Kaisa Korhonen

Predicting mobile device battery life

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.2.2011

Thesis supervisor and instructor:

Prof. Jukka Manner

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Kaisa Korhonen

Title: Predicting mobile device battery life

Date: 28.2.2011 Language: English Number of pages:6+56

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor and instructor: Prof. Jukka Manner

Mobile phones and other mobile devices have one resource in common that is essential to all of them: battery life. Once the battery is empty, the device is useless until it is recharged. Limited battery life has become an issue especially in the newest mobile devices. The evolution of mobile devices during the last ten years has changed them from mobile phones into multi-functional mobile computers. Today's mobile device can contain for example a web browser, a music player and a navigator. This evolution has made the devices increasingly useful and important in everyday situations, but at the same time the average power consumption of the devices has increased, leading to shorter and less predictable battery life.

This thesis examines how the remaining battery life could be estimated and indicated to the user in an intuitive way. The target platform of the work was the Linux-based mobile computer, Nokia N900. The work shows that estimating the remaining battery life on the target device in a reliable manner is feasible. The outcome of the work is a battery monitoring application, named BattInfo, which provides the user with remaining battery life estimations, such as call time and music playback time. BattInfo was also tested on a group of N900 users, and further improved based on the results from the testing.

Keywords: Mobile device, battery life, energy consumption, software implementation

AALTO-YLIOPISTO SÄHKÖTEKNIIKAN KORKEAKOULU	DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Kaisa Korhonen

Työn nimi: Mobiililaitteen akkukeston ennustaminen

Päivämäärä: 28.2.2011 Kieli: Englanti Sivumäärä:6+56

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka Koodi: S-38

Valvoja ja ohjaaja: Prof. Jukka Manner

Matkapuhelimet ja muut mobiililaitteet ovat kaikki riippuvaisia yhdestä kriittisestä tekijästä: akkukestosta. Kun akku tyhjenee, laite on hyödytön kunnes käyttäjää pystyy uudelleen lataamaan laitteen. Akkukesto on muodostunut ongelmaksi etenkin uusimmissa älypuhelimissa. Matkapuhelinten kehitys viimeisen vuosikymmenen aikana on muutanut ne yksinkertaisista kannettavista puhelimista monitoimilaitteiksi. Uusimmat älypuhelimet voivat sisältää esimerkiksi internet-selaimen, mediasoittimen ja navigaattorin. Tämä kehitys on tehnyt puhelimista entistä hyödyllisempiä jokapäiväisissä tilanteissa, mutta samalla laitteiden keskimääräinen energiankulutus on kasvanut, johtaen lyhyempään akkukestoon.

Tässä työssä tutkitaan miten akkukestoa pystyttäisiin ennustamaan ja esittämään käyttäjälle intuitiivisella tavalla. Työn kohdelaite oli Linux-pohjainen älypuhelin, Nokia N900. Työssä näytetään, että akkukestoa todellakin on mahdollista ennustaa luotettavalla tavalla. Työn tuloksena syntynyt akunmonitorointisovellus, BattInfo, näyttää käyttäjälle monipuolista tietoa jäljellä olevasta akkukestosta, esimerkiksi jäljellä olevan puheajan ja musiikinsoittoajan. Ryhmä N900-puhelimen käyttäjiä testasi BattInfoa, ja saatujen tuloksien perusteella sovellusta edelleen parannettiin.

Avainsanat: Mobiililaite, akunkesto, energiankulutus, sovelluksen toteutus

Preface

This thesis was done as a research project under the supervision of professor Jukka Manner. I would like to thank Jukka for his guidance and feedback, that greatly motivated me during the work.

I would also like to thank my friends Atte, Pekka and Juho for the splendid years at TKK, as well as Salvatore and my family for their support and encouragement during my studies and this work.

Otaniemi, 28.2.2011

Kaisa Korhonen

Contents

1 Introduction

Mobile phones and other mobile devices have one resource in common that is essential to all of them: battery life. Once the battery is empty, the device is useless until it is recharged. An ideal mobile device would never have to be recharged by the user: it would recharge itself wirelessly or reproduce the energy it needs from movement. While this ideal device remains to be invented, we must deal with the challenges set by limited battery life.

Limited battery life has become an issue especially in the most recent generation of mobile devices. The evolution of the devices during the last ten years has changed them from mobile phones into multi-functional mobile computers. Today's mobile device can contain for example a web browser, a music player, a navigator and a wealth of networked applications. This evolution has made the devices increasingly useful and important in everyday situations, but at the same time the average power consumption of the devices has increased, leading to shorter and less predictable battery life. It has become difficult for a user to estimate when the battery of the mobile device will run out and what can still be done with the remaining battery capacity.

One way to relieve this problem is to inform the user about the remaining battery capacity on the device. The current methods of showing battery-related information in most mobile devices, not only in phones but also in cameras, mp3 players and laptops, typically consist of 1) a bar that roughly shows the battery charge level and 2) an audio alarm that warns about low battery. These two methods have remained practically the same for the last ten years, even though the mobile devices themselves have developed tremendously and would allow for more advanced methods to be used. As an example of a more advanced method, modern cars tell the driver how many kilometers can approximately be driven with the remaining amount of fuel. Also laptops give the user an estimate about the remaining battery life.

The goal of this thesis is to examine how the remaining battery life in a mobile device could be estimated and to design and develop a battery monitoring application for a Linux based mobile computer, Nokia N900. The battery monitoring application should provide the user with more elaborate information about the battery charge level and remaining battery life, for example the remaining call time and remaining audio playback time. Also the possibility of estimating charging time is studied.

The work shows that estimating the battery life on the target device is possible and relatively straightforward: it can be calculated by dividing the battery charge level reported by the device by the current demand of the device. Estimating charging time revealed to be more challenging, but also possible. Based on this information, a battery monitoring application, named BattInfo, was developed. BattInfo monitors the battery status and, based on the average current demand of different tasks measured offline, calculates battery life estimations for the tasks, such as the remaining standby time, call time, web browsing time and video playback time. BattInfo presents all the information via a graphical user interface. A user test was conducted on BattInfo, which showed that BattInfo is indeed a useful, trustworthly application, that helps the users to manage their batteries. The results of this work are applicable not only for mobile and smart phones, but for practically any multi-purpose device that runs on battery power, such as a laptop or a tablet computer.

The remainder of this work is organized as follows. Chapter 2 describes the evolution of mobile devices and discusses how the evolution has affected the battery life. Chapter 3 presents various research papers that aim at reducing the energy consumption of mobile devices. Chapter 4 studies the possibilities for predicting battery life and charging time on the target device. It also presents the measurements that were conducted in order to find out the power consumption of different tasks (e.g. calling, web browsing) on the N900. Based on the findings in Chapter 4, a battery monitoring application, named BattInfo, is designed and implemented. BattInfo is described in detail in Chapter 5. Finally, a user test conducted on BattInfo is presented after which the application is evaluated and conclusions are made.

2 Evolution of mobile devices

According to a user study conducted in 2005 by TNS, the world's largest custom market research specialist, the most wanted feature in the future mobile device is "two days of battery life during active use" [1]. Short battery life was, and still is, an issue in the mobile devices. This chapter describes the evolution of mobile devices during the last 10-15 years, explaining the reasons that have lead to this situation and the possible remedies. An overview of the currently available battery monitoring applications for mobile devices is given in the end of the chapter.

2.1 Increasing power consumption

The use cases of mobile devices have changed a lot during the past years, as illustrated in Figure 1 [2]. The uses cases of the first mobile phones were simple: calling and possibly text messaging. Since most people only make a few phones calls per day and also because calling and text messaging were more expensive than today, the phones were in standby state most of the time.

As the phones then started to develop, new features such as a calendar, mp3 playback, and FM radio were embedded into the devices. As there were more use cases, the phones were not left in standby state for long times, instead they were used throughout the day for various purposes. The mobile phones became smart phones, and as they were used more than before, the power consumption of the phones increased. The newest generation of mobile devices is characterized by an increasing amount of connected, always-online applications. For example social networking and localization services are increasingly popular [2]. The increased usage of bandwidth and running many applications simultaneously in the background have increased the energy demand of the devices even more.

Also GSM Arena provides interesting figures about mobile device features in their article "Mobile phone evolution: features in focus" [3]. The article depicts the penetration of different interfaces and features in mobile phones during the last decade. Figure 2 for example shows the yearly proportion of mobile phones equipped with a Wireless Local Area Network (WLAN) interface. The green bars in the figure indicate the absolute average, in which each phone model has the same statistical importance (same weight). The orange bars indicate the weighted average, which takes into account the fact that different phone models are of different importance, based on the amount of times that the model was looked up from the GSM Arena's database. Thus, when calculating the weighted average a popular phone model has a higher weight than an unpopular one does. During the last ten years, the proportion of mobile phones equipped with a WLAN interface has grown almost 40% measured by the absolute average and the growth is even bigger if measured by the weighted

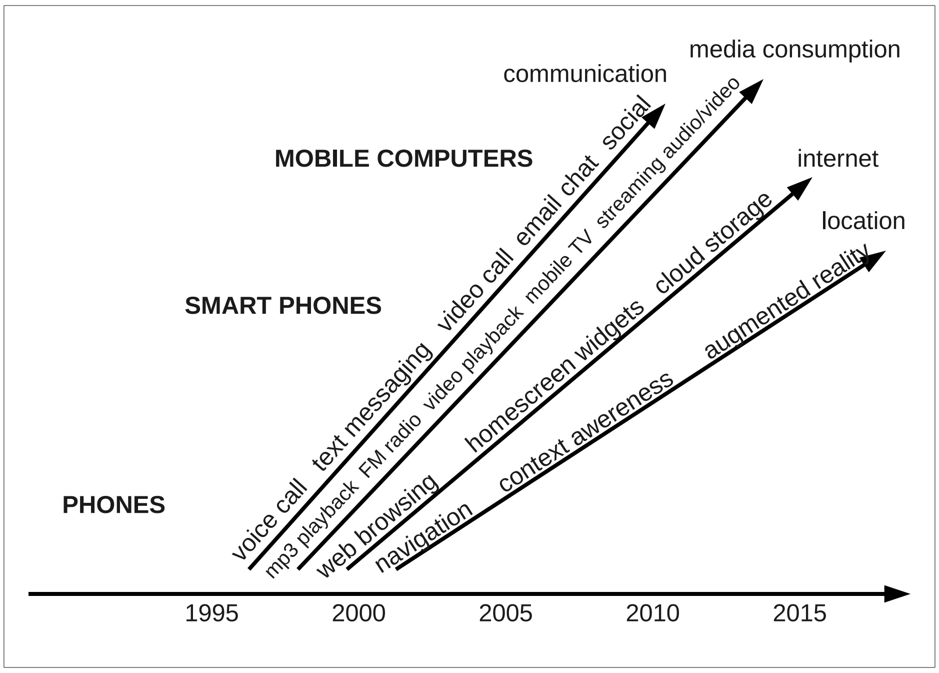


Figure 1: The evolution of the use cases of mobile devices [2].

average.

The article provides the same figures also for Bluetooth, FM radio and Global Positioning System (GPS) interfaces. Each of them has steadily grown its penetration in mobile phones during the last ten-year period. Thus, a mobile device today can have five radio interfaces (cell radio, WLAN, bluetooth, FM and GPS interfaces), whereas ten years ago the phones only had one or two. Also the penetration of cameras embedded in mobile phones has increased almost up to 100%. Another feature that has a big effect on the phones' power consumption is the screen size. It is known that the screens account for significant energy demand in the modern phones [4]. Figure 3 shows the average size of the mobile phone display since 2000. The average display size has grown more than 60% if measured by the absolute average, and almost 100% if measured by the weighted average, furhter increasing the overall energy consumption of the mobile devices.

Thus, the average power consumption of the mobile devices has grown significantly. The battery technologies have also developed, but not nearly as fast as the devices. Figure 4 illustrates the gap between the amount of energy needed by a Nokia mobile device for two days' active usage, and the amount of energy that can be provided by a standard-sized battery, according to Nokia's internal analysis [2]. The red curve in the figure shows how the energy demand of the devices has grown and is estimated to grow in the coming years. The energy available from a standard size battery,

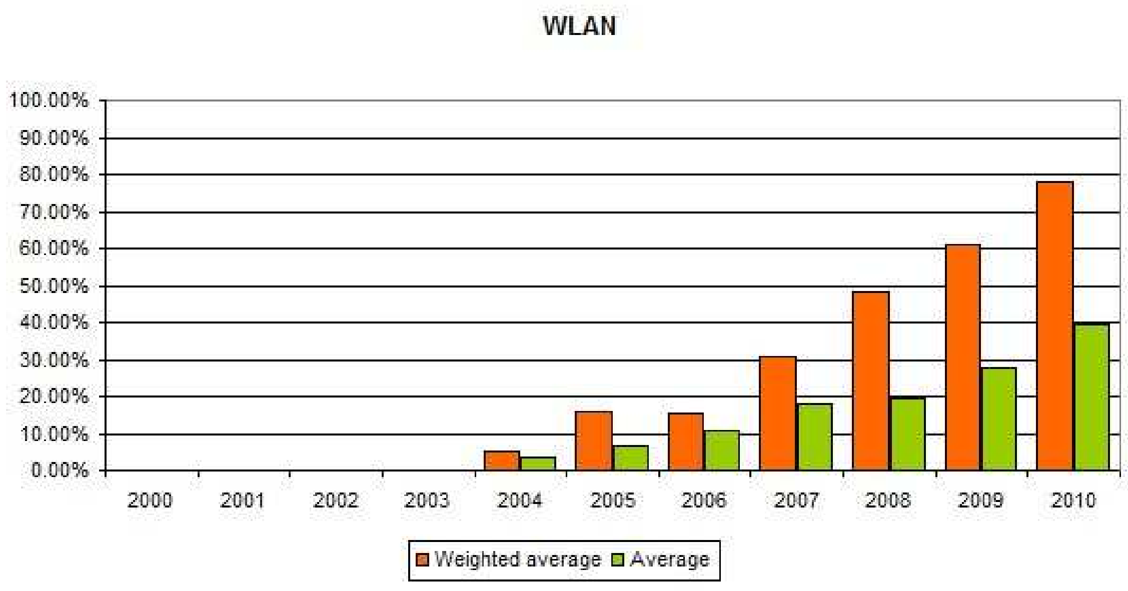


Figure 2: WLAN interface penetration in mobile phones [3].

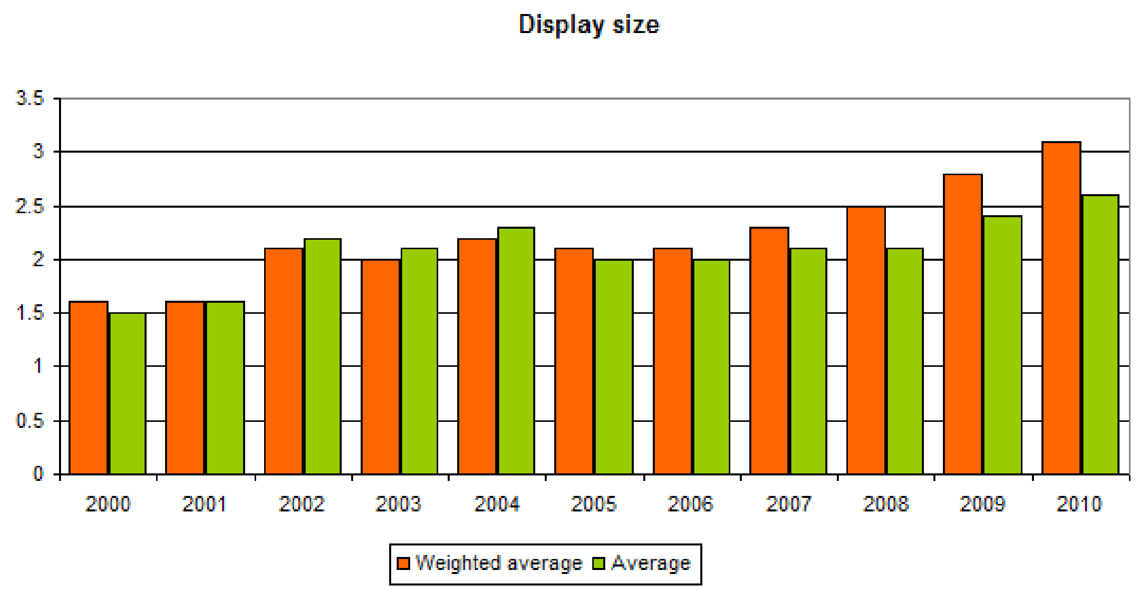


Figure 3: Mobile phone display size [3].

depicted in the figure with the blue curve, is clearly not sufficient.

