AIRS: A Mobile Sensing Platform for Lifestyle Management Research and Applications

Dirk Trossen[1] and Dana Pavel[2]

[1] Cambridge University Cambridge, UK
dirk.trossen@cl.cam.ac.uk
[2] University of Essex Colchester, UK
dmpave@essex.ac.uk

Abstract. Utilizing mobile devices for gaining a better understanding of one's surrounding, physiological state and overall behavior has been argued for in many previous works. Despite the increasing usage of mobile devices for research in this space, few platforms developed are readily available for supporting the wider research community. This paper presents a mobile sensing platform that allows for exploiting the latest and ever-increasing capabilities residing in mobile devices. While we highlight the main design and implementation characteristics of this solution, we also outline our experiences with this platform for typical usage scenarios in lifestyle management.

Keywords: mobile sensing, gateway, platform, lifestyle management, context awareness.

1 Introduction

The importance of mobile devices and their capabilities has long been recognized within research projects such as [1-4, 23] as well as commercial solutions such as [5,6]. This is due to mobile devices becoming increasingly more powerful in recent years. Processor speeds have exceeded 1GHz with storage capacities in the tens of GBs. Connectivity options now span from short-range Bluetooth over WLAN to high-speed cellular, while capabilities to locate mobile devices are almost ubiquitous nowadays. Furthermore, the penetration of smartphones has surpassed 50% in some markets such as the US or the UK throughout 2011.

Beyond hardware improvements, the mobile software space has exploded as well, with applications created for any possible usages. Such dramatic growth in mobile applications is driven by easier to use development tools as well as the support of an ecosystem provided by companies such as Apple or Google. Using such tools, it is possible to create applications capable of harvesting a growing pool of information that originates from or can be collected through such devices.

There is no need to justify here the advantages of a platform-based approach. Platforms are found now at various levels within computing architectures and works such as [7] discuss the advantages of this approach within embedded systems. What

C. Borcea et al. (Eds.): MobilWare 2012, LNICST 65, pp. 1–15, 2013.

we argue for is the need for an open-source, **widely available** mobile sensing platform that is flexible enough to be used for various purposes, allows for both automatic and manual input and not only enables new applications but also provides valuable support for user research. While other mobile-based sensing platforms have been developed during the years (e.g., [1][3][4]), we think there is value in presenting our platform, which can be immediately downloaded and used by the research community, therefore minimizing the time it takes to deal with sensing-specific issues and, instead, focusing on developing advanced algorithms that make use of such collected information. Our initial motivation behind creating a mobile-based sensing platform and gateway started a long time ago, with a Symbian-based platform [2], when it became clear to us that mobiles will become the more pervasive computing devices, with ever-increasing capabilities for collecting, processing and interacting with end users. However, the more recent developments of mobile devices, software development environments and even user attitudes towards sensing, allowed us to greatly improve the platform by making it easier to add new sensors, functionalities and user interaction means.

Based on our work and experiments within the area of lifestyle management applications[1], we have continuously improved the platform to address requirements of such application area, including allowing end users to get more involved in collecting and interpreting information through their mobiles.

In this paper, we discuss challenges, design solutions and implementation issues as well as the scenarios and experiments we have conducted to test our platform. For this, we organize the remainder of the paper as follows. We start by describing the setting in which we have been using our platform; present the challenges we encountered and the derived requirements while also including references to related work. Such challenges and requirements are important as they drive the design of our platform, which we describe before presenting our current implementation. We further include details about our experiments with the platform. We finally conclude our paper and discuss future work.

2 Scenarios and Challenges

Our recent platform development has been driven by our activities within lifestyle management systems. For that, we have used and further developed the mobile-based platform as one main information provider within a larger system capable of collecting various user context information that covers various dimensions, such as physiological, spatial, social, environmental, or emotional [18]. The main goal of our system was to provide its user with support for better understanding what happened and why it happened by allowing information correlation within a complex space.

The area of lifestyle monitoring is very well represented both in research [1-4] as well as in the commercial space [5][6][10-15], with mobile phones providing means

[1] Some of the work described in this paper has been funded by EPSRC and TSB through the PAL project [17] (grant number TP/AN072C), a research project investigating future healthcare services in the context of self-monitoring and lifestyle management.

for data collection, processing and remote access. Utilizing mobile devices for such scenarios, however, comes with challenges, in particular since the devices are not dedicated sensor platforms but they are primarily meant for personal or professional use [3]. Many of these challenges have been identified and partially addressed within related work, with [21] providing a particularly good overview.

