UNIVERSITÄT KOBLENZ · LANDAU
Faculty of Computer Science

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of IT

Enhancing the ReMoteCare prototype by adding an SNMP proxy and video surveillance

Diploma Thesis
in Computer Science

written by:
Martin Fischer

Supervisor:
A/Prof. Dr. Elaine Lawrence, Dept. of Computer Systems, University of Techn., Sydney
Evaluator:
Prof. Dr. J. Felix Hampe, Institute for IS Research, University of Koblenz

Sydney, 31st March 2008

Statement of Authentication

I declare that this thesis is my own work and was written without literature other than the sources indicated in the bibliography. Information used from the published or unpublished work of others has been acknowledged in the text and has been explicitly referred to in the given list of references. This thesis has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education.

. . . . . .

(Place, Date) (Signature)

Acknowledgments

I would like to thank A/Prof. Dr. Elaine Lawrence and Prof. Dr. J. Felix Hampe not just for their help and assistance but also for giving me the wonderful opportunity to write this thesis in Sydney at the University of Technology. A word of thanks also goes to the team at UTS and Ulm University, namly Karla Navarro, Brian Lim, Leena Ganguli, Tommy , A/Prof. Donald Martin, Dr. Peter Leijdekkers and Dr. Frank Kargl for the great cooperation and help during my stay.
Special thanks go to my parents for all their support during my studies.
Last but not least I have to thank Rich and all friends of the *LockUndKeller* for showing me Australia and for giving me a great time in Sydney.

Publications

The following is a list of refereed publications resulting from this thesis:

- Yen Yang Lim, Marco Messina, Elaine Lawrence, Frank Kargl, Leena Ganguli, and **Martin Fischer**. Snmp-proxy for wireless sensor network. In *5th International Conference on IT: New Generations*. ITNG 2008, April 2008.

- Marco Messina, Yen Yang Lim, Frank Karl, Elaine Lawrence, and Don Martin. Implementing and validating an environmental and health monitoring system. In *5th International Conference on IT: New Generations*. ITNG 2008, Las Vegas, USA, 2008.

- **Martin Fischer**, Ying Yang Lim, Elaine Lawrence, and Leena Ganguli. ReMoteCare: Health Monitoring with Streaming Video. In *7th International Conference on Mobile Business*, Barcelona, Spain, 2008. under review.

- Frank Kargl, Elaine Lawrence, Ying Yang Lim, and **Martin Fischer**. Security and Privacy in Pervasive eHealth Monitoring Systems. In *7th International Conference on Mobile Business*, Barcelona, Spain, 2008. under review.

Contents

List of Figures

List of Tables

Listings

Chapter 1.

Introduction

Computers and especially computer networks have become an important part of our everyday life. Almost every device we use is equipped with a computer or microcontroller. Recent technology has even boosted this development by miniaturization of the size of microcontrollers. These are used to either process or collect data. Miniature senors may sense and collect huge amounts of information coming from nature, either from environment or from our own bodies [MLNL07]. To process and distribute the data of these sensors, wireless sensor networks (WSN) have been developed in the last couple of years. Several microcontrollers are connected over a wireless connection and are able to collect, transmit and process data for various applications. Today, there are several WSN applications available, such as environment monitoring [MRHO06], rescue operations [MMSS03], habitat monitoring [SOP+04] and smart home applications [Fis06].
The research group of Prof. Elaine Lawrence at the University of Technology, Sydney (UTS) is focusing on mobile health care with WSN [MLNL07]. Small sensors are used to collect vital data. This data is sent over the network to be processed at a central device such as computer, laptop or handheld device. The research group has developed several prototypes of mobile health care [MLNL07] [LGL06] [LLN06]. This thesis will deal with enhancing and improving the latest prototype based on CodeBlue, a hardware and software framework for medical care [Sch06].

1.1. Motivation and Background

In almost every part of the western world there has been an increase of the aging population due to sustained low fertility and increased life expectancy. In Australia, there has been an 18% increase of the over 65s and a 26% increase in 45-64s over the last 20 years [Aus04]. The demand for hospitals, nursing homes and also the expenses for healthcare will rise dramatically. Additionally, seniors who want to live independently in their own homes will also need monitoring in order to provide adequate help in case of emergencies.

[MLNL07] have developed a health care monitoring application based on WSNs. This system could be a possible solution to manage the care of the next generation of older people. But it is not just suitable for monitoring seniors and patients in hospitals, it can also be used for emergency care and disaster response. The system is based on CodeBlue [Sch06], a hardware and software infrastructure developed at Harvard University. Small sensors are used to collect data and form a network to transmit the collected data. The CodeBlue framework will be described in more detail in chapter 2.
Due to its flexibility there are several possible scenarios for CodeBlue:

- ***Emergency site***. In the case of a mass casualty incident (MCI) with a lot of victims it is essential to determine the patients that are in a critical situation and give them instant treatment. This process of determining the priority of patients is called *triage* [NK07]. During the triage the severity of all patients' injuries is estimated. With CodeBlue, the medical professionals at the site can equip the patients with mote sensors. The network will form itself in an ad-hoc manner and start transmitting vital data to the CodeBlue appplication. With this application medical staff can see the condition of many patients at a glance and make decisions based on this data.

- ***Hospital scenario***. It is essential for patients to be monitored in a hospital, especially on a intensive care unit (ICU) where patients have to be under surveillance 24/7. The CodeBlue sensor network could be used to accomplish this surveillance. Since the network is wireless, the patients would even be able to move within the hospital.

- ***Elderly care***. The elderly care scenario is similar to the hospital scenario. The prototype enables the monitored patients to live independently. Their vital data is collected while they live their regular everyday life. As [SFGB07] state, a lot of systems that improve the independency of elderly people lack communication with the community outside the house. CodeBlue is able to send vital data to the outside world and give the patients a lot of independence at the same time.

Current procedures in hospitals and health care are primarily based on paper-based communications [NK07], but current developments show that new systems could improve these procedures. Especially in triage processes and monitoring patients, computers could help the medical staff significantly. Several prototypical triage programs have been developed.
[DBM+05] have presented a novel computer triage program which has achieved remarkable results when compared to standard triage. Their system is meant for a clinical environment and is based on stationary devices. Another system for hospitals was developed by [SH06] and is called MobileWard. It is a is context-aware system and is able to discover and react autonomously according to changes in the environment.

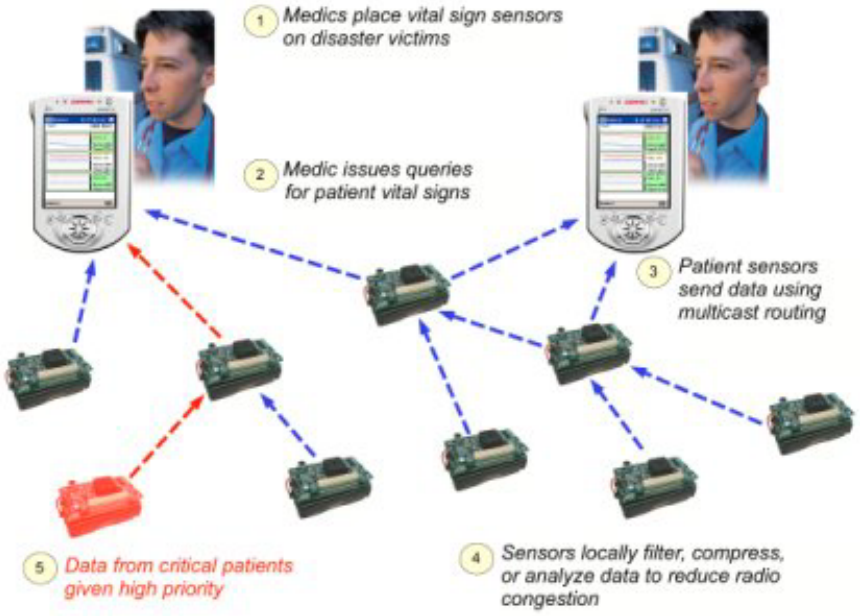

Figure 1.1.: CodeBlue architecture for emergency response. [Sch06]

A system focusing on emergency sites was developed by [NK07]. Their focus was on creating a prototype with hand held devices that assists paramedics on mass casualty incidents (MCIs).

All these systems have been developed to assist medical professionals in numerous situations. Some focus on hospital environment while others focus on emergency sites. CodeBlue is able to deal with different kind of scenarios. It is suitable for hospital scenarios as well as emergency sites.

Three key elements are essential for computer aided health care solutions: access, quality and value. The access includes data access from anywhere, any time and any how. The quality issues include offering high level of patient care by establishing integrated information repositories. The value includes provision for effective and efficient healthcare delivery.

Wireless sensor networks could provide all these three elements: access over gateways to other networks, quality by gathering and processing medical information and value by speeding up processes in medical environments. In addition to that the patient will also benefit from those networks, for example by gaining freedom to walk around in a hospital or home while monitored over a wireless network.

1.2. Wireless Sensor Networks

Recent technology advances include integration and miniaturization of physical sensors, embedded microcontrollers and radio interfaces on a single chip [MLNL07] and the use of the

802.15.4/ZigBee protocol to communicate. These sensors with radio interface are called *motes*. They were developed at the University of California, Berkeley, and are resold by Crossbow Technology Inc[1]. They are able organize themselves into a wireless network, sharing data with one another and with computers. This feature makes it very easy to set up a network very quickly. Especially on sites with no infrastructure, motes can be used to build an ad hoc network. An ad hoc network is a group of wireless mobile nodes which self organize into a network in order to communicate. These networks can be utilized for almost any kind of usage. The main focus of recent studies is patient monitoring with WSN. Already, a number of commercial and research groups are involved in developing tools or systems for patient monitoring based on particular WSNs.
Besides their size, one of the advantages of motes is the power consumption. They are powered by battery and can run up to one week continuously [MFJWM04].

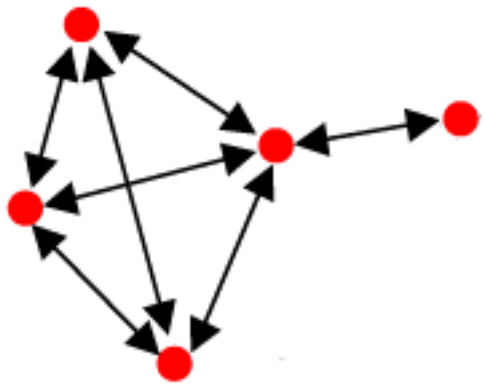

Figure 1.2.: A simple mesh topology. [Fis06]

A common topology of WSNs is a *mesh network*. In a mesh network as shown in figure 1.2 every node in the network has several connections to other nodes and has its own routing table to be able to forward packets. If one node is removed from the network the data packets still will reach its destination by taking another route through the network. This makes WSNs less vulnerable to network failures due to a damaged or offline mote.
This is one of the reasons why WSNs are used in extreme environments and situations. In a mass casualty incident for example the unambiguous mapping between the patients and their records is crucial for the functionality of the system. This is not a trivial task in disaster scenarios, since patients and paramedics move about very dynamically [NK07]. A mesh network of motes can easily adapt to changing condition and assure the transmission of vital data.

[1] `http://www.xbow.com`

There are other types of topologies such as star or cluster tree topology [Fis06] although they are not very common. Since the motes have enough hardware resources, a mesh topology is easy to set up and very reliable.

The type of mote used for the prototype based on CodeBlue is the MicaZ mote (see figure 1.3). Besides that, CodeBlue also supports Mica2 and Telos motes [MFJWM04]. MicaZ motes have a IEEE 802.15.4 compliant RF transceiver and are able to transmit data at a rate of 250 kbps with an approximate range of 100 meters [Cro07a]. MicaZ motes are based on the Atmel Atmega128L microcontroller. The devices have a small amount of memory (128 KB) and can be programmed (using the TinyOS operating system) to sample, transmit, filter, or process vital sign data [Sch06].

As you can see in table 1.1, the power consumption of the MicaZ mote is significantly lower than the Mica2 mote when transmitting data. Since transmitting data over the radio is a main application of the motes, the usage of MicaZ motes increases battery life significantly.

Figure 1.3.: MicaZ mote without sensor board [Cro07a]

Operating Current (mA)	**MicaZ**	**Mica2**
ATMega 128L, full operation	12 (7,37MHz)	12 (7,37MHz)
ATMega 128L, sleep	0,010	0,010
Radio, receive	19,7	10
Radio, transmit (1mW power)	17	27
Radio, sleep	0,001	0,001

Table 1.1.: Current draw of Mica2 and MicaZ [Cro07a]

1.3. Objective Of This Thesis

The research group at the University of Technology has enhanced the CodeBlue framework to a new prototype called *ReMoteCare*. They developed a SNMP-Proxy which reads log-files generated by ReMoteCare and uses this data to send SNMP messages into the local network. Right now this proxy is a stand-alone application and two steps are required to send the medical data via SNMP. First, the ReMoteCare application writes all medical data to a log file. In the second step the SNMP-proxy reads this log file and provides the data via SNMP to the network. This was developed as a proof of concept to show the ability and the benefits of sending medical data over SNMP.

The first objective of this work is to integrate the SNMP-proxy into ReMoteCare and generate SNMP-messages in real-time. By doing this patients can be monitored not just on the ReMoteCare GUI (see 2.3) but also over the local area network or internet with any kind of SNMP management tool.

The second goal of this work is to integrate a webcam into the ReMoteCare system, as, especially when monitoring elderly people and in hospitals, not just the medical data is important. It is also very useful to have a visual surveillance of the persons monitored. Medical staff can verify the status of a patient visually and may gain valuable life saving time to make appropriate decisions.

1.4. Prototyping

Prototyping is a common methodology in system development. In contrast to a linear development process, prototyping introduces a cyclic procedure of developing, reviewing and modifying the prototype [Sch99]. Prototyping can be a powerful tool in assisting the requirements and specification process. However prototypes are rarely developed because they are considered too expensive [GS81]. Especially in *Throwaway* or *Rapid Prototyping* where a prototype will eventually be discarded rather than becoming part of the finally delivered system, the costs can increase enormously. Using expensive components such as hardware for wireless sensor networks or medical equipment adds even more costs to the development process since different prototypes might need different types or versions of hardware.

A prototype is often used as a proof-of-concept. New features or technologies are added to the prototype to show the feasibility of the system. In addition to that, prototypes can be used to get a feedback from the future users already during the developing process. This could be very useful for medical systems. Health care providers, specifically physicians, have certain common work environments and behaviors across different regions and specialties. Similar to other experienced professionals, they are uncomfortable when placed in a position

of (information technology) novice [GR97]. During the process of prototyping doctors could have influence on the developed system while staying in a familiar environment.
As described in 1.3, this work will enhance the prototype currently developed at the faculty of IT at UTS. Building prototypes for ubiquitous computing is still challenging. Two main challenges, which have to be faced in developing such systems, are costs and time.
In various projects it can be seen that a significant part of the research effort (in terms of money and man-power) is spent on the framework in which the actual research is to be carried out [KS06]. To overcome this problem the research group at UTS follows the prototyping approaches described in [KS06]. According to that there are two approaches to prototyping (besides developing a prototype from scratch):

1. **Restriction and Repackaging** Physically repacking existing devices and hence restricting the set of interactive features (e.g. the buttons or screen space) is the first approach. For example, if a simple input device with a touch screen is needed that is easy to operate for elderly people and does not overwhelm them with too many buttons and switches, a Compaq IPAQ could be provided with a different housing to make it look like a completely different device. The screen could be restricted and only visible through a round hole in the new housing. This would save time in the developing process since the system does not have to be developed from scratch and is already available [PRS+03].

2. **Modification and Extension** This approach also starts with already developed products or technology. This product is then modified to add more or other functionality. For example the buttons on a wireless mouse can be replaced by some other kind of switch so that disabled people are able to operate the button with their feet. This is much easier than developing a new input device for disabled people from scratch.

According to [KS06] prototypes derived from standard commercially available components should be used to develop prototype systems. They argue that this will even help to ease the development process and decrease the time until market launch. At the same time the use of off-the-shelf products will reduce the costs for the prototypes.
Since this approach is quite reasonable, this work follows these paradigms. First of all, the ReMoteCare system is not built from scratch. A significant part is already integrated in the CodeBlue framework now. Additionally, all the enhancements to the ReMoteCare prototype are based on standard systems. These systems are changed or configured to fit the demands of this work.
Furthermore this work will follow the prototyping approach of modification and extension mentioned above.

In contrast to developing new technologies and creating new inventions from scratch, this work will try to use integrate already developed technology into the ReMoreCare prototype. Future prototypes will be developed based on the current ReMoteCare prototype using *Evolutionary Prototyping* rather than using the already mentioned Throwaway Prototyping.

1.5. Structure Of This Thesis

The rest of this thesis is structured as follows:
Chapter 2 describes the CodeBlue framework on which the ReMoteCare prototype is based. It outlines the general architecture as well as the hardware and software portion of the system.
Chapter 3 deals with the implementation of the management proxy. At first, medical monitoring systems are compared to network management tools. With that in mind the chapter gives a brief overview of suitable protocols. After that, the implementation itself is described in detail.
The second goal of this thesis, the integration of a webcam, is explained in chapter 4. After a short discussion about video surveillance, two approaches are discussed in this chapter.
Chapter 5 describes various tests and experiments that were run on the system to prove the functionality of the ReMoteCare prototype. Finally, chapter 6 discusses future work and concludes this thesis.

Chapter 2.

The CodeBlue Framework

The ReMoteCare prototype is based on CodeBlue. CodeBlue is a scalable hardware and software infrastructure for medical health care developed at Harvard University [Sch06]. They developed a range of medical sensors integrated with the commonly-used Mica2, MicaZ and Telos mote designs. These include a two-lead electrocardiogram (ECG) and a pulse oximeter. Additionally, the mica sensor board with environment sensors such as for light, temperature and sound was also integrated in CodeBlue. This chapter will give an overview of the CodeBlue framework. This includes the architecture, the wireless sensor network implemented in TinyOS and the graphical user interface that runs on a computer or laptop.
As described in 1.1 ReMoteCare integrated CodeBlue which can be used in different kind of scenarios. Figure 2.1 shows the scenario of an elderly care facility where ReMoteCare is used to monitor the inhabitants. In addition to the stationary ReMoteCare GUI, the medical staff can use hand held devices such as PDAs to connect (e.g. over a wi-fi connection) to the ReMoteCare system.

