

Position of a Body Area Network

A design for integration of a Body Area Network with existing location provision technologies.

Author: H. Schaap
Date: January 2005
Faculty: Electrical Engineering, Mathematics and Computer Science
Group: Architecture and Services of Network Applications (ASNA)
Tutor: dr.ir. A.T. van Halteren



Index

1	ABSTRACT.....	2
2	INTRODUCTION.....	3
2.1	MOTIVATION.....	3
2.2	CONTEXT OF BODY AREA NETWORK.....	3
2.3	RESEARCH QUESTION.....	5
2.4	APPROACH.....	5
2.5	REPORT STRUCTURE.....	6
3	BAN TECHNOLOGIES.....	7
3.1	OVERVIEW.....	7
3.2	BLUETOOTH PROTOCOL.....	8
3.2.1	<i>General</i>	8
3.2.2	<i>Core specification</i>	9
3.2.3	<i>Profiles</i>	11
3.2.4	<i>In practice</i>	13
3.3	GPRS PROTOCOL.....	14
3.3.1	<i>General</i>	14
3.3.2	<i>Protocol and building blocks</i>	14
3.3.3	<i>Characteristics</i>	16
3.3.4	<i>In practice</i>	17
3.4	EDGE AND UMTS.....	17
3.5	HTTP.....	18
4	POSITION PROVISION TECHNOLOGIES.....	20
4.1	GPS.....	20
4.2	DGPS.....	20
4.3	ALTERNATIVES.....	21
4.4	NMEA.....	21
4.5	GPSD.....	22
4.6	JSR-179.....	23
5	INTEGRATION BAN AND POSITION PROVISION.....	24
5.1	SERVICE DESCRIPTION BAN LPS.....	24
5.1.1	<i>Decomposition of the BAN LPS</i>	25
5.1.2	<i>Decomposition of the MBU</i>	26
5.1.3	<i>Decomposition of the Central server</i>	28
6	IMPLEMENTATION BAN LPS.....	30
6.1	USED HARD AND SOFTWARE.....	30
6.2	PATIENT SIDE IMPLEMENTATION.....	30
6.2.1	<i>Location provider</i>	30
6.2.2	<i>MBU communication service</i>	30
6.2.3	<i>Interaction communication service</i>	30
6.2.4	<i>MBU</i>	30
6.3	HEALTHCARE CENTER SIDE.....	33
6.3.1	<i>Central server</i>	33
7	EVALUATION.....	36
7.1	RESEARCH QUESTIONS.....	36
7.2	EVALUATION OF DESIGN.....	36
7.3	EXPERIENCES DURING IMPLEMENTATION.....	37
8	REFERENCES.....	39

1 Abstract

With the advancement of wireless technology, new applications become possible. The MobiHealth project is a collaboration of relevant parties to investigate new mobile applications in the healthcare sector. As part of this project the Body Area Network (BAN) has been developed.

The main purpose of the BAN is to make it possible for patients who need permanent monitoring to be fully mobile. The BAN is worn by a patient and basically consists of a set of lightweight devices that monitor and wirelessly transmit certain bio signals (vital signs) to a BackEnd System at a Healthcare center. A monitoring healthcare specialist retrieves the patient data over a reliable wired connection.

To increase the freedom of movement of the patient there is a need to locate a patient. This thesis considers the possibilities to integrate the current BAN prototype with position provision. Also it gives a possible implementation solution. Accurate research questions and an approach to answer these questions are defined.

In order to integrate the BAN with position provision this thesis examines the concept of the BAN and the relevant BAN technologies. The focus is on the wireless technologies Bluetooth and General Packet Radio Switching (GPRS), because of their important role of communication means.

Secondly this thesis gives an overview of different technologies to provide a location. Here the focus is on Global Positioning System (GPS), but alternatives and supporting technologies, to integrate position provision in a system, are also examined.

Based on the existing BAN prototype, the possible position provision technologies and performance criteria it is possible to formulate a service that integrates the current BAN prototype with position provision. This BAN Location Provision Service (BAN LPS) makes it possible to locate patients while they are not in a healthcare center. To make this possible the patient wears some equipment that is part of the BAN LPS and can be integrated with the BAN to make it possible for a Healthcare specialist to see the location of this patient on a map.

This thesis describes the service of the BAN LPS. The BAN LPS has a patient side and a healthcare center side that interact using GPRS technology. Both sides are decomposed in functional building blocks that can be implemented separately.

To retrieve an implementation this thesis gives an overview of important implementation choices.

Finally this thesis gives an evaluation of the design and implementation of the prototype. This evaluation is based on design criteria and

2 Introduction

2.1 Motivation

With the advancement of wireless technology, new applications become possible in the healthcare sector. Removing the restrictions imposed by wires and cables enables patients to benefit from increased mobility. For example, ambulant patients would normally have to remain at a healthcare centre if they require regular health monitoring, even though they are not confined to a hospital bed. With a wireless monitoring device it becomes possible for them to return to the comfort of their own home. Such an application not only improves the patient's quality of life, but also benefits healthcare insurers when it comes to disease and care related costs. A healthcare centre saves costs, e.g. food and housing, by enabling patients to spend more of their time away from the centre.

These and other mobile health applications are the topic of investigation for the MobiHealth project. The MobiHealth project is a collaboration of fourteen partners from five European countries, all relevant parties are involved: hospitals, universities, medical service providers, mobile network operators, mobile application service providers and mobile infrastructure and hardware suppliers.

The MobiHealth project considers healthcare services for patients without restricting their freedom to move around. As a part of the project, the so-called MobiHealth *Body Area Network* (BAN) has been developed to enable remote monitoring of ambulant patients. Although it is possible it to remotely monitor a patient, there is still the need to locate a patient.

The University of Twente is one of the participating partners in the MobiHealth Project. The study Telematics at the faculty computer science considers among other things distributed computer systems like the BAN. This project will extend the BAN with position provision.

More about the MobiHealth project can be read on the website [<http://www.mobihealth.org/>]

2.2 Context of Body Area Network

The main purpose of the BAN is to make it possible for patients who need permanent monitoring to be fully mobile. To fulfill that objective a personal lightweight monitor system is created that is completely customized to the patients needs. The BAN is worn by a patient and basically consists of a set of lightweight devices that monitor and wirelessly transmit certain bio signals (vital signs) to a BackEnd System. Healthcare centers can then retrieve this data over a reliable wired connection.

Concrete, the BAN consists of one or more sensors, depending on the patients' need, which measures specific data of the patient, for example blood pressure. This data is sent to a Mobile Base Unit (MBU), which can for example be a Personal Digital Assistant (PDA). The communication between sensor and MBU can be over a wire or via Bluetooth. Thereby, the MBU is in contact with a Healthcare centre. Vital information is periodically or only when it has reached a critical value sent to a central server at the Healthcare centre. For this, the wireless data transmission technologies Global Packet Radio Service (GPRS) or Universal Mobile Telecommunications System (UMTS) are used. A monitoring health specialist can login on the server and ask for the patient information that will be represented on for example a simple PC.

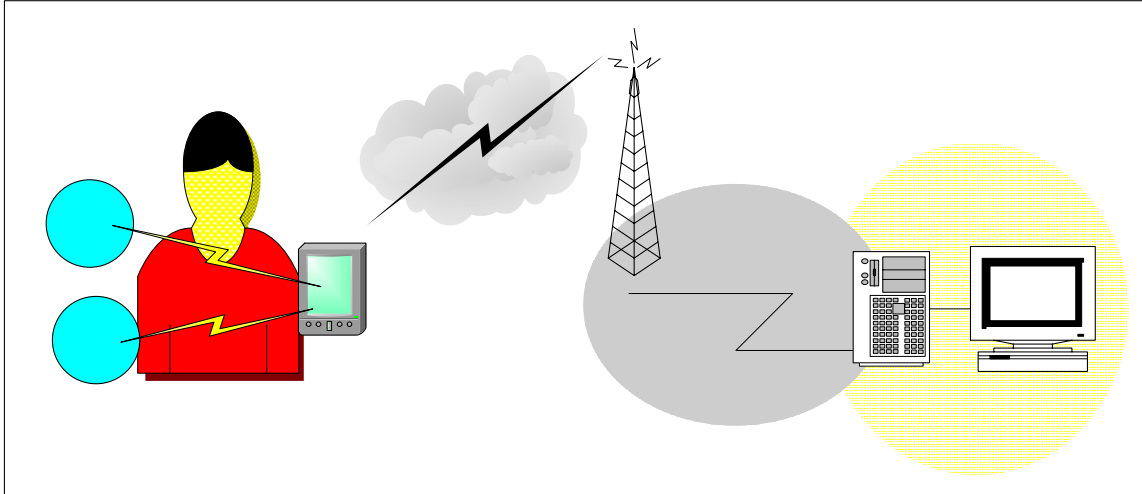


Figure 1.1: Body Area Network, consisting of sensors and PDA, connected to healthcare centre.

GPRS / UMTS
Wireless connect

At this moment a prototype of the BAN has been developed which satisfies the goals of the MobiHealth project. However, research on the project goes on. One of the opportunities is to extend the BAN with position provision. The information that is send by the MBU should not only contain bio information, but also data about *where* the patient is located.

Consider the following case:

Imagine a pregnant woman who must constantly be monitored because she has an increased risk for complications. At this moment, that woman is resting home. When there are indications of potential harm for her or her child, for example pre-mature uterus activity, a doctor can intervene.

Sensor

PDA

With the Body Area Network technology, the woman simply must wear some sensors which measure her uterus activity. When that rises too much, an emergency signal will be given and not only the uterus activity information, but also the position of the patient will be sent to the hospital and reflected on a map. With that information the monitoring doctor can decide to take contact or send help. In the last case, he exactly knows where to find the patient.

Body Area Network

The position information gives the women the freedom of not being in a health care centre. With sensors built in a belt for example she has more freedom of movement. When the sensors reach a defined critical level, here position will be sent to the monitoring specialist. With the received bio- and position information the monitoring specialist can act in a very adequate manner.

With the current technologies it is possible to locate the exact position of a device or a person. The most common and accurate position provision technology is Global Positioning System (GPS). However, to integrate this technology with the BAN certain major problems must be overcome.

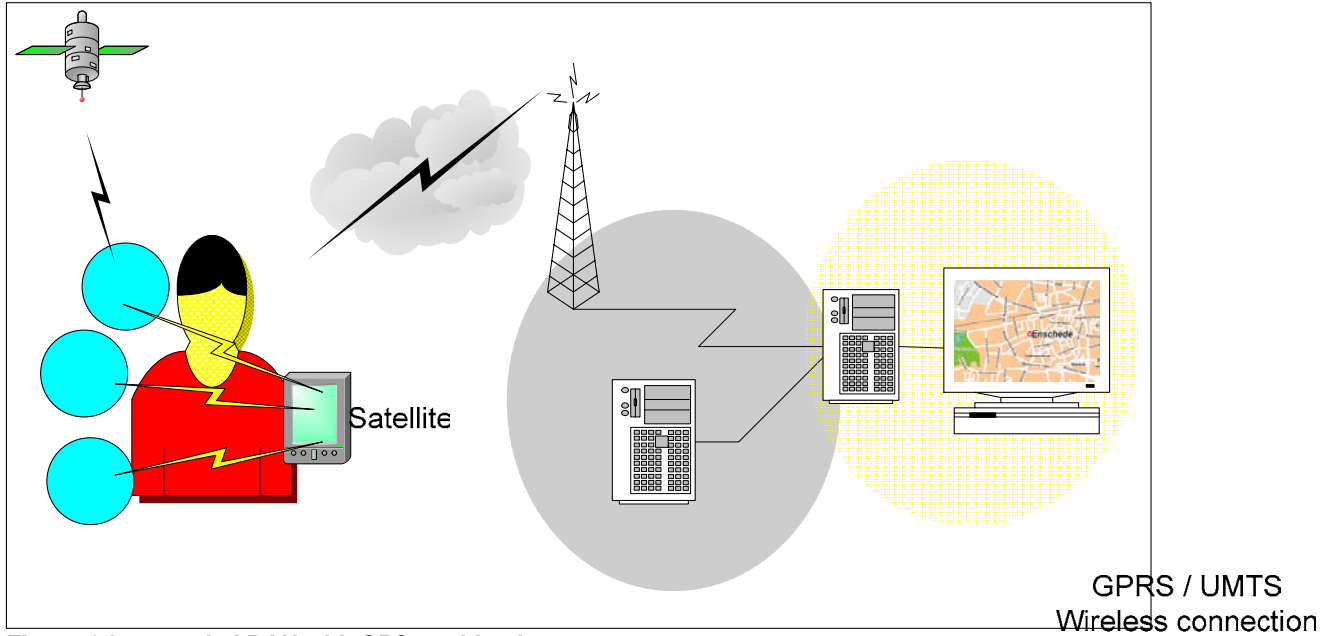


Figure 1.2: extended BAN with GPS position locator.

In this project I will consider the possibilities to integrate the current BAN prototype with position provision. I must couple a provision sensor to the MBU, which will be a GPS transceiver. Data communication between the sensor and the MBU will be via Bluetooth. For data exchange between the MBU and the central server at the healthcare centre GPRS or UMTS is used. For this, it is very important that data is sent at an efficient way, so the position and information of the patient is only sent when necessarily. This information automatically must be represented on a geographical map that is provided by a third external party.

2.3 Research Question

The objectives of this project are as follows:

1. Examine the technologies that are used in the Body Area Network. Hereby only the relevant technologies will be examined in detail, for more information I will refer to secondary literature.
2. Examine in what way the position information of a Body Area Network can be provided. This consists of detailed relevant technology description, but also of problems and solutions that will overcome that problem.
3. Modulate a possible solution for integrating the position provision in the existing Body Area Network. Thereby, the position of a BAN must be determined with a sensor integrated with the BAN and be shown on a remotely located map. A technical description of how the solution works is given.
4. Implement a prototype of a Body Area Network with position provision according to the designed model.

2.4 Approach

To fulfill the research question given in section 1.4 a systematic approach is necessarily. First I must know more about the context of the BAN, I must understand what the used techniques are and how they fit in the context of the BAN. To understand this better I split up functionality and abstract from irrelevant details.

When the functionality of the current prototype of the BAN is clear, I will take a look at how the BAN can be extended with position provision techniques. Alternatives will be looked for I will describe exactly how position provision techniques can be integrated within the BAN.

Before a prototype of a BAN with position provision can be developed I must make some decisions considering the design. A rational consideration is made between possible solutions. Based on the design concept a prototype will be developed. Finally critical analyzes of the research, the used techniques, development choice and prototype will be given.

2.5 Report structure

In this report the specified objectives are worked out in a systematic way:

Chapter 1, the *Introduction*, gives an overall picture of what this report is about.

Chapter 2, *BAN technologies*, gives an overview of BAN technologies and their relations. All technologies used in this project are described. The focus is on characteristics that are critical for the use in the BAN. For more common information a reference will be given. The technologies are:

- 2.2 Bluetooth
- 2.3 GPRS
- 2.4 EDGE and UMTS
- 2.5 HTTP

Chapter 3, *Position provision technologies*, gives a detailed description about the working of position provision technologies. For the BAN relevant specific characteristics are highlighted. The most important position technology is GPS (3.1), but also some alternatives / supplements are given (3.2).

Chapter 4, *Integration BAN and Position provision*, gives a solution for integrating a position sensor in the Body area network. A few design alternatives will be evaluated thereby every detail is highlighted.