Because of the big gap between the energy needed by the mobile device and the energy available from the battery, the average battery life of the mobile devices in active use is less than two days. In addition to shorter battery life on average, the battery life has become less predictable. An older mobile phone could only be used for calling and text messaging. Thus, calling was the most energy-demanding task

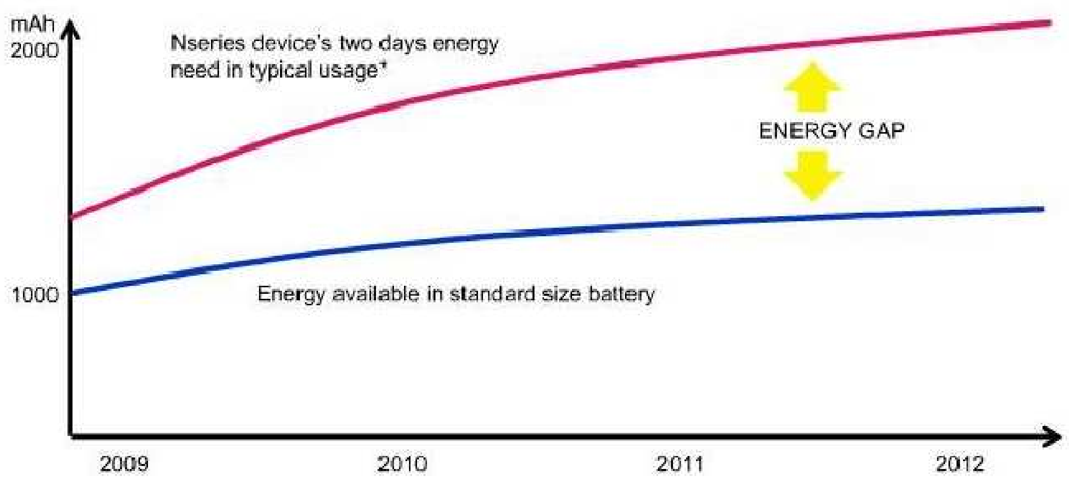


Figure 4: Mobile phone energy need vs. energy availability[2].

that could be done with the phone. It was relatively easy for the user to predict how fast the battery would drain during the day. In today's mobile phones the maximum power consumption has grown, which means that there is a greater variation in the power consumption of the phone. In standby state without an internet connection the battery can last for several days. If the user instead has a GPS-based map application running in the background while browsing the web and listening to music, the battery can be completely drained in a couple of hours. The battery life is thus more unpredictable in the new mobile phones.

2.2 Methods for improving battery life

The two basic ways to improve battery life are: maximizing the energy available to the mobile device and minimizing the energy needed by the device. Maximizing the energy available to the device can be done by improving the battery capacity (without increasing the battery size). In the past years the battery capacity has improved by 5-10% annually [2], which has not been enough to compensate for the increased energy demand of the mobile devices. Charging speed and convenience are other factors that affect the amount of energy available to the mobile device. If charging was fast and possible nearly everywhere (in offices, cafeterias, buses, bus stops), the short battery life would not be such a serious problem, since the device could be charged almost anywhere and anytime. Also the utilization of new energy sources, for example solar energy, energy from movement and wireless energy, would make the charging more convenient, and further improve the situation.

The second approach is to minimize the overall energy consumption of the device. There are many ways to achieve this. The ones mentioned in [2] are: improving hardware efficiency, new power management techniques, optimizing cellular networks and improvements in the user interface (software design). According to [2], software design is the potential means for reducing the power consumption, but so far, no significant improvements have been introduced and the battery life has remained short.

2.3 Need for battery life prediction

The approach taken in this work is to study how the battery life could be predicted and presented to the user. Instead of trying to directly address the problem of short battery life, the work concentrates on improving user satisfaction. While battery life remains short, it is better for the user to at least know about the remaining battery life than to be left with the short, unpredictable battery life. As a matter of fact, regardless of whether the battery life is long or short, the user satisfaction would most likely improve if the remaining battery life of a mobile phone, mp3 player or a camera was clearly visible to the user.

Another reason why battery life prediction is needed is that the users of the mobile devices rely more on their devices today than ten years ago. The mobile phone has become an all-in-one device, having applications such as a phone/address book, calendar, email, mp3 and video player, camera, web client and navigator. This evolution has made the devices increasingly important in everyday situations. The user might rely on finding her way to her destination using the navigator of the mobile device or on looking up the place of a business meeting in the middle of the day from the calendar of the mobile device. It is also more and more common that the mobile device is the only phone that a person owns [5], implying that it should be available in case of emergency. An unexpectedly empty battery would be a disturbing, or even a harmful distraction in all the above-mentioned cases. As a result, there clearly is a need to inform the user about the battery life left in the mobile device.

Also a user survey, conducted in parallel to this work by researchers from Helsinki Institute for Information Technology, revealed the users' need for more eloborate battery information. The survey consisted of both street questionnaires and of diaries kept by some users and it aimed to study the behaviour of mobile device users regarding the battery and the battery life. The results revealed that users do worry about running out of battery and that they have different methods for trying to cope with a nearly empty battery, such as avoiding usage of the device, sending text messages instead of calling and turning the phone silent. The conclusion that can be drawn from the survey is that the users are not sure how long their battery will last and what should be done in order to save battery. This again indicates a need for better battery monitoring applications.

2.4 Battery monitoring applications

Even though the processing power and storage capabilities of mobile devices have been growing exponentially, following Moore's law, the battery management techniques are much left behind [6]. As mentioned in the introduction, the default battery management techniques of mobile devices are the battery charge level meter and an audio alarm for low battery, and they have remained practically unchanged for years. The battery charge meter typically has only coarse accuracy and it does not give the user any idea about the remaining battery life. The battery-low audio alarms are typically played only when the battery is close to empty, when it might already be too late for the user to find a charger. Furthermore, the user does not know how long the battery will last and what can still be done with the remaining capacity after the battery-low alarm has been played.

More advanced battery monitoring applications do exist for some mobile phones, but they generally need to be separately installed by the user from an application store. myBatteryLife is a battery monitoring tool for iPhone. When the application is launched, it shows the accurate battery charge level in percentage and the remaining battery life for five different tasks (calling, web browsing using 3G and WLAN, audio playback and video playback) in a full-screen window.

Battery Monitor is a similar application for Nokia's Symbian devices, released in the Nokia Ovi Store in the end of 2010. It shows estimates for the remaining active, talk, web browsing and music playback time. The remaining active time is calculated based on the online-measured power consumption history of the device and the other values are based on offline-measured power consumption of the task. Also a charging time estimate and other statistics on battery usage are shown. The user can also place a Battery Monitor widget on the desktop, showing the estimated time for one of the tasks. This application is similar to ours, yet our application has more features and was in user studies by the time the Nokia's application was released.

When our work started in the spring 2010, a similar application to myBatteryLife and Battery Monitor could not be found for the Nokia N900 mobile computer, which is the target device of this work. There are at least two applications, BatteryGraph and BatteryEye, which monitor and log the battery information on the device and then show the power consumption history as graphs, but neither of them provides any estimations on the remaining battery life. Both of the applications are clearly targeted to the more technical users. The goal of this work is therefore to implement a battery monitoring application for the N900, that is able to estimate the remaining battery life and present it to the user in an appealing, easily understandable manner.

2.5 Summary

This chapter studied the evolution of mobile phones from the perspective of power consumption and battery life. As the mobile devices have changed from simple phones to multi-purpose mobile computers, they now have many more use cases than before and the power consumption has increased. The mobile devices have become more important and useful in the everyday situations, but at the same time the battery life has become shorter and less predictable.

The methods for informing the user about the state of the battery have remained the same for years and are clearly not sufficient. While the mobile devices themselves have developed quickly, and have now big colour displays that make it possible to create visually appealing graphical user interfaces, the battery charge meter has not developed during the past years. Also a survey that examined the users' behaviour regarding the battery showed that the users cannot know when their battery will run out and that they are troubled by it. Hence, there is a substantial need for better battery monitoring techniques and applications.

3 Reducing the power consumption of mobile devices

In order to keep the battery life of mobile devices long enough while more and more interfaces and features have been embedded into them and the displays have grown, methods for reducing the power consumption of mobile devices have been actively researched. As stated in the previous chapter, the improvements in batteries have not been enough to satisfy the increasing energy demand of the devices, but one can argue that also heat is an issue: even if the batteries could provide more energy, the power consumption of the mobile devices must be decreased because high energy consumption results in battery heating. This can be experienced for example with the target device of this work, the N900: during a 30-minute Skype call the device warms up so much that talking on it becomes uncomfortable. Therefore, even if better, more energy-dense batteries were available, they cannot alone be the solution for the problem. This chapter presents numerous research in the area of reducing mobile device power consumption.

3.1 Wireless communications

In a modern mobile device, wireless interfaces and wireless data transfer account for a significant amount of the total energy consumption of the device. As mobile packet data is becoming more common [7] and more always-online applications, e.g. social media and location services, are running on the devices, the amount of energy consumed by wireless communications is further growing. The typical set of wireless interfaces in a mobile device are the WLAN, Bluetooth and the cellular interfaces. There are different approaches to minimizing the energy consumed by wireless transfer. One can for example try to minimize the energy consumed by an individual interface, optimize over several interfaces (by choosing the least energy-consuming interface for transfer), minimize the amount of time that the interfaces are used or minimize the amount of data transferred. All these approaches and a few others are covered in this section.

3.1.1 Reducing WLAN energy consumption

Chapter 2 showed how the penetration of the WLAN interface in mobile devices has grown during the last years. WLAN is characterized by higher transmission speed (compared to a 3G connection) but also by high energy consumption. Consequently, reducing the WLAN energy consumption is very important considering the mobile devices' battery life. The research in this area is numerous. Some of the recent work is pointed out here, leaving the more thorough study for the interested reader.

The energy efficiency of MAC protocols for WLANs is studied in [8]. The paper integrates the various issues and challenges of MAC protocols for WLANs that have previously been researched individually, and classifies the MAC protocols based on their energy efficiency. The power management of the IEEE 802.11 based infrastructure mode WLAN is investigated in [9] and the optimal power management parameters are sought. While most of the research about the parameter configuration is based on simulation, the authors propose two queuing models (the M/G/1 and D/G/1 queues) to analyze the power management scheme by theoretical means. They find that the analytical models are strongly supported by the simulation results. The paper identifies the optimal parameters for energy efficiency and discusses them to achieve maximum energy efficiency without degrading the overall system performance.

The performance of the Power Save Mode (PSM) of the IEEE 802.11 infrastructure WLAN is analyzed in [10]. Using a performance analysis, the authors find the maximal listen interval, which maximizes the average percentage of time that a station stays in Doze (i.e sleep) state (and thus, consumes less energy) while satisfying the required quality of service on the mean and the variance of packet delay. The authors of [11] propose a centralized power saving mode (C-PSM), in which the 802.11 access point selects the optimal PSM parameters, optimizing the total energy efficiency for all clients. Using extensive simulation experiments the authors show that the C-PSM outperforms the standard PSM of the 802.11 WLAN, and reduces the power consumption by as much as 76%. Finally, a priority based power saving mode (PBPSM) is proposed in [12]. The PBPSM assigns different channel access priorities to different stations so that the energy efficiency of the stations that are in the power-save mode increases.

3.1.2 Optimizing over multiple network connections

As already mentioned, today's mobile devices are typically equipped with several radio interfaces, such as the cell radio, WLAN and Bluetooth interfaces. Each of these interfaces have their own characteristics (e.g. speed and range) which naturally affect the power consumed by them. One approach for saving energy in the mobile devices with multiple radio interfaces is to always choose the energy-wise cheapest interface.

A paper by Perrucci et al. [13] presents the results of measurements that were conducted in order to compare the energy consumption of text messaging, voice and data for a mobile device in 2G and 3G networks. The device used for the measurements was a Nokia N95 smartphone running the Symbian OS as operating system. The results of the measurements show that text messaging and voice services consume less energy in the 2G than in the 3G network. Instead, when large volumes of data are downloaded, 3G connection consumes less energy. The results imply that, in order to achieve longer battery life, a user should stay connected to the 2G network by default, and only use the 3G network when data transmission in needed.

Pering et al. present in their paper [14] CoolSpots, which is a system that enables a wireless mobile device to automatically switch between the WLAN and the Bluetooth interface when transmitting/receiving data in order to save energy. The work is based on the fact that while WLAN is a lot faster than Bluetooth, it also consumes a lot more power. While the idle power consumption of WLAN is of the order of 890 mW, Bluetooth's idle power consumption is only 120 mW. CoolSpot explores the policies to enable switching among the two radio interfaces based on the application bandwidth requirements, power and distance. Using experimental validation of the CoolSpot system the authors are able to reduce the energy consumption of the wireless subsystem by more than 50%.

The selection of an energy-wise optimal interface in a mobile device is studied also in [15]. The paper states that seamless handover and resource management are essential for achieving good Quality of Service (QoS) in mobile devices and that accessing e.g. UMTS and WLAN interfaces simultaneously can reduce the handover latency and data loss in heterogeneous handover. But accessing multiple interfaces simultaneously increases the mobile device energy consumption significantly. The paper thus presents an efficient interface selection mechanism that takes into account the battery power consumption in addition to the QoS.

A measurement study by Balasubramanian et al. [16] studies and characterizes the energy consumption of network activity over GSM, 3G and WLAN, with the goal of reducing the energy consumption of mobile applications that use these network technologies. It is found that in both GSM and 3G, a large fraction of the energy is wasted in the high-power states after the completion of a typical transfer. This energy is referred to as the tail energy. The tail energy of GSM is smaller than that of 3G, but, as GSM has a lower data rate, more time is spent in the high-power state while transmitting. It is also found that in WLAN the association overhead (finding an access point and connecting with it), is comparable to the tail energy of 3G, but that the data transfer is more efficient. Based on the above findings, the authors develop the "TailEnder", a protocol for scheduling the data transfers of common mobile applications so that their energy consumption is reduced. TailEnder can be used for two classes of applications: delay-tolerant applications (e.g. e-mail and RSS feeds) and applications that can benefit from prefetching (e.g. web browsing). The performance of TailEnder is evaluated on three applications (email, news feeds and web search), and for each of them, TailEnder is shown to reduce the energy consumption significantly.

3.1.3 Use of parallel TCP connections

The effect of using parallel TCP connections to the energy consumption of a mobile device is studied by Nurminen in [17]. Nurminen states that "as mobile phones are increasingly used for web browsing, emailing, multimedia and other data communication activities, the management of the connections in an energy-efficient way is important". The idea of the paper is to study whether using parallel TCP connections, which is shown to improve throughput, will also improve the energy-efficiency. TCP downloads during 3G voice calls or VoIP calls or when some other TCP streams are active at the same interface are given as examples in which parallel TCP connections can save energy. The paper shows that if the applications were engineered to take advantage of the phenomenon, significant energy saving could be achieved.

3.1.4 Localization

The newest mobile devices are often equipped with a GPS interface that can be used for localization and navigation purposes. The problem of continuous GPS usage in mobile devices is its high power consumption, which drains the battery very quickly. This is why the methods for lower-power localization have also been researched.

