Design of an energy efficient body area network based on Bluetooth 4.0 low energy for medical applications

by

Rene Ruck, B.Sc.

Thesis

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

August 2013

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Rene Ruck

August 25, 2013

Acknowledgments

Give many thanks and stuff...

Rene Ruck

University of Dublin, Trinity College
August 2013

Design of an energy efficient body area network based on Bluetooth 4.0 low energy for medical applications

Rene Ruck, M.Sc.
University of Dublin, Trinity College, 2013

Supervisor: Meriel Huggard

The major task of medical personnel and institutions in the present is dedicated to treat people in the case of emergency or sickness. This system will have to face drastic changes due to the shift towards an increasingly ageing society in the future. This development requires a shift towards a more proactive and monitored approach. With improvements in miniaturization and wireless communication technology medical monitoring gets more comfortable for patients as well as medical staff.

With a continuously increasing number of smart phones owned by individuals, limited analysis and diagnosis is not reserved to medically trained persons any more. Current mobile technologies are still heavily battery and lifetime constrained. Context awareness can help optimising workload and communication. In turn this optimisation can improve battery lifetime. The purpose of this dissertation is the development of an approach for intelligent wireless communication based on Bluetooth 4.0 low energy. The targeted context for this approach will be in Body Area Networks (BAN).

This dissertation proposed a way for energy efficient wireless communication achieved by situation adaptive change of connection parameter for Bluetooth 4.0 low energy connections. The first design was based on a connectionless approach utilising the Bluetooth low energy advertising mode. Implementing this idea proved out of scope for this dissertation. Based on a connection based communication a dedicated GATT service to configure the behaviour of embedded sensor devices was defined. The implementation of an embedded prototype included the dedicated GATT service as well as adaptive behaviour to detected outlier in the measured sensor data. An Android application as central data sink and controlling entity handles transmitted data and enabled the user to manually configure connection parameter.

Since the focus of this work was on energy efficient communication with adaptive connection parameter, the measured data did not consist of data with medical relevance. Also the analysing task to determine outstanding measurements did not involve medical knowledge or expertise.

Contents

Chapter 1

Introduction

Health is of every ones concern. Treating people in the case of sickness or emergency is currently the major task of medical personnel and institutions. This system will have to face drastic changes due to the shift towards an increasingly ageing society in the future. Following the Organisation for Economic Co-operation and Development (OECD)s regular world health report, the cost for health care has drastically increased over the last decade [31].

The current development requires a shift towards more a more proactive and monitored practice. Technology can help improve medical surveillance and shorten reaction times in cases of health critical situations. Improvements in miniaturization and wireless communication for medical devices, medical monitoring gets more comfortable for patients as well as medical staff. Wireless communication enables patients to freely move, even go home or continue their daily duties while medical parameter are monitored.

With a continuously increasing number of smart phones owned by individuals, limited analysis and diagnosis is not reserved to medically trained persons any more. With this pervasive mobile technology and ubiquitously usable sensors personal medical monitoring can blend into every ones daily technological surroundings. More diverse long term medical observation-data will yield better insights into different medical and social aspects.

Current mobile technologies are still heavily battery and lifetime constrained. With the help of the increasing number of sensors, mobile handsets are equipped with, context awareness starts entering the world of mobile applications. Context awareness can help optimising workload and communication. In turn this optimisation can improve battery lifetime. The purpose of this dissertation is the development of an approach for intelligent wireless communication based on Bluetooth 4.0 low energy. The targeted context for this approach will be in Body Area Networks (BAN).

Chapter 2

State of the Art

The first part of this chapter will be dedicated to an overview of current examples of wireless sensors in mobile health care. The overview will distinguish between wearable and implanted sensors. Within these two classifications the examples given are grouped according to the type of data they measure. The second part discusses wireless communication frequencies and techniques. Followed by chapter about security and privacy for medical data. The following chapter will introduce Bluetooth low energy and its special properties. The last part will consider Embedded prototyping boards and Bluetooth low energy enabled handsets.

2.1 Body area network

Body area networks (BAN), also called wireless body are networks (WBAN) or body sensor networks (BSN), are networks of smart miniaturized devices (motes) that are able to sense, process and communicate. These devices are designed to be small in size and battery powered so that they are wearable or implantable. These size and operational constraints bring them a inherent limitation on power consumption. Motes can be interconnected via a central processing unit that coordinates the network or can be organized into a self managing mesh network. The task of each individual mote can differ widely depending on their purpose. The main goal is most typically the collection of physical or environmental data. The rather limited connectivity of individual motes can be extended using single/central units with extended wireless capabilities as gateways to forward acquired data. These gateway units are usually not as resource constrained as the rest of the motes in the network. For example they could be mobile handsets or a dedicated devices with extended battery capabilities.

WBAN technology was first mooted in 1995 [REF] focusing on the idea of using wireless personal area network (WPAN) technologies to implement communications near and around the human body. With the advancing of miniaturizing sensors and circuits, WBAN were quickly adopted for medical applications. In the following section an overview of the past and current development in wearable sensors and sensor networks for medical applications is provided.

2.1.1 Wearable Sensors

The first type of sensors considered is the *pulse oximeter*. A pulse oximeter is a medical device that indirectly measures the oxygen saturation levels(SpO2) in an individual's blood, as well as the changes in blood volume in the skin. Typically, a pulse oximeter is attached to a finger or an earlobe, and it consists of red and infra-red light-emitting diodes (LEDs) and a photodetector. Changes in the absorption rate of the skin can be reasoned to the actual blood pressure.

A wearable photoplethysmogram (PPG) sensor ring is described in [45]. As a ring this sensor is more comfortably to wear and is more likely to be worn over for longer monitoring periods. This approach has been refined [4] where the resilience of the sensor to motion as well as changes in ambient light is increased. To reduce the power consumption a high frequency, low duty modulation scheme was applied.

A group at Harvard University [37] combined a BCI micro power oximeter with a variety of wireless sensor platforms. The resulting wireless sensor serves as part of the larger medical sensor network, *CodeBlue*. A pulse oximeter was also developed [3] as part of the multi node medical monitoring system AMON (Advanced care and alert portable telemedical monitor).

A good overview of wearable bio-sensor-systems of multi-parameter physiological detection systems is provided in [32]. While wearable sensor networks for the visually impaired formed the focus of [11].

Another node for the medical sensor network *CodeBlue*, [15], integrated an ECG sensor into a Micra2 mote. This sensor only uses two electrodes to capture an ECG signal. There is also an EGC sensor available for the AMON sensor platform. The Human++ project involves a single-lead ECG sensor node [34]. A wireless three pad system that can synthesise a conventional 12-lead ECG signal from the three available inputs has been developed [8, 9].

The AMON platform mentioned above provides a blood pressure sensor that uses an inflating cuff around the patient's wrist to get blood pressure readings. This method is limited to single measurements and the inflated cuff caused discomfort to some users. These problems have been addressed in [44]. They created a BF watch sensor that works without an inflating element. The pulse transmit time (PPT) method is used to measure the blood pressure.

Activity/Motion detection is another area which integrates the use of an accelerometer and gyroscope to determine the amount of movement, or lack of, a body or part of a body. An example for a wireless sensor node using an accelerometer is the BodyANT system [24]. It revealed it's potential usefulness in the monitoring of uncontrolled body movements that can happen to Parkinson's patients under medical treatment.

The area of Electroencephalography (EEG) also provides great opportunities to improve comfort and usability of long term EEG monitoring. Recording sessions are usually limited to 20 minutes. This often means that significant data is not recorded if events take place outside of this 20 minute window. Hence lighter and more comfortable long term monitoring solutions can be of great benefit. An example of such a system is using their Wireless Intelligent SEnsor (WISE) system to record EEG signals [22], ther examples include [14, 13]. They also developed a wireless sensor system based on the Mica2 platform that is capable of acquiring 2 channels of EEG data.

2.1.2 Implantable Sensors

Wireless Endoscopic Capsules

As far back as 1981 the feasibility of wirelessly communicating capsules for diagnostic and therapeutic purposes has been discussed. Limitations in available technology hampered the realization of this idea. In the 1990s then experiments with a first prototype were performed [7]. With improvements in miniaturization and imaging technologies, such as the CMOS image sensor, made this idea more feasible. In 2000 Given Imaging (Yoqneam, Israel) presented the first generation of commercially available wireless endoscopic capsules (WEC). The M2A^{TM} was the first WEC for investigating the small bowel [28]. With a size of 26x11mm it is easy to swallow and immune to the abrasive environment in the stomach and small intestine. Figure 2.1 shows a set of commercially available endoscopic capsules.

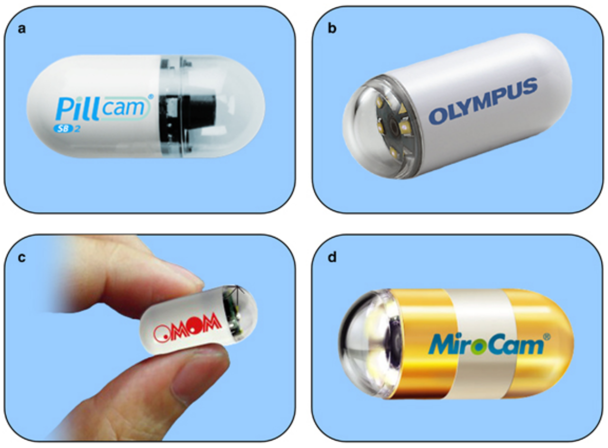


Figure 2.1: Wireless capsule endoscopes (a) PillCam SB2 (Given Imaging), (b) Endo Capsule (Olympus America), (c) OMOM (Jinshan Science and Technology), (d) MiroCam (IntroMedic)

A complete capsule should consist of the following six parts: locomotion, vision, telemetry, localization, energy and manipulation tools. The need to adapt to the different environments of esophagus, small bowel or colon, currently available products only include a subset of this modules.
For example the PillCam^{TM} SB2 (Given Imaging) is equipped with a camera (CMOS) on one end and takes pictures with a resolution of 256x256 pixel. The camera captures 2 images per second illuminated by 6 led. The built in battery time is stated as 9 hours. After ingestion the capsule is carried through the intestine passively via peristalsis and randomly captures images of the lumen wall with a view angle of 156 degree. As described below in 2.2.2 the PillCam^{TM} SB2 communicates with external devices using the IMS band of around 433Mhz. Since the capsule's main purpose is diagnosis it does not incorporate manipulation tools.
A comprehensive overview of current endoscopic capsules and their technical specifications can be found in Table 2.1.

2.2 Wireless Communication Technologies

The following chapter will give an overview of the wireless technologies important for this work. Starting with an overview of wireless short-range, low-power communication technologies. The second part will focus on Bluetooth and and its newest derivative Bluetooth 4.0 low energy (BLE).

Product	Company	Camera	View angle	Capture frequency	Battery time
PillCam^{TM}SB	Given Imaging	256x256px (CMOS)	140°	2 fps	8h
PillCam^{TM}SB2	Given Imaging	256x256px (CMOS)	156°	2 fps	9h
PillCam^{TM}ESO	Given Imaging	2x 256x256px (CMOS)	140°	14 fps	20min
PillCam^{TM}ESO2	Given Imaging	2 x 256x256px (CMOS)	169°	18 fps	20min
PillCam^{TM}COLON	Given Imaging	2 x 256x256px (CMOS)	156°	4 fps	10h
PillCam^{TM}COLON2	Given Imaging	2 x 256x256px (CMOS)	156°	4-35 fps	10h
EndoCapsule^{TM}	Olympus	192x1080px (CCD)	145°	2 fps	8-10h
MiroCam^{TM}	IntroMedic	320x320px (CMOS)	150°	3 fps	11h
OMOM	Jinshan	640x480px (CMOS)	140°	2	7-9h

Table 2.1: Overview of currently available capsule endoscopes and their important characteristics

2.2.1 Wireless Frequencies

The wide variety of available, unlicensed frequencies is given in Table 2.2. It can be seen that these are spread across the spectrum from 13 MHz to 5 GHz. The current research and development listed in Table 2.3 yields a great variety of used wireless frequencies. Most of them are placed in the Industrial, Scientific and Medical (ISM) or the Medical Implant Communication Service (MICS) radio band. These are not licensed and available for public use. Some of these bands (e.g. 2.4 GHz) are internationally available, others are restricted to specific global regions. The choice of the right transmission frequency is crucial for the successful use in WEC.

The ISM bands spans a range from 6 MHz up to 244 Ghz. Low frequencies of around 13MHz are often used in short range communication e.g. for RFID or inductive systems. The available bandwidth at those low frequencies is only a few KHz making it difficult to establish a reliable and resilient data connection. Higher frequencies should be used to increase both the range and available data-rate. Therefore the MICS band includes frequencies between 401 and 406MHz intended to use in medical implants. It was initially defined and standardised in 1999 by the Federal Communications Commission (FCC) in the U.S. and later, in 2003, in Europe by the European Telecommunications Standards Institute (ETSI). Its intended use is to provide a high level of comfort, better mobility and better patient care while providing mobility device telemetry [6].

The Wireless Medical Telemetry Service (WMTS) defined by the FCC is only used in the U.S. The bands set aside for WMTS are 608-614 MHz, 1395-1400 MHz and 1427-1432 MHz. The available ISM band for Europe in this range is located between 902 -

928 MHz.
Even higher frequencies provide better bandwidth and range but due to incompatible size, power and radiation restrictions are not suitable for the use in ingested systems.