The biggest challenge we have encountered is **battery life**. While advances in processor speeds or storage capabilities have largely been following Moore's Law, battery capacity has developed at a slower pace. Hence, any solution for mobile sensing must be sensitive to battery consumption. As mobile phones are still primarily used for other purposes, any sensing platform must cater to the need of an end user to sustain a certain level of battery that can be used beyond the desired mobile sensing task. One solution is the **configurability** of the platform, allowing for setting larger intervals for polling sensors, such as location and wireless connectivity (wifi, signal strength, etc.). Such options allow the end users to tradeoff the requirements of the experiments with their own needs, e.g., regarding battery life or storage.

Within self-monitoring scenarios, even when end users do not permanently record, there is still a considerable amount of data being generated. Therefore, **storing** and **synchronizing** recorded data has to be taken into account. Here we encountered various models, such as remote provisioning of such data [12][15] or utilizing the local storage of the mobile device [10][13][16]. We found that a platform created for self-monitoring has to provide solutions for storing information both locally and remotely. While local storage capacities have increased, there is still the issue of **safety of data** when considering how likely mobile devices are to be misplaced, stolen or destroyed. Hence, any solution needs an easy way to sync stored data, both within end user's own data space and with other trusted parties. This brings in the issue of **connectivity**. While data connectivity has improved in recent years, simply relying on always-on wireless connectivity can limit the applicability of the sensing platform. Instead, any solution should support a wide range of syncing (and sharing) options, from real-time (if the scenario demands it) to periodic.

Given the continuous addition of sensors on mobile devices as well as external ones (which can use the mobile device as a sensing gateway), a mobile sensing platform has to be designed with **extensibility** in mind, as also pointed out in [8]. Another challenge that comes from the continuous development of mobile devices, and their increased complexity is the impossibility of anticipating all malfunctioning scenarios. Therefore, for scenarios that require long-running experiments it is important to create platforms that ensure persistence of the measurement itself as well as for its recordings, e.g., through automatic restarting in the case of failures and emergency data saving.

Beyond technical challenges involved in building such platforms, using such mobile sensing platforms for recording user information poses major challenges with regard to user needs and concerns. A major challenge relates to **privacy**, as most of the user information collected is of a personal nature, as also discussed in [4]. Another important issue that arises from a sensing solution running on a device with a different main purpose as well as a (still) reduced screen capability is related to **user interactions**. Any solution should blend into the (device) platform-specific interaction model to avoid overburdening the end user. However, within scenarios

such as the ones we have considered, purely relying on automated data collection is not enough. For instance, people like to add their own annotations, which help them identify interesting moments during the day. Therefore, we provide means for such interactions allowing for exploiting user's knowledge and enriching automatic recognition algorithms such as [16].

Social communication is an essential part of our lives and the current trend we can observe is that people are willing to share more and more information. Therefore, such platforms have to provide means for **sharing** either individual or aggregated information with various circles. However, such sharing has to happen under the control of the end user.

A specific challenge arises from our ambition to serve the wider research community. While open sourcing is a means to ensure platform extension, it is not enough. Traditionally, many projects in this area have built their own platforms, which survived for a number of years and were then discontinued. We provide here an actively growing Android-based mobile sensing solution that allows for extensive sensing and is already available in the application store, ready to be installed, configured and used according to any research needs.

3 The AIRS Platform

In this section, we describe the AIRS platform from design to implementation. The design takes into account the various challenges we encountered when building lifestyle management applications.

We chose Android for a number of reasons, the main ones being: (1) its flexibility in terms of customizing user interfaces and interactions, as it allows for controlling font and icon sizes (important in healthcare scenarios), as well as easier interactions and increased awareness through widgets and the notification bar; (2) allowing access to a large number of sensors as well as system information without requiring special root rights; (3) the potential for integrating with future healthcare products through the Bluetooth Health Device Profile (HDP), supported by the most recent Android platform release Ice Cream Sandwich (4.0.4).

The AIRS platform offers the following functionalities:

- Supporting and integrating a wide range of current and future sensors
- Sensor configuration interface, allowing for customizing certain platform settings and behaviors, polling intervals and accuracy levels for certain sensors as well as adding or removing certain sensors
- Quick start mode from the main application launcher screen, using the last selected sensors (if they are still available)
- Inspecting and visualizing current recordings through the notification bar
- Provide two widgets, one used for free-text user annotations and one used for mood-related annotations
- Local recording, where sensors values are stored in a phone-local database
- Remote recording, where data is sent to a remote server for storage.

For simplicity, we describe in this paper only the local recording mode.

