Smart Garbage Management

Final Report

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

Introduction 3
Problem Statement and Solution 3
Requirements 3
Functional Requirements 3
Non-Functional Requirements 3
Intended Users and Uses 4
Residents 4
Collectors 4
Assumptions and Limitations 4
Assumptions 4
Limitations 5
End Product Deliverables 5

Revised Project Design 6
Initial Design Decision 9
Change to Final Sensor Package 7
Switch to Genetic Algorithm 9
Final Design 9

Implementation Details 9
How Will the Project Be Deployed to Waste Management Companies 11
How Users Will Interact with Their Garbo's 11
How Trash Collectors will Interact With Garbo 11
How Garbo Will Achieve the Mission Objective 11

Testing process and testing results 11
Testing Garbage Bin Sensor 13
Testing Algorithm 13
Testing App with AWS Server 13
Testing Communication Between Server/App/Database 13
Testing Communication Across Hardware & Software 13

Related products & Related Literature 14
Client-provided Literature 14
""Hey #311, come clean my street!" - 14
Applications Already Available 14
Enevo 14
SmartBin 15

Appendices 16
Appendix: I - "Operation Manual" (Setup/demo/test) 16
Instruction guide on how to install Garbo 17
Instruction guide on how to pair app to Garbo 17
How to use on daily basis 17

Frequently Asked Questions 18
Appendix: II - “Alternative/ other initial versions of design” 18
Appendix: III - “Notes” 19
Appendix: IV - “Hardware / Software Components” 20
Schematic 20
Routing Algorithm 21
System Connection Code 21
Full IoT Diagram 22
User Dashboard 23

Introduction

Problem Statement and Solution

In 2013, Americans generated approximately 254 million tons of garbage. That quantity has steadily increased over the past half-century. 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. Collection routes also do not take into account the rates that individual collection vehicles fill up at. This creates situations where some garbage trucks have to stop their route to empty the vehicle, while others are not completely full after running their whole route.

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. The sensor package then transmits the compiled information to the cloud. The information received in the cloud is stored in a database. Each night the application will take the garbage bin information to build clusters of nearby bins that need to be picked up and build fuel-efficient routes to pick up the clusters. This means that the garbage will be picked up when it needs to be as opposed to being on a fixed schedule. Our application will give users a better garbage collection experience, allow waste management to better allocate their limited resources, and be better for the environment.

We named our solution "Garbo".

Requirements

Functional Requirements

- Trash bin device must determine approximate weight and height of contents
- Garbage sensor communication must be secure, verifiable, and guaranteed to reach the cloud
- Generated routes must ensure every bin is picked up

Non-Functional Requirements

- Scalable back end
- Heterogeneity of systems

- System must be user friendly and cost effective
- User data must be protected

Intended Users and Uses

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

Residents

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 have the ability to show each resident their individual patterns of trash generation, allowing the customer to gain additional insight into how they throw away trash.

Collectors

Garbage collectors will use the platform to develop better logistics and planning for their service. Garbage collectors will be able to use our product to generate optimized routes that minimize drive time and maximize efficiency. The app will also help garbage collectors discover trends in their customers' behavior. This new data-driven approach will allow collectors to allocate resources more efficiently throughout the year by giving them the ability to forecast using historical data. This will enable them to give workers more time off during slow parts of the year by running fewer routes. They will also be able to ensure that there will be enough collectors on staff when trash generation is expected to peak.

Assumptions and Limitations

We assume the following about the usage of our platform:

Assumptions

- 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 and/or may be unable to provide traditional mains power where the garbage cans are stored and used.
- 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 resource limitations.
- Sensor package form factor will be designed such that it easily fits on the lid of a garbage can.
- Sensor package recharges on its own without customer interaction.

End Product Deliverables

At the end of the first semester, our garbage bin sensor device was capable of communicating with Amazon IoT via MQTT messages over an LTE Cat M1 connection. We also had an understanding of the sensors needed to build the garbage bin sensor. These sensors are responsible for measuring the trash height, weight, and the location of the trash bin. After we selected the types of sensors to use, we started selecting parts that could be used for a finalized proof of concept device. This means each component is capable of withstanding our environmental requirements and uses the minimum amount of power possible. We developed a bare-bones version of our web application and had a clustering algorithm that worked for samples of 200 nodes.

The second semester we assembled the selected components into a finalized fully functional and ruggedized prototype that can be used to demonstrate our idea to potential clients or investors. This device will be able to satisfy all the requirements we established, including the ability to be attached to a standard garbage bin, delivering trash and location data to the cloud, and demonstrating the non-invasive charging methods that will keep the device always powered on. We also built a working web prototype that allows a user to query a noSQL database and returns a garbage route on an Open Street Maps view. Routing was done through a genetic algorithm that takes in user settings and a weight and distance matrix of all of the full trash cans on that user’s network.

