MeterMinder

Managing Home Energy and Water Usage

Final Report

ECE 1778 - Creative Applications for Mobile Devices

Main Report: 1,982 words

Apper Context Section: 463 words

Trevor Plint (998920057)

Cai Durbin (998689174)

Sam van Berkel (999518641)

April 2013

Introduction

Buildings use almost a third of all energy consumed in Canada. MeterMinder is an application designed to give home owners feedback by connecting their utility consumption with occupant behaviour and home improvements.

The goal of the app is to provide home owners with tools for reducing their energy and water consumption. This is done by allowing the user to record and analyze utility meter readings, as well as collect information about devices within the home and how frequently they are used. The motivation behind this application is twofold: to help individuals understand and reduce their utility consumption, and to build a large dataset of home energy use data for academics and policymakers.

Overall Design

Figure 1 shows the block diagram on which we based the design for the MeterMinder app.

The core of the application is the set of four “managers” whose purpose it is to manage the collection of data. In this version of the app, these data are readings (associated with meters), device usage events (associated with devices), daily mean temperatures (weather) and occupancy records (location).

The principal means for entering utility consumption data is through the app’s UI, in the form of manual meter readings. The exception is for Toronto Hydro smart-meters, which automatically send hourly consumption data to a customer-accessible website once a day. In this case, our application automatically collects these readings via our own webservers.

Device and usage data is entered using Near-Field Communication (NFC) technology. Using NFC tags minimises the work associated with logging device usage. A user simply has to tap their phone against the NFC sticker to complete the process.

Weather and location data are automatically collected in the background. We collect daily mean temperatures for the user’s home location from openweathermap.org. Currently this data is used to partition heating loads and model gas usage. We also monitor the user’s location once an hour and determine if the user is “home” or “not home”.

Once the data has been collected, it is used by the data processor block. This block is a set of classes that handle the analysis of the data. These analyses typically involve the use of several of the data types. The end results are useful statistics and charts.

The final component of the app is the tips manager. The tips manager uses the analyses to construct a set of actionable items for the user. These items include relevant background information and a summary of the expected impact on annual utility usage and costs.

