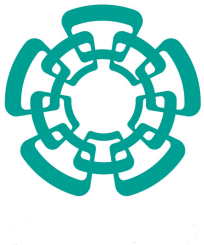
Smart Usage of Context Information for the Analysis, Design, and Generation of Power-Aware Policies for Mobile Sensing Apps



Technical Report

LTI-TR-2014-07

Student Rafael Pérez-Torres

Advisors César Torres Huitzil Phd, Hiram Galeana Zapién Phd.

Information Technology Laboratory,
CINVESTAV Tamaulipas

Abstract

TBW (To-be-written)

Keywords. Provide five key words or phrases that describe your topic, methods, and results.

1 Preamble

This sections describes the introduction to the problem, the problem statement itself, objectives and hypothesis of the thesis work.

1.1 Introduction

In recent years the smart devices have seen an increasing usage by people in every day activities. According to the Ericsson Mobility report [17] there were 1,900 million of smartphone subscriptions and 300 million of mobile PCs, tablets and mobile router subscriptions in the 2013. It is expected to have 5,600 million and 700 million of subscriptions, respectively, by the end of the 2019.

For people it is common to employ a smart device, like the smartphone, to perform work activities such as telephony tasks (texting, calls), sending emails and surfing the web, or even for entertainment purposes like playing videogames or multimedia resources. Here, the important idea is the acceptance and adoption of smartphones by society which transform these devices in a direct channel to obtain information and establish communication with people [14].

Besides its Internet-enabled features, a typical and modern smartphone also includes a set of physical sensors and additional circuitry that allows to improve the interaction with user. In this way, it is possible to detect the orientation of the phone and adapt the screen accordingly or play the next song by shaking the phone. Moreover, the inclusion of sensors in smartphones opened the path for performing computation considering aspects of the user environment. When smartphones use the data delivered by sensors to create a representation of the environment where the user is and employ such representation in their behavior they become *context-aware*.

The term *context* refers to the set of environmental states and settings that either determines the application's behavior or in which an application event occurs and is interesting to the user [6]. A context-aware mobile app is such that adapts its behavior based on changes detected in any source of contextual information. An important subset of context-aware mobile apps is composed by the location aware apps, also known as location based services (LBS) [8, 24]. This family of mobile apps focus on the detection of changes in the location data of the device and adapting its behavior accordingly.

Both location and context aware mobile apps share the behavior of access sensors to become aware and react accordingly. From the hardware usage perspective, these apps can be categorized as mobile sensing apps, which represent a current trend of research in the mobile computing area [3, 9]. A mobile sensing app is such that its behavior relies on analyzing data that is collected from sensors over long periods of

Feature	Average power consumption (watts)
Processor (1%)	0.06
Processor (100%)	0.41
Accelerometer	0.05
Bluetooth	0.28
Microphone	0.26
Screen	0.23
Wi-Fi scan	1.37
GPS	0.32
3G radio (idle)	0.47
3G radio (sending)	1.11

Table 1.1: Average energy consumption of a Nokia N95 smartphone, from [8]

time. Typically, the analysis processes performed over data consist in classification tasks for detecting specific patterns that describe user activities. Thanks to this, the range of applications that mobile sensing apps have is wide and it is even increasing due to the addition of new sensors to mobile devices.

1.2 Motivation

Among the many reasons that lead to the success and acceptance degree of smartphones, the next ones are the most influential:

- Their Internet enabled features.
- Their increasing computing and storage capabilities.
- The diversity of sensors embedded on them.
- The possibility of installing new mobile applications, or *apps*.

Despite all of these benefits, it should be noted that the more computing, storage, communication, and sensing technologies included in smartphone, the more its energy consumption. Table 1.1 shows the average power consumption of the main embedded components of the Nokia N95 smartphone. It can be identified that wireless communication interfaces, GPS and screen are the most energy consuming elements. Such situation is typical in most of the mobile platforms.

Unfortunately, the current advances in battery technologies are not evolving at the same pace than the rest of electronic components [23] of the smartphone. In fact, battery is only growing up to 5% each year according to [11]. In this sense the energy is a scarce, limited, and competed resource for any mobile platform [15]. As in the case of any other resource, the energy requires efficient techniques for its management considering that, different than other resources, once a unit of energy is employed it

can not be reused in future [21]. The energy cannot be longer considered an optional issue but a key component for mobile app development [12].

1.3 Problem statement

Typically, mobile sensing apps access sensors in a continuous way over long periods of time. This represents a high energy consumption due to task duration and the overhead generated by turning sensor on and off. Such usage may lead to a quick battery drain that prevents smartphone utilization for other activities.