In [18], Kjaergaard et al. build a system named EnTracked, which schedules location updates (location data retrieved with GPS and sent using 3G to an internet-connected server) to both minimize energy consumption and optimize robustness. The paper states that using large intervals between location updates reduces energy consumption, but it also reduces the accuracy, since a pedestrian or a cycling target can move quite far in a few minutes. EnTracked thus uses estimation and prediction of system conditions and mobility in order to schedule the location updates, and it is reported to achieve significant energy savings.

Another adaptive location service for mobile devices, called "a-Loc" is presented in [19]. a-Loc reduces the energy consumed by localization based on two facts. Firstly, the location accuracy required by applications varies with location and secondly, in addition to GPS, mobile devices have other methods for sensing location, such as WLAN triangulation, cell-tower triangulation and Bluetooth vicinity. The authors use "searching for the nearest pizzeria" as an example. If the user is currently in a densely populated area, high location accuracy is needed in order to localize the nearest pizzeria. If the user instead is in a forest, 10 kilometers outside a city, lower location accuracy is sufficient and a lower-power method, such as cell-tower triangulation, can be used. The authors show that the a-Loc system can be implemented on a real mobile phone (Android G1 and AT&T Tilt phones) and that the energy required by localization is decreased if the system is used.

3.1.5 Proxy-assisted energy saving

A proxy server is a server that assists the client (a mobile device) in communicating with content servers. The proxy works as an intermediary between the client and the server forwarding, caching, retransmitting and even re-scheduling the packets sent between the client and the server. Commonly proxies have been used for web caching in order to provide content distribution and faster content download, and as components of firewalls [20]. They have also been used for transcoding content to make it better suitable for the target device. Recently it has been recognized that proxies can also be used for reducing the power consumption of mobile devices.

Two proxy servers for reducing the WLAN power consumption are presented in [20]. The baseline behind the work is that WLAN, which has an active and a sleep mode, should spend more time in the sleep mode and make fewer transitions between the two modes in order to reduce its energy consumption. The work concentrates on web browsing and media streaming, which both have high power consumption. The idea is to shape the traffic destined to the mobile device such that the packets are delivered in bursts, extending the idle intervals of the mobile device's WLAN interface. The media streams are also transformed into less energy- and bandwidth-consuming forms. The authors present a Power-Aware Streaming Proxy (PASP) and compare it to a Power-Aware Web Proxy (PABP). Both proxies are shown to reduce the WLAN energy consumption while streaming/web browsing.

Another approach for proxy-assisted energy saving in mobile (and other) devices is given in [21]. The paper states that a lot of energy consumed by network-connected hosts is wasted because without the network connectivity, the hosts could stay asleep the majority of the time. Having to maintain the network connectivity disables the power-saving features of many hosts. Two solutions are presented for this problem: 1) redesigning network protocols and applications and 2) using a Network Connectivity Proxy (NCP) that maintains the network presence on behalf of the host, allowing the host to stay in sleep mode.

3.1.6 Data compression

The transmission of a single bit over a wireless interface can consume more than 1000 times of energy compared to a single 32-bit computation [22]. Based on this fact, compression of data that is about to be transmitted seems like a potential means for reducing the energy consumption of mobile devices. Three papers, [22]-[24], all make contributions on this subject.

In [22], Barr et. al state that in order to really save energy, the data compression must be done with care. Many data compression algorithms require, in addition to the computation, an extensive amount of memory accesses, which can greatly increase the overall energy consumption of the compression. Thus, one might end up increasing the energy consumption instead of descreasing it if data compression is used carelessly. The paper studies and analyzes several lossless compression algorithms and tries to find out the situations in which compression is beneficial. Based on their findings, the authors develop an energy-aware data compression strategy that uses data compression prudently and reduces the overall energy consumption. The authors also note that decreasing the amount of transmitted data can also provide second-order benefits, such as reduction in packet loss and less contention in the network.

The subject of data compression in order to save energy is further studied in [23]. It is noted that if the transmission rate of the wireless medium is high, the benefit of the compression from the point of view of saving energy is reduced or even eliminated. The paper studies the benefits achieved from data compression with respect to the speed and signal strength of the wireless network to be used. When the signal strength is low, the data transmission would take longer and thus consume more energy, which makes data compression highly beneficial. The signal strength might however improve during the transmission, which is why an adaptive compression scheme is needed. The paper shows by experimental means that the overall energy consumption can be remarkably decreased using such an adaptive compressing scheme.

The use of data compression and its impact on energy consumption is also studied in [24]. The authors of the paper state that the compression ratios achieved by the compression algorithms depend highly on the type of the data that is compressed. For example, image compression and text compression are likely to result in very different compression ratios. Nine compression programs are examined in the paper and they are evaluated based on the compression time, compression ratio and the energy consumed. Using the Nokia N810 for experiments, the authors show that a carefully chosen compression scheme can allow energy savings up to 57% (uplink) and and 50% (downlink) when visiting popular web sites using a WLAN connection.

3.2 Displays

The displays of the modern mobile devices have become larger and brighter, enabling high user-satisfaction for video playback, web browsing, navigating, gaming and various other functions. But the growing size and resolution have also increased the display power consumption so that the displays nowadays consume a significant amount of energy in the mobile devices.

In the paper "A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems" [25], the authors also state that the power consumption of an LCD display system is significant on mobile devices. They show that the display frame buffer and buses are major energy consumers and that the energy consumption of these components is proportional to the number of frame buffer accesses during the display sweep operation. Thus, reducing the number of frame buffer accesses reduces the power consumption of the display.

The authors design and implement an incremental, adaptive compression algorithm based on run-length encoding (RLE). The algorithm is tested on five different kinds of applications (a warehouse manager, an e-book reader, an image viewer, a map viewer and a movie player) in order to find out the average compression gain and compression overhead. The power reduction achieved by the frame buffer compression algorithm is significant: the power consumption of the target components (the frame buffer and memory and its associated buses) is improved by about 50% to 66%. The overall power consumption in the prototype implemented by the authors is reduced by 10% to 15% in most of the test cases.

In [26], Iyer et al. have a more user-centric approach on reducing the display power consumption. The basis of the work is a major user study that concentrates on understanding the screen usage patterns of real users in order to find opportunities for display power reduction. The user study reveals that, on average, the users only used 60% of the screen area available, thus identifying a promising opportunity for reducing the display energy consumption. The user study further shows that in many cases, the information could be displayed on much simpler, lower-power displays without loosing visual quality. Based on this information, the authors develop the "Dark Windows" windowing environment, which changes the brightness and color of areas of the screen that are not of primary interest to the user in order to save energy.

The results obtained from testing the "Dark Windows" indicate that, with acceptable trade-offs in the brightness and colour of the user interface, significant energy savings can be reached. The user study was conducted on laptop and desktop computers, which is why the results obtained are not directly applicable for hand-held mobile devices with smaller displays, as Luo, one of the authors of [26] states in her related PhD dissertation [27]. However, Luo points out subsequent user studies by HP researchers, which show the acceptance of the Dark Windows design also on hand-held mobile devices.

3.3 Computation offloading

The basic idea behind computation offloading is the same as behind proxy-assisted communication that was described earlier: moving workload from the mobile device to a more powerful entity in order to save energy in the mobile device. Whereas in proxy-assisted communication the task of e.g. performing reliable communication is moved from the mobile device to the proxy server, in computation offloading a computationally heavy operation is moved from the mobile device to be computed in a more powerful server.

The original idea of computation offloading was to increase performance by moving computationally heavy operations from thin clients to rich servers. Only recently it has been discovered that offloading computation using the available communication channels can also help to reduce the energy consumption of the mobile devices [28]. When large amounts of computation is needed that could be offloaded with a relatively small amount of communication, offloading becomes energy-wise beneficial.

In [29] the idea of computation offloading for smartphones is elaborated and a practical system design is presented. The offloading system is evaluated on two smartphone applications: the first one is a multimedia content analysis application that performs object recognition of images and the second one is a distributed game that also involves images and object recognition. Both of the applications require relatively heavy computing, and using computation offload, the authors are able to reduce the battery consumption (and speed up the computation) significantly. Computation offloading is thus a promising method for further decreasing the mobile device power consumption.

3.4 Summary

This chapter presented the numerous research that has been done to decrease the energy consumption of mobile devices. First, the various methods for reducing the energy consumption of wireless communications were introduced. Methods for reducing the energy consumption of WLAN interface, which is commonly embedded in the mobile devices, were studied, as well as optimizing over multiple wireless interfaces in order to always choose the energy-wise cheapest interface for transmission/reception. The research on reducing the energy consumption of GPS and localization has also been studied and two systems, EnTracked and a-Loc, have been presented as possible energy-saving solutions. Also the research on using proxies and data compression for energy saving were presented.

The mobile device displays consume a significant amount of energy and research in this area was presented. Based on the research, there is a great potential in saving energy used by the displays. Finally, computation offloading was described and two research papers studying its affect on energy consumption, were viewed.

4 Predicting battery life and charging time

The possibilities of predicting battery life on the target device, the Nokia N900 mobile computer, are studied in this chapter. First, the information that the N900 can provide about the battery is examined in order to know how an application running on it could monitor the battery status. Tests are run to produce battery dissipation curves, that give more information about the battery discharge behaviour and can be used for the actual battery life prediction. The power consumption of several tasks on the N900 is measured. Also the need for estimating charging time is discussed and the possibilities of estimating it on the N900 are studied.

4.1 Predicting battery life on the N900

Maemo, the software platform of the N900 provides battery identification, monitoring and charging service through the battery management entity (BME) [30]. The system command "hal-device bme" shows all the information that the BME provides about the battery. An example output of this command is shown in Appendix A. Using signals delivered by D-Bus (described in Chapter 5), a program running on Maemo can read and monitor several pieces of information regarding the battery, including the battery voltage, the battery charge level, information about whether a charger is plugged in and the type of the charger.

In [31], Wen et al. study how to predict battery life using battery dissipation curves obtained offline together with power dissipation history observed online. The baseline behind their idea is that the battery dissipation curve, which is a function of the battery voltage with respect to time, has the same shape for constant but differing workloads. Figure 5 shows the battery dissipation curves for two constant workloads. The curves start from the open circuit voltage (full battery), and finish at the cut-off voltage (empty battery). As can be seen, the curves have the same shape, but the workload F2 drains the battery faster. By transforming these offline-measured voltage curves into a linear form, the authors are able to accurately predict the battery life implied by any constant workload using only a partial voltage curve of that workload obtained online. Using this method the authors are able to predict the battery life with an error that is significantly smaller than the error produced by the Advanced Power Management (APM), the standard power management tool in Linux systems.

Based on [31], the voltage reported by the N900 seemed like the best candidate to be used for battery life prediction. A program was written to log all the changes in the battery voltage (battery.voltage.current in Appendix A) together with the times at which the changes occurred. The following test was then run:

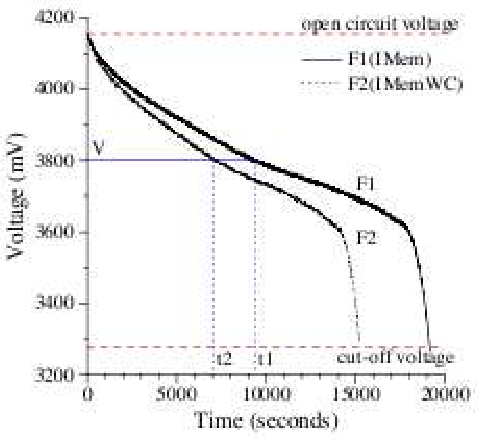


Figure 5: The dissipation curve of a Li-ion battery for two constant workloads [31].

1. charge the battery until it is full
2. start the logger program
3. start a constant-load task on the device, e.g. standby, video playback or audio playback
4. run the task until the battery is empty.

By plotting the voltages recorded by the logger program as a function of the times at which the voltage changes were received, battery dissipation curves similar to the ones in [31] could be produced. Figure 6 shows an example of a battery dissipation curve for audio playback on the N900. The test was performed also for standby and video playback, and the shape of the dissipation curve was the same with different constant loads, exactly like was observed in [31].

The battery management entity also reports the battery charge level in milliampere-hours (battery.reporting.current in Appendix A). Ampere-hour is a commonly-used unit for reporting the electric charge contained in a battery. For example, the theoretical maximum capacity of the default battery of the N900 is 1320 mAh, meaning that it could deliver a current of 1320 milliamperes for one hour. Alternatively, it could deliver a current of 132 mA for 10 hours. In order to examine the reliability of the reported charge level, the logger program was modified to log the changes in the charge level instead of the voltage, and the test described above was repeated. Figure 7 shows the charge level during audio playback. The charge level drops linearly with respect to time when a constant-load task is run on the device, implying that the reported value is reliable and that it is an excellent base for estimating the remaining battery life. Only when the battery is nearly empty, the charge level in

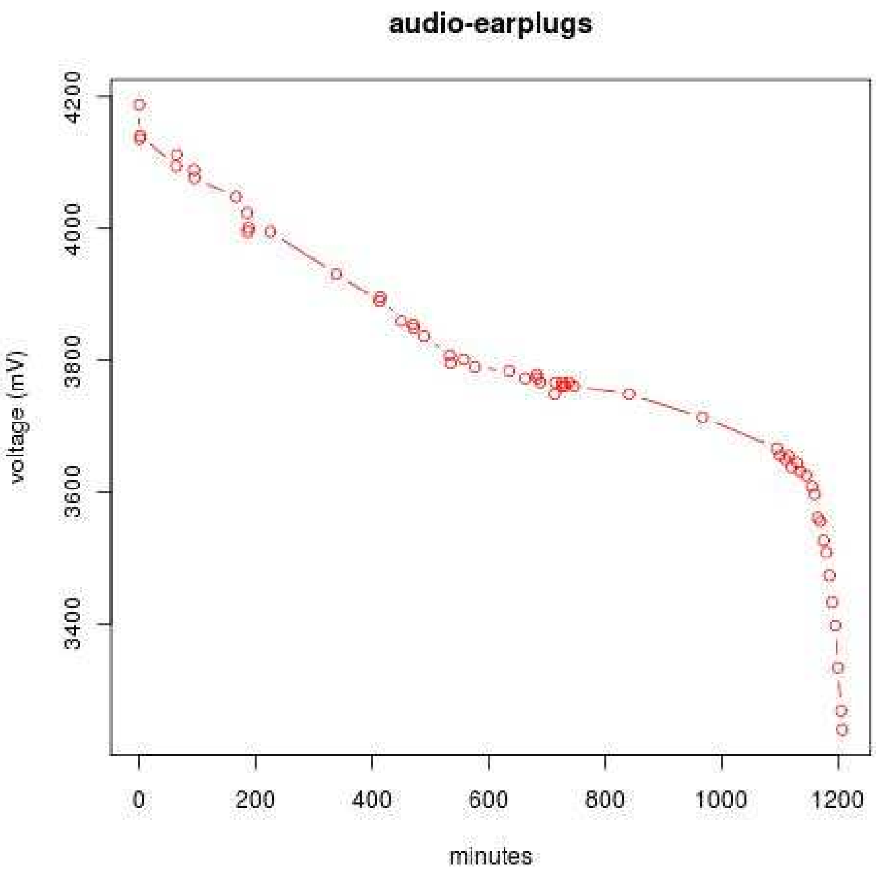


Figure 6: Battery dissipation curve (voltage) of Nokia N900 with constant audio playback.

millampere-hours is not updated correctly anymore. For example in Figure 7 the device keeps reporting the same value of 5 mAh for about 100 minutes, during which the device still works properly. Only after that the device shuts down. In the tests that were run during this work, the lowest value that the device reported was always 5 mAh, but the reason for this phenomenon remained unclear.