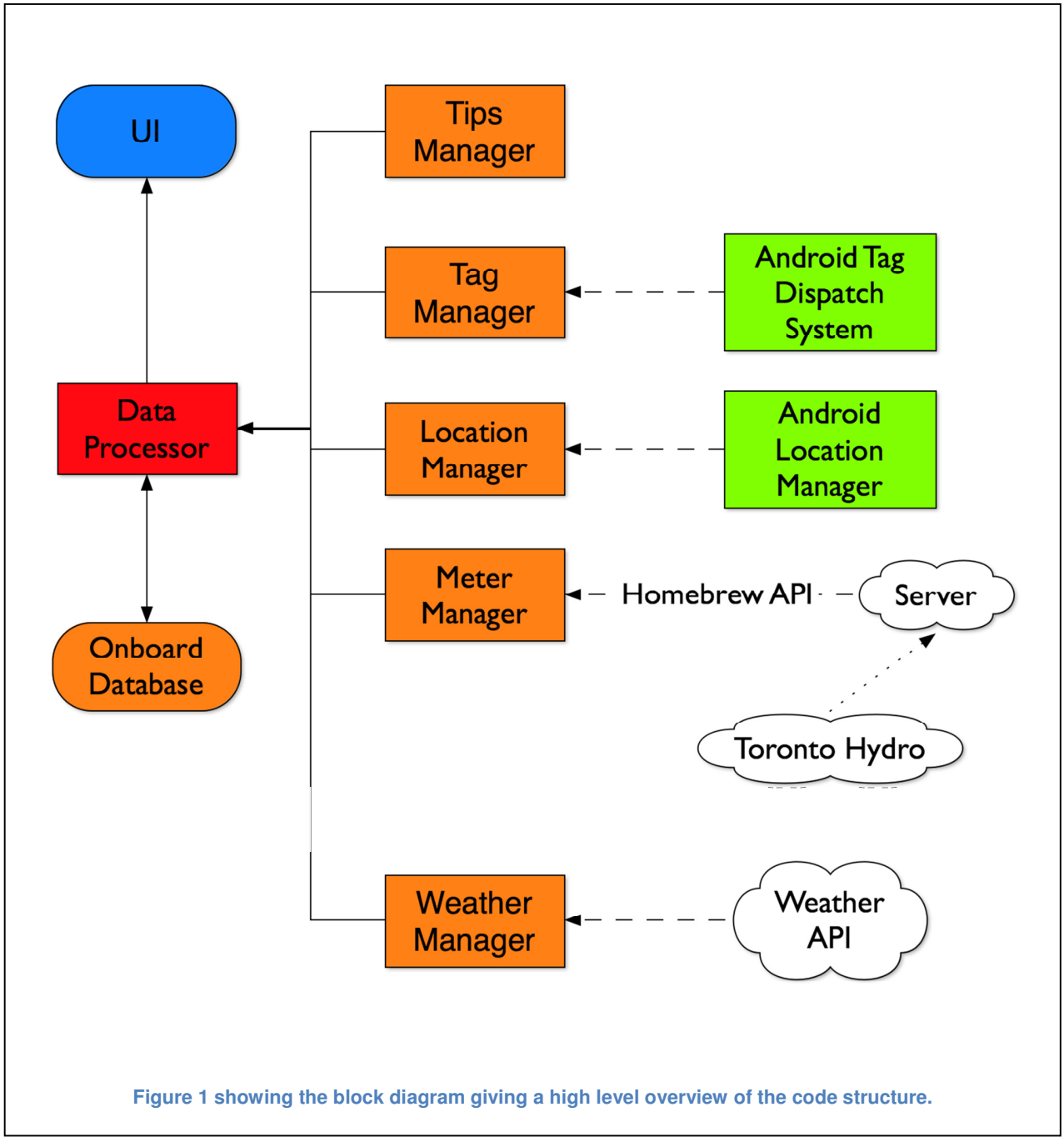

Figure 1 showing the block diagram giving a high level overview of the code structure.

Functionality – Web

Our website is running on the Heroku platform and is accessible at http://damp-mountain.herokuapp.com. Its main purpose is to allow users to register their Toronto hydro details with us, manage the collection and storage of the associated meter readings and serve them in a convenient manner to our mobile application.

Given this goal, the main website's user interface simply allows new users to sign up and then displays two summary graphs of their readings. These pages are shown in Table 1. Each page has an HTML version (which are accessed by browsers) and JSON version. The mobile app uses this JSON API to check which users are signed up and download the appropriate readings.

The second component of the server infrastructure is a Ruby script which gets the user list from the website and copies the meter readings for each user from Toronto Hydro's server to our website. This script runs as an hourly scheduled task on Heroku.

Table 1 showing screenshots from the MeterMinder website.

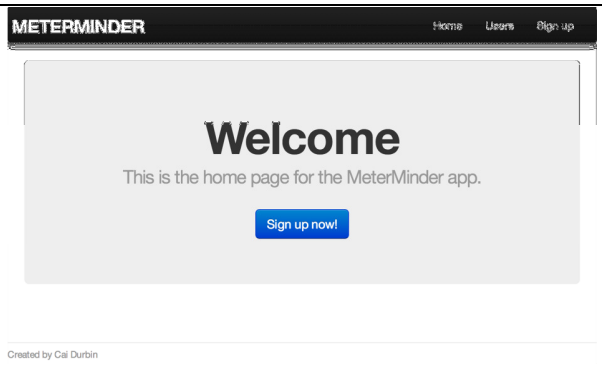

Figure 2 showing the homepage.