Chapter 5, *development prototype*, explains how the solution of chapter 4 is implemented.

Finally, chapter 6, gives an overall *evaluation*. Strengths, weaknesses, opportunities and threads will be given of the developed solution.

At last there is a *reference* (7) list of used literature.

3 BAN technologies

3.1 Overview

The Body Area Network is developed to enable remote monitoring of patients. It is a distributed system with on one end the patient (BAN) and on the other end the monitoring health specialist (server). The BAN has the role of service provider and is the central server of service user. Within the BAN information about the patient is collected via sensors. The MBU sends this information to a central server, where a monitoring health specialist collects the information (see figure 1.1). Obviously, the communication can be divided in internal BAN communication and external communication between the BAN and the health care centre.

Figure 2.1 gives an architecture. Gray squares are functional building blocks, they represent physical devices. The colored layers are protocols; every layer performs its own service, the black ellipses are communication points between the layers. It seems for an end user, for example the monitoring specialist that it receives its information directly from the MBU, this is represented with the horizontal arrows, but in reality he receives the information from the underlying service layer that at his turn receives its information from the communication layer, the vertical arrows.

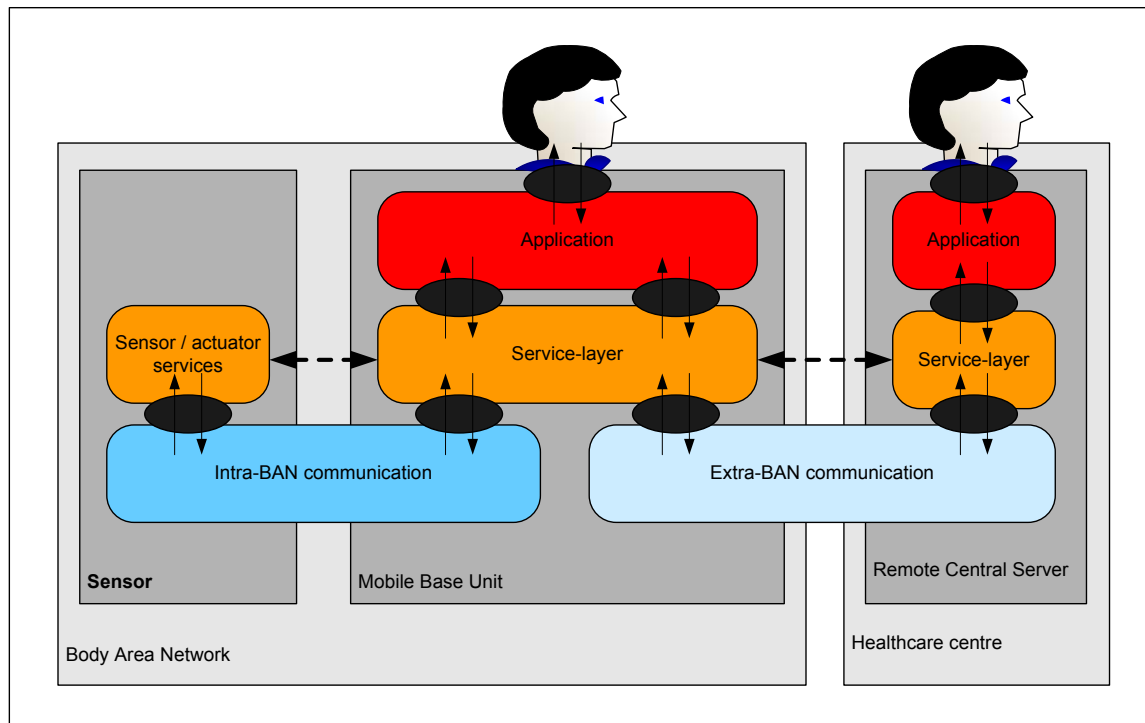


Figure 2.1: Architecture and functional building blocks of the total system.

The communication between entities within a BAN is called intra-BAN communication, concrete this is the communication between sensors and the MBU (PDA). Because it isn't very comfortable for a patient to be surrounded with wires, wireless communication is preferred for this. The short range wireless Bluetooth technology is the most useful solution for this problem. In figure 2.1 this is represented as the dark blue layer.

The external communication between the MBU and the central server is called extra-BAN communication, the light blue layer in figure 2.1. For this purpose GPRS or UMTS technologies are used.

In this scheme the application represents what the end user sees, red in figure 2.1. There are two types of applications, one within the BAN, on a MBU, the other on a remote central server at the healthcare centre. The application on the BAN can vary from a simple representation of collected sensor data to complex functionality for complicated data analyses. The second type of application also represents collected sensor data but thereby adds functionality for providing offline or online data processing, according to the MBU status.

There is a layer between the communication layer and the application; that is the service- layer represented in orange. The service layer masks the applications from the specific characteristics of the underlying communication network. It takes care of aspects like identification of a BAN, reliability and security of a connection, the limited capacities of a MBU and the scalability of the whole system. These aspects are out of the scope of this project more information can be found in the references [FIDJ03].

Now the overall structure is clear we'll take a closer look at the details. In the next chapters all used technologies and communication points will be highlighted.

3.2 Bluetooth protocol

In this chapter, we explain the working of Bluetooth and how this protocol is used in the BAN. In figure 2.2 the read area shows where Bluetooth is used in the MobiHealth project.

It is the responsibility of the Bluetooth protocol to set up a connection between a MBU and a sensor, to send data over this connection and to maintain or close this connection. Bluetooth implements the Intra-BAN.

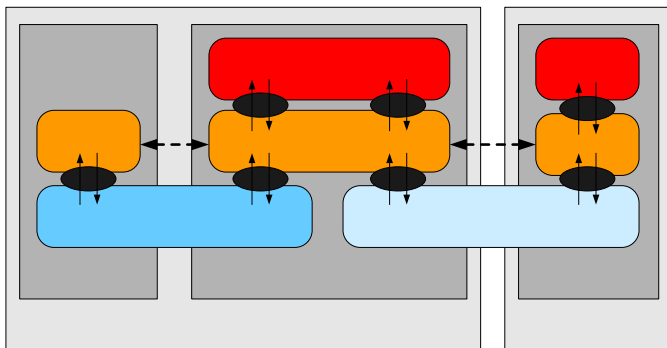
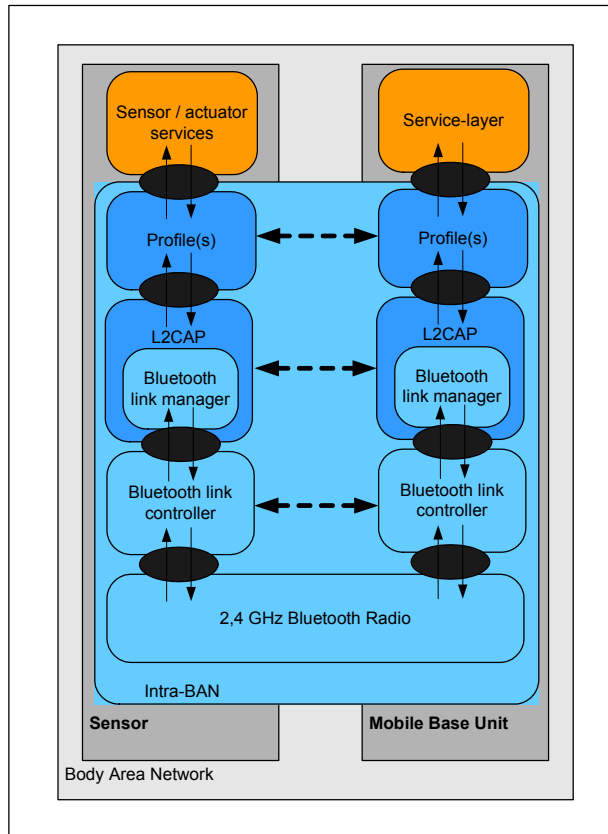


Figure 2.2: Role of Bluetooth in overall scheme

3.2.1 General

The Bluetooth wireless technology was created to replace the cables that are used to connect small mobile devices and their peripherals. It was developed by the Bluetooth Special Interest Group, or SIG. The SIG consists of the following companies: 3Com, Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba. It is a short-range communication standard, enabling wireless data communication between small, low powered devices at ranges of about 10 meter.

The Bluetooth Special Interest Group, or SIG, created a 2-part specification that defines the Bluetooth standard. The first part is the Core Specification, which defines the basics of the Bluetooth standard. The second part is the Profiles book. This part defines a selection of messages and procedures, gives an unambiguous description of the air interface for the specified service(s) and use case(s) and gives implementation suggestions for developers. The profiles are meant to guaranty interoperability between different product, brands and manufacturers.



In figure 2.3 the Bluetooth Intra-BAN is more specified. The lighter blue layers are specified in the core specification, the darker blue layers in the profile specification. It can be seen that there is an overlap of the specifications, that is because some profiles need specific procedures or messages in the link manager layer. The used profile depends on the purposes of the application.

In the next section the Bluetooth protocol will be explained according to figure 2.3.

Figure 2.3: Architecture of the Bluetooth protocol

3.2.2 Core specification

Radio layer:

The lowest layer, the Bluetooth Radio Layer, operates at the frequency band at 2,4 GHz. This layer consists of a transmitter that sends information to a receiver by using Gaussian Frequency Shift Keying. Hereby, a carrier wave is shifted over the frequency domain to distinguish a '1' bit from a '0' bit. In this layer the exact criteria for the transmitter and receiver are determined; this includes aspects like modulation characteristic, spurious emission, power criteria and other performance issues. More details about this can be found in the Bluetooth Core Specification v1.1 part A Radio Specification (See references).

Link Control layer:

To exchange information between two or more Bluetooth devices a channel is applied between these devices. Over this channel a connection or 'link' is made. This link can be between two or more Bluetooth devices. The Bluetooth link controller carries out low-level link protocols, like channel control, link control and packet control. This is described in the baseband specification.

A channel is represented by a pseudo random hopping sequence hopping through 79 or 23 Radio Frequency (RF) channels, the RF hop frequency to be used is derived from the Bluetooth internal clock value. On a channel information is exchanged through packets that are sent on fixed moments according to the internal clock. The frequency hop transceiver (transmitter + receiver) is used to overcome interference and fading; each packet is transmitted on a different hop frequency.

The Bluetooth system allows a point to point and a point to multi point connection. When two or more units share the same channel, this is called a piconet. One of the units acts as a master, while the other(s) act(s) as a slave. Between master and slaves two types of connections can be established. A Synchronous Connection-Oriented (SCO) link is a point to point connection between the master and the single slave, a Asynchronous Connection-Less (ACL) link is a point to multipoint link between the master and all the slaves participating in the piconet.

The data on the piconet is conveyed in packets. There are six types of packets that all have a specific function, that is: *device identification information*, *status acknowledgement information* (NULL packet), *polling information* (POLL packet), *special control information* (FHS packet) and the above mentioned SCO and ACL information. These packets are specified in a fixed format and consist of three entities: the *access code*, the *header* and the *payload*. The access code has a fixed length of 72 bits and identifies all packets that are exchanged on the channel in the piconet. The header has a fixed length of 54 bits and consists of link control information, i.e. *piconet member address information*, the *type of packet*, *flow information*, *acknowledge information*, *sequence information* and *header error check*. The Payload can range from zero to a maximum of 2745 bits. The payload depends on the type of the packet and can be empty or consist of asynchronous data or synchronous voice information. Also the bandwidth depends on the type of packet and can be up to 723,2 kb/s.

Link manager layer:

The purpose of the link manager layer is to set up and control a Bluetooth link and hide this information for higher levels. This layer filters out and interprets signals on the receiver side and doesn't propagate it to higher levels.

The link manager layer uses special messages, Link Manager Protocol (LMP) messages for link set-up, security and control. These messages are transferred in the payload and are distinguished by a reserved value in the packet header. The Link Manager Protocol essentially consists of a number (13) of PDU (protocol Data Units) which are sent from one device to another. The PDU's have different purposes like detecting other link managers, agreeing upon authentication, encryption, Quality of Services, clock synchronization etc.

3.2.3 Profiles

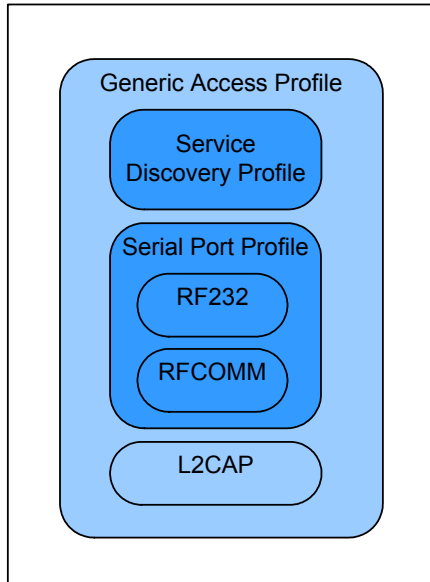


Figure 2.4: Profile Structure

Within this project the *Serial Port Profile* (SPP) as a replacement of a cable between a position sensor and a MBU is the most relevant Bluetooth profile. This profile is supported by two other profiles, the *Service Discovery Profile* (SDP) and the *Generic Access Profile* (GAP), what means that the SPP reuses parts of the SDP and GAP profiles.

Figure 2.4 gives a overview of the profile structure and the dependencies between the profiles.

A short overview of the working procedures is given in the next section.

Generic access profile (GAP)

The GAP profile describes how to use the lower layer levels radio, link control and link management and takes care of user interface aspects, access procedures that are used by other profiles and guarantees link establishment.

L2CAP

The Logical Link Control and Adaptation Layer Protocol (L2CAP) permits higher level protocols and applications to transmit and receive L2CAP data packets.

L2CAP is layered over the Baseband Protocol and resides in the data link layer. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions.

Service Discovery Protocol

The applications on both sides are typically legacy applications, able and wanting to communicate over a serial cable (which in this case is emulated). But legacy applications cannot know about Bluetooth procedures for setting up emulated serial cables, which is why they need help from some sort of Bluetooth-aware helper application on both sides.

The service discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.

Serial port profile

The Serial Port Profile defines the protocols and procedures for devices using Bluetooth for RS232 (or similar) serial cable emulation.

RS232

RS-232 was created for one purpose, to interface between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE) employing serial binary data interchange. So as stated the DTE is the terminal or computer and the DCE is the modem or other communications device.

With serial communications data is transferred from sender to receiver one bit at a time through a single line or circuit. The serial port takes 8, 16 or 32 parallel bits from your computer bus and converts it as an 8, 16 or 32 bit serial stream. The name serial communications comes from this fact; each bit of information is transferred in series from one location to another. [RSSS98]

The scenario covered by the Serial Port Profile deals with legacy applications using Bluetooth as a cable replacement, through a virtual serial port abstraction (which in itself is operating system-dependent).

RFCOMM

The Serial Port Profile defines how a RFCOMM connection should be established between two devices. RFCOMM emulates full 9-pin RS232 serial communication over a L2CAP channel. [KJHO04]

TS 07.10 is used by GSM cellular phones to multiplex several streams of data onto one physical serial cable. RFCOMM is the Bluetooth adaptation of GSM TS 07.10, only a subset of the TS 07.10 standard is used, and some adaptations of the protocol are specified in the Bluetooth RFCOMM specification. The name RFCOMM comes from a Radio Frequency (RF) oriented emulation of the serial ports on a PC.

The RFCOMM protocol supports up to 60 simultaneous connections between two BT devices. The number of connections that can be used simultaneously in a BT device is implementation-specific. For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them.