3.1 Main Abstractions

Let us refer to Figure 1 for the various classes being realized in our platform and outline how this particular design addresses the aforementioned challenges.

The sensors that can be recorded by the platform are represented by *Sensor* objects and their values can be provided by various resources, being physical (e.g., phone microphone, light sensor) or virtual (e.g., calendar, user annotations). A sensor can be either simple (i.e., when using a single resource) or complex (i.e., when using data from multiple resources). The actual recording is realized through a *Handler* class, which implements Discover() and Acquire() methods that are specific to the set of sensors included within that abstraction. The class also provides interfaces for resource management (destroyHandler()) as well as sharing of data (Share()). The extensibility requirement is addressed by integrating the various Handlers into a *HandlerManager* class, which instantiates the implementations at platform start.

The configurability challenge is addressed by providing a *HandlerUI* implementation for certain handlers. These implementations are made available through the *HandlerUIManager.*

When starting the local recording, each Handler implementation is instructed to discover the available sensors it implements, creating a *Sensor* instance for each available sensor. Each *Sensor* instance is inserted into the *SensorRepository,* which allows for retrieving a value instance at any time.

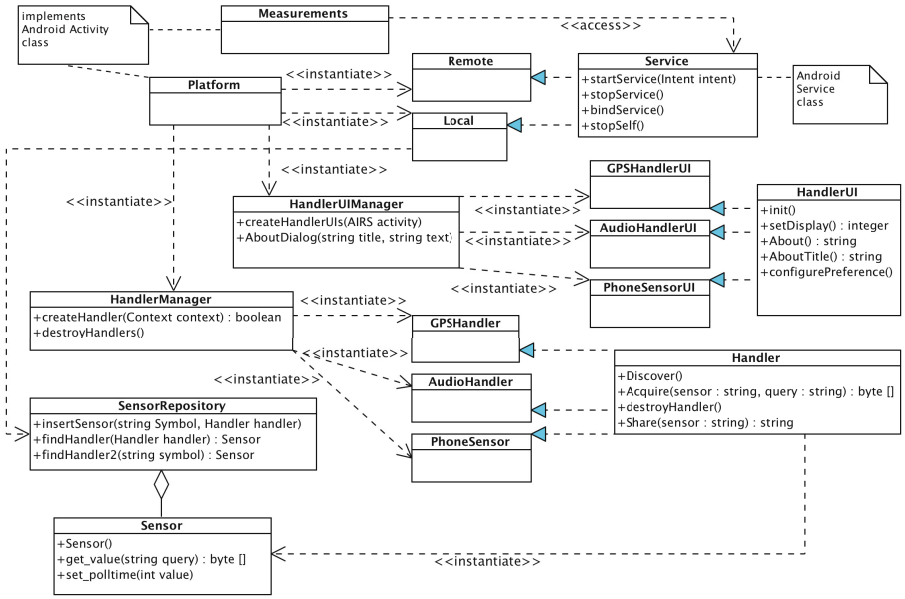

Fig. 1. AIRS implementation diagram

Since we directly base our platform on the Android design and implementation guidelines, as outlined in the SDK [20], any interaction with the end user is implemented as a so-called *Activity* [20]. The *Platform* class in Figure 1 is the main activity, which is started through the icon in the application launcher of Android. This activity provides access to the configuration for the overall platform as well as the handlers that expose a *HandlerUI* implementation. The main activity also allows for launching the local recording. For this, a long-running *Local* service is started, directly realizing the Android concept of a *Service* [20]. Before the service is started, a user dialogue allows for selecting the particular sensors to be recorded or perform a quick start. The current recording can be controlled by the *Measurements* activity, launched when clicking on the appropriate icon in the Android notification bar. The activity displays the latest value for each recorded sensor and also allows for pausing/resuming or exiting a recording.

3.2 Supported Sensors

The number and type of sensors supported by our platform have been increasing, driven by our applications scenarios, any new needs found through user experiments and through the growing ability of the Android system to access information.

As a consequence, our platform currently supports a wide range of information to be recorded. Apart from physical sensors that include location (of various kind such as based on GPS or cell information), gyroscope, accelerometer, pressure, temperature as well as magnetometer, the platform integrates a large variety of platform information such as tasks running, RAM size (used memory), headset status, battery status, cell information, and many more[2]. Given the inherent challenges of getting an accurate ambient temperature through the phone sensor as well as our increased usage of data connectivity, we also utilize web services for gathering information such as the local weather, humidity, wind speed and so on. Furthermore, we also support sensors that can be attached via Bluetooth technology, such as the Alive heart and activity monitor [9]. Figure 2 shows the various information types (left side) currently supported by our platform, in relation to processed information derived from these sensors along several user context dimensions, as implemented in work described in [18]. Based on these types, the current platform implementation exposes in excess of 60 sensor values.

