THE PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA MINISTRY
FOR HIGHER EDUCATION AND SCIENTIFIC RESEARCH
IBN KHALDOUN UNIVERSITY TIARET
FACULTY OF APPLIED SCIENCES
ELECTRICAL ENGINEERING DEPARTMENT

Dissertation submitted as a partial fulfilment of the requirements for Master's degree

Field: Science & Technology

Sector: Electrical Engineering

Specialty: EMBEDDED SYSTEMS

Theme

Smart Waste Management with IoT

Presented by

- **CHERFOUH KHALDIA NOURELHOUDA**
- **GUEFFAF MARWA**

Supervised by

- **Dr. AHMED SAFA**

Board of Examiners

• Dr. KOWDRIA MOUHAMED	MCA	Examiner
• Dr. BENBAID HOUARI	MCA	Precident
• Dr. ADDA BENATTIA ABDERAHMANE	MCA	Examiner

Academic year: 2023/2024

Acknowledgements

We are very thankful to all the people who helped us with our graduation project.

*First, we are very grateful to our **Parents**, **Family, and Sisters**. They loved us and supported us all these years. If not for them, we would not be here today.*

Our friends "Sara, Kacem, Zaki , and Aboubaker " also helped and encouraged us a lot. We thank them for being there for us.

*we also thank our teacher, **Dr. Safa AHMED**. He guided us and supported us throughout the project. He was patient with us, shared his knowledge, and encouraged us.*

*We thank the members of the **LGEP** laboratory for letting us use their workspace and for their mental support.*

*We also thank the members of the **jury** who agreed to read and evaluate our work.*

*After continuously seeking divine blessings from **ALLAH**, we were given the help, health, and strength to start and finish this project. Finally, we thank all the professors and people who supported us in any way during this project. Their encouragement meant a lot to us.*

Table of Contents

List of Figures

List of Tables

General Introduction

Public health and environmental policies are connected. Public authorities need to have a smart waste disposal system. This system should be cost-effective. It should also improve public health, local environment quality, and trash transportation.

Smart urban waste management has emerged by combining traditional IT systems with remote communication systems. Many scholars have suggested using low-power wide-area networks and data layers to transfer waste system data. They have also looked at data management technologies. Some studies talk about digital technologies but not specifically waste management information. However, these studies are still related.

This memoir look at the possibilities and challenges of disposing urban trash. It considers the currently available functions and technologies. To handle the waste, we have a fleet of trucks. We discuss transporting garbage using Internet of Things (IoT) ideas. We present the concept of Smart Waste Management (SWM) for waste treatment.

Cities face increasing waste management problems due to urbanization and population growth. Effective waste management shows how well public authorities handle trash. Removing waste from public and urban areas is complicated. It requires different levels of public funding. Improper waste disposal impacts the environment and public health. So waste transportation is very important.

Economic analysis shows the link between environmental sustainability and public health. Addressing urban cleaning meets public health goals. It reduces health hazards from improper waste disposal. In the past, waste segregation lacked effective technology and options.

Now, IoT-based technology allows easy adoption and monitoring of waste management in smart cities. This leads to better quality of life and fewer health hazards due to lower pollution. The smart waste system provides real-time information on garbage bin status. It lowers costs and enables efficient waste collection.

Waste management involves collecting, transporting, and disposing of garbage, sewage, and other societal waste. Waste accumulates in urban areas from homes, businesses, hotels, hospitals, and schools. Proper waste management is crucial for keeping smart cities clean and protecting the environment and public health.

CHAPTER I:
Smart Waste Management Using IoT

I.1 Introduction

Waste management has emerged as a critical issue in urban areas, especially in developing countries. As an undesirable byproduct of human activity, waste processing presents significant challenges for city authorities. Over the past decade, the volume of waste has increased at an alarming rate, posing threats to both the environment and human health. Unmanaged waste can contaminate soil and groundwater, leading to disease proliferation and air pollution.

Waste management involves not only the handling of waste but also converting trash into valuable resources. It is a service essential for every homeowner and business owner, allowing for the effective and safe disposal of outdated items.

Echoing Peter Drucker's words [1], "you can’t manage what you do not measure," it is evident that accurate data is vital in waste management. Currently, many waste collection systems are outdated, leading to excessive or insufficient servicing of public space bins. In fact, over-servicing of these bins can be as high as 81%, which results in unnecessary labor and fuel costs, and contributes significantly—up to 60%—to higher carbon emissions due to inefficiently planned routes and resultant traffic.

Waste management systems encompass all activities and protocols necessary to handle waste from its generation to disposal. This includes transportation, disposal, regulation, and oversight.

Effective waste management is crucial for protecting the environment and ensuring public health and safety. The challenges of waste disposal sites and the increasing global population demand that we address this issue urgently. Poor waste management leads to increased pollutant emissions, ozone layer depletion, and emerging diseases, underscoring the need for robust waste management systems.

This memoir aims to design and implement an IoT-based solution for efficient waste management. By harnessing the power of the Internet of Things (IoT), the proposed system seeks to overcome the inefficiencies in current waste management practices and the lack of real-time monitoring. The IoT solution will enable real-time monitoring of waste levels in bins, optimize transportation routes, and provide valuable data for decision-making.

The design and implementation of this IoT-based solution have the potential to revolutionize how cities and municipalities manage their waste, leading to improved efficiency, cost savings, and environmental benefits. If successful, this approach could serve as a model for other cities and municipalities aiming to optimize their waste management practices.

I.2 Literature Review

A smart waste management system for places with sparse house populations was proposed in the most current work. This may be achieved by setting up an effective waste management system and a central base station that transfers data garbage levels from RF transmitter nodes. Garbage instances would employ a stop-and-shop-based routing protocol to send knowledge-containing messages.

Therefore, by anticipating when a rubbish transportation container will fill up, this method sought to improve garbage transporters efficiency and lessen the labor burden in remote regions

with low population density. Waste was identified locally, and the rubbish transportation system's operation was improved.

I.2.1. Existing Research on Waste Management and IoT

Internet of Things in Waste Management Having a trash can or bins on every corner of a city, where residents may dispose of their garbage, is one of the most popular techniques to waste management. To make recycling easier, different trash types—such as paper and glass—often have their own containers. The fact that the container frequently overflows and discourages people from throwing away their garbage in favor of leaving it on the ground near the bin is one of the main issues with this approach. This is referred to as the overflow issue. Having bins that can wirelessly communicate their fill status to a centralized system is one way that IoT may help with this. Not only can it monitor waste, but it can also be used to communicate with the trash compactors that are automatically installed in each bin and activated when the bin is full. This means that fewer trips are made by the trash trucks to pick up waste, which can reduce the amount of CO2 emissions that are produced during trash collection. The Internet of Things (IoT) offers the possibility of connecting people, processes, data and things around us through sensors, networks and intelligence[2]. IoT has found huge potential uses in waste management system and utilities[3]. The sensors used within IoT will enable a range of new capabilities that are not currently possible or economic[4]. Through sensors, networks, and intelligence, the Internet of Things (IoT) provides the opportunity to link people, processes, data, and objects in our environment. IoT has enormous potential applications in utilities and waste management systems. Many new capabilities that are not now feasible or cost-effective will be made possible by the sensors utilized in the Internet of Things. The worldwide waste management business is highly fragmented, with the four biggest players (Waste Management, Republic Services, Veolia Environment, and Suez Environment) holding a combined market share of just around 25%. This helps to explain the size of the waste management market. If a single business or nation is able to capture a sizable portion of the market, this indicates that there is a big market potential.

Figure I. 1 a guide on IoT-powered smart waste management practices

I.2.2 Studies on Load Cells

Load cells are a crucial component in many monitoring and measurement applications, including waste management systems. These sensors are designed to accurately measure the

weight or force exerted on them, making them ideal for detecting the fill levels of garbage bins or containers.

Several studies have explored the use of load cells in waste management scenarios. Zhang et al. (2015) conducted research on the integration of load cells into a smart garbage monitoring system. Their study demonstrated the effectiveness of these sensors in providing real-time data on the weight of waste within bins, enabling more efficient collection routes and schedules.

Similarly, Mburu et al. (2019) investigated the use of load cell technology in monitoring the fill levels of dumpsters. Their findings highlighted the ability of load cells to reliably detect when bins are nearing full capacity, allowing for timely collection and preventing overflow situations.

Beyond waste management, load cells have found applications in various other industries, such as industrial automation, weighing systems, and force measurement. Their robustness, accuracy, and versatility make them a popular choice for monitoring and control applications.

One notable advantage of load cells is their compatibility with Internet of Things (IoT) systems. By integrating load cell data into an IoT platform, continuous remote monitoring and analysis become feasible. This enables real-time insights into the weight or force measurements, facilitating prompt detection of anomalies or critical situations. Furthermore, load cell data can be combined with other sensor readings, such as motion detectors or fill-level sensors, to provide a comprehensive understanding of the monitored system's state. This multi-sensor approach enhances the overall reliability and robustness of the monitoring solution.

As technology continues to evolve, researchers and engineers are exploring innovative ways to enhance the performance and capabilities of load cells. Advancements in sensor design, signal processing, and data analysis techniques hold the potential to further improve the accuracy, reliability, and ease of integration of these sensors in various applications.

I.2.3 MQTT Communication Technologies

MQTT is more reliable and efficient, allowing for the simultaneous notification of multiple devices with minimal data loss. Therefore, MQTT was used throughout the project to facilitate IoT connectivity. This suggests that ongoing initiatives to manage digital trash and create smart cities are responsive in terms of their capacity to reduce landfill waste, recycle garbage, and utilize waste sustainably.

Through the utilization of effective communication tools, the Internet of Things (IoT) can permit the negotiation of several endpoints that are discreetly scattered. These tools serve as a linking medium between other applications and programs. However, a privately held Transmission Control Protocol (TCP) architecture provides the flexibility of transferring data between different devices simply and securely through the usage of TCP sockets in Internet of Things connections. But among the cloud technologies we have access to include AMQP, ADMQ, and MQTT. Additionally, several forms of quality of service (QoS) are decided by the reimbursement on this particular occasion.

The current study is exclusive to this field of study, and its objective is to provide information for efficient waste management and advancements in the future of MQTT communication technologies. Because of these specific reasons, MQTT needed to be assessed.

These included integrated waste management, its location in relation to contemporary smart cities, and the design and functionality of the WebSocket communication protocol.

In addition to this, its capacity to ensure reliability, flexibility, and consistency in message transmission is what sets it apart. MQTT is simple to use, lightweight, and adaptable to any software.

Due to a lack of contemporary waste control technology and inadequate consideration of the data collecting and assessment system throughout the strategic waste management decision-making process, Furthermore, this advancement on the Internet of Things has made it possible to use an effective replacement for current waste management systems to improve waste control in smart cities[5].

I.2.4 Bin Status Monitoring

It is observed that garbage is frequently ignored, not cleaned in a timely manner, and that improper routing causes delays and fuel waste. Therefore, real-time monitoring will assist in obtaining the most recent data on garbage collection. If the appropriate bins are not taken out of the garbage containers, a lot of money will be squandered. Cans of garbage that are not covered and are just partially filled also smell. Therefore, using containers to their maximum potential helps save money. Regular garbage clearing also helps to minimize spills and odors in the surrounding area.

One of the most important components of a smart waste management system is the tracking of the bins in a city or area. This function is primarily implemented in urban areas, educational institutions, healthcare facilities, and commercial buildings that have extensive internal waste management systems. Effective waste management involves clearing garbage as soon as it reaches a threshold, collecting waste on a scheduled or need-based basis, and—most importantly—keeping track of the state of the bins in real time[6]

The garbage is kept in the bins for a predetermined amount of time and is often loaded with a variety of waste materials. Acquiring up-to-date information on the trash cans not only lowers the expenses associated with the waste removal procedure but also helps prevent garbage from spilling out of the cans, which contributes to environmental damage. Aslam and Sami Ullah (2020) state that it not only deteriorates the place's cleanliness but also could worsen the surrounding area's visual appeal. Regular garbage collection, appropriate disposal, and trash clearing all contribute to excellent hygiene.

Figure I. 2 smart waste management using IoT

I.2.5 Gaps and Challenges in Current Approaches

While IoT-based waste management systems have made significant advancements, several crucial obstacles remain to be overcome for widespread adoption and success. Addressing these issues is imperative, as they have the potential to limit the effectiveness, scalability, and sustainability of these systems. The following sections elaborate on the main shortcomings and challenges associated with current approaches to IoT-based waste management.

I.2.5.1 Data Reliability and Accuracy

Sensor Maintenance and Accuracy: Ensuring the quality and reliability of sensor data is one of the biggest challenges. For instance, load cells are delicate instruments that require regular calibration and maintenance to maintain their accuracy. Environmental factors such as temperature fluctuations, humidity, and physical shocks can lead to inaccurate weight readings. Protocols for routine maintenance and calibration are essential, but they can be expensive and labor-intensive.

Data Integrity and Consistency: Ensuring the reliability and consistency of data gathered from various sensors is crucial. Inconsistent data can lead to poor decisions, such as inefficient routing of garbage collection vehicles or missed pickups. Implementing robust data validation and error-checking procedures is necessary to ensure data integrity.

I.2.5.2 Infrastructure Requirements for Scalability

Logistic and Technical Challenges: Scaling up IoT-based waste management systems to encompass large urban areas presents logistical and technical obstacles. Widespread implementation requires a network of sensors, data processing units, and communication modules as infrastructure. In a large city, this infrastructure needs to be established and maintained with careful planning and significant funding.

Network Congestion and Latency: These issues may exacerbate as more devices connect to the network. Real-time monitoring and decision-making rely on the communication network's ability to handle the increased traffic without experiencing significant delays.

I.2.5.3 Cost and Implementation

Initial Setup and Maintenance Costs: Implementing an IoT-based waste management system can be prohibitively expensive, especially for smaller municipalities or underdeveloped regions. The initial setup involves purchasing and installing sensors, communication modules, and data processing equipment. Ongoing maintenance costs related to system updates, repairs, and calibration must also be considered.

Return on Investment (ROI): Adoption of these technologies hinges on demonstrating a clear ROI. Waste management companies and municipalities must be convinced that the benefits—such as reduced operating costs and environmental impact—outweigh the upfront and ongoing expenses.

I.2.5.4 Standardization and Interoperability

Lack of Standard Protocols: A major issue is the absence of common protocols and interfaces across different IoT devices and platforms. The use of proprietary communication protocols by various manufacturers makes it challenging to integrate devices from multiple

vendors into a single, functional system. Well-defined interfaces and protocols are essential for ensuring seamless communication and integration, as highlighted by [Tan and Lim (2019)].

Interoperability Challenges: Ensuring that the various components of the IoT ecosystem can interoperate with one another is crucial. This involves ensuring that data processing devices, communication modules, and sensors from different manufacturers can successfully exchange and communicate with each other.

I.2.5.5 Energy Utilization

High Energy Requirements for Communication Devices and Sensors: Many IoT devices, particularly those deployed outdoors, operate on batteries. Excessive energy consumption could necessitate frequent battery replacement or recharging, which is not always feasible. emphasizes the need for energy-harvesting technologies and energy-efficient designs to sustain IoT-based waste management systems.

Sustainability: Developing energy-efficient hardware and software solutions is essential for the long-term viability of IoT-based waste management systems. Energy-harvesting technologies, such as solar energy, can alleviate the issue but increase the system's complexity and cost.

I.2.5.6 Data Privacy and Security

Data Security: Ensuring the security of data transmitted over IoT networks is critical. IoT platforms can be vulnerable to cyberattacks, including unauthorized access and data breaches. Robust authentication and encryption procedures are necessary to secure sensitive data.

Privacy Concerns: IoT-based waste management systems often collect data on household waste habits, which may raise privacy concerns. Ensuring the anonymization and responsible use of this data is imperative to maintain public trust and comply with data protection regulations.

I.2.5.7 Technical and Human Resource Challenges Technical Expertise

Implementing and maintaining IoT-based systems requires technical expertise. Waste management organizations and municipalities may face challenges in attracting and retaining qualified personnel to operate these systems.

Awareness and Training: Operators and maintenance staff must receive adequate training to manage IoT systems effectively. This includes the ability to operate the technology, troubleshoot issues, and analyze data to make informed decisions.

I.2.5.8 Environmental and Physical Challenges

Extreme Conditions: IoT devices installed in outdoor waste bins must withstand challenging environmental conditions such as extreme temperatures, rain, dust, and physical impacts. Designing equipment resilient enough to endure these conditions without frequent failures is quite challenging.

Physical Damage and Vandalism: Waste bins and the associated IoT devices are vulnerable to physical damage and vandalism. Ensuring the security and durability of these devices is essential for their long-term operation.

I.2.5.9 Integration with Existing Systems

Compatibility with Legacy Systems: Many communities already have existing waste management systems in place. Integrating new IoT-based solutions with current infrastructure and processes can be challenging, requiring careful planning and execution.

Change Management: Implementing new technology often necessitates changes in organizational workflows and procedures. Effective change management practices are crucial for facilitating the adoption of new systems and ensuring a smooth transition.

IoT-based waste management systems have come a long way, but there are still a number of imp.

Figure I. 3 IoT waste management solution

I.3 Conclusion

efficient waste management techniques are crucial for protecting the environment, safeguarding public health, and fostering sustainable development. The increasing volume of waste and its associated issues, such as pollution and health risks, cannot be effectively addressed by traditional waste management methods.

On the other hand, IoT technology integration in waste management systems provides a game-changing solution that boosts productivity, lessens environmental effect, and promotes sustainable practices. Real-time garbage level monitoring, and better resource use are made possible by IoT-based smart waste management systems. These features result in major advantages including lower operating costs and more efficiency.

Urban environments are changing because of the confluence of waste management and IoT technology, which is simplifying garbage, and creating a sustainable culture. Cities may improve garbage, minimize overflow, and allocate resources efficiently by utilizing IoT-driven waste management systems. These actions will ultimately contribute to a cleaner, greener, and more sustainable future.

In summary, the integration of IoT technology offers a promising solution to revolutionize waste management, enhance efficiency, and promote sustainable practices for a cleaner and healthier environment. Traditional waste management practices, on the other hand, fall short in addressing the challenges posed by increasing waste production.

CHAPTER II:
System Overview

II.1 Introduction

In this chapter, we will go over the system architecture of our IoT-based monitoring system in detail. The architecture employs a network of interconnected devices to efficiently gather, transmit, and display data from a load cell sensor.

The key components of the system include Raspberry Pi, Raspberry Pi Pico, a load cell, an MQTT broker, and a user interface.