2.1. Architecture

The CodeBlue software framework consists of two main parts. The first one is the wireless sensor network. This part is implemented in *TinyOS* [SCL+05]. TinyOS is a operating system developed for wireless embedded sensor networks. The programming language used with TinyOS is NesC, a C dialect. TinyOS is event driven and has a component based architecture. This makes it easy to develop new application by simply wiring several already developed components together. The software portion of CodeBlue will be described in section 2.2 in more detail.
The second part is the CodeBlue graphical user interface (GUI) which is implemented in Java. The GUI is used as a interface to interact with the sensor network. The user can select patients and see their data on the display. The Java application will be described in section 2.3.

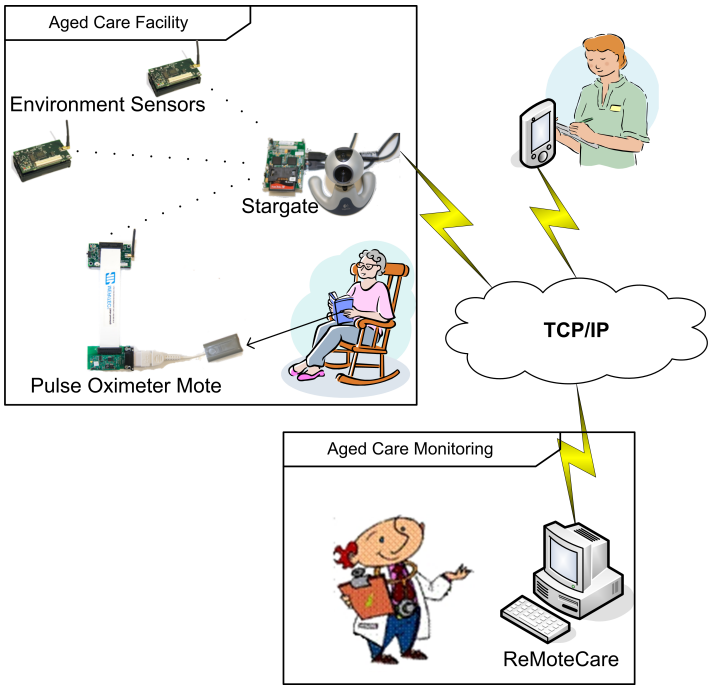

Figure 2.1.: ReMoteCare in elderly health monitoring [FLLG08]

2.1.1. Channel Architecture

CodeBlue is intended to act as an 'information plane' tying together a wide range of wireless devices used in medical settings. It uses a publish/subscribe architecture. The sensors publish their data into channels. Users can subscribe to that channel to receive the sensor data. This architecture minimizes the amount of traffic in the network because sensor data has to be transmitted to a channel just once and multiple users can subscribe to that channel [SCL+05]. Users subscribe to a channel by using the CodeBlue Query (CBQ) layer. Queries are sent to that layer from the graphical user interface. The query structure is described in more detail in section 2.3.

Although the publish/subscribe communication decouples the concerns of devices generating data from those receiving and processing it, several considerations have to be taken into account:

- Due to a limited bandwith on the radio channel, sensors should publish data at a reasonable rate.
- Some form of multihop routing is necessary when sensor devices are out of range (see section 2.2.1).
- Route paths have to be dynamic since sensor devices are able to move.

These considerations lead to a routing protocol that is able to adapt to changing conditions in the network.

2.1.2. Routing in CodeBlue

The two major challenges for routing in a wireless sensor network are the hardware and bandwith constraints and the constantly changing network conditions. Hardware and power resources on a mote are very limited. So, the chosen routing protocol has to work very efficiently. On the other hand, the routing protocol has to adapt to node failures, nodes out of range and similar event very quickly to ensure that the medical data reaches its destination as soon as possible. CodeBlue is able to route information through the network with the help of the Adaptive Demand-Driven Multicast Routing (ADMR) protocol [JJ01]. Chapter 2.2 gives a more detailed description of the routing protocol.

2.1.3. Security

Especially when dealing with medical data, security and privacy play an important role. Right now there are no security features implemented in CodeBlue. The data is sent over the network in clear and anybody connected to the network can subscribe to a data channel and read the patients data. Several research groups are working to resolve that issue. The major focus is on encryption and authentication within the wireless network.
The already mentioned resource constraints affect the security considerations as well. Table 2.1 summarizes the three major points of security. In conventional networks, message

Method	**Comment**
Key Establishment	Traditional method will not be able to scale up adequately with hundreds of notes.
Trust setup	Necessary to establish a secure and efficient key distribution for large scale networks.
Privacy and authentication	Research has indicated that software only cryptography is practical with motes.

Table 2.1.: Security considerations [LLN06]

authenticity, integrity, and confidentiality are usually achieved by an end-to-end security mechanism such as SSH [Ylo96], SSL [Ope], or IPSec [CGHK97] because the dominant traffic pattern is end-to-end communication; intermediate routers only need to view message headers and it is neither necessary nor desirable for them to have access to message bodies. This is not the case in sensor networks. The dominant traffic pattern in sensor networks

is many-to-one, with many sensor nodes communicating sensor readings or network events over a multihop topology to a central base station [KSW04].
TinySec is a approach developed by [KSW04] to implement security on the link layer. TinySec encrypts the data payload and authenticates the packet with a message authentication code. The message authentication code is computed over the encrypted data and the packet header. In authentication only mode, TinySec authenticates the entire packet with a message authentication code, but the data payload is not encrypted. The authors found RC5 and Skipjack to be most appropriate for software implementation on embedded microcontrollers [KSW04].
Other security approaches use elliptic curve cryptography (ECC) to develop efficient encryption algorithms, such as [LKN07] and [GMF+05]. ECC is based on the mathematical problem to solve discrete logarithms. ECC can offer equivalent security with substantially smaller key size. a 160-bit ECC key provides the same level of security as a 1024-bit RSA key and 224-bit ECC is equivalent to 2048-bit RSA. Smaller keys result in faster computations, lower power consumption as well as memory and bandwidth savings making ECC especially appealing for resource-constrained environments. More importantly, the performance advantage of ECC over RSA increases as security needs increase over time [GMF+05].
As outlined several security approaches are currently developed and the ECC seems to be promising. However, none of those technologies have been implemented into CodeBlue and the UTS team aims to follow this up in their next prototype.

2.2. The Wireless Sensor Network

The wireless sensor network of CodeBlue might consists of MICA2, MICAz or Teleos motes. In the ReMoteCare system, just MICAz motes are used. Every mote has a NesC application installed. This application is in charge of collecting data, transmitting data and routing packages through the network.
Two mechanisms are essential for the wireless network: routing and discovery. These processes are described in the following sections.

2.2.1. Adaptive Demand-Driven Multicast Routing

The routing plays a key role in sensor networks. In the medical scenario both patients and caregivers are expected to be mobile. Many patients may be ambulatory and free to roam about the hospital ward. Even those confined to hospital beds may be transferred between wards or temporarily moved for surgery or imaging. On an emergency site, the mobility of patients and medical staff is also essential.

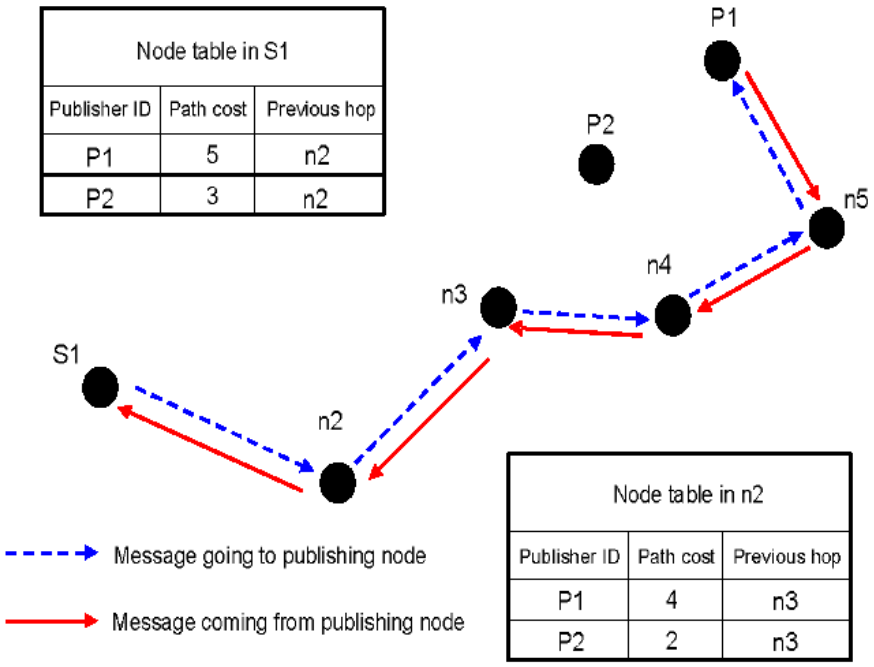

Figure 2.2.: A simple ADMR routing example. [KKG07]

The CodeBlue routing layer is based on the Adaptive Demand- Driven Multicast Routing (ADMR) protocol [JJ01]. The authors of CodeBlue chose ADMR because it is simple and has been extensively studied in simulation [SCL+05]. ADMR has been developed in order to reduce unnecessary traffic within the network and therefore increase the battery life of the devices. When a mote requests to publish data, a route discovery process is initiated [SCL+05]. Every CodeBlue node maintains a node table indexed by the publisher node ID. Each node table entry contains the path cost from the publisher to the current node, as well as the previous hop in the best path from the publisher. The route discovery process maintains the best paths from publishers to subscribers by periodically propagating a controlled broadcast flood that updates the node tables on all intermediate nodes. Figure 2.2 depicts the path construction procedure between publishing node P1 and node S1. At first, S1 sends a registration message to P1 through intermediate nodes n2 to n5 which become forwarders. Upon reception P1 will transmit the corresponding data to S1 through the forwarders [KKG07].

2.2.2. Discovery

In order to build selforganizing networks, CodeBlue has to automatically detect new nodes in the network. To discover new motes, CodeBlue uses a simple discovery protocol. Each mote periodically broadcasts metadata into a channel. This metadata includes mote ID and available sensors. Each mote that wants to detect new motes can subscribe to that broadcast channel. By default every mote in the network is subscribed to that channel and is able to

detect new motes and their available sensors.

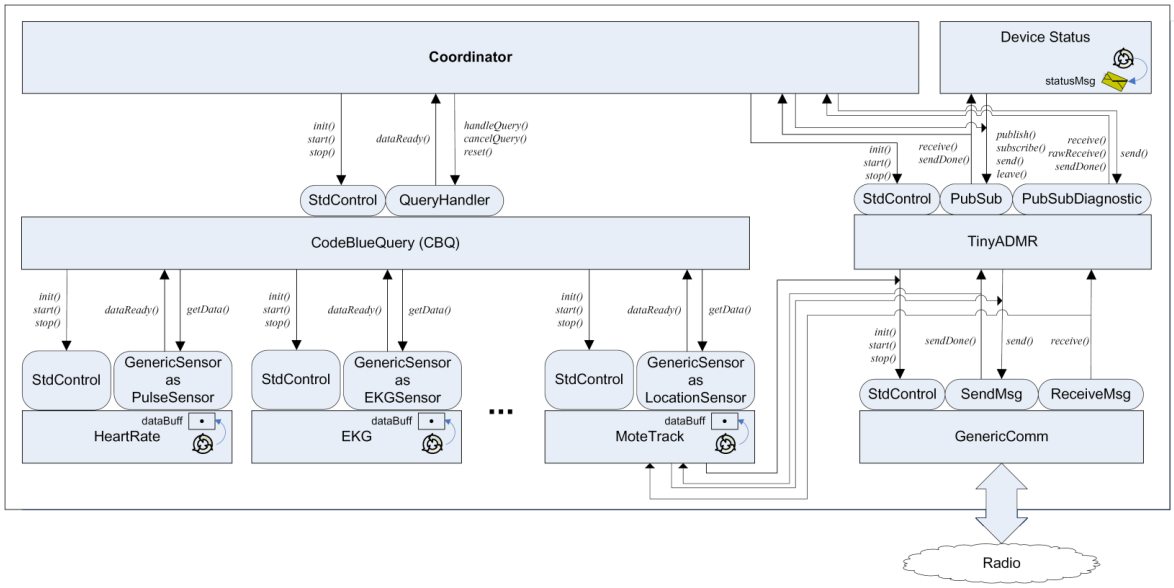

Figure 2.3.: The CodeBlue software architecture [SCL+05]

The sensors on a mote are accessed via the architecture outlined in figure 2.3. For each sensor type the `GenericSensor` interface is used to access the sensor. Data from the sensor can be requested by `getData()` and the interface signals a `dataReady()` event when the data is in the buffer. The event return a pointer to that buffer.
Each sensor component also implements the TinyOS StdControl interface allowing the associated hardware to be powered on or off by using the `start()` or `stop()` functions. A component has to be initialized before the first usage by `init()`.
CodeBlue uses a query layer (CodeBlue Query Layer, CBQ) to get data from the sensors. allows the receiver to establish communication pathways by specifying the sensors, data rates, and optional filter conditions that should be used for data transfer. Queries are generated by end-user-devices such as a laptop running the CodeBlue GUI. The `coordinator` receives query messages from the radio, processes the message (e.g. for debugging) and forwards the query to the CBQ component.

2.3. The Graphical User Inferface

The user can interact with the CodeBlue sensor network with the CodeBlue graphical user interface (GUI). The GUI is a Java application and can run on any PC or laptop computer. All data from the network is sent to the GUI via a base station mote that is connected to that computer.

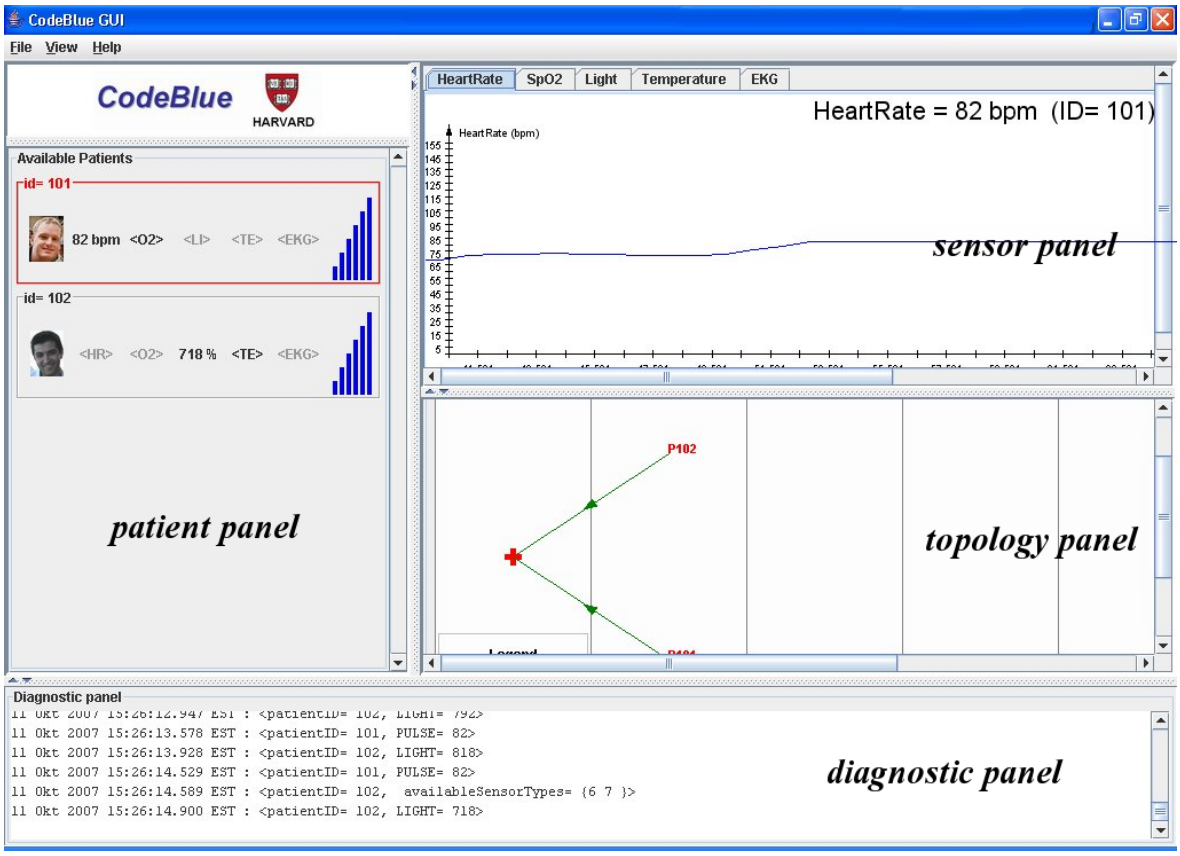

Figure 2.4.: The original graphical user interface of CodeBlue

The CodeBlue GUI as shown in figure 2.4 is separated into two columns. The left column shows the list of patients currently connected to CodeBlue. For each patient the ID, a picture, available sensors and a 'strength bars' indicating the network path quality to this patient sensor is shown [SCL+05]. When a patient is selected from the *patient panel*, the sensor data will be displayed in the *sensor panel* in the right column. The tabs for light and temperature have been added by the team at UTS. In addition, an overview of all current sensor values is displayed in the patient panel. In the sensor panel each sensor has its own tab. Recent sensor values are visualized using a graph. The current sensor value is also displayed with the patient ID in the upper tight corner of the sensor panel.

The *topology panel* is located underneath the sensor panel. All motes connected to the system are displayed here according to the topology. Lines represent the connections between the Motes. This way multi hop routing paths can easily recognized.

Underneath the two columns is the *diagnostic panel*. All output and debug messages are printed to this panel.