Assuming that the current demand of a task, i.e. the amount of current that a task consumes from the battery, is known, the estimated battery life for that task can be calculated using the equation:

$$t_{estimated} = \frac{C_{mAh}}{I_{task}}, \tag{1}$$

where C_{mAh} is the charge level informed by the device and I_{task} is the current demand of the task in question in milliamperes. For example, if the battery reports a charge level of 900 mAh and the current demand of a task is 10 mA, the battery will last for

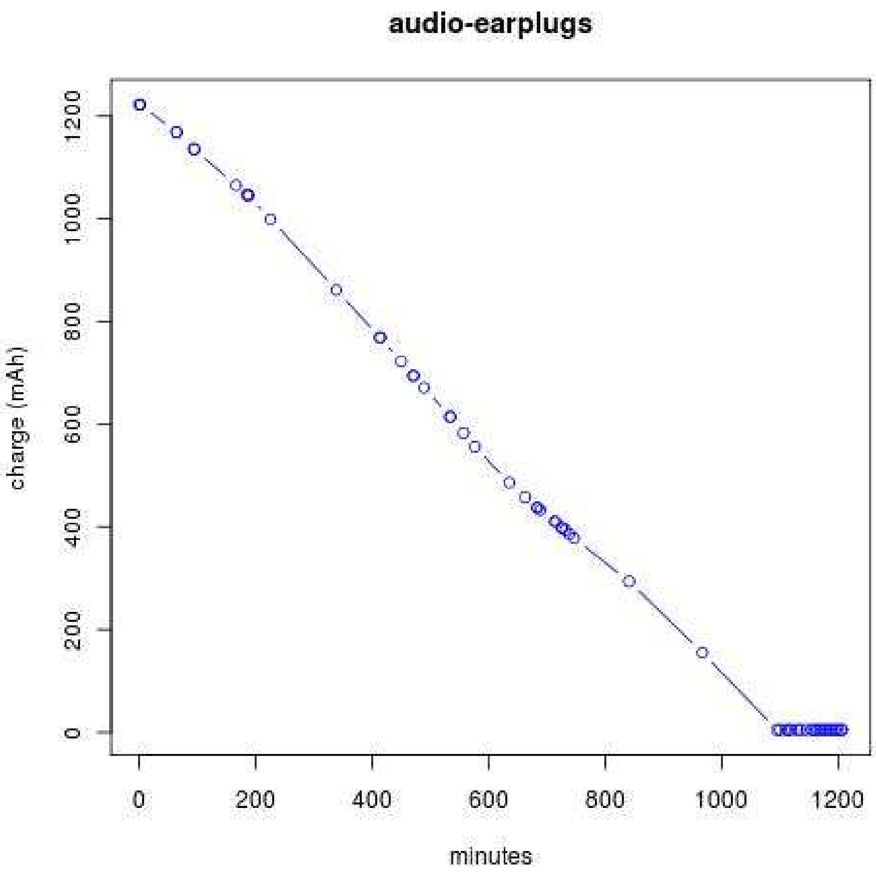


Figure 7: Battery dissipation curve (charge level) of Nokia N900 with constant audio playback.

90 hours running that task. Thus, in order to estimate for example the remaining call time, audio playback time and web browsing time, the average current demand of each task must be measured. The measurements that were conducted on the N900 to find out the current demand of several different tasks are presented next.

4.2 Power consumption measurements

As was concluded above, the battery life estimation for a task can be calculated by dividing the battery charge level (reported in milliampere-hours) by the current demand of the task in question. Therefore, the average current demand of the tasks for which the battery life wanted to be estimated, needed to be measured. This section presents the tasks that were chosen for the measurements and the setup used to measure the current demands. The results are discussed and compared to the battery life information reported by Nokia.

4.2.1 Measurement cases

The measurement cases, divided into constant-load and variable-load tasks based on whether the task in question requires user interaction, are listed below:

Constant load tasks:

- Standby, display off
- Standby, display on (default display brightness)
- Music playback (medium volume)
- Video playback (default display brightness, medium volume)
- 3G internet connection on
- WLAN internet connection on
- WLAN internet connection on and Skype client in online state

Variable load tasks:

- Calling
- Web browsing
- Text messaging
- Playing games
- Navigation

Measuring the current demand of the constant load tasks is straightforward: the device is left performing the given task for the measurement period during which the instantaneous current demand is recorded periodically. The average current demand can then be calculated from the recorded values. No user interaction is needed, and the measurements should produce similar results from one measurement to another.

Measuring the current demand of the variable load tasks needed to be thought out more carefully, since the energy consumption of the tasks depends on user-interaction. To measure the energy demand of calls, two things needed to be considered. First, when should the measurement be started: when the call is answered, once the "Call" button is pressed, or earlier? Since the goal was not to find out the theoretically exact power consumption of calls, but to inform the user about how long the device can be used for calling, also the selection of a contact/number prior to the actual call was included in the measurements. Secondly, the length of the test calls needed to be set. Based on information from the authors own call records, the average length of a phone call is around 2 minutes. Thus, to measure the energy demand of phone calls, ten phone calls of length from 90 seconds to 150 seconds were made. Separate measurements were made for GSM and WCDMA calls.

The power consumption of web browsing depends on both what kind of web pages are browsed and how frequently a new web page is downloaded (and also on the type of connection that is used). A user reading text-based documents, downloading a new page every 10 minutes consumes a lot less energy than another user browsing photos in Facebook. The web sites that were used in the measurements were chosen among the list of the most popular websites in the world, listed by DoubleClick [32]. The list includes, among others, Facebook, Google, Yahoo, and BBC. Chinese websites were excluded. Separate measurements were done for WLAN and 3G internet connections.

For measuring the power consumption of text messaging, twenty text messages of 100 characters were written and sent on the device. Also the average time that it took to write one message was measured. This was done because it would not make sense to report the remaining text messaging time. Instead an estimate on how many text messages can be written with the remaining battery capacity should be provided.

To measure the power consumption of playing games, two games were chosen: ”Blocks” which is by default installed on the N900 and ”Angry Birds”, an increasingly popular game that can be downloaded to the N900 from Ovi Store, Nokia’s application repository [33].

For navigation, the default map application of the N900, Nokia Ovi Maps, was used. Before starting the measurements, the map application was started and enough time was waited in order to find the initial GPS coordinates. The 3G network connection was enabled during the navigation. Since the measurements were conducted inside, no movement was involved. Hence, during the measurement the map application was only used for browsing the map, finding different locations and calculating routes between the locations.

4.2.2 Measurement setup

The setup for measuring the current demand of the tasks consisted of the mobile device, a resistor of 0.1 Ω, an external voltage source and a voltage meter. The battery of the mobile device was removed and the device was connected to the external voltage source. The voltage source and the mobile device were connected in series together with the resistor. The voltage meter was then connected in parallel to the resistor. The circuit diagram of the measurement setup is presented in Figure 8.

The voltage source was then adjusted to output a constant 4.1 V to power up the mobile device and the task to be studied was started. Since the resistance of the voltage source is high (a lot higher than that of the resistor), it can be assumed that all the current goes through the resistor. The voltage source keeps measuring the instantaneous voltage and using Kirchoff's laws, the instantaneous current demand of the N900 can be calculated using the following equation:

$$I = \frac{4.1V - U}{R}, \tag{2}$$

where U is the instantaneous voltage measured by the voltage meter and R is the 0.1 Ω resistance.

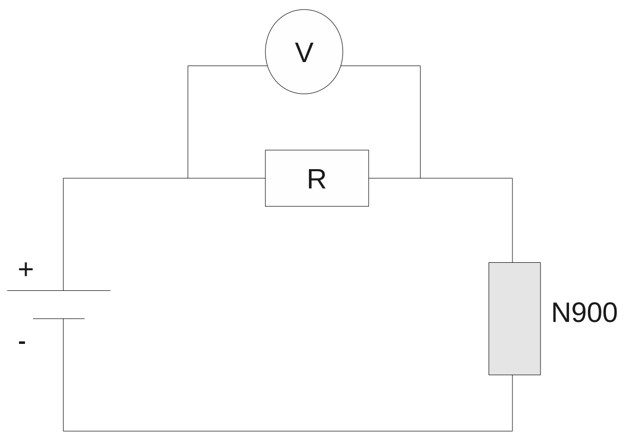


Figure 8: Circuit diagram for measuring the current demand of a mobile device.

4.2.3 Results

The average current demand of all the tasks listed in Section 4.2.1 are presented in Figure 9. The tasks with energy consumption lower than 200 mA are all actions, where the display is off. Sending text messages does not consume much more energy than only having the screen on. The tasks that consume more than 300 mA all keep the screen on and either transmit data using the internet connection (web browsing) or otherwise require a lot of computing (e.g. playing Angry birds).

One should take into account that the measurements were conducted without any movement. The energy consumed by for example calling, navigation and web browsing (using a 3G connection) could further increase if the device was moving.

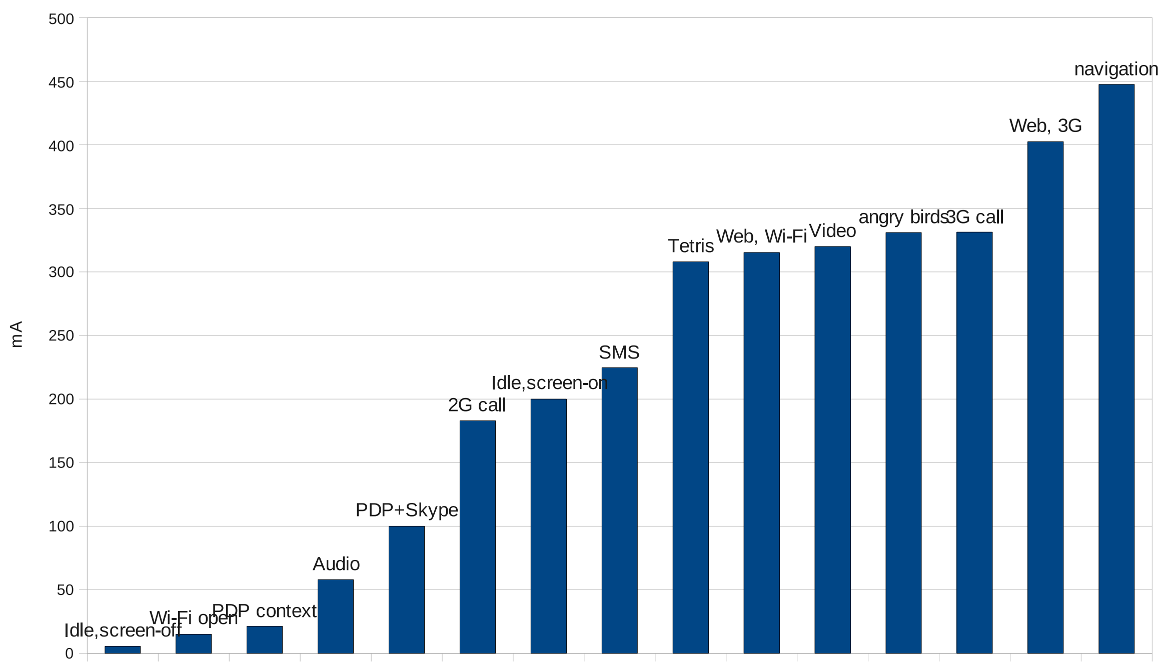


Figure 9: The current demand of different tasks on the N900.

Nokia reports some N900 battery life estimations in the technical details of the N900 [34]. The estimations provided are the following:

- Talk-time: Up to 6.3 / 4.3 hours (GSM / WCDMA)
- Standby time: Up to 278 /250 h (GSM / WSCMA)
- Active online usage: Up to 1 day
- Continuous browsing time: Up to 5.5 / 8 hours (WCDMA / WLAN)
- Video playback time: Up to 5.6 hours
- Music playback time: Up to 24.5 hours

These times are presumably based on the theoretical maximum battery capacity of 1320 mAh and they can converted to the current demand by dividing 1320 mAh with the time given. For example 6.3 hours of GSM talk time means that the current demand of GSM phone calls would be 210 mA (= 1320 mAh / 6.3 h). Table 1 below shows a comparison between the current demands calculated from the battery life figures reported by Nokia and the current demands obtained from the measurements in the previous section. The current demands reported by Nokia for standby, calling, video and audio are around 10% lower than the current demands measured here. The current demands reported for web browsing with both WLAN and WCDMA are close to 50% less of the ones measured here.

Table 1: Drain rate comparison.

Task	**Nokia (mA)**	**Our rates (mA)**
Standby (GSM/WCDMA)	5/7	5.5/8
Calls (GSM/WCDMA)	210/300	183/330
Video	270	320
Audio	56	58
Web browsing (WLAN)	165	314
Web browsing (WCDMA)	240	402

4.3 Charging time estimation

While researching the options for predicting battery life, the idea of predicting also charging time came up. According to a case study made in 2008 by World Business Council for Sustainable Development (WBCSD) [35], two-thirds of the power consumed by a Nokia mobile phone comes from the energy consumption of the chargers' no-load mode. The no-load mode means that the charger is plugged into the mains and thus, consumes energy, but the device is not actually being charged. The no-load mode occurs when the charger is not unplugged from the mobile device when the battery of the device is already full or when the charger is left plugged into the mains when the device is not connected to it anymore. Mobile device manufacturers are aware of this waste of energy and the no-load energy consumption of the chargers has been continuously reduced. For example Nokia reduced the no-load energy demand of their chargers by 70% from year 2000 to 2006 [35].

The no-load mode energy consumption could be further decreased with the help of users. According to [35], Nokia was the first to introduce battery-full alarms and a notification that asked the user to unplug the charger from the wall when the charging is finished. But if the device is left charging during the night, the charger will not be unplugged from the wall until the morning. If the device instead informed that "the battery will be full in two hours" when the charger is plugged in, the user might for example charge the device in the evening and unplug the charger before going to sleep. Thus informing the user about the remaining charging time could both increase user satisfaction (by giving her more information) and help in saving energy.

The battery management entity provides three values that can be used to detect when a charger is plugged in (maemo.charger.connection_status, maemo.rechargeable.-charging_status and battery.rechargeable.is_charging in Appendix A). Knowing the charge level of the battery, the charging time prediction could be calculated using the following equation:

$$t_{estimated} = \frac{C_{max} - C_{now}}{I_{charger}}, \tag{3}$$

where C$_{max}$ is the maximum charge level of the battery, and I$_{charger}$ is the charging rate of the charger used. The maximum charge level is reported by the BME (properties named battery.reporting.design or battery.-reporting.last_full, shown in Appendix A) but the charging rate is not. Furthermore, the N900 comes with two chargers: a wall charger and a USB charger, the charging rates of which differ from each other. The chargers can be identified by the charger type reported by BME (maemo.charger.type in Appendix A). For example the type of the default wall charger (Nokia AC-10E) of the N900 is "wall_charger".

To study the charging curves and to find out the charging rates of the two chargers, the logger program from the previous section was run while charging the phone. The battery was first drained to almost empty. The logger program was then started and a charger was plugged in. Once the battery was full, the logger program was stopped and the charge levels were plotted with respect to the times at which the charge levels were reported. It appeared that once the charger is plugged in, the battery charge level is not updated anymore. An update is received only when the battery is full or the charger is unplugged. As a result, the log files only showed the battery charge level at the moment before plugging in the charger and the final charge level once the battery was full. To obtain the charging curves, the charger was periodically unplugged from the device during the charging in order to receive updates of the charge level. The charging curves for the default wall charger and for the USB charger (Nokia CA-101) are shown in Figures 10 and 11.

Even though the device was in standby state, i.e. the current demand of the device was constant during the charging, the charging curves are not linear. This is due to the way that lithium-ion batteries are charged. First, a constant charging current is applied until a given voltage limit per battery cell is reached. After this, the charging current is continuously decreased until full charge is reached. For example, a battery could be charged up to 80 % of full charge in one hour, after which another hour would be needed for the remaining 20 %.