Configuration and roles

The following roles are defined for this profile:

Device A (DevA) – This is the device that takes initiative to form a connection to another device.

Device B (DevB) – This is the device that waits for another device to take initiative to connect.

Establish Link and Set up Virtual Serial Connection

This procedure refers to performing the steps necessary to establish a connection to an emulated serial port (or equivalent) in a remote device. The steps in this procedure are for device A:

1. Submit a query using SDP to find out the RFCOMM Server channel number of the desired application in the remote device.
2. Optionally, require authentication of the remote device to be performed. Also optionally, require encryption to be turned on.
3. Request a new L2CAP channel to the remote RFCOMM entity.
4. Initiate an RFCOMM session on the L2CAP channel.
5. Start a new data link connection on the RFCOMM session, using the aforementioned server channel number.

After step 5, the virtual serial cable connection is ready to be used for communication between applications on both sides.

Accept link and establish virtual serial connection

This procedure is for device B for accepting a virtual connection:

1. If requested by the remote device, take part in authentication procedure and, upon further request, turn on encryption.
2. Accept a new channel establishment indication from L2CAP.
3. Accept an RFCOMM session establishment on that channel.

4. Accept a new data link connection on the RFCOMM session. This may trigger a local request to authenticate the remote device and turn on encryption, if the user has required that for the emulated serial port being connected to (and authentication/encryption procedures have not already been carried out).

3.2.4 In practice

To use Bluetooth one need to purchase a Bluetooth device, there are no further costs for using Bluetooth.

The MBU and the position provision sensor, the GPS receiver, contain a Bluetooth transceiver. Those transceivers emulate the serial cable ports described as RS-232; this is done by RFCOMM protocol. Then signals are sent from one device tot the other over a L2CAP channel. The underlying Bluetooth layers hide complex functionality for the profiles and the application.

With Bluetooth eight devices can be connected over the same channel using a piconet, seven users and one slave. So it is possible to connect up to 7 sensors (blood, temperature, position, etc.) to one MBU.

The speed depends on the type of service application, voice, data, or a combination of both. The SCO packets that are defined for voice have a 64 kb/s speech transmission speed, the maximal speed of data transmission is 723,2 kb/s.

A standard Bluetooth device is able to communicate with other Bluetooth devices within a range of 10 to 20 meters. There are stronger transmitters and more sensitive receivers so higher ranges can be made, but this is not relevant for the BAN.

To make sure that no malicious computer program can attack the BAN or other people can see personal information, communication with external Bluetooth devices should be prevented. In the BAN, the link manager layer makes sure that only the Bluetooth devices within the BAN communicate with each other.

Bluetooth references: [COSP] [PRBO] [PRIN] [ETSI]

3.3 GPRS Protocol

3.3.1 General

GPRS is a European Telecommunication Standards Institute (ETSI) standard [ETSI]. Overall, it is a complex standard based on the widespread Global System for Mobile communications (GSM) standard for mobile telephony.

In this chapter, we explain the working of General Packet Radio Service (GPRS) and how this protocol is used in the BAN. In figure 2.5 the red area shows where GPRS is used in the MobiHealth project.

It is the responsibility of the GPRS protocol to set up a connection between a MBU and a central server, to send data over this connection and to maintain or close this connection. GPRS implements the Extra-BAN. The user data must be transparently transferred between the external packet data network and the GPRS mobile station.

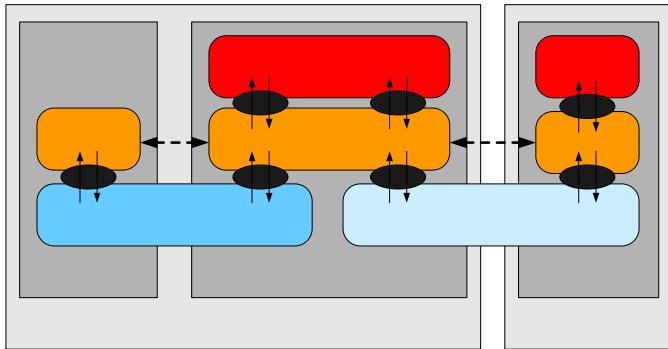


Figure 2.5: Role of GPRS in overall scheme.

3.3.2 Protocol and building blocks

It is out of the scope of this project to go through every detail of the GPRS protocol, therefore only a general description of the building blocks and their working is given.

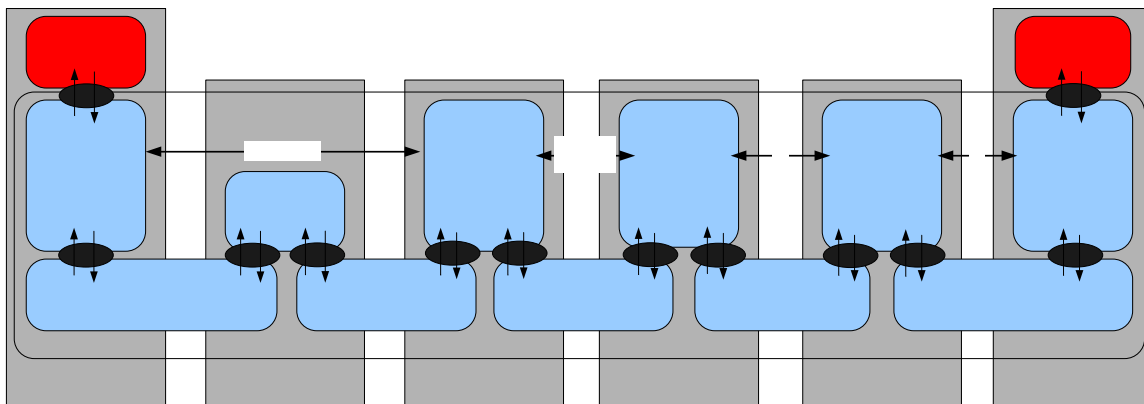


Figure 2.6: strongly simplified overview of GPRS protocol and relevant building blocks.

The GPRS network is an extension of the GSM network; it uses the GSM network elements but is enhanced with some specific features to enable packet switched data communication.

Intra-BAN communication:

Extra-BAN communication

The sending service layer element offers IP packets to the upper GPRS protocol layer. After transmission, an IP packet is offered to the receiving side service layer element. The transmission is split into two segments between the Gateway GPRS Support Node (GGSN) and Serving GPRS Support Node (SGSN), and the SGSN and GPRS mobile station. Methods known as encapsulation and tunneling are applied; the user data packet is equipped with specific protocol information which hides functionality for the underlying GPRS network. Also this enables an easy introduction of future inter-working protocols like UMTS: Operators interested in adapting their networks to UMTS in the future can reuse investments in the SGSNs, GGSNs, and the transmission network in between.

Mobile Base Unit (MBU)

At the mobile Base unit there must be a GPRS transceiver. This sends data to or receives data from the Base station. A *logical* connection between the GPRS mobile station and the SGSN is maintained. Over this logical connection packets are transparently transmitted between the GPRS mobile station and SGSN.

Base Station (BS)

The Base station consists of a controller that handles the radio resources and a transceiver that handles the communication to and from the MBU. The transceivers are the long antennas one can often see along the road. The GPRS specific coding schemes are implemented here; most operators will only introduce CS-1 and CS-2 code schemes because this can normally be implemented with a BS software update, the CS-3 and CS-4 code schemes require modifications to the transceiver and may not be implemented as rapidly (see table 2.1 below). The controller manages which part of the radio resources are allocated to circuit switched communication (GSM) and packet switched communication (GPRS), this realizes the 'data on demand concept'. The base station is connected to the Serving GPRS support node.

Serving GPRS Support Node (SGSN)

An SGSN is connected to the Base Station, to neighboring SGSN's and Gateway GPRS Support Nodes (GGSN). The SGSN keeps track of the individual MBU's location and performs authentication and authorization functions. It is the task of the SGSN to determine the most appropriate path available for transmitting data packets from their source to their destination therefore it contains a database with the location of the neighbor entities.

There is a logical link between the SGSN and the mobile unit (MBU). The SGSN manages this link; this includes establishment, maintenance and release. A logical link between the SGSN and the handheld can be maintained even if there are no physical resources in use.

Gateway GPRS Support Node (GGSN)

The GGSN provides the inter-working with external packet switched networks like the IP based Internet (IPv4 or IPv6). It is connected to SGSN's via an IP backbone on one side and to Internet packet routers on the other side. To the external network, the GGSN looks like an ordinary router. For communication with SGSN's tunneled IP packets are used.

Server at healthcare centre

Information between the GGSN and the server at the healthcare centre is exchanged over the Internet (IPv4 or IPv6). Routers are used to forward the data packets to the correct destination.

3.3.3 Characteristics

GPRS is based on packet switched data communication technology. This brings a few specific characteristics that are important for this project.

High Transfer Rate

Relative high transfer rates can be achieved using channel bundling and new coding schemes. With channel bundling, up to 8 timeslots per Time Division Multiple Access (TDMA) frame can be combined. Depending on the codec speed, see table below, this allows for transmission speeds of up to 171.2 kbps (8 timeslots at 21.4 kbps).

Codec	Data rate (kbps)
CS-1	9.05
CS-2	13.4
CS-3	15.6
CS-4	21.4

Table 2.1: Codec speed with the maximum data rate.

No limit on packet length

One of the specific features of a packet switched network is that the length of the information that must be sent is not relevant. The information is split up in smaller parts (IP data packets) and per packet sent over the network. So it overcomes the 160-character limit currently placed on short messages.

Pay for data use

The user is charged for the amount of send data, not for the duration of a connection. As said, a logical link is established between SGSN and the Mobile unit, however, when no data is sent over this connection, a user doesn't have to pay it. In the SGSN and GGSN statistical information, including billing information is collected. The subscriber only pays for the sent en received data, not for a duration of time.

Increased efficiency on the air interface

GPRS extends GSM and uses the same frequency band. To ensure efficient use, capacity allocation on demand is used: a cell's physical channels can be dynamically allocated for circuit switched and packet switched use. For example, in a cell with two transceivers, there are 14 physical channels available to transmit traffic. If the operator wants to ensure a call completion probability of 98% for circuit switched calls, on average less than 9 physical channels are used. The remaining resources are used as a spare for peak traffic situations. Those spare resources can be used for packet switched traffic. If circuit switched traffic increases, more physical channels can be allocated for circuit switched use "on demand" and some packet switched users may have to wait to continue downloading their data.

At last, asymmetric resource allocation is used: Uplink and downlink resources are allocated separately and may differ in size/capacity/rate.

3.3.4 In practice

In the BAN a HTTP application in the service layer generates IP packets and sends these to the underlying GPRS protocol. This protocol delivers IP packets at the destination.

There are a few things to take care of when using GPRS as underlying communication layer:

- The data to send must be reduced as much as possible; this reduces the cost and increases the speed.
- It might be possible that communication is not possible because of the dynamic resource allocation. For a medical solution this is not allowed.

Because coding schemes CS-3 and CS-4 are not fully implemented yet, the practical maximal transmission speed is between 72,4 (CS-1) and 107,2 (CS-2) kb per second and this is only achieved when all eight time slots are allocated to GPRS. Further, telecom operators give higher priority to circuit switched data traffic over packet switched traffic, so in practice the capacity of the GPRS network will be even less than that.

References: [TEKM] [MOBIL] [VOCA] [EEOU]

3.4 EDGE and UMTS

The capacity and transmission speed of the current GSM/GPRS networks can be increased by integrating new technologies. To enhance GSM networks EDGE and UMTS are developed. Because a mobile network that offers higher Quality of Services is quite relevant for the MobiHealth project a short overview is given of these technologies.

Enhanced Data rates for GSM Evolution (EDGE), also called Enhanced-GPRS (EGPRS) is a pre-third Generation (3G) technology that increases the data throughput of GSM/GPRS networks. By adopting a new (enhanced) radio modulation technology, used at the operators existing GSM radio frequency spectrum, data rates up to 473 kb/s are achieved. EDGE needs to be implemented at the Base station of the GPRS networks.

Universal Mobile Telecommunications System (UMTS) is a third generation (3G) mobile network. It is not the only 3G network architecture, but it is based on the existing GSM/GPRS(/EDGE) network structure and therefore the most practical choice for telecom operators to implement.

UMTS uses Wideband Code Division Multiple Access (WCDMA) radio access technology to access a new frequency band; 1885-2025 MHz and 2110-2200 MHz. This whole new access network is build and implemented at the Base stations of the current mobile network. Because the UMTS and GSM/GPRS networks use different frequency bands that (should) not interfere, the communication between the MBU and the SGSN is different for the two networks. Also this implicates that UMTS is not a replacement of the GSM/GPRS network but an enhancement to extend the total mobile end-services.

WCDMA in UMTS enables transmission speeds of up to 2 Mbit/s for voice, video, data and image transmission. This is about 10 times the traffic capacity of GSM while the costs are relative low.

UMTS and EDGE are being standardized in the Third Generation Partnership Project (3GPP). This represents a collaboration of 437 operators, vendors and standardization institutes worldwide and is part of the ITU/IMT-2000 standard.

For the MobiHealth project and the BAN, the development of EDGE and UMTS offer great opportunities. Because the data rate and network capacity is enormously increased with these technologies it becomes possible to couple more sophisticated sensors to the BAN that generate more data that can be monitored by a healthcare centre.

References: [UMWO] [PROT] [UMFO] [CELL] [GSAC] [WEBO]

3.5 HTTP

The HyperText Transfer Protocol (HTTP) is the de facto standard for transferring World Wide Web documents, although it is designed to be extensible to almost any document format. HTTP Version 1.1 is documented in RFC 2068.

In the MobiHealth project HTTP is used for higher level communication between the BAN and the server at the healthcare centre.

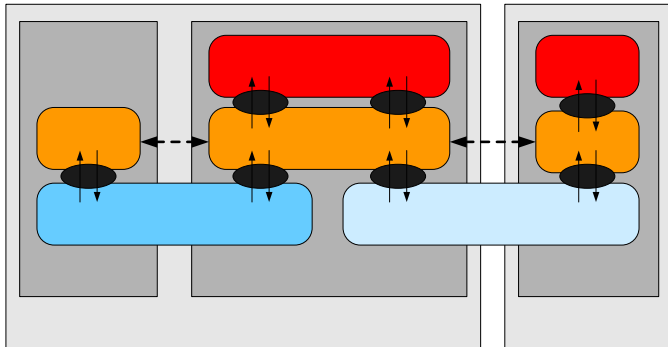


Figure 2.7: Role of HTTP in overall scheme.

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods and headers that indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI), as a location (URL) or name (URN), for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail as defined by the Multipurpose Internet Mail Extensions (MIME) that defines the format of messages.

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP (Simple Mail Transfer Protocol), NNTP (Network News Transfer Protocol), FTP (File Transfer Protocol), Gopher, and WAIS protocols (Wide Area Information Servers). In this way, HTTP allows Extensible BAN communication hypermedia access to resources available from diverse applications.

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta-information, and possible entity-body content.

Most HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In the simplest case, this may be accomplished via a single connection (v) between the user agent (UA) and the origin server (O).

request chain ----->
UA -----v----- O
<----- response chain

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

A more complicated situation occurs when one or more intermediaries are present in the request/response chain but is outside the scope of this thesis to go through all the possibilities.