The user can subscribe to data channels by simply clicking on patient's available sensors displayed in the patients list. This will generate a query that is then sent to the appropriate mote in the network. The mote then will start publishing sensor data on the channel defined

by the query. A CBQ query is specified by the tuple $\langle S, \tau, chan, \rho, C, p \rangle$. S represents the set of node IDs that should report data for this query and τ is the sensor type representing a specific physiological sensor. Examples of sensor types include heart rate, SpO_2, EKG, temperature and light. This model allows a single node to support multiple physical sensors. Results from the query should be published to the ADMR channel $chan$. The query also specifies the sampling rate ρ and an optional count C of the total number of samples to retrieve from each node (if C is unspecified, it is assumed to be infinite) [SCL+05].

2.4. Conclusion

In this chapter, the author has described the CodeBlue Framework on which the ReMoteCare prototype is based. He illustrated the general architecture as well as the wireless sensor network and the graphical user interface.
The next chapter describes the implementation of the management proxy.

Chapter 3.

Implementing a Management Proxy

In order to enhance the usability and increase the number of scenarios in which CodeBlue can be used, it is important to forward the medical data to other networks. At the same time, it is important to monitor and manage the wireless sensor network in terms of network management.
This chapter describes the development of a management proxy which is able to forward the sensor data to other network (e.g. IP based networks such as local area networks) and which is able to manage and monitor the network. Especially in medical applications the reliability of devices is essential. The ability to monitor the network will add safety features to the application. For example, the motes could be monitored on their battery power and even before they run out of battery and fail, the batteries can be easily replaced.
The forwarding of sensor data into IP networks will increase the flexibility of the ReMoteCare prototype. Right now the communication of the WSN is limited to the connection between the GUI and the mote network. With the the help of the proxy almost any kind of application could be able to display and process the sensor data from the motes. In particular Network management systems (NMS) are suitable to process data generated by ReMoteCare.
This chapter will give a brief overview of network management and its relation to medical monitoring systems. Section 3.2 discusses suitable protocols for the management proxy. Section 3.3 will give a overview of the Simple Network Network Management Protocol (SNMP) which was chosen for the implementation. Finally, the last two sections will outline the work already done on the management proxy and describe the architecture of the actual implementation.

3.1. Conceptualisation of Network Management

Standard network management is one of the key tasks in every network. When it comes to wireless sensor networks, this task is even more important because of the huge number of possible devices in the network. At the same time, managing patients in a medical context

has also much in common with classic network management. It is all about collecting period data samples (like link usage or heart rate) from a large number of entities (like switches or patients), interpreting this data, triggering alarms or predicting trends [MLK+08].
A common model for network management is *FCAPS*, or *Fault Management*, *Configuration Management*, *Accounting Management*, *Performance Management* and *Security Management* [MS05]. These conceptional areas were created by the International Organization for Standardization[1] (ISO) to aid the understanding of the major functions of network management systems.
The following sections will outline Fault Management, Configuration Management and Security Management and related them to network management in a medical WSN application as well as look at them from a medical perspective. Accouting Management and Performance Management are important for medical monitoring systems as well but the aspects are very similar to the one in classical network management so their discussion is left out at this point.

3.1.1. Fault Management

The goal of Fault Management is to detect, log and notify users of problems that occur in the network or system [MS05]. Fault management is used to increase the reliability of the network and minimize the downtime of the system. WSNs already have some kind of fault management implemented. If a node of the network fails the network will adjust to the new condition and route the packets over another path to its destination.
From a medical perspective, any life threatening incident could be considered a fault. Where in a switch the increasing number of malformed packages may indicate a upcoming problem the instability of heart rate could do so for patients. Unlike in classical network management the focus is not on isolating and resolving the problem but on detecting a critical condition of a patient.
Besides the medical aspect, fault management for the WSN has also to be considered. Failure of network nodes due to dead batteries could be prevented by monitoring the power level and by replacing the batteries before they fail.

3.1.2. Security Management

The objectives of a security management subsystem are to control access to network resources to ensure that sensitive information cannot be accessed without appropriate authorization [Nav08]. This fact is essential for medical systems since any patient data is highly

[1] `http://www.iso.org`

sensitive. In network management various sytems have been designed specifically for this purpose [MS05]. These include:

- Firewalls
- Intrusion detection Systems
- Antivirus systems
- Policy management and enforcement systems

Especially the last point applies for a medical system that handles sensitive data. Protecting the system with a password, creating several access level policies and encrypt the communication are the aspects of Security Management that have to be transferred into the medical domain.

3.1.3. Configuration Management

The goal of configuration management is to monitor network and system configuration information. In classic network management this information may include the number of network interfaces of a device, speed of those interface and firmware version for example.
In medical terms this would relate to the current condition a patient is in. However, not just medical data has to be taken into account. Environmental variables such as light, room temperature and (if available) video images will provide additional information that helps medical staff to make decisions.

3.1.4. Summary

As seen, medical monitoring has a lot of overlapping concerns with network management. Therefore it is a straightforward approach to use network management systems (NMS) for medical applications. Still, some considerations have to be made. According to [MS05] there are four levels of activity in network management:

- ***Inactive***. No monitoring is being done and if an alarm is received it would be ignored.
- ***Reactive***. No monitoring is being done. A problem is solved as soon it occurs.
- ***Interactive***. Network components are monitored but the troubleshooting has to be done interactively to isolate the root cause.
- ***Proactive***. Network components are monitored and the system provides a root-cause alarm for the problem and initiates a predefined automatic restoral process where possible.

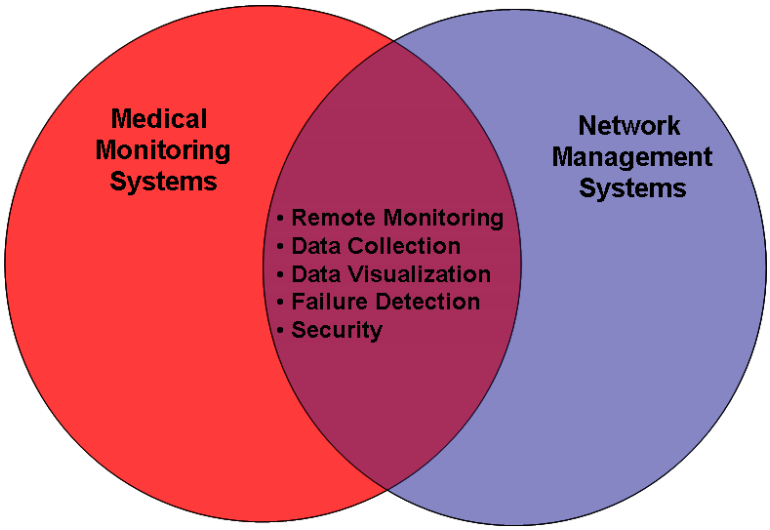

Figure 3.1.: Intersection between a NMS and a medical monitoring system. (based on [Nav08])

Medical system would fit only in the last two categories. "No monitoring" as the first two activity level suppose is just not acceptable. Meanwhile, the proactive approach has to be altered in that way that the system itself is not yet able to provide adequate solutions (such as cardiopulmonary resuscitation for example) for medical emergencies. It can only alert the carer to these problems

Figure 3.1 illustrates the main intersection between NMS and medical monitoring systems. Remote Monitoring, Data Collection, Data Visualization, Failure Detection and Security are the main aspects that overlap in both segments. Therefore a suitable protocol would need to be able to handle all five aspects. The selection of a suitable protocol is described in the following section.

3.2. Suitable Protocols

The development of a management proxy pursues two goals. The first goal is to forward medical data into a local network or even the internet. The system should be easily integrated with existing monitoring systems. The second goal is the monitoring of the wireless network itself. Therefore, this discussion of suitable protocols focuses on standard management protocols and the considerations made in section 3.1. In addition these protocols will be compared to the simple network management protocol (SNMP) since SNMP is the most accepted and implemented network management protocol. Some of these protocols have already been implemented on the motes. This section will give a brief overview of these protocols.

3.2.1. Universal Plug and Play (UPnP)

The Universal Plug and Play (UPnP) protocol is similar to SNMP as it requires large amounts of memory and computational power. As a result, the Business Operating Support System (BOSS) framework has been implemented in order to allow objects in the Sensor Networks that do not use the UPnP protocol to communicate with UPnP enabled Control Devices such as a Computer Terminal or a PDA. Additionally, UPnP operates on top of TCP/IP, thus allowing the sensor network to be easily connected to other networks [SKLS05]. Unlike SNMP however, the UPnP protocol also uses the XML format in order to interpret and transfer messages between the sensor network and the control point [SKLS05]. Other differences include the Security features used in order to accommodate privacy and protection issues. While Security for UPnP only involves User Authentication only, SNMP Version 3 improves on this by also providing end to end encryption for all communication.

3.2.2. Simple Network Management System (SNMS)

The Simple Network Management System (SNMS) is different from other management protocols as it only generates network traffic when directly responding to human action [DGA+05]. This improves energy efficiency as well as memory efficiency, and also minimises the impact on the computation and processing required. Because SNMS also occupies very little memory it was used on motes in [LDCO07]. Nevertheless it does not have other features such as Security. This protocol can also be applied directly to the motes, however it does not conform to the TCP/IP standard as SNMP does.

3.2.3. Sensor Network Management Protocol (sNMP)

Sensor Network Management Protocol (sNMP) is another technology that was compared to SNMP. sNMP involves the periodic management of network states, and uses two main methods of topology discovery: TopDisc and STEAM. While TopDisc is very scalable [LDCO07], the communication overhead produced huge impacts on the energy consumption. The STEAM algorithm was developed as an alternative in order to address this issue, however this results in the trade-off between energy efficiency and computational overhead [DN07]. Additionally, sNMP has no Security features, and is not part of TCP/IP.

Consequently, SNMP's implementation is a preferable alternative as it is used at the Control Point only, thus minimizing energy consumption and communication overhead issues. The SNMP protocol will be discussed in the following section.

3.3. SNMP

The TCP/IP standard for network management is the *Simple Network Management Protocol* (SNMP) [Com00]. It was introduced in 1988 to meet the growing need for a standard for managing Internet Protocol (IP) devices. SNMP provides its users with a "simple" set of operations that allows these devices to be managed remotely [MS05]. Network devices can be monitored by querying them over SNMP. Those devices will then answer to the queries with the appropriate answer. Queries can be generated by a large number of management tools. Management tools at the same time can also visualize the data transfered over SNMP (figure 3.2 shows the iReasoning MIB browser graphing light sensor data).
SNMP is a device independent standard to manage devices in a network. It allows multiple managers to talk to multiple devices (agents).

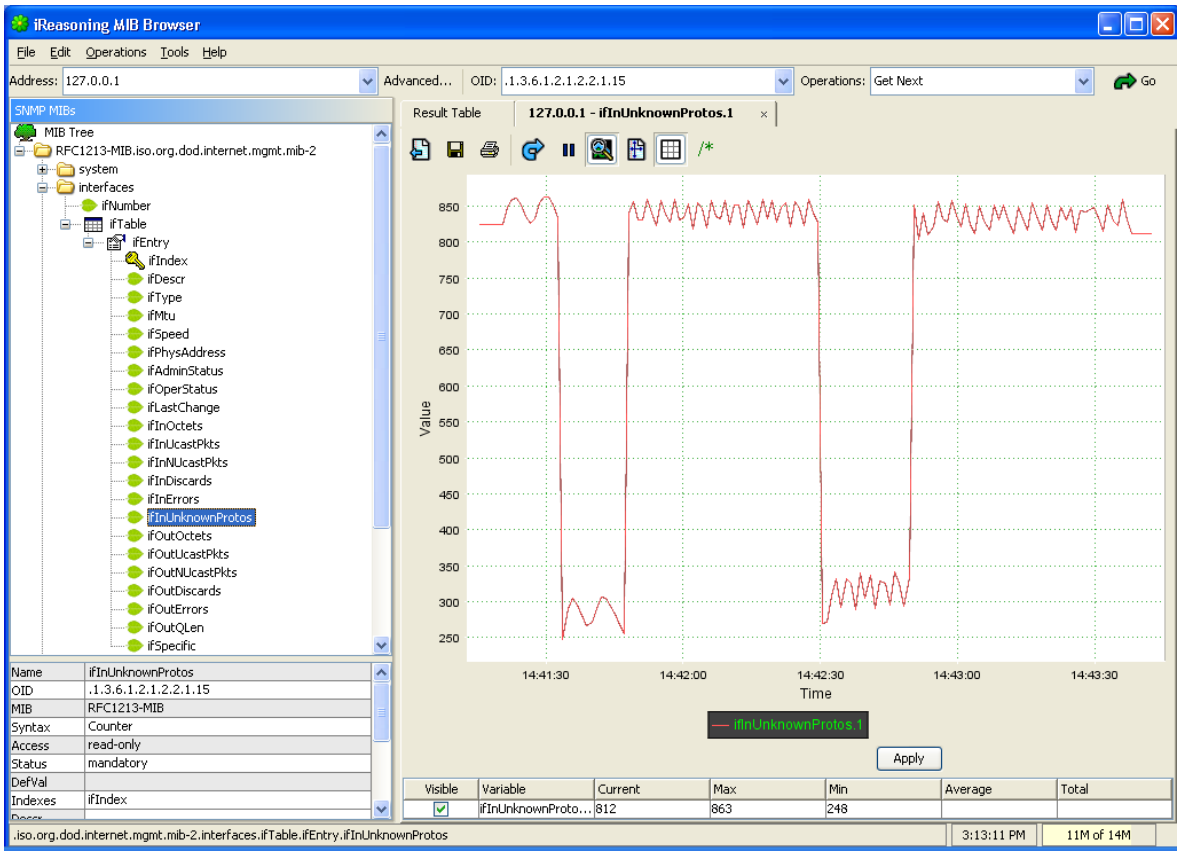

Figure 3.2.: MIB Browser displaying sensor reading for light

3.3.1. Benefits of SNMP

Making the data available via SNMP allows the use of existing network management tools with only slight modifications for medical monitoring. In essence, and as mentioned before

in section 3.1, patient monitoring and network monitoring are similar. Using SNMP, one monitoring and analysis station can supervise a large number of sensor networks from a central place.
SNMP is well established and a wide variety of clients and analysation tools are available. For this work the *iReasoning MIB browser* [iRe07] is used to display the SNMP data (see figure 3.2). Of course, more advances clients should be considered in the future, e.g. network management systems like *HP Network Node Manager* [Hew07] (formerly known as *HP Open View*). Ultimately, such network management tools could evolve to automated medical supervision tools that can even predict health problems [MLK+08].
Additionally, SNMP is very flexible and can be easily adapted to medical monitoring (e.g. by creating a customized MIB, see chapter 3.3.2).
As partly described above, SNMP was chosen because of the following reasons:

- SNMP runs on top of the wide spread IP protocol. This makes the proxy flexible to communicate through different kind of networks.
- SNMP could not just be used to transmit medical data but also (in future releases of ReMoteCare) could manage the wireless network itself.
- SNMP is well established and a wide variety of management tools already exist.
- The security model of SNMPv3 introduces reliable authentication and encryption functionality.
- Due to its simple set of commands, SNMP scales very good for any size of network.

SNMP covers all areas of network management. It can detect faults in the system and alert staff to take action in terms of Fault Management. It can monitor different kind of system as in Configuration Management and last but not least SNMPv3 enables authentication and encryption which is used for Security Management.

3.3.2. Management Information Base (MIB)

Each network device managed by SNMP must keep control and status status information that can be accessed by the SNMP manager. For example a router keeps statistics on the status of its network interfaces, incoming and outgoing packet traffic, dropped datagrams and error messages generated [Com00]. SNMP itself does not describe which information on a device can be accessed. This is accomplished by another standard, the Management Information Base (MIB). The MIB specifies which information a managed device has to provide. This information is divided into several categories. Table 3.1 shows a few examples of categories.

MIB category	Includes Information About
system	The host or router operating system
interfaces	Individual network interfaces
at	Address translation (e.g. ARP mappings)
ip	Internet Protocol software
icmp	Internet Control Message Protocol software
tcp	Transmission Control Protocol software
udp	User Datagram Protocol software
ospf	Open Shortest Path First software
bgp	Border Gateway Protocol software
rmon	Remote network monitoring
rip-2	Routing Information Protocol software
dns	Domain Name System software

Table 3.1.: categories of MIB information [Com00]

An advantage of the use of MIB is that a single SNMP management tool can be used to manage several types of devices. The tool only needs to load another MIB file that matches the MIB file of each device.
In addition to the standards that specifiy MIB variables and their meanings, a separate standard specifies a set of rules to define and identify MIB variables. The rules are known as the *Structure of Management Information* (SMI) [Com00]. The SMI standard specifies that a MIB has a tree structure and is noted in Abstract Syntax Notation One (ASN.1). Each entry in the MIB tree has a specific identifier. The identifier is also called *object identifier* (OID) This makes every object unambiguous because its name consists of numeric labels along a path from the root to the object. Figure 3.3 shows part of the OID tree, for example the object "private" in the tree has the OID 1.3.6.1.4. When a SNMP agent needs to access a specific object, it traverses the OID tree to find the object.
A MIB browser can be used to view the hierarchical OID tree. Like the iReasoning MIB browser shown in figure 3.2, MIB browsers are very often able to display a graphical view of the object data.