Additionally, processors of current smartphones are designed for managing the heavy interaction with user and the execution of mobile apps. A continuous sensor reading is out of their actual scope and because of this a large waste of energy is generated when the processor is only active for instructing sensor readings [16].

Also, current mobile platforms do not include out of the box mechanisms to access sensors periodically. API's[1] offered by manufacturers only provide support for basic tasks, such as turning sensors on and off and reading data from them, but ignore mobile app's business logic. Mobile app requirements (like precision in data being read), smartphone constraints (like battery level) and additional information are ignored by mobile OS[2].

Therefore, there is the need of a specialized framework that considers previous elements and allows the generation of smart policies for performing continuous sensor readings. A policy is a high level rule that defines the usage that sensors should observe in order to keep low energy consumption and accomplish mobile app requirements.

The *smartness* of policies can be achieved by leveraging the user's context obtained from data delivered by sensors. At low level, the context information can be identified by a *pattern identifier* mechanism fed by raw data coming from sensors. For example, the element being identified may refer to a mobility pattern useful for generating a policy to access GPS; if the pattern describes motion at high speed, the policy may instruct GPS readings more frequently than if user is moving at low speed.

Hence, the pattern becomes a descriptor of user's context, and can be the input for a *policy generator* mechanism that generates the policy that adapt the sensor usage, reduce the energy consumption and achieve mobile app objectives.

A relevant aspect in the generation of these smart policies is the need for energy and precision hints. Those are necessary since sensor usage can only be improved

[1] API refers to Application Programming Interface, which is a library with a set of methods that performs logical related tasks. This library is available to programmers for software construction.

[2] Mobile OS, Mobile Operating System.

when the mobile app specifies the precision level required. This precision level dictates the granularity in user activity tracking. However, the finer the granularity, the higher the energy consumption. Because of this, there is a trade-off between the precision and the energy consumption of mobile sensing apps.

The proposed tesis aims the creation of needed mechanisms for the policies generation, and their implementation using the the GPS receiver and mobility patterns as proof of concept elements.

The problematic detected can be achieved by dividing it into two main problems, the pattern identification and the policy generation. The pattern identification problem refers to the detection of a pattern in the data delivered by sensors. This pattern helps to obtain information about user's context which is helpful to add smartness to the policy generation process.

On the other hand, the policy generation problem is related to the definition of a new duty cycle for accessing sensors. This new duty cycle should reduce energy consumption and at the same time address the mobile app requirements.

1.3.1 Pattern identification

Given a set $V = \{v_1, v_2, \ldots, v_n\}$ of data values read from sensor S in the time interval $T = [t_1, t_2]$, find the behavior pattern $Pattern_S$ that represents the activity of user.

$$PatternIdentifier(V) \longrightarrow Pattern_S \in Patterns \tag{1.1}$$

Where $Patterns$ is a set of patterns that represent an interesting state in the user activity. For instance, considering mobility data obtained from the GPS receiver, the set $\{no\_movement, walking, running, vehicle\_transportation\}$ can represent these interesting states. The set of patterns is to be defined and conforms a step in the methodology.

1.3.2 Policy generation

Given the detected pattern $Pattern_S$ in data from sensor S, parameters for assigning weight to energy eh and precision ph, and physical constraints status pc of a mobile device, find a policy to adapt the duty cycle of sensors.

$$PolicyGeneration(Pattern_S, eh, ph, pc) \longrightarrow DutyCycle_S \tag{1.2}$$

The policy will be generated considering the trade-off between energy and precision parameters that are specified by the mobile app, since both factors have an implicit impact on each other.

1.4 Hypothesis

Smart policies generated through contextual information can be employed to reduce the energy consumption in a mobile device when performing continuous sensor readings.

1.5 Objectives

1.5.1 Main objective

Reduce the energy consumption when performing continuous sensor readings in mobile devices by making use of context information.

1.5.2 Particular objectives

- Identify behavior patterns which can provide meaningful context information from raw data collected by sensors.
- Generate smart policies for sensor usage from context information, mobile app requirements and mobile device constraints.

1.6 Contributions

- A mechanism for detecting patterns from the data read by sensors of mobile devices (specifically the GPS receiver). These patterns represent information about user's context.
- A mechanism for generating policies for accessing sensors. Such mechanism considers application requirements (energy and precision hints), level of mobile device constraints, and context about user situation (using the pattern detected by previous mechanism). These smart policies will allow to reduce the energy consumption while accessing sensors in a continuous way.
- A software element able to read data from sensors using the policies generated by the described mechanisms and transmit these data to an external server.

