Medical Sensor Network Infrastructures

Jacob Andersen

PhD Dissertation

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS ·

Department of Computer Science
Aarhus University
Denmark

Medical Sensor Network Infrastructures

A Dissertation
Presented to the Faculty of Science
of Aarhus University
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Jacob Andersen
1st December 2009

Abstract

Throughout the health-care sector from the emergency response situation, at hospitals, and to chronic disease management, much time is spent on performing and recording the results of various measurements and tests. Often large instruments are used (e.g. ECG monitors), or a complicated procedure is required, such as taking a blood sample, mailing it to a lab, where it is analysed and the result returned by mail.

Due to the continuing size and cost reduction of electronic equipment, future medical sensors will be much smaller, cheaper and often disposable. Furthermore, integration of these sensors with the electronic health record (EHR) IT-systems will save a lot of work (and human errors), as the sensor readings will be directly recorded in the patient's records by the sensors themselves, rather than by a transcription performed by a busy clinician.

Although this development has been going on for at least a decade, most sensors are still quite big, heavy and difficult to operate, and a lot of research is revolving around minimising the instruments and making them easier to use. Several research experiments have demonstrated the utility of such sensors, but few of these experiments consider security and access control, and how this impacts the usability. A typical design involves a general service discovery architecture connecting sensors automatically with nearby displays, but does not consider how non-authorised displays or users should be prevented from accessing the sensor readings.

Other researchers have been developing power-efficient security mechanisms for sensor networks. However, most of this work ignores the special usability demands from the clinical use-scenarios: set-up must be fast, and key pre-distribution is problematic if disposable sensors are discarded after being used for only a short while.

This tension between simple use and security in a low-power clinical environment is the main theme of this dissertation. Un-secure medical equipment will never pass official certification by national health authorities, but on the other hand, experience shows that if using the equipment is difficult or time-consuming, clinicians will either reject it, or circumvent the security mechanisms, leaving the system open to attacks. It is therefore essential to ensure both that a solution is secure, and that it will actually work in practise. In order to demonstrate that this is possible, a prototype platform

is constructed as part of the work presented by this dissertation. This platform offers experiments with various user interfaces and key establishment methods on real hardware and with realistic power and resource constraints.

The main scientific contributions of this dissertation are:

- An experimental prototype platform consisting of sensor hardware and a software library exposing pluggable programming interfaces for sensor user interfaces and key-establishment protocols. This library was designed to offer easy usability testing and energy efficiency measurements of different concrete user interface and key-establishment protocol implementations.

- A medical sensor user interface proposal, offering fast and easy deployment and configuration of sensors—including inspecting the current state of the deployment. The user interface is based on minimal hardware requirements (a single push button and a group of light-emitting diodes) and an early implementation of this user interface was tested with a group of clinicians, who found it easy to use and understand.

- A key-establishment protocol offering fully secure sensor set-up using elliptic curve based public-key cryptography, including an evaluation of this protocol's time and energy consumption, and its impact on the battery lifetime. The protocol is fast enough to be useful in a hectic environment and requires only a small amount of energy, making it useful even with watch-size battery cells.

- A tool to precisely measure and compare energy costs during sensor software development. This tool is designed for easy measurements of particular parts of the software—e.g. the user interface or the key-establishment protocol. Along with the development of this tool, several valuable lessons pertaining to low-power software development in general was attained, such as the importance of regular energy evaluations of an application throughout its development (in contrast to a single evaluation of the final application).

Contents

Part I

Overview

Chapter 1

Introduction

At hospitals and throughout the health sector in general, lots of medical sensors like thermometers, blood pressure meters, electrocardiography (ECG) monitors, and pulse-oximeters are used to monitor the condition of patients. Some measurements, such as temperature, will be taken periodically, while others, such as ECG monitoring a patient with a heart condition, must be done continually. A significant amount of the nurses' time at many hospital wards is spent on measuring and recording temperature, blood-pressure and other values for the patients' medical records in order to monitor progress and discover relapse. Often the nurses have to wake up patients who are sleeping in order to complete this task—especially early in the morning when they are collecting up-to-date values for the morning conferences. As a lot of work is involved in this medical record-keeping, and since cables obstruct the procedure of moving patients, and tie patients in need of exercise to their beds, clinicians are requesting new wireless sensor technology which can ease this task of record-keeping.

While recent years of IT and electronics development have brought about a plethora of small wireless gadgets, only few wireless sensors are used in hospitals today. The goal of this dissertation has been to investigate some of the fundamental problems preventing the adoption of this technology from happening and to propose appropriate solutions.

1.1 Motivation

Countless research projects have explored potential sensor uses, developed new sensors, or found new ways to collect and use sensor data for medical purposes. Examples of these will be presented later. By combining the visions from a few of these projects [39,17,55,69,62,43,29,58], the following slightly futuristic scenario can be imagined:

Emergency call—a man was hit by a cardiac arrest. As the ambulance arrives, resuscitation is performed and a patch-sized ECG sensor is attached to the patient's chest. At the hospital a cardiologist review the live sensor readings along with the patient's medical history at his office, and over the radio, he prescribes the necessary heart medicine, which the paramedic administers on the way to the hospital. During his stay at the hospital, the patient will be taken to different wards for examination, perhaps an operation and subsequent monitoring. Whenever the patient arrives at a new location, the sensor readings automatically becomes available on a nearby monitor. Furthermore, the cardiologist and other clinicians involved in the treatment can get remote access to the current and historic sensor readings at any time. When the patient is released from the hospital, he will continue to wear the ECG sensor, which will then notify both the cardiologist and the patient if irregular activity is detected. If another cardiac arrest is detected, the sensor can even make an autonomous emergency call, perhaps using GPS positioning to report its precise location.

Of course the patient will probably not wear the same sensor from the ambulance until he is released from the hospital. Hygiene and medical reasons may require sensor replacements; still, the key idea is, that from a technical point of view, there should be no need for sensor replacements—except, perhaps, due to battery lifetime.

The terms *Pervasive Computing*, *Ubiquitous Computing*, and *The Disappearing Computer* are often used to describe the current evolution of information technology. From a past where many users shared a single computer, through a time with almost one-to-one relationship between users and computers, we now have many computers (or IT-gadgets) per person. Continuing this trend, in a very near future, each person is envisioned to possess an un-manageable number of devices. This is why one of the main challenges is to make this new technology auto-configurable, so that new devices discover and configure themselves according to the environment they are brought into, rather than depending on the user to manually configure them, as we have been used to for decades. The IT will simply "disappear" into the buildings, appliances and clothes, and automatically pop up and do its job whenever the user needs it; for instance displays may be embedded in refrigerator-doors and bathroom mirrors, so one can write a grocery list at the refrigerator or check the daily schedule while brushing one's teeth.

Similarly, in the health-care sector imagine general-purpose displays embedded for instance in walls, hospital beds, stretchers, or in the arm of the ambulance personnel's jacket sleeves (as envisioned in [83]). If a sensor automatically connect wirelessly to nearby displays the need for a built-in display vanishes, and the sensor can be made much smaller. The sizes of sensors are

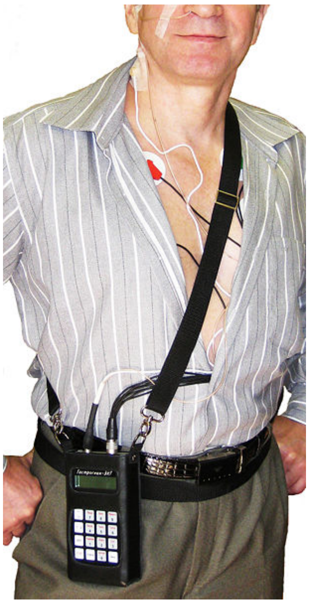


Figure 1.1: A wearable sensor, “Gastroscan-ECG”, recording ECG and pH in the stomach and at various points in the oesophagus for up to 24 hours. The sensor is used to determine whether a patient’s chest pains are caused by heartburn, i.e gastric acids in the oesophagus, or real heart problems. [Picture from the Wikimedia Commons collection]

actually in most cases not dictated by the sensors themselves; rather, it is typically a consequence of the need for a display, where the measurements of the sensors can be read, a battery, and perhaps a keypad to configure the sensor. For instance, a wearable ECG sensor as the one in figure 1.1 could be reduced to a simple patch, like the one found in [32, 39], if the keypad and display were removed, and a smaller battery could be used.

Consider as another example a typical digital mouth or rectal thermometer: the sensor itself and the electronics could be fitted into a volume about the size of a grain of sand. However, the display, battery, and the need of a handle lead to the size and shape that we are familiar with today. A pill-sized thermometer was in fact developed by NASA during the 1980s for monitoring the temperature of astronauts [51]. This thermometer pill is swallowed and will transmit the body temperature to an external receiver for a couple of days—until it leaves the body naturally. Today, this pill is produced by HQ Inc [47], and is still quite expensive. It is believed to have saved at least one professional football player’s life already, as it is used by U.S. national football league clubs for monitoring their players in order to prevent the dreaded heat strokes.

Despite commercial availability for more than a decade and the obvious advantages of eliminating cables and minimising sensors, the new wireless sensors have not yet driven old sensors out of the market, as one might have expected, due to several reasons, including the bulkiness of the devices (due to large batteries) or the increased complexity of use. In some contexts—e.g. accidents [54]—it is also crucial that sensors do not impose an extra work overhead.

The physical connection (being a common casing or a connecting cable) between a sensor and its display, has the following important advantages over the wireless (no physical connection) situation:

Power Transmitting a sensor reading through a cable requires virtually no energy, while a wireless transmission requires a significant amount of energy at both the transmitter and the receiver end. Furthermore, if the sensor and display are separate units connected by a cable (as e.g. a pulse-oximeter) a power source at the display-end will typically supply the sensor through the cable; in the wireless case, the sensor will need its own power source.

Security A cable connection has some inherent properties protecting the communication against adversarial third-parties (who have no access to the physical connection). These properties must be explicitly provided by a wireless connection:

- **Privacy** A third-party cannot eavesdrop. The (unencrypted) information generated by the sensor can only be accessed by the display.
- **Authenticity** The readings received by the display was sent from the sensor it is connected to; they cannot come from a different sensor, and a third-party cannot manipulate or counterfeit the data.
- **Dependability** The communication is reliable; a third-party cannot break or disturb the communication.

Tangible user interface The cable is a tangible physical connection between the sensor and its display, which is easy to inspect, manipulate, and understand. Imagine two identical sensor-display pairs in the same room (which is not uncommon at a hospital!), and replace the cables by wireless connections. How can the user be sure which sensor's data is displayed on which display? How can the user inspect the connection, and repair it, when something goes wrong?

The failing success for current state-of-the-art wireless medical sensors can in most cases be attributed to the lack of proper wireless solutions to one or more of the above-mentioned points; and while this is a huge problem, only few research projects have addressed this issue. Numerous research

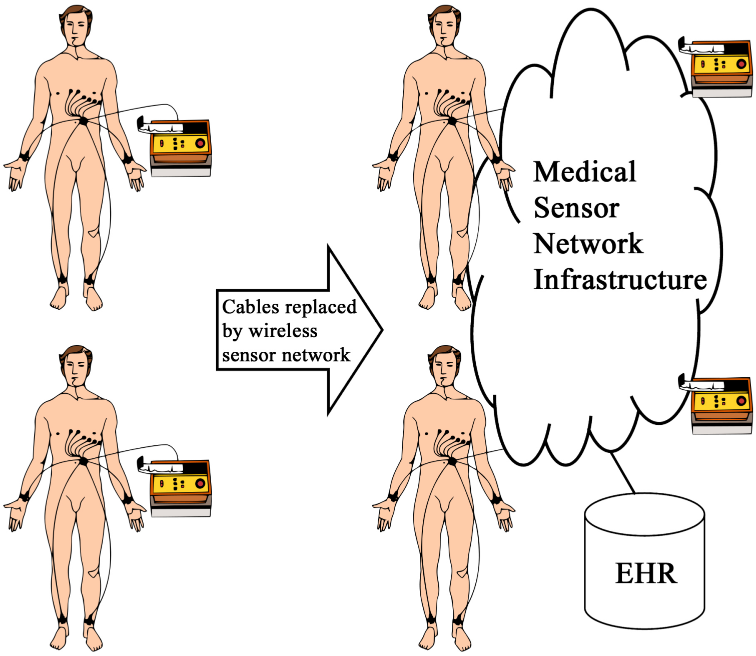

Cables replaced by wireless sensor network
Medical Sensor Network Infrastructure
EHR

Figure 1.2: Physical (cable) connections between sensors and their respective displays will be replaced by a *Medical Sensor Network Infrastructure.* In addition to delivering sensor readings to local displays, the infrastructure may deliver the readings to authorised personnel at remote locations or to the patient's Electronic Health Record (EHR).

projects succeed in demonstrating how medical sensors can be minimised—and how this is going to be very useful in the future health-care sector; however, they arrive at their conclusions by either ignoring some or all of these points, or by assuming, that these can be addressed later with only a small penalty.

Since the above-mentioned points are properties of the cable (physical connection) between the sensor and the display, when a wireless infrastructure is used in place of a cable, the replacement infrastructure has to offer the same properties. This *Medical Sensor Network Infrastructure* (figure 1.2) is the object of this dissertation. A main goal is to demonstrate how such an infrastructure may be constructed in a practical way—with only a small penalty.

1.2 Research Questions and Method

Throughout many research projects, the sensor devices are imagined to be very small—perhaps implants—running on battery power from a small watch-type battery or powered by a small generator which transforms mechanical, thermal or chemical energy from the sensor's environment into electrical power [92, 101, 116, 86]. In most of these projects, sensors are expected to last for days, weeks, months or even years, without replacing or recharging batteries, often as part of a more permanent monitoring programme. The projects always conclude that this kind of sensor technology is very promising and useful, as for instance the amount of time spent by nurses "logging sensor readings" is reduced or because doctors can access the sensor readings and histories from his office. Meanwhile, state-of-the-art sensors have the size and weight of cell phones due to large batteries, and still they only last for a few days [111, 80, 37], or in the best case perhaps a few weeks.

Now, ever since Alessandro Volta invented the battery more than 200 years ago, progress has been slow. Indeed, new and better battery technologies have been invented and are still developed, but the progress is not remotely like the well-known Moore's law for electronics, and no huge breakthrough should be expected soon. Hence, for all practical purposes, we can assume, that the amount of energy stored per mass unit or volume unit, is almost constant. If the devices are to be made smaller, proportionally less energy will be available. IT devices have rarely been designed to conserve power. Cell phones may be considered an exception to this rule, but still their batteries are much bigger than batteries that go into pacemakers or eatable thermometer pills, and still, the latter kind of equipment is expected to last longer, so power consumption must be orders of magnitudes lower. Traditional software architectures and network protocols were also not designed to conserve power (i.e. minimise the number of CPU cycles and the amount of memory spent).

In addition to this, medical sensors should meet the demands of the clinical world, they must be easy and safe to use and provide different kinds of foolproof functionality to prevent errors. They must also be secure and not allow unauthorised access, and all this while not wasting unnecessary clock cycles in order to obtain acceptable size and lifetime.

Now, the *serious* challenge emerge when the three issues of usability, security, and resource constraints are combined. Figure 1.3 shows the complexity of the problem space: The three issues, the trade-off relations between them, and examples of external factors that influence these issues. As illustrated on the figure, any issue-pair will be in conflict: Security is problematic when resources are constrained, especially if public-key cryptography is used, as this requires lots of memory and CPU cycles. Usability and security do not fit well together either. User names and passwords or biometric logins (such as fingerprint readers) are hard to imagine on small platforms like e.g. an ECG sensor embedded in a patch or a thermometer in a pill. And how exactly are keys distributed and authenticity guaranteed if disposable sensors are bought in large quantities for one-time use? Or what about handing over sensor access automatically when a patient is moved

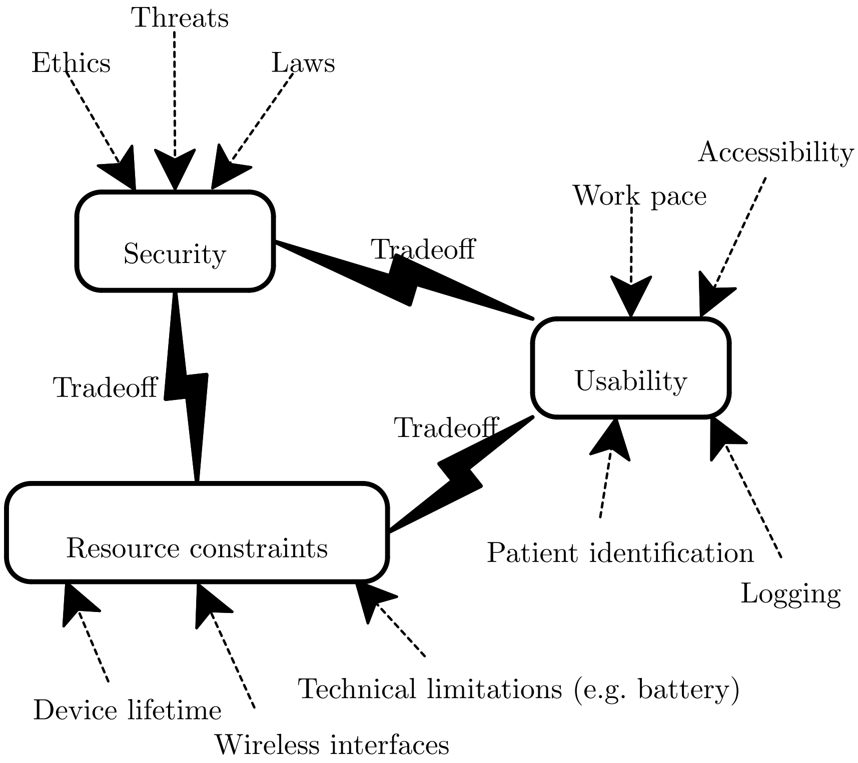


Figure 1.3: The Problem Space.

from one hospital to another? Finally, the key to low power consumption is to keep the sensor in the deepest possible sleeping mode as often and as long as possible. This, of course, does not combine well with usability. For instance, how do one ensure a sufficient responsiveness from the sensor? If the sensor has turned off its radio receiver, it is not capable of receiving a wake-up call. Therefore, user interaction patterns, as well as network protocols, must be carefully designed to allow the sensor to go to sleep most of the time.

As observed in the previous section, commercially available wireless sensors generally have good security properties (if not, they could not be certified for the market), but compared to the visions of the numerous research projects, the usability properties are lagging far behind. Also, the devices are usually bulky due to big batteries and still have a relatively short battery lifespan. Related research projects in general, deal with only one or two of these issues, typically leading to solutions where it is hard to imagine how solutions to the remaining third issue can be included. For instance, the CodeBlue project [67] (which will be outlined in section 2.3) start out considering all of these three issues, but end up with a prototype [98], where the user interface depends on public service discovery and no privacy whatsoever—

hardly a basis for an implementation of strict privacy, access control and logging.

1.2.1 Research Questions

The overall scientific aim of this PhD project has been to explore the limitations of the triangle of figure 1.3 in an attempt to demonstrate the feasibility of tiny (much smaller than state-of-the-art) sensors which are fully secure and have functional user interfaces. The research questions all revolve around the question of how the deployment and configuration of medical sensors in health-care settings (hospitals, emergency response, and homes) may be supported. More specifically, this dissertation will address the following questions:

Question 1:
How to construct appropriate user interfaces for even the smallest resource constrained sensor, usable at hospitals, emergencies and in the home?

Question 2:
How to design a secure wireless infrastructure obeying legal obligations and allowing the use of sensors straight from the factory (i.e. with no a priori knowledge about the environment they will be placed in), being easy to use and taking the technical restrictions into account?

Question 3:
How can an experimental research platform be designed to facilitate flexible and efficient experimentation with the issues, dependencies and limitations of the triangle in figure 1.3?

Question 4:
Will it be feasible to let common principles guide the designs across all three contexts (hospital, emergency response, homes—cf. section 2.2) such that a sensor can be used everywhere, and maybe even roam freely between the infrastructures[1]with little or no human intervention (so that e.g. sensors can be replaced when it is convenient or when dictated by hygienics, and not when dictated by technology limitations)?

Note that these questions can probably never be answered independently, as the resource shortage is so important in the world of sensor networks.

[1] Throughout the related work, hospital sensors are used only at hospitals, emergency response sensors only in trauma care and so on. Today, "roaming" is more or less unthinkable, but perhaps a common infrastructure would create new possibilities.

Consequently, every single detail can have a major impact on the entire system.

In the early stage of this project, an additional question was addressed: how to integrate the medical sensors in hospital and emergency IT infrastructures. This work was suspended, but will be revisited in the future work section 7.2.

1.2.2 Research Method and Approach

As most of the research questions revolve around exploring the limitations of tiny devices, the primary method has been that of experimental computer science, building proof-of-concept prototypes in order to demonstrate the ideas in practise.

The problem of replacing the cable between a sensor and its display by a wireless connection may seem small at a first glance. However, as it is affected by a complex interplay of problems, the only convincing way to prove that the problem has been solved would be to build a prototype solution. This solution should work reasonably well, so it can be tested against all requirements in an environment resembling the real-world complexity.

The hardware devices used for prototyping have not entirely been standard off-the-shelf items. Actually, a few hardware modules had to be developed to accommodate the needs along with the software drivers necessary to use these modules. As the hairy details of this rather extensive work will probably not make the most thrilling literature, they have been pushed to the back of this dissertation, and are available in the appendices.

To guide the prototype development—the user interface in particular—ideas and inspiration was collected from other projects (as reported in section 2.2) and a number of conversations with different clinicians. Users have *not* been directly involved in the development process, as the case is in the tradition of participatory design; instead, different alternative implementations have been—and will be—constructed which the users will then be asked to evaluate and comment on.

1.2.3 Contributions

The main scientific contributions of this dissertation include:

- An experimental prototype platform consisting of sensor hardware and a software library exposing pluggable programming interfaces for sensor user interfaces and key-establishment protocols. This library was designed to offer easy usability testing and energy efficiency measurements of different concrete user interface and key-establishment protocol implementations. The platform addresses Question 3 regarding the design of an experimental platform, and the architecture of the library was designed with no assumptions about the operating environment (hospital, emergency scene, or home), as a response to Question 4 about common principles for all three contexts. To some degree, the platform also addresses Question 1 and Question 2, as some basic security mechanisms and assumptions on the user interface was built into the platform.

- A medical sensor user interface proposal, offering fast and easy deployment and configuration of sensors—including inspecting the current state of the deployment. The user interface primarily addresses Question 1 and Question 4, and is based on minimal hardware requirements (a single push button and a group of light-emitting diodes) and an early implementation of this user interface was tested with a group of clinicians, who found it easy to use and understand.
- A key-establishment protocol offering fully secure sensor set-up using elliptic curve based public-key cryptography, including an evaluation of this protocol's time and energy consumption, and its impact on the battery lifetime. The protocol is fast enough to be useful in a hectic environment and requires only a small amount of energy, making it useful even with watch-size battery cells. The protocol primarily addresses Question 2, but like the platform library described above, it was designed with no assumptions on the operating environment, and it even includes a mechanism to facilitate roaming as requested by Question 4.
- A tool to precisely measure and compare energy costs during sensor software development. This tool is designed for easy measurements of particular parts of the software—e.g. the user interface or the key-establishment protocol. Along with the development of this tool, several valuable lessons pertaining to low-power software development in general was attained, such as the importance of regular energy evaluations of an application throughout its development (in contrast to a single evaluation of the final application). This tool addresses Question 3 about facilitating efficient experimentation.

1.3 Dissertation Structure

Following the general introduction and overview of the dissertation presented here, chapter 2 offers an introduction to the general area of sensor networks, and to the special demands found in the health-care domain. The reader will be guided through some of the state-of-the-art equipment and technologies used today, as well as a presentation of some of the related research projects in this area. The aim of the background chapter is to provide the reader with a deeper understanding of the problems pertaining to the separation of sensors and displays in the clinical context.

The chapters following chapter 2, deals with different themes. In chapter 3, the problem of creating a suitable user interface for small medical sensors is investigated, and a proposed solution is presented. Chapter 4 presents the platform which was constructed to carry out the experiments. The theme of chapter 5 is security and protocols for secure sensor configuration, and chapter 6 provides some general lessons learnt from working with low-power application development.

Part I ends in chapter 7 with directions for future work and overall conclusions.

In part II, the papers listed below are reproduced. At the end of the dissertation, a number of appendices present some of the technical details and documentation about the prototype hardware and software modules.

1.3.1 Papers

The papers reproduced in part II, cover work and contributions related to the topic "Medical Sensor Network Infrastructures", and they are presented in chronological order according to the list below.

[1] J. Andersen and J. E. Bardram, "BLIG: A new approach for sensor identification, grouping, and authorisation in body sensor networks," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, ser. IFMBE Proceedings, S. Leonhardt, T. Falck, and P. Mähönen, Eds., vol. 13. Berlin: Springer, Mar 2007, pp. 223–228.

[2] J. Andersen, B. Lo, and G.-Z. Yang, "Experimental platform for usability testing of secure medical sensor network protocols," in *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2008) in conjunction with the 5th International Summer School and Symposium on Medical Devices and Biosensors (ISSS-MDBS 2008).* IEEE, Jun 2008, pp. 179–182.

[3] J. Andersen and M. T. Hansen, "Energy bucket: A tool for power profiling and debugging of sensor nodes," in *The 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM 2009).* IEEE, June 2009, pp. 132–138.

[4] J. Andersen, "Secure group formation protocol for a medical sensor network prototype," in *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2009).* IEEE, December 2009, IN PRINT.

[5] J. Andersen, "Towards both Usable and Secure Protocols for Medical Sensor Networks," in *International Conference on Wearable and Implantable Body Sensor Networks (BSN 2010)*, 2010, PLANNED FOR SUBMISSION.

The contributions of these papers are summarised and put into context of related work in part I of this dissertation.

1.4 Acknowledgements

First of all to all my supervisors, Lars M. Kristensen, Jakob E. Bardram, Henrik B. Christensen and Jakob I. Pagter, and to my colleagues at the Centre for Pervasive Healthcare, a great thanks for all your time and the inspiring discussions we have had, which helped me shape this project into its current form. Also, thanks to my office "room-mate" Morten T. Hansen, with whom I have enjoyed many discussions—many of which revolved around the subjects of energy efficiency and security in Wireless Sensor Networks.

Thanks also to the Pervasive Sensing group at the Imperial College London, headed by professor Guang-Zhong Yang, for hosting me for a 5 months visit. A special thanks to Dr. Benny P. Lo, who was a great help to me during hardware design and production.

Last but definitely not least, a huge thanks to my lovely wife, Mey Ien, who (being a nurse) was my original inspiration for this project, and without whom I would never have finished it. Your support is everything to me!

Jacob Andersen,
Århus, 1st December 2009.

Chapter 2

Background

The problem of separating sensors and displays may at first seem small and trivial, but as hinted to in the introduction, it is actually challenging and no existing solutions come close to being ideal. The difficulty is caused by the complexity of the environment with many conflicting requirements, which must be thoroughly understood before a solution can be developed.

In order to equip the reader with a deeper understanding of the problems treated in this dissertation, this chapter presents selected background information. This is done first from a technical perspective in section 2.1, then in section 2.2 from the clinical (users') perspective, and finally through a survey of the current state-of-the-art equipment and recent research results in section 2.3.

2.1 Sensor Networks, Medical Sensors, and Applications

At hospitals and throughout the health-care sector, some of the most common wired sensors are ECG monitors and pulse-oximeters. These are used as examples throughout this dissertation, and are introduced below for the sake of the reader, who may not be fully familiar with such devices.

ECG (electrocardiography) monitors attach up to 10 electrodes to the body, recording the electrical activity in different sections of the heart. A wire from each electrode connects to an input on a monitor or perhaps a printer, typically located on a table beside the bed.

Pulse-oximeters (a.k.a. saturation monitors) measures the oxygen saturation in the blood (SpO_2) and the pulse by analysing the absorption of light from a couple of Light Emitting Diodes (LEDs) after it has passed through (or been reflected from) the skin. The most common place to perform this measurement is through a fingertip using a small clip attached to the finger, restricting the patient's use of that hand since the clip is connected to a bed-side monitor with a cable.

2.1.1 Why use Wireless Sensors?

As argued in the introduction, physically separating sensors from their displays is not as straightforward as one might have expected, so the question of why this is necessary is reasonable to raise.

First of all, patients need to get out of bed as soon as possible and as much as possible, since it is harmful to stay in bed for more than a few days, as the body will start to deteriorate [56]. Therefore, when patients are recovering from a health condition, they often need to be mobilised as early as possible, implying that cables restraining them to their beds during the recovery are problematic, to say the least.

One of the most time-consuming problems caused by cables, is that when patients are moved, all wires must be unhooked during transport and then reattached when the patient reaches his destination. Sometimes (for instance when ECG monitoring the patient) a temporary transport monitor is attached during transport, with the result that the procedure of detaching/reattaching wires must be repeated twice each time the patient is moved. A closer study of the intra-hospital transport of ECG monitored patients was made in [62], who also proposed using wireless ECG sensors to save time and trouble.

A study by the Lewin Group performed in October 2002, cited by the wireless ECG vendor LifeSync [61], states that nurses spend 40 minutes on each ECG monitored patient every day dealing with the cables and cable-related problems, such as alarms caused by unstable connections or cables falling off. The study claims that an average of 3 work-hours can be saved per patient (over the entire admission) if the wireless alternative (produced by LifeSync) is used instead. Unfortunately, the original publication from the Lewin Group does not appear to be available at their website, so only a few citations are available. Although the exact premises of this study is unclear, the conclusion that a significant amount of time can be saved with wireless ECG sensors is indisputable.

In addition to this, a number of research projects have also shown several health and economical benefits from around-the-clock wireless monitoring of patients suffering from chronic illness [43] or patients going through rehabilitation [52].

2.1.2 Medical Sensor Networks

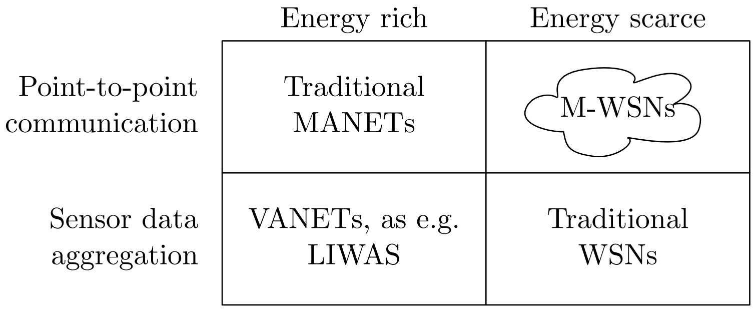
Wireless networks of mostly battery operated mobile devices has been the subject of much research during the past couple of decades. As explained in section 1.1, auto-configuration is of the essence due to the potential number of devices—and also due to the mobility, as devices may appear, move around, and disappear at any time. Many variants of *Mobile Ad-Hoc Networks* (MANETs) of typically battery powered devices have been proposed with different purposes and using different wireless technologies. Unfortunately, only few have made it to actual products—an example of a product, which offers auto-configured networking with neighbouring devices, is the XO laptop from the One-Laptop-Per-Child (OLPC) programme [82].

The MANET research typically focuses on establishing traditional network communication between mobile devices, having decent amounts of computational power and energy at their disposal, such as laptops and cell phones. Over the last decade, however, new applications appeared where the devices would have to run on battery power for very long periods and still be able to participate in ad-hoc wireless networks. This gave birth to a new class of networks, known as *Wireless Sensor Networks* (WSNs). Sensor networks can be defined for instance like [40] does: *"a set of small autonomous systems, called sensor nodes which cooperate to solve at least one common application. Their tasks include some kind of perception of physical parameters."* The purpose of a typical WSN is to sense some phenomenon, and each device (or *node*) operates for weeks, months, maybe even years on a single small battery—thus the power resources in WSNs may be several orders of magnitudes less than those of typical MANETs.

Similar to many other technologies, WSNs can be traced back to military research, and examples of recent military research projects using this technology include WSNs for fast and accurate sniper localisation [71], and field surveillance [6]—a technology which has the potential to replace the unpopular use of minefields. In both of these cases, a large number of small and cheap nodes are scattered over the target area using an air plane (perhaps remote controlled), and the nodes will form a network and collaborate on sensing the common phenomenon (the position of the sniper, or unauthorised intrusion).

WSNs have turned out to be a strong technology for many different applications, that used to be very difficult to address. Examples include: environmental monitoring like the Glacsweb [73], investigating the internal dynamics of a Norwegian glacier; early warning against forest fire in the FireBug [34] project by equipping an entire forest with a fire detector system; and the Great Duck Island project [66], non-intrusively monitoring birds' nesting habitats.

When browsing through WSN research projects like the examples presented above, one does not have to spend many minutes before it becomes evident, that the conservation of energy is the most problematic issue. Clearly, this issue is shared with wireless medical sensors, as these must be as small as possible (tiny sensors can only have tiny batteries), and we want the medical sensors to operate for as long as possible—in some cases permanently. Other common properties of WSNs and wireless medical sensors include serious memory and computing power limitations, as this is a natural consequence of the desire to minimise size as well as cost—disposable sensors should not be expensive. Due to these similarities between wireless medical



	Energy rich	Energy scarce
Point-to-point communication	Traditional MANETs	M-WSNs
Sensor data aggregation	VANETs, as e.g. LIWAS	Traditional WSNs

Figure 2.1: Sensor Network Taxonomy.

sensors and WSNs and the great body of research in the WSN community, most wireless medical sensor research is based on the WSN foundation, and this dissertation choose to follow this tradition, hence the term *medical sensor network* (M-WSN) in the title.

The definition of “sensor networks” as stated earlier involves nodes collaborating on some common sensing task. On the other hand, in a medical sensor network, each sensor has a unique task of measuring some phenomenon of a particular person—other sensors measure either a different phenomenon on the same person, or a similar phenomenon on a different person. In medical sensor networks, sensors may not collaborate and share sensor readings—in fact, if the data from a sensor on one person would find its way to a sensor on a different person, it would be considered a security breach. It would seem that the term “sensor network” is ambiguous, and the problem is that there are two completely orthogonal problems coexisting under the common designation of “sensor networks”. The first problem considers size, resources and energy limitations of nodes, while the second problem deals with the perception of a (single) physical phenomenon. In fact, a taxonomy of wireless networks could be made using two axes, having energy availability as one dimension and the issue of, whether “sensing a common physical phenomenon” takes place or not as the other dimension. This taxonomy is illustrated in figure 2.1, and discussed in detail below.

Traditional MANETs consist of nodes with plenty of power in networks typically using TCP/IP and point-to-point communication. Research issues include how to do auto-configuration and routing in networks with nodes that are highly mobile. Traditional WSNs (as presented above) will be found in the energy scarce and ‘sensor data aggregation’ category. Here aggregation of the recorded data about the common phenomenon is performed by the collaborating sensor nodes in order to minimise the total amount of data transmission—and thereby also minimising the power consumption. Multi-hop point-to-point communication is rarely used, and complex network stacks with advanced packet routing and transport layers (like for instance TCP/IP) are out of the question.

Two categories remain. Energy rich networks that are dedicated to sensing a common phenomenon and uses in-network data aggregation can be found for instance in Vehicular Ad-Hoc Networks (VANETs), which is ad-hoc networking among vehicles (and possibly road side equipment, such as roadsigns). An example of this can be found in the LIWAS project [64, 20], where the common phenomenon is the state of the road (dry, wet, icy) and the sensors are embedded in cars and trucks. As plenty of power is available, the problems in this class of networks are very different from those of the traditional WSNs. For instance, the LIWAS prototype nodes communicate using WiFi and the TCP/IP protocol stack.

The final category in this taxonomy is the energy scarce networks that are not sensing a common physical phenomenon. Here most medical sensor networks should be placed. M-WSNs do not measure a common physical phenomenon. Rather, each sensor measures its own phenomenon, and different nodes in the network may be doing their measurements on different patients. No aggregation of data takes place inside the network. Instead, all data will be sent to a display or to some database server—typically using point-to-point communication. The only exception from this rule is if some personal server node collects all sensor readings from a single patient for pre-processing before forwarding it to the final recipient. However, this personal node would only collect readings from sensors on a single patient and never from multiple patients.

Despite these differences between M-WSNs and traditional WSNs, they share a very large subset of the problems identified in WSNs, and thus a lot of the research results from the WSN domain can be used as inspiration or perhaps even to some extent be partially applied to M-WSNs.

Typical problems addressed by WSN research, but also relevant for M-WSNs include:

Energy consumption This is of course the main issue, and an issue with which all other problems are related.

Latency At times, WSN data packets must travel a long distance, which implies that WSN nodes must participate in relaying data from other nodes.[1] When radio transceivers are kept off most of the time, due to power conservation, how is it ensured that a message arrives at its final destination in only a short amount of time? In M-WSNs this can be very relevant, for instance if the message is a life critical alarm.

Accuracy What is the precision of measurements? Again this may involve a trade-off against power consumption.

[1] When transmitting a data packet over a long distance, it would require more energy and perhaps larger radios to transmit the packet in a single hop, than the total amount of energy spent by all nodes involved in a chain of shorter hops.

Fault tolerance Nodes have a tendency to die when the battery runs out. This and other faults (e.g. signal interference) must be addressed.

Scalability We can have a huge number of nodes (the WSN trend is also known by the name "Smart Dust"). In M-WSN we need to ensure, that the network will continue to work even if it grows beyond imagination (hospitals or emergencies sites with many patients in a small area, for instance after a major incident).

Security This includes ensuring the privacy and authenticity using very small processors and memory. An adversary attempting to break in may use powerful computers, so in M-WSNs it is obviously necessary with strong security—in fact, this is required by law, as medical data are handled.

Mobility of the phenomenon, the sensor, or the observer (the recipient of the sensor data). For M-WSNs the phenomenon will in most cases not move in relation to the sensor. However, as the sensor follows the patient, it can be necessary to roam the sensor between different networks when it moves in relation to the observer. Also, in M-WSNs the observer may sometimes move around (e.g. a portable display) or change over time.

2.1.3 Wireless Technologies

A wide selection of standards exist for radio communication between wireless sensor nodes. One of the most popular WSN choices is the IEEE 802.15.4 standard presented in the following section, but a number of alternatives exists, including:

WiFi Offers plenty of bandwidth, but at a high power-cost. WiFi was designed to offer wireless LAN connectivity within a radius of 50–100 meters. Due to the high power cost, WiFi is rarely used for WSNs.

GSM/GPRS/UMTS The cell phone standards. These standards require a very high amount of power. However, they can be relevant when connectivity is important but only necessary for short time intervals.

Bluetooth Was designed by Ericsson to replace the cable between cell phones and headsets or laptops. Offers lower data rates (no more than 3 Mbit/s) at a lower power than WiFi. This is a *cable replacement protocol*, and therefore communication is always point-to-point, except during service discovery. [59, 9] both report that Bluetooth is not suited for WSN purposes due to the relatively high power demands. The range is about 10 meters, although this can be extended to 100 meters. A limited number of devices can communicate simultaneously in any given area, however, so extending the range will significantly reduce the number of possible concurrent conversations.

The key problem with all the above standards is, that they were not developed for the low-power applications of WSNs. Therefore, constructing equipment which uses any of these wireless interfaces result in bulky cell-phone-size devices due to big batteries. Furthermore, for Bluetooth devices the receiver must be moved along with the sensor as the range is limited to a personal area (a radius of about 10 meters). Bluetooth will also form a distinct network around each pair of sensor and display, and these networks will compete for the wireless bandwidth in an un-coordinated fashion (because all networks are autonomous and inter-network co-operation is not possible). Therefore, only a small number of Bluetooth networks (sensor-display pairs) can coexist in the same area at the same time without heavy interference which could paralyse the equipment.

IEEE 802.15.4 and ZigBee

The IEEE 802.15.4 standard (which constitute the lower layers of the ZigBee protocol stack) was designed for sensor networks offering ultra-low power consumptions with a data rate of 250 kbits/s—but since packets are small and transmission can only take place a small fraction of the time, the maximum data rate experienced will be significantly less than this number. The low data rate could seem to be a problem for medical sensors, however [97] reports that typical medical sensors produce such small amounts of data, that plenty of bandwidth remains available. For instance a 3-lead ECG can settle for 600 bits/s, and most medical sensors would produce even less data than this. This implies that hundreds of sensors can be expected to easily coexist in an 802.15.4 network.

The key idea behind ZigBee [117] is that it should replace current control devices, such as remote controls for consumer electronics, light and ventilation controls etc. If this vision is fulfilled, ZigBee will in time be as pervasive as WiFi is today—perhaps even more—and if all homes and hospitals offer ZigBee infrastructures with gateways to the Internet, this could become the common wireless standard for medical sensors in the future.

It would be fair to assume that sensors, which produce larger amounts of data, also require more power, and therefore already have large batteries. Therefore, such sensors should be capable of using WiFi instead for all heavy data transfers.

2.2 Use Contexts

In order to understand the concrete needs and demands of the clinical settings, for which the wireless sensor technology must be designed, a closer study of the contexts, in which the technology would be used, is appropriate. Wireless medical sensors are applicable in, at least, the following three contexts:

Hospitals Operating theatres, intensive care units, and patient monitoring at any hospital ward and during patient transport. Primary users will be nurses and hospital physicians.

Home care Monitoring elderly citizens, enabling them to stay in their own homes rather than nursing homes. Monitoring and supporting people with chronic conditions and patients going through rehabilitation. Primary users will be general practitioners, visiting nurses, and physiotherapists. Secondary users include the elderly citizens or patients themselves, family members, and home care workers.

Emergency response From everyday traffic accidents or heart attacks to large-scale incidents. Primary users will be ambulance personnel, paramedics, doctors and fire fighters.

Other contexts may also be relevant, e.g. sports (monitoring athlete performance during practice), and fire-fighting (monitoring the well-being and temperature of fire-fighters). The focus of this dissertation, however, is on the three contexts listed above, with the main focus at the emergency context, as this is the most demanding of the three.

During the initial phase of the work presented in this dissertation, the author participated in a number of different research projects involving medical sensing in all of these three contexts, in order to gain a deeper understanding of the problems from the users' perspectives. The following subsections reports the findings from these studies, along with a study of the legal security requirements, which this kind of equipment must be in compliance with.

2.2.1 Hospitals

In the context of the *Activity-based Computing* (ABC) project [7, 17] (and a related *iHospital* project [49]) a number of workshops were carried out involving clinicians from hospital wards—a surgical ward at the hospital in the town of Horsens in particular. In addition to this several discussions with clinicians from various surgical hospital wards was held throughout the course of the project.

In 2008, the Danish Medical Association produced a policy on health-care IT, stating their visions and goals on the proliferation of information technology in all health-care sectors [65]. According to this, an imperative design criterion is that:

> *" [. . .]IT-systems must be designed to improve treatment, not delay or harm it. If the clinicians must unnecessarily disrupt their activities in order to type—or verify—information, which are not relevant in the current context, the system will loose its value as it increases the risk of malpractice. "*
>
> [65, p.7, my translation]

The main conclusions from these studies and activities can be summarised by the following list of demands from hospital users:

Wireless with unlimited range within the hospital. Clinicians have a strong dislike for cables, as they add to the clutter around a patient, fall off, get entangled with each other and with the tubing, and hinder the mobility of patients—and the clinicians working around the patient. Furthermore, wireless sensors with limited range is a problem when a patient who needs continuous monitoring, moves around or is moved.

Uncomplicated operation. The clinical work is characterised by multitasking and frequent interruptions. Therefore, equipment which require a lot of unnecessary (technical) attention from the clinician is problematic, as this draws her attention away from the job at hand, and adds to the number of conceptual different tasks she needs to juggle concurrently. In addition to this, the user interface of the sensor should only require a single hand, as a clinician would often need to support the patient or perform a different task using the other hand while operating a sensor.

Automated logging. Clinicians are required to do a lot of “paperwork”, documenting their actions and observations for future reference. If the sensors could assist by performing some of this work, clinicians can be relieved, thereby saving time.

2.2.2 Home Care

In order to study the use of medical sensors for home care, the author observed the *ElderTech* experiment [12,13]. In the homes of 7 elderly citizens a Bluetooth-enabled bathroom scale and blood-pressure meter was installed along with a laptop equipped with software (dubbed *Roberta*) developed by IBM collecting the measurements from the two sensors—along with a number of other functions, such as medication administration, calendar, and communication between the elderly, home care workers and clinicians.

A noteworthy occurrence during this pilot test was when one of the 7 elderly used the two sensors for daily measurements for 3 weeks without anyone noticing that the measurements were never recorded properly by the system.

The medication administration module in Roberta was not a success. While this may not be directly related to the medical sensors, it carries an important lesson, which applies to the sensor technology as well: the medicine module was designed from a clinician's point-of-view, but the elderly users of this application would handle their medication in very different ways, and the view presented by Roberta made little sense to them (for more details, see [84]). In most cases, the "user" of the sensor technology may be a clinician, but in some cases the "user" could be the patient; and patients and clinicians do not necessarily view or understand the technology—or the medical condition, for that matter—in the same way.

As a part of the ElderTech project, we also arranged 3 workshops. One workshop with a group of elderly citizens (still able to live in their own homes) and their relatives, one with home care workers, and one with nurses and physicians from a geriatrics department at a hospital. The theme of these workshops were, how to best use technology to enable elderly citizens to stay in their own homes for as long as possible. While the clinicians—and to some degree the home care workers as well—were prepared to fill up the homes of the elderly with all kinds of technology, almost transforming their homes to hospital rooms, the elderly themselves were focused on living normal lives with minimal intrusion from technology. This was a clear lesson that the interests of clinicians and patients may often conflict, and that both groups must be involved when developing new technology.

In summary, the demands from the home care context include:

Ultra-low power. When monitoring a patient with some chronic condition, sensor batteries should preferably last for very long periods.

Simple use. When the patient becomes the operator of the sensor, it becomes even more important that the equipment (the user interface in particular) is simple and intuitive. The patient may not be as well trained, or understand the technology and its purpose as well as the clinicians do.

Patients are users. Design not just for the clinicians, but also with the patients' needs in mind—in particular when sensors are to be used outside the hospital. Technology should be minimally intrusive. For instance, a sensor to be worn permanently should be discretely hidden under the clothes whenever possible.

2.2.3 Emergency Response

Compared to hospitals, the emergency scenario can complicate things a lot. First of all, the lighting conditions may vary a lot—from blinding sunlight to almost complete darkness. The working conditions are also worse than in a hospital. Beds and plenty of equipment are usually not available at emergency scenes. Instead patients are placed on stretchers or perhaps directly on the ground. Often many patients balance between life and death and must be treated in a hurry. During a major accident, like the train accident described below, many patients must be examined, prioritised and monitored, leaving only few seconds per patient for the initial examination. The patient-to-clinician ratio is a lot higher than in a trauma centre, and the patients' identities are often unknown to the workers, complicating things even more compared to the hospitals.

The author participated in a number of workshops concerning major incidents arranged by the PalCom *IT Support in Major Incidents* project [83]. The workshops involved professionals working with major incidents, including paramedics, trauma teams, ambulance personnel, police, fire-fighters, hospital trauma centre staff and coordinators. Also, in connection with the PalCom Major Incidents project, the author observed a 3-day medic team course on pre-hospital emergency response. This course focuses on both minor everyday incidents like traffic accidents and cardiac arrests, and major incidents like bus or train accidents. During the course, current practice and state-of-the-art technology was taught, and a number of training scenarios for 3-person medic teams (such as car collisions, fires, and shoot-ups) were performed at the fire-brigade's training facilities in Aarhus. Furthermore, a major incident exercise was performed, described in detail below.

From the workshops, and from other field studies carried out by individuals from the PalCom research group, we have learnt a lot about how the professionals do their work, and what is important to them. Many of the important points from this study can be found in [54].

One important point is, that the equipment used for major incidents and disasters must be the same equipment that the rescue workers use for their everyday tasks. Therefore, when designing equipment for small everyday incidents, one has to keep in mind, that it must also scale to large-scale incidents. Rescue workers might be persuaded to use special equipment designed only for large incidents during exercises, but they will *never* use it during a *real* major rescue operation, if they have no experience in using it from their everyday jobs.

At a typical major incident scene, the medical treatment area will be set up close to the accident scene, and is divided into three subareas: triage and waiting area, treatment area, and pick-up area. The fire-fighters bring all the casualties to the triage and waiting area (usually the medical personnel will not be allowed on the actual accident scene for safety reasons; only fire-fighters are allowed here). A clinician (doctor or nurse) will be receiving the casualties, performing initial examinations and triage, and a secretary follows this clinician, noting the observations on the accident cards and keeping a log of all patients. People who do not need physical treatment are handed over to the police, who will arrange for their transport and psychological help. Triage divides the patients into four categories. 0: Dead or cannot be saved with the available resources; 1: Immediate treatment needed; 2: Treatment as soon as possible; and 3: Treatment can wait. Patients in categories 1 and 2 are then moved to the treatment area, where doctors and nurses will be providing first aid and further treatment according to the priorities. Eventually, the patients will be moved to the pick-up area, where they are monitored until they can be picked up by an ambulance. The medical coordinator will be located at the pick up area. His job is to keep an overview over all patients and their needs for hospital treatments. It is important that patients are sent directly to a hospital specialist unit where proper treatment can be started right away. For instance, patients with large internal bleedings must be sent directly to an appropriate operating theatre, while patients with severe head traumas must go directly into a neurosurgery specialist unit. The medical coordinator communicates with a hospital coordination unit (via a radio link) to report what kinds of treatments are needed, and the hospital coordinator reports back, which hospitals have the required capacity. Based on ambulance availability, the medical coordinator decides when, where and in which order the patients are transferred to the hospitals.

The major incident exercise at the pre-hospital course described below provides excellent examples of the difficult circumstances, people and equipment must be capable of coping with during a disaster. Only two medical secretaries signed up for this course, and since all participants were divided in three groups for this exercise (which was then repeated 3 times), the author was offered the chance to participate in the exercise playing the role of the medical secretary of the third group. This was a very instructive experience. During the other two repetitions, the two other groups of medic teams were observed—in particular the triage doctor and the medical coordinator.

The exercise scene, figure 2.2, was a train derailing with about 40–50 casualties (played by a couple of very committed 9th grade classes from a local school). A medical staff of 6 people (including the medical secretary), plus some fire-fighters, helpers (young men from the Danish Home Guard, who would be moving the stretchers and monitoring patients waiting for treatment) and ambulance-men were assigned to rescue all these people, delivering first-aid, prioritise their treatment, and decide which hospital they should go to. And the job of the medical secretary (played by the author in one of the three repetitions) would be to document all of this by filling out the *accident cards* of the patients (described in [54]) and keeping a log of all patients, including their status and which hospital they were transferred to.

The key lessons learnt from this exercise along with all the workshops are summarised below:

Few seconds per patient. The triage doctor or nurse spend only a very short amount of time with each patient. Within less than a minute or so, the patient's vital signs are checked, injuries are identified, in-

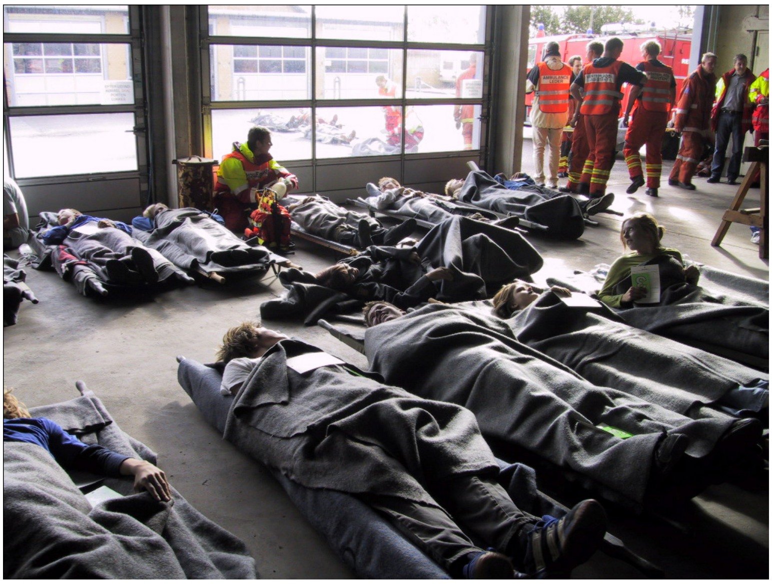


Figure 2.2: Train derailing exercise. In the background, outside the windows, firefighters deliver the casualties as they are rescued from the train wreck, and triage is performed. To the right (just outside the picture), casualties are treated at the treatment area. The patients in the foreground are being monitored, while they wait for ambulances to pick them up. The ambulance pick-up area is to the left.

cluding the possibility of internal bleedings and traumas to the central nervous system (head, neck and back), and the patient is categorised in one of the 4 categories. The secretary writes all findings on the accident card. If the patient is conscious and only if time permits, the patient's name, Social Security Number (SSN) and address will also be written on the accident card. If the task of installing a sensor on the patient must be performed in this narrow window of time as well, it cannot take more than a few seconds.

Interrupted work. The work of the triage team is often interrupted. As soon as the triage doctor can identify that the patient belongs to category 3 (treatment can wait)—or perhaps even 2 (treatment as soon as possible), he may have to move on to the next patient and then perhaps come back to this patient later, when all patients in a more critical condition have been passed on.

Postponed work. Prolonged tasks, such as filling out the accident cards, are often postponed until time permits. This, for instance, implies that many accident cards will be only partially completed, and experience from real major incidents shows [54], that they may not be used at all!