Revised Project Design

Initial Design Decision

One of the first parts of the design process for the hardware team was picking a wireless communication technology for the garbage sensors. Our most important evaluation criterion was that the technology had long range and consumed low power. Cost and complexity for deploying the wireless technology were also considered.

One of the first ideas considered was to attach each garbage sensor to the associated resident's Wi-Fi network. This would allow each device to communicate independently on a pre-existing network. However, most homeowners' wireless networks are protected by a WPA2 personal protocol, and using the residents' credentials could create security risks. Additional problems could arise from a resident not having a Wi-Fi network, or their network failing.

Communication over a mesh network topology was also considered. Products like the ESP32 and DIGI XBEE line of modules feature software development kits that feature the ability to create and communicate across mesh networks. The greatest benefit was potentially only one router or gateway would be needed to connect the mesh network to the database. However, this cannot be guaranteed and problems would arise from isolated mesh networks that are unable to connect to the central database. Additionally, a mesh network would require all devices to constantly be on in order to allow any devices to communicate with the database. Requiring all devices to be constantly on and communicating would require much higher power consumptions on each garbage bin sensor.

Another technology that was considered was LoRa. LoRa is a sub-gigahertz chirp spread spectrum wireless technology. One of the largest benefits to LoRa is the long range that can exist between endpoint devices and a gateway. LoRa also consumes little power. Each LoRa endpoint device would be connected to a LoRa gateway. One downside of LoRa was the limited bandwidth, but the bandwidth needed for our design was low to begin with. The largest downsides our team found were infrastructure costs and complexity. For waste management companies with a collection area wider than a single LoRa gateway's range, multiple LoRa gateways would need to be placed around the waste management's service area.

The last wireless communication technology we considered was LTE-M. LTE-M encompasses both LTE Cat-M1 and NB-IoT, two independent technologies. NB-IoT was still in development in the spring, so it was not considered. LTE Cat-M1 is a machine-to-machine communication network that exists on the latest mobile communications standard. Cat-M1 operates in half-duplex mode, which lowers the bandwidth but allows lower power consumption and greater communication distances. One huge benefit was that the networks were already rolled out by

AT&T and Verizon. Another benefit was that the network's range was only limited by the service provider's coverage. Testing showed that connection to the web was very easy, and using MQTT for data transmission was straight-forward. The only downside was the cost of each data plan per device.

Our team decided that LTE-M would be the best technology to start with due to its existing wireless networks and ease of integrating with AWS. LTE-M also allowed the largest range without having to use multiple LoRa gateways.

Change to Final Sensor Package

The ultrasonic and load cell sensors performed well during our initial testing. The ultrasonic sensor met our figures for accuracy in the range that would cover the average curbside garbage can. Although the ultrasonic sensor isn't as ruggedized as we would have liked, we believe that it will be acceptable due to it being recessed in the top of the lid. The load cell integrated well into the bottom of the trash can that we had selected. The load cell was extremely rugged, being machined out of stainless steel and capable of measuring up to 200kg. The MAX1454 sensor conditioner was added to the signal path of the load cell and is capable of compensating for temperature and calibrating the load cell to very accurate ratiometric output.