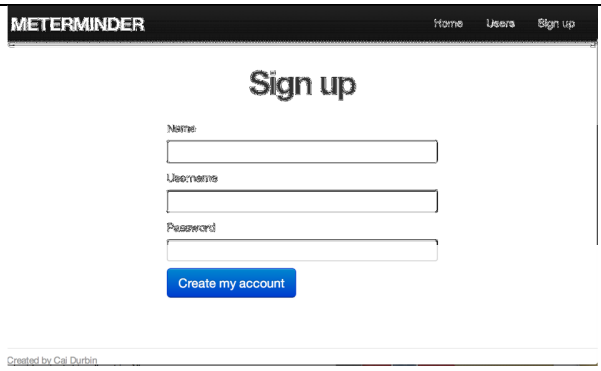

Figure 3 showing the user signup page.

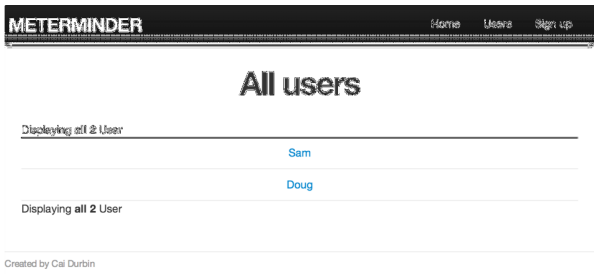

Figure 4 showing the list of users currently signed up.

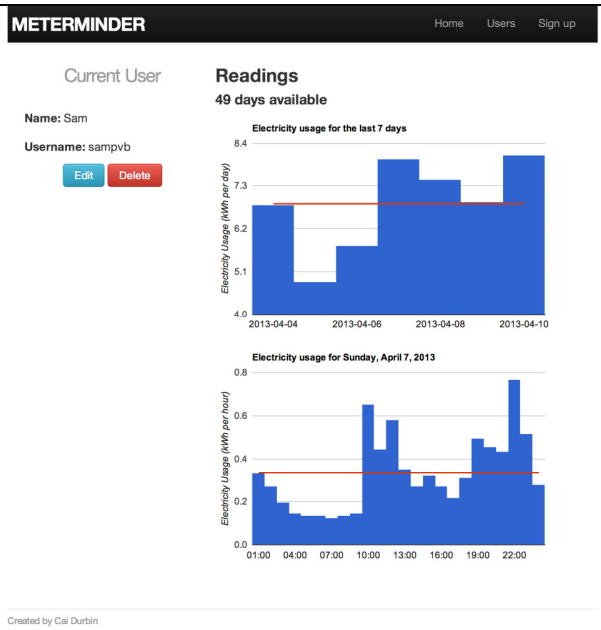

Figure 5 showing the users "profile" page.

Functionality - Android

The app is split into four tabs as shown in Figure 6. These are, from left to right, the meters, devices, tips and settings tabs. The plus button in the top right allows the user to add new meters or devices, along with logging new meter readings and device usage (Figure 7).

When adding a new meter (Figure 8), the user is first asked to select the type and associated units. The available options are "Electricity", "Toronto Hydro", "Gas" and "Water". A name for the meter is automatically generated from its type, but can be changed. If the Toronto Hydro type was selected, the user is also asked to enter their username and password. When the user clicks "Save" the app checks that the provided Toronto Hydro credentials have also been added to our website.

Making a new reading (Figure 9) first requires the user to select the meter. The current time and date are automatically entered, but these can be changed by clicking the appropriate button. The user then enters the reading and clicks save.