Patient identification. Discussing a patient, who is not nearby, can be very difficult, as patients are very hard to identify. For instance "The middle-aged guy in a dark-green coat over there [pointing], who injured his face and left leg" could be a way to identify a patient.

Partitioned teamwork. The triage team, the first aid / treatment team and the medical coordinator are all busy solving different tasks at different locations. In the case of a major incident with many casualties, the three teams will be separated by a physical distance as well, meaning they may have to use radios, if they need to communicate. This, among other things, complicates patient identification even further.

Familiar technology. At major incidents only well-known technology is used. This implies that sensors will only be used at major incidents if they are used at everyday accidents. Hence, sensors designed to be used in small accidents should also scale to major incidents.

Monitoring sensor readings. Although the sensor readings can be essential at a major incident scene, there is not enough personnel to continuously monitor the sensors on location. Therefore sensors must either work autonomously, deciding the condition of the patient on its own and give a clear alarm, if the patient is dying or rapidly deteriorating, or sensors must transmit the readings to a central station (perhaps at a hospital) where all patients may be monitored. If a central monitoring station is used, some mechanism for *locating* the individual patients is necessary (cf. the patient identification problem above).

2.2.4 Security Regulations and Related Work

As medical sensors handle sensitive information about the patient, special considerations regarding security are necessary. Privacy of medical information is of paramount importance, and is regulated by law as well as ethical standards.

The overview presented here is based on Danish laws and ethical standards, and while the particular details of legislation may vary from one country to another, similar principles apply in most countries.

Regarding the clinicians' obligations, section 2.7 of the Ethical Guidelines for Nurses [28] states: *The nurse must protect confidential information about the patient.* The same requirement is made clear in the Danish Health Act [90] chapter 9. This chapter of the Health Act explains how the clinicians (and other health-care professionals) must protect information about their patients and only access necessary information regarding a patient after this patient has given an informed consent. In case such a consent cannot be obtained (for instance if the patient is unconscious), it is up to the responsible clinician to judge if accessing the information is important and in the best interest of the patient, and if it is, the clinician is obliged to make a record of the incident and to inform the patient as soon as possible [90, §42a.5]. This rule is commonly known as *Værdispringsreglen* in Danish—roughly translated this means *Rule of Priority Reversal*.

Regulations on nursing records [89] direct what should and must be recorded and how the records may and should be updated: Every record must be signed by the clinician who made it, noting when it was made (for electronic records a digital signature solution may be used). A record may never be altered, and electronic record systems should be designed to enforce an append-only rule, which means that if an error is identified in an existing record, a correcting record must be added to rectify this wrong—once again time stamped and signed by the clinician responsible for the entry.

Laws do not just regulate the actions of clinicians, but set up standards for IT equipment as well. The Danish Personal Information Act [88] defines any medical information about a person as *sensitive*, and requires that it must be treated carefully. In particular, according to §41.3, precautionary measures must be taken to avoid accidental or deliberate (illegal) deletion, alteration or unauthorised access or leaks.

More specific instructions can be found in the Information Security Guidance Note from the Danish National Board of Health [102]. This guide defines the standard of IT for Danish health-care, hence it deserves special attention. Several sections in this guidance note are devoted to the aforementioned Priority Reversal rule, and the note explains that health-care IT systems must be designed to quickly handle situations where a clinician invokes this rule.

It is the responsibility of the local management to identify and evaluate threats—general as well as concrete—and take appropriate precautions, choosing a balance between security measures and convenience. For instance, if a public person is admitted to the hospital, there will be a high risk of journalists lurking around, chasing information about his condition, and extra (non-standard) measures may be necessary to prevent information leaks [102, sec.3.1].

The guidance note states that *"the purpose of logging and surveillance is to uncover unauthorised actions"*, and that *"all user activities, anomalies and security incidents must be logged and stored"* (author's translations). In particular, any invocation of the Priority Reversal rule must be emphasised in the log. Log entries must include at least the identity of the clinician performing the action, date and time, identity of the patient(s) involved, and the type of the action. Special precautions must be in place to protect against illegitimate alterations or deletions from the logs. The Information Security Act makes a dispensation from this logging requirement specifically for medical equipment, when readings (including stored personal information about the patient) are accessed locally [87, §19.5]. This implies, that for instance reading the ECG on a bed-side monitor may be performed without (automatic) logging.

According to the guidance note as well as the Personal Information Act, any medical information sent over an open network, such as any wireless network or the Internet, must be encrypted using a strong encryption algorithm, and the communicating parties must verify each other's identities.

Security considerations regarding medical WSNs was examined in great detail by [18]. Many different threats and possible solutions are mentioned, for instance the importance of privacy of sensor readings, so that unauthorised parties cannot get access to the readings or at least not learn the identity or position of the source of the readings. Hardware based solutions are also considered to some problems, for instance the use of controllable directional antennas to make it more difficult for adversaries to eavesdrop on conversations and at the same time save RF energy, thus reducing transmitting power. Problems with adversarial behaviour among clinicians and other health-care employees are also considered:

> *" Changes to a patient record to avoid lawsuit or to intentionally alter a patient's treatment are examples of attacks.*
>
> *The automated sensor network used for treatment and control of patient's health should be protected against attacks such as disabling monitoring devices, switching off or crashing computers, installation of flawed software, and disabling of sending/receiving messages. "*
>
> [18, p.3]

In [72] another concern is brought up regarding (chronically ill) patients wearing sensors on a daily basis. The very presence of the sensor device may be a privacy issue in itself, the patient may not like to disclose to his employer, acquaintances or medical insurance agent, that he is suffering from a medical condition. This aspect should also be taken into account when designing such devices.

A comprehensive overview of most security concerns in wireless medical sensors and actuators was presented in [41]. This article explores a number of problems, including the importance of the Priority Reversal rule presented earlier. The article also presents various types of adversarial behaviour, for instance a patient who would attempt to manipulate his own morphine dose from a medication pump.

As a final reminder that security for medical IT must be designed with regard to the users, [16] reports many interesting observations from a hospital ward. For instance, a common practise at this ward was to write user names and passwords for the Windows login directly on the PC monitor and to leave the PC without logging off, so others may use it without logging in—undermining security as well as traceability, as laid out by the laws and the Information Security Guidance Note. All this because the log-in and log-out steps would steal an unreasonable amount of time. This subject is also treated by the Danish Medical Association's health-care IT policy [65, p.7], which states that current systems are to rigid, are the cause of inefficiency and waste of time, and furthermore that the way the systems are designed, encourages anarchy toward them.

2.3 State-of-the-art and Related Work

Today medical sensors and displays generally have a 1-to-1 relationship, usually having the sensor and display integrated physically into a single unit, e.g. thermometers or blood-pressure meters, or perhaps into two units connected by a cable, e.g. ECG or pulse oximeters.

Some manufacturers have developed equipment that simply replaces the cable with some wireless interface and protocol, e.g. the LifeSync Wireless ECG system [61], which replaces the cables between the ECG electrodes and the monitor by a Bluetooth connection, and [80], which does the same thing for a pulse oximeter. The problems with Bluetooth was already explained in section 2.1.3, and of course these sensors are rather big and heavy, and they need the receiver within a distance of 10 meters. A paper at the LifeSync web-page [61] claims that their wireless Bluetooth-based ECG system can be expected to work well in hospitals; these claims are based on tests using only 8 systems for 18 hours—a very weak argument for the scalability of their system (cf. the discussion in section 2.1.3). Another problem with both instruments is how to inspect existing connections. The LifeSync wireless ECG has a very nice user interface for establishing the Bluetooth connection using a small token (it is unclear how exactly connections are established with the pulse oximeter), but once a connection is established, it is not possible to visualise it. Therefore, if several sensors and monitors are used in the same area, some other solution must be invented (such as writing names or numbers on the devices with an ink pen).

Another very common approach is to extend sensors with wireless (usually Bluetooth) capabilities in addition to their own displays, so that readings can be optionally transmitted to a nearby computer or cell phone. [8] offers bathroom scales and blood pressure meters with this feature.

Wireless monitoring systems (a.k.a. *telemetry*) have been developed for intensive care monitoring at hospitals. GE Healthcare offers a wireless extension to their patient monitoring system [37] allowing the patients free movement at the hospital within the system's area of coverage, while monitoring ECG, SpO_2, blood pressure and temperature. The size and weight of the units are in the order of cell phones and the battery lifetime is no more than 5 days. The wireless technology is a proprietary system, so the cost of covering an entire hospital (or even a few wards) with their proprietary base stations can be quite significant.

Welch Allyn [111] also provides a wireless extension to their telemetry systems. The Micropaq has similar features as the one presented in the previous paragraph. However, it uses regular WiFi for the network infrastructure—which makes the infrastructure cheap. This of course comes with a penalty as WiFi was not designed for this type of use: the size of the device is bigger (in the order of a PDA), it weighs half a kilo-gram and the battery lasts for only about a single day.

The *Electronic Patch* project [32, 39] seeks to develop wearable sensors with the form-factor of a patch. Different sensors are demonstrated including ECG and a pulse oximeter. The project proves that it is indeed feasible to build wireless sensors small enough (including the battery) to fit inside a patch, and user tests on patients from the “home care” context show, that the patients are very happy with this type of sensor. However, the project makes no attempt to solve the problem of user interfaces.

In [67] a number of critical challenges to be addressed in the CodeBlue project are identified:

1. Secure, reliable, ad-hoc wireless communication under limited energy and computational resources. Different priorities should be supported as well, for instance, alarms should be quickly delivered even if the network is congested.

2. The trade-off between computational power and the strength of data encryption, and the accompanying usability issues: Clinicians must be able to easily identify themselves, assign access rights and transfer these rights when, for instance, the patient is taken to a(nother) hospital. The attention is drawn to the fact that existing authentication systems are far too rigid in this respect.

3. The lack of a common programming framework and a flexible protocol suite.

Furthermore, the CodeBlue team dismisses the use of WiFi class devices, and the use of traditional middle-ware architectures, such as agents, remote procedure calls, and virtual machines (like Java), as these approaches inherently spend too much power [69].

In a more recent technical report [98] the CodeBlue team proposes an architecture for ad-hoc networks of medical sensors. However, security issues have been totally ignored in this architecture, which is based on general service discovery—all data are sent unencrypted, and no mechanism for authenticating the users is in place, nor even considered. Furthermore, the fact that sensors appear in the user interface identified only by a sensor ID seems disturbing, as this makes it easy to mix up patients. Even if some sort of database was added to provide a mapping between sensor IDs and patient names, someone would still have to provide these bindings and how will mistakes at this stage be prevented?

The Agent Based Casualty Care (ABC Care) project at Dartmouth College (New Hampshire) was designed for soldiers in combat [112, 113]. The soldier is outfitted with sensors and a PDA-size device monitoring his condition. The device reports back to the headquarters, placing the soldier in one of five categories—the four triage categories mentioned in section 2.2.3 plus a "not injured" category. The device determines the state of the soldier using 3 types of input: The readings of the medical sensors (ECG, SpO_2), asking the soldier himself about his condition, and asking the soldier's "buddy" (another soldier on the same team) about his condition. The ABC Care project has tested two different approaches of assessing the condition of the soldier: A strict rule based system and a fuzzy logic system. Both approaches works quite well (more than 90 % accurate).

The ABC Care project exploits that power and communication possibilities are available in the standard outfit of a combat soldier. Whether a similar solution could work for a major emergency remains to be tested. However, from the field-study observations described in section 2.2.3, it is clear that a few patients will be seeking attention by screaming, but they are rarely the ones who need the most attention (as they can mount the physical energy required to scream, while others cannot). As the input from the patient has a major impact on the triage category computed by the device, this would give the attention-seeking patient a tool to draw attention to himself, and since the average victim of an emergency cannot be assumed to demonstrate the same discipline a professional trained combat soldier is expected to demonstrate, this suggests that at least some modification will be necessary.

The UbiMon project at Imperial College, London [79], seeks to develop a sensor network suitable for hospital use. Their architecture is based on Body Sensor Networks (BSNs). Sensor prototypes are built using a locally developed variant of the Berkeley Telos mote [21]. Around each patient a BSN is formed consisting of the sensors and a PDA or a cell phone capable of reaching the main network using WiFi, GPRS or UMTS. On the main network a server collects the readings, do appropriate analysis, and store the data in a patient database.

Chapter 3

BLIG: a User Interface for Medical Sensors

One of the inevitable questions one must face when dealing with small wireless medical sensors, is the question of how they can be operated and connected to display devices. This was also the first research question on the list in section 1.2.1, and a proposal to answer this question is the subject of this chapter.

The results and contributions of this work were presented in the paper *BLIG: A New Approach for Sensor Identification, Grouping, and Authorisation in Body Sensor Networks* [1] found on page 119 ff.

3.1 Motivation

As pointed out in section 1.1, a physical connection (common casing or a cable) between a sensor and its display provides power for the sensor, privacy and authenticity for the data transmission, and a tangible user interface, which is easy to understand. Since this chapter focuses on the user interface perspective, let us examine this point a bit closer. A physical connection offers the following features to the user:

Manipulation By the act of plugging a cabled sensor into a socket on a display, the user's intent of having the data from this particular sensor displayed on that particular display is clearly communicated in a non-ambiguous way toward the equipment.

Inspection By inspecting the connections, the user can learn how sensors and displays are connected. In particular, if a user ever discovers a cable going from a sensor on one patient to a monitor placed at another patient's bedside, she will immediately know, that something is wrong.

Introducing wireless connections, some user interface mechanisms must be in place to handle the manipulation and inspection of connections. As anyone who have attempted to attach a Bluetooth headset to a cell phone will know, setting up and inspecting a single connection is not always easy. Now, imagine several identical phones and headsets (all connected in pairs) becoming mixed-up; the task is to figure out which headset is connected to which phone.

In the medical sensor setting the challenge is even greater, as sensors can be very small with very different form-factors, such as pills or patched.

3.2 State-of-the-art and Related Work

The most common method of establishing the relation between a sensor and the sensor readings in state-of-the-art equipment is some variant of the *cable-cutter* approach, where the physical cable is replaced by an invisible, "logical" cable between a sensor and its display.

For example, the LifeSync [61] units come with a small token (a small plug about the same size as a cell phone SIM card). This token is used to set up the connection between the transmitter and the receiver units. Upon this set-up, the token is removed from a port on the receiver (where it is initially located) and inserted into a port on the transmitter unit for a short duration. When the token is returned to the port on the receiver unit, a beep will confirm success. The token guarantees privacy, as it carries a strong encryption key. However, there is no way to tell which transmitter belongs to which receiver if several units are mixed up. Of course, low-tech solutions, such as stickers or a permanent marker pen could be used to remedy this deficiency.

Telemetry systems used at intensive care units, such as [111, 37], have a number of inputs. Each input is associated with a monitoring station—either cabled or wireless. As there is no real difference between the two, this approach is merely a variant of the cable-cutter approach.

The CodeBlue project [67, 98, 69] mentioned in section 2.3 uses service discovery in their current prototype, however, they have not yet incorporated any kind of security, and all sensor data is available at any display. A sensor ID will accompany the readings, but it is up to the user to make the logical link between this numeric ID and the patient whom the data relates to. There is no privacy enforcement; however, the main focus of this work seems to be on emergency sensors, so in this case, security may be of less importance, as the sensors are only used during life-saving first-aid work, and not for continued patient monitoring. According to [98], US regulations do not require privacy during life-saving procedures.

In the PalCom Major Incidents project [53, 83, 54, 55], a sensor named *BlueBio* was developed. This sensor is a single unit which is placed on the upper chest of a patient and picks up the individual's vital signs. An adhesive wrist-band containing an RFID chip will accompany this sensor, and this wrist-band is placed on the patient much like the luggage id-tags used in airports are attached to suitcases. A display device contains an RFID scanner, and the sensor data will be received by the display by scanning the RFID tag. Incidentally, the argument for placing the RFID tag in a wrist-band rather than on the sensor itself, is that the upper chest region of the patient (where the sensor must be placed) will usually be covered by either clothes or blankets.

A fourth approach found in research projects uses some kind of identification device worn by the patient. Any sensor attached to the patient is set up to deliver its readings exclusively to this identification device. The identification device can be set up to forward sensor readings to various displays or database servers along with the identity of the patient (possibly in a secure manner).

The UbiMon project [79] uses this approach; each patient carries a PDA or cell phone which acts as a personal identification node and a bridge between this Body Sensor Network (BSN) and the LAN/WAN.

When using this fourth approach, a method is needed to set up the connections between a sensor and the identification device. One way of doing this was proposed in [46]. In this proposal, two or more nodes can be explicitly bonded by shaking them together. Built-in accelerometers will detect the movement pattern, and because these patterns are similar for both nodes, they will connect to one another.

In [10] another method is proposed for setting up the connection between a sensor and a patient identification node in a BSN. This method gives each clinician a personal pen with an infrared transmitter. This pen is then used to set up a connection between a sensor node and a patient's identification node. This is simply done by pointing the pen toward the sensor node and the identification node while pushing a button on the pen.

Finally, if multiple sensors are touching the skin or have access to the heartbeat of the same person they may even be able to find each other automatically [33, 14, 85].

3.2.1 Discussion

The cable-cutter approach has some drawbacks when it comes to inspecting how sensors and displays are connected. As mentioned above, the LifeSync device has no means of doing this at all. But even if the device has some way of showing which peer it is connected to, this identity would probably be some kind of numeric ID—e.g. a network address. To inspect the connections, one would have to compare these numbers in order to verify that everything is in order. This can be very tedious and time consuming. Furthermore, with this approach sensors do not know the identity of the patient, so when the sensor readings are transmitted to a database server to be stored in the patient's medical file, the server must know the identity of the patient wearing that particular sensor. This is exactly what happens in common telemetry systems like that of Welch Allyn [111]: when a new patient is admitted for monitoring, his identity is typed into the central monitoring station's computer (which would usually be located in an office nearby).

The service discovery approach and the RFID-based approach are no different. Still, the sensor has no idea, who the patient is, so the logical link between the two must be provided elsewhere. Service discovery is an approach commonly used in the pervasive health-care research of sensors. However, to the best of the author's knowledge no commercially available sensors for hospitals use this method yet—probably due to the strict security demands for hospital IT equipment discussed in section 2.2.4, which would be difficult to combine with this kind of service discovery approach.

All of these three approaches share the same two drawbacks: First, the link between a sensor and a patient must be provided, typically by typing it into a computer placed in an office—i.e. *not* the same room as the patient, thereby increasing the risk of human error caused by accidentally mixing up patients. In an extensive study of the work in a surgical hospital ward, [16] identifies this pattern of splitting a single task into two—one at the bedside and one at the office—as the source of much stress among clinicians and increased risk of errors. Second, every time the patient is moved to another location, this whole procedure must be repeated, multiplying the risk of errors—and the time consumed.

The last approach presented above, on the other hand, implies that the true identity of the patient (e.g. name and social security number) is carried by the patient in the identification device, and any sensor placed on him will have access to this identity. Therefore, as the patient is moved around the hospital—or even to another hospital—his identity follows him inside the body sensor network, he is wearing. Furthermore, when a new sensor is installed on the patient, it is immediately logically connected to the identity of the patient. This is no longer a task performed in an adjacent office (away from the patient), and therefore the risk of mixing up patients has been reduced. In addition, the identification task is performed only once (when the identification device is installed), no matter how often the patient is moved to new locations. This reduces the risk of mix-ups even further, and also reduces the work overhead of moving patients significantly.

Still, a major problem remains: none of the above solutions offer an answer to the problem of establishing visible and tangible connections. How can a user quickly inspect the sensors on a patient to make sure they are set up correctly? For instance, it is important to the clinician, to be confident that the newly attached sensor was in fact set up to be paired with the correct patient's identification node—and not by accident the neighbouring patient's identification node, which may be less than a couple of meters away. No solution is given to this problem.

3.3 BLIG: Blinking LED Indicated Grouping

Before the sensor user interface proposal is introduced, let us examine the principles which have been guiding the design of this user interface—as well as the entire platform presented in later chapters.

3.3.1 Design Principles and Goals

The main design principle may appear to be trivial, but it has some extensive consequences as explained below.

Main design principle:

The wireless technology should not be inferior to existing (cabled) technology.

Recalling the points from the motivation in section 3.1 about the benefits of physical connections (easy manipulation and inspection), this design principle can be broken down to the following:

Fast and easy manipulation A cable can be plugged into a socket very quickly, typically using only one hand. As a nurse may need the other hand to support the patient or to carry other equipment, the "one hand" part can be quite important. Therefore, a goal of this user interface is, that it must be possible to "attach" a sensor in less than 5 seconds using a single hand.[1]

Inspection Tangible and easy-to-understand feedback to the user. This includes a confirmation during the above-mentioned sensor-attaching procedure in order to verify that the connection was established correctly (corresponding to the "click" sensation, experienced when a plug is correctly inserted into a socket); an ability to easily inspect existing connections; and of course a means for troubleshooting, when problems occur.

Security A physical connection provides privacy and authenticity of sensor readings. This chapter will not go further into details about the goals, and how they are achieved, as this will be thoroughly covered by the following two chapters. Suffice it to say, that the security of the chosen platform puts a requirement on the user interface: the (clinical) user performing an action (such as attaching a sensor to a patient) must be authenticated—meaning that she must show (and prove) her identity.

[1] The word "attach" here refers to the effort of establishing the wireless connection only—corresponding to the act of plugging a cord into a socket of a stationary display. Of course, many types of sensors require more from the user, but this extra work is not included in the "one hand in 5 seconds"-goal stated here.

Resilient and scalable Disconnections, node failures, infrastructure failures, and congestion or deliberate jamming of the wireless link are problems that may occur with wireless sensors, which does not occur when physical connections are used. Furthermore, when physical connections are used, any number of sensors and displays can be used simultaneously in a confined area (such as an emergency scene) without interfering. But if wireless sensors are used, they will have to share a common wireless medium, and then congestion may be likely to occur. The goal of this work is to ensure, that the equipment always works locally (close to the patient). This makes the solution "not worse" than physically connected equipment, as a display can always be connected to sensors on a patient to monitor readings locally, if chaos rules the network.

3.3.2 User Interface Hardware for Sensors

Facing the problem of creating a user interface for sensors such as thermometer pills or ECG patches, a major question is: What kind of hardware can be used to solve this problem? Certainly a keyboard and an LCD display will not work on such a small scale—and for disposable sensors, they would also be too expensive.

When selecting the hardware to be used as basis for the user interface, the following demands must be considered:

- Sensors may be *small* and have various shapes / form factors.
- The hardware must be *cheap*, as many sensors will be disposable in the future.
- *Low-power* operation is essential.

In the proposal presented here, the user interface is based on one push-button, and a group of three light emitting diodes (LEDs) in the primary colours (red, green, and blue). One advantage of this combination is, that it can be used with any sensor device—no matter how small, or what shape it is. A button need not be very big, for instance a pill sized sensor can be squeezed between two fingers, and a patch may be sensitive to bending or pressure. Furthermore, LEDs can be made very small (certainly less than a mm^2 of silicon). At the same time, LEDs are very easily seen at a distance, and in bright sunlight as well as darkness. LEDs are certainly cheap, and a button can be designed in a number of different ways—including some, that are cheap. The button does not require any power, however, LEDs *do* require quite a bit of power, but we shall revisit this issue later (section 3.3.6), looking into ways to reduce power consumption for extremely low-power operation.

With just a single push-button for user input, the expressive power of the user is quite limited. Pressing the button at a certain *place* (close to some point of interest), on the other hand, can make a lot of sense. Hence, the button and LEDs are supplemented by proximity detection. In the next chapter (section 4.3.1) the actual hardware implementation is discussed, but for the sake of the discussion here, it suffices to say that devices with this proximity detection capability are able to detect and identify other nodes with similar capabilities within a very short range—less than half a meter.

In summary, the hardware chosen here, fulfils most or all of the requirements stated above:

	LEDs	Button	Proximity
Cheap	√	√	?(1)
Small, various shape	√	√	?(1)
Low power	?(2)	√	?(1)

(1) Hardware-dependent, cf. section 4.3.1

(2) Variable, cf. section 3.3.6

3.3.3 Authorisation-nodes

As stated in the list of design goals above, authentication of the clinical user is a necessity each time a sensor node is attached to a patient. As the user is obviously located near the node being attached when this is performed, the proximity detection described above can be used to authenticate the user, if the user is carrying some device, which has the same proximity detection hardware, and is capable of proving the identity of the user.

This device, known as the *authorisation-node* (as it will also be used to prove that the user is authorised to perform the action, as described in the following chapters) could for instance be a part of the uniform—probably most likely the name tag.

Since authorisation-nodes are not disposable, and does not have to be particularly cheap, they may be equipped with more advanced hardware and capabilities. A node is used to authenticate the identity of the user, i.e. prove that the user really *is* the person she claims to be, and therefore it should feature some kind of log-in procedure to ensure that it can be used only by this person. A fingerprint reader would be one possible approach for secure login on a device of the size and shape of a name tag. It would probably be appropriate in most situations (although there would be some exceptions due to hand hygienics and the use of gloves).

The authorisation-node worn by the clinician cannot be allowed to become the object of the user's attention—the user should be focused at the patient and the work at hand (for instance installing a sensor). Since the user should never have to look at the authorisation-node, visual feedback to the user from this node in the form of displays or even LEDs would be impossible. Instead, the authorisation-node has a buzzer capable of producing different sounds.

A single button on the authorisation-node may also be practical in some scenarios, for instance for inspecting the local sensor deployment (on multiple patients) as described later.

3.3.4 Group-based Node Organisation

The main lesson from the discussion in section 3.2.1 was, that the identity of a patient should be known to the sensors attached to him, in order to ease the task of moving the patient and reduce the risk of patient identity mix-ups.

The approach used in some related work uses a special patient identification node, which is attached to the patient and contains his identity. The sensors on the patient are "dumb" in the sense that they do not know the patient identity, but they are configured to deliver the readings exclusively to this identification node, which will then forward the readings—now tagged with the patient identity—to the final destination. This approach has a few drawbacks:

- The patient identification node constitutes a *single-point-of-failure.*
- Each patient must carry at least 2 nodes—one sensor and the identification node.
- Installing the identification node can be an unacceptable and time-consuming overhead—especially at emergencies.

On the other hand, this approach has a number of benefits. If the patient identification node is much more powerful than the average sensor (for instance, a cell-phone-size device rather than a pill or a patch), it may boost the wireless range, provide better connectivity such as WiFi or even 3G, or provide a large memory for sensor data storage.

For the platform proposed in this dissertation, nodes will be organised in groups, such that there will be one group per patient. In contrast to the approach above, which uses a special identification node, all nodes in this platform are aware of the patient identity, and no special node is necessary. This resolves the drawbacks listed above.

In a context where better range, better connectivity, or extra memory are necessary, a *powerful* device offering this service can be added to the patient. This would usually be the case in the home care context, where the limited range of the radio transceiver in a small sensor may not be adequate—in particular if the patient is leaving the home, and cell phone networks are used to relay the data. Notice, that in case the powerful device fails or runs out of energy, the remaining network will continue to work, and local displays may be attached to monitor the sensors. In the emergency context (and at hospitals) where sensors can reach the wireless gateways directly, such a device would just be an unnecessary overhead, and would probably not be used.

3.3.5 Fast and Easy Manipulation

To perform a change in the sensor set-up, the user can interact with the sensors on a patient in three different ways, called the *configuration changes*:

Configuration changes:

- Install a new sensor on a patient.
- Remove a sensor from a patient.
- Change a patient property (e.g. update the patient identity).

The first two (installing and removing sensors) must take place near the patient (for obvious reasons), whereas the property change may be performed remotely—and even automatically. In this chapter, however, only the property change initiated locally by a user is discussed.

Notice that displays and actuators (such as medicine, fluid or nutrition dispensers) can be used in the exact same way as sensors—they relate to the clinical users in the exact same way. Henceforth, the word "sensor" will be used with this generalised definition: a device acting as either sensor, actuator, or display for a particular patient.

With the push button and group of LEDs found on a sensor the following user actions and feedbacks are possible:

The user can press the button:

- Short push (< 1 second)
- Long push (> 1 second)

The following signals may be displayed by the LEDs:

- Searching for nearby devices (seamless fading between colours).
- Found nearby device (2 fast white flashes).
- Group blinking (synchronised blinking among all nodes in that group).
- Error indication (fast, aggressive red blinking).
- Confirm power-off (fast, aggressive blue or green blinking).
- Power-off (slow fade from white to black).

And in addition to feedback from the sensor, the authorisation-node can provide audible signals ("OK" and "error").

Install Sensor

Prior to installing a sensor, the user must log onto her authorisation-node. This step is discussed further in section 4.3.3, but here it suffices to say, that the authorisation-node must be in a state where it is prepared to authenticate the user when requested to do so.

Figure 3.1 describes the BLIG user interface. Each state has an associated LED activity as indicated by the name (cf. the list above). Button push events, along with proximity detection and timeouts cause state transitions.

Installing the first sensor on a patient—thus forming a new group—is accomplished by pushing the button, holding the sensor close to the authorisation-node and pushing the button a second time. By bringing the sensor into the authorisation-node's proximity range, the user is authenticated. In many situations, installing a sensor on a patient must be logged by the clinical user in the patient's medical records. With this authentication step, the action can be automatically noted in such a log. The buzzer of the authorisation-node will produce a sound to confirm that this log entry was correctly made.

Additional sensors are added to the group in a similar manner, but rather than pushing the button the second time, the node is brought close to one of the existing group members. Sensors in the same group will share a common blink pattern, called the *group blinking*, which is unique for each group. This was the origin of the user-interface name BLIG: *Blinking LED Indicated Grouping*. Further details about the group blinking is offered by the paper *BLIG: A New Approach for Sensor Identification, Grouping, and Authorisation in Body Sensor Networks* [1] found on page 119 ff.

Remove Sensor

A sensor may remove itself—or simply disappear—if it runs out of battery or somehow detects that it has been detached from the patient. In addition to this, a user may explicitly remove the sensor by turning it off. Figure 3.1 shows the user interaction needed for this. When a sensor is active, a long

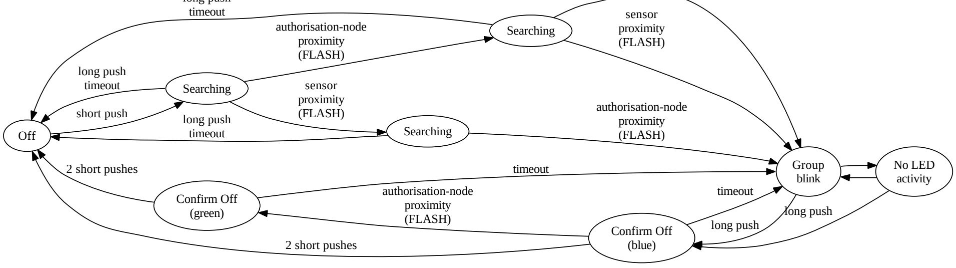


Figure 3.1: BLIG user interface state diagram.

button push causes the sensor to issue an “are you sure?”-warning. If the user confirms by another two short button pushes within a few seconds, the sensor will turn off.

In some situations, the user may wish to log the removal in the patient’s record with her signature, in the same manner as sensor installations are logged (procedures at a hospital may even require this). If this is the case, the user can bring the removed sensor within her authorisation-node’s proximity range in order to sign a log message stating that *she* was responsible for removing the sensor.

Change Property

Forming a group of sensors on a patient (by installing and removing sensors) is not enough. As explained earlier, all sensors in a group must be aware of the identity of their patient and where to deliver their data.

The *properties* of a group includes the name and Social Security Number (SSN) of the patient of the group, identities of servers allowed to access sensor readings, and information about which users are authorised to make further changes. Examples of property changes include:

- Input the name and SSN of the patient.
- Moving a patient from one ward to another (changing the information about which users are authorised to access sensors).
- Moving a patient from one hospital to another (changes the identity of servers as well as the information about authorised users).

In practise, inputting the name and SSN of the patient would be the only property change which would have to take place manually. After this is done, a hospital IT system would be capable of assuming control over the sensor group, performing all subsequent property changes. If a patient is transferred from one hospital (ward) to another, the IT system of the hospital can automatically initiate the necessary configuration changes remotely.

In order to type in the name of the patient (or perform any other configuration change locally), the user will need either a PDA or a local display with a keyboard. The user will be authenticated by the PDA or display by bringing the device close to the authentication-node or by performing a “conventional” login procedure (e.g. input user-name and password). If a PDA (which is not a member of the sensor group) is used, it is also brought close to one of the sensors in the group. This will start the group blinking on the sensors as well as a designated area of the display of the PDA.

3.3.6 Inspection

The group blinking (the unique common blink pattern for a group mentioned above) is used whenever a group relationship indication is needed. Each group of sensors forms a unique (long) pseudo-random sequence of colours, and all nodes in the group will slowly and synchronously light matching colours. For instance, with 7 different colours the probability of showing the same colour must be $\frac{1}{7}$. If the sequence is truly random, the probability of getting a synchronised n-step sequence is $\frac{1}{7^n}$, and this calculation does not even take phase differences into consideration. After a couple of steps, if two sensors are still presenting a synchronised blinking pattern, the probability that they are *not* in the same group is negligible. On the other hand, two sensors which are in the same group will always present identical and in-phase blinking patterns. Further details about the group blinking can be found in the paper on page 119 ff.

If all correctly installed sensor nodes would always be presenting this group blinking, inspection would be a simple matter of checking that all nodes on a patient are blinking synchronously, while nodes on different patients are not. However, the LEDs consumes about 0.5 joules per minute[2], and in comparison a watch battery cell (LR621) contains 81 joules, so keeping the LEDs on all the time would not be a realistic solution for long-term sensors. As shown by the graph in figure 3.1, when a sensor is installed and active, it can turn the LED blinking off and on—the conditions for triggering these on/off transitions are left unspecified on the graph edges, but will be described below.

In addition to being either ‘on’ or ‘off’, sensors with minimal energy resources may offer a third in-between state as well (not shown on the graph), where the LEDs emit a short flash in the proper colour lasting only a fraction of a second at the start of each colour time-slot (rather than the normal continuous light emitted in the ‘on’ state). This could save as much as 90 % of the energy at the cost of a slightly less user friendly interface. Some nodes have to emit the continuous light in order to allow colour comparison, and each node using this flashing mode rather than the normal blinking will slow down the inspection—hence, this mode should be used with care.

A number of different methods to trigger transitions between the group blinking on and off states can be realised:

Simple timeout Group blinking (for all nodes) is turned off after a number of seconds inactivity—and turned back on when a user starts a configuration change (e.g. installing a sensor).

Variable timeout Sensors with minimal resources use shorter timeout durations than other sensors (or use the flashing state described above after a short period).

Local on/off A short push on any sensor button will switch the state of the group blinking between on and off for all sensors in that sensor’s

[2] This could be more or less depending on chosen intensity—which could adapt to the ambient light conditions.

group. Quickly turning the LEDs on and off in an entire group using this method may even be a faster and easier way to inspect the group than waiting to compare group blinking colours.

Global on/off A push on the button of an authorisation-node could signal all groups in the vicinity (radio-range) of the user to turn on the group blinking. This way a user can inspect the sensors on multiple patients without having to approach each one of them. This method has a security weakness, as a remote adversary may succeed in launching a denial-of-sleep attack at the nodes, depleting their batteries; hence, it should be used with care.

The utility of each of these methods will be tested with the users in future work (cf. section 3.5).

3.3.7 Resilience and Scalability

Of course all wireless technologies are subject to congestion on the wireless medium, as well as deliberate or accidental jamming. The risk of congestion can be minimised by using radios with a relatively short range (low transmission power), as this will imply that only neighbouring nodes within a short range will be competing for the wireless channel. Deliberate or accidental jamming can never be completely prevented, so in this particular aspect, no wireless sensor-display connection technology can ever be as stable and resilient as a cable. This is an inherent problem with wireless technologies, which shall not be examined further in this dissertation. Instead, we choose to *assume*, that local wireless communication is always possible.

Only local communication is used for the configuration changes described here, so these changes can always be performed even with no infrastructure or server connections available. Possible configuration changes include adding a local display to the group of sensors (as displays are treated just like any sensor as explained earlier), and this guarantees, that as a fall-back strategy the sensor group can always *at least* be used exactly like the current (cabled) technology with a display alongside the patient.

With regards to scalability, the same argument holds: the local nature of the groups (and configuration changes) ensures a stable local operation even in large networks with possible bottlenecks at servers or network hubs.

3.4 Evaluation

To test the user interface ideas proposed by BLIG, a number of prototypes have been constructed over the years. Figure 3.2 shows the 4 generations, from the very first proof-of-concept at the left to the current prototype platform to the right.

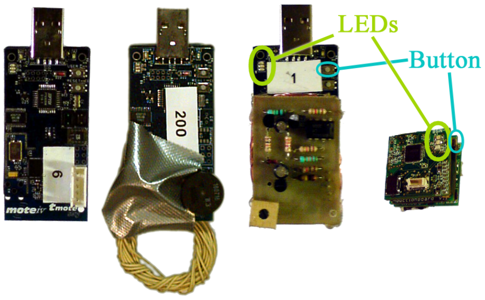


Figure 3.2: The four prototype generations. From the first very basic BLIG experiments on the node to the left, to the current prototype platform as described in chapter 4 to the right.

The left-most prototype was built to demonstrate the BLIG group synchronised blinking only, and it did not have any proximity detector. The second generation, which included a proximity detector, was used for usability experiments (only two copies were built).

The third prototype generation shown on figure 3.2 included a slightly improved proximity detector capable of transmitting a single 5-bit value (the node address). 9 copies of this version were built, and the first full horizontal prototype implementation (including both manipulation and inspection elements) was constructed.

The purpose of the fourth and current prototype generation was to implement a complete and secure platform. This prototype will be presented in chapter 4, and as explained in that chapter, a fifth generation of the hardware platform is already planned.

The first two prototype generations were demonstrated at two workshops arranged by the PalCom Major Incidents project in the fall of 2005 and spring 2006 involving participants from the local fire brigade and police force, emergency workers and trauma centre physicians. The reactions were very positive. As the first prototype generation was not equipped with the proximity detection hardware, one had to press a button on both units to indicate proximity. The clinicians were of course informed about this deficiency and did not seem to have any problems abstracting from it.

The participating clinicians expressed their appreciation with the easy grouping of the sensors and the tangible way groups can be recognised. Also, it was appreciated that the blinking LEDs are easy to see in all weather conditions, bright daylight and at night (which had not been considered at that time), and that no small displays are involved, making it easy to inspect connections at a distance and without reading glasses. A few simple scenarios were played around the table, and the clinicians demonstrated that they could easily use and understand this method. They also clearly encouraged continued development.

One of the participating clinicians expressed a concern regarding whether a colour-blind person will be able to use the equipment. However, total colour-blindness is a very rare condition, and the normal type of colour-blindness, dichromacy (which is the inability to see one of the primary colours), would still allow the user to distinguish between the remaining two colours. Furthermore, at the workshop the idea of using different rhythms to supplement the colour changes was proposed. This may also solve the problem of colour-blind users, and should be tested in future work.

Later generations of the BLIG prototype (generation 3 in particular) have been tested by a number of people (mostly having no clinical background) in order to ensure, that the user interface is easy to learn and intuitive to use.

User tests involving clinicians from all three use context (hospitals, home care, and emergency response) was planned from the beginning. However, memory requirements of the full prototype implementation grew beyond the capacity of the chosen fourth-generation hardware, as explained in the subsequent chapters, and it was not possible to implement a fully functional prototype. The options now were to implement either a horizontal prototype with all the user interface functionality necessary for the user tests but no real substance below the surface, or to implement a vertical prototype with a minimal user interface (i.e. *not* suited for user tests) but a full implementation below the surface. The latter option got the higher priority, and in the end, there was not enough time to do the preparations for the planned user tests.

3.5 Conclusion and Future Work

In this chapter the BLIG user interface for medical sensors was proposed, which can be used for any kind of tiny medical sensor—including pills and patches. The user interface offers group creation, inspection, and changes using only one button, a group of LEDs, and a proximity detector. Furthermore, this particular hardware choice (LEDs and a single button) makes the user interface applicable under diverse conditions, such as indoor/outdoor, darkness/sunshine, and using gloves.

An early implementation of this concept was tested with a group of clinicians and ambulance personnel, who found it to be easy to use and understand. Later versions have not yet been tested with clinical professionals, however, informal tests with nonprofessionals have shown similar results.

A proper user test involving clinicians from the three use contexts (hospitals, home care, and emergency response) was planned. However, there was not enough time to carry out these plans, so this part remains future work. The goal of these future user tests will be to investigate the following usability parameters:

Learnability How quickly will new users learn to perform basic operations?

Efficiency How quickly will trained users perform basic operations, such as adding a sensor to a patient, or establishing an overview?

Memorability When users have not been using the UI for a while, how easily do they reestablish proficiency?

Errors How often do users make errors? What is the severity? And how easily can they recover?

Satisfaction How pleasant is it to use the UI?

The idea of using the blink pattern of LEDs to convey the notion of grouping among devices to an observing human is new, and needs to be tested further in the future:

- How fast should the colours change?
- Are some blinking patterns better than others?
- Would different rhythms be helpful—in particular, if the clinician is colour-blind?
- Could blinking patterns and rhythms for instance be used to convey information about the condition of the patient? In the case of a major accident where triage is necessary, could the blinking pattern carry a message about the victim's condition (the triage category) somehow superimposed onto (or incorporated into) the group blinking?
- For how long time should the BLIG group blinking continue after the user interaction is completed? As the main BLIG power consumption is due to the LEDs, it would be important to develop ways to minimise their use. The power saving methods presented in section 3.3.6 should all be tested with users as well.

It will probably be the case that different BLIG flavours are preferred in different contexts, but the information about the current context and BLIG flavour may be stored in the authorisation-nodes, and then relayed to the relevant sensor nodes. However, the final judgement on this matter must be left to the users.

Chapter 4

Prototype Platform

With the initial work on the BLIG user interface presented in the preceding chapter, a number of open issues remain to be explored with the users. To do this, a fully functional and secure prototype is necessary.

Results and contributions from this work can be found in the two papers: *Experimental Platform for Usability Testing of Secure Medical Sensor Network Protocols* [2] found on page 125 ff., and *Towards both Usable and Secure Protocols for Medical Sensor Networks* [5] found on page 143 ff.

4.1 Motivation

From the discussion in the chapter 1—and the triangle on figure 1.3 on page 9 in particular—the key observation is that a very complex interplay exists between user interfaces, security, and resources in medical sensor networks. In order for a proposed solution to be convincing, it must be demonstrated in practise that it is fast and easy to use, even with a full level of security and in settings as realistic as possible. Since a one-size-fits-all solution is unlikely, several proposals are going to be tested in many different scenarios.

The previous chapter proposed the BLIG user interface, which still has a number of questions to be worked out with the users (cf. the list on page 50). Furthermore, badly designed security measures are likely to be rejected or circumvented by the users (cf. [16]), therefore different practical solutions should be developed, and then compared and evaluated by the users. This implies that the number of different prototypes will be quite significant, so the goal of the work presented in this chapter is to minimise the amount of effort needed to develop a single prototype.

In this chapter a prototype platform is proposed consisting of a software framework and the sensor and authorisation-node hardware necessary to run it. This platform enables rapid construction of prototypes, implementing different BLIG flavours and various security levels and properties. As a typical user workshop will involve comparing a number of different prototypes, fast changes between the prototypes under test will be necessary in order to do comparisons and decide which one is better—just like an eye examination at an optometrist. The software framework of the platform allows multiple prototypes to be compiled into a single sensor or authorisation-node application. Switching between the different prototypes is simply done by broadcasting a command over the radio to the nodes. Furthermore, due to the uniform environment offered by the platform, technical tests (timing and energy consumption) can easily be performed with detailed comparisons of different BLIG flavours and security schemes.

4.2 Related Work

A vast number of research projects addresses medical sensor network from a usability point of view and some deal with security issues. However, none of the research this author is aware of, addresses the full problem space of the triangle of figure 1.3 on page 9 (all 3 edges and sides) with prototype experiments. The CodeBlue project did initially consider the entire triangle [67] and the need for a flexible prototyping framework. In [68] they present their initial experiments with public-key cryptography (based on elliptic curves), but in more recent work, they appear to have shifted their goals and now focus only on un-secure networks for the emergency context [98]. Details on the CodeBlue project were presented in section 2.3.

Privacy of data sent over wireless networks involves encryption, and to this end, either key generation or key distribution is needed. [57] and [76,104] presents key distribution solutions based on intersecting polynomials. The basic idea of these schemes is that a polynomial is programmed into each node in the network, and each pair of nodes will find a shared key if each node evaluates its polynomial in a point defined by the other node's ID. The schemes are a bit more sophisticated than indicated here, but seems to be very efficient with regards to memory and computation requirements—and will allow millions of nodes with a small amount of keying material (i.e. it is possible to fit it into the limited memory of sensors).

The major problems with this type of key distribution are first of all that it requires the existence of some IT administrator who will pre-load the keying material into each node before it can be used, and the node will only be capable of communicating with other nodes programmed by the same IT administrator. In the case of a major emergency, demanding multiple response teams from different administrative areas, nodes cannot communicate across the administrative borders. The second major problem is the fact that an adversary capable of capturing enough nodes could calculate the master key used to generate the polynomials, and then decrypt all communication. In case this happens, all nodes programmed using this master key must be re-programmed.

Both of these problems disqualifies this kind of key distribution for disposable sensors. It would be a huge burden to program a key into each and every pill or patch used at a hospital, and it would be impossible to control what happens to the sensors after they leave the hospital, including preventing an adversary from capturing them.

The classical way to establish a secure communication over a fast public network is to transmit a smaller amount of secret information over a slow or low-capacity private channel. Take for instance the code-book delivered in a sealed container by a courier used to encrypt telegraph communication in the old days. This methodology was examined by [114, 115] under the designation *multichannel communication*. In a modern context, the idea is to utilise a slow and/or low-capacity auxiliary channel between the communicating parties at the beginning of the communication to create an encryption key for the much faster public network. This channel could be completely private (as in the courier example above), or a channel which the adversary can monitor but cannot manipulate—precluding man-in-the-middle attacks. The latter property is named *data-origin-authenticity*, as the data received on this channel originated from the intended party (the one which the user wanted it to originate from and not from a hidden adversary). Other proposals which generate keys based on an auxiliary channel that may be monitored but not manipulated by adversaries can be found in [11, 45].

Since medical sensors are often placed directly on the body, a number of research projects have explored how to use the body itself as a low-capacity auxiliary channel. An example of this can be found in [33] where weak electrical signals can be exchanged between sensors attached to the body (skin) of the same person. It is unclear how sensitive this method would be to electromagnetic noise from the environment, or the person being connected to electrical instruments or grounded. A different approach was suggested by [14, 85]: since many medical sensors have some kind of perception of the heartbeat of the patient, this shared information can be used to establish a secure connection between the sensors. This is very useful if the sensors have some kind of natural way of sensing the heartbeat—which of course is the case for sensors like ECG, pulse-oximeters and blood pressure meters. However, for some devices this will not work, especially those that do not touch the body, such as scales or intravenous medicine pumps, or those placed at locations where the heartbeat is undetectable. Furthermore, if the sensor does not have a natural sense of the heartbeat—like those mentioned above—using this solution will add to the complexity, energy consumption, size, weight, and price of the sensor. For small and disposable sensors this will not be acceptable.

In [100] the *Resurrecting Duckling security policy model* is proposed. In this model, a factory-fresh device is *imprinted* by the first device it discovers when it first comes to life—just like a newly hatched duckling is imprinted by the first moving thing it sees, believing it is its mother. The discovery will typically be performed on some data-origin authentic auxiliary channel, and during the imprinting phase the duckling device receives a *soul* from the mother, which could be a master certificate or key. Any device which attempts to communicate with the duckling henceforth, will have to present credentials authorised using this key or certificate. Furthermore, the duckling device may be *killed*, returning to the factory-fresh "soul-less" state where it can be imprinted yet again by a (possibly) new mother.

As the Resurrecting Duckling security policy model fits the motivation of the platform proposed here exactly (cf. Question 2 on page 10) and also complements the BLIG user interface perfectly, this security policy model is adopted by the platform.

4.3 Platform Design

In this section, an overview of the platform for prototype development is presented. First, section 4.3.1 presents the hardware components, the operating system, and programming language used to construct the sensors and authorisation-nodes. The platform also offers servers, gateways and a simple network infrastructure. These building blocks and the infrastructure linking it all together are presented in section 4.3.2.

Following the presentation of all the fundamental components of the platform, section 4.3.3 considers the security properties of prototypes built on top of this platform, and finally section 4.3.4 wraps it all up with a presentation of the sensor software architecture.

4.3.1 Hardware and Operating System

In the early WSN years, the University of California at Berkeley developed a range of prototype sensor boards called *motes*, consisting of a small low-power micro-controller (MCU) and a radio transceiver. They also designed an extension to the C programming language by the name *nesC* [78, 36] (where "nes" is short for Network Embedded Systems) and a new operating system called *TinyOS* [105, 44]. Together nesC and TinyOS make it easy to write energy-efficient programs for the MCU, and with the off-the-shelf available motes it is now easy for researchers within the WSN area to concentrate more on the problem at hand, rather than prototype engineering. Today, a number of alternatives exists for hardware as well as operating systems.

Many different MCUs are available from vendors such as Atmel, MicroChip, Infineon and Texas Instruments, and radio transceivers—and radio standards, for that matter—exist in many different flavours as well. Since all developers have their own particular preferred MCU/radio combination, a huge number of different mote designs exist, many of which are on the market as off-the-shelf products. One particular popular combination is the Texas Instrument MSP430 MCU and CC2420 radio transceiver (IEEE 802.15.4 and ZigBee compliant). This pair was chosen for the open hardware design *Telos* developed at Berkeley. Two versions of this design was made: revision A and revision B, with different versions of the MCU and a couple of additional peripherals in the B revision.

On the operating system scene, a few alternatives to TinyOS can be found. *Contiki OS* [70], developed at the Swedish Institute of Computer Science and *MANTIS OS* [27] from the University of Colorado at Boulder are a couple of the most popular alternatives, both written in plain C offering many libraries for rapid prototyping. The advantage of course is less work for the developer, however, the backside of the coin will be less control with power and memory resources, and probably a less efficient result (energy consumption as well as execution time).

For the platform presented in this chapter, TinyOS and the Telos hardware was chosen. The MSP430 MCU has many relevant capabilities, and the CC2420 radio includes a fast cryptographic engine (AES block cipher) which offers hardware acceleration for some of the cryptographic functions presented in the subsequent chapter (section 5.3). Furthermore, the fact that the CC2420 radio is ZigBee compliant implies that experiences from this platform regarding aspects such as energy consumption, and data throughput may feed back into the discussion of whether ZigBee is suited for medical sensors (cf. section 2.1.3).

For the early prototypes described in the previous chapter (cf. figure 3.2 on page 48), the *Tmote Sky* [106] motes sold commercially by the company Mote*iv* (now Sentilla [96]) was used. These motes implemented the Telos rev. B design. However, for the full platform presented in this chapter, the *BSN mote* [21] designed by Dr. Benny Lo at the Imperial College in London was chosen. This mote is a Telos rev. A design, and it was preferred because of its physical size and a modular design which enables easy connection of different boards, as demonstrated on figure 4.1.

The BLIG user interface proposed in the previous chapter requires that 3 LEDs (red, green, blue), a push button, and proximity sensing is available on all sensors. On authorisation-nodes, LEDs are not necessary, but instead a buzzer must be present, along with some means for logging in and out.

The Telos rev. A design (and thus the BSN motes) already have the three LEDs available, but the remaining devices must be provided by add-on modules.

A fingerprint reader module was constructed for authorisation-node login, and the details of this design can be found in appendix B. This module is always placed on top of the stack of boards and offers login by swiping a finger—and ordinary button functionality by simply tapping the reader surface. The board is shown on figure 4.2. This module is only used on authorisation-nodes, so it does not matter that it covers the LEDs of the BSN board, as they are only used by sensors.

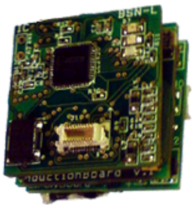


Figure 4.1: The BSN mote. Many modules with different functions can be stacked together due to the common bus, passing through each circuit board (all having a male connector on top and a female connector on the bottom side). This picture shows three modules. The upper-most board is the BSN mote, the Induction Board is in the middle, and at the bottom, a board with a rechargeable battery.

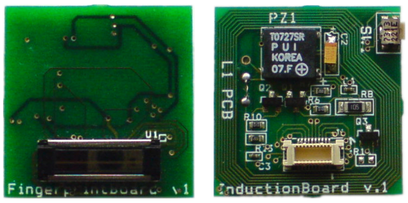


Figure 4.2: The fingerprint reader and the Induction Board.

In order to achieve security by imprinting using multi-channel communication, the proximity detection is realised by a *short-range communication channel*. When a node wishes to detect other nodes in its proximity, it will begin a transmission on this channel. If another node receives this transmission, it means that it is in range.

As the short-range communication should only be possible over a short distance (less than half a meter or so), an ordinary radio is not suited for this purpose. Many technologies exists: light, sound, and electromagnetic induction are just some examples, and each technology has advantages and disadvantages. For instance if visible light or audible sound were used, it would be difficult for an adversary to inject signals undetected by the user, but on the other hand the light or sound of non-adversarial communication may be annoying to the user. These solutions would also require expensive hardware and much energy.

For the platform presented in this chapter, electromagnetic induction is used for short-range communication. The *Induction Board* shown on figure 4.2 was designed to provide all the functions necessary for a BSN mote based sensor used with the BLIG user interface: short-range communication based on electromagnetic induction, a push button and a buzzer (used only with the authorisation-nodes). The design of this module is presented in appendix A.