Name	Frequency	Bandwdth	Country/Region	Current Projects
MICS	402–405 MHz	300 KHz	U.S., AU, EU, JP	PillCam (Given Imaging)
WMTS	608–614 MHz	1.5 MHz	U.S., Canada	None
	1395–1400 MHz	5-6 MHz		
	1427–1432 MHz			
High Freq. ISM	2400–2483 MHz	20 MHz	Worldwide	[40]
	5725–5875 MHz	40 MHz		
Mid-ISM	433–434 MHz	several kHz	Worldwide	[10], [39]
	315 MHz			[46]
	865–868 MHz	200–500 kHz	Europe	
	902–928 MHz	several MHz	U.S., CA, AU	[46]
Low ISM	13.55–13.567 MHz	several kHz	Worldwide	none
	26.95–27.283 MHz	14 kHz		
	40.66–40.70 MHz	13 MHz		

Table 2.2: Overview of unlicensed wireless communication bands

2.2.2 Wireless Communication in Pill Cameras

Due to their self-imposed size restrictions electronic pills have limited capacity to store captured images. Hence images need to be transferred wirelessly to a receiver or workstation with a higher storage capacity. Initially low frequencies and simple constructs where used. Sensors communicated with basic transmitter using Colpitts or Hartley circuits, to transfering data from within the body to external receivers. Monitoring of organs with simple values like temperature, pH or pressure was possible. However, the possibilities were limited due to the capabilities of existing insufficient electronic components.

More recent approaches have worked with quite low frequencies (UHF-433 IMS or lower). These are attractive due to their efficiency and the broad availability of components, that operate at these frequencies. Higher frequencies suffer from the high attenuation of the human body. There are plenty of components available for higher IMS frequencies: ZigBee (IEEE 802.15.4), Wi-Fi (IEEE 802.11) and Bluetooth (IEEE 802.15.1) all operate at 2.4 GHz. The major disadvantage of working at these high frequency is that devices start interfering with each other when located in the same environment [36].

In [33], Park, et al. use simple AM (amplitude modulation) for telemetric communication. Their design includes a mixer and an oscillator circuit together with a CMOS image sensor and a loop antenna to form a capsule-shaped telemetry device. To control the device and receive video data two independent communication channels have been used. Control signals are transmitted on a carrier frequency of 433 MHz and the video signal is modulated onto 315 MHz. A goal of this devices was to show that it is possible to handle two independent communication channels in an integrated circuit.

In 2009 [38] Valdastin *et al.* realized an approach to achieve data rates of around 2Mbps for high resolution images. They successfully demonstrated a transfer rate of 15-20fps with the help of image compression algorithms. A simple Colpitts circuit operating at 144MHz was used. The low frequency employed that has the disadvantage that it requires a larger antenna which again increases the size of the whole construction.

Improved diagnostics necessitate the transfer of higher resolution images. It has been successfully shown [47] that higher resolution images can communicate using an ultra wide band (UWB) frequency that achieves a transmission range in excess of 50cm. The advantages of this system is the attainment of data rates of 100Mbit/s and more enabling the use of higher resolution cameras as higher image transfer rates can be achieved.

2.2.3 Wireless communication in commercially available products

Commercially available products, like the one from Given Imaging, represent the state-of-the-art in current wireless technologies for capsules. Given Imaging's Pillcam SB uses a proprietary Zarlink RF module for communication at a frequency of 401-406Mhz [6] and provides a data rate of 800kbps. Frequency-Shift-Keying (FSK) modulation is used.

Furthermore there are commercially available endoscopic capsules from Olympus (EndoCapsule) and RF System Labs (Norika). In both their cases there is no information available about the communication chip set or frequencies used. However it can be assumed that these are ranged within the IMS or MICS bands, from 400 to 433Mhz.

Project	Image Sensor	Frequency	Data Rate	Modulation
Park 2002 [33]	OV7010 CMOS (510 x 492 px)	315/433 MHz	NA	AM
Valdastri2004 [39]	Multichannel (-)	433 MHz	13 kb/s	ASK
Kfouri2008 [23]	ICX228AL CCD (758 x 494 px)	UHF	250 kb/s	NA
Chen2009 [10]	VGA (307 x 200 px)	433 MHz	267 kb/s	NA
Thone2009 [38]	MT9V013 VGA (640 x 480 px)	144 MHz	2 Mbps	FSK

Table 2.3: Overview of Endoscipic Capules in research

2.2.4 Bluetooth

Bluetooth is a short range communication standard for exchanging data within the ISM band from 2400–2480 MHz. Bluetooth is used to span PANs between stationary and mobile devices. Designed by the telecommunication vendor Ericsson in 1994 it was initially intended to provide a high level of security. Bluetooth was originally defined in the IEEE 802.15.1 standard, but this standard is no longer maintained. The Bluetooth Special Interest Group (SIG) consisting of around 18,000 member companies out of the areas of telecommunication, computing, networking, and consumer electronics, maintains the specification and guides its further development.

Bluetooth divides the ISM band from 2400 to 2483.5 MHz into 79 channels, each 1 MHz wide. To avoid collisions with other devices within a frequency band Bluetooth implements Adaptive Frequency-Hopping (AFH). At the time of designing Bluetooth 1.0 the only modulation scheme available was Gaussian frequency-shift keying (GFSK) which is still used today. With the advert of newer modulation techniques the more recent Bluetooth specifications include these. These define the extensions of Bluetooth 2.0 with (EDR) which stands for Enhanced Data Rate and implements 4-DPSK and 8-DPSK schemes. The resulting transfer rates are given in Table 2.4.

Bluetooth has as a master-slave structure whereby a master can maintain up to seven simultaneous connections. The protocol allows the devices to switch roles on agreement. All connected devices listen to the master's clock and all packet exchange is based on this central clock. This clock defines ticks at 312.5 μs intervals, two clock ticks make up a slot of 625 μs, while two slots make up a slot pair of 1250 μs. Transmitted packages can be 1, 3 or 5 slots long. When sending only single-slot packages the master transmits on even slots and receives on odd slots. The slave then will receive on even slots

Technology	Range (m)	Power cons.	Frequency Band	Data Rate
Bluetooth 1.2	$\leq$ 10m; up to 100m	$\leq$ 30mA	ISM 2.4 GHz	1 Mbit/s (0.7 Mbit/s)
V.2.0 + EDR	$\leq$ 10m; up to 100m	$\leq$ 30mA	ISM 2.4 GHz	3 Mbit/s (2.1 Mbit/s)
V.3.0 + HS	$\leq$ 10m; up to 100m	$\leq$ 30mA	ISM 2.4 GHz	24 Mbit/s
V.4.0	up to 50 m	$\leq$ 15mA	ISM 2.4 GHz	1 Mbit/s (0.27 Mbit/s)
ZigBee	$\leq$ 100m	$\leq$ 47mA	ISM 2.4 GHz	0.25 Mbit/s
Wifi	250 m	116 mA	IMS 2.4 GHz	866.7 Mbit/s

Table 2.4: Brief overview of available wireless communication technologies and the properties of relevance for this work

and send on odd ones. When transmitting larger packets the master will always start transmitting in an even slot.

Bluetooth devices can be divided into 3 classes depending on the transmitted signal strength. The effective range of a device varies due to propagation conditions, material coverage, production variations, antenna configurations and battery conditions. These are summarised in Table 2.5 below.

Class	Maximum permitted power		Range (m)
	(mW)	(dBm)	
Class 1	100	20	100
Class 2	2.5	4	10
Class 3	1	0	1

Table 2.5: Classes of Bluetooth devices, their permitted transmission power the resulting maximum range range

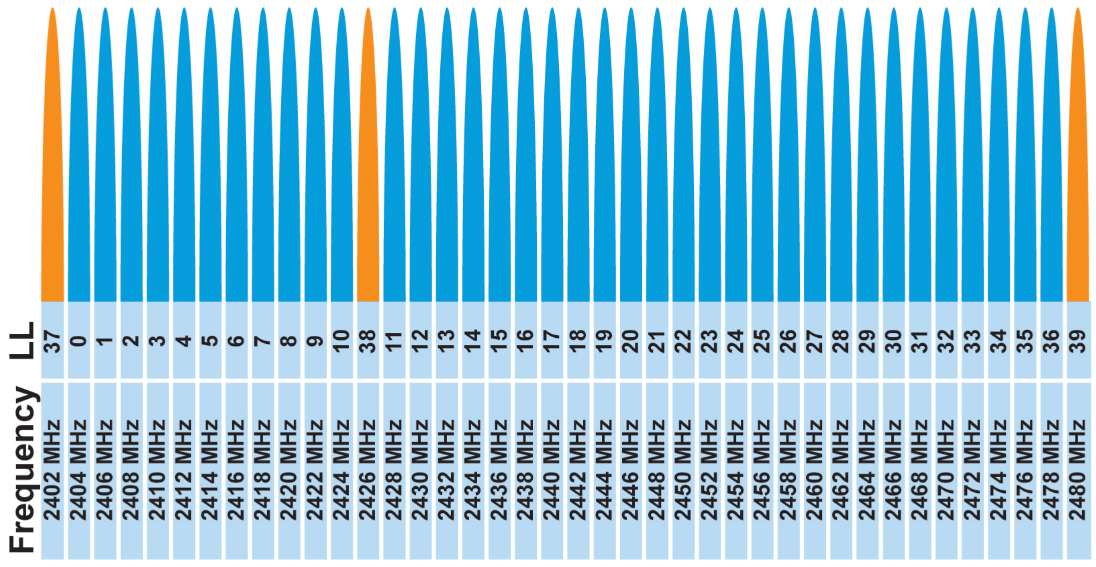
Bluetooth low energy (BLE)

Bluetooth low energy (BLE), originally introduced as WiBree, is a subset of Bluetooth v4.0 with a new protocol stack designed for the fast set up of simple connections. As a low energy alternative to the standard protocols from Bluetooth v1.0 to v3.0, BLE was designed to enable wireless connectivity for small devices running on a coin cell battery. Bluetooth low energy was introduced together with Bluetooth v4.0 in 2010.

BLE uses a GFSK modulation. This allows it to reuse many of the existing components of Bluetooth radios. As specified in IEEE 802.15.4 BLE works within the 2.4 GHz band using 40 channels over 2MHz. The RF output power is inherited from regular Bluetooth and goes up to +10bBm. Nevertheless in a body area network context an output power of around 0dBm might be totally suitable.

Bluetooth low energy divides the 2.4 GHz band into 40 channel of each 2 MHz. Figure 2.2 illustrates the allocation of Bluetooth low energy channels onto the 2.4 GHz band. Communication channels are coloured in blue, advertising channels in red.

To mitigate for interference in the quite crowded 2.4 GHz band BLE uses frequency hopping. Compared to regular Bluetooth, BLE dwells longer on each frequency. Within the available 40 channels there are three dedicated for advertising purposes to provide a way to discover nearby devices and services. The same channels will be used for transmitting connection requests and for connection parameter negotiation. After a connection is successfully initiated the regular communication channels will be used.



LL
Frequency
37 0 1 2 3 4 5 6 7 8 9 10 38 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 39
2402 MHz 2404 MHz 2406 MHz 2408 MHz 2410 MHz 2412 MHz 2414 MHz 2416 MHz 2418 MHz 2420 MHz 2422 MHz 2424 MHz 2426 MHz 2428 MHz 2430 MHz 2432 MHz 2434 MHz 2436 MHz 2438 MHz 2440 MHz 2442 MHz 2444 MHz 2446 MHz 2448 MHz 2450 MHz 2452 MHz 2454 MHz 2456 MHz 2458 MHz 2460 MHz 2462 MHz 2464 MHz 2466 MHz 2468 MHz 2470 MHz 2472 MHz 2474 MHz 2476 MHz 2478 MHz 2480 MHz

Figure 2.2: Bluetooth low energy channels. 37 communication channels(blue) and 3 advertising channels(red). Advertising channels are located in different parts of the spectrum to provide immunity against interference

Establishing a connection

A Bluetooth device can run in different modes, depending on the required functionality. These are advertising mode, scanning mode, master device mode or client device mode. A device in advertising mode will regularly send out information that advertises its provided services and will respond with further information if requested. Devices in scanning mode listen to advertising packets and request additional information if required.

The stack, Figure 2.3, is designed to provide the possibility to opt-out of functionality that is not required. For example an advertising only device can be stripped of any scanning and connection functionality to enable a radically smaller design.

To establish a connection, one device has to be in advertising mode and the other one in initiator mode. The initiator mode is quite similar to the listening mode with the main difference being that the device is intended to establish a connection. The initiating device listens for advertising packages from devices it wants to connect to and sends out connection requests if discovering a desired device. After successfully establishing a connection the initiating device switches into master device mode and the other one will be the client(slave) device. A client(slave) device can only maintain one connection at a time whereas a master device can hold as many connections as their system can manage. This enables client(slave) devices to be as efficient, in terms of energy consumption and hardware cost, as possible.

