ICST Transactions
on Mobile Communications and Applications Research Article ICST.ORG

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

D. Trossen[1,*] and D. Pavel[2]

[1]TecVis LP, Colchester, UK
[2]TecVis LP, Colchester, UK

## Abstract

Utilizing mobile devices for gaining a better understanding of one's surrounding, physiological state and overall behaviour 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.

Received on 02 April 2013; accepted on 05 September 2013; published on 16 December 2013



doi: 10.4108/mca.1.3.e8

## 1. Introduction

The importance of mobile devices and their capabilities has long been recognized within research projects such as [1-4, 22] 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 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 provides valuable support for user research. While other mobile-based sensing platforms have been developed during the years (e.g., [1][3][4][29]), 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 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 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

* Corresponding author. Email: dirk.trossen@cl.cam.ac.uk

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, 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 and provide an outlook how a platform like AIRS can contribute to the wider understanding about one's environment. We also briefly provide two examples for using our platform, namely processing and visualising recordings directly on a mobile device as well as controlling device functions. 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 user information across various context dimensions, such as physiological, spatial, social, environmental, or emotional [17]. The main goal of our system was to provide 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 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 [20] 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][14]. 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 gateway), a mobile sensing platform has to be designed with *extensibility* in mind, as also argued in [8]. A challenge that comes from the increasing complexity of mobile devices is the impossibility of anticipating all malfunctioning scenarios. Therefore, it is important to ensure *persistence* of the measurement itself as well as for its recordings, e.g., through automatic restarting in failure cases and emergency data saving. This is particularly important for long-running experiments.

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 we can observe the trend 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, sharing must happen under the user's control.

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 since Android platform release 4.0.4 (Ice Cream Sandwich); (4) a substantial user base, given the wide adoption of Android as a smartphone platform.

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 behaviours, 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, secure 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.

Since we directly base our platform on the Android design and implementation guidelines, as outlined in the SDK [19], any interaction with the end user is implemented as a so-called *Activity* [19]. 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* [19]. 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.

