15-549: Embedded Systems Design Project Proposal

Team 11

Little Brother

5/9/16

Alex Maeda
Manuel Diaz Corrada
Tom Mayo-Smith
Ramsey Natour

Abstract

The goal of Little Brother was to explore the efficacy of a sensor system built upon the principle of data muling through a smartphone application. This system allows users to gather data from the environment in areas with low connectivity and is cheaper, more power efficient, and easier to set up than systems designed to transfer data through a cellular modem. When users running our app walk near Little Brother, the device establish a Bluetooth® Low Energy (BLE) connection to the user's smartphone and automatically transfers its data logs. When the smartphone has connectivity, it pushes the logs to our server, where users can visualize the data and remotely configure Little Brother devices through our web application. Our findings suggest that such a system can be very effective. While we did not explore the social dynamic, we found BLE to have a long enough range and quick enough connectivity that users walking near the device can reliably act as data mules. We also found the impact to users' data rates to be minimal. Because of this, Little Brother can be used in a variety of applications, including: aiding researchers in developing nations that don't have reliable cellular coverage, gathering data in urban environments to perform anomaly detection and help emergency response teams, and increasing the library of real-world data to which machine learning researchers have access.

Table of Contents

1. Project Description

Little Brother is a remote sensing device designed to gather a variety of data about the environment and push these data to the cloud by crowdsourcing the bandwidth of smartphones around the device. We will develop a smartphone application that allows Little Brother to connect to nearby phones running the app through BLE. Smartphones with our app act as “data mules”, automatically collecting logs from devices in the vicinity of the phone and pushing these logs to our server. Our server will store data logs in a database and provide a webapp to make configuration changes to Little Brother, see the location of our devices, and visualize the data.

Our crowd-sourcing capabilities allow us to both gather data in near real-time from areas traditionally out-of-reach due to lack of WiFi access and operate at a lower power consumption and cost. We envision Little Brother as an economic solution to a wide range of applications: from industrial and environmental sensing to traffic analysis, air pollution monitoring, and general anomaly detection. Our end goal is to build a system able to make useful inferences about the environment based on a large amount of sensor data and make these observations freely available to the public.

2. Design Requirements

Our project consists of three main subsystems: Little Brother , our smartphone app, and our server. Little Brother will gather data from its array of sensors, save these data, and scan for smartphone devices in the area to upload these data to the cloud. Our smartphone app will facilitate communication with the device, push logs to our server, and allow registered users to make configuration changes to the device. Our server will store sensor logs for each Little Brother device, provide a webapp to interact with the data, and also make remote configuration changes.

2.1 The Little Brother embedded device shall:

a. collect data on surrounding environment using sensors
b. push data to cloud by automatically making BLE connections to smartphones running the app
c. be able to operate 24/7
d. be easy to set up and configure

2.2 The smartphone app shall:

a. automatically connect to Little Brother devices through BLE
b. download sensor logs from devices and push sensor logs to the server
c. allow for users to register new devices
d. allow users to view raw data or analysis of data
e. allow users to customize the operation of the app, including when to upload logs
f. have minimal impact on users' cellular data and battery life

2.3 The server shall:

a. store sensor logs to a database
b. provide a webapp to allow users to view raw data and analysis of data
c. show the location of Little Brother devices
d. allow for users to be registered to make configuration changes to Little Brother devices

