Bachelor thesis

MDP for Symbian.

Mikael Persson, Erik Jonsson

LIU-IDA/LiTH-EX-G–08/002–SE
MDP for Symbian.

Department of Computer and Information Science, Linköpings Universitet

Mikael Persson, Erik Jonsson

LIU-IDA/LiTH-EX-G–08/002–SE

Bachelor thesis: 15 hp

Level: C

Supervisor: Håkan Johansson,
Cybercom Sweden West

Examiner: Henrik Eriksson,
Department of Computer and Information Science, Linköpings Universitet

Linköping: january 2007
The content of this report describes a bachelor thesis performed on commission by Cybercom Sweden West. The report describes techniques, methods and development tools used during the project. The purpose of this project is to demonstrate the usefulness with a new Bluetooth profile called Medical Device Profile (MDP) on the Medica trade fair 14-11-2007. The MDP profile that will be released in turn of the year 07/08 is primarily intended to be used in medical devices.

The demonstration is to be made by building a demonstrator consisting of an application running on a smartphone with Bluetooth abilities. The application will handle the Bluetooth connection between the smartphone and the oximeter, the data encryption and other functionalities and presenting the figures received form the oximeter in a Graphical User Interface (GUI).

The final demonstrator consists of a smartphone application programmed in Symbian C++, which communicates with the oximeter using the Bluetooth Serial Port Profile (SPP). The application runs on UIQ 3.0 based smartphones and displays heart rate and %SpO2 (The percentage of oxygen saturation in blood), which the application receives from the oximeter.

The original idea was to use the MDP profile by porting Cybercom’s C version of the MDP to Symbian OS for the Bluetooth communication, depending on various factors described more in detail inside the report this was not done. The purpose of the project was still reached, even though another profile then MDP was used. This was done by replacing the considered oximeter for an older version which is using the SPP. By using SPP the same result will be demonstrated but with an older technique, and this will achieve the same result on Medica trade fair. The project was demonstrated on Medica with a successful result.
Abstract

The content of this report describes a bachelor thesis performed on commission by Cybercom Sweden West. The report describes techniques, methods and development tools used during the project. The purpose of this project is to demonstrate the usefulness with a new Bluetooth profile called Medical Device Profile (MDP) on the Medica trade fair 14-11-2007. The MDP profile that will be released in turn of the year 07/08 is primarily intended to be used in medical devices.

The demonstration is to be made by building a demonstrator consisting of an application running on a smartphone with Bluetooth abilities. The application will handle the Bluetooth connection between the smartphone and the oximeter, the data parsing, other functionalities and presenting the figures received form the oximeter in a Graphical User Interface (GUI).

The final demonstrator consists of a smartphone application programmed in Symbian C++, which communicates with the oximeter using the Bluetooth Serial Port Profile (SPP). The application runs on UIQ 3.0 based smartphones and displays heart rate and %SpO2 (The percentage of oxygen saturation in blood), which the application receives from the oximeter.

The original idea was to use the MDP profile by porting Cybercom’s C version of the MDP to Symbian OS for the Bluetooth communication, depending on various factors described more in detail inside the report this was not done. The purpose of the project was still reached, even though another profile then MDP was used. This was done by replacing the considered oximeter for an older version which is using the SPP. By using SPP the same result will be demonstrated but with an older technique, and this will achieve the same result on Medica trade fair. The project was demonstrated on Medica with a successful result.

Preface

This bachelor thesis work has been both fun and very instructive for us. For this we have many people to be thankful to who has been very helpful. To begin with we would like to thank Kristian Palm, Häkan Johansson and Magnus Öhrlund who gave us the chance to work with the demonstrator project on Cybercom Sweden West.

Big thanks to Thomas Bredhammar for all guidance during the project, and our examiner Henrik Eriksson for valuable comments on the report.

Last we would like to thank all personnel in Cybercom’s Huskvarna office, for all kinds of guidance during this project.
## List of abbreviations

### Bluetooth profiles

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Profile Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2DP</td>
<td>Advance Audio Distribution Profile</td>
</tr>
<tr>
<td>AVCTP</td>
<td>Audio/Video Control Transport Profile</td>
</tr>
<tr>
<td>AVDTP</td>
<td>Audio/Video Distribution Transport Profile</td>
</tr>
<tr>
<td>AVRCP</td>
<td>Audio/Video Remote Control Profile</td>
</tr>
<tr>
<td>BIP</td>
<td>Basic Imaging Profile</td>
</tr>
<tr>
<td>BPP</td>
<td>Basic Printing Profile</td>
</tr>
<tr>
<td>BNEP</td>
<td>Bluetooth Network Encapsulating Protocol</td>
</tr>
<tr>
<td>CIP</td>
<td>Common ISDN Access Profile</td>
</tr>
<tr>
<td>CTP</td>
<td>Cordless Telephony Profile</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Profile</td>
</tr>
<tr>
<td>GAP</td>
<td>Generic Access Profile</td>
</tr>
<tr>
<td>GAVDP</td>
<td>Generic Audio/Video Distribution Profile</td>
</tr>
<tr>
<td>GOEP</td>
<td>Generic Object Exchange Profile</td>
</tr>
<tr>
<td>HFP</td>
<td>Hands-Free Profile</td>
</tr>
<tr>
<td>HCRP</td>
<td>Hard Copy Cable Replacement Profile</td>
</tr>
<tr>
<td>HSP</td>
<td>Headset Profile</td>
</tr>
<tr>
<td>HID</td>
<td>Human Interface Device Profile</td>
</tr>
<tr>
<td>ICP</td>
<td>Intercom Profile</td>
</tr>
<tr>
<td>MDP</td>
<td>Medical Device Profile</td>
</tr>
<tr>
<td>OBEX</td>
<td>Object Exchange Profile</td>
</tr>
<tr>
<td>OPP</td>
<td>Object Push Profile</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Networking Profile</td>
</tr>
<tr>
<td>SDP</td>
<td>Service Discovery Protocol</td>
</tr>
<tr>
<td>SPP</td>
<td>Serial Port Profile</td>
</tr>
<tr>
<td>SYNC</td>
<td>Synchronization Profile</td>
</tr>
<tr>
<td>UIQ</td>
<td>User Interface Quartz</td>
</tr>
</tbody>
</table>

**Bluetooth layers**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2CAP</td>
<td>Logical Link Control and Adaptation Protocol</td>
</tr>
<tr>
<td>MCAP</td>
<td>Multi-Channel Adaptation Protocol</td>
</tr>
<tr>
<td>RFCOMM</td>
<td>Radio Frequency Communication</td>
</tr>
</tbody>
</table>

**Others**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFH</td>
<td>Adaptive Frequency Hopping</td>
</tr>
<tr>
<td>AO</td>
<td>Active Object</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BSD</td>
<td>Berkley Socket Distribution</td>
</tr>
<tr>
<td>Bluetooth SIG</td>
<td>Bluetooth Special Interest Group</td>
</tr>
<tr>
<td>GFSK</td>
<td>Gaussian Frequency Shift Keying</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IR</td>
<td>Infra-Red</td>
</tr>
<tr>
<td>ISDN</td>
<td>Integrated Services Digital Network</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial, Scientific, and Medical</td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Messaging Service</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSAL</td>
<td>Operating System Abstraction layer</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>ULP</td>
<td>Ultra Low Power</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra Wide Band</td>
</tr>
</tbody>
</table>
## Contents

