Smart Garbage Management

Design Document v1

Team sddec18-08
Client/Advisor: Prof. Goce Trajcevski
Team Members/Roles:
Steven Brown - Hardware Design
RJ Duvall - Web Development
Brendan Finan - Mobile Development
Sam Johnson - Big Data
Colin McAllister - Embedded Systems
Nicholas Pecka - Networking
Team Email: dec1808@iastate.edu
Team Website: sddec18-08.sd.ece.iastate.edu
Revised: 02/04/2018, v1

0. Introductory Material

0.1 List of Definitions

- Garbage: Residential waste.
- Collect/Collection: The act of picking up garbage.
- Resident: A homeowner who has garbage to collect.
- Collector: An employee of a waste management company, tasked with collecting garbage
- Admin: A user in charge of an instance of our software.
- App: The Android mobile application component of the project, including both the Resident and the Collector functionality.

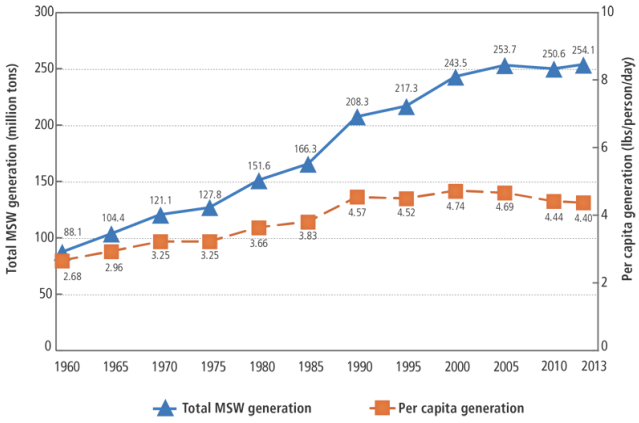
1. Introduction

1.1 Acknowledgement

Goce Trajcevski - Idea creator and IoT mastermind

1.2 Problem and Project Statement

Problem Statement: In 2013, Americans generated approximately 254 million tons of garbage. That quantity has steadily increased over the past half-century, shown in the diagram on the right (image link). However, garbage collection techniques have not significantly changed over the same time period, still relying on static routes and scheduling. This antiquated technique does not account for each individual's unique waste disposal habits, creating inefficiencies where one resident's garbage is picked up too often, while another's garbage is not picked up enough.



Solution Approach: Our solution is to add a small sensor package to each resident's garbage can. The sensor package routinely monitors the container's waste level, weight, and location and transmits the compiled information to the cloud. The information received by the cloud is then transformed into meaningful information that can create smarter logistics for garbage collection and also provide additional insight into residents' waste behavior.

1.3 Operational Environment

The environment of our mobile app is a mobile phone. We are planning on supporting Android 4.0 and later. We will also require our app to have access to the internet.

Due to the large range of environments that smart garbage containers will be exposed in, the sensor package installed on garbage containers will need to be water and dust resistant. The sensor package is also expected to perform in extreme temperatures due to the variability of climates in the United States. The IEC standard 60529, commonly known as the IP Code, will be used to standardize the weather rating of the enclosures. Since the product will be outside for potentially long durations, the smart garbage container will be designed with an IP66 rating. The sensor package will also be designed to withstand other environmental threats like vibration, shock, and salt-water.

The initial design for the sensor package uses a solar panel to recharge its battery. Since the garbage can will be outside for collection days, the sensor package can only use only the amount of power generated in that time.

1.4 Intended Users and Uses

Our platform aims to make waste collection more enjoyable for both residents and service providers.

Residents will use one of the platform's mobile applications to communicate with their service provider and gain further insight into their waste behavior. The resident's mobile application will notify them when they should place their garbage bin on the curb for their scheduled collection day. The user could also alert their service provider if their garbage bin needs to be collected as soon as possible. Additionally the mobile application could show each resident their individual trends for their trash generation, allowing the customer to gain additional insight into how they throw away trash.

Garbage collectors will use the platform to develop better logistics and planning for their service. Garbage collectors will want optimized routes that minimize drive time and maximize efficiency.

Collectors will also want to be able to discover trends in their customers behavior. This will allow collectors to allocate resources more efficiently throughout the year, giving workers more time off during slow parts of the year and ensuring that there are enough collectors on staff when trash generation is expected to peak.