3. Functional Architecture

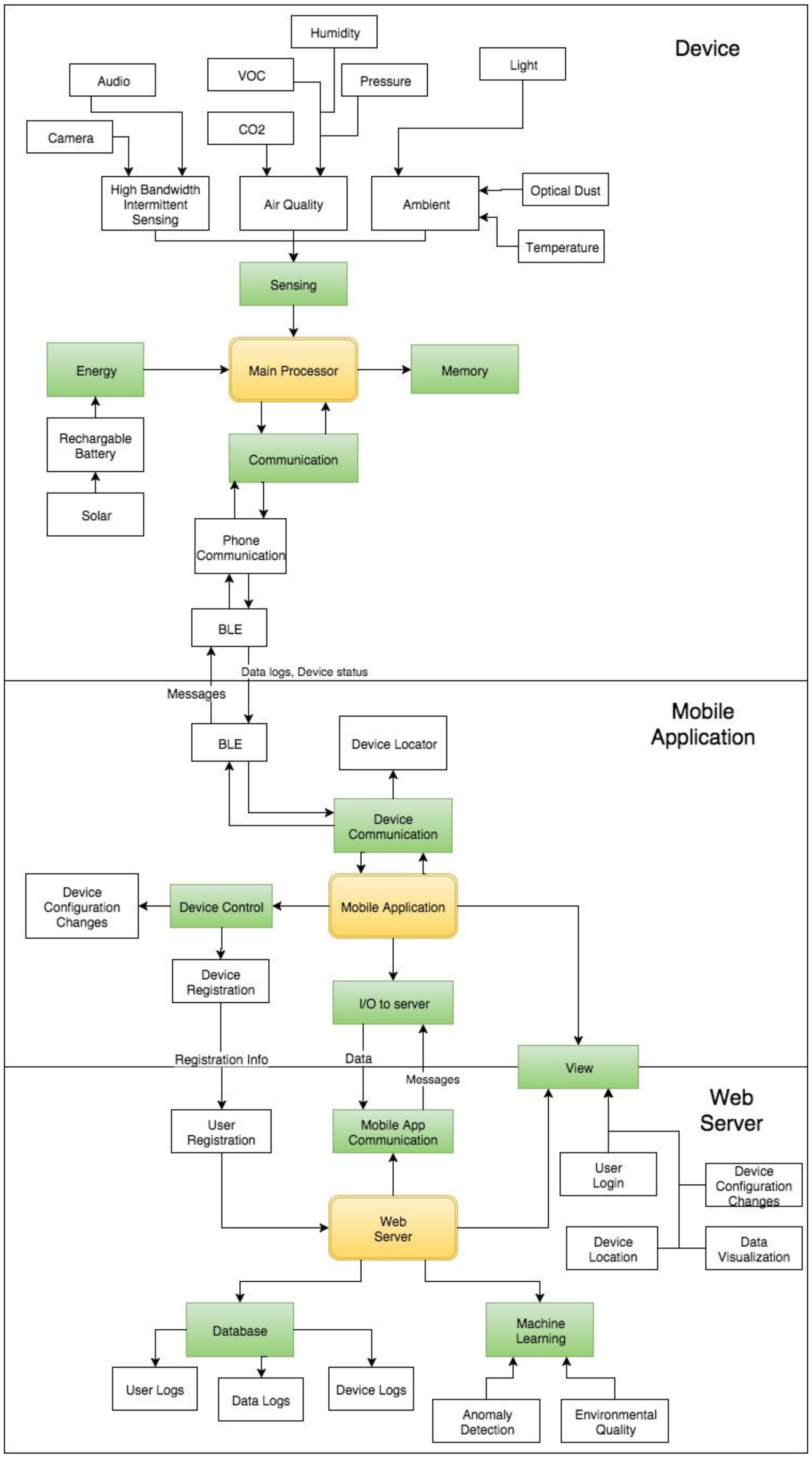

Figure 1: Functional Diagram

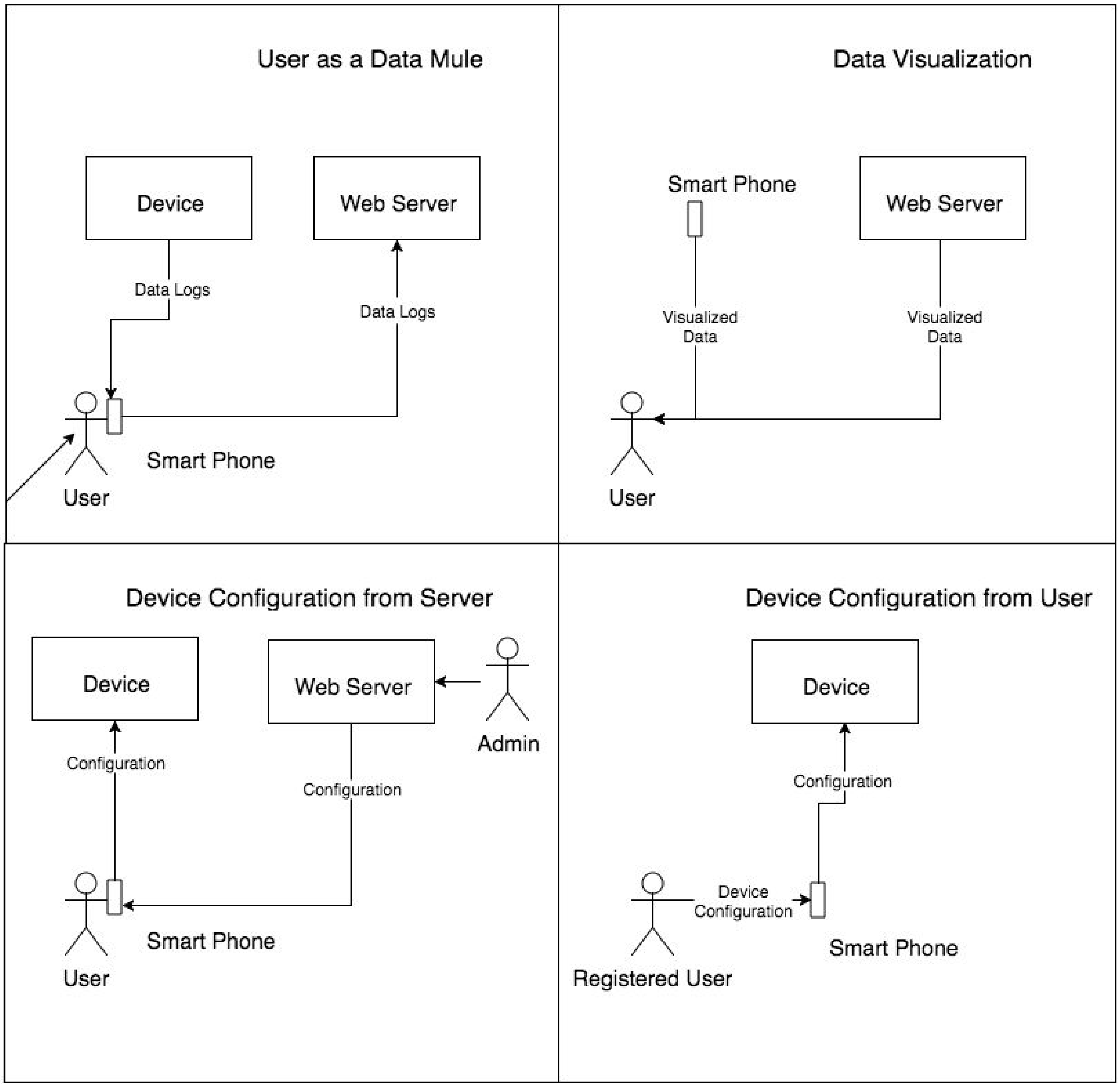

Figure 2: 4 User Interaction Use Cases
Top Left: User acts as passive data mule when near the device
Top Right: User uses the smartphone app to view data received from the device
Bottom Left: User passively transmits a remote configuration update to the device
Bottom Right: Registered user makes configuration changes from the smartphone app

4. Design Trade Studies

4.1 Project Idea

Little Brother has gone through many changes throughout the design process. The project began as an idea for a sensor hub that the user would attach to his or her mobile device in order to enhance its sensing capabilities and give app developers more hardware libraries to work with. App developers would be able to write apps using hardware that doesn't yet exist on the mobile device and these apps would act as an incentive for users to purchase hardware enhancements. We decided that the scope of this project was too broad and decided to focus on creating a sensor hub that would exist separate from phones but would still interact with them in some way. After reading about issues relevant to the city of Pittsburgh, we decided to go with data muling through the smartphone with the aim of creating a better solution for localized air pollution monitoring. Localized air pollution is air pollution that differs significantly from regional averages over a relatively small area. It is particularly relevant in Pittsburgh due to its history and unique landscape [1]. Monitoring stations are expensive to set up and require a cellular reception to move data [2]. Since populated areas are also areas where policy makers care more about air pollution and since air pollution affects all people equally, we saw this as a good opportunity for data muling.

4.2 Embedded Device

We divide the design decisions for the device into two subcategories: sensor choices and choice of SoC.

4.2.1 Sensor Choice

Since Little Brother will be able to support a variety of sensors, we decided to choose 3 sensors for our prototype that covered SPI, I2C, and analog communication. Each sensor further needs to be powerable by our 3V battery without voltage regulation, be simple to modulate (for demo purposes), be simple to calibrate, and not interfere with each other's operation.

