Embedded System Entrepreneurship

Ajeeth Venkat Prasat (5588067),
Keerthana Mohan Raj (5581522)
Sheren Narayanan (5453401),
Yaswanthkumar Sirumugai Karunakaran (5581486)

Smart Garbage Can

by © IES Lab

Date: August 15, 2024
Supervisors: Prof. Dr. Oliver Amft, Mr. Mohammed Bourouah

Abstract

The Smart Garbage Can system revolutionizes waste management by integrating sensor technology and artificial intelligence. This innovative solution alleviates the challenges faced by facility managers, who oversee multiple locations with unpredictable municipal waste collection schedules. Employing an accelerometer sensor, the system captures acceleration data during the garbage collection process. This data is processed by an embedded machine learning model (TinyML) within the garbage can's controller to detect tilt motions indicative of the bin being emptied. Upon detection, real-time notifications are sent to the facility managers, enhancing operational efficiency and contributing to a cleaner and more sustainable environment.

Contents

1. Introduction

1.1. Problem Statement and Proposed Solution: An Introduction

Waste management is an essential component of modern urban environments and encompasses an array of tasks including the collection, transportation, and disposal of garbage. Facility managers, tasked with overseeing waste management across multiple buildings, face numerous challenges. Among these, keeping track of unpredictable municipal garbage collection schedules and ensuring timely retrieval of bins are particularly taxing. Inefficient practices not only burden facility managers but also contribute to environmental pollution.

In the midst of burgeoning technological advancements, integrating sensor technology and artificial intelligence into waste management practices presents an opportunity to address these issues. The advent of sensors, capable of measuring an array of physical properties such as pressure and acceleration, has transformed various sectors including automotive, aerospace, and consumer electronics. Accelerometers, for instance, are capable of detecting changes in velocity and position, while pressure sensors can monitor variations in altitude. These sensors have proven to be remarkably reliable even in adverse conditions, making them ideal candidates for various applications.

In the Smart Garbage Can project, the MPU6050 Sensor, is utilized to monitor the acceleration data of garbage bins. The MPU-6050 is an integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor.

Complementing the sensor board is the Pico controller, embedded with the powerful RP2040 microcontroller chip. This dual-core ARM Cortex-M0+ processor is capable of handling complex operations, and in this context, it processes the data from the MPU6050 sensor through a TinyML model.

The integration of these cutting-edge technologies in the Smart Garbage Can system streamlines waste management for facility managers. Through real-time monitoring of acceleration data and intelligent processing, the system can accurately detect when a garbage can has been emptied and subsequently notify the manager. This technological intervention promotes efficiency, reduces time spent on manual monitoring, and fosters a cleaner, greener urban environment.

1.2. Theoretical Foundations of the Sensor and Controller: A Detailed Analysis

In the Smart Garbage Can system, the core components include the MPU6050 sensor and the LiPo Pico with an RP2040 chip serving as the controller. This section elaborates on the technical foundations, specifications, and advantages of these components.

1.2.1. Sensor: MPU6050

The MPU6050 sensor, integral to the Smart Garbage Can system, is an always-on inertial module featuring a 3D digital accelerometer and a 3D digital gyroscope. It is instrumental in detecting the motion and orientation of the garbage can.

Specifications:

- Operating Voltage: 1.71 V to 3.6 V
- Accelerometer Full Scales: $\pm 2/\pm 4/\pm 8/\pm 16$ g
- Gyroscope Full Scales: $\pm 250/\pm 500/\pm 1000/\pm 2000$ dps
- Output Data Rates (ODR): up to 1 kHz
- Interface: I^2C

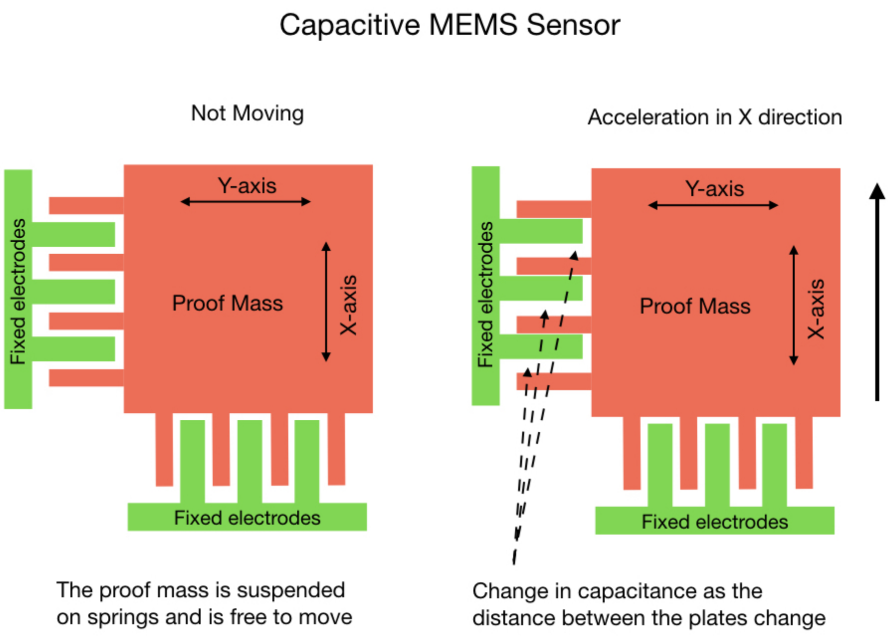

Capacitive MEMS Sensor
Not Moving
Acceleration in X direction
Y-axis
X-axis
Proof Mass
Fixed electrodes
Fixed electrodes
Y-axis
X-axis
Proof Mass
Fixed electrodes
Fixed electrodes
The proof mass is suspended on springs and is free to move
Change in capacitance as the distance between the plates change

Figure 1: MPU6050 Capacitive Sensing Unit

Source: The MPU6050 Explained

Working Principle:

The MPU6050 integrates a 3D accelerometer and a 3D gyroscope in a single package, enabling it to measure both linear acceleration and angular velocity. From Figure 12, it is clear that the sensor's micro-mechanical structures vary their capacitance based on acceleration or rotational motion, and these changes in capacitance are detected and processed to yield the measurements

1. Accelerometer:

The accelerometer portion of the MPU6050 measures linear acceleration along three axes (X, Y, and Z). Linear acceleration is essentially the rate of change of velocity with respect to time. This allows the device to detect various aspects of motion, such as whether the object is moving or at rest, or if it is tilted at a particular angle.

2. Gyroscope:

The gyroscope portion of the MPU6050 measures angular velocity, which is the rate of change of angular position over time, along three rotational axes. This helps in understanding how an object is rotating or turning.

3. Data Generation and Features:

The MPU6050 can process raw data from both the accelerometer and the gyroscope to compute motion-related data such as Significant Motion. A notable feature of MPU6050 is its interrupt generation capability. It can be configured to trigger interrupts when specific conditions are met, such as when a certain motion is detected. This is particularly useful in power optimization, as the system can be programmed to sleep mode and be activated only when an interrupt is generated. Additionally, MPU6050's embedded FIFO (First In, First Out) memory block allows data to be stored and batch-processed, which helps in reducing the communication load on the system and further optimizing power consumption.

4. Communication Interfaces:

The MPU6050 sensor can communicate data to a microcontroller or other devices through the I^{2}C interface. This facilitates easy integration into a variety of systems and applications.

Advantages:

- Low power consumption.
- High sensitivity and accuracy.
- Interrupt generation for motion-triggered actions.
- Robustness in various conditions.

1.2.2. Controller: LiPo Pico with RP2040 Chip

The LiPo Pico controller, enhanced with the RP2040 microcontroller, is a powerful processing unit integral to the Smart Garbage Can system. The Pico Lipo will be powered by Lipo Battery which will be interfaced along with Pico Lipo.