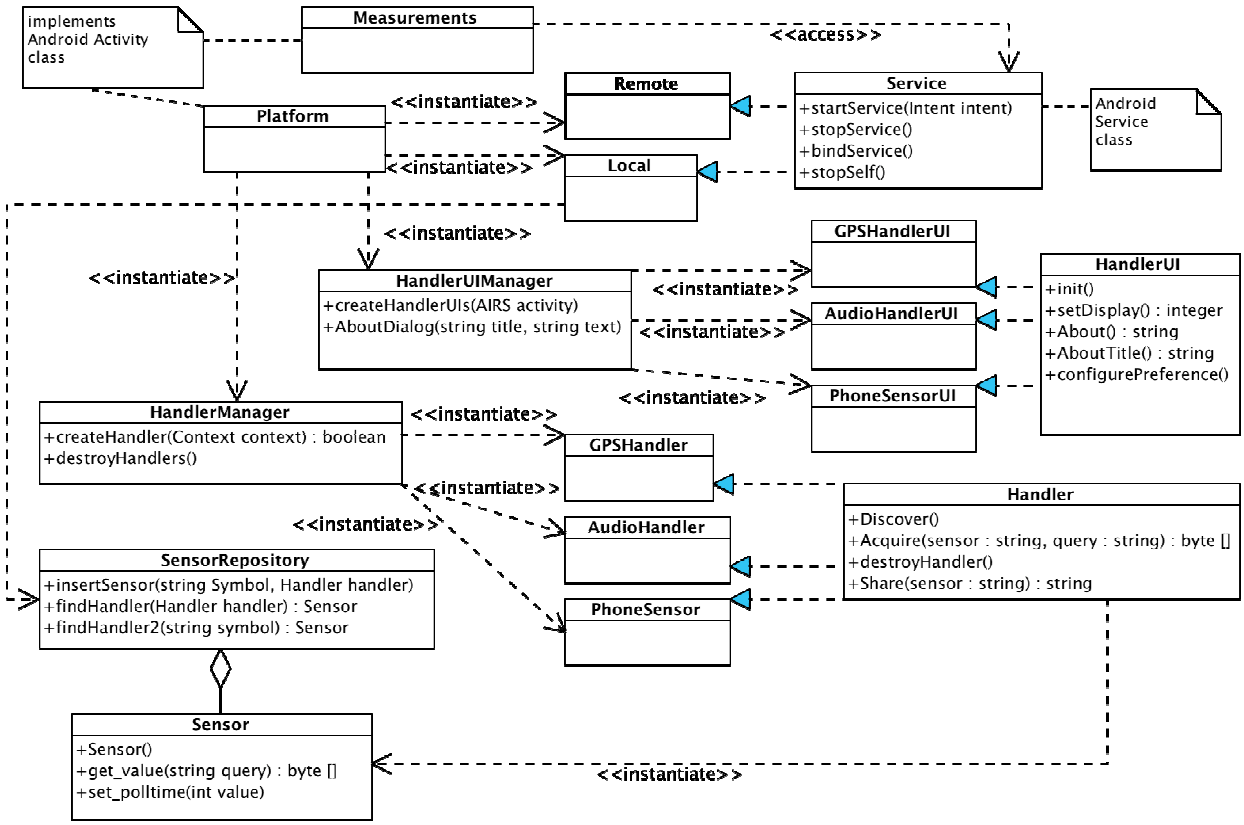


**Figure 1.** AIRS implementation diagram

## 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. 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 normally. Given the inherent challenges of determining 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 support sensors that can be attached via Bluetooth, such as the Zephyr HxM heart rate 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 [17]. 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 Zephyr heart rate monitor [9] by implementing the particular BT-level protocol. This Handler then provides access to three different sensors supported 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, 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. With this, we aim at providing information with minimal resource consumption, similar to system-level logging and analytic tools. Secondly, any Handler should verify the existence of any necessary resource before using it, catching any 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.