II.2 Design and Implementation of an IoT-Based Smart Waste Management System

Block diagram showing the different components used in the Smart Dustbin system

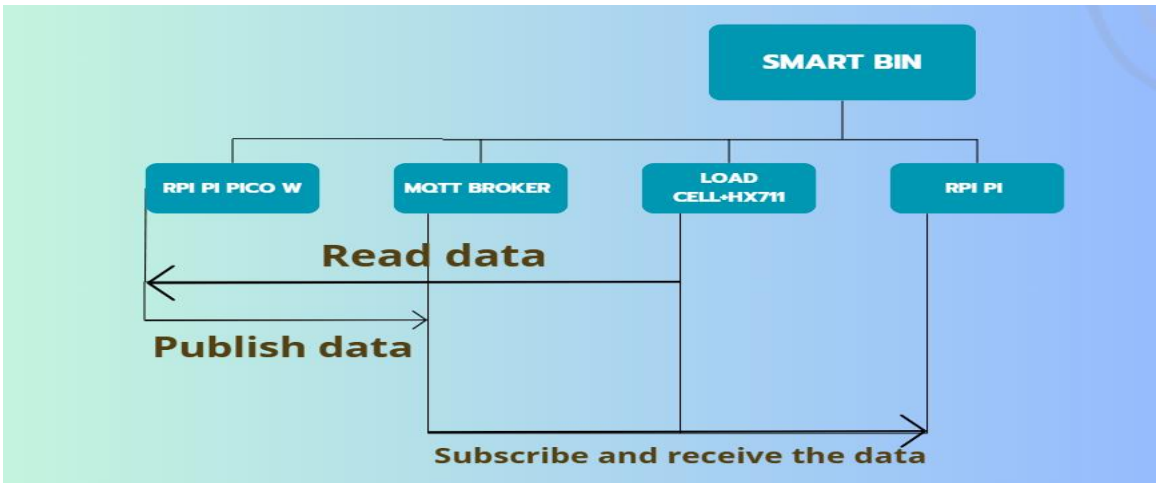


Figure II. 1 Block Diagram of the proposed System

Our smart waste management system comprises several key components:

Sensors and Devices:

- Load Cells: An electronic sensor that We deploy it within waste bins to measure the weight of the waste. These sensors provide real-time data on bin fill levels.
- Raspberry Pi Pico W: A small and cost-effective development board that runs on the Micro Python operating system. Each waste bin is equipped with a Raspberry Pi Pico W microcontroller, which interfaces with the load cell. The Raspberry Pico processes weight measurements and communicates with the central system.
- MQTT Broker: We use the MQTT (Message Queuing Telemetry Transport) protocol for efficient communication between devices. An MQTT broker manages data exchange among sensors, microcontrollers, and the central processing unit (CPU).

Central Processing Unit (CPU):

- Raspberry Pi: A single-board computer more powerful than Raspberry Pi Pico and running on the Raspbian operating system. The Raspberry Pi acts as the brain of our system. It subscribes to MQTT topics, receives data from multiple bins, and performs data aggregation and analytics.
- Data Processing: The CPU aggregates weight data, calculates fill percentages, and predicts when bins will be full. It also stores historical data for trend analysis.

- User Interface and Alerts: A graphical interface that allows users to view information about the status of the waste container. This interface can be a:
  - Alerts: Residents receive notifications (e.g., “Bin empty!”) via the interface, allowing them to dispose of waste promptly.

It is indeed a possibility to implement route optimization and a web/mobile interface in the Raspberry Pi-based smart waste management system. The Raspberry Pi's capabilities make it suitable for running route optimization algorithms and hosting a user-friendly web or mobile interface.

- Route Optimization: Using algorithms, the system can optimize waste collection routes based on real-time data from the bins. This reduces travel time, fuel consumption, and operating costs for the waste management operations.
- Web/Mobile Interface: Waste management personnel and residents can access a user-friendly interface, either through a web application or a mobile app. This interface can display real-time bin fill levels, collection schedules, and alerts, enabling efficient monitoring and communication.

The Raspberry Pi's computing power and ability to run Python and other programming languages make it a viable platform for implementing these features. With the appropriate software development and integration with the existing system components, route optimization and a web/mobile interface can be introduced to enhance the overall efficiency and user experience of the smart waste management system.

Component	**Description**	**Role in Smart Waste Management**
Raspberry Pi Pico W	A microcontroller with Wi-Fi capability	Measures weight using load cell and sends data
Load Cell HX711	A precision weight sensor	Provides accurate weight measurements
Raspberry Pi 3	A higher performance computer module	Receives weight data and processes it
IoT Platform MQTT	A messaging protocol for the Internet of Things	Facilitates communication between devices

Table I: description component for our system

II.2.1.2 Data Flow

II.2.1.2.1 Load Cell Measurements

- The load cell continuously measures the weight of waste in the bin.
- Raspberry Pico reads this data and converts it into a digital signal.

II.2.1.2.2 MQTT Communication

- Raspberry Pico publishes the data to an MQTT topic (e.g., “waste_bins”).
- The MQTT broker receives the data and manages subscriptions.
- Waste management personnel and other subscribers can access this topic.

II.2.1.2.3 Central Processing and Analytics

- Raspberry Pi subscribes to the “waste_bins” topic.
- It receives real-time data from all bins.
- The CPU processes data, calculates fill levels, and predicts when bins need collection.

II.2.1.2.4 User Interaction

- Residents and waste management personnel access the user interface.
- They view bin status, collection schedules, and alerts.
- Personnel optimize collection routes based on real-time data.

The figure below shows the overall flowchart diagram of the implemented garbage monitoring system.

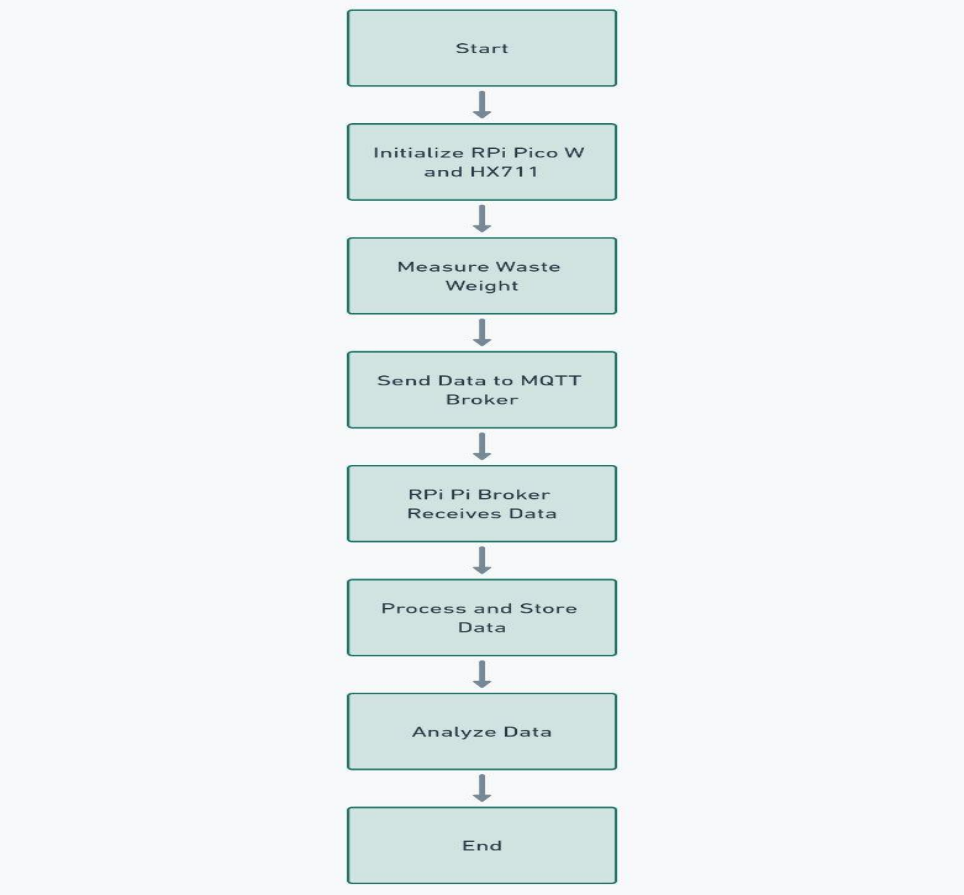


Figure II. 2: Flowchart for our system waste management

II.2.1.3 Communication Protocols

The system employs various communication protocols to ensure efficient and dependable data transfer:

MQTT (Message Queuing Telemetry Transport): A lightweight messaging protocol suitable for IoT applications, facilitating the publish-subscribe model for seamless data transfer between the Raspberry Pi Pico, MQTT broker, and Raspberry Pi.

Raspberry Pico publishes information about MQTT headers (e.g., "trash bins"). An MQTT client receives data and manages subscribers**.**

Wi-Fi: Enables wireless communication between devices, allowing the Raspberry Pi and Raspberry Pi Pico to connect with the MQTT broker over a wireless network, providing seamless communication between devices, promoting flexibility and ease of deployment.

Operation	Description
Publish	Raspberry Pi Pico W publishes the weight data using load cell HX711
Subscribe	Raspberry Pi 3 subscribes to receive the published weight data
Analysis	Raspberry Pi 3 analyzes the received data to determine if the bin is empty or full
Notification	Raspberry Pi 3 sends data or alerts based on the analysis

Table II: Explain how MQTT works in our smart wastebins

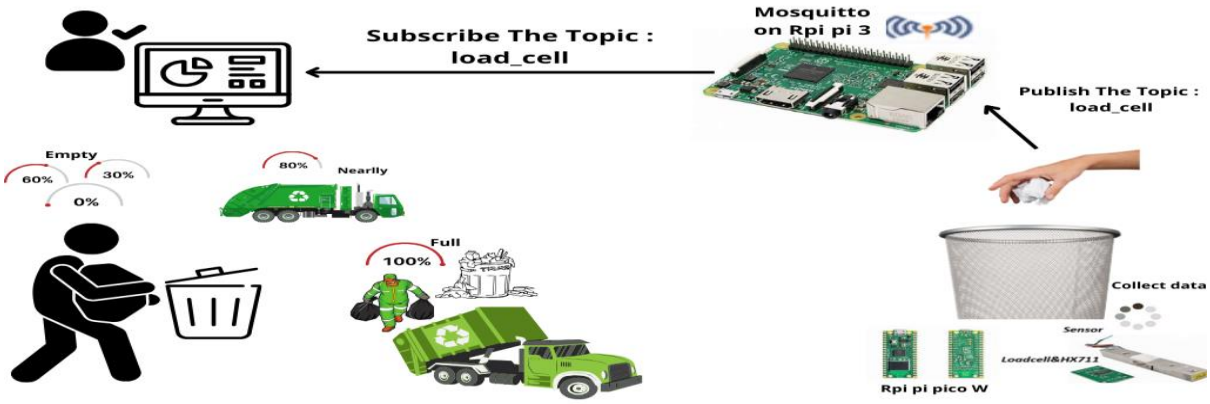


Figure II. 3: how works our smart wastebins

II.3 Conclusion

The system architecture described here leverages the strengths of the Raspberry Pi and Raspberry Pi Pico w, combined with the efficiency of MQTT and Wi-Fi protocols, to create a robust and scalable IoT solution. This architecture ensures accurate data acquisition, reliable data transmission, and user-friendly data display, making it ideal for various monitoring applications.

Our IoT-based smart waste management system offers several benefits:

1. Easy to implement: The system uses readily available and easy-to-use components.
2. Scalability: Additional components can be added to meet specific requirements.
3. Adaptability: The system can be customized to fit different scenarios.
4. Cost-effective: The system uses cost-effective components.
5. Reliable: The MQTT protocol provides a reliable connection for data transmission.
6. Maintainable: The system is easy to maintain and update as needed, including reduced operational costs, lower carbon emissions, and cleaner streets. By integrating technology, we can create more efficient waste collection processes, contributing to sustainable urban environments. As cities continue to grow, such innovative solutions become essential for effective waste management.

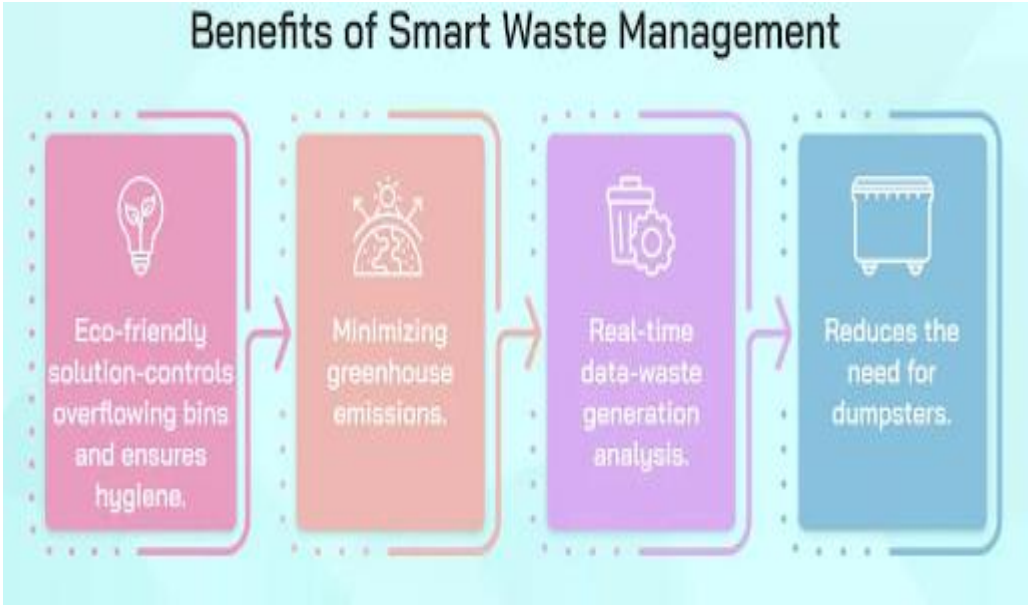


Figure II. 4 Benefits of Smart Waste Management

CHAPTER III:
Hardware & Software

III.1 Introduction

This chapter lays the groundwork for the project's software and hardware. It introduces essential tools for development, including the Raspberry Pi Pico W and Micro Python for code upload. We will explore development environments like Thonny.

Additionally, the chapter covers communication protocols like MQTT and the Mosquitto broker on Raspbian. On the hardware side, We will explore the Raspberry Pi Pico W and its integration with the HX711 load cell, a system designed to measure and monitor garbage weight in real-time. This comprehensive introduction equips you with the software and hardware knowledge needed for building the smart garbage monitoring system in subsequent chapters.

III.2 Hardware Implementation

In this section, we focus on the implementation of a smart garbage monitoring with weight sensing. We explore the utilization of hardware components, including using raspberry Pi Pico W HX711, and load cell is a system designed to measure and monitor the weight of garbage in real-time.

III.2.1 Boards

In our project for smart garbage management systems, we are utilizing the Raspberry Pi Pico W, and the HX711 load cell. These components provide essential capabilities for data acquisition, processing, communication, and control, enabling efficient monitoring and management of waste levels. The Raspberry Pi Pico W serves as the central processing unit, interfacing with the HX711 load cell to measure the weight of the garbage, with rpi pi 3 Wi-Fi connectivity, allows for remote data transmission and system updates, ensuring real-time tracking and optimization of waste collection services.

III.2.1.1 Raspberry Pi Pico W [7]

Raspberry Pi Pico W is a microcontroller board based on the Raspberry Pi RP2040 microcontroller chip realized in 2022.

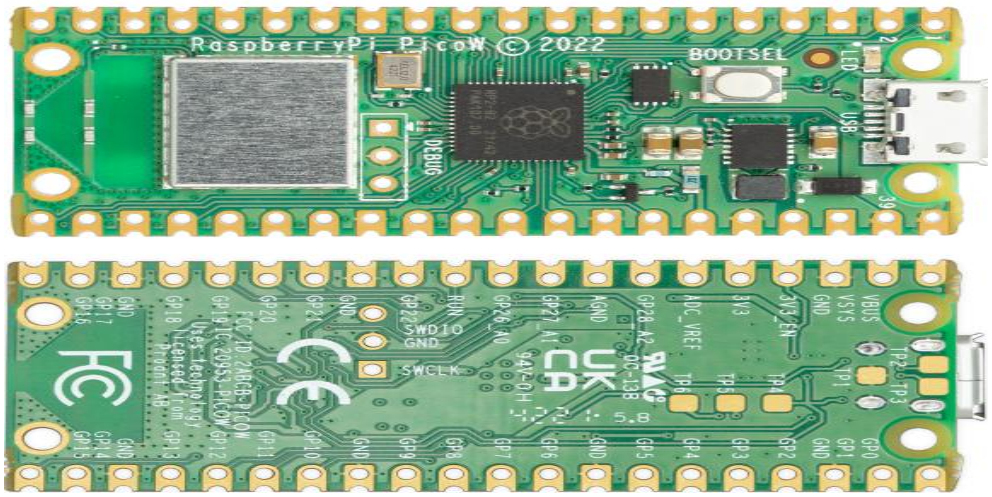


Figure III. 1 Raspberry pi pico w

Raspberry Pi Pico W is a cost-effective development with a 2.4GHz wireless interface.

A. Specification and Features:

➢ RP2040 microcontroller with 2MB of flash memory

➢ On-board single-band 2.4GHz Wireless interfaces (802.11n)

➢ Micro USB B port for power and data, flash reprogramming

➢ 40 pins 21mmx51mm 'DIP' style PCB with 0.1" through-hole pins also with edge castellations.

➢ 26 multi-function 3.3V GPIO including 23 digital-only, with 3 ADC capable

➢ 3-pin Arm serial wire debug (SWD) port

➢ flexible power supply options via micro-USB, external supplies or batteries

➢ High quality, low cost, high availability

➢ Comprehensive SDK, software examples and documentation

➢ Form factor: 21 mm × 51 mm

➢ CPU: Dual-core Arm Cortex-M0+ @ 133MHz

➢ Memory: 264KB on-chip SRAM; 2MB on-board QSPI flash

➢ Interfacing: 26 GPIO pins, including 3 analogue inputs

➢ Peripherals:

- 2 × UART
- 2 × SPI controllers
- 2 × I2C controllers
    - 16 × PWM channels
    - 1 × USB 1.1 controller and PHY, with host and device support
    - 8 × PIO state machines

➢ Connectivity: 2.4GHz IEEE 802.11b/g/n wireless LAN, on-board antenna Bluetooth 5.2

- Support for Bluetooth LE Central and Peripheral roles
- Support for Bluetooth Classic Input power: 1.8–5.5V DC Operating temperature: -20°C to +70°C

➢ Input power: 1.8–5.5V DC

➢ Operating temperature: -20°C to +70°C [8]

B. Pico W pinout:

There are a few RP2040 GPIO pins that are utilized by the board internally:

- GPIO29 OP/IP wireless SPI CLK/ADC mode (ADC3) to measure VSYS/3.

- GPIO25 OP wireless SPI CS - when high also enables GPIO29 ADC pin to read VSYS.
- GPIO24 OP/IP wireless SPI data/IRQ.
- GPIO23 OP wireless power on signal.
- WL_GPIO2 IP VBUS sense - high if VBUS is present, else low.
- WL_GPIO1 OP controls the on-board SMPS power save pin (Section 3.4).
- WL_GPIO0 OP connected to user LED.