Once a reading has been successfully saved, a summary page (Figure 10) is displayed which gives some quick facts about the new reading relative to previous readings. These

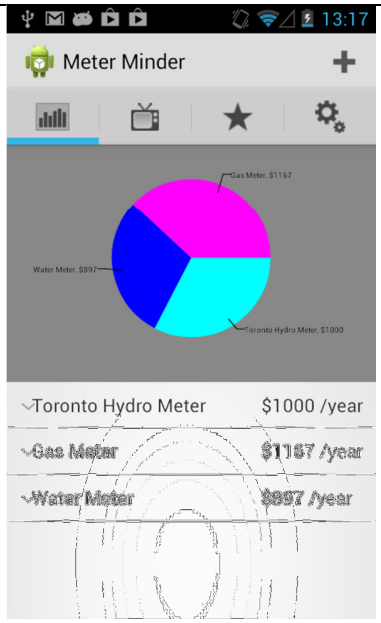

Figure 6 showing the meter tab, the meter list and the summary pie chart.

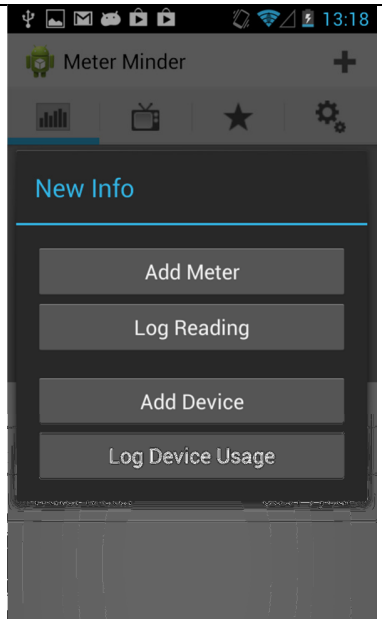

Figure 7 showing the result of clicking the add button.

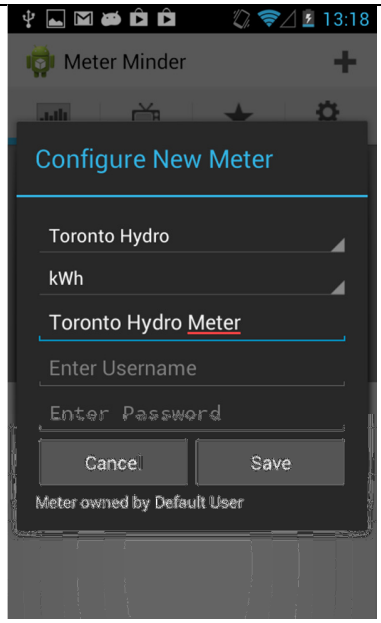

Figure 8 showing the UI for adding a new meter. Also used when editing an existing meter.

include the time since last reading, amount of the utility consumed (and dollar value) over this period and a comparison of usage relative to the average for this meter.

The meter tab (Figure 6) shows a list of all meters currently being tracked by the app. Each row in the table shows the meter's name and an estimate of its yearly cost to the user. These cost estimates are also shown in a pie chart when no meter is selected. Examples for a gas meter and a Toronto Hydro meter are shown in Figures Figure 11 and Figure 12. Long pressing a row in the table gives the user the option to edit or delete the meter or view its readings (Figure 13).

The graph shown depends on the type of the meter. For all non-Toronto-Hydro meters, it shows each reading as a bar whose height is the usage per day since the previous reading. In the case of a gas reading (Figure 11), we also show a black line which gives the historical average for the meter based on a linear regression of all readings and daily mean temperatures for the user's home. This enables the user to quickly determine if they have been more or less efficient than their own average, after correcting for temperature.

Figure 9 showing the UI for adding a new reading. Also used when editing an existing reading.

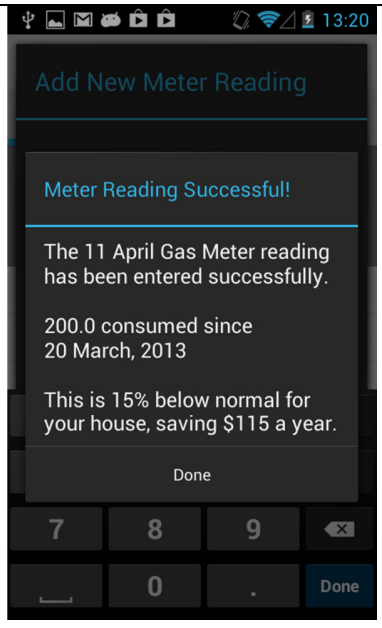

Figure 10 showing the summary page presented after a reading has been added successfully.

