Feel the World: A Mobile Framework for Participatory Sensing

Theophilos Phokas[1], Hariton Efstathiades[2],
George Pallis[2], and Marios D. Dikaiakos[2]

[1] Department of Computer Science, University of California, Los Angeles
[2] Department of Computer Science, University of Cyprus
tphokas@gmail.com,
{cs07eh2,gpallis,mdd}@cs.ucy.ac.cy

Abstract. Nowadays, smartphones have almost replaced basic feature phones all over the world. Today's smartphones come equipped with an increasing set of embedded sensors, computational and communication resources. All these gave developers the ability to design and implement a wide variety of applications in the domains of healthcare, social networking, safety, environmental monitoring and transportation. This paper presents a novel middleware platform, called Feel the World (FTW) which provides third party programmers, with little phone programming experience, the ability to develop applications that enable people to sense, visualize and share information about the world they live in.

Keywords: Participatory sensing, mobile applications, mobile data collection.

1 Introduction

Technological advances in sensing, computation, storage and communication have brought closer than ever a global sensing device that enables a great number of novel applications that weren't available in the past. Small inexpensive sensors, low power processing but mostly the fact that today's smartphones come equipped with an increasing set of embedded sensors, computational and communication resources, gave developers the ability to design and implement a wide range of applications. Analysts estimate that 5 billion people worldwide use mobile phones[1] - more than half the world's population.

While the real numbers are debatable, it is clear that the proliferation of smartphones equipped with cutting-edge sensing technology and high-end processors opens new horizon in participatory sensing having a tremendous impact on our society. In this context, smartphones represent an ideal computing platform to develop urban sensing applications across a wide variety of domains, such as social networking, environmental monitoring, healthcare and transportation [7,10]. Therefore, a key challenge in realizing the potential of participatory

[1] `http://www.physorg.com/news185467439.html`

sensing is to ensure the ease of developing and deploying such applications, so that the developer does not have to reinvent the wheel.

To address this challenge, we present a novel middleware platform, called Feel the World (FTW). FTW provides to third party programmers, with little phone programming experience, the ability to develop applications that enable people to sense, visualize and share information about the world they live in. FTW framework abstracts implementation details, like service and thread implementations, in order to provide a convenient way to handle sensors both embedded and external. Even though capturing sensor data is quite easy using the provided SDKs, developing a continuous sensing application that takes into account network traffic and the resources of the device (e.g., battery, processor utilization) on which the application is running is not a trivial task. Although several frameworks have been developed in order to address this challenge (e.g. MyExperience [9], Jigsaw [8], Funf [1], SociableSense [11], ODK Sensors [3]), FTW differs from the existing platforms since it provides a dynamic control of sensors (sampling rate) based on the specific technical specifications and the current resources state (battery level, number of running applications) of the mobile device. In addition, FTW provides a mechanism that determines when to upload the collected data to the server based on the priority of data, Wi-Fi availability and other configuration parameters of the mobile device. The key contributions of our work can be summarized as follows:

- A novel open source framework for developing people-centric sensing android applications. Through FTW, developers would be able to exploit and configure all the embedded sensors of mobile phone as well as external sensors. The general configuration properties of FTW are the sampling rate, the duration of each data collection, the priority of data and the running environment (background/foreground). Additionally, developers can specify whether or not the data will be uploaded on a server and how often this will take place. FTW framework can be downloaded from `http:/grid.ucy.ac.cy/FTW/`.
- We evaluate the proposed framework demonstrating its expressivity, the utilization of resources and the ease of use in developing a people-centric application using its built-in features.

The rest of this paper is structured as follows. Section 2 gives a brief overview of relevant frameworks. Sections 3 presents the system architecture of the proposed framework. Section 4 demonstrates the FTW framework. Section 5 gives an insight to the work that will follow on the framework and concludes the paper.