2 Theoretical framework

This section describes basic aspects about terms and elements involved in the problem statement which are also fundamental for a better comprehension of the state of art of this research.

2.1 Context aware computing

Since their introduction to market, mobile devices have evolved from basic cell phones and media players to complex and sophisticated devices like smartphones [5,20]. Sensors, advanced wireless communication interfaces and other technological improvements have provided them with more features to be employed in any daily activity.

Furthermore, this set of computing elements have been responsible of the invention of the mobile computing area. Mobile computing are those computing tasks that can be executed in portable devices while in motion, without losing their computing abilities and accessing to resources at remote locations through wireless networks. That is, compute tasks anytime and anywhere.

After the invention of mobile computing, the notion of location and context in terms of mobility emerged. Location is a simple concept indicating the position of the user in a referenced system. The system can be such abstract as the GPS or very specific like indoors, e. g. the cafeteria at university campus.

On the other hand, context is a broader and more generic concept. Oxford dictionary defines concept as:

> The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood.

This definition gives an idea of its abstractness.

Regarding to mobile computing, context still being a non-fully established term. In [2] several definitions of context are listed:

- Context contains information addressing "where you are, who you are with, and what resources are nearby", [Schilit et al. 1994].
- Context contains "any information that can be used to characterize the situation of an entity", [Dey and Abowd 2000].
- Context comprises "elements for the description of this context information [that] fall into five categories: individually, activity, location, time, and relations", [Zimmermann et al. 2007].
- Context is the "set of variables that may be of interest for an agent and that influence its actions", [Bolchini et al. 2009].

- Context is "a four-dimensional space composed of *computing context*, *physical context*, *time context*, and *user context*", [Chen and Kotz 2000].

This research considers the latter definition along its development.

2.2 Mobile sensing apps

Current trends show there is a large set of applications on which smartphones can be used for. Most of these applications leverage the smartphones' mobility features and their ability to sense the environment, which turn them into *omni-sensors* [15].

Given the chance to perceive environment, data coming from sensors can be analyzed to detect patterns that represents information about user's context. The context then can be used to feedback user with information about his or her activities and adapt smartphone's behavior accordingly.

As stated, the set of mobile apps that are able to perform tasks related to data collection from sensors and discovering information are called *mobile sensing apps*. Take bullets off

The improvements in data analysis algorithms and similar techniques employed by mobile sensing apps locally at the smartphone and / or in the cloud are increasing the *smartness* of these devices day by day. In this way, smartphones can get involved in high level logical activities, not only in simple sensor readings or abstract data analysis tasks.

Because of this, the notion of smartphone will be upgraded to *cognitive-phone* [3]. This new class of devices will be able to understand people life patters, reason about their health and well-being and even intervene on their behalf.

2.3 Stages of mobile sensing apps

The duty of mapping raw data into meaningful information for user can become quite complex. This complexity level depends on the specific characteristics of the sensor being used and the mobile app requirements.

Although the differences in mobile apps and the sensors embedded in mobile devices, some steps are common in their internal functionality. The list of these steps include the stages of sensor reading, an optional filtering, feature extraction, classification, and post-processing [18]. These stages can form a loop, as seen in Figure 2.1, that powers up the mobile app.

A basic description of the stages is as follows:

Sensor reading It includes the request to sensors for reading data from the environment. Common tasks performed in this stage are windowing and framing.

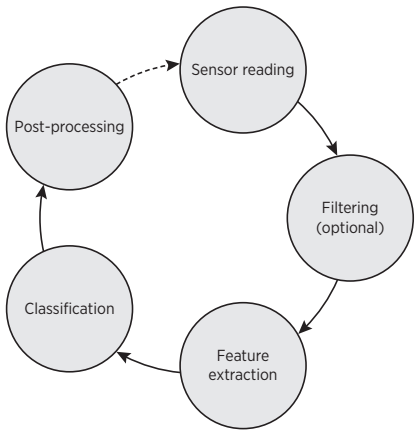


Figure 2.1: Common stages of mobile sensing apps.

Filtering (optional) In this stage is possible to filter out uninterested data, like outliers or noise. Filling data gaps can also be done. The purpose of the stage is to prepare a clean and uniform set of data for its use in further steps.

Feature extraction It performs the extraction of features that helps to discriminate between activities in the classification stage. The feature extraction to be employed depends on the type of mobile app being developed. The features to be computed can be:

Deterministic Simple deterministic transformations of raw sensor data, for example frequency content.

Probabilistic Calculated with probability measures, for example the user likelihood of being in a certain location.