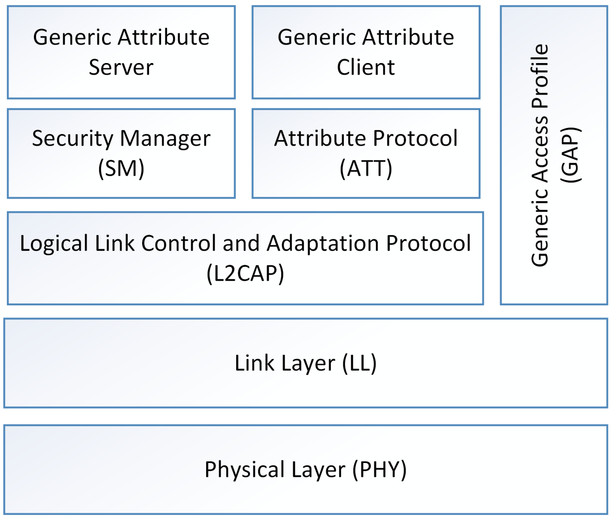


Figure 2.3: Bluetooth low energy stack

BLE packages

A BLE frame starts with a 1 byte preamble followed by 4 bytes with the access address. The access address consists of an access code combined with the used channel number. The actual payload (PDU) can be between 2 and 39 bytes. A 3 bytes CRC section comes at the end. Illustrated in 2.2.4.
On the advertising channel the payload (PDU) consists of a 2 bytes header and up to 37 bytes of actual payload, depending on the type of advertising package whereas a scanning device can request additional information which again can be up to 31 bytes. Hence a advertising device can distribute up to 68 byte of data before even establishing a connection.

The data channel payload (PDU) is consits of a 2 byte header and up to 37 bytes of actual payload. If the connection uses a link layer encryption a Message Integrity Checksum (MIC) with 4 bytes is attached at the end. The payload can carry link layer control information or application data. The expected payload can be defined in the header. The smallest package is of 10 bytes in size and the largest possible is 47 bytes. The frequency of advertising packets can be varied between 20ms and 10s. Advertising packages will be sent out on all available advertising channels. In contrast the scanning device can vary its scanning intervals as well as the time to scan per interval.

After a connection is established, the master device will transfer a collection of connection critical data to the client. These are the connection interval and the slave latency. The connection interval defines the time between connection events i.e. data package exchange sequence. The slave latency defines the number of communication intervals the client can ignore without loosing the connection. This enables client devices to further optimize their power consumption. A client can then request an update of these parameters which better suits its needs.

A communication event is always initiated by a master transmission. This serves as an anchor point to calculate the time for the next event. Within a communication event the master and slave alternate in the transmission packets. If one side stops sending packets the current event is considered to be finished. Each side has to wait until the next communication event before communication can resume.

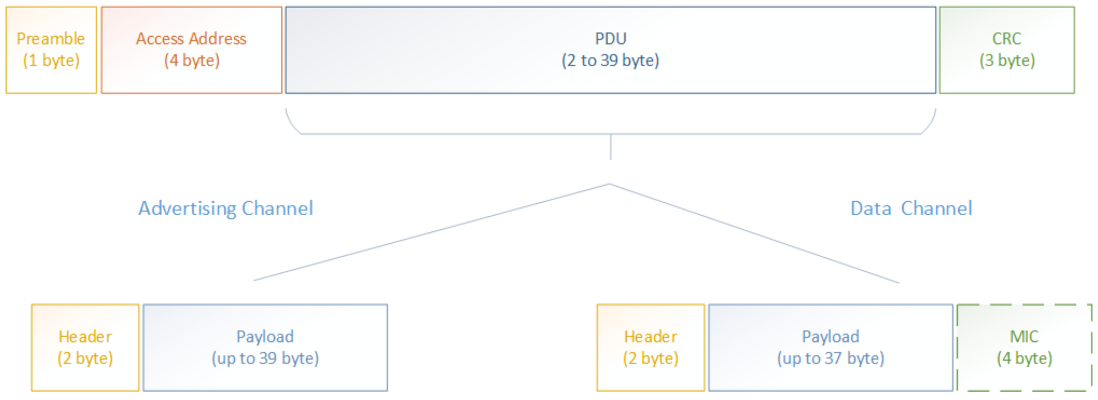


Figure 2.4: Bluetooth low energy over the air frame

2.3 Security and Privacy

Medical data is highly sensitive in nature. People have a natural concern about the security and privacy of their personal medical data. There is a great risk of abuse, e.g. identity theft when medical records get into the wrong hands.

This is why this topic has received much attention in the literature. Researchers have considered the question of secure communications from a personal medical device over a mobile mediator to an external endpoint [21]. They propose to split the secure communication into two parts. Part one contains the secure communication between the person and a mobile device and the second focuses on the secure communication between the mobile device and a "network access point or remote server". The communication from the person to the mobile device should be authenticated using biometrics. Fingerprints are well suited for this purpose and more accessible than "traditional passwords and PIN codes". The second stage of the communication should be secured by physical layer authentication and a Markov key exchange.

H. Alanazi *et al.* [1] and Y. Ren [35] provide a quite comprehensive overview and analysis of the literature including security factors, encryption techniques, patient rights, ethics and Electronic Medical Record systems. In their analysis of current legal guidelines together with available cryptographic techniques suitable for the usage in medical environments. The authors conclude:

> "In fact, encryption methods are efficient way to protect data."

but

> "...however, a system that guarantees confidentiality, integrity, authenticity, availability and non-repudiation has not appeared in the literature."

Followed by a comparison of cryptographic techniques and algorithms, their performance and applicability for securing medical data.

2.3.1 Security measures in commercially available products:

Encryption of transmitted data is not mentioned in the available documentation. Moreover, tear downs of available devices do not give any reason to expect the images and data that are transmitted from a Pillcam are encrypted in any way [27, 26]. Saving energy and valuable space within the tiny spatial constraints of the devices is most likely the reason for this.

2.4 Embedded Prototyping Boards

Embedded Prototyping boards are a convenient way to develop embedded systems without designing and manufacturing a circuit board with the required capabilities. These boards are normally already quipped with a small scale embedded processor (normally an ARM or ATMEL) and are capable of being extended in various ways. These extensions are called shields. Shields are embedded systems in their own right, but without a central processing unit. They extend the basic board with specific functionality like the ability to communicate via IEEE 802.11 (Wi-Fi) or nearly any other functionality that can be realized with an embedded system. Table 2.6 shows a very narrow overview of available development boards. A more extensive lost can be found in [43].

2.4.1 Arduino

Created in 2005 in Italy, Arduino is an open source hardware platform with a single 8-bit Atmel AVR micro controller. It has been designed to be easy accessible and usable especially by people without a background in software development. The available IDE (Integrated Development Environment) is derived from the 'ProcessingÂť programming language and the Wiring project. Programs for Arduino are written in C or C++. One of the very strong aspects of the Arduino platform is its extensibility. With circuit expansion boards called shields an Arduino can be extended to include any kind of functionality like GPS, LCD, Wi-Fi, Bluetooth, etc. Access to additional functionality is usually provided as software library with the extension board. This feature makes it flexible and suitable for many different kind of embedded projects.

Name	Processor	Flash/SRAM (kb)	Voltage	BLE
Arduino Uno	ATmega328P @ 16MHz	32 / 2	5 V	yes
Arduino Mega2560	ATmega2560 @ 16MHz	256 / 8	5 V	yes
Arduino Nano	ATmega328 @ 16MHz	16 / 1 or 32 2	5 V	yes
LilyPad Arduino	ATmega168V @ 16MHz	16 / 1	2.7-5.5 V	yes
BeagleBoard	TI OMAP3530 @ 720MHz	256 MB / 256 MB	28V	no
Tinkerforge (master)	ATSAM3S4C	256 / 48		no

Table 2.6: Short overview of embedded development boards

2.5 Bluetooth low energy enabled handsets

Prerequisite of using Bluetooth low energy with a mobile handset is the availability of a BLE enabled chipset in a device. Apples iPhone^{TM} is using BLE enabled chips since the iPhone^{TM}4s (released October 2011). All following iPhone^{TM} model are subsequently equipped with a BLE enabled chipset. Next to Apple devices several handsets by HTC and Samsung, running the Android operating system, are BLE enabled. Details can be found in table 2.7.

Vendor	Name	OS	Release date
HTC	HTC One	Android 4.1	April 2013
	One X+	Android 4.1	April 2012
	Droid DNA	Android 4.1	Dec 2012
Apple	iPhone 4s	iOS	Oct 2011
	iPhone 5	iOS	Sept 2012
	iPad (3rd, 4th gen)	iOS	March 2012
	iPad Mini	iOS	Nov 2012
Nokia	Lumia 620, 820, 920	Windows Phone 8	2012
Samsung	Galaxy SIII	Android 4.0	May 2012
	Galaxy S4	Android 4.2	Apr 2013
	Note II	Android 4.1	Sept 2012
Microsoft	Surface Pro	Windows 8	Oct 2012
Blackberry	Z10, Q10	BlackBerry 10	Jan 2013

Table 2.7: Bluetooth low energy enabled handsets

2.6 Android and its communication capabilities

Android is an operating system for smart phones and tablet computer. It is actively developed by Google and by the end 2012 it held a 75 percent of the global smart phone market[19]. Android stands out because of its open character. Google consequently publishes all source code for the Android operating system in the public domain. This makes Android popular amongst developers as they can relatively easy comprehend and adapt Android functionality to their own purposes. The Android operating system and framework is mainly written in Java and so are the applications (Apps) for Android.

Android supports a wide variety of communication standards and protocols, the protocols and standards listed below are far from complete and only comprise those relevant to the work described in this dissertation.

2.6.1 Communication protocols supported by Android

From the first public available release Android supported the communication standard IEEE 802.11 (Wi-Fi) and IEEE 802.15.1 (Bluetooth). With API level 12 (Android 3.1) the Android framework gained the ability to act as a USB host and communicate with or control attached USB devices. With the introduction of API level 14 (Android 4.0) Android has been extended to include the ability to use ISO/IEC 18092 / ECMA-340 (**N**ear **F**ield **C**ommunication) [12], Wi-Fi Direct [2] and can talk to Bluetooth devices with the CEN ISO/IEEE 11073 (Medical / health device communication standard) [20].
The available communication capabilities are constantly updated and adapted to new versions of standards. Today the current Android API 18 (Android 4.3) supports Bluetooth v4.0 low energy with a set of GATT profiles.

2.7 Summary

This chapter outlined an overview of the state of the art in body area networks for wireless medical monitoring technology and research. The second part gave an overview about wireless communication technologies. In this chapter wireless frequencies as well as wireless communication in Pill Cameras was discussed. Bluetooth and Bluetooth low energy were introduced separately. Since privacy and security are important topics, they were covered in this chapter as well. The last section focused on embedded prototyping boards, Bluetooth low energy enabled handsets and a review of Android's wireless communication capabilities.

Chapter 3

Design

This Chapter will cover the design and the process that led to the final design. Several important questions will be discussed. The first discussion will be about ways that influence the power consumption in mobile devices. The question of the placement of intelligence is discussed in the second part. This leads to the next important question which is about determining outlier in measurements. The first design approach discusses the utilisation of the Bluetooth low energy advertising mode. Several transmission strategies and parameter are discussed in the following chapter. The following discussion around connectionless and continuous connections will address a central issue in the design process. At last the resulting GATT protocol and security aspects will be discussed.

3.1 Ways to influence power consumption

As mentioned in chapter 2.1, single motes usually run on a battery and are therefore heavily power constrained. It is essential to the most out of the provided resources in terms of CPU cycles and battery lifetime. This chapter will discuss a number of the features of in a BAN system that influence the power consumption of a single mote as well as the whole system are reviewed.

3.2 Discussion where to locate the intelligence

The decision on where to analyse data and then make decisions on the basis of this data is of great impact onto the amount of power a device has to use. Next to the wireless communication the processing unit is the main power draw in a system. Therefore this decision has to be weighed up properly. There are two extremes available, first the intelligent sensor:

3.2.1 Intelligent sensor nodes

Intelligent sensor nodes process data and detect critical points in time independently and each note decides by them self when to rise the sampling rate. This also applies to the task of reducing the sampling rate after an emergency period with increased measuring frequency. The decision logic for a device depends on the individual use case the device is intended for and also the data it has to measure. A critical value in one use case e.g when measuring blood pressure, is probably not critical in another use case where values occur in a different range or order.
In the case of placing the decision logic into the embedded device, the mote permanently has to perform analysing tasks, hence that leaves less time for transmitting if we assume the sending of data as a blocking procedure. As well as measuring since every analysing task takes time and reduces the maximum measuring rate. But probably the most significant impact is the increased power requirement by the extended processor usage.
For a dataset consisting out of several motes, if these nodes act independently then this can have a negative impact on the homogeneity of the overall dataset. While some nodes may make the decision to increase the sampling rate, there could be a number of nodes that still sample on a lower sampling rate. This could distorts the resulting overall dataset and conclusions made. This behaviour is most useful if each node measures data for a different use case. In the case of many nodes measuring the same kind of data, the sampling rate should be homogeneous.
Embedded systems are typically constrained in their available memory (see Table 2.6). Analysing large datasets requires a not insignificant amount of memory, including both temporary memory as well as permanent storage.
An advantage of placing analysis and decision making on the node-side is the concomitant reduction in transmissions. The node buffers measurements without forwarding them. In this case the RF chip can stay in a sleep state. Next to this likely reduction in the value of transmissions is the significant higher reaction time. On device analysis means that reactions to eventual critical situations can be much faster then when the data has to be transmitted to the central device first for an analysis.

