# Table of Contents

Chapter 1 - Introduction........................................................................................................... 2  
  1.1 Motivation......................................................................................................................... 2  
  1.2 Objectives......................................................................................................................... 2  
  1.3 Methodology...................................................................................................................... 2  
  1.4 Documentation.................................................................................................................. 3  
  1.5 Bluetooth.......................................................................................................................... 3  
Chapter 2 - PCWin32 Platform.................................................................................................. 8  
  2.2 Introduction....................................................................................................................... 8  
  2.3 Bluetooth adapters........................................................................................................... 8  
  2.4 Bluetooth libraries............................................................................................................ 9  
  2.5 Summary of Bluetooth Compatibilities............................................................................ 12  
  2.6 Bluetooth client-server application.................................................................................. 12  
  2.7 Serial communication...................................................................................................... 14  
Chapter 3 - Intel iMotes and Tiny OS....................................................................................... 16  
  3.1 Introduction....................................................................................................................... 16  
  3.2 Intel iMotes........................................................................................................................ 16  
  3.3 TinyOS............................................................................................................................... 17  
  3.4 Communications on iMotes.............................................................................................. 17  
  3.4.1 Serial............................................................................................................................ 17  
  3.4.2 Bluetooth.................................................................................................................... 20  
Conclusion.................................................................................................................................... 20  
Chapter 4 - Mobile platforms................................................................................................... 22  
  4.1 Introduction....................................................................................................................... 22  
  4.2 Mobile device.................................................................................................................... 22  
  4.3 Development platforms for Mobile Devices .................................................................... 23  
  4.4 References....................................................................................................................... 24  
Chapter 5 - Benchmark Applications....................................................................................... 25  
  5.1 Introduction....................................................................................................................... 25  
  5.2 Summary of tests performed............................................................................................ 25  
  5.3 Silent Presence control..................................................................................................... 25  
  5.4 Remote iMote monitor....................................................................................................... 29  
Chapter 6 - Conclusions.......................................................................................................... 33  
  6.1 Introduction....................................................................................................................... 33  
Chapter 7 - Project Management............................................................................................ 35  
  7.1 Introduction....................................................................................................................... 35  
Annexes...................................................................................................................................... 40  
  Bluetooth addresses list.......................................................................................................... 44  
  Getting started with aveLink BT SDK for Java........................................................................ 46  
  Getting started with Aventana............................................................................................... 52  
  Getting started with iMotes and TinyOS.................................................................................. 54  
  Getting started with Symbian............................................................................................... 64

1
Chapter 1 - Introduction

1.1 Motivation

Pervasive computation is the integration of the computer in the daily person life, in a way that computers are not perceived as differentiated objects. The objective is insert “intelligent devices” in the environment and in the objects that we can use every day, in order that the persons could interact with them in a natural way, in all situations and circumstances.

Those “intelligent devices” are special-purpose computer system designed to perform a dedicated function. Unlike a general-purpose computer, such as a personal computer, an embedded system performs one or a few pre-defined tasks, usually with very specific requirements, and often includes task-specific hardware and mechanical parts not usually found in a general-purpose computer. Since the system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product. Embedded systems are often mass-produced, benefiting from economies of scale.

This technology is implanted in reduced placed where a person can not access in, for reduced dimensions or for being a dangerous for human health. For example, if we were interested in measure an oven temperature in an industrial environment. Our “intelligent device” will be placed inside the oven and will send the temperature to a main computer using a wireless communication protocol. From the main computer it will be monitorized the temerature of the most critical parts of the factory.

From other side, there are Handheld devices (also known as handheld computer or simply handheld). A handheld is a pocket-sized computing device, typically using a small visual display screen for user output and a miniaturised keyboard for user input. Almost all those device provides mechanism for wireless communication.

In the example shown in the previous paragraphs the main computer is a “differentiated object” in the system. This contradicts, the definition of a pervasive system. But if we replace a main computer for one or several handhelds for monitoring the temperature. The human responblie for checking that all the system is working, will not need be in the same room than the main computer. He checks from its handheld and visually if the oven is working properly or not. Now the example matches with the definition of a pervasive system.