Reference: [RFCE] [RFC2]

4 Position Provision Technologies

In this section we focus on techniques that make position provision of a BAN possible. At this moment the GPS technology is widespread used for all kind of provision position applications. In section 3.1, 3.2, and 3.3, we take a closer look at this is a proven technology. Because there are a few withdraws in using GPS for a medical solution we also take al look at addition technologies.

4.1 GPS

The Global Positioning System (GPS) is a navigation system consisting of 27 satellites (24 in operation and three extras in case one fails) orbiting the Earth twice a day and continuously transmitting information to a ground-based receiver, enabling the receiver to compute its position. Originally developed by the U.S. Department of Defense (DoD), the system is now widely used by the public for navigation and scientific purposes. By coupling a GPS receiver to a BAN, the latitude, longitude, and elevation of a patient can be traced any place on Earth. Schematically this can be seen in figure 3.1.

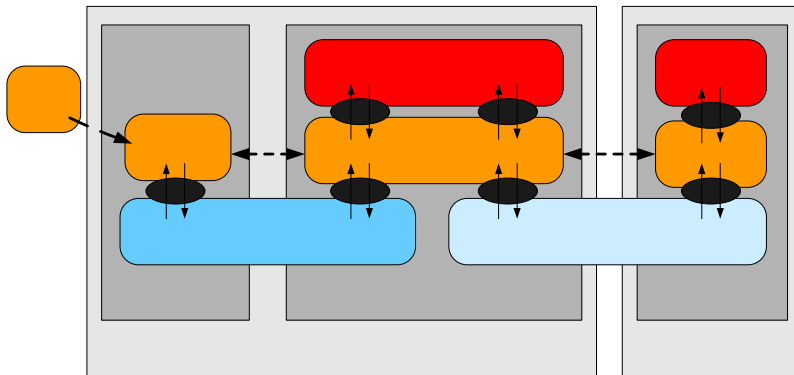


Figure 3.1: Role of GPS in overall scheme

A GPS receiver must pick up the high-frequency low power radio signals of four or more satellites. The receiver knows *where* the satellites are so it knows from who it receives a signal. Because the receiver knows *when* the signal is sent, it can calculate, based on the speed of light, the exact distance to the satellite. Based on this information it can calculate its position with the mathematical principle tri-lateration, where the interception point of the four distance spheres around each satellite is the position of the receiver.

Because of small incorrectness' of measurement, it is almost impossible to find an exact intersection point, however, it isn't difficult for the receiver to calculate a adjustment where there is. Based on this adjustment the receiver synchronizes its internal clock with the atomic clocks of the satellites, again to know exactly *when* a signal is sent.

4.2 DGPS

Differential GPS (DGPS) is an extension of 'normal' GPS. There are external aspects that influence the accuracy of GPS, like a derivation of the speed of light because of the composition of the atmosphere or signals that bounce off large objects. DGPS helps to correct these errors by adding a station to the system that knows its own position exactly. With this and with the signals it receives from satellites it can calculate signal correction information. The station broadcasts this information and DGPS equipped receivers in the area can calculate their position in up to 100 times more accurate.

DGPS can be used for the BAN, in some cases it is preferable to have a high accuracy. When a patient (wearing a BAN) for example is taking a walk along a river and suddenly needs medical

Intra-BAN communication

Extra

help, it would be very useful for the medical team to know on which side of the river they have to be.

References: [HSWO] [BERK] [GPSI] [WSRC]

4.3 Alternatives

LORAN (Long Range Aid to Navigation) is an example of a radio navigation system using land-based radio transmitters and receivers to allow mariners to determine their position. The fixed stations transmit precisely synchronized a signal to mobile receivers. Based on the difference of the time of arrival the signal the mobile receiver calculates its position. Three or more stations are needed to calculate a position.

In the same way the location of a mobile device can be gathered using a cellular network like the GSM. By measuring the signal strength and / or the time delay the distance between three or more transmitters can be calculated and so the position is determined.

At last there are short-range positioning methods like Bluetooth Local Positioning. This also calculates the position by measuring the signal strength and / or the time delay, but a great advantage is that this technique can be used indoors.

4.4 NMEA

The National Marine Electronics Association (NMEA) is a non-profit association composed of manufacturers, distributors, dealers, educational institutions, and others interested in marine electronics environment. It has developed specifications with the purpose to define the interface between various pieces of marine electronic equipment.

The current available NMEA standards are *NMEA Installation Standard*, for professional installation, *NMEA 2000*, which is constructed to interconnect marine electronic equipment onboard vessels, and *NMEA 0183*. The NMEA 0183 interface standard defines electrical signal requirements, data transmission protocol and time, and specific sentence formats. These sentences are the output of the GPS receiver and must be interpreted by the MBU in the BAN.

The idea of NMEA is to send a line of data called a sentence that is totally self contained and independent from other sentences. The advantage of this is that whatever device or program that reads the data can watch for the data sentence that it is interested in and even ignore other sentences that are irrelevant.

Each sentence starts with a "\$", a two letter "talker ID", a three letter "sentenceID", followed by a number of data fields separated by commas and terminated by an optional checksum and a carriage return/ line feed. A sentence may contain up to 82 ASCII characters. If data for a field is not available, the field is empty and no space is between two commas. In this way the receiver should locate the desired data fields by counting commas.

The optional checksum fields consist of a "*" and two hex digits that represent the exclusive OR operation of all the characters between the "\$" and "*"

The standard also allows individual manufacturers to define proprietary sentence formats. These start with "\$P", then a 3 letter manufacturer ID, followed by manufacturer specific data in the standard sentence format.

Most computer programs that provide real time position information understand and expect data to be in NMEA format.

There is a number of standard sentences each with a specific goal, like output to an autopilot, or to a depth meter. The only relevant sentences for this project are those who provide a position: the "GLL" (geographical position in latitude and longitude), "GGA" (Global Positioning System Fix Data) and RMC (Recommended Minimum Navigation Information) sentences. An example of GLL is:

```
$GPGLL,4916.45,N,12311.12,W,225444,A
```

This sentence must be interpreted as follows:

\$	start sentence
GP	talker ID: Global Positioning system receiver
GLL	sentence ID: Geographical position
4916.45	Latitude 49 deg. 16.45 min.
N	North
12311.12,	Longitude 123 deg. 11.12 min. West
W	West
225444	Fix taken at 22:54:44 UTC
A	Data valid

As can be seen there is no checksum in this sentence. This sentence is sent at 4800 baud (bits/second) to a computer over for example a RS232 serial port, see section 2.2.3.

When a GPS receiver will send position information to a MBU it will do this in the form of the NMEA sentence GLL. The GLL sentence consists of 35 characters, each character is 8 bits long, so it should takes 0,06 second to send only this sentence. Because there will be some overhead of the underlying transport layer (Bluetooth) the travel time will increase, but it is realistic to say that at least every second a new NMEA GLL sentence can be received by the MBU.

GGA - Global Positioning System Fix Data

RMC - Recommended Minimum Navigation Information

Reference:

[NMEA] [GPSI] [NEUT] [VOIL] [PCPT]

4.5 GPSD

GPSD was meant to make programs which allow you to show the position of the GPS on an electronic map and make this hardware independent so it can more easily be used by other programs, like chart plotters.

It takes input from a device capable of generating positional data (like a GPS or a Loran (Long Range Aid to Navigation) receiver) and translates it into another, very simple format.

GPSD is a Java Daemon thread. First a GPSD server accepts a TCP connection from a client and then waits for an external request of the client on port 2947.

The request is a plain text string containing letters that specify what information is requested. When a request is received GPSD publishes a response on the same port 2947. The servers' response start with "GPSD" followed by a list of return values separated by commas and ending with a "\0" character.

Table 3.2 shows possible request and the responses.

Character	Meaning	Unit	Data Type	Example response
P	Lat/Lon	Decimal Degrees	2 Doubles separated by a whitespace	P=47.667615-22.327687
D	Date/Time		String	D=02/04/2003 21:15:14
A	Altitude	Meters	Double	A=83.500000
V	Speed	Knots	Double	V=0.000000
S	Status	0=No GPS, 1=No Signal, 2=2D Fix, 3=3D Fix	Integer	S=1
M	Mode	0=No GPS, 1=GPS, 2=DGPS	Integer	M=1

Table 3.2 GPSD requests and responses (from <http://www.pygps.org/gpsd/protocol.html>)

With GPSD it is possible to display only the information that the client (subscriber) asks for. The GPS receiver receives a lot of information in a lot of sentences, by translating this information into GPSd format it becomes possible for the client to ask a “p” and receive just the latitude and longitude.

References:

[PYGP] [FAHA] [GPSW] [MAAR]

4.6 JSR-179

Java Specification Request-179 (JSR-179) is a location Application Program Interface (API) for Java 2 Platform, Micro Edition (J2ME) and is developed under the Java Community Process. This means that JSR-179 describes a standard how to connect a location application program that is running on a small mobile terminal, like a PDA, with the environment and how to let it communicate with it.

We have seen that it is possible to obtain a location in different ways. These location providers have different criteria for operation, for example GPS is for free, but it is possible that the user must pay for other location providers. Because of the design pattern of JSR-179 it is up to the implementation of the application to decide which location provider is selected: the JSR-179 API is based on a creational pattern that is called the “*factory method*”. The specification contains the basic classes needed to request and get a location result. One of these classes, the LocationProvider class, represents the module that is able to determine the location of a terminal. However this LocationProvider class is just an *interface* for creating a specific Location Provider instance. Based on defined criteria of the different location providers the application selects which *specific* LocationProvider is instantiated.

In other words, the application specifies criteria for selecting a location provider and then instantiates the LocationProvider that is able to fulfill these criteria.

This pattern makes it possible to create an object oriented solution where all objects have their own separated concerns and functions. The main advantage of this is that it makes the solution very scalable. When, for example, it is desirable to use another (new) location provision device to locate the terminal, only a new implementation of the LocationProvider class has to be written.

Reference: [GSRA] [JSRS]

5 Integration BAN and Position provision

This chapter describes the design of the BAN location provision service (BAN LPS). The BAN LPS is an addition to the existing BAN and extends the BAN with location provision. The design process is a combination of top-down and bottom up approach, it starts with the service description of the overall service but there are functional components available that will fulfill parts of the BAN LPS, (bottom-up design). The design considers important aspects like scalability, reliability, efficient implementation and authentication of the service.

5.1 Service description BAN LPS

The BAN LPS makes it possible to locate patients while they are not in a healthcare center. To make this possible the patient wears some equipment that is part of the BAN LPS to locates the patient and make it possible for a Healthcare specialist to see the location of a patient on a map.

Figure 4.1 shows the BAN LPS as underlying service layer for the patient and the Healthcare specialist. The BAN LPS makes it possible for the patient to send its location information to the Healthcare specialist, this is shown with the white arrow. The black ellipses are interaction points of end users and the underlying service.

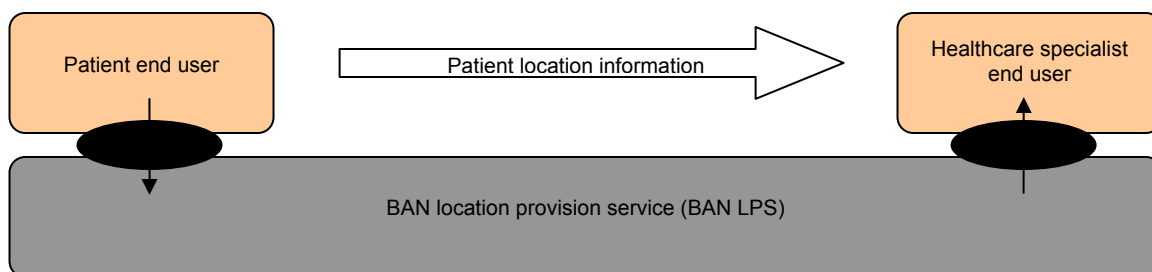


Figure 4.1: the BAN location provision service and two end users

Patient end user

There are multiple patients that need to be supported by the BAN LPS. A patient end user can communicate in one way with the BAN LPS by giving a trigger. The trigger activates the BAN LPS to send the patient location information to the Healthcare center and displays it on a map for the Healthcare specialist end user. A trigger can be caused by manually pressing a button, but also a signal from a sensor, for example a blood pressure sensor that gives an alarm. The trigger is the only one interaction between the patient and the BAN LPS.

Healthcare specialist

There are multiple Healthcare specialist end users that need to monitor the location of patients. A healthcare specialist end user is on a different location than the patient. The healthcare specialist just monitors a screen. When it receives a map with the location of a patient, it can take adequate action.

Also between the BAN LPS and the Healthcare specialist is only one interaction: the BAN LPS gives a healthcare specialist end user the location of a patient by displaying it on a map.

BAN Location Provision Service

Because the service should support multiple patients and multiple healthcare specialists, the BAN LPS is able to identify them to give the correct healthcare centre the correct patient information.

To locate the position of a patient there is functionality that is able to determine the location. Also there is functionality to retrieve and display de location information on a map.

Because it should be possible to display location information at a Healthcare center, there is functionality at the patient side of the BAN LPS that forwards the location information to a Healthcare centre side. This implies that there is functionality at the healthcare side that waits to receive the information from a BAN. Also, there must be functionality at the Healthcare centre side that displays the location information on a map.

According to the amount of data that is send, the communication between the patient side and the Healthcare centre side costs money, so it is very important to keep the data traffic as low as possible.

Because the patient side of the BAN LPS is a mobile unit the resources are scarce. It runs on batteries so it is important to use it economically so it lasts longer. Also processor and memory capacity are limited and should be used efficiently.

It is undesirable that information of a patient becomes known to unauthorized parties. Therefore the information of a patient must be secured and not be accessible for unauthorized people. For this reason it also should be possible to authenticate the users of a BAN.

5.1.1 Decomposition of the BAN LPS

Figure 4.2 shows a decomposition of the BAN LPS in multiple parts, the 'Location Provider', the 'MBU', 'Central server', the 'Map Provider' and the 'Communication service'. Also there is made a separation between the patient side and the Healthcare centre side of the service. The patient side contains the functionality that is physically located at the patient, the Healthcare center side contains the functionality that is physically located on the healthcare center and on the supporting map provider.

Communication services

The purpose of the communication service is to exchange the information between the other (higher)functional building blocks. There is a difference between the communication services, the MBU communication service is at the patient side of the BAN LPS, the Interaction communication service is between the patient side and the Healthcare center side, finally the Extern communication service is at the Healthcare center side.

Chapter 2 describes possible communication technologies that can be used here.

MBU

The MBU system part will contain the functionality to receive a trigger from the Patient end user and communicate with the Communication service.

Each MBU has a unique identifier to identify a calling patient. The MBU contains the functionality to request a location from the MBU communication layer after it has received a trigger from the patient. The MBU communication layer applies this request to a Location provider that gives the location. The communication between the MBU and the Location provider is depicted with the white arrow 'Location'. When the MBU receives a correct location, it is able to forward this information to the central server (the arrow Patient location information) over the Interaction communication service.