List of Figures ................................. 1
List of Tables ................................. 3

1 Introduction ................................. 5
  1.1 Background ............................... 5
  1.2 Purpose .................................. 5
  1.3 Structure ................................ 5
  1.4 Sources ................................ 6

2 Task .......................................... 7
  2.1 Requirements ............................. 11
  2.2 Preliminary study ......................... 12

3 Theory ........................................ 15
  3.1 Bluetooth ................................ 15
    3.1.1 Background ......................... 15
    3.1.2 Bluetooth special interest group .... 16
    3.1.3 Bluetooth technology ............... 16
    3.1.4 Core specification .................. 16
    3.1.5 Bluetooth Profiles .................. 17
    3.1.6 Comparison between wireless technologies 18
    3.1.7 The future of bluetooth ............ 19
  3.2 Symbian OS ............................... 20
    3.2.1 Background .......................... 20
    3.2.2 Active objects ...................... 20
    3.2.3 Descriptors ......................... 21
    3.2.4 Error handling ...................... 22
    3.2.5 Symbian signed ...................... 24

4 Application ................................. 25
  4.1 Architecture ............................ 25
  4.2 Implementation .......................... 28
  4.3 Development tools ....................... 31
    4.3.1 Software Development Kit .......... 31

4.3.2 Integrated Development Environment .................... 34
4.3.3 Bluetooth sniffer ........................................... 34

5 Result 37
5.1 Requirements fulfilled ........................................... 37
5.2 Requirements not fulfilled ....................................... 37
5.3 Demonstrator ....................................................... 38

6 Discussion 39
6.1 Conclusion .......................................................... 39
6.2 Reason for not fulfilling requirements ............................. 39
6.3 Suggestions to alterations of the application ...................... 39
6.4 Difficulties during project .......................................... 41
   6.4.1 Capabilities .................................................... 41

Bibliography 51
List of Figures

2.1 Original use case architecture .......................... 7
2.2 Final use case architecture ............................ 8
2.3 Original high-level use case ............................ 9
2.4 Final high-level use case ............................... 10
3.1 Piconets and scatternet ................................. 18
3.2 Active object high-level view ........................... 21
3.3 Exception with memory leak ............................. 23
3.4 Exception without memory leak ......................... 23
4.1 Application structure .................................. 25
4.2 Manager class flow chart ............................... 26
4.3 Oximeter data format .................................. 27
4.4 Application flow chart .................................. 29
4.5 Application: main views ............................... 29
4.6 Application: menu / device search .................... 30
4.7 Application: log view .................................. 31
4.8 UIQ emulator ........................................... 33
4.9 Carbide.C++ debugging ............................... 34
4.10 Sniffer session ........................................ 35
5.1 Demonstrator ........................................... 38
6.1 Desirable application architecture ..................... 40
List of Tables

3.1 Bluetooth device classes .................................... 17
3.2 A comparison between wireless technologies .............. 18
4.1 Smartphone manufacturers .................................. 32
Chapter 1

Introduction

1.1 Background

Cybercom Sweden West is a consulting company with an office in Huskvarna that specializes in Bluetooth solutions. They are an associate level member of the Bluetooth SIG (Chapter 2.1.1) and members of the medical devices profile workgroup[56].

The medical device profile (MDP) is a new Bluetooth profile that provides a standard of wireless communication for medical and fitness devices. The profile is designed to have a low power-consumption and a high level of security. The MDP profile is dependent on the Bluetooth Multi-Channel Adaptation Protocol (MCAP) which is another newly developed Bluetooth profile[62]. MCAP is created to support MDP but may be the basis for other future Bluetooth profiles. Cybercom Sweden West was developing an implementation of the MDP and wanted to demonstrate it at the international trade fair and congress: Medica in Düsseldorf (14-11-2007)[60, 61].

Cybercom wanted a demonstrator consisting of a Bluetooth enabled oximeter and a smartphone with Bluetooth technology to display heart rate and oxygen saturation transferred from the oximeter.

1.2 Purpose

The purpose is to provide Cybercom Sweden West with a demonstrator which illustrates the usefulness of the coming Bluetooth MDP to Medica trade fair (14-11-2007).

1.3 Structure

In the beginning the reader is given a short background and a description of the project with the requirements followed by a theory part to help the reader understand the substance of the report. After the theory part a chapter explains what has been accomplished and the report is ended with the result. The reader is expected to have basic knowledge in computer SW/HW.
The text in this report is written in times new roman, references in Vancouver style and the date format used is ISO-8601 international date format.

**Task**
This section introduces the reader to the thesis project, its requirements and explains the work that has been done during the preliminary study.

**Theory**
This section is a theoretical part which introduces the reader to Bluetooth and Symbian which are the two underlying technologies used in this thesis project.

**Application**
This section gives a detailed explanation of the applications architecture, implementation and ends with a discussion about the development tools that have been used.

**Result**
The Result section lists which of the project requirements that were fulfilled and which were not.

**Discussion**
The Discussion section contains the conclusion of the thesis rapport and explains why some of the requirements were not fulfilled, the section also suggests alterations to the design if development of it should be continued.

### 1.4 Sources
A majority of the references are internet addresses, the reason for this is that platforms like Bluetooth, Symbian OS and UIQ are developing continuously and thereby making it very difficult to obtain updated material in print.
Chapter 2

Task

The original use case that formed the basis of the preliminary study is presented in Figure 2.3. A Bluetooth enabled oximeter with the Bluetooth MDP implemented was to send information about heart rate and the oxygen saturation of the blood to a Bluetooth enabled smartphone. The task would be to implement MDP on the smartphone and to write an application on top of the MDP that would display heart rate and oxygen saturation to the user i.e. figure 2.1. The application should also record a log file of the heart rate and be able to send the file to a FTP-site or a database on a remote computer via GPRS or UMTS. The preliminary study revealed that the use case in Figure 2.3 would

![Application Diagram](attachment:image.png)

Figure 2.1: Original use case architecture

be very difficult to realize and that a new use case had to be developed Figure 2.4. The oximeter was replaced with an oximeter that used the Bluetooth Serial Port Profile (SPP) to communicate. SPP was already implemented in the smartphone which meant that the MDP layer could be removed and that the


7
demonstrator would have a better chance of being done in time for the Medica trade fair i.e. figure 2.1.

![Diagram](image)

Figure 2.2: Final use case architecture
Figure 2.3: Original high-level use case
Figure 2.4: Final high-level use case
2.1 Requirements

The requirements where set to meet the purpose of the application, even though all the requirements do not have the same priority they are all listed in the same manner below. Some requirements where not set from the beginning e.g. filtering for devices in device search this was shown to be necessary because of the amount of Bluetooth devices in the Medica trade fair Bluetooth area. The two requirements server application and send data to server application by using GPRS and UMTS was a secondary priority if there was time left before the trade fair and the smartphone-oximeter communication was finished. When it was realized that the server application would not be finished in time, send the log file to a remote device and log file name suffix with date and time were introduced as two new requirements.