The motivation for realize this project is cover the necessity that exists on a pervasive system for controlling and retrieve information from its “intelligent devices” from any point of its influence area. For this PFC the “intelligent devices” will be an Intel iMote and the handheld will be a smart-phone.

1.2 Objectives

The objective of this PFC will be build a pervasive system. In order to do that, we will have to study which are the communications mechanism that are provided in the Intel iMote platform and in the smart-phone platform.
It will be added an extra PCWin32 platform, because there are tools that help us to monitorize the communications. The applications for the iMote and the smartphone are developed and downloaded from the PCWin32 platform. During the project it we developed several applications that will test the platforms communications capabilities and depending on the results obtained it will be designed the final pervasive system.

We will focus on interconnect the different platforms with bluetooth protocol. Because it is a wireless protocol, wider range (up to 100m.), low power consumption, it is integrated in almost all smartphones and in some Motes. So the first thing that will do, it is studying bluetooth, because this protocol has been designed for building pervasive systems and exists in the three platforms.

In the next section, it will be studied the behavior of Bluetooth protocol on the platform PCWin32. It will be analized the different libraries available on the market for this platform.

In the second chapter are presented the bluetooth devices and libraries to be tested. From this chapter we will conclude which library is compatible with which bluetooth device and the libraries limitations. For the final application we need to know which libraries and which bluetooth devices can be used. Due to the fact that the iMotes have a serial port it will be tested a serial communications library.

On the third chapter, it is presented the iMote and its operating system TinyOS. How to develop applications on this platform and will developed iMote applications for testing its communications capabilities (Serial and Bluetooth).

On the fourth, it will be shown application development for mobile devices on Mobile development platforms (J2ME and Symbian). It will be developed a test application for checking its communication capabilities (Bluetooth).

Finally on the fourth chapter, depending on the results taken on the three parts it will developed a sample application that will show up the interconnection among the three platforms.

1.3 Methodology

For knowing the interconnection possibilities that each platform offer us, we will study each platform independently.

For each platform we will take a look first to the physical interfaces that we have (Bluetooth, Serial, USB....). After that we will take a look at the different possibilities that Bluetooth protocol bring us on the focused platform studied. Depending on the results obtained it will be created a sample application that shows up the results obtained on the study.

With this methodology different interconnection possibilities raises. And it is more feasible to find the correct interconnection formula among the different platforms involved.

Once finalized the study on each platform and depending on the results obtained we will interconnect the different platforms.
All documentation related with the Configuration Management has been collected in the Annex because I consider that kind of documentation can distract from the objectives of the study. Nevertheless this documentation saves a lot of time in case that you were interested in install the tools in another computer or solve known problems when you work with the tools.

### 1.5 Bluetooth

Bluetooth is a wireless protocol that for its characteristics: wireless protocol, low power consumption, wide range (100 m) and provided in almost all the smartphones and in some Motes. Use bluetooth protocol will help us to integrate all different elements that are part of our pervasive system. In this section are explained the main tips of bluetooth protocol.

#### 1.5.1 History

Old Harald Bluetooth united Denmark and Norway. Bluetooth of today will unite the worlds of computers and telecommunications (hopefully longer than the few years Harald's Viking kingdom survived). In 1994 Ericsson Mobile Communications initiated a study to investigate the feasibility of a low-power low-cost radio interface between mobile phones and their accessories. In Feb 1998, five companies Ericsson, Nokia, IBM, Toshiba and Intel formed a Special Interest Group (SIG) [1]. The group contained the necessary business sector members - two market leaders in mobile telephony, two market leaders in laptop computing and a market leader in digital signal processing technology. It was estimated that before year 2002, Bluetooth will be a built-in feature for more than 100 million mobile phones and several million communication devices ranging from handsets and portable PCs to desktop computers and notebooks.

#### 1.5.2 Description

Bluetooth offers a wireless interconnection among different devices that has this technology, like mobiles, PDA, digital cameras, laptops, printers, or simpy any that any device that its manufacturer wants, using always a safe radio connection. The range that use to have those devices are 10 meters for saving energy, because almost all those devices use batteries. The range can be increased to 100 meters (very close to Wi-Fi) but increasing its consumption [3]. For improving the communication its desirable that no big physical object will interpose, for instance a wall.