Aside from the GPIO and ground connections, the main 40-pin interface has seven extra pins:

- VBUS is the micro-USB input voltage, which is attached to the micro-USB port's pin 1. The nominal voltage is 5V (or 0V when USB is not connected or powered).
- VSYS The on-board SMPs create 3.3V for the RP2040 and its GPIO using the main system input voltage, which ranges from 1.8V to 5.5V.
- 3V3_EN connects to the on-board SMPS enabling pin and is pulled high (to VSYS) with a 100kΩ resistor. To disable 3.3V and power off the RP2040, short this pin low.
- 3V3 is the main 3.3V supply to the RP2040 and its I/O, which is generated by the on-board SMPS. This pin can power external circuitry, with a maximum output current depending on the RP2040 load.
- ADC_VREF is the ADC power supply (and reference) voltage, and is generated on Pico W, if improved ADC performance is needed, this pin can be utilized with an external reference.
- AGND is the ground reference for GPIO26-29. There is a separate analogue ground plane running under these signals and terminating at this pin. This pin can be linked to digital ground in cases where the ADC is not in use, or its performance is not crucial.
- RUN is the RP2040 enable pin and has an internal (on-chip) pull-up resistor to 3.3V of about ~50kΩ. To reset RP2040, short this pin low.

In conclusion, there are six test points (TP1-TP6) that can be accessed as necessary, such as when utilizing the module as a surface mount one. They're:

- The ground known as TP1 (close-coupled ground for differential USB signals).
- TP3 USB DP, TP2 USB DM.
- TP4 WL_GPIO1/SMPS PS pin (do not use).
- TP5 WL_GPIO0/LED (usage not advised).
- TP6 BOOTSEL.

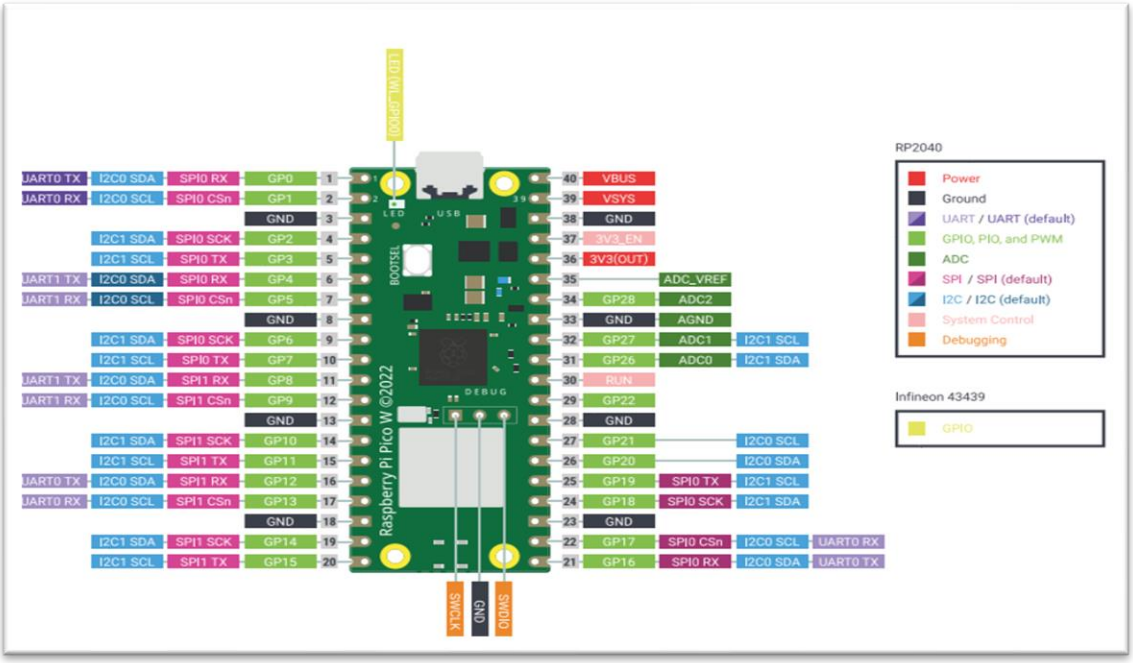


Figure III. 2 The pinout of the Pico W Rev3 board

III.2.1.2 Raspberry Pi 3 model B

The Raspberry Pi 3 Model B is a significant upgrade from the previous models. It was released in February 2016 and is the earliest model of the third-generation Raspberry Pi [9]

Figure III. 3 Raspberry pi 3 model b

A.Specification:

➢ Processor: Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz

➢ Memory: 1GB

➢ Connectivity:

- 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
- 4 × USB 2.0 interface

➢ Video and sound: • MIPI CSI camera port

• 1 x full size HDMI

• MIPI DSI display port

• 4 pole stereo output and composite video port

➢ Multimedia: H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics ➢ SD card support: Micro SD format for loading operating system and data storage Input Power:

• 5V/2.5A DC via micro-USB connector

• 5V DC via GPIO header

• Power over Ethernet (PoE)-enabled (requires separate PoE hat)

➢ Operating temperature: 0-50°C

➢Production lifetime: Raspberry Pi 3 Model B+ will remain in production until at least January 2028

➢ sensor

➢ load cell

A. Introduction

Figure III. 4: load cell

Load cells are instruments that measure force and are widely used in industrial and scientific applications. They can determine the weight of an object, the force a fluid exerts on a surface, or the tension within a cable. Typically, a load cell is made up of a metal shaft or bar, which is supported on each end by a strain gauge.

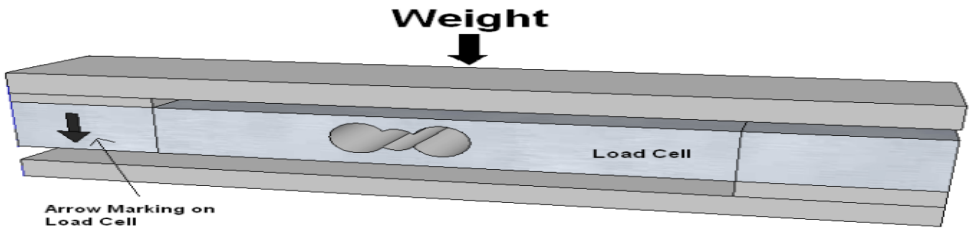


Figure III. 5: load cell sensor

Load cell is a force sensing module, which consists of a precisely engineered metal framework with tiny components called strain gauges positioned at strategic points throughout. Because of its design, load cells only register one force and ignore other applied forces. The load cell produces a very small electrical signal, which calls for specific amplification. Thankfully, all the electrical output amplification and measurement will be handled by the HX711. It amplifies the signal from a load cell and provides a clean, stable digital value. Force can only be measured in one way using load cells. Since portions of the load cell that were previously running under compression are now functioning under tension, and vice versa, they will frequently measure force in different directions, although the sensor sensitivity will differ.

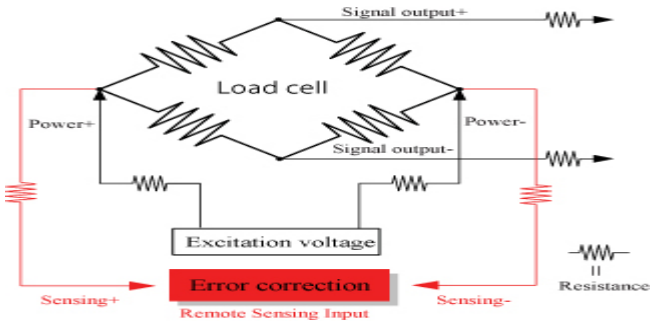


Figure III. 6: circuit of load cell

B. Load cell function:

The strain gauges detect the beam's distortion because of an applied force and generate a signal that is proportionate to the applied force. A variety of load cell types and models are available.

In bar strain gauge load cells, two strain gauges measure compression and two measure tension, while the fourth gauge detects bending distortion when torque is applied to the bar. The cell is configured in a "Z" shape, which facilitates the precise measurement of slight resistance changes from these four strain gauges when they are connected in a Wheatstone bridge configuration. [10]

When the force applies to the load cell, two of the strain gauges will compress (green) while the other two will stretch (blue).

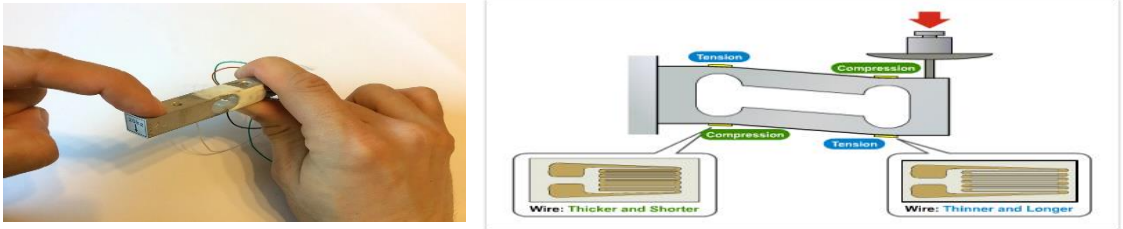


Figure III. 7: More in-depth diagram of strain gauges on bar load cells when force is applied

C. Calibration: [11]

A simple formula is usually used to convert the measured mv/V output from the load cell to the measured force: Measured Force = A * Measured mV/V + B (offset). It is important to decide what unit your measured force is - grams, kilograms, pounds, this load cell has a rated output of 1.0±0.15mv/v which corresponds to the sensor's capacity of 20kg to find A we use:

- Capacity = A * Rated Output

- A = Capacity / Rated Output
- A = 20 / 1.0
- A = 20

Since the Offset is quite variable between individual load ells, it is necessary to calculate the offset for each sensor.

Measure the output of the load cell with no force on it and note the mv/V output measured by the Whitebridge.

Offset = 0 - 20 * Measured Output

D. Specification:[12]

Mechanical:

➢ Housing Material: Aluminum Alloy.

➢ Load Cell Type: Strain Gauge.

➢ Capacity: 20kg.

➢ Dimensions: 55.25x12.7x12.7mm.

➢ Mounting Holes: M5 (Screw Size).

➢ Cable Length: 550mm.

➢ Cable Size :30 AWG (0.2mm).

➢ Cable - no. of leads :4 (RED WHITE GREEN BLACK).

Electrical:

➢ Precision: 0.05%.

➢ Rated Output: 1.0±0.15 mv/V.

➢ Non-Linearity: 0.05% FS.

➢ Hysteresis: 0.05% FS.

➢ Non-Repeatability: 0.05% FS.

➢ Creep (per 30 minutes): 0.1% FS.

➢ Temperature Effect on Zero (per 10°C): 0.05% FS.

➢ Temperature Effect on Span (per 10°C): 0.05% FS.

➢ Zero Balance: ±1.5% FS.

➢ Input Impedance:1130±10 Ohm.

➢ Output Impedance:1000±10 Ohm.

➢ Insulation Resistance (Under 50VDC) ≥5000 MOhm.

- ➢ Excitation Voltage:5 VDC .
- ➢ Compensated Temperature Range: -10 to ~+40°C.
- ➢ Operating Temperature Range: -20 to ~+55°C.
- ➢ Safe Overload: 120% Capacity.
- ➢ Ultimate Overload: 150% Capacity.

Load cell has red,black,green and white wires.

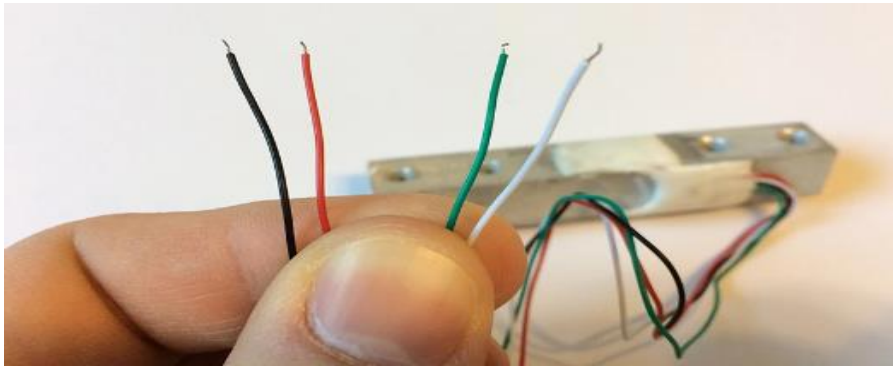


Figure III. 8 wires of sensor

HX711 Load Cell Transmitter:

Dual-Channel 24 Bit Precision A/D weight Pressure Sensor Load Cell Amplifier and ADC Module is a small breakout board for the HX711 IC that allows you to easily read load cells to measure weight. By connecting the module to microcontroller will be able to read the changes in the resistance of the load cell and with some calibration. will be able to get very accurate weight measurements.

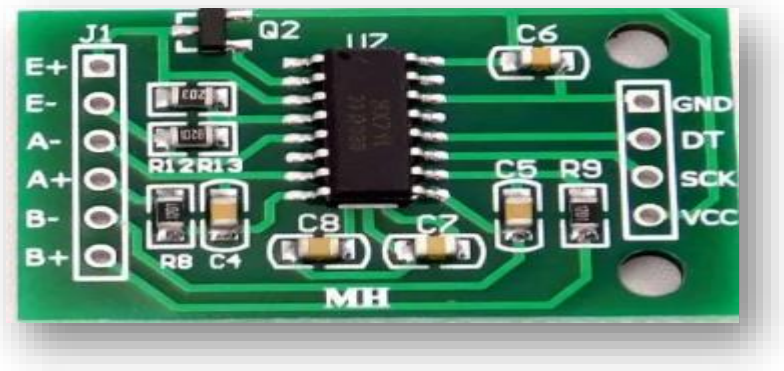


Figure III. 9: driver Hx711 24 bit

This 24-bit analog to digital and signal conditioning module is made specially to interact directly with a bridge sensor in industrial control applications, weight scales, and weight sensors. On the HX711 chip, the module is built.

Since the output range of a strain gauge is typically quite tiny in load cells and weight sensors, the signal must be amplified prior to processing to avoid the introduction of errors.

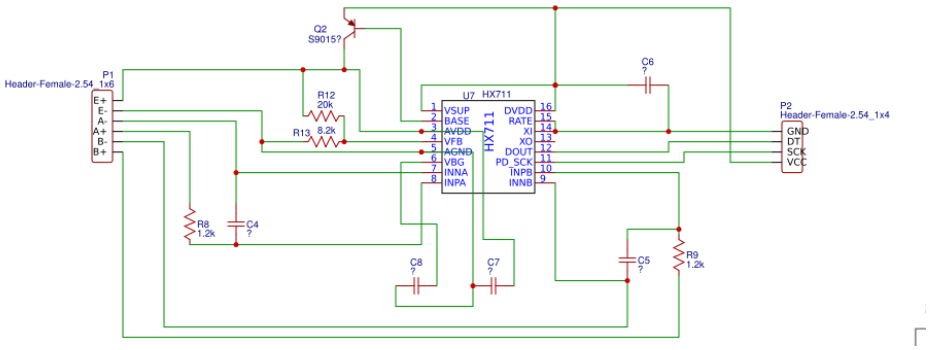


Figure III. 10: circuit of hx711

This module increases the precision of the measurement by amplifying the weight sensor and transforming it from an analog to a digital sensor. The microcontroller will get a serial output. Two weight sensors can be attached to the module simultaneously. Additionally, the module uses an out+ pin to directly supply power to the sensors.

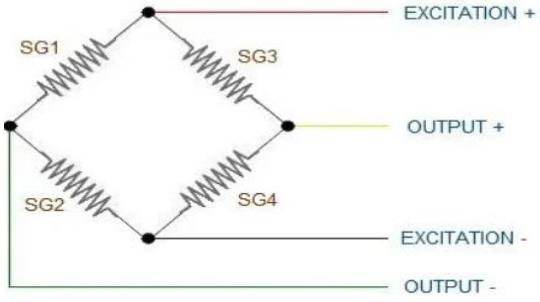


Figure III. 11: Excitation & Output of Module

Load cell wiring

Features & Specification:

➢ Two selectable differential input channels

➢ On-chip active low noise PGA with selectable gain of 32, 64 and 128

➢ On-chip power supply regulator for load-cell and ADC analog power supply

➢ On-chip oscillator requiring no external component with optional external crystal

➢ On-chip power-on-reset

➢ Simple digital control and serial interface: pin-driven controls, no programming needed

➢ Selectable 10SPS or 80SPS output data rate

➢ Simultaneous 50 and 60Hz supply rejection

➢ Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA

➢ Operation supply voltage range: 2.6 ~ 5.5V

➢ Operation temperature range: -40 ~ +85℃

➢ 16 pin SOP-16 package

PIN OUT:

The HX711 communicates with the microcontroller using a two-wire serial interface, similar to I2C but simpler in implementation, it is a simpler protocol with only two lines (Clock and Data) and does not require additional addressing or control lines. This protocol uses two pins:

1. Clock (CLK): Synchronizes data transmission.
2. Data (DOUT): Transmits the data from HX711 to the microcontroller

These two pins allow the microcontroller to read data from the HX711 module in a synchronous manner, where the clock signal from the microcontroller coordinates the timing of the data transmission.

To connect a load cell to the HX711 module, 4 cables are required, the colors generally used are Red, Black, White and Green. Per color corresponds to a signal as shown below:

- Red: Excitation voltage +, E+, VCC
- Black: Excitation voltage -, E-, GND
- Green: Amplifier -, Signal -, A-
- White: Amplifier +, Signal +, A+

III.2.2 Load Cell Connection and Calibration

For wiring the load cell and HX711 with the Raspberry Pi Pico W, we using GPIO pins and similar wiring principles as with the Raspberry Pi Pico W.

Here we are showing how to wire it:

- Load Cell to HX711:
  - Red wire (VCC) to E+ on HX711
  - Black wire (GND) to E- on HX711
  - White wire to A- on HX711
  - Green wire to A+ on HX711
- HX711 to Raspberry Pi Pico W:
  - VCC on HX711 to 3V3 on Pico W
  - GND on HX711 to GND on Pico W
  - DT (Data) on HX711 to a GPIO pin (GP12)
  - SCK (Clock) on HX711 to a GPIO pin (GP13)

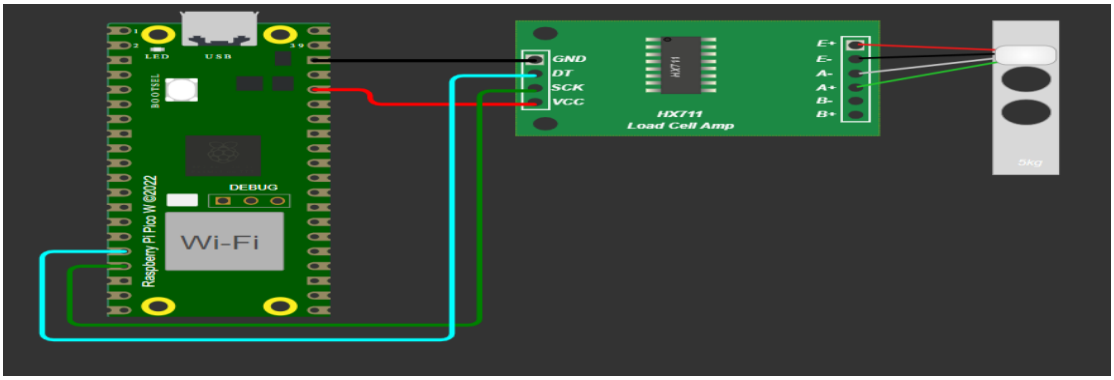


Figure III. 12: schema of our system

III.2.3 Wiring and Power Considerations

III.2.3.1 Wiring Considerations

Cable Management: We will make sure all cables are neatly routed and secured with cable ties. Loose hanging wires risk getting accidentally disconnected or damaged.