Implement MDP

The application shall implement MCAP and MDP in the application.

Search for the device

The application shall be able to search for available Bluetooth devices.

Display heart rate and oxygen saturation

The application shall display heart rate and oxygen saturation.

Display a graph

The application shall be able to display a graph which displays heart rate and time.

Record a log file

The application shall be able to store a file containing heart rate during a predetermined duration.

Send the log file to a remote device

The application shall be able to send the log file to a remote device using e.g. Bluetooth.

Log file name suffix with date and time

The application shall name the log file with a suffix containing date and time.

Send data to server application by using GPRS or UMTS

The application shall be able to send the log file to an ftp-site or a database using GPRS or UMTS.
Server application

A pc application shall be developed with a database to store and display the log files that has been sent from the smartphone.

Filtering of devices in device search

The application shall only display oximeters or devices which are in the same class, when it is searching for Bluetooth devices.

2.2 Preliminary study

Cybercom had developed a version of the MDP that used an Operating System Abstractation Layer (OSAL) to make porting of the profile easier. The OSAL would have to be modified to integrate MDP into Symbian OS. Bluetooth communication would be accessed through a socket interface. The MDP depends on another new Bluetooth protocol named MCAP that provides MDP with capabilities such as multiple simultaneous data channels to one device and automatic reconnection to devices that has been disconnected. MCAP/MDP resides above the L2CAP layer in the Bluetooth architecture.

The problem then arose about how to access functions in the MDP that was written in ANSI C from the application written for the Symbian OS in Symbian C++. It was decided to try to isolate the MCAP/MDP part and access it through a TCP socket interface.

Symbian handles multithreading by the use of active objects. The OSAL was constructed for operating systems that is using a thread bases multithreading system and proved to be very difficult to change to use active objects instead. The task was therefore changed to implementing the functionality of MCAP and MDP in Symbian C++.
Rewriting MCAP and MDP proved to be too time consuming and a demonstrator would not have been done in time for the Medica trade fair. It was then decided that the task was to be changed to writing an application that used a different hardware that communicated through Bluetooth SPP. By using SPP we could implement our Bluetooth communication using RFCOMM (Radio Frequency COMMunication) that was already implemented in Symbian.
Chapter 3

Theory

3.1 Bluetooth

Bluetooth is a specification for wireless communication, it is designed to have a small form-factor, low power consumption and a low price. The technology uses the frequency band between 2.4-2.85Ghz. The band is called the Industrial, Scientific, and Medical (ISM) band and is unlicensed in most parts of the world[20]. The Bluetooth technology has expanded to many areas and over 1.5 billion Bluetooth enabled devices has been sold to date. Some of the areas of use are listed below.

- Mobile phones and accessories e.g. headsets and headphones.
- Personal computers and accessories e.g. Bluetooth dongles, keyboards and mice.
- Automotive e.g. GPS and handsfree.
- Game controllers e.g. Sony sixaxis and nintendo wii remote.
- Medicinal and fitness e.g. oximeters.
- Office e.g. printers and speakerphones.

3.1.1 Background

The development of Bluetooth began in 1994 when Ericsson Mobile Communications conducted a study to find a suitable replacement to infra-red (IR) links as a cable-replacer for mobile-phones and their peripherals. The advantage of a radio-link such as Bluetooth over an IR-link is that no line of sight is needed between the transceiver and receiver. Interest grew for the technology and in February 1998 the Bluetooth Special Interest Group (SIG) was founded and the first core specification was released in July 1999. The Bluetooth name was taken from a 10th century Danish king called Harald Blåtand that supposedly united Norway and Denmark. The name draws parallels to the unifying of the telecom and computer industries into one standard for short distance wireless communication that Bluetooth is intended to do[19].
3.1.2 Bluetooth special interest group

Bluetooth SIG is an organization that develops new Bluetooth specifications and work to strengthen the Bluetooth brand. Bluetooth SIG was created in 1998 by Ericsson, IBM, Intel, Nokia and Toshiba, those companies together with Microsoft, 3Com Lucent and Motorola that joined in 1999 were the SIG promoters[21]. Today the SIG promoter companies consists of

- Ericsson AB
- Intel Corporation
- Lenovo Pte Ltd.
- Microsoft Corporation
- Motorola Inc.
- Nokia Corporation
- Toshiba Corporation

Promoter is the highest level in a hierarchy of membership levels that consists of promoters that are represented at the board of directors, associate members that can influence the specifications and get to review specifications before they are finalized\(^1\) and adopter members that can use nearly finalized versions of the specifications\(^2\). Every company that implements Bluetooth technology must be at least an adopter member of the Bluetooth SIG, and to this date there are over 9000 members. All products that use Bluetooth technology has to undergo a qualification program to ensure that the product complies with the Bluetooth specification.

3.1.3 Bluetooth technology

Bluetooth SIG develops specifications for implementation of Bluetooth technology, the specifications are divided into two parts, a core specification that all Bluetooth products has to comply with and a profile part that defines standards for specific functions e.g. the Headset profile (HSP) defines how a headset should communicate with a Bluetooth device.

3.1.4 Core specification

The latest version of the Bluetooth core specification, version 2.1 was released in July 2007, the main features of Bluetooth 2.1 are described below.

**Transmission rate**

Since core specification version 2.0 there has been two transmission rate modes defined. The basic rate mode with a transmission rate of 1 Mbps and the enhanced data rate with a transmission rate up to 3Mbps.

---

\(^1\)From version 0.50
\(^2\)From version 0.90
Frequency hopping

As several other technologies communicate on the 2.4 GHz band, interference can be a problem. Bluetooth technology solves this by implementing frequency hopping. The Bluetooth transmitter is changing carrier frequencies 1600 times per second, following a pre-selected sequence of 79 carrier frequencies. By hopping between frequencies, the device lessens the chance of transmitting on a frequency with interference\[21\]. Bluetooth version 1.2 introduced adaptive frequency hopping (AFH), a technique that detects if a frequency is interfered and then removes that frequency from the hopping sequence.

Power consumption

The transmitters and receivers are divided into three power classes depending on transmitting power and receiving sensitivity\[19, 20\].

<table>
<thead>
<tr>
<th>Class</th>
<th>Maximum transmitter power</th>
<th>Maximum range (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>100mW (20dBm)</td>
<td>100</td>
</tr>
<tr>
<td>Class 2</td>
<td>2.5mW (4dBm)</td>
<td>10</td>
</tr>
<tr>
<td>Class 3</td>
<td>1mW (0 dBm)</td>
<td>1</td>
</tr>
</tbody>
</table>

Security

Ad hoc network

Ad hoc is latin and means “for this purpose”. An ad hoc network is a network where the devices connect directly to each other without the use of a base station. When two or more Bluetooth devices connect to each other a piconet is created. A piconet can support a total of eight devices. One of the devices is selected to be the master of the piconet and the connected devices all synchronize their clock with the master clock and follow the same frequency hopping pattern. If two piconets are connected they form a scatternet\[55\]. A device can be slave in one piconet and a master in the other as show in figure 3.1 where node 2 is master of piconet 2 but a slave in piconet 1. Piconets communicate with each other using time multiplexing.