The bandwidth depends on its kernel version:

<table>
<thead>
<tr>
<th>Version</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.1</td>
<td>723.1 Kbps</td>
</tr>
<tr>
<td>Version 1.2</td>
<td>1 Mbps</td>
</tr>
<tr>
<td>Version 2.0 + EDR:</td>
<td>2.1 ~ 3 Mbps</td>
</tr>
</tbody>
</table>

*Table 1: Bluetooth version and its bandwidth*
The transmission power is divided in 3 categories:

<table>
<thead>
<tr>
<th>Class</th>
<th>Maximum Permitted Power (mW)</th>
<th>Maximum Permitted Power (dBm)</th>
<th>Range (approximate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>100 mW</td>
<td>20 dBm</td>
<td>~100 meters</td>
</tr>
<tr>
<td>Class 2</td>
<td>2.5 mW</td>
<td>4 dBm</td>
<td>~10 meters</td>
</tr>
<tr>
<td>Class 3</td>
<td>1 mW</td>
<td>0 dBm</td>
<td>~1 meters</td>
</tr>
</tbody>
</table>

Table 2: Bluetooth classes

1.5.3 Topology

The big difference with other wireless nets, like IEEE 802.11 (Wi-Fi), designed for devices that are inside a building or in its nearabout; is that those devices that use wireless PANs IEEE 802.15, including Bluetooth, will be able to communicate in any point of the planet, even inside a plane or ship without the help of any access point.

![Illustration 3: Piconet with one slave (P2), with several (P3) and scatternet (P1+P2+P3)](image)

The bluetooth architecture is organized in piconets, that are two or more devices sharing one channel; one of the acts as master, and the rest as slave. A scatternet is made of several piconets.

1.5.4 Bluetooth stack

The main principle in mind when developing the Bluetooth Protocol Architecture has been the maximization and the re-use of existing protocols for different purposes at the higher layers. The one main advantage is that existing (legacy) applications can be adapted to work with the Bluetooth Technology. The Bluetooth Protocol Architecture also allows for the use of commonly used application protocols on top of the Bluetooth-Specific protocols. In simpler terms, this permits new applications to take full advantage of the capabilities of the Bluetooth technology and for many applications that are already developed by vendors; they can take immediate advantage of hardware and software systems, which are also compliant with the Specification [2].
1.5.4.1 HCI

The HCI provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. Essentially this interface provides a uniform method of accessing the Bluetooth baseband capabilities. The HCI exists across 3 sections, the Host - Transport Layer - Host Controller. Each of the sections has a different role to play in the HCI system.

1.5.4.2 SDP (Service Discovery Protocol) protocol

The Service Discovery Protocol (SDP) provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

SDP involves communication between a SDP server and a SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client may retrieve information from a service record maintained by the SDP server by issuing a SDP request. If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes, but it does not provide a mechanism for utilizing those services.

Normally, a SDP client searches for services based on some desired characteristics of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any a priori information about the services. This process of looking for any offered services is called browsing.

1.5.4.3 SPP (Serial Port Protocol) protocol

The Serial Port Profile defines the requirements for Bluetooth devices necessary for setting up emulated serial cable connections using RFCOMM between two peer devices. The requirements are expressed in terms of services provided to applications, and by defining the features and procedures that are required for interoperability between Bluetooth devices.

Essentially, the Serial Port Profile defines the protocols and procedures that shall be used by devices using Bluetooth for RS232 (or similar) serial cable emulation. The scenario covered by this profile deals with legacy
applications using Bluetooth as a cable replacement, through a virtual serial port abstraction (which in itself is operating system-dependent).

### 1.5.4.4 L2CAP protocol

Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability and segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

L2CAP is based around the concept of *channels*. Channel is a logical connection on top of baseband connection. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol. Multiple channels can share the same baseband connection.

### 1.5.5 Bluetooth API

From the HCI and lower layers are implemented by the bluetooth hardware manufacturer. For dealing with upper HCI layers there are two big platforms Symbian and J2ME (Sun Microsystems api's). For Developing applications for Symbian and J2ME platform you can see in the annex of this documentation two articles that talks about it.