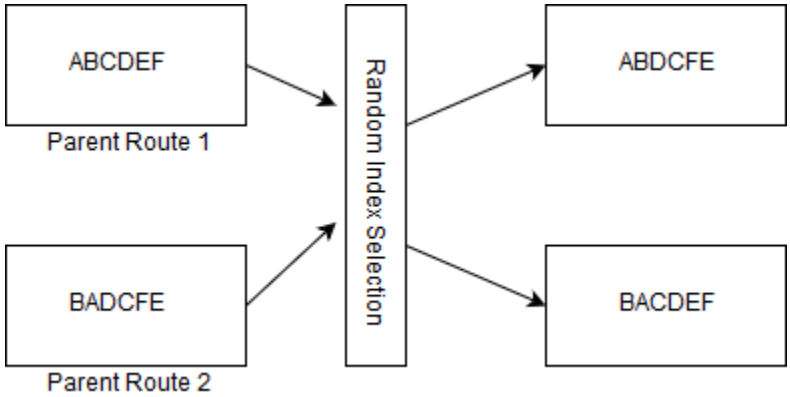
Switch to Genetic Algorithm

Our initial plan for creating routes was to build clusters of garbage bins by creating n-directed minimum spanning trees where edges were scored by combination of distance weight. We pivoted off of this after realizing two major design flaws. First, this implementation focused on the weight of each individual bin instead of the total weight of garbage in the truck, which would produce sub-optimal routes for gas consumption. Second, the generation of the directed spanning tree performed poorly on graphs with more than a thousand nodes. Additionally, a bug in the library caused it to occasionally produce cycles. From there, we looked into the idea of focusing on high-weight areas to cluster, but hit dead ends while looking into how to do that. After our first PIRM presentation, Dr. Daniels recommended we look into using a genetic algorithm to build our route. We researched genetic algorithms for two weeks. After looking at a few genetic algorithm heuristics for the vehicle routing problem, we decided that implementing a genetic algorithm would be great for our project.

Our project's routing problem is a subset of the Vehicle Routing Problem. We have some number of trucks and some number of bins. Each bin needs to be visited exactly once, and a weight value needs to be moved from the bin to the truck. However, the trucks have a maximum carrying capacity before they overflow. How could you get each bin picked up in the most efficient way? The Vehicle Routing Problem is in a class of problems known as NP-Hard, which means the finding an exact solution takes an exponential amount of time relative to the number

of inputs. For a municipality that needs to get hundreds or even thousands of trash cans picked up, that is simply not possible. Fortunately, there are many heuristics that can create a route that is ‘good enough’ for practical purposes. A genetic solution is almost perfectly suited for this type of searching problem, so after review of the situation, we took Dr. Daniels’ advice and pursued creating a genetic algorithm for our routing.

The philosophy behind the genetic algorithm is that if you take a population of random routes and build a scoring system to determine how close to the optimum route they are, you can combine the best parts of the best routes to eventually get very close to the optimum route. A big advantage to using genetic algorithms for problems like this is that they are time-bounded by the efficiency of our fitness and mutation algorithms. As long as we could implement the fitness and mutation algorithms in linear time, the route-building algorithm runtime would only expand linearly as more bins are added.



Example of Crossover where index selected is two.

When routes are requested by a user, our web service makes a database call to find all trash bins in the users network that are above the weight or volume threshold set by the waste management company. An API is call is made to Graph Hopper, an Open Street Maps routing utility. A matrix of the street distances between each point is passed to the routing algorithm.

The algorithm works by creating an initial population of random routes and scoring them based on a function of distance and weight carried by the truck at each stop. This scoring function means that the routing algorithm will prefer picking up lighter bins first, provided that it doesn’t make the truck cover a notably greater distance. This feature helps the truck to consume less fuel, as the truck will be carrying less weight as it travels. A new generation is created in two phases. First, some number of “genetic winners” are selected in a “selection tournament” by taking random samples from the population and selecting the best score in the random sample. Then “crossovers” are selected by doing another “selection tournament” and taking pairs from the group of winners and splicing them together at a random point to create a “child route” . After the splice, both child routes are checked to make sure that all of the bins are contained in the route. If not, a list of missing bins is created and they replace the repeated bins in a random order. After the crosses are done the “crossovers” are iterated over and some portion of them are selected to be “mutated” by reversing the order of a subset of the route. Crossovers and mutations create “genetic diversity” in the routes, and help the routing algorithm avoid being

caught in a local minimum score. After the “genetic winners” and crossovers are created, the populations are merged and a new set of routes forms a new generation. The process is repeated on the new population of routes. Once the set number of new generations has been generated, each route is scored and the route with the best score is returned as the winning route.

Final Design

Our final design encapsulates several improvements over our original design. First, the change from a LoRa network to an LTE-M signal demonstrates research we did on the protocols. Additional components we added, such as GPS, tailored our device as an IoT can enhancement. Our change to a genetic algorithm reflects redesign and the iterations we performed after receiving feedback and criticism. We are confident in our finished product and feet that our iterative, agile approach to this project has served us incredibly well. Allowing our design to adapt and evolve with the challenges we faced has helped us to build a project that is much more nuanced and advanced than our initial design. Shifts such as only focusing on full bins for pick up solved a number of problems, both functional and non-functional. The final design that we created would not have been possible without following the principles of agile project management and iterative design.

Implementation Details

How Will the Project Be Deployed to Waste Management Companies

Before each trash sensor is deployed, the device will first be flashed with software. Additionally, a SIM card will need to be activated and placed in the device in order to communicate via LTE-M. After flashing and SIM insertion, the sensor will be sealed in a waterproof enclosure and will be ready for deployment. Our team assumed that these steps would be conducted when the device is manufactured or before it is delivered to the waste management company.

