed.

Taken together, the literature makes a clear case: the technical building blocks for smart bin monitoring exist and work, but nobody has yet put together a module that not only detects fill level in real time but also interprets the behavioral patterns behind it. That gap is exactly what this research addresses.

1.3 Research Gap

The literature makes a strong case that IoT-based bin monitoring works. What it has not delivered yet is a system that goes beyond detecting fill level to explaining the behavior behind it. That is the gap this research targets.

The most obvious limitation of existing systems is that they are reactive by design. A bin reaches 90% full, an alert fires, a truck is dispatched. That is useful, but it tells you nothing about why the bin reached 90% in the first place. Was it gradual accumulation over a normal workday? A single illegal dumping event? A pattern of high-frequency short-duration lid openings suggesting misuse? Current systems treat all of these identically. They see the outcome but not the cause, which means the response is always the same regardless of whether the situation calls for a collection truck, a maintenance check, or a compliance report.

Second, while Al Mamun et al. [5] showed that rule-based algorithms can classify bin states reliably, no published system has gone the next step of implementing a behavior-aware anomaly detection engine that specifically identifies distinct event types illegal dumping, sudden fill spikes, and abnormal lid patterns as separately classified and separately reportable signals. The engineering to do this at the embedded level, with edge processing on an ESP32 before data even reaches the cloud, has not been demonstrated in the existing literature.

Third, the event data that most systems do collect tends to be poorly structured for downstream use. Timestamps are inconsistent, event types are not categorized, and there is no standard schema that a machine learning model or a routing algorithm can reliably consume. This makes it hard to build a pipeline where the output of one module becomes a clean input to the next which is exactly what a multi-module system like this one requires.

Together, these three gaps point to a clear and addressable problem. This module tackles all three: behavior-aware anomaly detection that classifies not just fill states but the events behind them, edge-level processing for low-latency alerts, and a structured event log schema designed from the ground up to feed the ML and routing modules built by the rest of the team

2. OBJECTIVE

2.1 Main Objective

The overall goal of the group project is to build and test an intelligent waste management system that replaces fixed-schedule collection with demand-driven, data-informed decisions - combining IoT bin sensing, machine learning composition prediction, and dynamic route optimization into one working framework.

2.2 Specific Objectives

My specific objectives for the Smart Fill-Level and Occupancy Monitoring module are:

1. Build a multi-level fill sensing system using an HC-SR04 ultrasonic sensor and ESP32 microcontroller that classifies bin fill into Low (0–25%), Medium (26–50%), High (51–80%), and Critical (81–100%) stages and sends timestamped readings to the cloud continuously.
2. 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 — giving the system the behavioral context it needs to make sense of fill-level changes.
3. Design and implement a rule-based anomaly detection engine, running on both the ESP32 and Firebase Cloud Functions that identifies illegal dumping, sudden fill spikes, and abnormal lid-activity rates as distinct event types and triggers appropriate real-time alerts.
4. Store all fill-level readings and event logs in a structured Firebase and Azure cloud schema that supports daily and weekly usage analytics and serves as a clean, queryable feature dataset for the ML Composition Prediction and Predictive Routing modules.
5. Validate the module through controlled simulation of normal and anomalous scenarios, measuring fill classification accuracy (target >95%), anomaly detection accuracy (target >90%), false-positive rate (target <5%), and alert latency (target <3 seconds).
6. Contribute to reducing overflow incidents and unnecessary collection trips by ensuring the data this module produces is reliable and well-structured enough to be a genuine asset to the team's downstream prediction and routing systems.

3. METHODOLOGY

3.1 Research Approach

This research follows a Design Science Research (DSR) approach [6], which is the right fit here because the goal is not just to study the problem but to build something that solves it. DSR is well established in IT research precisely because it treats artefact development — the act of designing and building a system as a legitimate and rigorous form of scholarly contribution, as long as the design is evaluated against real criteria and the findings are generalizable beyond the specific instance.

In practice, this means the research moves through iterative cycles: design the module, implement a prototype, test it against defined scenarios, measure what works and what does not, and refine. The process is grounded in actual sensor data collected from prototype bins, supplemented where necessary by synthetic datasets built from published waste accumulation statistics. Any synthetic data will be clearly labeled as such throughout the research outputs.