3.2.2 Stupid sensor nodes

A node designed to be as 'stupid' as possible would only measure data and forward these to a central device for analysis. Assuming a suitable connection exists. In this case the embedded device has only a small buffer that stores a minimal amount of data in case a connection is not available. In this scenario the central device aggregates all the collected information and analyses them. A central device like a smartphone is equipped with s a significant greater amount of memory and calculation power. Since embedded nodes do not have any self-regulation mechanisms, the central device instructs the embedded devices. These instructions can include changes in sampling rate or new connection parameter settings. These can also include a general configuration of its purpose, like the data to measure and the sensors to use.
Central controlling of nodes may lead to the gathering of a more homogeneous and coordinated dataset. Due to the larger memory available to the central device, decisions can be made based on a larger data set and therefore using much broader knowledge. Nevertheless analysing larger data sets requires more calculations and therefore increases the time until a reaction can be expected. Once the dataset grows too large, the power of a mobile handset is not sufficient and at this point it may be necessary to call on external data mining facilities.
For the embedded devices this scenario requires less calculation effort but the gains in terms of energy efficiency are hard to quantify. The possible energy savings due to increased transmission of new values should not be neglected, whereas a higher rate of transmission can have a negative impact on node lifetime. Nonetheless the difference in reaction time as described above is a factor to be taken into consideration.

3.2.3 Conclusion

Both of the extremes discussed above are not ideal. As always, a combination of both yields the best results. In this case a minimal amount of analysis on the embedded mote can provide an increase reaction time. Calculations over a broader set of measurements can be made on the handset side. This can then configure the embedded devices to optimise measurements in the long term. Depending on the memory capabilities of the embedded devices, buffering non-critical data can reduce the amount of transmissions needed and can increase the time the RF chip-set spends in sleeping mode.

3.3 Determining critical points in measurements

Determining when the measured data indicates critical behaviour is a central element in deciding on which transmission strategy to use. A balance between accuracy and power demand has to be found. Extensive calculations lead to better and more accurate results but at the expense of an increased drain of valuable energy.

3.3.1 Trend estimation

1. **Trend model:** The trend model is a statistical method to support the analysis of time series data. The traditional approach is the *Trend-Season-Model*. With this model an educated estimation of tendencies within the given dataset can be made, under consideration of seasonal anomalies.

2. **Moving Average:** A moving average is a kind of *finite impulse response filter* to analyse datasets. Therefore a new dataset will be created which consists of average values of equal sized subsets of the original dataset. This technique is used to analyse and smooth time series data. This process can also be applied in a weighted variation.

3.3.2 Outlier analysis

Another way of determining critical or interesting points within the data is with the help of Outlier analysis [REF]. An outlier is a measurement value that does not fit with the past or the expected future values. The expected value of a future measurement is normally defined using a scattering area around an expected value. The definition of the concept of an outlier:

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [18].

Outlier analysis is all about determining if a given value is an outlier and therefore a measurement error or a valid result. There are several approaches that all depend on the assumption that all values are normally distributed. Example for this approach is the *Grubb's test for outliers* [17], [16]. An alternative approach has been proposed by Walsh [42], [41] does not depend on the assumption that the data follows a normal distribution.

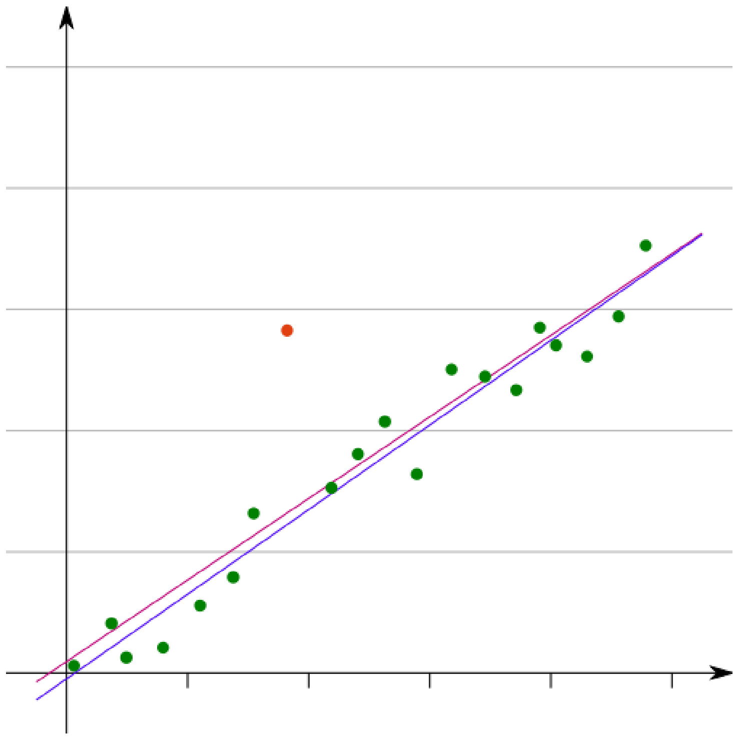


Figure 3.1: An outlier measurement. The blue regression line has been calculated without the outlier, the violet with.

Time Series Data and Data Streams

Time series data is a time dependent series of data values. Measurements can be taken at regular intervals (equidistant) or can be sampled at random intervals. The data collected at each point in time can consist of single or multiple values.

3.4 Utilising Bluetooth low energy advertising

The advertising mode is a legacy from former Bluetooth specifications. In Bluetooth low energy advertising is designed more flexible to provide additional ways for improving energy efficiency. Advertising periods can be between 20ms and 10 sec. As described in section 2.2.4 it is possible to transmit a decent amount of information already within the advertising packages. This is the point where the first idea for a design started.

Considering the amount of bytes that can be transmitted and the required memory for measurements it seems possible to utilise advertising packages to communicate measured data. Normally advertising package transport service definitions a device provides to the word. Replacing this data now with a minimum set of meta data and actual measurements, provides a way for omitting an actual connection. This connectionless approach removes the additional overhead of establishing and maintaining a connection between two devices.

3.5 Transmission parameter and strategies

As described in section 2.2.4 a Bluetooth low energy connection has several parameters that can be used to adjust and tune the connection. Other parameters include the sampling rate and the buffer size and these can be used to configure the behaviour of the embedded mote. Parameters of interest in this dissertation are:

1. **Connection Interval:** Interval in-between communication windows. Connection Interval requested, in multiple of 1.25 ms.
2. **Slave Latency:** Number of connection intervals that can be ignored by the slave.
3. **Sampling Rate:** Number of measurements taken per second, in Hz.
4. **Buffer Size:** Maximum amount of measurements that can be stored. This value can also serve as a threshold for the number of measurements to buffer before making the next transmission.

These parameters are all inter-related to each other. In an attempt to reduce the amount of information that has to be transferred, the following formulas connect the above parameters:

$$BufferSize(CI, SL, SR) = \frac{ConnectionInterval * SlaveLatency}{SamplingRate}$$

$$SamplingRateinHz(CI, BS, SL) = \frac{ConnectionInterval}{BufferSize} * SlaveLatency$$

Since *Buffer Size* and *Sampling Rate* can be calculated from the other given values, configuring the mote can be achieved by changing the connection parameter. The mote gets informed about the change of connection parameter like the *Connection Interval* and can calculate its resulting configuration using this new value.
There is no way provided by the Samsung BLE API, nor by the Android BLE API, to directly influence the *Connection Interval* or the *Slave Latency* from the Master/Android side. Hence the client/mote has to apply these changes. Therefore the master has to instruct the client/mote of the new connection parameter. This will only be possible within one of the connection windows defined by the combination of *Connection Interval* and *Slave Latency*. This is a definitive drawback of the implementation created in

this dissertation.

3.5.1 Transmission Strategies

Two general modes of transmission follow from the above discussion.However their use in a real-world implementation are strongly constrained by the available resources like hardware as well software components.

1. Buffered:
In order to extend sleeping periods to save energy, measurements are internally buffered and transmitted in bulk. Buffer size and sleeping time length are directly related, therefore they have to be matched to each other. Also the availability and amount of internal memory to keep buffered data depends on and is constrained by, the hardware platform used.

2. Real Time
The real time strategy aims to deliver measured data to an evaluating and displaying device as quick as possible. Every measurement will be directly transmitted. This results in a higher energy demand to serve the need for rapid data delivery.

3.6 Connect on demand vs. continuous connection

In the previous section several suggestions for possible connection strategies have been introduced. Deciding which one to implement leads to the general question of whether to use a continuous connection or to tear down the connection as soon as it is no longer required. To decide which approach might yield the better energy efficiency we have to have a look at the power consumption of the different components chosen to build the prototype.
These are on one side the Arduino Uno board and the nRF8001 Bluetooth Low Energy RF chip. A description about how to measure the power consumption of the nRF8001 has been provided [29]. There are also sample values listed for a heart rate monitoring use case which is similar to the targeted use case in this work. Table 3.1 shows the measured power consumption of the nRFF8001 chip in different phases of communication establishment and communication. From this we can get an idea of the power consumption to expect. Due to the voltage drop over the 10Ω resistor used, the current for I_{Idle} and I_{sleep} are too low to detect as they can be assumed to be 0.01mA or lower.
The specification states the input voltage is in the range of 1.9 - 3.6V. The Arduino Uno provides 3.3V to external devices.

Markers	Current	Time	Consumption
I_{Wakeup}	3.8mA	0.6ms	0.99mW
$I_{Standby}$	0.9mA	2.4ms	2.97mW
I_{RX}	13.5mA	0.6ms	44.55mW
I_{TX}	12.5mA	0.6ms	41.25mW
I_{MCULL}	3.5mA	1.2ms	11.55mW
$I_{MCUhost}$	5.0mA	1.9ms	16.50mW

Table 3.1: Typical power consumption of the nRF8001 BLE chip in a heart rate monitoring scenario

As described in section 2.2.4 a BLE device has to communicate every x ms with the master device. This every connection interval is given by the master device initially but can be changed by either party to suite their needs and optimise use. To provide opportunities to improve energy efficiency the slave device can ignore a certain number of these connection intervals and remain in a sleeping mode for longer.
It is hard to determine the power consumption of an Arduino Uno board since the board provides various ways to power it. Depending on the way it is powered, some components like the FTDI are activated or not. In the case of the FTDI it is assumed to be powered down if no USB connection if present. All experiments are supposed to run with the Arduino Uno connected via USB and outputting debugging information over the serial interface. Following the specification, USB provides between 4.75V and 5.25V (5V $\pm 5\%$).
To make an estimation easier only the power outlay of the Bluetooth low energy component will be considered.
The standard connection interval set up in the BLE shield uses is 997.5ms. This means we have a power consumption of 117.81 mW every 997.5 ms. This value can be improved by adjusting the connection interval as well as by increasing the slave latency.

3.6.1 Conclusion

Sending out advertising packets costs a lot of energy. Moreover the length of time spent advertising prior to connection establishment is unspecified. This can lead to long spikes of high power consumption. The energy cost of scanning on the handset side should also not be ignored. Bluetooth low energy can be optimised for sleeping states with very low energy consumption. This feature should be exploited. Hence in Bluetooth low energy a continuous connection with the right connection parameter is more efficient then one that continuously disconnects and reconnects.

3.7 Connection control protocol

To control the client a protocol, which defines the parameters, is required. Buffer Size, and hence the Sampling Rate, can be calculated every time the Connection Interval and the Slave Latency is set. In addition, either the desired Sampling Rate or Buffer Size has to be given. To communicate with a Bluetooth low energy device the GATT (Generic Attribute Profile) has to be used. A specific GATT profile consists of a number of *services* a device provides to the world. These services in turn consist of *Characteristics*. *Services* and *Characteristics* are uniquely defined by a 16- or 128-bit UUID. The specific UUID of each service will be combined with a base UUID. This base UUID can be either the Bluetooth SDP base UUID or a chipset manufacturer specific identifier [*Bluetooth Core Specification v4.0, Volume 3, Part G, 3.1cl*]

Bluetooth Base UUID: 00000000-0000-1000-8000-00805F9B34FB

$$128\_bit\_UUID = 16\_bit\_value * 2^{96} + Bluetooth\_Base\_UUID$$
$$128\_bit\_UUID = 32\_bit\_value * 2^{96} + Bluetooth\_Base\_UUID$$

Listing 3.1: Example for a typical service definition in xml

```
<Service Type="local" PrimaryService="true">
       <Name>ConnectionControl</Name>
       <Uuid>1111</Uuid>

         ...

       <Characteristic>
           <Name>SlaveLatency</Name>
           <Uuid>1113</Uuid>
           <DefaultValue>0</DefaultValue>
           <UsePresentationFormat>1</UsePresentationFormat>
           <UserDescription></UserDescription>
           <MaxDataLength>1</MaxDataLength>
           <AttributeLenType>1</AttributeLenType>
           <ForceOpen>false</ForceOpen>
           <Properties>
               <WriteWithoutResponse>false</WriteWithoutResponse>
               <Write>true</Write>
               <Notify>true</Notify>
               <Indicate>false</Indicate>
```

```
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="4" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>60000
               </PeriodForReadingThisCharacteristic>
            <PeriodForProperties>
                <Read>60000</Read>
                <Notify>60000</Notify>
            </PeriodForProperties>
        </Characteristic>

        ...

    </Service>
```

For our purpose the following field in the service and characteristic definition are important. These are *Name, Uuid, MaxDataLength and Write.*

- **Name:** The name is only of minor importance. This value translates into the pipe name used to address characteristic values on the embedded device.
- **Uuid:** Unique 16-bit UUID to identify this characteristic. This UUID will be combined with the Bluetooth base UUID as described above. This UUID also has to be unique to not to collide with already defined services and reserved name spaces for future definitions.
- **MaxDataLength:** This field constrains the maximum transferable data for this characteristic.
- **Write:** If this field is not set to *'trueÂt',* transferring data from a master device to the embedded client is impossible.