3.1.5 Bluetooth Profiles

Bluetooth profiles are created to improve the interoperability between devices. The profiles define standards for common use cases. New profile specifications are continually developed and updated by the Bluetooth SIG to comply with requests from the members. The profile specifications defines dependencies on other profiles, all profiles are dependent on the Generic Access Profile (GAP) where information about how to discover and connect to other devices are described.

A complete list of the profiles and protocols can be found in, respectively, Appendix A and Appendix B.
3.1.6 Comparison between wireless technologies

There are several other technologies for wireless communication, this section will compare a few of the more common technologies. Infrared link is a cable replacement technology that is using light to communicate. Light transmitting and receiving limits the range and makes the technology sensitive to obstacles, it is also very directional. A positive effect of the directionality and limited range is that it makes it difficult to listen in on the communication, therefore the technology is used for electronic payment.

Zigbee is a low power wireless technology that also communicates on the 2.4 GHz band. The main focus for Zigbee technology is remote sensors and control.

802.11g is the common standard for wireless broadband. It has focus on capacity and has a power consumption that is several times greater than Bluetooth and Zigbee.

Table 3.2: A comparison between wireless technologies

<table>
<thead>
<tr>
<th></th>
<th>Bluetooth</th>
<th>802.11g</th>
<th>Zigbee</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>up to 3Mbps</td>
<td>54 Mbps</td>
<td>up to 250kbps</td>
<td>up to 16Mbps</td>
</tr>
<tr>
<td>Power consumption</td>
<td>low</td>
<td>higher</td>
<td>very low</td>
<td>low</td>
</tr>
<tr>
<td>Range</td>
<td>up to 100m</td>
<td>140m</td>
<td>100m</td>
<td>1m</td>
</tr>
</tbody>
</table>
3.1.7 The future of bluetooth

The Bluetooth SIG will likely incorporate Ultra Wide Band (UWB) into the next version of the core specification. Bluetooth SIG has joined forces with Wimedia alliance[49] to develop a Bluetooth version that is using UWB technology. Tests has shown that transmission speeds could be increased to 480 Mbps at close ranges[50]. Another feature that is likely to be described in the next specification is Ultra Low Power Bluetooth (ULP)[51]. The ULP Bluetooth technology will provide Bluetooth functionality to devices where battery life is essential e.g. button cell battery devices like wrist watches and medical and fitness sensors.
3.2 Symbian OS

3.2.1 Background

The Symbian OS is an OS developed and distributed by Symbian LTD. Symbian OS is today by far the most commonly used OS for smartphones. The second quarter of 2007 Symbian LTD distributed 75% of the sold smartphones worldwide with OS, which sums up to over 18 million licenses sold in three months[7]. Symbian origins from another similar OS developed for use in handheld devices and were called EPOC. EPOC was developed by Psion which is one of the companies involved in the alliance which founded Symbian LTD. The other companies in the alliance were Ericsson, Nokia and Motorola[2].

Symbian OS is developed like most of the commercial OS available for desktops, meaning it supports preemptive multitasking, multithreading and is memory protective. Although there are some differences from a laptop computer OS in design constraints, e.g., limited resources and battery power. The constraints are like this because it shall be able to run on small handheld devices, these constraints have led to some specific features which are discussed later.

Symbian OS implements Active Objects (AO) and asynchronous functions to support multiple operations in parallel, even though Symbian supports multiple threads within the same process, they are not commonly used. The next section describes the concept of AO in more detail, because a good understanding for AO is important in the process of developing software in Symbian. Other important and Symbian specific features discussed in this chapter are cleanup stack, descriptors and Symbian signed [3, 8].

3.2.2 Active objects

Symbian OS applications normally run on one single thread, to manage this and still have the characteristics of multithreading, AOs are introduced. AOs are classes implemented by the developer self and inherit from CActive. AOs simplify asynchronous programming which is used very frequently in Symbian OS. To achieve this sort of multithreading with AOs with only one thread the OS also needs the active scheduler. This is a scheduler that is event-driven and switches between different AOs as they occur. It is possible to implement thread switching in Symbian by using the class RThread this is usually not done and is considered complicated. Functions which are included in the different Symbian APIs are usually implemented in one synchronous and one asynchronous version. The asynchronous version is implemented with an extra parameter which is a TRequestStatus object. TRequestStatus is a status variable which could be “polled” to retrieve execution status of the function i.e. KErrNone if function returned correctly. The asynchronous function must not be implemented in connection to an AO, but to “poll” this variable after calling the asynchronous function would mean the same thing as calling the synchronous version from the beginning. The correct way of using the asynchronous function would be to call CActive( ) which is derived from CActive this indicates that the AO has a asynchronous function to handle. The thread will start the execution of the asynchronous function, and immediately after it has been called it will return the status variable (pending) which is derived from the class TRequestStatus. This tells the calling thread that the asynchronous function are executed and
now runs in parallel with the calling thread but on a background thread. Now the application thread can do other things while the asynchronous function runs in background and returns ending status when it is done by invoking the request semaphore. This can be compared with a normal synchronous function which would have forced the calling thread to execute the function until it is completely finished[8, 13].

![Diagram of Active Object high-level view](image_url)

Figure 3.2: Active object high-level view

[15]

If the active scheduler has noticed an event by the request semaphore, the next thing it will do is to search the event handler and look for AOs which are set to active by SetActive( ). Then pick the one which not equals to status pending and call this AOs RunL( ) function. Because the event loop in the active scheduler is driven by one single thread the implemented scheduling method is a non-preemptive method, meaning it will not be able to interrupt the running AO for execution of another one. The last calling AO therefore has to wait until running AOs synchronous function RunL( ) is completed. RunL( ) needs to be implemented with a lot of consideration. Optimal is to put as much functionality as possible into the AOs asynchronous functions, and minimize the RunL( ). Because as long as the RunL( ) is in execution all the other AOs will be prevented from execution time.

### 3.2.3 Descriptors

A specific Symbian method used to keep memory management safe is descriptors. Instead of Strings in C, Symbian are using descriptors. The first 4-bytes of each descriptor are what their name reveals self-describing. The first 28 bits
reveals the length of the descriptor and the last 4-bits of the starting 4-bytes tells which type of descriptor it is. These first describing 4 bytes are later used in the memory-layout.

By knowing the length of the each descriptor, Symbian OS keeps memory from being overrun. If the descriptor operates outside its allocated memory area, the OS treats this as programming error and raises an exception which is called panic. A panic terminates the program immediately[9, 18].

3.2.4 Error handling

The memory management and error handling is important to small handheld devices like smartphones, which are considered to have limited recourses. Other characteristic too handheld devices are that they are often supposed to run for month, even years without having to reboot. This is why the following methods are essential to master if one wants to develop efficient software in Symbian. There are a few different ways of implementing error handling in Symbian, the most basic is to handle the majority of the API functions with return codes. This is not a unique thing for Symbian but still very useful, examples of return codes are KErrNone and KErrNoMemory. This might not be the most efficient way of implementing error handling but still useful. A more sophisticated and Symbian specific way of handling errors and recover from them is the Leave/Trap mechanism. Leave is very much like exceptions in C/C++, in fact it is a lightweight form of exceptions. The reason of this is that exceptions were initially considered to use up to much memory and when Symbian C++ was written throw/catch was not part of the C++ standard.