3.2 Multi-Level Fill Sensing Design

The fill sensing setup is straightforward in concept: an HC-SR04 ultrasonic sensor sits at the top of the bin interior, pointing downward at the waste surface. It works on the time of-flight principle - it fires a 40 kHz sound pulse, waits for the echo, and calculates distance from how long that takes. The formula is:

$$\text{Distance (cm)} = (\text{Time } (\mu\text{s}) \times 0.0343) / 2$$

Where 0.0343 cm/ μ s is the speed of sound at standard conditions. Converting to fill percentage is then:

$$\text{Fill \%} = ((\text{Bin Height} - \text{Measured Distance}) / \text{Bin Height}) \times 100$$

Waste surfaces are uneven and reflective in unpredictable ways, so raw ultrasonic readings can be noisy. To deal with this, the firmware applies a sliding window average over the last five readings before classifying. This cuts transient noise by up to 70% and adds only about 2.5 seconds of latency at a 2 Hz sampling rate - completely acceptable for a waste bin that fills over hours, not seconds.

Once filtered, the reading maps to one of four levels:

Level	Fill Range	Alert Status	Action Triggered
Low	0 – 25%	None	Log reading to cloud
Medium	26 – 50%	Informational	Log + update dashboard
High	51 – 80%	Warning	Log + push advisory notification
Critical	81 – 100%	Critical Alert	Log + trigger overflow alert + notify routing module

Table 1: Fill-Level Classification Thresholds and Corresponding Alert Actions

3.3 Lid-Opening Event Tracking

Tracking lid events is simpler than fill sensing but arguably more interesting from a behavioral perspective. A magnetic reed switch on the lid hinge breaks its circuit when the lid opens and restores it when the lid closes. The ESP32 listens for these transitions via a GPIO interrupt, meaning it reacts instantly rather than polling — no lid event gets missed.

Every time a lid opens or closes, the firmware creates a structured record and sends it to Firebase:

- `bin_id` - which bin this event came from
- `event_type` - either `LID_OPEN` or `LID_CLOSE`
- `timestamp` - UTC time in ISO 8601 format, recorded to the millisecond
- `fill_level_at_event` - the fill percentage captured right before the lid moved
- `lid_duration_ms` - how long the lid stayed open, calculated when the `LID_CLOSE` event fires

These records are what make the behavioral analysis possible. Lid frequency, access duration, and timing patterns can all be derived from this log - and those patterns are what the anomaly detection engine looks for.

3.4 Anomaly Detection Engine

The anomaly detection engine is where this module becomes genuinely different from a standard fill-level monitor. It runs as a rule-based event processor across two layers: primary rules run on the ESP32 firmware itself for the fastest possible response, while more complex pattern analysis runs in Firebase Cloud Functions. Keeping the critical alerts at the edge means a suspicious event triggers a response in under a second, without waiting on cloud round-trips.

The five detection rules currently implemented are:

Anomaly Type	Detection Rule	Severity	Alert Action
Illegal Dumping	$\Delta\text{Fill} > 25\%$ per single <code>LID_OPEN</code> event	HIGH	Flag event + notify supervisor
Sudden Fill Spike	$\Delta\text{Fill} > 30\%$ in < 60 seconds (no lid event)	HIGH	Alert + flag for manual review
Abnormal Lid Rate	> 15 lid events in 60 min window	MEDIUM	Advisory alert generated
Extended Lid Open	Lid open > 120 seconds continuously	LOW	Log event + dashboard flag

Overflow Imminent	Fill $\geq$ 81% AND fill rate > 5%/min	CRITICAL	Immediate alert + trigger routing module notification
--------------------------	--	-----------------	---

Table 2: Anomaly Detection Rule Set for Smart Fill-Level Module

3.5 Data Transmission and Cloud Integration

Data leaves the ESP32 over Wi-Fi using MQTT, a lightweight publish-subscribe protocol that is well suited to event-driven IoT applications. Fill-level readings go out every 30 seconds by default, while lid events and anomaly alerts are published the moment they are detected - no polling delay, no batching.

On the cloud side, Azure IoT Hub receives the incoming data and routes it two ways: to Azure Blob Storage for long-term archiving and ML feature extraction, and to Firebase Realtime Database for low-latency real-time access. Firebase Cloud Functions handle the cloud-side anomaly rules and dispatch push notifications to operator dashboards and mobile endpoints. The two platforms complement each other well - Firebase is fast and reactive, Azure is scalable and analytically powerful.