2 Related Work

Participatory sensing enables collection of environmental sensory data by ordinary citizens, through devices such as mobile phones, without requiring any pre-installed infrastructure. Despite the radical increase of mobile applications mainly due to the popularity of smartphones and app stores, there are still only a limited number of applications for participatory sensing purposes [4]. The reason

Table 1. Mobile Frameworks for Participatory Sensing

	Jigsaw [8]	MyExperience [9]	SensLoc [6]	SociableSense [11]	Funf [1]	ODK [3]	FTW
`Embedded Sensors`	microphone accelerometer GPS	GPS Bluetooth	WiFi accelerometer GPS	microphone accelerometer Bluetooth	all	all	all
`External Sensors`	no	no	no	no	no	yes	yes
`Task Inference`	yes	yes	yes	yes	no	no	no
`Send data to server`	no	yes	no	yes	yes	yes	yes
`Dynamic Sampling`	no	no	no	yes	no	no	yes
`Decision module`	no	no	no	yes	no	no	yes

resides to the programming challenges of implementing such applications and especially due to resource constraints that prevent adoption of these applications in under-resourced environments. This main constraint reveals the challenges that participatory sensing still has to face: (1) Continuous sensing by making mobile applications more efficient and (2) creating mobile applications easier by leveraging the complexity of building such applications. Many frameworks and platforms have been proposed addressing the aforementioned areas. Some related work focus on specific sensors and tasks trying to achieve maximum efficiency, while other ones succeed a more general approach of the problem trying to ease the development of participatory sensing using all sensors.

Jigsaw [8], SociableSense [11], SensLoc [6] MyExperience [9], ODK Sensors [3] and Funf [1] are some indicative recent frameworks that enable mobile phones to collect sensor data important to a users daily life. The Jigsaw [8] mobile sensing platform balances the performance requirements of the applications and the resource demands of sensing continuously on the smartphone. SociableSense [11] is a framework that captures user behavior in office environments, while providing the users with a quantitative measure of their sociability with their colleagues. Novel energy efficient mechanisms that are adaptive with respect to the changing context and optimized according to the requirements of users collecting the sensor data have been implemented. SensLoc [6] is a platform that provides location context to applications by using a combination of acceleration, Wi-Fi, and GPS sensors to find semantic places, detect user movements, and track travel paths. MyExperience [9] is a framework for collecting data and capturing user's feedback. The innovative idea behind MyExperience is that it does not only collects quantitative data, but also qualitative such as user's opinion for the sensed event. Moreover, MyExperience provides an easy configuration mechanism throughout an XML-based configuration file. A key characteristic of all these frameworks is that they are focused on specific sensors and sensing tasks (like location queries, activity inference, etc). The ODK Sensors framework [3] provides a single sensing interface for both built-in and external sensors, hiding many details involved in developing sensing applications. The ODK Sensors framework provides an abstraction on which to develop and deploy user-level device drivers on Android smartphones.

The work most similar to ours is Funf [1], which is an open sensing framework developed by MIT Media Lab[2] for creating participatory sensing applications that takes account all of phone's embedded sensors. The core concept is to provide an open source, reusable set of functionalities, enabling the collection, uploading, and configuration of a wide range of data signals accessible via mobile phones. Funf contains reusable and expandable code and abstracts data and sensors through probe data structure.

The proposed FTW framework differs from existing ones in several aspects. Contrary to SensLoc, MyExperience and SociableSense, FTW does not focuses on specific sensors and task inference. Instead, FTW, ODK Sensors and Funf frameworks take into account all the phone's sensors. ODK Sensors framework aims at packaging software so that non-technical users can access external sensors from a locked mobile device running a stock version of Android operating system. As opposed to our work, Funf framework and ODK Sensors framework do not provide any mechanism for dynamic control for the process of sensing. Additionally, Funf and ODK Sensors framework do not support any mechanism that will decide when to upload the accumulated data to the server with respect to the priority of data, WiFi availability and programmer's configuration. Table 1 summarizes the key characteristics of the existing frameworks.

3 System Architecture

3.1 Overview

FTW uses the Android SDK and Java Runtime environment in order to develop the services that provides at the client-side and at the server-side respectively. The client-side component allows the collection of values from sensors embedded on the smartphone or from external sensors (the platform of Sensaris senspods[3], which is running under a RTOS-based kernel, has been implemented) connected to the device via Bluetooth connectivity. Collected values can be aggregated and/or transformed locally on the client and uploaded to the server-side component in real-time or at a later moment. The server-side component gathers, processes and visualizes the retrieved values. Android OS has been selected as the target platform for FTW framework because it is open source, has more features unlocked compared to the rival operating systems (background services) and supports connectivity with more external sensors. Our high level goal is to minimize effort for the developer. This includes both developers who are using

[2] Funf. `http://funf.media.mit.edu/index.html`

[3] Sensaris. `http://www.sensaris.com/`

the FTW API and Android library to build apps, as well as developers extending the FTW framework at the lower levels. Below, we list the design goals for FTW:

– be expandable ensuring the ease of developing applications so both experienced and inexperienced users will be able to use it effectively to create applications for participatory sensing, providing a high-degree of isolation between applications and sensor-specific code;
– support all the embedded sensors of mobile phones, facilitating the integration of new sensors into applications without requiring modifications to the OS configuration;
– provide a background service for monitoring the collection of data in order to ensure that the collection process doesn't affect user's experience. This service will dynamically configure sensors based on phone's technical characteristics and the phone's current resources state;
– provide functions which automatically determine when the data should be uploaded to the server.