Important to know when developing software for Symbian is that all functions with the suffix “L” in their names are implemented with the possibility that they might leave, the suffix “LC” means the same but it has already pushed the result to the cleanup stack (Explained later in this section). When exception is raised meaning that a leave has occurred the program execution will schedule the trap harness, which decides what action to take, this is further explained in Figure 3.1. Trap harness is the function that decides what action to take. Every function which has the possibility to leave does not need to have a trap harness, as long as there is one “higher up” in the “function chain”, meaning if Func1L( ) calls Func2L( ) and this continues up until e.g. Func10L( ) it is enough that Func1L( ) has a trap harness. There are a special trap harness for every AO, which is implemented as a virtual method called RunError( ), to implement your error handling for a AO this function needs to be overwritten. The action to be taken on leave is set by programming the TRAP or the TRAPD macro (trap harness). These two macros takes two parameters each, the only difference is that with TRAPD the leave code parameter is declared by the macro it self. The other parameter is the invoked function. These leave codes are not only helpful during error handling on target, they are also helpful in the software developing phase. If leave occurs it is easy to find what is wrong by searching the error codes in Symbian documentation or in the SDK help for SDK specific leaves. A leave will immediately interrupt the program execution, by returning every function until a TRAP/TRAPD (trap harness) is found. A result of this is that allocated memory might not be de-allocated. This situation is called a memory leak, and the cleanup stack is used to solve this problem. Consider the following two examples for explanation [12, 8]. Make sure the cleanup stack is
Figure 3.3: Exception with memory leak

```cpp
MyFunc1 ( )
{
    // Trap harness
    TRAP(LeaveCode, MyFunc2L( ));
    if(LeaveCode)
        // Action to take on leave
    else
        // KErrNone
    MyFunc2L( );
        
        MyClass *p = new(Eleave) MyClass;
        MyFunc3L( );
        Delete p;
        MyFunc3L( );
        // Leave invoked
}
```

Figure 3.4: Exception without memory leak

```cpp
MyFunc1 ( )
{
    // Trap harness
    TRAP(LeaveCode, MyFunc2L( ));
    if(LeaveCode)
        // Action to take on leave
    else
        // KErrNone
    MyFunc2L( );
        
        MyClass *p = new(Eleave) MyClass;
        CleanupStack::PushL(p);
        MyFunc3L( );
        CleanupStack::PopAndDestroy();
        Delete p;
        MyFunc3L( );
        // Leave invoked
}
```
balanced all the time by pop items which no longer constitute a memory leak threat.

3.2.5 Symbian signed

Symbian signed was introduced in version 9.0 this is a digital certificate, and it holds information about the applications origin which makes it traceable. Signing also gives the developer permission to utilize APIs which are modifying user data, functions which could result in a cost for the user, functions which gives access to the network or function which might change the appearance of applications installed on target (i.e. APIs which are protected with capabilities.). All this is to prevent viruses to spread among the mobile devices which use Symbian OS. Capabilities are a way to increase platform security, the excess of different capabilities is divided in to different levels. Basic capabilities can normally be used without any signing, while some capabilities demand a digitally signed application. Some capabilities might even demand manufacturer approval. The developer’s only way of being certain if the application demands signing or not even though no system- or device manufacturer capabilities are used is to install the .sis on target and test. Since this not only depends on Symbian OS capabilities, the Software Development Kit (SDK) and the device manufacturer also plays a part in this decision. E.g. UIQ 3.0 does not demand applications to be signed, opposed to S60 3rd edition which demands signing even if it could be self signed. The cost of signing small applications for a layman could be rather substantial, at least if system- or device manufacturer capabilities are used, which demands developers ID. The developer ID is a way of proving the identity of the developer, this cost somewhere around $200-$400. With this ID the developer is allowed to sign an unlimited amount of applications, as long as the ID is updated on annual basis[16, 5].
Chapter 4

Application

4.1 Architecture

The Structure of the application can be divided into three layers.

![Application Structure Diagram]

**Figure 4.1: Application structure**

**Active Objects**

The active object classes are using a EPOC socket (ESOCK) for the communication with the Bluetooth module in the smartphone. ESOCK is a Symbian OS socket API which is similar to Berkley Socket API created by Berkley Socket Distribution (BSD)[64], but except for the TCP/IP APIs there are also IR, Bluetooth and USB protocol included in ESOCK API[63]. A socket-server is opened when the class is instantiated and the three active objects that handles the Bluetooth communication all use that socket-server. The classes are called from the manager class and are asynchronous. The function returns immediately and gives the resources to another object. A callback function which is implemented in the manager class is called when the active operation is completed.

**Manager class**

The manager class is designed to provide a simplified interface for the GUI. The manager is implemented as a state machine that handles the connection and
configuration of the oximeter and calls a callback function that is implemented in the GUI. The configuration of the oximeter is done by sending a message to

START

CONNECT

CONNECT OK?

YES

TRANSMIT CONFIG MESSAGE

CONFIG MESSAGE ACK?

NO

YES

RECEIVE DATA

PARSE DATA

GUI CALLBACK FUNCTION

END

Figure 4.2: Manager class flow chart

the oximeter within 5 seconds of the connection. The oximeter is configured to transmit a package of three bytes every second. Figure 4.3 illustrates what these bytes contain.
Packet Description

Three types of data are transmitted 1 once per second.

### Byte 1 - STATUS

<table>
<thead>
<tr>
<th>BIT7</th>
<th>BIT6</th>
<th>BIT5</th>
<th>BIT4</th>
<th>BIT3</th>
<th>BIT2</th>
<th>BIT1</th>
<th>BIT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNSD</td>
<td>OOT</td>
<td>LPRF</td>
<td>MPRF</td>
<td>ARTF</td>
<td>HR8</td>
<td>HR7</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Bit 7 is always set*

### Byte 2 - HEART RATE

<table>
<thead>
<tr>
<th>BIT7</th>
<th>BIT6</th>
<th>BIT5</th>
<th>BIT4</th>
<th>BIT3</th>
<th>BIT2</th>
<th>BIT1</th>
<th>BIT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>HR6</td>
<td>HR5</td>
<td>HR4</td>
<td>HR3</td>
<td>HR2</td>
<td>HR1</td>
<td>HR0</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Bit 7 is always clear*

### Byte 3 - SPO2

<table>
<thead>
<tr>
<th>BIT7</th>
<th>BIT6</th>
<th>BIT5</th>
<th>BIT4</th>
<th>BIT3</th>
<th>BIT2</th>
<th>BIT1</th>
<th>BIT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP6</td>
<td>SP5</td>
<td>SP4</td>
<td>SP3</td>
<td>SP2</td>
<td>SP1</td>
<td>SP0</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Bit 7 is always clear*

The following are all active high:
- **SNSD:** Sensor Disconnect
- **OOT:** Out Of Track
- **LPRF:** Low Perfusion
- **MPRF:** Marginal Perfusion
- **ARTF:** Artifact
- **HR8 - HR0:** Heart Rate
- **SP6 - SP0:** SpO\textsubscript{2}

- An absence of signal, meaning the sensor is disconnected
- An absence of consecutive good pulse signals
- Amplitude representation of low signal quality
  
- **HR8 - HR0:** Standard 4-beat average values not including display holds
- **SP6 - SP0:** Standard 4-beat average values not including display holds

Figure 4.3: Oximeter data format
GUI

The GUI is designed to have an easy user interface (UI), with both touch screen and keypad user input alternatives. Even though selection in the application menu is only supported by the touch screen. But for authentication the pass code may be entered by the keypad or the touch screen. The following four classes form the basis of all the different Symbian GUI platforms and thereby also this UIQ based GUI:

- **Application class**
  Program instantiation.