1.5 Assumptions and Limitations

Assumptions

- We assume that Residents will have access to a cell phone with Android 4.0 or later.
- Residents will be willing to participate in our platform,
- Residents will not want to charge their garbage cans
- A waste management company will only collect in a single city
- All residential areas where the product will be used are covered by a cellular network

Limitations

- Our platform is designed for “traditional” residential areas: blocks of houses with a grid of interconnected streets.
- We will only be able to produce good routes: producing the best possible route may be beyond computational resources.
- Sensor package easily fits on the lid of a garbage can
- Sensor package recharges on its own without the customer

1.6 Expected End Product and Deliverables

After the first semester we expect to have a proof of concept sensor package. This proof of concept device will be able to measure both the trash height and weight. This data will be sent to the AWS platform along with the sensor package’s GPS location. The completed proof of concept device will be used to determine the optimal types of sensors and develop a system that can power the device. Once the proof of concept device is built, the second semester will consist of refining the device into a marketable prototype that can be proposed to a municipal waste company for further development and adoption.

2. Specifications and Analysis

2.1 Proposed Design

Our design

The mobile application will consist of different Activities, which can be accessed through the following pattern:

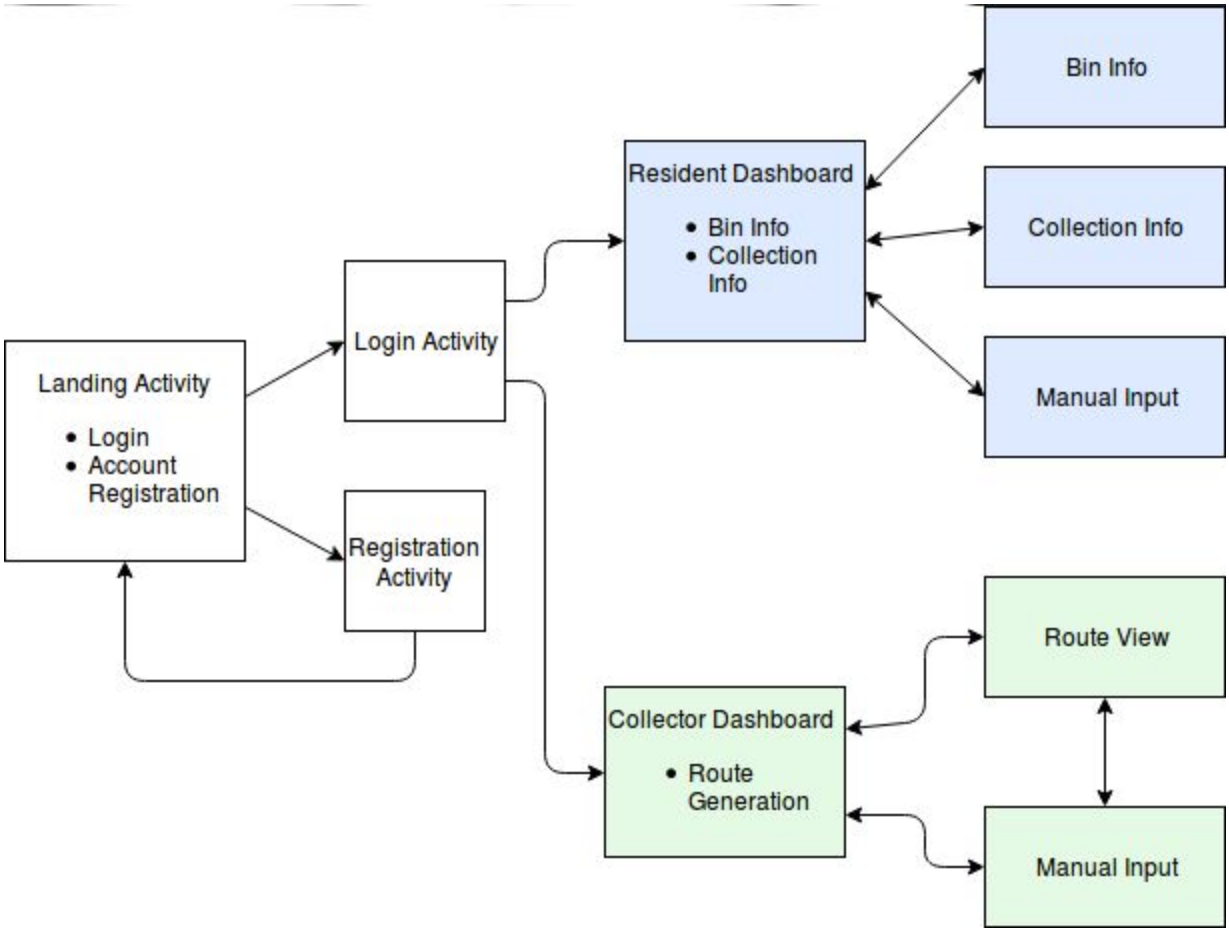


Fig X.x: Navigation of the Android Activities.