Classification It involves the execution of the mobile app specific classifier. Because of the complexity of human activities and the noise existing in sensor data, the classification algorithm are almost always probabilistic, although machine learning techniques have also been used [7]. The classifiers can make use of user's context to improve their performance.

This stage is the most power consuming one and the execution time may take from seconds up to hours, depending on the data being classified (e. g. in the case of human behavior recognition). Because of this, the classifier can be executed locally at the smartphone or in external elements like the cloud.

Post-processing The classification of data may trigger another continuous sensing-compute chain or other mobile app specific tasks such as displaying feedback to the user or initiating network communication.

2.4 Sensing scale of mobile sensing apps

Depending on the purpose of the mobile app, it may be required to obtain data from one or more users. The target audience size of the mobile app is known as the *sensing scale*. There are two sensing scales:

Individual sensing scale Involves collecting and analyzing data coming from a single person. Typically, data refer to track user's exercise routine and the information is not shared.

Community sensing scale Involves collecting and analyzing data coming from several users who share a common goal. It needs to respond to user specific needs of privacy and anonymity.

This sensing scale is often referred as mobile crowd-sensing and classic examples include traffic monitoring, detection of available parking slots, etc.

An implicit detail in this sensing scale is the need of a centralized node that acts as a sink of data where analysis processes are executed.

This research is conducted targeting the individual sensing scale since the policies to be generated consider analysis of data coming from a single person.

2.5 Paradigms of mobile sensing apps

The way how sensors are employed and whether explicit user input is required by a mobile app is known as the *mobile sensing paradigm* [9].

There are two sensing paradigms:

Opportunistic It aims automatic data collection without human participation at all. A remarkable issue of this paradigm is determining the best moment to perform a sensor reading.

Participatory It leverages the abilities of people requiring their participation to describe the data and to decide the best moment for their capture.

However, this human dependence can also be a source of errors and noisy data since user is able to upload mislabeled data or even to not to have participation at all.

This research is conducted by following the opportunistic paradigm since the smart policies to be generated will decide the duty cycle that sensors must observe, avoiding user participation.

2.6 Mobile sensing apps examples

Mobile sensing apps have evolved from apps with specific sensor usage, like using the accelerometer to change the orientation of the screen, to more complex tasks like detecting the user activity e. g. walking, running, climbing stairs, etc. The usage possibilities offered by mobile sensing apps are high and even increasing after a new sensor is embedded in a smartphone.

In [22] the *WalkSafe* mobile app is described. This app improves the safety of pedestrians that walk and talk. WalkSafe leverages the user's context employing the back camera of the smartphone to detect vehicles approaching the user, alerting her or him of a potential unsafe situation.

WalkSafe implements machine learning techniques for detecting the front and back views of moving vehicles. It employs image recognition algorithms trained off-line with datasets of positive (pictures of front and back views of vehicles) and negative (pictures of side views of vehicles and urban environments) samples producing a set of features for building the car detection model.

Once the model is created, it is uploaded to the phone for its usage in the on-line detection process. When a phone call is received, the smartphone takes pictures from the back camera, fixes their orientation with accelerometer data, and passed them to the detection module for their processing in real time. If a car approaching the user is detected, then a vibrating alert is emitted.

Another example of context information usage is *BeWell* [10], which is an app that can monitor and promote some aspects of physical and emotional well-being. BeWell continuously tracks user behavior along three health dimensions in an *opportunistic* sensing way.

Classifications algorithms are run directly on the phone to infer context information about user's sleep duration, physical activity, and social interaction from sensor data. BeWell assigns a score to these dimensions and gives a graphical feedback of them to the user. Sleep duration is inferred by detecting the absence of movement of phone at night. Physical activity is detected thanks to accelerometer inferring walking, stationary, or running states of the user. Social interaction is obtained thanks to microphone samples that are used to detect speech or silence time lapses.

A final instance of mobile sensing apps, that employs a *pluggable* sensor via Bluetooth, is *NeuroPhone* [4]. This app works employing neural signals obtained from an electroencephalography (EEG) headset to control smartphones.

Neurophone is a brain-controlled address book dialing app that leverages the generation of P300 [3] signals in human brain. The smartphone shows a sequence of pic-

[3] *P300* is a neuroscience term that refers to a positive peak with a latency of 300 *ms* that is elicited when

tures of address book contacts and a potential P300 brain is elicited when a photo matches the person whom the user wishes to call. The data generated by the EEG headset is transmitted to the smartphone, on which a classification process detects an actual P300 and triggers the contact's phone number dialing.

brain concentrates on a task specific stimulus among a pool of stimulus.