Depending on the abstraction level that the application works with with the data transferred and the need for controlling the communications protocol it will be more adequated one or another layer. For controlling the communications is more adequated HCI, and for sending bytes like in a serial line is would be RFCOMM.

Symbian api's only are implemented on mobile devices that use Symbian OS. With Symbian you can work with OBEX, SDP, RFCOMM, L2CAP and HCI layers.

If J2ME supports JSR082 then it will be possible to deal with the bluetooth APIs. JSR082 can deal with SDP, L2CAP and OBEX. JSR82 is in all mobiles that have bluetooth and works with J2ME platform, but for J2SE (laptops and desktops) is not included in the platform and its needed an extra library for working with. On the following table there are some of the libraries avialable for J2ME platform.

<table>
<thead>
<tr>
<th>Library</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlueCove</td>
<td>LGPL</td>
</tr>
<tr>
<td>Atinav</td>
<td>Trial 15 days</td>
</tr>
<tr>
<td>Aventana</td>
<td>Trial 15 days</td>
</tr>
</tbody>
</table>

*Table 3: Bluetooth libraries tested*

### 1.5.5.1 JSR082 – Bluetooth library for Java

The overall goal of this specification is to define a standard set of APIs that will enable an open, thirdparty application development environment for Bluetooth wireless technology. The API is targeted mainly at devices that are limited in processing power and memory, and are primarily battery-operated.
The main advantage of using a .jar library that implements jsr082 is that the source code that uses this library is able to work in another operating systems (MAC, Linux, Windows) and even in another hardware architectures mobile devices. In other implementations we depend on the operating system [3] or the hardware architecture [4].

As we are going to use a mobile J2ME compatible with bluetooth it will be very usefull to use this library, in order to use application with different platforms.

1.6 References

Chapter 2 - PCWin32 Platform

2.1 Motivation

We are going to study the Bluetooth and Serial communication protocol on PCWin32 because this platform will be the responsible for interconnecting indirectly the mobile platform and iMote platform.

2.2 Introduction

In this chapter we will study the communications mechanism existing for PCWin32 platform. Mainly focusing on Bluetooth, but serial protocol is shown as well.

It will presented the bluetooth adapters that have been used for testing the bluetooth communication in the PCWin32 platform. For each one I present its most significant features. On the following section are tested three libraries that works with bluetooth protocol and one library that works with the serial line. On the test performed we will see the compatibilities with the bluetooth adapters.

From the resulting test we will see the matches between libraries and communication devices and what kind of protocol we can work with.

2.3 Bluetooth adapters

The use of several Bluetooth adapters is due to the fact that each jsr82 bluetooth library implementation works with a concrete Bluetooth adaptor. We will use more than one adaptor for running client-server applications within the same laptop.

<table>
<thead>
<tr>
<th>Picture</th>
<th>Commercial name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Toshiba PA3455U - 1BTM" /></td>
<td>Toshiba PA3455U - 1BTM</td>
<td>Bluetooth version 2.0, Class II, Protocols SDP, SPP, DUN, FAX, LAP, OPP, FTP, HID, HCRP, PAN, BIP, HSP, A2DP, AVRCP, GAVDP</td>
</tr>
<tr>
<td><img src="image2.png" alt="Trust BT-220TP" /></td>
<td>Trust BT-220TP</td>
<td>Bluetooth version 2.0, Class II, Protocols SDP, SPP, DUN, FAX, LAP, OPP, FTP, HID, HCRP, PAN, BIP, HSP, A2DP, AVRCP, GAVDP</td>
</tr>
</tbody>
</table>
2.4 Bluetooth libraries

JSR82 is in all mobiles that have bluetooth and works with J2ME platform, but for J2SE (laptops and desktops) is not included in the platform and its needed an extra library for working with. On the following table there are some of the libraries available for J2ME platform.