Specifications:

- Dual-core ARM Cortex-M0+ processor
- Clock speed: up to 133MHz
- 264KB of SRAM
- 16MB of on-board Flash memory
- 26 GPIO pins
- Interface: 2× UART, 2× I2C, 2× SPI, 3× 12-bit ADC, 16× controllable PWM channels

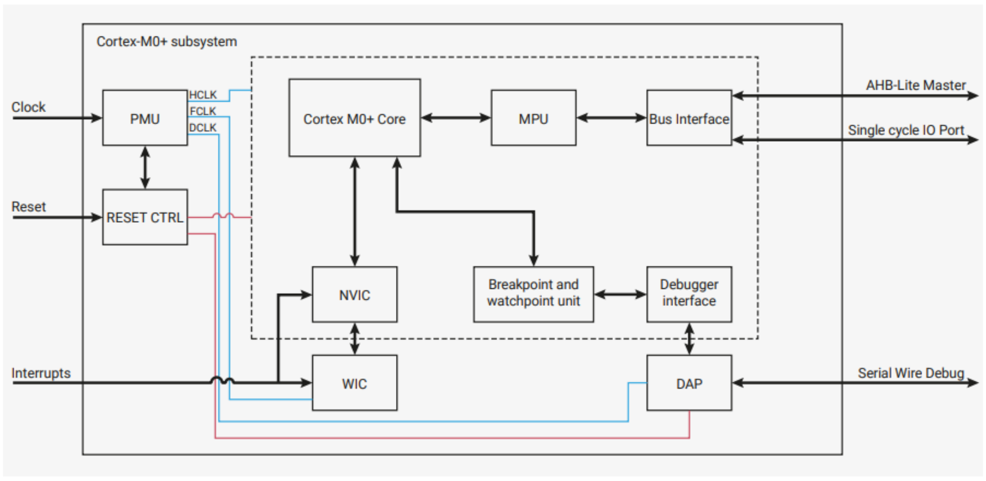


Figure 2: PICO Block Diagram

Source: RP2040 Datasheet

Working Principle: The LiPo Pico is built around the RP2040 chip, which boasts a dual-core ARM Cortex-M0+ processor. This processor is adept at handling multiple tasks simultaneously and executing complex algorithms. The chip contains an extensive 16MB Flash memory, which provides ample storage space for the program, libraries, and data.

The controller processes data from the MPU6050 sensor. With the sensor's interrupt signals, the LiPo Pico can efficiently manage power by being in sleep mode during inactivity and waking up when motion is detected. The controller executes the TinyML model, processes the data from the accelerometer to detect the tilt, and sends notifications to facility managers.

Advantages:
- Dual-core processor enables proficient multitasking.
- Extensive memory for elaborate applications.
- Versatile GPIO options for various interfacing needs.
- Efficient power management features.

Combining the MPU6050 sensor's capacitive accelerometer and MEMS gyroscope with the LiPo Pico's powerful RP2040 chip creates an advanced system capable of real-time monitoring and notifications, effectively optimizing waste management processes.

1.3. Leveraging Edge Impulse for Embedded Machine Learning: Unveiling the Potential of TinyML

In the era of IoT and smart systems, TinyML emerges as a game-changer, enabling machine learning algorithms to run on resource-constrained devices such as microcontrollers. It facilitates intelligent decision-making at the edge, reducing latency and enhancing efficiency. The Smart Garbage Can system harnesses the prowess of TinyML via the Edge Impulse platform to analyze and interpret sensor data in real-time.

Edge Impulse:
Edge Impulse is a prominent platform that democratizes and accelerates the development of TinyML models for edge devices. It offers a comprehensive set of tools for data collection, model training, and deployment, significantly streamlining the development process.

Utilization of Edge Impulse in the Smart Garbage Can System:
Data Collection: Through Edge Impulse, raw data from the MPU6050 sensor can be efficiently collected. This data, capturing the tilt and motion of garbage cans, is critical for training the machine learning model.

Model Training and Optimization: Edge Impulse's powerful training tools enable the creation of models that can interpret the sensor data. The platform also provides optimization features ensuring that the model is lightweight and efficient, an essential attribute for deployment on the LiPo Pico controller.

Deployment: Once the model is trained and optimized, Edge Impulse facilitates the deployment of this model onto the LiPo Pico. Here, the model continuously analyzes the accelerometer data.

Intelligent Analysis: The TinyML model deployed through Edge Impulse is capable of determining when a garbage can is emptied based on the tilt data from the accelerometer. Upon making this determination, it triggers the system to send a notification to the facility manager.

Scalability and Updates: As the system operates, new data can be fed back into Edge Impulse to retrain and update the model, allowing it to adapt to changes and improve its accuracy over time.

2. Related work

In recent years, the integration of Internet of Things (IoT) technology in waste management systems has emerged as a potent solution to the multifaceted challenges faced by this sector. Medvedev et al. [1] were among the pioneers in this integration, developing a smart waste bin system tailored for office environments. Utilizing sensors for automatic waste sorting and maintaining a record of waste disposal practices, their system encouraged environmentally responsible behavior among office workers. Interestingly, their system placed a special emphasis on waste sorting and had an educational component aimed at raising awareness about waste segregation.

In an urban context, Farah et al. [2] investigated how IoT technology could be harnessed to drive positive behavioral changes for improved waste management in smart cities. Their work emphasized the importance of data and feedback in motivating residents towards responsible waste disposal practices. They also tackled the challenges associated with the implementation of IoT-based waste management systems, and proposed strategies to overcome them, thereby recognizing the transformative potential of IoT in modernizing waste management. [1]

Adding a more pragmatic dimension, Vikram and Chaudhari [3] presented Smartbin, a novel system that utilizes sensors to monitor waste levels in bins and sends automated notifications to municipal services for collection when the bins are almost full. Their work highlighted how such an integration could foster cleaner and more sustainable urban environments by ensuring timely waste collection. It is noteworthy how they have aligned the technological deployment directly with municipal services.

Broadening the scope, Vermesan et al. [4] provided a comprehensive overview of IoT, discussing its architectures, applications, and the immense potential it holds in fostering smart city and smart factory concepts. What stands out in their work is the exhaustive exploration of IoT applications, and an analysis of the critical success factors and limitations of these technologies through various case studies.

Meanwhile, Kumar et al. [5] devised a smart garbage monitoring system, which integrated GSM and GPS systems with IoT. Their system alerts municipal authorities once garbage bins are almost full, ensuring timely collection. The combination of these technologies was posited as a way to streamline waste management strategies effectively. In doing so, their work showcased the feasibility and effectiveness of adopting a multi-technology approach for waste management.

Collectively, these studies elucidate the versatile applications of IoT in waste management, from fostering responsible waste disposal behaviors and integrating education, to practical deployment in synchronization with municipal services, and the benefits of multi-technology integration.

3. Technical Approach

In this project, an innovative and efficient solution was developed to alleviate the challenges faced by facility managers in monitoring and managing the waste disposal needs of multiple buildings. The core of this system is a retrofittable sensor circuit that is affixed to garbage cans for the purpose of detecting when they have been emptied. This sensor circuit is composed of a Pico LiPo microcontroller, known for its low power consumption and compact size, and MPU6050 acceleration sensor.

The MPU6050 accelerometer is particularly integral to the system as it is responsible for detecting the significant motion of the garbage cans. Rather than relying on an external central database, the collected data is stored in the flash memory of the Pico LiPo microcontroller, a design decision made to reduce complexity and reliance on continuous network connectivity.

To transform the raw sensor data into actionable insights, machine learning is employed. Edge Impulse, a leading platform for creating machine learning models for edge devices, is used for this purpose. The data stored in the flash memory is preprocessed to extract relevant features such as the degree of tilt. A TinyML model is then trained on this dataset using Edge Impulse, designed to discern the tilt characteristic of the garbage cans being emptied.

Upon successful training, the TinyML model is deployed back onto the Pico LiPo microcontroller. Herein lies the true innovation of this system - with the model deployed, the microcontroller can now analyze the accelerometer data in real-time and make instant decisions on whether the garbage can has been emptied.

This detection is not where the system's functionality ends. Once the bin-emptying event is detected, the microcontroller sends a real-time notification to the facility management. These notifications are not merely informational, as they play a vital role in enabling facility managers to optimize their waste bin collection schedules and routes.

Finally, to ensure that the system is robust and reliable in a real-world scenario, it is subjected to extensive testing across various garbage collection cycles. This involves monitoring the accuracy of the tilt detection and the timeliness of the notifications sent to facility management, and making any necessary refinements to the system.

In summary, through the intelligent integration of sensor technology, embedded systems, and machine learning, this project delivers a powerful tool for facility managers to upgrade their waste management practices and operations.

3.1. Sensor Circuit Design and Data Acquisition

3.1.1. Phase 1: Initial Design and Sensor Configuration

In the first phase of the project, an iterative approach was adopted for developing the sensor circuit. The initial step involved selecting a suitable microcontroller and sensor for preliminary testing. The Raspberry Pi Pico was selected as the microcontroller due to its robust capabilities and low power consumption. The MPU6050, a widely used accelerometer, was selected as the initial sensor for data collection.

To ensure future compatibility with the Edge Impulse machine learning platform, it was decided to program the Raspberry Pi Pico using the C programming language via the Arduino IDE, as opposed to the more conventional MicroPython.