Table 1: Sensor Specifications

Category	ADXL345	TMP102	OPT101
Measurement	Acceleration	Temperature	Light intensity
Price	7.17$	.5$	3.51$
Communication standard	SPI	I2C	Analog
Operating range(V)	2-3.6	1.4-3.6	2.7-36
Current	40uA	120uA	10uA

To satisfy these requirements, we chose to include a light sensor, a temperature sensor, and an accelerometer. These sensors are easy to modulate, do not interfere with each other, and represent the types of data Little Brother might be used for. We chose the Texas Instruments OPT101 photodiode to measure light, the TMP102 temperature sensor, and the Analog Devices ADXL345 accelerometer. These specific sensors were chosen among numerous alternatives because they were highly rated on hardware forums and were within our budget.

4.2.2 SoC Choice

For our choice of SoC, we researched several options and evaluated their tradeoffs. The boards we considered were:

Nordic NRF51822
Intel Edison
TI CC2541

Table 2: SoC Comparison

Category	Nordic NRF51822	Intel Edison	TI CC2541
Price	4.95 Dev Kit: 39.00	112.44	6.18 Dev Kit: 99.78
Processor	ARM Cortex M0	Intel Atom	8051 Microcontroller
Storage	128kb Flash 16 kb RAM	1 GB RAM 4 GB Flash	128 kb Flash 8 kb RAM
Power Usage	2.7 mW BLE	680 mW on WiFi 88 mw Idle	400 days
Connectivity	BLE	Wifi Direct,BLE	BLE
Sensor Capabilities	32 GPIO SPI/UART/I2C	40 GPIO pins SPI/UART/I2C	23 GPIO SPI/UART/I2C
Developer Tools	Extensive support and documentation, dozens of tutorials and code examples	Programmable in Python/Ruby as well as C/C++, Cloud Analytics support	Library for interacting with BLE hardware, tutorials and examples, BTool debugging tool

These options were chosen because they were the best representatives of our selection criteria. Our first tradeoff was to consider our communication method. The options we considered were BLE, Wifi Direct, and Zigbee. Between these, BLE was chosen because it allows the device to operate at a lower power point, was generally associated with cheaper components, and has a quick connection time between the central and the peripheral.

We ruled out the Intel Edison because it has excess processing power and features than we need for our use case. These drove up both the price and the power consumption. Between the nrf51822 and cc2541, the nrf51822 was chosen for several reasons. The device was cheaper and used less power than the TI equivalent. The SoC is more mature, with extensive documentation, support, and tutorials covering our use cases. The nrf51822 also allows us to purchase a Developer Kit designed to work with the tutorials and support all of the sensors we are considering. The Developer Kit will aid us in debugging and will provide a fallback in case we run into hardware problems on our custom PCB.

4.3 Mobile Application

Our primary choice for the design of the mobile app was to develop for Android, iOS, or both. We chose to develop for Android only because our team has prior experience developing Android applications and because there is extensive documentation and examples for developing BLE apps. To ensure we have enough time to demonstrate proof of concept, we elected to develop solely for Android. We targeted Android 4.1 because the newest API level contains a more advanced API for interacting with BLE devices.

The mobile app design is structured to maximize the availability of useful information to the users. From the main activity, users can see nearby devices, view the logs currently stored on his or her phone, perform first-time registration of devices, and make changes to the app settings. The activities gather data from a service running in a background thread that scans for Little Brother devices and attempts to download/upload logs on a duty cycle. We chose this design in order to separate the functionality we want to happen persistently from the functionality we need when a user is actively using the app.

4.4 Web Application

For our webserver, we chose to run on Amazon EC2 over several alternatives such as Heroku and Microsoft Azure because our team has prior experience developing web apps on EC2, AWS is a widely used and mature, and because the free tier of service is easy to set up and is sufficient for our purposes. We chose to develop the web app in Django because our team has prior experience in Python/Django and because Django is a widely used framework with many resources available to developers. Our webapp will store data in a MySQL database. MySQL is easy to configure with Django on an EC2 instance, supports fast queries, and has extensive developer support.

5. System Design

Little Brother is composed of three major subsystems that interact with each other to grab data from the environment and store it in the cloud. Each subsystem is composed of modules that help that component perform its task.