3.6 Data Collection and Ethical Considerations

Primary data will come from prototype bins deployed in controlled campus settings. Where real historical data is not available, synthetic datasets will fill the gap - generated from published waste statistics and clearly labeled as simulated throughout all research outputs. The project does not collect any personal data. Everything relates to bin conditions, usage frequencies, and environmental readings. An expedited ethical clearance will still be sought from SLIIT's review board to formally document data handling practices.

3.7 Validation Strategy and Performance Metrics

Validation will be done through structured controlled experiments. I will run the prototype through a set of scripted scenarios - normal usage patterns on one hand, and deliberately simulated anomalies (illegal dumping, sudden spikes, abnormal lid bursts) on the other. Because the ground truth is known in advance, it is straightforward to measure exactly how often the system catches what it should and how often it fires when it should not.

The metrics I will measure are:

- **Fill Classification Accuracy** - How often does the system put a reading in the right bucket (Low / Medium / High / Critical)? Target is above 95%. Anything below that and operators can't trust what they're seeing on the dashboard.
- **Anomaly Detection Accuracy** - Out of all the anomalous events I deliberately simulate, what percentage does the engine catch? Target is above 90%. Miss too many and the whole behavioral monitoring value proposition falls apart.
- **False-Positive Rate** - How often does the system cry wolf on a normal event? Target is below 5%. Too many false alarms and operators stop trusting the alerts entirely.

- Alert Latency - How long from the moment an anomaly occurs to the moment a CRITICAL alert lands in the cloud? Target is under 3 seconds. This is where the edge-processing design earns its keep.
- Data Transmission Reliability - What percentage of readings make it to the cloud without loss? Target is above 98%. Below that, fill-level trends start developing gaps that distort the ML module's inputs.

4. HIGH-LEVEL SYSTEM ARCHITECTURE

4.1 System Overview

The overall system sits across four layers, each with a distinct job. At the bottom, smart bins with embedded sensors generate a continuous stream of raw data. Above that, cloud platforms store and organize that data alongside external context like weather and calendar information. Next, machine learning models and analytics engines process everything into forecasts and status predictions. At the top, those outputs drive real operational decisions - which bins to collect, in what order, and where recycling resources should be pre-positioned.

It is worth being clear about how the four modules divide that work. My teammate Hemsara K.D.S handles the Environmental Sensing Smart Bin - temperature, humidity, gas, and weight readings from each bin. I handle the Smart Fill-Level and Occupancy Monitoring module, which is the volumetric and behavioral layer: how full is the bin, when was the lid opened, and does the usage pattern look normal. Both of our modules sit at the edge layer and feed data into the cloud. From there, Jayawardhane K.G.M.S uses that data to predict next-day waste composition, and Wijesiri A.M takes the fill and prediction data to optimize collection routes. The pipeline only works because each module produces structured, reliable data that the next one can actually use.

4.2 Smart Fill-Level & Occupancy Monitoring Module - Detailed Architecture

My module has three internal layers that work in sequence:

The first is the hardware layer. An HC-SR04 sensor at the bin ceiling handles fill measurement while a magnetic reed switch at the lid hinge handles access events. Both connect to an ESP32 via GPIO pins. The ESP32 runs the sensing loop - polling the ultrasonic sensor every 5 seconds and responding instantly via interrupt whenever the lid state changes.

The second is the edge processing layer. This is where the firmware earns its keep. Rather than blindly streaming every reading to the cloud, the ESP32 applies the sliding window filter, classifies the fill level, evaluates the anomaly rules, and only sends an immediate alert payload when something noteworthy happens. Routine readings go on their normal 30-second schedule. This keeps cloud traffic manageable and means critical alerts are not delayed by competing network activity.

The third is the cloud storage and processing layer. Firebase Realtime Database holds the live event stream - fill readings, lid events, and alert flags - in a JSON structure indexed by bin ID and timestamp. Azure Blob Storage keeps the full time-series archive for ML model training. Firebase

Cloud Functions sit on top of the incoming stream and run the more complex temporal pattern checks that are too heavy for the ESP32, then push notifications out to operator dashboards when needed.

4.3 Data Flow: Edge Sensors to Analytics Modules

The data flow moves in one direction through a clean pipeline. Sensors generate readings. The ESP32 filters, classifies, and evaluates them. Structured JSON payloads go to Azure IoT Hub via MQTT. IoT Hub splits the stream - Firebase gets the real-time copy, Blob Storage gets the archive. Firebase Cloud Functions watch the live stream and fire alerts when rules are triggered. The ML module (Component 3) pulls historical fill-level time series from Azure as training features. The routing module (Component 4) subscribes to live bin status from Firebase to decide which bins need collection today and in what order.