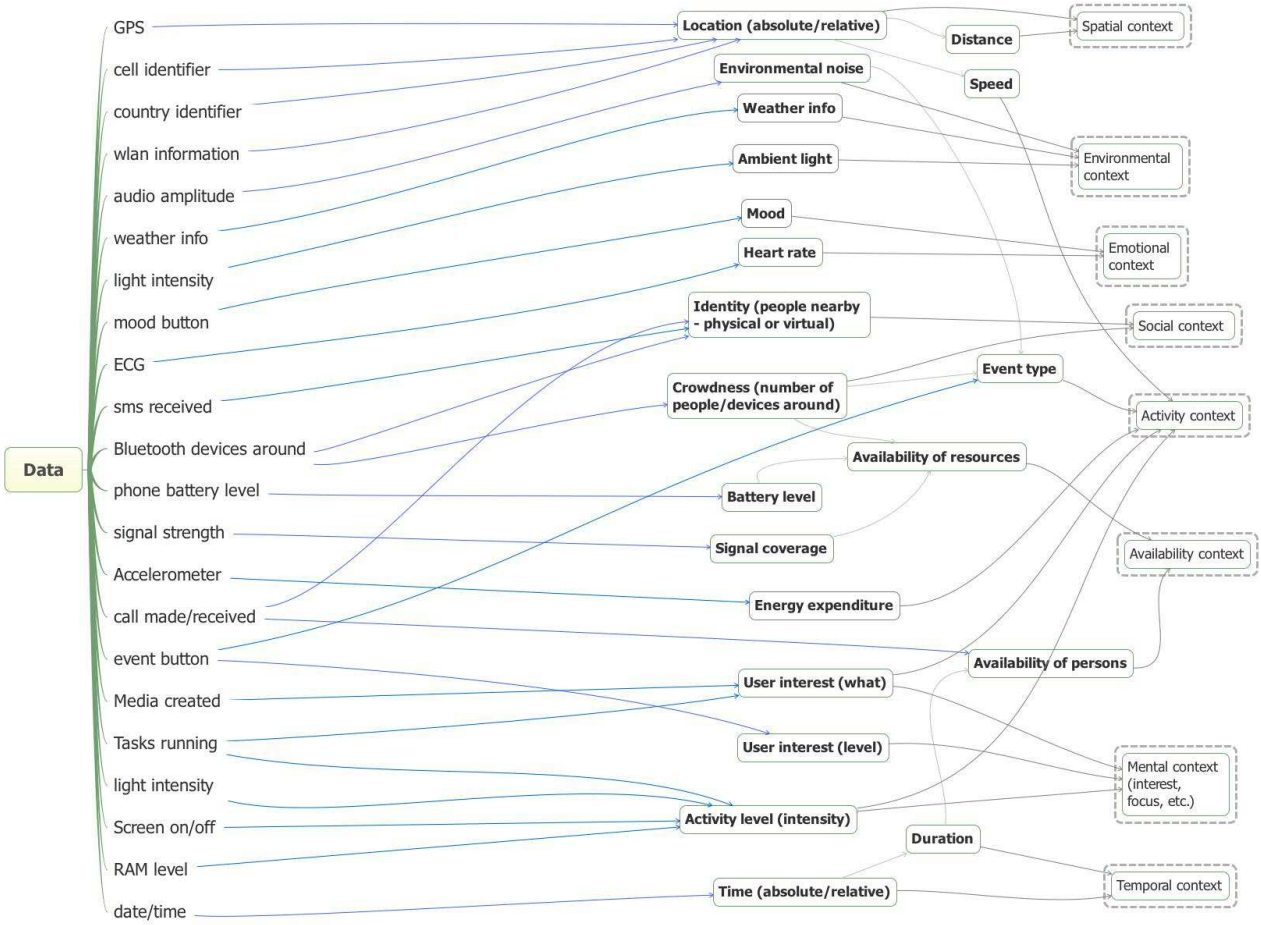


**Figure 2.** Sensors supported by AIRS and types of processed information

## 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 temporarily generates text 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 given 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 [17].

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 for reducing battery consumption. 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* [19] 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 the 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 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 provides 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 realised through polling, GPS is 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. Furthermore, the GPS handler provides an *adaptive mode* where GPS activity is entirely suspended when nearby known WLAN networks, which significantly reduces battery consumption in stationary scenarios.

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 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.

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 for adjusting its overall operation. Each *HandlerUI* implementation makes use of the Android concept of a *PreferenceActivity* [19], which minimizes any necessary code for the 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).

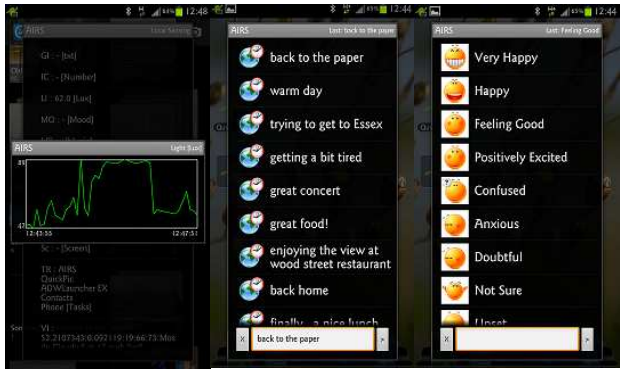


**Figure 3.** AIRS Screenshots: (a) visualization; (b)(c) annotation widgets

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. For this, we utilize the concept of an Android Widget [19] 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 BT-attached sensors, such as the Alive heart rate monitor [23] (via 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 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 battery consumption.

For these reasons, we present here results from experiments within a lifestyle recording scenario, where the mobile is used by a single person within a realistic setting over a month, in comparison with a more controlled scenario that only focused on recording 3 of the most battery consuming sensors: GPS, WiFi and Bluetooth in relation to location (a 'wardriving' scenario [18]). 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 frequently synchronizing content 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 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.

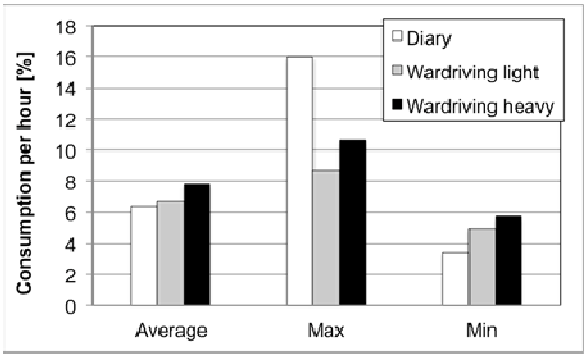


**Figure 4.** Battery consumption in various scenarios

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* [19] 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][23]. 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. From recording to understanding