4.3.2 Platform Building Blocks

The platform is built using a number of different devices shown on figure 4.3. A detailed description of these devices are given in the paper on page 125.

Sensor A mote representing a medical sensor or actuator (cf. section 3.3.5). The device *may* include a real medical sensor, but since the primary objective of the prototypes would be to test usability, having a physical sensor delivering real data is rarely important.

Authorisation-node A mote equipped with a fingerprint reader and a buzzer, representing a clinician's name-tag as described in section 3.3.3. Can be used for installing and removing sensors as described in section 3.3.5.

Extended authorisation-node A mote connected to a PDA, which can be used exactly like a normal authorisation-node for installing and removing sensors. Furthermore, it can be used for patient property changes, such as inputting the name or social security number (cf. section 3.3.5).

Display A mote connected to a laptop, which assumes the role of a local display (e.g. a bed-side display).

Gateway A simple (dumb) bridge between the LAN/Internet and the sensor network (IEEE 802.15.4).

Server and database Currently the main server and database purpose is to control and log the events at a user test. From a small server application the current prototype is selected and configured by broadcasting commands to all motes; data about user interactions and usage patterns is gathered from the motes for later analysis and stored in a database table. In the future (when user applications are introduced—see below), the server will also play the role as a hospital server, collecting sensor readings and simulating patient transfers (cf. section 3.3.5).

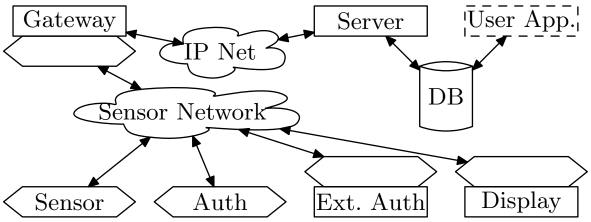


Figure 4.3: Platform building blocks—hexagons are devices realised using motes while rectangles are devices realised using PCs or PDAs (User applications remain future work).

User applications One of the future work directions (cf. sections 7.2.1–7.2.2) is to experiment with integration of medical sensors in hospital and emergency IT infrastructures. For such scenarios, the server will act as a hospital server, and user applications (e.g. electronic health record and journals) can retrieve the data from the database and visualise it for the user.

The sensor network in the current platform is single-hop only—no routing takes place. This is acceptable for user workshops, but of course in practise, routing in the sensor network may be necessary in locations where the gateway density is low—for instance at emergencies. In future versions of the platform, a networking layer featuring routing of sensor network packets may be added. At an early stage of this project, Doina Bucur proposed a routing protocol designed especially for the architecture of the platform [22].

4.3.3 Overall Security Framework

In section 2.2.4, the security considerations for sensor data in health-care settings were presented including the relevant legislation and related research. Overviews over the security issues in medical sensor networks were also published in [18] and [41]. According to these studies, the following groups of adversaries must be considered:

Clinicians Humans make mistakes and clinicians make no exception to this rule. However, clinicians who made a mistake may have an incentive to try to cover it up or blame it on someone else.

Patients Drug addict patients trying to manipulate their own medical record in order to get unnecessary drug prescriptions, and patients attempting to commit insurance fraud by counterfeiting their own medical record, or try to alter the records in order to sue the hospital with a malpractice claim.

Third parties Adversaries trying to gain unauthorised access to medical information about patients. This could for instance be tabloid journalists trying to spy on some celebrity, or it could be a sinister person trying to learn anything about the patients that could be used for a blackmail attempt.

The key point here is that adversaries are not just outsiders. Users (clinicians and patients) of the sensors may be malicious as well, and this possibility must be considered.

As a natural consequence of the short-range communication described in section 4.3.1, the class of third party adversaries can be divided into *internal* and *external* adversaries based on their capabilities:

External adversary This adversary has no physical access to the patient, whose sensor readings he is trying to access. The adversary is capable of eavesdropping and packet-injection on the wireless link, but is out of range for the short-range communication.

Internal adversary Due to the physical access to the patient—and thus the sensor nodes on this patient, an internal adversary will have access to the short-range communication channel, where he can launch for instance man-in-the-middle attacks.

Of course adversarial clinicians or patients always belong to the "internal" category.

This partitioning between external and internal adversaries is very useful. The main design principle of this work (stated in section 3.3.1) is that the new technology should not be inferior to the existing—and with the existing (cabled) technology, external parties have absolutely no access, while internal parties may tamper with equipment, install wiretaps and so on. Thus, the physical connection between a sensor and a display provides absolute security against external adversaries but only limited security against tampering. An equivalent level of security can be realised by using only strongly encrypted and authenticated communication on the wireless radio channel, since security against threats in a publicly accessible network is well-studied and has a number of widely accepted solutions.

According to (Danish) regulations [89], clinicians are required to document their actions. Each sensor and authorisation-node has a (reasonably large circular) buffer dedicated to log messages, where a report about all configuration changes (cf. section 3.3.5) this node has participated in will be stored. By automatically collecting and logging these reports, the sensor network can save the clinicians a lot of work, and with the authentication provided by the participating authorisation-nodes, log messages can be automatically signed by the clinical user.

One way to protect the sensor groups against internal adversaries will be to block all attempts to perform a configuration change without proper user authentication, and make sure that the user identity is included in the log reports. Then malicious activity can always be traced to the responsible user. The challenge is to ensure that log entries are not forged by clever adversaries, and this will be one of the topics treated by the following chapter.

Prior to installing a sensor, the user must log onto her authorisation-node. Exactly how this is done and for how long time a login is valid, will depend on the circumstances and a pragmatic assessment. For instance, at an operating theatre a user should not have to touch the authorisation-node; instead keeping the login active for the entire working day could be appropriate, as the risk of loosing the authorisation-node is low. On the other hand, at an open clinic, due to the risk of authorisation-node theft, a continuous login may not be used at all, and instead a fingerprint verification would be required each time a sensor is installed.

Trusting Sensors

The concept of imprinting used by the Resurrecting Duckling security policy (cf. section 4.2) matches the use of disposable sensors, which cannot be preprogrammed with any information about the environment they will be used in. When a sensor is powered on, it will act like the newly hatched duckling and identify its "mother"—which in this case could be the clinical user, or to be more precise: her authorisation-node. When reusable sensors are reset, any previously accumulated knowledge used when monitoring earlier patients, will be erased, and they will again act as factory fresh nodes.

As the sensor is imprinted, it receives a soul from its mother. This soul may contain directions on how to connect to a server for sensor data delivery including the credentials to use for this delivery. These directions must be prepared by the IT staff managing the server.

Forming groups of sensors on a patient (as required by BLIG) is performed by adding one sensor at a time. The mother of the first sensor installed on a patient will be the authorisation-node of the clinician installing the sensor, and the directions which the IT staff has programmed into the authorisation-node will constitute the soul of the sensor (and hence the entire group of sensors).

As a subsequent sensor is installed on the patient, the *group* will be the mother of the new sensor and hand over the common soul of the group to the new sensor. Furthermore, the new sensor is only accepted into the group if the authorisation-node used in the install process can be *authorised* by the soul of the group (hence the name "authorisation-node"). If for instance the soul of the group only accepts authorisation-nodes from hospital X (probably because the clinician who installed the first sensor works at this hospital), any subsequent node must be added using authorisation-nodes issued by hospital X as well.

Logs (as explained above), user authorisation, and proximity of nodes in the BLIG install protocol can now be combined to form an argument for trusting sensors. The fact that the BLIG install procedure succeeded is a proof that the nodes involved were located at the same place at the same time (within a few seconds), and therefore that the sensors are located on the same patient, if the clinician performing the installation can be trusted. Every time an authorisation-node participates in a successful install procedure, it will produce an audible beep (cf. section 3.3.3) to ensure that its owner is aware of the procedure (an adversary will not be capable of cheating a clinician's authorisation-node without a high probability of being discovered). Each time a sensor is added to an existing network, the clinician (represented by her authorisation-node) is authorised according to the soul of the group, and a log message is generated. Finally, a sensor can combine the log message of its own installation with those of all its ancestors, forming a genealogy all the way back to the original first sensor. This genealogy can then be presented as a proof that the sensor is "purebred" and therefore can be trusted.

If the conditions for accepting clinicians must be changed (for instance the patient is transferred to a different hospital), the soul of the sensor group can be updated. To accommodate this, the property configuration change (introduced in section 3.3.5) is used.

One of the great advantages of this model of trust is that it can be established completely off-line. No connection to any server is necessary at any time to establish communication and trust between the nodes. Only local communication between the nodes is required.

Another advantage of this model is, that only sensor nodes which have been properly installed into the group, are trusted by that particular group and allowed to communicate with group members. This implies that groups are completely disjoint, so an adversary cannot use access to one group to gain access to another, and he can *at most* get access to data about the patient of the group he breaks directly into. Assuming that the strong encryption used on the wireless channel cannot be broken by any adversary, the adversary who successfully breaks into a sensor group must be internal, which means that he must have had physical access to the patient—and group of sensors—at some point in time. But even today, such an adversary may install surveillance equipment or rogue sensors, so this does not contradict the main design principle (cf. section 3.3.1).

The Rule of Priority Reversal (cf. section 2.2.4) states that a clinician must always be able to gain access, if she finds it necessary. If the clinician is not authorised (or perhaps if her authorisation-node is currently unavailable), this can be supported using a local display by simply turning all sensors off (killing the ducklings) and reinstalling them. For this purpose all displays must include a dummy authorisation-node, which is self-authorising, i.e. *not* affiliated with any Certificate Authority (see below), not even the hospital it belongs to.[1] This issue, including an alternative solution is discussed further in section 4.4.

Certificate Authorities

Three different types of actors participate in the configuration change protocols: the sensors, the authorisation-node, and one or more *Certificate Authorities* (CAs) delegating trust to the authorisation-node. A CA issues a signed certificate of authorisation to the authorisation-node. This certifi-

[1] A side-effect of this is that anyone may buy a sensor and a display at the local drugstore, connect the two devices using BLIG *without* an authorisation-node. Hence, all sensors can be used by clinical as well as non-clinical users.

cate is stored in the memory of these nodes (this step is performed by the local IT administrator before the authorisation-node can be used).

The relationship between sensor groups, CAs, and authorisation-nodes can be many-to-many-to-many, as authorisation may be handled at different levels of granularity. For instance, the hospital may have a global CA, which is used to issue certificates to all authorisation-nodes, each ward may have a local CA, and each team—and perhaps even each clinician—may have one. An authorisation-node will then be programmed with a number of different certificates according to the owner's affiliations—hospital(s), ward(s), and teams(s). On the other hand, the patient will be admitted to a ward and treated by one or more teams—perhaps at different wards or even different hospitals. The soul of the sensor group will keep a list of all the CAs that may access the patient's data, and when an authorisation-node is used for a configuration change, it will choose a certificate that was signed by one of the CAs accepted by the soul of the sensor group.

If a CA knows the necessary key to authorise clinicians to perform configuration changes on a group, it can use the same key to perform such configuration changes directly (assuming it has a network connection to that particular group). This is exactly the mechanism needed to establish connections between sensor groups and servers subscribing to log and sensor data, and in order to keep the platform simple, the CA keys used to sign authorisation-node certificates can also be used as server keys.

Private and Shared Data

Sensor nodes contains the information found in table 4.1. In addition to an ID of the node, the sensor will keep a full copy of its genealogy and a block of data shared among all sensors in a group. Apart from the `group-id` and `group-key`, these values are kept synchronised among all group members by the platform—using the `group-key` to secure the wireless communication needed for this synchronisation.

The shared group data comprises the following elements:

`group-id` This is the public ID of the group (generated when the group was created).

`group-key` The key used to secure internal communication among nodes in the group. A node is a member of the group if and only if it knows the `group-key`.

`group-time` The common time used to synchronise BLIG group blinking and radio duty-cycling among group nodes.

`ca-pk[]` This is a list of CA public keys of the trusted CAs of this group. Authorisation-nodes presenting valid certificates signed by one of these CAs are authorised to perform configuration changes, and remote

Table 4.1: Data kept in each node and shared among nodes in a group.

`n-id`	ID of the node.
`n-genealogy`	Ancestors and their certificates.
`group-id`	Public ID of the group.
`group-key`	Secret key used for encryption and authentication.
`group-time`	Common clock.
`ca-pk[]`	List of CAs trusted by this group.
`meta`	Information about the patient.

Table 4.2: Data kept in an authorisation-node.

`a-id`	ID of the node.
`a-cert`	A signature of a CA on `a-id`.
`ca-pk[]`	Public key of one or more CAs.

servers communicating using one of these keys are authorised to perform remote configuration changes and subscribe to log and sensor data.

`meta` Contains various information about the patient such as his name, Social Security Number, blood type, and little more. The amount of data stored here must be limited to the most vital information, as this is replicated on all nodes—even those with limited memory. Extra data (such as complete medical records) should not be stored here, but only in nodes with plenty of memory. The `meta` storage also saves as many log entries as possible.

Authorisation-nodes are prepared by the hospital's IT department (or a similar certificate authority) with the information found in table 4.2. A node will have an ID, at least one CA public key, and one or more certificates proving that `a-id` is a valid authorisation-node signed by one of the CAs. The list of keys found in `ca-pk[]` are used for imprinting of newly-formed sensor groups.

4.3.4 Software Architecture

As explained in the motivation of this chapter, the platform was designed so that several prototypes can be compiled into a single sensor node, with prototype switching and configuration controlled from a central server. The prototype code distinguishing one prototype from others has been encapsulated in modules with fixed interfaces to the remaining platform code.

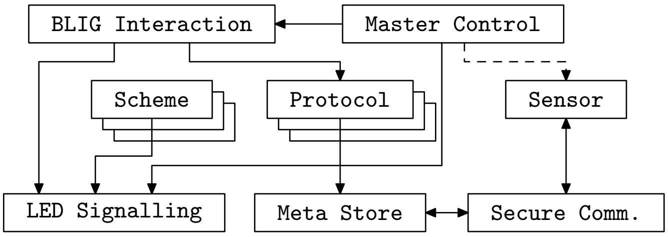


Figure 4.4: The software architecture of sensors.

Two such "pluggable interfaces" exists: `Scheme`, dealing with the BLIG issues for sensors (group blinking look-and-feel), and `Protocol`, dealing with imprinting, trust, authentication, and key generation for both sensors *and* authorisation-nodes.

A number of different `Protocol` and `Scheme` implementations may be compiled into a single sensor and authorisation-node application. All nodes must use the same `Protocol` implementation at any time (otherwise communication will fail), and the central server may broadcast a command ordering all nodes to switch to another `Protocol` and perform a reset. A `Scheme` defines the group blinking pattern as well as the presentation of this blinking pattern (e.g. timeouts, and steady light vs. flashing, cf. section 3.3.6). Of course the blinking patterns must agree among sensors, however sensors with different ways of presenting this common pattern can coexist—for instance a powerful sensor may implement steady blinking with a long timeout while a sensor with a small battery uses flashing and a much shorter timeout. Therefore, the central server may either broadcast commands instructing all sensors to use the same `Scheme`, or it may instruct each individual sensor to use a certain `Scheme`.

The software architecture of sensor nodes of this platform is illustrated in figure 4.4. Each box represents a subsystem (typically one or more TinyOS components) and the arrows describe the interface connections and communication between these subsystems. The software architecture of authorisation-nodes is similar, with the only exceptions that the `Sensor`, `LED Signalling`, and `Scheme` boxes are missing, and instead a login and buzzer subsystem handles the fingerprint authentication and buzzer sounds.

The following is a brief outline of the function of the individual components. Further details are provided by the two papers on page 125 ff. and 143 ff.

The `BLIG Interaction` subsystem implements the BLIG user interface interaction described by figure 3.1 and uses the LED signals listed on page 43 implemented by `LED Signalling` (except the group blinking, which is implemented by the current `Scheme` as explained above).

`Protocol` defines the communication between the nodes using the short-range communication and radio during a configuration change. The outcome of configuration changes are keys and log messages which will be stored in the `Meta Store` (along with patient data, such as name and social security number). Details will be presented in section 5.2—particularly in the box on page 72.

The `Secure Communication` subsystem implements symmetrically encrypted and authenticated radio communication. Sensor data are delivered to displays and servers using this subsystem. Keys are fetched from the `Meta Store` where they are all stored (the symmetric keys are generated by the `Protocol`). The `Secure Communication` and `Meta Store` subsystems collaborate on a number of different tasks:

- Group/patient information, time, and logs are synchronised among all members of a group using dissemination and time-sync protocols. For this job, protocols from the TinyOS libraries are used, as dissemination and time synchronisation are well-studied subjects from sensor network research [60, 38].
- Managing subscribers of sensor and log data (and the session keys used).
- Configuration changes initiated remotely by a server (for instance transferring the patient to a different hospital, cf. section 3.3.5). The current `Protocol` may be used to validate signatures.

The `Master Control` subsystem manages the platform during user tests, receiving and executing the prototype selection commands issued from the server regarding the current `Scheme` and `Protocol` choice. The `Master Control` will be instructed to disable itself when technical tests (timing and energy consumption) are performed, as it would otherwise influence the measurements.

4.4 Conclusion and Future Work

In this chapter a platform consisting of software libraries and hardware for medical sensor network prototype development and test was presented. The platform was designed for technical (timing and energy consumption) and usability evaluations of the configuration change protocols and user interfaces of the prototypes. The platform supports co-existence of several prototype implementations on the sensor hardware, in order to allow fast switching between prototype implementations during a usability test.

The platform software implements the BLIG user interaction, which was defined by figure 3.1 on page 44, and all prototypes must adhere to this specification. The platform does *not* specify when transitions between the two states “Group blink” and “No LED activity” must happen, nor does it specify what the group blinking should look like. These issues are left to be defined by the concrete prototype implementation. The platform hardware includes all the peripheral components required by BLIG: LEDs (red, green, blue), button, proximity detector (the short-range communication), a fingerprint reader, and a buzzer.

During sensor install and when the sensor is turned off, the platform lights the LEDs and communicates on the short-range communication channel. This total round-trip requires less than 1 J of energy and for a disposable sensor this happens only once in its entire lifetime (for a reusable sensor with a rechargeable battery, it is fair to assume that it happens only once for each time it is recharged).

When the sensor has joined a group, the platform software will keep the group clock and the group properties stored in the `Meta Store` synchronised among all group members. These updates only require the radio to be on for very short durations at regular intervals, and require no heavy calculations. With the current hardware implementation, the short-range communication board must be in receive mode most of the time, and this draws a constant current of 2 mA. As explained below this will be fixed in future work, so that no current will be needed to monitor the short-range channel.

To conclude, the platform hardware and software library by itself requires less than 1 J of energy plus a small amount of radio communication during a sensor's lifetime—a requirement which should be acceptable with even the smallest battery. Concrete prototype implementations will define the conditions for the BLIG group blinking and the protocol used for configuration changes, and of course these implementations will add some energy requirements on top of this.

By design, the platform hardware and software offers the following features (regardless of the concrete prototype implementation):

No pre-programming. When a sensor node is powered on, it will be imprinted by the first peer it encounters on the short-range communication channel.

Automatic logging. All configuration changes are logged with the responsible (clinical) user's signature. The platform disseminates these log messages among the group members, and if a connection is available to the network, the log messages can automatically be handed over to the patient's electronic records.

Off-line operation. The platform was designed such that local operation requires no network access. When the network is unavailable, a local display can always be added to access the sensors locally.

Supporting the Rule of Priority Reversal (cf. section 2.2.4). It is a requirement, that all displays must implement an authorisation-node which is self-authorised (i.e. the display is its own certificate authority). It is always possible to turn a sensor off, and by turning it on again, it can be imprinted by the display and will begin delivering its readings there. Since only the sensor and a display is needed (i.e. no authorisation node), anyone can do this at any time—but if the hospital server is actively monitoring the sensors, the action will probably be discovered immediately and an alarm could be triggered.

Roaming. If the patient is moved to a different unit (ward or hospital), the certificate authority key (`ca-pk`) of the new unit is added to the sensor group's list of accepted servers. This can be done locally by a patient property change, but can also be performed remotely by one of the currently accepted servers. For instance, if a patient is currently admitted to ward X and must be transferred to ward Y, as a doctor or nurse records the transfer in the patient's records, the server of ward X will automatically contact the sensor group and configure it to accept the server of ward Y.

As mentioned earlier, future work includes a redesign of the Induction Board for short-range communication to add passive (zero-current) monitoring of incoming signals. If a signal is detected, this circuit will wake up the micro-controller, which in turn will power up the receiver to receive the packet.

The current Induction Board has a fixed range, but from a usability point of view, it would be interesting to study the optimal "proximity" range. Therefore, a future version of this board should include a way to adjust the range.

Electromagnetic induction was originally chosen as the method for short-range communication because the necessary circuit is quick and easy to build. However, many other methods could be used instead, and an interesting direction for future work would be to examine and compare the alternative methods with respect to for instance price, size, energy consumption, and security properties. For example, what would the price and increased complexity be, if the laws of physics (speed of light/sound or the near-field effects between the antennas) are used to guarantee the maximum range of the communication interface? Could the (long range) radio hardware be used also for the short-range communication—with the differences partly implemented in software (a software-defined radio), saving hardware costs at the expense of more complex software?

Sensors like for instance bathroom scales or blood pressure meters are often used with many patients—a single measurement for one patient at a time. Adding these *promiscuous sensors* to the sensor groups of all patients who use them would potentially grant these sensors full access to the private data of all these patients. This would make a promiscuous sensor a very appealing point of adversarial attacks, as an adversary would gain a lot more by tampering with a bathroom scale than for instance a disposable sensor—and the former would probably be a lot easier to accomplish. Therefore, a future work path could be to introduce a new class of *temporary sensors.* A temporary sensor would be temporarily "installed" using BLIG for a single measurement only, and will never fully join the group (i.e. it presents the group blinking but actually never learns the `group-key` or any other private group data). Instead, it connects to one of the sensors of the group and delivers the measurement to this node. This sensor node will act as a proxy and forward the measurement to the relevant displays or servers.

Lastly, the way to deal with the Priority Reversal rule as mentioned above may be improved in future work. When an unauthorised user (clinician) needs access to the sensors, the current approach is to shut off all sensors on the patient and start a new group using a local display. While this approach does grant the user the necessary access, the cost of destroying the original sensor group and cutting off the connection to hospital servers is quite high. Perhaps a better solution would be to allow temporary unauthorised connections between already installed sensors and local displays. This would preserve the connection between sensors and servers, and could instead activate an alarm to notify the regular staff of the unusual action which is being performed. The full impact on the security of the system by this extension would have to be studied further, though.

Chapter 5

Secure Group Configuration

The subject of this chapter is protocols implementing the pluggable interface of the prototype platform described in the previous chapter. The purpose of such a protocol is to define the communication taking place on the radio and short-range communication link between nodes involved in a local configuration change (cf. section 3.3.5). Two very different protocols are presented along with a discussion of their security properties.

The main results and contributions of this work were presented in the paper *Secure Group Formation Protocol for a Medical Sensor Network Prototype* [4] found on page 137 ff. Also, the paper *Towards both Usable and Secure Protocols for Medical Sensor Networks* [5] found on page 143 ff. contains results of this work.

5.1 Motivation

In current (cable-based) technology, accessing sensor data and configuring sensors can only be performed by an individual who has physical access to the patient. An adversary attempting to access the patient or the sensors (e.g. a nosy journalist) can be blocked by physical access restrictions—locking a door or even posting a sentry. Restricting individuals with physical access to the patient (including the patient himself) from accessing the sensor data or manipulating sensors is much more difficult, and pragmatic countermeasures are chosen when a concrete threat exists.

As explained in section 4.3.3, the introduction of the short-range communication channel encourages the distinction between internal and external adversaries, where an external adversary has only limited access to the short-range channel. Recalling the main design principle from section 3.3.1 (that the new technology should not be inferior to the existing technology), this distinction is extremely handy and motivates the two-fold purpose of the work presented in this chapter.

5.1.1 External Security

First of all, by designing the security mechanisms appropriately, external adversaries should be denied any access. This is trivial if the external adversary has absolutely no chance of eavesdropping on the short-range communication, injecting packets, or manipulating data—then the transmissions on the radio link can be secured using keys exchanged on the short-range channel. In practise, however, a maximum-range guarantee may be difficult (and expensive) to implement in the short-range communication hardware, as there may exist some strategy which an external adversary could use to gain some access to the short-range communication with a non-negligible probability. For instance, a TV remote control could be regarded as a short-range communication device, as it would only work within a few metres of the TV under normal circumstances. However, by replacing the infrared LED by a powerful infrared laser aimed carefully at the receiver (and perhaps a telescope for communication in the opposite direction), the TV may be controlled from a long distance (perhaps several kilometres). Similar range-extending modification may be feasible for most—if not all—inexpensive short-range communication technologies.

The challenge here is to find a balance between the short-range channel hardware requirements (price, size and energy) and the key-exchange software requirements (time and energy). Therefore, the first purpose of this work is to propose and compare a number of different key-exchange protocols making different assumptions about the short-range channel hardware—e.g. eavesdropping possible or impossible; packet injection possible, impossible, or unlikely. Combined with an analysis of the cost of realising these different assumptions in hardware (which is outside the scope of this dissertation), a well-balanced solution can be developed.

5.1.2 Internal Security

The second purpose of the work presented in this chapter relates to countermeasures against threats from individuals with access to the patient (including the patient himself). This is more complicated, especially in light of the requirement of the "Rule of Priority Reversal" (cf. section 2.2.4), stating that a clinician must be granted access if she insists that it is necessary. As stated above, the current practise is pragmatic in the sense that if a concrete threat exists, extra time and resources will be allocated. For instance, if a patient is a drug addict or suicidal, he will be monitored more closely, and if irregularities are suspected among the staff, logs could be carefully checked.

Due to the Priority Reversal Rule an insisting clinician must always be granted access. With current technology (thermometers, ECG monitors etc.) this is possible even if the clinician is off-duty or has lost her name tag. As the new technology should not be more restrictive than existing technology (cf. the design principle in section 3.3.1), access restrictions for local configuration changes should not be the foundation of all security against internal adversaries. A better approach would be to keep a log of all the configuration changes. This way, at least illegal actions can be discovered with high probability. This approach may be supplemented with (automatic) log auditing—perhaps capable of alarming the staff in real-time if irregularities are detected. If users know this is done, and that illegal actions would be discovered with high probability (perhaps even with evidence pointing toward the culprit), it would discourage this type of adversarial behaviour.

The challenge to be addressed by the protocols in the work presented here, is how to ensure that log entries are correct, i.e. to protect the logs against accidental or intentional alterations. Log entries are the result of configuration changes (and thereby the result of a successful protocol run), so the protocol should ensure that a log entry is made, and that it accurately reflects an action which took place. As the technically skilled internal adversary would be capable of full access to the short-range communication channel and node tampering, cryptographic protocols must be used to protect the log integrity. However, cryptographic protocols tend to be slow and energy demanding on sensor hardware, so the challenge is to find a proper balance between an acceptable level of security against internal adversaries, processing speed (with a resulting user interface latency), and energy consumption.

In summary, the second purpose of the work presented in this chapter is to experiment with different protocol proposals in order to determine what kind of security can be realised, and at what cost—in terms of time, energy, and increased software complexity (hardware price).

5.2 Install a New Sensor on a Patient

Although the local configuration changes (cf. section 3.3.5) includes removing sensors and changing patient properties, this dissertation considers only the part of a protocol dealing with installing a sensor on a patient. The performance of the install part is much more critical, as sensors must sometimes be installed in a hurry in life-critical situations—while removing a sensor or inputting the name of the patient would rarely be a priority under such circumstances. Furthermore, the remove and property change parts of a protocol are less complex than the install part.

Protocols are subject to the BLIG user interface, and must be able to conform to the user interaction patterns defined by BLIG—including running in real-time and responding quickly to user actions. Figure 5.1 shows a simplified excerpt of figure 3.1 with the states and transitions of the BLIG user interaction relevant for the install-part of a protocol. The BLIG user interface install procedure was defined in section 3.3.5.

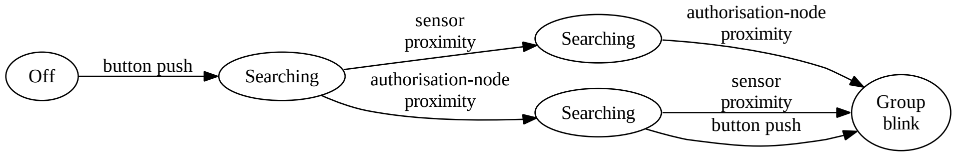


Figure 5.1: BLIG user interaction for sensor installation.

Two fundamentally different installation procedures exist: *bootstrapping* and *association*. Bootstrapping is the procedure of installing the first sensor on a patient, forming a new sensor group; and association is the procedure of adding another sensor to an existing sensor group. In figure 5.1, bootstrapping is performed by the bottom path (button push, authorisation-node proximity, button push), whereas association is realised by the upper two.

For each node proximity edge indicated in the graph, exactly one data packet is transmitted on the short-range communication channel from the new sensor to the node in proximity, optionally followed by one packet in the opposite direction. Apart from these packet exchanges, all communication defined by the protocol must take place on the radio channel. Of course radio usage should be minimised as well, due to energy consumption.

The intended outcome of the protocol was specified in section 4.3.4, and all the constraints imposed on the protocol can be summarised as:

Protocol implementation constraints:

The implementation (including all communication on the short-range channel) is subject to the BLIG user interface as described above, and must run in real-time, with the following outcome:

`group-id` The identity of the group of sensors, cf. section 4.3.3.

`group-key` Key used for internal communication within the group of sensors, cf. section 4.3.3.

`ca-pk[]` list of keys of trusted servers, cf. section 4.3.3.

Log message created by the authorisation-node participating in the protocol. It is up to the protocol to define whether this log message should be digitally signed or not—and where and when this signature should be verified.

5.3 Library of Cryptographic Functions

A library was developed to support protocol construction, featuring the following general cryptographic methods—all built on top of the AES-128 block cipher:

- Transmitting and receiving encrypted and authenticated data packets over the radio using symmetric-key cryptography.

- A cryptographically secure hash function.

- A cryptographically secure pseudo-random number generator (based on the AES block cipher in CTR mode).

As explained in the Energy Bucket paper [3] on page 129, the AES block cipher can be computed much faster using the hardware AES processor on the CC2420 radio chip [24] (available on both Tmote Sky and BSN motes) compared to a software implementation. However, the TinyOS driver for the CC2420 radio was not designed to use this AES processor—in particular, the driver was not even designed for the necessary control over the radio chip power level.

Therefore, a new TinyOS CC2420 radio driver had to be developed, adding support for transmitting and receiving secure packets according to the full IEEE 802.15.4 standard [48, 24], and addressing some of the issues and recommendations found in the thorough analysis of the security of this standard provided by [93]. Furthermore, this revised radio driver exposes a TinyOS interface to allow applications easy direct access to the CC2420 hardware AES processor. The hash function and pseudo-random number generator described above was built on top of this interface. Further details about this implementation is given in the paper [4] found on page 137 ff.

5.4 `NaiveProtocol`: a Simple Protocol

The first protocol, ***`NaiveProtocol`***, which is little more than a stub implementation, serves as a baseline for the technical and user experience evaluations of more advanced protocols. The `NaiveProtocol` offers minimal energy consumption and the best possible user experience (no latency due to lengthy calculations); however, the protocol offers only limited security features.

The security of the `NaiveProtocol` is based on the following naive assumptions (hence the name):

- The short-range communication channel cannot be accessed by an external adversary—neither eavesdropping nor packet injection or modification is possible for any out-of-range party.

- Internal parties are always trusted, which means that security measures against threats from internal adversaries are not considered.

- A user will always discover and discard tampered nodes, so an (external) adversary cannot plant a corrupted sensor node on a patient.

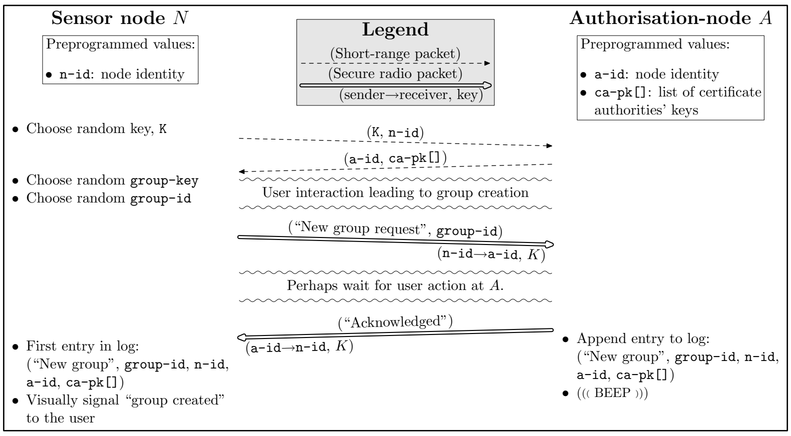

Sensor node N
Preprogrammed values:
• n-id: node identity
Legend
(Short-range packet)
(Secure radio packet)
(sender→receiver, key)
Authorisation-node A
Preprogrammed values:
• a-id: node identity
• ca-pk[]: list of certificate authorities' keys
• Choose random key, K
(K, n-id)
(a-id, ca-pk[])
• Choose random group-key
• Choose random group-id
User interaction leading to group creation
("New group request", group-id)
(n-id→a-id, K)
Perhaps wait for user action at A.
("Acknowledged")
(a-id→n-id, K)
• First entry in log: ("New group", group-id, n-id, a-id, ca-pk[])
• Visually signal "group created" to the user
• Append entry to log: ("New group", group-id, n-id, a-id, ca-pk[])
• (((BEEP)))

Figure 5.2: Bootstrapping protocol for the `NaiveProtocol`.

Since all internal parties are trusted and external parties are not capable of eavesdropping or manipulating the short-range communication channel, or tricking an internal party into installing a corrupted sensor, the symmetric key to be used for radio communication can be transmitted in clear text on the short-range channel.

The details of the bootstrapping part of this protocol is shown in figure 5.2. Here N is the sensor node forming a new group and A is the authorisation-node. The association protocol is similar, except that the sensor node already attached to the patient will hand over the existing `group-key`, `group-id`, and `ca-pk[]` to N in a final encrypted radio packet, after receiving the acknowledgement from A.

In this protocol proposal, log messages are not signed by A, as internal parties are trusted not to forge the logs. A simple signature scheme could be added with only minimal overhead (a few extra bytes in the last "Acknowledged" message), by having some shared secret between A and a hospital server. A could concatenate this secret to the log message, compute the secure hash function (described in the previous section), and send the result to N in the last message. Although only the hospital server would be capable of verifying this signature and detect log-forge attempts, this signature would make it more difficult for an internal adversary to manipulate the logs undetected.

5.5 `AlphaProtocol`: a Secure Protocol

Although the `NaiveProtocol` may be useful in situations where speed is essential (such as emergencies), its assumptions about the honesty of people with direct access to the patient are too lenient for general use. Furthermore, as explained in the motivation, the assumption that external parties can never access the short-range communication channel will probably be unrealistic, due to the low-cost restrictions that apply to the hardware.

Recalling the discussion in section 2.2.4, clinicians treating a patient—and even the patient himself—may exhibit adversarial behaviour [41, 18]. Hence, even communication on the short-range channel cannot necessarily be trusted.

In this section a much more secure `AlphaProtocol` is proposed, based on the following more realistic set of assumptions:

- *External* parties may eavesdrop but cannot inject or modify packets on the short-range communication channel. This channel implements *data-origin-authenticity* (cf. section 4.2 and [115]), which means that data originated from a source chosen by the user—but with no guarantee that the user herself is honest and chose a non-compromised source.
- An honest user will always discover and discard tampered nodes, so an *external* adversary cannot plant a corrupted sensor node on a patient (without collaborating with an *internal* adversary).
- A user will not commit a dishonest act if this can easily be discovered and traced back to her.
- An honest user will carefully protect her authorisation-node, PDA, and other devices she is logged onto—preventing others from stealing her identity.

Notice that the three last assumptions must also be used to argue for the security of current technology.

To construct this protocol, *Public-Key Cryptography* (PKC) methods were used. While PKC is used extensively on more powerful wireless platforms, such as cell phones, the use of these methods on low-power sensors was until very recently considered impractical.

5.5.1 Related Work

In a paper from 2004, [110] describes an implementation of RSA for sensor networks. This implementation was built on a MICA mote, which has capabilities roughly comparable to the hardware platform used in the work presented here. The small exponent version of RSA ($e = 3$ as exponent in all public keys) is used, and only the public operations are implemented on the sensors, spending 14.5 seconds for a single modular exponentiation. The time used for private exponentiation (with large exponents) was not measured, however an extrapolation predicted tens of minutes. Key generation is not even considered on sensor nodes. The paper also reports of a Diffie-Hellman key exchange implementation over the discrete logarithm problem, spending up to 3 minutes.

An *Elliptic Curve Cryptography* (ECC) based Diffie-Hellman key exchange implementation presented in [68] completed in 34 seconds—also running on the MICA mote in 2004.

A more recent ECC implementation, *TinyECC* presented in [63] (published 2008) performs much better. The basic ECC point multiplication is computed in less than 2 seconds.[1] An even more impressive implementation is the *NanoECC*, presented in [103], claiming to perform the same computation in less than one second. Both of these implementations are running on the same micro-controller, that is used on the platform presented in this dissertation.

5.5.2 Protocol Outline

In this section a brief overview over the `AlphaProtocol` is offered. A much more detailed description was published in the paper *Secure Group Formation Protocol for a Medical Sensor Network Prototype* [4] found on page 137 ff., and this section will only sketch the key points.

For the PKC computations necessary in this protocol, a library of a few basic functions was developed. The library is originally based on the TinyECC implementation described above, but was extensively modified to fit the requirements of the prototype platform. Rather than using the *Elliptic Curve Digital Signature Algorithm* (ECDSA), a signature scheme based on the Schnorr protocol [94] was implemented, as it can be optimised to run much more efficiently.

The primary line-of-defence against internal adversaries are the log entries constructed each time a configuration change is performed. Using PKC protocols, these log entries are signed by the authorisation-node participating in the protocol after all claims in the log message have been thoroughly verified. A secondary line-of-defence is the buzzer introduced in section 3.3.3. This buzzer will sound a clear beep each time the credentials are used to sign a log message. In some cases (depending on the risks), the user may have to confirm by pressing a button or swiping a finger over fingerprint reader. As we assume that an honest user protects her credentials, the logs offers non-repudiation—which means that any configuration change can be traced back to the responsible user.

The initial data communication on the short-range channel is used to exchange public keys to be used in a classical Diffie-Hellman key exchange, which provides the necessary session keys to continue the protocol on the radio channel.

In case of a **bootstrap** (first sensor installed on a patient) the log message *"New group `group-id` created by `n-id` and imprinted with `ca-pk[]`—signed `a-id`"* is prepared by the authorisation node (where `a-id` and `n-id`

[1] The Diffie-Hellman key exchange involves two point multiplications. With TinyECC, this can be performed in less than 4 seconds.

are the identities and public keys of the authorisation-node and the new sensor node participating in the protocol). The authorisation-node must now check the correctness of this log message. Since it knows its own identity, the identity of the new sensor node (which it knows from the short-range packet exchange), and the list of keys of trusted servers, ca-pk[] (which is chosen by the authorisation-node itself), the only things to verify are: that group-id is a *new* group, and that n-id is a member of this group. The sensor node constructs a proof of these two facts and sends it over the radio, and if the authorisation-node can accept this proof, it will produce a clear beep to alert the user, and then hand over a copy of the log message to the sensor.

In case of an **association**, the log message will be *"Node **n-id** added to group **group-id** by **s-id**—signed **a-id**"*, where s-id is the identity and public key of the sensor already placed on the patient. The sensor node placed on the patient checks whether the authorisation-node is authorised to make this configuration change by checking that the digital signature provided by the authorisation-node is signed by any of the certificate authority servers accepted by the sensor group. Meanwhile, the authorisation-node checks that s-id is a member of group-id and that s-id will accept n-id as a new member. If this is the case, the authorisation-node will provide a clear beep and hand over copies of the log message to both sensors. If the sensor on the patient accepts the authorisation of a-id and verification of the log signature also succeeds, the new sensor will be accepted into the group. No private information about the group is transmitted to the new member until the log message (and authorisation-node) has been accepted by the sensor on the patient.

External Adversaries

Security against external adversaries is simply a matter of encrypting and authenticating all radio communication using session keys generated by the classical Diffie-Hellman key exchange after exchanging public keys on the short-range channel. Usually, when using the Diffie-Hellman key exchange the public keys are authenticated in order to prevent man-in-the-middle attacks. In this case, however, man-in-the-middle attacks cannot be launched by an *external* adversary on the short-range channel (as the adversary would have to inject information), so this simple scheme is sufficient.

An external adversary may attempt to launch attacks using captured nodes, but being external implies that he cannot capture a node already installed on a patient. Capturing an authorisation-node would not be of any help either (even if the owner was logged in), as it must be used close to the patient, who the adversary is trying to get access to (sensor groups are completely isolated from each other by nature, so accessing a different patient, which the adversary *has* access to, would not be of any help). We assume that honest users will discover and discard tampered sensor nodes upon install, and un-tampered sensors will reset to factory settings when installed on a patient. Therefore uninstalled sensor are also of no use to the external adversary.

Internal Adversaries

The main differences between external and internal adversaries are, that the internal adversaries have full access to the short-range communication channel and may tamper with nodes.

The protocol is constructed in such a way, that no sensor leaks any private information until the log message has been verified—and the log message is not even transmitted to the sensor until the authorisation-node produced a beep. This implies that even with full access to the short-range channel, an adversary cannot get access to any private data without creating a log message using his credentials. For instance, an adversary may try to learn the `group-key` of some sensor group by installing a tampered sensor (from which he will be able to extract the key). However, the key will not be revealed until the log message is permanently stored in the sensors in the group (and perhaps even forwarded to a server), so if the adversary uses his own credentials and the leak is discovered, evidence will point toward him.

Another possible attack for an internal adversary could be to try to launch a man-in-the-middle attack on the short-range channel when an unsuspecting user is performing a legitimate configuration change in order to hijack the session and steal that user's credentials without arising suspicion. First of all, the technology for the short-range channel should be chosen to make this type of attacks difficult (for instance demanding some hidden node closer to the user's authorisation-node than the sensor she is installing). Secondly, the session will fail—the new sensor will continue to search for an authorisation node even when the authorisation-node beeps, or the new sensor will start blinking in a different group blinking pattern than expected (cf. the BLIG user interface description in section 3.3.5).

Tampering with a sensor already installed on a patient can at most give access to sensor readings from that particular patient. As sensor groups are isolated from each other, no access to other patients would be possible, and for any configuration change a signed log message from an authorised user is still required. This situation is therefore not worse than with current technology, where a bug could be installed on a sensor by an internal adversary; and in both cases the leak may later be discovered by an honest user (since we assume that a tampered node can always be recognised by an honest user).

Tampering with a sensor which is not installed on a patient yields nothing unless it is later installed on a patient. Since no honest user will do this, the adversary has to do this himself. And as we assume that a user will not behave dishonestly if the act can be traced to him, he must attempt to steal an(other) authorised user's identity. This will be difficult as we assume that honest users carefully protect their identities, and would hear the beep—and discover the adversary, who must be nearby.

Tampering with authorisation-nodes may first of all be very difficult, as these nodes can be equipped with anti-tampering measures which would make the node self-destruct. Assuming tampering *is* possible, the credentials stored on the node must be encrypted and only be accessible when the owner has logged in. Tampering with an authorisation-node when its owner is not using it (for instance deactivating the buzzer) would be discovered by the owner before she logs in, and she will discard the node. If a user tampers with her own node, she would just learn her own private credentials (as this is the only non-public information stored there), and she cannot use that for anything, except posing as herself. As honest users protect their authorisation-nodes when they are logged in, it will be difficult for an adversary to get his hand on such a node without being discovered.

5.5.3 Evaluation

The current implementation of the AlphaProtocol performs the bootstrapping in less than 10 seconds and association in about 13 seconds when executed on the MSP430 micro-controller at 8.4 MHz. The implementation is still not optimal—a few lengthy operations (such as large integer modular divisions) could be removed by optimising the code further. Also, with more RAM available, Shamir's trick (double multiplication using a two-dimensional sliding window) is supposed to reduce the execution time by at least a couple of seconds [63].

In order to evaluate the AlphaProtocol energy consumption, the *Energy Bucket* (presented in the next chapter) was used to measure executions of the bootstrap as well as the association protocols on a sensor node. Figure 5.3 shows this measurement setup with a 3 second excerpt of the measurement data showing the current draw during one of the signature verifications.

On the current hardware platform, the AlphaProtocol consumes 0.1 J

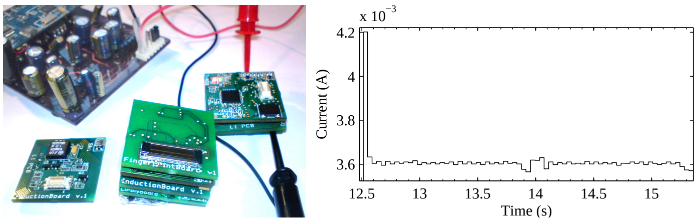


Figure 5.3: A picture of the measurement setup along with a 3 second current/time plot showing the typical current draw during a single signature verification.

for each protocol execution (small differences between the bootstrap protocol and the association protocol on the S and N sides exist on the second decimal place only). This number includes the energy spent on the computation as well as communication on both the short-range and the radio interface, while energy spent on the user interface (LEDs) is not included.

In comparison, the BLIG user interface spends about 0.5 J/min (cf. section 3.3.6), which implies that after just 12 seconds of BLIG blinking, the user interface will have spent more energy than the `AlphaProtocol`. A watch battery (LR621) contains 81 J, so this would approximately be the amount of energy available for the smallest sensor. As a small sensor would only have to go through a few (say, 5) configurations changes in its entire life time, there will be plenty of energy left for the sensor.

Further details about the `AlphaProtocol` evaluation are available in the paper on page 137 ff.

5.6 Conclusion and Future Work

Two configuration change protocols were presented in this chapter: the `NaiveProtocol` which is fast but has very poor security properties, and the `AlphaProtocol`, offering much better security but requiring a lot more memory and processing power.

These two protocols are just the first in a line of planned protocols with various combinations of assumptions and requirements regarding the short-range communication hardware, threats, and (internal) adversary capabilities. These protocols achieve different flavours of security, for instance with different strategies for where and when to check the log signatures. With experiments, the costs of these different flavours can be charted in terms of running time (user interface latency), energy consumption (battery life), and software complexity (capabilities and price of hardware).

A proper user test involving clinicians was planned from the start. However, with the `AlphaProtocol` and (in particular) the security libraries, the prototype software grew larger than what could fit into the memory of the chosen micro-controller (MCU). This problem was realised too late, when there was no time to switch to a different platform, and consequently the user testing was postponed. The focus shifted toward technical tests, which could still be performed on the chosen MCU by testing the individual components separately, and tests with users remain future work.

In order to build the full platform, a bigger MCU is necessary. The MSP430 processor has recently become available in an updated version with up to 256 kB flash available, which should be more than enough for this prototype with any protocol. Furthermore, this newer processor may run up to 25 MHz and can perform 32 bit multiplications on the hardware, which suggests that the `AlphaProtocol` execution time can be reduced from

13 seconds to below 1 second. With this reduction, a prototype with a full level of security *and* a complete and responsive user interface is within reach.

Rather than using the general-purpose MCU for the ECC point multiplications, dedicated hardware might prove useful (just like the CC2420 AES processor mentioned in section 5.3). Therefore, an orthogonal future work direction would be to examine the utility of such hardware. [35] reports of a simulation of an ECC hardware implementation which will spend just 324 μJ on a digital signature verification. Of course extra silicon for this co-processor would make the sensor slightly more expensive, but if the reduced power consumption would make the battery smaller, this could outweigh the cost of the co-processor—and result in a smaller, lighter, and perhaps even cheaper sensor.

Chapter 6

Developing Low-power Applications

Most of the development of the short-range communication (appendix A) and the fingerprint reader (appendix B) took place during the author's 4 months visit at the Imperial College, London. Experiences from this work inspired a practical way to improve the job of developing low-power applications.

The results and contributions of this work were presented in the paper *Energy Bucket: a Tool for Power Profiling and Debugging of Sensor Nodes* [3] found on page 129 ff.

6.1 Motivation

In order to debug an application running on a sensor node, the application developer will often use the LEDs for debug signalling and in some cases he may print text messages to the host PC terminal via the serial USB connection (cf. the TinyOS 2.x `printf` and `diagmsg` libraries). The use of these methods can be quite cumbersome, and in some cases they would cause resource conflicts (the LEDs or serial connection may already be in use by the application). If a test produces only a small amount of data, these data can be cached in the sensor node RAM and transferred after the test completes; otherwise, these methods are too slow[1] to be useful when fast and timing-critical code must be tested.

During the development of the fingerprint reader and short-range communication software driver development, timing details would be checked by outputting some information about the internal state of the drivers on a couple of digital I/O pins of the micro-controller (MCU). These states would be recorded by a multi-channel oscilloscope (sometimes along with other values, such as the node's current draw) and analysed.

[1] Fast LED changes cannot be perceived by the human eye, and the serial connection speed is limited and requires significant RAM and CPU resources, which may disturb the application under test.

At the Imperial College this work could be performed in a lab with all the necessary equipment at hand, but it would still take some time to prepare all the wires for each measurement—and even more time to properly interpret the results afterwards. Back in Aarhus it was not even possible to perform the application development in a lab environment, and a colleague of the author, Morten T. Hansen, had similar problems comparing the energy consumption of different sensor network routing protocols. An examination of existing solutions revealed, that the existing tools are both expensive and not at all optimised for the sensor network application developer's needs, so together we chose to design one that is.

6.1.1 Goals of this Work

One of the tools most commonly used by developers for debugging when the LEDs and serial connection are not sufficient, is the oscilloscope. In addition to being both bulky and expensive, a lot of manual effort is required from the user in order to do a simple measurement of energy consumption, and as demonstrated in section 6.3, the accuracy may not be that good either.

The goal of this work was to design a tool with focus on *application development*, which include the following:

Support easy energy measurement, including measuring the energy consumption in different parts or subsystems of an application (defined and chosen by the developer), and making comparative studies of the energy consumptions of different implementations.

Support easy and fast debugging, which is obviously important to a developer—this includes some kind of support for inspection of the program state and flow.

Integration, in the sense that using this device fits the usual workflow of the application developer.

Small so it will fit on the desktop and be easy to carry around.

Cheap as each developer should have at least one at his disposal.

6.2 Related Work

A number of different energy meters and testbeds can be found in related research. The focus of these instruments, however, is mostly on the evaluation of entire applications—not on the needs of the developer during the development process. Most of the related work measures energy consumption by

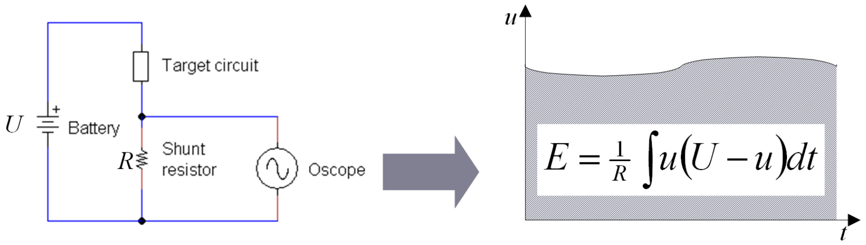


Figure 6.1: Using a shunt resistor to measure energy consumption.

reading the voltage over a *shunt resistor* as illustrated by figure 6.1.

Other methods exists. One method, which was proposed in [91], is designed for accelerated evaluation of the battery lifetime for sensor node programs. In this proposal, nodes are powered from a very large capacitor rather than a battery cell. The node will discharge the capacitor while executing the target program and die when the capacitor is depleted. The lifetime of the application is measured and used to predict the lifetime when using a real battery. Another approach could be to use a standard clamp am-meter (a.k.a. tong-tester), which is completely unobtrusive. Unfortunately, this method has a narrow dynamic range—a setup used in [75] was only capable of measuring 2 decades of current (0.4 mA to 40 mA).

A testbed energy measurement solution was proposed in [42]. This paper describes the *PowerBench*, which is a testbed designed primarily to compare routing protocols. The testbed manages up to 24 sensor nodes, and measures the energy consumption of each individual node. Its resolution is not adequate for detailed energy measurements of sleeping nodes, but as it was designed for routing protocol evaluation, nodes are not supposed to sleep much during the tests. Besides, due to the number of channels, low complexity and minimal price per channel was more important to the authors than high accuracy. (They report a cost of about €27 per channel).

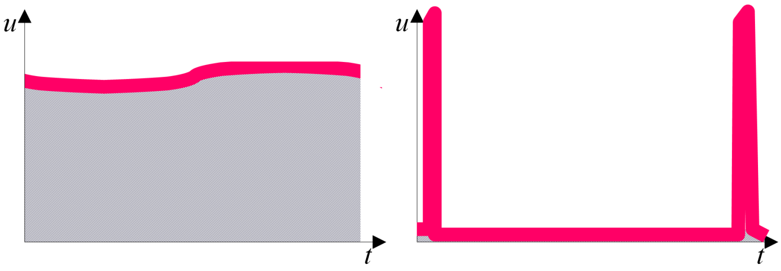
Rather than attaching the nodes to a fixed testbed, the energy measurement could be performed by a sensor board attached to the node, which is the approach pursued by [50] and [107]. Performing the energy measurement on a sensor board implies that the sensor application must be adapted to read the energy measurements itself, and forward the readings to some base station collecting the information; hence, this approach is more intrusive and may even influence the measurements. These energy meter sensor boards also consume a significant amount of energy themselves, making them suitable only for testing purposes, and they should not be used for actual sensor network deployments. However, the idea of having sensor node applications monitor their own energy consumption during deployment was pursued by *iCount* [31]. The key motivation for this work was that many sensor applications may benefit from a detailed knowledge of the energy resources—for instance routing protocols could avoid passing packets through nodes with nearly depleted batteries. The iCount energy sensor works only with sensors that use boost converters in their power supply, and is "free" in the sense that if the MCU has an unused counter input no overhead is added. The accuracy is only within 20 % which makes it unsuitable for precise energy measurements.