As described above, all sensors are accessed through Handler implementations. Usually, certain groups of sensors are realized by a single Handler providing a common way of accessing this group. For instance, a dedicated Handler implementation realizes the access to the Alive monitor by implementing the particular BT-level protocol. This Handler actually provides values from 6 different sensors provided by the monitor.

Furthermore, the design of the platform allows for directly integrating information processing into the platform through creating a hierarchy of Handler implementations,

[2] We do not utilize the camera as a sensor since Android requires the camera preview to be visible, which contradicts our requirement of being able to use the device as usual.

if so desired. Such decision is driven by factors such as disconnected operation, limiting the amount of data to be sent off for processing purposes, on-the-phone visualizations, "abstract and discard" operations, and so on.

Any addition or change to the supported sensor pool requires re-compiling and re-installing the platform. In order to address the extensibility as well as the battery life requirements, we recommend two important best practice guidelines. Firstly, Handler implementations should access information through callbacks instead of polling, making use of the various OS-level mechanisms that allow for minimizing overall battery consumption. Secondly, any Handler should verify the existence of any necessary resource before using it, avoiding runtime exceptions when the resources are not available. This is particularly important when integrating a new sensor that might not be widely available in most handsets.

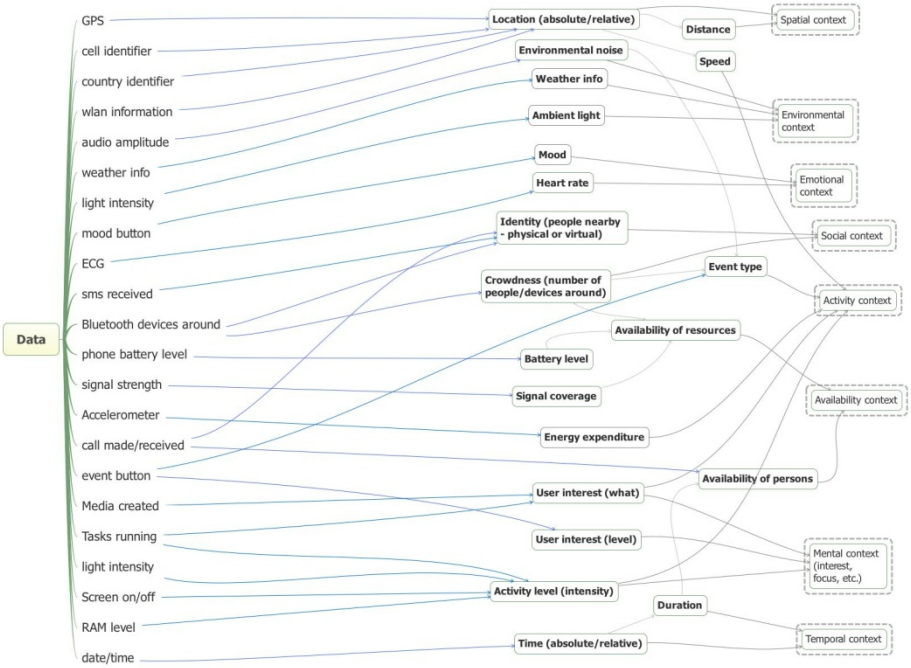

Fig. 2. Sensors supported by AIRS and types of processed information

3.3 Storing and Sharing

Local recordings are stored in an Android database within the local file system. This database approach provides additional security since the underlying file is only accessible to our platform, i.e., it cannot be read by other applications. At any time, the recordings can be synchronized via Android sharing options, such as Bluetooth (transferring the files to a laptop), email (sending the files over the Internet) or through any other installed means (e.g., Facebook, etc.). For this, the platform

generates on-the-fly text-based files that can be parsed at the receiving end. These temporary files start with a timestamp that indicate the start of the recording. Following this, every line carries three different entries. The first one represents the time relative to the initial timestamp, followed by the sensor ID as given in the discovery of each sensor. Finally, the value of the current reading is written in text-encoded format. If a sensor produces a multi-line string, each line is separated with a carriage return. Byte array recordings are written in separate files with the file name being recorded in the Value field.