The first step in deploying a trash bin sensor will be device configuration. When a garbage bin sensor powers on, it will check for a configuration file stored in its flash memory. If no configuration file is found, the device will try to connect to a pre-specified WLAN network with default credentials flashed on the device. If the network exists and the sensor is able to connect, it will wait until the AWS keys, certificates, and device configuration file are transferred to the device via FTP. The configuration file will contain the device-specific ID, trash company ID, calibration slope for the load cell, the AWS IoT Rest API endpoint, and file paths to the keys and certificates. A software tool would be used to create the configuration file and transfer the

configuration file along with the AWS keys and certificates. The tool will also create a barcode that will be placed on the bin sensor's exterior. Once the device has been configured and affixed with a barcode, it will be ready to be installed in the field. The strain sensor package will also be placed on the bottom of the bin and connected to the main device over a wire affixed to the trash can and attached via waterproof connectors.

The installation process has been designed to make retrofitting a sensor to a residential garbage bin quick and simple. First the installation technician will use a mobile application to scan the sensor's barcode and associate the device-specific ID with a residential address, which will be uploaded to a database in AWS.

One of the main considerations when designing the garbage bin sensor was to create a modular package that can be easily retrofitted to residential garbage bins currently in use by waste management companies. Since the sensor will need access to the top and bottom of the lid, a hole will need to be drilled in the waste bin for installation. A jig would aid an installation technician to correctly place the hole on the lid. After the hole is drilled, the sensor will be placed on top of the hole, with a fastener placed on the bottom of the sensor inside the lid.
After the sensor has been installed on the bin, the installation technician will activate the device on AWS. Once this last step has been performed the device will be ready for use. Opening the lid to activate the garbage sensor will allow the technician to ensure that data is sent to AWS as a final test.

In order for a company to set up their AWS cloud stack, they would first have to contract us. We would be in full control over the AWS stack, so our first step would be creating a customized AWS User for the Waste Company Company that will exist on our current IAM user. The company would have their very own NoSQL database for storing the residents trash can information. Companies could also create any additional rules that they need for their Lambda functions.

How Users Will Interact with Their Garbo's

Interaction with the device will be minimal. The device is activated by opening the lid of the garbage can. When the lid is closed, the device will measure the height and weight of the garbage that has been inserted. The device has a diagnostic LED to help the user troubleshoot if there is a lack of solar power or some other problem. All other user interaction is provided by the customer-facing app and cloud platform, which will indicate conditions such as the lid being left open or the can being full and in need of emptying.

How Trash Collectors will Interact With Garbo

Garbo will know when garbage has been collected by the weight and height in the can decreasing after being activated by the lid being opened. The trash collectors will not have any additional task to manage the device. All interaction with the device is through the mobile application and cloud service platform.

How Garbo Will Achieve the Mission Objective

After Garbo is deployed to a residential area or larger community, Garbo will analyze the current trash within the containers and send the data to the AWS server where it will be processed and stored within a database. Once the data has been stored, the information will be used in the genetic algorithm that will determine the nodes that the garbage collector will run for that day. The stops generated will form the most efficient route based on trash within bins Garbo is analyzing, dimensions of the truck collecting, and proximity of pick up points within the neighborhood. Garbo will use this data to achieve the mission objective of finding the most efficient route for garbage pickup.

Testing process and testing results

Testing Garbage Bin Sensor

Each power section of the circuit board was tested separately and in sequence without the main microprocessor populated to save it from damage if this testing found faults. These sections consisted of the battery management system, the wake on shake section and the 5-volt buck-boost section. This testing was mainly verification that the design was performing as anticipated and that all voltages were in their appropriate ranges and would survive under the anticipated downstream loads.

When this testing was completed the main microprocessor was socketed and the sensors and communications were tested using the code we developed to initially vet the sensors with modifications for the changes made during circuit board design. With all these tests passed, we moved on to microprocessor integration testing.

Initial testing from the spring consisted of testing software for each individual sensor. Each of the critical components were tested. The ultrasonic and infrared sensors were first tested for ease of integration and accuracy with our development board. Preliminary testing showed that the ultrasonic sensor had a much better field of view without losing accuracy. Additionally, the

output from the infrared proximity sensor was non-linear, whereas the ultrasonic sensors output is linear relative for distance. The GPS module was also tested with the development board. We found that the GPS accuracy was within a couple meters of actual position and also able to provide an accurate time for the module's location. The last piece of software that was initially tested was the MQTT transmission over LTE-M. A basic proof of concept program was conducted in the spring which demonstrated that the development board was able to interface with AWS over MQTT via LTE-M.