Subsequently, focus was shifted towards configuring the MPU6050 sensor. Two crucial parameters - data sampling frequency and sensor sensitivity - needed to be defined. Following extensive research, it was decided to set the data sampling frequency at 100 Hz and sensor sensitivity at $\pm$8g.

Since the Arduino IDE does not provide direct support for the RP2040 (the microcontroller on the Raspberry Pi Pico), an extensive investigation was conducted to find a workaround. Two options surfaced: "Earle's Arduino Core for RP2040 Boards" and "Mbed-OS Core for RP2040". A thorough comparison revealed several differences between these cores, including support for EEPROM, availability of peripherals, GCC version, and support for Programmable I/O (PIO). Given that the project required both sets of peripherals and EEPROM functionality, Earle's Arduino Core was chosen.

Utilizing Earle's Arduino Core, the team was successful in developing and compiling C code for the Raspberry Pi Pico and implemented a simple program to read acceleration data from the MPU6050 sensor at 100 Hz with a sensitivity of 8g. To set the sensor in required configuration, the sample rate can be calculated as follows:

$$\text{Sample Rate} = \frac{\text{Accelerometer\_Output\_Rate}}{1 + \text{SMPLRT\_DIV}} \tag{1}$$

3.1.2. Phase 2: Motion-Triggered Interrupt, Power & Storage Management

During the second phase, three primary challenges were identified and addressed:

Problem 1: Power Management

For efficient power utilization, the sensor circuit needed to operate in low power mode when data collection was not required.

Solution:

Sleep mode was implemented in the sensor circuit. The microcontroller was configured to wake up from sleep mode only upon receiving an interrupt from the sensor, indicating motion.

Problem 2: Motion-Triggered Interrupt Generation

The MPU6050 sensor supports motion-triggered interrupt generation. The sensors configuration enabled us to activate the Significant motion Interrupt.

Solution:

The team identified MPU6050's Significant motion as a suitable solution. The MPU6050 was configured to generate an interrupt in case of significant motion.

The default value of Acceleration Sensor output Rate is 1000 Hz. To set the sensor at 100 Hz We will have to divide the Default rate with 10 as the sample rate divisor. Hence, we set the SMPRT_DIV Register with 0b00001010 (0x0A in Hex). Then in order to set the acceleration sensor to ±8g scale, the ACCEL_CONFIG Register is set with 0b00001000 (0x08 in Hex).

Finally, setting the interrupt can be achieved by configuring the INT_ENABLE register with 0b00111000 (0x38 in Hex) which enables the Interrupt generation via I2C communication. The MPU6050_MOT_THR register was set to 0b0110 0100 (0x64 in Hex) to set the threshold value for motion detection to ±100mg. Simillary, the MPU6050_MOT_DUR Register was set to 0b00010100 (0x14 in Hex) to set the duration threshold value for motion detection to be 20 ms. The Interrupt generated by MPU6050 was pulsed, hence a interrupt signal was generated for 50 μ s.

Problem 3: Data Storage

A storage medium was required to store sensor data for later retrieval which will be sued for training the ML Model.

Solution:

An external SD card compatible circuit called "AdaLogger" was used, which communicates with the Raspberry Pi Pico via SPI to store the data on the SD card.

In addition to addressing these challenges, a custom Arduino core was developed to implement sleep functionality, using Earle's Arduino Core as a foundation and incorporating functionalities from the pico-extras SDK. The core was custom built with the GCC compiler on a Linux system to include sleep functions.

In conclusion, this phase successfully achieved data collection through the MPU6050 sensor, efficient power management through sleep mode, and data storage on an SD card, paving the way for the subsequent development phases.

3.1.3. Phase 3: Data Handling

In the third phase of the project, a crucial problem was identified and addressed to improve the efficiency of data storage.

Problem 1: Efficient Data Storage

The project required sampling data at 100 Hz, meaning the system needed to store sensor data within 10 microseconds. However, writing 12 bytes of data (3-axis accelerometer output as floating-point numbers) to an SD card exceeded this time frame, resulting in possible data loss.

Solution:

As an alternative to SD card storage, it was decided to use the flash memory in the microcontroller. However, the Pico's 2MB of flash memory was not sufficient for storing large volumes of data. The team investigated alternative RP2040 boards with larger flash memory capacities and eventually selected the Lipo Pico, which comes with 16MB of flash storage and supports interfacing with a lithium battery.

This selection allowed for storing 12 bytes of sensor data in the flash memory within 3 to 5 microseconds, ensuring efficient data capture without loss and allowing a buffer of 5 microseconds for the next data read-write cycle.

Phase 3 succeeded in refining motion-triggered interrupts by employing the MPU6050's significant motion function, allowing for more precise motion detection. Additionally, by adopting a different RP2040 board with larger flash memory, the issue of data storage efficiency was resolved. These advancements collectively contributed to enhancing the reliability and accuracy of the sensor system. Future work can now focus on further optimizing the system and possibly integrating machine learning models for even more sophisticated motion analysis.

3.1.4. Phase 4: Finalizing Circuit Development

Phase 4 marked a pivotal transition from the prototyping stage to the final assembly of the sensory device. The focus shifted from programming to the physical development of the circuit. The objective was to fabricate a compact, reliable, and efficient device capable of real-time data collection from a garbage can.

Transition to Dot Board

After confirming that the prototype was functioning as expected, the team decided to create a more robust and permanent circuit on a dot board. A Schematic of the circut was developed. Finally, a box Enclosure was developed to interface the circuit to the garbage Can.

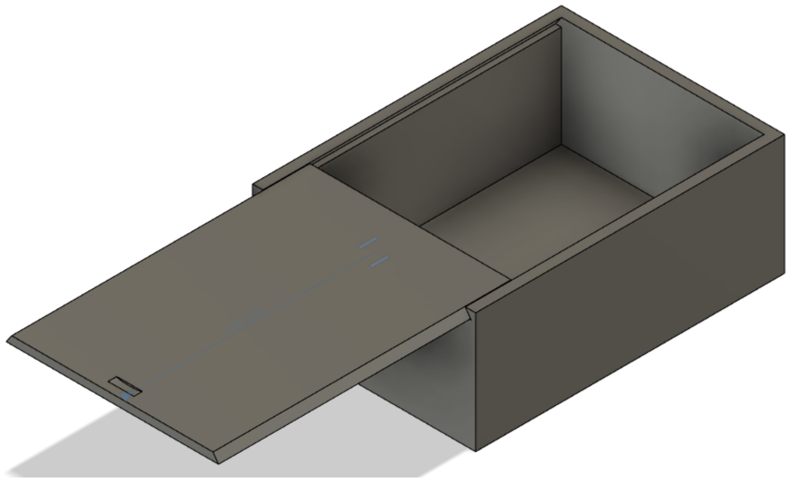


Figure 3: BOX Enclosure

Source: Autocad Fusion 360

Component Insertion and Testing

Upon completion of the soldering and conductivity check, the sensor and the Pico board were plugged into their respective sockets. The entire setup, integrating both hardware and software components, was tested to ensure it was functioning correctly.

Once the dot board assembly was verified to be functioning properly, it was attached to a garbage can for real-time data collection. The sensor's sensitivity and precision, combined with the efficient data storage system, enabled the device to accurately collect data on the garbage can's movements and tilts.

In Phase 4, the project progressed from the prototyping stage to the final assembly and implementation. Careful attention was paid to the physical construction of the circuit, ensuring that it was robust and reliable. The device's successful attachment to a garbage can demonstrated its practical application in real-world scenarios. Future steps could include scaling up production, further miniaturizing the circuit, improving power efficiency, and possibly integrating wireless communication for remote data access.

3.1.5. Phase 5: Data Acquisition

Data acquisition is the fundamental process that enables our Smart Garbage Can system to capture and store critical acceleration data during the garbage collection process. The operation of this process is facilitated by the meticulously developed sensor circuit, interfaced with the accelerometer sensor.

To start with, the sensor circuit is activated. Its principal function is to detect and register the acceleration during the garbage collection process, a pivotal parameter for our system. The accelerometer sensor mounted on the garbage can is responsive to tilt motions, capturing real-time data that will subsequently feed into the embedded machine learning model for further analysis.

The sensor's output data is converted from analog to digital, a transformation that enables it to be processed by the Pico LiPo's microcontroller. We opted for the Pico LiPo due to its superior capabilities in handling complex computations while maintaining low energy consumption.

After the data is successfully captured and converted, the next step is to store it in the Pico LiPo's flash memory. This process ensures that the crucial acceleration data is preserved, facilitating easy retrieval when needed.