Connections: For initial prototyping, a breadboard will simplify the wiring between the Pico, HX711, and load cell. But for a more permanent setup, we may look into a custom PCB that neatly houses the HX711 and load cell connections. This reduces risks of loose wires affecting measurement accuracy.

Signal Integrity: To minimize electrical noise, We will keep the wiring between the load cell and HX711 as short as possible. If the environment is electrically noisy, shielded cables can further protect the signal.

III.2.3.2 Power Considerations

Powering this system properly is crucial for stable and reliable operation. We will start by ensuring the Raspberry Pi Pico W receives a clean, consistent 3.3V supply - either through its micro-USB port or a dedicated regulated 3.3V power source. Using the USB is convenient for development, but a regulated supply may be preferred for the finally deployed system.

The HX711 can be powered directly from the 3.3V pin on the Pico W, which simplifies the power requirements and ensures compatibility

Having a single 3.3V source really simplifies things, as we can power the HX711 load cell amplifier directly from the Pico W's 3.3V pin. This amplifier is designed for ultra-low-power operation, so it can run efficiently right off that 3.3V line without needing a separate supply rail.

While the Pico W itself doesn't draw much current, we must account for the added load from the HX711 and load cell. While the Pico W itself doesn't draw much current, we must account for the added load from the HX711 and load cell. To calculate the total current draw across all components and ensure it doesn't exceed the maximum rated current capacity of the Raspberry Pi Pico W, we need to add up the current consumption of each component connected to the 3.3V supply. Here’s how you can do the math:

- Determine the Current Draw of Each Component
- Raspberry Pi Pico W: The typical operating current of the Pico W is around 50-100 mA, depending on what peripherals and features are being used.
- HX711: The HX711 typically consumes around 1.5 mA.
- Load Cell: The load cell itself doesn't draw current directly; its power consumption is accounted for in the HX711's consumption.

Sum the Currents:

- Raspberry Pi Pico W: Let's assume 80 mA (a midpoint in its typical range).
- HX711: 1.5 mA.
- Total current draw = 80 mA (Pico W) + 1.5 mA (HX711) = 81.5 mA.

Verify Against the Pico W's Maximum Current Capacity:

- The maximum current capacity of the Raspberry Pi Pico W's 3.3V supply pin is around 500 mA.

- Total current draw: 81.5 mA.
- Maximum capacity: 500 mA.
- Since 81.5 mA is well within the 500-mA limit, the setup is safe in terms of current draw.

To further increase power resilience, we may investigate adding some filtering capacitors right at the 3.3V pins of the HX711 and Pico W. This can help smooth out any minor voltage fluctuations or noise on the supply line that could affect readings. And a simple diode or other protection circuit guards against reverse polarity connections.

With a clean, sized-right 3.3V supply distribution, power isolation, and protection measures, we can ensure this weighing system has a solid foundation for reliable data. Properly powering everything is a critical first step.

III.2.4 Integration with the Waste Bin

III.2.4.1 Load Cell Placement

Structural Support: The load cell needs to be mounted on a firm, flat surface that can support the bin's weight without flexing. For the waste bin, this could be at the base away from falling debris.

III.2.4.2 Load Distribution

We will use a mounting bracket or plate to evenly distribute the load across the load cell's surface for consistent measurements, regardless of how waste is distributed in the bin.

For mounting the load cell, we took advantage of 3D printing. We 3D printed a base frame and a weighing plate tailored to our needs. While not rated for the full 20kg capacity, these printed parts are plenty sturdy for our experiments.

3D printing allowed us to customize the mounting plates for ideal fit and spacing with the load cell. This ensures proper load distribution and those critical air gaps needed for accurate strain measurement.

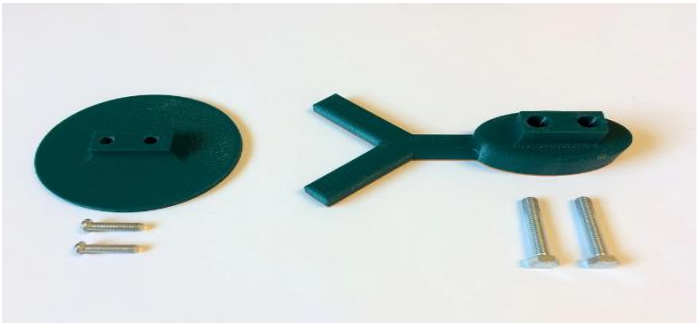


Figure III. 13: mechanism 3D printed

Proper mounting orientation is key. The load cell has an arrow indicating the force direction. We positioned it so this arrow points down, with the 3D printed weighing plate secured on top. This way, when weight presses down on the plate, that force transfers onto the load cell in the proper direction.

For the other end, we connected it directly to the 3D printed base frame below. This suspends the weighing plate between the two ends of the load cell. The top plate pushes down, while the base anchors, inducing the necessary mechanical strain throughout the body.

The strain gauges inside can then convert this deformation into an accurate electrical weight signal. Careful attention to the load cell's arrow orientation, combined with our precise 3D printed mounting solution, enabled reliable operation.

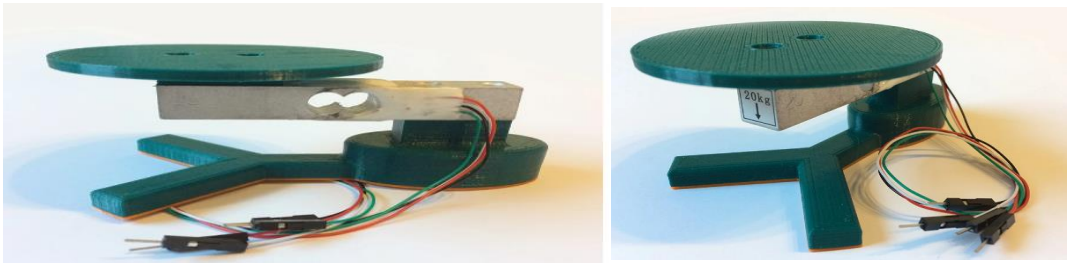


Figure III. 14: load cell with 3D printed

III.2.4.3 Protecting the Electronics

Environmental Protection: An enclosure will protect the Pico and HX711 from dust, moisture and harsh environments, while still allowing ventilation to prevent overheating.

III.2.4.4 Accessibility

We will position the electronics for easy access for maintenance, while still securing them against tampering.

III.2.4.5 Maintenance and Calibration

III.2.4.5.1 Regular Calibration

We will schedule periodic calibrations by placing known weights in the bin and adjusting the scale calibration factor in software to maintain accuracy long-term.

III.2.4.5.1 Maintenance Access

The installation will be designed for easy access to all components, with connectors used where possible for quick disconnection or replacement.

III.3 Software Integration

Real-Time Monitoring: Software will read the MQTT weight data and determine the bin's status in real-time, sending alerts when nearing or reaching capacity.

Data Logging: Weight data logs will be kept for analyzing usage patterns and optimizing collection schedules over time for maximum efficiency.

Proper wiring, power design, environmental protection, and software integration will all come together for a robust smart waste bin monitoring solution!

III.4 Software Development

In this section, we will delve into the detailed steps and processes involved in the software development aspect of our IoT-based smart waste management system. This includes the code

implementation for various components such as the Raspberry Pico firmware, MQTT client, Raspberry Pi software, and the user interface.

III.4.1 Code Implementation

The Raspberry Pi Pico can be programmed in various programming languages, including Circuit Python, Micro Python C /C++, and even in assembly language, making it suitable for a wide range of applications. In our project we are programming with:

- **Python:**

Python is a versatile language that can be used in many different domains, such as web development, data analysis, machine learning, and automation. It is also relatively easy to learn, which makes it a good choice for beginners.

Since its launch in 1991, the Python programming language, named after the popular comedy group Monty Python, rather than a snake, has become one of the most popular in the world. But just because it is so popular doesn't mean improvements can't be made to them, especially if you're working with a microcontroller. The Python programming language is designed for computer systems such as desktops, laptops, and servers. Microcontroller boards like the Raspberry Pi Pico W are small and compact, and they have very little memory; This means they cannot use the same Python language as their larger counterparts. This is where it comes into play. Originally developed by Damien George and first released in 2014, it is a Python-based programming language developed specifically for microcontrollers. While it includes many of the core Python features, they add a new set of features designed to use the hardware found in the Raspberry Pi Pico and other microcontroller boards. If you've programmed with Python before, you'll find it familiar. If not, do not worry, it is an easy language to learn.

Micro Python: is a full implementation of the Python 3 programming language that runs directly on embedded hardware like Raspberry Pi Pico. You get an interactive prompt (the REPL) to execute commands immediately via USB Serial, and a built-in file system. The Pico port of Micro Python includes modules for accessing low-level chip-specific hardware.[13]

- **IDE'S:**

We use IDE' s Thonny (Integrated Development Environment) and Vs code (visual studio) as the main development tools in our project. These IDE's enable rapid development and programming by providing an easy-to-use and efficient way to code, compile, and port software to compatible microcontrollers**.**

- **Visual Studio:**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). [14]

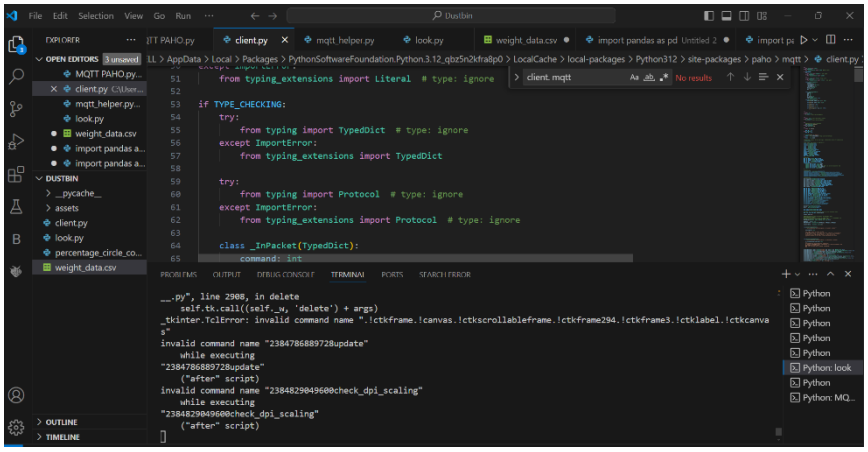


Figure III. 15: VS Code

- **Thonny:**

Thonny is a simple Python IDE designed for beginners, Thonny is an open-source IDE which is used to write and upload Micro Python programs to different development boards such as Raspberry Pi Pico, ESP32, etc . It is extremely interactive and easy to learn IDE as much as it is known as the beginner-friendly IDE for new programmers. With the help of Thonny, it becomes very easy to code in Micro python as it has a built-in debugger that helps to find any error in the program by debugging the script line by line.[15]

Thonny is an open-source Python IDE that allows you to write and upload Micro Python programs to various development boards such as Raspberry Pi Pico, ESP32, and others. It is incredibly dynamic and easy to learn IDE, as well as being renowned as the beginner-friendly IDE for beginners

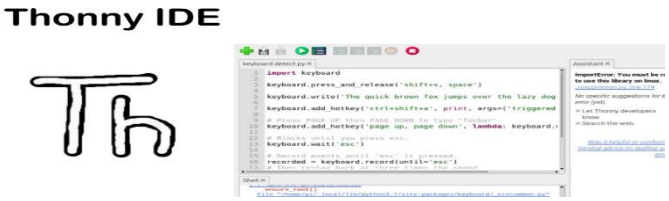


Figure III. 16: Thonny IDE

- **Raspberry Pico Firmware:**

In this section, we will learn how to use Thonny to develop and run programs.

It is very easy to get started with Raspberry Pi Pico using MicroPython;

1. Connect the Raspberry Pi Pico to your computer while holding the BOOTSEL button at the same time to enter bootloader mode and flash the firmware.

Figure III. 17: rpi pi pico w

2. Open Thonny IDE. Then go to **Tools** > **Options**. Select the **Interpreter** tab on the new window that opens.
3. Choose MicroPython (Raspberry Pi Pico) as the interpreter and try to detect port automatically for the Port. Then, click on Install or update MicroPython.

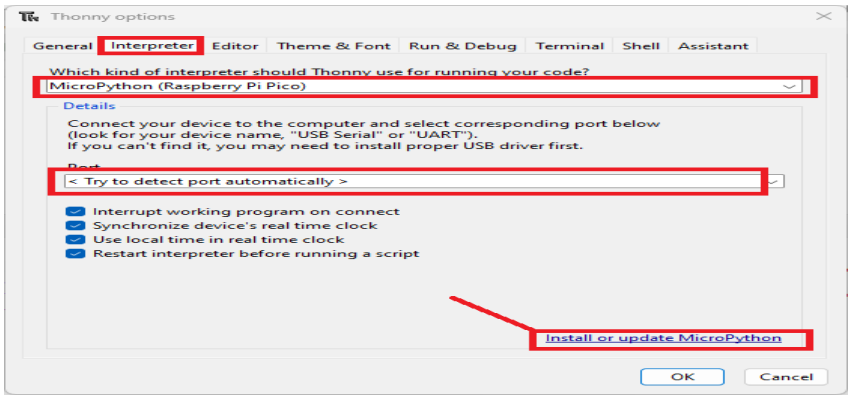


Figure III. 18: Thonny option

The following window will open.

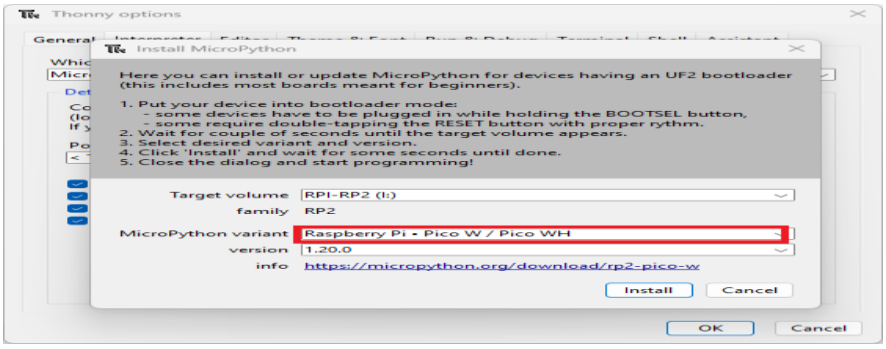


Figure III. 19: figure showing how install micropaython

4. Select the MicroPython variant according to the board you're using. In our case, we are using the Pico W. Select a different option if you're using the Pico.
5. Finally, click **Install**.
6. After a few seconds, the installation should be completed.

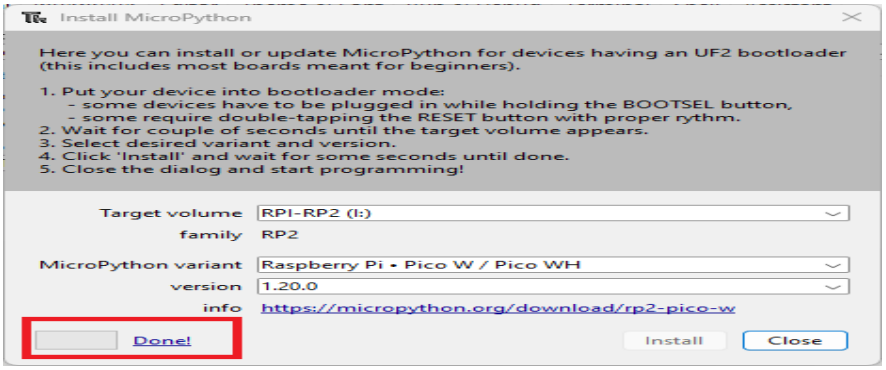


Figure III. 20: MicroPyhton in rpi pi pico w

7. You can close the window. You should get the following message on the Shell, and at the bottom right corner, it should have the Interpreter it is using and the COM port.[16]

III.4.1.1 Raspberry Pico Firmware for Load Cell Data Acquisition

The Raspberry Pico will be the workhorse of our data acquisition process, responsible for interfacing with the load cell and measuring the weight of the waste in each bin.

For this critical role, we're using the HX711_PIo library, which is specially designed for connecting HX711 load cell amplifiers to microcontrollers like the Pico. This library really streamlines the process of reading data from load cells by tapping into the Pico's powerful PIO (Programmable I/O) hardware capabilities.

The PIO is like a tiny programmable processor built into the Pico's chip, dedicated solely to handling I/O tasks. By offloading the load cell data acquisition to this separate PIO processor, we free up the Pico's main processor cores to handle other tasks without getting bogged down.

The HX711_PIo library gives us a set of convenient functions for initializing the PIO state machine that will interface with the HX711 amplifier. We can customize settings like the clock rate, data format, gain factor, and more.

Once initialized, the real magic happens with the library's read_data() function. This kicks off the PIO state machine which rapidly clocks in the 24-bit data stream coming from the HX711 amplifier. The incoming bits are deserialized, formatted, and the final weight value is calculated - all handled ultra-efficiently by the dedicated PIO hardware.[17]

Instead of our main application code having to meticulously clock in each bit using elaborate timing loops, the PIO seamlessly takes care of that low-level I/O work in the background. This allows our firmware to simply call read_data() whenever it needs an updated weight reading.

By leveraging the HX711_PIo library and the Pico's PIO capabilities, we get fast, accurate, and reliable load cell data acquisition without taxing the main processors. Combine that with the Pico's low cost and compact form factor, and we've got an excellent embedded data acquisition solution for our smart waste bin project.

- **Communication Protocol:**

In our project, we use MQTT (Message Queuing Telemetry Transport)

These protocols enable efficient and dependable communication between devices and systems, allowing for the interchange of data and commands necessary for successful monitoring and control in our project's context.

MQTT Protocol: was designed for the Internet of Things (although it was not called that at the time) allows messages to pass in both directions between clients and servers.

- **MQTT Characteristics**
  - ✓ MQTT is a "PUSH" system, in which producers send data to brokers.
  - ✓ MQTT protocol is considered a lightweight protocol in which its header size is 2 bytes.
  - ✓ MQTT is designed for low power consumption as it is used for low power IOT devices.

✓ MQTT communication operates as a publish-subscribe system. Devices send signals about a given topic. All devices that have subscribed to that topic receive the message.