The average charging rates calculated based on the measurements were 680 mA for the wall charger, and 345 mA for the USB charger. Using these charging rates, the charging time could be roughly estimated. The estimation could be further improved by calculating it piece-wise. For example. the charging rate of the wall charger remains constant (around 900 mA) until 800 mAh is reached, after which

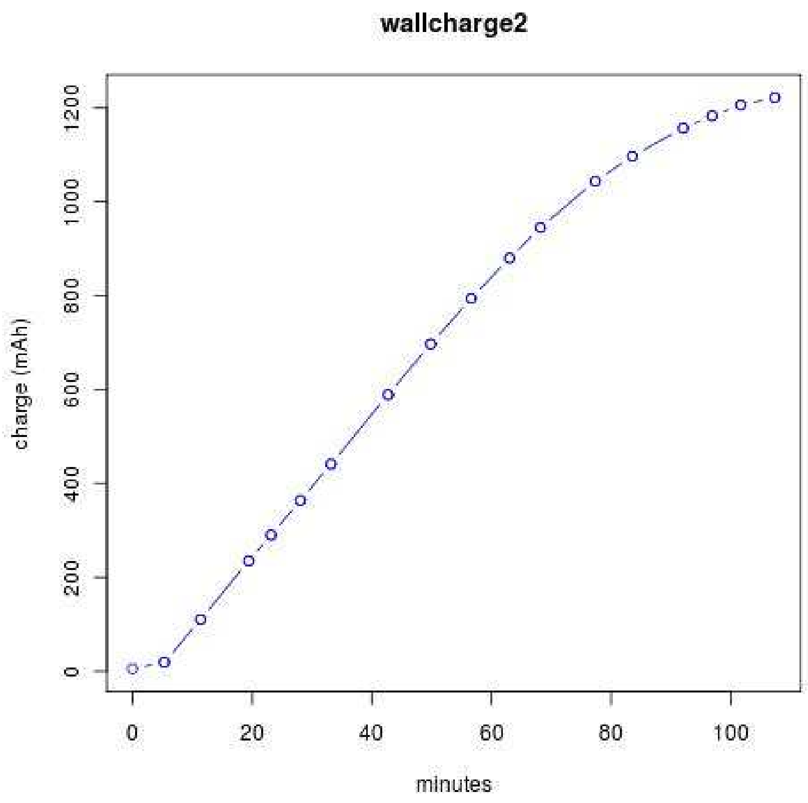


Figure 10: Battery charging curve for the N900 default wall charger (Nokia AC-10E).

the rate slowly decreases.

The problem of this approach is that there are several different types of chargers for the N900, each of which can have a different charging rate but the same type. For example, another wall charger (Nokia AC-5E) that can be connected to the N900, appears in the system with the same type ("wall_charger") as the default wall charger of the N900 but it has a different charging rate. This would result in an incorrect charging time estimation.

4.4 Summary

The possibilities for predicting battery life and charging time on the N900 mobile computer were examined in this chapter. The initial idea was to use the battery voltage level reported by the N900 for the prediction, as was done in [31]. After further research it was found out that the device is able to directly report the charge level of the battery in a reliable manner. Dividing the battery charge level by the current demand of a task (e.g. audio playback) produces the battery life estimation for the task in question. The power consumption measurements that

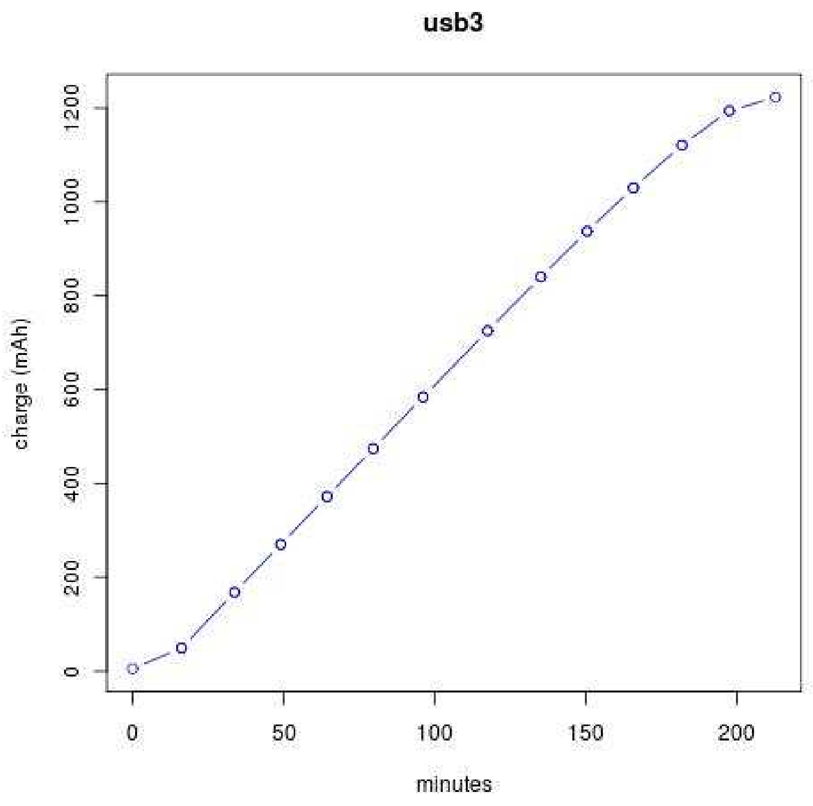


Figure 11: Battery charging curve for the N900 USB cable (Nokia CA-101).

were conducted to find out the current demand of different tasks together with the results were also presented. Thus, it was found out that providing the N900 users with reliable battery life estimations for different tasks is feasible and relatively straightforward.

The last section of the chapter stated that leaving the mobile device chargers plugged into the mains after the charging is finished stands for a significant amount of energy used by mobile devices. Providing mobile device users with charging time estimations could both increase user satisfaction and help in saving energy. The charging curves for two different chargers of the N900 were obtained, and it appeared that the charging rates do not remain constant when charging the battery from empty to full. Also, identifying the chargers appeared to be a problem, since two chargers with different charging rates can appear in the system with the same name. A charging time estimate could anyway be provided for example for the default wall charger of the N900 using a piece-wise defined charging rate.

5 Implementing the battery monitoring application

The implementation of a battery monitoring application for the Nokia N900 mobile computer is described in this chapter. The application was named BattInfo, and this name is used from here onwards to refer to it. The first subsection of this chapter states the goals and requirements that were set on the application. The software platform of the N900 and the services provided by it are described in the second subsection. Sections 5.3-5.5 present BattInfo and explain how it actually works. Also the three additional features of BattInfo (two battery-low alarms and a power-save mode) are described. A user test that was conducted in order to evaluate BattInfo and to receive feedback from it is presented in the following chapter.

5.1 Goals and requirements

The goal of this work was to develop a battery monitoring application that gives the user more elaborate information about the remaining battery capacity and battery life. The current method for showing the remaining battery capacity in most devices is the battery charge meter, that only has coarse accuracy. Furthermore, the battery charge meter does not give any information about the remaining battery life, which would be more useful to the users.

The battery monitoring application should provide the user with battery life estimations for different tasks that are typically performed with the mobile device, such as remaining talk time, audio playback time, video playback time, navigation time and web browsing time. Also a battery life estimation based on the usage history, which will help the user to estimate how long the battery will last with her typical usage pattern, should be provided.

The information should be presented to the user via a self-describing graphical user interface (GUI), and all of its functions should be easy to use. Part of the information that the application is able to provide shall be presented to the user via a desktop widget, that is always visible and allows the user to find out the status of the battery with a quick look. The desktop widget should be customizable so that the user can choose which battery life estimations, e.g. talk time and audio playback time or talk time and accurate battery capacity, are shown. The full set of information that the application provides should be hidden by default, but easily accessible from the desktop widget by one or two clicks.

5.2 Maemo

Maemo is a software platform developed by Nokia for mobile devices [36]. The first Maemo device was the Nokia 770 Internet Tablet, released in 2005. The newest version of Maemo is Maemo 5, also known as Fremantle, which is used on the N900. Maemo is based on the Linux operating system kernel and uses Debian package management.

D-Bus

D-Bus is a message bus system used for interprocess communication (IPC) and remote procedure calls (RPC) on Maemo. There are two well-known buses, the system bus and the session bus. The system bus delivers signals from the Hardware Abstraction Layer (described below), such as "new hardware device added" or "hardware state changed". The D-Bus session bus is specific to each user session, and could provide a service such as opening the user's web server. Applications that want to use D-Bus must have a service name. Each service can have one or more objects, each of which can then have one or more interfaces. In addition to the service name, they have a path for their objects, and methods/signals with some interfaces. Each interface has a defined set of properties that can be accessed using the methods and signals provided by the interface. Table 2 shows two examples of complete D-Bus addresses.

Table 2: D-Bus addresses.

D-Bus addresses	**Battery status**	**Network status**
Service name	org.freedesktop.hal	com.nokia.phone.net
Object path	/org/freedesktop/Hal/devices/bme	/com/nokia/phone/net
Interface	org.freedesktop.Hal.Device	Phone.Net
Signal	PropertyModified	radio_access_technology

The set of D-Bus objects varies from a Linux system to another, depending on the underlying hardware. A command line tool called dbus-monitor [37] can be used to display the signals sent by both applications and the system on D-Bus. This tool was used for finding the set of signals sent on the N900, that could be used by the battery management application.

Hardware Abstraction Layer and Battery Management Entity

The Hardware Abstraction Layer (HAL) is a software project for Unix-like computers [38], that offers a toolkit for discovering and configuring the computer hardware and allows for applications to discover and use the many different hardware and their special properties through a uniform API. The Battery Management Entity (BME) is a part of HAL and, as the name suggests, it reports information concerning the battery. BattInfo uses the information provided by BME/HAL and delivered by D-Bus for calculating and updating battery life estimations.

Application development on Maemo

The Maemo development environment supports by default the C language for application development, but support can be added for C++ and Python languages as well. After the PR1.2 update to Fremantle, also Qt has been supported. Qt is a C++ cross-platform application and UI development framework [39], and in addition to Maemo, it currently supports also the following platforms: embedded Linux, MAC OS X, Windows, Linux/X11, Meego and Symbian. Applications such as Google Earth, Skype and Opera web browser are developed using Qt. With the idea of later porting the application also for other mobile platforms, Qt was chosen as the programming language for BattInfo.

5.3 Application overview

When BattInfo is launched, a desktop widget is placed on the current desktop of the device. Figure 12 shows a screenshot of the BattInfo desktop widget, displaying the accurate battery charge level and the estimated remaining standby time. From the settings dialog, shown in Figure 13, the user can customize the information shown in the desktop widget. The three other options shown in the figure are explained later.

When the desktop widget is clicked, the BattInfo main window is opened (Figure 14). From the main window, the user can easily navigate to the settings and other options of BattInfo. Clicking the "Show battery state" button opens the "battery state" view, shown in Figure 15. Initially, clicking the desktop widget opened directly the "battery state" view, but based on a few preliminary user tests, this design did not encourage the users to explore the application. The main window of Figure 14 was thus added in order to facilitate and motivate navigation inside the application. However, the user test (presented in Section 6) in the end showed that the main window was considered useless and was thus removed from the final version of BattInfo.

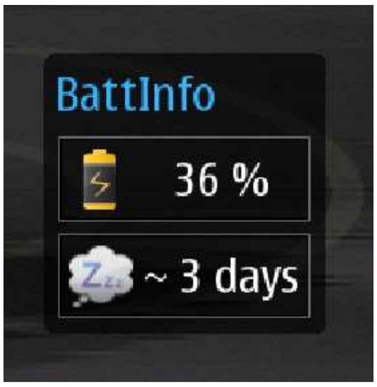
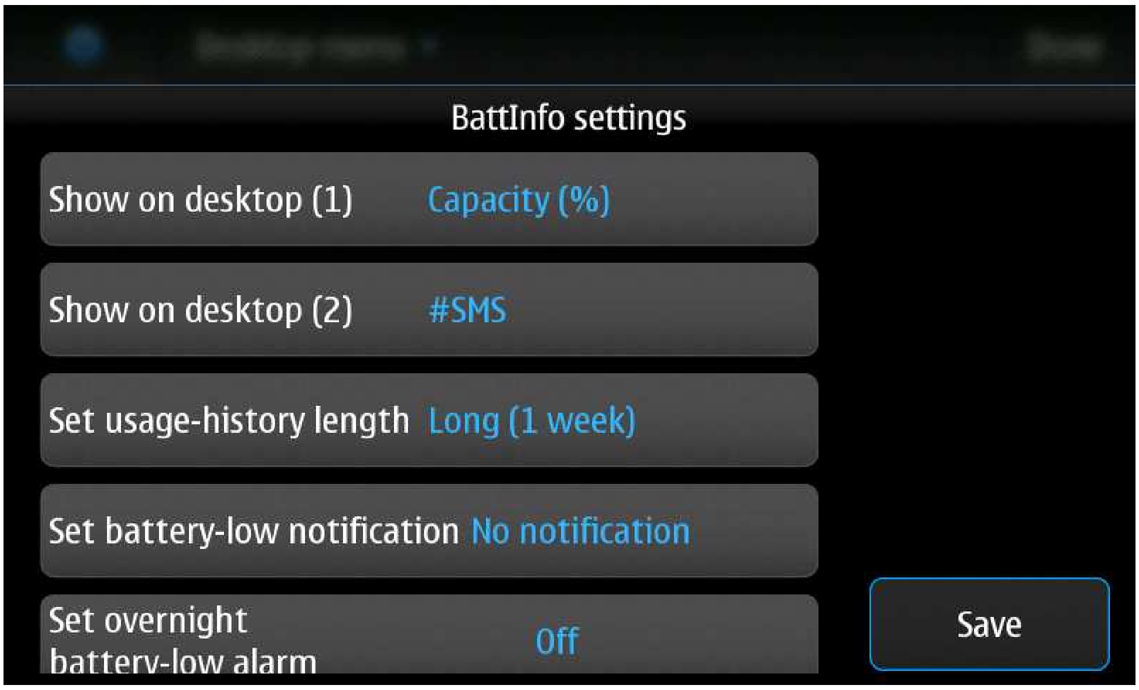


Figure 12: BattInfo desktop widget.


BattInfo settings

Show on desktop (1) Capacity (%)

Show on desktop (2) #SMS

Set usage-history length Long (1 week)

Set battery-low notification No notification

Set overnight battery-low alarm Off

Figure 13: BattInfo settings dialog.

The values under "Estimated battery life", excluding the usage-based estimate, are calculated based on the battery charge level and on the demand of current of the tasks. In the example figure, the battery could provide for example 1 hour of video playback or alternatively 6.5 hours of music playback. In addition to informing the user about the remaining battery life, the values reported on the left side of the window can help the user to understand which activities have high power consumption and which do not.

The battery life estimations are calculated according to Equation 1, where I_{task} is the current demand of the task in question and C_{mAh} is the battery charge level. If the task is calling, the current demand depends on the device's network status, i.e.