3 State of art

This section introduces a description and classification of works aiming to address the energy problem generated by employing sensors continuously in mobile sensing apps. A special focus is given to the pure software techniques, which will be pursued during the development of this research.

3.1 Approaches for addressing the energy issue

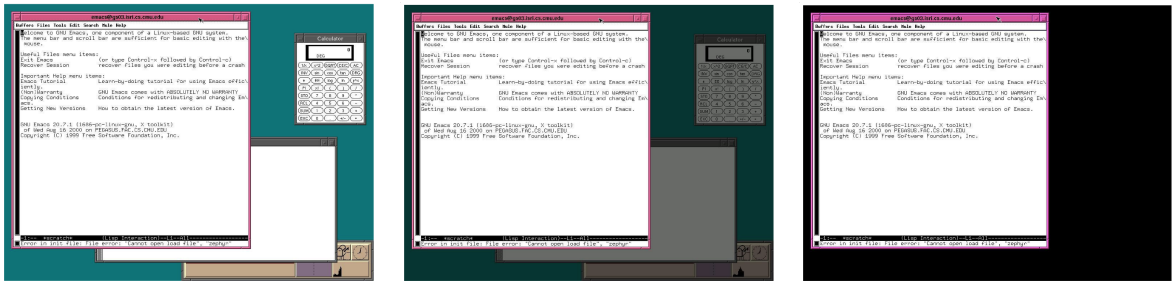
As stated, the advances produced in several technological fields have contributed to the acceptance of smart devices by society [9, 18]. However, although computing and storage capacity issues have been addressed and solved partially with every new generation of mobile devices, the energy consumption is still an open issue because of the slow rhythm of the technological advances related to it and because the inclusion of newer and more sophisticated sensors that impose a higher energy demand.

In this sense, the battery consumption concern has been under research since the introduction of mobile devices into the market, and also shows an evolution in the objectives pursued by scientists.

The research in energy management in mobile devices started with broad techniques and guidelines applied when building mobile application systems. In [13] the authors address the energy consumption issue from a design point of view by introducing the idea of the *requirements-aware energy scale-down* approach.

This approach states that both hardware and software should scale their features and energy usage to meet a variety of design points. According to it, if there are no hard energy constraints, the software can use the hardware components at maximum performance in order to accomplish the mobile app requirements. On the other hand, in case of energy constraints the mobile app must be aware and react accordingly by modifying via software the hardware usage for consuming less energy and still accomplish the application requirements. The described work was implemented targeting different hardware elements like screen, processor and wireless radio communication interfaces. For example, the screen implementation aims the design of energy aware GUI's. Such GUI's change the luminescence and color of non-active portions of the screen to reduce power consumption, as shown in the Figure 3.1.

The idea of *requirements-aware energy scale-down* approach serves as a base for the creation of specific techniques for addressing the energy consumption issue. However, the work introduced by authors focused on broad guidelines that, while helpful, left behind important details that can be obtained from contextual information of user. It is noteworthy that at the time of the coinage of this approach the smart devices proliferation was in its infancy.



(a) Original interface (b) Background half dim (c) Background fully dim

Figure 3.1: An example of energy aware GUI proposed by [13]

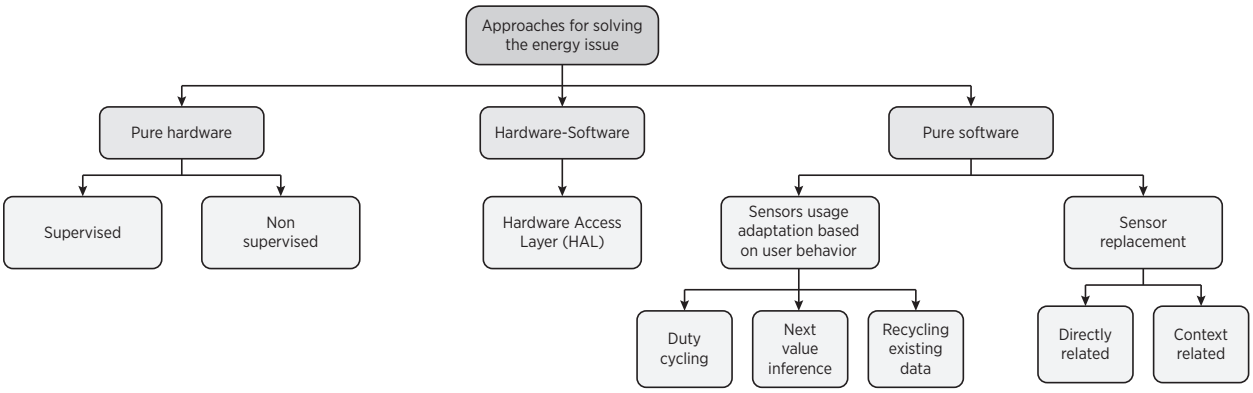


Figure 3.2: Taxonomy of approaches for solving the energy issue.