As the primary data repository, the flash memory's role cannot be overstated. It stores the raw acceleration data, making it available for subsequent processes, including data preprocessing and the training of the TinyML model. Importantly, this memory unit has been specifically selected due to its high capacity and high-speed data access.

To capture a broad spectrum of data and enrich the learning capacity of our machine learning model, we conducted five types of motion tests with the garbage can:

Still: In this data collection, the garbage can was left still. The aim was to capture and quantify the 'rest' state, where no acceleration is expected.

Linear Motion: The can was moved strictly in the x and z directions, simulating the limited movements experienced during regular usage.

Trajectory: This data collection was the most indicative of the garbage can's real-world application. The can was moved in a trajectory mimicking that of being lifted by a garbage truck during the collection process.

Ramp Up: For this test, we positioned the garbage can on an inclined ramp to mimic a situation where the garbage can may have to be moved uphill. This scenario provides unique acceleration data due to the gravitational pull acting against the motion of the can.

Ramp Down: This test was designed to mimic the scenario where the garbage can is being moved downhill. The unique acceleration data in this case is due to the gravitational pull acting in concert with the motion of the can.

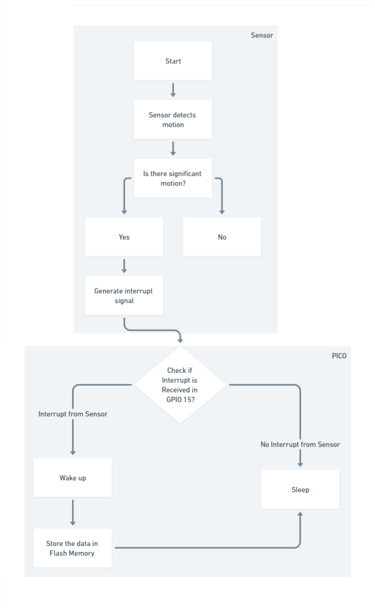

Sensor
Start
Sensor detects motion
Is there significant motion?
Yes
No
Generate interrupt signal
PICO
Check if Interrupt is Received in GPIO 15?
Interrupt from Sensor
No Interrupt from Sensor
Wake up
Sleep
Store the data in Flash Memory

Figure 4: Data Acquisition Software Flow Chart

Through the successful acquisition of data from the sensor circuit and its storage in the Pico LiPo's flash memory, we lay a robust foundation for our system's subsequent steps: data preprocessing and TinyML model training and testing. As such, this stage is instrumental in ensuring the effectiveness of our smart garbage can system in enhancing operational efficiency and contributing to a cleaner and more sustainable environment.

We plan to continue refining this process to maximize data accuracy, minimizing noise and outliers that could potentially impact the TinyML model's performance. Future improvements may also include optimizing the data acquisition process for power efficiency to extend the garbage can's operational longevity on a single charge.

In conclusion, our approach to data acquisition provides a practical, effective, and sustainable solution to the challenges faced by facility managers in handling waste management, aligning with our objective to revolutionize this sphere through the integration of sensor technology and artificial intelligence.

3.2. Machine Learning Model Development

The development of the machine learning model forms the core of the Smart Garbage Can system. This process involves several critical steps: data preparation and cleaning, feature extraction using Digital Signal Processing (DSP), model training, model testing, and model deployment. We leveraged Edge Impulse, an advanced development platform, to facilitate these tasks.

3.2.1. Data Preparation and Cleaning

Once our data was acquired, we commenced the crucial step of data preparation. Importing the recorded data into Edge Impulse, we started by labeling the data based on the five motion models: still, linear, and trajectory, ramp up and ramp down. This categorization allowed for a more streamlined and efficient training process for the machine learning model.

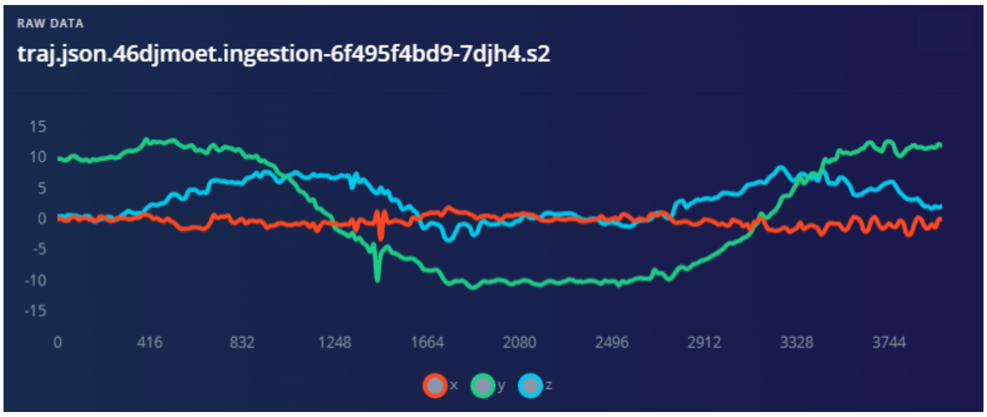


Figure 5: Plot of Trajectory Dataset

Source: Edge Impulse

Next, we performed data cleaning, eliminating known outliers and incorrect measurements that might skew the model's learning process. This stage ensured that the training and testing datasets consisted only of pertinent and accurate data. The cleaned data was then split into a training dataset, used to train the model, and a test dataset, employed later to evaluate the model's performance.

3.2.2. Digital Signal Processing (DSP)

Digital Signal Processing (DSP) is a critical step in the preprocessing of our sensor data. It serves to convert the raw, often noisy, sensor readings into more relevant and meaningful features that our machine learning model can learn from. This includes filtering out high-frequency noise and extracting specific patterns that correspond to our desired motion classifications. Through DSP, we enhance the model's capacity to make accurate predictions, optimize computational resources, and streamline our model training process, all contributing to the improved efficiency of our Smart Garbage Can system.

The selected data was then processed using spectral analysis in Edge Impulse. This approach was favored due to its ability to transform the time-domain sensor signals into the frequency domain. Spectral analysis excels in scenarios involving repetitive motion, such as our case, as it makes it easier to discern patterns by concentrating on the frequencies that hold the most information and reducing noise.

A low-pass filter with a cutoff frequency of 19Hz and an order of 6 was chosen. This selection aimed to filter out high-frequency noise while preserving the core signal components related to the bin's motion. The cutoff frequency and order were selected based on empirical testing and the specific characteristics of our dataset.

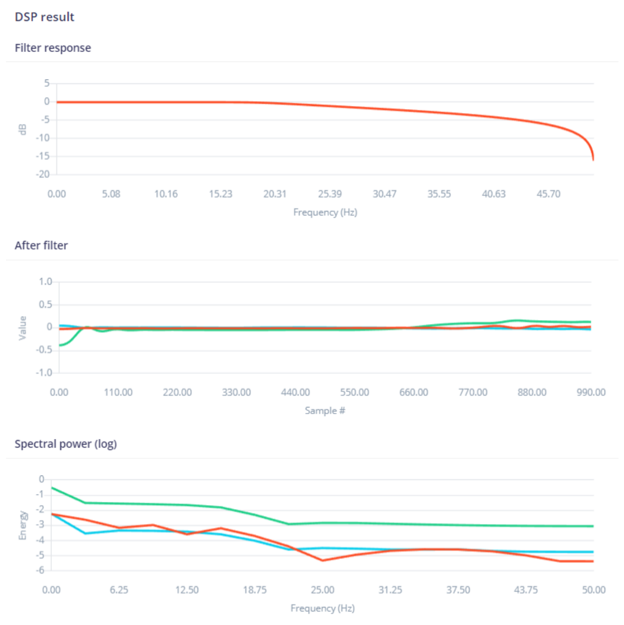


Figure 6: Result of DSP

Source: Edge Impulse

Additionally, we chose the Fast Fourier Transform (FFT) method for spectral analysis, with a FFT length of 32. This decision was driven by FFT's ability to quickly convert time-domain signals into frequency-domain signals, making the patterns in our repetitive motion data more visible. The choice of 32 as the FFT length allowed us to obtain a sufficient level of detail from our frequency analysis while managing computational complexity. Figure 6 shows us the results of the DSP.

3.2.3. Model Training

Following the data preparation and DSP setup, we proceeded to the model training phase. A classifier was chosen as the learning block due to its ability to effectively learn and discern patterns in data. In our case, the classifier was trained to identify patterns that correspond to the five motion models: still, linear, trajectory, ramp up & ramp down.

Our neural network classifier was designed with an input layer of 33 neurons, corresponding to the 32-point FFT output plus a bias term. This was followed by two dense layers, each containing 20 neurons. These hidden layers were tasked with learning complex features and interactions from the input data, facilitating the model's learning process.

