Abstract—In this paper, we present an Internet of Things sensor system to monitor the fill level of municipal trash cans. This sensor system incorporates a cloud service that stores historical data and triggers SMS alerts. Evaluation shows that the system operates reliably in realistic conditions.

Keywords—Internet of Things, smart cities, waste

I. INTRODUCTION

Smart cities have been identified as a promising potential application domain for the Internet of Things, with a wide range of possible services that can benefit city administration and citizens alike [1]. One service that can be provided in a smart city is smart waste management. Public trash cans detract from the surrounding environment when they are full for long periods of time. On the other hand, it can be an expensive operation to send garbage trucks to every trash can in the city; if cans are empty, the journey accomplishes nothing. Cities develop rough algorithms for minimizing cost of various municipal services such as collecting trash, but Internet of Things sensors can improve these services by notifying relevant public works officials when particular trash cans are full.

We begin with background on related work applying Internet of Things concepts to waste collection in section II, discuss the chosen implementation on both the hardware and software side in section III, cover the experimental portion in section IV, and evaluate the results in section V before concluding and presenting ideas for future work in section VI.

II. RELATED WORK

The problem of using Internet of Things technology to optimize waste collection has been studied in the research literature and addressed through commercial products. Some smart trash research considers "pay as you throw" weight-based billing for residential collection, which could motivate residents to reduce their waste [2], [3]. Other research focuses on sorting and properly routing contents for appropriate recycling and other processing [4].

A deployment of smart trash cans in Barcelona has been successful and received considerable media attention [5], [6]. These trash cans sense and report their fill levels, with the goal of allowing the city to plan efficient collection routes.

Enevo markets a smart trash monitoring solution [7]. This system involves a sensor node that monitors fill level, energy usage, and other internal state of the compactor, reporting readings via CDMA or GPRS. The accompanying services provide real-time and historical reports on the data, and directly integrate with waste haulers to automatically schedule optimized pickups.

III. IMPLEMENTATION

A block diagram of the system is shown in figure 1. The system implementation is described in the following sections, including hardware construction and software flow.

A. Hardware

The u-blox C027-U20 microcontroller board [10] is the heart of the sensor node system. It features a NXP LPC1768 MCU with an ARM Cortex-M3 processor, a LISA-U200 cellular module that can communicate over WCDMA/HSPA and GPRS/EDGE, and a MAX-M8Q positioning model based on GNSS signals from GPS/GLONASS/BeiDou/QZSS. It also has a variety of GPIO pins used to interface with peripherals. Notable ports on the board include:

- GNSS antenna: Connects to GPS unit
- Cellular antenna: Connects to cellular modem
- Power supply: Requires 7V to power the system
- SIM: Stores information to access cellular data network
- USB: Interface to program the board

The Grove Base Shield SLD01099P [11] plugged into the GPIO pins of the C027 to facilitate easier interfacing with the peripherals. The shield consists of a number of 4-pin analog and digital JST connectors on the top, with a set of pins on the bottom that matches the headers of the C027. The connectors feature 2 signal lines along with VCC and GND.

We used an ultrasonic distance sensor HC-SR04 [12] to measure the trash level, and a temperature sensor [13] to measure weather conditions and determine if the trash can catches fire. Both sensors were wired to JST connectors on the Grove Base

BigBelly produces compacting trash and recycling stations [8]. These stations use a solar panel to charge an internal battery for communication and compacting. Compaction occurs automatically when the stations are full. They use GPRS communication for monitoring and management. BigBelly stations have been deployed in a number of cities, universities, and other environments, mostly in the U.S.

SmartTrash markets a monitoring system for trash compactors [9]. This system involves a sensor node that monitors fill level, energy usage, and other internal state of the compactor, reporting readings via CDMA or GPRS. The accompanying services provide real-time and historical reports on the data, and directly integrate with waste haulers to automatically schedule optimized pickups.

Smart City Trash Cans

Timothy Ambrose, Corey Ford, and Michael Norris
Shield, with the temperature sensor using a connector linked to an ADC input of the MCU. The temperature sensor only measures up to a maximum of 125°C, but anything at this temperature would clearly indicate a problem.

The board’s voltage regulator required 7V input for operation. A wall-connected AC power adapter sufficed for the lab environment, but another solution was needed for field testing. A USB solar panel and battery were found not to supply sufficient voltage, so we used a rechargeable lead-acid 12V battery alone.

B. Sensor Software

The structure of the code developed to run on the C027 processor is shown in figure 2a. The source is available at https://developer.mbed.org/teams/Garbage-Collectors/code/SLOTrashHTTP/.

At startup, the code initializes the cellular modem and GPS, and connects to the cellular data network. These are simply left connected, although for practical deployment they could be turned off in between data transmissions to save power (see section V-B).

The temperature calculation code in figure 3 was derived using the Steinhart-Hart equation [14]. Distance sensor code uses MBED libraries [15] to measure the time between sending out an ultrasonic signal at 40kHz [12] and sensing the return signal. GPS readings are received using the NMEA protocol over an internal serial bus.