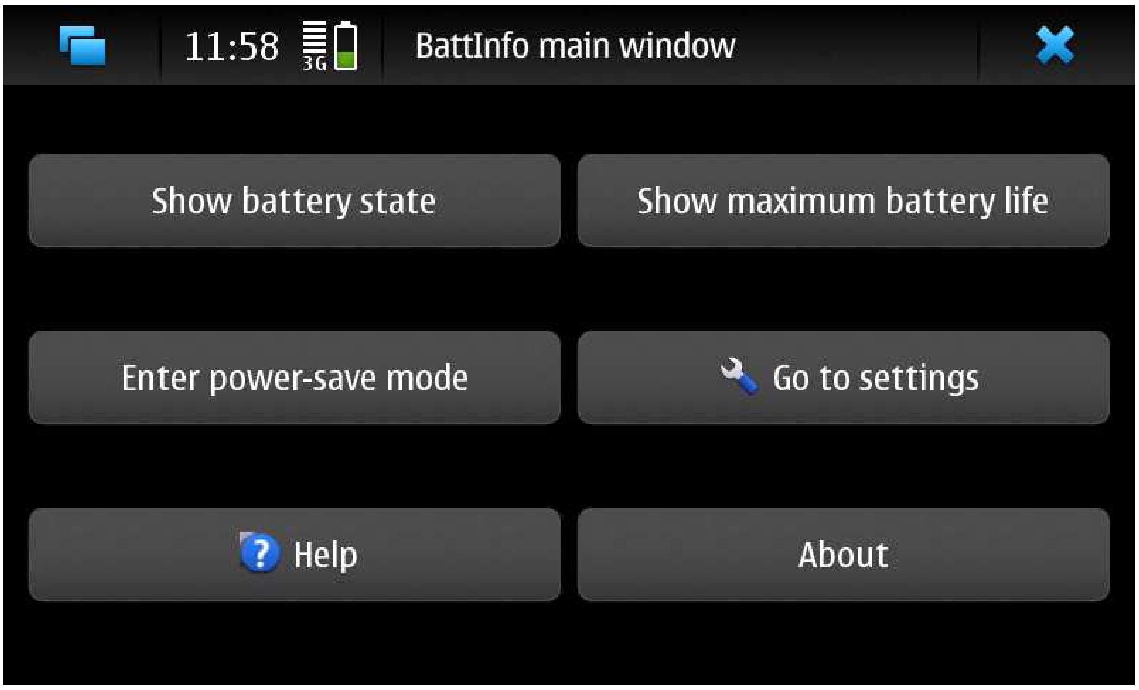


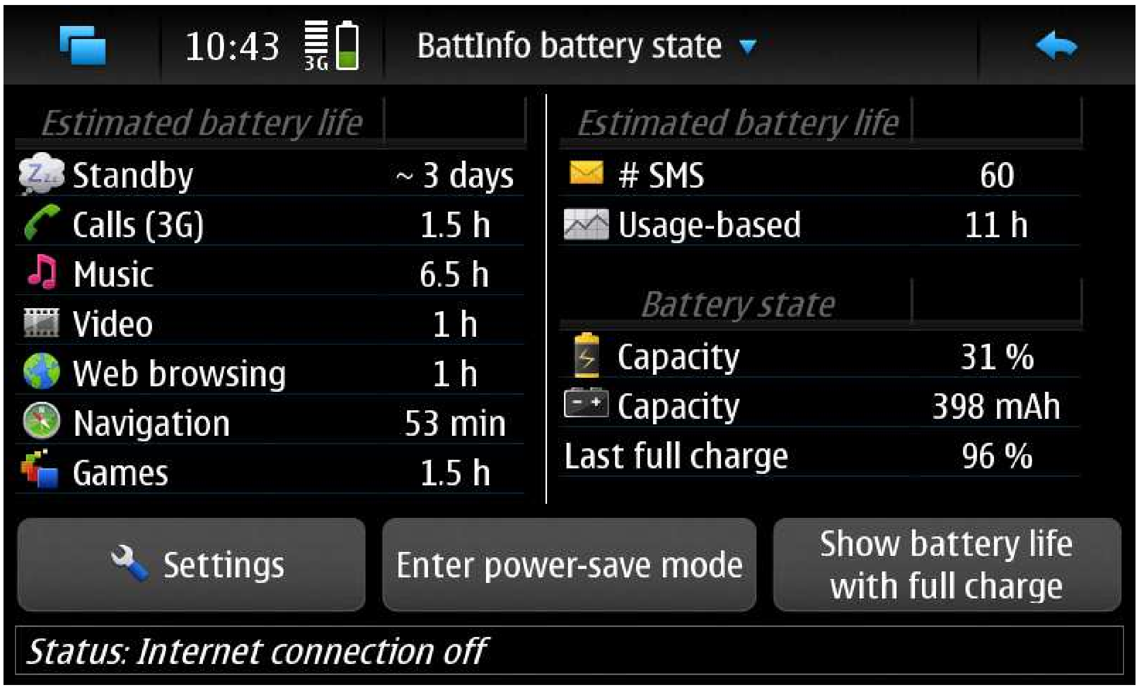
Figure 14: BattInfo main window.

whether the device is connected to a GSM or to a WCDMA network.

BattInfo also takes into account the amount of power consumed by an internet connection when calculating the estimated battery lives. If an internet connection is enabled, the battery life will naturally be shorter. Figure 16 shows a screenshot of the BattInfo main window with same battery charge level as in Figure 15 but with a WLAN connection on. As can be seen by comparing the two screenshots, the estimated battery lives are much shorter when the internet connection is on. The power consumption of the internet connection is not taken into account when calculating the usage-based battery life.

Equation 1 produces the estimated battery life t, the unit of which is hours. The calculation can result in values such as 0.643 hours or 243.25 hours, which are not very user-friendly. Therefore the estimated battery life is rounded according to the following rules:

1. If $t < 1h$: convert t to minutes and round it to the closest full minute
2. Else if $1h < t <= 10h$: round t to the closest half an hour
3. Else if $10h < t <= 48h$: round t to the closes full hour
4. Else (i.e. $t > 48h$): convert t to days (48h) and round it to the closest full day.



10:43 BattInfo battery state

Estimated battery life	
Standby	~ 3 days
Calls (3G)	1.5 h
Music	6.5 h
Video	1 h
Web browsing	1 h
Navigation	53 min
Games	1.5 h

Estimated battery life	
# SMS	60
Usage-based	11 h

Battery state	
Capacity	31 %
Capacity	398 mAh
Last full charge	96 %

Status: Internet connection off

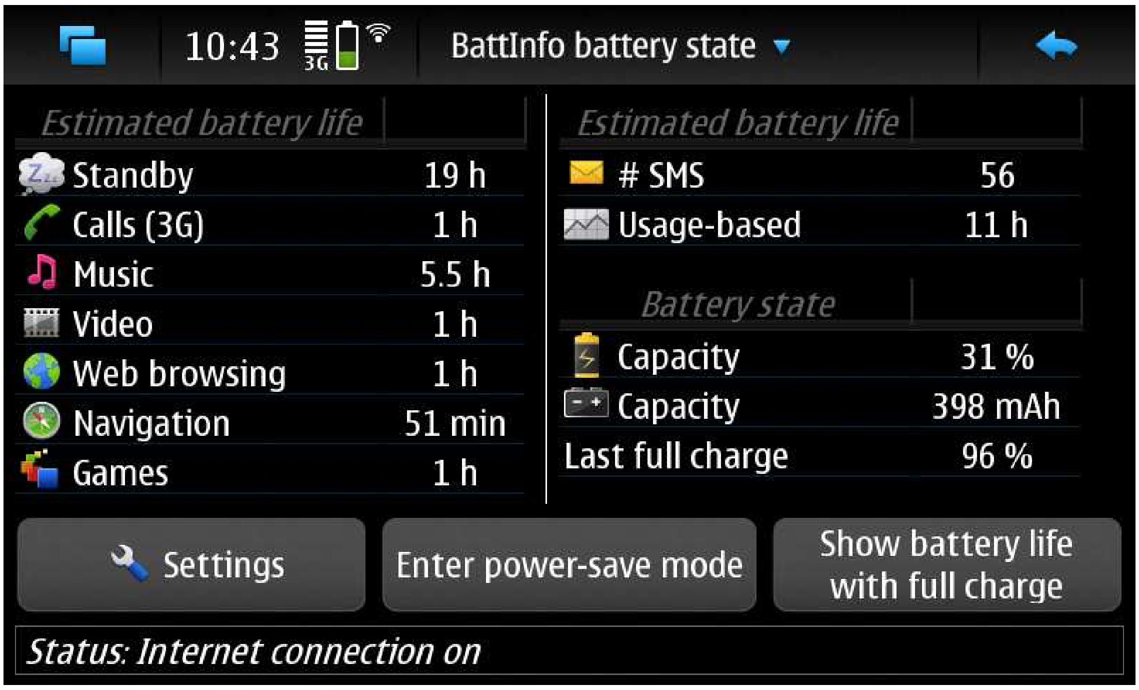
Figure 15: BattInfo battery state.

BattInfo monitors the battery state (battery charge level) and the network state (whether the device is connected to a GSM or to a WCDMA network) using signals sent on the D-Bus by two entities: the battery management entity and the ”Phone.net” entity. The complete D-Bus addresses of these two are shown in Table 2. The status of internet connections is monitored using a signal provided by the QNetworkConfigurationManager class.

5.4 Estimating battery life based on usage history

As shown in Figure 14, BattInfo also calculates a usage-based battery life estimation, which is based on the power consumption history of the device. This gives the user an idea of how long the battery will last if the device is used more or less in the same way as it was used before.

The initial idea for recording the power consumption history was to log the battery charge level every one hour (or using another fixed interval), and then calculate the average current demand during every interval. The usage-based battery life estimation could then be calculated using Equation 1. This method was implemented, but it appeared that the battery management entity of the device stops reporting the battery charge level when the device goes into ”inactive state”, for example when the screen is turned off. The reason for this is presumably power-saving: there is no reason to update the battery charge level if nobody can see it. Consequently, the



10:43 BattInfo battery state

Estimated battery life	
Standby	19 h
Calls (3G)	1 h
Music	5.5 h
Video	1 h
Web browsing	1 h
Navigation	51 min
Games	1 h

Estimated battery life	
# SMS	56
Usage-based	11 h

Battery state	
Capacity	31 %
Capacity	398 mAh
Last full charge	96 %

Status: Internet connection on

Figure 16: BattInfo battery state, internet connection enabled.

approach of logging the battery charge level periodically did not work, and another method needed to be thought out.

The approach that was next implemented and that was found to work correctly, was to record the times at which new battery charge level updates are received. The average current demand can then be calculated online using the equation below:

$$I_{usage-based} = \frac{C_i - C_{i-1}}{t_i - t_{i-1}}, \tag{4}$$

where C_i and t_i are the newest charge level update received and the current time, and C_{i-1} and t_{i-1} are the previous reported charge level and the time at which it was reported. After this the usage-based battery life estimation can be calculated using Equation 1.

In general, a history-based estimation is the more accurate the longer the history taken into account when calculating the estimation [31]. The length of one week was considered long enough to provide reliable usage-based battery life estimations and it is used as the default history-length in BattInfo. In some cases the user might be interested in having a battery life estimation based on only the recent usage-history, which is why BattInfo allows for the user to change the length of the history that is considered when calculating the usage-based estimation. The user can choose from three options: short (2 h), medium (24 h), long (1 week). The set of options was reduced to this set of three in order not to give the user too much choice, which might just be confusing. The history-length can be changed from the settings dialog, shown in Figure 13.

5.5 Additional features

BattInfo has three features in addition to the battery life estimation, each of which is targeted to detect and cope with low battery. The first one of them is a battery-low alarm that can be configured by the user. The user can decide whether the alarm is active or not, and set the charge-level in percentage (e.g. 10, 20 or 30 %) at which the alarm is triggered. The purpose of this alarm is to enhance the default battery-low alarm of the N900, which is played only when the battery charge level reaches 0 %.

The second additional feature is an overnight battery-low alarm, the purpose of which is to notify the user in the evening if the battery would be critically low the next morning. If the alarm is activated, BattInfo checks the battery charge level at a specified time in the evening and calculates the remaining standby time. If the capacity would be critically low the next morning, the alarm is be played. Both the battery-low alarm and the overnight battery-low alarm can be turned on and configured from the settings dialog (Figure 13).

After testing, a major problem with these alarms was noticed. As it was already stated, the battery charge level is not updated while the device is inactive. This means that if the device has stayed in standby state for a longer time, the reported battery charge level can suddenly drop many units when the device is activated again. To demonstrate the problem with the alarms, let us consider a case where the user has set the battery-low alarm to notify when the battery charge level reaches 30%. At 32%, the device is left idle and the screen is turned off. After a couple of hours the user turns the screen on again, and a charge level update of 26% is received. Thus, the battery-low alarm has not been triggered correctly. The overnight battery-low alarm suffers from the same problem. A possible fix for this problem would be to programmatically activate the system, e.g. by turning on the screen and then immediately turning it off it in order to receive a charge-level update when needed. This fix would require additional energy, which is not wanted if the battery is already low, and therefore the fix was not implemented. However, the alarms were not removed from the application. They were kept in BattInfo, but a note informing the user about their possible malfuctioning was added to the help dialog of the application.

The third feature added to BattInfo is the power-save mode. The idea of the power-save mode is to easily allow the user to save battery capacity when the battery charge level is low. The three things that the power-save mode does in order to reduce the power consumption of the device are: 1) dimming the screen 2) closing the internet connection and 3) forcing the phone to GSM network. These power-saving actions are activated by running the following system commands:

1. gconftool -s -t int /system/osso/dsm/display/display_brightness 1
2. run-standalone.sh dbus-send –system –dest=com.nokia.icd /com/nokia/icd_ui com.nokia.icd_ui.disconnect boolean:true
3. run-standalone.sh dbus-send –system –type=method_call –dest=com.nokia.phone.net /com/nokia/phone/net Phone.Net.set_selected_radio_access_technology byte:1

When the power-save mode is exited, the display brightness is set back to the brightness level before entering the power-save mode and the radio access technology is set back to "dual mode". Internet connections are not re-established. The activation of power-save mode is indicated in the BattInfo desktop widget by a green icon, as shown in the figure below. BattInfo exits from power-save mode automatically when the charger is plugged in. The user can also exit the power-save mode manually by clicking a button in the main window: when the power-save mode is activated, the button "Enter power-save mode" in Figure 14 changes into "Exit power-save mode".

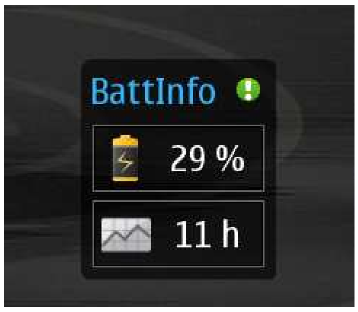


Figure 17: BattInfo desktop widget when power-save mode is activated.

5.6 State machine and module structure

The state machine of BattInfo is depicted in Figure 18. When the application starts, it loads the possible settings. The settings are used to save the user's choises for what is shown on the desktop widget, the settings of the two alarms, and the collected usage-history. Thus, after e.g. a restart of the device, the application will startup with the same configuration as before. After this, the initial state of the device, i.e. the status of the battery, network and internet connection, is queried.

Based on the state of the device, BattInfo calculates the necessary battery life estimations and shows the desktop widget on the desktop. At this point BattInfo

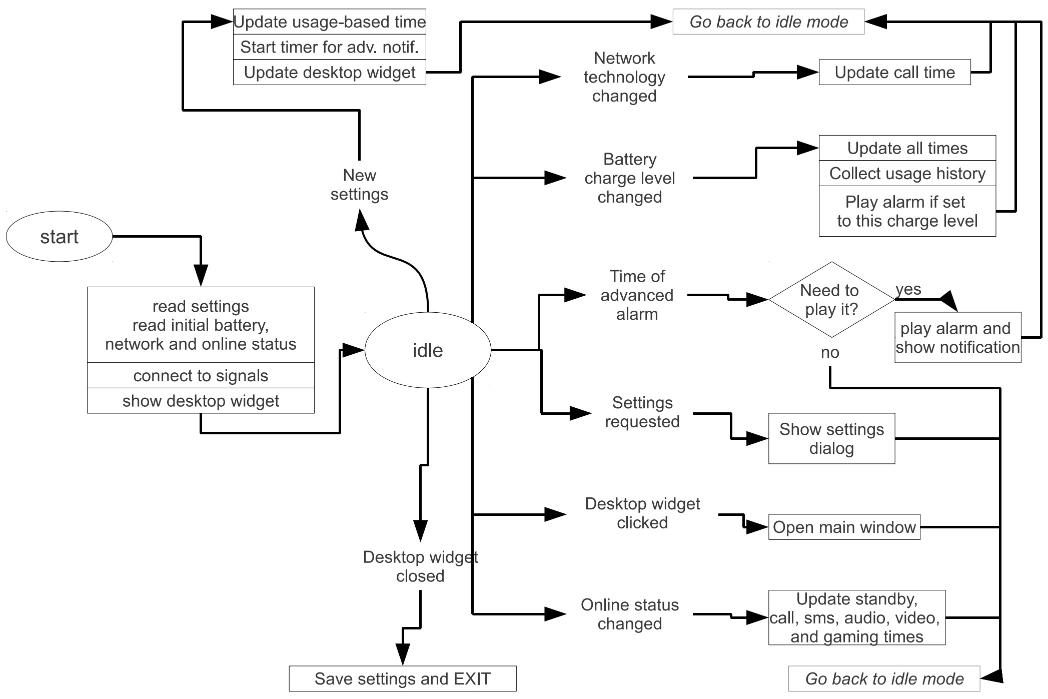


Figure 18: BattInfo state machine.

connects to the signals from D-Bus, QNetworkConfigurationManager and to the signals triggered by user input. Also the possible alarm for the advanced notification is set. When the time of the notification arrives, a signal is delivered to BattInfo, and it will calculate whether it is necessary to play the alarm.