The key point is that this module is not just a sensor unit - it is a data producer for the whole system. The quality and structure of what it generates directly affects what every downstream module can do with it.

4.4 Scalability and Security Considerations

The architecture is built to grow. Adding more bins means registering more devices with Azure IoT Hub - no changes to the cloud processing logic needed. Azure supports millions of concurrent device connections, so scaling from a campus pilot to a city-wide deployment is an infrastructure question, not a redesign question. Security is handled at every layer: TLS encryption on all MQTT connections, X.509 certificate-based device authentication via Azure Device Provisioning Service, role-based access control on Firebase, and AES-256 encryption for data at rest. Nothing sensitive leaves the device or the cloud unprotected.

5. USER REQUIREMENTS

5.1 Stakeholder Identification

Five main groups have a stake in what this module produces:

- Waste Collection Operators - They need to know which bins are actually ready for collection, not just which ones are on today's schedule. Real-time fill status and overflow alerts let them make that call without guesswork.
- Facility Managers (Campus/Office) - They care about compliance and accountability. When a bin fills up suspiciously fast or a lid event pattern looks wrong, they need a log they can point to, not just a vague alert.
- Municipal and Institutional Supervisors - They need weekly and daily usage summaries to plan staffing, justify budgets, and report to regulators. The event history this module stores feeds directly into that.
- Recycling Facility Operators - They work downstream of the collection process. Better fill-trend data helps them anticipate incoming volumes and get sorting resources ready in advance, which is something the ML module (Component 3) translates into direct composition forecasts.

- System Administrators - They need a reliable, secure platform they can actually manage: add devices, adjust thresholds, and push firmware updates. The architecture supports all of this without requiring a system redesign.

5.2 Functional Requirements

The nine functional requirements below define exactly what the module must be able to do. Each one maps directly to either the fill sensing, lid tracking, anomaly detection, or cloud integration sub-systems described in the methodology.

ID	Requirement	Description
FR-01	Multi-Level Fill Sensing	The bin fill level must be measured continuously and sorted into four states: Low (0 - 25%), Medium (26-50%), High (51 - 80%), and Critical (81-100%). Each reading gets a label, not just a number.
FR-02	Real-Time Cloud Transmission	Fill readings must go to Azure IoT Hub and Firebase with timestamps at least every 30 seconds. If a reading is held back longer than that, the system is effectively blind.
FR-03	Lid-Event Tracking	Every lid open and close must be caught, timestamped, and logged alongside the fill level at that moment and how long the lid stayed open. No events should be missed.
FR-04	Illegal Dumping Detection	If the fill level jumps by more than 25% within a single lid-open event, the system must flag it. That kind of spike in one access is not normal usage.
FR-05	Sudden Fill Spike Detection	If fill level rises by more than 30% in under 60 seconds with no lid event to explain it - sensor interference, bin damage, or something else unusual - the system must catch and flag it.
FR-06	Abnormal Lid Activity Detection	More than 15 lid-open events in a 60-minute window is not normal for a single bin. The system must detect this pattern and raise an advisory alert.
FR-07	Overflow Risk Alerts	When fill level hits 81% or above and is still rising faster than 5% per minute, a CRITICAL alert must go out immediately. This is the highest-priority signal in the module.
FR-08	Event History Storage	All fill readings, lid events, and alert flags must be stored in Firebase in a consistent, queryable structure. Operators need to pull daily and weekly summaries from this without writing custom queries.

FR-09	Routing Module Integration	Live bin status must be published to a shared cloud endpoint that the Predictive Routing module (Component 4) can read. This is what connects my module to the collection optimization pipeline.
--------------	----------------------------	--

Table 3: Functional Requirements of the Smart Fill-Level & Occupancy Monitoring Module

5.3 Non-Functional Requirements

Beyond what the module does, there are clear expectations for how well it does it. The targets below are based on what would make the system genuinely useful in a real operational environment, not just technically functional in a lab.