After reading from both sensors and the GPS receiver, an HTTP request is constructed. This consists of an HTML form-encoded body with three parts:

- **name**: Name of the sensor (hardcoded)
- **value**: Distance, temperature, and GPS readings, plus an indication of whether this is the first loop iteration, all encoded in JSON
- **mac**: HMAC computed over the JSON value using a secret key, as hexadecimal

This is submitted as an HTTP POST request to a predefined endpoint /sensor on the application server, using the built-in TCP processing of the cellular modem. After doing so,
/* Calculate the resistance of the thermistor from analog voltage read. */
resistance = (float) 10000.0 * ((65536.0 / analogRead) - 1.0);

/* Convert the resistance to temperature using Steinhart-Hart equation */
temperature = (1/((log(resistance/10000.0) / 3975) + (1.0/298.15))) - 273.15;

The code waits for a certain interval of time (10 seconds in our implementation, but minutes or even hours would not be unreasonable in practical deployment) before repeating the loop.

C. Server Software

The cloud component of the system is a simple web application, developed using Python and the Flask framework [16] on top of a SQLite database. It accepts readings from the sensor, interfaces with Twilio to send SMS alerts, and serves visualizations of historical data to users. The source code for the application is available at https://github.com/coyotebush/trash-map. It was deployed on a Rackspace cloud server instance running Ubuntu 15.04.

Upon receiving a POST request at /sensor, the application first retrieves the database record representing the sensor by the “name” supplied in the request. If the record contains a cryptographic key, the application computes an HMAC over the JSON “value” from the request and compares this against the “mac” from the request; if these do not match, the request is rejected with a “403 Forbidden” error. Otherwise, the value is inserted into the database along with the current timestamp. This process is shown in figure 2b.

After a value is inserted, several of its properties are used to potentially trigger SMS alerts to a preconfigured list of phone numbers. This functionality uses Twilio [17] to provide SMS capabilities, including a phone number from which to send messages and APIs to communicate with the service. First, if the “first” property is set to 1, a greeting message is sent. Then, if the “distance” property is lower than a per-sensor threshold set in the database, and no alert has been sent for that sensor within 10 minutes, a message is sent indicating that “the trashcan is full”.

In addition to accepting requests from the sensor, the application accepts requests from web browsers to show visualizations of the data. A request for / returns a map using Leaflet [18] with a marker at the coordinates of each known sensor (shown in figure 4). These markers contain a link to /graph/<name>, a request to which returns a Dygraphs [19] display of “distance” and “temperature” properties of all records for the particular sensor.

Note that the server does not use GPS coordinates from the sensor, because the sensor node is not expected to move from the target trash can. The GPS capabilities may be useful for implementing zero-configuration deployment or anti-theft measures in the future.

IV. EXPERIMENT

The primary test of this system was conducted overnight at a city park in San Luis Obispo.

A. Project Enclosure

The electronics were housed in a plastic Radio Shack 2x4x6” project box to reduce possibility of contamination. Two
holes were drilled in the box to allow the distance sensor to measure the level of trash. One hole was drilled to allow the cellular antenna attached to the C027 board to fit. Another hole was drilled on the side to allow power and ground lines and a coaxial cable for the GPS. Finally, three holes were drilled in the lid to secure the C027 board to the enclosure via machine screw. This assembly is shown in figure 5.

We superglued magnets to the lid of this enclosure (opposite the distance sensor side) to secure it to the inside of the metal trash can lid. The GPS module was itself magnetic, and attached to the trash can lid separately from the project box. The 12V battery sat outside the trash can for initial testing, and then was later placed between the trash bag and the trash can for overnight testing. The installation is shown in figure 6.

B. Field Testing

We took the sensor node enclosure and battery to Throop Park, San Luis Obispo. Initial testing consisted solely of securing the enclosure under the edge of a trash can lid, and ensuring that the distance to the trash measured correctly. Trash bags were found to occasionally interfere with the sensor reading, necessitating proper sensor placement.

Overnight field testing followed initial field testing. We installed the sensor on a trash can (located at N 35.29564° W 120.67659°) overnight. Several hours into the test, readings stopped being sent to the server. We soldered a switch into the power line, configured a watchdog timer, and reattached the battery, after which the problem did not recur. We removed the sensor during a demonstration the following day.

V. Results and Discussion

A. Distance and Temperature

Figure 7 plots the distance and temperature readings recorded during the overnight test. There is a gap and visible jump in both values from approximately 20:30–21:30, when maintenance was performed on the sensor following a failure.

Overall, the distance reading is steady. There is some oscillation during the night as the sensor detected reflections from different objects in the trash can. During the day, there are minor changes as a small amount of trash was added by park users and wind shifted the contents around. At the very end of the test, we intentionally blocked the distance sensor to trigger an alert.