We now present two examples for applications built on top of the AIRS platform. Our first one facilitates the move from recording lifestyle-related parameters to understanding what and why something has happened. For this, we extend our discussion regarding the sensors that can be recorded in AIRS (see Figure 2) by elevating the recorded information onto the level of possible context information. This processing of information can provide deeper insights into activities of users, their surroundings and spatial context, their social interactions as well as their emotional state. Information at that level can be utilised for various scenarios. For instance, in the user studies performed in [24] the usages for AIRS recordings[†] ranged from memory recollection over stress management to outpatient-like monitoring. In all of these scenarios, various context dimensions of Figure 5 were utilised to help users understand the recorded information.

Crucial to understanding the recorded information is, ultimately, its presentation to the end user. The work in [17][24] outlines a *storytelling* approach for visualising lifestyle recordings to end-users that relies on a narrative approach. Here, the various context dimensions shown in Figure 5 are visualised to the user through a sequence of *meaningful events*, each of these events enriched with (processed) context information that aids the understanding of the user as to what happened around this event. The visualisation uses engaging multimedia elements in the form of icon graphics as well as backgrounds, complemented by textual descriptions of the context information.

In an attempt to bring such processing and visualisations means to mobile users, we have released the mobile application *Storica* [25], which provides similar

[†] The experiments in [24] included information from other platforms than AIRS, such as desktop activities as well as stand-along physiological sensors. AIRS recordings, however, were central to all user experiments.

processing and visualisation as shown for the desktop system in [24] but optimised for mobile devices. Figure 6 shows examples of the visualisation means that are provided by Storica. Access to the AIRS data is provided through a calendar-based entry screen (Figure 6a). For each day, the user can visualise her personal story of events (Figure 6b), display a track of movements (Figure 6c), enriched with context information for each position along the track as well as enjoy her captured media in a context-enriched manner. Around each event or position, the user can view detailed information of the included context through timeline graphs, tag clouds as well as call and SMS logs.