Currently, the popularity of smartphones and the mobile sensing apps have made the energy issue evident and specific mechanisms have been proposed to address it. A revision of the literature shows that there are three families of techniques to face the energy issue (Figure 3.2):

- Pure hardware.
- Hardware-software.
- Pure software.

The pure-hardware approach is located at hardware level and is agnostic of the entire software platform. The hardware-software approach is located at the lowest levels of the mobile OS in connection with hardware elements. It can be seen as the development of new hardware drivers. Finally, the pure-software approach is located at top of the mobile OS stack and it is agnostic of the hardware platform. Works of this approach make use of the API offered by the mobile OS for accessing sensors, collect data, analyze these data and readapt the usage of sensors.

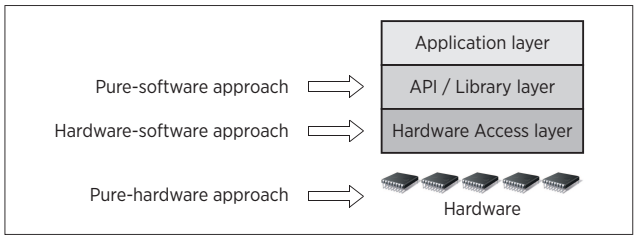


Figure 3.3: The relation between approaches for solving the energy issue and layers of a mobile platform.

As can be seen in Figure 3.3, the energy issue is a cross-layer problem that can be analyzed and addressed from several perspectives. It is noteworthy that although there is a taxonomy of approaches, the solutions found in literature may include a combination of these techniques. This is a suggestion of the complexity of the problem and even about the evolution of the techniques, since ideas proved to be working efficiently in software soon or later are implemented directly in hardware.

In next sections a brief description of the pure hardware and hardware-software approaches is presented, followed by a detailed review of the pure software approach since the special features and advantages offered by it.

3.1.1 Pure hardware approach

Despite the fact current mobile processors include features at the electronic level like DPM (Dynamic Power Management) and DVFS (Dynamic Voltage and Frequency Scaling) that are helpful to reduce the energy consumed by them, these techniques are not enough to solve the energy issue in mobile platforms. The major drawback with the usage of these features is the complexity of the mobile processors, whose static power consumption remains high (approximately 200mW) when the processor is not in sleep mode [16]. Additionally, the current architecture of most mobile platforms require that for the correct operation of the phone, other components have to be operational, which increases the energy consumption.

The set of works under the pure-hardware approach tries to perform a redesign in the hardware platform of mobile devices. This redesign implies the isolation of all of the components associated with the measurement of physical signals inside a new unit with a dedicated low power processor. Such change must not carry the modification of mobile apps logic.

This new hardware unit can obtain a substantial reduction in the energy consumption since it is energy aware by design. The basic behavior involves that the sensors report their readings to the embedded low power processor, which is capable of do-

ing light computing operations like filtering. While the hardware unit is working on reading and filtering data, the rest of the smartphone's hardware platform is able to reach its sleep mode.

The pure hardware approach can be materialized following two distinct implementations, based on the level of data complexity that the platform can handle. The supervised variant does not have knowledge about the information that is handled and it is restricted to receive sensor data requests and deliver data without running any sort of filtering or processing.

The non supervised variant is also able to deliver raw data, but additionally can detect higher level information thanks to the data processing algorithms. For example, it might measure the distance covered by user, the amount of steps done, or identify the type of activity performed by the user (no movement, walking, running, etc.) completely by itself.

An example of the supervised variant is LittleRock [16]. Its design includes a special unit in charge of the access to sensors. When any mobile app requires to perform sensing tasks it has the option to employ the LittleRock unit or use it only as a bridge and access directly to sensors. The relevant features of this design are:

Power independence LittleRock is powered directly from battery and not from other internal electronic items. Because of this, the main circuitry can be turned off when the phone is in sleep mode and keep LittleRock accessing to sensors.

Interrupt LittleRock offers interruption features, making it possible to wake up the phone and inform it about events discovered in data delivered by sensors.

Re-purposing LittleRock allows to reprogram its main processor to adapt the processing of sensor data and achieve mobile app requirements.