3.2 FTW Data Types

FTW has a common interface for each unique data type. Specifically, each type of sensor data collected by the system is encapsulated as a conceptual "IData" object. IData gives an homogenous representation of sensor data (generic sensor data, location data, WiFi data) in the framework. Figure 1 depicts the IData interface. The abstraction for implementing and operating sensors is captured by BaseSensor class (see Figure 2). Using the modular BaseSensor architecture, it is easy to add new sensors to the system, or swap existing sensors with an improved version. All sensors support a common set of behaviors, and each one defines a set of configuration parameters that control it, and the format of its output. Sensors can be configured locally on the device or remotely through the server. Figure 3 depicts the class diagram of FTW implementation. It is an automatically generated diagram which shows the dependencies between the classes. BaseSensor class represents an abstract sensor. It contains the android service implementation and the code which is responsible for the communication with the modules of the FTW, such as communication service and monitoring service. Inheriting BaseSensor class, we can create our own sensor instances for sampling sensor data. Currently, 16 different types of sensor data are collected. Specifically, FTW software framework includes all the embedded sensors of the phone as well as external environmental sensors from the Sensaris senspods platform.

3.3 Data Formats

Sensor data are saved by default as CSV values and stored locally in the phone. Each write is made on a different file in order to prevent complete data loss in case of file corruption and to allow periodic data upload to the back-end server. Since many users do not have a mobile data service plan, the system

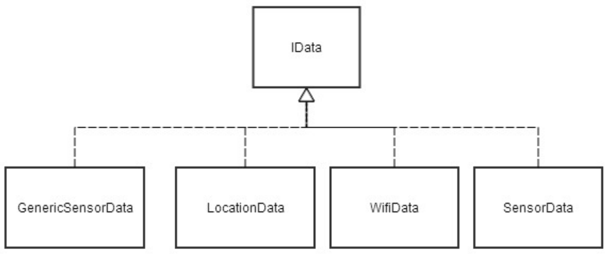


Fig. 1. IData interface

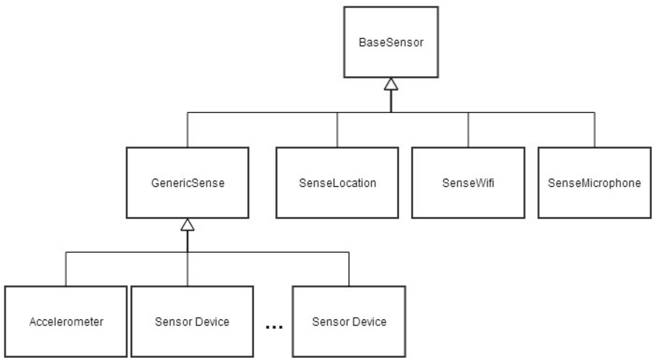


Fig. 2. BaseSensor Class

has been designed by default in the absence of network access and the phone accumulates the collected CSV files locally. When server connection is established (for example a participant connects to Wi-Fi), the system attempts to upload files in order to be further processed.

3.4 FTW Components

FTW is a two-sided application with the client-side running on the mobile device and the server-side running on the server. On server-side we provide a simple Application Programming Interface (API) for easy communication with the client-side of the framework. More specifically, two modules are provided: Messaging manager and File transfer manager, which both aim to leverage the complexity of creating the receiving on server-side. On client-side our main purpose is to provide a modular framework for creating new sensing modules by abstracting the way we handle data. Figure 4 depicts the architecture of the FTW middleware platform. In the following paragraphs the modules of the proposed FTW framework are described.