Once transferred, there are many possibilities to save and work on the data. For instance, we provide a Java program that parses the recordings and saves the data into a MySQL database. Once in the database, the data can be accessed and processed in any way desired. For example, in the mentioned PAL project, data collected through the AIRS platform is combined with data collected from other sources, such as physiological sensors and desktop, further interpreted and visualized through PHP-based scripts, utilizing a story-based approach for depicting interesting moments during a day [18].

Another way to share individual sensor values is provided in the *Measurements* activity (started through the notification bar). Here, individual readings can be seen and shared through any system-internal content provider, after long-pressing the particular sensor in the list of values. While such provider could be Bluetooth or email, it also allows for sharing the value through social networks like Facebook or Google+. To enable such sharing, every Handler implements a human-readable text for each individual sensor.

3.4 Addressing the Battery Consumption Issue

Let us now return to one of the most important issues within the usage of platform, namely the battery consumption.

Within our platform, we rely on three approaches to cater to the need for conserving battery. For that, all handlers attempt to utilize callback functions wherever provided by the Android operation system. For this, we register a so-called broadcast receiver [20] to a particular event (e.g., the cellular signal level). An acquisition thread for this particular sensor then simply sleeps until the OS provides the most recent value through the registered callback function. This significantly reduces overall battery consumption compared to polling mechanisms. In the current realization of the platform, only five groups of information are realized through polling, namely Bluetooth (for discovering surrounding devices), audio (for surround noise measurements), WLAN (for detecting SSID and signal strength of surrounding access points) as well as the RAM size and running tasks of the system. We consider the last two as being less relevant for battery consumption since retrieving this information consumes little power (assuming polling intervals of several seconds and beyond). WLAN and BT are power-expensive resources (although the latest BT version 4.0 significantly reduces consumption, according to specifications). The same holds for the noise level measurements, for which frequent recordings through the local microphone are required.

For all polling mechanisms, the intervals for polling can be configured by the end users, giving them control over the overall consumption. In addition, WLAN scanning can be aligned with the overall device policy, if desired by the end user (i.e., on many devices, WLAN is set to sleep once the screen is switched off).

The end user can also configure to only record values when there is user activity, i.e., when the screen is turned on. This gives a significant control over the power usage of these particular sensors, while still leaving the ability to set a critical level for stopping the recording altogether. Although not implemented through polling, GPS is considered another heavy battery consumer, when used in recordings, especially when its availability varies and frequent signal re-scanning is required. However, the configuration settings allow for determining minimal intervals as well as timings for recording new location values. This allows for using efficient Android callback functions instead of frequent polling. This results in no platform activity in cases where the end user remains stationary.

The platform also provides a setting that exits the recording when a defined battery level is reached (e.g., 30%). With that, users can define their desired amount of battery that should be preserved. The user is notified through the Android notification bar once such killing setting has been executed.

3.5 User Interactions in AIRS

We have mentioned before that one crucial aspect in our experiments was to be able to allow for user interactions in order to (1) configure recording parameters according to various needs and constraints; (2) interact with the running recording for visualizing what is being recorded; (3) allow end users to input their own annotations. We describe here how the platform addresses all these aspects.

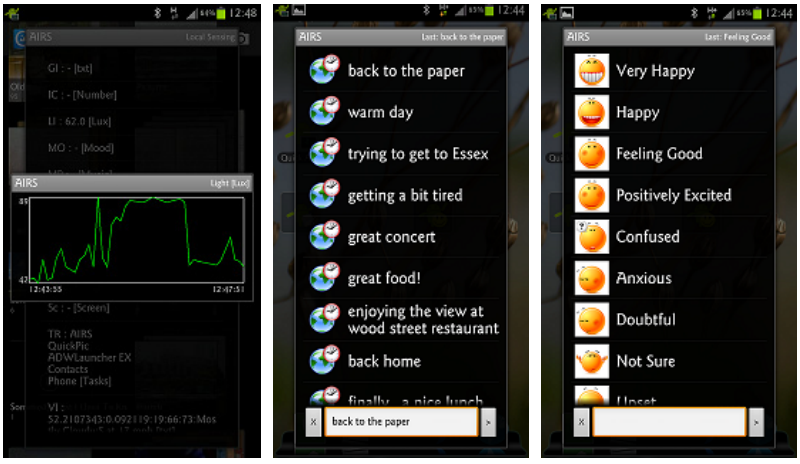

Fig. 3. AIRS Screenshots: (a) visualization; (b)(c) annotation widgets

The configuration mode for setting up the various recording parameters is enabled by the various *HandlerUI* implementations that expose settings for certain sensors (accessed through a *Handler*). Furthermore, the platform itself provides settings that allow for adjusting its overall operation. Each *HandlerUI* implementation makes use of the Android concept of a *PreferenceActivity* [20], which minimizes any necessary code for the particular configurations.