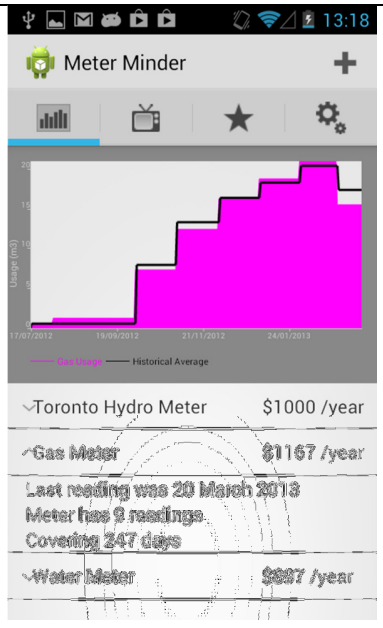

Figure 11 showing the result of selecting a gas meter in the meter list. Bars show mean daily usage for periods between readings.

Since the data for a Toronto Hydro meter is higher resolution than a regular meter, we use a special graph to summarise it (Figure 12).

The tips tab contains a list of all the tips that have been generated for the user (Figure 14). Currently the app generates tips for each meter and device, which these are regenerated when new data is added. Each tip in this table shows the amount saved in terms of both the relevant utilities and dollars. Associated with each tip are a brief description and a more detailed explanation (Figure 15).

Not shown in the list of tabs in the NFC functionality. When an NFC tag comes within range of the phone, it is scanned and passed to the application if the tag has the correct format. If the tag is blank, the user is presented with a registration screen (Figure 16). The tag is also written with a unique identification number, which is used to identify the database entry made by the registration page. If the tag is not blank, and is registered in the database, the user is asked for an estimated time of use. If an event is logged, a database entry is written. After a device log event, a statistics page is displayed (Figure 18).

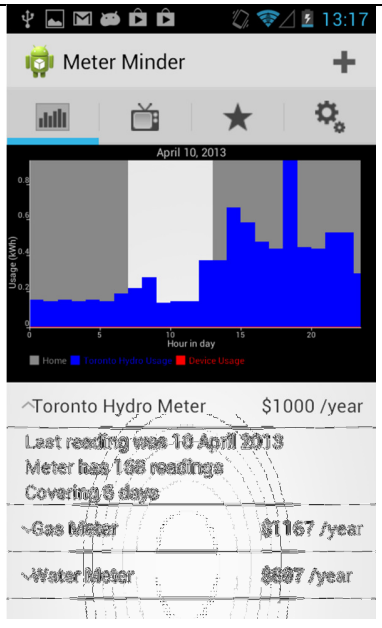

Figure 12 showing the result of selecting a Toronto Hydro meter in the meter list. Bars are hourly usage. Background shows period s when user is home (grey) or not home (white).

Figure 13 showing the list of readings associated with a meter. This is accessed by long pressing a meter in the meter list or clicking on the meters detail draw

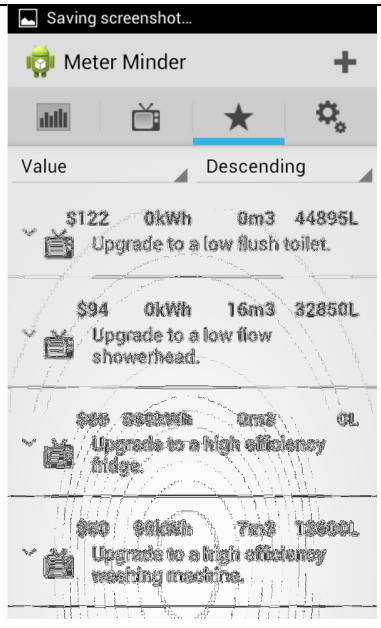

Figure 14 showing the tips tab containing a list of example tips.

