# Final Report - Implementation of Smart Bin

School of Computer Science & Statistics
Trinity College Dublin (TCD)
Ireland
tiliu@tcd.ie, zhye@tcd.ie, dengji@tcd.ie, xkong@tcd.ie, keumm@tcd.ie

***Abstract*—The issue of handling waste in urban areas is a common challenge faced by all cities. In an attempt to find a better solution, our group has tried to apply IoT technology and explore the potential of using "smart bin" to manage waste more effectively.**

***Keywords— Waste management, IoT, Smart bin***

## I. INTRODUCTION TO THE PROJECT

Urban waste management is a hard-dealing problem that all cities are struggling with. Common dustbins installed along the road are usually filled over with trash but with no concern to be cleared up when they are packed completely with overflowing trash. This results from the increasing population in the urban city [1] and the poor designs of waste collection. To be specific, the waste management companies have no knowledge of the current level of trash inside the bin, and the location of the one that overflowed which results in a delay in waste collection. Leveraging the IoT technology on the dustbin is anticipated to improve the efficiency of waste collection as the dustbins are "self-aware" enough so that they can automatically "share" their own states with the operational staff via network in a timely manner. The cooperation of different sensors or devices can equip the dustbins with multiple functions that can facilitate the running of waste collection.

### A. Trash Level Detection

Combining the distance measuring sensor (e.g., ultrasonic sensor) with the dust bin can provide real-time monitoring of the garbage level. The city manager can view the current trash level of each Smarbin through our dashboard. There are four indications of the trash level in terms of three intervals (e.g., [0 – 30%], [30% – 70%], [70% - 100%]), which are displayed by three colors respectively, which are green, yellow, and red.

### B. Weight Detection

The weight detection equipment is not included in our final implementation. However, the combination of weight sensor and ultrasonic sensor giving the insight of the internal situation of bin capacity is still worth being examined, using which the compactor would function only the ratio of weight to height reach the threshold rather than being utilized every time the trash thrown in. An advantage of introducing this mechanism is, to an extent, balancing the tradeoff between the energy consumption and internal space utilization.

### C. Compactor

Equipping a compactor within the bin is always a good idea for managing the waste collection as it can maximize the space utilization as much as possible. As the concern of energy efficiency is usually prioritized in the area of edge devices, the compactor would only be utilized when the ratio of weight to height reach the threshold as discussed in the previous section. Therefore, the compactor would not result in prohibitive energy consumption. Moreover, the compactor can response to the cleaning staff who has the authorization to access our application, in which he/she can send the instruction for compressing the trash by pressing a button.

### D. GPS

On the application level, the GPS sensor is used to display the location of each dustbin on the dashboard. On the waste management level, the GPS information can be used to ease the workload of waste collection. For example, our application applies shortest path algorithm to ensure that all dustbins (i.e., ones that need to be collected) would be visited in an optimal order. In our stretch goal, this function can be enhanced by cooperating with transportation sector to avoid congestion.

### E. Humidity detection

This function seems to be trivial, however with our consideration that the smell would significantly influence the first impression of people who are living in or visiting this city. This idea derived from a usual phenomenon that the trash smells stinky in a humid environment no matter if it results from wet weather or liquid splashed inside the bin. In such cases, the aforementioned level of trash would be dramatically increased so that the wetted bins would be collected in a more frequent manner.

### F. OTA

With thousands of bins across Dublin City, it is impossible to update the firmware manually via physical access. Instead, we apply OTA (Over-The-Air update) in our smart bin, which refers to the ability to update the software on a device remotely.