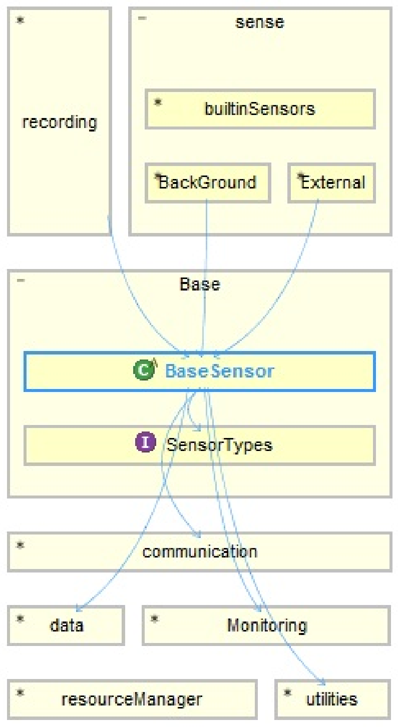


Fig. 3. FTW Class Diagram

Client-Side Data Layer

- Foreground Sensing. An abstract implementation that enables developer to create an instance for a foreground sensor configuring the sampling rate, duration and priority of data. The collection of data is made in the foreground and stops when the application is closed.
- Background Sensing. An abstract implementation that enables developer to create an instance for a background sensor configuring the sampling rate, duration and priority of data. The specific module is implemented as an Android service and the sampling is continuing even when the foreground application is closed.
- External Sensing. This module leverages the complexity of integrating an external sensor in the framework. FTW provides a mechanism to connect and pair the external sensor with the mobile device.
- Resources Service. This module runs on the background and collects data which are related with the resources of the mobile device. Collected data can be categorized in the following two categories: 1) static data, which rarely or never change (processor clock, ram) and 2) dynamic data, which are changing over time (battery level, running processes).