- **Document class**
  Handles data

- **UI class**
  Instantiate the view and handle program logic (User inputs).

- **View class**
  Main view control.

This application GUI is basically designed with one base view, which is overwritten when new input is given to the GUI. In the application- and the document class only minor details are changed from the wizard build\(^1\). The document class is more often used when data needs to be stored. In opposite from these two, the next two are classes which contains a lot of application specific code. The UI class defines which action to take on different program-/user input. The view class includes these functionalities:

- **CQBTUISelectDialog**
- **CQIKSENDasDialog**
- **Save logfile**
- **Logfile naming format**

CQBTUISelectDialog and CQIKSENDasDialog are both UIQ specific classes which are used to implement the Bluetooth search dialog and the send alternative dialog. With save log file means the functionality to store oximeter input values to logfile. The logfile are named after this format: Cybercom year/month/hour/min.

### 4.2 Implementation

The flowchart explains the components and the functionality of the application.

---

\(^1\)Wizard build is included in the Carbide.c++, helps building the most basic GUI application.
Main view

The main view is the first thing that is displayed when the application is started. Main view displays heart rate and oxygen saturation when the application is connected and is receiving data. Before a connection is made the main view displays NOT CONNECTED and if the application receives a bad signal or if the oximeter is detached from the fingertip of the user then the main view displays NO SIGNAL.

Select device

When connect is selected in the menu, a UIQ specific class called CQBTUIServiceSelectDialog is used. The class provides a list of available Bluetooth devices and
returns the Bluetooth address of the selected device.

![Figure 4.6: Application: menu / device search](image)

**Connect**

The Bluetooth address obtained with CQBTFUISelectDialog is used when the connect request function is called in the manager class. The manager enters the state machine where it connects to the oximeter. The next step is to send a configuration message to the oximeter to select the transmission format of the data. The manager then reads the answer to the configuration message, if it is successful the manager will start to send data from the oximeter to the GUI. The GUI will then display heart rate and oxygen saturation in the Main view.

**Log view**

When log 10 seconds is selected in the menu and the device is connected to the oximeter the application enters log view. Log view displays a graph with heart rate value on the Y-axis and time in seconds on the X-axis. 10 heart rate values are displayed and are also stored into an array. If the application receives a bad signal or if the oximeter is detached from the fingertip of the user then the main view displays NO SIGNAL.
Save Logfile

Save logfile writes the array from log view to a file and adds date and time as a suffix to the file name.

Select Device and send Logfile

A UIQ specific class named CQikSendAsDialog is then used to send the log file to a remote device. The class first displays a dialog with four options of transferring the log file to the remote device: Bluetooth, SMS, MMS and IR. The user then selects one of the four ways and sends the Log file. The application returns to the main view after sending the file.

4.3 Development tools

4.3.1 Software Development Kit

The application consists of a GUI which is implemented by using UIQ 3.0 SDK. This is a software layer on top of Symbian OS which helps different smartphone manufacturers to design their UI individually. UIQ technology has been a Symbian Ltd subsidiary until November 2006, when Sony Ericsson bought it and really brought up the struggle of smartphone UI platforms mainly with Nokias S60. Another new competitor UIQ needs compete with is Apples Iphone UI, this is to be done by increasing the number of phone manufacturers involved in UIQ. First out was Motorola which bought half the shares of UIQ technology from Sony Ericsson in October 2007 and the two company’s future plan is to get more manufacturers involved in UIQ to make it a more powerful UI, and get more people involved in UIQ based feature development[14].

By using UIQ 3.0 specific APIs the application becomes limited to smartphones which supports this version of the SDK, a full list of all available smartphone which supports this SDK is shown in table 4.1. Different display sizes could also create problems for this application, this is a doubtful design and the reason is handled in the discussion. The applications two main criteria’s are simplicity to use, and interesting design. The following smartphone manufacturers
Table 4.1: Smartphone manufacturers

<table>
<thead>
<tr>
<th></th>
<th>UIQ 3.2</th>
<th>UIQ 3.1</th>
<th>UIQ 3.0</th>
<th>UIQ 2.1</th>
<th>UIQ 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sony Ericsson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1</td>
<td>W960i</td>
<td>W950</td>
<td>P990</td>
<td>M600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P910</td>
<td>P900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola</td>
<td>Motorizer Z10</td>
<td>Motorizer Z8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M1000</td>
<td>A1000</td>
<td>A925</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A920</td>
</tr>
<tr>
<td>Arima</td>
<td></td>
<td></td>
<td>U308</td>
<td>U300</td>
<td></td>
</tr>
<tr>
<td>BenQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P30</td>
</tr>
</tbody>
</table>

This SDK is free and available for download on UIQ's homepage (http://developer.uiq.com/devtools_uiqsdk.html). It includes full documentation of all the UIQ APIs, example applications and an emulator. In this project an extension package of this emulator was used called M600i extension pack. Figure 5.1 show a “print screen” of the emulator. This emulator was used during the project with great satisfaction. In combination with a USB Bluetooth dongle and the NOKIA S60 BT driver the emulator was used to connect to the oximeter for debugging purposes. The emulator contains a text based program called eshell, which are used for command line debugging when application misses GUI.
Figure 4.8: UIQ emulator
4.3.2 Integrated Development Environment

For compilation and debugging we used Carbide.c++ which is an Integrated Development Environments (IDE), UIQ supports the following IDEs:

- Carbide.c++
- CodeWarrior 3.0 and 3.1
- Visual Studio .NET 2003
- VistaMax 2.0

Carbide.c++ is an eclipse-based development environment which focuses on mobile development; it is used to develop Symbian C++ applications. There are a few different versions of this IDE, the express version was used in this project, and this because it is free for download and considered to meet all requirements for debugging. [59]

Figure 4.9 shows the project application being debugged with the development environment Carbide.c++ and the emulator from UIQ.

![Figure 4.9: Carbide.C++ debugging](image)

4.3.3 Bluetooth sniffer

One of the tools used for developing the application was the Frontline FTS4BT FTS4BT Wireless Bluetooth Protocol Analyzer & Packet Sniffer. The Frontline
soft- and hardware made it possible to analyze and record all data sent between two Bluetooth enabled devices. The sniffer has to know the Bluetooth addresses of the monitored devices and it has to be active when the devices are being paired. A recorded session is displayed in fig 4.10.

Figure 4.10: Sniffer session
Result

Section 5.1 and 5.2 lists whether the requirements were reached or not. If the requirements were not reached the reason will be explained in the following chapter.