The receival of the signals triggers different actions in the application, as shown in the state machine. For example, when the battery charge level changes, all the battery life estimations (or those visible to the user) are updated and usage-history is collected, but when the network technology changes from e.g. 2G to 3G, only the remaining call time is updated. Upon receiving a signal, BattInfo performs the necessary calculations and accordingly updates the desktop widget (and the main window if it is visible). After this, BattInfo stays in the idle state until the next signal is received.

BattInfo consists of three objects: the desktop widget, the main window and the settings dialog. The desktop widget contains most of BattInfo's functionalities, in addition to showing and updating the desktop widget. It loads/writes the settings from/to a file when the application starts/exits, takes care of reacting to the different signals and, when requested, opens the main window or the settings dialog. The main window and the settings dialog are separate objects, that only send/receive signals to/from the desktop widget. The main window, for example, does not follow the status of the battery itself. Instead it receives signals from the desktop widget and updates itself accordingly. Similarly, the settings dialog does not read or write the application settings to a file: the desktop widget maintains the settings information and communicates them to the settings dialog via signals when necessary. The communication of BattInfo and the device and BattInfo's internal communication is depicted in Figure 19.

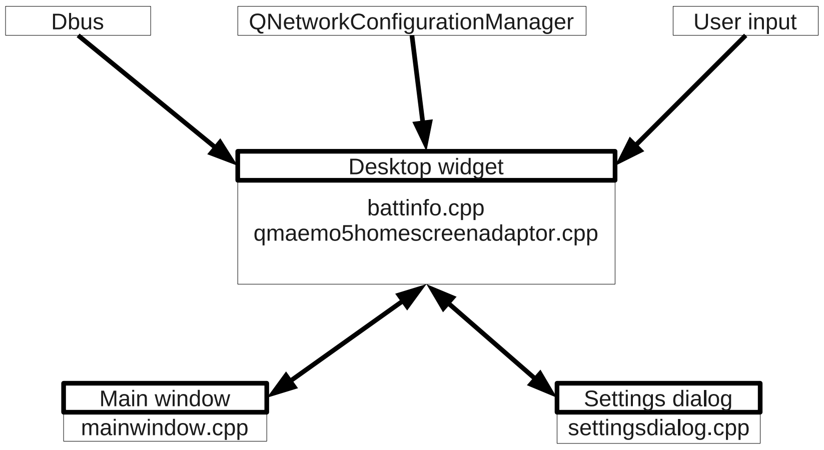


Figure 19: BattInfo's module structure.

5.7 Summary

This chapter started by defining the goals that were set to the battery management application. It stated that the application should give the user more elaborate information on the remaining battery life and present the information via an intuitive, easy-to-use graphical user interface. The implementation platform (Maemo) and its most important components were described next, after which the actual implementation, named BattInfo, was presented.

BattInfo monitors the battery charge level, and based on the average current demands of different tasks (measured in Chapter 4), calculates several battery life estimations. All the information is shown to the user via a graphical user interface, which consists of a desktop widget and a bigger, full screen window. BattInfo takes into account the complete state of the device (battery, network and internet connection state) when calculating the battery life estimations. Also the three additional features of BattInfo, two battery-low alarms and a power-save mode, were described. It was noticed that the battery-low alarms do not work properly in all situations, but they were not removed from the application. Instead a note was added to the application's help dialog to inform the user about the possible malfunctioning.

6 User testing

In order to get feedback about BattInfo and to examine its usefulness and usability, a user test was conducted. The user testing was conducted in collaboration with usability specialists from Helsinki Institute for Information Technology (HIIT). This chapter describes the user tests, their results and how the results influenced the development of Battinfo.

6.1 Arrangements and questionnaires

First, in order to make the application easily installable for the test users, a Debian package was created and it was placed inside a repository. The repository was made available online, and the users were sent links to the repository, allowing one-click install. The process of distributing the application to the users and making it easily installable is described in Appendix B.

The user study started with a background questionnaire, which collected demographic information (such as age, gender and nationality) about the users and the users received a link to the BattInfo repository upon answering a demographic questionnaire. After installing and starting to use BattInfo, three questionnaires were sent to the users: the first one after three days, the second one after one week and the last one after two weeks of usage.

The first and the third BattInfo questionnaires consisted of two types of questions: 1) statements, such as "BattInfo helps me to manage my phone battery" and "I find BattInfo easy to navigate", to which the users had to respond by choosing the most appropriate alternative from the five following options: "I totally disagree", "I disagree", "I neither agree or disagree", "I agree" and "I totally agree", and 2) word pairs, such as "Practical - Impractical", "Dull - Captivating" and "Confusing - Clearly structured", where the users had to choose from a seven-scale the word they considered more appropriate for describing BattInfo. The majority of the questions in the first and the third questionnaires were the same. The second questionnaire was the Attrakdiff Single Evaluation feedback form [40], which also consists of word-pair questions.

6.2 Results

The test users were mainly researchers and professors from Aalto University's Department of Communications and Networking (Comnet). The majority of them was Finnish, male, and 25-44 years old. The test users were clearly more technically oriented than an average user. Initially, 16 users enrolled in the study and 15 of them answered to the demographic questions. 12 users completed the first questionnaire about BattInfo and 10 users the second one. The whole survey was completed by nine users. Four of the users who answered to all the questionnaires were also interviewed face to face after the test period in order to get more detailed feedback.

The four statements below were contained in both the first and the third questionnaire:

1. Taking BattInfo into use has been easy.
2. BattInfo helps me to manage my phone battery.
3. I find BattInfo easy to navigate.
4. I use BattInfo to plan my battery usage.

The distribution of the answers to these four statements are shown in tables 3 (results from the first questionnaire) and 4 (results from the third questionnaire). Overall it can be concluded that BattInfo was relatively easy to be taken into use and most users also thought that it was also easy to navigate within the application.

Table 3: Distribution of the answers to the statements for the first questionnaire. (*The actual questionnaire used "I neither agree or disagree" instead of "Neutral".)

	I totally disagree	**I disagree**	**Neutral***	**I agree**	**I totally agree**
Statement 1	8.3%	8.3%	0.0%	58.3%	25.0%
Statement 2	8.3%	25.0%	8.3%	58.3%	0.0%
Statement 3	0.0%	8.33%	33.3%	41.7%	16.7%
Statement 4	0.0%	50.0%	33.3%	16.7%	0.0%

Table 4: Distribution of the answers to the statements for the final questionnaire. (*The actual questionnaire used "I neither agree or disagree" instead of "Neutral".)

	I totally disagree	**I disagree**	**Neutral***	**I agree**	**I totally agree**
Statement 1	0.0%	0.0%	0.0%	66.7%	33.3%
Statement 2	11.1%	0.0%	0.0%	77.8%	11.1%
Statement 3	0.0%	22.2%	22.2%	44.4%	11.1%
Statement 4	11.1%	33.3%	22.2%	33.3%	0.0%

In the word-pair questions of the first and the third questionnaire, the users had to choose from a seven-scale the most appropriate alternative. The distribution of the responses, for both the first and the third questionnaire, is shown in Table 5. On each row, the upper values are from the first questionnaire and the lower values from the final questionnaire. Based on the results, BattInfo is considered more undemanding than challenging, more practical than impractical, pleasant instead of unpleasant, attractive instead of ugly and rather trustworthy. In addition, most users would probably continue using the application in the future. The questionnaires also asked whether the users had used the power-save mode. Most of the users had never used it, but some users reported using it somewhat regularly.

Table 5: Distribution of the answers of the word-pair questions for the first and the third questionnaire. For each row, the upper values are from the first questionnaire, and the lower values from the final questionnaire. (*C.s = Clearly structured, Not tr. = Not trustworthy.)

	1	**2**	**3**	**4**	**5**	**6**	**7**	
Undemanding	16.7% 11.1%	41.7% 22.2%	0.0% 44.4%	25.0% 22.2%	8.3% 0.0%	0.0% 0.0%	8.3% 0.0%	Challenging
Dull	16.7% 0.0%	8.3% 11.1%	16.7% 11.1%	25.0% 33.3%	25.0% 22.2%	8.3% 22.2%	0.0% 0.0%	Captivating
Practical	0.0% 0.0%	25.0% 44.4%	33.3% 33.3%	0.0% 11.1%	16.7% 0.0%	8.3% 22.2%	0.0% 0.0%	Impractical
Confusing	8.3% 0.0%	0.0% 0.0%	33.3% 0.0%	16.7% 33.3%	8.3% 22.2%	25.0% 44.4%	8.3% 0.0%	C.s.*
Pleasant	0.0% 0.0%	41.7% 44.4%	8.3% 33.3%	41.7% 11.1%	8.3% 11.1%	0.0% 0.0%	0.0% 0.0%	Unpleasant
Ugly	8.3% 0.0%	0.0% 11.1%	8.3% 11.1%	33.3% 22.2%	16.7% 33.3%	33.3% 22.2%	0.0% 0.0%	Attractive
Trustworthy	0.0% 0.0%	33.3% 33.3%	16.7% 22.2%	33.3% 33.3%	8.3% 0.0%	0.0% 11.1%	8.3% 0.0%	Not tr.*
Would not use	0.0% 0.0%	8.3% 11.1%	0.0% 0.0%	25.0% 22.2%	16.7% 22.2%	16.7% 33.3%	33.3% 11.1%	Would use

The results of the Attrakdiff questionnaire are shown in Figure 20 below. The figure reveals that the user interface of BattInfo was considered "neutral", and that there is room for improvement in terms of usability and in terms of hedonic quality. The "dullness" of the graphical user interface of BattInfo came out also in the personal interviews - BattInfo clearly was not considered graphically stylish and appealing.

In addition to the above-mentioned dullness of the GUI, also the following lackings and issues came up in the free textual feedback of the questionnaires and in the user interviews:

- The main window (Figure 14) was considered useless and disturbing.
- The several horizontal and vertical lines in the battery status view (Figure 15) are disturbing, and furthermore, the "Estimated battery life" headers are misplaced, since the estimated times are actually shown in the second column.
- The "Status" label, which shows whether the internet connection is on or off (Figure 15), is unclear and occupies too much space.

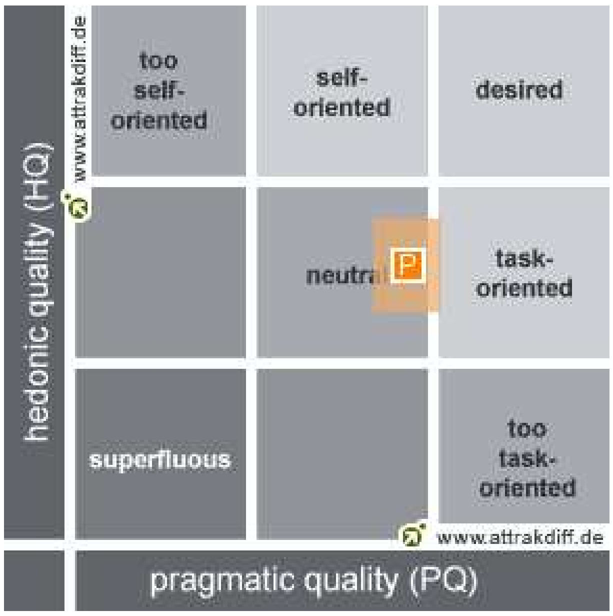


Figure 20: How user-friendly and attractive is BattInfo [40]?

- The battery state view was considered boring.
- The two battery capacities (% and mAh) have different icons even though they report basically the same thing.
- It should be possible to place more (or fewer) than two items in the desktop widget.
- The green icon for power-save mode (Figure 17) was not clearly enough associated with power saving: one user was not sure what it meant.
- Two users would have liked to have the possibility of setting a longer usage-history than the maximum of one week that BattInfo used, e.g. two weeks or one month.

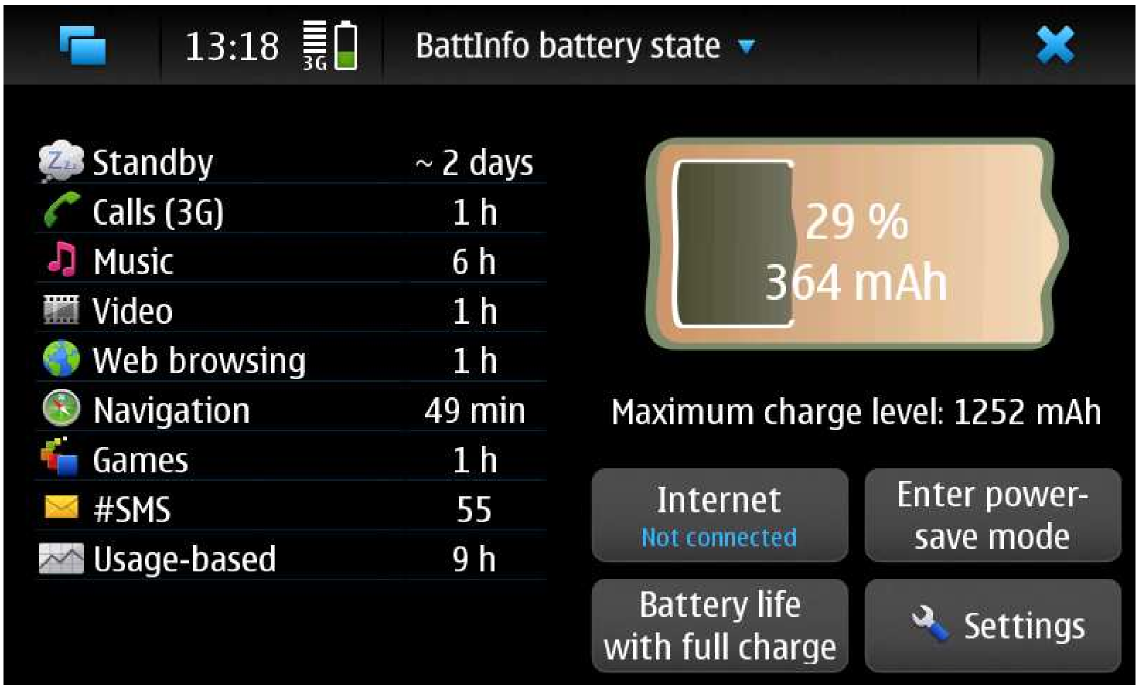
The questionnaires did not have questions about the usage of the battery-low alarm and the advanced notification provided by BattInfo, but at least the users that were interviewed did not use these functionalities. One user also wrote in the free textual feedback that:

> *"The battery life on the N900 is so bad that it has to be charged every day. This is partly because I use it for browsing as well. Because I have to charge the phone every day anyway, the alarm system on the battinfo is not important for me. It might be more important if my battery lasted longer, as it would for example with an N8 phone."*

6.3 Conclusions and changes to BattInfo

All in all, the feedback from the user test was encouraging and positive and indicates that BattInfo fulfills the requirements that were set to it in Section 5.1. BattInfo was seen as a useful application that gives valuable information to the users about the battery. Also the representation of the information was clearly understandable. The visual look of BattInfo was still clearly unfinished. Users were hoping for a ”more professional look”, especially regarding the battery state view, which was found visually unpleasing by most users.