Figure III. 21: MQTT Publish Subscribe

MQTT offers several key benefits:

- **Lightweight and efficient**: MQTT minimizes the resources required by clients and network bandwidth.
- **Bidirectional communication**: MQTT facilitates communication between devices and servers, supporting publishing and subscribing. It also allows broadcasting messages to groups of devices.
- **Scalability**: MQTT can scale to support millions of devices or "things" in an IoT or IIoT ecosystem.
- **Quality of Service (QoS) levels**: MQTT specifies different QoS levels to ensure reliable message delivery.
- **Persistent sessions**: MQTT supports persistent sessions between devices and servers, reducing reconnection time over unreliable networks.
- **Security features**: MQTT supports TLS encryption for message confidentiality and authentication protocols for client verification.

MQTT Basic Concepts:[18]

- Messages: Messages are the information that you want to exchange between your devices. It can be a message like a command or data like sensor readings, for example.
  - ➢ Topics: Topics are the way you register interest for incoming messages or how you specify where you want to publish the message .[19]
- Publish/Subscribe: In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages o for example Device 1 publishes on a topic.

Device 2 is subscribed to the same topic that device 1 is publishing in, so device 2 receives the message.

Figure III. 22: how two devices publish & subscribe in the same topic

- Broker: The MQTT broker is responsible for receiving all messages, filtering the messages, deciding who is interested in them, and then publishing the message to all subscribe clients.

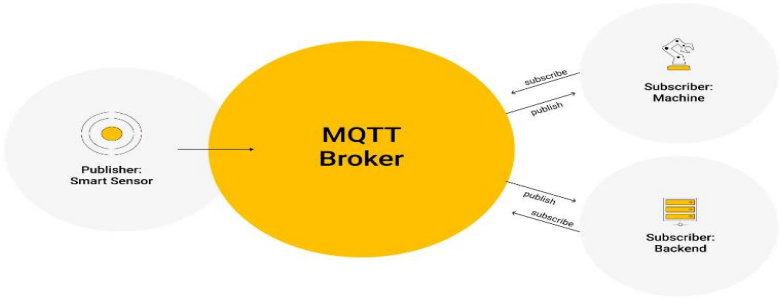


Figure III. 23: MQTT Broker

- **How MQTT work :[20]**

The Internet of Things (IoT) thrives on communication, and that's where the MQTT protocol comes in. Imagine a network of resource-constrained devices, like smart waste bins, needing to exchange information. MQTT provides a lightweight yet efficient way for these devices to publish (send) and subscribe (receive) data to a central server called a broker.

Here is a breakdown of how MQTT works:

- **Establishing a Connection:** An MQTT client (like our smart bin) initiates a connection with the broker using a standard TCP/IP port. The broker then checks if this client has an existing session (think of it like a saved login). If it does, the old session can be resumed. For new clients, a fresh session is established.
- **Authentication:** Security is important! MQTT offers two options. For secure connections, clients can authenticate with the broker using a username and password. For open (public) brokers, anonymous connections might be allowed, but this is not recommended for sensitive data.
- **Communication:** Once connected and authenticated, the fun begins! Clients can now perform various actions:
  - **Publishing:** Clients can publish data (like the fill level of a bin) on specific topics (e.g., "bin_status"). The message content is in a binary format and can be anything relevant to the application.
  - **Subscribing:** Clients can subscribe to specific topics. The broker will then forward any messages published on those topics to the subscribed clients. This ensures that devices only receive the information they need.
  - **Unsubscribing:** Clients can unsubscribe from topics they no longer need to receive updates on.
  - **Pinging:** Clients can periodically send "ping" messages to the broker to keep the connection alive and confirm it is still open.
  - **The Key Takeaway:** MQTT is a publish-subscribe messaging protocol that enables efficient communication between resource-constrained devices and a central broker. By enabling targeted data exchange, MQTT plays a crucial role in connecting the ever-growing world of IoT devices.

Important Points to Note:

- Clients do not have addresses like in email systems, and messages are not sent to clients.
- Messages are published to a broker on a topic.
- The job of an MQTT broker is to filter messages based on topic, and then distribute them to subscribers.
- A client can receive these messages by subscribing to that topic on the same broker
- There is no direct connection between a publisher and subscriber.
- All clients can publish (broadcast) and subscribe (receive).
- MQTT brokers do not normally store messages.

III.4.1.2 MQTT Client Code for Data Transmission

III.4.1.2 .1 Overview of MQTT in our project

Our IoT-based smart waste management system utilizes the MQTT (Message Queuing Telemetry Transport) protocol for its efficient and reliable data communication. MQTT's lightweight nature makes it ideal for resource-constrained devices and low-bandwidth networks prevalent in IoT deployments.

The Raspberry Pi Pico, running MicroPython firmware, acts as the primary data acquisition unit, collecting weight data from load cell sensors attached to waste bins. To seamlessly integrate with the MQTT infrastructure, we leverage the umqtt_simple library, a streamlined MQTT client specifically designed for MicroPython environments.

This choice offers several advantages. Firstly, umqtt_simple seamlessly integrates with the Raspberry Pi Pico, ensuring efficient implementation within its limited resources. Secondly, it focuses on essential functionalities like connect, disconnect, publish, and subscribe, minimizing processing overhead. Finally, its simplicity facilitates rapid development, allowing our team to focus on core functionalities rather than complex MQTT implementation details.

Weight data acquired by the Raspberry Pi Pico is published to an MQTT broker using umqtt_simple. This broker acts as a central hub, distributing data to other system components, like the Raspberry Pi. The Raspberry Pi subscribes to relevant topics to receive and process the published data.

By leveraging MQTT and the lightweight umqtt_simple library, our system achieves efficient and reliable data communication, enabling real-time monitoring and analysis of waste levels across various locations.

III.4.1.2.2 Overview of Setting Up Mosquitto MQTT Broker on Raspberry Pi

Setting up Mosquitto, a lightweight and efficient MQTT broker, on a Raspberry Pi is a crucial step in developing a robust IoT system. Mosquitto acts as the central hub for MQTT messaging, allowing various devices to communicate seamlessly. This setup involves several key steps, starting with the installation of Mosquitto and its client tools using the Raspberry Pi's package manager. Once installed, enabling the Mosquitto service to start at boot ensures continuous operation. Basic configuration adjustments, such as setting up logging, listeners,

and optional authentication, help tailor the broker to specific project needs. After configuring Mosquitto, testing the setup with Mosquitto client tools validates that the broker can handle publish/subscribe operations effectively. By running Mosquitto on a Raspberry Pi, developers can create a scalable, reliable communication infrastructure that facilitates real-time data transmission and processing, integral for applications like remote sensing, home automation, and other IoT projects. This setup not only ensures efficient data handling but also supports secure and manageable device interactions within the network.

- **Paho MQTT**

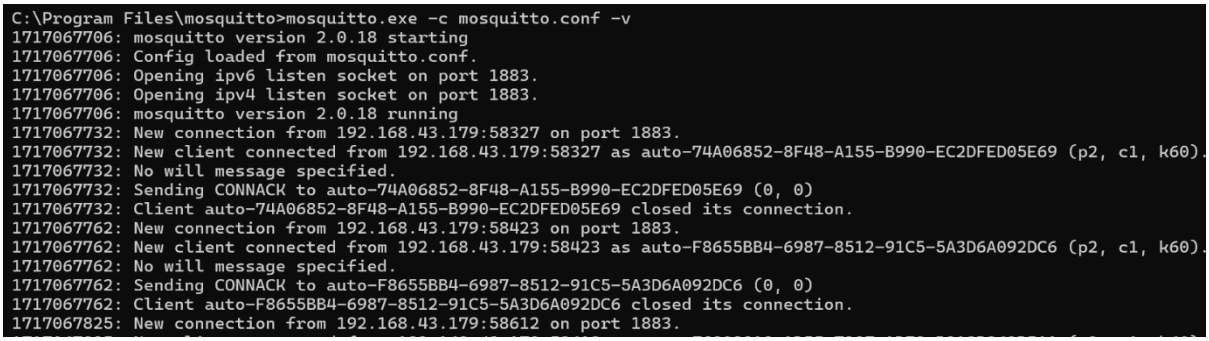


Figure III. 24: mosquitto configuration

The Raspberry Pi, due to its relatively powerful hardware and full Linux operating system, is well-suited to act as both an MQTT client and a data processing unit in an IoT setup. This part of the system will subscribe to MQTT topics, receive data transmitted by devices like the Raspberry Pi Pico, and then process or store this data for further actions such as analysis, visualization, or triggering responses.

Figure III. 25: Mosquitto Publish/Subscribe

III.4.1.2.3 Setting Up the MQTT Client

To receive data, the Raspberry Pi will use the Paho MQTT client, a popular MQTT library that provides a flexible way to handle incoming MQTT messages.

We install Paho-MQTT using the pip command in the command prompt: pip install paho-mqtt.

Key Components:

1. MQTT Client Setup: Initialize the client and connect it to the MQTT broker.
2. Subscription to Topics: Subscribe to the MQTT topics that the Raspberry Pi Pico is publishing to.
3. Message Handling: Define callback functions to handle incoming messages.

4. Reconnection Logic: Ensure the client can handle disconnections and reconnect automatically.

III.3.4 User-interface

In our project, we utilize tkinter and customtkinter as GUI frameworks. Tkinter serves as a Python binding to the basic tkinter framework, whereas customtkinter offers a modern approach. Both frameworks provide the necessary tools and capabilities to create intuitive and visually attractive graphical interfaces for our system.

Tkinter:Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets that allow developers to build interactive applications with buttons, labels, entry fields, menus, and more.

- **Python Tkinter Widgets:**

In the following sections, we cover basics of Tkinter, widget,etc

A GUI program uses a variety of controls, including buttons, labels, scrollbars, radio buttons, and text boxes.

Tkinter refers to these small GUI components or controls as widgets

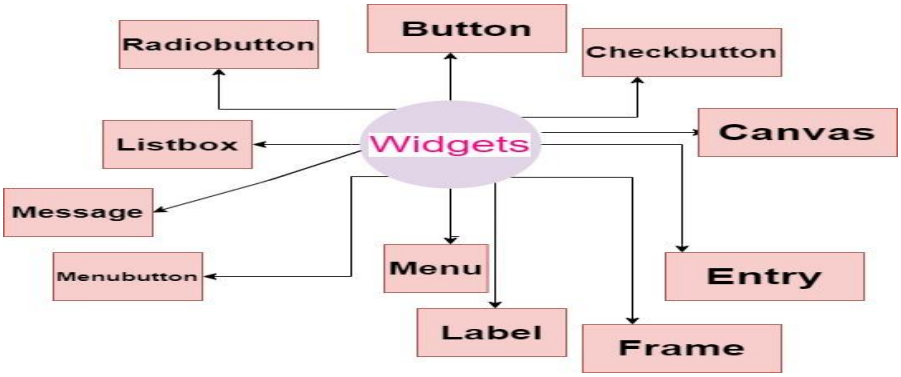


Figure III. 26: Python Tkinter Widgets

[21]There are 19 widgets available in Python's Tkinter package. All of the widgets are given here, along with their basic descriptions:

List of widgets	description
button	If you want to add a button in your application, then the **Button widget** will be used. A button is clickable, and the user can click the button widget to perform any action
canvas	To draw a complex layout and pictures (like graphics, text, etc.) the **Canvas Widget** can be used.
checkButton	If you want to display a few options as checkboxes, then the **Check button widget** can be used. It allows you to select multiple options at a time.
Entry	To display a **single-line text field** that accepts values from the user **Entry widget** can be used.

Frame	In order to group and organize other widgets, the Frame widget can be used. Basically, it acts as **a container that holds other widgets**
Label	To Provide a **single-line caption** to another widget Label widget can be used. It can **contain images** too.
Listbox	To provide a user with a list of options the **List box widget** can be used
Menu	To provide commands to the user **Menu widget** can be used. Basically, these **commands are inside the Menu button**. This widget mainly creates **all kinds of Menus** required in the application.
Menubuton	The **Menu button widget** is used to display the menu items to the user
Message	The **Message widget** mainly displays a message box to the user. Basically, it is a **multi-line text which is non-editable**
Radionbutton	If you want the number of options to be displayed as radio buttons, then the **Radio Button widget** can be used. You can select one at a time.
Scale	**Scale widget** is mainly a **graphical slider** that allows you to select values from the scale.
Scrollbar	To scroll the window up and down the **Scrollbar widget** in Python can be used.
Text	The **Text widget** mainly provides a **multi-line text field** to the user where users enter or edit the text and it is different from Entry.
toplevel	The **Toplevel widget** is mainly used to provide us with a separate window contain
SpinBox	The **SpinBox** acts as an entry to the "**Entry widget**" in which value can be input just by **selecting a fixed value of numbers**.
PanedWindows	The PanedWindows is also a container widget that is mainly used to handle different panes
LabelFrame	The label Frame widget is also a container widget used to mainly handle complex widgets.
MessageBox	The message box widget is mainly used to display message in desktop apllication

Table III: widgets of Tkinter

- **CustomTkinter:**

CustomTkinter is a python UI-library based on Tkinter, which provides new, modern and fully customizable widgets. They are created and used like normal Tkinter widgets and can also be used in combination with normal Tkinter elements. The widgets and the window colors either adapt to the system appearance or the manually set mode ('light', 'dark'), and all CustomTkinter widgets and windows support HighDPI scaling (Windows, macOS). With CustomTkinter you'll get a consistent and modern look across all desktop platforms (Windows, macOS, Linux).[22]

- **List of basic widgets**

List of widgets	Description
CTkCheckBox	A checkbox that can be toggled on and off.
CTkComboBox	A combination of a drop-down list and an entry widget.
CTkButton	A customizable button widget.
CTkEntry	An entry widget that allows the user to input a single line of text.
CTkLabel	A widget used to display text or images.
CTkOptionMenu	A menu that allows the user to select a value from a list.
CTkSwitch	A switch that can be toggled on and off.
CTkTabview	Creates a tab view, like a notebook in Tkinter.
CTkSrollbar	A scrollbar for scrolling widgets like CTkTextbox.
CTkTextbox	A textbox that is scrollable in vertical and horizontal directions.
CTkFrame	A container for other widgets.
CTkSegmentedButton	Aset of options where only one option can be selected at a time.
CTkSlider	A slider widget for selecting a value from a range.
CTkProgressBar	A widget that shows the progress of a task.
CTkRadioButton	A set of options where only one option can be selected at a time
CTkScrollableFrame	A frame that can be scrolled vertically and/or horizontally

Table IV: widgets of CustumTkinter

➢ **Benefits of Using Both Frameworks:**

- **Leveraging tkinter's foundation:** You gain access to the core functionalities and vast library of widgets offered by tkinter.
- **Enhanced aesthetics with customtkinter:** You can create modern and visually attractive user interfaces that elevate the user experience.
- **Simplified development:** Customtkinter streamlines development by providing pre-built widgets with a modern look and feel, saving you time and effort compared to creating custom styles from scratch.

III.5 Data Processing and Decision Logic

In this section, we will explain how the received data from the waste bin is processed to determine the bin's status and the decision-making rules that trigger alerts and notifications for operators. The focus will be on weight calculations, bin status determination, and the logic for making decisions based on the data.

Data Processing Steps	Description
Weight Calculations	Initial and ongoing measurements to monitor fill levels, assessing weight to estimate bin status.
Bin Status Determination	Categorization of bins based on their calculated weight as empty, half-full, or full.
Decision-making for Operators	Rules-based triggers for actions: alerts for nearly full bins, notifications for scheduled pickups or issues.

Table V: Description of Data Processing

III.5.1 Weight Calculations

The data processing begins with calculating the weight of the waste in the bin. This involves converting the raw data from the load cell into meaningful weight measurements.

- **Raw Data Conversion:**
  - The HX711 load cell amplifier sends raw digital values to the Raspberry Pi Pico.
  - These raw values need to be converted into weight measurements using the calibration factor obtained during the calibration process. The formula for conversion is Weight (kg) = Raw Value / Calibration Factor is used to convert the raw digital values obtained from the HX711 load cell amplifier into meaningful weight measurements in kilograms (kg).
- **Explanation:**
  - Weight (kg): This represents the calculated weight of the waste in the bin, expressed in kilograms (kg).
  - Raw Value: This refers to the unprocessed digital data received from the HX711 load cell amplifier. It is typically a numerical value that needs to be converted into a weight measurement.
  - Calibration Factor: This is a crucial parameter that accounts for the sensitivity of the load cell and ensures accurate weight calculations. It is determined during the calibration process, where a known weight is placed on the load cell, and the corresponding raw value is recorded. The calibration factor is then calculated by dividing the known weight by the raw value.

Example: If a 10 kg weight gives a raw output of 800,000 from the HX711, the calibration factor would be 10 / 800,000 = 0.0000125 kg per unit.

- **Applying the Formula:**
  - Obtain Raw Value: Acquire the raw digital value from the HX711 load cell amplifier. This value might be represented in various formats depending on the specific communication protocol and hardware setup.

- Retrieve Calibration Factor: Ensure you have access to the calibration factor obtained during the calibration process. This value should be stored securely and easily accessible for calculations.
- Perform Calculation: Substitute the raw value and calibration factor into the formula:

$$\text{Weight (kg)} = \text{Raw Value} / \text{Calibration Factor}$$

- Interpret Result: The resulting value represents the weight of the waste in the bin, expressed in kilograms (kg).

This formula is essential for converting raw load cell data into meaningful weight measurements. It enables the system to accurately assess the weight of the waste and make informed decisions regarding waste management. The calibration factor plays a critical role in ensuring the accuracy of these calculations.

III.5.2 Bin Status Determination

Once the weight is calculated, the next step is to determine the status of the waste bin. This involves comparing the calculated weight against predefined thresholds.[23]

Threshold Definitions: We need to define clear weight thresholds for the different bin statuses. Here's what we're thinking:

- - Empty: Weight < 1 kg
- - Partially Full: Weight between 1 kg and 15 kg
- - Full: Weight 20 kg or higher

Of course, these thresholds can be adjusted depending on the actual bin capacity and the typical weight of the waste it'll be holding. We will want to fine-tune them once we have some real-world data.

Status Evaluation: To determine a bin's status, we will simply compare its calculated weight against these predetermined thresholds. Whichever threshold range the weight falls into, that'll be the status we assign to that bin.

We will make sure to update the status accordingly whenever new weight data comes in. No sense working with stale info!

Status Reporting: Once a bin's status is updated, we will package up that status and send it over to the Raspberry Pi for further processing. We can use that same MQTT protocol we're already using to transmit the raw weight data.

III.5.3 Decision-Making Rules for Operators

When it comes to managing these bin statuses, we need a solid set of rules to make decisions and alert our operators. Here's how We will handle it:

Alert Thresholds:

We will define clear conditions for when alerts should be triggered, like:

- Immediate alert if a bin is 100% full

- Warning notification if a bin is partially full, so we can plan for its collection

Notification Mechanisms:

There are a few different ways we can notify operators about bin statuses:

- Visual indicators like LEDs right on the bins themselves
- Audible alarms or buzzers to grab attention
- Remote notifications via email, text message, or a dedicated mobile app

Decision Logic:

We will develop a robust decision logic framework that spells out exactly what actions to take based on a bin's status.