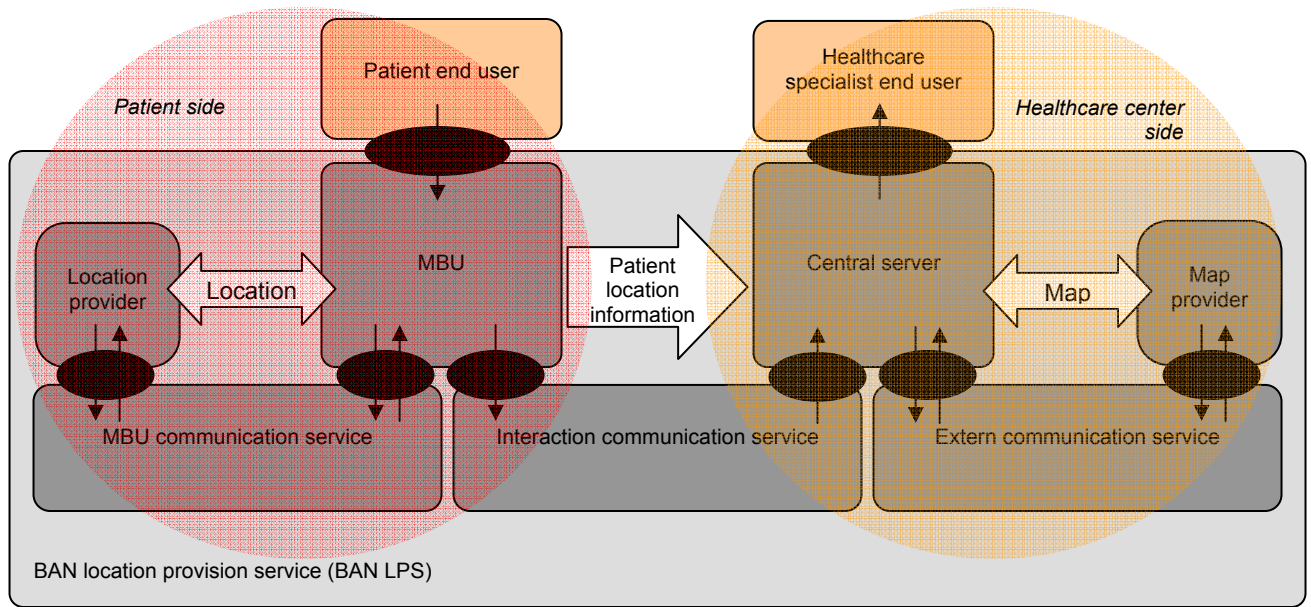


Figure 4.2: decomposition of the BAN in functional building blocks

Location Provider

Like the MBU the Location provider is located at the patient side of the BAN LPS, it is physically on the same location as the patient and the MBU. The Location provider contains functionality to determine a location. The functionality is separated from the MBU functionality because there are multiple possibilities to retrieve a location. In order to guaranty a correct patient location it is preferable to use more than one location provider services for a patient. The services of a Location provider block can be fulfilled by existing services. Chapter 3 describes different location providers that implement the functionality of the Location provider.

Central server

The central server is at the Healthcare centre side between the communication layer and the Healthcare specialist. When it receives Patient location information from the Interaction communication layer, it requests for a map to the Extern communication layer and retrieves it from the Map provider to display the location to the end user.

Map provider

The Map provider waits for a request, when it receives the request with location information, it positions the coordinates on a map and send this back to the requesting Central server. The map provider is at the Healthcare centre side of the BAN LPS, however it need not to be at the same physical location, there are multiple existing possibilities to provide this service.

5.1.2 Decomposition of the MBU

The functionality of the MBU is specified in more detail. The functionality consists of three functional blocks, a 'Trigger', a 'Processing location information' and a 'Forwarding location information'.

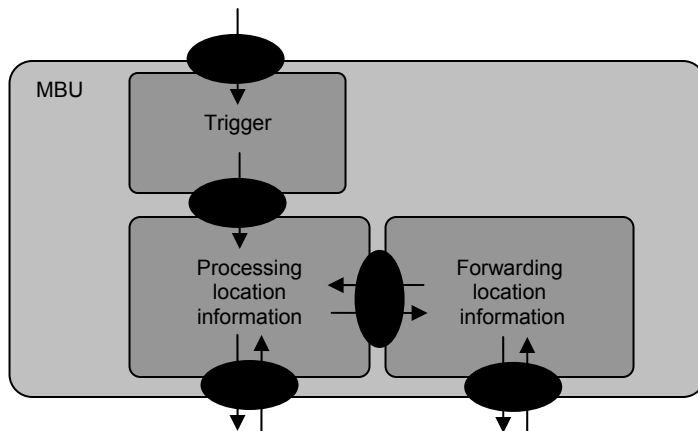


Figure 4.3: decomposition of the MBU in functional building blocks

Trigger

The trigger has two interaction points, one with the patient and one with the Processing location information block. The trigger listens for an external 'trigger signal', it will receive this from the patient end user. When a trigger happens, the trigger activates the process of sending location information to the Healthcare center by activating the Processing location information block.

This trigger functionality should be able to communicate with other sensors in the BAN so it becomes possible to automatically activate the BAN LPS when it receives a certain value of the BAN sensor, for example when blood pressure is too high.

Processing location information

The Processing location information function block has three interfaces: it is activated by a trigger, it requests for location information and receives it from the MBU communication service and it signals the location information to the Forwarding location information service block.

When the processing location information block receives a trigger it determines what Location providers are available and requests the location from a available Location provider. It checks the accuracy of the received location information and activates the Forwarding location information.

Because a signal is not always available (for example when a GPS receiver is inside a house) the Processing location information validates the status of the signal.

Forwarding location information

The Forwarding location information block has two interfaces, after it is activated by the Processing location information block it requests the location with a certain time interval. It forwards patient location information to the Healthcare center over the Interaction communication service.

In order to reduce the data traffic over the Interaction communication service every time it requests a location the Forwarding location information block calculates if the location of a patient has changed. Only when the patient is moving the location is sent. However, the information should be send at least once a certain time interval to indicate to the Healthcare center that there is still a problem, also when the patient is not moving.

There are multiple Healthcare centers this means that there are multiple central servers. To identify a central server they have an unique identifier. The Forwarding location information block also contains functionality to send the information to the correct healthcare center and identifies the patient.

The functionality of the forwarding location is separated from the rest to make it possible to implement different forwarding implementations. According to the type of the connection there are different optimal implementations.

Paragraph 3.6 describes an interface for the implantation of the Processing location information.

5.1.3 Decomposition of the Central server

The functionality of the Central server is specified in more detail. The functionality consists of four building blocks 'Presentation', 'Information processing', 'Information reception' and 'Map provision'.

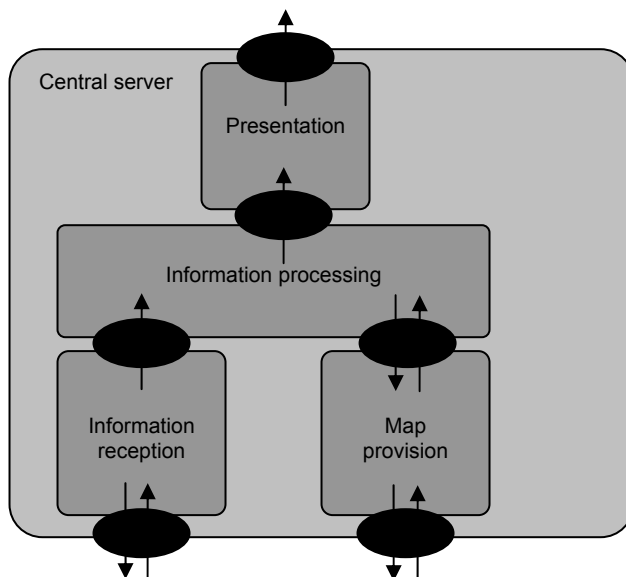


Figure 4.4: decomposition of the Central server in functional building blocks

Information reception

The Information reception block listens for an indication from the external Communication service block. The information is applied to the Information processing service block. It is separated from the information processing block to make it possible to support multiple patients with one MBU. It implements the reception of information independent of the further information processing.

Because one healthcare specialist must be able to monitor multiple patients, the implementation of the Central server must support the communication with multiple MBUs. This is done by the Information reception block, that supports patient location indications from multiple MBUs.

Information processing

The information processing block receives an indication with patient location information from the information reception block then it notifies the Map provision that a new map must be generated. The received map is not stored but directly presented to the Presentation block.

It is obvious that the data is dynamically generated; the location information can change in time when a patient is on the move. The information processing block creates this dynamic information provision.

It is possible that more than one healthcare specialists works with one Central server. To make it possible that multiple healthcare specialists can monitor the same patient the Information

processing block stores the latest received information. A database makes a scalable solution and facilitates the identification of a patient.

Presentation

The presentation block separates the presentation to the end user from the rest of the Central server. This makes it possible to separate the implementation of the presentation from the other information processing. Also multiple instances of the presentation service can interact with one information processing block. Also the presentation service checks the identification of the healthcare specialist.

Map provision

The Map provision receives a request for a map from the Information processing service block. It forwards the request to the external map provider block over the External communication service. When it receives the map it forwards it to the Information processing block.

There are different map providers that facilitate the same functionality but at different costs. It is useful to make it possible to distinguish between them. Different map providers have different interfaces and ways of communication. For this reason the Map provision service block is separated from the Information processing block.

6 Implementation BAN LPS

This chapter describes a possible implementation of the BAN LPS. Section 5.1 gives a short overview of the used hard and software, section 5.2 describes implementation of the patient side of the BAN LPS is described. Each subsection describes a service part, based on the functional building blocks. Section 5.3 describes the implementation of the Healthcare center side.

6.1 *Used hard and software*

The BAN LPS prototype described in chapter 4 is implemented with the programming language Java using the programming environment JBuilder 9 enterprise edition.

The client side of the BAN LPS runs on a COMPAQ iPAQ Pocket PC. In order to do so a Java runtime environment is installed.

At last a GPS Daemon is installed on the COMPAQ iPAQ Pocket PC.

6.2 *Patient side implementation*

6.2.1 Location provider

Chapter 3 describes possible techniques to obtain location information. For the prototype a GPS handheld transceiver is used. (Emtac Bluetooth GPS) This device knows its location using the GPS (paragraph 3.1) and communicates with the environment using Bluetooth (paragraph 2.2). When the device is activated it permanently broadcasts a wide range of NMEA sentences (paragraph 3.4).

6.2.2 MBU communication service

The communication between the Location provider (GPS transceiver) and the MBU is over Bluetooth, described in paragraph 2.2. The authentication of communication between these two devices can be guaranteed with the settings on the MBU. Because there are no costs for using Bluetooth there are no financial limits for the use of this communication channel.

6.2.3 Interaction communication service

The communication between the MBU and the Healthcare center is over GPRS, described in paragraph 2.3.

6.2.4 MBU

Trigger

The functionality of the Trigger block activates the Processing location information functionality when it receives an alarm trigger from a patient.

This block will be implemented in the most simple way, for the prototype it is enough to 'push a button' to activate the system. However, because the trigger is a separated entity it can be improved without affecting the rest of the implementation

Processing location information block

A GPS Daemon tool listens for the reception of the NMEA information from the GPS transceiver. This tool is *not* part of the implementation of the processing location information block. It is installed on the hardware to simplify the reception of location information, see paragraph 3.5.

The Java JSR-179 API is used as a standard interface for the implementation of the Processing location information functionality (paragraph 3.6). A main class is activated by the trigger. This main class selects available location providers. And activates the Forwarding location information functionality.

For the prototype only a GPS LocationProvider implementation obtains the location information from the GPS Daemon. This implementation sets up a connection to port 2947 and request the location information from this port. It receives the location information from the GPS Daemon and validates the correctness of the signal. To reduce processor capacity, this implementation only request and processes the location information with a time interval of 5 seconds. Not every request gives correct location information so this interval cannot be to long.

The use of the JSR-179 makes the implementation scalable because it is possible to add other location implementations that listen for location information of other location providers.

Forwarding location information block

The forwarding location information functionality sends the patient location information to the Healthcare Centre. Because the use of data traffic over GPRS is charged the traffic should be reduced as much as possible.

To implement the interface of the forwarding location information block there are three possibilities:

1. by using Java Remote Method Invocation (RMI) or
2. by using HTTP.
3. opening a socket

An RMI implementation provides a framework for Java objects to communicate regardless their location. This means that the client on the MBU can access the server at the Healthcare centre as if they are executing in the same runtime environment. If an entire distributed application will be implemented in Java and will not need to integrate with other platforms, then RMI may be preferable. To make an application more interoperable and not fully tied to Java, but still using RMI technology, the CORBA platform could be used. However this implementation is quit complicated.

The information to be exchanged is periodic and quite limited, the emphasis is not on the connection and thus can be kept simple. The use of sockets is not preferable because the opening of sockets takes relative much processor capacity.

The most common way to do so is encapsulating the sending information in HTTP. By simply giving a HTTP POST request to the HTTP client, that is send over TCP. At the receiving site there must be functionality that listens for requests.

The implementation of the HTTPClient is based on the 'Jakarta Commons HTTPClient' what is a standard package providing a framework for working with the client- side of the HTTP protocol with most recent HTTP standards and recommendations. This is developed by the Commons Jakarta Apache project group.

Reference:
[JAAP]

Schematical overview of implementation of MBU

The implementation of the MBU is schematically depicted in figure 5.1 . On the PDA there is a Bluetooth receiver that receives the location information from the GPS transceiver. It applies this signal to the GPS Daemon. The implementation runs on a Java Virtual Machine and consist of the entities 'Trigger', 'JSR-179' and 'HTTP client'.

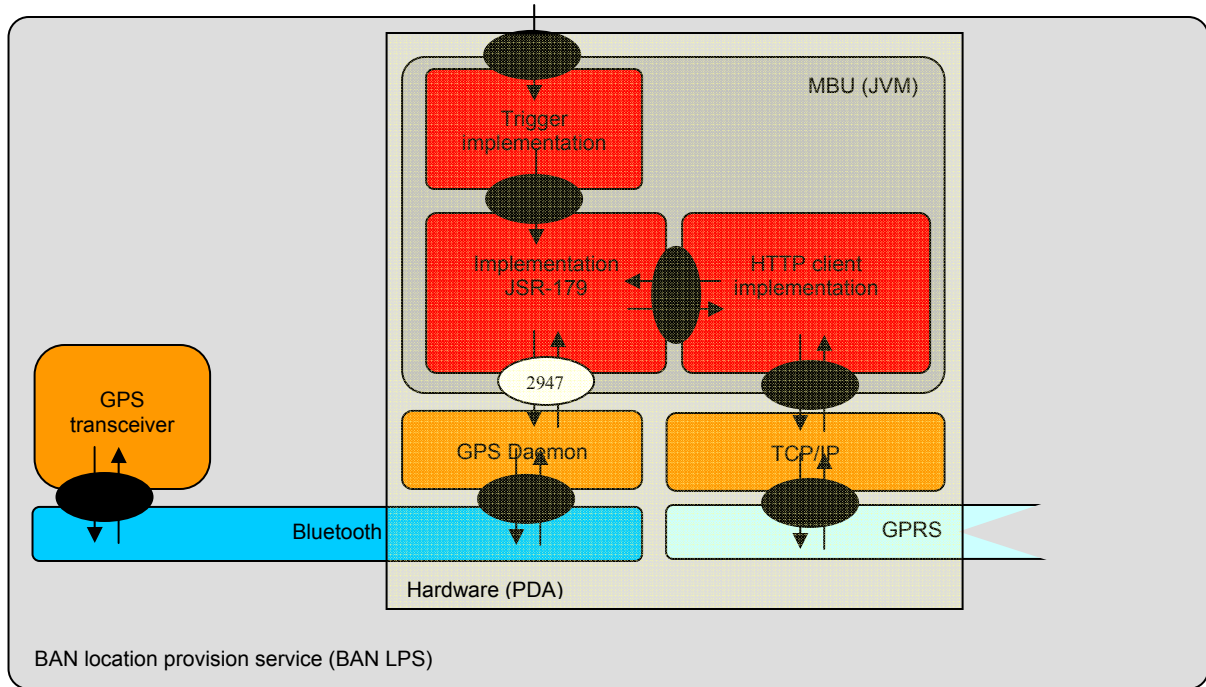


Figure 5.1: protocol stack of MBU

In figure 5.2 an overview of the implemented classes is given. There are two packages; 'javax.microedition.location' contains the implementation of JSR-179, 'nl.utwente.msp.jsr179' implements the Location provider and the HTTPClient.