As of right now we have developed an initial idea for a sensor package prototype. The sensor package will include four individual sensors that detect location, lid movement, trash height, and trash weight. These sensors are connected to the PyCom FiPy, which houses the MCU that controls the device. The PyCom FiPy comes equipped with a 4G LTE-M modem which is capable of transmitting low bandwidth, high range messages using very little power. The PyCom FiPy and sensors will be powered by either an array of supercapacitors or some type of rechargeable

battery. The capacitors or battery would be charged by a solar panel. Some type of circuit will be built that manages charging of the battery or capacitor and regulating the voltage to the PyCom FiPy.

The parts to build the initial prototype have been placed on order. Once the parts arrive we will begin testing each of the components individually. For example an infrared proximity sensor and an ultrasonic sensor have been selected as possible sensors for detecting trash height. The sensor that performs best in testing will be incorporated into the proof of concept device. Once all sensors are selected. Software will be written to interface each of the sensors with the FiPy.

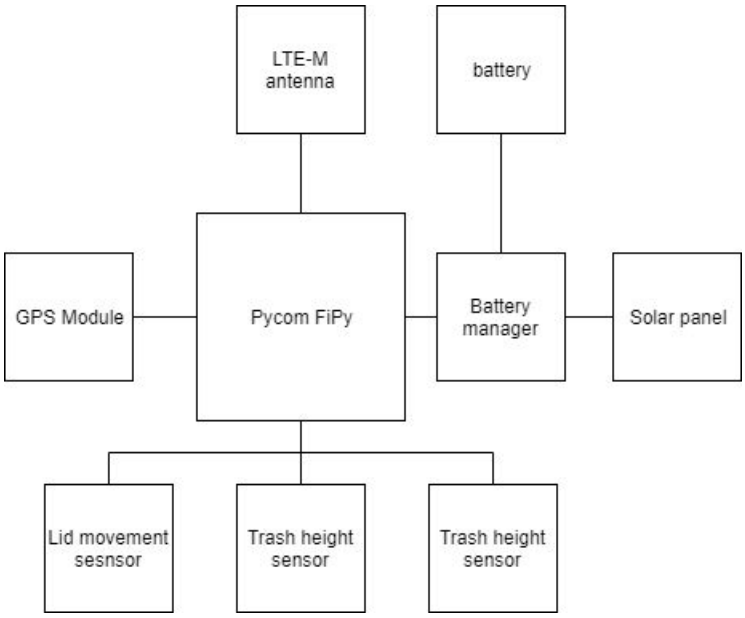


Fig X.X: Abstract diagram of hardware components

After the sensors have been interfaced, the software to implement the flow diagram shown below will be written. Software to connect the FiPy with the cloud via MQTT will also be written as well.