3.4. Recent Work

One goal of the WSN research group at UTS is to integrate SNMP into the ReMoteCare prototype and to use network management tools to monitor patients. Because WSNs are at an early stage of development with a lack of international standards (various research groups

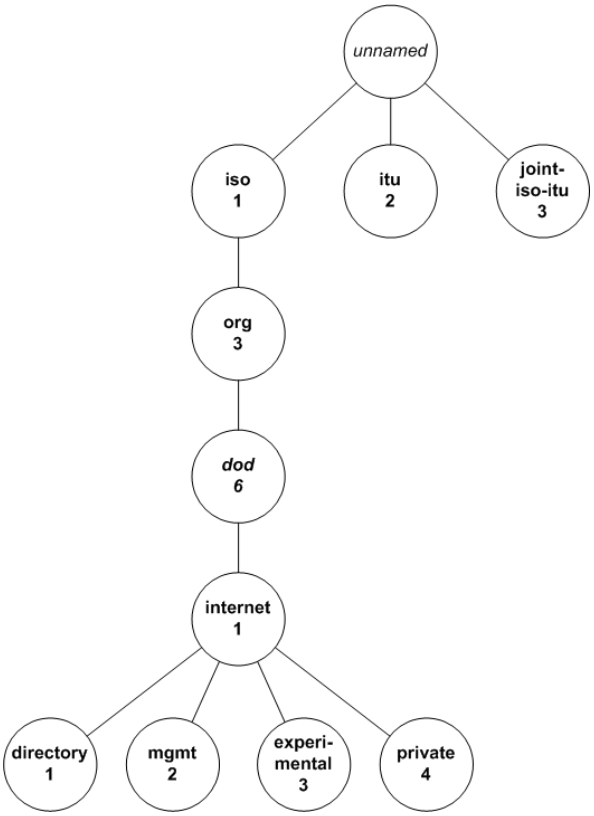

Figure 3.3.: Part of the hierachical object identifier namespace [Com00]

are currently working on IP based communication in WSN, e.g. [MF06]) among various WSN manufacturers, the CodeBlue "proprietary" communication protocol had to be utilized with an additional proxy entity for the prototyping of ReMoteCare. In order to address these limitations the Proxy Server Organization Model (shown on figure 3.4) was incorporated to allow non-SNMP managed objects, like the motes, to communicate with a network manager [Nav08].
The proxy architecture is often used for legacy systems management when devices without an SNMP agent have to be managed [Sub99]. The proxy converts management information into a set that is compatible with SNMP and communicates with the SNMP manager.

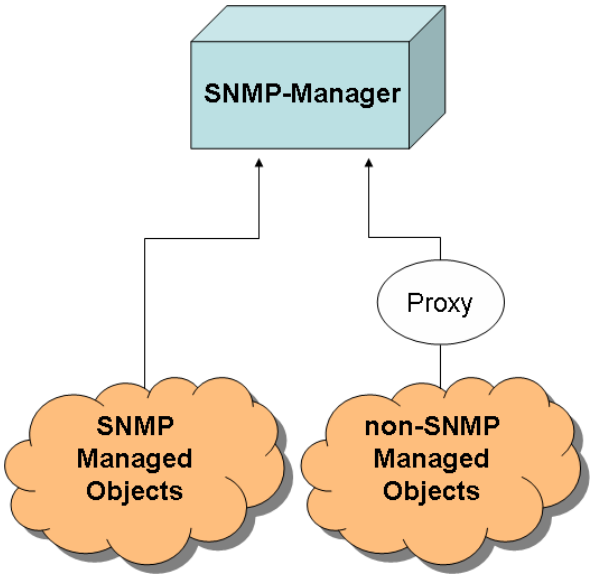

Figure 3.4.: Proxy Server Organization Model. [Sub99]

The first version of the SNMP proxy working with ReMoteCare was developed by [LML+08]. They developed the proxy as a proof of concept to show that SNMP is able to transmit medical data. The proxy is a stand-alone application which reads the log file generated by ReMoteCare and converts the data into an SNMP table. This table can be queried with any SNMP management tool. Two steps are required to send data via SNMP. In the first step all sensor output of the ReMoteCare GUI is written into a log file. The entries in that log file have the following standard format.

```
<date><time>:<patient ID,type of sensor=sensor data>
```

The following lines indicate that a new mote with the ID 111 has been detected and two types of sensor have been detected:

```
14 Aug 2007 10:45:11.898 EST: New patient: id 111
14 Aug 2007 10:45:11.904 EST: <patientID=111,availableSensorTypes={1 2}>
```

OID	name	used as
1.3.6.1.2.1.2.2.1.1	ifIndex	mote ID as number
1.3.6.1.2.1.2.2.1.2	ifDesc	mote ID as string (mote-xx)
1.3.6.1.2.1.2.2.1.6	ifPhysAddress	mote ID as hex value
1.3.6.1.2.1.2.2.1.10	ifInOctets	pulse sensor value
1.3.6.1.2.1.2.2.1.11	ifInUcastPkts	light sensor value
1.3.6.1.2.1.2.2.1.16	ifOutOctets	oxygen sensor values
1.3.6.1.2.1.2.2.1.17	ifOutUcastPkts	temperature sensor values

Table 3.2.: OIDs used by the SNMP proxy prototype

An actual log file entry for a sensor reading would look like this.

```
14 Aug 2007 10:45:16.939 EST: <patientID=111,PULSE=80>
```

The proxy parses the log file and passes the data to the SNMP interface.
For the SNMP interface the *SNMP for Java* (SNMP4J) implementation was used. SNMP4J is an enterprise class free open source and state-of-the-art SNMP implementation for Java [FK07]. It provides all necessary classes and interface to realize an SNMP agent. To keep things simple the agent was derived from the SNMP4J standard example. This agent has a a static SNMP table which holds the sensor data and provides it for SNMP requests.
As OIDs for the SNMP objects a starting OID of `1.3.6.1.2.1.2.2.1` was used. In regular usage of MIB, this OID belongs to the data values of a network interface. Table 3.2 shows the OIDs used with the original meaning and the sensor data published under that OID.

3.5. Architecture

As described in section 3.4, the SNMP proxy already exists as a stand alone application. This application had to be integrated in the ReMoteCare prototype.
Usually, the SNMP agent (which is in our case the SNMP proxy) is responsible to collect the manageable data from the device (which is the ReMoteCare system). In this system though the data is written to the SNMP proxy by ReMoteCare. Since ReMoteCare uses a subscribe/unsubscribe architecture, the SNMP proxy does not know which sensor it should query, therefore the data is passed on to the SNMP proxy by ReMoteCare.
A `snmpProxy` object is added to the ReMoteCare GUI to provide SNMP functionality. ReMoteCare can communicate with the SNMP proxy by using the following functions:

- `addMote (int moteId)`

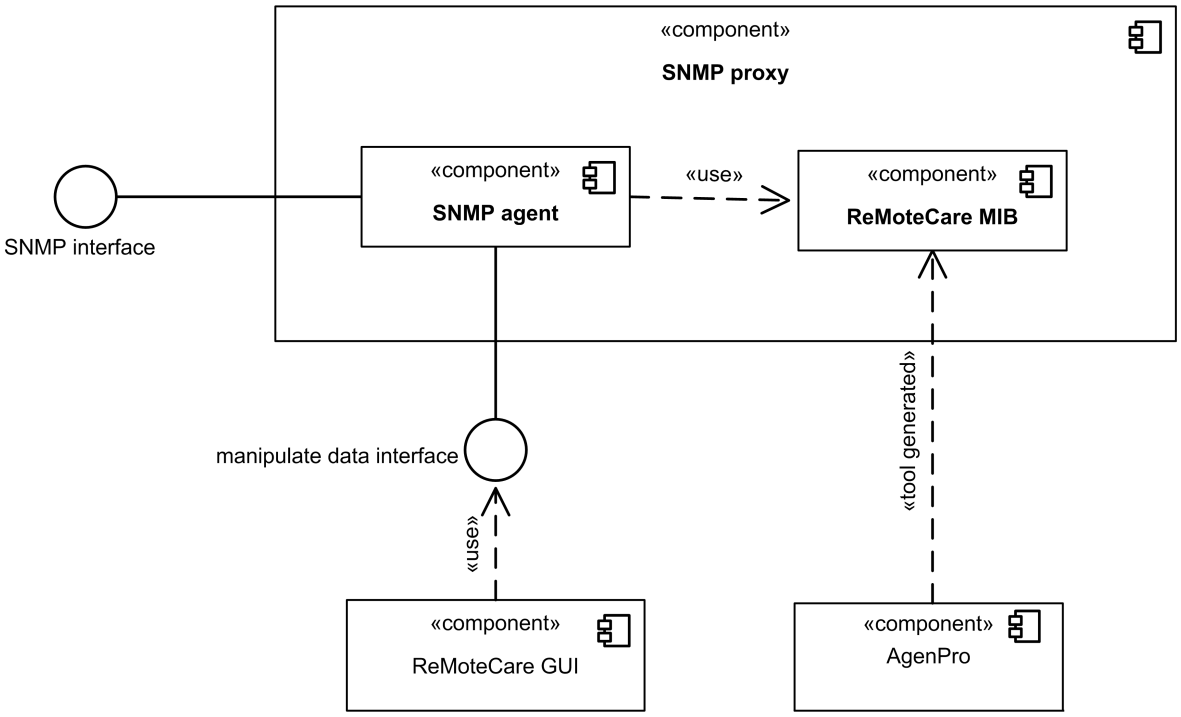

Figure 3.5.: Architecture of the SNMP proxy

- `setData (int moteId, short sensorType, int data)`

These functions call the appropropiate functions of the ReMoteCare MIB.
The general architecture of the SNMP proxy is illustrated in figure 3.5. The ReMoteCare GUI passed the medical data to the SNMP agent. The Agent uses the generated MIB class to store and manage the medical data. Network management systems can connect to the agent over the standard SNMP TCP port 161. The SNMP agent supports all three SNMP versions. For version 3, the manager has to authenticate itself with a password. This will be used in future releases of ReMoteCare to secure the system.
The MIB class was generated as described in section 3.5.2 by the *AgenPro* code generator.

3.5.1. Using a special Management Information Base

As mentioned before, the first prototype of the SNMP proxy used a static table for the SNMP data. It was derived from the SNMP4J agent example. To keep things simple, the developers did not change the OID for the data. They used the OID `1.3.6.1.2.1.2.2.1`, which is the OID for network interfaces, to publish medical data. Not all data fields of a network interface are being used, so the table contains a lot of empty columns.
As you can see in figure 3.3 there is also a experimental subtree of the OID tree. This subtree can be used for drafts and experiments [Fra07]. This subtree has the OID `1.3.6.1.3`. Since ReMoteCare with the SNMP proxy is still a prototype, the objects for the sensor data

are placed in this subtree. In a productive system this subtree can be moved to a more suitable position in the OID tree. The complete MIB file created for the ReMoteCare subtree can be seen in appendix A.
The ReMoteCare MIB is divided into 4 entries. The first two entries are used for the IP adress and the URL of the Stargate. The next two entries are tables. The first table holds all the medical data collected by the sensors. The second table is a management table. In this table information about the network such as signal strength, hop-count, etc are kept. Figure 3.6 show the created OID tree. The management table is not used for the ReMoteCare prototype at this time. In future releases this could be used to manage the network and motes.

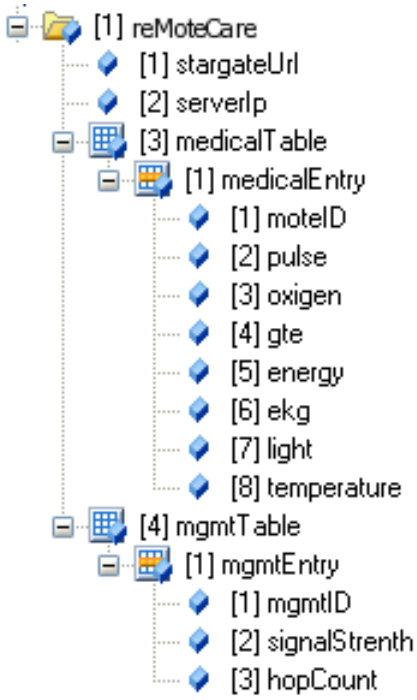

Figure 3.6.: The MIB tree created for ReMoteCare.

3.5.2. Implementing the ReMoteCare MIB

Code Generation is often used in the context of Model Driven Engineering (MDE). The prerequisite for effective code generation is a modeling language that expresses domain concepts effectively. As mentioned before for the SNMP domain such a modeling language is the Structure of Management Information (SMI) [Foc07b]. For the SNMP4J-Agent a code generator called *AgenPro* can be used to generate the SNMP-Agent stub code [Foc07a]. AgenPro generates code for the SNMP4J API when the SNMP4J-Agent code generation template is used. The code generation template carries the knowledge about the SNMP4J-Agent API domain. Thus, AgenPro can be used with other templates to generate code for

other agent APIs, but it can only generate code for the SNMP domain because it can process SMI specifications only. Figure 3.7 show the schematic process of the agent development based on AgenPro code generation. AgenPro parses a MIB file and generates a Java class that contains all the appropriate fields and methods. Additionally to the generated methods the methods for interacting with ReMoteCare have to be added. These will be described in the following section.

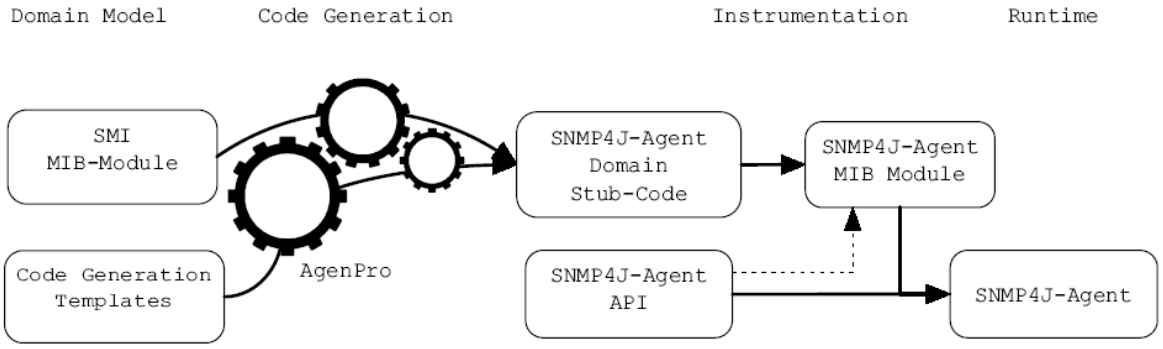

Figure 3.7.: Schematic representation of the agent implementation process based on AgenPro code generation. [Foc07b]

3.5.3. Interface of SNMP proxy

This section will describe the interaction of ReMoteCare and the SNMP proxy. The already mentioned functions `addMote` and `setData` call the `addMote` function or the `changeMedicalEntry` function of the ReMoteCare MIB respectively.
The `addMote` function will add a new row in the medical table of the ReMoteCare MIB. The appropriate code is shown in listing 3.1. The function creates a empty row with the patient ID as index and OID. To keep numbers and OIDs small and since all mote IDs start at 101, the new OID is patient ID minus 100. The `MedicalRowEntryFactory` is generated by AgenPro.

```
public void addMedicalEntryRow(int patientID) {

 MedicalEntryRowFactory factory=new MedicalEntryRowFactory();
 Variable[] values=new Variable[]{new Integer32(patientID-100),
                    new Integer32(0), //pulse
                    new Integer32(0), //oxygen
                    new Integer32(0), //GTE
                    new Integer32(0), //energy
                    new Integer32(0), //EKG
                    new Integer32(0), //light
                    new Integer32(0)};//temp

 OID index=new OID(new int[]{patientID-100});

 MOTableRow row= factory.createRow(index, (Variable[])values);

 ((DefaultMOMutableTableModel)medicalEntryModel).addRow(row);

}
```

Listing 3.1: Function to create a new row in the SNMP table

The `changeMedicalEntry` function updates an entry of a particular mote with a new sensor value. The mote ID is used to get the appropriate row of the medical table. The sensor type defines the column that hold the to be updated sensor value. Table 3.3 shows the sensor types with their column ID. Column 0 holds the mote ID.

```
public void changeMedicalEntry(int moteID,int column,int value){
 OID index=new OID(new int[]{moteID});
 Variable val=new Integer32(value);
 DefaultMOMutableRow2PC row2Update=
      (DefaultMOMutableRow2PC)medicalEntryModel.getRow(index);
 row2Update.setValue(column, val);
}
```

Listing 3.2: Function to change sensor data in the medical table

3.5.4. Security

Security Management plays an essential role in terms of network management. When combining network management with medical data this becomes even more important. The security portion of the SNMP proxy relies on the standard security mechanisms of the SNMP standard.

PULSE	1
OXYGEN	2
GTE	3
ENERGY	4
EKG	5
LIGHT	6
TEMP	7

Table 3.3.: Constant values for each sensor type defined by CodeBlue

While SNMPv1 and SNMPv2 only use so called *community strings* for authentication, SNMPv3 introduces user-based authentication [MS05]. RFC 2274[2] defines the User Security Model (USM) for SNMP. The USM supports authentication and privacy mechanisms. To support these functions, an SNMP engine requires two values: a privacy key (privKey) and an authentication key (authKey). USM allows the use of one of two alternative authentication protocols: HMAC-MD5-96 and HMAC-SHA-96. HMAC uses a secure hash function and a secret key to produce a message authentication code; HMAC is widely used for Internet-based applications and is defined in RFC 2104[3]. For HMAC-MD5-96, HMAC is used with MD5 as the underlying hash function. For HMACSHA-96, the underlying hash function is SHA-1 [Sta98].
For this prototype and for testing purposes, all three SNMP versions are implemented. SNMPv1 and SNMPv2 operate with the standard community string `public` and `private`. For SNMPv3 a user *SHADES* is created. This user has two passwords. In order to connect to the SNMP agent via SNMPv3 the SNMP manager has to authenticate itself with the authentication password. The privacy password is used to encrypt the SNMP communication.

3.6. Conclusion

In this chapter the author described the implementation of the management proxy based on SNMP. He related network management to health monitoring and pointed out the similarities. From that perspective he derived a suitable protocol for the management proxy, which turned out to be SNMP. He illustrated the architecture and implementation of the SNMP proxy and the used MIB file.
In the next chapter the author will describe the integration of the web cam into the ReMoteCare system.

[2] `http://rfc.net/rfc2274.html`
[3] `http://rfc.net/rfc2104.html`

Chapter 4.

Integrating a web cam

In terms of network management, Fault Management is one of the most significant areas. Its goal is to detect problems in the system. This is not always clear to detect from a medical point of view. False alarms can accur quite easily as the following scenario describes.
The scenario is set in an Aged Care Facility with patient Alice and a healthcare staff member, Bob. On Bob's screen he has access to Alice's latest vital sign information, her medical history and her location, as well as the room temperature and light. Having access to this information enables Bob to make fast decisions and send immediate help to Alice if necessary. Alice's heart rate goes beyond the predetermined threshold and the video appears on the main screen. In this scenario, Alice had been bending down to retrieve the television remote control from the floor and this effort had caused a sharp spike in her heart rate. He observes Alice relaxing as she has retrieved the remote control and he watches as her heart-rate slows. This is clearly not an emergency so he does not have to set the emergency response in motion. The video surveillance helped Bob to make the right decision [FLLG08].
This chapter will describe the integration of a webcam into the ReMoteCare system. The webcam is connected to a Stargate programming board and the pictures are addressed over a HTTP connection to the *Stargate*. Two different approaches are tested. The first approach publishes still images over a webserver running on the Stargate. The second approach provides a video stream in real time. Both setups will be presented in the following sections after describing the hardware and software setup and a brief discussion about video suveillance.