The settings page allows the user to turn location tracking on and off (Figure 19). Pressing the “I Am Home” button causes the app to attempt to get the users current GPS location, and write it as the working “home” location. If the “Use My Home WiFi” option is enabled, it will also write the current router’s MAC address as a future “quick reference”, to potentially bypass using the battery-intensive GPS search.

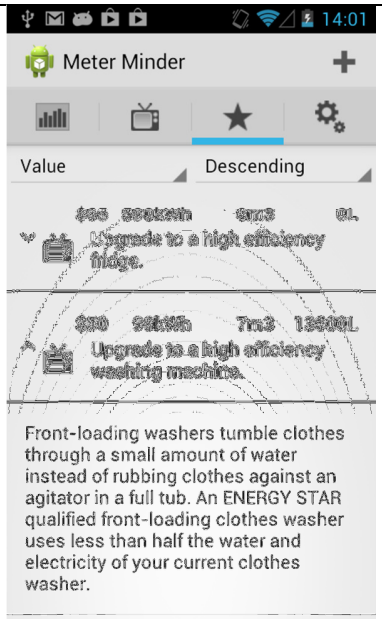

Figure 15 showing the result of selecting a tip in the tips table.

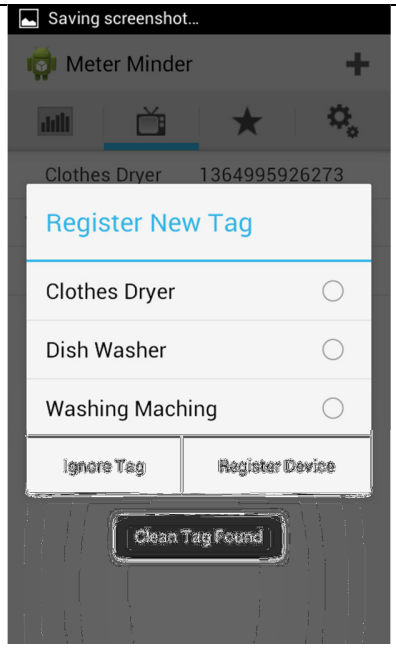

Figure 16 showing the registration of a new device upon discovery of an NFC tag.

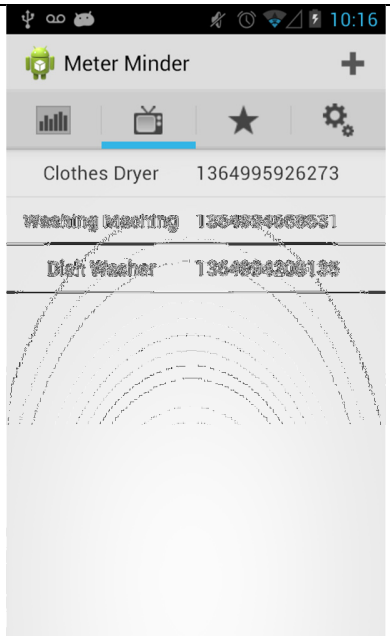

Figure 17 showing a list of devices registered using NFC tags (NFC tags not shown)

Lessons Learned

In general, the development of the app went as planned. If we were to begin working on a professional version of this app, we would consider developing the user interface earlier and have real home owners provide their feedback. While there was a certain amount of user testing from the earliest point of development, having a number of dedicated test homes for the duration of the project would have been a powerful development tool for the group.

Overall there was good communication between the apper and the programmers and we met frequently in order to review progress and to rehearse presentations.