Automated Actions:

If our waste collection is automated, we can configure the system to trigger those collection actions automatically when a bin hits full status. Like firing off a message to the collection service.

Data Logging & Analysis:

Every weight measurement and bin status will be logged for later analysis. This data will help optimize our collection schedules and make the whole waste management process more efficient over time. We can analyze patterns in the data and adjust our thresholds or decision logic if needed.

Scheduling Optimization:

Speaking of optimized scheduling, we will integrate all this bin data into a management system that can plot out the most efficient collection routes based on current statuses.

Notifications:

For real-time updates, we will use MQTT or a similar protocol to push bin status data to operators or a central management dashboard as it comes in.

The goal is to create a streamlined system that alerts the right people at the right time, automates actions, when possible, optimizes scheduling, and just gets smarter through continuous data analysis. Efficient waste management made easy!

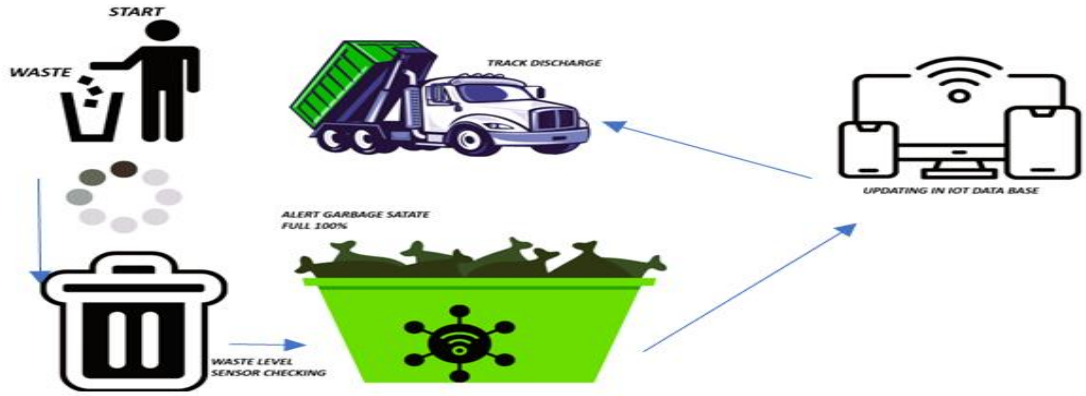


Figure III. 27: Efficient waste management

III.6 User Interface Design

The user interface (UI) for the smart waste management system is crucial for operators to monitor the system effectively. This section covers the design considerations for the operator interface, focusing on real-time bin status display, historical data visualization, and alerts and notifications. The interface is created using Tkinter and CustomTkinter.

III.6.1 Real-time Bin Status Display

A crucial aspect of the user interface is to provide operators with a real-time display of the waste bin status. This involves integrating sensor data from the bins to continuously monitor their fill levels. The user interface should be intuitive and visually appealing, using color-coded indicators (green for less than 50% empty, yellow for 50-70% nearly full, and red for above 100% full) to convey the bin status at a glance. This feature helps operators make quick decisions regarding waste collection routes and prioritize bins that need immediate attention.

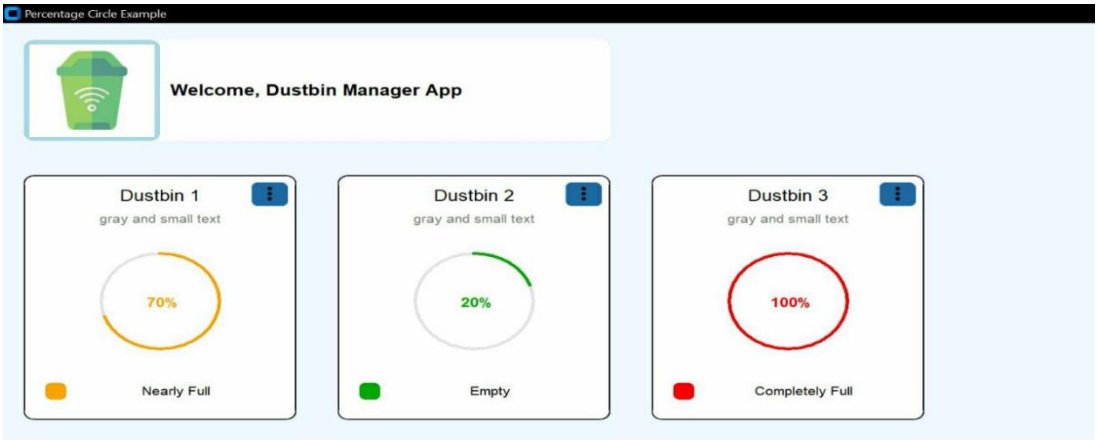


Figure III. 28: User Interface Design Considerations for our Smart Waste Management System

III.6.2 Historical Data Visualization

Visualizing historical data is essential for analyzing waste generation patterns over time. This can be achieved by utilizing data visualization tools like Pandas in Python to read CSV (Comma-Separated Values) files containing historical sensor data from the bins. Graphs and charts can display trends such as daily, weekly, or monthly waste levels, allowing operators to identify peak times and optimize collection schedules accordingly. The user interface should provide easy access to these visualizations, with options to filter data by date range and bin location.

- **Pandas**:

Pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.[24] To install pandas just open cmd and type: pip install pandas [25]

With Pandas module up and running, you can import your data into a Data Frame or Series and use Pandas' extensive functionality to manipulate, clean, and analyze that data. Key features and functions of Pandas include:

1. **Data cleaning**
   Pandas offers various functions for cleaning and transforming your data, such as filling in missing values, dropping columns or rows, deleting NULL values and renaming columns.
2. **Data filtering and selection**
   Pandas allow for a range of fine filtering and selection functions, based on highly granular conditions. So, no matter how complex the data is, you can extract the exact information you want.
3. **Data aggregation**
   With Pandas, you can perform aggregation operations like groupby, pivot, and merge to summarize and restructure your data.
4. **Data visualization**
5. Pandas integrates with the popular data visualization library, Matplotlib, allowing you to create various types of plots and charts from your data.

- **The benefits of learning Pandas library:**

Why should you use Pandas? There are several reasons to use Pandas for data analysis and manipulation, including but not limited to:

- Efficient data handling

Pandas provides a functional framework for handling large datasets with ease. The library is built on top of NumPy, which ensures fast and efficient numerical operations.

- Flexibility

Pandas offers an arsenal of functions and methods for data manipulation, and it is a flexible tool for all sorts of data scientist and manager tasks.

- Easy integration with other libraries

Pandas integrate seamlessly with popular Python libraries like NumPy, SciPy, and Matplotlib, creating powerful pipelines for data analytics.

- Wide adoption and support

Pandas are widely used in the data science community, so you'll find ample resources, tutorials, and support through online forums.

- Readability

The Pandas package has a clear and concise syntax, so it is easy to read and understand. This readability makes your code easier to append and maintain, driving smooth collaboration with others and longevity for your projects.

- Handling diverse data sources

Once you install Pandas and start importing data from diverse sources, Pandas let you efficiently process that data.

This includes reading and writing data sources such as CSV files, Excel files, and SQL databases. This versatility makes Pandas libraries a popular solution through a range of fields, where data comes in diverse sets and formats.

- **Matplotlib:**

Matplotlib is a fundamental Python library for creating static, animated, and interactive visualizations. It offers a comprehensive set of tools and functionalities that cater to various plotting needs across different scientific disciplines and data analysis workflows.

To install Matplotlib using pip command in the cmd :

pip install matplotlib

To plot with Matplotlib, we must import its Pyplot module using the command below:

use import matplotlib.pyplot as plt

We will discuss 10 Matplotlib advantages that will inform your hiring:[26]

1. Matplotlib provides a simple way to access large amounts of data

With Matplotlib, developers can create accurate plots based on huge amounts of data. When analyzing these data plots, good developers will make it easier to see patterns and trends in the data sets. Thus, Matplotlib simplifies data, making it more accessible.

2. It is flexible and supports various forms of data representation

As noted above Matplotlib supports data representation in bar charts, graphs, scattered plots, and other forms of visualization. This flexibility means that it can adapt effectively to your company's needs.

3. It is easy to navigate

The Matplotlib platform isn't too complex. Hence, both beginners and advanced developers can apply their programming skills to the platform, producing professional results. Matplotlib also has subplots that further facilitate the creation and comparison of data sets.

4. It ensures accessibility by providing high-quality images

Since the main goal of Matplotlib is to provide a way to access and display data, its plots and images must be of high quality. To meet this requirement, Matplotlib provides high-quality images in various formats, such as PDF, PGF, and PNG.

5. It is a powerful tool with numerous applications

Matplotlib's data-visualization qualities can be used in various forms, such as Python scripts, shells, web application servers, and Jupyter notebooks. As such, its operations are versatile.

6. It is useful in creating advanced visualizations

Matplotlib is primarily a 2D plotting library. However, it includes extensions that developers can apply to create advanced 3D plots for data visualization. In this way, the platform ensures that working with data is easier and more productive.

7. It is open source, saving you cash

An open-source platform requires no paid license. Because Matplotlib is free to use, you save the extra cost you usually incur when producing data visualizations.

8. It is extensive and customizable

The Matplotlib platform can fit any of your company's needs because it includes many types of graphs, features, and configuration settings. Experienced developers can tweak its features to suit particular objectives and projects.

9. It can run on different platforms

Matplotlib is platform independent. This means it can run smoothly no matter what platform you use. Whether your developers use Windows, Mac OS, or Linux, you can expect high-quality results.

10. It makes data analysis easier

Due to its numerous features, plot styles, and high-quality results, Matplotlib makes data analysis easier and more efficient. It also helps save the time and resources you would have spent analyzing large datasets.

Unlike other data-visualization platforms, Matplotlib in Python only requires a few lines of code to generate a plot for data sets.

III.6.3 Alerts and Notifications

To ensure timely waste management, the system should be capable of sending alerts and notifications. These can be triggered when bins reach a certain fill level or when unusual patterns are detected (e.g., a bin filling up faster than usual). Notifications can be sent via multiple channels, such as email, SMS, or push notifications on mobile devices. The user interface should include a configuration panel where operators can set thresholds for alerts and choose their preferred notification methods.

- **Implementing the User Interface:**
  - Dashboard Design: Create a central dashboard that provides an overview of all bins, displaying real-time status, historical data visualizations, and alert settings in one place. This ensures that operators have all the necessary information at their fingertips for effective decision-making.
  - Interactive Maps: Utilize interactive maps to visualize the geographic locations of the bins with their real-time statuses. This helps operators visualize the spatial distribution of waste levels and plan efficient collection routes.
  - Responsive Design: Ensure the user interface is accessible on various devices, including desktops, tablets, and smartphones. This allows operators to monitor and manage the waste management system remotely, improving operational efficiency.
  - Data Export and Reporting: Provide options to export data and generate reports. This feature can aid in regulatory compliance and presenting performance metrics to stakeholders, promoting transparency and accountability.

By integrating these design considerations, the user interface for the smart waste management system will be comprehensive, user-friendly, and effective in optimizing waste

collection operations. It will provide operators with the necessary tools and information to make informed decisions, streamline processes, and promote sustainable waste management practices.

III.7 Conclusion

This chapter has laid a strong foundation for the development of the IoT-based smart waste management system. By exploring the hardware and software components required, it has provided a comprehensive understanding of the core elements that make up the system.

The detailed coverage of the Raspberry Pi Pico W's specifications and its integration with the HX711 load cell has equipped us with a solid grasp of the hardware implementation. The ability to accurately measure and monitor the weight of the waste in real-time is a crucial aspect of the system, and the chapter has thoroughly explained the load cell's functionality and the calibration process.

The discussion on the MQTT protocol and its utilization for efficient data communication between devices and the central server is particularly insightful. The chapter's explanations on the publish-subscribe model, the role of the MQTT broker, and the implementation of the MQTT client on the Raspberry Pi are all instrumental in ensuring reliable data transmission within the system.

Furthermore, the chapter's focus on the user interface design considerations, such as the real-time bin status display, historical data visualization, and the integration of alerts and notifications, demonstrates a well-rounded approach to the system's usability and user experience. The leveraging of Python libraries, including Tkinter, CustomTkinter, and Pandas, further enhances the system's capabilities in data processing, visualization, and decision-making.

CHAPTER IV:
Testing and Validation

IV.1 Introduction

In this chapter, we will thoroughly detail the testing procedures conducted to validate the functionality and reliability of the waste management system developed using Raspberry Pi Pico with HX711 load cell interface and MQTT communication protocol.

IV.2 Testing and Validation

How does our system work?

The RPi Pico W acts as the publisher device which collects data from the load cell using the HX711 driver. Using the MQTT protocol, RPi Pico W publishes the weight data from the load cell to an MQTT broker.

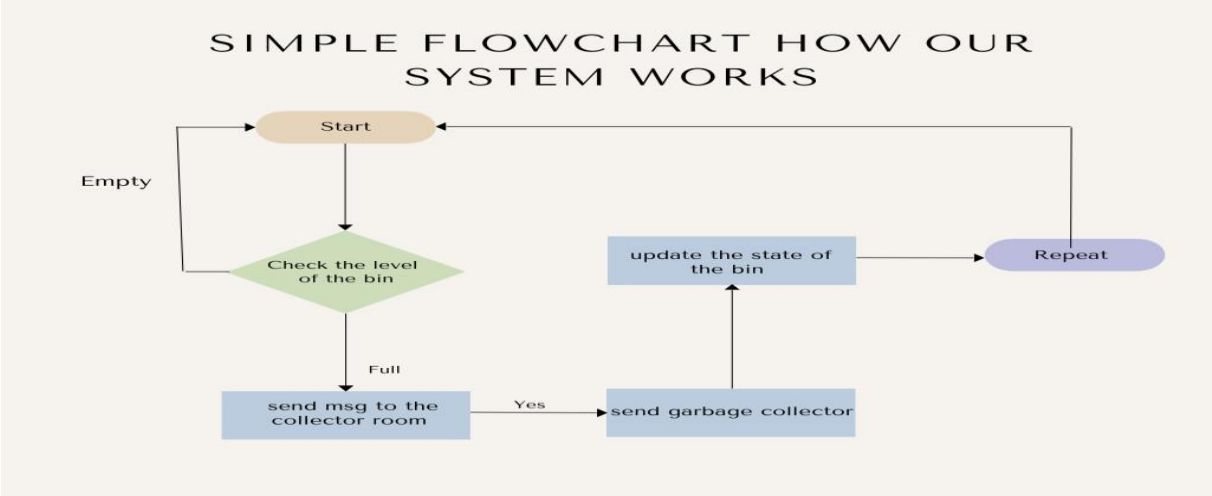
The RPi PI acts as the subscriber device which receives the weight data from the MQTT broker. Based on the received weight on the RPi, a custom Tkinter interface shows the bin status.

The interface displays a circular percentage that rotates in color according to the bin's fill level.

The MQTT protocol allows communication between the publisher (Raspberry Pi Pico W) and subscriber (Raspberry Pi) devices, allowing weight data from the load cell to be delivered to the interface.

The Pi Pico W acts as a lightweight IoT device, sending weight data to the MQTT broker, while the RPi is a more powerful device, receiving data and providing bin status on the custom Tkinter interface.

This system takes advantage of the Internet of Things concept by utilizing the Raspberry Pi Pico W as a low-cost, low-power device for collecting sensor data and publishing it to an MQTT broker. With its enhanced computational ability, the RPi subscribes to the MQTT broker, receives data, and indicates the bin status in a user-friendly interface.



- **Load Cell Accuracy Testing:**

We evaluated the precision of our load cell weight measurements by testing with known calibrated weights across different weight ranges and bin fill levels. The load cell readings were

compared against the true weights, and we verified that the measurements fell within acceptable tolerance limits.

Figure IV. 1: tare of the known weight

We compared the measured values against the expected values to assess the precision and consistency of the load cell readings. This testing phase aims to ensure the reliability and accuracy of the weight data collected by the system.

Figure IV. 2: Electronic scale

For optimal stability and reliability in voltage supply, we used an external 3.3V power supply instead of relying on the internal regulator of the Raspberry Pi Pico w. This approach is particularly beneficial for delicate components like the HX711 load cell, which is used in precision measurement scenarios such as our smart garbage system. A steady power source not only boosts the accuracy of measurements but also safeguards our microcontroller and sensors from potential damage caused by voltage fluctuations.

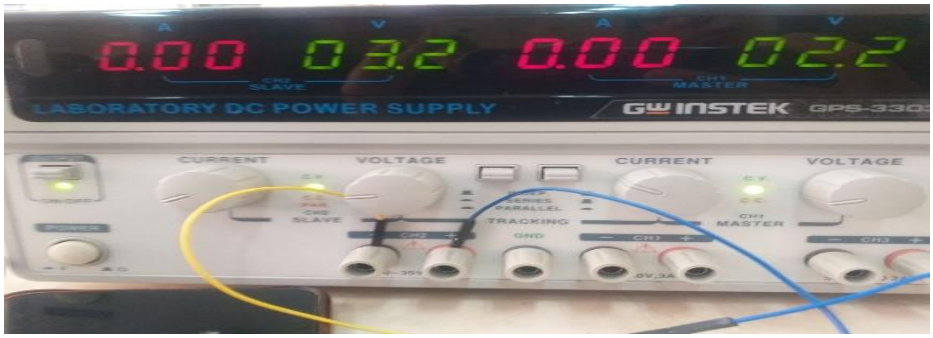


Figure IV. 3: Testing with 3.3v used Power Supply

- **MQTT Communication Reliability:**

How Reliable is Our Messaging System?

We tested if data could be sent and received properly using MQTT and how well our messaging system (MQTT) works under different network conditions. We sent messages from a Raspberry Pi Pico W to another Raspberry Pi and measured how many messages arrived successfully. We also checked to make sure the weight values weren't changed during transmission.

- **Bin Status Accuracy Validation:**

We filled the bins to different levels and We checked if the system correctly updated the bin status (empty, partly full, full). Here, we rigorously how validated the accuracy of our bin status indicators to ensure the system could reliably determine if a bin was empty, partially full, or completely full. This involved two key testing methods:

1. Physical Inspection Comparisons: We compared the percentage fill levels reported by the system against physical inspections of the actual bin contents. By checking the system's readings against real-world conditions, we could verify the bin status algorithm was correctly interpreting the sensor data.
2. Waste Composition Testing: We also tested the system's performance with different types and densities of waste materials. This allowed us to confirm that the bin status algorithm could accurately account for variations in waste composition that might impact how fast a bin appears to be filling up based solely on weight measurements. From light, fluffy materials to dense, compact waste, we aimed to cover a wide range of real-world scenarios.

Throughout both testing methods, we simulated different fill levels in a controlled environment, filling the bins to precise levels and monitoring if the system triggered the appropriate status updates. We transmitted bin status data via MQTT and validated that the user interface accurately always reflected the reported status. This rigorous testing validated the integrity and reliability of the bin status information provided to operators through the system.