4.1. Discussion about video surveillance

Video surveillance always raises controversial discussions. Especially after 09/11 and the fight against terrorism, video surveillance plays an important role in our everyday life. A recent study calculates that if you live in London - the most surveilled of modern cities - you will appear on camera some 300 times a day just going about your normal business [MM07]. The London Closed-circuit television (CCTV) system - one camera per 15 inhabitants -

played a starring role in the media stories of the capture of those associated with the so-called July 2007 "Doctor's plot" which saw failed bombing in London and Glasgow [Lan07].
The discussion about video surveillance is mostly concerned with monitoring public areas. Integrating a video camera into the ReMoteCare system will bring this surveillance into private life. In Britain for example video surveillance in public is widely accepted. But monitoring people's home brings the discussion to the next level. Using live video streaming raises privacy concerns. Transmitting unprotected visual-signals over a network carries the risk that someone can monitor these transmissions.
On the other hand is video surveillance a common practice in hospitals and patient monitoring. In addition to the increase in overall security and safety, surveillance cameras can provide invaluable visual evidence for investigations of criminal activity and other specific events that have taken place within the hospital [Vid08]. The Faculty of Nursing, Midwifery and Health[1] also uses video surveillance for training purposes in their high fidelity simulation lab [Fac08].
Medical data in general is very sensitive. The video data should be also considered in the same way. Therefore access, usage and storage of this data has to controlled and protected [Kar01]. Cameras are effective tools for monitoring many sections of a hospital, but patient privacy should be considered when determining whether or not cameras should be placed in a facility's more private areas [Vid08]. Video surveillance enables medical staff to quickly verify a situation and reduce the number of false positive alarms. When monitoring elderly people not only the medical data is important. It is also very useful to have a visual oversight of the persons monitored. With the help of a picture or even better, a video, medical staff could quickly check on the patient's states.

4.2. Stargate

In order to communicate with other network such as IP networks, the motes need a gateway to that network. The SPB400 or *Stargate* shown in figure 4.1 is such a gateway. It provides several interfaces which are described in the following section. The Stargate is also able to host applications (such as a webserver or video streaming server) that interact with the motes and the connected networks.

4.2.1. Hardware

Stargate provides a small linux operating system to host all kinds of applications [Int05]. The Stargate it self has a PCMCIA slot and a slot for compact flash cards. Motes like the

[1] `http://www.nmh.uts.edu.au`

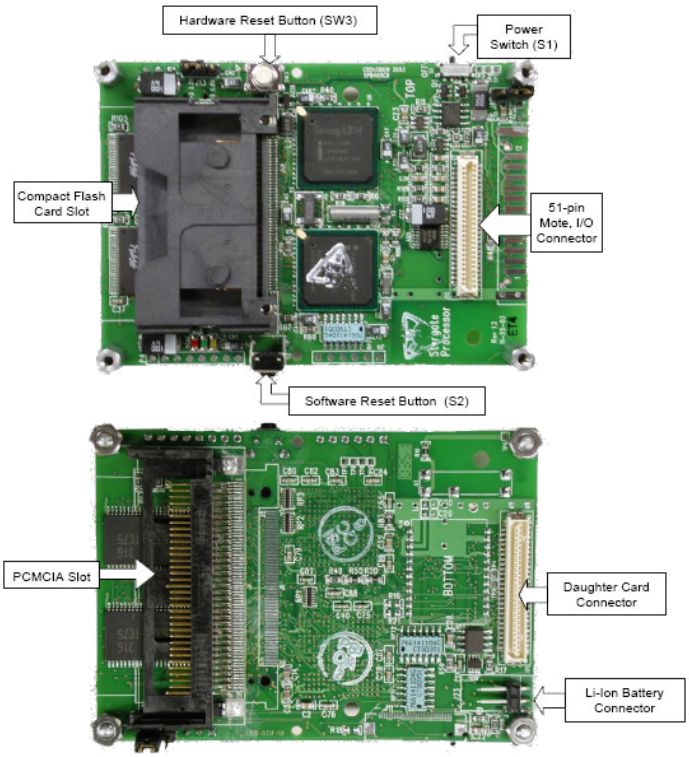

Figure 4.1.: Stargate Gateway [Dex07]

MICAz can be connected over a serial standard Mote connector [Cro07b].

The Stargate ships with a daughter card (shown in figure 4.2). This card has a ethernet port, a RS-232 interface as well as a USB port. It is connected to the Stargate via a 51-pin daughter card interface. The Stargate can also be powered over the daughter card. Due to this wide range of connections, the Stargate is highly flexible and can act as a gateway between several networks.

The Stargate is based on a 400MHz, PXA55 XScale processor which was developed by Intel. It has 64 MB SDRAM as well as 32 MB Flash memory. The flash memory is used to store the operating system (see 4.2.2) and the installed applications. This memory can be upsized by a compact flash card that can be mounted into the filesystem.

4.2.2. Software

The operation system running on the Stargate is a standard open source Linux distribution with kernel 2.4.19. The operating system and additional software are provided by the PlatformX Project founded by Intel [Int05]. Stargate supports 802.11 and Bluetooth wireless tools and provides programming and communication utilities for programming Motes. The current release number of the Stargate software is 7.3

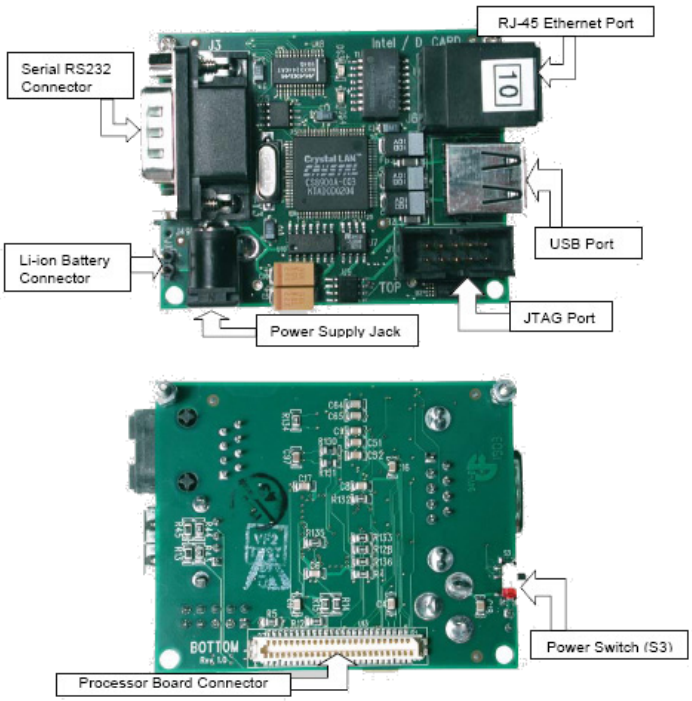

Figure 4.2.: Stargate daughter card [Dex07]

As of this release, Stargate supports the *Itsy Package Management System* (iPKG). iPKG is a very lightweight package management system. It was designed for Linux installations with severe storage limitations such as handheld computers [www07].
iPKG is adresses these limitations in several ways:

- The control programs themselves are small, (currently about 13kB)
- The installed meta-data tries to be only what is absolutely essential, (currently about 38kB for a 16MB compressed flash iPAQ distribution)
- The available packages are small. (The idea is that the package tree should be as fine-grained as possible. Much of this still needs some work)

With iPKG, the installation of applications is quite simple. The installation of the apache webserver for example comes down to one single line:

```
ipkg -verbose_wget install apache
```

iPKG will figure out the dependencies and download five packages (if a internet connection is available) and display progress bars for the installation process.

4.3. First Approach

The first approach was to publish still images with the help of a webserver. The camera is connected to the Stargate via USB. Acroname Robotics have developed a simple application, called *grabber*, that takes a single picture from the camera and saves it to the file system of the Stargate [Acr06]. This picture file then can be published with a webserver. The webserver shipped with the Stargate is the Apache webserver 1.3.26 [Cro06].

The camera used is a *Logitech QuickCam Pro 3000*. The camera has a maximum resolution of 640x480 pixel and has a built in microphone [Rut01]. It is connected to the Stargate via the Universal Serial Bus (USB). Figure 4.3 shows the camera connected to the testbed. The microphone is not used in the current implementation but might be considered in future versions of the video surveillance. That would have the advantage that patient and health carer could communicate.

4.3.1. Configuration

As mentioned above, the grabber application takes a single picture and saves it as a .png file. In order to refresh the picture, the grabber application has to be scheduled and run periodically.

To accomplish this, the scheduling service *uschedule* [Ohs07] is used. This service can be installed with the help of iPKG. Unfortunately, the repository at www.handhelds.org only holds an old version (0.5.6, whereas the current version is 0.7.1) of uschedule. This version does not support the use of time offsets. So, a job can not be scheduled to run for example every ten seconds. The job has to be scheduled at 10, 20, 30, 40 and 50 seconds after a full minute as well as at the full minute itself.

The popular *cron* application is also available for the Stargate's operating system. The drawback of *cron* is that the smallest scheduling period of a job is one minute. Even though the cron deamon *crond* checks the cron table every 30 seconds jobs are executed just every minute.

This makes *cron* unsuitable for the purpose of medical surveillance. A refresh rate of one minute is unacceptable since the condition of a patient could change within that minute dramatically.

There is no special configuration for the apache webserver necessary. The server listens on standard http port 80 and allows access to the files located in the `/home/apache/public_html` directory. For simplicity a symbolic link is created from the output file of the grabber application (which is located under `/home/acroname/aBinary`) to the `public_html` directory.

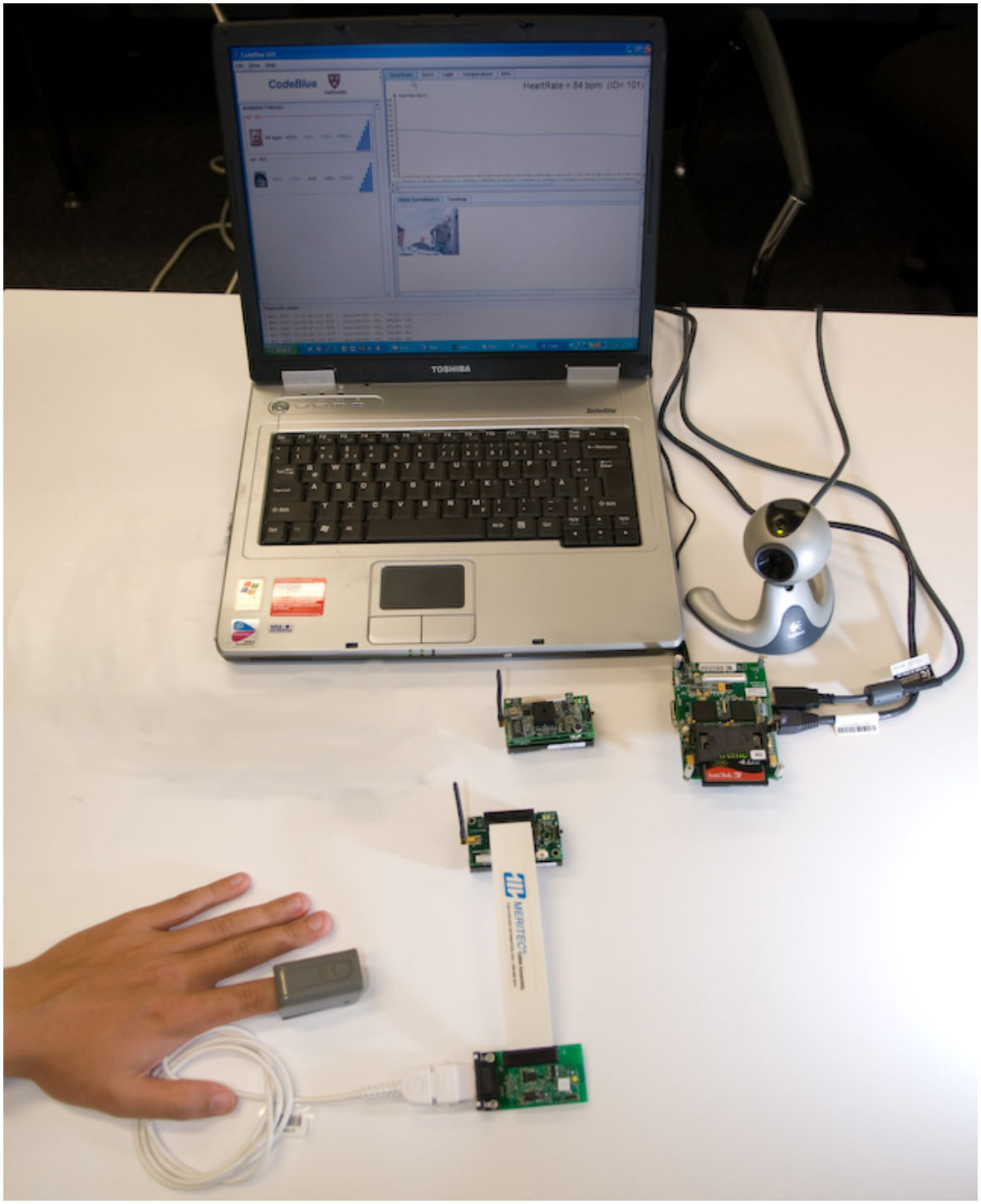

Figure 4.3.: Testbed setup with camera and Stargate

A simple webbrowser based on a `JEditorPane`[2] is integrated into the ReMoteCare GUI. This webbrowser connects to the Stargate via HTTP and displays the image. Every 10 seconds the image is reloaded from the server.
This first approach does not address any security concerns. The connection between GUI and Stargate is made with regular HTTP instead of the encrypted HTTPS. Changing the server side of this communication to HTTPS is quite simple. The following steps are necessary:

1. Generate a site certificate and a corresponding private key (OpenSSL could be used for that [Ope]) and install it on the Stargate.

2. Change the configuration of the Apache webserver to listen to port 443. At the same time a new virtual host should be created with SSL enabled and the generated keys declared in the `httpd.conf` configuration file.

3. To add authentication to the webserver a `.htaccess` file can be added to the directory were the image is stored.

On the client side, the simple webbrowser has to be changed as well. The JEditorPane does not support HTTPS connections. A new HTTPS client based on the Java Secure Sockets Extension[3] has to be developed. This client could connect to the server and forward the pages to a JEditorPane that displays the picture.
These security features were not implemented because the evaluation of this approach (described in the following section) proved this technology to be not as suitable for medical surveillance as live video is.

4.3.2. Evaluating the first approach

The first approach of the video integration provides a still image from the webcam. The image has a size of 320x240 pixel and was updated every 10 seconds. The image quality and size is sufficient to recognize important details of the picture (e.g. the facial expression of a patient). At the same time the picture is good enough to monitor larger areas such a rooms or hall ways. However, this approach has several disadvantages:

- a rather big webserver has to be installed and configured on the Stargate in order to publish one single picture.

- The integration via JEditorPane into the GUI was not working reliably. From time to time the refresh of the image did not work properly and just an empty frame was displayed.

[2] `http://java.sun.com/javase/6/docs/api/javax/swing/JEditorPane.html`
[3] `http://java.sun.com/javase/technologies/security/`

- The configuration of the image update takes quite some effort. Scheduling `uschedule` to refresh time of ten seconds takes 6 jobs to be scheduled. For shorter refresh rates, even more jobs have to be scheduled.
- After all, live video would provide much better information. If an refresh of an image fails, the information shown in the GUI might be up to 20 seconds old or even worse, no information is shown for up to 10 seconds. This is not acceptable for life critical monitoring.

Especially because of the last disadvantage mentioned above, the first approach was dropped. The second approach with a live video stream will be described in the following section.

4.4. Video Streams

The first approach to integrate a camera into ReMoteCare was dealing with still pictures that were refreshed on a regular basis. More up-to-date information is provided by video streams. Therefore a video stream with three main features is introduced to ReMoteCare. These features are:

1. Providing live video footage of the webcam in real time
2. Being able to record the video to the harddrive for later review
3. Alerting the medical staff by opening up an alert window that show the current video stream

This approach uses the same hardware as the first approach. The Apache webserver and the scheduling application are not used anymore.
Figure 4.4 illustrates the architecture of the video surveillance. The ReMoteCare GUI establishes two TCP connections to the Stargate. The first connection is on port 9000 to the serialforwarder to communicate with the motes in the WSN. All medical data is transfered over that connection. The second connection is is made on port 9192 to the video streaming server. The webcam is connected via USB and the streaming server provides the images over HTTP.