Once the circuit board was developed and passed initial voltage and short testing, the other components were tested, including the accelerometer and sleep circuitry. The accelerometer was tested by programming it into the desired mode, where an interrupt pin would trigger power to be supplied to the development board when subjected to high acceleration. A tilt switch was also used to detect lid movement. Both methods were logically ORed together in order to compare how well each method worked in detecting lid movement. Lastly, the ability for the development board to toggle a pin to turn off the power supply was tested.

Once all the components were tested, the final prototype software was installed to verify that the prototype could meet all design criteria and be integrated with the rest of the project stack.

Testing Algorithm

Because the algorithm is a heuristic that is not guaranteed to return the best possible route, our testing was focused on what percentage of the time the algorithm generated the best route in different situations. We did this by creating a few routes that would be trivial for a human to determine the optimal routes and running a simulation thousands of times and seeing how often the algorithm returned that route. For each of these we used a population of 200 routes, produced 25 generations, and ran a simulation 1000 times. We focused on three situations:

1. A small graph, that could be solved by hand

   Optimal route found 100% of iterations

2. A linear graph with one truck picking up all bins

   Optimal route found 100%

3. A graph with two linear clusters and two trucks.

   Optimal route found 98.7%

Testing App with AWS Server

We tested our mobile app to ensure communication between our AWS Server and our users. To ensure communication, we simulated calls between the app and the server in a variety of conditions. We tested our app on different supported devices, to ensure that the interface rendered on different screens. We used location spoofing to test different location values. Finally, we tested multiple input variations, and malformed requests to ensure that our programs failed correctly when bad input was given.

Testing Communication Between Server/App/Database

To start, we tested this by showing that a GPS point for the trash can that was in the database could be taken by the Trash Collector's Route View and displayed on the Android Application. This was achieved by using a AWS Lambda function through a Lambda invoker on the application end. This AWS Lambda function would simply get the GPS point from the database and return it back to the client on the Android App side. These simple tests were branch out after proving to be successful by creating tests that would do multiple GPS points across multiple Trash Collectors running the app.

Testing Communication Across Hardware & Software

Initial testing of the garbage bin sensor's communication to the main database consisted of creating an MQTT connection with AWS to send basic JSON packages. The first tests connected over Wifi to avoid debugging both LTE-M and connecting to AWS. Basic testing showed that the software worked for creating an MQTT connection. The tests were repeated over LTE-M. After successfully testing over LTE-M, the test code was used to create the final software for the prototype.

After the prototype was created, the software was tested to ensure that all data was correctly transmitted through MQTT messages via LTE-M.

Related products & Related Literature

Client-provided Literature

““Hey #311, come clean my street!” -

Three million tweets regarding civil complaints were collected. These tweets were analyzed using sentiment analysis techniques, and mapped to their geographic origins. Using the sentiment data, a structure is created for addressing real-time civil complaints.

Applications Already Available

i. Enevo

Enevo is a Boston-based waste management company with 32 employees. They partner with hauling companies to provide corporations with a comprehensive waste pickup service. Like our project, they offer a distributed sensor and interfaces for monitoring and controlling trash pickups. They finance their operations primarily by

taking a cut of the savings they create for corporations.

ii. SmartBin

SmartBin is a company that creates their own fill-level device, similar to our device. They also offer an interface for monitoring devices, creating optimized pickup routes, and generated reports on pickup metrics.

Appendices

Appendix: I - “Operation Manual” (Setup/demo/test)

Instruction guide on how to install Garbo

- Manufacturer
    a. Program board while assembling device
    b. Register SIM card and place into slot located on the board
- Installation technician
    a. Pre-installation
        i. Ensure set-up wifi network is available
        ii. Shake garbage sensor to wake
        iii. Use program tool to set up device with AWS and place config files on device
        iv. Place printed barcode on device
    b. Installation
        i. Use jig to cut hole in lid of garbage can
        ii. Scan barcode with installation app to associate sensor with current address
        iii. Place garbage sensor on top of the hole drilled into the lid
        iv. Fasten sensor onto lid by screwing fasteners on bottom of lid
        v. Active the sensor using the installation app
        vi. Open and close the sensor and confirm via the app that the garbage sensor is working

Instruction guide on how to pair app to Garbo

The app pairs with Garbo through the web services provided to it. Using the corresponding table, the app displays the current garbage level to the user and alerts them when garbage levels reach the maximum.