- **User Interface:**

The GUI of our IoT-based system is specifically designed to efficiently manage waste collection by monitoring bin statuses and weight metrics. The main interface shows real-time updates on different bins, indicating how full they are with labels like "Nearly Full," "Completely Full," "Empty," and specific percentages like "23%". Each bin has an ID and location for easy identification, and you can get more details by clicking the button in the top right corner of the bin's dashboard card. This helps quickly identify which bins need attention and plan the waste collection routes.

The system also includes a data analysis section that uses a line graph to show the weight of the waste over time, allowing users to see trends in waste accumulation. This data helps predict future waste production and plan resource allocation. Through rigorous testing and validation procedures, we ensure that our smart waste bin system meets the required standards for accuracy, reliability, and usability, providing waste management teams with a robust and effective solution for optimizing their operations.

The user can see the following features and elements in this interface:

- Dustbin ID: to know which dustbin is being monitored
- Location of the dustbin this will be future work
- Current Fill Level: indicating if it is empty, nearly full, or full

If there are many dustbins, you can easily know which dustbin is empty, nearly full, or full. This interface allows users to view and analyze data in both graph and table formats (CSV file).

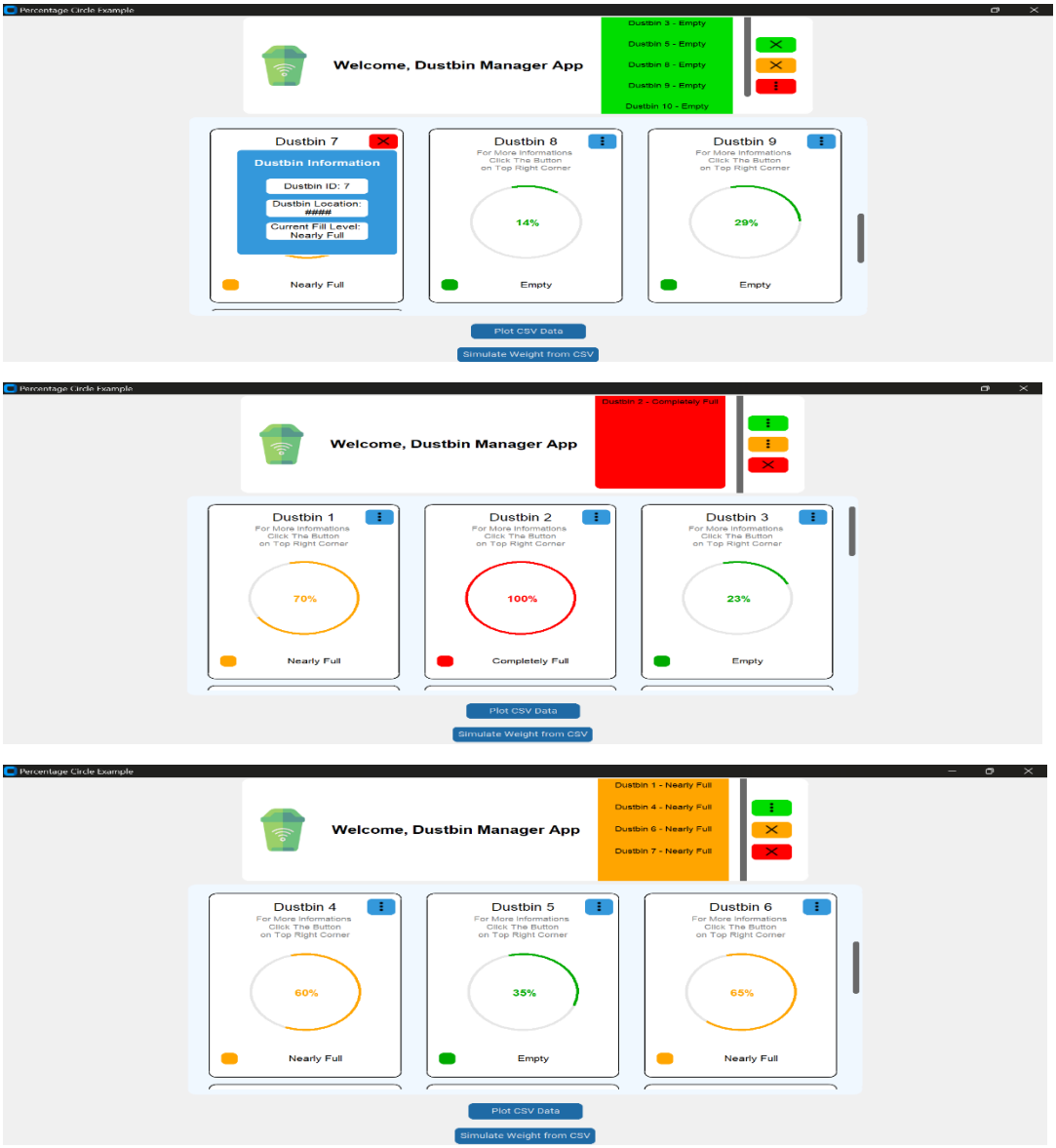


Figure IV. 4: GUI's of Dustbin's when is empty or full, nearly full

This interface allows the user to plot the data points in a table. The user can update the data displayed on this table through user selection on the columns date, time, and weight. It also allows a user to plot a specific range of dates and times by variable for the X-axis and by variable for the Y-axis and step size. The interface allows the user to search within the data display. It can allow someone to export data as a .csv file and enable plotting. It makes data visualization, analysis, and customization easier to handle and understand; therefore, this interface is valuable in providing insights that help make informed decisions.

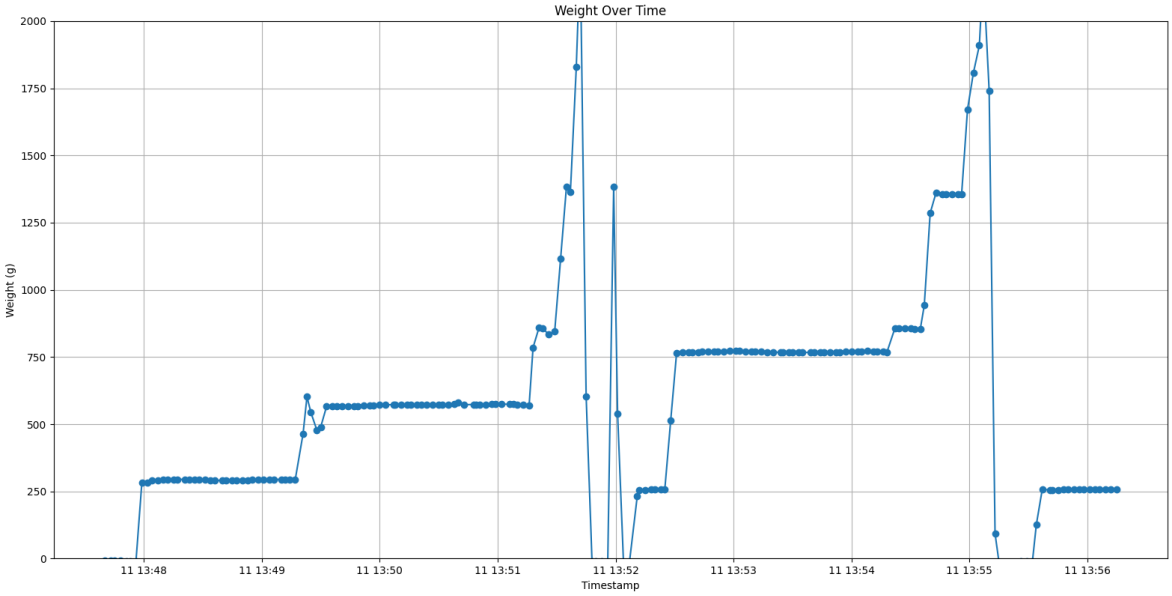


Figure IV. 5: plot value over time

IV.3 Results and Discussion

In this section, we will present the findings derived from the testing procedures outlined in the previous section. And how our smart garbage monitoring:

1. Accuracy of Weight Measurements:

The results of the load cell accuracy testing will reveal the system's ability to consistently provide precise weight measurements. We will analyze the deviations between the measured values and the expected values to determine the overall accuracy of the weight data.

Our testing:

First test: Without weight

Figure IV. 6: First test

Second test:

We checked if the weights measured by the load cells matched the real weights measured by the balance electronic

```
weight value:  86.11582
weight value:  86.29585
weight value:  86.54602
weight value:  86.66315
weight value:  86.84264
weight value:  86.88795
weight value:  86.8491
weight value:  86.77059
weight value:  86.85743
weight value:  86.97192
weight value:  86.94027
```

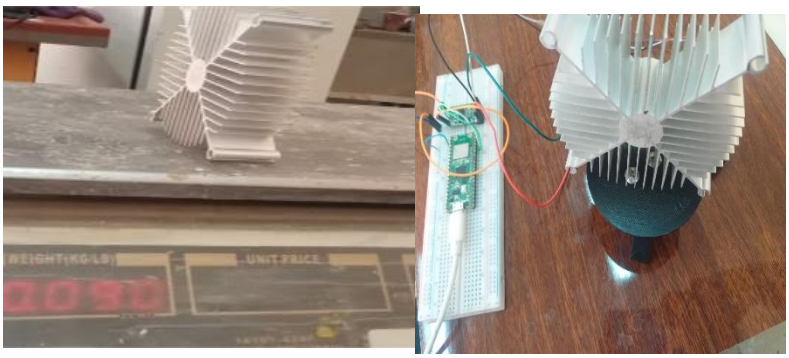


Figure IV. 7: Second test

Third test:

We tested the load cells by putting known weights on them

Here we put a known weight of 500g with 3.3v

```
weight value:  498.3771
weight value:  498.4699
weight value:  498.3569
weight value:  498.1885
weight value:  498.0163
weight value:  497.9769
weight value:  497.9038
weight value:  498.1003
weight value:  498.2207
weight value:  498.2746
weight value:  498.3447
weight value:  498.2621
weight value:  498.3448
weight value:  498.2312
```

Figure IV. 8: Third test

From our testing, here is what we found:

The load cell readings were very accurate compared to known weight

2. Effectiveness of the Bin Status Indicator:

The validation of the bin status indicator will demonstrate the system's reliability in accurately detecting the fill level of the waste bin and promptly updating the status information displayed in the user interface.

The bin status indicator promptly updated the status information displayed on the user interface, ensuring real-time visibility into bin conditions:

The figure below shows that when the garbage bin is at a low level, it means that the amount of garbage in the bin is minimal.

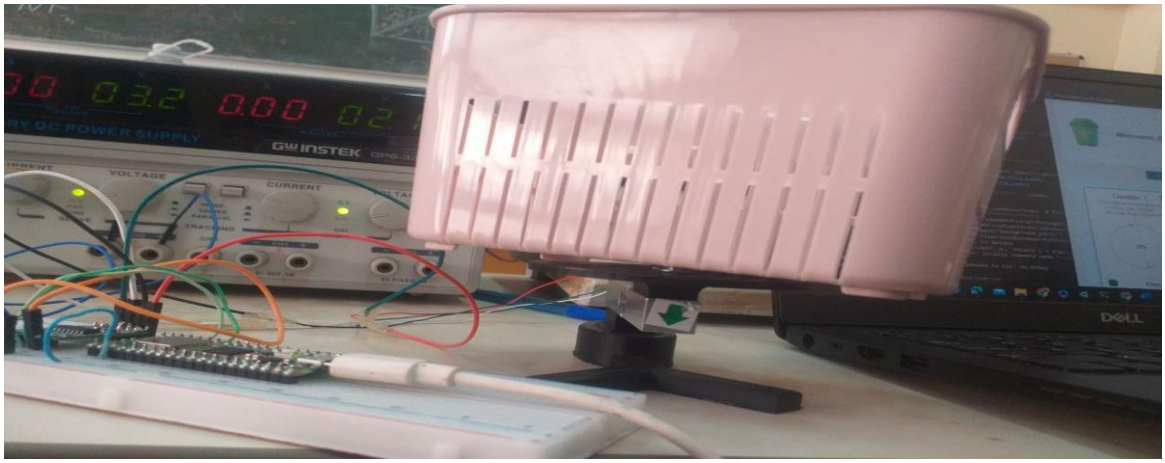


Figure IV. *9:bin when low level*

The figure below indicates that the garbage bin level falls between 25% and 50% of the bin's depth.

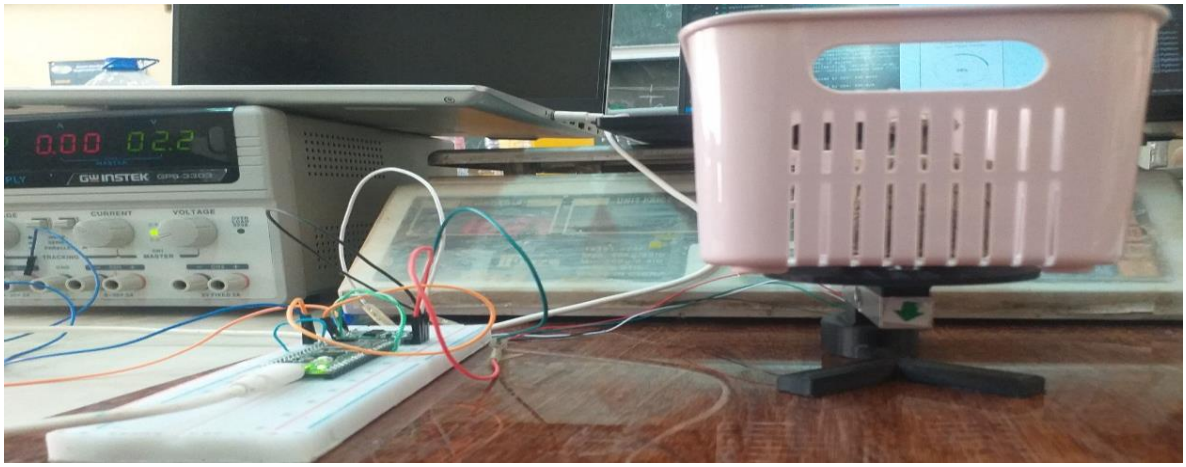


Figure IV. *10:bin when level falls*

The figure below shows that the garbage bin level is at full.

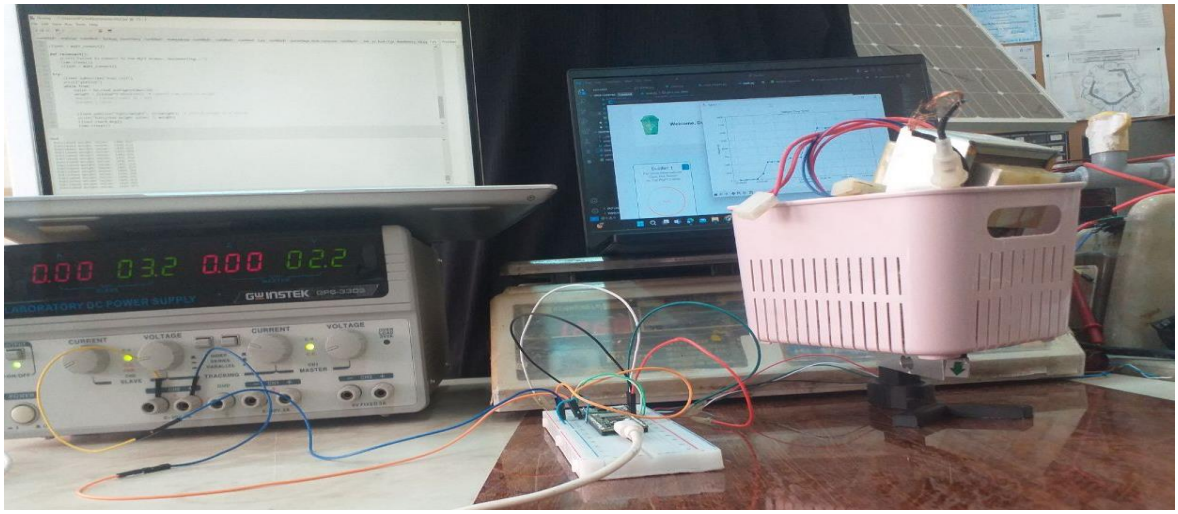


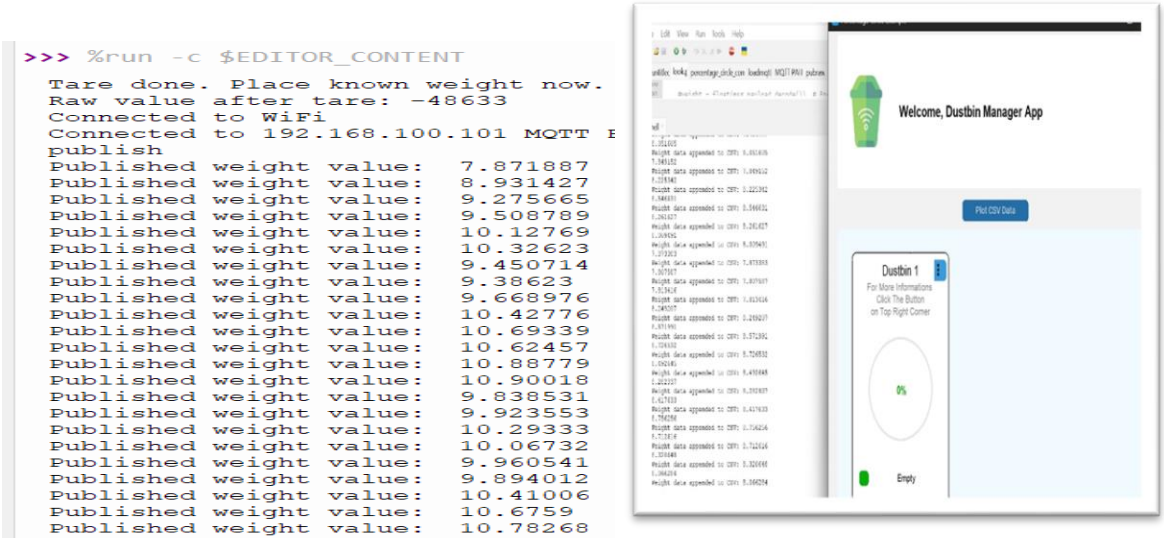
Figure IV. *11:bin when full level*

The system correctly determined and displayed the bin fill status and confirmed the reliability in accurately detecting the fill level of waste bins.

3. **Interface:**

The graphical illustrations in the report depict how the user interface visually represents the bin fill level status using color-coded circular indicators:

- When the bin is at a low level (between 0% and 50% full), a green circular indicator is displayed.