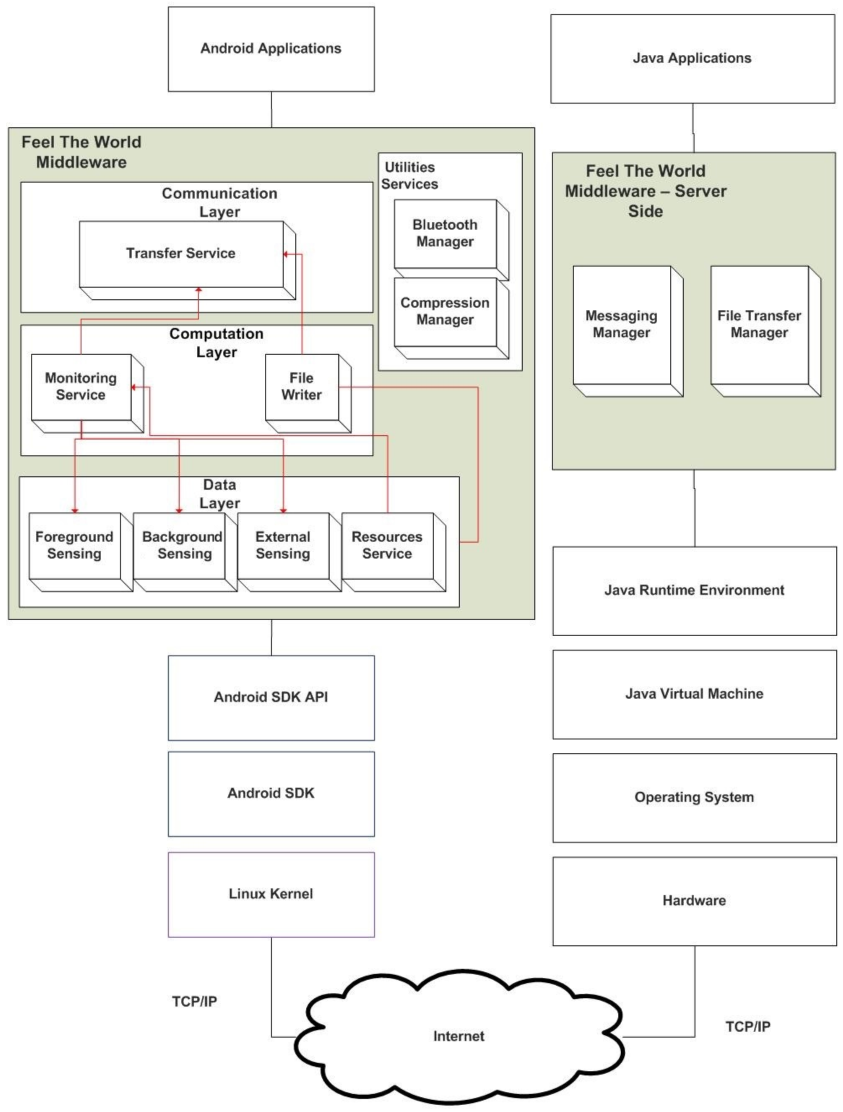


Fig. 4. FTW System Architecture

Computation Layer

- Monitoring Service. This module is responsible for monitoring the collection of sensor data. Monitoring service is able to dynamically configure sensors based on phone's technical characteristics and the phone's current resources state. Using this module, the developer can set specific rules that affect the sampling of sensor data. For example, when battery is below 20% then the sensors will stop sampling. Monitoring service includes a decision mechanism

which decide when to upload the accumulated data to the server with respect to the information received (e.g., priority of data, Wi-Fi availability and programmer's configuration). The decision module processes all information received and generates an action plan when to upload the data to the server.
- File Writer. This module writes data to CSV files.

Communication Layer

- Transfer Service. This module is responsible for data communication with the back-end server. It includes rules whether the data should be sent to the server. By default, when a participant connects to Wi-Fi, the system attempts to upload files.

Utilities Services

- Bluetooth Manager. This module provides a useful API for discovering and pairing nearby Bluetooth devices from the application. It also provides the capability of exchanging data between the mobile and the Bluetooth device.
- Compression Manager. This module compresses data using a widely known compression approach, the Deflate algorithm [5]. The Deflate algorithm divides the compressed data in variable-length blocks using a sliding window that can be up to 32 kilobytes large. Several compression algorithms have been tested [2]. Specifically, we tested the following compression techniques: a) Huffman coding, b) a combination of Huffman and Run-Length coding, c) ZIP compression, and d) Deflate compression. Deflate compression technique achieved the best compression ratio for both small and large length of strings. Huffman coding and Run-Length did not achieve data compression since these encodings are mainly used for compressing images. According to [12], experimental results have also shown that Deflate is the most energy-efficient algorithm.

In the next section, we have developed a few example applications to demonstrate reuse, flexibility, and extensibility of the framework.

4 Evaluation

In this section, we demonstrate the FTW framework capabilities. Specifically, we focus on demonstrating its expressivity, the utilization of resources and the ease of use in developing a people-centric application. All experiments were conducted using Google Nexus-S mobile phone.

4.1 Expressivity

With the term "expressivity" we refer to the ability of framework to utilize as many sensors as possible and the provided freedom to the configuration of

Table 2. FTW Expressivity

Sensor	Background Sampling	Rate (samples/sec)	Duration (sec)	Monitoring	Log data	Send to server
`Accelerometer`	yes	600	60	yes	yes	yes
`Accelerometer`	no	100	30	no	no	no
`Light sensor`	yes	500	60	yes	yes	yes
`Microphone`	yes	500	60	no	yes	no
`WiFi`	yes	1000	-	no	no	no
`Location-GPS`	yes	2000	-	no	no	no
`Location Network`	yes	2000	-	yes	no	no

sensors. In order to demonstrate the expressivity of FTW we have instantiated several sensors with different characteristics and configurations. We can see from Table 2 where we present 7 indicative sensor instances, each with different characteristics. For example, we have two accelerometer instances running with different sampling rate and duration, one running in foreground and the other in background implemented as a service.

4.2 Resources Utilization

In order to understand the automatic monitoring service and the way it handles battery utilization, we created two similar applications. Both applications gather accelerometer data and have the following configuration: background service runtime environment, sampling rate 30 seconds and duration 5 seconds. Their only difference is that the first application utilizes automatic monitoring service whilst the second one does not. The monitoring service handles the sampling rate of the first application based on the current battery level. The monitoring service runs on the background on specific rate and each time it runs sets the sampling rate as follows:

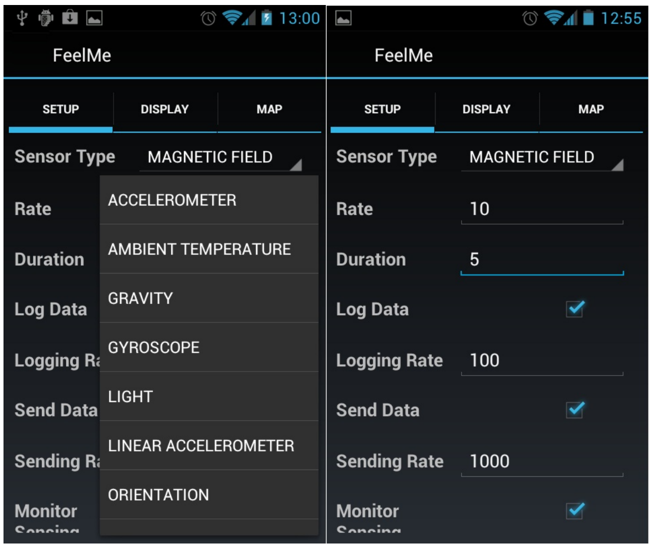
$$(\%\, of\; used\, battery\; +\; 1)\; *\; initial\; sampling\, rate \tag{1}$$

For example, when battery reaches 80%, then monitoring service will set sampling rate at 1.2 * initial sampling rate. This means that as long as the battery extinguishes, the sampling rate is becoming slower and slower. Our purpose here is to understand the impact of monitoring service in the utilization of resources at the mobile devices. In order to address it, we consider two similar applications which are running for 4 hours; the phone is plugged in to the power supply. We observe that the battery consumption is the same for both applications. This was expected since battery level was 100%. Next, we detach the phone from power supply and we observe that the battery utilization of the application which uses the monitoring service of FTW is gradually under 4% while the second one has a battery utilization of 6%. From these two applications, we observe that monitoring service utilizes in an efficient way the resources of a mobile device since it dynamically adapts the sampling rate as well as the duration of sampling.

4.3 Ease of Use

In order to demonstrate the ease of use of FTW framework, we developed a people-centric application for gathering data, called FeelMe. FeelMe application can be used by scientists or self-trackers that are willing to gather and share their daily data, their environment, or even their location. The motivation of FeelMe is to provide citizens a wide range of crowdsourced city information including traffic flow, road reports, parking places, and even warnings about where the latest speed traps have been set up. The user can take a picture of a place or an area where a problem has been detected through the mobile device's camera and report it with a description and title. The external sensor devices, which are embedded with GPS, detect CO_2, HUV, motion, noise measurements and sent these data to the mobile device through Bluetooth connectivity, and then the data are sent to server through TCP. The server stores all these data and depict them in a live map.

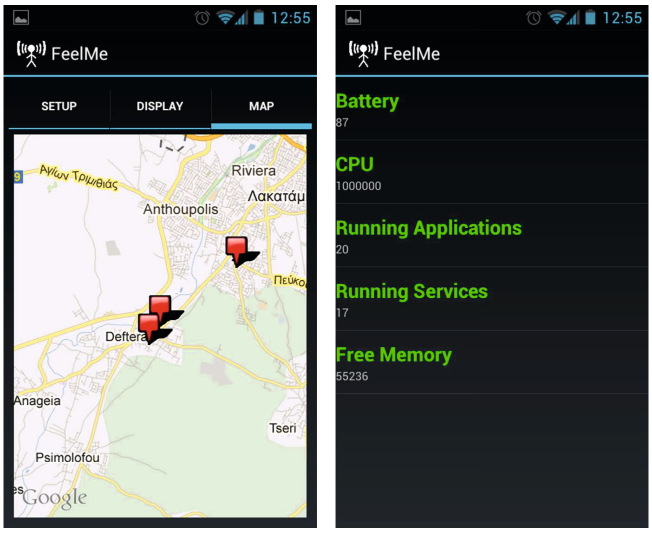
On the left screen in Figure 5, the user can choose the sensor that he is willing to gather data. FeelMe provides a wide range of settings, such as sampling rate, duration of each sampling etc. On the right screen in Figure 5, the user can observe the stream of collected sensor data. Moreover, the application collects location data and displays them on a map as depicted in the left of Figure 6. Finally, the application displays the current resources of the phone (on the right



Sensor Type MAGNETIC FIELD
Rate
Duration
Log Data
Logging Ra
Send Data
Sending Ra
Monitor Sensing

Sensor Type MAGNETIC FIELD
Rate 10
Duration 5
Log Data ✓
Logging Rate 100
Send Data ✓
Sending Rate 1000
Monitor Sensing ✓

Fig. 5. Sample screenshots: FeelMe Application setup display



Battery
87
CPU
1000000
Running Applications
20
Running Services
17
Free Memory
55236

Fig. 6. Sample screenshots: (left) FeelMe Application map display, (right) FeelMe Application resources display

screen of Figure 6). For the development of this application we used the built-in features of FTW through its API. To sum up, the development of FeelMe application required about 320 lines of code.