How to use on daily basis

Resident

1. Pull up Garbo app on device
2. Navigate to Resident Dashboard
3. Inside Resident Dashboard
    a. Pick up date & time displayed
    b. Garbage tip of the day displayed
    c. Settings to pair and unpair Garbo
        i. Pair button pairs Garbo
        ii. Unpair button unpairs Garbo
    d. Icon in middle to view inside trash bin
        i. Click icon and sends to trash bin view
    e. Percentage of trash within trash bin

Collector

1. Pull up Garbo app on device
2. Once in app navigate to Collector Dashboard
3. Inside Collector Dashboard
    a. Cans collected, subtracted, and missed displayed with a total number value
    b. Button for manual input
        i. Navigates to manual input page
    c. Button for subtract stop

i. Adds to number of subtracted stops

d. Button for undo subtracted stop

i. Deducts from number or subtracted stops

e. Button for route view

i. Navigates to map view of current route

Frequently Asked Questions

1. My Garbo won't pair
    a. Please turn off phone and then turn back on and try again
2. My Garbo won't unpair
    a. Close out of app and then reopen and retry
3. Can I have multiple Garbo's paired to one device?
    a. Currently you may only have 1 Garbo devices paired per device
4. Why can't I access the resident dashboard
    a. Only residents may access this dashboard
5. Why can't I access the collector dashboard
    a. Only collectors may access this dashboard
6. The resident app says the garbage collector should of been here by now
    a. The time displayed is an estimate and your collector should arrive shortly

Appendix: II - "Alternative/ other initial versions of design"

To deliver the most reliable and cleanest design we initially wanted to create a residential garbage bin with the strain sensor and rest of the electronics embedded into the bin itself. Our

team decided to abandon this idea due to high costs of adoption for waste management companies because an existing waste management company would need to purchase a whole new fleet of garbage bins. Our team also was lacking the mechanical engineers and designers to deliver a proof of concept trash bin.

On the routing algorithm side there are a lot of improvements that could be made if we had more resources. One thing we've talked about as future work is implementing a system that makes the trucks prioritize right turns over left turns. There is a lot of literature out there about how much more efficient turning right is. The time spent idling waiting to turn left is wasted fuel and wasted time. Many delivery companies, like UPS and Amazon, build routes so there are very few left turns if possible. We believe that this is one thing that could possibly greatly improve our results if we had access to better mapping software that would allow us to prioritize that while finding distances.

Appendix: III - "Notes"

Our testing for hardware only consisted of one garbage sensors. We were unable to test the ceiling of scalability for our solution. Theorized bottlenecks would most likely occur at the AWS IoT endpoint and trash bin database. Since each device is independently connected to the LTE-M network, our team felt that the wireless technology would be able to scale from very limited adoption and testing to large scale implementation.

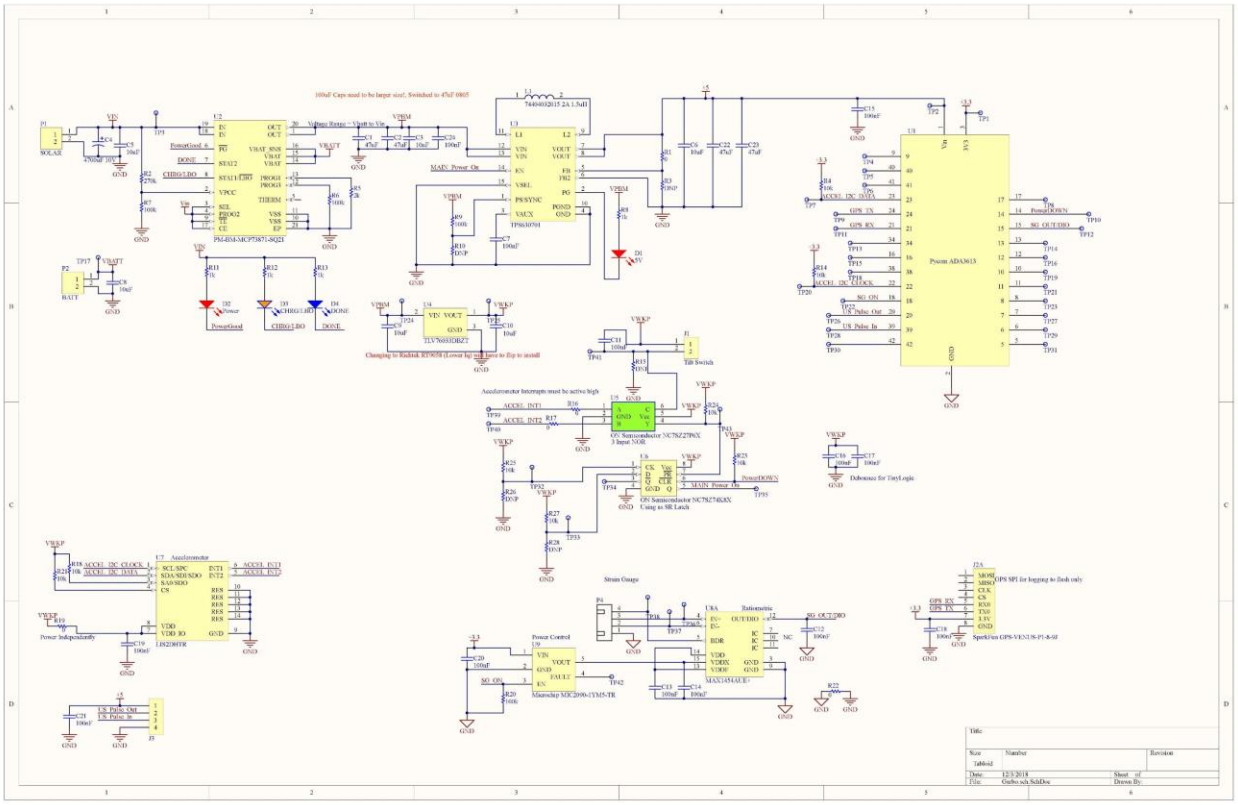
This garbage sensor developed by our team was specifically developed for residential waste management, however the hardware team focused on creating a device that would only need to be modified slightly to be used in an industrial setting. The major considerations would be scaling up and using multiple load cells for the larger weight and size of dumpsters in addition to using multiple ultrasonic distance sensors to measure the larger surface area in a dumpster.