4.4.1. The image streaming server

Acccording to our our prototyping approach we use an already established stream server *camserv* to provide the video stream to the network. *camserv* is a open source project and supports all USB webcams that can be run with a Video4Linux driver. [Tra02]

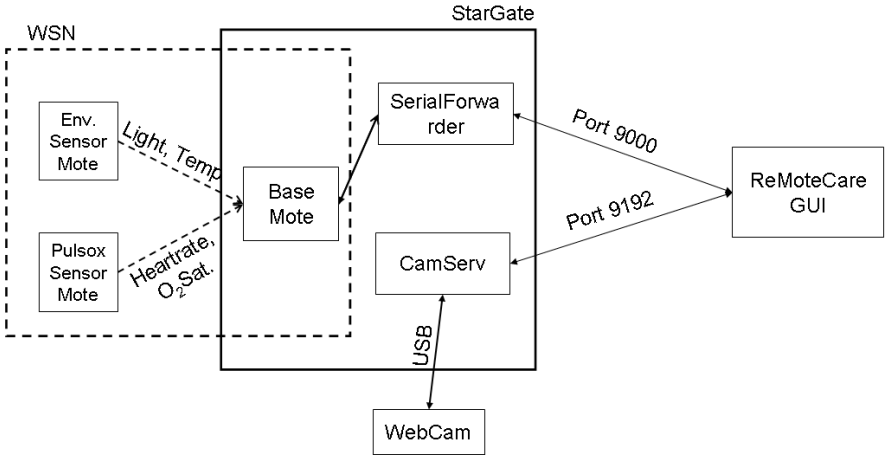

Figure 4.4.: Architecture of the video surveillance

Camserv is highly configurable and modular. It is controlled by a config file. This file is divided into several section. Each sections is responsible for a certain module of camserv. Listing 4.1 shows part of the configuration file. The `video_v4l_qcam` section controls the video driver behavior. `text_banner` and `time_stamp` are filters that are applied to the video image. They add a text message in the upper left corner and the current time in lower left corner of the picture. This might be an important feature when there is more than one camera in the system. The `fixed sections` define the general behavior of the camserv application, such as TCP port to listen on and what video device to use.

```
[video_v4l_qcam]
path          /usr/local/lib/camserv/libvideo_v4l.so.0
device_path   /dev/v4l/video0
width         320
height        240
port          0
color         30000
hue           30000
contrast      30000
brightness    30000
whiteness     30000
autobright    0

[text_banner]
path      /usr/local/lib/camserv/libtext_filter.so.0
text      Camera 1, Floor 4
bg        #000000
fg        #ffffff
x         0
y         0
```

```
mangle    0
fontname  6x11

[time_stamp]
path      /usr/local/lib/camserv/libtext_filter.so.0
text      Time: %X
bg        #000000
fg        #ffffff
x         0
y         230
mangle    1
fontname  8x8

##########################################
#  Begin Fixed Sections                  #
##########################################
[socket]
listen_port  9192
max_frames   0
max_bytes    0
max_seconds  0

[filters]
num_filters      3
filter0_section  time_stamp
filter1_section  text_banner
filter2_section  jpg_filter

[video]
video_section  video_v4l_qcam
width          320
height         240
maxfps         0
memhack        1
```

Listing 4.1: The camserv config file

The camserv application can be run without any command line parameters. It will open a socket on the in the configuration file specified port and provide the video stream on that port. Any webbrowser can connect to that port to display the video stream.
The video stream itself consists of a sequence of JPEG images. A mime-type content type image/jpeg informs the browser to expect an image. The images in the stream are separated by the following two lines followed by a blank line.

```
--ThisRandomString
```

```
Content-type: image/jpeg
```

4.4.2. Integration in the GUI

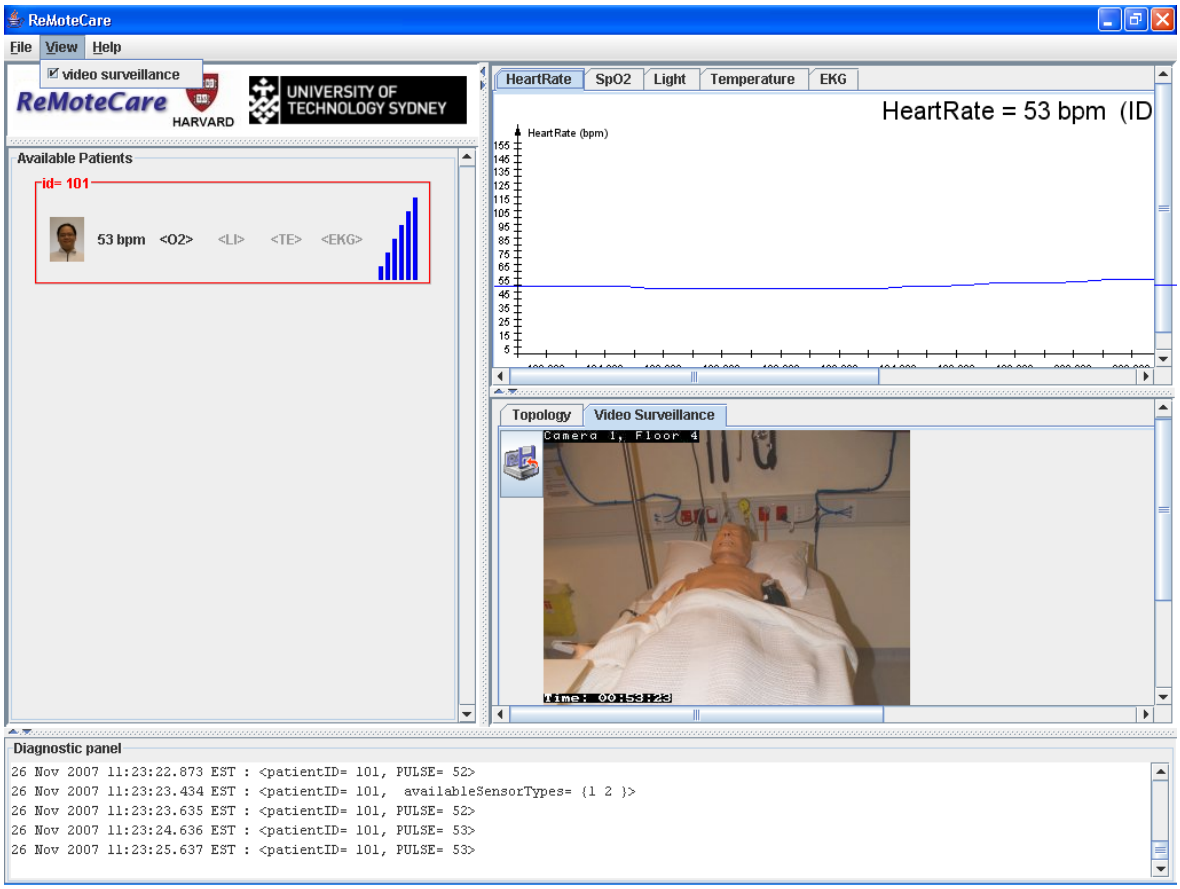

Figure 4.5.: Video picture within ReMoteCare

The video of the web cam is not assigned to a particular patient. It provides additional general information that might be important for every patient. It is useful to be able to see the video no matter which patient is selected from the patient summary panel. Therefore, the video stream can be viewed with two different methods. The first method is the web cam panel (see figure 4.5). This panel is displayed as a new tab next to the topology panel in the GUI. It can be activated over the View menu.

The second method is the alert window which will open up when a critical condition of a patient is reached. This method will be described together with the requirements of a critical condition in section 4.5.

The web cam panel consists of a JEditorPane (to connect to the camserv application and display the images) and a button to toggle the *record mode*. This mode allows the user to save the video stream to the hard drive. This feature will be described in the following

section.

4.4.3. Creating video files

Saving the video stream to the hard disc will give the medical personnel the possibility to review the video for evaluation and training purposes.
If the record mode is turned on, the image files displayed in the web cam panel are saved as JPEG images onto the harddrive. The current time stamp is used as a filename. At the same time, the filename is written to a vector to be processed later on.
The camserv application delivers approximately 6 images per second. After 360 images (which is about one minute of video stream) the images are converted into a QuickTime (`.mov`) video file. Listing 4.2 show the sourcecode that saves the images and starts the conversion after 360 images.

```
//save image to disk if record mode is enabled
if (isRecordMode()){

  Date now=new java.util.Date();
  String fn="video/"+String.valueOf(now.getTime())+".jpg";
  try {
    //save image as jpg
    ImageIO.write((RenderedImage) image,"jpg",new File(fn));
  } catch (IOException e) {
    e.printStackTrace();
  }
  counter++;

  imageList.add(fn);
  if (counter>=360) {  //one minute recorded. convert to .mov
    counter=0;
    filename="video/"+String.valueOf(now.getTime())+".mov";
    convertFiles(filename);
  }

}
```

Listing 4.2: saving images to harddrive

The source code for converting images to a video is taken from a code sample from the Java Media Framework[4] by Sun Microsysem [Sun07]. A custom `PullBufferDataSource` is designed to read compressed JPEG data from the vector that contains a list of JPEG still image files. For each file read, a `Buffer` object is created and the compressed data is set on

[4] `http://java.sun.com/products/java-media/jmf/index.jsp`

the Buffer object. The custom PullBufferDataSource is then used to create a `Processor`. The `Processor`'s output is set to generate QuickTime data. The output `DataSource` of the `Processor` is then hooked up to a file `DataSink` to save the bits to a movie file.

4.5. Triggering an alert window

In health monitoring it is always important to detect anomalies. A sudden change of heart rate can be an indicator for a critical condition for the patient. To notify the medical staff of an emergency situation a notification window will pop up in the user interface. This window shows the live video stream. At the same time the record mode is turned on and the video stream is saved to the harddrive.
In addition to the medical critical condition a trigger for the light sensor is implemented. When monitoring a room it might be useful to activate the camera when the light is switched on. The light sensor will recognize that and open up the alert window.

4.5.1. Parameters for critical condition

The parameters that indicate a critical condition as defined by [GGW+05] are shown on table 4.1. In a real environment these parameters have to be adjusted based on the medical record and the actual condition of the patient. Due to lack of the appropiate hardware the heart rate stability and the blood pressure conditions are not implemented. The heart rate stability requires a electrocardiograph (ECG) and for measuring blood pressure a blood pressure cuff is needed.

Alert Type	**Detection Parameter**
low SpO_2	$SpO_2 < 90\%$
bradycardia	$HR < 40bpm$
tachycardia	$HR > 150bpm$
HR stability	max HR variability from past 4 readings > 10%
BP change	systolic or diastolic change > $\pm 11\%$

Table 4.1.: Alert detection parameters [GGW+05]

All these conditions are implemented in the ReMoteCare GUI. Everytime a sensor reading is processed, the sensor values are checked if a critical condition is reached. If that is the case, a new window is opened. This window show the video stream of the camera. At the same time the recording mode is turned on and the video stream is saved to the hard disc. This allows medical staff not only to check the patient visually but also allows the doctors to review the video later on for analysis purposes.

4.5.2. The alert window

The alert window as shown in figure 4.6 is just a simple window that has a single button to start and stop the recording of the video stream. By default the video stream is recorded as soon as the alert window pops up. The window also stays on top of all other windows on the desktop.
The implementation of the alert window is the same as the web cam panel. In addition to that an information text is displayed underneath the actual image. This is helpful to see which sensor of which patient caused the critical condition. The reason of the critical condition (e.g. high heartbeat rate) is also shown in that info text.
The testing of the alert window was done with all the other test that are described in chapter 5.

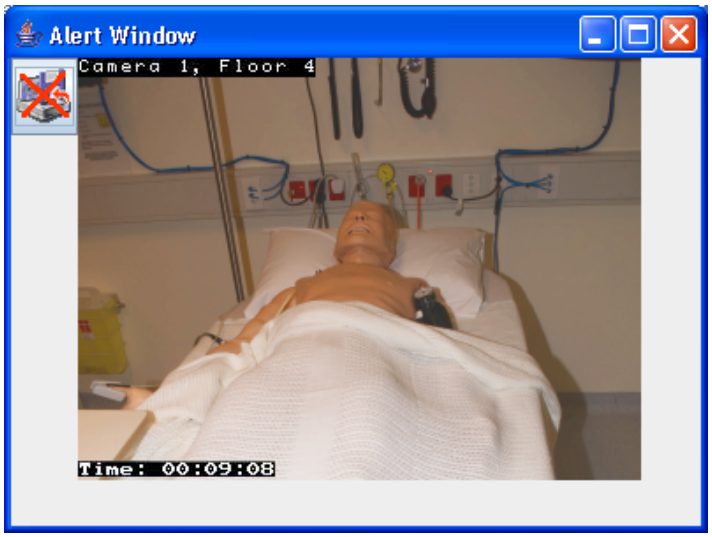

Figure 4.6.: The alert window of the ReMoteCare GUI

4.6. Conclusion

In this chapter, the author described the implementation of video surveillance into the ReMoteCare prototype. The first approach of still images proved to be not suitable for medical surveillance. Therefore, a second approach with video stream was developed and implemented. In addition to that an alert window was implemented that alerts the carer when certain medical conditions are reached.
The next chapter will evaluate and test the ReMoteCare system in several scenarios.

Chapter 5.

Testing the ReMoteCare prototype

Since medical systems are often used in lifecritical situations or environments, their functionality and reliability has to be extraordinarily good. This also holds true for the ReMoteCare prototype.
This chapter will describe several tests that were run with the ReMoteCare prototype. The tests pursue several goals. First of all, the reliability of the system and the responsiveness are tested. In addition to that, the functionality of the newly developed components (i.e. the SNMP proxy and the video surveillance) of the ReMoteCare prototype have to be tested.
These tests and the results are described in the following sections.
Section 5.1 will describe the test for the overall responsiveness of the system. Several scenarios are tested on the system. At the same time, the SNMP proxy was tested with these scenarios.
The second part will analyze the functionality of the video surveillance integrated into the prototype. Again, several scenarios were developed to see the performance of the camera in real-life situations.

5.1. Testing the responsiveness

The first set of test were made in cooperation with [MLK+08]. In order to validate the architecture developed, a series of experimental scenarios are developed and to be tested in the laboratory. The various scenarios allow evaluating the reactivity of the monitoring system to changing conditions of the subjects.

5.1.1. Experimental Scenarios

In the first scenario was tested with two subjects: a 28 year old male person and a over 60 year old female. The subject is attached the pulse oximeter to the finger and has the Heart Rate and Oxygen measured for a total of 6 minutes. For the first minute the subject is in a

sitting position relaxing to establish the resting heart rate. During the next two minutes the patient steps slowly, and then spends one minute resting before finally stepping quickly for the last two minutes of the scenario. This scenario aims to understand how the system would react to fast changes and extreme conditions in the patient status, particularly the heart beat values.
In the second scenario the ReMoteCare prototype was compared to a different health monitoring system. Dr Valerie Gay and Dr Peter Leijdekkers from the Faculty of IT developed a Personal Health Monitor[1] (PHM) based on BlueTooth. Their system is able to collect electrocardiograph (ECG) data. ECG is far more accurate than a pulse oximeter and therefore is suitable for comparing the two systems. Two subjects (both 28 year old males) were tested in a similar scenario to scenario 1. The resting period of one minute in between was left out. At the same time, the medical data was transmitted via SNMP to a SNMP manager (in this case the iReasoning MIB Browser) to verify the functionality of the SNMP agent in the ReMoteCare prototype.

5.1.2. Experimental Results

Minor data losses were experienced but overall the system responded very well during the test bed experiments. The graphs in figure 5.1 show the results of the testing of the first scenario.

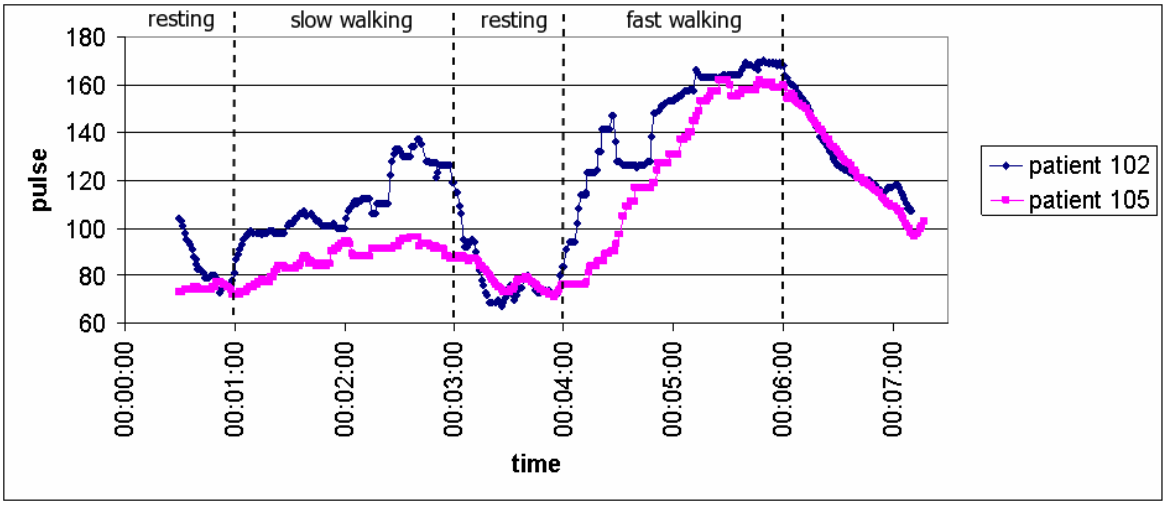

Figure 5.1.: Heartbeat rate of the two subjects in scenario 1 [MLK+08]

The younger male (#102) shows a higher range of heart beats compared with the older female (#105) as expected. Moreover the changes which occurred between the resting and the

[1] http://www.personalhealthmonitor.net

walking stages are well detected. Smooth changes for the older female patient (#105) and steep changes for the younger male (#102) are recognized by the system quickly and accurately in response to physical activity changes during the test. Analysis of the complete log files showed that the data losses experienced were minimal.

In the second scenario, the ReMoteCare prototype responded very similarly to the PHM. Figure 5.2 shows the graphs of both systems. After two minutes the sensor values start to divert. This is due to the fact that the pulse oximeter is meant to be used with little movement of the patient [Smi07]. The two subjects moved their hands quite a lot during fast walking and the sensor readings became inaccurate on the ReMoteCare system.