The experimentation conducted by authors indicates that LittleRock can operate up to 830 days whe this and the phone are in sleep mode using a 1340 mAH battery. Among the results, it is noteworthy that LittleRock is faster than the phone itself do collect a single sample of data from sensors due to the complexity of the software stacjk that the main processor has to handle.

Another relevant example of the supervised variant is reported in [7]. Despite the fact this work was not implemented in a smartphone unit, it also introduces a mobile sensing platform (MSP) that is composed in a similar way than LittleRock. This MSP is basically a sensor board attached to a wireless node that includes a 32 bits ARM7 processor, Bluetooth interface and a compact flash bay for storage. It can be noted that the MSP is representing the extra hardware unit with sensors and a dedicated low power processor, which can deliver the collected data via Bluetooth to an external computing unit for analysis and classification tasks.

The autonomous variant has been already implemented in the Apple iPhone. Starting from the iPhone 5s, these smart devices include an ARM coprocessor that is in charge of interacting with several sensors like accelerometer, gyroscope and magnetometer in order to obtain data about user's fitness [19].

Apple has introduced into iPhone OS a software API, the Core Motion Framework [1], that allows mobile developers to make use of the new hardware architecture and access and consume these information. The Core Motion Framework is able to deliver information refering to amount of steps or distance that user has covered and even the type of activity performed by him. Additionally, this API allows requests for raw data from sensors being the update frequency the only mandatory argument.

Despite the fact this implementation will lead to energy savings, it presents two major drawbacks. The first one is that the step counter mechanism is always running in the background even if no applications are requesting data. The second one is that this implementation does not provide support for continuous readings and lacks the idea of self adapting the sensor's duty cycle in order to reduce energy consumption.

A common problem of any hardware approach resides in that hardware designers can not foresee the needs of any future mobile development and implement it directly in the circuits. In this sense, the effort of hardware designers is affected by a trade off between the features provided out-of-the box and the flexibility features offered by the produced hardware platform.

3.1.2 Hardware-software approach

Another approach found in literature describes a combination of hardware and software techniques to optimize the energy consumption in continuous sensing mobile apps.

This technique aims to contribute with a reviewed and improved software API that allows to allocate computational and sensing tasks in components others than the predefined by the platform's original design.

This approach allows to instruct an additional low power processor in the smartphone to execute any arbitrary instruction. Since the smartphone's processor can be kept idle there is a potential energy saving.

The work presented in [18] leverages the presence of several low power processors (LP) in the latest smartphones. The utilization of these LP's can be done by exposing their functionality through a layered API.

Through several simulations, authors show that such processors can bring substantial energy savings, specially when executing frequent sampling & buffering and basic arithmetic operations.

That work describes two main challenges, the selection of a suitable LP and guidelines for deciding where to allocate the execution of a given task.

To select an adequate LP, the key factor is the wake up transition delay in terms of energy. So, any processor with a small wakeup transition delay is suitable as LP. For deciding where to perform the execution of a task, the next guidelines are proposed:

Execution in main processor Tasks found to be more efficient in the main processor should be executed there, ignoring the transition penalty.

Execution in LP Tasks found to be more efficient in the LP, and that are very frequent within the mobile app, should be executed in the LP. Sampling and buffering tasks belong to this category.

Execution in main or in LP Tasks found to be more efficient in the LP, and that are not frequent within the mobile app, should be executed in the LP. However, a special mechanism to evict them should be included. This might happen when tasks of other mobile apps request the LP.

3.2 The software approach

3.2.1 Sensor usage adaptation based on user behavior

3.2.2 Sensor replacement

References

[1] APPLE, *Core Motion Framework Reference*, Sep 2013.

[2] P. BELLAVISTA, A. CORRADI, M. FANELLI, AND L. FOSCHINI, *A Survey of Context Data Distribution for Mobile Ubiquitous Systems*, ACM Computing Surveys, 44 (2012), pp. 24:1–24:45.

[3] A. CAMPBELL AND T. CHOUDHURY, *From Smart to Cognitive Phones*, Pervasive Computing, IEEE, 11 (2012), pp. 7–11.

[4] A. CAMPBELL, T. CHOUDHURY, S. HU, H. LU, M. K. MUKERJEE, M. RABBI, AND R. D. RAIZADA, *NeuroPhone: Brain-mobile Phone Interface Using a Wireless EEG Headset*, in Proceedings of the Second ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds, MobiHeld '10, New York, NY, USA, 2010, ACM, pp. 3–8.

[5] A. CHARLESWORTH, *The ascent of smartphone*, Engineering Technology, 4 (2009), pp. 32–33.

[6] G. CHEN AND D. KOTZ, *A Survey of Context-Aware Mobile Computing Research*, tech. report, Dartmouth College, Hanover, NH, USA, 2000.