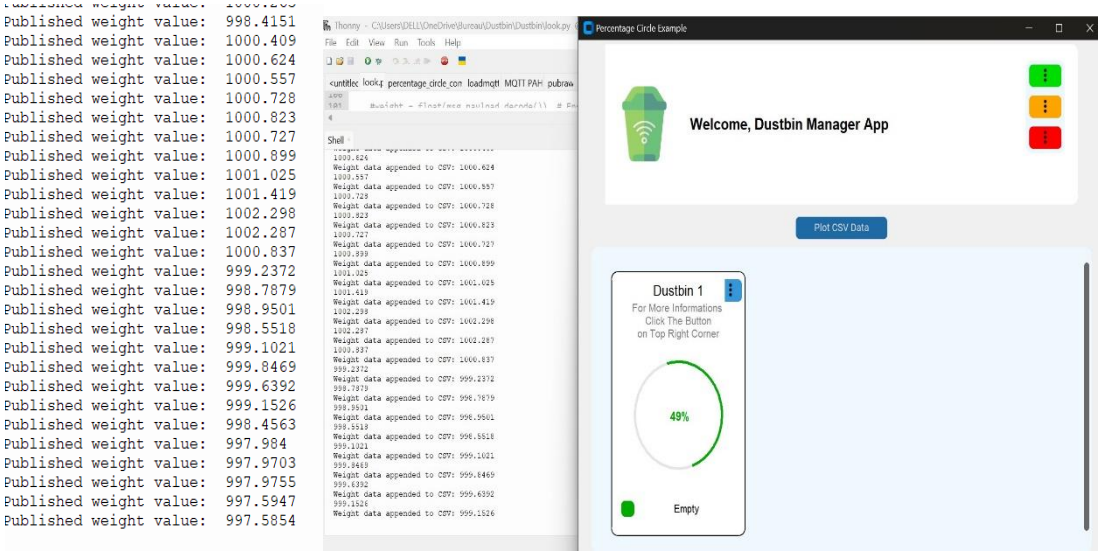


Figure IV. *12: result of bin is at low level*

- As the bin fills up to a moderate level (between 50% and 98% full), an orange circular indicator appears, indicating it is nearly full.

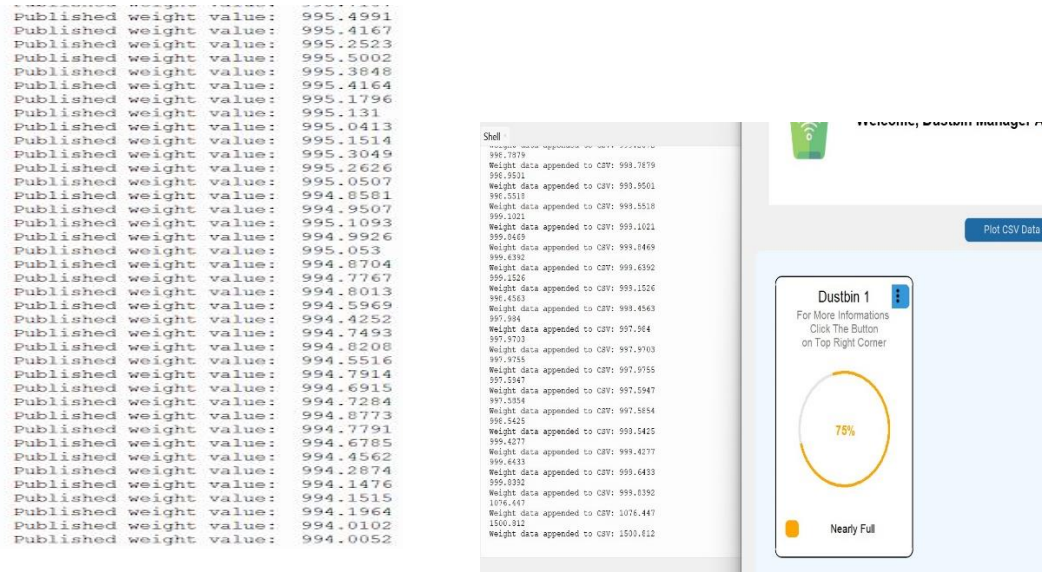


Figure IV. *13: result of bin is at nearly full*

- Once the bin reaches 100% capacity, a red circular indicator is shown, alerting users that the bin is completely full.

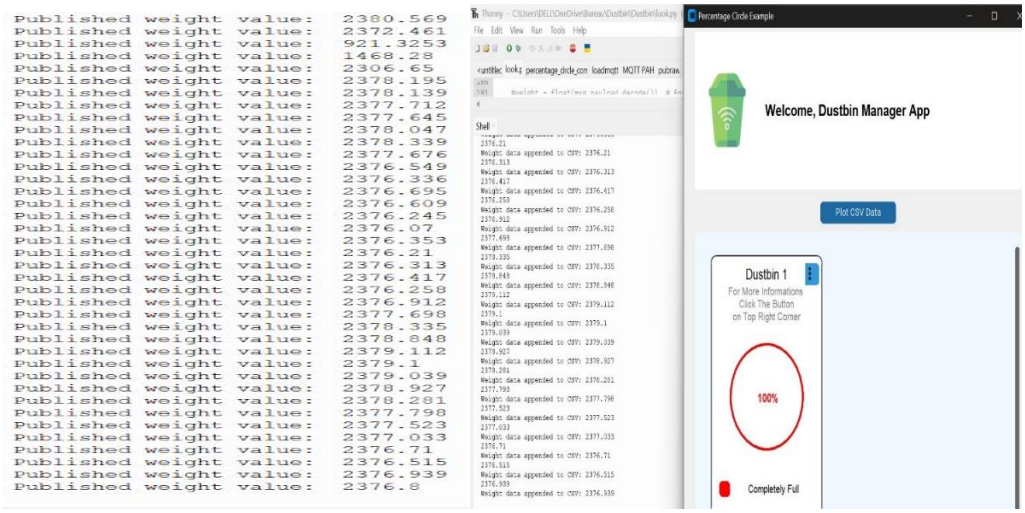


Figure IV. *14: result of bin is at full*

CSV Data Interaction: Users can import weight data from CSV files to simulate and visualize weight changes directly in the GUI, enhancing predictive analysis and resource planning.

	1 Timestamp	2 Weight		1 Timestamp	2 Weight
98	11 June 2024 13:52:01	538.8146	149	11 June 2024 13:54:18	768.0825
99	11 June 2024 13:52:04	-14.4534	150	11 June 2024 13:54:22	855.9575
100	11 June 2024 13:52:07	-14.80728	151	11 June 2024 13:54:24	856.0858
101	11 June 2024 13:52:11	232.2783	152	11 June 2024 13:54:27	855.7375
102	11 June 2024 13:52:12	254.9093	153	11 June 2024 13:54:30	855.6924
103	11 June 2024 13:52:15	256.1449	154	11 June 2024 13:54:32	855.5251
104	11 June 2024 13:52:18	257.3105	155	11 June 2024 13:54:35	855.3883
105	11 June 2024 13:52:20	257.0261	156	11 June 2024 13:54:37	942.4882
106	11 June 2024 13:52:23	256.6266	157	11 June 2024 13:54:40	1285.821
107	11 June 2024 13:52:25	256.4664	158	11 June 2024 13:54:43	1361.651
108	11 June 2024 13:52:28	512.7541	159	11 June 2024 13:54:46	1356.884
109	11 June 2024 13:52:31	764.2675	160	11 June 2024 13:54:48	1356.775
110	11 June 2024 13:52:34	767.8	161	11 June 2024 13:54:51	1357.05
111	11 June 2024 13:52:37	768.2461	162	11 June 2024 13:54:54	1356.741
112	11 June 2024 13:52:39	768.679	163	11 June 2024 13:54:56	1356.869
113	11 June 2024 13:52:42	768.8557	164	11 June 2024 13:54:59	1671.655
114	11 June 2024 13:52:44	769.3054	165	11 June 2024 13:55:02	1808.04
115	11 June 2024 13:52:47	769.1593	166	11 June 2024 13:55:05	1909.077
116	11 June 2024 13:52:50	769.5935	167	11 June 2024 13:55:07	2183.611

Figure IV. *15: result of csv data*

Graphical Displays:

The interface includes a line graph showing the weight trends over time, allowing users to analyze patterns in waste accumulation.

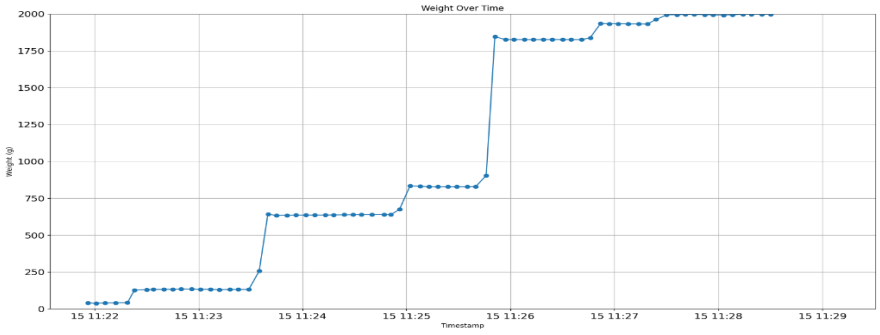


Figure IV. *16: result of graph weight over time*

• Overall Performance: The full system operated reliably during testing with minimal issues

IV.4 Conclusion and Future Work

Our smart bin monitoring system passed all the tests! We successfully:

- Accurately measured bin weights
- Correctly displayed bin fill levels
- Enabled reliable data communication
- Designed an intuitive user interface

However, there are some limitations and areas we can still improve:

- Optimize MQTT communication further

- In the proposed model, we connected a single dustbin to MQTT to gather data. Next, we will plan to connect all the dustbins together.
- Enhance user interface based on feedback to evaluate the usability and intuitive design we will conduct user trials involving waste management personnel. These trials involve realistic usage scenarios, and we will gather feedback on aspects such as data visualization, interaction flows, and overall user experience.

This testing will help identify areas for improvement and ensure the user interface meets the operators' needs.

- Add predictive maintenance features
- will add ultra-sonic sensor to detect level or distance
- Implement optimized collection route planning

With continuous improvement, our smart bin system can revolutionize waste operations!

In conclusion, we can say the testing and validation procedures conducted in this chapter affirmed to us the functionality and reliability of the waste management system developed. However, we will also identify any limitations or areas for improvement, such as the need for further optimization of MQTT communication for enhanced reliability in challenging network conditions.

- **Future work:**

we hope to upgrade the system with advanced capabilities like:

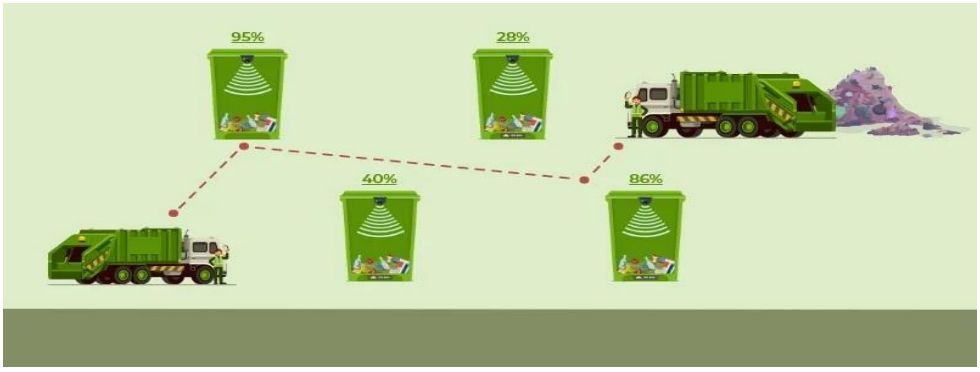


Figure IV. *17:* smart garbage with iot

- Using AI to predict when bins will fill up
- Automatically planning the most efficient collection routes
- Integrating with third-party systems and services

And in the next phase of our project, we plan to introduce exciting enhancements that will significantly improve the efficiency and effectiveness of waste management. One of our key developments is the integration of cameras for monitoring improper disposal, smell sensors for environmental odor detection, and moisture sensors for assessing bin conditions This technology will streamline the sorting process right at the source, drastically reducing the need for manual labor and making waste management more efficient.

Additionally, we envision an advanced automated system capable of collecting and sorting waste1 around the bins. This robotic mechanism will detect litter, pick it up, and ensure it is placed in the correct bin, whether dry or wet. This automation will not only enhance the precision of waste segregation but also help maintain cleaner surroundings by addressing waste that may not directly reach the bins.

We aim to create a system that not only meets the current needs but also adapts to future demands, ensuring a cleaner and more efficient approach to handling waste.

General Conclusion

In conclusion, our smart garbage management system with IoT demonstrates the potential of advanced technology in revolutionizing waste handling and collection processes. By integrating load cell sensors, our project enables real-time monitoring of garbage levels within bins across the city. This system provides municipalities with timely updates on bin status - whether empty, partially full, or full - facilitating more efficient waste collection routes and schedules.

A key feature of our system is its ability to accurately measure garbage levels and promptly relay this information to both the community and municipal authorities. This enables drivers to be informed precisely when and where collection is needed, thereby optimizing resource allocation and minimizing unnecessary trips. The incorporation of motion detection via load cells identifying when objects are being placed in already full bins.

Importantly, our system's interface and software have been designed with adaptability in mind. They can be easily modified and redeveloped to meet the specific requirements of different city municipalities, ensuring that the solution can be tailored to diverse urban contexts. This flexibility, coupled with the potential for further research and development, promises to continually boost the system's efficiency and performance.

The IoT backbone of our smart garbage management system facilitates seamless communication between bins, central servers, and end-users. By leveraging protocols such as MQTT, we ensure reliable data transmission, enabling real-time decision making and responsive waste management strategies.

While our current implementation shows considerable promise in enhancing urban waste management, we recognize the potential for future enhancements. These could include the integration of cameras for monitoring improper disposal, smell sensors for environmental odor detection, and moisture sensors for assessing bin conditions. Such additions would further increase the system's comprehensiveness and effectiveness.

Overall, this smart garbage management system with IoT offers a forward-thinking solution to age-old urban challenges. It empowers municipalities with the tools to make informed decisions, optimize resource utilization, and promote cleaner, more sustainable city environments. As we continue to refine and expand this technology, we move closer to realizing truly smart, efficient, and environmentally conscious cities.

Abstract

Waste management problems worsen with rapid population growth, leading to health issues due to unsanitary conditions. "Smart Waste Management" utilizing IoT is a key solution. It involves embedding electronics in public bins for real-time monitoring of waste levels. This data optimizes garbage collection routes, reducing fuel costs. Load sensors improve waste data collection efficiency. Analyzing this data enables governments and authorities to enhance intelligent waste management strategies.

الملخص

مع الزيادة السريعة في عدد السكان، تتفاقم المشاكل المتعلقة إدارة النفايات بشكل كبير. هذا التدهور يخلق ظروفًا غير صحية للسكان المجاورين، مما يؤدي إلى انتشار الأمراض المعدية والأمراض. لمعالجة هذه المشكلة، تُعتبر إدارة النفايات الذكية المستندة إلى إنترنت الأشياء (IoT) الحل الأكثر فعالية وشعبية. في النظام المقترح، سيتم تجهيز صناديق القمامة العامة بأجهزة مدمجة تُمكِّن من مراقبة مستويات النفايات في الوقت الفعلي. ستُستخدم البيانات المتعلقة بمستويات النفايات لتوفير مسارات مثلى لعربات جمع القمامة، مما سيقلل من تكاليف الوقود. ستزيد أجهزة استشعار الحمل من كفاءة البيانات المتعلقة بمستويات النفايات. سيساعد التحليل المستمر للبيانات المجمعة السلطات البلدية والحكومية على تحسين الخطط المتعلقة بإدارة النفايات الذكية.

Résumé

Les problèmes liés à la gestion des déchets deviennent de plus en plus graves en raison de la croissance démographique si rapide. Cela conduit au développement de maladies infectieuses et de troubles chez la population locale en créant des circonstances non sanitaires. La gestion intelligente des déchets basée sur l'IoT est l'approche la plus efficace et la plus utilisée pour résoudre ce problème. La solution proposée prévoit l'installation d'électroniques embarquées dans des poubelles publiques pour permettre la surveillance en temps réel des niveaux de déchets. Les itinéraires des camions de collecte des ordures seront ajustés en utilisant les données sur les niveaux de déchets, ce qui permettra d'économiser des dépenses en carburant. Les données sur les niveaux de déchets seront recueillies plus efficacement grâce aux capteurs de charge. Les gouvernements et les autorités locales pourront améliorer leurs stratégies de gestion intelligente des déchets grâce à l'étude continue des données recueillies.

Bibliography

Bibliography

[1] https://medium.com/centre-for-public-impact/what-gets-measured-gets-managed-its-wrong-and-drucker-never-said-it-fe95886d3df6

[2] B. Sigongan, J., P. Sinodlay, H., Xerxy P. Cuizon, S., S. Redondo, J., G. Macapulay, M., O. Bulahan-Undag, C., & Migan Vincent C. Gumonan, K. (2023). GULP: Solar-Powered Smart Garbage Segregation Bins with SMS Notification and Machine Learning Image Processing. [PDF]

[3]Sutjarittham, T. (2021). Modelling and Optimisation of Resource Usage in an IoT Enabled Smart Campus. [PDF]

[4] Ghahramani, M., Zhou, M., Molter, A., & Pilla, F. (2022). IoT-based Route Recommendation for an Intelligent Waste Management System. [PDF]

[5] Barbosa, D., Antón, S., Paulo Barraca, J., Bergano, M., C. M. Correia, A., Maia, D., & A. R. M. Ribeiro, V. (2020). Portuguese SKA White Book. [PDF]

[6] Zeng, F., Pang, C., & Tang, H. (2024). Sensors on Internet of Things Systems for the Sustainable Development of Smart Cities: A Systematic Literature Review. ncbi.nlm.nih.gov

[7] https://components101.com/sites/default/files/component_datasheet/pi-pico-w-datasheet.pdf

[8] Raspberry Pi Pico W Datasheet (components101.com)

[9] https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf

[10] https://manualzz.com/doc/17523942/datasheet-3134---micro-load-cell---0-20kg----czl635-contents

https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all

[11] https://diyprojectslab.com/arduino-with-load-cell-hx711-module/#google_vignette

[12] https://instrumentationtools.com/load-cell-working-principle/

https://instrumentationtools.com/load-cell-working-principle/

[13] MicroPython - Raspberry Pi Documentation

[14] Documentation for Visual Studio Code

[15] Getting Started with Raspberry Pi Pico using Thonny IDE (microcontrollerslab.com)

[16] Getting Started with Raspberry Pi Pico (and Pico W) | Random Nerd Tutorials

[17] https://github.com/robert-hh/hx711/blob/master/hx711_pio.py

[18] https://www.hivemq.com/blog/how-to-get-started-with-mqtt/#heading-introduction-to-mqtt

[19] https://www.eginnovations.com/documentation/Mosquitto-MQTT/What-is-Mosquitto-MQTT.htmvvvvv

[20] http://www.steves-internet-guide.com/mqtt-works/

[21] Python Tkinter Widgets - Studytonight

[22] Official Documentation And Tutorial | CustomTkinter (tomschimansky.com)

[23] https://tutorials-raspberrypi.com/digital-raspberry-pi-scale-weight-sensor-hx711/#google_vignette

[24] pandas · PyPI

[25] Introduction to Pandas in Python: Uses, Features & Benefits (learnenough.com)

[26] Top 10 advantages of Matplotlib in Python - TG (testgorilla.com)