Category	Requirement	Target / Specification
Performance	Alert Latency	CRITICAL alerts must reach the cloud within 3 seconds of detection. Slower than that and the point of real-time monitoring is lost.
Performance	Data Transmission Rate	At least 98% of sensor readings must make it to the cloud successfully. Packet loss above 2% starts affecting the reliability of fill-level trends.
Reliability	System Uptime	The module must stay online at least 99% of operational hours. Outages mean blind spots in bin monitoring.
Security	Data Encryption	All MQTT traffic uses TLS 1.2 or higher. Data stored in Azure and Firebase is encrypted with AES-256. No sensor data travels or rests unprotected.
Usability	Dashboard Accessibility	An operator looking at the dashboard should see the current fill status of any bin within 5 seconds of the latest reading. Stale data defeats the purpose.
Scalability	Device Capacity	Adding more bins to the system should not require changes to the cloud architecture. The design must support at least 500 bin nodes without redesign.
Maintainability	Firmware Updates	Firmware must be updatable over Wi-Fi using the ESP32's built-in OTA capability. Nobody should need to physically access a deployed bin to push a firmware fix.

Table 4: Non-Functional Requirements

6. COMMERCIALIZATION PLAN

6.1 Target Market and Customer Persona

The most natural early markets for this system are places that already have waste problems and accountability pressures: university campuses that need to manage dozens of bins across a large area and want evidence when waste disposal rules are broken; corporate office parks with sustainability reporting obligations; and municipal local authorities managing high-traffic public bins where overflows create both health risks and reputational damage.

The customer persona in the primary segment is a Facilities or Operations Manager at a medium to large institution - somewhere between 500 and 5,000 people - who is responsible for waste operations and sustainability compliance. They want to know what is happening across their bins without dispatching someone to physically check each one, and they want evidence when something goes wrong, not just a report that a bin was full.

6.2 Value Proposition

The module delivers value in four concrete ways. It cuts unnecessary collection trips by giving operators real visibility into what bins actually need attention. It creates an accountability trail - timestamped event logs that show not just that a bin was full, but how it got that way, which matters when misuse needs to be investigated or reported. It reduces the environmental footprint of waste collection by feeding cleaner, more reliable data into the routing decisions that determine when trucks go out. And it lowers operational costs, with a realistic 20 - 40% reduction in collection frequency for well-managed deployments once demand-driven scheduling replaces fixed routes.

6.3 Revenue Model and Pricing Strategy

The commercialization model is a SaaS subscription combined with hardware leasing for the sensor units. Pricing is tiered by fleet size - a smaller entry plan for campuses running 10 to 50 bins, and a larger enterprise plan for municipal deployments at 500 or more. One-time installation fees and optional premium support contracts round out the revenue streams. Per-bin hardware costs for a prototype unit sit at roughly LKR 8,500 to 12,000, which is competitive against commercial alternatives that often deliver less capability for more money.

6.4 Competitive Advantage and Intellectual Property

Most commercial smart bin products on the market measure fill level and stop there. This system goes further by combining fill data with lid event tracking and behavioral anomaly detection giving institutions something they actually need but cannot get elsewhere: evidence of why bins fill the way they do. The integrated pipeline from edge sensing through ML prediction to dynamic routing also means this is not a standalone sensor product but a complete operational decision support system, which is a meaningfully different and harder-to-replicate proposition. IP considerations include the potential for a provisional patent on the behavioral anomaly detection approach, with open-source hardware components clearly separated from the proprietary software and schema contributions.

7. BUDGET AND JUSTIFICATION

Item	Category	Est. Cost (LKR)	Justification
ESP32 Development Board (x2)	Hardware	3,000	Primary microcontroller for sensor integration and MQTT transmission; x2 for redundancy and parallel testing.
HC-SR04 Ultrasonic Sensors (x4)	Hardware	1,600	Fill-level distance measurement; x4 for multiple bin prototypes.
Magnetic Reed Switches (x4)	Hardware	800	Lid-opening event detection; low cost, interrupt-driven.
Miscellaneous (Resistors, Wires, PCB)	Hardware	2,000	Supporting electronics for circuit assembly and sensor interfacing.
Microsoft Azure IoT Hub (Free Tier)	Cloud Services	0	Free tier provides 8,000 messages/day, sufficient for prototype scale.
Firebase Realtime Database (Spark Plan)	Cloud Services	0	Free Spark plan provides 1 GB storage; adequate for prototype event logging.
Arduino IDE / VS Code / PlatformIO	Software Tools	0	Open-source development environments; no licensing cost.
Python / scikit-learn (Analytics)	Software Tools	0	Open-source ML and analytics libraries.
Human Effort (480 hours @ LKR 150/hr)	Development	72,000	Estimated developer effort over two semesters; costed at standard undergraduate rate for budget accountability.
TOTAL		79,400	

Table 5: Budget and Justification for Smart Fill-Level & Occupancy Monitoring Module

8. WORK BREAKDOWN STRUCTURE (WBS)

The table below breaks my work into twelve tasks across two semesters. Semester 1 focuses on building and connecting the core hardware and firmware. Semester 2 shifts to the anomaly detection engine, cloud-side processing, validation experiments, integration testing with the other team modules, and final dissertation writing.