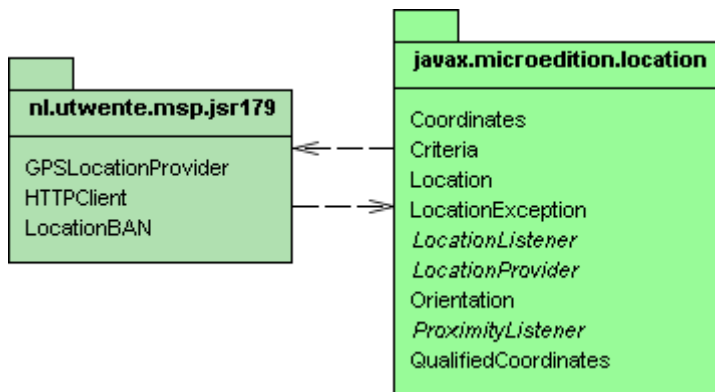


Figure 5.2: Diagram with overview of implemented classes

In figure 5.3 the class diagram of this implementation is given.

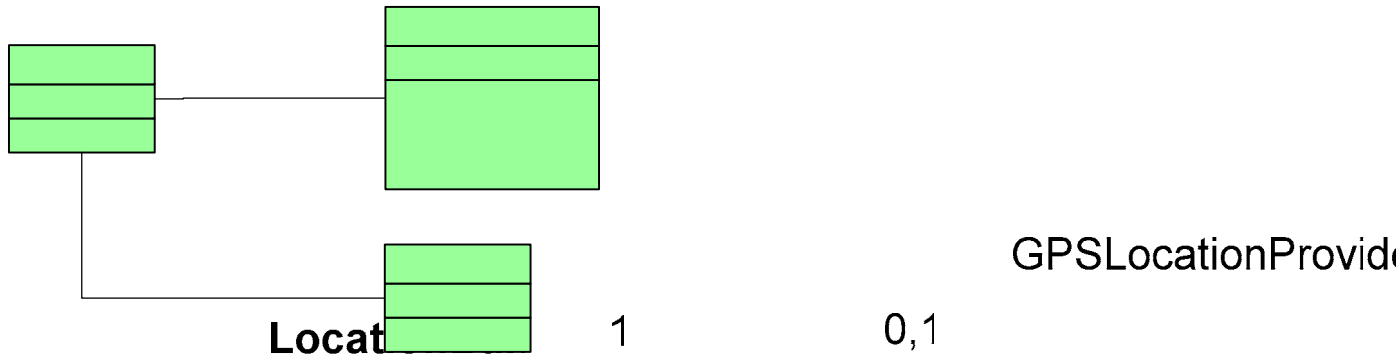


Figure 5.3: Class diagram of package nl.utwente.msp.jsr179

The class LocationBan contains the main method. This method invokes the implementation of JSR-179 and the two classes GPSLocationProvider and HTTPClient.

More information about the implementation of the MBU can be found in the appendix with the code.

6.3 Healthcare center side

6.3.1 Central server

Information processing

Because the programmed with Java, there are two options to implement the Information processing functionality:

- using a Servlet or
- using Java Service Pages.

Both perform the following tasks:

- Generate dynamic content by reading data sent by the user, e.g. from an HTML form.
- Look up information about the HTTP request, e.g. client host name.
- Process the request and generate a result, e.g. writing to a database
- Format the result, e.g. embedding information in HTML document
- Set HTTP response parameters
- Return a document.

JSP's are an extension on top of the Servlet API, they are dynamically compiled into Servlets. The most important difference between the two is that JSP separates the presentation from the content; a JSP can be seen as an HTML web page that embeds application logic to generate dynamic content.

Therefore, for the BAN Location Provision service, JSP pages are most interesting. The Information processing block contains Java functionality that on request dynamically generates an HTML web page.

Both the Information reception functionality as Presentation functionality can be implemented using JSPs. The implementation of the Information processing functionality consists of a

- JSPConnection package and a
- JSPRepresentation package.

A feature of JBuilder is that it has already installed the Tomcat 4.0 server, so during creation this has only to be set in the configuration. Also JBuilder generates java Beans automatically to separate the presentation from the functionality. This is very useful because it makes the implementation very scalable.

The most common way to store data is to create a MySQL database because this is a proved and highly accepted technology. The database is invoked with specific queries. The prototype does not implement a database. The functionality of the JSPConnection package consists of 'set' and 'get' methods to respectively read the input of the parameters and store them. The prototype writes the parameters latitude, longitude, altitude and speed to a textfile when the set methods are invoked.

The structure of the JSPPresentation package is equal to that of the JSPConnection. The JSP file has a GET method containing the input fields to present the parameters for the visitor of the HTML page. It invokes the functionality of the JSPPresentationBean with tags. The get method reads the text files. By putting meta information in the 'head' it is possible to let the HTML page automatically refresh itself at a regular interval.

Information reception

This HTML form is generated dynamically by the information processing implementation. The JSPConnection consists of a JSP file that has a POST method form containing several input fields for receiving the parameters of the BAN and tags to invoke the functionality of the Java file. This is the Java JSPConnectionBean file that contains the functionality for accessing and processing the received parameters.

Map provision

This implementation receives a request from the Information processing implementation and invokes the external Map provider to request a Map. A possible implementation is the Microsoft MapPoint client which provides a map with the location information. Because a lack of time this is not implemented.

Presentation

The presentation makes it possible for the end user, the Healthcare specialist, to monitor a patient. This HTML form is generated dynamically by the information processing implementation. An HTTP request to the IP address of the Presentation results in the presentation of a map with the location of the patient.

To guaranty the privacy of patient information the HTML form is accessible after authentication using a login module. Also it is possible to create a secure connection to the central server by tunneling the data over the (Inter)network This is implemented with the Secure Socket Layer (SSL) protocol. It is not the scope of this project to implement one of these for the prototype.

For the implementation of the central server the protocol stack looks like this:

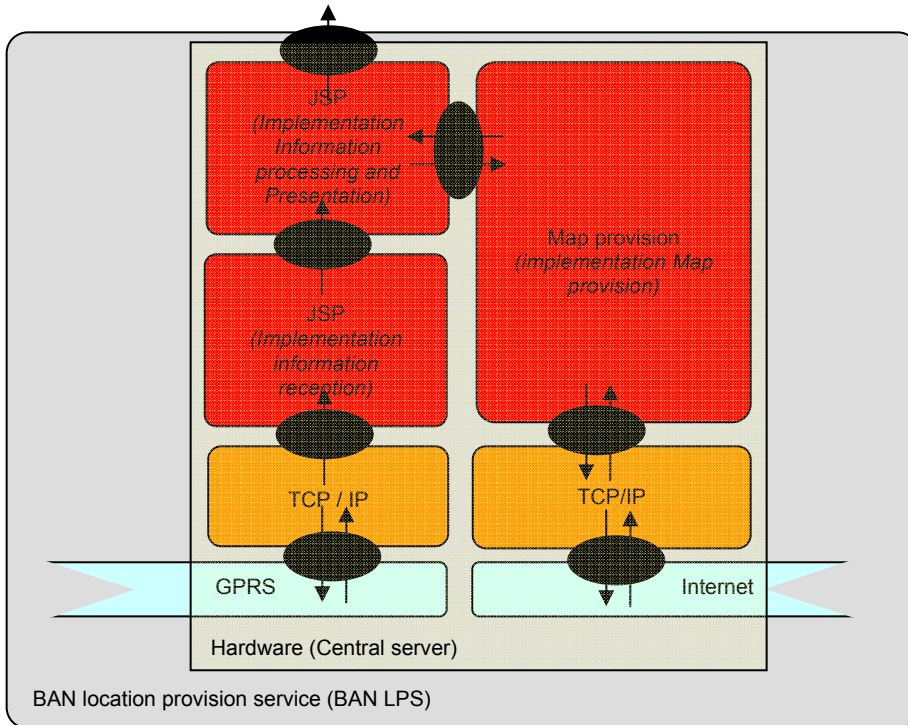


Figure 5.4: Protocol stack of central server

Reference:[WIKI] [JAME] [MIWS]

7 Evaluation

7.1 Research questions

This section evaluate the research question described in section 1.3.

1. Examine the technologies that are used in the Body Area Network. Hereby only the relevant technologies will be examined in detail, for more information I will refer to secondary literature. Chapter 2 gives a detailed description of this.
2. Examine in what way the position information of a Body Area Network can be provided. This consists of detailed relevant technology description but also of problems and solutions that will overcome that problem. This is examined in chapter three. Different aspects of the Location provision process are examined and give a detailed view of the current technological possibilities.
3. Modulate a possible solution for integrating the position provision in the existing Body Area Network. Thereby, the position of a BAN must be determined with a sensor integrated with the BAN and be shown on a remotely located map. A technical description of how the solution works is given. This is described in chapter 4. First the highest level services is defined; this consists of functional building blocks that together form the Location provision service of the BAN.
4. Implement a prototype of a Body Area Network with position provision according to the designed model. This is described in chapter 5. Implementation solutions of the functional building blocks are given, alternatives are considered. The code can be viewed in the appendix. The system works, only no map provision is invoked. Thereby, it is obvious that this implementation is a very simple prototype. The system must be extended for a great deal to generate a practical application.

7.2 Evaluation of design

The next part considers the most important design aspects of the BAN LPS and gives suggestions.

Scalability

Considering the scalability the current system is quite flexible, the JSR-179 offers possibilities to extend the functionality and thereby increase the service of the BAN at the MBU site. Also by separating the receiving and processing site from the presentation site of the central server it is possible to extend both sites in an independent way. For example, to extend the presentation to the monitoring specialist by showing more detailed patient information.

What should be done to make a more efficient and scalable design is to implement a data base to store the patient (MBU) information. This is now written to text files.

Reliability

It is obvious that a location provision service only based on GPS is not reliable. When a MBU is inside a building no GPS location information can be received. By creating more location providers that calculate the location in different ways the systems reliability would increase. This could be done by implementing location providers as described in section 3.3.

In the Location provision service, there is a lot of communication between different devices: between Location sensor and MBU, between MBU and central server, between monitoring health care specialist and central server and at last between the external map provider and the central

server. To be able to give a correct judgment about the reliability it is useful to examine what the risk of failure is and how that could be decreased, for each communication line.

It is useful for an end user to see more information of the received signal. JSR-179 offers possibilities to give a detailed status of the information. Also it is more reliable to permanently check the status and to send the last *correct* information when a trigger happens.

Efficiency implementation

As described, during implementation it appeared that the GPS Daemon did not function properly. I recommend to discard the use of GPSD, it is more effective to read direct from the Transceiver. To do so, functionality should be changed but it is possible to directly read NMEA sentences. By doing so the overall application on the MBU becomes more efficient.

The HTTPClient sends the parameters in the POST request as name-value pairs in an array. In the JSPConnectionBean for every String in this array a separate method is written to access the parameters. This is not very efficient, especially when the number of parameters becomes larger. It would be better to create just one input field in the JSP page that reads an array object and that the Bean functionality dynamically creates variables based on this array. I have read that this is very good possible, but did not implement it because of a lack of time.

Considering the data and cost reduction, the prototype is at this moment not efficient. However only the latitude and longitude is sent over GPRS, it is even sent when the location has not changed. It should only be sent when the location of a patient changes.

Authentication and security

In the prototype there is no form of authentication at all. Because the system processes personal and critical information, this should of course be secured. Information should be stored and encoded, end users should first have permission to access information, the information should be sent over secured connections, etc.

7.3 Experiences during implementation

This section describes my experiences during the implementation. Most of the problems seem to be easy but took a lot of time to solve. I used Netcat, specifications, the API, help functions, the Internet, secondary literature and of course a lot of print lines in the text editor to find out what was going on and solve the problems I ran into. However, not all problems are solved because finding or implementing their solutions simply would take too much time.

The first tests on the PDA only generated errors because I had used a method from the J2EE API instead of the J2ME API. This stupid mistake took a lot of useless programming time.

GPSD is the daemon that is positioned between the GPS receiver and the application (JSR-179). It should make the implementation of the application more easily by filtering NMEA output on request of the application. I found out that there were a few problems with the GPS Daemon that was installed on my iPAQ. By using the Linux test program Netcat I found out that the Daemon did not recognize all the specified commands. For example it did not return a value (NULL) for the command "t" that should return the current track. This was also the case for other commands that however are less relevant for the BAN.

Also the GPS Daemon consequently returned "NULL" at every first time requesting an "s". I also found this by testing with Netcat.

Another problem was the not locking of the GPS transceiver. This not only made it very difficult to test my program, but also made it clear that using position provision only with a GPS receiver is far by reliable when it is located inside a building. For this reason, in my implementation of the

application I have not made a distinction between a DGPS and GPS signal because just fixing the signal appeared to be hard enough.

In addition, for some reason sometimes the Bluetooth connection between the GPS transceiver and the iPAQ suddenly broke down.

It was hard to find out how to send information to the central server. First I was trying to use the Java.net packages to make a connection to the central server over HTTP. It appeared that the HTTPClient from Jakarta offered a better solution. But to let this work I had to download "Commons-httpclient-2.0.1" and "Commons-logging-1.0.4". It took a lot of time to figure this out and configure this properly. When this finally was done, implementing the HTTPClient was not difficult anymore.

According to the design, the location information of the BAN should be written to a database among other things because of scalability. I didn't make it to implement this. The most common database solution is MySQL but I was simply not allowed to install MySQL on my workstation. After that I have tried to generate a JBuilder database, the code of it is out-commented, but I also stopped that because this was to time consuming.

The last part of the project is to present the location of the BAN with a dot on a map. I have spent a lot of time searching for online map providers that respond to input of coordinates. Although there are a lot of map providers that give the location based on address information I couldn't find one that satisfies my requirements. The other option was to create a local map providing service. This should be done by using 'Microsoft map client'. But using this appeared to be more complex, so finally, because of lack of time the presentation on a map is discarded and only the coordinates are given.

During implementation of the JSP pages on the central server the use of separating functionality in different building blocks (packages) was already rewarded. This made it possible to create the presentation to the end user completely independent of the functionality of the 'BAN communication' functionality. By storing the information in a standardized database, both parts can be separated developed and updated.

All this information exchange is over external communication channels and carries therefore a certain risk.