Users who would like to participate in and contribute to *FeelMe* install the FTW runtime from `http:/grid.ucy.ac.cy/FTW/` on their mobile smartphones. Currently, FTW uses the Android SDK (Software Development Kit[4]).

5 Conclusion – Future Work

The great increase of smartphone users creates a need and opportunity to involve these devices in data gathering mechanisms that will provide projects with the needed data. Our focus in this work was to provide a framework that is scalable, expandable and maintainable. FTW ensures the ease of developing applications so both experienced and inexperienced users will be able to use it effectively to create applications for participatory sensing. Finally, an Android application has been built using the FTW framework, which makes use of many of its built-in features. The application can be used by citizens interested in collecting and exploring information related to the mobile device, its environment, and its user's behavior.

A key characteristic of the proposed framework is its extensibility. As future work, we plan to extend FTW framework so as to take into account tasks priority, security, and storage issues. In terms of tasks priority, the proposed

[4] `http://developer.android.com/sdk/index.html`

framework does not support any policy since it considers that all tasks have the same priority. However, some tasks are more time critical than other ones and therefore they should have higher priority. Regarding security issues, collected data contain sensitive private information. Although data are sent anonymously, no encryption/decryption protocol is used between sender-receiver in the existing framework. For the furure, we plan to extend FTW so as to allow storing and sending data in encrypted format for security purposes. In terms of storage aspects, in the current version of FTW framework the data are stored as CSV files. In the future, we plan to switch to SQL-lite in order to be able to index data and retrieve them more efficiently.

Acknowledgement. This work was supported by the third author's Startup Grant, funded by the University of Cyprus.

References

1. Aharony, N., Pan, W., Ip, C., Khayal, I., Pentland, A.: Social fmri: Investigating and shaping social mechanisms in the real world. Pervasive Mob. Comput. 7(6), 643–659 (2011)
2. Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Trans. Comput. Syst. 24, 250–291 (2006)
3. Brunette, W., Sodt, R., Chaudhri, R., Goel, M., Falcone, M., Van Orden, J., Borriello, G.: Open data kit sensors: a sensor integration framework for android at the application-level. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys 2012, pp. 351–364. ACM, New York (2012)
4. Das, T., Mohan, P., Padmanabhan, V.N., Ramjee, R., Sharma, A.: Prism: platform for remote sensing using smartphones. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys 2010, pp. 63–76. ACM, New York (2010)
5. Deutsch, P.: Deflate compressed data format specification version 1.3 (1996)
6. Kim, D.H., Kim, Y., Estrin, D., Srivastava, M.B.: Sensloc: sensing everyday places and paths using less energy. In: Proceedings of the 8th ACM Conference on SenSys, pp. 43–56. ACM, New York (2010)
7. Campbell, A.T., Eisenman, S.B., Lane, N.D., Miluzzo, E., Peterson, R.A., Lu, H., Zheng, X., Musoles, M., Fodor, K., Ahn, G.-S.: The rise of people-centric sensing. In: IEEE Internet Computing, pp. 12:12–12:21 (July 2008)
8. Choudhury, T., Lu, H., Yang, J., Liu, Z., Lane, N.D., Campbell, A.T.: The jigsaw continuous sensing engine for mobile phone applications. In: Proceedings of the 8th ACM Conference on SenSys, MobiSys 2010, pp. 71–84. ACM, New York (2010)
9. Froehlich, J., Chen, M.Y., Consolvo, S., Harrison, B., Landay, J.A.: Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones. In: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, MobiSys 2007, pp. 57–70. ACM, New York (2007)

10. Lane, N.D.: Community-aware smartphone sensing systems. IEEE Internet Computing 16(3), 60–64 (2012)
11. Rachuri, K.K., Mascolo, C., Musolesi, M., Rentfrow, P.J.: Sociablesense: Exploring the tradeoffs of adaptive sampling and computation offloading for social sensing. In: Proceedings of the 17th MobiCom, Las Vegas, USA, pp. 71–84. ACM (2011)
12. Tsiftes, N., Dunkels, A., Voigt, T.: Efficient sensor network reprogramming through compression of executable modules. In: 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2008, pp. 359–367 (2008)