PowerTOSSIM [99] takes a different approach by extending a simulator with energy estimation capabilities. This approach has the advantage that the estimation can be performed entirely in software on the developer's workstation, avoiding the need for any measurement equipment, but as the estimation is based on a simulation, it may not be accurate, as the simulation may not capture all real-life phenomena. A solution to this could be to do the energy estimation in real sensor nodes. This is the basic idea of [30], where a sensor OS is modified slightly to monitor the state of each hardware component, keeping a count of the total energy consumption, with only a small computational overhead. This can also be used as an iCount [31] alternative to keep track of remaining energy in a deployed network.

6.3 The Energy Bucket

Looking through the "wish list" in section 6.1.1, it is clear that an oscilloscope or a similar expensive instrument would not be an ideal solution, due to the cost as well as the physical size. Furthermore, since the method of reading the voltage over a shunt resistor involves some degree of uncertainty (illustrated on figure 6.2 by the thickness of the red line), extra caution is necessary when this method is used to measure energy consumed in narrow pulses, which is typical for sensor node applications. In order to calculate the energy consumption, the measurements must be integrated. Most electrical equipment draw a steady current like the graph on figure 6.2a; and in this case the method can be successfully applied, as the grey hatched area represents the "certain" energy consumption, while the area of the red line represents the uncertainty (which in this case is only a small fraction of the total area). As most sensor network applications draw current in short bursts, the situation now becomes that of figure 6.2b. This demonstrates, that when measuring the energy consumption of sensor applications, the demands on the precision of the measurement equipment are much more strict. As explained in detail in appendix C.1.1 and the paper on page 129 ff., standard oscilloscopes do not even meet these requirements.

In order to address this issue of measurement precision in our design, we narrowed our options down to two approaches: either the shunt resistor voltage method using an audio ADC (which would have a reasonable voltage resolution and an acceptable temporal resolution), or some variant of a charge pump (explained below). The charge pump method was chosen, as a simpler circuit and also much less data processing was expected.



(a) A good case (b) A bad case

Figure 6.2: A good and bad case for the shunt resistor voltage method.

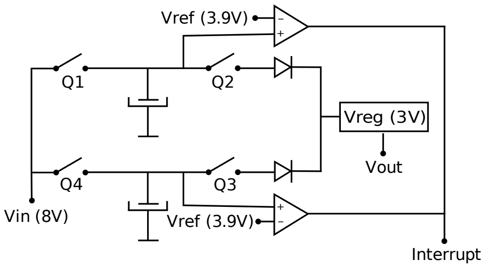


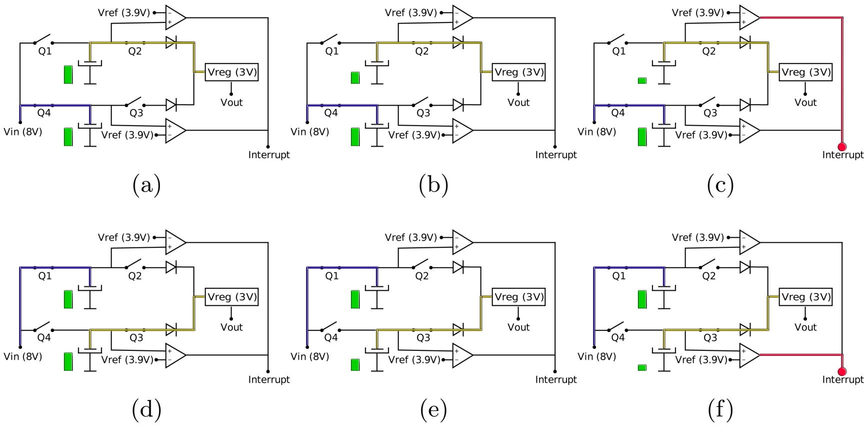
Figure 6.3: The Energy Bucket overview schematic.

6.3.1 Energy Meter Design

An overview schematic of the circuit is provided by figure 6.3. The charge pumping was implemented using a couple of capacitors which are repeatedly charged to a voltage of 8 V and discharged to 3.9 V. One of these two *bucket capacitors* will be charged while the other one is discharging through the target circuit. When the lower voltage level is reached, the other bucket capacitor is switched in instead, and the first will be switched out and recharged. This completes a single "stroke" of the pump. The operation is illustrated on figure 6.4, showing two such strokes.

The four switch symbols, Q1–4, represent transistors controlled individually by software running on a Tmote Sky mote [106]. A comparator for each bucket capacitor detects when the voltage reaches 3.9 V and triggers an interrupt on the Tmote MCU, which will then respond by executing the switchover between the capacitors.

Following the charge pump circuit, a voltage regulator regulates the output voltage (to be fed to the target circuit) down to 3.0 V. As each stroke of the charge pump delivers a certain fixed amount, Q, of charge to the target circuit, and the voltage over the target circuit is regulated to a fixed voltage, $U = 3$ V, the pump will complete one stroke each time the target circuit consumes the energy $E = Q \cdot U$. The name *Energy Bucket* was derived from this process, as each stroke of the pump signifies the consumption of one



(a) (b) (c)

(d) (e) (f)

Figure 6.4: The Energy Bucket modus operandi.

.

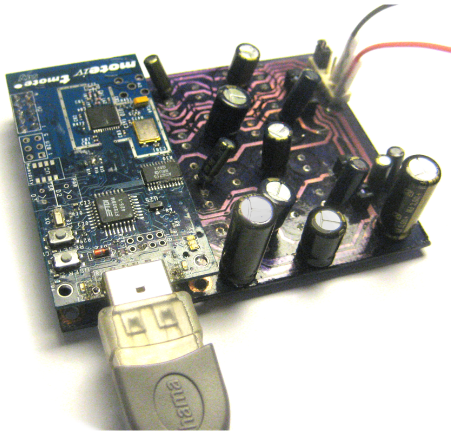


Figure 6.5: The Energy Bucket device.

“bucket of energy”. In theory, measuring the energy consumption of the target circuit is simply a matter of counting the number of pump strokes, but as explained in the paper on page 129 ff. and in appendix C.2.7, it turned out to be a bit more complicated than that to get exact measurements.

Figure 6.5 shows the Energy Bucket device; notice the Tmote Sky mote which controls the measurement process on top of the Energy Bucket board. Measurements are transmitted to a host PC via the USB port. Further details about the hardware and a full schematic can be found in appendix C

The Energy Bucket device was calibrated and tested as reported in the paper on page 129 ff. and in more detail in appendix C. Over the 5-decade current range 1 μA–100 mA, the accuracy of the Energy Bucket is ±2 % or better.

Develop code The sensor network application is written. . .

Compile . . . and compiled at the host PC.

Install The binary is transferred to the sensor mote flash memory.

Run ***and measure energy consumption*** The application is tested by executing it on the sensor mote. *During this test, the energy consumption is measured.*

Figure 6.6: The *extended development cycle* (the iterative cycle for sensor network application development).

6.3.2 USB Cord

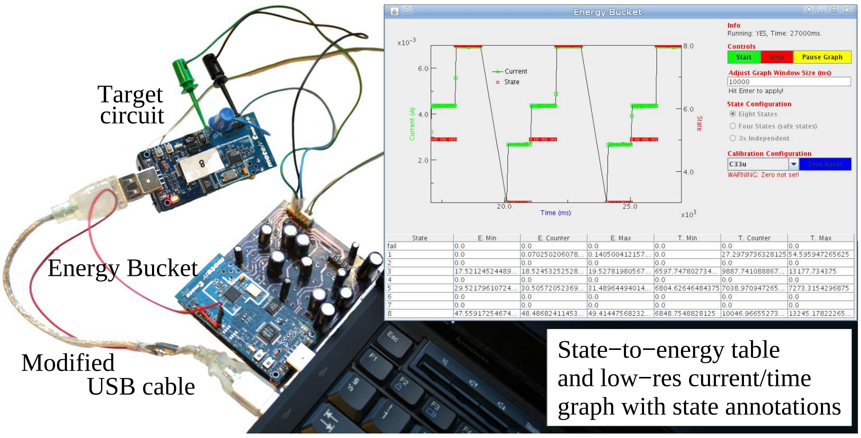
One of the goals of this work (cf. section 6.1.1) was to simplify the job of measuring the sensor energy consumption during application development, by making this an integrated part of the work-flow. The *development cycle*, defined as the iterative process of programming, compiling, installing and testing applications, can be *extended* as illustrated in figure 6.6 to include the energy measurement as an integrated part of the testing phase.

A tedious and potentially time consuming task at the "run and measure" step would be disconnecting the target circuit from the flash programmer used in the install step and attaching it to the Energy Bucket—and afterwards reattaching it to the programmer in the subsequent iteration of the cycle. As both sensor mote platforms used in this work (Tmote Sky and BSN) use USB as the flash programming interface, a solution for this particular arrangement was devised, automating this interchange.

A short USB extension cord was modified to include a power switch. This enables the target sensor mote to remain attached to the USB cable during a measurement as the Energy Bucket will cut off the USB power when running, and the entire development cycle can be completed without any need to attach or remove a single cord. This full arrangement is shown on figure 6.7. It is important, though, that the Energy Bucket and the extension cord are attached to the same PC or USB hub in order to prevent ground loops and related problems. More detail about this cable can be found in appendix C.3.

6.3.3 Usage

During application developing, the developer typically needs information about the energy consumption in particular sections of the code, for instance in order to compare different implementations or to evaluate progress (e.g. check whether the latest code modifications improved energy efficiency or not).



Info
Running: YES, Time: 27000ms.

Adjust Graph Window Size (ms)
10000
Hit Enter to apply!

State Configuration
● Eight States
○ Four States (safe states)
○ 3x Independent

Calibration Configuration
C33u
WARNING: Zero not set!

State	E. Min	E. Counter	E. Max	T. Min	T. Counter	T. Max
fail	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.070250206078...	0.140500412157...	0.0	27.2979736328125	54.595947265625
2	0.0	0.0	0.0	0.0	0.0	0.0
3	17.52124524489...	18.52453252528...	19.52781980567...	6597.747802734...	9887.741088867...	13177.734375
4	0.0	0.0	0.0	0.0	0.0	0.0
5	29.52179610724...	30.50572052369...	31.48964494014...	6804.62646484375	7038.970947265...	7273.3154296875
6	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0
8	47.55917254674...	48.48682411453...	49.41447568232...	6848.7548828125	10046.96655273...	13245.17822265...

State−to−energy table
and low−res current/time
graph with state annotations

Figure 6.7: The Energy Bucket in action.

Most energy measurements (in particular those using the shunt resistor voltage method) delivers the results as high resolution graph data of current versus time, and a great effort is often required from the application developer to interpret these information—extracting the interesting bit of information from this vast amount of data. If for instance the developer wishes to know the energy consumption in a particular section of the code, he will have to somehow identify the start and end points in the current/time data and integrate the section between the points. This is usually a quite time-consuming task, which is why the developer may not find it worth the effort.

In most situations, high-resolution data/graphs of current versus time are irrelevant to an application developer (while a low-resolution graph may be a practical way to visualise the application activity). Instead, what the developer really needs is just a simple scalar value—the amount of energy consumed in a section of code of his own choice. In order to realise this vision, however, some mechanism is necessary to help the application developer define the section he wants the measurement to cover. One solution could be to allow some source code annotation markings of the "in" and "out" points.

We chose a slightly more general solution, allowing source code annotations of a *state number* (an integer in the range 1–8). A small TinyOS 2.x library exporting the following standard C function through a header file, may be included anywhere in the target program:

```
void energy_state(const energy_state_t state);
```

The current state number is signalled to the Energy Bucket by the target MCU using 3 general I/O lines. Most sensor prototype platforms (including the Tmote Sky and BSN motes) have plenty of general I/O lines available, so reserving 3 lines for energy measurements should be possible in most cases.

At compile-time, the developer may choose to compile the application with the real library or an alternative empty skeleton. The real library method outputs the chosen state number on the three general I/O pins on the target MCU, which only requires a couple of extra MCU instructions (clock cycles) causing only a minimal overhead. The three pins are connected to three inputs on the Energy Bucket, and the Energy Bucket software presents the energy readings in a table according to the state number along with the total energy consumption and a combined current/time and state/time graph—see figure 6.7.

In the current implementation of the Energy Bucket, program state is only recorded once per pump stroke, which implies that a state that is visited for a shorter duration than the length of a pump stroke may not be registered. Furthermore, the granularity of energy measurements are fixed (cf. the Q in section 6.3.1 which equals approximately 145 μC with the capacitors chosen in the current circuit—corresponding to an energy of 435 μJ). As a consequence, while the Energy Bucket measures the overall energy consumption of the target circuit quite accurately, the exact distribution among the different states will be more uncertain.

In order to address this granularity problem, the Energy Bucket software keeps track of the minimum and maximum energy spent in each state (cf. figure 6.7). Whenever a state change happens between two pump strokes, half a bucket of energy is added to each of the two states' counters, as we cannot know when the state change took place exactly. Meanwhile, a full bucket of energy is added to both states' *maximum-counters*, while nothing is added to the *minimum counters.* Assuming that at least one pump stroke takes place each time a state is visited, the minimum and maximum counters will be totally reliable (within the 2 % accuracy of the instrument), and the true energy consumption will lie somewhere between those bounds.

6.4 Evaluation

The Energy Bucket has been used over a period of a couple of months of application development—most notably, the development of the alternative radio stack described in section 5.3 and secure protocol of chapter 5. A crucial characteristic of the Energy Bucket is the fact that it offers energy measurements with virtually no overhead. This, encourages its use, as the developer will experience, that for the development of many types of sensor applications, the benefits from using the Energy Bucket far outweigh any inconveniences.

During the development of the alternative radio stack, the coarse current/time graph proved very useful, as the program behaviour in this case would be immediately apparent—in this particular case the program would be testing elements of the radio stack which would explicitly or implicitly affect the power level of the radio chip, and this could be inspected directly on the graph.

As mentioned in the motivation, under *normal* circumstances (when an energy meter is unavailable or cumbersome to use), a sensor network application developer uses the LEDs for debug signalling and in some cases and may print text messages to the host PC terminal via the serial USB connection. With 3 LEDs there are only 8 different colour combinations, and the serial connection has a number of limitations as well—it cannot be used when interrupts are disabled, uses a lot of memory and involves tasks and interrupt handlers, disturbing the duty cycling pattern of the MCU. In contrast, the Energy Bucket is completely non-invasive and in many situations, the current/time graph (possibly with the state/time overlay, cf. figure 6.7) will provide more detail than three LEDs ever could.

Perhaps the most important lesson from this work was gained the very first time the Energy Bucket was ever used, as it revealed a programming flaw in a software library, which had gone unnoticed for more than 6 months.

The final Induction Board hardware (for the BSN mote) and software (cf. appendix A) was developed during the author's stay at the Imperial College. After the circuit board design had been completed, a few weeks of waiting for the production of the boards was spent completing and testing the software library on the prototype, which was built using Tmote Sky motes.

All this was done in the lab using an oscilloscope (as it happened before the Energy Bucket was conceived), and as the circuit boards for the BSN mote arrived from the manufacturer, only the most basic functions related to the hardware were tested using the oscilloscope (the `InductionPulse` interface implementation, cf. appendix A.2). At the time the author did not find it necessary to test the full communication library (the `InductionTx` and `InductionRx` interface implementations) on the BSN motes using the oscilloscope, as it was "just" an additional software layer which had already been fully tested on the Tmotes.

About 6 months later the Energy Bucket was almost completed and as an experiment, it was used to measure the energy consumption of a BSN mote running an arbitrary application—which by chance happened to be using this communication library. The result was a surprise, as the application turned out to occasionally draw about 20 mA more current than it was supposed to.

Measurements on the Induction Board hardware showed no signs of any malfunction—in fact the leak would still be present when the board was detached. The intermittent pattern of the current leak led to the conclusion that a software error was causing it, and debugging using the iterative development cycle (figure 6.6) quickly led to isolation of the problem.

In order to get the timing right in the packet encoding/decoding module, a couple of debug signals were monitored by the oscilloscope during

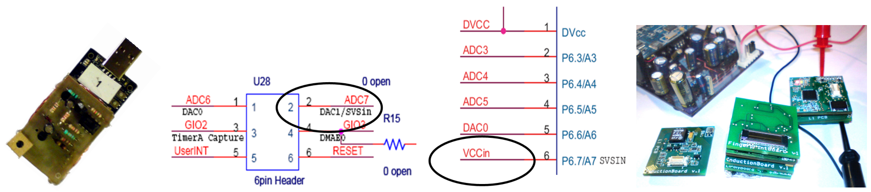


Figure 6.8: Difference between the Tmotes (Telos rev. B) and BSN motes (Telos rev. A) designs.

the testing phase, and a pre-compiler constant controlling this was never disabled as it should have been. One of the debug ports used was port `6.7` on the MSP430 MCU, which is exported for external use on the Tmote Sky. On the BSN mote, however, this port is connected directly to the positive supply rail, so a low output on this port caused a short-circuit, cf. figure 6.8.

Since the application worked perfectly well on both platforms, and the author had no plans of performing extra lab tests on the BSN mote (as the application had already been thoroughly tested on Tmote Sky motes), this bug would probably have gone unnoticed without this tool. If the application had been using the radio (which also draws about 20 mA when active), a power profile of the buggy application may not have aroused any suspicion due to its complexity, so the key lesson here is, that energy measurements should *not* be limited to a single measurement when the full application is finished, but should rather be used continuously for early programming error discovery. This will furthermore convey a better understanding of the behaviour of the application under development.

This experience also proved, that having instant energy measurements available while debugging (and programming in general) is indeed a very powerful tool, as the power fingerprint of an application may reveal a lot more about what is going on, than three LEDs. The true strength of the Energy Bucket is that it integrates so well into the development cycle (figure 6.6). In the above example, debugging was typically performed by commenting out a few lines of code, so the "develop code" step would often take only a couple of seconds. This would make the "compile" and "install" steps the most time-consuming bottleneck, as the "run and measure" step also would complete in a few seconds.

As a final note, after this work was presented at the SENSORCOMM 2009 conference, several sensor network application developers attending the conference approached the author—or sent an email afterwards—asking for more details, and confirming that they too experience the same struggle with energy measurements during application development, proving that this device addresses a real need in the sensor network community.

6.5 Conclusion and Future Work

The main lesson learnt from this work has been the importance of having energy measurements incorporated into the everyday sensor application development cycle, rather than as isolated (and rare) lab experiments. The Energy Bucket proved itself useful in particular when developing code that manages I/O and hardware state as reported in the previous section, and this type of code is very common in many sensor network applications.

The Energy Bucket stands out from competing measurement tools in two aspects. First, its ability to deliver energy measurements as pure numbers rather than graphs (which require post-processing). Secondly, it is capable of categorising the energy consumption according to the program state chosen by the application developer.

The Energy Bucket device also proved very accurate, despite its low price and simple design. With an accuracy better than ±2 % over the relevant 5-decade range of current draw, it is safe to conclude, that the energy measurement results are more accurate, than the results obtained from a normal oscilloscope using the serial shunt resistor measurement method. The total cost of the components was around €50 in addition to the Tmote Sky used as controller. Incidentally, a Tmote with a defective radio (which is useless in most sensor network experiments anyway) was used.

The main problems to be addressed in the future are the low temporal resolution and the uncertainty on the distribution of energy among the different states.

Each time a state is visited, ± 1 bucketful of energy is added to the uncertainty of that state. If the state only consumed a few bucketfuls of energy during the visit, ± 1 is a huge uncertainty. This will be the case for states that are visited for very short durations and/or draws very low currents.

A cure for this problem—which also improves the temporal resolution—is smaller “buckets” (capacitor values). This of course would imply higher switching frequencies, and the software running on the Tmote—and possibly the hardware as well—would have to be redesigned. Adding a couple of flip-flops and a few gates to the circuit, the charge pumping could be performed in hardware, and the Tmote would only have to count the number of pump strokes. As the counter input of the MSP430 MCU can operate as high as 10 MHz [77], capacitor values could be chosen such that, say, 5 MHz would correspond to 100 mA; then a current draw of 2 μA would correspond to 100 Hz, and the bucket size would be 60 nJ.

Another way to address the problem could be to perform a hardware detection of state changes to force a switching immediately. Of course then the amount of charge remaining on the switched-out capacitor would have to be measured, and one possible way to solve this problem could be to use an ADC—perhaps one of the MSP430 MCU’s available ADC inputs—

to measure the capacitor voltage. A similar method was successfully used in [25] at high frequencies, demonstrating that this could actually work.

One of the reasons for choosing the charge pumping method was because less data post-processing was expected—simply counting the number of energy buckets should have been sufficient. As explained in appendix C.2.7 at equation (C.9), the translation from bucket count to energy turned out to be non-linear (with an exponent slightly below 1). In order to compensate for this non-linearity, the data is currently differentiated, after which the transformation formula is applied and the result integrated. A possible cause of the non-linearity could be the dielectric material used in the electrolytic capacitors, so in the future, experiments with other capacitor technologies should be performed—when the capacitor values are smaller, non-electrolytic capacitors (ceramic, polyester etc.) may be used instead. Hopefully the need for the exponent in the translation will disappear, and all these extra calculations can be simplified.

Finally, a measurement setup using the shunt resistor voltage method with an audio ADC was never attempted. It would be interesting to investigate which method is simpler to implement, and also which method performs better. Hence, a possible future path for this work could be to build such a device in order to do a comparison.

Chapter 7

Conclusions and Future Work

For the conclusion, a recap of the research questions from section 1.2.1 along with a few comments will be presented, followed by a general summary of the scientific contributions of this work. Finally, an overview of future work directions is presented.

7.1 Conclusions and Contributions

> **Question 1:**
> How to construct appropriate user interfaces for even the smallest resource constrained sensor, usable at hospitals, emergencies and in the home?

The experimental research platform proposed in chapter 4 provides only the most basic user interface hardware, namely a single push button and a group of LEDs. These components can be produced at low cost in almost arbitrarily small sizes, and any shape and form imaginable—and still be easy to operate. A key advantage of LEDs is, that they are easy to observe from any angle and under very different lighting conditions (bright sunlight to complete darkness); however, they do draw a significant amount of power, compared to, for instance, LCD displays with no background light (the kind which is used for digital wrist watches). In addition to the button and LEDs, the research platform offers short-range communication hardware, which may also be used for proximity detection (as nodes will only be able to communicate when they are in close proximity of each other).

User interfaces based on the hardware proposed above can be embedded in virtually any small sensor, including pills and patches, and can be used anywhere—also under the harsh conditions found at emergency sites.

In chapter 3 the BLIG (Blinking LED Indicated Grouping) user interface for medical sensors was proposed. Using only the button, LEDs, and proximity detection provided by the platform (as described above), this user interface offers group creation, inspection, and changes. An implementation of this concept was tested at an early stage on a group of clinicians and ambulance personnel, who found that it was easy to use and understand. Later versions were only tested with nonprofessionals, but with similar results.

The main BLIG power consumption is due to the LEDs, hence a major concern is limiting the use of these. In general, the need for the LEDs to be on, depends on the level of sensor configuration activity found at (or in close vicinity of) the patient. This suggests that the power consumption will be high at emergencies and very low for home care. As sensors used at emergencies are used for much shorter durations than sensors used for permanent monitoring in the homes, acceptable power consumption should be achievable in any case. However, further studies of concrete use scenarios will be necessary to reach certain conclusions concerning this point.

Question 2:

How to design a secure wireless infrastructure obeying legal obligations and allowing the use of sensors straight from the factory (i.e. with no a priori knowledge about the environment they will be placed in), being easy to use and taking the technical restrictions into account?

Chapter 5 presented the `AlphaProtocol` capable of establishing secure communication between sensors, displays and hospital infrastructures. The protocol uses imprinting to support the use of factory-fresh sensors.

The question also emphasises that the protocol must take usability considerations into account in order to ensure, that it will be useful in practise. In order to do this, the protocol was designed to complement the BLIG user interface. On the current hardware and software implementation, however, the protocol execution speed is still too slow to be useful, as the BLIG grouping action will take up to 13 seconds, when used with this protocol. Calculations show that a simple upgrade of the platform to the latest microcontroller (MCU) generation will boost performance, and we can expect all BLIG grouping actions to complete within 1 second.

Measurements show that a single grouping action with this protocol requires 0.1 J (not including energy for the BLIG LED signalling). As the watch battery LR621 contains 81 J, even a sensor using this tiny battery will be capable of executing this protocol a few times during its lifetime with no major battery lifetime impact. A disposable sensor will only be used once (i.e. added to a group once), hence only a couple of executions are necessary.

Question 3:

How can an experimental research platform be designed to facilitate flexible and efficient experimentation with the issues, dependencies and limitations of the triangle in figure 1.3?

The triangle referred to by this question relates the problems of usability, resource constraints, and security. The primary resources considered in this dissertation are device size and lifetime, and of course the energy limitations, which is a direct consequence of the small size and long lifetime combined with the limited energy density of batteries.

In chapter 4 an experimental prototype platform was presented. This platform was designed to facilitate easy usability testing (such as user workshops) as well as technical evaluations of different implementations:

The platform was designed with user workshops in mind, so that different BLIG flavours and secure protocols may be tested with the users with minimal preparation time needed between experiments. In particular, nodes can be reconfigured, constants adjusted, or implementations substituted by simple commands broadcast over the radio link.

Energy measurements in particular, was covered by the Energy Bucket (presented in chapter 6), which offers easy energy measurements of selected parts of an application. This simplifies the work of comparing different prototypes, as the energy consumptions of the "competing" sections of code can be directly compared, while the energy consumed in irrelevant sections of code or during idle waiting for user response or the arrival of messages (which may vary from one experiment to another) can be ignored.

Finally, the experimental prototype platform includes a library offering a number of primitives for easy security experimentation, such as a cryptographic hash function, symmetric cryptography, elliptic curve cryptography functions.

The full implementation of this experimental prototype platform was never tested, as it turned out to be larger than what could fit into the memory of the BSN mote MCU (60 kB flash and 2 kB RAM). For the same reason, it was never tested at a real user workshop. However, different subsets of the platform were successfully used for different tests: user interface implementations with no security to test user interaction, and security implementation with minimal user interface (and no workshop reconfiguration support) to test security and energy consumption. In the future the current MCU will be replaced by a more recent version, which has sufficient memory available to fit in the full implementation.

Question 4:

> Will it be feasible to let common principles guide the designs across all three contexts (hospital, emergency response, homes—cf. section 2.2) such that a sensor can be used everywhere, and maybe even roam freely between the infrastructures with little or no human intervention (so that e.g. sensors can be replaced when it is convenient or when dictated by hygienics, and not when dictated by technology limitations)?

Throughout the development of the prototype platform, the user interface and the key-establishment protocol, the demands found in sections 2.2.1–2.2.3 have all been carefully observed. The BLIG user interface was designed for the demanding emergency context, and offers fast and easy deployment and inspection of the sensors. Still, BLIG proved easy to use and appears to meet the needs and requirements of hospitals as well as home care, although it has not yet been properly tested in those contexts. The key-establishment protocol proposed in chapter 5 offers a level of security, which is adequate for all three contexts; and as explained in section 4.4, roaming (e.g. moving a patient from one ward or hospital to another) can be offered as well—even automatically, if integrated with an EHR system.

Concluding on the question, *"Will it be feasible to let common principles guide the designs..."*, the answer appears to be affirmative: using the proposed sensor infrastructure in all three contexts seems to be feasible indeed. However, a pilot test involving all three contexts should be conducted, in order to test this claim and definitely answer this question.

Based on the discussion above, the scientific contributions from the work of this dissertation can be summarised as follows:

- An experimental prototype platform consisting of sensor hardware and a software library exposing pluggable programming interfaces for sensor user interfaces and key-establishment protocols. This library was designed to offer easy usability testing and energy efficiency measurements of different concrete user interface and key-establishment protocol implementations.

- A medical sensor user interface proposal, offering fast and easy deployment and configuration of sensors—including inspecting the current state of the deployment. The user interface is based on minimal hardware requirements (a single push button and a group of light-emitting diodes) and an early implementation of this user interface was tested with a group of clinicians, who found it easy to use and understand.

- A key-establishment protocol offering fully secure sensor set-up using elliptic curve based public-key cryptography, including an evaluation of this protocol's time and energy consumption, and its impact on the battery lifetime. The protocol is fast enough to be useful in a hectic environment and requires only a small amount of energy, making it useful even with watch-size battery cells.

- A tool to precisely measure and compare energy costs during sensor software development. This tool is designed for easy measurements of particular parts of the software—e.g. the user interface or the key-establishment protocol. Along with the development of this tool, several valuable lessons pertaining to low-power software development in general was attained, such as the importance of regular energy evaluations of an application throughout its development (in contrast to a single evaluation of the final application).

7.2 Future Work

The future work can be categorised in three main directions:

- Hardware platform upgrade
- User tests and user interface experiments
- Integration of sensors in hospital and emergency IT infrastructures

As mentioned in the future work sections of the earlier chapters, a number of limitations need to be addressed in the prototype platform hardware.

Most importantly, a micro-controller upgrade from the current processor (MSP430F149) to the new and improved MSP430x5 series, will provide the memory needed to get the full prototype software up and running, and the processor speed to run the secure `AlphaProtocol` in real-time. Along with this processor upgrade, the induction circuitry could be included on the new processor board (to avoid the extension board); and in case this is done, the circuitry could be upgraded to feature an adjustable communication range and zero-current listen capabilities (cf. section 4.4).

Secondly, the Energy Bucket needs a hardware revision in order to get a better temporal resolution and less uncertainty on the distribution of energy between the different states (cf. section 6.5). This would improve the quality of the energy experiment results.

The two purposes of the experimental prototype platform was to facilitate user tests and technical evaluations. So far, it has been used only for the latter. When the technical upgrades listed above have been implemented and the platform is mature, it should be used for user experiments in all three use contexts (hospitals, emergency response, and home care). Furthermore, some more elaborate experiments with the BLIG group blinking could be performed to investigate what kinds of blinking patterns (rhythms, colours) would be easiest to perceive.

The third future work direction is to investigate how sensors can be integrated in hospital and emergency infrastructures. As mentioned in section 1.2.1, this was one of the main goals in the early stage of this project. Two novel ideas have been considered, as outlined in the subsequent subsections, and both may be further investigated in future work.

7.2.1 Activity-Based Sensor Networks (ABSN)

Throughout the previous chapters it was implied, that a hospital or general practitioner would have a suitable IT system to which the sensor readings

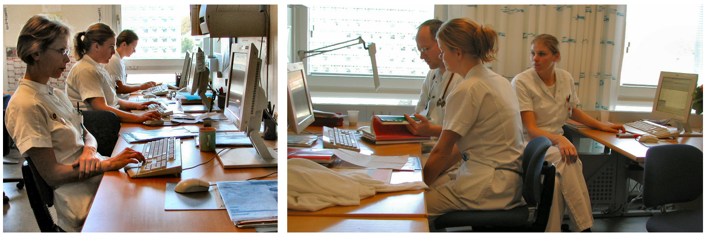


Figure 7.1: The personal computer paradigm influencing hospital clinical work.

could simply be handed over—e.g. an EHR system. Although this is a perfectly fine assumption, the systems currently found were often designed as if doctors and nurses are working at a traditional PC in an office. For a general practitioner who spends most of his time behind a desk in his clinic, this may be adequate; but at hospitals desk-work used to be the exception rather than the rule, and now the IT development has the unfortunate side-effect of drawing clinicians away from patients and into the office as illustrated by figure 7.1.

The goal of the *Activity-Based Computing* (ABC) project [7,26,15,17,23] is to develop a common operating system for pervasive computing devices—ranging from small devices (such as sensors) to large devices (such as conference room computers with interactive multi-user wall-size displays). As indicated by the name, ABC is based on the users' activities rather than hardware, applications, and data, which are the foundations of current operating systems. When the ABC operating system knows what a user is doing (or wants to do), it will automatically collect the relevant data and documents, and present them using appropriate applications on nearby displays.

The *Activity-Based Sensor Network* (ABSN) is an effort toward including sensors, sensor networks, and sensor data into the ABC paradigm. One of the main problems of ABSN is, how to automatically link sensor data to activities. For instance, if a doctor is evaluating the treatment of Mr. Hansen's infection, and a nurse recently installed a temperature sensor on Mr. Hansen, the sensor data and history should automatically pop up on the doctor's screen (without requiring him to explicitly request this information).

Further details on the current status of this work can be found in appendix D, and future work includes the construction of a proof-of-concept prototype of the ideas presented in the appendix used on real-life scenarios.

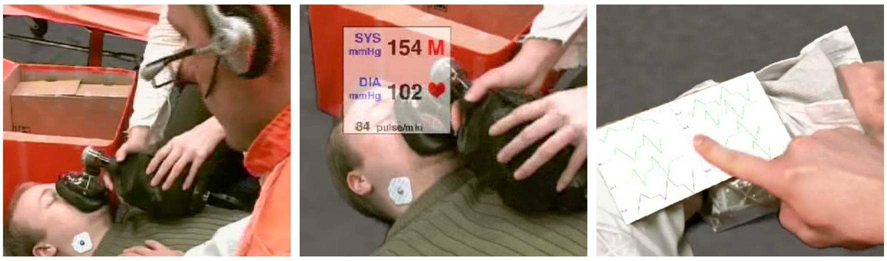


Figure 7.2: Three still images from the PalCom Major Incidents project's future bus accident vision video. The leftmost image shows the paramedic using a headset with a small display. This display adds a box of sensor readings above each injured person the paramedic looks at, as illustrated on the middle image (seen through the eye of the paramedic). The rightmost image shows the idea of a sleeve display presenting sensor readings from a number of persons.

7.2.2 Emergency Site Sensor Data Overview

One of the goals of the PalCom Major Incidents project [83] (mentioned in section 2.2.3) was to explore ideas for technological support for managing information regarding the conditions of the injured. In the early stage of this project, a video was made with the vision of a future bus accident scenario. In this video, sensor data are visualised to the paramedics using a head-mounted display. By looking at an injured person through this display, the paramedic will see the sensor readings floating in the air above that person. Furthermore, the jacket sleeve of the paramedic is equipped with a large display capable of showing an overview of the state of multiple persons. Figure 7.2 shows a few still images from this video illustrating these ideas.

Future work include the construction of a proof-of-concept prototype demonstrator of these technologies based on the prototype platform presented in this dissertation. The basic functionality of this demonstrator will be as follows:

- A computer in the first ambulance arriving at the emergency scene will act as the server for the accident scene sensor network. According to procedure at (major) accidents, the first ambulance arriving at the scene will always be the last one to leave.

- The authorisation-nodes of the ambulance personnel will imprint the sensors with the key of the above-mentioned server and furthermore activate a special *emergency mode* in which the sensor BLIG group blinking will continue uninterrupted until the patient leaves the emergency scene. Furthermore, the group blinking will carry a superimposed high-frequency blink-signal (not visible to the human eye) encoding the `group-id` of the group.

- A camera in the headset of the paramedic will pick up high-speed video (at a frame-rate sufficient to decode the above-mentioned signal). Image processing will identify the location of all sensors in the current view and decode their respective `group-ids`.

- The headset contacts the server in the ambulance requesting data for the individual `group-ids` found in the current view. The data is added as an overlay onto the current view using some head-mounted display technology capable of creating this *augmented reality* [19] effect. Sensors close by will create a large box with detailed information, whereas sensors at a distance will show few information—or perhaps just a coloured dot (red for critical readings, green for stable readings).

- The sleeve display shows sensor readings from the current patient as well as earlier patients who have not yet left the emergency site. Recalling the list in section 2.2.3, patient identification is a difficult challenge in the emergency scenario, and with sensor readings from multiple patients on the sleeve display, this could become a major issue. One possible solution could be to use the camera in the headset to capture an image of the patient. This image could be used as a small icon beside the sensor readings, or as a background image behind the sensor readings on the sleeve display.

One thing that became very clear from the PalCom Major Incidents workshops is, that the medical personnel are requesting some kind of technology to replace the accident cards (cf. section 2.2.3), since they have little time to complete these forms. Furthermore, it is very relevant for doctors at the hospital to get early access to observations, and in particular imagery (still photos) are requested and would be extremely useful. For emergencies, a large (flash) memory device capable of holding media data such as speech and imagery could be installed on the patient as a regular sensor node. The camera and microphone in the headset of the paramedic could then be used to record images of the patient along with a spoken version of the accident card (it would be possible to dictate the contents of the accident card into the headset microphone while using the hands to treat the patient). The memory device can forward recorded speech to a medical secretary service at the hospital when the network resources are sufficient. The medical secretary will type in the dictation and return the text file to the memory node on the patient, as text in most situations will be more practical and faster to browse than a spoken recording.

Bibliography

Papers Included in this Dissertation

[1] J. Andersen and J. E. Bardram, “BLIG: A new approach for sensor identification, grouping, and authorisation in body sensor networks,” in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, ser. IFMBE Proceedings, S. Leonhardt, T. Falck, and P. Mähönen, Eds., vol. 13. Berlin: Springer, Mar 2007, pp. 223–228.

[2] J. Andersen, B. Lo, and G.-Z. Yang, “Experimental platform for usability testing of secure medical sensor network protocols,” in *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2008) in conjunction with the 5th International Summer School and Symposium on Medical Devices and Biosensors (ISSS-MDBS 2008)*. IEEE, Jun 2008, pp. 179–182.

[3] J. Andersen and M. T. Hansen, “Energy bucket: A tool for power profiling and debugging of sensor nodes,” in *The 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM 2009)*. IEEE, June 2009, pp. 132–138.

[4] J. Andersen, “Secure group formation protocol for a medical sensor network prototype,” in *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2009)*. IEEE, December 2009, IN PRINT.

[5] ——, “Towards both usable and secure protocols for medical sensor networks,” in *International Conference on Wearable and Implantable Body Sensor Networks (BSN 2010)*, 2010, PLANNED FOR SUBMISSION.

Cited Work

[6] "29 palms experiment." [Online]. Available: http://www-bsac.eecs.berkeley.edu/~pister/29Palms0103

[7] "Activity-based computing web site." [Online]. Available: http://www.activity-based-computing.org

[8] "A&d medical telemonitoring." [Online]. Available: http://www.andmedical.com/and_med.nsf/html/telemonitoring

[9] M. I. Bagüés, J. Bermúdez, A. Burgos, A. Goñi, A. Illarramendi, J. Rodriguez, and A. Tablado, "An innovative system that runs on a pda for a continuous monitoring of people," in *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, 2006, pp. 151–156.

[10] H. Baldus, K. Klabunde, and G. Müsch, "Reliable set-up of medical body-sensor networks," in *EWSN'04 – 1st. European Workshop on Wireless Sensor Networks*, ser. LNCS, no. 2920. Berlin, Germany: Springer, 2004, pp. 353–363.

[11] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *NDSS'02 – Network and Distributed System Security Symposium 2002*, Feb 2002.

[12] S. A. Ballegaard, J. Bunde-Pedersen, and J. E. Bardram, "Where to, roberta?: reflecting on the role of technology in assisted living," in *NordiCHI'06: Proceedings of the 4th Nordic conference on Human-computer interaction*. New York, NY, USA: ACM, 2006, pp. 373–376.

[13] S. A. Ballegaard, T. R. Hansen, and M. Kyng, "Healthcare in everyday life: designing healthcare services for daily life," in *CHI'08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2008, pp. 1807–1816.

[14] S.-D. Bao, Y.-T. Zhang, and L.-F. Shen, "A design proposal of security architecture for medical body sensor networks," in *BSN'06 – International Workshop on Wearable and Implantable Body Sensor Networks*. IEEE, 2006, pp. 84–87.

[15] J. E. Bardram, "Activity-based support for mobility and collaboration in ubiquitous computing." in *Proceedings of the Second International Conference on Ubiquitous Mobile Information and Collaboration Systems, Lecture Notes in Computer Science*, ser. LNCS, L. Baresi, Ed., no. 3272. Springer, 2004, pp. 166–180.

[16] ——, "The trouble with login - on usability and computer security in ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 9, no. 6, pp. 357–367, December 2005.

[17] J. E. Bardram and J. Bunde-Pedersen, "Iaso – an activity-based computing platform for wearable computing," in *ICDCSW'05 – 25th IEEE International Conference on Distributed Computing Systems Workshops.* IEEE, Jun 2005, pp. 484–490.

[18] A. Bhargava and M. Zoltowski, "Sensors and wireless communication for medical care," in *DEXA'03 – Proceedings 14th International Workshop on Database and Expert Systems Applications.* IEEE, Sep 2003, pp. 956–960.

[19] O. Bimber and R. Raskar, "Modern approaches to augmented reality," in *SIGGRAPH'06: ACM SIGGRAPH 2006 Courses.* New York, NY, USA: ACM, 2006, pp. 1–86.

[20] J. Brøndsted, K. M. Hansen, and L. M. Kristensen, "An infrastructure for a traffic warning system," in *International Conference on Pervasive Services.* IEEE, Jul 2005, pp. 136–145.

[21] "BSN node specifications." [Online]. Available: http://bsn-web.org/index.php?article=926

[22] D. Bucur and J. E. Bardram, "Resource discovery in activity-based sensor networks," in *Proceedings of the 1st International Conference on Pervasive Computing Technologies for Healthcare 2006*, 2006.

[23] J. B. Bunde-Pedersen, M. Mogensen, and J. E. Bardram, "The abc adaptive fusion architecture," in *MPAC '06: Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2006).* New York, NY, USA: ACM Press, 2006, p. 1.

[24] "Cc2420 2.4 ghz ieee 802.15.4/zigbee-ready rf transceiver." [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc2420.pdf

[25] N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: case study of ARM7TDMI," in *ISLPED'00: Proceedings of the 2000 international symposium on Low power electronics and design.* New York, NY, USA: ACM, 2000, pp. 185–190.

[26] H. B. Christensen and J. E. Bardram, "Supporting human activities – exploring activity-centered computing," in *UbiComp'02 – Fourth International Conference on Ubiquitous Computing*, ser. LNCS, no. 2498. Springer, Sep 2002, pp. 107–116.

[27] "Contiki web page." [Online]. Available: http://www.sics.se/contiki/

[28] Danish Nurses' Organisation's Congress, "Ethical guidelines for nurses," May 2004. [Online]. Available: http://www.dsr.dk/msite/text.asp?id=45&TextID=61

[29] A. Deery, D. Chambers, D. Moriarty, E. Connolly, and G. Lyons, "Clinical trials of a wireless lan based patient monitoring system," in *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, Jun 2006, pp. 479–484.

[30] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based online energy estimation for sensor nodes," in *EmNets'07: Proceedings of the 4th workshop on Embedded networked sensors.* New York, NY, USA: ACM, 2007, pp. 28–32.

[31] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy metering for free: Augmenting switching regulators for real-time monitoring," in *International Conference on Information Processing in Sensor Networks.* Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 283–294.

[32] "Electronic patch website." [Online]. Available: http://www.eplaster.dk/

[33] T. Falck, H. Baldus, J. Espina, and K. Klabunde, "Plug 'n play simplicity for wireless medical body sensors," *Mobile Networks and Applications*, vol. 12, no. 2, pp. 143–153, Jun 2007.

[34] "Firebug." [Online]. Available: http://firebug.sourceforge.net/

[35] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Third IEEE International Conference on Pervasive Computing and Communications Workshops, 2005. (PerCom 2005 Workshops)*, March 2005, pp. 146–150.

[36] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *PLDI'03 – Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation.* ACM Press, Jun 2003, pp. 1–11.

[37] "Ge healthcare: Patient monitoring wireless solutions." [Online]. Available: http://www2.gehealthcare.com/

[38] R. Gusella and S. Zatti, "The accuracy of the clock synchronization achieved by tempo in berkeley unix 4.3 bsd," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 847–853, 1989.

[39] R. G. Haahr, S. Duun, E. V. Thomsen, K. Hoppe, and J. Branebjerg, "A wearable "electronic patch" for wireless continuous monitoring of chronically diseased patients," in *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2008) in conjunction with the 5th International Summer School and Symposium on Medical Devices and Biosensors (ISSS-MDBS 2008)*, June 2008, pp. 66–70.

[40] T. Haenselmann, "An fdl'ed textbook on sensor networks." [Online]. Available: http://pi4.informatik.uni-mannheim.de/~haensel/sn_book/

[41] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Pervasive Computing*, vol. 07, no. 1, pp. 30–39, Jan–Mar 2008.

[42] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen, "PowerBench: A scalable testbed infrastructure for benchmarking power consumption," in *Int. Workshop on Sensor Network Engineering (IWSNE)*, June 2008.

[43] P. A. Heidenreich, C. M. Ruggerio, and B. M. Massie, "Effect of a home monitoring system on hospitalization and resource use for patients with heart failure," *American Heart Journal*, vol. 138, no. 4, pp. 633–640, 1999.

[44] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, Nov 2000.

[45] J.-H. Hoepman, "The ephemeral pairing problem," in *Financial Cryptography*, ser. LNCS, no. 3110. Springer, 2004, pp. 212–226.

[46] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *Ubicomp'01 – Ubiquitous Computing, Proceedings*, ser. LNCS, no. 2201. Springer, Sept 2001, pp. 116–122.

[47] "Hq inc: Cortemp temperature pill." [Online]. Available: http://www.hqinc.net/

[48] *IEEE Standards 802.15.4*, IEEE Computer Society. [Online]. Available: http://www.ieee802.org/15/pub/TG4.html

[49] "iHospital web site." [Online]. Available: http://www.ihospital.dk/

[50] X. Jiang, P. Dutta, D. Culler, and I. Stoica, "Micro power meter for energy monitoring of wireless sensor networks at scale," in *6th International Symposium on Information Processing in Sensor Networks, 2007. (IPSN 2007)*, April 2007, pp. 186–195.

[51] W. D. Jones, "Taking body temperature, inside out," *IEEE Spectrum*, vol. 43, no. 1, pp. 13–15, Jan 2006.

[52] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2:6, Mar 2005.

[53] G. Kramp, M. Kristensen, and J. F. Pedersen, "Physical and digital design of the bluebio biomonitoring system prototype, to be used in emergency medical response," in *Pervasive Health Conference and Workshops, 2006*, Nov–Dec 2006, pp. 1–11.

[54] M. Kristensen, M. Kyng, and E. T. Nielsen, "It support for healthcare professionals acting in major incidents," in *SHI 2005, 3rd Scandinavian conference on Health Informatics*, O. Hejlesen and C. Nøhr, Eds. Virtual Centre for Health Informatics, Aug 2005, pp. 37–41.

[55] M. Kristensen, M. Kyng, and L. Palen, "Participatory design in emergency medical service: designing for future practice," in *CHI'06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA: ACM, 2006, pp. 161–170.

[56] N. J. Kristoffersen, Ed., *Almen Sygepleje 3. Patient og sygeplejerske. Krop, omgivelser og handlinger*. København: G.E.C. Gads Forlag, 1997.

[57] J. Lee and D. R. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," in *Selected Areas in Cryptography*, ser. LNCS. Springer, 2004, no. 3357, pp. 294–307.

[58] P. Leijdekkers and V. Gay, "Personal heart monitoring system using smart phones to detect life threatening arrhythmias," in *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, 2006, pp. 157–164.

[59] M. Leopold, M. B. Dydensborg, and P. Bonnet, "Bluetooth and sensor networks: a reality check," in *SenSys'03 – Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 103–113.

[60] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *NSDI'04: First symposium on networked systems design and implementation*, 2004.

[61] "Lifesync wireless ecg." [Online]. Available: http://www.wirelessecg.com

[62] Y.-H. Lin, I.-C. Jan, P. C.-I. Ko, Y.-Y. Chen, J.-M. Wong, and G.-J. Jan, "A wireless pda-based physiological monitoring system for patient transport," *IEEE Transacions on Information Technology in Biomedicine*, vol. 8, no. 4, pp. 439–447, Dec 2004.

[63] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *International Conference on Information Processing in Sensor Networks, 2008. (IPSN '08)*, April 2008, pp. 245–256.

[64] "Liwas, life warning system." [Online]. Available: http://www.liwas.dk/

[65] Lægeforeningen (Danish Medical Association), "Lægeforeningens politik for sundheds-it (dma's health-care it policy; available in danish only)," April 2008. [Online]. Available: http://www.laeger.dk/portal/page/portal/LAEGERDK/LAEGER_DK/POLITIK/POLITIKPAPIRER/POLITIKPAPIRER_LAEGEFORENINGEN/LF%27s%20politik%20for%20sundheds-it/L%C3%A6geforeningens%20politik%20for%20sundheds-it.pdf

[66] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.* ACM Press, 2002, pp. 88–97.

[67] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *International Workshop on Wearable and Implantable Body Sensor Networks*, Apr 2004. [Online]. Available: http://www.bsn-web.org/public/David_Malan_-_abstract.pdf

[68] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *IEEE SECON'04 – First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks.* IEEE, 2004, pp. 71–80.

[69] K. L. D. J. Malan, T. R. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, “Sensor networks for emegency response: Challenges and opportunities,” *IEEE Pervasive Computing*, vol. 03, no. 4, pp. 16–23, Oct-Dec 2004.

[70] “Mantis web page.” [Online]. Available: http://mantis.cs.colorado.edu

[71] M. Maroti, G. Simon, A. Ledeczi, and J. Sztipanovits, “Shooter localization in urban terrain,” *Computer*, vol. 37, no. 8, pp. 60–61, 2004.

[72] T. Martin, E. Jovanov, and D. Raskovic, “Issues in wearable computing for medical monitoring applications: A case study of a wearable ecg monitoring device,” in *Proceedings of the IEEE Fourth International Symposium on Wearable Computing.* IEEE, Oct 2000, pp. 43–49.

[73] K. Martinez, J. K. Hart, and R. Ong, “Environmental sensor networks,” *Computer*, vol. 37, no. 8, pp. 50–56, 2004.

[74] “Microchip xlp technology home page.” [Online]. Available: http://www.microchip.com/XLP

[75] A. Milenkovic, M. Milenkovic, E. Jovanov, D. Hite, and D. Raskovic, “An environment for runtime power monitoring of wireless sensor network platforms,” in *SSST’05: Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory*, March 2005, pp. 406–410.

[76] O. G. Morchón, H. Baldus, and D. S. Sánchez, “Resource-efficient security for medical body sensor networks,” in *BSN’06 – International Workshop on Wearable and Implantable Body Sensor Networks.* IEEE Computer Society, 2006, pp. 80–83.

[77] “Msp430x15x, msp430x16x, msp430x161x mixed signal microcontroller (slas368e).” [Online]. Available: http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf

[78] “nesc: A programming language for deeply networked systems.” [Online]. Available: http://nescc.sourceforge.net/

[79] J. W. P. Ng, B. P. L. Lo, O. Wells, M. Sloman, N. Peters, A. Darzi, C. Toumazou, and G.-Z. Yang, “Ubiquitous monitoring environment for wearable and implantable sensors (ubimon),” in *UbiComp’04 – The Sixth International Conference on Ubiquitous Computing, Poster Proceedings.* UbiComp’04, 2004. [Online]. Available: http://ubicomp.org/ubicomp2004/adjunct/posters/ng.pdf

[80] “Nonin medical avant 4000.” [Online]. Available: http://www.nonin.com/ProductDetail.aspx?ProductID=12

[81] M. Odersky, P. Altherr, V. Cremet, B. Emir, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, and M. Zenger, *An Introduction to Scala*, 1st ed., Programming Methods Laboratory, EPFL, Switzerland, Aug 2005.

[82] "One laptop per child (olpc)." [Online]. Available: http://www.laptop.org

[83] "PalCom it support in major incident web site." [Online]. Available: http://www.ist-palcom.org/application-areas/major-incidents/

[84] L. Palen and S. Aaløkke, "Of pill boxes and piano benches: "homemade" methods for managing medication," in *CSCW'06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. New York, NY, USA: ACM, 2006, pp. 79–88.

[85] C. C. Y. Poon, Y.-T. Zhang, and S.-D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 73–81, Apr 2006.

[86] E. K. Reilly, E. Carleton, and P. K. Wright, "Thin film piezoelectric energy scavenging systems for long term medical monitoring," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2006, pp. 38–41.

[87] "Bekendtgørelse nr. 528 af 15/06/2000—sikkerhedsbekendtgørelsen (available in danish only)." [Online]. Available: https://www.retsinformation.dk/Forms/R0710.aspx?id=842

[88] "Lov nr. 429 af 31/5/2000—persondataloven (available in danish only)." [Online]. Available: https://www.retsinformation.dk/Forms/R0710.aspx?id=828

[89] "Vejledning nr. 9229 af 29/4/2005—vejledning om sygeplejefaglige optegnelser (available in danish only)." [Online]. Available: https://www.retsinformation.dk/Forms/R0710.aspx?id=10045

[90] "Lovbekendtgørelse nr. 95 af 7/2/2008—sundhedsloven (available in danish only)." [Online]. Available: https://www.retsinformation.dk/Forms/R0710.aspx?id=114054#Kap9

[91] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso, "Experimental evaluation of lifetime bounds for wireless sensor networks," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, Jan 2005, pp. 25–32.