The user can also interact with the platform while the recording is running through the notification bar. The *Measurements* activity allows for inspecting recent recorded values as well as accessing certain visualizations for certain sensors (by pressing on the corresponding item), as seen in Figure 3(a).

As part of our experiments with designing and building lifestyle management systems, it was essential that we better understand what people consider most interesting to be captured within their daily stories. As the mobile phone is one of the most likely devices to be used every day and in multiple situations, we realized that the AIRS platform would be best suited to collect such information, especially in relation to the other recorded information provided by sensors. For this, we utilize the concept of an Android Widget [20] by directly placing a user interface element on the user's home screen. This interaction is one the closest abstraction to pressing a button to annotate while also allowing for adding a meaning to such operation. Figure 3(b) shows the interface for the user annotation, which allows for any text to be inserted and even remembered (by configuring the list size). The user can select a previous annotation or add a completely new one. While emotion recognition is making progress even on mobile phones [16], humans are still better suited to recognize and describe their own emotions. For this reason, we also created a widget that allows for fast mood-related annotations. The user can select from a set of 12 pre-defined mood icons or use an own mood description.

These two widgets connect to two specific Handler implementations of the platform. The value selected or defined through these two widgets is treated the same as any other platform sensor.

4 Usage-Based Experiments and Their Challenges

While the previous sections focused on highlighting certain aspects we consider essential in understanding our platform, we describe next the experiments and experience we have had with using this platform within the lifestyle management setting. What is special about such scenarios is that they usually require recording a multitude of sensors (in order to create a diverse user context picture) and for longer periods of time (in order to cover more aspects of user's daily activities and life).

However, with large amount of data comes the challenge of making sense of it as well as identifying what is really of interest to the end user. As mentioned before, our experiments were mainly focused on better understanding what people consider of importance during the day. For this, we conducted recording experiments with six end users over several days, followed by semi-structured interviews.

We started our experiments by using available physical annotation means provided by the Alive monitor (a binary button). Based on the received feedback from experiments and user interviews, we realized that there is a lot of value in allowing end users to self-annotate their data with their own words, as it makes it much easier to remember what was going on at a certain moment as well as reflect on what has happened before, after, who was there, why she put that annotation and so on. This insight led us to introduce the widget-based annotation means presented in Section 3.5. While coming out of a need to identify interesting moments in time, the interaction means provided by our platform became an interesting study on what goes on in the process of annotating, as users became more aware of what was really the most meaningful description of the situation at hand. Even more, it became obvious that given such tool, end users will try not to replicate information recorded through the AIRS sensors, such as location, focusing instead of descriptions hard to capture through automatic means.

Apart from this specific input regarding annotation, our experiments generally showed the value of having an extendable, controllable and interactive mobile-based sensing platform, as it allowed us to collect and correlate user meaningful lifestyle information both automatically (objective) and human-driven (subjective) instead of using commonly available methods, such as periodic polling or questionnaires.

However, within such recording scenarios, battery consumption becomes a real challenge as it affects the length of the recording as well as the likelihood of users performing such recordings with their own mobile phones. An obvious route to obtaining an insight into battery consumption is through experiments but measuring battery consumption within real-life scenarios is riddled with challenges. Firstly, the used devices are of personal nature (in contrast to purpose-built sensing devices) and each user has different, often parallel usages. Also, each user's environment and movement patterns differ, making statements about using features such as GPS, WLAN or BT futile since the exact environment of the experiment (defined by effort it takes to obtain a GPS fix, the number of access points or BT devices as well as the frequency of scanning) cannot be kept identical between users or even the same user within different situations. Hence, battery statistics are bound to vary significantly.

Furthermore, the variety of available handsets makes any study regarding battery consumption difficult since consumption will inevitably vary according to processor generation, radio chipset and radio environment (such as positioning of the antenna in the case of WLAN or BT) and even OS configurations. Hence, battery statistics can at best be given for certain (reference) devices.

Also, the general consumption caused by the various callback sensors is very difficult to normalize since their consumption will heavily depend on the particular rate of triggering the callbacks. Given the nature of the information (such as battery charging, handset plugged in/out, change in radio signal), this rate inevitably depends on the particular usage scenario and any artificially defined usage scenario is therefore of little value to understanding the overall consumption expectation. In all this, the configurability of the platform adds additional variance to any statement of battery consumption.