The final output layer consisted of three neurons, corresponding to the three classes (still, trajectory, linear, ramp up & ramp down). This setup ensured that the model was able to distinctly recognize and categorize each type of motion.

For the learning rate, we opted for a value of 0.0005. The learning rate is a crucial parameter in the training process; it determines the size of the steps the model takes in the optimization process. If the learning rate is set too high, the model might overshoot the optimal point, while a low learning rate, although more precise, might make the training process too slow. A learning rate of 0.0005 was chosen as it struck a balance, allowing the model to learn effectively without compromising speed or accuracy.

For the different types of motion tests, the machine learning model demonstrated the following accuracies:

Linear Motion: In the linear motion test, where the can was moved strictly in the x and z directions, simulating the limited movements experienced during regular usage, the model achieved an accuracy of 100%. This perfect accuracy suggests that the model is exceptionally proficient at detecting and categorizing regular use motions.

Ramp Down: In the scenario where the garbage can was positioned on a downhill ramp, mimicking a situation where the garbage can is moved downhill and thus capturing unique acceleration data due to gravitational pull acting with the motion of the can, the model showed an accuracy of 77%. While relatively lower than the others, this accuracy still indicates the model's ability to reliably classify this type of motion.

Ramp Up: When testing the garbage can moving uphill on a ramp, simulating a scenario that provides unique acceleration data due to gravitational pull acting against the motion of the can, the model accuracy was found to be 0%. This score suggests that significant improvement is needed in the training of the model to accurately categorize this specific movement.

Still: For the 'rest' state, where the garbage can was left still and no acceleration was expected, the model achieved perfect accuracy at 100%. This result shows that the model is exceptionally efficient at recognizing when the can is not in motion.

Trajectory: In the trajectory motion test, where the can was moved in a trajectory mimicking that of being lifted by a garbage truck during the collection process, the model demonstrated an accuracy of 92%. This high accuracy reflects the model's strong ability to accurately categorize this real-world motion scenario, which is of significant importance in the intended use-case of our smart garbage can system.

These results underscore the potential of our machine learning model in differentiating various movements and states of the garbage can, aiding in the accurate detection and notification of garbage collection events. The differing accuracies across the various movements also highlight areas for future improvement, particularly for the 'Ramp Up' scenario. Over fitting of the model was checked and some generalization of model we performed to refine the model which lead to increased accuracies and more reliable garbage collection detection.

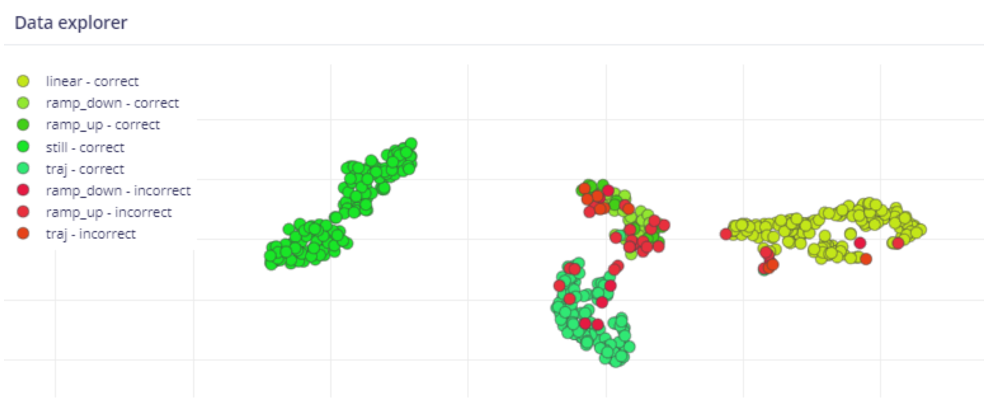


Figure 7: Classification Output of Trained Model

Source: Edge Impulse

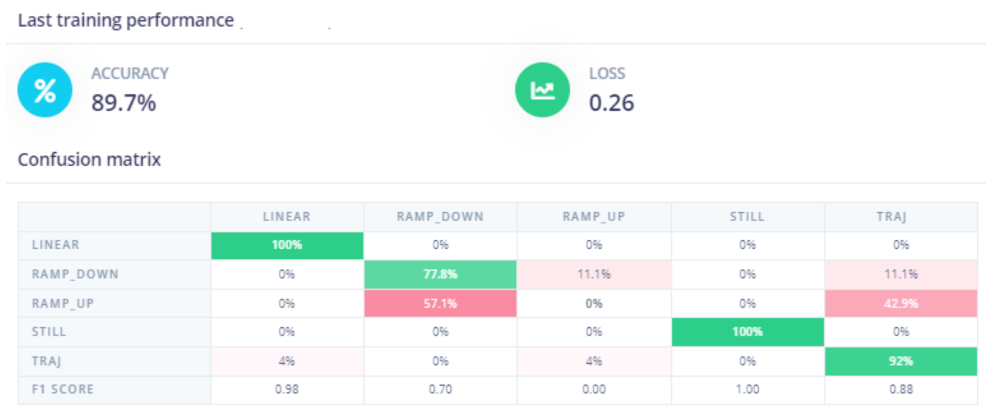
The high accuracy rates in trajectory scenario validate the model's capacity to efficiently learn and discern the patterns associated in detecting our preferred motion, providing a strong foundation for effective real-time motion detection in the subsequent deployment phase.

3.2.4. Model Testing

After the training process, the effectiveness of the model was put to the test. We utilized Edge Impulse's live classification feature to facilitate this crucial step. This tool allows for real-time testing of the model by feeding sensor data directly into Edge Impulse via a communication (COM) connection.

We simulated each of the five motion models—still, linear, trajectory, ramp up and ramp down—and provided the corresponding sensor data to the model. The objective was to evaluate the model's ability to correctly classify real-time data, mimicking the conditions the Smart Garbage Can system would encounter in a practical setting.

Remarkably, the model successfully classified each of the motion models accurately in all cases. This result not only validated the training phase but also demonstrated the model's robustness and reliability when faced with real-world data. It provided strong assurance of the model's capacity to distinguish between the different motions and perform consistently in operational conditions.



Last training performance

ACCURACY 89.7%

LOSS 0.26

Confusion matrix

	LINEAR	RAMP_DOWN	RAMP_UP	STILL	TRAJ
LINEAR	100%	0%	0%	0%	0%
RAMP_DOWN	0%	77.8%	11.1%	0%	11.1%
RAMP_UP	0%	57.1%	0%	0%	42.9%
STILL	0%	0%	0%	100%	0%
TRAJ	4%	0%	4%	0%	92%
F1 SCORE	0.98	0.70	0.00	1.00	0.88

Figure 8: Confusion Matrix of ML Model

Source: Edge Impulse

These successful test results cleared the way for the final phase of our machine learning model development: model deployment.

3.2.5. Model Deployment

Having successfully passed the testing phase, our TinyML model was ready for deployment. We selected the Arduino Library as our desired output format due to its compatibility with the Pico controller. This format was particularly suitable for our purpose as it allows the model to be seamlessly integrated into microcontroller projects programmed with the Arduino IDE.

Upon exporting the model as an Arduino Library, we integrated it into our Pico controller. This integration essentially transformed our Smart Garbage Can system into an intelligent device capable of processing raw sensor data, discerning between different motions, and sending real-time notifications to facility managers.

This process of model deployment marked the culmination of our machine learning model development, thus equipping our Smart Garbage Can system with the ability to transform raw sensor data into actionable insights. The deployed model, capable of identifying the distinct tilt motions indicative of the bin being emptied, is central to the system's enhanced operational efficiency. It offers real-time notifications to the facility managers, thereby contributing to a more effective and sustainable waste management system.

3.3. Hardware Integration and Machine Learning Model Deployment

The deployment circuit development and the integration of the ML model form the final step in the Smart Garbage Can project. It involves incorporating the ML model with a sleep functionality, integrating the NodeMCU with the Arduino Cloud IoT service, and the circuit development, assembly, and packaging.

3.3.1. Integrating ML Model along with Sleep Functionality

Our first task was to integrate the machine learning model with a sleep functionality. We developed the system to be in a sleep mode initially, consuming less power, and only wake up when an interrupt from the sensor occurs due to significant motion. This interruption causes the Pico controller to start reading data from the sensor and feed it into the ML model.

The model then evaluates the motion and determines if it corresponds to the action of emptying the garbage can. If no such motion is detected, the Pico controller returns to sleep mode until the next interrupt. This feature is crucial for energy efficiency, ensuring that the system is only active when necessary.