[92] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, "Power sources for wireless sensor networks," in *Wireless Sensor Networks, First European Workshop, EWSN*, ser. LNCS, no. 2920. Springer, 2004, pp. 1–17.

[93] N. Sastry and D. Wagner, "Security considerations for ieee 802.15.4 networks," in *WiSe'04 – Proceedings of the 2004 ACM workshop on Wireless security.* ACM Press, 2004, pp. 32–42.

[94] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, Jan 1991.

[95] N. Schärli, S. Ducasse, O. Nierstrasz, and A. P. Black, "Traits: Composable units of behaviour," in *Proceedings of ECOOP – Object-Oriented Programming*, ser. LNCS, no. 2743. Springer, 2003, pp. 248–274.

[96] "Sentilla home page." [Online]. Available: http://www.sentilla.com

[97] K. Shimizu, "Telemedicine by mobile communication," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 18, no. 4, pp. 32–44, Jul 1999.

[98] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, "Sensor networks for medical care," TR-08-05, Division of Engineering and Applied Sciences, Harvard University, Tech. Rep., 2005.

[99] V. Shnayder, M. Hempstead, B.-R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *SenSys'04: Proceedings of the 2nd international conference on Embedded networked sensor systems.* New York, NY, USA: ACM, 2004, pp. 188–200. [Online]. Available: http://www.eecs.harvard.edu/~shnayder/ptossim/

[100] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, ser. LNCS, no. 1796. Springer, 2000, pp. 172–182.

[101] I. Stark, "Thermal energy harvesting with thermo life," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2006, pp. 19–22.

[102] Sundhedsstyrelsen (The Danish National Board of Health), "Informationssikkerhed – vejledning for sundhedsvæsenet (information security guidance note; available in danish only)," Sundhedsstyrelsen, Feb 2008. [Online]. Available: http://www.sst.dk/publ/Publ2008/SDSD/Infosikkerhed_vejl08.pdf

[103] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, “NanoECC: Testing the limits of elliptic curve cryptography in sensor networks,” in *Wireless Sensor Networks*, ser. LNCS, no. 4913. Springer, 2008, pp. 305–320.

[104] D. S. Sánchez and H. Baldus, “A deterministic pairwise key predistribution scheme for mobile sensor networks,” in *SecureComm’05 – Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks.* IEEE, Sep 2005, pp. 277–288.

[105] “Tinyos.” [Online]. Available: http://www.tinyos.net/

[106] “Tmote sky datasheet.” [Online]. Available: http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf

[107] T. Trathnigg and R. Weiss, “A runtime energy monitoring system for wireless sensor networks,” in *3rd International Symposium on Wireless Pervasive Computing, 2008. (ISWPC 2008)*, May 2008, pp. 21–25.

[108] D. Ungar and R. B. Smith, “Self: The power of simplicity,” Sun Microsystems, Inc., Mountain View, CA, USA, Tech. Rep. SMLI-TR-94-30, 1994.

[109] “Upek home page.” [Online]. Available: http://www.upek.com

[110] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, “Tinypk: securing sensor networks with public key technology,” in *SASN’04 – Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks.* ACM Press, 2004, pp. 59–64.

[111] “Welch allyn telemetry monitoring.” [Online]. Available: http://www.welchallyn.com/wafor/hospital/connectivity.htm

[112] S. M. Wendelken, S. P. McGrath, and G. T. Blike, “Agent based casualty care – a medical expert system for modern triage.” [Online]. Available: http://er3people.dartmouth.edu/abccare/papers/abccare-suzanne.pdf

[113] ——, “A medical assesment algorithm for automated remote triage,” in *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 4, Sep 2003, pp. 3630–3633.

[114] F. L. Wong and F. Stajano, “Multi-channel protocols,” in *Security Protocols*, ser. LNCS, no. 4631. Springer, 2005, pp. 112–127.

[115] ——, “Multichannel security protocols,” *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, Oct 2007.

[116] E. M. Yeatman, “Rotating and gyroscopic mems energy scavenging,” in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2006, pp. 42–45.

[117] *ZigBee Alliance web site*, ZigBee Alliance. [Online]. Available: http://www.zigbee.org

Part II

Papers

BLIG: A New Approach for Sensor Identification, Grouping, and Authorisation in Body Sensor Networks

Jacob Andersen[1] and Jakob E. Bardram[2]

[1] Centre for Pervasive Healthcare, Department of Computer Science, University of Aarhus, Aarhus, Denmark
[2] IT University of Copenhagen, Copenhagen, Denmark

Abstract— **Using body sensor networks (BSN) in critical clinical settings like emergency units in hospitals or in accidents requires that such a network can be deployed, configured, and started in a fast and easy way, while maintaining trust in the network. In this paper we present a novel approach called *BLIG* (Blinking Led Indicated Grouping) for easy deployment of BSNs on patients in critical situations, including mechanisms for uniquely identifying and grouping sensor nodes belonging to a patient in a secure and trusted way. This approach has been designed in close cooperation with users, and easy deployment and ease of use are top priorities. We present an initial implementation and evaluation of the presented technology.**

Keywords— **body sensor network, WPAN, healthcare**

I. Introduction

The goal of wireless body sensor networks (BSNs) is to avoid cables and the drawbacks that come with them; cables are inherently unstable as cables and plugs tend to loosen or break under moderate stress, and cables tend to get tangled into each other. Patients with wired monitoring equipment are fixed to a bed or a chair and cannot easily move, or be moved, around. This has made the use of sensor technology impracticable in critical medical situation where sensors must be deployed in seconds and where patients need to be moved frequently. In our study of medical work in settings like emergency units in hospitals and larger accidents, we have discovered that sensors are not used at all, even though the same users say that sensors in many situations would be very useful [1]. Cables, however, have two very important features: first, the grouping of sensors and peripheral units like displays is *palpable* because you can see which nodes are attached to each other. Second, the sensor network is to a large degree *secure* – it is very difficult for an adversary to gain access to the medical data.

The overall objective of our work is to create a wireless BSN which is easy to deploy and use, while maintaining an appropriate level of security and privacy. In particular, we want a wireless BSN to include the two important features of *palpability* and *security* that a cabled BSN possesses. Such a BSN is especially required in critical medical circumstances like large accidents. Based on our studies of monitoring in e.g. the home of patients, there is clear evidence that these requirements are also valid in such more mundane areas of medical monitoring

This paper presents BLIG (short for: Blinking Led Indicated Grouping), which is an approach to deployment of wireless body sensor networks (BSN) on patients in critical situations, like accidents. We present the requirements for BLIG and explain how the design fulfils these requirements for fast, easy, and secure deployment. We then present the current implementation of BLIG and report from a user evaluation session with professional paramedics trying out the technology. We then conclude the paper and shortly discuss our plans for further work.

A. Problems

Figure 1 illustrates the problems BSNs inherently are dealing with. Throughout healthcare, usability issues are always important, demanded by the critical situations and the quick work pace. Security issues, such as privacy of medical information and logging of events are required by law and ethics. Furthermore, as battery technology is developing at a very slow rate, and energy storage per unit of volume or mass is lower than we would like it to be, we want to conserve power as much as possible and to avoid large and bulky equipment.

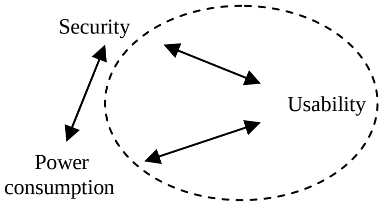


Fig. 1 The BSN problems and their relations. The dashed ellipse encloses the issues addressed by BLIG.

Solving the three problems tend to be problematic, as they conflict with each other, as illustrated by the arrows on the figure: good security properties requires lots of memory and processing power – conflicting with power conservation

– and keys may need to be distributed and users must log in and out, conflicting with simple and easy use. Also when designing the user interfaces, power saving requirements can have a huge impact on what is possible and what is not.

As illustrated by the dashed ellipse on the figure, BLIG was designed to be extremely simple to use, and at the same time have some reasonable security and power properties.

B. Requirements

Our design is grounded in various user-centred design processes ranging from supporting emergency workers involved in large accidents [1], to supporting mobility in hospitals [2], to supporting home care monitoring of patients and elderly people [3,4]. Based on this work we have distilled the following set of requirements:

- **Fast:** Most medical work is time-critical and often done on the move, so deployment of BSNs has to be swift. Data from a train collision emergency exercise showed that only a few seconds are spent with each victim during triage (sorting and prioritising victims). Hence, our goal is that deployment must take *only a few seconds*.
- **Easy:** Deployment takes place in a hectic environment where the users have to attend to many parallel tasks. Hence, the deployment must be very easy. Our goal is to enable users to deploy sensors *using only one hand*, while using the other hand for other things, such as supporting the patient.
- **Resilient and scalable:** The BSN must work reliably in a rugged environment like an outdoor accident scene or in the home, with little or no infrastructure available. Hence, deployment operations must only rely on communication between the actual units involved – i.e. no server or similar network resources should participate. This requirement is to ensure that the basic functionality can be accessed even during a major server or network breakdown. This requirement also ensures a scalable network since network latency will not slow down the process of adding or removing sensors.
- **Secure:** Medical data should be protected and access to it authorised. The goal is that the wireless BSN should not be inferior compared to existing (cabled) technology. The requirements are that communication should be encrypted, only authorised parties can gain access to the network and its data, sensors should be able to provide a security proof, and no unauthorised node should be able to deduce anything about the sensor (apart from its presence). [1]
- **Palpable:** Users need to be able to understand the BSN and investigate its configuration. The goal similarly is that investigating the wireless BSN should be just as easy as investigating how a cabled BSN is set up. The requirement is that users within a few seconds can tell which nodes belong to the same group, i.e. patient.

II. Related work

A method proposed by Baldus et. al. in [5] gives each clinician a personal pen with an infrared transmitter. This pen is then used to arrange sensor nodes on a single patient together in a group. The main problem remains: how can users quickly investigate these groups to inspect which nodes are members? No answer is given to this question.

The CodeBlue project [6] uses service discovery to establish links between sensors and displays. As new sensors are attached to patients, they just pop up at the displays. However, apparently there is no restriction limiting which displays are allowed to show the data, which means that everyone can get access to the information and no logging of who views the data takes place. Furthermore, only a numeric sensor id identifies the data source. This makes it hard to figure out which patient the data originates from. This implies a greater risk of mixing up patients and therefore more time spent checking and double-checking the associations between patient and sensor id.

III. Blig design issues

It is important to realise that the requirements presented above are of a *technical* as well as a *usability* nature. We need a method for BSN identification, pairing, and authorisation which is technically adequate, while ensuring that users can use it quickly, with a high level of confidence, and understand the logical connections.

The BLIG method offers a solution to the usability issues: how to set up sensors in a fast, easy, and palpable way. Along with BLIG, we are developing a corresponding protocol with good security and power consumption properties. Although this protocol is far beyond the scope of this paper, we will give a few hints towards the relation between these properties and BLIG.

An example scenario, using the BLIG at an emergency could be this: an emergency worker wants to add 3 sensors to a patient. He takes the first sensor, turns it on, brings it close to his id-tag and then places it on the patient. He then

[1] It is beyond the limits of this paper to engage in a thorough threat analysis. But a few examples include: Adversaries gaining access to medical information about a patient, adversaries counterfeiting sensor readings and thus causing ill-treatment of the patient, and adversaries jamming the radio channels or launching a Denial-of-Service attack.

takes the next sensor, turns it on, brings it close to his id-tag and brings it close to the first sensor (in arbitrary order), and finally places it on the patient. Likewise for the third sensor (which is brought close to either one of the existing sensors as well as the id-tag). Later when he needs to reuse a sensor, he detaches it from the first patient, turns it off and on (reset) and repeats the above procedure with another patient.

A. Identity of the patient

In most – if not all – existing patient monitoring equipment (such as [7,8]), and many related research projects (such as [6]), the sensors have no information about the identity of the patient. Instead, the sensors simply deliver the data to some (perhaps predefined) destination. It is then the job of the clinician to associate the incoming data at this destination with the identity of the patient. This involves two steps: attach the physical sensor to the patient, and associate the data with the identity of the patient (e.g. by typing in the patient's Social Security Number (SSN) at the monitoring central). If the second step does not take place at the patient's bed, the result is an increased risk of mistakes.

Data produced by the sensors are logically tied to the identity of the patient. However, in many cases the identity of the patient is unknown (e.g. in the accident scenario) and even if known, it is impractical to enter patient names or SSNs into sensors in a BSN. Therefore, in line with the method presented by Baldus et al. [5] we propose to group sensors on a patient together. This way, the identity of the sensor group is not depending on whether the true identity of the patient is known or not. When the identity of the patient is established, a mapping between the patient's "true id" and the sensors' group-id (the shared identity of all sensors on a single patient) can be made, e.g. by adding a PDA with proper communication capabilities to the sensor group and type in the "true id", such as the name or SSN. This procedure is carried out only once for each hospitalised patient – in contrast to once for each time a sensor is attached, changed or the patient is moved, which is the case for current sensors. Furthermore, the procedure would normally be carried out at the bedside, reducing the risk of errors such as patient mix-ups.

In contrast to the method suggested by Baldus et al. [5], however, we do not require the use of a special patient identification node or tag. We are proposing to let the first sensor attached to the patient generate a unique group-id. This id will be replicated to all sensors subsequently added to the patient. This way, not one specific sensor has a special role. This will make the sensor network more robust with respect to single point of failure.

B. Grouping

Once the group-id has been established (using the first sensor node) other sensor nodes can be attached to the BSN group. Hence, we need a method of grouping and ungrouping sensors. This is done by handing over the group-id to the node joining the group. Each sensor has a button for grouping (which could also serve as power button). When a patient has no sensors attached, the first sensor will generate its own unique group-id. Successive sensors are added to the group by pressing the grouping button while holding the new sensor close to any one of the already attached sensors. Note that we can use any node since the group-id is replicated amongst all participating nodes. Short range communication hardware (to be explained later) will be used to add the new sensor to the group of the nearby sensor. By 'short range' we mean no more than about half a meter, i.e. not a radio transceiver. For security reasons, which will be explained below, the user (clinician or emergency worker) doing the grouping must carry an authorisation-node (e.g. part of an id-tag), which shall also participate in the attachment procedure. Removing a sensor from a group is simply a matter of powering this sensor off. When a sensor is removed from the patient the remaining sensors will keep the original shared group-id.

C. Security

The short range communication channel is used to add new sensors to a group. During this procedure the new node must use the short range communication hardware to communicate with both an existing sensor on the patient and an authorisation-node worn by the user. These three nodes (or two, if the new node is the first to be attached to this patient) will cooperate to generate an unforgeable certificate for the new sensor. This certificate will be used by the new sensor to prove to others in or outside the group (e.g. another sensor or a display) that it truly belongs to the group and is in fact placed on the corresponding patient. Assuming "short range"' means less than about half a meter, the procedure places all nodes at the same place at the same time. An audible sound from the user's node during this procedure ensures that we have the user's attention; therefore the certificate is a proof that this user has approved the grouping.

If all users and their authorisation-nodes are trusted, this argument (by transitivity) states that all sensors belonging to the same group (with valid certificates) will in fact be placed on the same patient. Each time the authorisation node participates in an action, it provides a clear audible sound. This will alert the user if an adversary should try to abuse his or her node for an unauthorised action.

Due to the authorisation certificates a user can (perhaps remotely) inspect a group of nodes using e.g. a PDA to investigate who initially deployed the nodes. Furthermore, by (Danish) law [9] clinicians are required to document their actions. Automatically collecting and logging these certificates, the sensor network can now provide this documentation, saving the clinicians a lot of work.

D. Group palpability

In contrast to e.g. Baldus et al. [5], the grouping can be performed using only one hand and the user should be able to deploy a sensor within a few seconds. Our usability goals, however, were more ambitious: the user must also be able to quickly inspect and understand the grouping.

For this purpose, the presented design incorporates a mechanism which allows a sensor to reveal which group it belongs to. Each sensor node has a number of light-emitting diodes (LEDs) in different colours. All nodes in a group can present a synchronised blinking behaviour – i.e. the LEDs on all nodes belonging to one group will slowly and synchronously blink in matching colours. Each group will have its own unique blinking pattern (very long sequence of colours), which helps the user(s) to quickly verify whether all nodes on one patient are correctly grouped and whether the sensors on two (adjacent) patients belong to different groups (hence the name BLIG: Blinking Led Indicated Grouping). Sensors that fall off a patient are easy to re-establish on the correct patient.

To the user, the colour sequence will appear to be random. Technically, the sequence will be a repetitive pattern with a very long period, determined uniquely as a function of the group id and controlled by a common clock maintained between the nodes belonging to a group. This scheme is similar to the Frequency Hop Spread Spectrum (FHSS) radio modulation method used by e.g. Bluetooth (IEEE802.15.1) [10]. Instead of hopping among a set of radio channels, this scheme will hop among a set of colours. Clock drift is compensated by exchanging clock information between nodes. Since clock drifts in the order of up to tens of milliseconds are not a problem and the oscillators are rather accurate, this synchronisation can be a very rare event (i.e. not a scalability issue).

E. Hardware and power saving

The sensors' primary communication uses radio links. Furthermore, the sensors should have hardware for short range communication to be used in grouping and authorisation. Also, the sensors will need at least one button and a group of LEDs in different colours.

In order to save power, the LEDs should not be blinking all the time – only when a new sensor is added and up to a maximum of a few minutes after that. On the scene of a major emergency this behaviour might be unwanted, but then a command could be broadcast in the sensor networks at the emergency site, forcing all sensors to keep blinking until they leave the site. Furthermore, in the development of the protocol care has to be taken towards conserving power as well. Generally, we want the sensors to consume as little power as possible. Authorisation nodes, on the other hand, are allowed to spend more power, since they should only last for a single shift before being recharged. Therefore, radio and short range communication receivers can be switched on all the time and heavy computations can be performed here. This is exploited by the underlying protocol.

Short range communication can be achieved using a number of technologies. The key requirement is that communication can only take place if the devices are in a close proximity of each other – in the order of about half a meter. A number of technologies could be used, including light, sound, infrared light, ultrasound, and electromagnetic induction. For our prototypes, we have chosen induction, and we are not going to concern ourselves further with this issue. However, if BLIG is brought beyond the prototype implementation, further investigations will be necessary. In particular the security properties of each type should be compared: we assume that the local communication is truly *local*, but would it be possible for a technically skilled adversary to establish a "local" communication link at a distance? For instance, if infrared was used instead, could the adversary use an infrared laser from an adjacent room or through a window? Would visible light be better? – A visible light laser would certainly be easier to spot than an infrared laser. Other properties, like power consumption of different technologies, should also be examined.

IV. Prototype

A prototype intended for workshop based proof-of-concept testing has been developed. This prototype implementation was developed only to demonstrate the BLIG grouping mechanism, thus the secure protocol mentioned earlier, which we are also developing, has not been implemented in this prototype yet. The node hardware platform we use is the Berkeley mote of Telos revision B design ("Tmote sky" from Mote*iv*) extended with a coil and a couple of amplifier circuits for the short range communication channel. Figure 2 shows two motes. Mote 7 is a regular sensor node, which will be attached to patients, and mote

200 is a "clinician mote", i.e. it plays the role of the clinician's authorisation-node. This mote also has a piezoelectric buzzer. Figure 3 shows a rough schematic of the extension: two amplifiers built from a few transistors and an op-amp IC takes care of bridging the digital input and output from the mote's microcontroller with the coil. Both amplifiers can independently be turned off and on in order to save power. As of now, we use a very simple frequency modulation technique to transmit a number between 0 and 31 over the communication channel in less than 100 milliseconds. This is more than sufficient for the current prototype, but in the future, as the secure protocol will be implemented, we will have to improve this, in order to get a fast general data packet transfer.

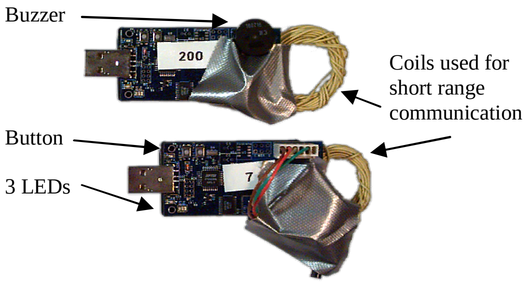


Fig. 2 A clinician mote and a regular sensor mote

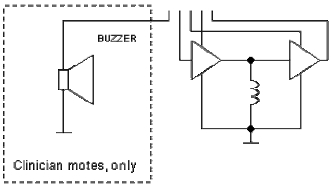


Fig. 3 Schematics of the short range communication extension.

Regular sensor nodes have 4 different states: ungrouped, searching, grouped, and error. When the node is turned on or reset, it enters the ungrouped state. While in this state, all LEDs are off. Pressing the grouping button, the node will enter the searching state for a maximum of 10 seconds, during which the LEDs will be glowing (with smooth intensity variations). If the node meets an authorisation-node on the short range channel within the time limit, it will generate a new group-id and enter the grouped state using this group-id. If the node meets both an authorisation-node and another regular node, which is already in the grouped state, within the time limit, the node will receive the group-id and current time from the other regular node. The node will then enter the grouped state using this group-id and synchronise its clock. If the node does not meet an authorisation-node within the time limit, it will enter the error state. In this state the red LED will be blinking aggressively. As a node enters the grouped state, it will begin blinking slowly. Having 3 coloured LEDs (red, green, and blue), there are 8 different combinations available. The combination chosen at any given time is a function of the group-id and a sequence number, which is incremented at a fixed interval.

Authorisation-nodes work in a straight-forward way: they simply provide a short beep sound from the buzzer each time they encounter a regular node in the searching state on the short range channel.

A PC running a small terminal-based Java program can be used to monitor the network, listing the sensors attached to each "patient" (group-id), and the identity of the user who attached it to the patient. The Java program is also capable of showing sensor readings.

V. Preliminary evaluation

Earlier prototype implementations without the short range communication hardware[2] were demonstrated at two workshops arranged by the PalCom project [11] in the fall of 2005 and the spring of 2006, involving participants from the local fire brigade and police force, emergency workers and trauma centre physicians. The reactions were very positive.

The participating clinicians expressed their appreciation with the easy grouping of the sensors and the tangible way groups can be recognised. Also, it was appreciated that the blinking LEDs are easy to see in all weather conditions, bright daylight and at night (something we never even considered), and that no small displays are involved, making it easy to inspect connections at a distance and without glasses. A few simple scenarios were played around the table, and the clinicians demonstrated that they could easily use and understand this method. They also clearly encouraged us to continue the development.

VI. Future work

At the time of this writing, we are planning to begin involving participants from a hospital ward. This hospital ward has both operating theatres and intensive care.

We are currently planning a new workshop where we are going to test the current version of the prototype described above. At this workshop we will be using people acting as patients and "faked" sensor readings on the displays. We

[2] "Proximity" between two units was emulated by pressing a button on both units

will be testing how quickly the clinicians are able to perform the different tasks, establish an overview of the local sensors and identify errors.

Later on, prototypes equipped with real sensors (ECG, pulse oximeter, blood pressure meter etc.) will be constructed and used in workshops, and perhaps as one part of a larger pilot test planned in the near future at the above mentioned hospital.

The pros and cons of different kinds of local communication hardware should be examined in regard to power consumption and how to ensure true local communication. As mentioned earlier, we have chosen not to look further into this subject at the moment.

The idea of using the blink pattern of LEDs to convey the notion of grouping of devices to an observing human is new and needs to be tested further, involving experts in the human visual perception capabilities. How fast should the colours change? Are some blinking patterns better than others? What if the clinician / emergency worker is colourblind? Would different rhythms be helpful? Could blinking patterns and rhythms for instance be used to convey information about the condition of the patient? In the case of a major accident (like a train derailing) where triage is necessary, could the blinking pattern carry a message about the victim's condition (on a scale from "OK" to "critical care needed")?

More work need to be put into the protocol. In particular, we need to formalise and prove the security properties. We also need to do some more extensive scalability analysis and tests – the technology should be capable of handling rather large-scale emergencies with high densities of victims (think of train collisions or terror attacks).

VII. Conclusion

The idea of grouping all sensors on a patient together is not revolutionising. Others have gone down this road before us [5,12]. Our contribution is to the way this group identity will be made tangible for the user. The user will need a fast, one-button method of adding (and removing) a sensor device to a group, and a way to quickly inspect the correctness of the grouping (all sensors on the same patient are in the same group and the sensors on two different patient do not belong to the same group).

We have presented a new method of setting up networks of medical sensors on and around patients in hospitals or at emergency scenes. Besides easing the monitoring of the patient, this method offers a simple and intuitive way of identifying the patient, Furthermore, we are confident that the underlying protocol, which we are currently developing, will prove to be as secure against hardware (including network) failures and most adversary attacks, as the current (cabled) technology.

Acknowledgements

The research presented in this paper is done as part of the Activity-Based Computing Project (http://activity-based-computing.org/) funded by the Danish Research Council under the NABIIT programme, and as part of the PalCom project under the 6th Framework Programme, Information Society Technologies, Disappearing Computer II (http://www.ist-palcom.org).

References

1. Kristensen M, Kyng M, Nielsen E. T. (2005) IT support for healthcare professionals acting in major incidents, SHI 2005, 3rd Scandinavian conference on Health Informatics, pp 37–41
2. Bardram J E, Bossen C (2005) Mobility Work – The Spatial Dimension of Collaboration at a Hospital, Computer Supported Cooperative Work, 14(2), pp. 131–160 DOI 10.1007/s10606-005-0989-y
3. Bardram J E, Bossen C, Thomsen A (2005) Designing for transformations in collaboration: a study of the deployment of homecare technology, GROUP'05: Proceedings of the 2005 international ACM SIGGROUP conference on supporting group work, pp. 294–303 DOI 10.1145/1099203.1099254
4. Aaløkke S, Bunde-Pedersen J, Bardram J E (2006) Where to, Roberta?: Reflecting on the role of technology in assisted living, NordiCHI'06: Proceedings of the 4th Nordic conference on Human-computer interaction, pp 373–376 DOI 10.1145/1182475.1182515
5. Baldus H, Klabunde K, Müsch G (2004) Reliable Set-Up of Medical Body-Sensor Networks, EWSN'04 1st European Workshop on Wireless Sensor Networks, Springer LNCS 2920, pp. 353–363
6. Shnayder V, Chen B, Lorincz K, Fulford-Jones T R F, Welsh M (2005) Sensor Networks for Medical Care, Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University
7. GE Healthcare: Patient monitoring solutions, www.gehealthcare.com/usen/patient_mon_sys/wireless_and_telemetry/products/telemetry_sys/
8. Welch Allyn telemetry monitoring, monitoring.welchallyn.com/products/wireless/
9. Danish Ministry of Health, guidelines on nursing records (in Danish only), www.retsinfo.dk/DELFIN/HTML/C2005/0922960.htm
10. Haartsen J C, Zürbes S (2002) Frequency hop selection in the Bluetooth radio system, IEEE 7th International Symposium on Spread Spectrum Techniques and Applications, vol 1, pp. 83–87 DOI 10.1109/ISSSTA.2002.1049291
11. PalCom "major incidents" web site, www.ist-palcom.org/examplesOfWork/accidents.php
12. Ng J W P, Lo B P L, Wells O, Sloman M, Peters N, Darzi A, Toumazou C, Yang G-Z (2004) Ubiquitous Monitoring Environment for Wearable and Implantable Sensors (UbiMon), UbiComp'04: The 6th International Conference on Ubiquitous Computing, Poster Proceedings.

Experimental Platform for Usability Testing of Secure Medical Sensor Network Protocols

Jacob Andersen and Benny Lo and Guang-Zhong Yang

Abstract— **Implementing security mechanisms such as access control for clinical use is a challenging research issue in BSN due to its required heterogeneous operating responses ranging from chronic diseases management to emergency care. To ensure the clinical uptake of the BSN technology, appropriately designed security mechanisms are essential. Several experimental sensor network platforms have emerged in recent years targeted for clinical use. However, few of them consider the importance of security issues such as privacy and access control, and how these can impact the usability of the platform, while others develop BSN security without considering how a prototype implementation would be received by clinicians in real-life situations. The purpose of this paper is to present our initial effort in building a flexible experimental platform for providing a basic infrastructure with symmetric AES encryption of sensor and configuration data with suitable user interfaces. The pluggable module provides the protocol for authentication and key generation such that modules with different security properties and respective user interface consequences can be easily compared and evaluated.**

I. INTRODUCTION

Recent advances in sensor network technologies for clinical use have called for the need for simple yet effective techniques for data security management in such networks [1]. Microprocessors on sensor platforms are orders of magnitudes slower than those of desktop computers, thus limiting the sophistication of the algorithms that can be used. On these sensor platforms, one significant problem is the latency introduced to the user interface by lengthy calculations. If an authentication protocol requires, for example, a certificate validation as a result of a clinician's access request, a delay of several seconds can be common before access is granted. If the protocol further demands access to a network server, the latency could be even longer. Therefore, a theoretically appealing solution may never be realisable in practice.

Another major significant issue is that on one hand the network security must be rigid and not allow unauthorised access, but on the other hand – like a typical Hollywood movie – an "access denied" message must come with an "override" button to bypass security in critical situations where proper authentication cannot be obtained.

Because of these problems, it is a great technical challenge to create medical sensor network infrastructures that are secure and at the same time are easy to use and flexible. We recognise a great need to create prototype implementations of secure sensor networks and testing these with the end-users (both clinicians and patients) in realistic scenarios, in addition to the "classical" theoretical and laboratory based experiments.

The main focus of the platform proposed in this paper is the sensor network applications which require frequent user involvement – often under stress, like hospitals and emergencies. Implants and sensors for long-term monitoring are rarely operated under time-critical circumstances and may not provide extensive user interface. Such applications would demand a different treatment.

A. Problematic Issues

Many issues have great impact on practical use of this technology. For the sake of simplicity, we shall only name a few, but a more thorough analysis is provided by [2].

1) Adversaries and threats: Adversaries can generally be grouped in two categories: external and internal. An external adversary can try to gain information about patients and may attempt to affect the treatment plan (in order to harm the patient) by the usual network attacks: eavesdropping, packet injection, man-in-the-middle attacks etc. Internal adversaries include clinicians, patients, and visitors. Clinicians may try to cover up malpractice by hiding or blaming it on someone else. The patient may try to commit insurance fraud, or (e.g. a drug addict) affect medication. Visitors may be anyone else who get access to the patient. This group poses even more difficulties in sensor network security than the external adversaries, as they may have direct access to the user interface of the sensors, and perhaps the ability to tamper with the equipment. We will not consider threats to the network itself, such as denial-of-service attacks.

2) Authentication vs. availability: We already mentioned the problem of creating a system which only allows authorised access but at the same time can be overridden by any clinician in an emergency. A solution must be robust even when confronted with the aforementioned internal adversaries.

3) Off-line operation: As sensor networks may be used in environments where no network infrastructures are in place or available, the basic (local) operation of the sensor network must be able to work off-line. However, extended features, such as remote control of actuators and EHR storage of sensor data, may be offered by network infrastructures when available.

4) No key pre-programming: As we imagine many types of sensors, including disposable ones like patches (measuring e.g. ECG), sterile dressings (monitoring wounds), pills

J. Andersen: Department of Computer Science at the University of Aarhus, Denmark andersen@daimi.au.dk

B. Lo and G.-Z. Yang: Department of Computing at the Imperial College, London, UK {benny.lo,g.z.yang}@imperial.ac.uk

(measuring temperature) etc., we reject key pre-programming as an unfeasible option. Sensors must be ready for use as they leave the factory, we cannot have the IT department of a hospital programming all sensors with different keys or certificates.

II. PLATFORM DESIGN

One of the major questions in the design of this platform was how to make a user interface that is general enough to be useful with different types of sensors. Our answer to this question was BLIG (Blinking LEDs Indicated Grouping), presented in [3]. This user interface requires only a single push-button and a group of LEDs of different colours. In addition, it uses two communication interfaces, a normal radio for long range communication, and a short-range communication link for detection of node proximity. BLIG defines two node types:

A *Sensor-node* is a sensor, actuator (e.g. a medicine pump), storage device (e.g. patient medical history) or personal monitor (e.g. a bedside monitor). Such devices are always logically linked to a patient, and in BLIG all "sensor-nodes" on one patient will be joined together to form a closed group. Devices need at least one push-button and a group of LEDs to perform all basic functions.

Authorisation-nodes are linked to the users (most often clinicians). The node can be embedded in a physical token linked to a user (like a name-tag), in which case the node is always used by that particular user. Alternatively it can be a more powerful device like a PDA, in which case multiple users can use the same device by logging in. Devices need at least one push-button and the means to provide an audible feedback to the user (as the user should never have to focus on this node during operation).

BLIG provides the user with interface operations needed to organise and inspect the groups of sensor-nodes. Furthermore, if X and Y are two authorisation-nodes, and sensor-nodes A, B and C are members of a group, in addition to sharing keys and information about the patient, the nodes would carry some signed messages, for instance: "B is a member of the same group as A – signed X" and "C is a member of the same group as B – signed Y".

Some of the questions we intend to explore with our prototype are: how such messages should be signed; if and when the signatures should be checked as these actions are relatively "expensive" computationally.

Another question is, whether X and Y should be trusted and allowed to perform these actions at all. This question is closely linked to the problem mentioned earlier; what a factory-fresh sensor should trust when key pre-distribution is not feasible. In order to solve the latter, we adopt the following method in our basic prototype design:

A. Establishing Trust

The resurrecting duckling was proposed in [4]. For our prototype, adopting this strategy means that whenever a sensor-node is turned on or reset it will trust the first authorisation-node that it communicates with – and any server this authorisation-node trusts. For our example above, this would imply that if X and Y were able to prove that they are trusted by a server, which was trusted by the authorisation-node that originally created the group, then the operations would be accepted, otherwise not. When, how, and whether these certificate tests should be performed is subject to the concrete protocol.

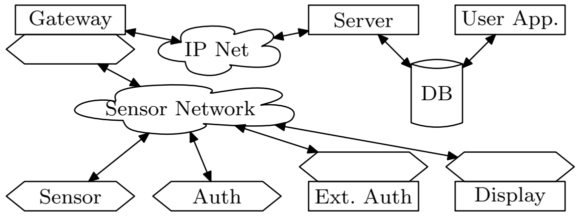


Fig. 1. Prototype building blocks – Hexagons represent motes, rectangles are PCs or PDAs

It is a fundamental assumption that all sensor readings can be delivered (on encrypted links) to any trusted server, and that any command received from a trusted server will be accepted. The security of such servers is beyond the scope of this investigation.

B. Building Blocks

A full workshop prototype consists of a number of different devices, which we shall present briefly here. Figure 1 shows the different device types, and of course several copies of each device type will be used at a workshop.

1) Sensor-nodes: For the evaluation platform, the sensor-nodes will be either real ECG or SpO_2 sensors, emulated sensors (generating fake data), or memory devices intended to store information about the patient. The physical node will be realised as either a BSN mote [5] or a Tmote Sky [6] mote equipped with a push-button, three LEDs of different colours, a short-range communication circuit, and perhaps a sensor circuit.

2) Simple Authorisation-nodes: These nodes resemble a part of the uniform like a name-tag, and will be realised as Tmotes or perhaps BSN motes. The nodes are equipped with a push-button and a buzzer as well as the short-range communication circuit. Depending on the protocol and use-scenario that will be tested, the nodes may be equipped with fingerprint readers as well. As the device has capabilities at least equivalent to a chip card, it is acceptable as login credentials at (Danish) hospitals according to the Danish Hospital IT Guidance Note [7], and since battery must only last for a single shift (8 hours), its computational capabilities exceeds those of a normal sensor-node.

3) Extended Authorisation-nodes: Realised as a Tmote based simple authorisation-node attached to the USB port of a PDA running a Java program. Basically it is an authorisation-node with extra capabilities, such as inspecting a group of sensors (read logs and data) and configuring sensor groups (type in name, patient id / SSN, sensor parameters etc.)

4) Local Displays: Acting as local points of access to the sensors (such as bed-side or watch-type monitors) the displays are realised as regular sensor-nodes (Tmotes) attached to the USB port of a laptop or a PDA executing a Java program. In addition to being able to show sensor-readings and perform configuration tasks, the device will be capable of acting like an authorisation-node to set up private connections to sensor-nodes.

5) Gateways: Just a dumb bridge between the sensor network and the IP network realised as a Tmote attached to a PC.

6) Servers: A server plays the role of a hospital infrastructure or equivalent, whatever that might be (e.g. EHR server). Encrypted (AES) remote access to and from the sensor network passes through this server. Whether a server uses a single global key or one key per sensor group will be protocol dependent. Multiple servers may be used in experiments to test procedures for handing over patients between different domains – e.g. an ambulance to a hospital, or one hospital to another. Servers also record some extra information about the progress of the experiment that may be useful in the evaluation of the workshop results, e.g. exact (millisecond) time-stamps of individual steps (button presses etc.) of the user interaction. A server is realised as a Java application and data is stored in a MySQL database.

7) User Applications: Applications used to browse current and historic sensor readings. The purpose is to provide a remote access to sensor readings simulating a EHR-like environment to the workshop user; it may however be omitted if it is deemed unnecessary in the workshop scenarios, as this is not essential to our investigation. If realised, the applications will be implemented in Microsoft Access.

C. Software Architecture

As one of the main purposes of the platform is to provide a framework that allows experiments with the set-up protocol, this is an independent module in the architecture of the platform. Different protocols can be incorporated in a prototype and switched during a workshop session by a simple command broadcasted in the sensor network. Only sensor- and authorisation-nodes participate in the set-up protocol (the server is affected only on a "preference level"), hence the remaining building blocks presented in the previous section will not be affected by these experiments.

To keep this presentation simple, however, we will describe only the overall software architecture of sensor-nodes as authorisation-nodes are very similar and reuse most of the design and code.

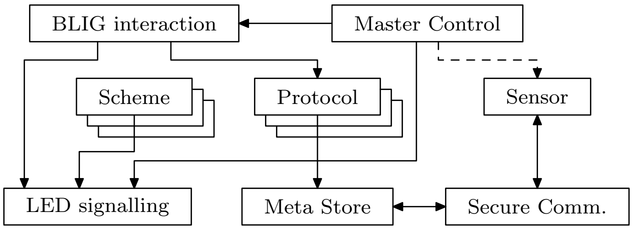


Fig. 2. Sensor-node software architecture

Figure 2 describes the overall structure of the sensor application. Each box represents a functional block consisting of a number of TinyOS 2 components. The main blocks of interest for this presentation are the "`BLIG Interaction`" and "`Protocol`" blocks. As illustrated by the figure, a number of protocols can be compiled into the sensor application, and the current protocol is selected by a command from the `Master Control` block, which handles the overall prototype preferences and receives commands broadcast to the sensors. If a sensor-node is not equipped with a physical sensor (ECG or SpO_2) it can implement a sensor block which can act as different sensors, faking readings. This can also be controlled by remote commands. Finally, as we will also be experimenting with different ways to signal the current state with the LEDs to the user, multiple signalling schemes may be implemented, which can be chosen as well.

When a sensor is successfully added to an existing group, the outcome of the procedure will be a group-id and a group key shared among all nodes in the group (the group-id will be globally unique, and the key will only be known to members of the group). These values will be stored by the protocol in the "`Meta Store`" block. Whenever a new group is created, these same values will be generated from scratch and stored as well, and in addition to these values, a number of server IP addresses and encryption keys (symmetrical AES keys) may be stored in the `Meta Store` block – the imprinting of the duckling.

The `Meta Store` holds a number of different values shared by the nodes of a group. The group-id and key was mentioned above, but also patient name, SSN etc. can be stored. Furthermore, the store contains pairs of IP addresses and encryption keys of trusted servers. When a new node has been added to a group, it receives a copy of the entire store from other nodes of the group, and all nodes keep the store synchronised by dissemination. The "`Secure Communication`" block handles all encrypted communication on the radio interface. This block can only communicate with a peer when its IP address and encryption key is found on the list in the `Meta Store`. Access to read and write all values in the `Meta Store` *except* the group key is granted to any peer, as well as the access to read sensor data and configure the sensor.

III. Discussion

By referencing to the technical challenges raised earlier, what kind of security can we accomplish with the proposed platform?

A. External and Internal Adversaries

A very naïve protocol can assume that the short-range channel is secure by its limited range and transfer secret keys (generated by a random number generator) in plain-text. This would not guard against attacks from internal adversaries, and may not even guard against externals, as they may be able to build a device which can pick up the communication from a long distance.

A better solution would be to use multi-channel security protocols [8]. Assuming that an adversary cannot inject traffic on the short-range link (but may be able to eavesdrop), we can get a good protection against attacks from external adversaries.

Better security against internal adversaries may be obtained at an increased cost of complexity, for instance by signing all messages and using authorisation-node certificates issued by a hospital server.

Security measures at hospitals are very often pragmatic procedures, which can be adapted to the actual threat level. For instance, a celebrity may be guarded to avoid intruders, and on the other hand a drug addict or a suicidal patient may be monitored more closely than other patients. We are planning to mimic this with different protocols implementing different types of security – with different consequences for the usability.

B. Usability Issues and Authentication vs. Availability

In the introduction, the problem of creating simple authentication was raised along with the issue of keeping the network available in emergencies. One possible approach would be to let a sensor network allow full local access to any clinician having an authorisation-node which can provide a certificate issued by a server which the sensor network trusts. In practice this will imply that any clinician at the hospital, at which the patient is admitted, will have full access. Any access will be logged with the staff-id of the clinician, though, so if a clinician, who is not treating the patient accesses the data, it will be noted. This solution fits very well with current practice, but it is somewhat heavy, as the sensors will have to validate the certificates.

Regarding the availability problem, whenever a display device is available, the fundamental design of the platform guarantees that the sensor network (except implants or edible sensors) can easily and quickly be restarted, simply by turning all sensors off – the duckling is killed and resurrected. Current and future readings will be available on the display. However, historic data and other stored information will be lost. This scenario is (at least) not worse than today's technology, as you will always be able to get the current readings in an emergency.

A slightly better solution would be to device a protocol, which will always allow local displays to be attached by anybody, still only showing current readings, but not having to kill the entire network in the process. This is similar to the common situation at hospitals where anyone with physical access to a patient (in principle) would be able to read bedside monitors, but not journal data. However, this security notion would probably be too weak outside a hospital setting. A variant of this solution could be to only allow the local display connection for a limited amount of time, requiring manual reconnection after a timeout.

A protocol variant could also grant anyone permission to add a sensor to an existing network, but sensor readings will be flagged as "untrusted" until confirmed by a trusted user.

IV. CONCLUSIONS AND FUTURE WORKS

We are in the process of building an experimental platform offering quick and easy development of medical sensor network prototypes with different security features for usability testing.

The current status of the platform is that the hardware has been designed and built. The sensor software implementation (TinyOS) is almost complete and tested (a few modules missing). The server and display software is still under early development. An early implementation of the BLIG user interface was tested with users as reported in [3].

V. ACKNOWLEDGMENTS

Elements of the research presented in this paper was done as part of the Activity-Based Computing project and the PalCom project at the University of Aarhus and the SAPHE project at the Imperial College.

The Activity-Based Computing project (http://activity-based-computing.org/) is funded by the Danish Research Council under the NABIIT programme; the PalCom project (http://www.ist-palcom.org) is funded by the EU's 6th Framework Programme; and the SAPHE project (http://vip.doc.ic.ac.uk/saphe) is funded by the UK Technology Strategy Board.

REFERENCES

[1] G.-Z. Yang, Ed., *Body Sensor Networks*. Springer London, 2006.

[2] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Pervasive Computing*, vol. 07, no. 1, pp. 30–39, Jan–Mar 2008.

[3] J. Andersen and J. E. Bardram, "BLIG: A new approach for sensor identification, grouping, and authorisation in body sensor networks," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, ser. IFMBE Proceedings, S. Leonhardt, T. Falck, and P. Mähönen, Eds., vol. 13. Berlin: Springer, Mar 2007, pp. 223–228.

[4] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, ser. LNCS, no. 1796. Springer, 2000, pp. 172–182.

[5] "Bsn node specifications," bsn-web.org/index.php?article=926.

[6] "Sentilla home page," www.sentilla.com.

[7] Sundhedsstyrelsen (The Danish National Board of Health), "IT-sikkerhedsvejledning for sygehuse (Hospital IT guidance note – available in danish only)," www.sst.dk/publ/Publ2002/IT_sikkh_sgh_korr.pdf, Sundhedsstyrelsen, Jul 2002.

[8] F. L. Wong and F. Stajano, "Multichannel security protocols," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, Oct 2007.

Energy Bucket: a Tool for Power Profiling and Debugging of Sensor Nodes

Jacob Andersen and Morten Tranberg Hansen
Department of Computer Science
Aarhus University
Aarhus, Denmark
Email: {jacand,mth}@cs.au.dk

Abstract—**The ability to precisely measure and compare energy consumption and relate this to particular parts of programs is a recurring theme in sensor network research. This paper presents the *Energy Bucket*, a low-cost tool designed for quick empirical measurements of energy consumptions across 5 decades of current draw. The Energy Bucket provides a light-weight state API for the target system, which facilitates easy score-keeping of energy consumption between different parts of a target program. We demonstrate how this tool can be used to discover programming errors and debug sensor network applications. Furthermore, we show how this tool, together with the target system API, offers a very detailed analysis of where energy is spent in an application, which proves to be very useful when comparing alternative implementations or validating theoretical energy consumption models.**

Index Terms—**power profiling, debug, tool, sensor network.**

I. Introduction

Energy efficiency is one of the main concerns in the development of sensor networks. The resource restricted sensor nodes have limited energy supplies, and in order to increase their lifetime, energy efficiency has been considered in many aspects of sensor network research, from platform design [1], to link layers [2], to network layers [3] and applications [4]. This motivates the need for an instrument that offers empirical energy measurements, ranging from detailed evaluations and comparisons of algorithms to verifications of energy consumption models. Such an instrument should not only be used for benchmarking of the final application or protocol implementation; instead, the instrument should be available throughout the development process, instantly showing improvements—or revealing setbacks and errors.

The goal of this work has been to create a tool which will be useful for the sensor network programmer. It must be easily integrated into the development cycle of coding, compiling, loading, executing, and evaluating. Furthermore, it must be low-cost and small, so that it can be used on the desktop—or wherever the programmer wants to use it. The use of big and expensive equipment (such as oscilloscopes) discourages this type of use and leads to experiments in a lab detached from the development cycle. This means that the programmer will not benefit from an instant energy profile of minor code updates, which may reveal intricate information about the running program.

Most energy measurement solutions samples the current through the target system every small time interval in order to produce a graphical representation of the current as a function of time. This leads to enormous amounts of data, but often the question, which the programmer seeks to answer, is something like "how much power did this section of the application consume?" or "under which conditions will protocol A be more energy efficient than protocol B?". To answer such questions, a single scalar value showing the total energy consumption would suffice; perhaps supported by a low-resolution graph for clarity. Hence, all the tiny details picked up with the conventional methods (using, e.g., an oscilloscope) are only occasionally useful.

We claim that often the programmer will benefit more from a tool which offers a concise overview of the energy consumption in chosen parts of the running program as a simple table—where the relevant parts are selected using annotations in the source code. This information can be more useful than a high-resolution (current vs. time) data set, in which the programmer would manually have to identify the relevant time intervals and perform the necessary integrations to obtain the same results.

The *Energy Bucket* is an energy meter designed specifically to be used for sensor network programming. The tool is accompanied by a low memory footprint library which allows the programmer to easily annotate sections of the target program with state numbers. The Energy Bucket will then report the amount of energy spent in each state. To speed up the development process, the use of the Energy Bucket may be incorporated in the build system to launch an energy measurement right after installing a program on a sensor node.

In section II we review related work and in section III we describe and evaluate the precision of the Energy Bucket. In section IV we demonstrate the usage of the tool with three case studies before we conclude and describe future work in section V.

II. Related Work

Most energy measurements found in sensor network research has been performed in the classic lab setup using a digital storage oscilloscope, a specially designed data acquisition board [5], or even a sensor node [6] to measure the voltage over a shunt resistor in series with the target system, followed by integration using some software, typically MATLAB. There are, however, quite a few alternative energy measurement methods designed for a number of different purposes.

One of these alternative methods, presented in [7], was designed for an accelerated evaluation of battery lifetime for sensor node programs. Instead of powering the node from a battery cell, a very large capacitor is used. The node will discharge the capacitor while executing the target program and die when the capacitor is depleted. The lifetime of the application is measured and used to predict the lifetime when using a real battery.

Another method, explored in [8], is the use of a clamp ammeter (a.k.a. tong-tester). This method has the advantage of being completely unobtrusive. Unfortunately, it is also very susceptible to noise. The setup used in [8] was only capable of measuring 2 decades of current (0.4 mA to 40 mA) before hitting the noise-floor.

In a number of situations, a sensor node will benefit from being capable of measuring its own energy consumption—for instance routing protocols may use this information to route packets around nodes with almost depleted batteries. The *iCount* presented in [9] may be used on sensor nodes that use a typical DC-DC boost converter in its power supply. This is basically just a matter of connecting the control signal of the boost controller IC to a counter input of the microcontroller (MCU). This adds energy measurement to the sensor node for "free"—in the sense that if the MCU has an unused counter input and if the sensor node needs a boost converter anyway, no overhead is added. The accuracy is only within 20 % which makes it unsuitable when precise energy measurements are needed.

More accurate measurements can be obtained using SPOT [10] (or the similar solution in [11]). The SPOT is a sensor board containing a complete and very accurate energy meter that can be read using the sensor node I^2C bus. This solution also enables the sensor node to measure its own energy consumption, but the problem using it in a deployment is that the energy meter consumes a fair amount of power itself.

Rather than measuring the energy consumption on actual hardware, it is possible to completely simulate it. PowerTOSSIM [12] is one way of realising such simulations. Based on detailed measurements of the current draw for each component (MCU, radio, flash, LEDs, sensors) in each mode (sleep, active, etc.), the energy consumption may be calculated for each node in the simulation. This approach has the advantage that it can be performed entirely in software on the developer's workstation, avoiding the need for any measurement equipment.

When it is possible to simulate the energy consumption, why not perform the simulation on the sensor node itself? This is the basic idea of [13], where a sensor OS is modified slightly to monitor the state of each hardware component, keeping a count of the total energy consumption—with only a small computational overhead. This has the advantage over PowerTOSSIM that the simulation is performed in a realistic environment which may capture real-life phenomena that are not captured by the simulated world of PowerTOSSIM. This can also be used as an iCount [9] alternative to keep track of remaining energy in a deployed network.

Although all of the above methods *can* be used during development, it will be quite tedious and laborious when a programmer needs a concise overview of the energy spent in different program parts instead of the total energy consumption.

[14] explores a way to create an energy consumption overview on the process level of Linux-based sensors. The kernel task scheduler is modified to keep energy accounts for each process. The energy spent by each process may be monitored in real-time by the user through 'etop' (a modified version of the 'top' utility) which lists the energy consumption and actual current draw for each running process.

III. The Energy Bucket

As explained in the introduction, we want an energy meter which can produce a simple table of the energy consumption of different sections of a program—based on source code annotations. The SPOT [10] solution mentioned above can be adapted to do this; however, this solution may interfere with the program under test in a couple of ways. First, the I^2C bus, which is used to fetch the readings, is also used for radio and external flash communication on many platforms. Second, all the extra logic required to manage and store the readings (and transmit them to a host) may even end up spending significant time and energy compared to the program being tested. Furthermore, we need a tool that is usable with multiple hardware platforms, rather than being a "sensor board" designed for one particular hardware platform.

The Energy Bucket hardware delivers a constant voltage of 3.0 V to the target system while measuring the delivered charge (note that energy equals charge times voltage). In addition to the power supply output, the Energy Bucket has three high-impedance digital input lines which may be used to read the state of the target program (the particular choice of 3 inputs was arbitrary, and more may be added later).

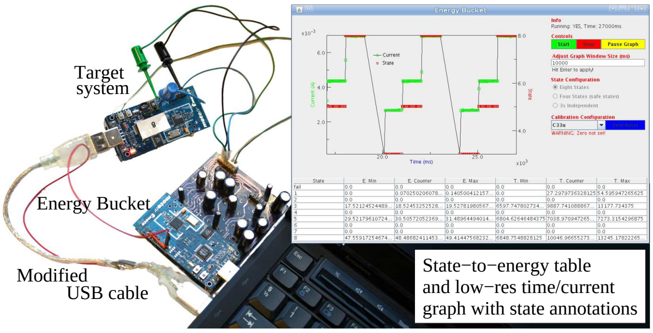
A small TinyOS 2.x [15] library exporting the following standard C function through a header file, may be included anywhere in the target program:

```
void energy_state(const energy_state_t state);
```

At compile-time, either the real library or an alternative skeleton (with no implementation) may be chosen. The real library outputs the chosen state (0–7) to three general I/O pins on the MCU which are connected to the three inputs lines on the Energy Bucket. Most sensor prototype platforms have plenty of general I/O lines available, so this requirement should be easily met.

A. Measurement Setup

Figure 1 shows a typical measurement setup. The target system is connected to the development computer for easy re-programming through a modified USB cable, and to the Energy Bucket in order to collect online measurement data. The USB cable is disabled by the Energy Bucket when running



Info
Running: YES, Time: 27000ms.

Controls

Adjust Graph Window Size (ms)
10000
Hit Enter to apply!

State Configuration
● Eight States
○ Four States (safe states)
○ 3x Independent

Calibration Configuration
C33u
WARNING: Zero not set!