5.1 Requirements fulfilled

- Search for the device
- Display heart rate and oxygen saturation
- Display a graph
- Record a log file
- Send the log file to a remote device
- Log file name suffix with date and time
- Filtering of devices in device search

5.2 Requirements not fulfilled

- Implement MDP
- Send data to server application by using GPRS or UMTS
- Server application
5.3 Demonstrator

Figure 5.1 illustrates the completed demonstrator.
Discussion

6.1 Conclusion

The purpose of this thesis is defined in the beginning of this report as “The purpose is to provide Cybercom Sweden West with a demonstrator which illustrates the usefulness of the coming Bluetooth MDP to Medica trade fair (14-11-2007).” A demonstrator was ready in time for Medica trade fair but it did not use the medical device profile. The difference in functionality between a MDP version and the SPP version of the demonstrator would have been small if any, and as a proof of concept the demonstrator was successful.

6.2 Reason for not fulfilling requirements

As it is written in the conclusion the difference for a user between the usage of SPP instead of MDP is minor if any. When the project were started the intention was to port MDP to Symbian using a OSAL but Symbian does not utilize the same CPU scheduling as the windows ported version of MDP. To rewrite the MDP was shown to be a too big challenge in this short time frame, and this is why the MDP implementation was not fulfilled.

The server application was a low priority requirement from the beginning and it was decided rather early to make use of the log file option instead, where the file can be transferred by SMS, MMS, IR or Bluetooth. This file is MS Excel compatible for graph plotting on PC.

6.3 Suggestions to alterations of the application

If the demonstrator should be developed to an end user application there are some alterations that could be considered.

Port to S60

In order to reach a larger number of users, the application could be ported to Nokia series 60. The manager class and the active object classes could be
transferred unchanged but the GUI framework would need to be changed to Nokia S60. Both these two platforms are basically based on the same Symbian classes but there are UIQ specific APIs (e.g. Bluetooth search dialog) that are used. Another significant difference is the touch screen UI which are used in UIQ and not S60.

**GUI improvements**

As the GUI is currently design it encapsulates too much functionality. It would be a better design to separate as much functionality as possible from the GUI layer. Figure 6.1 shows an alternative to this situation.

![Figure 6.1: Desirable application architecture](image)

Another improvement on the GUI would be to use dialogs or more then one view, with each its own control. The GUI framework is quite complicated and time consuming to get started with, this is why the demonstrator application is not designed this way. Our GUI development started of in the wrong direction and it became too late to change when this was noticed. If this is changed the screen resolution problem we have would disappear.

**Add application functionality**

The current application can only record 10 heart rate values and save them into a log file; this is for demonstration purpose only, and it is the same situation when it comes to the graph which only plots 10 sec.

This would have been changed in an end user product. The application could also be extended to allow the user to select the recording duration and the oxygen saturation values within the same file as the heart rate values.

**Send data by GPRS or UMTS**

Further the application could be used for the medical purposes as it original was intended to demonstrate, namely some sort of server application which the stored log-file could have been uploaded to. This is shown in the original use case i.e. sending the log file to a server using GPRS. This would except for the application also need a server application.
6.4 Difficulties during project

6.4.1 Capabilities

A rather small but yet annoying problem during application development, was the problem to get a “heart-beat” API function to light up the background on the display. Background to the problem is; if this is not done and the smartphone has default setup, the display will fade out after 10 sec. and that is an undesired effect when demonstrating on the trade fair. The APIs used to modify these setups are considered to be harmful and therefore protected by manufacture capabilities. The solution was to manually setup this in the phone, which is accepted due to non-end user product.
Appendix A

Bluetooth profiles

Advanced Audio Distribution Profile (A2DP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP &amp; GAVDP[39].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>This profile defines how audio is distributed between two devices. The profile differs from the Hands-Free Profile (HFP) by using an ACL link instead of SOC and by providing mono or stereo audio of higher quality. A typical use case of this profile could be a media player that transfers music to wireless headphones[39].</td>
</tr>
</tbody>
</table>

Audio/Video Remote Control Profile (AVRCP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP[40].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>This profile describes how to remotely control Bluetooth enabled audio and video equipment by sending commands to alter for example volume or channel. The profile uses AVCTP to transport control commands[40].</td>
</tr>
</tbody>
</table>

Basic Imaging Profile (BIP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP,SPP &amp; GOEP[34].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>BIP describes how images should be transferred between devices. The profile handles resizing and conversion between image formats[34].</td>
</tr>
</tbody>
</table>

Basic Printing Profile (BPP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP,SPP &amp; GOEP[41].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>This profile describes how to send documents to a printer. The printer has a list of supported document formats and can be discovered by a Bluetooth device that is using BPP if the Printer has got the required document format[41].</td>
</tr>
</tbody>
</table>

## Common ISDN Access Profile (CIP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP[43].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Defines how to access ISDN over Bluetooth. Often used in ISDN access points. [43].</td>
</tr>
</tbody>
</table>

## Cordless Telephony Profile (CTP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP[45].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Bluetooth SIG has defined a use case called: 3-in-1 phone, where fixed network telephony services can be accessed by a Bluetooth enabled device through a base station. CTP was designed to provide a standard for implementations of the 3-in-1 phone use case. An example of the use case could be a Bluetooth enabled mobile phone that can make calls via the land line when close to a Bluetooth enabled base station[45].</td>
</tr>
</tbody>
</table>

## Fax Profile (FAX)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP &amp; SPP[37].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Defines how to implement FAX functionality with Bluetooth, typical situation is when mobile phone/modem is used as gateways for a PC[37].</td>
</tr>
</tbody>
</table>

## File Transfer Profile (FTP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP &amp; GOEP[32].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Defines how one Bluetooth device can browse, transfer and manipulate a file system on another Bluetooth device[32].</td>
</tr>
</tbody>
</table>

## Generic Access Profile (GAP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP[46].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>All other profiles depend on GAP. GAP describes how to discover and connect to other devices[46].</td>
</tr>
</tbody>
</table>

## Generic Audio/Video Distribution Profile (GAVDP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP[31].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Describes how to set up two devices for audio and video streaming, and how to control that stream. This profile forms the base on which A2DP and VDP depends[31].</td>
</tr>
</tbody>
</table>

## Generic Object Exchange Profile (GOEP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP &amp; SPP[30].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>Defines object transfers between devices. All other profiles that are using OBEX depend on GOEP[30].</td>
</tr>
<tr>
<td>Profile Name</td>
<td>Dependencies</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Hands-Free Profile (HFP)</td>
<td>GAP &amp; SPP[47]</td>
</tr>
<tr>
<td>Hard Copy Cable Replacement Profile (HCRP)</td>
<td>GAP[27]</td>
</tr>
<tr>
<td>Headset Profile (HSP)</td>
<td>GAP &amp; SPP[48]</td>
</tr>
<tr>
<td>Human Interface Device Profile (HID)</td>
<td>GAP[26]</td>
</tr>
<tr>
<td>Intercom Profile (ICP)</td>
<td>GAP[25]</td>
</tr>
<tr>
<td>Object Push Profile (OPP)</td>
<td>GAP[28]</td>
</tr>
<tr>
<td>Personal Area Networking Profile (PAN)</td>
<td>GAP[28]</td>
</tr>
</tbody>
</table>
Serial Port Profile (SPP)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP, RFCOMM[24].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>The SPP defines how to use virtual serial ports provided by the RFCOMM protocol [24].</td>
</tr>
</tbody>
</table>