The, for the purpose of controlling the embedded client, a new GATT service is defined to provide the following characteristics.

1. **Connection Interval:** A more detailed description of this parameter in section 2.2.4. As described in section 3.5.1 the current implementation of the Android BLE API does not provide a direct way to manipulate the *Connection Interval* from the handset side. Therefore the handset has to inform the embedded device about desired connection changes.

2. **Slave Latency:** The *Slave Latency* is the same case as the *Connection Interval*.

3. **Sampling Rate:** This field defines the *Sampling Rate* for the embedded device. Within the prototypic implementation this value stands for the general heart beat the embedded device will work.

4. **Buffer Limit:** The *Buffer Limit* defines a part of the decision parameter on which the embedded device decides when to send buffered values to a connected handset.

Following the formulas proposed in section 3.5.1, missing fields can be calculated from the remaining values.

3.8 Security Aspect

Just broadcasting measurement data gives rise to a great privacy issue since everyone can capture these broadcast data packages.
As described in 2.2.4 a BLE device can have a white list with devices that it is allowed to communicate with. This technique also covers the provision of additional information from an advertising frame. Hence everyone can receive the information about available data but only eligible devices actually get the data. How to get a certain device on this white list is a different problem and won't be covered by the prototype. Possibilities like a pairing procedure with NFC or an kind of controlled pairing might be suitable.

3.9 Summary

This chapter discussed the design process in this dissertation. The first key questions that were discussed are ways to influence power consumption in embedded devices and where to locate intelligence and decision logic. The discussion about outlier analysis outlines several ways to determine unusual measurements. The proposed design utilising the advertising mode of Bluetooth low energy revealed to exceed the scope of this dissertation. The following discussion about connectionless vs. continuous connections led to the conclusion that a continuous Bluetooth low energy connection with the right parameter yields great energy efficiency potential. The last chapter introduces the resulting Connection Control Protocol, a GATT service designed to control an embedded device and Bluetooth low energy connection parameter. Security aspects are also addressed in the last chapter.

Chapter 4

Implementation

This chapter draws on the discussions and conclusions of the previous chapter. It describes the implementation of the embedded mote and the mobile client. Firstly the components chosen for implementing the prototype are detailed, followed by a description of the implementation of the designed GATT service on both the embedded mote and mobile handset. Since the exchange of data is a central aspect of this work, section 4.4 dwells upon the processes needed on both sides to handle incoming and outgoing data exchange on a Bluetooth low energy connection. Finally the implemented behaviour of the handset- and the embedded application is explained full.

4.1 Used components

Following the discussion of embedded prototyping boards in section 2.4 it was decided to chose an Arduino Uno board due to its ready availability and flexible extensibility. In particular an Arduino Uno Revision R3 was used. To enable the Arduino with Bluetooth low energy functionality a BLE shield from Red Bear Labs was chosen. The Bluetooth low energy chip used on this shield is the Nordic Semi nRF8001. Further documentation and specifications can be found in [30]. As it was necessary to simulate several input sources some sensors were required. The easiest way of extending the prototype with a variety of sensors was to make use of a Tinkerkit sensor board.

On the mobile handset side a Samsung Galaxy S4 (GT-I9505) was used. It provides a Bluetooth low energy enabled chip and the Samsung provided Bluetooth low energy SDK was employed. Table 4.1 provides a comprehensive overview of the used components.

Component	Manufacturer	Type
Arduino Uno	Arduino LLC	A000066
BLE Shield	Red Bear Labs	
Sensor Shield	Tinkerkit	
Hall Sensor	Tinkerkit	
Galaxy S4	Samsung	GT-I9505

Table 4.1: Overview of used components

4.2 Getting the BLE shield up and running

In order to run the Red Bear Lab BLE shield with the nRF8001 library by Nordic Semi some special wiring is required. The library for the BLE shield provided by Red Bear Lab uses the SPI interface to communicate with the shield and the nRF8001. Figure 4.1 shows the pins that have to be connected in order to enable the library to access certain pins on the nRF8001 directly. A modified version of the library that does not require this wiring can be found in the Bluetooth Developer Forum [5]. Unfortunately it was found to be quite unstable when trying in use.

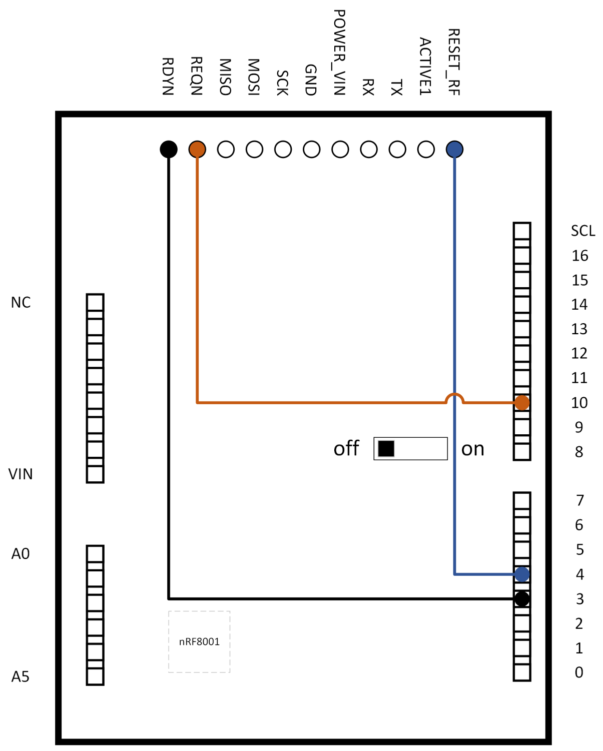


Figure 4.1: Red Bear Lab BLE shield schematic 3 pins have to be connected to I/O pins to gain direct access the nRF8001

4.3 Connection Control Service

The nRF8001 has to be initialized with a profile consisting of the GATT service definition, gapp-settings, hardware-settings and current-settings. The profile definition is done in XML. Creating and editing this file can be done manually or with the nRFgo Studio provided by Nordic Semiconductors. This profile is then compiled by nRFgo Studio into header files containing the profile definitions in binary. The exact compiling process in unknown. The resulting binary data seems to be smaller then the input file, one possible explanation for this is that it is likely that some form of optimisation is carried out as part of the compilation process. The code in listing 4.1 shows a shortened version of the final profile. The full definition can be found in the appendix, listing A.1.

Listing 4.1: shortened version of the nRF8001 profile definition

```
<Profile Version="1.3">
    <SetupId>1</SetupId>
    <Device>nRF8001_Dx</Device>
    <Service Type="local" PrimaryService="true">
        <Name>Device Information</Name>
        ...
    </Service>
    <Service Type="local" PrimaryService="true">
        <Name>Heart Rate</Name>
        ...
    </Service>
    <Service Type="local" PrimaryService="true">
        <Name>ConnectionControl</Name>
        <Uuid>1111</Uuid>
        <Characteristic>
            <Name>ConnectionInterval</Name>
            <Uuid>0029</Uuid>
            ...
        </Characteristic>
        <Characteristic>
            <Name>SlaveLatency</Name>
            <Uuid>1113</Uuid>
            ...
        </Characteristic>
        <Characteristic>
            <Name>SamplingRate</Name>
            <Uuid>1114</Uuid>
            ...
        </Characteristic>
        <Characteristic>
            <Name>BufferLimit</Name>
            <Uuid>1115</Uuid>
            ...
        </Characteristic>
    </Service>
    ...
</Profile>
```

4.4 Transmitting and handling data

This section describes how one can transmit data and handle incoming data on an Arduino as well as with the Samsung BLE SDK on Android. As previously noted Bluetooth low energy only communicates via the manipulation of characteristic values. These values can be changed locally and the new value transmitted to other devices in the network.

4.4.1 On the Arduino

The library to access the nRF8001 BLE chip on an Arduino provides an interface that abstracts the lower level communication with the chip itself. This interface can be used to establish connections, configure certain properties of the nRF8001 and can also send and receive data on an active Bluetooth low energy connection.

Sending data

The library provided by Nordic Semiconductors abstracts a lot of the steps needed to change characteristic values and transmit them to connected devices. The only drawback is that the given interface only provides a function to send a single value at a time. In the design chapter section 3.5.1 the idea of sending values in bulk to optimise the efficiency of each transmission was introduced. To permit the sending of values in bulk the existing interface had to be extended. Fortunately the library provided is open source. Listing 4.2 shows the implemented function. The function retrieves the oldest element in the buffer until the maximum available payload is reached. In the case of the nRF8001 this is 22 bytes. Then the data is sent over the connection. The current configuration automatically handles the reception of acknowledgements or the re-transmission of lost data packages.

Listing 4.2: Method to send bulk data over a Bluetooth low energy connection

```
void sendData()
{
  heart_rate_set_support_contact_bit();
  heart_rate_set_contact_status_bit();

  uint8_t data_index = 0;
  current_heart_rate_data[data_index] &=
                     ~HEART_RATE_FLAGS_MEAS_SIZE_BIT;
  current_heart_rate_data[data_index] &=
                     ~HEART_RATE_FLAGS_ENERGY_EXPENDED_STATUS_BIT;
  current_heart_rate_data[data_index] &=
                     ~HEART_RATE_FLAGS_RR_INTERVAL_SUPPORT_BIT;
  data_index++;
  while(data_index < HR_MAX_PAYLOAD && buffer_size() > 0) {
    current_heart_rate_data[data_index] = buffer_pop();
    data_index++;
  }
  Serial.print(F("Data Index - "));
  Serial.println(data_index);
  boolean success = lib_aci_send_data(
            PIPE_HEART_RATE_HEART_RATE_MEASUREMENT_TX,
      (uint8_t *)&current_heart_rate_data[0],
                        data_index);

  Serial.print(F("Send Data - "));
  Serial.println(success);
  if(success) {
    radio_ack_pending = true;
  }
}
```

Receiving data

A polling approach is used to handle incoming characteristic changes. The function $aci\_loop()$ gets called in every cycle and looks for the arrival of aci_events. These aci_events are then buffered in a pipeline pointing to a memory address on the nRF8001. In the case of an available aci_events the function executes a big switch-case statement to determine the appropriate reaction to this event. Depending on the events op-code ($aci\_evt- > evt\_opcode$) cases will be handled differently. The important case for receiving data is $ACI\_EVT\_DATA\_RECEIVED(0x8C)$. The changed characteristic is encoded within the pipeline number ($aci\_evt- > params.data\_received.rx\_data.pipe\_number$). Each characteristic is represented by a separate pipe. The new values of the characteristic can be accessed by reading the array returned from $aci\_evt- > params.data\_received.rx\_data.aci\_data$. The array values are always represented as $uint8\_t$ numbers. When obtaining the values it is important to consider the endianness and size of the expected values and apply a transformation if necessary. Listing 4.3 shows a shortened version of the process to handle incoming events and characteristic changes. The full code can be reviewed on the CD attached to this dissertation.

Listing 4.3: Code snipped of handling incoming data

```
void aci_loop()
{
  // Enter the if statement only when there is a ACI event
  // available to be processed
  if (lib_aci_event_get(&aci_state, &aci_data))
  {
    aci_evt_t * aci_evt;
    aci_evt = &aci_data.evt;

    switch(aci_evt->evt_opcode)
    {

     ...

    case ACI_EVT_DATA_RECEIVED:
      Serial.println(F("DATA RECEIVED"));
      Serial.print(F("Pipe #: 0x"));
      Serial.println(aci_evt->params.data_received.rx_data.
          pipe_number, HEX);

      uint16_t value;
      switch(aci_evt->params.data_received.rx_data.pipe_number)
      {
        case PIPE_CONNECTIONCONTROL_CONNECTIONINTERVAL_RX_ACK_AUTO:
          Serial.print(F("CONNECTION_INTERVAL: "));

          // Two bytes
          // big-endian!
          value = aci_evt->params.data_received.rx_data.aci_data[1] +
          ((uint16_t) aci_evt->params.data_received.rx_data.aci_data
             [0]<< 8);
          connection_interval = value;
          Serial.println(value);
        break;

        ...

      }
    }
  }
}
```

4.4.2 On Android

Currently an Android device can only run a Bluetooth low energy connection as a master device.

Sending data

Sending data from an Android client is not as abstracted as on the Arduino. To send a characteristic value over a connection several steps are necessary. The first step is to change the value of a specific characteristic locally and then send the changed characteristic value to a specific client. To check if a characteristic has been successfully sent it is necessary to implement the callback function `public void onCharacteristicWrite( BluetoothGattCharacteristic characteristic, int statusId)`. The callback gets called with a status message regardless of whether the transmission was successful or not. The status message also indicates an encountered problem in the case of failure. Listing 4.4 shows some status messages as defined in the Samsung BLE SDK.