ID	Task Description	Deliverable	Semester	Duration (weeks)
1.1	Literature review on fill sensing, anomaly detection, and cloud IoT	Annotated bibliography	S1	2
1.2	Hardware procurement and circuit design for HC-SR04 and reed switch	Component list + circuit diagram	S1	1
1.3	ESP32 firmware development: ultrasonic measurement and fill classification	Working firmware v1.0	S1	3
1.4	Lid-event tracking implementation: interrupt handler + event logging	Firmware v2.0 with lid logging	S1	2
1.5	MQTT integration: transmission to Azure IoT Hub and Firebase	Cloud-connected prototype	S1	2
1.6	Proposal documentation	Submitted proposal report	S1	2
2.1	Anomaly detection engine: embedded rule implementation	Firmware v3.0 with anomaly detection	S2	3
2.2	Firebase Cloud Functions: cloud-side rule processing and alert dispatch	Deployed cloud functions	S2	2
2.3	Controlled validation experiments: normal vs anomalous scenarios	Validation test results	S2	3

2.4	Integration testing with Component 3 (ML) and Component 4 (Routing)	Integration test report	S2	2
2.5	Performance evaluation and metrics analysis	Evaluation report	S2	2
2.6	Final dissertation writing and submission	Submitted dissertation	S2	4

Table 6: Work Breakdown Structure – Gamaariya C C

9. GANTT CHART

The chart below lays out the full 32-week timeline across both semesters. Blue cells show active task periods. The Gantt is intentionally front-loaded with hardware and firmware work in Semester 1, freeing Semester 2 for the more complex anomaly detection, integration testing, and dissertation writing.

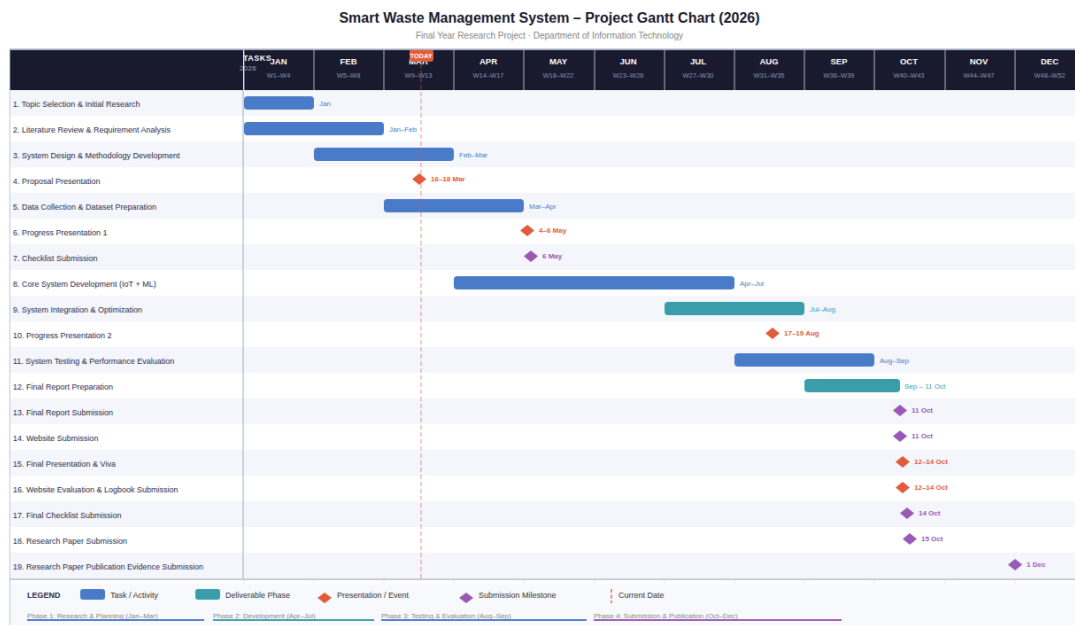


Figure 4: Gantt chart

Four milestones anchor the timeline: Proposal submitted by end of Week 12 (S1). Prototype and validation experiments done by Week 8 (S2). Integration with team modules complete by Week 12 (S2). Final dissertation submitted by Week 16 (S2).