<table>
<thead>
<tr>
<th>Name</th>
<th>Protocols</th>
<th>Platform</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aventana</td>
<td>SDP,SPP,L2CAP</td>
<td>PCWin32</td>
<td>14-Ev. days</td>
</tr>
<tr>
<td>Atinav</td>
<td>SDP,SPP,L2CAP</td>
<td>PCWin32</td>
<td>14-Ev. days</td>
</tr>
<tr>
<td>Bluecove</td>
<td>SDP,SPP</td>
<td>PCWin32</td>
<td>GPL</td>
</tr>
<tr>
<td>BlueZ</td>
<td>SDP,SPP,L2CAP</td>
<td>PCLinux</td>
<td>GPL</td>
</tr>
<tr>
<td>Rococo</td>
<td>SDP,SPP,L2CAP</td>
<td>PCWin32</td>
<td></td>
</tr>
</tbody>
</table>

In the previous chapter were presented some libraries that implemented jsr082, so in this chapter we are going to install and evaluate them. The bluetooth addresses from the devices used in this test are collected in the article Bluetooth address list placed in the annex.

2.4.1 Test methodology

The bluetooth libraries choosen for testing have been Aventana, Atinav and Bluecove. Those libraries is an implementation of jsr082 for PCWin32 platform. So the test will be the same for the three libraries because the libraries implements a common interface (jsr082).The test is a java application that uses SDP protocol. It has been created three different projects with the common test source file.

2.4.1 aveLink BT SDK for Java

The article Getting started with aveLink BT SDK for Java placed on the Annex of this documentation explains how to install the library and how to create and build a project that use AveLink library. At the end of the article are described the problems that I have found for working with this library and how I solve them.
aveLink BT SDK for Java (Atinav) is a library that implements jsr082 for java applications, J2SE and J2ME as well. This library it is 15-day evaluation, and has no limitation on the bluetooth adaptors to work with. You can download this library from [5], but first you have to register.

As I mentioned before, each library works with a specific bluetooth adapter. In this case the adapter that works is Toshiba PA3455U – 1BTM. Take in count that the installation of this library cancels the adaptor vendor drivers and software, so the software that originally worked with the adaptor stops working.

2.4.1.1 Testing the library

In order to check that the library is properly installed and working, it has been developed a simple application that uses the library.

The sample application will be used for testing the each jsr082 library implementation. For loading the application all that you have to do is import the project C:\projects\final\TestLibrary.Atinav from Eclipse [6].

For import the project open Eclipse IDE, and on Menu File|Import|Existing projects into workspace, Select root directory: C:\projects\final\TestLibrary.Atinav and Finish.

And, execute the project:

If all is configured properly you will see how are being detected all the devices that are in the nearabouts. If have some problems take a look at the article Getting started with aveLink BT SDK for Java placed in the annex.

Illustration 5: Click the "play" button for launching the application

Illustration 6: The console window reports the bluetooth devices found in the surrounding area wit Atinav library.
On the top of the screen you can read the name and library version, the bluetooth address followed by the bluetooth alias name (in this case is the library name) and some communication parameters. Afterwards the application begins to inquiry for nearabout bluetooth devices and finds the EZURIO pcmcia bluetooth adapter.

### 2.4.2 Aventana library

The article *Getting started with Aventana* placed on the Annex of this documentation explains how to install the library and how to create and build a project that use Aventana library. At the end of the article are described the problems that I have found for working with this library and how I solve them.

*Aventa* is another library that implements jsr082. This library it is 15-day evaluation, and has limitation on the bluetooth adaptors to work with, in this case is 3. You have to say which bluetooth addresses you will work with when you request for the library [8].

In this case the adaptor that works is **Trust BT-220TP** and **EZURIO pcmcia bluetooth**. This library works without problems with the adapter vendor drivers and software.

If all is configured properly you will see how are being detected all the devices that are in the surrounding area. If have some problems take a look at the article *Getting started with Aventana* placed in the annex.

#### 2.4.2.1 Testing the library

In order to check that the library is properly installed and working, it has been developed a simple application that uses the library. The application is the same that with the previous library in order to check the library compatibility. Proceed as the previous library to import and execute the project.