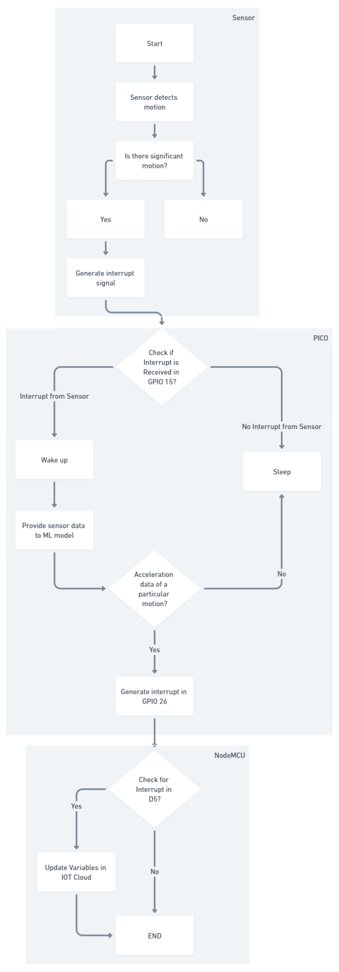

Sensor
Start
Sensor detects motion
Is there significant motion?
Yes
No
Generate interrupt signal
PICO
Check if Interrupt is Received in GPIO 15?
Interrupt from Sensor
No Interrupt from Sensor
Wake up
Sleep
Provide sensor data to ML model
Acceleration data of a particular motion?
No
Yes
Generate interrupt in GPIO 26
NodeMCU
Check for Interrupt in D5?
Yes
Update Variables in IOT Cloud
No
END

Figure 9: Deployment Software Flow Chart

3.3.2. Integration of NodeMCU with Arduino Cloud IoT Service to Send Notifications

Once the ML model detects the action of the garbage can being emptied, it triggers the Pico controller to send a signal to the NodeMCU board through one of its digital pins.

Upon receiving this signal, the NodeMCU which is connected with WiFi will initiate a communication with the Arduino Cloud IoT service. It sends a notification to the IoT dashboard indicating that the garbage can has been emptied. This information can be viewed by facility management companies, enabling them to track the emptying status of the garbage cans in real-time.

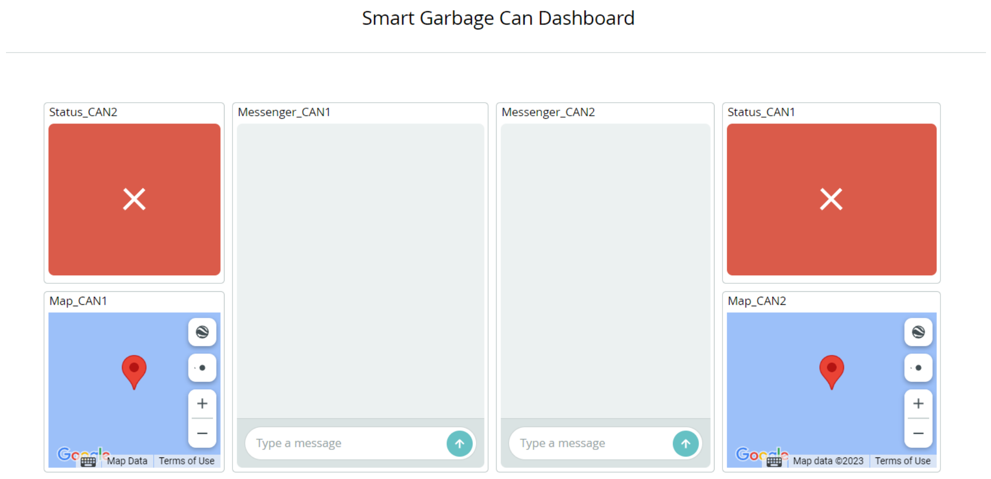


Figure 10: IoT Dashboard

Source: Arduino Cloud IoT

3.3.3. Circuit Development, Assembly, and Packaging

The final step involved developing a single integrated circuit that included the Pico Lipo controller, the acceleration sensor, and the NodeMCU board. The NodeMCU was connected to the Pico controller through one of its digital pins, establishing a communication link between the two devices.

To package our integrated circuit, we designed a custom 3D enclosure. The enclosure provided protection to the circuit components while allowing necessary connections and sensor interactions with the environment. The design of the enclosure was carefully considered to ensure it could be conveniently placed in a garbage can without interfering with its primary functions.

After assembling the circuit and verifying its functionality, the assembly was fitted into the 3D enclosure. This ensured that all the components were adequately protected while maintaining their operational capabilities.

This encapsulated the entirety of the Smart Garbage Can system. The system is now capable of automated waste management, from motion detection to real-time notifications, all while preserving energy efficiency through its sleep functionality. The 3D enclosure provides a practical, durable, and unobtrusive packaging solution, making the system ready for real-world implementation.

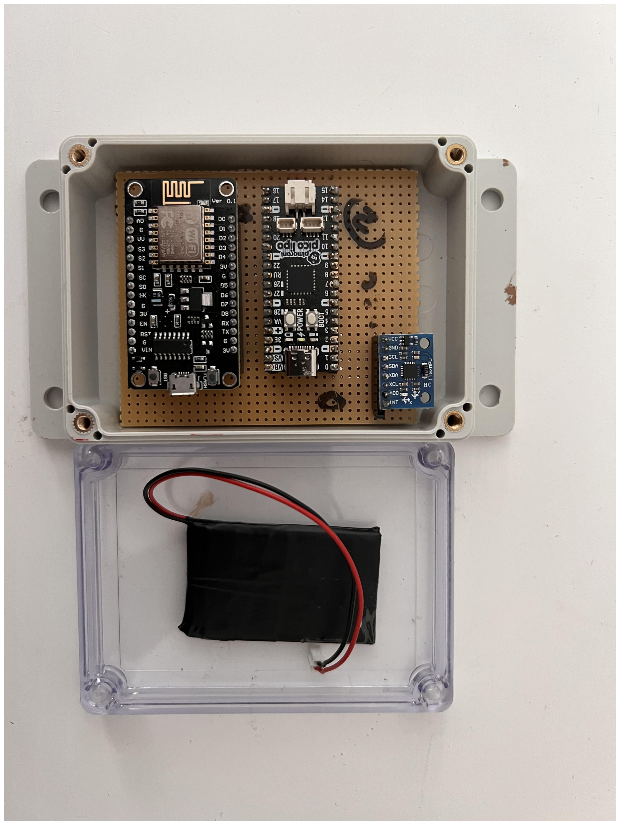


Figure 11: Enclosure for Circuit

3.4. System Testing and Validation

After the assembly was enclosed within the custom 3D package, it was time to validate the functionality of the entire Smart Garbage Can system. Comprehensive testing is crucial to ensure the effectiveness of our design and the reliability of the system in a real-world scenario.

3.4.1. Rigorous Testing

The first stage of our validation involved rigorous testing, aimed at examining the performance of our system under various motion scenarios.

Testing Still, Linear & Ramp Motions

We simulated situations where the garbage can was left still, moved linearly or moved in a ramp, cases where we did not expect to receive any notifications on the IoT dashboard. These simulations tested the system's ability to ignore irrelevant motions and avoid false alarms. As expected, the system correctly identified these scenarios and did not send unnecessary notifications, demonstrating its accuracy in motion classification.

Mimicking Trajectory Motion

In the next phase, we mimicked the trajectory motion that a garbage can undergoes when it is lifted by a garbage collection truck. As the system is designed to detect this specific motion and trigger a notification, we anticipated a corresponding alert on the IoT dashboard. As expected, the system detected the motion correctly and sent a notification, proving its reliability in detecting the act of garbage disposal.

In conclusion, through rigorous testing, we confirmed that our Smart Garbage Can system is reliable, accurate, and ready for wider deployment. The successful system validation ensures that the system can effectively contribute to enhancing operational efficiency and promoting a more sustainable environment in waste management.

As illustrated in Figure 9, the control flow sequence, beginning with the initiation of an interrupt from the Sensor to the final updates on the IoT Dashboard, is thoroughly outlined. The flowchart serves as a comprehensive visual aid, detailing each phase and subsystem involved in the process. It begins with the sensor interrupt, which is pivotal in triggering the rest of the operations. This elaborate sequence underpins the operational mechanism of our smart garbage can system, highlighting the integration of various components and technologies from the hardware level, the embedded machine learning model, to the IoT-based notification system.