10. REFERENCES LIST

- [1] World Bank, "What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050," World Bank Group, Washington D.C., 2018. [Online]. Available: <https://datatopics.worldbank.org/what-a-waste/>
- [2] 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.
- [3] T. Anagnostopoulos, A. Zaslavsky, K. Kolomvatsos, A. Medvedev, P. Amirian, J. Morley, and S. Hadjiefthymiades, "Challenges and Opportunities of Waste Management in IoT-Enabled Smart Cities: A Survey," IEEE Transactions on Sustainable Computing, vol. 2, no. 3, pp. 275–289, 2017. doi: 10.1109/TSUSC.2017.2691049.
- [4] Md. 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 Journal, vol. 15, no. 1, pp. 561–567, Jan. 2015. doi: 10.1109/JSEN.2014.2345477.
- [5] Md. 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 Systems with Applications, vol. 48, pp. 76–88, Apr. 2016. doi: 10.1016/j.eswa.2015.11.025.
- [6] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," MIS Quarterly, vol. 28, no. 1, pp. 75–105, Mar. 2004. doi: 10.2307/25148625.
- [7] V. Perumal, S. V. Divya, and N. Vishal, "Smart Dustbin using ESP32 for Waste Management," IRO Journal on Sustainable Wireless Systems, vol. 6, no. 4, pp. 309–317, 2024. doi: 10.36548/jsws.2024.4.002.
- [8] 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.
- [9] M. A. Hannan, M. Arebey, H. Basri, and R. A. Begum, "Intelligent Solid Waste Bin Monitoring and Management System," Australian Journal of Basic and Applied Sciences, vol. 4, no. 10, pp. 5314–5319, 2010.
- [10] M. 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 Management, vol. 34, no. 2, pp. 281–290, Feb. 2014. doi: 10.1016/j.wasman.2013.10.030.

11.APPENDICES

Appendix A: Hardware Component Specifications

Component	Key Specifications	Role in Module
ESP32 WROOM-32	Dual-core 240MHz, Wi-Fi 802.11 b/g/n, 34 GPIO pins, 4MB Flash	Central microcontroller; manages sensor I/O, MQTT transmission, and edge anomaly rules
HC-SR04 Ultrasonic Sensor	Range: 2–400 cm; Accuracy: ± 3 mm; Operating Voltage: 5V; Frequency: 40 kHz	Fill-level distance measurement via time-of-flight principle
Magnetic Reed Switch (NO)	Operating voltage: 3.3–5V; Response time: < 1 ms; Contact rating: 0.5A @ 100V DC	Lid-state detection via magnetic field interruption

Appendix B: Firebase Realtime Database Schema

The following JSON structure illustrates the event log schema used in Firebase Realtime Database:

```
smart_bins/  
  {bin_id}/  
    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 }
```

Appendix C: Risk Analysis Matrix

Risk	Likelihood	Impact	Mitigation Strategy
Ultrasonic sensor noise causing wrong fill-level classifications	Medium	Medium	Run a sliding window average over 5 readings before classifying. Calibrate against known distances before deployment. Retest after any physical bin changes.
Wi-Fi drops cutting off cloud data transmission	Medium	High	Buffer readings locally on the ESP32's SPIFFS storage and retry transmission when connectivity returns. Use a queued MQTT publish pattern rather than fire-and-forget.
Too many false-positive anomaly alerts making operators ignore them	Medium	High	Require two consecutive rule triggers before raising a CRITICAL alert. Tune thresholds against real validation data, not just theoretical values. Log all false positives during testing and adjust.
Not enough real-world bin data to validate with	Low	Medium	Generate synthetic datasets based on published waste statistics. Label them clearly as simulated. Use the campus pilot deployment to collect at least some authentic data before final validation.
Azure or Firebase costs running over budget at scale	Low	Low	Stay on free tiers for the prototype phase. Azure IoT Hub free tier covers 8,000 messages/day; Firebase Spark plan covers 1 GB. Monitor usage and only upgrade if the pilot exceeds those limits.