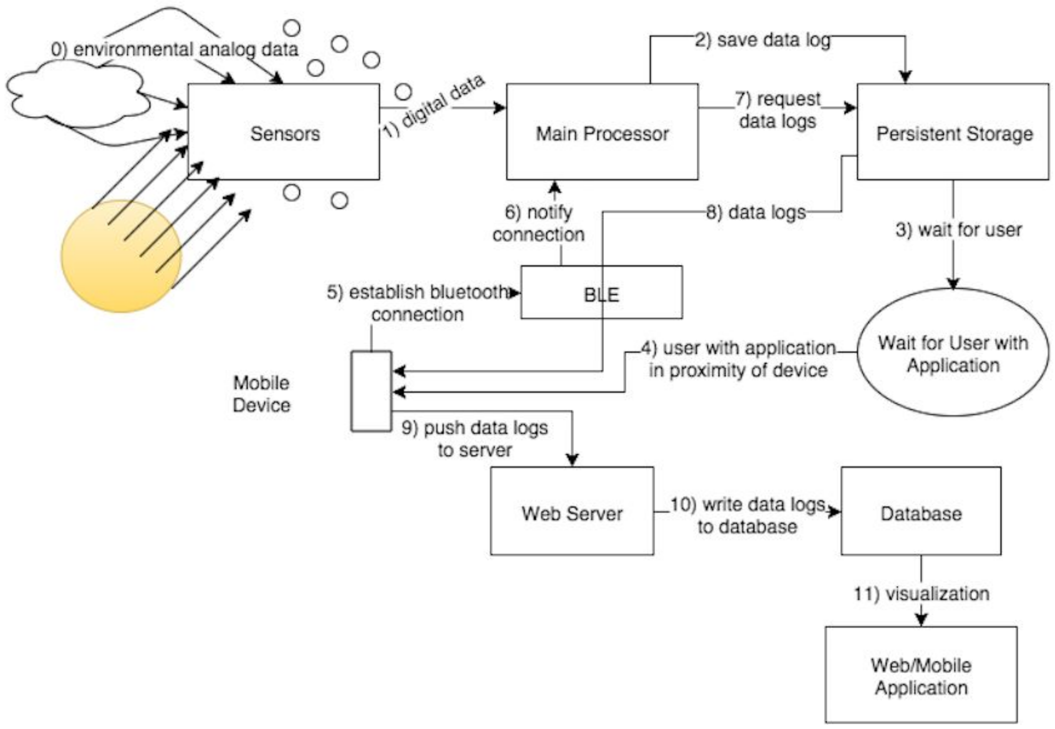

Figure 4: Flow of Data

5.1 Little Brother Embedded Device

Little Brother is responsible for gathering data from its sensors, storing these data in logs, monitoring for available data mules in the environment, pushings its logs to available data mules, adjusting its configuration based on remote settings, and regulating its state to ensure 24/7 operation.

5.1.1 Sensing Capability

Little Brother has an array of sensors that can be divided into two categories: ambient sensing and sensors specific to the use case of the device. Ambient sensors allow Little Brother to collect generic information about the environment in the form of temperature, light level, and audio (with decibel sensing). These general environmental data provide useful information to our users, who will be able to read the data directly on their phones and remotely through our webserver. These data are also useful for regulating the lifecycle of the device itself. For example, through an ambient light level sensor Little Brother could put itself to sleep when it is dark and not many users are moving near the device and wake up when the light level increases. Other sensors can be added to target specific use cases and are not constant among all devices.

For our demo, we will be using a light sensor, a temperature sensor, and an accelerometer. Although our design allows for more advanced capabilities, we chose these sensors because they are easy to modulate and cover a breadth of hardware communication methods.

5.1.2 Log Management

Little Brother uses a Nordic nrf51822 SoC with built-in RAM, flash memory, and a BLE chip. Our microprocessor will poll its sensors at a configurable interval and store the data in logs located in RAM. Logs will be broadcasted in FIFO order and when device storage is full the oldest logs will be overwritten first. This ensures that the logs most likely to have been transmitted are overwritten first. This also ensures that, barring that case, newer data take priority over older data since they are generally more useful to the user.