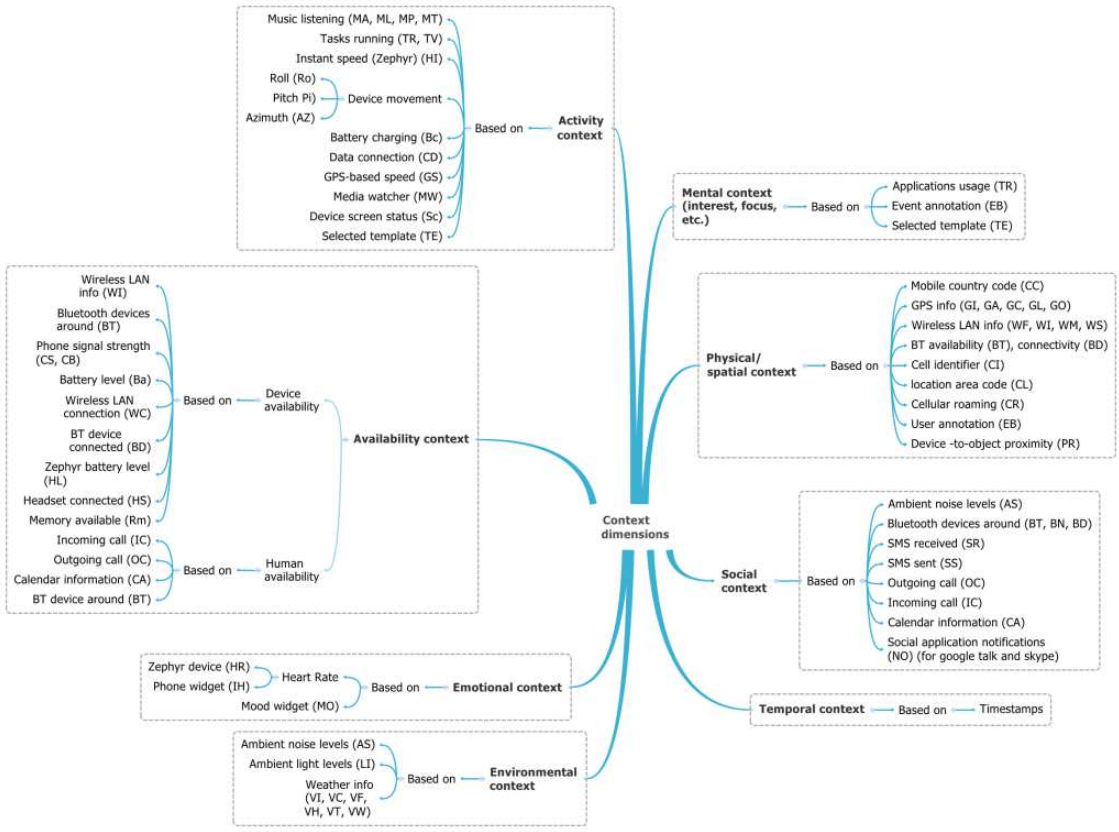


**Figure 5.** Possible contexts to be determined from AIRS recordings

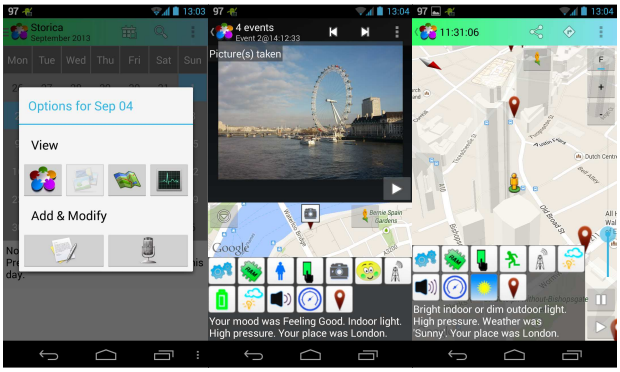


**Figure 6.** Visualisations in Storica: Calendar-based entry (a), story-based view (b) and map-based view with fly-over effects (c)

While being in an early stage of deployment, Storica provides an insight how AIRS recordings can directly be enjoyed by end users on their recording device, while having the option to synchronise the information to their laptop in order to integrate with the desktop system as described in [24]. Storica supports experiences across several user devices, the sharing across a user group through available social networks as well as enriching the personalisation of the experiences through changing the settings of stories presented to end users by supporting so-called *diary modes*. In the future, we will explore further adaptations of the presentation to the user based on dedicated application use cases, such as patient diaries, mood diaries and alike.

## 6. From recording to controlling

Our second example of an application that can be realised on top of the AIRS platform is a control application, utilising the recorded information to vary mobile device parameters and settings.

AIRS Ruler [26] is an application example in which the end user can define simple rules that act upon AIRS recordings with various system changes, such as switching Bluetooth or WiFi radio or changing the desktop wallpaper. Envisioned scenarios for such application can be to show the AIRS annotated mood information as a specific wallpaper (e.g., *if mood is happy, show wallpaper 1*) or control the WiFi radio based on user annotations (e.g., *if event is travelling, then switch off WiFi*). Rules are defined through a graphical UI, which allows for selecting supported sensors and actions as well as combining conditions with AND/OR operators (see Figure 7 for example screenshots).

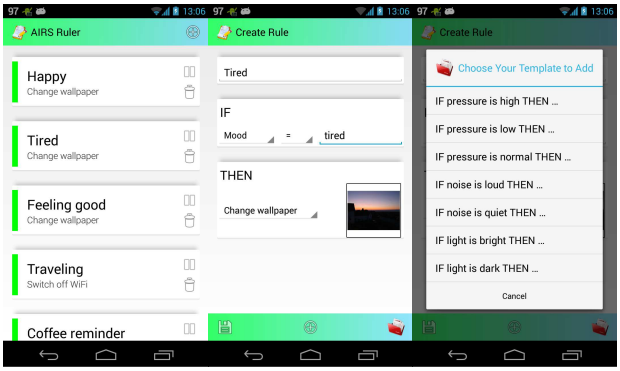


**Figure 7.** Rule definition in AIRS Ruler (a) list of rules (b) individual rule (c) templates for ease-of-use

In contrast to the Storica example in Section 5, AIRS Ruler does not utilise the database of AIRS itself but relies on so-called *AIRS intents*, based on common Android system intents [19]. These intents are lightweight Android mechanisms to carry data from one application to another. In our case, AIRS sends one such intent message for each recording[‡], while AIRS Ruler only subscribes to the intents of those sensors it needs to execute its rules. With that, AIRS Ruler is only executed when a required sensor has been recorded in AIRS and it demonstrates how to easily implement applications on top of AIRS, while leaving the efficient retrieval of the information to the AIRS platform.