One big issue we hit was bugs relating to the original Cluster/Routing algorithm. The algorithm for building a directed spanning tree is pretty complicated and very hard to actually implement. Thankfully, early last spring we found a Python library that claimed it could do it. It worked well for our initial implementation and even gave us some good results for our spring senior design presentation. We left that semester feeling pretty confident with that part of our design. However, within the first week of the fall semester we had started testing that implementation with larger graphs and found we were getting really odd results. After some testing we found that for graphs that were 1000 or more nodes the library code would create directed spanning trees with cycles, which made it unusable for our purposes. From here we had three options: Fix the existing library, write our own directed minimum spanning tree utility, or pivot and find a new

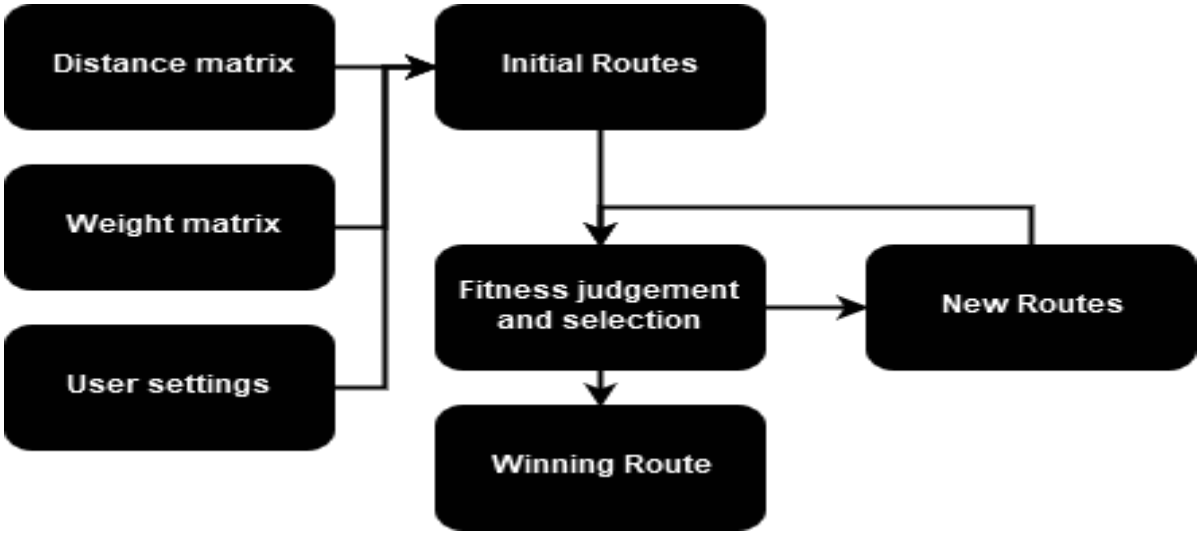
way to build routes. Thankfully Dr. Daniels advised us to look into genetic algorithms and it led us to building our current routing solution.