5.1.3 Data Mule Interaction

Little Brother will utilize the Bluetooth Low Energy protocol for scanning for available data mules in its environment. Using BLE minimizes power consumption on the device [3]. A BLE link takes roughly 10 ms to connect and can happen without user intervention, making it ideally suited for our use case.

Our BLE stack will be constructed with two services: a device info service and a log service. The device info service will broadcast device metadata needed by the smartphone to upload logs to our server. Characteristics of this service include: device name, device id, sensors, and configuration revision. Device info characteristics begin at default values on first time bootup and can be written to once by a client for new device registration. The log service will broadcast the oldest available log that has not been transmitted. When a log has been successfully read, Little Brother will swap the currently broadcasted log for the next oldest untransmitted log and notify the smartphone that a new log is available. This model allows Little Brother to quickly push as many logs as it can while a mule is in range.

5.1.4 Operating Condition Management

Little Brother will have two means to affect change in its operating behavior. Based on its current power state, available internal storage, and availability of data mules, Little Brother will be able to change the rate at which it is polling its sensors. For example, when sunlight is readily available and many data mules are in the area, Little Brother will poll its sensors at a quicker rate. Alternatively, during night hours when few data mules are available and light is low, Little Brother will will gather data and look for mules at a slower rate to conserve power and internal storage. Little Brother will also be configurable remotely through a web server interface by users with admin privileges and from smartphones registered to that device.

5.2 Smartphone Application

The smartphone application satisfies the data mule requirement. The application is responsible for automatically connecting to Little Brother devices in the environment, pulling data logs from these devices, pushing these data logs to cloud storage, sending configuration updates to Little Brother, and viewing data logs locally.

5.2.1 Device Communication

Our smartphone app will communicate with Little Brother through the BLE hardware device on Android phones with API level 21 or higher. If a configuration update is bound for the device, the app will also upload the new configuration details to the device. For unregistered devices, users can perform a first-time registration by writing to the device info characteristics of the device.

5.2.2 Server Communication

The app will communicate with the server in two directions. In one direction, the app will upload logs directly to the server and register new devices. In the other direction, the app will request for new configuration updates so that they can be applied to the device.

5.2.3 Data Interaction

The smartphone app will allow users of the app to view data logs gathered from the device. This is useful in cases where Little Brother is intended for private use or cases where Little Brother is intended to provide data in locations where cellular coverage is also sparse, such as hiking trails.

5.2.4 Configuration Updates

The smartphone app will allow for users to be registered to make configuration changes to specific Little Brother devices from their smartphone. This allows for use cases where Little Brother is intended for private use.

5.3 Server

The server provides a central endpoint for data gathered from Little Brother devices distributed over a potentially large area. The server is responsible for storing data and user logs in a database, providing a web app to visualize the data and the location of Little Brother devices, allowing admins to make remote configuration updates to devices, and running algorithms to provide automated analysis of data where appropriate.

5.3.1 Database

The web server will store logs generated by Little Brother devices and from user activity on the smartphone app to a database. The server will throw out duplicate and corrupt logs before writing to the database. The server will also ensure that database transactions are performed atomically.

5.3.2 Web Application

The web application will provide a frontend user-friendly interface for our device. The webapp will use the Google Maps API to show a map with Little Brother devices marked as pins. The webapp will also allow for different data gathered over time from Little Brothers to be visualized in the form of graphs and tables. This webapp will be mobile-friendly so that users can also access the map and graphs from their smartphones. To help minimize overhead in managing Little Brother devices, the web app will provide an authentication system and an interface to allow registered users to make changes to how Little Brother devices are operating in the environment. The webapp will ensure that data gathered from public devices is easily accessible to the public.

5.3.3 Data Analysis

Our server will have the capability to run machine learning algorithms on the data being collected. We will use the quantity and cohesive nature of the data collected to train our algorithms to make useful inferences from the data and detect anomalies in the environment.