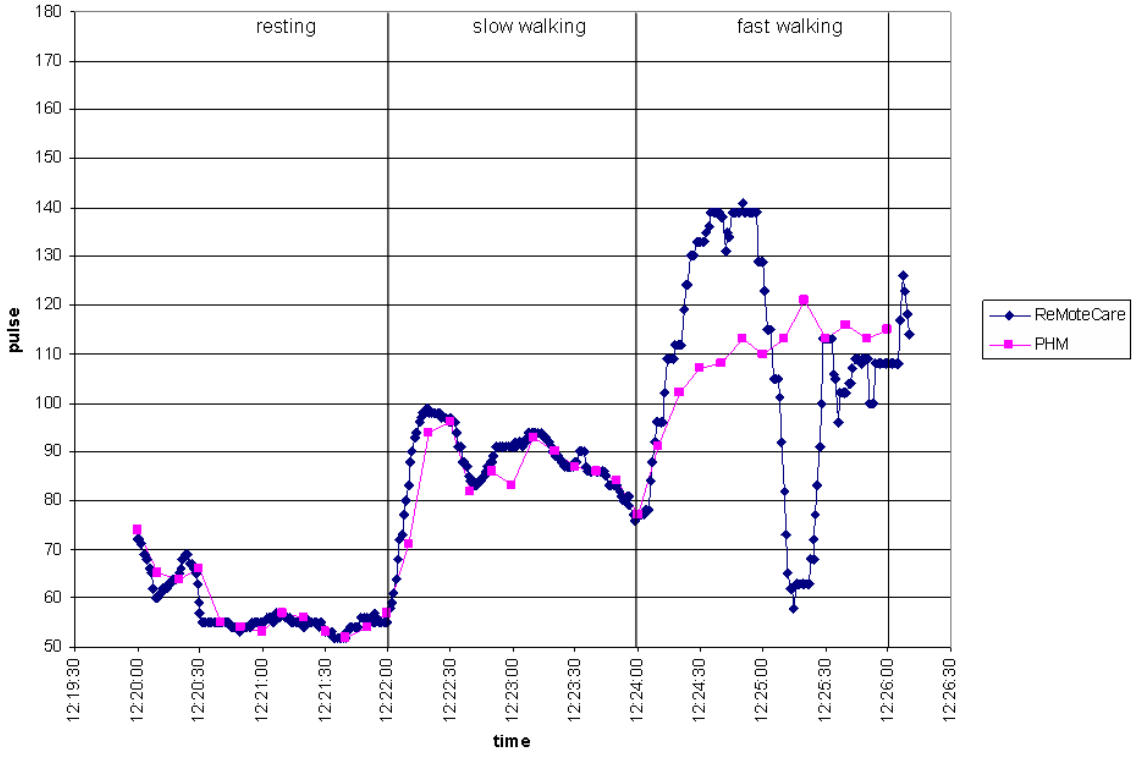

Figure 5.2.: Comparison ReMoteCare vs. Personal Health Monitor

The SNMP proxy also showed a good performance in the test. The SNMP proxy was polled from a remote computer at an interval of 1 second. The collected values were graphed and can be seen in figure 5.3. It can be seen that the graph is almost identical with the graph generated from the ReMoteCare log files.

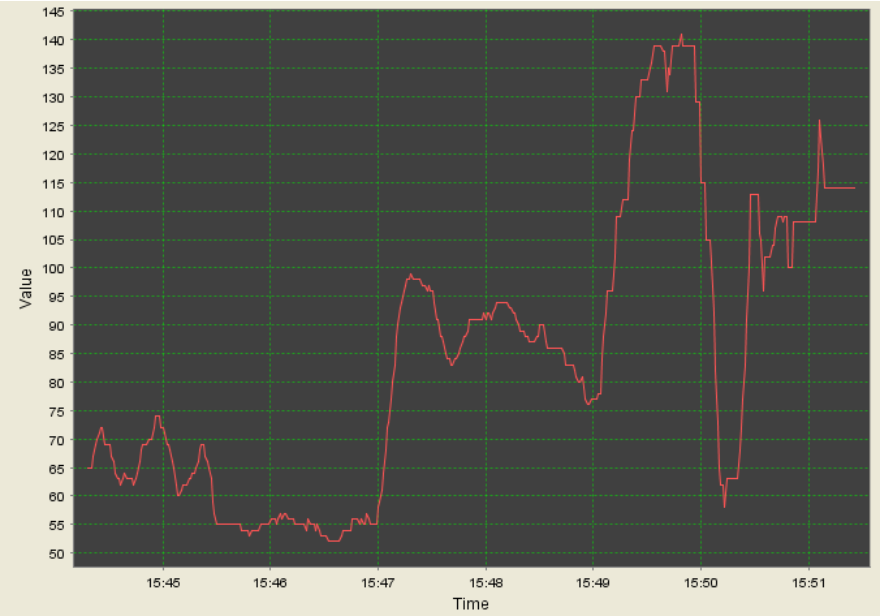

Figure 5.3.: Graph of SNMP data

5.2. Testing the video surveillance

This second series of tests aimed to prove that the video surveillance and the triggering of the alert window is working correctly. Again, a testing scenario was developed.

5.2.1. Testing scenarios

To test the functionality of the alert window, the trigger values of table 4.1 are used. However, not all parameters could be simulated in the laboratory, e.g. the blood oxigen saturation of 91% could not be reached by holding the breath and also the bloodpressure parameters could not be tested due to the lack of required hardware. To overcome the problem of the blood saturation the parameter was set to 95%.

In this test we were aiming at scenario 3 described above. The goal is to see if the video camera was switched on once the blood oxygen level reach the predetermined conditions as stated in table 4.1 on page 45.

Two subjects were asked to spend 1minute resting to obtain a normal resting heart rate. After 30 seconds of deep breathing they were then asked to hold their breath for as long as they could while we observed the blood oxygen measurement with the pulse oximeter. This scenario was tested on a 28 year old male (subject 1) and a 33 year old female (subject 2) person.

5.2.2. Testing results

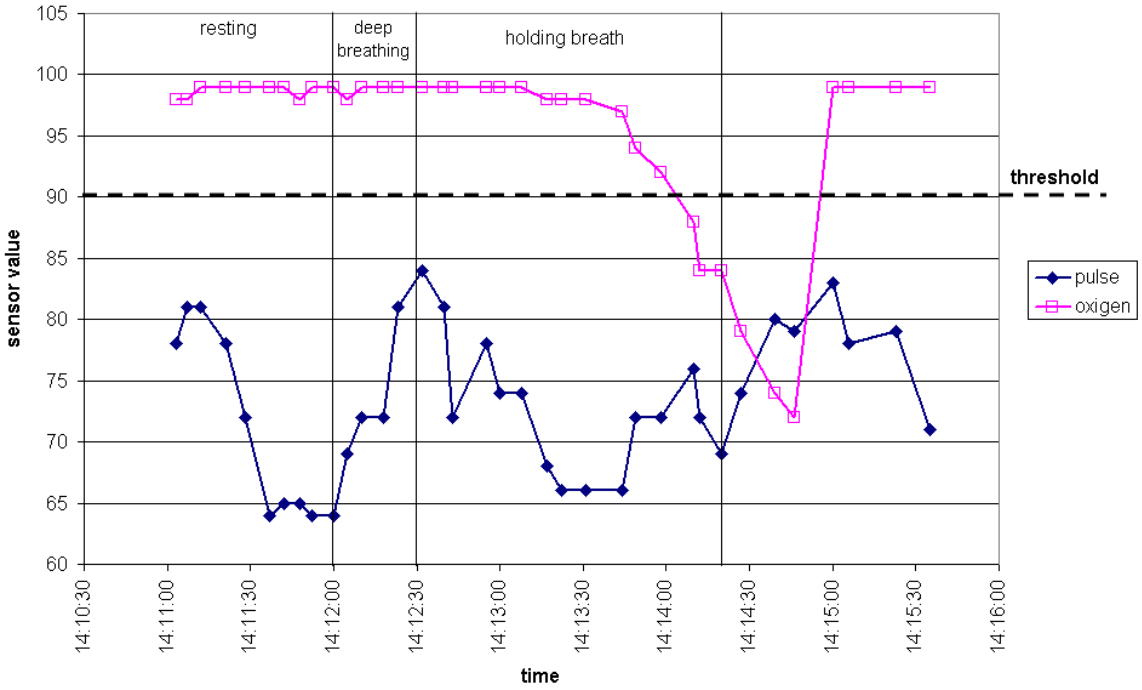

Figure 5.4.: Pulse and oxygen values of subject 1

As seen in figures 5.4 and 5.5 the system responded well to the changing conditions of the subjects. Due to technical limitations the pulse oximeter is not able to measure a oxygen saturation below 80% correctly [Smi07]. That is why the oxygen value of subject 1 even drops below 75%. The values of subject 2 are more reasonable.
The alert window opened up reliably for both subjects as soon as the oxygen value dropped below 90% for the male and 92% for the female. The variability of the threshold value illustrates the flexibility of ReMoteCare. Figures 5.4 and 5.5 show the sensor values with the predetermined alert threshold. As can be seen from the graphs the pulse rate increased as the oxygen levels dropped. Many persons who suffer sleep apnea exhibit this sort of behaviour and therefore video alerts in, for example, old person's home would prove useful for carers.

5.3. Evaluation

The experiments showed that a live video stream is a major enhancement to the ReMoteCare system. It enables medical staff to quickly verify a situation and reduce the number of false positive alarms. In addition, the ability to record the video stream gives medical professionals the capability to analyse critical situations afterwards. The reliability of the system was

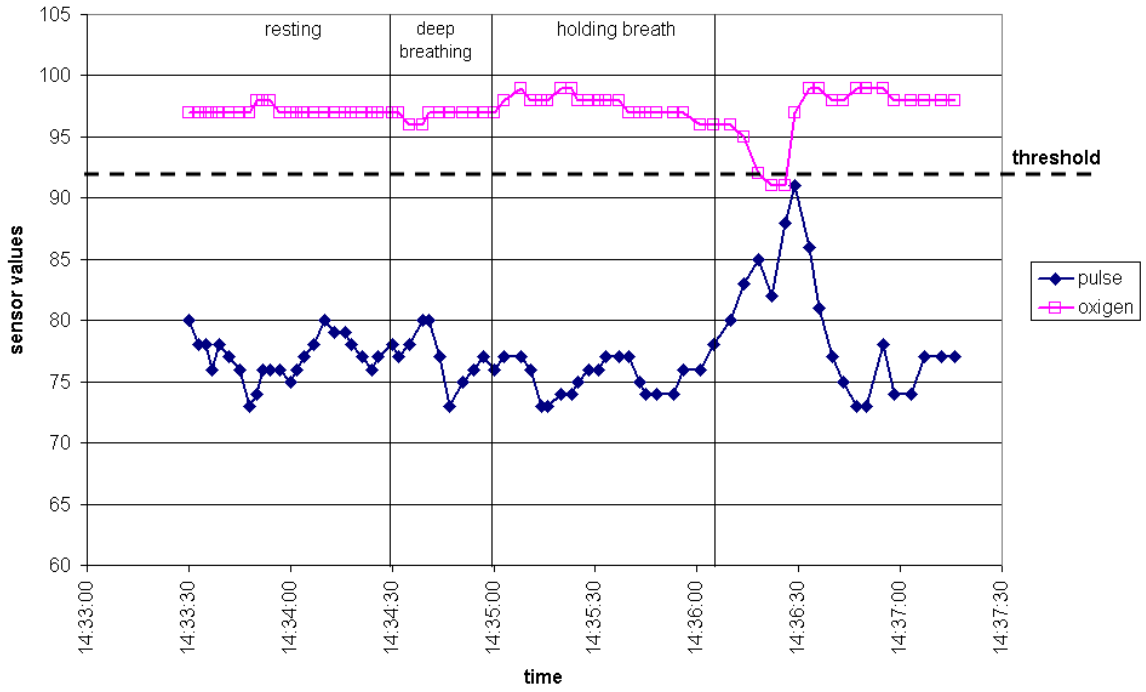

Figure 5.5.: Pulse and oxygen values of subject 2

proved by comparing the sensor readings to two different monitoring systems. ReMoteCare showed correct results as long as the movement of the monitored person was limited to walking. As explained pulse oximeters are best used for sedentary patients but for testing purposes the pulse oximers proved adequate. This problem could resolved by using a ECG based sensor instead of the used pulse oximeter.

Chapter 6.

Conclusion

This work dealt with the enhancement of the wireless sensor network application ReMoteCare. It advanced the recent work of a SNMP proxy and introduced a video camera for surveillance purposes.
This chapter will summarize this work and will give advice for future developments on the ReMoteCare prototype.

6.1. Summary

The first part of this work integrated the SNMP proxy into the ReMoteCare prototype. This enhanced ReMoteCare not just in terms of flexibility but also in terms of network management.

1. The ReMoteCare system is now able to propagate medical data into any IP based network. Monitoring stations can be placed in a remote location with any network connection. Any standard NMS is able to monitor patients over a secure connection.

2. Yet not fully implemented, the SNMP proxy added the possibility to manage the network itself. All major areas of network management (i.e. Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management) are covered by the functionality of SNMP and therefor available for the ReMoteCare prototype.

Chapter 3.1 showed that network management and patient monitoring have a lot in common. Therefore, network management system can be easily adapted for patient monitoring. The usage of the widely spread management standard SNMP helped to make this integration easy.
The video surveillance of ReMoteCare was introduced in chapter 4. Despite the privacy concerns, video surveillance is a major enhancement to the monitoring system. Alerting

medical staff in case of anomalies and the ability to detect false alarms quickly will make processes in medical environments more efficient. Nevertheless, careful attention has to paid to security and privacy. These issues are not addressed in this work. Future prospects in section 6.2 will give a brief overview of possible improvements.
The tests of chapter 5 proved the prototype to work reliable. The performance in comparison to other monitoring system was acceptable. In addition to that, the new developed components worked reliable and appeared to be a great enhancement.

6.2. Future Prospects

Several issues about the ReMoteCare prototype remain for future research. In addition more ideas about improving the prototype were developed while working on this project. This section will give a summary of future improvements.
First of all, security is a major issue. As pointed out in previous chapters, no security features (except for the SNMPv3 mechanisms) are implemented in ReMoteCare. This leaves a huge field of study. A general introduction to the security issues is given in [KLFL08]. Besides that, several proposals have already been made to integrate security and privacy on the motes using software/security protocols such as Sizzle [GMF+05], TinyPK [WKfC+04], TinySec [KSW04] and TinyECC [LKN07].
In addition to that, other alternatives such as the implementation of a VPN tunnel (as proposed by [Kar01]) or the use of the 802.1x standard may be worth further investigation. The first security alternative that has been considered for ReMoteCare is the use of a VPN tunnel. Currently, ReMoteCare uses two TCP connections to communicate to the Stargate. With the implementation of software such as OpenVPN on the Stargate, it is possible to pass these connections through a single VPN tunnel. This increases the security as all data is passed through one dedicated circuit, making it easy to add a form of encryption to prevent unwanted access. Additionally, because OpenVPN is a SSL based solution, it is easy to provide application level security for most web applications [Yon03]. This is because SSL is designed specifically to ensure that client/server applications can communicate while preventing eavesdropping, tampering or message forgery [FKK96]. A further advantage exists as OpenVPN would be implemented on the Stargate, rather than the motes themselves. Consequently, problems associated with the amount of memory and processing power available should be minimised. As a result, the implementation of OpenVPN may be a solution that may be further examined in the future.
Another alternative for the implementation of security into the ReMoteCare System is the use of the IEEE 802.1X, which is a standard for port-based Network Access Control. It aims to authenticate and authorise devices within the IEEE 802 LAN infrastructures. In

the event of the failure of the authentication and authorisation process, further access to the given port is prevented [IEE04]. Despite this, authentication in 802.1X only takes place at the beginning of the connection, thus making the connection vulnerable to tampering by a party that does not have permission to access the medical data that is within the ReMoteCare System. Additionally, the implementation of this standard also requires the use of additional hardware such as a third party authentication device (i.e. a RADIUS server). This results in additional expenses; however, because of a comparatively low overhead, this may be a solution that can be considered for future implementation within the ReMoteCare system.
Additionally, some other features could be explored as they are "nice-to-have". These include:

- Enhancing the SNMP proxy to send out traps for certain events.
- Manage the WSN and the motes via SNMP
- Integrate new sensors (ie a blood glucose monitor for diabetes patients)
- Integrate a secure data storage system for medical data

All these features are most likely to appear in future releases of ReMoteCare.

Bibliography

[Acr06] Acroname Robotics. Garcia teleoperation example. http://www.acroname.com/garcia/tutorials/aTeleOp/aTeleOp.html#e11, 2006.

[Aus04] Australien Bureau of Statitics. *Australien Social Trends: Scenarios for Australia's aging population*, 2004. http://www.abs.gov.au/Ausstats/abs@.nsf/2f762f95845417aeca25706c00834efa/95560b5d7449b135ca256e9e001fd879!OpenDocument.

[CGHK97] P.-C. Cheng, J. A. Garay, A. Herzberg, and H. Krawczyk. A security architecture for the internet protocol. *IBM Syst. J.*, 37(1):42–60, 1997.

[Com00] Douglas E. Comer. *Internetworking with TCP/IP*. Prentice Hall, 2000.

[Cro06] Crossbow Technology Inc. Stargate developer's guide. http://www.xbow.com/Products/productdetails.aspx?sid=229, 2006.

[Cro07a] Crossbow Technology Inc. Crossbow technology : Wireless sensor networks : Micaz 2.4ghz - wireless module. http://www.xbow.com/Products/productsdetails.aspx?sid=101, 2007.

[Cro07b] Crossbow Technology Inc. *Stargate: X-SCALE, PROCESSOR PLATFORM*, 2007.

[DBM+05] S.L. Dong, M.J. Bullard, D.P. Meurer, I. Colman, S. Blitz, B.R. Holroyd, and B.H. Rowe. Emergency triage: comparing a novel computer triage program with standard triage. *Acad Emerg Med*, 12(6):502–7, 2005.

[Dex07] Dexel Autonomous Systems Lab - Drexel University. Stargate tutorial. http://www.pages.drexel.edu/~ttl28/tutorials/stargate-tutorial.html, 2007.

[DGA+05] Prabal Dutta, Mike Grimmer, Anish Arora, Steven Bibyk, and David Culler. Design of a wireless sensor network platform for detecting rare, random, and

ephemeral events. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 70, Piscataway, NJ, USA, 2005. IEEE Press.

[DN07] Budhaditya Deb and Badri Nath. Wireless sensor networks management. `http://www.research.rutgers.edu/~bdeb/sensor_networks.html`, 2007.

[Fac08] Faculty of Nursing, Midwifery and Health. High fidelity simulation lab. `http://www.nmh.uts.edu.au/about/facilities/simulation-lab.html`, 2008.

[Fis06] Martin Fischer. Zigbee in der hausautomation. research paper, Universität Koblenz, 2006.

[FK07] Frank Fock and Jochen Katz. Snmp4java. `http://www.snmp4j.org`, 2007.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The ssl protocol version 3.0. IETF internet-draft, 1996.