Appendix: IV - “Hardware / Software Components”

Schematic

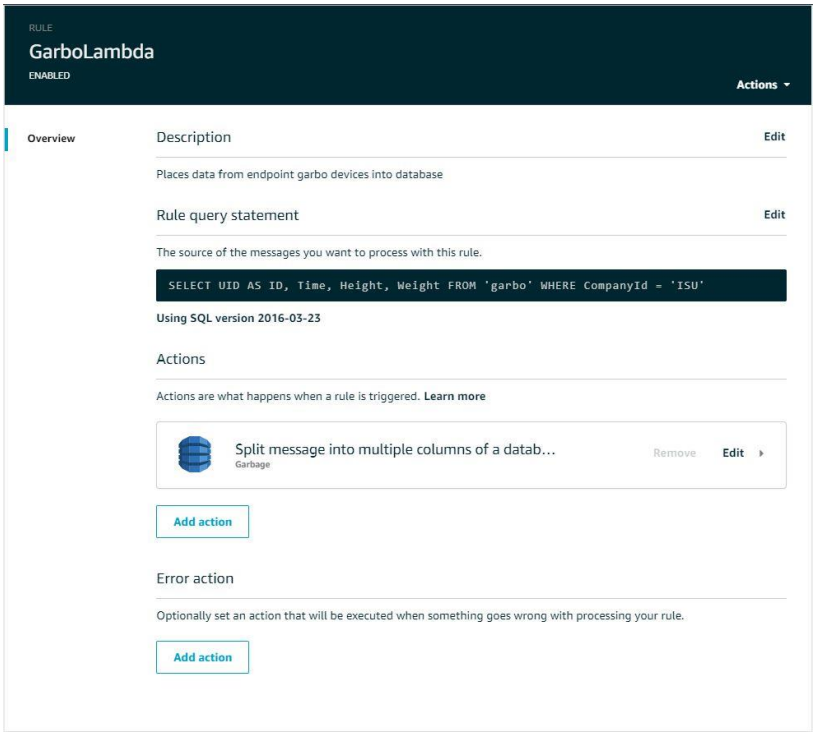


Routing Algorithm

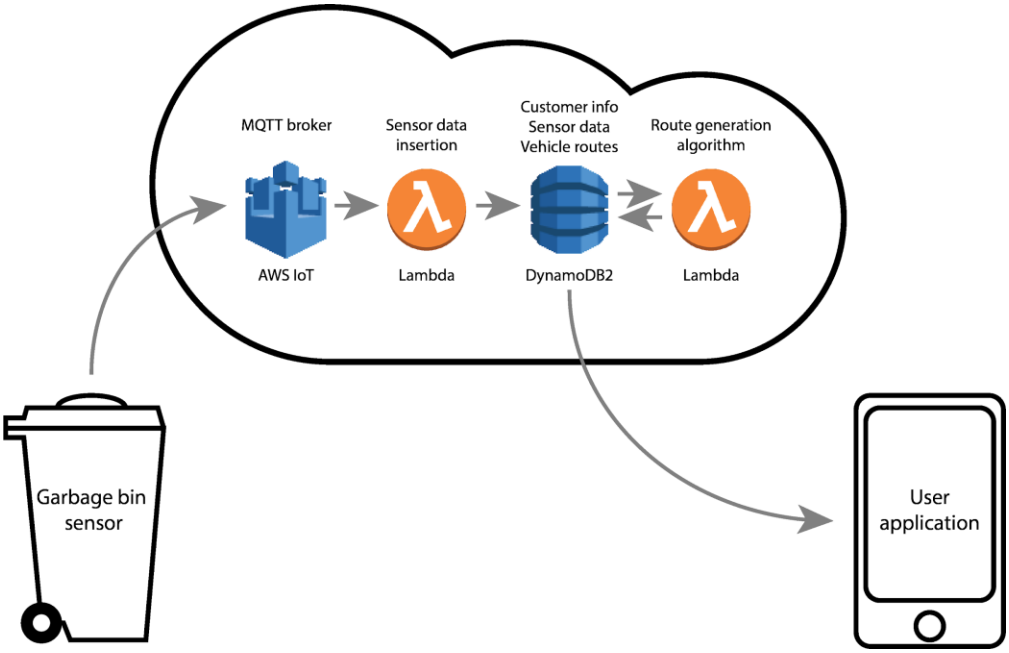


System Connection Code

```
SELECT UID AS ID, Time, Height, Weight FROM 'Garbo' WHERE CompanyId = 'ISU'
```



Full IoT Diagram



User Dashboard