[7] T. CHOUDHURY, S. CONSOLVO, B. HARRISON, J. HIGHTOWER, A. LAMARCA, L. LEGRAND, A. RAHIMI, A. REA, G. BORDELLO, B. HEMINGWAY, P. KLASNJA, K. KOSCHER, J. LANDAY, J. LESTER, D. WYATT, AND D. HAEHNEL, *The Mobile Sensing Platform: An Embedded Activity Recognition System*, Pervasive Computing, IEEE, 7 (2008), pp. 32–41.

[8] M. B. KJÆRGAARD, *Location-based services on mobile phones: minimizing power consumption*, Pervasive Computing, IEEE, 11 (2012), pp. 67 –73.

[9] N. LANE, E. MILUZZO, H. LU, D. PEEBLES, T. CHOUDHURY, AND A. CAMPBELL, *A survey of mobile phone sensing*, Communications Magazine, IEEE, 48 (2010), pp. 140–150.

[10] N. LANE, M. MOHAMMOD, M. LIN, X. YANG, H. LU, S. ALI, A. DORYAB, E. BERKE, T. CHOUDHURY, AND A. CAMPBELL, *BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing*, in Intl. ICST Conf. on Pervasive Computing Technologies for Healthcare, IEEE, 4 2012.

[11] X. MA, Y. CUI, AND I. STOJMENOVIC, *Energy Efficiency on Location Based Applications in Mobile Cloud Computing: A Survey*, Procedia Computer Science, 10 (2012), pp. 577 – 584. {ANT} 2012 and MobiWIS 2012.

[12] Y. MAN AND E. C.-H. NGAI, *Energy-efficient automatic location-triggered applications on smartphones* , Computer Communications, 50 (2014), pp. 29 – 40. Green Networking.

[13] R. N. MAYO AND P. RANGANATHAN, *Energy Consumption in Mobile Devices: Why Future Systems Need Requirements-Aware Energy Scale-down*, in Proceedings of the Third International Conference on Power - Aware Computer Systems, PACS'03, Berlin, Heidelberg, 2004, Springer-Verlag, pp. 26–40.

[14] R. PEREZ-TORRES, *Middleware para el soporte de aplicaciones móviles sensibles a la ubicación y al contexto basado en cómputo en la nube.*, master's thesis, LTI Cinvestav Tamaulipas, November 2012.

[15] R. PEREZ-TORRES AND C. TORRES-HUITZIL, *A power-aware middleware for location amp; context aware mobile apps with cloud computing interaction*, in Information and Communication Technologies (WICT), 2012 World Congress on, IEEE, Oct 2012, pp. 691–696.

[16] B. PRIYANTHA, D. LYMBEROPOULOS, AND J. LIU, *Littlerock: Enabling energy-efficient continuous sensing on mobile phones*, Pervasive Computing, IEEE, 10 (2011), pp. 12–15.

[17] R. QURESHI, *Ericsson Mobility Report*. PDF, June 2014.

[18] M.-R. RA, B. PRIYANTHA, A. KANSAL, AND J. LIU, *Improving Energy Efficiency of Personal Sensing Applications with Heterogeneous Multi-processors*, in Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp '12, New York, NY, USA, 2012, ACM, pp. 1–10.

[19] S. SATHIAH, *A closer look: Apple iPhone 5s M7 Motion Coprocessor*, Oct 2013.

[20] A. SCHMIDT AND D. BIAL, *Phones and MP3 Players as the Core Component in Future Appliances*, Pervasive Computing, IEEE, 10 (2011), pp. 8–11.

[21] N. VALLINA-RODRIGUEZ AND J. CROWCROFT, *Energy management techniques in modern mobile handsets*, Communications Surveys Tutorials, IEEE, 15 (2013), pp. 179–198.

[22] T. WANG, G. CARDONE, A. CORRADI, L. TORRESANI, AND A. T. CAMPBELL, *WalkSafe: A Pedestrian Safety App for Mobile Phone Users Who Walk and Talk While Crossing Roads*, in Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, HotMobile '12, New York, NY, USA, 2012, ACM, pp. 5:1–5:6.

[23] O. YURUR, C. LIU, AND W. MORENO, *A survey of context-aware middleware designs for human activity recognition*, Communications Magazine, IEEE, 52 (2014), pp. 24–31.

[24] Z. ZHUANG, K.-H. KIM, AND J. P. SINGH, *Improving Energy Efficiency of Location Sensing on Smartphones*, in Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, New York, NY, USA, 2010, ACM, pp. 315–330.