8 References

- [ReM0] Biemans, F.P.M. and Brussee, R. and Lacob, M.E. and Jonkers, H. and Lankhorst, M.M. and Porskamp, P.A.P. and Slagter, R.J., 2002, *Reference Models for Networked Applications, 2nd edition*, Telematica Instituut, Enschede, 96p.
- [JANP] Hughes, M. and Shoffner, M. and Hamner, D., 1999, *JAVA Network Programming*, Manning Publication Co., Greenwich, 807p.
- [JASS] Lewis, J. and Loftus, W., *Java Software Solutions, 2nd edition*, Addison Wesley Longman Inc, USA, 780p.
- [FIDJ] Dokovsky, N. and Halteren, A. and Widya, I., *BANip: enabling remote healthcare monitoring with Body Area Networks*, University of Twente, Enschede, 11p.
http://aps.ewi.utwente.nl/public/bibliografie/FIDJI2003_final.pdf (July 2004)
- [KJHO] Hole, K., August 2004, *Bluetooth Part 7: RFCOMM*,
<http://www.kjhole.com/Standards/BT/BT-PDF/Bluetooth7.pdf> (June 2004)
- [RSSS] Clark, N., August 98, *RS232 Serial Port*,
<http://www.ctips.com/rs232.html> (June 2004)
- [TEKM] TekTronix, Inc, 2000, *GPRS Protocol Testing in the Wireless World*,
http://www.tek.com/Measurement/App_Notes/2F_14253/eng/2FW_14253_0.pdf
(June 2004)
- [MOBIL] Mobile Streams, 2001, *All about General Packet Radio Service (GPRS) on mobile networks mobileGPRS*, <http://www.mobilegprs.com/gprs.asp?link=1> (July 2004)
- [VOCA] Marin, J.A.T., 2004, Vocal Technologies LTD., One stop provider of communications software, *GPRS Protocol Stack*, http://www.vocal.com/data_sheets/gprs_protocol_stack.html (July 2004)
- [EEOU] Vallstrom, J., 1997, *General Packet Radio Service (GPRS)*
<http://www.ee.oulu.fi/~fiat/gprs.html> (July 2004)
- [COSP] Bluetooth SIG inc., 2001, Specification Volume 1 Specification of the Bluetooth System, *Core*, Bluetooth SIG, 1084p.
http://www.comms.scitech.susx.ac.uk/fft/bluetooth/Bluetooth_11_Specifications_Book.pdf
(June 2004)
- [PRBO] Bluetooth SIG inc., 2001, Specification Volume 2 Specification of the Bluetooth System, *Profiles*, Bluetooth SIG, 452p.
http://www.comms.scitech.susx.ac.uk/fft/bluetooth/Bluetooth_11_Profiles_Book.pdf
- [PRIN] Stone Street One Inc., 2002-2003, Enabling the wireless revolutions, *An introduction to Bluetooth*,
<http://www.stonestreetone.com/PDF/IntroductiontoBluetooth.pdf> (July 2004)
- [UMWO] UMTS World.com, 2003, UMTS World WCDMA specification and information page, *MCDMA (UMTS)*, <http://www.umtsworld.com/technology/wcdma.htm> (August 2004)
- [PROT] Protocols.com, *UMTS Family*, <http://www.protocols.com/pbook/umtsfamily.htm> (August 2004)
- [UMFO] umts-forum.org, 2004, UMTS Forum, *Network evolution*,
http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/3G_Network_gsm (August 2004)
- [CELL] Cellularonline, CellularOnline one of the worlds favorite mobile portals, *GSM Data Speed Evolution To UMTS*, http://www.cellular.co.za/data_speed_evolution.htm (June 2004)
- [GSAC] Global mobile Suppliers Association, 2004, *Evolution of GSM to 3G/IMT-2000 via GPRS/ EDGE/ WCDMA*, http://www.gsacom.com/gsm_3g/index.php4 (July 2004)
- [WEBO] Jupitermedia Corporation, 2004, Webopedia: Online Computer Dictionary for Computer and Internet Terms and Definitions <http://www.webopedia.com/TERM/C/CDMA.html> (June 2004)
- [RFCE] RFC Editor, 2004, RFC Editor, *Official Internet Protocol Standards*,
<http://www.rfc-editor.org/rfcxx00.html> (July 2004)
- [HSWO] How Stuff Works inc., 2004, How Stuff Works, *How GPS Receivers Work*,
<http://electronics.howstuffworks.com/gps6.htm> (July 2004)
- [BERK] Berkley, 2004, California Oak Mortality Task Force, *Monitoring Committee SOD Monitoring GPS Information*, <http://camfer.cnr.berkeley.edu/oaks/GPSProtocol.html>
(July 2004)
- [GPSI] Bennet, P., 2004, NMEA-0183 and GPS Information, *NMEA data*,
<http://www.gpsinformation.org/dale/nmea.htm> (July 2004)

- [WSRC] Rupprecht, W., 2004, DGPS corrections over the Internet, <http://www.wsrcc.com/wolfgang/gps/dgps-ip.html> (July 2004)
- [NMEA] NMEA, 2002, National Marine Electronics Association, Publications and standards, *NMEA 0183 Standard For Interfacing Marine Electronic Devices* (Version 3.01) <http://www.nmea.org/pub/index.html> (July 2004)
- [GPSI] DePriest, D., NMEA data, <http://www.gpsinformation.org/dale/nmea.htm> (July 2004)
- [NEUT] Leadtek Research Inc., *GPS Protocol Reference Manual* (Revision 1.30), http://neutrino.phys.washington.edu/~berns/archive/Leadtek/GPS_Protocol_ReferenceManual.pdf (July 2004)
- [VOIL] Couderc C., 2004, Bus automobile CAN, *Standard marine NMEA 2000*, http://www.voilelec.com/pages/can_nmea.php (July 2004)
- [PCPT] Piechulla, W., 2002, Understanding NMEA, <http://pcptpp030.psychologie.uni-regensburg.de/trafficresearch/NMEA0183/> (July 2004)
- [PYGP] Gpsd, *A gps Service Daemon*, <http://www.pygps.org/gpsd/gpsd.html> (June 2004)
- [FAHA] Handhelds.org, 2004, The Familiar Project, <http://familiar.handhelds.org/> (July 2004)
- [GPSW] Center for Advanced Studies, Research and Development in Sardinia (CRS4), 2004, The Web Around The Corner: Augmenting the Browser with GPS, GPSWeb (v1.2), <http://www.crs4.it:8000/gpsweb/14> (July 2004)
- [MAAR] Karlsen, E., 1997, A Users Plea for Daemon Threads, <http://www.mail-archive.com/glasgow-haskell-users@haskell.org/msg00175.html> (July 2004)
- [GSRA] Raj, G. S., 1999, Web Cornucopia, *The Factory Method (Creational) Design Pattern* <http://gsraj.tripod.com/design/creational/factory/factory.html> (July 2004)
- [JSRS] JSR-179 Expert Group 2003 (Nokia Corporation), *Location API specification for Java™ 2 Micro Edition* (Version 1.0), http://www-users.cs.umn.edu/~czhou/docs/jsr179/jsr179_FinalRelease10.pdf (July 2004)
- [WIKI] Wikipedia, 2004, Wikipadia the free encyclopedia, *Internet protocol suite*, <http://en.wikipedia.org/wiki/TCP/IP> (July 2004)
- [JAME] Hadzic, D., 2004, *INTRODUCTION TO JAVA MICRO EDITION (J2ME)* (VERSION: 1), http://heim.ifi.uio.no/~dinkoh/docs/javame_introduction_ver1.pdf (July 2004)
- [JAAP] Apache Software Foundation, 2004, The Apache Software Project, *HTTP Client*, <http://jakarta.apache.org/commons/httpclient/> (august 2004)
- [MIWS] Microsoft Corporation, 2004, Microsoft MapPoint, *Mappoint Webservice*, <http://www.microsoft.com/mappoint/webservice/default.aspx> (September 2004)
- [ETSI] European Telecommunications Standards Institute (ETSI), 2004, Telecom Standards, <http://www.etsi.org> (June 2004)
- [RFC2] Fielding, R. and Gettys, J. and Mogul, J. and Frystyk, H. and Masinter, L. and Leach, P. and Berners-Lee, T., June 1999, Hypertext Transfer Protocol, *HTTP/1.1 IETF RFC 2616 (Draft Standard)*, 176 pages, www.ietf.org/rfc/rfc2616.txt (June 2004)
- [PALO] Palo Wireless, Bluetooth Recourse Centre, <http://www.palowireless.com/infotooth/tutorial/baseband.asp> (June 2004)

Position of a Body Area Network

A design for integration of a Body Area Network with existing location provision technologies.

Appendix 1: Implementation

Author: H. Schaap
Date: January 2005
Faculty: Electrical Engineering, Mathematics and Computer Science
Group: Architecture and Services of Network Applications (ASNA)
Tutor: dr.ir. A.T. van Halteren



Index

1	PACKAGE JAVAX.MICROEDITION.LOCATION	2
1.1	CLASS COORDINATES	2
1.2	CLASS CRITERIA	3
1.3	CLASS LOCATION	4
1.4	CLASS LOCATIONEXCEPTION	5
1.5	CLASS LOCATIONPROVIDER	5
1.6	CLASS QUALIFIEDCOORDINATES	6
2	PACKAGE NL.UTWENTE.MSP.JSR179	6
2.1	CLASS GPSLOCATIONPROVIDER	6
2.2	CLASS HTTPCLIENT	9
2.3	CLASS LOCATIONBAN	11
3	PACKAGE JSPPRESENTATION	12
3.1	CLASS JSPPRESENTATIONBEAN	12
3.2	JSPPRESENTATION	14
4	PACKAGE HTTPSERVER	15
4.1	CLASS JSPCONNECTIONBEAN	15
4.2	JSPCONNENCTION	16

1 Package javax.microedition.location

1.1 Class Coordinates

```
package javax.microedition.location;

public class Coordinates {

    static int DD_MM;
    static int DD_MM_SS;
    private double lat_ = 12;
    private double lon_ = 13;
    private float alt_ = 14;

    /**
     * Creates a new instance of Coordinates
     */
    public Coordinates(double latitude, double longitude, float altitude) {
        lat_ = latitude;
        lon_ = longitude;
        alt_ = altitude;
    }

    float azimuthTo(Coordinates to) {
        throw new RuntimeException("Not Implemented! Used to compile Code");
    }

    static String convert(double coordinate, int outputType) {
        throw new RuntimeException("Not Implemented! Used to compile Code");
    }

    static double convert(String coordinate) {
        throw new RuntimeException("Not Implemented! Used to compile Code");
    }

    public float distance(Coordinates to) {
        throw new RuntimeException("Not Implemented! Used to compile Code");
    }

    public float getAltitude() {
        return alt_;
    }

    public double getLatitude() {
        return lat_;
    }

    public double getLongitude() {
        return lon_;
    }

    public void setAltitude(float altitude) {
        alt_ = altitude;
    }
}
```

```
public void setLatitude(double latitude) {
    lat_ = latitude;
}

public void setLongitude(double longitude) {
    lon_ = longitude;
}
}
```

1.2 Class Criteria

```
package javax.microedition.location;
```

```
public class Criteria {

    static int NO_REQUIREMENT = 0;
    static int POWER_USAGE_LOW = 1;
    static int POWER_USAGE_MEDIUM = 2;
    static int POWER_USAGE_HIGH = 3;

    //hard criteria:
    private boolean costAllowed_;
    private boolean speedCourseRequired_;
    private boolean addressInfoRequired_;
    private boolean altitudeRequired_;
    private int powerConsumption_;

    /**
     * Creates a new instance of Criteria
     * set default values to Criteria
     */
    public Criteria() {
        costAllowed_ = true;
        powerConsumption_ = NO_REQUIREMENT;
        speedCourseRequired_ = false;
        addressInfoRequired_ = false;
        altitudeRequired_ = false;
    }

    //cost criteria
    public void setCostAllowed(boolean costAllowed) {
        costAllowed_ = costAllowed;
    }

    public boolean isAllowedToCost() {
        return costAllowed_;
    }

    //Powerconsumtion criteria
    public void setPowerConsumption(int level) {
        if ((level >= NO_REQUIREMENT) && (level <= POWER_USAGE_HIGH)) {
            powerConsumption_ = level;
        }
    }

    public int getPowerConsumption() {
```

```
    return powerConsumption_;
}

//Speed en course criteria
public void setSpeedAndCourseRequirement(boolean speedCourseRequired) {
    speedCourseRequired_ = speedCourseRequired;
}

public boolean isSpeedAndCourseRequired() {
    return speedCourseRequired_;
}

//address info criteria
public void setAddressInfoRequirement(boolean addressInfoRequired) {
    addressInfoRequired_ = addressInfoRequired;
}

public boolean isAddressInfoRequired() {
    return addressInfoRequired_ = false;
}

//altitude criteria
public void setAltitudeRequirement(boolean altitudeRequired) {
    altitudeRequired_ = altitudeRequired;
}

public boolean isAltitudeRequired() {
    return altitudeRequired_;
}
}
```

1.3 Class Location

```
package javax.microedition.location;
```

```
public class Location {

    public final static int MTA_ASSISTED = 1;
    private static QualifiedCoordinates coordinates_;
    static float speed_;
    static float course_;

    /**
     * Creates a new instance of Location
     */
    public Location(QualifiedCoordinates coord) {
        coordinates_ = coord;
    }

    /**
     * Returns the coordinates of this location and their accuracy.
     * Static method so it can be invoked directly from other classe
     */
    public static QualifiedCoordinates getCoordinates() {
        return coordinates_;
    }
}
```

```
//returns the speed
public static float getSpeed() {
    return speed_;
}

//returns the course
public float getCourse() {
    return course_;
}

//set the speed
public static void setSpeed(float sp) {
    speed_ = sp;
}

public static void setCourse(float co) {
    course_ = co;
}
}
```

1.4 Class LocationException

```
package javax.microedition.location;
```

```
public class LocationException extends Exception {

    /**
     * Creates a new instance of LocationException
     */
    public LocationException() {
    }

    public LocationException(String s) {
    }
}
}
```

1.5 Class LocationProvider

```
package javax.microedition.location;
```

```
import nl.utwente.msp.jsr179.GPSLocationProvider;
```

```
public abstract class LocationProvider {

    public static int AVAILABLE = 1;
    public static int TEMPORARILY_UNAVAILABLE = 2;
    public static int PERMANENTLY_UNAVAILABLE = 3;

    /**
     * Creates a new instance of LocationProvider
     */
    protected LocationProvider() {
    }
}
```

```

}

/**
 * factory method; return new specific LocationProvider object
 * check first the criteria for the selection of the location provider
 *
 */
public static LocationProvider getInstance(Criteria criteria) {
    if ((criteria != null) && (!criteria.isAllowedToCost()) && (criteria.isSpeedAndCourseRequired())) {
        return new GPSTLocationProvider();
    }
    /*
    possible for future use:
    else{
    We need a LocationProvider that may cost money, for example a UMTS LocationProvider
    return new UMTSLocationProvider();
    }
    */
    return new GPSTLocationProvider(); //default return GPSTLocationProvider
}

public static Location getLastKnownLocation() {
    throw new RuntimeException("Not Implemented! Used to compile Code");
}

public abstract Location getLocation(int timeout) throws LocationException;

public abstract int getState();

public abstract void reset();

public abstract void setLocationListener(LocationListener location, int interval, int timeout, int
maxAge);
}

```

1.6 Class QualifiedCoordinates

```
package javax.microedition.location;
```

```

public class QualifiedCoordinates extends Coordinates {

    /** Creates a new instance of QualifiedCoordinates */
    public QualifiedCoordinates(double latitude, double longitude, float altitude, float horizontalAccuracy,
float verticalAccuracy) {
        super(latitude,longitude,altitude);
    }
}

```

2 Package nl.utwente.msp.jsr179

2.1 Class GPSTLocationProvider

```
package nl.utwente.msp.jsr179;
```

```

import javax.microedition.location.*;
import java.io.*;