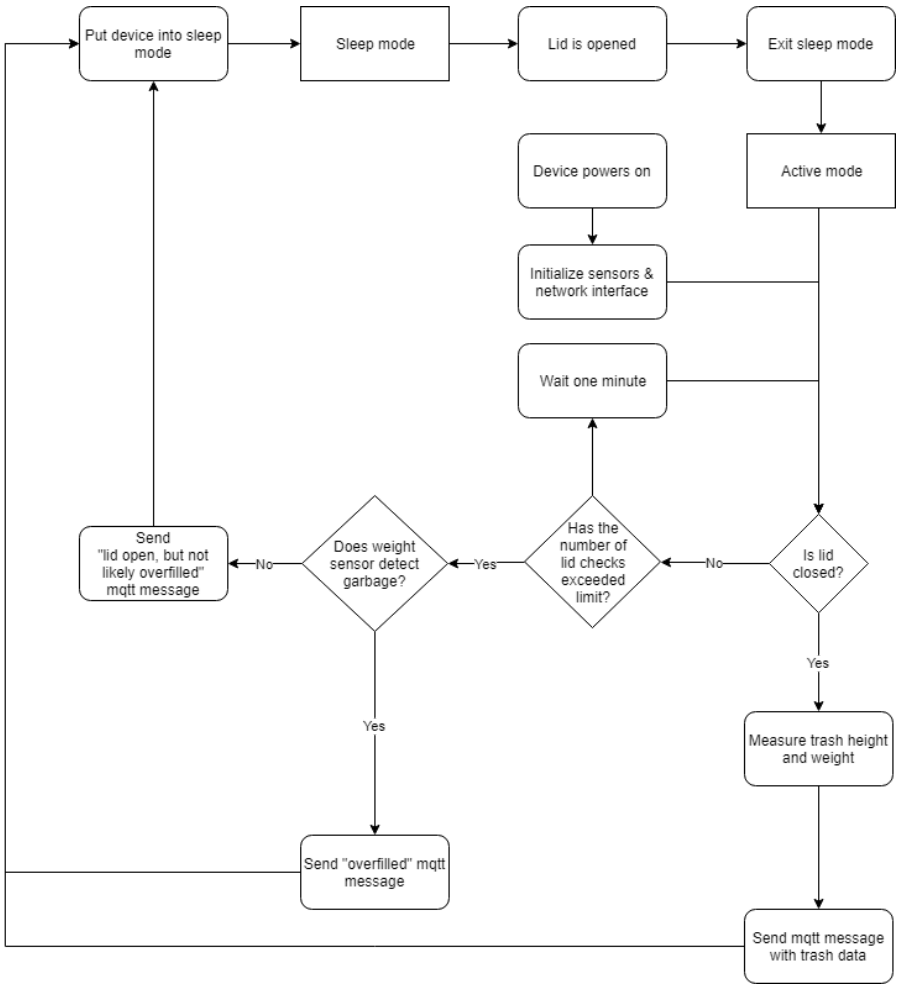


Figure X.x: Sensor package flow

3. Testing and Implementation

hardware parts with a mobile app. As such, we will need to test both the hardware and software as well as the integration.

Hardware:

The hardware will consist of a device mounted to the lid of a standard residential style garbage can that will be able to detect the distance to the bottom of the can by some sensing means that will be determined after testing a few candidate sensors. A weight sensor may also be attached to the bottom of the can such that the weight of the trash can be used in decision making. These sensors will be connected to a microprocessor with radio peripherals that will be able to transmit the state of the weight and height of garbage to our server side infrastructure.

Our hardware testing plan is as follows:

The sensor elements will be individually tested using calibrated laboratory equipment available in our labs to very their function, calibration and suitability. This tentatively includes tests for:

- Distance sensing using a LDPE and paper targets every 15 centimeters from 0 to 2 meters and measuring the sensor output using a multimeter.
- Weight sensing using known weights from every 5kg from 0 to 100kg and measuring the sensor output using a multimeter
- Switch testing using an oscilloscope to verify

The energy harvesting and storage system will be tested using a dummy resistive load that will simulate approximate load of our actual system. This will generate storage capacity rating and approximate usable lifetime figures. The test will pass if it can retain more energy than required by our worst use case.

The remaining piece of the hardware to be tested is the microcontroller which in this case has numerous radio interfaces. LoRa, LTE-M, SigFox and WiFi. Not all of the capabilities of the radios may be tested individually due to lack of receivers to test them with and with the large expense in both time and equipment it would require to test them in a controlled manner. However these radios will be covered by integration testing and WiFi in particular will be exercised extensively during development.

We currently do not have any intention to do accelerated lifetime testing or combined environmental testing due to the proof of concept nature of this project. However we would advise that prior to a product being commercialized that these types of tests be performed.

Software:

For software we will be required to test in four major areas: The algorithm for clustering garbage bins, the route creation algorithm, connectivity between the mobile app and the server, and general usability. The clustering algorithm will be tested by manually creating datasets that range from very easy to hard to cluster and comparing the performance of our algorithm with the optimal clustering. For route creation, we will randomly generate clusters of garbage bins that need to be picked up and compare the routes generated to already existing garbage routes and other algorithms used to generate shortest paths. Testing connectivity will be done by simulating use cases on the mobile app and assuring that the users will be able to do them. General usability will be tested by having people with varying levels of technical literacy attempt to use some function on the app. We will also periodically check user satisfaction with a quick survey screen.

3.1 Interface Specifications

Interfacing between the hardware component and the software system will be accomplished through MQTT messages on Amazon Web Services' WebSocket Protocol.

We decided on MQTT (specifically MQTT-SN which is just MQTT but for sensor networks) as it is designed to function well with low power usage which is a main concern in our project. With low power usage in mind, MQTT's support an offline procedure which will save power by buffering messages while in sleep mode and then deliver them upon wake. Since we also only need short bursts of communication between our network of trash cans and database MQTT's work great as one of their main designs is to interact well over low bandwidth and short messages.

Our group decided on using Amazon Web Services (AWS) for our server and AWS has a protocol that supports MQTT's which is AWS's WebSocket protocol. Specifically the message broker supports the use of MQTT to publish and subscribe over Signature Version 4 Signing Process (SigV4) authentication.

http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf -Link for reference on MQTT-SN

3.2 Hardware and Software

Software

- Android Version 4.0 or later

3.3 Results

Results will be based on the outcomes of our our testing plan.

4 Closing Material

4.1 References

PyCom documents

AWS IoT services

4.2 Appendices

Bill of Materials