For these reasons, we present here results from experiments within a lifestyle recording scenario, where the mobile phone is used by a single person within a realistic setting over a month, in comparison with a more controlled recording scenario that only focused on recording 3 of the most battery consuming sensors: GPS, WiFi and Bluetooth in relation to location (a 'wardriving' scenario [19]).

The lifestyle scenario involved one of the authors using the platform during one month of usual usage of his personal mobile phone. Information recorded included GPS, BT, noise level as well cellular information (signal strength, location area, cell identifier), activity information (headset status, mood and event widget input, call as well as SMS information) and system information (RAM, battery, tasks running, music played, files created). GPS and Bluetooth were configured for 30 seconds updates while surrounding noise was determined every three seconds (recording for one second to determine the noise level). With this, we generated a moderate to heavy load created by our platform. The end user made use of his handset within the typical range of activities, including synchronizing content frequently during office hours (from 9am to 7pm). The data is averaged over a month and includes activities from office work over home working to international travel. Recording was conducted from about 9am to 8pm, on occasions longer when there were late evening activities.

Our diary experiment was conducted with a Galaxy Nexus on Android 4.0.2, while two Samsung Galaxy S with Android 2.3.6 were used for the 'wardriving' scenario, carried at the same time to encounter similar environmental conditions. In the latter case, the handsets differed in their configuration of the polling interval (15 seconds for the 'heavy' and 30 seconds for the 'light' case). In order to emulate a dedicated wardriving usage, the handsets were not used throughout the measurements for anything else, eliminating any variance through user usage.

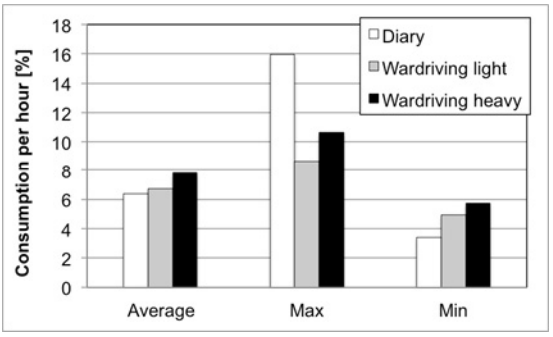

Fig. 4. Battery consumption in various scenarios

Figure 4 shows the battery consumption for these usage-based experiments. The diary use case results in a larger variance since the handset was normally used (the maximum value, for instance, is caused by a prolonged browsing session during a domestic travel). On average, the platform consumed about 6.3% battery per hour for the activity recording, with an average battery consumption of the phone without recording at around 2.5%. With that, such recording is possible throughout a normal

working day (of, say, about 12 hours) without recharging. Although less callback sensors are used in our second scenario, the usage of WLAN (in exchange for the noise recording) leads to an increase in consumption. We explain this with the necessary wakelock [20] on the WLAN radio in order to perform the frequent scanning. Hence, WLAN never switches off. We can see that increasing the polling interval for WLAN only leads to a small increase from 6.7 to 7.7%.

The takeaway from our experiments is that the battery consumption of our platform is moderate even in experiments that record a significant number of sensors. Using wireless radio resources increases the overall battery consumption, which is expected. This is even more the case when using, e.g., BT-attached sensors like the ones in [9]. Their individual consumption, however, heavily depends on the used radio protocol as well as the rate of communication. Newer technologies, such as BT 4.0, are expected to reduce power consumption for these scenarios.

5 Conclusions and Future Work

Given the almost ubiquitous availability of mobile handsets as well as their ever-increasing capabilities, utilizing their power is desirable for many mobile sensing scenarios. This is especially the case within the lifestyle management area that is concerned with increasing self-awareness through self-monitoring, information processing and visualizations. In order to focus research and development on what matters, namely the intelligence to make use of the increasing pool of information that could be gathered, a platform approach is essential as it can accommodate individual or group requirements. Although we see the area of self-monitoring through mobile phones taking off (both in research and the mobile application area), mainly fragmented, short-lived or purpose-oriented solutions are created.

There are currently few generic and widely available mobile device based sensing platforms that provide the wide range of features we have described, combining both automatic as well as user-based information gathering, perfectly suited for self-monitoring scenarios where not everything of value to users can be sensed or recognized automatically. In this paper we provided the main challenges we have encountered in our work together with several design and implementation choices we made in order to address them. We specifically addressed one of the essential challenges of any mobile-based sensing platform, which is battery consumption. Our experiments show that the platform allows for sustaining daily recording activities over a wide range of information without significantly degrading the overall device performance.