On the top of the screen you can read the name and library version, the bluetooth address followed by the bluetooth alias name (in this case is the machine name) and some communication parameters. Afterwards the application begins to inquiry for nearabout bluetooth devices and finds the Toshiba PA3455U – 1BTM. It is quite recommended check this log with the one obtained wint the AveLink library.
2.4.3 BlueCove

BlueCove is a LGPL licensed JSR-82 implementation that currently interfaces with the Microsoft Bluetooth stack. Originally developed by Intel Research and currently maintained by volunteers. You can download it from [9].

Due to the Microsoft Bluetooth stack only supporting SPP connections, BlueCove also only supports SPP connections. The operating system supported is currently limited to Windows XP SP2 and newer, because the Microsoft Bluetooth stack is not available on other operating systems. If someone writes code to support another stack and/or operating system, it will be considered for inclusion. BlueCove does also not support OBEX, but there are other projects that can (possibly) be used to achieve OBEX functionality with BlueCove.

2.4.3.1 Installation

Once downloaded from [9] place bluecove.jar in your class path and do not forget copy intelbth.dll into \c:\windows\system32.

In order to check that the library is properly installed and working, it has been developed a simple application that uses the library. The application is the same that with the previous library in order to check the library compatibility. Proceed as the aventana library to import and execute the project.

![Illustration](image.png)

*Illustration 8: No bluetooth device is detected with this implementation of BlueCove library.*

In the log we can see that the library does not found any bluetooth device to work with and the test for detecting the nearabout bluetooth devices fails.

2.5 Summary of Bluetooth Compatibilities

This table summarizes the compatibility between the different libraries implementations and the bluetooth adaptors used
### 2.6 Bluetooth client-server application

Up to this point we know the compatibilities among the bluetooth libraries and bluetooth adapters. So we are able to develop any client-server applications.

#### 2.6.1 The application

The purpose of this is to interconnect two applications through SPP bluetooth protocol.

![Bluetooth connection diagram](image)

**Requirements**

The PCWin32 application will:

- The server will be on for offering some services and waiting for customer connections. Once is established the connection with a customer, the server sends a message to the customer and retrieves the message received from the customer.
- The Customer will look up in the surrounding area any bluetooth device that offers one specific service. Once is found, the customer sends a message to the server and retrieves the message received from the server.

#### 2.6.2 Build and loading

The Server application is using Atinav and Toshiba bluetooth adaptor. And the customer is using Aventana with PCMCIA bluetooth adaptor. The customer detects the server, makes a connection and sends/receives data.

From eclipse, import the projects C:\projects\final\ClientSPP.Aventana and C:\projects\final\ServerSPP.Atinav and run the projects.

**Client console:**

<table>
<thead>
<tr>
<th></th>
<th>Atinav</th>
<th>Aventana</th>
<th>BlueCove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba PA3455U</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Trust BT-220TP</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Ezurio PCMCIA - BT</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*Table 4: Bluetooth hardware and libraries compatibility*
The client console window reports, among other information, the services provided by the server: Service IV, Service I, Service II and Service III. The server URL is presented as well. Afterwards the connection between the server and the customer is performed, the data received from the server is “I’m the Server”.

**Server console**

The server reports that a connection has been successful and print out the data received from the Customer.

The server console window reports its URL and waits for customer connection. Once the connection is performed, the data received from the client is “I’m the customer”.

Illustration 9: The Customer console window

Illustration 10: The Server console window
2.7 Serial communication

Even that the focus of this project is study the baggage protocol. It is interesting work with other known protocols in the case that the baggage communication could not be achieved.

In telecommunications, **RS-232** is a standard for serial binary data interconnection between a *DTE* (Data terminal equipment) and a *DCE* (Data Circuit-terminating Equipment). It is a wired protocol.

2.7.1 Libraries

The most adequate for working are *javacomm* [10] because are implemented in java. So it will be more adequate in case that an application have to work also with a jsr82 baggage library and.

The Java Communications 3.0 API is a Java extension that facilitates developing platform-independent communications applications for technologies such as Smart Cards, embedded systems, and point-of-sale devices, financial services devices, fax, modems, display terminals, and robotic equipment.

You can download the libraries from [http://www.fh-bochum.de/fb3/meva-lab/tools/](http://www.fh-bochum.de/fb3/meva-lab/tools/). For installing it, uncompress it and leave the .jar library into the .lib folder of your current project and copy win32com.dll into C:\windows\system32.