Listing 4.4: A set of status codes from the Samsung BLE SDK

```
GATT_SUCCESS = 0;
GATT_READ_NOT_PERMITTED = 2;
GATT_WRITE_NOT_PERMITTED = 3;
GATT_INSUFFICIENT_AUTHENTICATION = 5;
GATT_REQUEST_NOT_SUPPORTED = 6;
GATT_INVALID_OFFSET = 7;
GATT_INVALID_ATTRIBUTE_LENGTH = 13;
GATT_INSUFFICIENT_ENCRYPTION = 15;
...
```

Receiving data

For receiving data the BLE SDK for Android by Samsung provides a subscription approach with callback. After service discovery takes place an app can subscribe to characteristic changes. Incoming characteristic value changes will be automatically delivered via a callback. Listing 4.5 shows a a shortened version of the "onCharacteristicChangedÂťÂť callback. It is unclear what approach is adopted to obtain characteristic changes on the lower levels of the the implementation. The BluetoothGattCharacteristic that gets delivered as a parameter represents the changed characteristic and also contains the new value. The value is represented as a byte array. As in the retrieval of characteristic values on the Arduino, it is crucial to take care of the endianness as well as the size in bytes of the expected data.

Another critical point is the difference in data formats between Android and Arduino. The Arduino data values are transmitted as unsigned integers with a size of 2 bytes ($uint8\_t$). Java, and therefore Android, does not support unsigned integer values. This problem has not been addressed in the implementation since the focus was more on configuring the connection rather then on the transmitted data. A conversion step is necessary to get valid results when reading the data. After extracting the data values, they can be further processed in the usual manner. In the case of this application the values are forwarded with an ActivityMessageHandler to an Activity that displays the values.

Listing 4.5: Handling incoming data callback on Android

```
public void onCharacteristicChanged(BluetoothGattCharacteristic
    characteristic) {
   Log.i(TAG, "onCharacteristicChanged");
   Bundle mBundle = new Bundle();
   Message msg = Message.obtain(mActivityHandler, HRP_VALUE_MSG);

   int length = characteristic.getValue().length;
   int[] hrmValues = new int[characteristic.getValue().length -2];

   if(isContentArray(characteristic.getValue()[0])) {

     for (int i = 1; i < characteristic.getValue().length-1; i++) {
        hrmValues[i - 1] = characteristic.getValue()[i];
     }

    mBundle.putIntArray(HRM_VALUE, hrmValues);
```

```
        msg.setData(mBundle);
        msg.sendToTarget();
      }
      ...
}
```

4.5 Changing connection parameter

Changing connection parameters means changing the *Connection Interval* and *Slave Latency*. Since it is not possible to change these connection values with the BLE SDK on Android, this section will exclusively describe how to change connection parameters with the Nordic Semi BLE library. As in the case when sending data, the library abstracts most of the steps required.
The function $boolean\ \ lib\_aci\_change\_timing(uint16\_t\ \ minimun\_cx\_interval,\ \ uint16\_t\ \ maximum\_cx\_interval,\ \ uint16\_t\ \ slave\_latency,\ \ uint16\_t\ \ timeout)$ starts the process. The commands parameters are constrained in the range of values to use. Details can be found in the product specification [*nRF8001 Product specification, Rev1.1, chapter 24.18.3*]

To handle process responses, the event handling cycle from listing 4.3 has to be extended to handle ACI_EVT_TIMING events. Handling these events enables us to react to a possible rejection of the requested connection parameter change by adjusting the request parameter or abort the attempt. The complete process needed to change the connection parameter is set out in figure 4.2.

4.6 The embedded node (Arduino)

The purpose of the embedded node is to measure and collect data independently and then transfer this data to a connected mobile client. Figure 4.3 gives a flow chart of the repetitive behaviour of the embedded device.

When the embedded node is starting it first initialises itself. Within the initialisation step the serial connection and basic environmental parameters are set up and initialized. In this step the nRF8001 is also initialised with the pre-compiled profile for the chip (see section 4.3). After this step the device enters a process loop. First step is always the reading of data from the sensors followed by the buffering of these values. The implemented buffer is a linked ring buffer that extends until the available memory

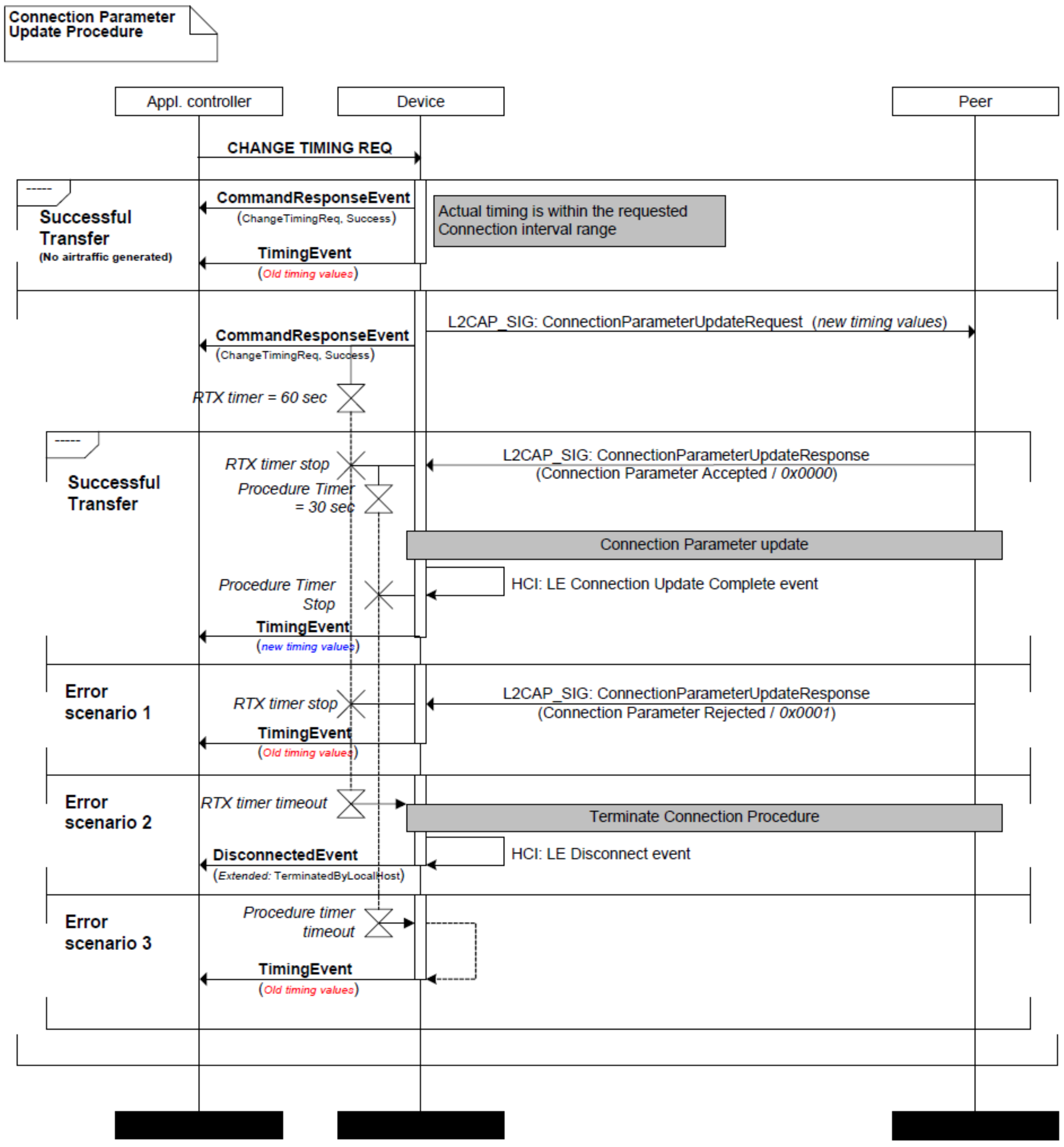

Connection Parameter Update Procedure
Appl. controller
Device
Peer
CHANGE TIMING REQ
Successful Transfer
(No airtraffic generated)
CommandResponseEvent
(ChangeTimingReq, Success)
TimingEvent
(Old timing values)
Actual timing is within the requested Connection interval range
L2CAP_SIG: ConnectionParameterUpdateRequest (new timing values)
CommandResponseEvent
(ChangeTimingReq, Success)
RTX timer = 60 sec
Successful Transfer
RTX timer stop
Procedure Timer = 30 sec
L2CAP_SIG: ConnectionParameterUpdateResponse
(Connection Parameter Accepted / 0x0000)
Connection Parameter update
Procedure Timer Stop
HCI: LE Connection Update Complete event
TimingEvent
(new timing values)
Error scenario 1
RTX timer stop
L2CAP_SIG: ConnectionParameterUpdateResponse
(Connection Parameter Rejected / 0x0001)
TimingEvent
(Old timing values)
Error scenario 2
RTX timer timeout
Terminate Connection Procedure
DisconnectedEvent
(Extended: TerminatedByLocalHost)
HCI: LE Disconnect event
Error scenario 3
Procedure timer timeout
TimingEvent
(Old timing values)

Figure 4.2: Procedure followed for changes in the timing sequence

limit is reached. When this happens the oldest value is overwritten.

The next step in the procedure is data analysis. The requirement was that the embedded device should be able to independently detect outliers in the measured data. Due to the limited processing capabilities of the device and for the purposes of energy efficiency, the device only considers a small range of measured data when seeking to determine the presence of outliers. As discussed in chapter 3.2 and 3.3 determination

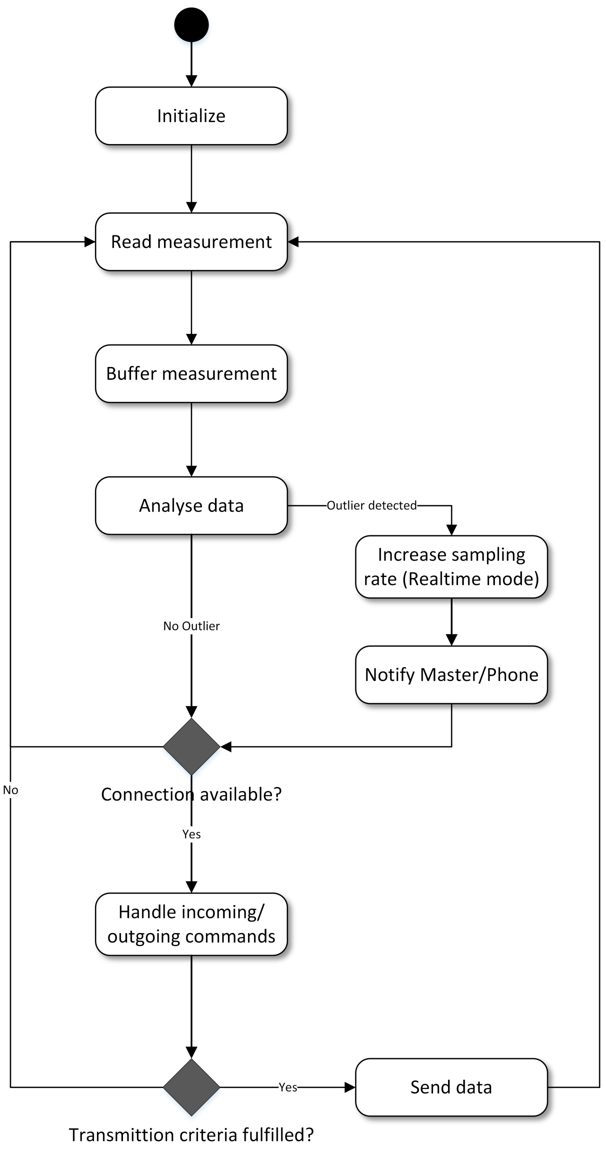


Figure 4.3: Sequence diagram mote/client

criteria can be manifold. In this implementation upper and lower thresholds are used to categorise the current measurement as an outlier or not. In the case of the used hall sensor this threshold is exceeded if a magnet approaches the sensor.

In the case of a detected outlier, the sampling rate will be increased and the buffer limit decreased. The next step is to inform the connected handset of the detection and the applied changes. The handset can refuse the decision by sending a request with different settings or it can accept the changes. Handling of the possible reaction by the handset is done in the following 'handle incoming commands' step. Of course this is only carried out after it is confirmed that a connection is available.

In the prototype implementation the buffer limit serves as a decision criteria for the next process-step. Deciding when to make a transmission can depend on diverse parameter and logics. If this step evaluates to true, a transmission is initiated.In order to be as efficient as possible the communication packet is filled with the oldest elements in the buffer until the maximum available payload is reached or there are no more measurements available. After a successful transmission, the cycle starts again by reading sensor values.

4.7 The Android client

The Android application serves as an endpoint for the Bluetooth connection and the embedded motes. Received data can be stored and processed in place or transmitted on. It was necessary to work with a BLE enabled phone as endpoint with the an appropriate chipset. An API to access BLE in Android has been made available with the release of API lvl 19 (Android 4.3). To work with Android versions before API lvl 19 a vendor specific library is required to access the BLE functionality. Typical sendors include Samsung, Intel, Texas Instruments and Broadcom. They provide their own API to access BLE functionality on Android. As an aside it should be noted that the BLE API introduced in API lvl 19 heavily inherits technology and functionality from the Broadcom BLE stack [25].