State	E. Min	E. Counter	E. Max	T. Min	T. Counter	T. Max
fail	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.070250206078...	0.140500412157...	0.0	27.2979736328125	54.595947265625
2	0.0	0.0	0.0	0.0	0.0	0.0
3	17.52124524489...	18.52453252528...	19.52781980567...	6597.747802734...	9887.741088867...	13177.734375
4	0.0	0.0	0.0	0.0	0.0	0.0
5	29.52179610724...	30.50572052369...	31.48964494014...	6804.62646484375	7038.970947265...	7273.3154296875
6	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0
8	47.55917254674...	48.48682411453...	49.41447568232...	6848.748828125	10046.96655273...	13245.17822265...

Figure 1. Using the Energy Bucket

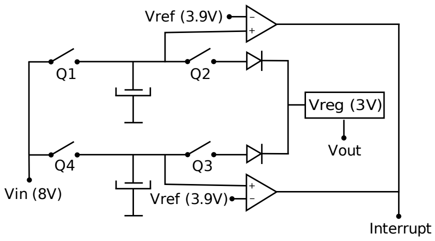


Figure 2. Overview schematic

so that the target system does not draw any current from the USB connection. This setup enables easy integration of the Energy Bucket in the previous mentioned development cycle. Figure 1 also shows how the developer can follow the online current draw and a charge to state table on the development computer.

B. Energy Measurement Method

Sensor node power profiles are characterised by long periods of ultra-low current combined with short bursts of high(er) current. As both types may contribute significantly to the overall energy consumption, neither can be ignored in the measurements [10].

The common method of sampling the voltage over a shunt resistor using some A-D conversion (e.g., a digital storage oscilloscope) requires a subsequent integration of the data. Hence, this method yields precise results only when the ADC's voltage uncertainty is negligible compared to the minimal voltage over the shunt resistor, and the ADC's time period is negligible compared to the width of the bursts. As sensor node current consumption ranges over 5 decades (1 μA – 100 mA), this implies that a (linear) ADC must have a voltage resolution of 18 bits or more, which makes most affordable oscilloscopes (with the typical 8 bits resolution) unsuitable for this task.

The approach taken by the Energy Bucket is to count the number of charge/discharge cycles of a buffer capacitor. The Energy Bucket precisely controls the voltages which the capacitor is charged to and allowed to discharge to. Thus, each cycle will always transfer the same amount of charge—one "bucketful". Furthermore, by keeping the output voltage at a fixed level, each bucketful of charge equals a bucketful of energy, hence the name "Energy Bucket".

Figure 2 shows a block schematic of the Energy Bucket hardware. The four switch symbols represent transistors controlled by the Energy Bucket software running on a Tmote Sky [16]. Two identical electrolytic capacitors are used as buffer caps, so that one can be charged to 8.0 V, while current to the target circuitry is drawn from the other cap. When the voltage over the discharging cap falls to 3.9 V, the two caps are switched. A comparator for each buffer cap detects when it is time to trigger a switching and signals this to the Tmote. A voltage regulator adjusts the output voltage down to 3.0 V. Apart from the Tmote Sky, all parts used are common low-cost off-the-shelf components with a total price of around €50.

Currently, all switching events are time-stamped and sent to the host computer, on which a Java program will perform all calculations. This, however, causes a bottleneck at the serial communication between the Tmote Sky and the host computer, as the maximum UART baud rate is 115,200. Each packet is currently 10 bytes long, so this limits the Energy Bucket to 1152 switchings per second. Since the measured frequency is proportional to the current draw of the target system, which ranges over about 5 decades, this implies that the worstcase time between switchings may be as long as 2 minutes. This is not a problem, if all we want to do, is to measure the overall energy consumption. Still, it becomes difficult to measure the time and energy spent in each individual state, when significant time elapses between the readings. The current solution to this problem is simple: when a state change happened during the elapsed interval, half of the total time and energy of this interval is assigned to the new state and the previous state respectively. However, the Energy Bucket also keeps track of minimum and maximum energy and time measurements for each state and these counters will be increased by either zero or the full time and energy of the interval. We will know for sure, that the real energy consumption and time will be within these "confidence intervals", so if these intervals are reasonable narrow, the result is acceptable. In the Future Work section, we propose yet another approach to fix this problem.

The Energy Bucket circuitry was dimensioned to deliver currents up to 150 mA. In order to reach this current, the minimal capacitance is given by:

$$C = \frac{150 \text{ mA}}{1152 \text{ Hz} \cdot 4.1 \text{ V}} = 31.8 \ \mu\text{F}$$

and we chose to use 33 μF capacitors. Alternatively, if we performed all calculations locally on the Tmote Sky, the UART limitation would vanish, and the maximum frequency would be many times higher, resulting in a much improved temporal resolution.

C. Evaluation and Calibration

In order to calibrate and evaluate the accuracy of the Energy Bucket, we adjusted the output to 3.0 V (using a regular low-cost multimeter) and performed 49 measurements with different fixed combinations of resistors (0.1 % tolerance) in the range 20 Ω–30 MΩ, corresponding to 0.1 μA–150 mA.

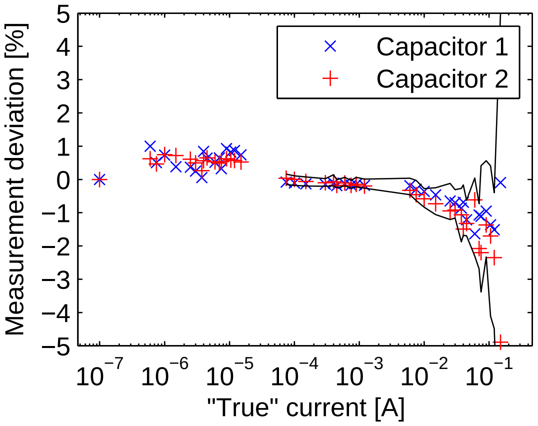


Figure 3. Evaluation results

We used 30 out of these 49 measurements (those in the 1 μA–1 mA interval) for calibration, while all measurements were used to evaluate the resulting calibration.

We chose to perform two separate calibrations, one for each capacitor, as small differences between the capacitors and other components in each of the two circuit paths can be expected. Hence, from each measurement dataset from resistors under 100 kΩ, we picked 100 switching timestamps—50 for each capacitor; and from the remaining (much smaller) datasets, we picked only two timestams, i.e., 1 per capacitor, as the duration for each discharge grows to almost half an hour for the 30 MΩ resistor.

As the total charge is expected to be proportional to the number of switchings (a fixed amount of charge, Q, is transfered each time), we expected the current (which is always a constant in these measurements) between switchings to be proportional to $1/t$, where t is the time between the two switchings, or at least a linear function, $I(t) = Q \cdot (1/t) - I_0$, where I_0 is a constant current leak caused by component imperfections. Analysing the data, we found that this was not exactly the case. A double logarithmic plot of the data showed that in fact the current between switchings is a function $I(t) = Q \cdot (1/t)^e - I_0$ where the exponent, e, is a constant slightly smaller than 1. Experiments indicated that capacitors of same type and value have almost identical e values, while different capacitors (either different type *or* different value) can have significantly different $(1 - e)$ values.

Using the 30 measurements with currents in the interval 1 μA–1 mA, we get the values for Q and e, and using the 30 Ω measurement, we get the value for $I_0 = -17$ nA, i.e., a small current is leaking *into* the circuit:

$$\begin{aligned} I_1(t) &= 144.7\ \mu C \ \cdot\ (\tfrac{1}{t})^{0.9897} + 17.3\ nA \\ I_2(t) &= 146.7\ \mu C \ \cdot\ (\tfrac{1}{t})^{0.9892} + 17.8\ nA \end{aligned}$$

A plot of the percentual deviation of the measured current from the "true" current is shown in figure 3. As mentioned earlier, each measurement point for resistors below 100 kΩ represent an average of 50 switching timestamps. The black lines show the minimal and maximal value among each of the 100 values of both capacitors. The "true" current values are really calculated as 3.0 V/R, where R is the resistor value used, and it is an implicit assumption that the output voltage is constant. This assumption is not accurate, however. In fact, under high load (>100 mA) the voltage fluctuates a bit and may even drop as low as 2.8 V—which is a 6 % drop in voltage giving rise to up to a 6 % error in the "true" current as well. This explains the reduced accuracy and general drop in the curves of figure 3, and from the measurements shown on this graph, we can conclude, that the Energy Bucket measures *current* and *charge consumption* within ±2 % or better, over more than five decades of current consumption (1 μA–100 mA). The instrument also measures *power* and *energy consumption*, but due to saturation of the output voltage regulator, the accuracy drops a bit when the current draw exceeds 50 mA.

Component tolerances—as high as 20 % for the buffer capacitors—suggest that environment (primarily ambient temperature) and ageing effects should be considered. All our experiments were carried out in an office under normal room temperature conditions. Each time the Energy Bucket is used, we also make a few test measurements using the reference resistors in order to check the calibrations, and so far no discrepancies have been observed. As the device is only 6 months old, ageing effects may still appear in the future. However, as the primary purpose of the tool is *not* to deliver accurate absolute measurements, but rather to do comparative studies (such as comparing the energy consumption of different implementations of a component), using the latest calibration will be sufficient most of the time. Redoing the calibration would be recommended prior to any measurement where absolute accuracy is essential.

IV. Case Studies

In this section we demonstrate the use of the Energy Bucket as a tool for debugging, comparing alternative implementations, and validating energy consumption models.

A. Debugging: Telos A vs. Telos B

In order to back up our claim that measuring the energy consumption should be a fundamental part of writing and debugging programs, we present a concrete case where the Energy Bucket—in fact, the first time it was ever used—revealed a programming flaw in an application, which had gone unnoticed for more than 6 months.

One of the authors developed a communication library containing some extremely timing-critical parts, and in order to debug some library procedures, two digital signals were output to a multi-channel oscilloscope. The use of these outputs would be enabled by a precompiler constant, defined in the Makefile.

The library was developed and tested on Tmote Sky motes, which is a telos revision B design [16]. The library, as well as an application using the library, was tested on this platform as well, including a test of its energy consumption—which was within the design limits. The application was, however, deployed on a mixture of Tmote Sky and BSN [17] motes—the latter being a telos revision A design.

After installing the application on a BSN mote, the Energy Bucket revealed that this node was drawing 20 mA more current than expected. We proceeded to use the Energy Bucket in an iterative “alter code, compile, load, execute and measure” process in order to narrow down the reason for this current drain. The error turned out to be in the Makefile, as the above-mentioned precompiler constant definition was never deleted. One of the debug ports used was port 6.7 on the MSP430 MCU, which is exported for external use in the Telos revision B design. On the Telos revision A design, however, this port is connected directly to the positive supply rail, so a low output on this port caused a short-circuit.

Since the application worked perfectly well on both platforms, and we had no plans of performing extra lab tests on the BSN mote, since the application had already been thoroughly tested on Tmote Sky motes, this bug would probably have gone unnoticed, if we did not have this tool. Furthermore, this experience proved, that having instant energy measurements available while debugging (and programming in general) is indeed a very powerful tool, as the power fingerprint of an application may reveal a lot more about what is going on, than three LEDs.

B. Comparison: CC2420 vs. MSP430 CCM security

Sensor network researchers and developers often evaluate novel problem solutions by comparing them to previously proposed ones [4]. The dominant performance metrics in such comparisons include time and energy: time measurements are important when estimating the throughput of a routing or data processing algorithm and energy measurements are important with regard to lifetimes. In this section we show how to use the Energy Bucket to compare the energy and time consumption of the CC2420 radio inline Counter and Cipher Block Chaining Message Authentication Code (CCM) mode security mechanism [18] to a similar software implementation[1] when used with packet transmission. It is known that the CC2420 inline CCM security mechanism will outperform a similar software implementation with relation to a time metric [19], but the fact that the radio could be turned off when doing the software security operation could favour it with relation to an energy metric.

The CC2420 radio inline CCM security mechanism works on packets already present in the CC2420 transmit and receive buffers. Doing packet transmission the read/write to these buffers needs to be done anyway, so the security overhead only consist of initialising the security (setting keys and nonce) and performing the actual encryption/decryption.

We implemented the CC2420 inline CCM security operation and ported the similar software implementation to TinyOS 2.x [15] running on an MSP430 MCU based Tmote Sky [16] connected to the Energy Bucket that recorded energy consumption for the interesting program states. Table I shows a comparison of the measured energy and time consumption with confidence intervals for one encryption using the two different implementations. Each value is an average of 12000 encryption operations. The CC2420 inline CCM security implementation makes use of the default TinyOS 2.x CC2420 radio stack which puts the radio in receive mode when on. This favors the software implementation as the CC2420 security operations only require the radios oscillator to run.

Table I
Energy Bucket CCM evaluation results

	Energy [mJ]			Time [ms]		
	min		max	min		max
CC2420	10.05	11.13	12.21	0.17	0.21	0.24
MSP430	50.13	50.37	50.58	8.56	8.62	8.68

Table II
States

State	Description
Starting	Start the radio’s voltage regulator, start the radio’s oscillator, wait for oscillator to stabilize, enable receive state.
Transmitting	Set packet headers and transmission power, write packet to the radio’s transmit buffer, wait an initial back-off time, do clear channel assessment, transmit the packet, wait for acknowledgment (optional).
Stopping	Stop the radio’s voltage regulator.

We see that according to the time measures the CC2420 radio inline CCM security mechanism is $8.62/0.21 = 42$ times faster and uses $50.37/11.13 = 4.5$ times less energy than the favored similar software implementation running on the MSP430 MCU. To verify the time measures we did another experiment in which we measured the duration of one encryption operation using the MCU’s microsecond timer. Note that these time measures also verifies the related energy measures as time and energy is allocated to the states in the same way. Averaged over 250 encryptions, the CC2420 inline CCM security mechanism and the similar software implementation took 0.189 ms. and 8.68 ms, respectively. These values are within the confidence intervals of the times measured by the Energy Bucket.

C. Model validation: Packet transmission

Evaluating sensor network programs with regard to energy efficiency is often done from an energy consumption model of the program [4], [12], [13]. The model divides the program into a set of fixed states, S_i, with an associated current, I_i, that can be derived from experimental evaluation or datasheet lookups. The total charge consumption of the program is then calculated from the time, T_i, spent in each state:

$$Q = \sum_i T_i \cdot I_i$$

In this section we show how we used the Energy Bucket to validate an energy consumption model.

Inspired by a problem from the SensoByg project [20] we evaluate the energy consumption of a single sensor node that periodically broadcasts its acquired data to a potential receiver.

[1] http://gladman.plushost.co.uk/oldsite/AES/

Table III
MODEL VALIDATION RESULTS

Model	NoAck	AckReliable	AckUnreliable
Starting time	2.64 ms	2.64 ms	2.64 ms
Transmitting time	10.20 ms	12.45 ms	18.02 ms
Stopping time	0.21 ms	0.21 ms	0.21 ms
Calculated charge*	0.23 mC	0.27 mC	0.37 mC
Energy Bucket	NoAck	AckReliable	AckUnreliable
Measured charge*	0.20 mC	0.24 mC	0.34 mC
Deviation	NoAck	AckReliable	AckUnreliable
Absolute	17.36 %	12.82 %	8.53 %
Corrected	-2.74 %	-3.75 %	-3.22 %

* per sent packet

For the sake of simplicity, we leave out the sensor readings and instead transmit a static 28 byte data packet. The transmission of the data packet consists of starting the radio, transmitting the packet, and stopping the radio again. The application was implemented in TinyOS 2.x [15] running on a Tmote Sky [16]. The states of the program are described in detail in Table II.

The transmission of a packet can be done with or without an acknowledgment. When enabling acknowledgments, the time spent in the *transmitting* state depends on the time the sensor node has to wait for the acknowledgment. We derive three variations of the model: one without the use of acknowledgments (NoAcks), one where a receiver acknowledge the packet immediately using software acknowledgments (AcksReliable), and one where the expected acknowledgment from the receiver is lost (AcksUnreliable).

We measured the time spent in each state for the three variations of the model using the MCU's microsecond timer and calculated the charge consumption per transmitted packet based on the currents listed in the CC2420 radio datasheet [18]. The TinyOS 2.x CC2420 radio stack implementation puts the radio into receive mode when started, so we let the current in the *starting* and *stopping* states be the current of the radio in receive mode (19.7mA) and the current in the *transmitting* state be the current of the radio in transmit mode (17.4mA). Note that this is an overestimate of the current in the *starting* and *stopping* states as the radio will not be fully on the entire time. On the other hand, this could be neutralised by the smaller current drawn by the MCU, not included in the model, and the underestimate of the *transmitting* state as it is inevitable that the radio will not spend some time in receive mode here. The first part of Table III shows the time measure for each state and the calculated charge consumption per packet (time multiplied by the theoretical current) for the three variations of the model. The time values are averaged over 1000 sent packets with a variance of zero for the starting and stopping times and 7.58, 11.84, and 7.51 for the NoAck, AckReliable, and AckUnreliable transmitting times, respectively. These variances are due to the randomized initial backoff time and delay in the acknowledgement (for AckReliable). The experimental values from the Energy Bucket shown in the second part of Table III is the average charge consumption per packet calculated from the consumption of a sensor node transmitting 1000 packets.

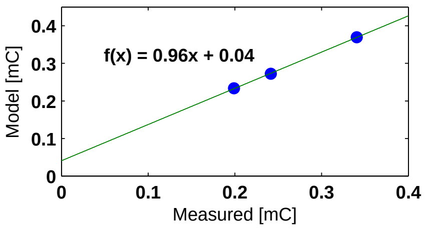


Figure 4. Relation between model and measured charge

Our experiments show that the calculated charge consumption values from the model deviate from the measured values. In Figure 4 we plot the relation between the model and measured values and see that the regression curve shows a constant deviation of 0.04 mC. This has to be due to the overestimate of the charge consumption in the *starting* and *stopping* states and explains why the absolute deviation shown in Table III is decreasing with the increasing transmission time. If we correct the model with this constant factor the corrected deviation between the model and the measurement stays within an acceptable 4% (c.f. Table III) and we conclude that this corrected model is valid.

V. CONCLUSION AND FUTURE WORK

We designed and evaluated the Energy Bucket, a tool for relating energy consumption to parts of a target program by measuring the energy usage. In case studies, we demonstrated that the Energy Bucket is a valuable tool for programmers when writing and debugging programs, comparing alternative implementations, and validating energy consumption models. The case studies emphasized the benefits of having a tool that offers instant power profiling and can be integrated into the development cycle. The tool facilitates early programming error discovery and conveys a better understanding of the target system's behaviour.

The Energy Bucket achieved an accuracy of less than 2 % over five decades of current draw. However, the accuracy of the energy to state measured by the Energy Bucket depends on the time resolution of the measurements. Using smaller capacitors will increase the resolution, but the current system has performance limitations with regard to how fast it can alternate between the capacitors.

Future work include optimizing the Energy Bucket for performance by reducing the packet size for the serial connection or by eliminating online serial communication by calculating the charges used in each state on the Tmote Sky instead of the host computer. In situations where very high time resolution is required the capacitor switching could be implemented in hardware to decrease the source of error caused by the switching delay. Using hardware-based switching and feeding the switching signal to a timer input on the MCU, the maximum switching frequency could be as high as 10 MHz on the current MCU [21]. Choosing capacitors such that, say, 5 MHz would correspond to 100 mA, a current draw of 2 μA would correspond to 100 Hz. An orthogonal approach to improve the accuracy of the energy to state measurements is to force a capacitor switch at each state change. This requires a way to measure the unused charge remaining in the capacitors when they are switched out—e.g., using one of the ADCs available on the MCU. This solution would be highly accurate, similar to the approach used in [22].

Acknowledgement

The research presented in this paper was partially funded by the Activity-Based Computing project (http://activity-based-computing.org/) and the SensoByg project (http://www.sensobyg.dk/english/) at Aarhus University.

References

[1] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 48.

[2] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 95–107.

[3] K. Lin and P. Levis, "Data discovery and dissemination with dip," in *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 433–444.

[4] R. Musaloiu-E, C.-J. Liang, and A. Terzis, "Koala: Ultra-low power data retrieval in wireless sensor networks," *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pp. 421–432, April 2008.

[5] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen, "PowerBench: A Scalable Testbed Infrastructure for Benchmarking Power Consumption," in *Int. Workshop on Sensor Network Engineering (IWSNE)*, 2008.

[6] L. Selavo, G. Zhou, and J. Stankovic, "Seemote: In-situ visualization and logging device for wireless sensor networks," Oct. 2006, pp. 1–9.

[7] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso, "Experimental evaluation of lifetime bounds for wireless sensor networks," *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pp. 25–32, Jan.-2 Feb. 2005.

[8] A. Milenkovic, M. Milenkovic, E. Jovanov, D. Hite, and D. Raskovic, "An environment for runtime power monitoring of wireless sensor network platforms," *System Theory, 2005. SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on*, pp. 406–410, March 2005.

[9] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy metering for free: Augmenting switching regulators for real-time monitoring," *International Conference on Information Processing in Sensor Networks*, vol. 0, pp. 283–294, 2008.

[10] X. Jiang, P. Dutta, D. Culler, and I. Stoica, "Micro power meter for energy monitoring of wireless sensor networks at scale," *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pp. 186–195, April 2007.

[11] T. Trathnigg and R. Weiss, "A runtime energy monitoring system for wireless sensor networks," *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, pp. 21–25, May 2008.

[12] V. Shnayder, M. Hempstead, B.-R. Chen, and M. Welsh, "PowerTOSSIM: Efficient power simulation for tinyos applications," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004. [Online]. Available: http://www.eecs.harvard.edu/~shnayder/ptossim/

[13] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*. New York, NY, USA: ACM, 2007, pp. 28–32.

[14] T. Stathopoulos, D. McIntire, and W. Kaiser, "The energy endoscope: Real-time detailed energy accounting for wireless sensor nodes," April 2008, pp. 383–394.

[15] P. Levis, D. Gay, V. Handziski, J.-H. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, J. Polastre, P. Buonadonna, L. Nachman, G. Tolle, D. Culler, and A. Wolisz, "T2: A second generation os for embedded sensor networks." Tech. Rep., 2005.

[16] "Tmote sky datasheet." [Online]. Available: http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf

[17] "BSN node specifications." [Online]. Available: http://bsn-web.org/index.php?article=926

[18] "Cc2420 2.4 ghz ieee 802.15.4/zigbee-ready rf transceiver." [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc2420.pdf

[19] M. T. Hansen, "Asynchronous group key distribution on top of the cc2420 security mechanisms for sensor networks," in *WiSec '09: Proceedings of the second ACM conference on Wireless network security*. New York, NY, USA: ACM, 2009, pp. 13–20.

[20] "The sensobyg project." [Online]. Available: http://www.sensobyg.dk/english

[21] "Msp430x15x, msp430x16x, msp430x161x mixed signal microcontroller (slas368e)." [Online]. Available: http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf

[22] N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: case study of arm7tdmi," in *ISLPED '00: Proceedings of the 2000 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2000, pp. 185–190.

Secure Group Formation Protocol for a Medical Sensor Network Prototype

Jacob Andersen
Department of Computer Science, Aarhus University
Aabogade 34, 8200 Aarhus N, Denmark
jacand@cs.au.dk

***Abstract*—Designing security mechanisms such as privacy and access control for medical sensor networks is a challenging task; as such systems may be operated very frequently, at a quick pace, and at times in emergency situations. Understandably, clinicians hold extra unproductive tasks in low regard, and experience from user workshops and observations of clinicians at work on a hospital ward show that if the security mechanisms are not well designed, the technology is either rejected altogether, or they are circumvented leaving the system wide open to attacks [1].**

Our work targets the problem of designing wireless sensors to be both secure and usable by exploring different solutions on a fully functional prototype platform. In this paper, we present an Elliptic Curve Cryptography (ECC) based protocol, which offers fully secure sensor set-up in a few seconds on standard (Telos) hardware. We evaluate this protocol's time and energy consumption, and its impact on the battery lifetime.

I. Introduction

Several experimental sensor network platforms have emerged in recent years targeted at clinical use, but only few consider the importance of security issues such as privacy and access control, which can have a huge impact on usability. Others develop secure protocols that are not useful on sensor platforms. A calculation which could be performed in fractions of a second on a PC spends several seconds or minutes on a sensor. Major problems include user interface latency due to lengthy calculations, and a demand for reliable authentication even when network infrastructure or servers are unavailable.

Setting up sensors must be a quick and easy task, and at the same time, only authorised personnel should be allowed to do this—and of course all wireless communication must use strong encryption. As sensor size and cost decrease over time, sensors which are big and expensive today will be disposable tomorrow (for instance in the form of sterile patches or pills), when the cost of cleaning and re-use exceeds the cost of the sensor itself. For this reason we choose to reject any solution, which would require that the hospital's staff (IT department) must preprogram local keys into all sensors prior to their use on the network, as such solutions would not scale—disposable sensors must be ready to use straight from the factory.

This paper presents a protocol for validating user credentials and building the necessary keys, as a clinician sets up a sensor on a patient. The work presented here is part of a larger project, aiming to construct a complete prototype sensor network infrastructure, designed primarily for use in hospitals and at emergencies, with the requirements given above (among others). A description of our prototype platform and overall goals can be found in [2], and the sensor user interface, BLIG (Blinking LEDs Indicated Grouping), was presented in [3].

In addition to a presentation of the protocol, the contributions of this paper include the results of an evaluation of the time and energy requirements of the protocol on the current sensor hardware, and also some considerations regarding its impact on battery lifetime of sensors with very small batteries.

After a look at some relevant related work, we examine the requirements and assumptions we can make about the protocol's environment. The protocol is presented and discussed, along with the evaluation, and we conclude with some steps, we are planning to take to improve performance.

II. Related Work

The CodeBlue project [4] explores the use of sensor networks at emergencies and in hospitals. A service discovery based approach is taken, so that data from new sensors just pop up on nearby monitors. However, there are no restrictions on, who will get access to the data, and also no encryption is used on the network. This solution can be acceptable at an emergency (which also seems to be their primary objective), but it will not be acceptable in a hospital.

In [5] each clinician has a personal pen, which is used to set up sensors. Although the authors did not include any security or authorisation mechanism in this proposal, they discuss how this could easily be added on top, and their proposal becomes more appealing than CodeBlue, as it identifies (and potentially approves) the clinician operating on the sensors.

[6] explores security issues in medical sensor networks, including the often overlooked issue of adversarial behaviour among authorised users (clinicians). Another great analysis of medical sensor network security problems was done by [7].

Several public key cryptography (PKC) implementations for sensor networks have appeared in recent years. Elliptic curve cryptography (ECC) currently seems the most promising method. One of the most optimised implementation for TinyOS is TinyECC [8], which can perform the basic ECC point multiplication in less than 2 seconds on the MSP430x1 micro-controller (MCU) family found on the Tmote Sky sensor nodes. NanoECC [9] manages to do the same calculation in less than 1 second on the same platform, clock frequency and current draw, but with different curve parameters and what appears to be a more efficient implementation.

III. Fundamental Concepts

Sensor nodes are organised in groups around patients in a 1-to-1 relationship. In order to establish the various (symmetric) keys necessary for wireless communication in the medical sensor network, a few PKC based *configuration change* protocols are used whenever a (clinical) user configures the network:

- Add a new sensor to a patient.
- Remove a sensor from a patient.
- Change patient property (e.g. move to another hospital).

This paper only looks at the "add" protocols, as this is the most complicated situation, and the other protocols are similar. We need two distinct protocols for this task: adding the first sensor node to a patient, *bootstrapping*, which involves the formation of a new group; and adding a sensor to an existing group (patient carrying at least one sensor), *association*.

The players in the protocols are as follows:

N: The new sensor being attached to the patient.

S: Sensor already attached to a patient (only association).

A: The *authorisation-node*—a device (name tag, PDA) identifying the clinician performing the configuration.

A-nodes can be more powerful and expensive than sensors. They are programmed with the keys of the current "security domain" (e.g. a hospital) and require login. Tampering would be detected by an honest user, and they may be tamper-proof.

As mentioned earlier, sensors cannot be preprogrammed with any keys. Therefore, a new sensor node, N, will learn who to trust by the first node it communicates with after it is turned on. Inspired by [10], we call this process *imprinting*.

The user interaction required to add a sensor to a patient is given by BLIG [3] and shall not be repeated here. The bootstrapping and association protocols are started after a clinician brings the new sensor, N, close to her A node and possibly (for association) a sensor, S, attached to the patient. This proximity enables a one-packet interchange on a *short-range communication channel*. For the sake of the discussion in this paper, we assume that this channel implements *data-origin authenticity* [11], which basically means that an out-of-range adversary may be able to eavesdrop on the communication, but cannot intercept or inject information. Any data received on this channel originated from the node, that the user intended.

A. Adversarial Behaviour and Countermeasures

The short-range communication channel gives rise to a distinction between *internal* and *external* adversaries.

External: Someone who has no physical access to the patient, whose network he is attempting to access (out of range for short-range communication).

Internal: This is a person who has direct access to the patient (perhaps even the patient himself).

While security against external threats in a network is well-studied with a number of well-tested solutions, threats posed by people with access to the patient may be much more complex. However, the goal here is not "absolute security" whatever that means, but rather security comparable to existing technology. For instance, the cable from a sensor to a display provides full security against external adversaries but only limited security against tampering (internal adversaries). Even today an honest clinician is assumed to discover such attacks.

An external adversary may try to capture a node in order to acquire access to some patient, but he cannot capture nodes already used on a patient, as this would require physical access to that patient. A factory-fresh sensor is of no use to the adversary. Should he successfully tamper with it, an honest clinician is assumed to discover it. Also, our adversary cannot use an authorisation-node for much, as he still needs physical access to the patient to perform a configuration change. If the patient is leaving the ward to stay in public areas, the sensor network may be locked, so that the "external" adversary cannot connect even if he gets physical access to the patient.

Internal adversaries include clinicians or hospital employees who may have accepted bribe from external parties or try to cover up malpractice. With access to the short-range channel they may launch man-in-the-middle attacks or tamper with nodes, but it is fair to assume, that they would not use their own credentials if it could be traced back to them. Patients and visitors may also exhibit adversarial behaviour (e.g. a drug addict trying to manipulate a medicine pump).

Security against the internal threats today are most often handled by pragmatic approaches: a drug addict may be monitored more than other patients; if a celebrity or a statesman is admitted to a hospital ward, security guards may be posted to keep journalists or other unwanted guests away.

The primary line-of-defence against all internal attacks are cryptographically signed logs. The configuration change protocol must ensure, that a correct log entry is recorded in each uncorrupted node. If at least one protocol participant is uncorrupted, a correct log entry will exist, and since correct log entries will have valid signatures, forging a log entry without valid credentials is not possible. If all sensors and authorisation-nodes connect to a hospital server, log activity may be monitored, generating reports about irregularities.

A secondary line-of-defence is the buzzer, which is an important part of any authorisation-node user-interface as explained in [3]. The buzzer allows audible feedback to the user from the authorisation-node to avoid requiring visual attention (the user never has to look at this node). For the sake of this discussion, we assume, that an adversary cannot capture an authorisation-node when its owner is logged in—as these nodes are quite powerful, they may be able to detect separation from the owner, or perhaps just log out automatically when idle for a few seconds (the concrete solution would probably be a pragmatic trade-off between usability and concrete threats). As the adversary cannot steal an authorisation-node, he may (literally) try to go behind the owner's back in order to steal her identity for a configuration change. But since all such actions are accompanied by a buzzer sound, it will be hard to avoid being noticed. Furthermore, the authorisation-node may be set up to require pressing a button, scanning a fingerprint or some similar action making it even more difficult.

Finally, *if* an adversary manages to get access to the sensor network on a patient (by uncovering the `group-key`), he would only get access to that particular patient, as the sensor groups are completely isolated from each other. Even with this access, he would still not be capable of performing any configuration change, as this requires a valid log message.

IV. Protocol description

For the protocol, we use elliptic curve domain parameters over $\mathbb{F}_p$ (p, a, b, G, n, h) given by secp160r1 [12]. We also use a cryptographic hash function, $\mathbb{H}$, described later.

To construct signatures and proofs-of-knowledge, we use the Schnorr algorithm [13] adapted for elliptic curves, instead of the Elliptic Curve Digital Signature Algorithm (ECDSA). The Schnorr algorithm is faster than the ECDSA, as it uses fewer large integer modular multiplications and no divisions. The Schnorr signature verification can in fact be performed in the Jacobian projective coordinate space, which implies that no divisions are necessary, while the ECDSA verification involves at least two large integer modular divisions (one for translating Jacobian coordinates to regular affine coordinates, and one explicitly required by the algorithm).

Before the protocols can begin, all nodes precompute a few values, in order to save execution time during the interactive part of the protocol. Disposable sensors may do this during a factory self-test (to avoid spending battery energy), and rechargeable sensors may do this while charging.

All nodes on a patient share a public `group-id` and a private `group-key` used for internal group communication, such as property change dissemination. This key is created by the first member of the group in the bootstrap protocol described below, and is never changed. Of course this implies, that there are no forward secrecy for this internal group communication; however, data exchanged internally between nodes constitutes the group history, and will be summarised for new nodes anyway, so this should not be a problem.

The nodes on a patient also share a list, `ca-pk`, of public keys of certificate authorities, which is kept synchronised among all group members. The elements on this list serve a twofold purpose. First, they are the public keys of authorities, who may issue certificates to authorisation-nodes (only A-nodes who can present a certificate signed by a member of this list will be allowed access to make configuration changes). Second, they are the public keys of servers, that are allowed to access sensor data from any sensor on this patient. Different elements of this list can represent different teams of clinicians, an entire ward, or perhaps even a hospital. Another scenario would be the emergency, where the paramedic A would put the ambulance service's key on the `ca-pk`, but could add the key of the trauma centre to this list as well. Adding and removing entries from the `ca-pk` list is also possible later (using the patient property change protocol).

A. Common notation

We do not distinguish between a node's public key and its ID (network address). In particular, node N has the ID `n-id`, which is the point `n-id` = `n-sk` $\otimes G$ on the elliptic curve where `n-sk` is a secret integer known only by N.

In the following A, N and S denotes honest players, while A', N' and S' denotes arbitrary players, who may be dishonest (controlled by an attacking adversary). Due to our data-origin-authenticity assumption about the short-range link, the initial exchange of IDs, and the Diffie-Hellman symmetric key generation, an adversary must know the private key or choose the ID of each node, he is impersonating, to succeed.

B. The bootstrap protocol

The object of this protocol, shown in figure 1, is to create a new group when the first sensor, N, is attached to a patient and becomes the first member of this patient's new group.

As N (and thus the group) comes to life, it is imprinted by A with a list, `ca-pk[]`, of IDs (public keys) of trusted entities. This list must contain at least one element—an authority who issued a certificate for A (there may be more than one to choose from). By choosing the elements of this list, A decides who will (initially) have access to this sensor group (patient).

In this protocol, A generates the log message "New group `group-id` created by `n-id` and imprinted with `ca-pk[]`—signed `a-id`". A chooses `ca-pk[]`, but has to check the remaining information before publishing the message, in particular *A must check that `group-id` is new, and that the corresponding `group-key` is known by N'*. In the protocol N' proves the knowledge of `group-key`$-r$ before r is revealed by A. If `group-id` was *not* new, N' could collude with a member of this group (who knows `group-key`) and calculate r, contradicting the ECC discrete log assumption. On the other hand, after N' has proved knowledge of `group-key`$-r$, r is revealed to N', so N' must be able to calculate `group-key`. This argument proves the entire log message claim.

N must also check that `group-id` is new and that the `group-key` = `group-id` $\otimes G$ (Φ_1 is chosen randomly and the latter is checked explicitly). Furthermore, N must check that A' correctly signed the message and presents a valid certificate signed by `ca-pk[0]`. N *cannot* check the "validity" of `ca-pk[]`, but this is precisely the point of the imprinting, and it is the feature, which ensures that anyone can get sensor access in an emergency: a display device must implement a dummy "self-authorised" A (more details in [2]), which may be used when "proper" credentials are unavailable.

A corrupt A' must not be capable of extracting knowledge from N other than what is necessary for the log message; but by the zero-knowledge property of the Schnorr algorithm in the random oracle model, the information going into the log message are the only data transmitted to A'. On the other hand, a corrupt N' should only be able to extract public data and a correct log message from A, but apart from r and R (garbage for the honest A), nothing else is transmitted.

C. The association protocol

The protocol shown in figure 2 is used when a sensor is added to an already existing group. In this case, A generates the log message: "Node `n-id` added to group `group-id` by node `s-id`—signed `a-id`". Prior to publishing this log message, A must check that `s-id` is indeed a member of

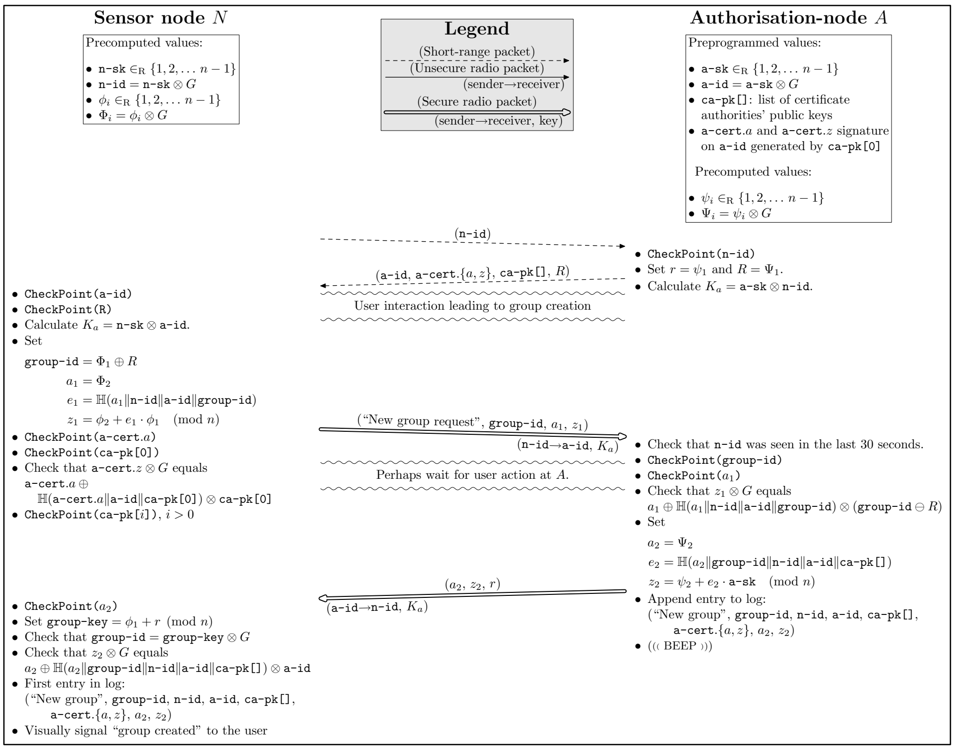

Sensor node N
Precomputed values:
• n-sk $\in_R \{1,2,\dots n-1\}$
• n-id = n-sk $\otimes G$
• $\phi_i \in_R \{1,2,\dots n-1\}$
• $\Phi_i = \phi_i \otimes G$
Legend
(Short-range packet)
(Unsecure radio packet)
(sender→receiver)
(Secure radio packet)
(sender→receiver, key)
Authorisation-node A
Preprogrammed values:
• a-sk $\in_R \{1,2,\dots n-1\}$
• a-id = a-sk $\otimes G$
• ca-pk[]: list of certificate authorities' public keys
• a-cert.a and a-cert.z signature on a-id generated by ca-pk[0]
Precomputed values:
• $\psi_i \in_R \{1,2,\dots n-1\}$
• $\Psi_i = \psi_i \otimes G$
(n-id)
• CheckPoint(n-id)
• Set $r = \psi_1$ and $R = \Psi_1$.
• Calculate K_a = a-sk $\otimes$ n-id.
(a-id, a-cert.$\{a,z\}$, ca-pk[], R)
User interaction leading to group creation
• CheckPoint(a-id)
• CheckPoint(R)
• Calculate K_a = n-sk $\otimes$ a-id.
• Set
group-id = $\Phi_1 \oplus R$
$a_1 = \Phi_2$
$e_1 = \mathbb{H}(a_1\|$n-id$\|$a-id$\|$group-id$)$
$z_1 = \phi_2 + e_1 \cdot \phi_1 \pmod n$
• CheckPoint(a-cert.a)
• CheckPoint(ca-pk[0])
• Check that a-cert.$z \otimes G$ equals a-cert.$a \oplus$ $\mathbb{H}($a-cert.$a\|$a-id$\|$ca-pk[0]$) \otimes$ ca-pk[0]
• CheckPoint(ca-pk[i]), $i > 0$
("New group request", group-id, a_1, z_1)
(n-id→a-id, K_a)
Perhaps wait for user action at A.
• Check that n-id was seen in the last 30 seconds.
• CheckPoint(group-id)
• CheckPoint(a_1)
• Check that $z_1 \otimes G$ equals $a_1 \oplus \mathbb{H}(a_1\|$n-id$\|$a-id$\|$group-id$) \otimes ($group-id $\ominus R)$
• Set
$a_2 = \Psi_2$
$e_2 = \mathbb{H}(a_2\|$group-id$\|$n-id$\|$a-id$\|$ca-pk[]$)$
$z_2 = \psi_2 + e_2 \cdot$ a-sk $\pmod n$
(a_2, z_2, r)
(a-id→n-id, K_a)
• Append entry to log: ("New group", group-id, n-id, a-id, ca-pk[], a-cert.$\{a,z\}$, a_2, z_2)
• (((BEEP)))
• CheckPoint(a_2)
• Set group-key = $\phi_1 + r \pmod n$
• Check that group-id = group-key $\otimes G$
• Check that $z_2 \otimes G$ equals $a_2 \oplus \mathbb{H}(a_2\|$group-id$\|$n-id$\|$a-id$\|$ca-pk[]$) \otimes$ a-id
• First entry in log: ("New group", group-id, n-id, a-id, ca-pk[], a-cert.$\{a,z\}$, a_2, z_2)
• Visually signal "group created" to the user

Figure 1. Bootstrapping protocol

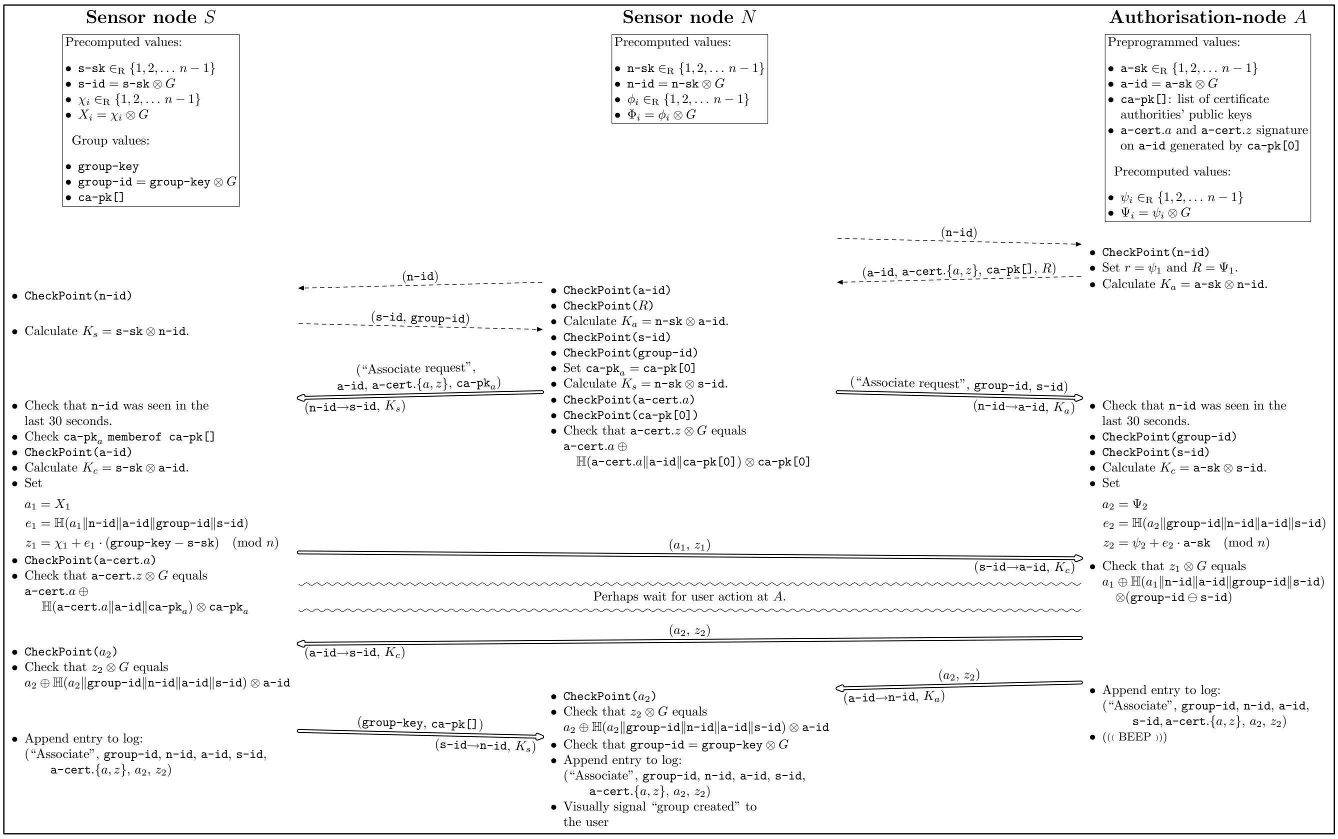

Sensor node S
Precomputed values:
• s-sk $\in_R \{1,2,\dots n-1\}$
• s-id = s-sk $\otimes G$
• $\chi_i \in_R \{1,2,\dots n-1\}$
• $X_i = \chi_i \otimes G$
Group values:
• group-key
• group-id = group-key $\otimes G$
• ca-pk[]
Sensor node N
Precomputed values:
• n-sk $\in_R \{1,2,\dots n-1\}$
• n-id = n-sk $\otimes G$
• $\phi_i \in_R \{1,2,\dots n-1\}$
• $\Phi_i = \phi_i \otimes G$
Authorisation-node A
Preprogrammed values:
• a-sk $\in_R \{1,2,\dots n-1\}$
• a-id = a-sk $\otimes G$
• ca-pk[]: list of certificate authorities' public keys
• a-cert.a and a-cert.z signature on a-id generated by ca-pk[0]
Precomputed values:
• $\psi_i \in_R \{1,2,\dots n-1\}$
• $\Psi_i = \psi_i \otimes G$
(n-id)
• CheckPoint(n-id)
• Set $r = \psi_1$ and $R = \Psi_1$.
• Calculate K_a = a-sk $\otimes$ n-id.
(a-id, a-cert.$\{a,z\}$, ca-pk[], R)
(n-id)
• CheckPoint(n-id)
• Calculate K_s = s-sk $\otimes$ n-id.
(s-id, group-id)
• CheckPoint(a-id)
• CheckPoint(R)
• Calculate K_a = n-sk $\otimes$ a-id.
• CheckPoint(s-id)
• CheckPoint(group-id)
• Set ca-pk$_a$ = ca-pk[0]
• Calculate K_s = n-sk $\otimes$ s-id.
• CheckPoint(a-cert.a)
• CheckPoint(ca-pk[0])
• Check that a-cert.$z \otimes G$ equals a-cert.$a \oplus$ $\mathbb{H}($a-cert.$a\|$a-id$\|$ca-pk[0]$) \otimes$ ca-pk[0]
("Associate request", a-id, a-cert.$\{a,z\}$, ca-pk$_a$)
(n-id→s-id, K_s)
("Associate request", group-id, s-id)
(n-id→a-id, K_a)
• Check that n-id was seen in the last 30 seconds.
• Check ca-pk$_a$ memberof ca-pk[]
• CheckPoint(a-id)
• Calculate K_c = s-sk $\otimes$ a-id.
• Set
$a_1 = X_1$
$e_1 = \mathbb{H}(a_1\|$n-id$\|$a-id$\|$group-id$\|$s-id$)$
$z_1 = \chi_1 + e_1 \cdot ($group-key $-$ s-sk$) \pmod n$
• CheckPoint(a-cert.a)
• Check that a-cert.$z \otimes G$ equals a-cert.$a \oplus$ $\mathbb{H}($a-cert.$a\|$a-id$\|$ca-pk$_a) \otimes$ ca-pk$_a$
• Check that n-id was seen in the last 30 seconds.
• CheckPoint(group-id)
• CheckPoint(s-id)
• Calculate K_c = a-sk $\otimes$ s-id.
• Set
$a_2 = \Psi_2$
$e_2 = \mathbb{H}(a_2\|$group-id$\|$n-id$\|$a-id$\|$s-id$)$
$z_2 = \psi_2 + e_2 \cdot$ a-sk $\pmod n$
(a_1, z_1)
(s-id→a-id, K_c)
• Check that $z_1 \otimes G$ equals $a_1 \oplus \mathbb{H}(a_1\|$n-id$\|$a-id$\|$group-id$\|$s-id$)$ $\otimes($group-id $\ominus$ s-id$)$
Perhaps wait for user action at A.
(a_2, z_2)
(a-id→s-id, K_c)
• CheckPoint(a_2)
• Check that $z_2 \otimes G$ equals $a_2 \oplus \mathbb{H}(a_2\|$group-id$\|$n-id$\|$a-id$\|$s-id$) \otimes$ a-id
(a_2, z_2)
(a-id→n-id, K_a)
• Append entry to log: ("Associate", group-id, n-id, a-id, s-id, a-cert.$\{a,z\}$, a_2, z_2)
• (((BEEP)))
(group-key, ca-pk[])
(s-id→n-id, K_s)
• Append entry to log: ("Associate", group-id, n-id, a-id, s-id, a-cert.$\{a,z\}$, a_2, z_2)
• CheckPoint(a_2)
• Check that $z_2 \otimes G$ equals $a_2 \oplus \mathbb{H}(a_2\|$group-id$\|$n-id$\|$a-id$\|$s-id$) \otimes$ a-id
• Check that group-id = group-key $\otimes G$
• Append entry to log: ("Associate", group-id, n-id, a-id, s-id, a-cert.$\{a,z\}$, a_2, z_2)
• Visually signal "group created" to the user

Figure 2. Association protocol

group `group-id` and willing to accept `n-id` as a new member. To confirm this, S' sends a proof to A that it knows both the `group-key` and `s-sk` and that it will accept `n-id`. If the `group-id` is a genuine group (i.e. sensors on a patient) which has not been compromised, an adversary cannot forge this proof. On the other hand, if the adversary is indeed controlling S', he could make S' claim to belong to another (compromised) `group-id` different from the $\overline{\texttt{group-id}}$ the clinician expected. This tampering should be easily detectable by an honest user (clinician) by noticing that the BLIG [3] blinking pattern of sensor N (in `group-id`) is different from the other (un-compromised) sensors in $\overline{\texttt{group-id}}$.

N must check the sanity of the log message received from A' and the `group-key` received from S'.

S acts as a sentry, guarding the patient's network against intruders, and must make sure that N' and A' will not be able to extract any information except what is used in the log message. Only if A' has proper credentials and delivers a correctly formed log message will N' be accepted into the group. S will then imprint N' with all relevant information.

D. Implementation notes

Our ECC implementation is shaped after the TinyECC [8] code and uses its large integer computation module (`NNM.nc`) unchanged. However, as communication and user interaction is needed in parallel with ECC calculations, we re-implemented the point calculations: an ECC calculation will be handed over to a task, performing one computation-step and re-posting itself until the computation is done. The overhead introduced is small, as the measurements found in [8] are similar to the measurements presented in the following section.

Our primary hardware platform is currently the BSN mote, which is a Telos rev. A design with an MSP430 MCU and just 2 kB RAM [14]. With this limited memory, sliding window multiplications and Shamir's trick (c.f. [8]) cannot be used.

We implemented a secure hash function $\mathbb{H}$ using the Merkle-Damgård construction on a Matyas-Meyer-Oseas (MMO) one-way compression function built using the AES block cipher, and used the AES hardware support on the CC2420 radio to perform the calculations, as earlier experiments showed this solution to be far more energy and time efficient than calculating AES on the MCU [15]. The CC2420 can receive the next clear-text buffer from the MCU and return the current cipher-text in a single duplex SPI bus transfer, saving almost half the communication time; so we chose MMO, as its clear-text input to the block cipher is independent of the output from the previous block cipher invocation.

As the MSP430F149 MCU does not have a DMA controller, the MCU will spend a significant amount of time moving bytes back and forth to the CC2420 radio, whereas with a DMA controller, the $\mathbb{H}$-computation can be performed truly in parallel with other tasks. For the Schnorr verification, this is particularly interesting, as $\mathbb{H}$ may be calculated in parallel with the $z \otimes G$ calculation (or the window precomputations when using Shamir's trick). A Schnorr verification from the protocol was measured on a Tmote Sky (with the MSP430F1611 MCU) *with* DMA to be 19.3 ms faster than on the BSN mote. The time spent computing $\mathbb{H}$ on the BSN was 19.5 ms, so on the Tmote only 0.2 ms *processor time* was spent. If the radio is on anyway, calculating $\mathbb{H}$ is almost free. If the radio is not on, a bit of extra power will be required, though. The CC2420 draws less than $\frac{1}{2}$ mA in the lowest power state necessary.

Table I
TIME MEASUREMENTS OF DIFFERENT CALCULATION "PRIMITIVES" AVERAGED OVER 260 EXECUTIONS ON RANDOM INPUTS.