In order to establish the platform as a possible basis for research and development activities alike, we released the work to the open source community as well as to the general software market [22] as free software. At the time of writing, more than 4000 users have downloaded the application with more than 400 active installations.

Apart from general application developers, we see the research community at large as a beneficiary of our work as the platform can be immediately downloaded and used, allowing researchers to focus on processing recorded information. We also see

our support for interaction as being useful in various user research studies or even for aiding automatic recognition of certain situations.

For our future work, we intend to focus on the information processing and visualization aspects involved when gathering such a multitude of information. Story-based approaches [18] to presenting information have so far yielded promising feedback from end users with many ideas for extensions. These ideas include correlating existing sensors with any type of media created during recording (e.g., pictures and videos taken). We also plan on extending the support for the wider community by enabling the addition of Handlers without the need to re-compile and re-install the platform. This will allow for establishing code repositories, which can be enriched over time by the wider community. These extensions are planned in collaboration with the wider research community, initiated through our software market and open source release. To foster this engagement with the community, we have set up a dedicated blog platform as well as an online manual that is directly accessible through the mobile application. We also provide increasing insight into example handlers with the attempt to encourage the development of novel extensions to the core platform. We also plan on making available code repositories for the wider community where handlers can be downloaded for free in order to optimize the AIRS platform for any experiment that is planned by community members. The most important community engagement, however, is the usage of the platform as well as the reporting of its usefulness, potential bugs and errors as well as suggestions for extensions. Our current online blog platform provides the means for this interaction through feature requests, blogging about new features, and often encountered Q&As.

References

1. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. IEEE Pervasive Computing 04(2), 51–59 (2005)
2. Trossen, D., Pavel, D.: NORS: An Open Source Platform to Facilitate Participatory Sensing with Mobile Phones. In: Conference on Mobile and Ubiquitous Systems: Networking and Services (2007)
3. Siewiorek, D., Smailagic, A., Furukawa, J., Krause, A., Moraveji, N., Reiger, K., Shaffer, J., Wong, F.L.: SenSay: A Context-Aware Mobile Phone. In: Seventh IEEE International Symposium on Wearable Computers (2003)
4. Sung, M., Pentland, A.: LiveNet: Health and Lifestyle Networking Through Distributed Mobile Devices. In: Workshop on Applications of Mobile Embedded Systems, MobiSys (2004)
5. Sportstracker (2010), `http://www.sports-tracker.com/#/home`
6. Endomondo (2012), `http://www.endomondo.com`
7. Carloni, L.P., De Bernardinis, F., Pinello, C., Sangiovanni-Vincentelli, A.L., Sgroi, M.: Platform-Based Design for Embedded Systems. The Embedded Systems Handbook (2005)
8. Trossen, D., Pavel, D., Singh, J., Bacon, J., Guild, K.M.: Information-centric Pervasive Healthcare Platforms. In: Pervasive Health Conference (2010)
9. Alive Technologies, “Alive Heart and Activity Monitor” (2010), `http://www.alivetec.com/products.htm`

10. WristCare, http://www.istsec.fi/eng/Emikakoti.htm
11. SenseWear BMS (2010), http://www.sensewear.com/BMS/solutions_bms.php
12. Philips Lifeline solutions (2010), http://www.lifelinesys.com/content/home
13. iFall (2010), http://www.imedicalapps.com/2010/04/ifall-android-medical-app/
14. OBS (2010), http://www.obsmedical.com/products
15. CardioNet patient solutions (2010), http://www.cardionet.com/patients_01.htm
16. Rachuri, K.K., Rentfrow, P.J., Musolesi, M., Longworth, C., Mascolo, C., Aucinas, A.: EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In: ACM Ubicomp (2010)
17. PAL project (2012), http://www.palproject.org.uk
18. Pavel, D., Callaghan, V., Dey, A.K.: Supporting Wellbeing Through Improving Interactions and Understanding in Self-Monitoring Systems. In: Handbook of Ambient Assisted Living – Technology for Healthcare, Rehabilitation and Well-Being, vol. 11. IOS Press (2012)
19. Wikipedia, "Wardriving" (2012), http://en.wikipedia.org/wiki/Wardriving
20. Android Developer online resources (2012), http://developer.android.com/index.html
21. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A Survey of Mobile Phone Sensing. Comm. Mag. 48, 140–150 (2010)
22. AIRS: Android Remote Sensing platform (2012), https://play.google.com/store/apps/details?id=com.airs
23. SENSEI FP7 project (2012), http://www.sensei-project.eu/