The temperature reading follows smooth curves throughout the solar cycle. It also reflects realistic observations: the coldest part of the night was just before sunrise around 6:00, and the sun came out briefly from approximately 12:00–13:00 before clouds and colder weather returned. In general, the readings are higher by as much as 5°C than published weather data; this may be explained by mis-calibration of the sensor, insulation provided by the project box, or some combination. However, the readings are still more than adequate for detecting fire, and can be calibrated more carefully, even on the application server if necessary, to serve as a weather station.

B. Power Measurements

Minimizing power consumption is an important goal of such a system, and it is important to know the expected time of life of the sensor. Table I shows measured power draw when the board is in different states. Processor sleep modes and peripheral poweroff were not implemented in the prototype software iteration, but can be added to improve power efficiency in the future. With specially fabricated chips designed to maximize power efficiency and communicate less frequently to the cloud, system battery life can be extended to ten years in ideal conditions, as demonstrated by the Enevo sensor implementation [7].

In the field test, our battery (capacity 5Ah) was nearly discharged after 24 hours. These calculations suggest that with improved power management, the system could operate for several days using the same battery. Since this is similar to the
(a) Distance measurements

(b) Temperature measurements

Fig. 7: Sensor measurements from 24-hour test

<table>
<thead>
<tr>
<th>State</th>
<th>Current Draw</th>
<th>Time in State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle or reading from sensors</td>
<td>86mA</td>
<td>9.97%</td>
</tr>
<tr>
<td>Cellular modem use</td>
<td>130mA</td>
<td>0.28%</td>
</tr>
<tr>
<td>Sleep mode</td>
<td>50mA</td>
<td>89.75%</td>
</tr>
</tbody>
</table>

**TABLE I: Power consumption**

<table>
<thead>
<tr>
<th>Component</th>
<th>As built</th>
<th>Improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>u-blox C027 board</td>
<td>$199.00</td>
<td>$149.00</td>
</tr>
<tr>
<td>Grove Base Shield</td>
<td>$8.90</td>
<td>$0.00</td>
</tr>
<tr>
<td>Temperature sensor</td>
<td>$2.90</td>
<td>$1.00</td>
</tr>
<tr>
<td>Distance sensor</td>
<td>$30.00</td>
<td>$5.00</td>
</tr>
<tr>
<td>12V battery</td>
<td>$20.00</td>
<td>$20.00</td>
</tr>
<tr>
<td>Project enclosure</td>
<td>$10.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>Machine screws and nuts</td>
<td>$0.30</td>
<td>$0.30</td>
</tr>
<tr>
<td>Magnets</td>
<td>$1.00</td>
<td>$1.00</td>
</tr>
<tr>
<td>Cellular SIM card</td>
<td>$10.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>Cellular data plan</td>
<td>$25.00</td>
<td>$1.47</td>
</tr>
<tr>
<td>Rackspace cloud server</td>
<td>$46.75</td>
<td>$5.00</td>
</tr>
<tr>
<td>Twilio service</td>
<td>$2.00</td>
<td>$2.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$282.10</td>
<td>$196.30</td>
</tr>
</tbody>
</table>

**TABLE II: System costs**

typical collection frequency of municipal trash cans, it would be plausible to continuously operate the system by replacing the battery when collecting trash. Alternately, a solar panel of sufficient voltage could be installed on each trash can and eliminate the need for external recharging.

C. Cost Analysis

Cost of building and maintaining the system is another important concern. We calculated the cost of our system as implemented, which includes both fixed costs (to build one sensor) and recurring monthly costs (to maintain the cellular connection and server, the latter of which could even be amortized across multiple sensors). We also researched potential cost improvements that could be to the implementation without significantly altering the design. In particular, the C027 board could be replaced with a variant using the simpler SARA-G350 GPRS modem; the Grove components could be replaced with standard hardware; the cellular data plan could be replaced with a specialized M2M-oriented data plan providing as little as 500KB monthly; and the server could be implemented on a much lower-specification cloud instance. Both actual and improved costs are shown in table II.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a working proof of concept for a “smart trash can” that can sense trash level and notify necessary city workers over the cloud. Ultrasonic distance readings are a reasonably reliable way to gather this data, though they are inconsistent when faced with complex surfaces. With our cost and power analyses, we also determine that this system can be made much cheaper and more power efficient. System cost can be reduced from that of our prototype implementation. Sleep modes can also be implemented, reducing power consumption of the system.

Further field trials would be useful to better assess the reliability of the distance sensor and in turn improve the SMS alerting algorithm. With data from multiple sensors, a possible feature for the application server is the calculation of an optimized and human-readable route that truck drivers can utilize to efficiently collect full trash cans.

ACKNOWLEDGEMENTS

We would like to thank Dr. Shivakumar Mathapathi of Dew Mobility and Dr. Foaad Khosmood for their guidance.
We would also like to thank Dr. Simon Koo of Ananse, Dr. Roland Krause of ICS, Michael Koster of ARM, Dr. Martin Serrano of DERI, and Robert Abatecola for presentations and support during the course. Finally, we would like to thank Greg Hermann and the City of San Luis Obispo for inspiration and funding for this and other “smart city” projects at Cal Poly.

REFERENCES