	Mean time [ms]	Std.dev [ms]
Schnorr sign	21.3	0.36
Schnorr verify	2799.8	24.0
$i \otimes G$	1411	24
$i \otimes P$ (any point)	1657	31
`CheckPoint(`P`)`	2.85	0.27
$+ \pmod{n}$	0.067	18e-6
$\cdot \pmod{p}$	0.678	0.0004
$\cdot \pmod{n}$	1.561	0.0016
$/ \pmod{n}$	84.9	2.34

Note also, that the $\mathbb{H}$ function described here only produces 128 bit hashes. This was done to simplify the code, but of course it will not offer the same level of security as full 160 bit hashes. This also had a side-effect, however, as the Schnorr signature verifications will be 300 ms faster than expected. This is caused by the 32 0-bits appearing as most significant bits in the point multiplication, causing the multiplication algorithm to use fewer point doublings, thus performing faster.

On the other hand, we are actually performing several divisions that could easily be removed by simple reduction. Furthermore, we are not using Shamir's trick. According to the measurements done by [8], we should be able to save almost a second per signature verification by introducing this optimisation. Finally, the NanoECC [9] implementation appears to be twice as efficient as TinyECC, so investigating the reason for this speedup may also improve our numbers.

We are planning on fixing all these things along with a hardware upgrade in the future.

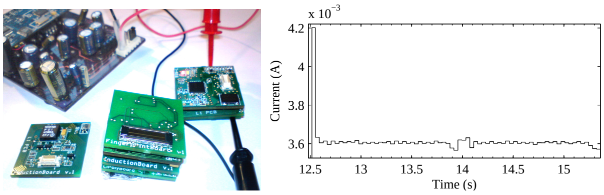
V. Evaluation

In order to get a full understanding of the time distribution among the different calculations used in the protocol, we measured the durations of the different primitives. The results are tabulated in table I. These measurements were done on the Tmote Sky platform using DMA and the sliding window optimisation (but *not* Shamir's trick) for point multiplication.

We evaluated the time and energy consumption on the protocol as well as the different building blocks of the protocol using the "Energy Bucket" [15]. Figure 3 shows the setup, and a graph of the current during one of the Schnorr verifications.

The energy consumption for the bootstrapping and association protocols is currently 0.1 J on the sensor nodes. This value includes energy spent on calculation and communication, but not energy spent on precomputations and the user interface; the latter may be several times more, depending on the user interface configuration (which is outside the scope of this paper). We do not worry about authorisation-node consumption.

For comparison, a Panasonic CR1025 coin cell battery (1 cm in diameter) contains 324 J, while a tiny LR621 watch battery contains only 81 J, so 0.1 J can be quite a lot, if the procedure



(a) Measurement setup

(b) Result plot excerpt

Figure 3. Energy consumption of the protocol running on the sensor node to the right is measured using the "Energy Bucket" [15] device seen in the back. The sensor node communicates with the authorisation-node seen at the front center. This particular authorisation-node is equipped with a fingerprint reader for easy and secure clinician login. At the bottom left, the "Induction Board" is shown, which offers short-range communication, one push-button at the upper-right corner, and a buzzer (the black square) used by authorisation-nodes to produce sounds. An excerpt of the resulting current/time plot showing the typical current draw during a Schnorr verification is shown on the right.

must be repeated often during the sensor's lifetime. However, it would be fair to assume, that patients at hospitals—and at emergencies as well—would carry at least one big sensor or some kind of patient ID wristband with a larger battery. This "patient ID node" would be used as S in most association protocols (as well as the other configuration change protocols mentioned earlier), and it would be fair to assume, that most tiny sensors would only have to execute these protocols 2–5 times during the entire battery lifetime. Under this assumption, the protocol requires less than 1% of the energy of a watch battery, which should be an acceptable price. [16] reports results of a simulated ECC hardware implementation: a digital signature verification can be done with 324 μJ. Translated to the protocols presented here, about 1 mJ would be required to do the protocol calculations on the sensor. Note that this does not include energy spent on communication, so this number cannot be compared directly to our measurements.

VI. Conclusion and Future Work

This paper has demonstrated, that medical sensors using ECC and imprinting along with short-range communication to establish secure communication, are indeed feasible. Our proposal protects the network (and the log in particular) against external and internal adversaries, with no sensor node key preprogramming of any kind. We also show that this proposal is viable on tiny and extremely low-energy devices.

On the current platform (MSP430x1 running at 8.4 MHz), the protocol uses slightly less than 10 seconds for bootstrapping and about 13 seconds for association, which is not acceptable for real applications. However, we are planning to upgrade to an MSP430x5 MCU, capable of running up to 25 MHz and equipped with a 32×32 bit hardware multiplier. Code analysis indicates, that about 85% of the ECC point multiplication execution time is spent in the multiplication of large integers subroutine; so upgrading from a 16×16 to a 32×32 bit hardware multiplier should improve the execution time significantly. Also a larger RAM would allow us to use Shamir's trick, and along with increasing the clock frequency by a factor 3, we should be able to get below 1 second. Furthermore, according to the specifications, the MSP430x5 series consumes less energy per instruction, so the energy efficiency should even be improved as well by this upgrade.

Acknowledgement

The research presented here was supported by the Activity-Based Computing and the PalCom projects at Aarhus University; and the SAPHE project at the Imperial College London.

The Activity-Based Computing project is funded by the Danish Research Council under the NABIIT programme (http://activity-based-computing.org/); the PalCom project (http://www.ist-palcom.org) is funded by the EU's 6th Framework Programme; and the SAPHE project is funded by the UK Technology Strategy Board (http://vip.doc.ic.ac.uk/saphe).

References

[1] J. E. Bardram, "The trouble with login - on usability and computer security in ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 9, no. 6, pp. 357–367, Dec 2005.

[2] J. Andersen, B. Lo, and G.-Z. Yang, "Experimental platform for usability testing of secure medical sensor network protocols," in *BSN'08: 5th Intl. Workshop on Wearable and Implantable Body Sensor Networks*. IEEE, Jun 2008, pp. 179–182.

[3] J. Andersen and J. E. Bardram, "BLIG: A new approach for sensor identification, grouping, and authorisation in body sensor networks," in *BSN'07: 4th Intl. Workshop on Wearable and Implantable Body Sensor Networks*, ser. IFMBE Proceedings, S. Leonhardt, T. Falck, and P. Mähönen, Eds., vol. 13. Berlin: Springer, Mar 2007, pp. 223–228.

[4] K. L. D. J. Malan, T. R. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emegency response: Challenges and opportunities," *IEEE Pervasive Computing*, vol. 03, no. 4, pp. 16–23, Oct-Dec 2004.

[5] H. Baldus, K. Klabunde, and G. Müsch, "Reliable set-up of medical body-sensor networks," in *EWSN'04: 1st. European Workshop on Wireless Sensor Networks*, ser. LNCS, no. 2920. Berlin, Germany: Springer, 2004, pp. 353–363.

[6] A. Bhargava and M. Zoltowski, "Sensors and wireless communication for medical care," in *DEXA'03: 14th Intl. Workshop on Database and Expert Systems Applications*. IEEE, Sep 2003, pp. 956–960.

[7] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Pervasive Computing*, vol. 07, no. 1, pp. 30–39, Jan–Mar 2008.

[8] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *IPSN'08: Intl. Conference on Information Processing in Sensor Networks*, Apr 2008, pp. 245–256.

[9] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the limits of elliptic curve cryptography in sensor networks," in *Wireless Sensor Networks*, ser. LNCS, no. 4913. Springer, 2008, pp. 305–320.

[10] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, ser. LNCS, no. 1796. Springer, 2000, pp. 172–182.

[11] F. L. Wong and F. Stajano, "Multichannel security protocols," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, Oct 2007.

[12] Standards for Efficient Cryptography Group / Certicom Research, *SEC 2: Recommended Elliptic Curve Domain Parameters*, 1st ed. [Online]. Available: http://www.secg.org/download/aid-386/sec2_final.pdf

[13] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, Jan 1991.

[14] "BSN node specifications." [Online]. Available: http://bsn-web.org/index.php?article=926

[15] J. Andersen and M. T. Hansen, "Energy bucket: A tool for power profiling and debugging of sensor nodes," in *The 3rd Intl. Conference on Sensor Technologies and Applications (SENSORCOMM 2009)*. IEEE, Jun 2009, pp. 132–138.

[16] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Third IEEE Intl. Conference on Pervasive Computing and Communications Workshops*, Mar 2005, pp. 146–150.

Towards both Usable and Secure Protocols for Medical Sensor Networks

Jacob Andersen

***Abstract*—Designing security mechanisms such as e.g. access control for wireless medical sensors is a challenging task; as such systems may be operated very frequently, at a quick work pace, and at times in emergency situations. Understandably, clinicians hold extra unproductive tasks in low regard, and experience from user workshops and observations of clinicians at work on a hospital ward show that if the security mechanisms are not well designed, the technology is either rejected altogether, or they are circumvented leaving the system wide open to attacks [1].**

Our work targets the problem of designing wireless sensors to be both secure and usable by exploring different solutions on a fully functional prototype platform. This paper offers an overview over the security framework of this platform, describes and discusses some of the key features and interfaces used to build secure prototypes, and finally illustrates how we implemented a simple prototype with limited security and how future prototypes with different and better properties will be designed.

I. Introduction

Several experimental sensor network platforms have emerged in recent years targeted clinical use, but only few consider the importance of security issues such as privacy and access control, which can have a huge impact on the usability. Others, on the other hand, develop secure protocols that are not useful on sensor platforms. A calculation which could normally be performed in fractions of a second on the desktop now spends several seconds – or even minutes – on the sensor platform. One major problem is the latency introduced to the user interface by such lengthy calculations.

The main goal of this work is to deliver a notion of "security" comparable to the existing (cable-based) technologies. However, as it is very easy to develop security which looks nice on paper, but is too slow and energy hungry in practice, we have taken a slightly different approach:

By building a platform, which is designed to provide a framework for easy prototype development, we plan to develop not just a single secure protocol, but a number of protocols having different properties. This presentation will not go into details about the architecture of the platform itself, as this was described in [2]. Instead, the primary contribution of this paper is a presentation of the framework for development of the secure protocols.

II. Trusting Sensor Nodes

Before we use a data stream from a sensor, we need to be confident that the sensor is in fact on the patient, it claims to be on – and attached correctly. To do this, we want to know *who* installed the sensor, and decide whether that person can be trusted. To accomplish this, all users (clinicians) will be carrying an "authorisation-node" – typically in the form of an intelligent name tag.

Sensors are organised in groups around patients, and a sensor-node starting a new group will trust any authority that is trusted by the authorisation-node which initiated the operation. We call this "imprinting" [3]. Each sensor-node keeps a "genealogy list" as a special section of its log file, listing ancestors all the way back to the sensor-node that created the network. Details can be found in [4], [2]. Whenever a configuration change adds a new node to a patient group, this new node will get the genealogy list from its "parent" (one of the existing sensors). All elements on the list are signed, so for instance if A and B are sensor-nodes in the same group (i.e. on the same patient), and X and Y are both authorisation-nodes approved by the authority trusted by imprinting, the list kept by B could look like:

$$\begin{array}{l}[\ A \text{ is a new group, signed } X;\\ \quad B \text{ is in the same group as } A \text{ signed } Y\ ]\end{array}$$

Each list element states that the 2 or 3 nodes were present at the same place at the same time – signed some authorisation-node. This certificate is stored in the log of all the nodes involved (in the case of the second list item, both A, B and Y keeps a copy). This means that even a succesful alteration or deletion in the log of one of the nodes can be detected.

The logs have append-only semantics – however, due to memory limitations, most logs will probably be circular buffers. Most of the time, nodes will be able to deliver log entries to on-line medical journal servers. It is a fundamental assumption, that log records cannot be altered or deleted (except overwritten due to buffer wrap-around) without physically tampering with the node.

A. Hardware Integrity

Authorisation-nodes are trusted as they are prepared by the hospital's IT staff. These devices requires some kind of login, and we assume, that any tampering would be detected by an honest clinician prior to logging in.

Furthermore, as these devices may be more advanced – and expensive – than the ordinary sensors, they may be equipped with anti-tampering mechanisms that erases the memory, if someone attempts to open the device.

Sensor-nodes arriving straight from the factory are trusted as well, and we assume that tampering can be detected by an honest clinician as well. This is actually no different than the

J. Andersen is a PhD student at the Department of Computer Science, Aarhus University, Denmark `jacand@cs.au.dk`

situation today: if someone have installed a wiretap or bug in a sensor at a hospital, the only line of defence is the staff's ability to discover it; hence this assumption is fully in line with our main goal of matching the level of security found in current practice.

III. Adversarial Behaviour

Hostile attacks can be divided into the following two categories:

External adversary
: Someone who has no physical access to the patient, whose sensor reading he is attempting to access.

Internal adversary
: This is a person who has direct access to the patient (perhaps even the patient himself).

Internal adversaries include clinicians and hospital employees who may be trying to cover up malpractice or who may have accepted bribe from external adversaries. However, it is fair to assume, that clinicians would never use their own credentials to do such things if this implies that the acts can be traced back to them. Patients and family (visitors) may also exhibit adversarial behaviour, for instance a drug addict may try to increase the dose of morphine from an i.v. medicine pump.

This partitioning between external and internal adversaries is helpful: While security against external threats in a network is well-studied and has a number of well-tested solutions, the threats posed by people who have access to the patient, may be a lot more complex. However, the goal here is not "absolute security" whatever that means, rather security comparable to existing technology. The cable between a sensor and a display found in existing equipment provides complete security against external adversaries but only limited security against tampering (i.e. internal adversaries).

Security against the internal threats today are most often handled by pragmatic approaches. For instance, if a drug addict is admitted to a hospital ward, he may be monitored more than other patients, to ensure that he does not steal any medicine. Along the same lines, if a celebrity or a statesman is admitted to a hospital ward, security guards may be posted to keep nosy journalists or other unwanted guests away. Perhaps the national intelligence services will even perform a background check on the staff of the ward to minimise the threat of internal adversaries.

A. External Attacks

The adversary will be capable of attacking the wireless link with the usual means (eavesdropping, packet injection etc.), but due to the physical distance between the adversary and the patient, access to the short-range communication link is either impossible or limited. If it is possible to guarantee that both eavesdropping and packet injection on the short-range link will be impossible except when in close proximity to the patient, then keys could even be transmitted in clear-text on this link without compromising external security. If only one of the properties can be guaranteed (e.g. that packet injection is impossible) then it is still a trivial task to generate a key that an external adversary cannot learn [5], [6].

An external adversary may also try to capture a node in order to acquire access to some patient, but he cannot capture sensor-nodes already in use on the patient, as this would require physical access to the patient. A factory-fresh sensor-node is of no use for the adversary, except if he can modify it and afterwards trick a clinician to use this sensor on the particular patient. This scenario would conflict with our assumption about the honest clinician being able to discover any tampering attempt. Also, our adversary cannot use an authorisation-node for much, as he still needs physical access to the patient in order to perform a configuration change.

If the patient is well enough to stay in public areas (leaving the ward), the sensor network could be locked for further configuration changes, so that the "external" adversary cannot create a connection even if he gets physical access to the patient in the public domain.

B. Internal Attacks

The class of internal attacks originates from an adversary who has physical access to the patient. It may include clinicians, visitors and even the patient himself. The internal adversary has full access to the short-range channel and may use tampered devices as well as attempting to launch man-in-the-middle attacks.

A typical goal would be to use the identity of some unsuspecting nurse to perform a configuration change that she never intended. For instance, if the nurse were intending to attach a new sensor to patient X, a node hidden under X's bed would intercept the communication and forward it to our adversary located at patient Y, where he would try to use the nurse's identity to perform a configuration change on patient Y instead.

The internal adversary may also attempt to use a captured authorisation-node, however, if the owner of this node is still logged in, there is no way to distinguish this scenario from authorised use. Therefore, we have to assume, that the authorisation-node will log out by itself – either after a fixed amount of time or by somehow detecting that it is removed from its owner. A clinician may succesfully tamper with her own authorisation-node and then log in and use it, but it is not possible to use an authorisation-node belonging to someone else.

The primary line-of-defence against all such attacks is the logs. The protocol used for configuration change must make sure, that a correct log entry is performed in all uncorrupted nodes. This means that if at least one participant in the protocol is uncorrupted, a correct log entry will exist. Furthermore, only correct log entries can have a valid signature, so forging log entries is not possible. Therefore it is unlikely that a clinician will use her own authorisation-node for fraudulent acts.

A secondary line-of-defence is the buzzer, which is a part of any authorisation-node as explained in [4]. The buzzer is an important part of the user interface, as it allows feedback to the user from the authorisation-node without requiring the user's

TABLE I
DATA KEPT IN EACH INDIVIDUAL SENSOR NODE, DATA SHARED AMONG SENSOR NODES IN A GROUP, AND DATA KEPT IN EACH AUTHORISATION-NODE.

node-id	Unique ID of the node (factory pre-programmed).
node-nonce	Node nonce generator.
node-genealogy	Ancestors and their certificates.
node-children	Children and their certificates.
group-id	Public ID of the group.
group-key	Secret key used for encryption and authentication.
group-time	Common clock.
group-CA-list	List of CAs trusted by this group.
group-meta	Information about the patient.
auth-id	Public id of the node.
auth-cert	A signature of a CA on auth-pk.
auth-log	A list of all successful groupings.
auth-CA	Public key of the CA.

attention – the user never has to look at the authorisation-node. However, the buzzer also serves another purpose: as it will be virtually impossible to capture an authorisation-node, our adversary may try (litterally) to go behind the nurse's back in order to steal her identity for a configuration change. But since all configuration changes will be accompanied by a sound from the buzzer, this will be hard to do without being noticed.

The problem of corrupted nodes can be summarised this way. Let A be an authorisation-node, P be a sensor node already attached to a patient, and N be a new sensor node. Suppose the configuration change of adding N to the network of P is supposed to take place, but some or all of the nodes are corrupted:

A and P or all three nodes are corrupt
: The notion of security makes no sense.

P corrupt $\vee$ N corrupt
: P and/or N must not be allowed to steal the identity of A or trick A to do anything except sign a log entry stating that N was added to P's group. If this log entry is ever produced, it will always be stored by A.

A corrupt $\vee$ N corrupt
: N must never be allowed access to P unless A delivers a signed and valid log entry stating that N was added to P, signed A. As the owner of the A node is the only one who can use this authorisation-node, she should notice if N or A is corrupt – and it is unlikely that a culprit would want to leave her own signature.

Notice that in a man-in-the-middle attack (like the example used earlier) the "node in the middle" would have to act like either A, N, P or any pair, in order to trick the un-corrupted node, so this class of attacks is covered as well.

IV. NODE CONFIGURATION DATA AND LOGS

Table I lists the data which is stored in each sensor-node and authorisation-node. Each sensor-node stores the information shown in the first two sections, while an authorisation-node keeps the information listed in the third section.

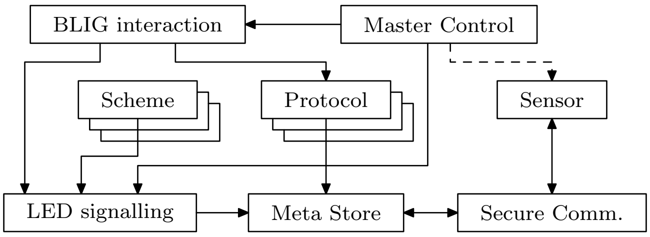


Fig. 1. Sensor-node software architecture

Each node has a *node-id*, which is a network address, and a persistent *node-nonce* store, which survives when the node is turned off. We assume that any pair of two uncorrupted nodes will have different *node-id*s and that a nonce generated by the *node-nonce* of an uncorrupted node will be unique.

node-genealogy, which was mentioned earlier, is the ancestor list, while *node-children* is the section of the sensor-node's log which keeps all the log messages of added sensor-node "children". The authorisation-node also keeps a log *auth-log* of configuration changes.

The *group-id* and *group-key* are generated by the sensor-node that created the group. The key is used *only* for intra-group communication, which is limited to keeping the *group-time*, the *group-CA-list* and the *group-meta* synchronised by dissemination. The *group-id*, *group-key* and *group-time* are used by the BLIG user interface [4] to generate a group identity blinking pattern.

The *group-CA-list* keeps all the public keys of trusted Certificate Authorities (i.e. hospital servers and similar). Certificates issued by any of these servers (such as all the authorisation-node certificates) may access the network and make configuration changes. Normally, the first element of this list will be provided by the authorisation-node during imprinting when a group is created. Later on in the lifetime of a group, more CAs may be added, e.g. if the patient is moved from one hospital to another.

The *group-meta* is a general (key,value) storage area, where a small amount of highly relevant data about the patient may be stored, such as Civil Registration Number, Name, Birthdate, Blood type etc.

Authorisation-nodes have a network address *auth-id*. In addition to this, the nodes keep a certificate on the validity of *auth-id* issued by a certificate authority, who has the public key *auth-CA*.

V. PLATFORM ARCHITECTURE

The purpose of the platform is to test different protocols for configuration changes, and the platform was designed in a way to support exactly that. More detailed information about the platform architecture can be found in [2]; here we will focus on the protocol for key generation.

At the centre of figure 1, the `Protocol` is located, which handles the communication during a configuration change, but in order to fully understand this component, a short introdution of the other components will be called for.

```
interface Protocol {
  ...
  command void start();
  event void firstEncounter(uint8_t securityLevel);
  command void createGroup(uint8_t securityLevel);
  event void joinedGroup();
  event void blinkReceived(bool on);
  command void blinkSend(bool on);
  ...
  command void login();
  command void logout();
  ...
  command void abort();
  event void error();
  ...
}
```

Fig. 2. `Protocol` interface excerpt.

The `MetaStore` keeps all the data and logs presented in table I. When the sensor-node is first added to a group, the `Protocol` will save *group-id*, *group-key* and *group-time* here along with *node-genealogy* and perhaps the first element of *group-CA-list*. When the node is already a member of a group and encounters another node, which seeks to join the group, the protocol may need to look up a CA public key in *group-CA-list* in order to verify the request – and if the process was successful, a new child must be added to the *node-children*.

All the AES encrypted communication goes through `SecureComm`. This module takes care of the dissemination of the *group-CA-list* and *group-meta*, and the time synchronisation of *group-time*. Furthermore, this module discovers remote servers, and if a server can present proper credentials issued by one of the known CAs, it will get access to sensor readings and configuration, updates of *group-CA-list* and *group-meta* will be accepted from the server, and log entries will be pushed to the server.

Figure 2 lists the key part of the interface between the `BLIG interaction` module and the `Protocol`, and figure 3 shows the automaton that governs the state of `BLIG interaction`. It should be noted that the `login` and `logout` commands – and the “Authorised” state – are for authorisation-nodes only, while the remaining states are used by sensor-nodes. The user interaction through BLIG [4] remains the same even with very different protocols. The only thing which is going to change, is the timing – how much time will each step require.

VI. The first `NaiveProtocol`

We shall only briefly sketch our first attempt towards a secure protocol, as it is not so interesting from a security perspective, and mostly serves as a skeleton implementation in order to test the entire platform:

As the `start` command is issued, the `NaiveProtocol` generates a random 128 bit number and transmits it over the short-range interface. When a response arrives, the process is repeated and the `firstEncounter` event is signalled. If the first encounter was an authorisation-node and no second node is found, a new group with a random key is created.

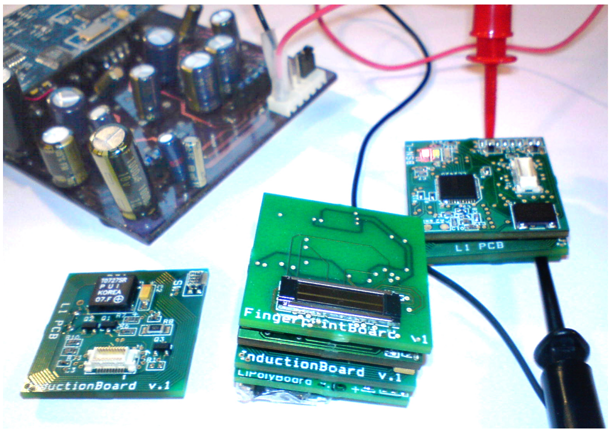


Fig. 4. Energy consumption of the protocol running on the sensor node to the right is measured using the “Energy Bucket” device seen in the background. The sensor node is communicating with the authorisation node seen at the front center. This particular authorisation node is equipped with a fingerprint reader to ensure easy and secure clinician login. At the bottom left corner, the “Induction Board” is shown, which offers short range communication using the printed coil, one push-button at the upper-right corner, and a buzzer (the black square) used by authorisation nodes to produce sounds.

Otherwise the sensor node already in the group will transmit the *group-id*, *group-key*, *group-time*, and *group-CA-list* via the radio link encrypted by the random number first sent to it by the short-range link. Furthermore, if imprinting is needed (only when the group is created), the authorisation-node sends the relevant data encrypted on the radio link in the same manner. No signatures are produced and no certificates are tested.

As explained earlier, this approach is fine, if we assume that the short-range link is truly “short-range” and that people with direct access to the patient can be trusted. However, these assumptions are not always valid.

VII. Improved Protocols

One of the greatest challenges in designing suitable protocols is the time for checking a signatures. This leads to the question of *when* must the signatures be checked and by *whom*. Furthermore, a signature could be temporarily accepted by the sensor until proved invalid. This could be implemented simply by allowing configuration change access to all who completes the protocol and delivers a signature. This signature will then be sent to the hospital server (or merely a more powerful device), and the sensor-node relies on the server to detect abnormalities. This solution does not provide a 100% safe sensor network, however, there is a very good chance that the attack – and the adversary’s identity – is discovered immediately.

Perhaps the sensor would be capable of testing the signatures by itself given a little more time (and energy), which leads to the idea of allowing temporary node additions – pending a signature check. Basically the idea is as follows: whenever a new node is added, the real *group-key* is not revealed until the signature has been verified. Instead, a temporary key is created, and the new node will not become a real member until the signature has been validated.

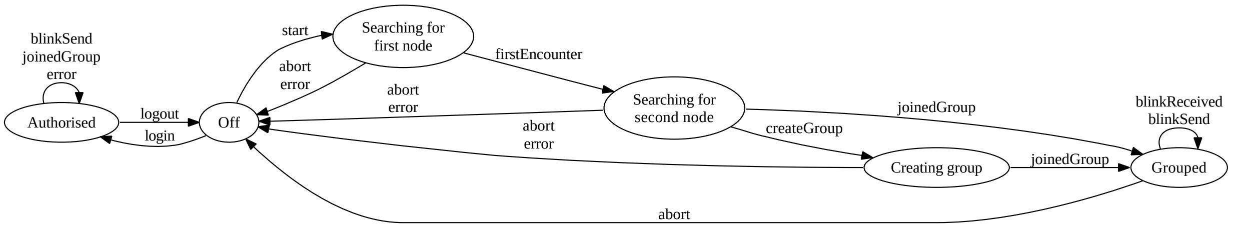


Fig. 3. `Protocol` state diagram

Another possible resource is the authorisation-node. It was mentioned earlier, that this node type may involve more expensive and powerful computing resources. As the authorisation-node must only last for a single work day before it is recharged, it may have computational power comparable to a cell phone or a PDA. This resource may be used in the design of protocols such that all or most heavy computations are placed at the authorisation-node.

All the ideas sketched above can be implemented, tested with users, and subjected to technical evaluations and comparisons on the platform. The first protocol implementing security against internal adversaries, dubbed the `AlphaProtocol`, has been implemented and tested [7], and a few variations are in the pipeline.

VIII. Conclusions and Future Works

We are in the process of finishing an experimental platform offering quick and easy development of medical sensor network prototypes with different security features for usability testing. The primary contribution of this paper is a presentation of the security framework we have chosen for this platform, and its underlying assumptions.

The current status of the platform is that the hardware has been designed and built. The sensor software implementation (TinyOS) is almost complete and tested. The server and display software is still under development. An early implementation of the BLIG user interface was tested with clinical users as reported in [4].

In addition to the `NaiveProtocol` presented in this paper, a protocol featuring security against internal adversaries was developed (presented and evaluated in [7]). More protocols with different security properties will be added in the near future, and when the selection is satisfactory, we will do another user interface test involving clinical users, this time featuring "realistic" delays and latencies in the user interface.

We also plan to perform a technical comparison of the proposed protocols, measuring the time and energy cost for each added security feature.

IX. Acknowledgments

Elements of the research presented in this paper was done as part of the Activity-Based Computing project and the PalCom project at Aarhus University, and the SAPHE project at the Imperial College London.

The Activity-Based Computing project (http://activity-based-computing.org/) is funded by the Danish Research Council under the NABIIT programme; the PalCom project (http://www.ist-palcom.org) is funded by the EU's 6th Framework Programme; and the SAPHE project (http://vip.doc.ic.ac.uk/saphe) is funded by the UK Technology Strategy Board.

References

[1] J. E. Bardram, "The trouble with login - on usability and computer security in ubiquitous computing," *Personal and Ubiquitous Computing*, vol. 9, no. 6, pp. 357–367, December 2005.

[2] J. Andersen, B. Lo, and G.-Z. Yang, "Experimental platform for usability testing of secure medical sensor network protocols," in *Proceedings of the 5th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2008) in conjunction with the 5th International Summer School and Symposium on Medical Devices and Biosensors (ISSS-MDBS 2008).* IEEE, Jun 2008, pp. 179–182.

[3] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols*, ser. LNCS, no. 1796. Springer, 2000, pp. 172–182.

[4] J. Andersen and J. E. Bardram, "BLIG: A new approach for sensor identification, grouping, and authorisation in body sensor networks," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, ser. IFMBE Proceedings, S. Leonhardt, T. Falck, and P. Mähönen, Eds., vol. 13. Berlin: Springer, Mar 2007, pp. 223–228.

[5] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *NDSS'02 – Network and Distributed System Security Symposium 2002*, Feb 2002.

[6] F. L. Wong and F. Stajano, "Multichannel security protocols," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, Oct 2007.

[7] J. Andersen, "Secure group formation protocol for a medical sensor network prototype," in *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2009).* IEEE, December 2009, IN PRINT.

Part III

Appendices

Appendix A

Short-range Communication

The *Induction Board* was developed as an extension board for the BSN platform, featuring all the extra hardware necessary to build the prototype platform described in chapter 4:

- short-range communication (featuring a bit-rate of up to 5377 bps),
- a push button, and
- a buzzer (only used by the authorisation-nodes)

This appendix offers some technical details of this extension board, followed by a brief presentation of the accompanying TinyOS libraries.

A.1 The Induction Board

The Induction Board was designed using the same dimensions as the BSN mote board [21], in order to allow stacking the boards together in a nice way as demonstrated on figure A.1. This figure shows two nodes communicating

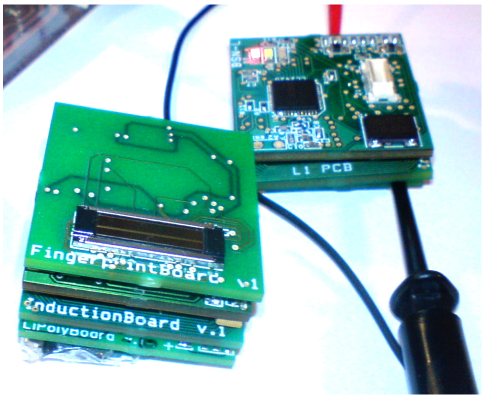


Figure A.1: Short range communication between two nodes.

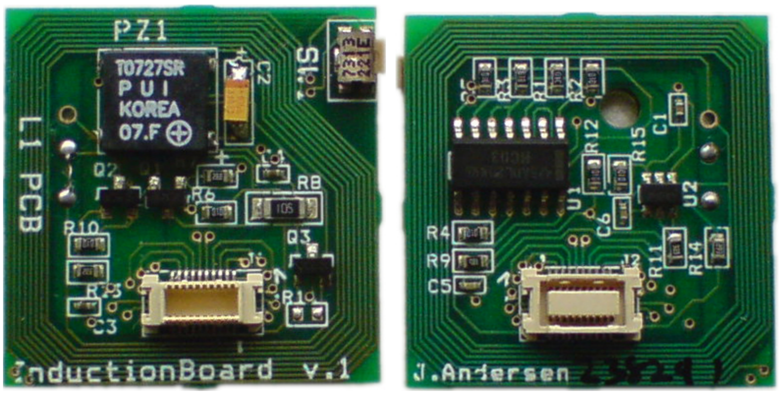


Figure A.2: The Induction Board top and bottom view.

over the short-range interface (an authorisation-node at the front left, and a sensor at the back).

Figure A.2 shows the top and bottom side of the board. Since the board is typically used in-between other boards (cf. figure A.1), the push-button (`SW1`) is located at the edge, so it can always be operated. The black square (`PZ1`) on the top layer is the buzzer used for authorisation nodes.

As suggested by the name *Induction Board*, short-range communication is realised using electromagnetic induction. The PCB is made up of 4 layers, and features a printed coil (seen along the edges on figure A.2), which extends through all 4 layers with a total of 37 turns. The transmitter produces magnetic pulses by drawing a current through its coil. These pulses are then picked up by the receiver coil and amplified.

The full schematic of the board is shown on figure A.3. The *Direction* line, which is connected to the `ADC4` pin on the MSP430 micro-controller (MCU), decides the direction (output or input) of the short-range communication. In *input* mode (low), the amplifier circuit around `Q3` and `U2` is turned on, and draws a small amount of current (about 2 mA). The amplification of this circuit is fixed to allow a range of about 5 cm.

The transmitting MCU pulls the *Data* line down resulting in a current through the transmitter coil, in turn inducing a current in one direction in the receiver coil. 61 μs later (2^{-14} s) the *Data* line is reset turning the transmitter's coil current off, resulting in an induced current in the opposite direction at the receiver coil.

The receiver amplifier is primarily sensitive to induced currents in one direction, so depending on the relative transmitter and receiver coil orientation, the receiver may be triggered by the first or second pulse, or perhaps both. The receiver amplifier is equipped with a timer (positive feedback by `C1`), which will assert the *Data* line for at least 72 μs, in order to guarantee that one pulse from the transmitter will not be received as two pulses.

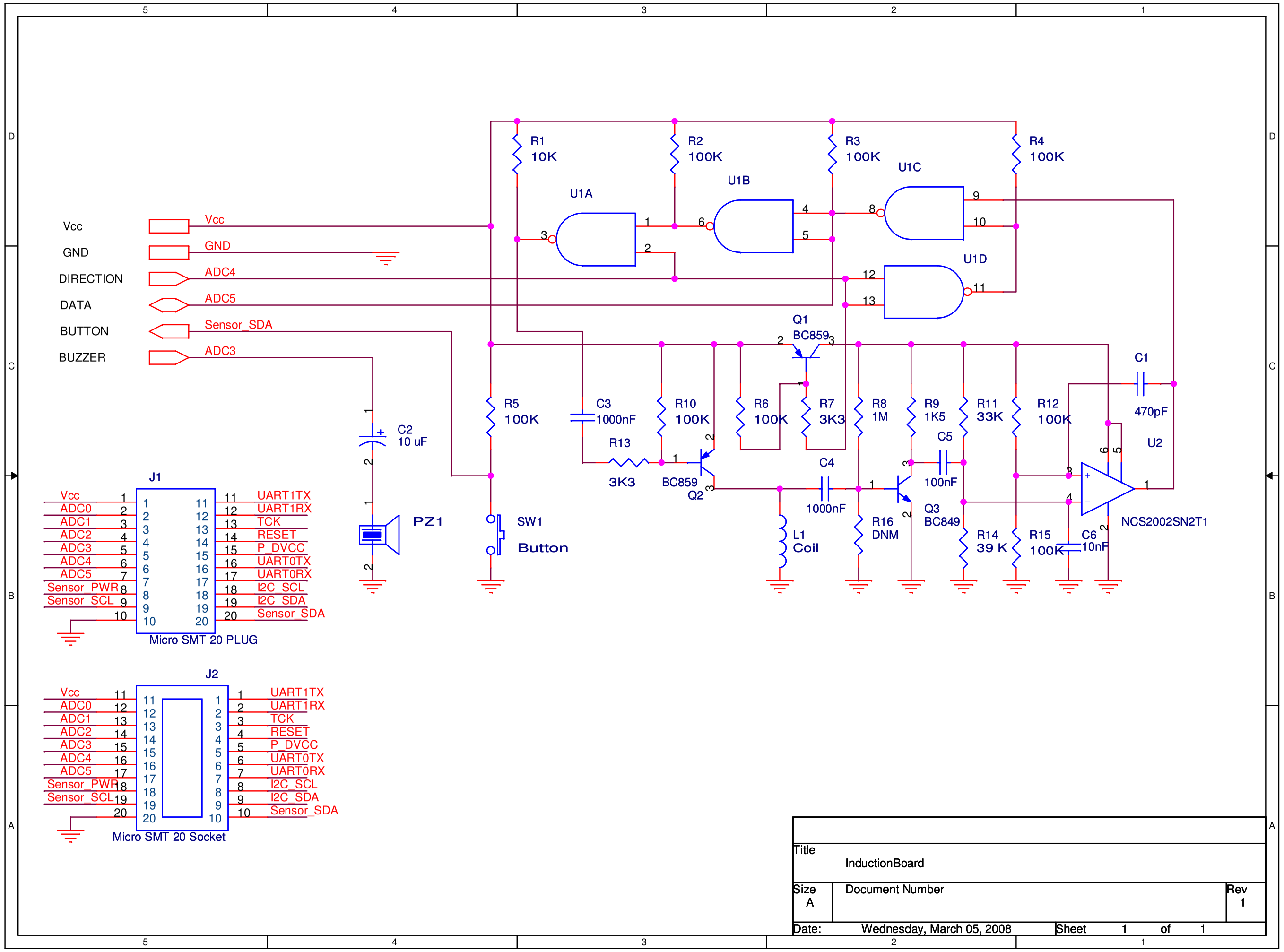
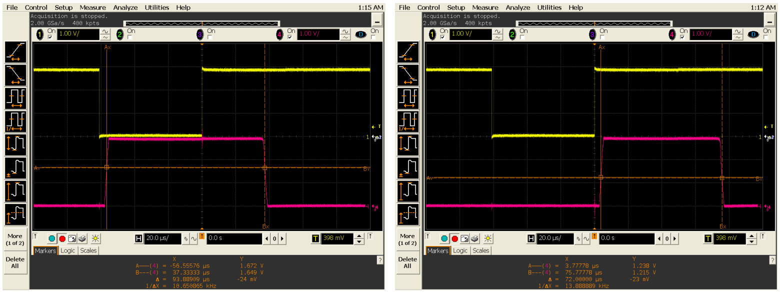


Figure A.3: Schematic of the short range communication board.



```
Acquisition is stopped.
2.00 GSa/s   400 kpts
```

```
                 X                    Y
A——(4) = -56.55576 µs         1.672 V
B---(4) =  37.33333 µs         1.649 V
      Δ =  93.88909 µs        -24 mV
  1/ΔX =  10.650865 kHz
```

```
                 X                    Y
A——(4) =  3.77778 µs          1.238 V
B---(4) =  75.77778 µs         1.215 V
      Δ =  72.00000 µs        -23 mV
  1/ΔX =  13.888889 kHz
```

Figure A.4: Pulse reception timing measurements with different transmitter/receiver orientations.

Figure A.4 shows measurements of the received pulse relative to the transmitted pulse. In both measurements, the yellow signal is the output of the transmitting MCU, and the magenta is the input on the receiving MCU. The left measurement shows the reception when the receiver picks up the first pulse—the second pulse is also picked up partially, but it just results in an extension of the pulse width (to 94 μs). The measurement to the right shows the result when the receiver fails to pick up the first pulse.

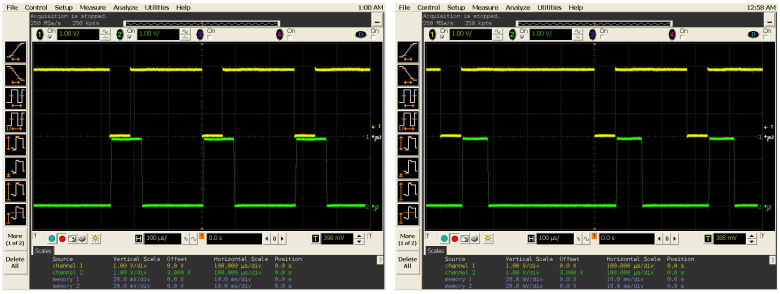
As explained above, the receiver circuit requires a current of about 2 mA to power an active amplifier. It is in fact possible to construct a passive receiver, capable of detecting a long series of pulses (perhaps tuned to a certain frequency) without an amplifier—i.e. with *no* power requirements. Such a passive receiver could be used to wake up the MCU and power on the receiver's amplifier when transmissions are detected.

In order to keep things simple, such a passive receiver was not included on this version of the Induction Board, however, this may be included on later versions.

The Induction Board proved very reliable and unsusceptible to noise and interference. It was tested with the data transmission library presented in the following section in many different noisy environments with no observed interference or data loss. The only exceptions were within 10 cm of an electric engine (power drill), and in a kitchen within 45 cm of an induction cooker at full power.

A.2 TinyOS Library

Conversion between series of pulses and digital data is a very common problem with well-known solutions; for instance, storing data on a magnetic or optical disk involves converting data into a series of magnetic flux reversals (on magnetic disks) or height changes (on optical disks). The main problem here is how to convert the pulses back to data, since no common clock is available.



```
Acquisition is stopped.
250 MSa/s    250 kpts
1 On 1.00 V/   2 On 1.00 V/   3 On   4 On   D On
H 100 µs/   0.0 s   T 398 mV

Source      Vertical Scale  Offset   Horizontal Scale  Position
channel 1   1.00 V/div      0.0 V    100.000 µs/div    0.0 s
channel 2   1.00 V/div      3.000 V  100.000 µs/div    0.0 s
memory 1    20.0 mV/div     0.0 V    10.0 ms/div       0.0 s
memory 2    20.0 mV/div     0.0 V    10.0 ms/div       0.0 s
```

```
Acquisition is stopped.
250 MSa/s    250 kpts
1 On 1.00 V/   2 On 1.00 V/   3 On   4 On   D On
H 100 µs/   0.0 s   T 388 mV

Source      Vertical Scale  Offset   Horizontal Scale  Position
channel 1   1.00 V/div      0.0 V    100.000 µs/div    0.0 s
channel 2   1.00 V/div      3.000 V  100.000 µs/div    0.0 s
memory 1    20.0 mV/div     0.0 V    10.0 ms/div       0.0 s
memory 2    20.0 mV/div     0.0 V    10.0 ms/div       0.0 s
```

Figure A.5: Pulse train.

To solve this problem and reconstruct the clock at the receiver, some kind of coding technique must be used, which can guarantee a low upper bound on the duration between pulses. The standard solution is to use a member of the *Run-Length Limited* (RLL) family of coding techniques. The term “run length” refers to the duration between pulses, and as the name suggests, these coding techniques guarantees an upper limit on this duration.

For the TinyOS library implementation, a coding scheme known as (2,7) RLL was used. This particular scheme was chosen because it is simple to implement, and also due to its efficiency; in fact, if t_0 is the minimum time between two pulses, (2,7) RLL achieves a bit-rate of $\frac{3}{2} \cdot t_0^{-1}$ (by comparison, this is a 50 % improvement over the naive coding scheme of encoding a ‘1’ as a pulse and ‘0’ as no pulse—and this coding scheme does not even feature run-length limiting).

In order to synchronise transmission and reception, and provide error detection, data is organised in packets of flexible length (1–256 bytes) protected by a 16 bit *Cyclic Redundancy Check* (CRC) value.

The start of a packet is indicated by a synchronisation header, which must be distinguishable from any code that can occur during normal data transmission. Since the (2,7) RLL code is used, no run-lengths (pause between pulses) greater than 7 clock-ticks will occur in normal data, so any run-lengths of at least 8 followed by a pulse, signifies the start of a packet.

In the chosen coding scheme, a pulse followed by 3 or more clock-ticks of silence, corresponds to the encoding of two 1-bits, so this bit sequence was chosen as the header signature. Following these two bits, the length of the data section of the packet follows. Valid lengths are 1–256, encoded by 8 bits (with 256 encoded as 0). Following the data section is the 16 bit CRC. The complete packet layout can be visualised as:

11	length (8 bits)	data (length · 8 bits)	CRC (16 bits)

Figure A.5 shows measurements of received pulses relative to the transmitted pulses. In both measurements, the yellow signal is the output pin of the transmitting MCU, and the green is the input pin on the receiving

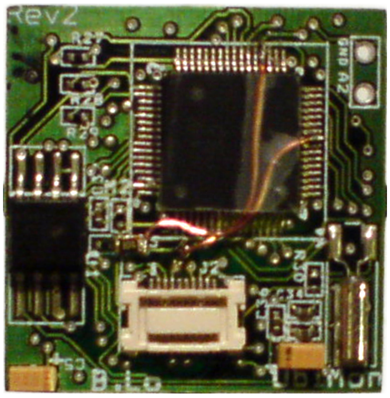


Figure A.6: The modified BSN board with two jumpers added.

MCU. Like figure A.4, the two images show the results of different transmitter/receiver orientations, and it is evident, that the only difference is a constant delay, which will not influence the interpretation of the pulses at the receiver.

The MSP430 MCU has a number of timer I/O pins, which may be used for automatic timing measurements and generation of pulses. Unfortunately, none of these pins are available in the BSN mote extension connector (nor on the Tmote Sky extension connectors, for that matter).

If a timer I/O pin is not used to generate and detect the pulses, the accuracy of the pulse generation as well as the detection must be performed by pulse generation/reception interrupt routines, and will depend on, whether these interrupt routines are allowed to run immediately, or are delayed by other interrupts. This implies, that the interrupt routines (or code sections which disable interrupt handling) in other application sub-systems could interfere with the pulse timing, causing packet damage and loss. This of course, would be an unacceptable consequence, so instead the BSN mote board was modified to export one of the unused timer I/O pins (`TB0`) to the extension connector. This was done by adding a jumper connecting `TB0` with `ADC5`, to which the *Data* line of the Induction Board is connected. Figure A.6 shows this modification—along with another modification, which enables a clock speed of up to 8 MHz on the MSP430 MCU (without this modification, it would only be capable of running at 4 MHz).

In the current implementation of the TinyOS library, the physical bit-rate was chosen as $\frac{2^{14}}{3} = 5461$ bps. The maximum data throughput at this bit-rate is 5377 bps (a difference of a factor $\frac{256}{256+4}$ at a packet size of 256). At this bit-rate the minimal time between pulses is 275 μs, which may seem a lot, when the pulse width is less than 100 μs (cf. figure A.4). It would probably be possible to double the bit-rate without causing any problems, but the bit-rate was chosen very conservatively, in order to avoid

Program A.1 `Button` and `Beep` interfaces.

```
interface Button {

  event void shortPress();
      Signals a button click (shorter than 1 second).
  event void longPress();
      Signals a button hold (longer than 1 second).
  command bool bootPress();
      Did the user press the button during boot?
}

interface Beep {

  command void shortLow();
      500 Hz tone playing 100 ms.
  command void longLow();
      500 Hz tone playing 500 ms.
  command void shortHigh();
      3 kHz tone playing 100 ms.
  command void longHigh();
      3 kHz tone playing 500 ms.
  command void tripleHigh();
      Three 3 kHz beeps (75 ms each with 50 ms pause).
  command void fourthUp();
      Signal consisting of two beeps (750 Hz + 1 kHz) for 100 and 200 ms.
  command void fourthDown();
      Signal consisting of two beeps (1 kHz + 750 Hz) for 333 and 666 ms.
  command void chirp();
      Signal consisting of a 1 s down-chirp starting at 2 kHz.
}
```

future problems (when the library is used by applications which interfere and behave badly).

Program A.1 lists the TinyOS interfaces for accessing the button and buzzer. The `Button` interface distinguishes between a long press (more than one second) and a short press, matching the similar distinction in the BLIG user interface (cf. chapter 3). The `Beep` interface provides a number of commands corresponding to different audio signals, which may be used for various cues or alerts.

The short-range communication library exposes two layers: a pulse layer and a packet layer. The pulse layer interface, `InductionPulse`, shown in program A.2, provides commands and events to fire and detect pulses. In order to support communication using series of pulses, this interface was

Program A.2 `InductionPulse` interface.

```
  async command error_t setIn();
```

Sets the induction hardware to input mode.

```
  async command error_t setOut();
```

Sets the induction hardware to output mode.

```
  async event void fired(uint16_t dt);
```

A pulse was received. The time argument describes the elapsed time since the previous pulse was received (in $\frac{1}{32768}$ seconds).

```
  async command error_t fire(uint16_t dt);
```

Send a single pulse now. If `dt!=0` use this pulse as the starting point of a pulse series, where the time difference between the two first pulses will be $\frac{dt}{32768}$ seconds.

```
  async event uint16_t fireDone(error_t status);
```

Event requesting the next time difference for a pulse series.

Program A.3 `InductionTx` and `InductionRx` interfaces.

```
interface InductionTx {

  command error_t send(uint8_t length, void *buffer);
```

Transmit the buffer via the induction interface.

```
  event void sendDone(void *buffer);
```

Signalled when a message was sent via the induction interface.

```
}

interface InductionRx {

  command error_t start(uint8_t length, void *buffer);
```

Starts the induction receiver for messages of maximum length given by the `length` argument.

```
  command void *stop();
```

Stops the induction receiver.

```
  event void *messageReceived(uint8_t length, void *buffer);
```

Signalled when a message was successfully received.

```
  event void bufferReturn(void *buffer);
```

Returns an Rx buffer which is no longer used by the induction subsystem.

```
}
```

designed to transmit and receive pulses using time values measured relative to the time of the previous pulse (rather than absolute time).

Program A.3 lists the `InductionTx` and `InductionRx` interfaces, that are used for packet-based communication using the (2,7) RLL encoded packets described above. Packets of any length in the interval (1–256) may be sent or received.

Appendix B

Fingerprint Reader

A sensor board containing a fingerprint reader was developed for the BSN mote platform [21] to be used as the primary user input interface for authorisation nodes. This appendix presents a brief overview of this sensor board, the accompanying TinyOS library, and a rechargeable battery board, which was built to accompany this fingerprint reader.

B.1 The Fingerprint Board

The fingerprint reader is based on a popular chipset from UPEK [109] commonly used for cell phones and laptops. This chipset comprises of a fingerprint swipe sensor, *TCS3C*, and a biometric image processor, *TDC42*.

The main features of this chipset includes:

- Enroll a fingerprint into a template database, or verify the fingerprint against the prints already stored in this database.
- Image capture: returns a bitmap image of the scanned finger.
- Navigation mode: finger movements are tracked—typically used for moving a cursor (similar to a touchpad).
- Power supply requirements: 3.0–3.6 V, 70 μA–164 mA.
- Serial connection (RS-232 or USB) to host processor.
- Low-current sleep mode (further details below) with fast wake-up when a finger touches the sensor surface.
- 4 kB non-volatile memory capable of containing a database of up to 21 fingerprint templates with an encrypted payload (e.g. a user password) which can only be decrypted and released when the corresponding fingerprint has been read—external flash may be added if 4 kB is not sufficient.

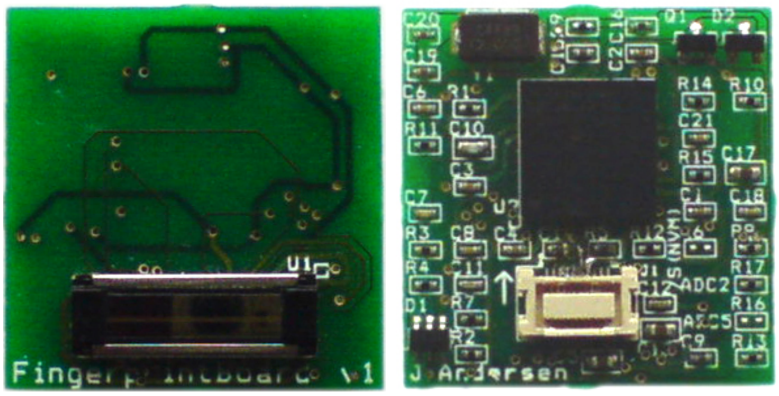


Figure B.1: The fingerprint reader board top and bottom view.

When the swipe sensor, sleeps with a current draw of about 100 μA, it is able to detect the touch of a finger, wake up in time to perform the image scan, turn on the image processor chip to perform the actual processing, and finally go back to sleep. When both chips are fully on, a current draw of about 110 mA can be expected (measurements of the current draw of the entire system including the BSN host mote with active radio have shown a total current draw of about 130–140 mA). According to the chipset's datasheet, currents as high as 164 mA are possible, but such high currents were never observed during the experiments described in this dissertation. As the chipset requires at least 3.0 V at these high currents (which is a lot more than the coin-cell batteries, normally used for the BSN motes, can deliver), a special rechargeable battery board was constructed as well, which is briefly presented in section B.3 below.

Figure B.1 shows the sensor board. The swipe sensor is the only component on the top side (to avoid fingers accidentally touching other components), and all remaining components, including the biometric image processor, are located on the bottom side. Figure B.2 shows how the fingerprint reader is used to construct an authorisation-node by stacking 4 boards together: the fingerprint reader, the BSN mote, the induction board, and finally at the bottom, the rechargeable battery board.

The full schematic of the fingerprint reader board is shown on figure B.3. It is (with only minor exceptions) identical to the reference schematic published by UPEK, and the chipset connects to the following MSP430 microcontroller pins on the BSN mote: `UART0` (`-RX` & `-TX`), port `1.6`, and `ADC5`. Port `1.6` on the MSP430 can be used to trigger interrupts, and the UPEK chipset will use this line to wake up the MSP430 when a finger touches the fingerprint reader.