AIRS Ruler has been released to the public in addition to the AIRS platform itself as a demonstrator for the capabilities of AIRS.

[‡] The end user agrees to the sending of these intents in the AIRS settings and can therefore switch off this mechanism, e.g., for privacy reasons.

## 7. 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 device performance, placing us on par with optimised system-level logging capabilities such as Google Analytics, while enabling a continuous monitoring of a wide range of information.

In order to establish the platform as a possible basis for research and development activities, we released the work to the open source community as well as to the general software market [21] as free software. At the time of writing, more than 12000 users have downloaded the application with more than 1600 active installations. To foster this engagement with the community, we have set up a dedicated blog platform as well as an online manual that is accessible through the mobile application. We also released a comprehensive JavaDoc documentation [27] as well as established a forum for developers and users alike [28]. With the latter, we provide insight into example handlers, provide the ability to issue feature requests and send bug reports, all with the attempt to encourage the development of extensions to the core platform. The most important community engagement, however, is the usage of the platform as well as the reporting of its usefulness for which we utilise our blogging platform.

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 extend the information processing and visualization means already provided by our Storica application as well as extend the desktop-based system described in [24]. 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. These extensions are planned in collaboration with the wider research community, initiated through our software market and open source release.

## Acknowledgements.

Some of the work described in this paper has been funded by EPSRC and TSB through the PAL project (grant number TP/AN072C), a research project investigating future healthcare services in the context of self-monitoring and lifestyle management. Dirk Trossen was with Cambridge University and Dana Pavel with University of Essex at the time of this research.

## References

[1] Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications, IEEE Pervasive Computing, vol. 04, no. 2, pp. 51-59, Apr-Jun (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., Fei Lung Wong: 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, http://www.sports-tracker.com/#/home, (2010)

[6] Endomondo, http://www.endomondo.com, (2012)

[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] Zephyr HxM heartrate monitor, http://www.zephyr-technology.com/products/hxm-bluetooth-heart-rate-monitor/, (2012)

[10] WristCare, http://www.istsec.fi/eng/Emikakoti.htm

[11] SenseWear BMS, http://www.sensewear.com/BMS/solutions_bms.php

[12] Philips Lifeline solutions, http://www.lifelinesys.com/content/home, (2010)

[13] iFall, http://www.imedicalapps.com/2010/04/ifall-android-medical-app/, (2010)

[14] OBS, http://www.obsmedical.com/products, (2010)

[15] CardioNet patient solutions, http://www.cardionet.com/patients_01.htm, (2010)

[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] 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 & Well-being, IOS Press, Vol. 11, (2012)

[18] Wikipedia, "Wardriving", http://en.wikipedia.org/wiki/Wardriving, (2012)

[19] Android Developer online resources, http://developer.android.com/index.html, (2012)

[20] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A. T.: A Survey of Mobile Phone Sensing. In: Comm. Mag., 48:140–150, (2010)

[21] AIRS: Android Remote Sensing platform, https://play.google.com/store/apps/details?id=com.airs, (2013)

[22] SENSEI FP7 project, available at http://www.sensei-project.eu/, 2012

[23] Alive Technologies, "Alive Heart and Activity Monitor", http://www.alivetec.com/products.htm, (2010)

[24] D. Pavel, V. Callaghan, A. K. Dey, F. Sepulveda, and M. Gardner, "The Story of Our Lives: From Sensors to Stories in Self-monitoring Systems," in 4th Computer Science and Electronic Engineering Conference (CEEC'12), 2012.

[25] Storica: Experience your Life, https://play.google.com/store/apps/details?id=com.storica, (2013)

[26] AIRS Ruler, https://play.google.com/store/apps/details?id=com.airsruler, (2013)

[27] AIRS documentation, http://tecvis.co.uk/software/airs/developer-information, (2013)

[28] AIRS forum, http://tecvis.co.uk/forum, (2013)

[29] mCrowd, http://crowd.cs.umass.edu/team.php, (2013)