4. Results

Throughout the development and implementation of our Smart Garbage Can system, we have been focused on delivering a solution that accurately detects and classifies the various movements of a garbage can in different scenarios. We have also sought to make this process energy-efficient, practical, and easy to manage for facility management companies.

Our data acquisition stage successfully captured various types of movements, including still, linear, trajectory, ramp up, and ramp down motions, providing us with a rich dataset for training our machine learning model. The Pico LiPo, selected for its excellent computational capabilities and energy efficiency, proved to be an effective choice for our embedded system, allowing the transformation of analog sensor data into digital and storing it in flash memory for further processing.

On the system integration side, our design included sleep functionality for energy efficiency, a key consideration for IoT devices. This allowed our system to remain dormant until an interrupt from the sensor triggered data reading and analysis.

The integration with Arduino Cloud IoT service for sending notifications to facility management companies proved to be effective, providing real-time status updates on garbage collection events. This feature significantly enhances operational efficiency by enabling prompt responses to garbage collection events.

In our rigorous testing, we conducted 100 tests in total, allocating 20 tests to each of the motion scenarios. The table below provides a detailed overview of the correct and incorrect classifications.

Motion Scenario	**Number of Tests**	**No. of Correct**	**No. of Incorrect**	**Accuracy**
Still	20	20	0	100%
Linear	20	20	0	100%
Ramp Up	20	2	18	10%
Ramp Down	20	15	5	75%
Trajectory	20	18	2	90%

Table 1: Performance of the Machine Learning Model in Different Motion Scenarios

Lastly, our enclosure design and assembly process proved to be successful, providing a protective casing for our integrated circuit while maintaining easy access to the device for maintenance and adjustments.

In conclusion, the results from our system testing and validation suggest that the Smart Garbage Can system can provide a valuable and efficient solution for facility management companies, revolutionizing waste management through sensor technology and machine learning. We believe that with further improvements and refinements, the potential and impact of our system can be even greater.

Hi this is test.

this is second para

5. Business Analysis

5.1. Problem Statement

Across Europe, facility managers juggle the responsibility of supervising waste disposal needs for multiple buildings. The unpredictability of municipal garbage collection schedules complicates this task, with garbage cans often left outside for extended periods post collection. This uncertainty necessitates frequent checks on the status of each bin across various locations, causing facility managers to squander valuable time and resources. These inefficiencies not only burden facility managers but also contribute to environmental degradation due to prolonged exposure of waste. Current practices are clearly in need of a technological upgrade to streamline operations, reduce human effort, and improve the environmental sustainability of waste management.

5.2. Solution

Our innovative solution, the Smart Garbage Can system, leverages sensor technology and machine learning to address these issues. The system uses the MPU6050 Sensor to monitor acceleration data from garbage bins, which is then processed through a machine learning model embedded within the controller. This system accurately detects when a bin has been emptied and subsequently sends a real-time notification to facility managers. This streamlined process reduces time spent on manual monitoring and fosters operational efficiency, contributing to a cleaner, more sustainable urban environment.

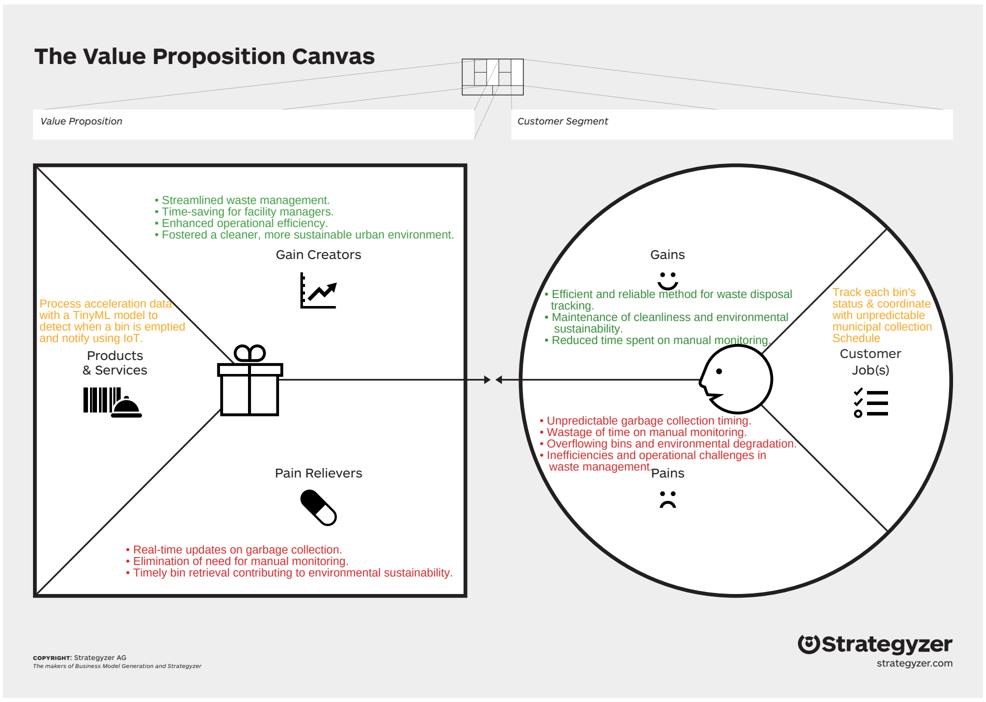

The Value Proposition Canvas
Value Proposition
Customer Segment
• Streamlined waste management.
• Time-saving for facility managers.
• Enhanced operational efficiency.
• Fostered a cleaner, more sustainable urban environment.
Gain Creators
Process acceleration data with a TinyML model to detect when a bin is emptied and notify using IoT.
Products & Services
Pain Relievers
• Real-time updates on garbage collection.
• Elimination of need for manual monitoring.
• Timely bin retrieval contributing to environmental sustainability.
Gains
• Efficient and reliable method for waste disposal tracking.
• Maintenance of cleanliness and environmental sustainability.
• Reduced time spent on manual monitoring.
Track each bin's status & coordinate with unpredictable municipal collection Schedule
Customer Job(s)
• Unpredictable garbage collection timing.
• Wastage of time on manual monitoring.
• Overflowing bins and environmental degradation.
• Inefficiencies and operational challenges in waste management
Pains
COPYRIGHT: Strategyzer AG
The makers of Business Model Generation and Strategyzer
Strategyzer
strategyzer.com

Figure 12: MPU6050 Capacitive Sensing Unit

Source: The MPU6050 Explained

5.3. Market Opportunity

5.3.1. Targeted Audiences:

This solution is particularly beneficial for facility managers of residential complexes, corporate buildings, educational institutions, hospitals and shopping malls. Furthermore, municipal corporations and waste management companies can benefit from integrating this technology into their operations.

5.3.2. Analysis & Sizing of the market:

According to a report, the global smart waste management market size was valued at USD 1.3 billion in 2019 and is expected to grow at a compound annual growth rate (CAGR) of 18.6% from 2020 to 2027. Europe generated 2.5 billion tonnes of waste in 2018, with the potential for significant growth. Given the broad applicability of the Smart Garbage Can system across different sectors, there is a significant opportunity to capture a sizeable share of this growing market.

5.4. Business Model

The primary revenue source for the Smart Garbage Can system would be the sales of our smart retrofittable IoT devices to the facility management companies. Additional revenue could be generated through offering subscription-based services for software updates and maintenance. With a good market demand, the business is likely to witness considerable growth in the coming years.

5.5. Future Scope

The future potential for the Smart Garbage Can system extends beyond just waste management. Smart bins with integrate device could be manufactured and sold to the organizations and waste management companies. The technology could be further developed to identify the type and amount of waste, which would facilitate more effective recycling practices. It could also be integrated with other smart city solutions for comprehensive environmental management.

Moreover, the solution's underlying technology – sensor data processed through machine learning models – has wide-ranging applications. It could be adapted for use in various other sectors, such as logistics or manufacturing, to monitor different types of assets and improve operational efficiencies. Here are some potential applications:

Vehicle Monitoring: The device with few customization can be used to monitor driver behavior, such as harsh braking or accelerating, or to detect accidents. This data can be used by fleet management companies, auto insurance companies, or even individual car owners to enhance safety and efficiency.

Structural Health Monitoring: The device can be used to monitor vibrations or movements in buildings, bridges, or other structures, which can help detect potential structural issues early.

Industrial Applications: In manufacturing, the device can be used to monitor machinery and detect any abnormal vibrations that might indicate a potential malfunction.

6. Conclusion

Throughout this study, we have presented the development and deployment of a smart garbage can system, employing a TinyML model embedded within the RP2040 microcontroller on the Pico LiPo, in conjunction with an accelerometer sensor for motion data acquisition. The system is designed to notify facility management companies when a garbage can has been emptied, aiming to improve waste management efficiency and promote sustainability.