Figure 4.4 shows a behaviour sequence diagram for the Android application. When starting the app the first step is always a check whether the Bluetooth module is activated or not. With an activated Bluetooth module the application steps into scanning mode to find nearby Bluetooth low energy devices advertising their services. Detecting a device with the a known ID (i.e. one that matches a specific pattern) the app will attempt to establish a connection.

Following the successful establishment of a connection, the handset will discover if the required services like the *Connection Control Service* are available. As described above in section 4.4.2, the app will then subscribe to changes in these services and characteristics. Once this has been taken place the app will wait for transmitted data, delivered as changes of characteristic values.

The analysis of received data can run in a concurrent process. Since the focus of building the prototype was on efficient connection parameter settings, the concurrent

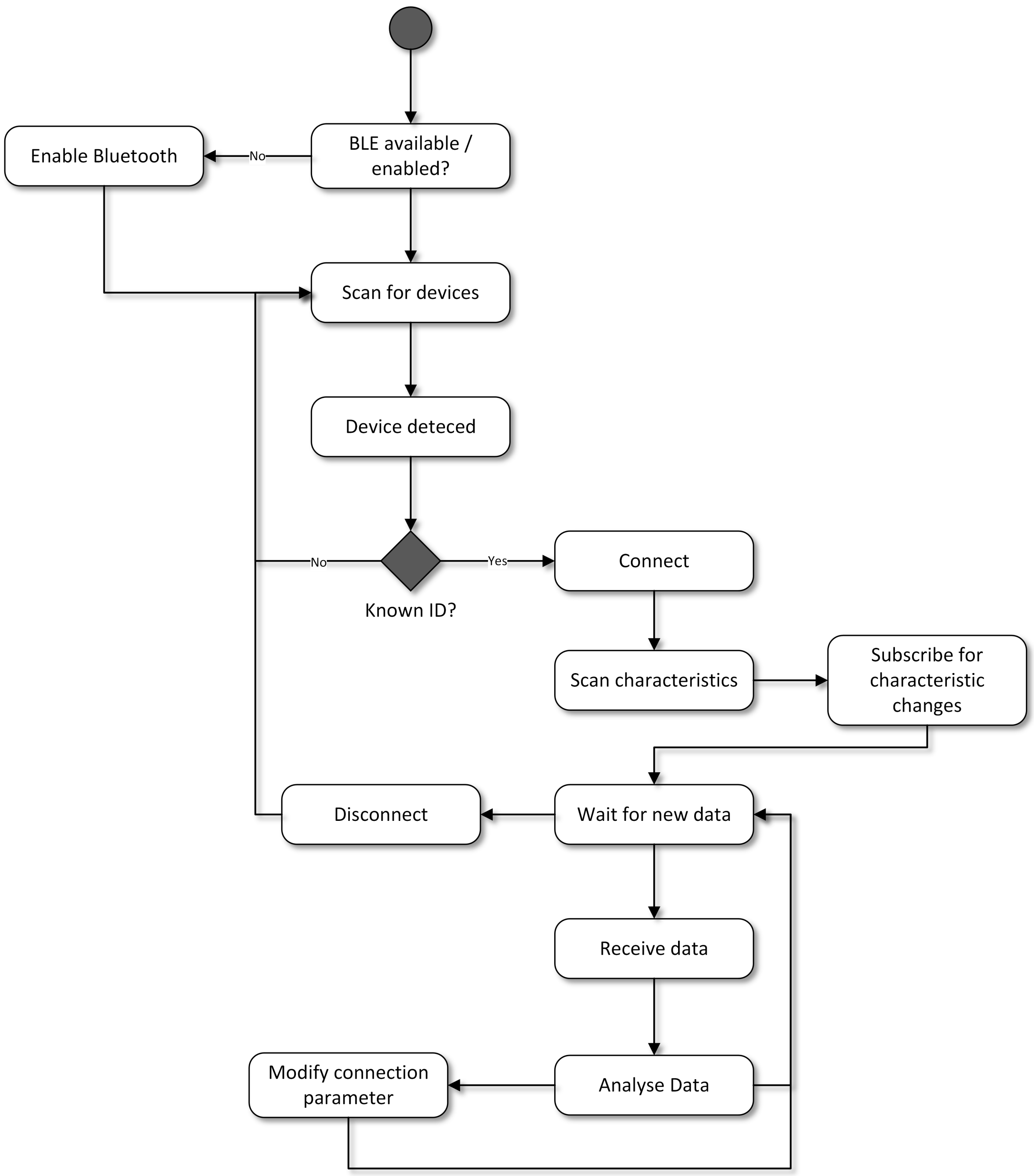
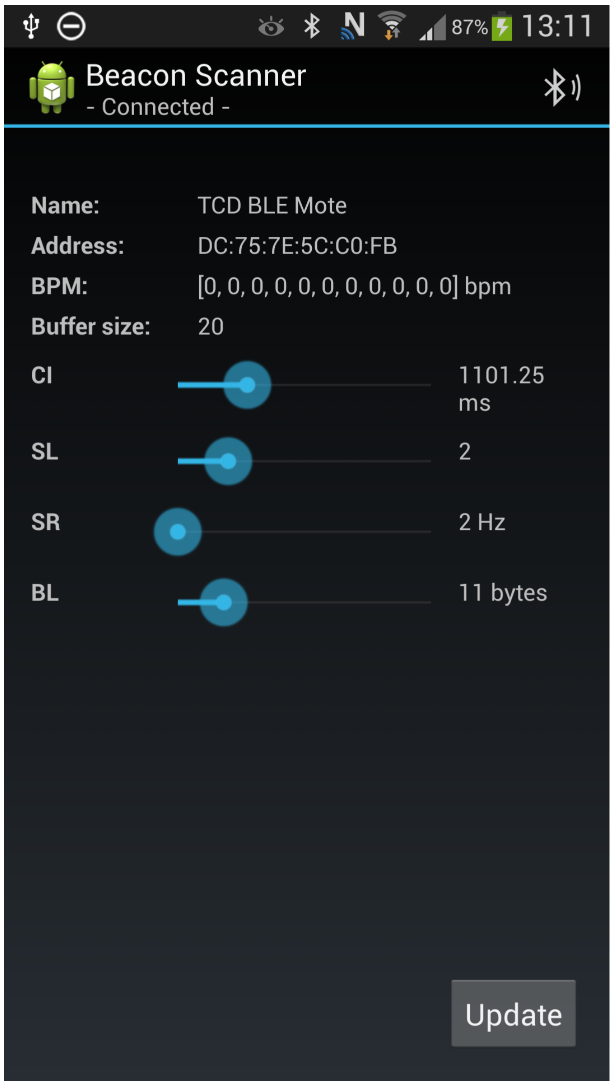


Figure 4.4: Behaviour of the Android client

analysis task implementation is implemented as a dummy. Figure 4.5 shows the Android apps GUI. For testing purposes the characteristics of the *Connection Control Service* can be set manually.



Beacon Scanner
- Connected -

Name:	TCD BLE Mote
Address:	DC:75:7E:5C:C0:FB
BPM:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] bpm
Buffer size:	20
CI	1101.25 ms
SL	2
SR	2 Hz
BL	11 bytes

Figure 4.5: The Android clients GUI

4.8 Summary

This chapter outlined the implementation of a prototype embedded mote as well as of a corresponding Android application. A detailed description of how to handle incoming and outgoing data streams on both both platforms is provided. Changes in connection parameters, with the help of the proposed service, has been achieved. The extension of the implemented behaviour to each side has been described.

Chapter 5

Conclusions

This dissertation proposed a way for energy efficient wireless communication achieved by situation adaptive change of connection parameter for Bluetooth 4.0 low energy connections. The first design was based on a connectionless approach utilising the Bluetooth low energy advertising mode. Implementing this idea proved out of scope for this dissertation. Based on a connection based communication a dedicated GATT service to configure the behaviour of embedded sensor devices was defined. The implementation of an embedded prototype included the dedicated GATT service as well as adaptive behaviour to detected outlier in the measured sensor data. An Android application as central data sink and controlling entity handles transmitted data and enabled the user to manually configure connection parameter.

Since the focus of this work was on energy efficient communication with adaptive connection parameter, the measured data did not consist of data with medical relevance. Also the analysing task to determine outstanding measurements did not involve medical knowledge or expertise.

5.1 Future work

Out of the scope of this work but not less interesting is the pure advertising approach. Completely omitting the overhead of communication establishment and dedicated transmissions, the connectionless approach holds potential for energy efficient communication. To determine the actual power consumption to make qualified comparisons a test bench has to be prepared. In the case of the nRF8001 Bluetooth low energy chip used in this work, Nordic Semiconductors provides a detailed documentation of how to measure the current power draw of this RF chip [29]. To perform these tests a special hardware setup is required.

To investigate into the possibility of a commercialisation smaller hardware is required. Implementing the proposed approach with a different hardware platform is the next logical step in the continuous improvement process. The design of a dedicated IC integrating RF chip, processor as well as sensors will drastically improve the size as well as power requirements. Considering especially power efficient components will create an optimal environment for future improvements.

To use this approach with sensors that capture actual medical data, appropriate analysis algorithms have to be developed and implemented. This applies for the Android application as well as outlier detection rules and algorithms running on embedded devices. Last but not less important comes the security aspect. Medical measurements are highly sensitive personal information and have to be protected. Bluetooth low energy provides techniques to ensure data originality as well as encryption for over the air communication. Using advanced security mechanisms will have an impact onto power consumption, future work might focus on choosing an optimum between security and energy demand.

Appendix A

Abbreviations

A.1 nRF8001 Profile definition and GATT service definition

Listing A.1: Complete nRF8001 profile definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AttributeServer>
<Profile Version="1.3">
    <SetupId>1</SetupId>
    <Device>nRF8001_Dx</Device>
    <Service Type="local" PrimaryService="true">
        <Name>Device Information</Name>
        <Uuid>180a</Uuid>
        <Characteristic>
            <Name>Serial Number String</Name>
            <Uuid>2a25</Uuid>
            <DefaultValue>1587</DefaultValue>
            <UsePresentationFormat>1</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>4</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>false</Write>
                <Notify>false</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>true</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="25" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>0</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties/>
        </Characteristic>
    </Service>
    <Service Type="local" PrimaryService="true">
        <Name>Heart Rate</Name>
        <Uuid>180d</Uuid>
        <Characteristic>
            <Name>Heart Rate Measurement</Name>
            <Uuid>2a37</Uuid>
            <DefaultValue></DefaultValue>
            <UsePresentationFormat>0</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>19</MaxDataLength>
            <AttributeLenType>2</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>false</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="1" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>0</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties/>
        </Characteristic>
        <Characteristic>
            <Name>Heart Rate Sensor Location</Name>
            <Uuid>2a38</Uuid>
            <DefaultValue>03</DefaultValue>
            <UsePresentationFormat>0</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>1</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>false</Write>
                <Notify>false</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>true</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="1" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>0</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties/>
        </Characteristic>
        <Characteristic>
            <Name>Heart Rate Control Point</Name>
            <Uuid>2a39</Uuid>
            <DefaultValue></DefaultValue>
            <UsePresentationFormat>0</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>1</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>true</Write>
                <Notify>false</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="1" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>0</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties/>
        </Characteristic>
    </Service>
    <Service Type="local" PrimaryService="true">
        <Name>ConnectionControl</Name>
        <Uuid>1111</Uuid>
        <Characteristic>
            <Name>ConnectionInterval</Name>
            <Uuid>0029</Uuid>
            <DefaultValue>6</DefaultValue>
            <UsePresentationFormat>1</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>2</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>true</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="6" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>60000</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties>
                <Read>60000</Read>
                <Notify>60000</Notify>
            </PeriodForProperties>
        </Characteristic>
        <Characteristic>
            <Name>SlaveLatency</Name>
            <Uuid>1113</Uuid>
            <DefaultValue>0</DefaultValue>
            <UsePresentationFormat>1</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>1</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>true</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="4" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>60000</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties>
                <Read>60000</Read>
                <Notify>60000</Notify>
            </PeriodForProperties>
        </Characteristic>
        <Characteristic>
            <Name>SamplingRate</Name>
            <Uuid>1114</Uuid>
            <DefaultValue>1</DefaultValue>
            <UsePresentationFormat>1</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>4</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>true</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="8" NameSpace="01" Unit="2722"/>
            <PeriodForReadingThisCharacteristic>60000</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties>
                <Read>60000</Read>
                <Notify>60000</Notify>
            </PeriodForProperties>
        </Characteristic>
        <Characteristic>
            <Name>BufferLimit</Name>
            <Uuid>1115</Uuid>
            <DefaultValue>1</DefaultValue>
            <UsePresentationFormat>1</UsePresentationFormat>
            <UserDescription></UserDescription>
            <MaxDataLength>1</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>true</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>true</AckIsAuto>
            <PresentationFormatDescriptor Value="0000" Exponent="0"
               Format="4" NameSpace="01" Unit="0000"/>
            <PeriodForReadingThisCharacteristic>60000</
               PeriodForReadingThisCharacteristic>
            <PeriodForProperties>
                <Read>60000</Read>
                <Notify>60000</Notify>
            </PeriodForProperties>
        </Characteristic>
    </Service>
    <Gapsettings>
        <Name>TCD BLE Mote</Name>
        <DeviceNameWriteLength>12</DeviceNameWriteLength>
        <LocalPipeOnDeviceName>false</LocalPipeOnDeviceName>
        <DeviceNameShortLength>0</DeviceNameShortLength>
        <Apperance>0340</Apperance>
        <SecurityLevel>0</SecurityLevel>
        <AuthenticationReq>0</AuthenticationReq>
        <IoCapabilities>0</IoCapabilities>
        <BondTimeout>0</BondTimeout>
        <SecurityRequestDelay>0</SecurityRequestDelay>
        <MinimumKeySize>7</MinimumKeySize>
        <MaximumKeySize>16</MaximumKeySize>
        <AdvertisingDataBondedBitmap>1f</AdvertisingDataBondedBitmap>
        <AdvertisingDataGeneralBitmap>1f</
           AdvertisingDataGeneralBitmap>
        <AdvertisingDataBrodcastBitmap>0</
           AdvertisingDataBrodcastBitmap>
        <AdvertisingDataBondedScanResponseBitmap>0</
           AdvertisingDataBondedScanResponseBitmap>
        <AdvertisingDataGeneralScanResponseBitmap>0</
           AdvertisingDataGeneralScanResponseBitmap>
        <AdvertisingDataBrodcastScanResponseBitmap>0</
```