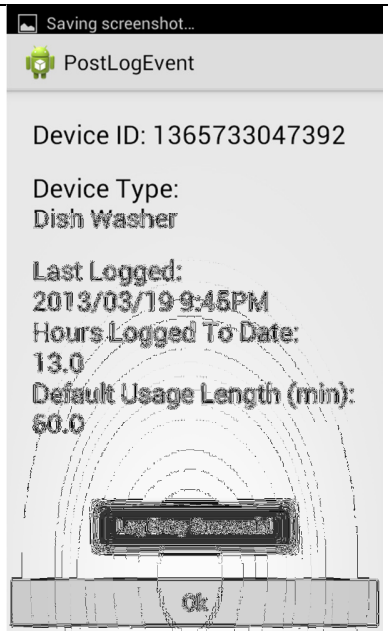

Figure 18 showing statistics from a successful device logging event.

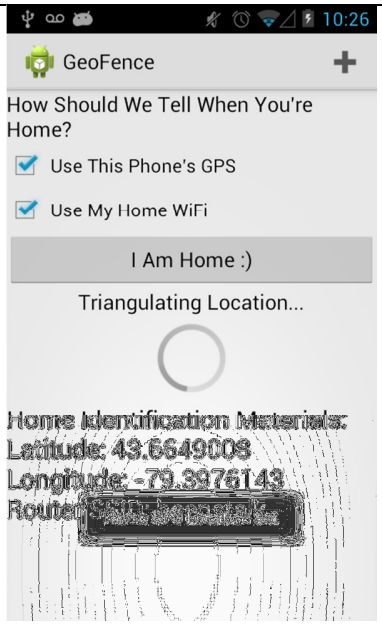

Figure 19 showing the location tracking settings page with tracking enabled

Group Member Contributions

Sam had three main roles as the apper for the MeterMinder project. In the early stages, his primary focus was on the design of the user interface for the app. Sam incorporated best practices from the Android and iPhone design guidelines to create mock-ups of the UI and later worked with the programmers to refine the user interface. Sam's second role was developing the methodology and scientific basis behind the calculations performed in the app. Some of these calculations were based on Sam's experience in the field, while others were based on research into existing databases. In the later stages of the project Sam focussed on supplying actual utility data with which to test the app, and experimenting with the app's functionality to identify technical issues and areas for improvement.

In the early phase of the project, Cai focused on building the database infrastructure that would be used throughout the app. He simultaneously constructed and deployed the application's website, including the modification of an existing Ruby script to scrape data from Toronto Hydro's website. The original script needed to be changed so that it did not require a desktop browser and so could be run on the server. Once this infrastructure was in place, Cai built the meter and weather managers, along with the data processor code used to analyse both these data. This included all associated UI elements, including the graphs used on the meter list page. Finally, Cai was responsible for programming the logic and UI elements required by the tips manager.

Trevor's contributions fell into three categories. The first was the development of the location based tracking. This involved writing a service class that would attempt to determine (based on available WiFi and GPS data) whether the user was home, away, or whether it was not possible to determine their location. The second was the detection of and the logical flow for the handling NFC tags. The system had to detect compatible tags and allow the user to write them with an ID number associated with a particular type of device, while being able to handle the case of defunct or deleted tag entries. The third component was linking the first two components to the central database by modifying the database helper classes to handle the required SQL fields.

Building Science Context

Sam is doing his Masters of Applied Science in the area of building science and building energy use. While large commercial or multiunit residential buildings have full time professional engineering staff managing their operation, single family homes are too small and too distributed to allow for this type of detailed analysis. As a result, houses must be studied from a systemic and ‘best practices’ viewpoint and only generic interventions that target commonalities across the housing stock (such as replacing light bulbs) can be implemented effectively.

The MeterMinder app allows for a more in-depth and technical energy use analysis of individual homes, without having to directly involve an engineer or building energy specialist. The app provides a framework through which standard building performance analysis procedures can be made available to the everyday home owner, allowing them to make more informed decisions about their home and giving them the tools to accurately evaluate the results. It is hoped that this feedback will influence the field of energy conservation and promote a more results-focused approach within the industry.