The system was rigorously tested under various motion scenarios including still, linear, ramp up, ramp down, and trajectory motions. The results showcased a high degree of accuracy in most scenarios, particularly in still and linear motions. Despite challenges encountered in ramp up and ramp down scenarios, the overall performance of the system remained commendable.

An important achievement of this study was the successful integration of the TinyML model with the hardware components - the sensor, Pico LiPo, and NodeMCU. The system was able to operate in a power-efficient manner by utilizing sleep functionality until motion detection interrupts were generated.

One of the main highlights of this project was the utilization of the Arduino Cloud IoT service for sending notifications to facility management companies. This real-time and interactive feature greatly enhances waste management processes by providing instant updates on garbage can statuses.

However, there is still room for improvement and refinement. Future work could focus on improving the model's performance in the ramp up and ramp down scenarios, enhancing data acquisition for power efficiency, and investigating the use of different sensors or machine learning models to increase overall system robustness and reliability. Finally the performace of the device when integrated with the garbage can in real time should be assessed.

In conclusion, the smart garbage can system has demonstrated the potential to revolutionize waste management practices through the innovative use of sensor technology and machine learning. This study lays a firm foundation for future research and development in this area, moving us one step closer to a cleaner and more sustainable environment.

Appendix

A. Pico Code with ML Model and Sensor Data

```
/*
 * Title: Activity detection using Raspberry Pico & MPU6050
 * Date: 28/07/2023
 * Author: Smart Garbage Can team
 * Brief: Code for getting Mpu6050 acceleration data and detect garbage can empty operation using tiny_ML model.
 * Pins used: GPIO 15 - MPU6050 interrupt pin, GPIO 0 - SDA (MPU6050), GPIO 1 - SCL (MPU6050), GPIO 26 - NODEMCU Interrupt
 */

#include <pico/stdlib.h>                // Pico Standard Library
#include <pico/sleep.h>                 // Pico Sleep Library
#include <hardware/rosc.h>              // Pico Ring Oscillator Library
#include <Adafruit_MPU6050.h>           // Adafruit MPU6050 Library
#include <Adafruit_Sensor.h>            // Adafruit Sensor Library
#include <Wire.h>                       // Arduino I2c Library
#include <Final_inferencing_motion.h>   // TinyML Library from Edge impulse
//#include <Final_inferencing.h>


bool awake = false;                    // Flag to check Sleep Status
static bool debug_nn = false;
#define interrupt_pin 15               // Interrupt from Sensor
#define nodemcu 26                     // Interrupt from Sensor

Adafruit_MPU6050 mpu;                  // Create a Sensor Instance

void setup(void) {
  // Try to initialize!
  if (!mpu.begin()) {                                    // Initializing MPU6050
    //Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  //Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);          // MPU6050 register write Range - 8G
  mpu.setFilterBandwidth(MPU6050_BAND_44_HZ);            // MPU6050 register write Filter Bandwidth  - 44 Hz
  mpu.setSampleRateDivisor(9);                           // MPU6050 register write Sample frequency - 1Mhz/9

    // Setup motion detection
  mpu.setHighPassFilter(MPU6050_HIGHPASS_0_63_HZ);       // MPU6050 register write High Pass Filter - 63 HZ
  mpu.setMotionDetectionThreshold(100);                    // MPU6050 register write Motion Detection Threshold - 100 mg
  mpu.setMotionDetectionDuration(20);                    // MPU6050 register write Motion Detection Threshold - 20 ms
  mpu.setInterruptPinLatch(false);                       // MPU6050 register write Interrupt Singal to Latched mode
  mpu.setInterruptPinPolarity(true);                     // MPU6050 register write Interrupt Singal to active LOW
  mpu.setMotionInterrupt(true);                          // MPU6050 register write Enable Significant Motion Interrupt Singal

    // Initialize Pins
  pinMode(LED_BUILTIN,OUTPUT);                           // Initialize PICO Built in LED as Output
  digitalWrite(LED_BUILTIN,HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN,LOW);
  delay(2000);
  pinMode(interrupt_pin, INPUT);                         // Initialize Sensor to Pico interrupt pin as Output
  pinMode(nodemcu, OUTPUT);                               // Initialize PICO to Nodemcu interrupt pin as Output
}

void loop()
{

  if (!awake){
    //Serial.println("in sleep");
    digitalWrite(LED_BUILTIN,HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN,LOW);
    delay(2000);
    sleep_run_from_xosc();                                            // Put PICO to sleep until Interrupt
    sleep_goto_dormant_until_pin(interrupt_pin, false, false);    // Wake PICO from sleep if there is a Falling Edge in the Interrupt Pin
    awake = true;                                                     // Set the awake falg to TRUE
  }
  else if (awake){
    digitalWrite(LED_BUILTIN,HIGH);
    uint8_t buf1[64]="traj";
    uint8_t buf2[64]="still";

    float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };                                        // Allocate a buffer here for the values we'll read from the IMU

    for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {
          uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 10000);                     // Determine the next tick (and then sleep later)
          sensors_event_t a, g, temp;
          mpu.getEvent(&a, &g, &temp);                                                                  // Get the data from the sensor using sensor_event
          buffer[ix] = a.acceleration.x;                                                              // Store the data in Buffer
          buffer[ix+1] = a.acceleration.y;
          buffer[ix+2] = a.acceleration.z;
          delay(40);
    }

    signal_t signal;
    int err = numpy::signal_from_buffer(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);     // Turn the raw buffer into a signal which we can the classify
    if (err != 0) {
        ei_printf("Failed to create signal from buffer (%d)\n", err);
        return;
    }

    ei_impulse_result_t result = { 0 };                                                             // Run the classifier

    err = run_classifier(&signal, &result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;
    }


    ei_printf("Predictions ");                                                                        // Print the predictions
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
    result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");

    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label, result.classification[ix].value);
    }

    #if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    anomaly score: %.3f\n", result.anomaly);
    #endif

    if (result.classification[0].value > 0.6){
      delay(500);
    }

   if (result.classification[1].value > 0.6){
      delay(500);
   }

   if (result.classification[2].value > 0.6){
      delay(500);
    }

   if (result.classification[3].value > 0.6){
      delay(500);
   }

   if (result.classification[4].value > 0.72){ // Flicker the LED if
       trajectory is detected
    digitalWrite(nodemcu,HIGH); // Generate the Interrupt to NodeMCU
    for(int i = 0; i < 5; i++){
        digitalWrite(LED_BUILTIN,HIGH);
        delay(500);
        digitalWrite(LED_BUILTIN,LOW);
        delay(500);
    }
    digitalWrite(nodemcu,LOW);
}

    awake = false;

        // Set the awake flag to FALSE
    digitalWrite(LED_BUILTIN,LOW);

  }
}
```

Listing 1: Pico Code

B. NodeMCU Code to Send Data to IoT Dashboard

```
/*
 * Title: NodeMCU to IOT Dashboard Update
 * Date: 28/07/2023
 * Author: Smart Garbage Can team
 * Brief: Code for sending the data to IOT Dashboard after interrupt is
    received.
 * Pins used: GPIO D6 - Interrupt from PICO
 */

#include "arduino_secrets.h"
#include "thingProperties.h"

#define INTERRUPT 14

void setup() {
  Serial.begin(9600);   // Initialize serial and wait for port to open
  delay(1500);    // This delay gives the chance to wait for a signal
  pinMode(INTERRUPT, INPUT);  // Initialize Interrupt pin as INPUT
  initProperties(); // Defined in thingProperties.h

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  // Connect to Cloud
  setDebugMessageLevel(2);  // Check for errors in Connections
  ArduinoCloud.printDebugInfo(); // Print the error in Connections
}

void loop() {
  ArduinoCloud.update();   // Update the Variables the Arduino Cloud

  if (digitalRead(INTERRUPT) == HIGH) // Check for Interrupt
  {
      status = true;  // Set Status to True
      message = "Garbage Collected from Can 1 | Seepark"; // Set Message
      location = Location(48.0081,7.8215); // Set Location to Seepark
      ArduinoCloud.update();  // Update the Variables the Arduino Cloud
      delay(5000);
  }
  status= false;  // Reset Status to False
  message = "Resetted"; // Reset Message to "Reseted"
}
```

Listing 2: NodeMCU Code

References

[1] G. Bishop and G. Welch, "Smart waste collection processes a case study about smart device implementation," *Proc of SIGGRAPH, Course*, vol. 8, pp. 27 599–3175, 2001.

List of Figures

List of Tables