```
            AdvertisingDataBrodcastScanResponseBitmap>
        <AdvertisingDataBondedBitmapCustom>1</
            AdvertisingDataBondedBitmapCustom>
        <AdvertisingDataGeneralBitmapCustom>1</
            AdvertisingDataGeneralBitmapCustom>
        <AdvertisingDataBrodcastBitmapCustom>0</
            AdvertisingDataBrodcastBitmapCustom>
        <AdvertisingDataBondedScanResponseBitmapCustom>0</
            AdvertisingDataBondedScanResponseBitmapCustom>
        <AdvertisingDataGeneralScanResponseBitmapCustom>0</
            AdvertisingDataGeneralScanResponseBitmapCustom>
        <AdvertisingDataBrodcastScanResponseBitmapCustom>0</
            AdvertisingDataBrodcastScanResponseBitmapCustom>
        <TxPowerLevelOffset>0</TxPowerLevelOffset>
        <MinimumConnectionInterval>400</MinimumConnectionInterval>
        <MaximumConnectionInterval>800</MaximumConnectionInterval>
        <SlaveLatency>0</SlaveLatency>
        <TimeoutMultipler>502</TimeoutMultipler>
        <ServiceToAdvertise>
            <Uuid>180d</Uuid>
        </ServiceToAdvertise>
        <CustomAdTypes>
            <AdType index="1">
                <Type>19</Type>
                <Value>4003</Value>
            </AdType>
            <AdType index="2">
                <Type>18</Type>
                <Value></Value>
            </AdType>
        </CustomAdTypes>
    </Gapsettings>
    <Hardwaresettings>
        <Clocksource>1</Clocksource>
        <ClockAccuracy>1</ClockAccuracy>
        <InitialTxPower>3</InitialTxPower>
        <HfClkSource>0</HfClkSource>
        <DcDcConverter>0</DcDcConverter>
        <ActiveSignalModeIndex>0</ActiveSignalModeIndex>
        <ActiveSignalToTickDistance>0</ActiveSignalToTickDistance>
        <DynamicWindowLimitingEnabled>true</
            DynamicWindowLimitingEnabled>
    </Hardwaresettings>
    <CurrentInput>
```

```
        <BatteryCharge>220</BatteryCharge>
        <Master32KhzClockAccuracy>10</Master32KhzClockAccuracy>
        <ConnectionInterval>1000</ConnectionInterval>
        <PercentOfTimeSleeping>10</PercentOfTimeSleeping>
        <PercentOfTimeAdvertising>11.1111</PercentOfTimeAdvertising>
        <AdvertisingInterval>1280</AdvertisingInterval>
    </CurrentInput>
</Profile>
```

List of Tables

List of Figures

Bibliography

[1] Hamdan O Alanazi, Hamid A Jalab, Gazi Mahabubul Alam, and B B Zaidan. Securing electronic medical records transmissions over unsecured communications : An overview for better medical governance. *Journal of Medicinal Plants Research*, 4(19):2059–2074, 2010.

[2] Wi-Fi Alliance. Wi-Fi Alliance Published Specifications, 2013.

[3] Urs Anliker, Jamie a Ward, Paul Lukowicz, Gerhard Tröster, François Dolveck, Michel Baer, Fatou Keita, Eran B Schenker, Fabrizio Catarsi, Luca Coluccini, Andrea Belardinelli, Dror Shklarski, Menachem Alon, Etienne Hirt, Rolf Schmid, and Milica Vuskovic. AMON: a wearable multiparameter medical monitoring and alert system. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 8(4):415–27, December 2004.

[4] H.H. Asada, P. Shaltis, A. Reisner, and R.C. Hutchinson. Mobile monitoring with wearable photoplethysmographic biosensors. *IEEE Engineering in Medicine and Biology Magazine*, 22(3):28–40, May 2003.

[5] Bluetooth Developer Forum. Webinar: Enabling Bluetooth Low Energy on Arduino Platform using Nordic nRF8001 module (Download nRF8001 SDK for Arduino), 2013.

[6] Peter D. Bradley. An ultra low power, high performance Medical Implant Communication System (MICS) transceiver for implantable devices. In *2006 IEEE Biomedical Circuits and Systems Conference*, pages 158–161. IEEE, November 2006.

[7] T. N. Millis C. P. Swain , F. Gong. Wireless transmission of a color television moving image from the stomach using a miniature CCD camera, light source and microwave transmitter. *Gut*, 39(A26), 1996.

[8] Huasong Cao, Haoming Li, Leo Stocco, and Victor C. M. Leung. Demonstration of a novel wireless three-pad ECG system for generating conventional 12-lead sig-

nals. *Proceedings of the Fifth International Conference on Body Area Networks - BodyNets '10*, page 29, 2010.

[9] Huasong Cao, Haoming Li, Leo Stocco, and Victor C M Leung. Wireless Three-Pad ECG System : Challenges , Design , and Evaluations. *Communications and Networks*, 13(2):113–124, 2011.

[10] Xinkai Chen, Xiaoyu Zhang, Linwei Zhang, Xiaowen Li, Nan Qi, Hanjun Jiang, and Zhihua Wang. A Wireless Capsule Endoscope System With Low-Power Controlling and Processing ASIC. *IEEE Transactions on Biomedical Circuits and Systems*, 3(1):11–22, February 2009.

[11] D. Dakopoulos and N.G. Bourbakis. Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):25–35, January 2010.

[12] Ecma Internationa. 340âĂŞNear Field Communication Interface and Protocol (NFCIP-1). *Standard ECMA-304*, 2nd Editio(December), 2004.

[13] Shahin Farshchi, Istvan Mody, and Jack W Judy. A TinyOS-based wireless neural interface. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 6:4334–7, January 2004.

[14] Shahin Farshchi, Paul H Nuyujukian, Aleksey Pesterev, Istvan Mody, and Jack W Judy. A TinyOS-enabled MICA2-based wireless neural interface. *IEEE transactions on bio-medical engineering*, 53(7):1416–24, July 2006.

[15] Thaddeus R F Fulford-Jones, Gu-Yeon Wei, and Matt Welsh. A portable, low-power, wireless two-lead EKG system. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 3:2141–4, January 2004.

[16] Frank E. Grubbs. Sample Criteria for Testing Outlying Observations. *The Annals of Mathematical Statistics*, 21(1):27–58, March 1950.

[17] Frank E. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1):1, February 1969.

[18] D M Hawkins. *Identification of outliers*. Chapman and Hall, 1980.

[19] IDC. Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter, According to IDC - prUS23771812, 2012.

[20] ISO/IEEE. ISO/IEEE 11073 standards, 2013.

[21] Vladimir I Ivanov, Paul L Yu, and John S Baras. Securing the communication of medical information using local biometric authentication and commercial wireless links. *Health informatics journal*, 16(3):211–23, September 2010.

[22] E Jovanov, D Raskovic, J Price, J Chapman, A Moore, and A Krishnamurthy. Patient monitoring using personal area networks of wireless intelligent sensors. *Biomedical Sciences Instrumentation*, 37:373–378, 2001.

[23] Moussa Kfouri, Ognian Marinov, Paul Quevedo, Naser Faramarzpour, Shahram Shirani, Louis W.-C. Liu, Qiyin Fang, and M Jamal Deen. Toward a Miniaturized Wireless Fluorescence-Based Diagnostic Imaging System. *IEEE Journal of Selected Topics in Quantum Electronics*, 14(1):226–234, 2008.

[24] Martin Kusserow, Oliver Amft, and Gerhard Tröster. BodyANT: Miniature wireless sensors for naturalistic monitoring of daily activity. In *Proceedings of the 4th International ICST Conference on Body Area Networks*. ICST, 2009.

[25] Sara Sinclair Brody Matthew Xie, Richard Hyndman. Google I/O 2013 - Best Practices for Bluetooth Development, 2013.

[26] Mikeselectricstuff. Another pill-cam teardown, 2012.

[27] Mikeselectricstuff. Youtube: Pill camera teardown, 2012.

[28] Andrea Moglia, Arianna Menciassi, Marc Oliver Schurr, and Paolo Dario. Wireless capsule endoscopy: from diagnostic devices to multipurpose robotic systems. *Biomedical microdevices*, 9(2):235–43, April 2007.

[29] Nordic Semiconductors. Measuring the Dynamic Current of nRF8001. 2012.

[30] Nordic Semiconductors. Nordic Semiconductors nRF8001, 2013.

[31] OECD. Public expenditure on health, 2013.

[32] A. Pantelopoulos and N.G. Bourbakis. A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, January 2010.

[33] H.J. Park, H.W. Nam, B.S. Song, J.L. Choi, H.C. Choi, J.C. Park, M.N. Kim, J.T. Lee, and J.H. Cho. Design of bi-directional and multi-channel miniaturized telemetry module for wireless endoscopy. In *2nd Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology. Proceedings (Cat. No.02EX578)*, pages 273–276. IEEE, 2002.

[34] Julien Penders, Bert Gyselinckx, Ruud Vullers, Michael De Nil, Venkatarama S. R. Nimmala, Jef van de Molengraft, Firat Yazicioglu, Tom Torfs, Vladimir Leonov, Patrick Merken, and Chris Van Hoof. Human++: From technology to emerging health monitoring concepts. In *2008 5th International Summer School and Symposium on Medical Devices and Biosensors*, pages 94–98. IEEE, 2008.

[35] Yonglin Ren, Richard Werner, Nelem Pazzi, and Azzedine Boukerche. Monitoring patients via a secure and mobile healthcare system. *IEEE Wireless Communications*, 17(1):59–65, February 2010.

[36] Soo Young Shin, Hong Seong Park, and Wook Hyun Kwon. Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b. *Computer Networks*, 51(12):3338–3353, August 2007.

[37] V Shnayder, B Chen, K Lorincz, TRFF Jones, and M Welsh. Sensor networks for medical care. *SenSys*, 17(1):1–18, November 2005.

[38] J. Thoné, S. Radiom, D. Turgis, R. Carta, G. Gielen, and R. Puers. Design of a 2Mbps FSK near-field transmitter for wireless capsule endoscopy. *Sensors and Actuators A: Physical*, 156(1):43–48, November 2009.

[39] Pietro Valdastri, Arianna Menciassi, Alberto Arena, Chiara Caccamo, and Paolo Dario. An implantable telemetry platform system for in vivo monitoring of physiological parameters. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 8(3):271–8, September 2004.

[40] Pietro Valdastri, Arianna Menciassi, and Paolo Dario. Transmission power requirements for novel ZigBee implants in the gastrointestinal tract. *IEEE transactions on bio-medical engineering*, 55(6):1705–10, June 2008.

[41] John E. Walsh. Some Nonparametric Tests of Whether the Largest Observations of a Set are too Large or too Small. *The Annals of Mathematical Statistics*, 21(4):583–592, December 1950.

[42] John E. Walsh. Correction to ”Some Nonparametric Tests of Whether the Largest Observations of a set are too Large or too Small”. *The Annals of Mathematical Statistics*, 24(1):134–135, March 1953.

[43] Wikipedia. List of Arduino boards and compatible systems, 2013.

[44] Carmen Y. Poon, Yee Wong, and Yuan-ting Zhang. M-Health: The Development of Cuff-less and Wearable Blood Pressure Meters for Use in Body Sensor Networks. In *2006 IEEE/NLM Life Science Systems and Applications Workshop*, pages 1–2. IEEE, July 2006.

[45] Boo-Ho Yang and Sokwoo Rhee. Development of the ring sensor for healthcare automation. *Robotics and Autonomous Systems*, 30(3):273–281, February 2000.

[46] Hong Yu and Rizwan Bashirullah. An asymmetric RF tagging IC for ingestible medication compliance capsules. *2009 IEEE Radio Frequency Integrated Circuits Symposium*, pages 101–104, June 2009.

[47] Mehmet R Yuce, Tharaka Dissanayake, and Ho Chee Keong. Wireless telemetry for electronic pill technology. In *2009 IEEE Sensors*, pages 1433–1438. IEEE, October 2009.