6. Project Management

6.1 Schedule

Table 3: Schedule

Date	Objective
2/28	Eagle schematic of board completed, research API's, design communication protocol between systems
3/13	Complete board assembly and hardware debugging, smartphone app design
3/27	beta app completed with data muling capability, device can transmit logs to phone, simple server setup
4/10	smartphone app completed, functional web app completed, data from device should be visual
4/24	device dynamically changes configuration, remote configuration updates allowed, private crowdsourcing allowed, web app completed, machine learning component added
5/1	field test, final debugging

6.2 Team Member Responsibilities:

1. Alex Maeda
    a. Primary Responsibility: Smartphone development, Webapp development
2. Manuel Diaz Corrada
    a. Primary Responsibility: Hardware
    b. Secondary Responsibility: Project Management/Documentation
3. Tom Mayo-Smith
    a. Primary Responsibility: Firmware
4. Ramsey Natour
    a. Primary Responsibility: Firmware/App Integration
    b. Secondary Responsibility: Webapp development, Smartphone development

6.3 Budget

Table 4: Budget

Description	Quantity	Unit Price	Total Price
J-Link 10-pin Needle Adapter (8.06.04)	1	$98.00	$115.30
SENSOR TEMPERATURE SMBUS SOT563	2	$2.05	$4.10

IC ACCEL SPI/I2C 3AX 14LGA	2	$7.17	$14.34
IC AMP MONOLTHC/PHOTOD 8-SOP	2	$12.19	$24.38
RES SMD 10.5KOHM 0.1% 1/16W 0402	25	$0.95	$23.75
1µF ±20% 6.3V X5R Ceramic Capacitor -55°C ~ 85°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	5	$0.11	$0.55
0.10µF ±10% 16V X7R Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	20	$0.10	$2.00
DEV KIT FOR NRF51 SERIES	2	$39.00	$78.00
DIODE SCHOTTKY 30V 1A SMINI2	1	$0.16	$0.16
Coin Cell Battery Holder - 20mm	1	$0.95	$0.95
4.7µF ±10% 6.3V X5R Ceramic Capacitor -55°C ~ 85°C Surface Mount, MLCC 0603 (1608 Metric) 0.063" L x 0.031" W (1.60mm x 0.80mm)	5	$0.11	$0.55
BALUN TRANSFORMER W/FILTER CSP	1	$0.52	$0.52
ANT 2.4GHZ 802.11 BLUETOOTH SMD	1	$3.16	$3.16
switch	1	$0.95	$0.95
20pF ±5% 50V C0G, NP0 Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	4	$0.10	$0.40
12pF ±1% 50V C0G, NP0 Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	5	$0.31	$1.55
32.768kHz ±20ppm Crystal 12.5pF 90 kOhm -40°C ~ 85°C Surface Mount 2-SMD	2	$0.76	$1.52
12MHz ±20ppm Crystal 8pF 100 Ohm -40°C ~ 85°C Surface Mount 4-SMD, No Lead (DFN, LCC)	2	$0.73	$1.46
47pF ±1% 100V C0G, NP0 Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0603 (1608 Metric) 0.063" L x 0.032" W (1.60mm x 0.81mm)	6	$0.43	$2.58
1µF ±10% 10V X7R Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0603 (1608 Metric) 0.063" L x 0.031" W (1.60mm x 0.80mm)	4	$0.10	$0.40
10000pF ±10% 50V X7R Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	4	$0.10	$0.40
4700pF ±10% 25V X7R Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	3	$0.10	$0.30

2200pF ±10% 50V X7R Ceramic Capacitor -55°C ~ 125°C Surface Mount, MLCC 0402 (1005 Metric) 0.039" L x 0.020" W (1.00mm x 0.50mm)	3	$0.24	$0.72
IC SOC 2.4GHZ 128K FLASH 48QFN	2	$4.11	$8.22
RES SMD 100K OHM 5% 1/10W 0402	3	$0.10	$0.30
FIXED IND 10UH 50MA 900 MOHM SMD	3	$0.14	$0.42
FIXED IND 15UH 220MA 900 MOHM	3	$0.27	$0.81
Custom Printed PCB	3	$36.66	$110

6.4 Risk Management:

Since this project is full-stack in scope and has many components, we can mitigate risk by dropping features that do not interfere with our basic workflow. We can drop the private smartphone interaction, machine learning, remote configuration updates, and dynamic power management without cutting off our ability to gather sensor data and data mule to push this data to cloud storage. We have scheduled meetings twice a week—once with a TA and once with each other--to ensure each member is held accountable for his responsibilites. We have divided up the work based on our individual strengths and programming experience. Alex and Ramsey are able to use the web app component of our project as their final project for 15-437 Web Application Development, helping ease the workload later in the semester. Tom has previous experience with machine learning and firmware and Manuel has more experience working with hardware than the other members of our group. All four of us have focused on software development and we feel capable to solve the technical challenges we will face.