[FLLG08] Martin Fischer, Ying Yang Lim, Elaine Lawrence, and Leena Ganguli. ReMoteCare: Health Monitoring with Streaming Video. In *7th International Conference on Mobile Business*, Barcelona, Spain, 2008. under review.

[Foc07a] Frank Fock. Agenpro v2.6. `http://www.agentpp.com/agen/agen.html`, 2007.

[Foc07b] Frank Fock. Snmp4j-agent & agenpro code generation. `http://www.agentpp.com/agen/agen.html`, 2007.

[Fra07] France Telecom SA. Oid repository. `http://www.oid-info.com`, 2007.

[GGW+05] T. Gao, D. Greenspan, M. Welsh, R. R. Juang, and A. Alm. Vital signs monitoring and patient tracking over a wireless network. In *The 27th Annual International Conference of the IEEE EMBS*, Shanghai, China, September 2005.

[GMF+05] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet (best paper). In *PERCOM '05:*

Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, pages 247–256, Washington, DC, USA, 2005. IEEE Computer Society.

[GR97] John Gosbee and Eileen Ritchie. Human-computer interaction and medical software development. *interactions*, 4(4):13–18, 1997.

[GS81] Hassan Gomaa and Douglas B.H. Scott. Prototyping as a tool in the specification of user requirements. In *ICSE '81: Proceedings of the 5th international conference on Software engineering*, pages 333–342, Piscataway, NJ, USA, 1981. IEEE Press.

[Hew07] Hewlett Packard. Hp network node manager. `https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119^1155_4000_100`, 2007.

[IEE04] IEEE Std 802.1x. IEEE 802.1X Standard for Local and Metropolitan Area Networks - Port-Based Access Control. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=4039841&isnumber=4039840`, 2004.

[Int05] Intel Corporation. Platformx with stargate. `http://platformx.sourceforge.net/`, 2005.

[iRe07] iReasoning Networks. Mib browser. `http://www.ireasoning.com/mibbrowser.shtml`, 2007.

[JJ01] Jorjeta Jetcheva and David B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, October 2001.

[Kar01] Atsushi Kara. Protecting privacy in remote-patient monitoring. *Computer*, 34(5):24–27, 2001.

[KKG07] Georgios Kambourakis, Eleni Klaoudatou, and Stefanos Gritzalis. Securing medical sensor environments: The codeblue framework case. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 637–643, Washington, DC, USA, 2007. IEEE Computer Society.

[KLFL08] Fank Kargl, Elaine Lawrence, Martin Fischer, and Yen Yang Lim. Security,privacy and legal issues in pervasive ehealth monitoring systems. In *7th International Conference on Mobile Business*, Barcelona, Spain, 2008. under review.

[KS06] Matthias Kranz and Albrecht Schmidt. Restriction, modification and extension of consumer devices for prototyping ubiquitous computing environments. In *ICDCSW '06: Proceedings of the 26th IEEE International Conference-Workshops on Distributed Computing Systems*, page 57, Washington, DC, USA, 2006. IEEE Computer Society.

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[Lan07] Mark Landler. Where little is left outside the camera's eye. The New York Times: `http://www.nytimes.com/2007/07/08/weekinreview/08landler.html`, July 2007.

[LDCO07] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver. Network management in wireless sensor networks. University of Western Australia, 2007.

[LGL06] Peter Leijdekkers, Valérie Gay, and Elaine Lawrence. Smart homecare system for health tele-monitoring. University of Technology, Sydney, 2006.

[LKN07] An Liu, Panos Kampanakis, and Peng Ning. Tinyecc: Elliptic curve cryptography for sensor networks (version 0.3). `http://discovery.csc.ncsu.edu/software/TinyECC/`, February 2007.

[LLN06] Einstein Lubrin, Elaine Lawrence, and Karla Felix Navarro. Motecare: an adaptive smart ban health monitoring system. In *BioMed'06: Proceedings of the 24th IASTED international conference on Biomedical engineering*, pages 60–67, Anaheim, CA, USA, 2006. ACTA Press.

[LML+08] Yen Yang Lim, Marco Messina, Elaine Lawrence, Frank Kargl, Leena Ganguli, and Martin Fischer. Snmp-proxy for wireless sensor network. In *5th International Conference on IT: New Generations. ITNG 2008*, April 2008.

[MF06] Karl Mayer and Wolfgang Fritsche. Ip-enabled wireless sensor networks and their integration into the internet. In *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, page 5, New York, NY, USA, 2006. ACM.

[MFJWM04] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[MLK+08] Marco Messina, Yen Yang Lim, Frank Karl, Elaine Lawrence, and Don Martin. Implementing and validating an environmental and health monitoring system. In *5th International Conference on IT: New Generations. ITNG 2008*, Las Vegas, USA, 2008.

[MLNL07] Marco Messina, Elaine Lawrence, Karla Felix Navarro, and Einstein Lubrin. Towards assistive healthcare: Prototyping advances in wireless sensor network (wsn) systems integration and applications. In *IADIS International Conference - Wireless Applications and Computing 2007*, Lisbon, Portugal, 2007.

[MM07] Katina Michael and M. G. Michael, editors. *From Dataveillance to Überveillance and the Realpolitik of the Transparent Society*. University of Wollongong, 2007.

[MMSS03] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele. Applying wearable sensors to avalanche rescue. *Computers and Graphics*, 27(6):839–847, December 2003.

[MRHO06] Kirk Martinez, Alistair Riddoch, Jane Hart, and Royan Ong. A sensor network for glaciers. In *Intelligent Spaces*, pages 125–138. Springer, 2006.

[MS05] Douglas E. Mauro and Kevin J. Schmidt. *Essential SNMP*. O'Reilly, Sebastopol, 2005.

[Nav08] Karla Felix Navarro. *A Heterogeneous Network Management Approach to Wireless Sensor Networks in Personal Healthcare Environments*. PhD thesis, University of Technology, Sydney, 2008.

[NK07] S. Nestler and G. Klinker. Using mobile hand-held computers in disasters. In *UbiComp Workshop on Interaction with Ubiquitous Wellness and Healthcare Applications (UbiWell)*, Innsbruck, Austria, September 2007.

[Ohs07] Uwe Ohse. uschedule: scheduling service. `http://www.ohse.de/uwe/uschedule.html`, 2007.

[Ope] OpenSSL. `http://www.openssl.org`.

[PRS+03] T. Prante, C. Röcker, N. Streitz, R. Stenzel, C. Magerkurth, D. van Alphen, and D. Plewe. Hello.wall - beyond ambient displays. In *Video and Adjunct Proceedings of UbiComp 2003*, October 2003.

[Rut01] Daniel Rutter. Webcam comparison: Logitech quickcam pro 3000, logitech quickcam express, lifeview robocam. `http://www.dansdata.com/webcams.htm`, 2001.

[Sch99] Wolfgang Schneider. Prototyping in der software-entwicklung. `http://www.ergo-online.de/site.aspx?url=html/software/software_entwicklung_prototyp/protot_konzept.htm`, 1999.

[Sch06] School of Engineering and Applied Sciences, Harvard University. *CodeBlue: Wireless Sensor Networks for Medical Care*, 2006. `http://www.eecs.harvard.edu/~mdw/proj/codeblue/`.

[SCL+05] Victor Shnayder, Bor-Rong Chen, Konrad Lorincz, Thaddeus R. F. Fulford-Jones, and Matt Welsh. Sensor networks for medical care. Technical report, Division of Engineering and Applied Sciences, New York, NY, USA, 2005.

[SFGB07] Hong Sun, Vincenzo De Florio, Ning Gui, and Chris Blondia. Participant: A new concept for optimally assisting the elder people. In *CBMS '07: Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems*, pages 295–300, Washington, DC, USA, 2007. IEEE Computer Society.

[SH06] B. Skov and Th. Hoegh. Supporting information access in a hospital ward by a context-aware mobile electronic patient record. *Personal Ubiquitous Comput.*, 10(4):205–214, 2006.

[SKLS05] Hyungjoo Song, Daeyoung Kim, Kangwoo Lee, and Jongwoo Sung. Upnp based sensor network management architecture. In *Proc. ICMU Conference*, 2005.

[Smi07] Smiths Medical PM, Inc. *BCI-3044 Pulse Oximeter Finger Sensor*, 2007. `http://www.smiths-medical.com/catalog/oximeters/handheld/3301/bci-3301-hand-held.html`.

[SOP+04] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks, 2004.

[Sta98] William Stallings. Snmpv3: A security enhancement to snmp. *IEEE Communications Surveys and Tutorials*, 1(1), 1998.

[Sub99] Mani Subramanian. *Network Management: An Introduction to Principles and Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[Sun07] Sun Microsystems, Inc. Generating a movie file from a list of (jpeg) images. `http://java.sun.com/products/java-media/jmf/2.1.1/solutions/JpegImagesToMovie.html`, 2007.

[Tra02] Jon Travis. Camserv, streaming video server. `http://cserv.sourceforge.net/`, 2002.

[Vid08] VideoSurveillance.com. Video surveillance for hospitals and medical facilities. `http://www.videosurveillance.com/hospital.asp`, 2008.

[WKfC+04] Ronald Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter 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*, pages 59–64, New York, NY, USA, 2004. ACM.

[www07] www.handhelds.org. Handhelds.org - open source for handheld devices. `http://www.handhelds.org/moin/moin.cgi/Ipkg`, 2007.

[Ylo96] T. Ylonen. SSH - secure login connections over the internet. Proceedings of the 6th Security Symposium) (USENIX Association: Berkeley, CA):37, 1996.

[Yon03] James Yonan. User-space vpn: History, conceptual foundations, and practical usage. `http://openvpn.net/papers/BLUG-talk/BLUG-talk.ppt`, 2003.

Appendix A.

ReMoteCare MIB file

This appendix shows the complete MIB file used with the SNMP proxy.

```
CodeBlue-MIB DEFINITIONS ::= BEGIN

IMPORTS
     IpAddress
          FROM RFC1155-SMI
     OBJECT-TYPE
          FROM RFC-1212;

codeBlue OBJECT IDENTIFIER
     -- 1.3.6.1.3.1
     ::= { 1  3  6  1  3  1 }

-- groups in CodeBlue

stargateUrl OBJECT-TYPE
     SYNTAX  IpAddress
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION ""
     -- 1.3.6.1.3.1.1
     ::= { codeBlue 1 }

serverIp OBJECT-TYPE
     SYNTAX  IpAddress
     ACCESS  read-only
     STATUS  mandatory
```

```
     DESCRIPTION ""
     -- 1.3.6.1.3.1.2
     ::= { codeBlue 2 }

-- list of motes

medicalTable OBJECT-TYPE
     SYNTAX  SEQUENCE OF MedicalEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION
          "A list of Motes."
     -- 1.3.6.1.3.1.3
     ::= { codeBlue 3 }

medicalEntry OBJECT-TYPE
     SYNTAX  MedicalEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION
          "A mote entry for a particular mote."
     INDEX {
          moteID }
     -- 1.3.6.1.3.1.3.1
     ::= { medicalTable 1 }

MedicalEntry ::= SEQUENCE {

     moteID      INTEGER,
     pulse       INTEGER,
     oxigen      INTEGER,
     gte         INTEGER,
     energy      INTEGER,
     ekg         INTEGER,
     light       INTEGER,
     temperature INTEGER }
```

```
moteID OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "Mote ID."
     -- 1.3.6.1.3.1.3.1.1
     ::= { medicalEntry 1 }

--*****************************************
-- entries of medical values
--*****************************************

pulse OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "Heartbeat rate of the patient."
     -- 1.3.6.1.3.1.3.1.2
     ::= { medicalEntry 2 }

oxigen OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "Oxygen saturation."
     -- 1.3.6.1.3.1.3.1.3
     ::= { medicalEntry 3 }

gte OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "whatever."
```

```
     -- 1.3.6.1.3.1.3.1.4
     ::= { medicalEntry 4 }

energy OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "Some kind of energy"
     -- 1.3.6.1.3.1.3.1.5
     ::= { medicalEntry 5 }

ekg OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "EKG values"
     -- 1.3.6.1.3.1.3.1.6
     ::= { medicalEntry 6 }

light OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "light"
     -- 1.3.6.1.3.1.3.1.7
     ::= { medicalEntry 7 }

temperature OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          "temperature"
     -- 1.3.6.1.3.1.3.1.8
```

```
     ::= { medicalEntry 8 }

mgmtTable OBJECT-TYPE
     SYNTAX  SEQUENCE OF MgmtEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION ""
     -- 1.3.6.1.3.1.4
     ::= { codeBlue 4 }

mgmtEntry OBJECT-TYPE
     SYNTAX  MgmtEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION ""
     INDEX {
          mgmtID }
     -- 1.3.6.1.3.1.4.1
     ::= { mgmtTable 1 }

MgmtEntry ::= SEQUENCE {

     mgmtID        INTEGER,
     signalStrenth INTEGER,
     hopCount      INTEGER }

mgmtID OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          ""
     -- 1.3.6.1.3.1.4.1.1
     ::= { mgmtEntry 1 }

signalStrenth OBJECT-TYPE
     SYNTAX  INTEGER
```

```
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          ""
     -- 1.3.6.1.3.1.4.1.2
     ::= { mgmtEntry 2 }

hopCount OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-only
     STATUS  mandatory
     DESCRIPTION
          ""
     -- 1.3.6.1.3.1.4.1.3
     ::= { mgmtEntry 3 }

END
```

Appendix B.

Camserv config file

This appendix shows the complete config file for the camserv application.

```
# video_basic: The 'basic' color-changing video module.
[video_basic]
path            /usr/local/lib/camserv/libvideo_basic.so.0
device_path     /dev/v4l/video0

# video_v4l_bttv:  Example of a common BTTV module for video4linux.
# port 0 == TV, port 1 = Composite 1, port 2 = Composite 2
# frequency == is the channel frequency for the TV
# autobright == 0 turns off autobrightness adjusting, otherwise it adjusts
#               the brightness of the picture every 'autobright' frames.
# brightmean == The mean pixel value that is the 'goal' of the autobright.
#               (0->255)
# brightx1->y2 == (x1,y1) top left coords, and (x2,y2) bottom right coords
#                 of a rectangle of which to take the average pixel value.
#                 this is then used in calculating the mean to adjust the
#                 brightness of the image.
# mode == the video norm to use:  0 == PAL, 1 == NTSC, 2 == SECAM, 3 == AUTO
# color,hue,contrast,brightness,whiteness = 0->60000, representing
# the value of each component.

[video_v4l_bttv]
path            /usr/local/lib/camserv/libvideo_v4l.so.0
device_path     /dev/v4l/video0
port            0
mode            3
#frequency      74.43
color           30000
hue             30000
contrast        30000
brightness      30000
whiteness       30000
```

```
autobright      1
brightmean      128
brightx1        0
brighty1        320
brightx2        0
brighty2        240

# FreeBSD BTTV driver:
# port 0  = Video
#      1  = Tuner
# Channel Sets:
# nabscst   1
# cableirc  2
# cablehrc  3
# weurope   4
# jpnbcst   5
# jpncable  6
# xussr     7
# australia 8

[video_fbsd_bttv]
path            /usr/local/lib/camserv/libvideo_fbsd_bttv.so.0
device_path     /dev/v4l/video0
port            1
width           320
height          240
autobright      100
#brightness     0
#chroma         180
#contrast       1000
channelset      2
channel         60

[video_v4l_qcam]
path            /usr/local/lib/camserv/libvideo_v4l.so.0
device_path     /dev/v4l/video0
width           320
height          240
port            0
color           30000
hue             30000
contrast        30000
brightness      30000
whiteness       30000
autobright      0
```

```
[jpg_filter]
path            /usr/local/lib/camserv/libjpg_filter.so.0
quality         100

# text_filters:  Text filters are used to provide text on your webcam
# bg,fg          == #RRGGBB if RGB camera, #CC if B&W camera,else 'transparent'
# x,y            == Coordinates for the text
# mangle         == 0 turns off mangling of the 'text', 1 turns it on
# text           == Text to display.  If mangling == 1, special metachars
#                   such as '%X' will be expanded -- see date(1) for lots of
#                   flags
# fontname       == 6x11 or 8x8 for the fontsize.

[hello_world_banner]
path            /usr/local/lib/camserv/libtext_filter.so.0
text            Camera 1, Floor 4
bg              #000000
fg              #ffffff
x               0
y               0
mangle          0
fontname        6x11

[time_stamp]
path    /usr/local/lib/camserv/libtext_filter.so.0
text            Time: %X
bg              #000000
fg              #ffffff
x               0
y               230
mangle          1
fontname        8x8

[static_filter]
path            /usr/local/lib/camserv/librand_filter.so.0
num_perline     20
coloredpix      0

#
# You can add the imlib2_filter to your filters list to display pictures
# over your own, or small regions, or whatever your heart desires.
[imlib2_filter]
path            /usr/local/lib/camserv/libimlib2_filter.so.0
```

```
file            /tmp/my_nasty_picture.png
x               0
y               0

########################################
#  Begin Fixed Sections                #
########################################

# socket parameters:
#       listen_port = port the camserv program listens on
#       max_frames  = Maximum # of frames to send to the client before
#                     closing the connection (0 disables)
#       max_bytes   = Maximum # of bytes to send to a client before
#                     closing the connection (0 disables)
#       max_seconds = Maximum # of seconds a client can be connected before
#                     being closed (0 disables)

[socket]
listen_port             9192
max_frames              0
max_bytes               0
max_seconds             0

[filters]
num_filters             3
filter0_section         time_stamp
filter1_section         hello_world_banner
filter2_section         jpg_filter


# [video] - This section is devoted to all things dealing with the pictures
#           taken by the input video module.  These are general things which
#           should be used by all video modules.
#
#           IMPORTANT:  If you are seeing cycling colours instead of the
#                       video for your camera, you need to change video_basic
#                       to video_v4l_qcam or video_v4l_bttv

[video]
video_section           video_v4l_qcam
width     320
height    240
maxfps    0
memhack   1
```

```
[main]
# To do a single time invocation of the output from the camserv,
# use output_snapfile which designates the output location, and
# output_presnaps to take a number of pictures before finally outputting
# the final image.
#output_snapfile                foo.jpg
#output_presnaps                100
```