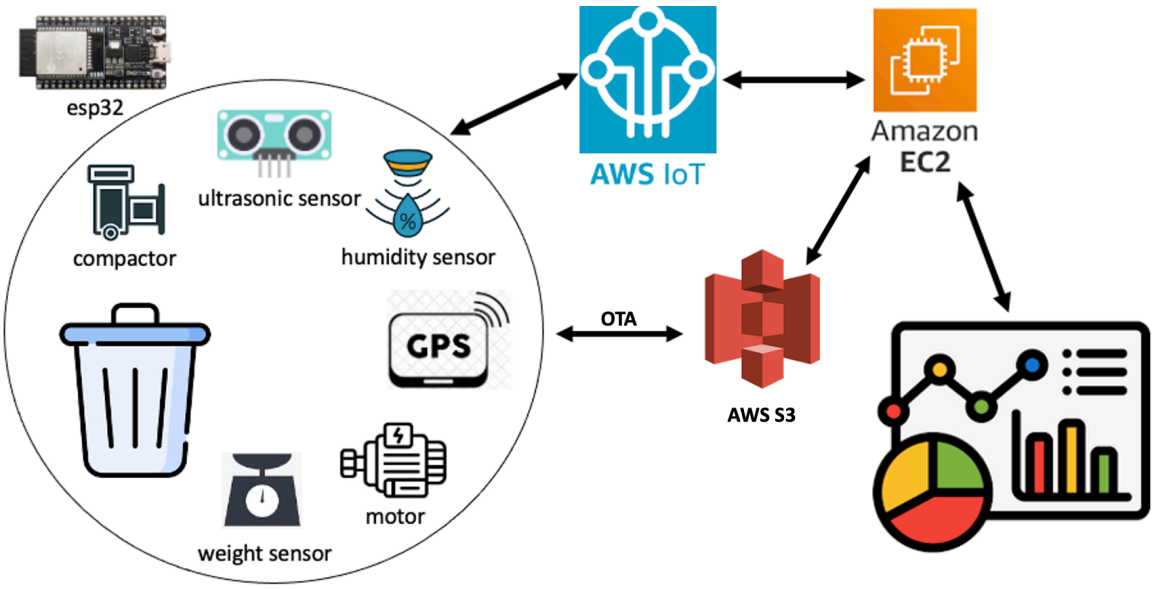


*Figure 1 Architecture of smart bin*

## II. Implementation

The implementation of this whole project mainly contains three parts (i.e., application, cloud services, and embedded system). The following section discusses the technologies applied in our implementation with our consideration of choosing this technology.

### A. Application

As it is required to display the real-time data on the dashboard, WebSocket is the ideal technology that offers real-time communication in a bi-directional manner, which means the backend service can push data to the frontend once it receives the sensor data from the embedded system, so that the dashboard can update itself instantly without refreshing the page to send another http request to fetch the newest data.

### B. Cloud Service

To smooth the development process and make our product as portable as possible, three cloud services (AWS IoT, AWS EC2, AWS S3) are selected to support the essential functions.

To build the connection between the application and the embedded system, AWS IoT is used for the enablement of passing the real-time data back and forth. We are using AWS EC2 to hold our backend service and the data persistence service. Moreover, AWS S3 bucket is used to store the newest firmware. When there is new update in AWS S3, our backend service would be notified, then it would fetch the URL of that new firmware and publish it to the AWS IoT, whose topic is subscribed by our embedded system. Once the embedded system acquires the URL, it would download the firmware via Https.

### C. Embedded System

The implementation of our embedded system mainly contains three modules (i.e., MQTT, Sensors, Coordinator).

**Network:** The MQTT module uses coreMQTT library, which is provided by AWS and offers the authentication for connecting to AWS IoT, security protocol (e.g., TLS), and several backoff algorithms. Except these, initiating the connection and reconnecting, and the interaction with WiFi module are built from scratch. The workflow of MQTT reconnection is illustrated in the figure 2. When the MQTT fa-

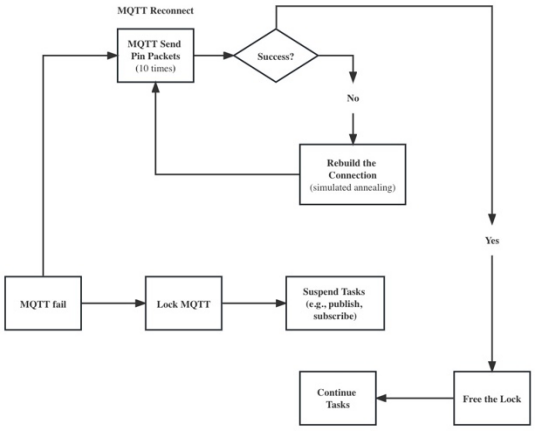


*Figure 2 MQTT Reconnection*

-ils, usually caused by unstable network or no Wi-Fi coverage, it would first send 10 pin packets, which is tuneable, to recheck the connection. After that, it rebuild the MQTT connection based on the backoff algorithm (i.e., simulated annealing) if all 10 pin packets failed. Besides, the MQTT object is locked if the connection was unavailable. Therefore all the tasks that are accessing the MQTT object would be suspended, in which case the CPU resource will not be assigned. After successfully rebuilding the connection, the MQTT object will be unlocked, so that all the suspended tasks can continue their jobs.

**Sensors:** We are using object-oriented programming (i.e., OOP) to implement all the sensors. Each sensor can be viewed as a single object that barely associates with others. Moreover,

they are running asynchronously on their own threads when their jobs start (e.g., monitoring). Memory for running each task is strictly assigned when it is created. Besides, getting data from sensors and sensors writing data usually refer to the same block of memory, where mutex is necessary to ensure accesses from different threads will not race with each other. Before the mutex is unlocked, the data will be deep copied, which can prevent the incoming data flushed the current one the other threads are trying to read. Last but not least, variables that are no longer needed will be garbage collected.

**Coordinator:** Coordinator is the object that is responsible for handling all the logics such as turning on the yellow led and turning off the green led in the meantime. Moreover, some of the tasks need to be executed in sequence, for example MQTT connection cannot start without the Wi-Fi connection. In this case, the Wi-Fi module and MQTT module are running on different thread and have no knowledge of each other. We are using EventGroup, an API provided by FreeRTOS, to synchronize these tasks, in which the end or state changes of one task will be the sign of initiating the other tasks.

## CONCLUSION

This report illustrates a concrete idea on how IoT technology can be applied to urban waste management by developing a smart bin project. The functionalities of our smart bin are explained first, such as trash level detection, weight detection, compactor, GPS, and humidity detection. After that, components (e.g., cloud services) that are integrated in this system are given. In the end, the implementation of the embedded system is detailed along with the consideration taken from the process of development.

## REFERENCES

[1] A. Sinha and P. Das, "Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry," 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 2021, pp. 1-4, doi: 10.1109/IEMENTech53263.2021.9614779.