Synchronization Profile (SYNC)

<table>
<thead>
<tr>
<th>Dependencies</th>
<th>GAP &amp; GOEP[23].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Information</td>
<td>SYNC defines how synchronization of personal information manager (PIM) items like calendar and address book are done.[23].</td>
</tr>
</tbody>
</table>
## Appendix B

### Bluetooth protocols

**Audio / Video Control Transport Protocol (AVCTP)**

| Protocol Information | AVRCP is used by AVRCP to send commands to control audio and video equipment.\[?\] |

**Audio / Video Distribution Transport Protocol (AVDTP)**

| Protocol Information | A transport protocol which defines how audio and video is streamed using Bluetooth.\[52\] |

**Bluetooth Network Encapsulating protocol (BNEP)**

| Protocol Information | BNEP is used to encapsulate other networking protocols such as IPv4 and IPv6 to be transmitted by a Bluetooth link. The profile is used by PAN.\[42\] |

**Object Exchange (OBEX)**

| Protocol Information | The OBEX profile defines a protocol which defines data objects and a communication protocol to transfer these data objects. These three profiles are using OBEX; the synchronization-, the file transfer- and the object push profile[36]. |

**Radio Frequency COMMunication (RFCOMM)**

| Protocol Information | RFCOMM is a transport protocol that emulates RS-232 serial ports.\[53\] |

---

**Service Discovery Protocol (SDP)**

| Protocol Information | SDP defines how a Bluetooth device shall find services on remote devices. [24] |

**Telephony Control Specification (TCS-Binary or TCP)**

| Protocol Information | Defines the signaling required to control voice and data calls. Protocol also defines how to handle groups of Bluetooth devices. [54] |
Appendix C

HEALTH LOG DEMONSTRATOR

The application displays pulse, HR, and saves a 10 second log which can be transferred by Bluetooth, IR, SMS or USB. The log can be displayed as a graph in e.g. Microsoft Excel.

To ensure that the phone backlight will not be disabled during demonstration, select "Main menu" -> "Tools" -> "Control panel" -> "Device" -> "Backlight" and deleted the "Power save enabled" checkbox.
(Sony Ericsson P1i path to Backlight is: Select "Main menu" -> "Control panel" -> "Device" -> "Backlight")

Oximeter:

1) Connect the cable, make sure that the device is in detectable mode (indicator light is blinking orange).

Phone:

2) Select "Main menu" -> "Tools" -> "Health Log" the application starts and "NOT CONNECTED" is displayed.

3) Select "more" -> "Connect" and wait until the preferred oximeter is found. Select the preferred oximeter. If the phone is not previously paired with the oximeter, the phone will ask for authentication. The pass code should be the last 6 digits in the device name (e.g. device name: Phone: Medical Inc...563000: Pass code: 563000). If the connection is successful "NOT CONNECTED" will change to display pulse and HRap02 and the oximeter indicator light will change to a green blinking light. If no signal has happened within 10 seconds the "more" -> "Connect" again (make sure that the right device is selected) wait another 10 seconds. If this does not solve the problem then reboot the oximeter by unplugging the battery and retry step 3 again.

4) To record data from the oximeter; select "more" -> "Log 10 seconds". The application will display a graph presenting pulse during 10 seconds. The application will then present four views to select the recorded data. Select the preferred method and enter the file. When the transition is complete the application will automatically go back to displaying pulse and HRap02.

The recorded data is saved locally with the same format (log/d/LogP1i_2009-01-01.dat). The path to the folder containing the logs files is: "Main menu" -> "Tools" -> "File manager" -> "media". The folder should be checked regularly to avoid that the phone runs out of memory.

The log file will be overwritten if more than one recording is done within the same minute.

This application has been tested with:
Sony Ericsson P1i
Sony Ericsson K800i
Sony Ericsson W900i
Bibliography

[1] Developing Software for Symbian OS, chapter 1 & 2 - Smartphones and Symbian OS, Symbian OS Quick Start
   *By Steve Babin, 2006*

   *Symbian.com, 14-01-2008*

   *Symbian.com, 14-01-2008*

   *Descriptors.blogspot.com, 23-11-2007*

   *Symbiansigned.com, 15-01-2008*

   *Symbian.com, 10-12-2007*

   *Symbian.com, 10-12-2007*

[8] Developing Software for Symbian OS, chapter 8 - Asynchronous Functions and Active Objects
   *By Steve Babin, 2006*

[9] UIQ 3 SDK » Symbian OS v9.1 » Symbian OS guide » Base » Using User Library (E32) » Buffers and Strings » Using Descriptors » Descriptor concepts » Descriptor basics
   *UIQ help documentation, Included in SDK*

   *Uiq.com, 11-01-2008*

UIQ help documentation, Included in SDK

[12] Developing Software for Symbian OS, chapter 4 - Symbian OS Programming Basics

By Steve Babin, 2006

[13] Symbian developer library, available in SDK help or on symbian.com


Symbian.com, 14-01-2008


Symbianone.com, 11-01-2008


By Steve Babin, 2006

[16] The complete guide to Symbian signed


Symbian.com, 11-01-2008

[17] Developing Software for Symbian OS, p. 89

By Steve Babin, 2006

[18] Developing Software for Symbian OS, Chapter 6 - Strings, Buffers and Data Collections

By Steve Babin, 2006

[19] Bluetooth:Connect without cables

Jennifer Bray, Charles F Sturman, 2002


Bluetooth.com

[21] Bluetooth operation and use

Robert Morrow, 2002


Bluetooth.com


Bluetooth.com, 15-01-2008


Bluetooth.com, 15-01-2008
Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

[46] Bluetooth profiles the definitive guide
Dean A. Gratton, 2003

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 15-01-2008

Bluetooth.com, 16-01-2008
   Bluetooth.com, 15-01-2008

   Bluetooth.com, 15-01-2008

   Bluetooth.com, 15-01-2008

   Bluetooth.com, 15-01-2008

   Bluetooth.com, 17-01-2008

   Bluetooth.org, 15-01-2008

[57] Developing Software for Symbian OS, p 332
   By Steve Babin, 2006

   Nonin.com, 14-01-2008

   Uiq.com, 14-01-2008

[60] http://www.medica.de/cgi-bin/md_medica/custom/pub/content.cgi?lang=2&oid=20846&ticket=g_u_e_s_t&redirect_kati_url=%2Fkaticgi%2Fkati%2Fvis%2Fcustom%2Fext%2Fshow_exhibitors.cgi%3Fcp%3D0%26backto%3Dindex%26portal_search%3Dcybercom%26ticket%3Dk9748928997635
   Medica.de, 14-01-2008

   Cybercomgroup.com, 14-01-2008

   Bluetooth.com, 14-01-2008
[63] Symbian developer library, available in SDK help or on symbian.com

Symbian OS v9.1 » Symbian OS guide » Comms infrastructure
» Using Sockets Server (ESOCK) » Sockets Client Overview

Symbian.com, 14-01-2008


Wikipedia.org, 14-01-2008