The app also has the potential to be very useful to researchers in the field of building science and building energy use. Although the energy and water consumption of household devices is published online, their frequency of usage is based largely on anecdotal evidence. If the app is eventually adopted by a large number of users, it could provide valuable information about how often occupants are using particular devices, and how this usage influences overall utility consumption.

Because the app tracks utility consumption, it can also provide researchers with data on the effectiveness of particular intervention strategies. Often equipment does not perform as well as it does in a controlled laboratory environment, and factors related to installation and building occupancy can impact actual performance in the field. By collecting utility data before and after retrofits take place, researchers can quantify the success of equipment replacement and use that information to recommend particular strategies or best practices.

Consider furnace replacements for example. Because the app corrects natural gas usage for weather, users of the app will be able to assess how much energy has actually been

saved by replacing their old furnace. By comparing their results with other people, those manufacturers and installers who deliver better results will be recognized and rewarded with increased business, encouraging industry-wide improvement. At the same time, researchers will be able to identify trends in furnace replacement, and which factors are most influential in achieving good results. This information can then be utilized to influence government incentive programs by requiring implementation of best practices from the research before home owners can qualify for funding. Research data may also help furnace manufacturers design equipment better suited to actual installation and operating conditions.

Future Work

In the current version, the app only communicates with the server when downloading Toronto Hydro readings. The next step in development would be to have the app also push the data it is collecting back to our server. The infrastructure is already in place on the server side.

Having all the data available in the cloud would open up the possibility for server-side computation and statistical analysis, and would allow for changes to the calculation methodology without having to push new versions of the app. It would also allow users to compare themselves to others, adding a social element to the app, and would allow researchers to collect home usage data.

At the app level, we would like to increase functionality by adding a larger number of appliances that users can track. It would also be interesting to add a series of home experiments which users can undertake to learn more about their house and how it consumes utilities.

Next Steps for the App

There is the potential for the development of this app to be continued in partnership with Sam’s employer. As a result, we are not interested in having the business school take up the application idea and do not wish the app to be open-sourced.

Third Party Tools

achartengine

http://achartengine.org

The open-source charting library for Android. We used it for all graphs and pie charts in MeterMinder. It is easy to set up and very flexible around data types. Our only criticism is that visual customisation of the charts can be tricky to get perfect.

Async HTTP Client

http://loopj.com/android-async-http

Java library that greatly simplifies HTTP requests and allows for asynchronous processing of results. It also includes classes to help process JSON and XML responses. We used this to manage the app's communications with our server.

ExpandableListFragment

https://gist.github.com/mosabua/1316903

A handy class for using an ExpandableList inside a fragment. This functionality is not currently provided by Android. See the ExpandableListView class for the non-fragment version (http://developer.android.com/reference/android/widget/ExpandableListView.html).

Ruby On Rails Tutorial

http://railstutorial.org

A very thorough, and free, book/tutorial for learning Ruby on Rails. It assumes little to no knowledge of Ruby and also covers good software development practices (e.g. GIT for version control). The tutorial uses Heroku throughout, making it an excellent introduction to setting up the website too.

Bootstrap

http://twitter.github.io/bootstrap

A very useful CSS Framework which makes setting up a nice looking website trivial. It is highly customisable and compatible with Ruby on Rails. Using this library meant that we had to write very little CSS.

Heroku

http://heroku.com

Heroku is a cloud application platform. It is very stable (since it is based on Amazon's web services) and accepts applications written in a variety of languages. Importantly for this course, a free tier is available that is surprisingly powerful and allows multiple applications to run constantly. Our setup was one app running the website (serving the API) and another app running a scheduled task that scraped from the Toronto Hydro Servers. The second task communicated with the first through our API.

Mechanize

http://mechanize.rubyforge.org/

A Ruby library which automates script interaction with websites. We used this to build the new version of the Toronto Hydro scrapper script which did not require desktop browsers and so could run on our Heroku server.