7. Conclusions

7.1 Lessons Learned

Over the course of our project, we encountered multiple setbacks. We learned that hardware needs to be done much earlier than software because it is much more time consuming to debug. Our initial PCB design failed and we didn't have time to revise it and order another one. We constructed some breakout shields for our dev board that had a few sensors sautered on, however, they were difficult to debug because it was hard to determine whether the hardware was malfunctioning or we were reading over the communication channel incorrectly. This resulted in us having to resort to the onboard temperature sensor of our dev board rather than the sensors we built. Firmware proved a steeper learning curve than we originally anticipated and we were only able to achieve basic functionality. Because of this, we scrambled at the last minute before our demo and ended up without communication between the device and the app. Android has a more complex interaction with the Bluetooth hardware and the communication from the device to the phone took longer to debug than we expected.

7.2 What Would You Do Differently?

If we could go back, we would ensure that hardware got completed much earlier than we budgeted for. Many of our issues stemmed from hardware problems that we could not rectify at the last minute. We would also have spread out the work for the project more because we were not able to complete the backlog of features we originally planned for.

7.3 Future Work

There is a lot of potential for data muling as a design principle to gather data from IoT devices. We demonstrated that the BLE protocol has sufficient range and bandwidth to push logs onto the phone in the time it takes a user to walk by the device. We did not have time to perform stress testing and see how Little Brother fares out in the environment or to analyze the social dynamic of getting people to volunteer as mules. Our prototype uses one basic sensor to demonstrate functionality but Little Brother's selling point it its ability to have its sensors be configured to solve a real-world problems that users care about. We also wanted to analyze a distributed data structure application where devices can also push logs to each other in order to minimize the time it takes to get the log pushed to a phone but did not have time.

8. Related Work

There are many devices similar to Little Brother, however, we did not find any direct competitors that compete with our end-to-end design. Competitors we saw typically implemented only a subset of our system [4].

8.1 Estimote

Estimote is a BLE beacon that uses advertisements to monitor users and incentivize them to certain behaviours. BLE beacons such as these are related to our project in the sense that their intended behavior is to be picked up passively without user intervention. However, the key difference for us is that we consolidate multiple types of data onto one device and we connect to the device with the intention of uploading the data, not redirecting user behaviour.

8.2 Sensoro Yunzi

The Yunzi uses similar sensors as Little Brother to collect data from the environment and offload them a mobile device. What the Yunzi does not do is connect to any device running the app with the goal of crowdsourcing pushing these data to the cloud. This is a core feature of Little Brother and gives us an advantage is turning data into useful analyses.

8.3 mCrowd

Through mCrowd people can utilize the sensors on their smartphones to complete crowdsourcing tasks set up by other users. We target a slightly different use case. Our app looks for sensing devices in the environment and automatically uploads the logs to our server. Users who would put up a crowdsourcing task could instead use our webapp directly to get the data that they want.

8.4 Park and Heidemann of USC

Implemented two Wide Area Sensor Networks (WASN) which pushed data via mobile phone data mules [5]. The first WASN monitored an oil field and explored how well a sensor network built on data muling would perform compared to SCADAs. The second WASN monitored an urban environment and explored how well this system could monitor trashcan overflow in the city. These systems were successful, albeit narrower in scope than Little Brother.

9. References

1. Balmes, John R., M.D. "The Geography of Air Pollution: Where Are Pregnant Women and Their Children at Risk?, IoES Event." *UCLA Enviorment and Sustainability*. N.p., n.d. Web. 13 Feb. 2016.

2. R, Micheal. "Exploring Air Quality Monitoring Using Intel® Edison." *IoT* -. N.p., 10 Apr. 2015. Web. 13 Feb. 2016.

3. Shah, Rahul C. "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks." (2008): n. pag. Web. 13 Feb. 2016.

4. Chong, Suan Khai. "Context-Aware Sensors and Data Muling." *2Analytical Services Development Biology PIRVic,* (2007): n. pag. Web. 13 Feb. 2016.

5. Park, Unkyu. "Data Muling with Mobile Phones for Sensornets." *ISI Technical Report ISI-TR-673* (2011): n. pag. Web. 13 Feb. 2016.