In order to improve the user experience, the main window (Figure 14) was completely removed, so that the full battery status can be opened by one click from the desktop widget. The battery state view was modified according to the feedback. Unnecessary lines and misleading header labels were completely removed and a battery icon was added to the window to improve the visual look. The new look of the battery state view is shown in Figure 21.


13:18 3G BattInfo battery state

Standby	~ 2 days
Calls (3G)	1 h
Music	6 h
Video	1 h
Web browsing	1 h
Navigation	49 min
Games	1 h
#SMS	55
Usage-based	9 h

29 %
364 mAh

Maximum charge level: 1252 mAh

Figure 21: Battery state view after modifications.

Also the indicator for the power-save mode, shown in Figure 17, was intented to be changed. Creating an icon that would both catch the user’s attention and remind the user of the power-save mode while not being too ”striking” was difficult. Thus the power-save mode indicator was left as it is until a better solution is found. The battery-low alarm and the advanced notification were completely removed since they did not work properly in all situations.

6.4 Summary

The user test conducted on BattInfo was presented in this chapter. The testing consisted of a background questionnaire and of three questionnaires, that were sent to the users after three days, one week and after two weeks of usage. The user test aimed to find out whether BattInfo fulfills the goals and requirements set to it in Section 5.1.

The results of the testing showed that BattInfo does indeed help the users to cope with their batteries and gives them important information about the battery life. The visual look of BattInfo was generally considered boring and not professional enough. Based on the feedback, BattInfo's appearance was slightly changed into a, hopefully, more appealing direction.

7 Conclusions and future work

This work started by describing the evolution of mobile devices during the last 10-15 years, and by discussing how this evolution has affected the battery life of the devices. The devices have developed relatively fast from simple portable phones into multifunctional mobile devices. Their use cases have changed, and they are now used throughout the day for various purposes, instead of only calling and sending text messages a few times a day. The devices have gained a bigger role in our daily lives, but at the same time, the battery life of the devices has become shorter and less predictable. The numerous research in the area of reducing mobile device energy consumption was also presented.

The target device of the work, the Nokia N900 mobile computer, was studied in Chapter 4. It was found out that the device is able to report the charge level of the battery in milliampere-hours in a reliable manner. Dividing the charge level by the amount of current consumed by the device produces the battery life estimation that was searched. To be able to provide the user with for example the remaining standby, call and audio playback times, the current demand of the tasks needs to be known. Thus, energy consumption measurements were conducted on the N900 to find out the average amount of current consumed by several different tasks.

Also the possibility of estimating charging time was studied. It appeared that when charging the lithium-ion battery from empty to full, the charging rate does not remain constant, even though the device was in standby state during the charging. Furthermore, the N900 can be charged with several different chargers, each with a different charging speed, and it is not possible to identify all the different chargers. These two issues make the charging time estimation challenging, but not completely impossible.

Based on the offline-measured energy consumptions of different tasks, a battery monitoring application was developed. The application was named BattInfo, and it consists of a customizable desktop widget showing a small subset of the information that BattInfo can provide and a full screen window, showing all the battery life estimations calculated by BattInfo, such as the remaining standby time, call time, navigation time and web browsing time. A user test on BattInfo was conducted in collaboration with researchers specialized in usability and user-experience. The feedback showed that basically BattInfo fulfilled its goals, but that there was definitely room for improvements, especially in the visual look of the application. Based on the users' feedback, BattInfo was slightly modified and its appearance was changed into a less technical and more appealing direction.

The work with BattInfo is likely to continue with a second user study, that would examine the current version of BattInfo and give feedback on whether the changes that were made have improved BattInfo. The application could also be further developed. One idea is to change the application so that no offline energy consumption measurements would be needed. The application could monitor the binaries running on the device, identify them, and measure the energy consumption of various tasks online, in the same manner as BattInfo now measures the usage-based energy consumption. This way the accuracy of the battery life estimation for e.g. web browsing could improve and the application could be ported to different devices without conducting the energy consumption measurements separately on each device. The application could also provide estimates only for those tasks and application, that the user frequently performs/runs. Thus, a user who uses a navigating application would see estimated navigation time on his device, whereas a user playing games could see estimated playing times for each game she plays. Charging time estimation was not implemented in BattInfo, and that is another interesting area for future work, especially making the charging time estimation accurate enough and working for all different types of chargers that are used.

References

[1] TNS Global Market Research. *TNS user study.* Cited 1.12.2010. Url: `http://www.tns.lv/?lang=en&fullarticle=true&category=showuid&id=2288`

[2] Ofversten, J. *Mobile Internet Battery Life. Challenges for application SW development.* Forum Nokia webinar. 16.12.2009. Cited 1.12.2010. Url: `http://www.forum.nokia.com/Library/Multimedia/Webinars.xhtml`

[3] GSM Arena. *Mobile phones evolution: Features in focus.* August 2009. Cited: 1.12.2010. Url: `http://www.gsmarena.com/mobile_phones_evolution_features-review-501p2.php`

[4] Y. Neuvo. *Cellular phones as embedded systems.* In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC), 2004, pp. 32-37.

[5] Finnish Communications Regulatory Authority. *The survey on the use of telecommunications services 2009* Cited 8.12.2010. Url: `http://www.ficora.fi/attachments/suomiry/5n2kRC9zk/Tutkimusraportti_2009_Telepalveluiden_kayttotutkimus.pdf`

[6] Ravi, N., Scott, J. and Iftode, L. *Context-aware Battery Management for Mobile Phones: A Feasibility Study.* In Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications, March 2008, pp. 224-233.

[7] Kivi A., Smura T., Toyli J. (2009). *Diffusion of Mobile Handset Features in Finland.* In Proceedings of the Eighth International Conference on Mobile Business (ICMB), June 2009, pp. 209.

[8] Ray, N. and Turuk, A. *A Review on Energy Efficient MAC Protocols for Wireless LANs.* In Proceedings of the Fourth International Conference on Industrial and Information Systems (ICIIS), December 2009, pp. 137.

[9] Lei, H. and Nilson, A. *Queueing Analysis of Power management in the IEEE 802.11 Based Wireless LANs.* IEEE Transactions on Wireless Communications, April 2007, Vol. 6, No. 4, pp. 1286.

[10] Baek, S. and Choi, B. *Performance Analysis of Power Save Mode in IEEE 802.11 Infrastructure WLAN.* In Proceedings of the International Conference on Telecommunications, June 2008.

[11] Xie, Y., Luo, X. and Chang, R. *Centralized PSM: An AP-centric power saving Mode for 802.11 infrastructure networks.* IEEE SarnoffSymposium, April 2009.

[12] Zhu, F. and Niu, Z. *Priority Based Power Saving Mode in WLAN.* In Proceedings of the IEEE Global Telecommunications Conference, November-December 2008.

[13] Perrucci, G., Fitzek, F., Sasso, G., Kellerer, W. and Widmer, J. *On the impact of 2G and 3G network usage for mobile phones' battery life.* In Proceedings of the European Wireless Conference, May 2009, pp. 255.

[14] Pering, T., Agarwal, Y., Gupta, R. and Want, R. *CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces.* In Proceedings of the 4th international conference on Mobile systems, applications and services, June 2006, pp. 220-232.

[15] Chowdhury, M., Jang, Y., Ji, C., Choi, S., Jeon, H., Jee, J. and Park, C. *Interface selection for power management in UMTS/WLAN overlaying network.* In Proceedings of the 11th International Conference on Advanced Communication Technology, February 2009, Vol. 01, pp. 795-799.

[16] Balasubramanian, N., Balasubramanian A. and Venkataramani, A. *Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications.* In Proceedings of the 9th Internet measurement conference, November 2009, pp. 280-293.

[17] Nurminen, J. *Parallel connections and their effect to battery consumption of a mobile phone.* In Proceedings of the 7th IEEE Consumer Communications & Networking Conference, January 2010.

[18] Kjaergaard, M., Langdal, J. and Godsk, T. *Demonstrating EnTracked a System for Energy-Efficient Position Tracking for Mobile Devices.* Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing (September 2010), pp. 367-268.

[19] Lin, K., Kansal, A., Lymberopoulos, D. and Zhao, F. *Energy-accuracy trade-off for continuous mobile device location .* In Proceedings of the 8th international conference on Mobile systems, applications, and services, June 2010, pp. 285-298.

[20] Rosu, M., Olsen, C., Luo, L. and Narayanaswami, C. *The Power-Aware Streaming Proxy Architecture.* First International Workshop on Broadband Wireless Multimedia, October 2004.

[21] Jimeno, M., Christensen, K. and Nordman, B. *A Network Connection Proxy to Enable Hosts to Sleep and Save Energy.* In Proceedings of the IEEE International Performance, Computing and Communications Conference, December 2008, pp. 101-110.

[22] Barr, K. C. and Asanovic. K. *Energy-aware lossless data compression.* ACM Transactions on Computer Systems, August 2006, vol. 24, no. 3, pp. 250-291.

[23] Maddah, R., Sharafeddine, S. *Energy-Aware Adaptive Compression Scheme for Mobile-to-Mobile Communications.* IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, August 2008, pp.688-691.

[24] Wang, L., Manner J. *Evaluation of data compression for energy-aware communication in mobile networks.* In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, October 2009, pp. 69.

[25] Shim, H., Chang, N., Pedram, M. *A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems.* In Proceedings of the 2004 Asia and South Pacific Design Automation Conference, January 2004, pp. 819.

[26] Iyer, S., Luo, L., Mayo, R. and Ranganathan, P. *Energy-adaptive display system designs for future mobile environments.* In Proceedings of the First International Conference on Mobile Systems, Applications, and Services, May 2003, pp. 245-258.

[27] Luo, L. *Designing Energy and User Efficient Interactions with Mobile Systems.* PhD thesis. School of Computer Science, Institute for Software Research, Carnegie Mellon University, 2008. pp. 90.

[28] Kumar, K., Lu, Y.*Cloud computing for mobile users.* Computer, Issue 99. March 2010.

[29] Kemp, R., Palmer, N., Kielmann, T., Bal, H. *Cuckoo: a Computation Offloading Framework for Smartphones.* MobiCASE 2010.

[30] maemo.org. *Maemo 5 Developer Guide.* Cited 20.2.2011. Url: `http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide`.

[31] Wen, R. and Krintz, C. *Online, Battery Lifetime Prediction for Embedded and Mobile Devices.* In Workshop on Power-Aware Computer Systems (PACS), 2003.

[32] Double Click. *The 1000 most-visited sites on the web* Cited: 9.12.2010. Url: `http://www.google.com/adplanner/static/top1000/`.

[33] Pitkanen, J. *Angry Birds laajentaa reviiriaan.* Tietokone-magazine, 11.12.2010. Cited 9.2.2011. Url: `http://www.tietokone.fi/uutiset/angry_birds_laajentaa_reviiriaan`.

[34] Nokia N900 Tech Specs, Power management. Web page. Cited 29.10.2010. Url: `http://europe.nokia.com/find-products/devices/nokia-n900/specifications#hardware-pow`.

[35] World Business Council for Sustainable Development. *Reducing mobile phone no-load energy demand.* Case Study, 2008. Cited 29.10.2010. Url: `http://www.wbcsd.org/plugins/DocSearch/details.asp?type=DocDet&ObjectId=MzA0MTM`.

[36] Maemo platform. Web page. Cited 29.10.2010. Url: `http://maemo.org/intro/platform/`.

[37] *dbus-monitor reference.* Web page. Cited 29.10.2010. Url: http://dbus.freedesktop.org/doc/dbus-monitor.1.html.

[38] *HAL reference.* Cited 29.10.2010. Url: http://www.freedesktop.org/wiki/Software/hal.

[39] Blanchette, J. and Summerfield. M. *C++ GUI Programming with Qt 4. 2nd Edition.* United States, Prentice Hall, 2009.

[40] Attrakdiff, Single Evaluation. http://attrakdiff.de/en/Services/AttrakDiff-Basic/Single-Evaluation/

A Output of ”hal-device bme” system command

The system command ”hal-device bme” shows all the battery-related information that the N900 battery management entity (BME) provides. The complete output of the command is shown below.

```
udi = '/org/freedesktop/Hal/devices/bme'
info.addons = 'hald-addon-bme' (string list)
maemo.charger.type = 'none' (string)
maemo.charger.connection_status = 'disconnected' (string)
maemo.rechargeable.charging_status = 'off' (string)
maemo.rechargeable.positive_rate = false (bool)
battery.present = true (bool)
info.product = 'Battery (BME-HAL)' (string)
info.subsystem = 'unknown' (string)
battery.is_rechargeable = true (bool)
info.udi = '/org/freedesktop/Hal/devices/bme' (string)
battery.charge_level.unit = 'bars' (string)
battery.remaining_time = 10800 (0x2a30) (int)
battery.type = 'pda' (string)
battery.charge_level.percentage = 62 (0x3e) (int)
battery.charge_level.design = 8 (0x8) (int)
battery.charge_level.capacity_state = 'ok' (string)
battery.rechargeable.is_discharging = true (bool)
battery.charge_level.last_full = 1 (0x1) (int)
battery.reporting.design = 1262 (0x4ee) (int)
battery.reporting.last_full = 5 (0x5) (int)
battery.reporting.current = 789 (0x315) (int)
battery.voltage.unit = 'mV' (string)
battery.voltage.design = 4200 (0x1068) (int)
info.category = 'battery' (string)
battery.voltage.current = 3881 (0xf29) (int)
battery.remaining_time.calculate_per_time = false (bool)
info.parent = '/org/freedesktop/Hal/devices/computer' (string)
battery.charge_level.current = 6 (0x6) (int)
battery.rechargeable.is_charging = false (bool)
info.capabilities = 'battery' (string list)
battery.reporting.unit = 'mAh' (string)
```

B Debian packaging of a Qt application for Maemo

In order to distribute the application to the test users, it had to be made easily downloadable and installable through the internet. Maemo devices use the Debian package management system for software distribution. The debian package creation for a Maemo device has some specific things that need to be taken into account when packaging the application. This appendix explains how the application was packaged so that it could be installed with the Maemo Application manager and placed in the right menu on the device, and how the package was then distributed.

First the basic structure of a debian package was created by running the command ”db_createorig” inside Scratchbox (a sandbox for emulating the ARMEL architecture) in a folder containing the files of the project. After this, the files ”control” and ”rules” inside were modified accoring to the instructions given in the Maemo developers guide.

The debian package also needs a desktop file, which defines how the application will be shown in the Maemo menus. The desktop file used for the battery monitoring application is shown belos.

```
[Desktop Entry]
Name=BattInfo
Type=qt
X-Path=/opt/battinfo/battinfo
X-Multiple-Instances=false
X-home-applet-minwidth=150
X-home-applet-minheight=150
```

Finally, the actual debian package was created with the command *dpkg-buildpackage -rfakeroot* inside Scratchbox. This command creates a .deb file which is installable on the N900.

Creating a one-click installation link

The initial idea was to simply distribute the debian file to the users, but it was found out that the Maemo application manager is not able to automatically download and install the dependencies of a single debian file. Instead, when the debian package was put into a repository, the application manager was able to handle the dependencies correctly.

In order to make the application downloadable and installable by one click, a private debian repository was built. After this, a .install file was created and a link to it was sent to the test users. Clicking the link adds the repository that contained BattInfo to the Maemo Catalogue list and after that installs the application together with all its dependencies. The ”battinfo.install” file is shown below.

```
[install]
catalogues = Comnet
package = battinfo


[Comnet]
name = Aalto University Comnet
uri = http://www.netlab.tkk.fi/ kakorho2/1118172628/apt/
dist = fremantle
components = free
```