```

```
import java.net.*;

public class GPSLocationProvider extends LocationProvider {

    private Socket gs;
    private OutputStream out;
    private DataOutputStream output;
    private InputStream in;
    private BufferedReader input;
    public static int state;
    static final String host = "localhost";
    static final int port = 2947;

    /**
     * Creates a new instance of GPSLocationProvider, based on criteria, tries
     * to open connection to port 2974 and sets state of received signal
     */
    public GPSLocationProvider() {
        try {
            gs = new Socket(host, port);
            out = gs.getOutputStream();
            output = new DataOutputStream(out);
            in = gs.getInputStream();
            input = new BufferedReader(new InputStreamReader(in));
            this.setState();
        }
        catch(UnknownHostException unknownHostException) {
            state = LocationProvider.PERMANENTLY_UNAVAILABLE;
            System.out.println("No connection to GPS tranceiver, unknown host");
            return ;
        }
        catch(IOException ioEx) {
            state = LocationProvider.PERMANENTLY_UNAVAILABLE;
            System.out.println("No connection to GPS tranceiver, IOException");
            return;
        }
    }

    /**
     * returns a location object, based on status of GPS Tranceiver.
     */
    public Location getLocation(int timeout) throws LocationException {
        int i = state;
        if(i==1) {
            System.out.println("    AVAILABLE; correct lock...");
        }
        if(i==2) {
            System.out.println("    TEMPORARLY_UNAVAILABLE; system cannot lock... last known
information:");
            state = LocationProvider.PERMANENTLY_UNAVAILABLE;
            throw new LocationException("TEMPORARLY_UNAVAILABLE");
        }
        if(i==3) {
            System.out.println("    PERMANENTLY_UNAVAILABLE; system cannot lock... last known
information:");
        }
    }
}
```

```
try {
    String p = "p"; // position
    output.writeBytes(p);
    output.flush();
    String positionString = input.readLine();
    int indexMinus = positionString.indexOf(" ");
    String latString = positionString.substring(8,indexMinus);
    String lonString = positionString.substring(indexMinus);
    double lat = Double.parseDouble(latString);
    double lon = Double.parseDouble(lonString);

    String a = "a"; // altitude
    output.writeBytes(a);
    String altGPSD = input.readLine();
    String altString = altGPSD.substring(8);
    float alt = Float.parseFloat(altString);

    String v = "v"; // speed
    output.writeBytes(v);
    String velGPSD = input.readLine();
    String velString = velGPSD.substring(8);
    float vel = Float.parseFloat(velString);
    Location.setSpeed(vel);
    /*
    String t = "t"; // course, unknown for GPS Daemon
    output.writeBytes(t);
    String trackGPSD = input.readLine();
    String trackString = trackGPSD.substring(8);
    float course = Float.parseFloat(trackString);
    Location.setCourse(course);
    */
    this.setState();
    QualifiedCoordinates coo = new QualifiedCoordinates(lat, lon, alt, 0, 0);
    return new Location(coo);
}
catch(IOException ex1) {
    System.out.println("IOException");
    return null;
}
catch(StringIndexOutOfBoundsException ex2) {
    System.out.println("StringIndexOutOfBoundsException ");
    return null;
}
catch(NullPointerException ex3) {
    System.out.println("NullPointerException in getLocation method ");
    //TEST CODE
    QualifiedCoordinates coo = new QualifiedCoordinates(115, 215, 315, 415, 12);
    return new Location(coo);
    //END TEST CODE
}
}

/**
 * Set state of LocationProvider, write request for status, read the response.
 * if not available, try 200 times, after that, set PERMANENTLY_UNAVAILABLE
 */
```

```
private void setState() throws IOException {
    int state_;
    boolean noFix = true;
    boolean MAX = true;
    int count = 1;
    while(noFix && MAX){
        output.writeBytes("s");
        String stateGPS = input.readLine();
        String sstate_ = stateGPS.substring(8);
        boolean s1 = sstate_.equals("1");
        boolean s2 = sstate_.equals("2");
        if(s1 || s2) {
            state_ = LocationProvider.AVAILABLE; // GPSD status 1
            noFix = false;
            state = state_; // set the status
        }
        else {
            count++;
            state = LocationProvider.TEMPORARILY_UNAVAILABLE; // GPSD status 2
        }
        if(count == 200) {
            MAX = false;
            state = LocationProvider.PERMANENTLY_UNAVAILABLE; // GPSD status 3
        }
    }
}

public int getState() {
    return state;
}

public void setLocationListener(LocationListener location, int interval, int timeout, int maxAge) {
}

public void reset() {
    try{
        gs.close();
        out.close();
        in.close();
        input.close();
        System.out.println("Connection closed...");
    }
    catch(IOException ex) {
        return;
    }
    catch(NullPointerException np) {
        System.out.println("NullPointerException by closing system");
        return;
    }
}
}
```

2.2 Class HTTPClient

```
package nl.utwente.msp.jsr179;
```

```
import javax.microedition.location.*;
import java.io.*;
import java.net.*;
import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.*;

public class HTTPClient extends Thread {

    private double la;
    private double lo;
    private double al;
    private float sp;
    private String lat;
    private String lon;
    private String alt;
    private String spe;

    public HTTPClient() {
    }

    /**
     * Creates a HTTPClient, parses post method on specific URL, with lat, lon
     * alt and spe as the parameters. A RetryHandler tries 3 times to make connection
     */
    public void run() {
        String url = "http://130.89.218.225:8080/jspWebApp/jspConnection.jsp";//
        "http://localhost:8080/jspWebApp/jspConnection.jsp";
        try {
            la = Location.getCoordinates().getLatitude();
            lo = Location.getCoordinates().getLongitude();
            al = Location.getCoordinates().getAltitude();
            sp = Location.getSpeed();

            lat = Double.toString(la);
            lon = Double.toString(lo);
            alt = Double.toString(al);
            spe = Float.toString(sp);

            //TESTCODE
            //String[] posInfo = {lat, lon, alt,spe};
            //for (int leng=0; leng < posInfo.length; leng++)
            //    System.out.println("posInfo is: " + posInfo[leng]);
            System.out.println("posInfo is: " + lat);
            //END TESTCODE
        }
        catch(NullPointerException nullPointerException) {
            System.out.println("NullPointerException by getLatitude in HTTPClient");
            return;
        }

        HttpClient client = new HttpClient();

        PostMethod method = new PostMethod(url);

        NameValuePair[] data = {
            new NameValuePair("lat", lat),
```

```

        new NameValuePair("lon", lon),
        new NameValuePair("alt", alt),
        new NameValuePair("spe", spe));

method.setRequestBody(data);

DefaultMethodRetryHandler retryhandler = new DefaultMethodRetryHandler();
retryhandler.setRequestSentRetryEnabled(false);
retryhandler.setRetryCount(3);
method.setMethodRetryHandler(retryhandler);

try {
    int statusCode = client.executeMethod(method);
    if (statusCode != HttpStatus.SC_OK) {
        System.err.println("HTTP request failed: " + method.getStatusLine());
    }
    byte[] responseBody = method.getResponseBody();
    // TEST CODE
    System.out.println(new String(responseBody));
    // END TEST CODE
}
catch (IOException e) {
    System.err.println("Failed to download file.");
    e.printStackTrace();
}
finally {
    method.releaseConnection(); // Release the connection.
    System.out.println("TCP connection closed");
}
}
}
}

```

2.3 Class LocationBAN

```

package nl.utwente.msp.jsr179;

import javax.microedition.location.*;
import javax.swing.*;

public class LocationBAN {

    public static Location location;
    public static LocationProvider test;
    public static HTTPClient client;
    public static boolean alarm = true;

    /**
     * main: specifies criteria, creates a LocationProvider object
     * When alarm status is true, every 2 seconds the getLocation method is invoked
     * every 6 seconds the HTTP client is invoked.
     */
    public static void main(String[] args) {
        try {
            Criteria gpsCrit = new Criteria();
            gpsCrit.setCostAllowed(false);
            gpsCrit.setSpeedAndCourseRequirement(true);

```

```
        test = LocationProvider.getInstance(gpsCrit);
    }
    catch (SecurityException securityException) {
        return;
    }
    try {
        //TEST CODE
        int count = 0;
        //END TEST CODE
        int i = 2;
        while(alarm) {
            location = test.getLocation(10); // try to retrieve location, time out is 10 seconds set status in
PERMANENTLY_UNAVAILABLE
            i++;
            if(i == 3) {
                client = new HTTPClient();
                client.start();
                i = 0;
            }
            Thread.sleep(2000);
            //TEST CODE
            count++;
            if(count == 11) {
                alarm = false;
            }
            //END TEST CODE
        }
    }
    catch(LocationException lex) {
        System.out.println("LocationException");
        return;
    }
    catch(NullPointerException exception) {
        System.out.println("NullPointerException");
        return;
    }
    catch(InterruptedException ie) {
        System.out.println("InterruptedException");
        return;
    }
    test.reset();
}
}
```

3 Package jspPresentation

3.1 Class JspPresentationBean

```
package jsppresentation;

import java.io.*;

public class JspPresentationBean {

    private String latitude = "unknown";
```

```
private String longitude = "unknown";
private String altitude = "unknown";
private String speed = "unknown";

//Access Latitude property
public String getLatitude () {
    try {
        FileReader fr = new FileReader("H:\\B-Project\\lat.txt");
        BufferedReader in = new BufferedReader(fr);
        String str;
        while ((str = in.readLine()) != null) {
            //System.out.print("Lat is: " + str);
            latitude = str;
        }
        in.close();
    } catch (IOException e) {
        System.out.println("IOException in getLatitude");
    }
    return latitude;
}

//Access Latitude property
public void setLatitude () {
}

//Access Longitude property
public String getLongitude () {
    try {
        FileReader fr = new FileReader("H:\\B-Project\\lon.txt");
        BufferedReader in = new BufferedReader(fr);
        String str;
        while ((str = in.readLine()) != null) {
            //System.out.print("Lon is: " + str);
            longitude = str;
        }
        in.close();
    } catch (IOException e) {
        System.out.println("IOException in getLongitude");
    }
    return longitude;
}

//Access longitude property
public void setlongitude () {
}

//Access altitude property
public String getAltitude () {
    try {
        FileReader fr = new FileReader("H:\\B-Project\\alt.txt");
        BufferedReader in = new BufferedReader(fr);
        String str;
        while ((str = in.readLine()) != null) {
            //System.out.print("Alt is: " + str);
            altitude = str;
        }
    }
}
```

```

        in.close();
    } catch (IOException e) {
        System.out.println("IOException in getAltitude");
    }
    return altitude;
}

//Access longitude property
public void setAltitude () {
}

//Access speed property
public String getSpeed () {
    try {
        FileReader fr = new FileReader("H:\\B-Project\\spe.txt");
        BufferedReader in = new BufferedReader(fr);
        String str;
        while ((str = in.readLine()) != null) {
            //System.out.print("Speed is: " + str);
            speed = str;
        }
        in.close();
    } catch (IOException e) {
        System.out.println("IOException in getSpeed");
    }
    return speed;
}

//Access longitude property
public void setSpeed () {
}
}

```

3.2 JSPPresentation

```

<html>
<head>
<title>JspPresentation</title>
<META HTTP-EQUIV="Refresh" CONTENT="5">
</head>

<jsp:useBean id="jspPresentationBeanId" scope="session" class="jsppresentation.JspPresentationBean"
/>
<jsp:setProperty name="jspPresentationBeanId" property="*" />

<body bgcolor="#ffffff">

<h1>
BAN Location Presentation site
</h1>

<form method="get">
<input type="hidden" name="latitude"> <br>
<input type="hidden" name="longitude"><br>
<input type="hidden" name="altitude"> <br>
<input type="hidden" name="speed"> <br>

```

```

<%--<input type = "submit" name= "submit" value = "Request Location of patient">--%>

    <p>
    <br>Value of latitude is :<jsp:getProperty name="jspPresentationBeanId" property="latitude" />
    <br>Value of longitude is :<jsp:getProperty name="jspPresentationBeanId" property="longitude"
/>
    <br>Value of altitude is :<jsp:getProperty name="jspPresentationBeanId" property="altitude" />
    <br>Value of speed is :<jsp:getProperty name="jspPresentationBeanId" property="speed" />
</form>

</body>
</html>

```

4 Package httpserver

4.1 Class JspConnectionBean

```

package httpserver;

import java.io.*;

public class JspConnectionBean {

    private String lat = "unknown";
    private String lon = "unknown";
    private String alt = "unknown";
    private String spe = "unknown";

    //Acces lat property
    public void setLat(String newLat) {
        try {
            lat = newLat;
            File file = new File("H:\\B-Project\\lat.txt");
            FileWriter fileWriter = new FileWriter( file );
            fileWriter.write( lat );
            fileWriter.close();
            System.out.println("\n      File textOutput written in setLat:" + lat);
        } catch (IOException e) {
            System.out.println( "IO error:" + e );
        }
    }

    public String getLat() {
        System.out.println("lat in Getlat(): " + lat);
        return lat;
    }

    //Acces lon property
    public String getLon() {
        return lon;
    }

    public void setLon(String newLon) {
        try {
            lon = newLon;
            File file = new File("H:\\B-Project\\lon.txt");

```

```

        FileWriter fileWriter = new FileWriter( file );
        fileWriter.write( lon );
        fileWriter.close();
        System.out.println("\n      File textOutput written in setLon:" + lon);
    } catch (IOException e) {
        System.out.println( "IO error:" + e );
    }
}

//Acces alt property
public String getalt() {
    return alt;
}

public void setAlt(String newAlt) {
    try {
        alt = newAlt;
        File file = new File("H:\\B-Project\\alt.txt");
        FileWriter fileWriter = new FileWriter( file );
        fileWriter.write( alt );
        fileWriter.close();
        System.out.println("\n      File textOutput written in setalt:" + alt);
    } catch (IOException e) {
        System.out.println( "IO error:" + e );
    }
}

//Acces spe property
public String getSpe() {
    return spe;
}

public void setSpe(String newSpe) {
    try {
        spe = newSpe;
        File file = new File("H:\\B-Project\\spe.txt");
        FileWriter fileWriter = new FileWriter( file );
        fileWriter.write( spe );
        fileWriter.close();
        System.out.println("\n      File textOutput written in setspe:" + spe);
    } catch (IOException e) {
        System.out.println( "IO error:" + e );
    }
}
}
}

```

4.2 JspConnention

```

<html>
<head>
<title>
    jspConnection
</title>
</head>

<jsp:useBean id="jspConnectionBeanId" scope="session" class="httpserver.JspConnectionBean" />

```

```
<jsp:setProperty name="jspConnectionBeanId" property="*" />

<body bgcolor="#ffffff">
  <h1>
    Connection to BAN
  </h1>

  <form method="post">

    <br>Enter lat : <input type = "text" name="lat"> <br>
    <br>Enter lon : <input type = "text" name="lon"> <br>
    <br>Enter alt : <input type = "text" name="alt"> <br>
    <br>Enter spe : <input type = "text" name="spe"> <br>

    <br><input type="submit" name="Submit" value="Submit">
    <p>
    Value of lat is :<jsp:getProperty name="jspConnectionBeanId" property="lat" /><br>
    Value of lon is :<jsp:getProperty name="jspConnectionBeanId" property="lon" /><br>
    Value of alt is :<jsp:getProperty name="jspConnectionBeanId" property="alt" /><br>
    Value of spe is :<jsp:getProperty name="jspConnectionBeanId" property="spe" />
  </form>
</body>
</html>
```