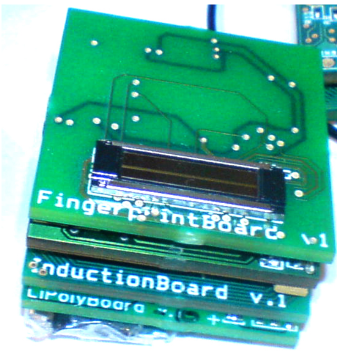


Figure B.2: The fingerprint reader and rechargeable battery board used for an authorisation-node.

B.2 TinyOS Library

The UPEK chipset communication protocol consists of three layers: application, transport, and link layer.

The *application layer* interface provides a number of commands, such as “enroll a finger into template database”, “scan and verify finger”, or “sleep”. Each command call returns a response consisting of a return code (error message) and optionally additional information (for instance, a “scan finger” command may return a bitmap image of the scanned finger). Command calls may block indefinitely waiting for the user to scan a finger, and can be aborted by a concurrent call to an “abort” command—causing the original call to return an “aborted” return code. During a command call which involves scanning a user’s finger, directions for the user may be signalled from the application layer, for instance “swipe was too fast, try again”, or “sensor dirty, please clean it”. Such messages should be handled by upper layers and presented to the user.

The *transport layer* sits between the application and link layers, and its primary job is to handle message fragmentation and defragmentation, as the link layer transmits only short packets (2 kB max), while the application layer’s messages and responses may be much longer.

The *link layer* offers a number of different functions, which can be summarised by the following list:

- Opening and closing a connection, resetting the sensor, changing baud rate and other connection-specific parameters.
- Error detection and reliable packet transfer—retransmission of lost or damaged packets.
- Detect and report a lost connection.

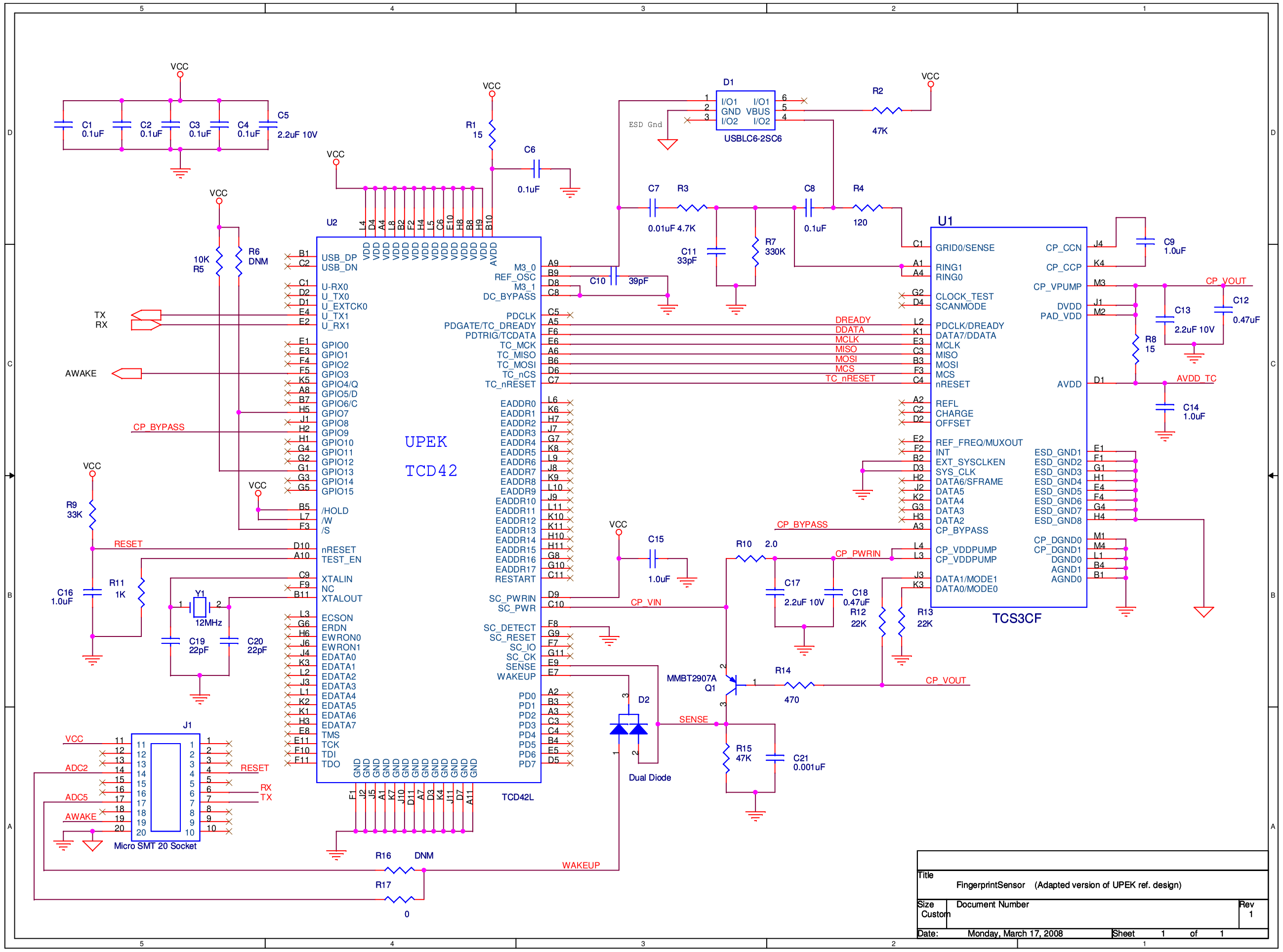


Figure B.3: Schematic of the fingerprint reader board. This is an adaptation of the reference schematic provided by UPEK.

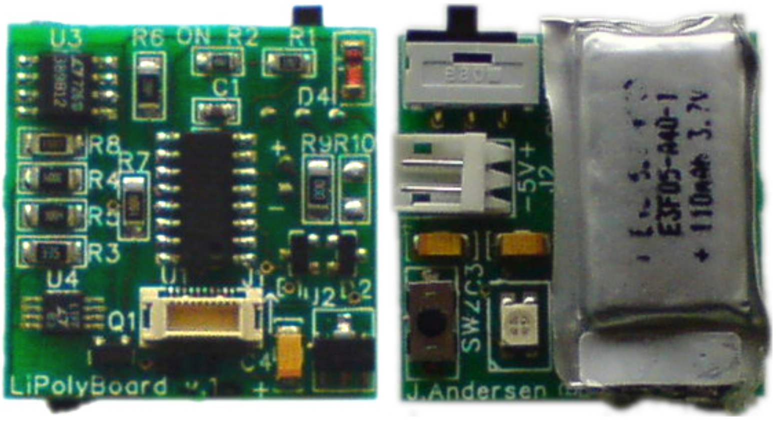


Figure B.4: The rechargeable battery board top and bottom view.

- Putting the sensor to sleep, and signalling when it wakes up by the touch of a finger.
- Handle cursor movements when the sensor is in navigation mode.

Currently, the TinyOS implementation of this communication protocol stack covers only the transport layer and most of the link layer[1]. This implies that TinyOS applications using the fingerprint reader must assemble the command message according to the application layer specification in a message buffer and hand it directly to the transport layer using the `PT_TransportLayer.sendMessage` command. Responses are signalled with the `PT_TransportLayer.receiveMessage` and `receiveDone` events, and it is up to the application to determine whether this message was a response to the original request, or some direction for the user. A full overview of the transport layer interface is provided by program B.1

B.3 Rechargeable Battery Board

The standard battery board for the BSN mote uses a 3 V coin battery (CR2430) which is not capable of keeping a voltage of 3.0 V under high load. As the fingerprint reader requires a supply voltage in the 3.0–3.6 V range, the rechargeable battery board was constructed to provide a small (and rechargeable) battery-based power supply for the BSN mote, capable of delivering the relatively high currents necessary to use the fingerprint reader while keeping the voltage steady above 3.0 V.

[1] A part of the link layer dealing with cursor movements when the sensor is in navigation mode was not fully implemented—for instance errors are not handled properly in this mode. As navigation is never used in the prototype described in this dissertation, this deficiency does not affect the prototype functionality in any way.

Program B.1 `PT_TransportLayer` Fingerprint reader transport layer API.

```
command error_t open(bool slowBus, uint32_t timeout, uint16_t receiveBufSize);
        Open the connection to the fingerprint module.

event void openDone(pt_status_t status);
        Signals that the open command has finished and the link layer is ready to accept messages.

command error_t close();
        Closes the connection.

event void closeDone();
        Signals that the close operation has finished.

command void abort();
        Sets the “abort” flag.

event void commFailed(pt_status_t status, bool fatal, void *buf1, void *buf2);
        Reports failure of the communication.

command error_t sendMessage(uint16_t messageLength, void *fragment, uint16_t fragmentLength,
                            bool inSessionSleep, void *buf1, void *buf2);
        A message will be sent to the fingerprint module.

event void *sentFragment(void *fragment, uint16_t *length);
        A fragment was sent successfully, and the transport layer is ready to transmit the next fragment.

event void sendDone(void *fragment, uint16_t length);
        Signalled when a complete message has been transmitted successfully, or before a commFailed event is signalled
        due to an abort or transmission error during message transmission.

event void receiveMessage(uint16_t length);
        Signals the reception of a message and reports the total length. The data will arrive with the events
        receivedFragment and receiveDone

event void *receivedFragment(void *fragment);
        Signalled when the receive buffer is full and the entire message was not yet received.

event void receiveDone(void *fragment, uint16_t fragmentLength, void *spareBuf);
        Signalled when a complete message was received.

command error_t rawContinue();
        The raw data mode continue command is sent to the fingerprint module.

command error_t rawCancel();
        The raw data mode cancel command is sent to the fingerprint module.

event void *receiveRawPacket(void *raw, uint8_t length);
        Signals the reception of a raw packet (cursor movements in navigation mode).

async command bool isAwake();
        Checks whether the fingerprint module is awake or not. This works even when no connection is open and
        asynchronously.

command error_t wakeup();
        Wakes up the sensor from in-session sleep.

event void wakeupByFinger();
        Signalled whenever the fingerprint module wakes up due to the touch of a finger (during out-of-session sleeps
        only).

command bool pauseReceive();
        Pauses the incomming message.

command error_t resumeReceive(void *buf);
        Resumes a previously paused reception.
```

As the BSN mote accepts supply voltages in the range 1.8–3.6 V, and the recommended supply voltage for the fingerprint reader is 3.3 V, this value was chosen as output voltage for the battery board in order to allow a small voltage-drop under high load.

The battery board is shown on figure B.4. The 3.7 V polymer lithium-ion rechargeable battery chosen for this board is located on the bottom side of the circuit board, along with the power switch, the reset button, an external power connector, and a charge status indicator LED.

Figure B.5 provides the full schematic of the rechargeable battery board. Rather than having an external battery charger, a complete charge circuit (`U1` and a few surrounding components) was included on the board, so any external 5 V supply can be used to charge the battery.

As any lithium-ion battery would be ruined if discharged below a certain voltage level, the battery board contains a protection circuit designed to cut off the power if the battery voltage falls below approximately 2.8 V. This circuit is realised by `U3`, `U4`, `Q1` and a few resistors.

Finally, a voltage regulator, `U2`, adjusts the output voltage to 3.3 V. This regulator serves a double purpose. First, it regulates the voltage to 3.3 V, protecting the mote circuit from higher voltages—the battery may deliver up to 4 V when fully charged. Secondly, it protects the battery from short-circuits and similar dramatic current overloads by cutting off the output.

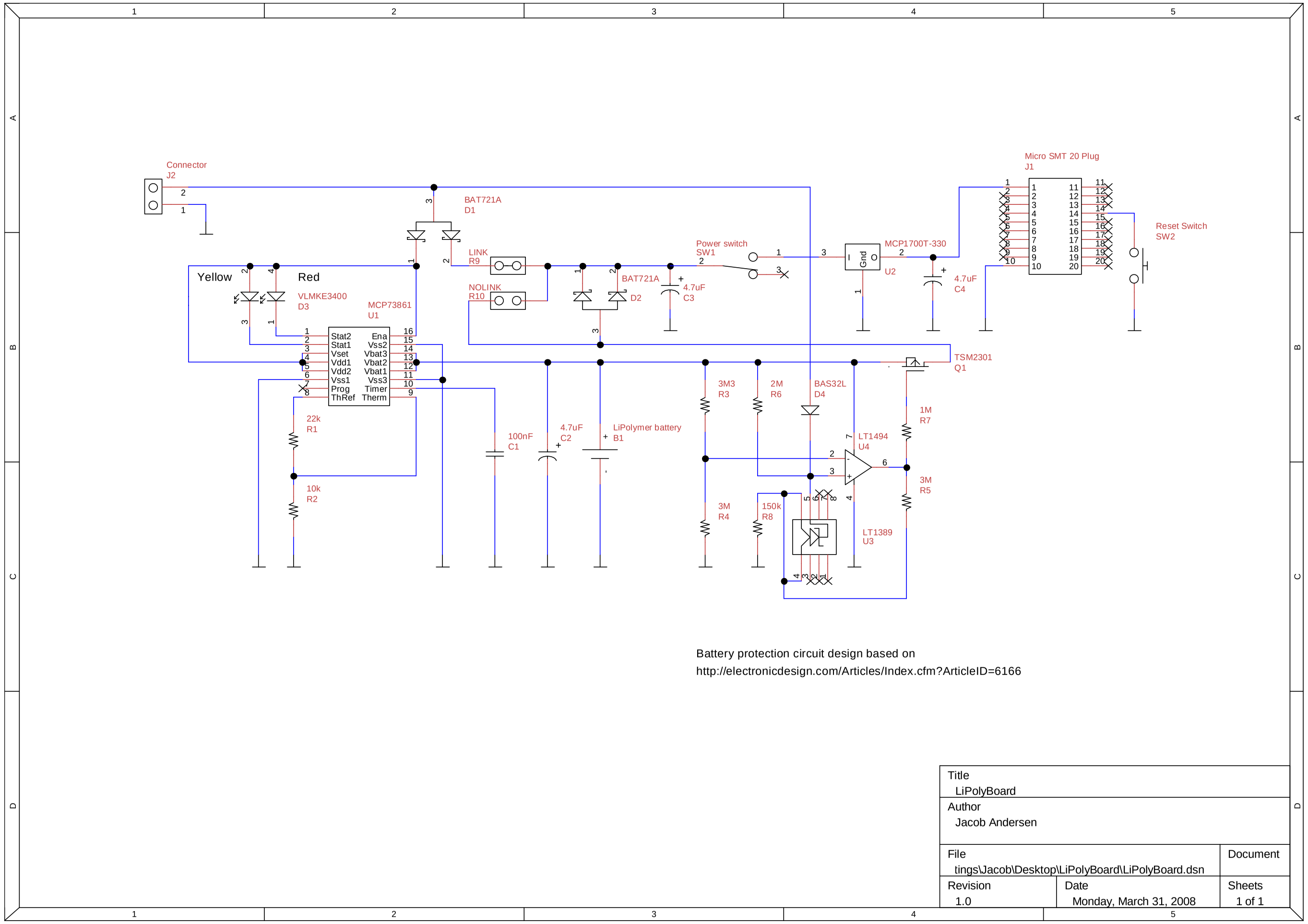


Figure B.5: Schematic of the rechargeable battery board.

Appendix C

Energy Bucket

This appendix offers the full schematic and some extra and more detailed information about the Energy Bucket presented in chapter 6 and the USB cable extended with an internal power switch. Following a discussion about energy measurement methods, the Energy Bucket circuit board is presented with a description of each of the main building blocks along with the calibration and accuracy measurements performed. After this, the modified USB cable is presented, and the appendix is concluded with a step-by-step walk-through of the measurement process.

C.1 Energy Measurement Method

The energy consumption of an electric circuit can be measured in a number of different ways, but the method most commonly found in papers describing experimental evaluations of sensor network applications involves measuring the voltage across a *shunt resistor*—a small resistor added in series between the Target Circuit (TC) and the power supply. This is also the method found in most of the related work cited in section 6.2.

C.1.1 The "Shunt Resistor Voltage" Method

The derivative of electrical energy consumption with respect to time (the power) can be calculated as the product of voltage and current:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = u_{TC}(t) \cdot i_{TC}(t) \tag{C.1}$$

where $u_{TC}(t)$ is the voltage over the TC and $i_{TC}(t)$ is the current through the TC at time t. Measuring $i_{TC}(t)$ is performed using an oscilloscope measuring the voltage over a fixed small resistor R (the shunt resistor), see the schematic in figure C.1. R is supposed to be orders of magnitudes smaller than the input impedance of the oscilloscope, and therefore the

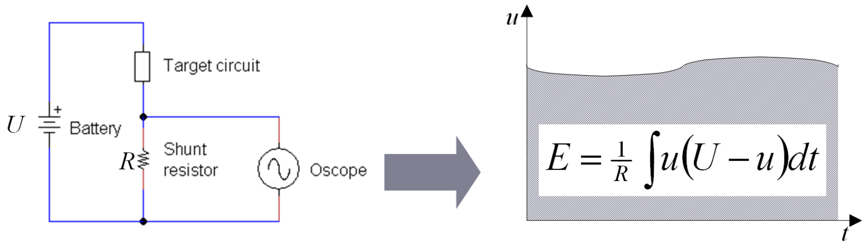


Figure C.1: Using a shunt resistor to measure energy consumption.

current through the shunt resistor equals the current through the target circuit, and

$$i_{TC}(t) = \frac{u(t)}{R} \tag{C.2}$$

where $u(t)$ is the voltage measured by the oscilloscope.

Assuming the power supply delivers a constant voltage, U, the voltage $u_{TC}(t)$ is given by

$$u_{TC}(t) = U - u(t) \tag{C.3}$$

and combining equations (C.1)–(C.3), we get the energy, $E(\tilde{t})$, consumed in the time-interval 0–$\tilde{t}$:

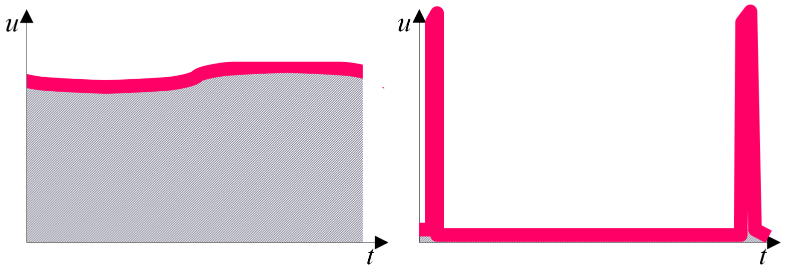
$$E(\tilde{t}) = \frac{1}{R} \int_0^{\tilde{t}} u(t) \cdot (U - u(t))\mathrm{d}t \tag{C.4}$$

As $u(t)$ is usually kept much smaller than U, this expression is often simplified to:

$$E(\tilde{t}) \simeq \frac{U}{R} \int_0^{\tilde{t}} u(t)\mathrm{d}t \tag{C.5}$$

Calculating the energy consumption is now simply a matter of integrating the measured voltage data as illustrated in figure C.1, and this is a widely used method, which works very well—in most cases.

The accuracy of the measurement (the computation of the integral) obviously depends on the granularity of the vertical and horizontal axes, but an often overlooked factor, influencing the accuracy, is the shape of the graph. Take for instance the graph shown in figure C.1, which is typical for most electronic equipment. Emphasising the vertical and horizontal uncertainties, we may get something like the graph shown in figure C.2a, with the red area illustrating the measurement uncertainty (while the hatched area is certain). Since the red area is much smaller than the hatched area, the relative accuracy of the measurement is high. Now, compare this to the measurement in figure C.2b using the same scale and horizontal and vertical accuracies. Clearly, the relative accuracy in this case is much worse. Note that the maximum voltage decides the scale used by the oscilloscope, so a better resolution could not be obtained by just selecting a lower voltage setting, as



(a) A good case

(b) A bad case

Figure C.2: A good and bad case for the shunt resistor voltage method.

this would overload the oscilloscope input. Since almost all sensor network applications would have a power profile similar to that of figure C.2b, using this measurement method for sensors should be done with great caution.

Another problem is caused by the fact that most oscilloscopes have a vertical (voltage) resolution of 8 bits, and in general, significantly better resolutions are both rare and expensive. This implies that only 256 discrete steps are available on the vertical axis. A typical sensor mote (like the Tmote [106]) may draw up to 50 mA when active (radio, external flash and LEDs on), perhaps even more with external sensor hardware. When the mote is asleep, it draws about 5 μA, and recent MCUs like the Microchip XLP [74] sleeps in the nano-Ampere range. If, for instance, the oscilloscope range (and resistor R value) was chosen to allow a maximum current of 50 mA, the step-size for an 8-bit oscilloscope would be approximately 200 μA, and it would be impossible to measure any sleep current, as illustrated in figure C.3. For many sensor network applications, this is unacceptable, as the energy consumption during sleep can contribute significantly to the overall energy consumption.

An instrument capable of measuring the full range of current draw from a sensor mote should at least be capable of covering the range 0.5 μA–100 mA, corresponding to a vertical resolution of 18 bits. In fact, the Microchip XLP micro-controllers are supposed to be able to go as low as 0.05 μA [74], and a mote equipped with the fingerprint reader from appendix B may require up to 150 mA when fully active, so the range may in some cases be larger.

While oscilloscopes in general have poor vertical resolution, they have very good horizontal (temporal) resolutions—often in the GHz range. However, a temporal resolution of this magnitude is not really useful for an instrument measuring the current along a power supply line. Usually a lot of decoupling capacitors are used in the target circuit to decouple subcircuits and isolate noise, and these capacitors will act as a low-pass filter on the power supply line, resulting in only slow variations (compared to the temporal resolution of the oscilloscope) in current draw.

Rather than using an oscilloscope to capture the voltage data, a much

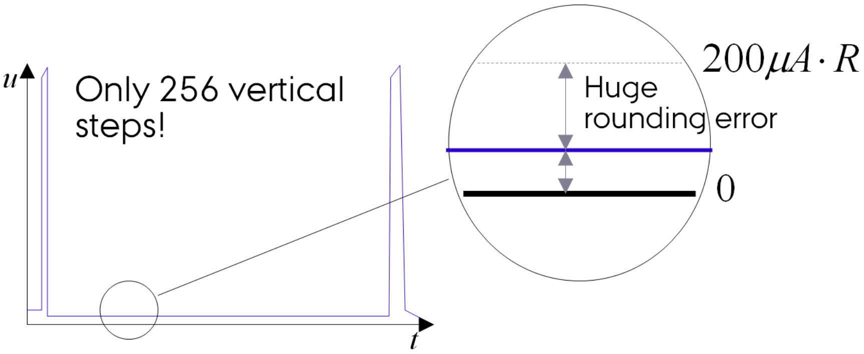


Figure C.3: Rounding error using an 8-bit linear voltage scale when the shunt resistor R is chosen to allow a maximum current of 50 mA. Note how sleep currents below 200 μA leads to huge rounding errors—sleep currents for sensor nodes may go as low as into the nA range!

slower analog-to-digital converter (ADC) with a higher voltage resolution could be used. ADCs used to digitise audio would be a cheap alternative; a resolution of 24 bits is quite common, and a chip containing a 24-bit ADC capable of sampling at 96 kHz may be purchased for a couple of Euros.

C.1.2 The "Charge Pump" Method

In order to measure the energy consumption, the shunt resistor voltage method measures the current through the target circuit and computes the integral. Since *current* is the derivative of *charge*, a more direct approach would be to measure the amount of charge, which passes through the target circuit. This would simplify the calculations. Assuming like before, that the voltage, U, over the target circuit is constant, the energy consumption is given by:

$$E(t) = U \cdot q(t) \tag{C.6}$$

where $q(t)$ is the amount of charge having passed through the target circuit at time t.

Charge can be measured using a *charge pump*. If the pump is constructed in such a way that each stroke delivers exactly the same amount of charge, measuring the total amount of charge delivered by the pump becomes a matter of counting the number of pump strokes.

C.2 The Energy Bucket Board

The Energy Bucket was designed as a Tmote Sky [106] extension board. A picture of the device is shown on figure C.4, with the Tmote mounted on top of the Energy Bucket circuit board. The circuit schematic is shown on

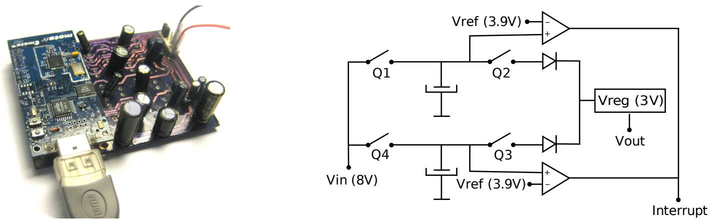


Figure C.4: The Energy Bucket with an overview schematic.

figure C.5 and consists of a number of functional blocks which will be explained below. The overall design of the device is illustrated by the overview schematic on figure C.4, and comprises the following main building blocks:

- 8 V voltage generator
- Charge pump circuit
- 3.9 V voltage reference and comparators
- Output voltage regulator (3.0 V)
- Status signals from target circuit (not shown on the overview schematic)

C.2.1 8 V Voltage Generator

The Energy Bucket is powered through the attached Tmote Sky, which in turn is powered from the USB port of the host PC. The Tmote operates at 3 V, and since the voltage of the USB supply is 5 V, the Tmote employs a low-dropout voltage regulator which is only capable of delivering an additional 200 mA.

As the Energy Bucket needs at least 150 mA at 8 V, a step-up converter starting from 3 V would require roughly 500 mA, which is much more than the Tmote voltage regulator would be able to deliver. Therefore, the 8 V supply is generated directly from the USB 5 V supply, and a boost converter (`U1`) followed by a voltage regulator (`U2`) generates 9 V and regulates it to 8.0 V, which is then made available over the `C5` capacitor.

The voltage generator is controlled by the Tmote via a signal generated by `U4B`, and is turned on only when needed.

C.2.2 Charge Pump Circuit

The charge pump comprises two identical circuit paths each with a “bucket” capacitor which is alternately charged to (almost) 8.0 V through a bipolar transistor (`Q2` and `Q3`) and then discharged through a MOSFET transistor (`Q8` and `Q9`). The four transistors are individually controlled by the Tmote

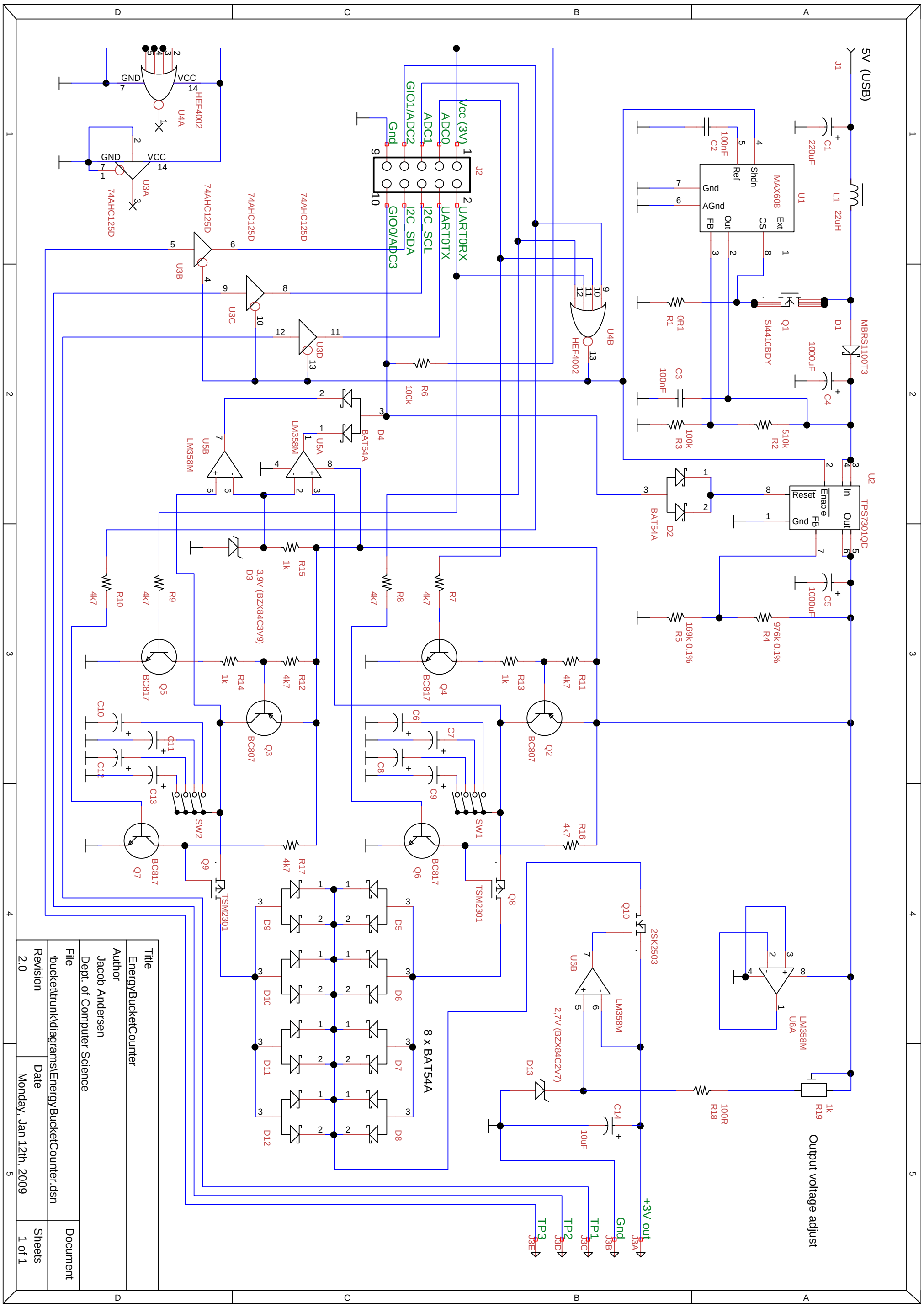


Figure C.5: Schematic of the Energy Bucket.

software as explained in section C.4. The circuit board was designed to fit 4 capacitors in each circuit path, and includes switches (SW1 and SW2) to individually select the capacitor(s) to use. Different capacitance may be chosen to accommodate different requirements for maximum current or temporal resolution (cf. the discussion in section C.2.6). A number of parallel schottky diodes are used to combine the currents from the two paths with minimal forward voltage drop.

C.2.3 3.9 V Voltage Reference and Comparators

The 3.9 V reference is realised by D3 and R15 and is fed to the two comparators in U5. When the voltage over either bucket capacitor runs below 3.9 V the corresponding comparator output goes low, causing an interrupt at the Tmote micro-controller (MCU).

C.2.4 Output Voltage Regulator

In the first version of the Energy Bucket, a standard low dropout voltage regulator was used to regulate the output voltage to 3.0 V. The energy consumption of this regulator turned out to be relatively high (compared to a sleeping sensor mote) and difficult to predict, and since it would draw its power from the Energy Bucket output, its energy consumption would be included in the measured energy.

This was not an acceptable situation, so a second version was designed with a custom built voltage regulator (Q10, U6B, D13 and R19). This voltage regulator does not disrupt the energy measurement, however, as reported in section C.2.7, it becomes slightly unstable with rising current.

C.2.5 Status Signals from Target Circuit

The three status signals from the target circuit are forwarded to 3 digital inputs on the Tmote through buffer gates in U3.

C.2.6 Capacitor Size

In the current implementation of the software, all pump stroke events are time-stamped by the Tmote and sent to the host PC using the serial USB interface. Each packet is 10 bytes long, and the maximum UART baud rate is 115,200 (with one start bit and one stop bit), which limits the Energy Bucket to 1152 pump strokes per second.

The Energy Bucket circuit was designed to deliver up to 150 mA, as the fingerprint reader (appendix B) may draw up to about 100 mA while the mote itself may draw up to about 50 mA. The "bucket size" should be as small as possible in order to get a good temporal resolution, and an upper bound on the capacity of the capacitors can be calculated as:

$$C = \frac{150 \text{ mA}}{1152 \text{ Hz } \cdot \ (8 \text{ V} - 3.9 \text{ V})} = 31.8 \ \mu\text{F} \tag{C.7}$$

For all the experiments presented in the following section, 33 μF capacitors was used, as this is the minimal capacity which is large enough to allow the Energy Bucket to operate over the entire current range. Smaller capacitors (10 μF) have been used for experiments which did not require high currents.

C.2.7 Calibration and Accuracy Experiments

In order to calibrate and test the accuracy of the device, a series of 49 measurements were performed with different fixed combinations of 0.1 % tolerance resistors as "target circuits". With the output voltage adjusted to 3.0 V (using a regular low-cost multimeter) and resistances ranging from 20 Ω–30 MΩ, the corresponding constant currents ranged from 0.1 μA–150 mA, and since small differences between the capacitors and other components in the two circuit paths can be expected, separate calibrations were performed for each capacitor. For the 32 resistor combinations with resistance below 100 kΩ (corresponding to 30 μA and up), 100 pump strokes was recorded (50 for each capacitor), and for the remaining 17 combinations only 2 pump strokes (1 for each capacitor) was recorded due to the very long stroke durations (more than 20 minutes for the 30 MΩ resistor).

As the total charge is expected to be proportional to the number of pump strokes and the currents in the measurements are constant, the stroke frequency, f_I, corresponding to the current I should be proportional to the current. Due to component imperfections a small leakage current, I_0, can be expected, so the following linear relation was anticipated:

$$I = Q \cdot f_I - I_0 \tag{C.8}$$

The measurements, however, diverged systematically from equation (C.8), and a graph analysis in MATLAB revealed the following correlation:

$$I = Q \cdot f_I^e - I_0 \tag{C.9}$$

for e slightly less than one. Further experiments revealed that e depends on the capacitor used. Different capacitors (for instance from different vendors) with identical capacitance may exhibit significantly different e values.

In order to determine the Q, e and I_0 values for the two capacitors, all the measurements of currents between 1 μA–1 mA (30 out of the 49 measurements) were used to determine Q and e through linear regression (using

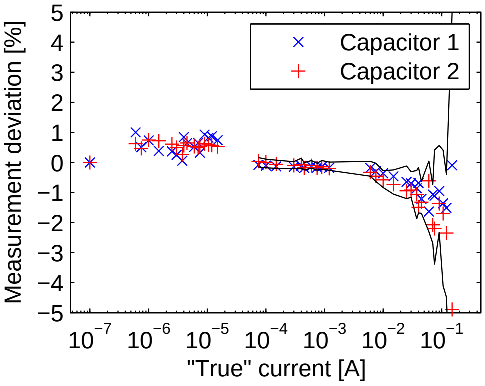


Figure C.6: The Energy Bucket calibration.

MATLAB). I_0 could then be determined using the 30 MΩ measurement. The result for a pair of 33 μF capacitors was found to be:

$$\begin{aligned} I_1 &= 144.7\ \mu C \ \cdot \ f^{0.9897} + 17.3\ nA \\ I_2 &= 146.7\ \mu C \ \cdot \ f^{0.9892} + 17.8\ nA \end{aligned} \tag{C.10}$$

In order to evaluate the calibration, all 49 measurements (including the 18 measurements not used for the calibration) were compared to the "true" current on the plot in figure C.6. The "true" current in this case is the current which is supposed to be drawn by the resistor, i.e.

$$I_{true} = \frac{3.0\ \text{V}}{R} \tag{C.11}$$

and the plot on figure C.6 shows

$$\frac{I_j - I_{true}}{I_{true}} \qquad \text{for } j \in \{1, 2\} \tag{C.12}$$

As 100 pump strokes were recorded for measurements from 30 μA and up, the × and + marks on the figure for these measurements show the *average* of all 50 strokes for each capacitor, while solid lines mark the minimal and maximal values—i.e. *all* 100 recorded pump strokes fall between these lines. Notice that the 30 MΩ measurement deviates 0 % because this measurement was used in the calibration to fix I_0.

Studying the plot it seems that the accuracy begins to drop at about 20 mA and decreases as the current grows. However, the reason for this behaviour does not necessarily reflect a decrease in the accuracy of current/charge measurements by the Energy Bucket; rather, it is a consequence of a problem with the reference I_{true}. Recall from equation (C.11) that this value depends on the assumption that the voltage over the resistor is a constant 3.0 V. Measuring this voltage reveals increasing un-stability and a

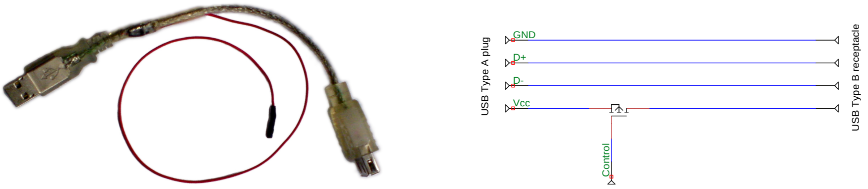


Figure C.7: The USB extension cord with power switch.

consistent voltage drop at higher currents. The voltage drops to 2.8 V at 150 mA, which equals a 6 % drop in the *real* I value compared to the I_{true} value used in the calculations above. This combined with the general unstability of the voltage regulator explains the behaviour found at the right end of the plot.

The experiments show, that the accuracy of the Energy Bucket is ±2 % or better over the 5-decade range 1 μA–100 mA for measuring *current* and *charge consumption*. The accuracy for *power* and *energy consumption* is almost as good, but drops a bit when the current is high (exceeds 50 mA).

Component tolerances—as high as 20 % for the bucket capacitors—suggest that environment (primarily ambient temperature) and ageing effects should be considered. All experiments so far have been carried out in an office under normal room temperature conditions. Over a period of several months the calibration was checked a few times, and is still checked occasionally. So far no discrepancies or ageing effects have been observed.

As the primary purpose of the tool is to do comparative studies (such as comparing the energy consumption of different implementations of a program), using the latest calibration will be sufficient most of the time.

C.3 The USB Extension Cable

A USB extension cable was modified by inserting a P-channel MOSFET transistor in the positive supply line. Schematic details and a picture of the modification are on figure C.7.

To turn on the power, the control signal must be well below 5 V (usually close to 0 V), and to turn off the power, the control signal must be at or above 5 V. As a minimum of 5 V is required and the Tmote operates at 3 V it cannot drive this transistor directly. Instead, the transistor gate is connected directly to the output of the 8 V voltage generator on the Energy Bucket board. The USB power must be turned off when energy measurements are performed, and this is also the time for the 8 V generator to be on.

Note that in order to prevent ground loops and other nasty electrical potential related problems when using the extension cable power switch with the Energy Bucket, the extension cable and Energy Bucket *must* be connected to the same PC or USB hub.

C.4 Energy Measurement Outline

The following paragraphs describes the energy measurement process step by step.

Initially, the four control lines, (ADC0, ADC1, ADC2, UART0RX) are at 0 V (logical 0). This causes all four transistors in the charge pump (Q2, Q3, Q8, Q9) to be off, so that no current can flow through the pump. Furthermore, the NOR gate U4B will keep the 8 V voltage generator off, which causes U2 to drive its $\overline{\texttt{Reset}}$ output to 0 V, which in turn fixes the interrupt line (GIO0) to a logical 0. If the USB extension cable is used, since the 8 V voltage generator is off, the power switch will be on, so the target circuit sensor mote may be programmed through the USB connection.

As an energy measurement is about to start, the Tmote MCU raises the ADC0 and UART0RX lines to logical 1 (3 V) and waits for the interrupt line (GIO0) to go high. This turns the two transistors Q2 and Q3 on to allow charging of both bucket capacitors, and it also causes the NOR gate U4B to set its output low, which will start up the 8 V generator.

At first, the voltage at U2's input will be 5 V as the boost converter (U1) has been off, but as U1 is starting up as well, the voltage will rise to about 9 V—as quickly as the boost converter can deliver the charge necessary to charge all the large capacitors. As soon as the voltage at U2's output exceeds 5 V, the USB extension chord will shut down the power switch, severing the direct connection between the host PC and the target system. When the voltage at U2's output is stable at 8.0 V (which implies that both bucket capacitors are charged and ready as well), U2 will release the $\overline{\texttt{Reset}}$ signal and the interrupt line (GIO0) goes high. This informs the Tmote MCU that the measurement can begin.

The MCU will start the energy measurement by lowering ADC0 and then raising ADC1. This will first stop charging the first bucket capacitor and then begin discharging it through the target circuit.

When the voltage over the first bucket capacitor gets below 3.9 V, the comparator U5A will detect this and drive the interrupt line (GIO0) low. The MCU receives this signal and responds with the following 4 steps:

Lower UART0RX This stops charging the second bucket capacitor

Raise ADC2 Begin discharging the second capacitor. Notice that the schottky diodes will cause the discharging of the first capacitor to stop at this point.

Lower ADC1

Raise ADC0 Starts charging the first bucket capacitor.

As soon as the first capacitor has been charged to at least 3.9 V, U5A will release the interrupt line, and the pump stroke is complete.

As the voltage over the second capacitor gets below 3.9 V, U5B detects this and causes an interrupt with the following response from the Tmote MCU: (Lower ADC0, Raise ADC1, Lower ADC2, Raise UART0RX).

The energy measurement is terminated by the Tmote MCU by lowering all four control lines (ADC0, ADC1, ADC2, UART0RX). This will stop the current from flowing through the charge pump, and due to the NOR gate U4B, the 8 V generator will shut down. Bleeding through various components (U5, U6, R4, R5, R15, R18, R19, D3, D13) will quickly discharge C5 to a voltage below 5 V, at which point the power switch in the USB extension cord will turn on, restoring the connection between the host PC and the target circuit.

Appendix D

Activity-Based Sensor Networks (ABSN)

This appendix deals with the use of data from medical sensor networks, in particular how to create easy remote access to the sensor readings with little or no configuration overhead.

This mismatch between applications designed for traditional personal computing (i.e. desk-work) and the clinical work at a hospital is the focal point of the *Activity-Based Computing* (ABC) project [7, 26, 15, 17, 23]. The transition from large computing centres and mainframe computers to personal computers and widespread use in offices and homes led to an operating system paradigm shift: While early operating systems (OS) were designed to manage data using applications and had an application-centered command line interface, later OSs, such as MacOS and Microsoft Windows, were designed for desk use (keyboard and mouse) and document management. Today, the emergence of ubiquitous computing calls for another OS paradigm shift. The use of computing devices are no longer restricted to a desk. Instead, we use a growing and diverse set of IT devices and "gadgets" for a wide variety of tasks that does not necessarily fall in the category of "document management". For instance, several people may collaborate on a single task and use many devices or tools in order to reach their goal. The ABC project is an attempt to create an OS support for this kind of collaboration and cross-platform work.

In the current version of the ABC framework, a "Computational Activity" (CA) is the fundamental entity around which everything else revolves. A CA models one real-world activity (RWA) of a person or a group of people. If a group of people share a CA, a number of tools supporting cooperative work are available [15] and the ABC framework ensures that the states of documents and data associated with this CA are kept consistent. A person can be engaged in only one CA at any point in time (assuming he is using only one computer), so if he is in fact multitasking between several RWAs, he will have to swap between the CAs.

The development toward IT becoming ubiquitously available is extremely useful in the clinical world—and especially hospitals, where the office use of computers is a disturbing factor in the work-flow [16] (cf. figure 7.1) and collaboration between clinicians is a fundamental part of all tasks. Hospital routines and use scenarios have thus been studied thoroughly as a significant part of the ABC background research.

The fact that ABC provides an awareness—at the OS level—of who the user is and what the user is currently doing, could be used to automatically judge the relevance of different sensor data, and when, where and how to present them.

D.1 Sensor Data Use Scenarios

To illustrate the potential advantages of using the ABC framework and the context awareness provided by ABC, rather than a traditional (not context aware) application, three scenarios to be used in the following discussion are presented below:

D.1.1 Train Accident

Recalling the train accident scenario from section 2.2.3, different clinicians would be using the sensor readings in different ways. Some would be using the readings to treat individuals while others would use the readings to monitor a larger group of patients, for instance performing triage or allocating and prioritising resources. Furthermore, sensor data for each individual patient can be forwarded to the relevant hospital, as soon as it is decided, which hospital the patient will be transferred to.

D.1.2 Morning Conference

At the morning conference at a hospital ward, doctors and nurses may discuss the schedule for the day, and the progress of each individual patient. Relevant sensor data could be aggregated and displayed at a wall display in the conference room. For instance, for post-surgical patients, it is relevant to monitor the development of the body temperature, as increasing temperature could indicate infections, thus demanding special attention throughout the day.

D.1.3 Ward Round

At the ward rounds, a doctor and a nurse visit each patient. Relevant sensor readings may be brought up on a PDA, tablet PC or a display mounted on or beside the bed. Of course only data regarding the current patient should be displayed to avoid confusion and mix-ups.

D.2 Proposal: Sensors as CA Participants

The goal of this work will be to exploit the ABC Operating System to eliminate the need for configurations in the previous examples. This can be achieved by inviting the group of sensors on a patient, Mr. Hansen, to join the CA of "Monitoring Mr. Hansen" as participants. Other participants of this CA could be the clinicians authorised to treat Mr. Hansen, who will then get access to the readings through the collaboration features of the ABC framework. The sensors will carry a block of meta information about the patient which could easily contain information about the monitoring CA, causing the sensor to be aware of the CA it is participating in—hence the name Activity-Based Sensor Network (ABSN).

A significant advantage from this approach would be, that moving patients from one place to another could be done without reconfiguring the sensors, as this would be handled entirely by the ABC framework. For instance in the train accident example above, sensors come to life at the accident scene with no knowledge of which hospital they will end up going to. Based on the context awareness found in the ABC framework together with knowledge about the affiliations of the participating clinicians, the ABC framework will figure out how to distribute the sensor readings. Initially, the sensors would form an ad-hoc network, and this would probably automatically change into a client-server structure as the patient is moved into a hospital in line with the hybrid architecture of the ABC framework [23].

D.3 Discussion: The Current ABC Paradigm and its Limitations

A notable drawback of the ABC framework in its present form, is that every participant of a CA share the exact same view. Browsing through the use scenarios above, it should be clear that this is not what we want. In the train accident example, some clinicians need details about one particular patient, while others (like the medical coordinator) need to get an overview of a large group of patients. Likewise, there will also be different needs at the morning conference compared to the ward round.

The root cause of this drawback is the current ABC conceptual model. As each CA is an independent entity unrelated to other CAs, there is no obvious way to handle an activity of monitoring all patients as a simple combination of all the activities of monitoring each individual patient. This problem can easily be generalised, as RWAs are often logically linked together. For instance, if a RWA is "Treating Mr. Hansen's stomach cancer" another one could be "Feeding Mr. Hansen". These two RWAs are different, though clearly related somehow—in particular if Mr. Hansen requires a certain diet. Yet another RWA could be "Medicating Mr. Hansen"—obviously

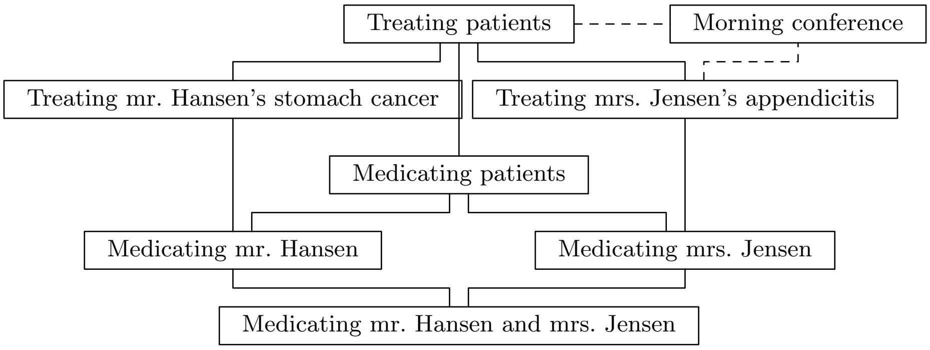

Treating patients
Morning conference
Treating mr. Hansen's stomach cancer
Treating mrs. Jensen's appendicitis
Medicating patients
Medicating mr. Hansen
Medicating mrs. Jensen
Medicating mr. Hansen and mrs. Jensen

Figure D.1: Relations between real-world activities.

again related to the first RWA. But what if Mrs. Jensen in the bed beside Mr. Hansen gets her medication at the same time? Perhaps the nurse then prepares the two patients' medication at the same time. In that case a RWA would be "Medicating Mr. Hansen and Mrs. Jensen" and this RWA would be related to both "Treating Mr. Hansen's stomach cancer" and "Treating Mrs. Jensen's appendicitis" (cf. figure D.1). There is no way to model such inter-dependencies in the current version of the ABC framework.

Looking at the example scenarios given above, similar patterns can be found. For instance, the morning conference is a long lasting repetitive RWA. It takes place every morning and involves most of the clinicians working that day as participants. On the other hand, the morning conference is all about planning the treatment of the currently admitted patients, so the morning conference RWA is clearly related to the RWAs of treating the individual patients—but only for as long as they are admitted to the ward.

D.4 An ABC Next Generation Conceptual Framework Early Proposal

During the first two years of this PhD project a significant effort was done towards re-thinking the concepts of ABC in order to ease the tasks of configuring sensors and managing sensor data in a hospital (ABC-based) infrastructure.

Unfortunately, this work was suspended before any of the ideas could be tested, so the task of building a prototype to demonstrate this proposal remains future work. The basic idea of this proposal is to incorporate relations between CAs corresponding to the relations found between RWAs, and the remainder of this section is a brief presentation of the thoughts on this subject so far.

As the traditional document-oriented view is supposed to be replaced by activity-orientation, documents and data should be tied to activities.

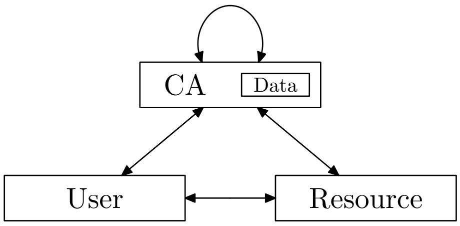


Figure D.2: Activities-Users-Resources model. Arrows demonstrate many-to-many relations.

This change can be compared to the earlier mentioned transition from application-oriented to document-oriented operating systems (OS). In the former type of OS, the user had to launch the specific application, he wanted to use, whereas in the latter type of OS, the user merely opens the document, and the OS has to figure out, which application to use and how to start it.

In a complex ubiquitous computing setting, the user may be accessing his documents and data from a number of different devices: Phones, PDAs, different workstations etc. Now it turns out to be essential that the user should not be expected to keep track of, where his data/documents are physically located. Instead the activity-based OS should take over. Therefore, a triangle (figure D.2) is proposed, with the following corners:

Computational Activity A CA is a container of data and documents. The user should not have to be aware of the physical location of his data – on which disc it was stored. Instead the OS should create the illusion that data is logically "stored" inside the CA. Also the CAs are the base of collaboration. Whenever multiple users have access to the same CA, they will share any documents or data stored inside this CA and the OS will automatically track changes, provide versioning, resolve conflicts etc. Furthermore, a CA can be associated to other CAs as explained below.

User A user can be a participant in a CA. This gives him access to read and write the data contained by the CA.

Resource A CA exist in the "cyberspace", while most users exist in the physical world, hence some kind of bridge is necessary. A user manipulates his CAs through available resources, which may include: hardware (PDA, workstation, wall display), software (browser, word processor), service (printer, scanner).

The 4 relations on figure D.2 are all many-to-many. In particular, a CA can be associated to any number of CAs!

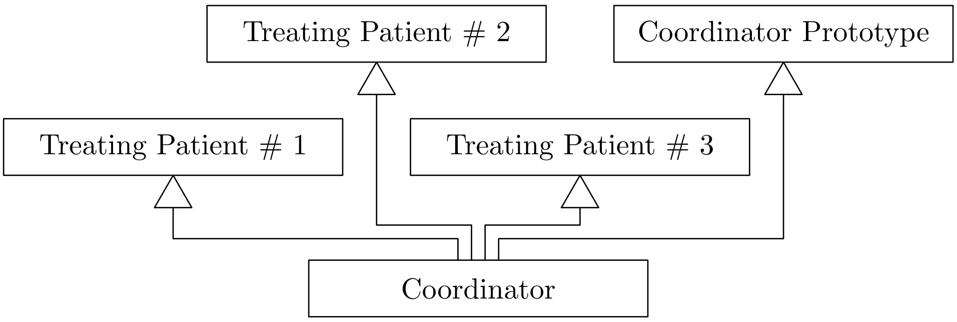


Figure D.3: Accident scenario revisited.

The associations between CAs need a semantics. The ideas have been developed based on different drawings like figure D.1 of relations between RWAs and inspired by prototype-based object-oriented programming (as found in e.g. the Self programming language [108]), and the modelling concepts of mixins and traits [95] found in recent OO languages like Scala [81].

One potential future work path would be to explore, whether it makes sense to adapt these concepts to the world of activities. For instance (cf. figure D.1) the morning conference CA of June 2nd could have multiple parent CAs (like traits or mixins): all patient treating CAs of patients admitted to the ward on June 2nd would be in a parent relation to the morning conference of that day, and consequently all data about the patients (stored in the patient treatment CAs), including sensor readings, would be accessible and subject to collaboration if other users (e.g. a radiologist) are accessing one of the CAs simultaneously. Furthermore, a prototype morning conference CA would also be a parent of the June 2nd morning conference CA. This prototype morning conference CA could contain some filters and preferences that would define how data from all the patient treatment CAs should be aggregated and displayed on the wall display in the conference room.

Figure D.3 shows how this could be used in the emergency scenario. Again a prototype CA is used. The medical coordinator participates in the treatment of all patients through a single CA, which by multiple inheritance is the child of every patient treatment CA, providing access to all sensor readings. Meanwhile, the medical coordinator prototype CA provides a better view—for instance by reducing the amount of details and emphasising critical data.