In the following chapter it is developed a sample application that communicates via serial port PCWin32 with iMote platform.

2.8 References

Chapter 3 - Intel iMotes and Tiny OS

3.1 Introduction

In this chapter we will be presented the iMotes hardware and its operating system TinyOS. Afterwards it will be studied the different ways that we have for communicating with the iMotes. It will developed sample applications for test its communication capabilities. The configuration management for working with the iMotes read the article *Getting started with iMotes and TinyOS* included in the annex.

3.2 Intel iMotes

The Intel Mote [2], [3], [4] is an enhanced sensor network node platform that is targeted for a variety of research and commercial applications. The focus of this platform development is to build a small, modular unit that can easily be customized for a variety of usages by snap-on expansion boards. The operating system software is based on TinyOS, which is a component based operating system specifically designed for sensor network applications by the University of California in Berkeley and the attached Intel Research Lab.

The Intel Mote provides enhanced computation and communication resources that allow more demanding sensor network applications to be supported. In particular, the use of the Bluetooth* radio protocol provides increased link reliability and security options. Bluetooth also provides a synchronized TDMA access scheme which allows the sender and receiver of a communications link to be powered down in-between active slots, thereby reducing the average power consumption.
3.3 TinyOS

TinyOS[1] is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS’s component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. TinyOS’s event-driven execution model enables fine-grained power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.

3.4 Comunications on iMotes

The intel iMotes have two interfaces for communicating: Bluetooth and Serial (RS232). In the following sections it will be developed applications for testing the iMote communications capabilities.

3.4.1 Serial

The serial communication is used by the iMote for transferring TinyOS applications into the iMote. It can be used as well for transferring data to any TinyOS application. In the following section is presented a sample TinyOS that reads and writes from the Serial line.

3.4.1.1 EchoSerial test (iMote side)

The purpose of this test the serial communication with the iMote, in order to see what is going on in and
send some to the commands to the imote. The root imote will be linked to the PCWin32 through a serial port (simulated by a usb connection).

The iMote application will:

- Echo any character that will be sent by the serial port, in order, to check that serial communication works.
- Each 10 secs. Will send a message by the serial port, in order, to see that the iMote is alive.

**Build and loading**

Open a cygwin window and from `/opt/tinyos-1.x/contrib/imote/apps/EchoSerial` folder type make install imote, and follow the window instructions.

Open Hyperterminal, create a new connection with the following characteristics:

![Hyperterminal configuration](Illustration_16: Hyperterminal configuration)

Click on call button, reset the iMote and wait for the responses...

**3.4.1.2 EchoSerial test (PCWin32 side)**

The purpose of this is to read the serial port and send some data. In one iMote will be loaded the EchoSerial application and from EchoSerial test we will retrieve all the information sent by the iMote and we will be able to send data to the iMote.

The iMote application will:

- Echo any character that will receive from the serial port
- Send its internal counter value.
Build and executing

From eclipse import the project from C:\projects\final\SerialEcho\lib and run the projects. The iMote echoes its internal counter through the Serial port.

Conclusions

From a java application we are able to read and write from the serial port.

In the hyperterminal log it is presented the iMote internal counter and echoed the characters from the keys pressed.

3.4.2 Bluetooth

The iMote has a bluetooth device for communicating with other iMotes and perhaps another bluetooth
devices. In the following sections will be implemented TinyOS applications that will test several parts of the bluetooth functionality.

### 3.4.2.1 OneTimeInquiry test

The purpose of this test the bluetooth inquiry functionality. Look up other bluetooth devices.

The iMote application will:
- Will detect any bluetooth device on the surrounding area.
- Once detected will print out its bluetooth address.

**Build and loading**

Open a cygwin window and from `/opt/tinyos-1.x/contrib/imote/apps/OneTimeInquiry` folder and type `make install imote`, and follow the window instructions.

**Conclusion**

Only had been detected the iMotes, the other bluetooth devices not.

### 3.4.2.2 Connection test

The porpouse of this test the bluetooth communication among the iMotes

The iMote application will:
- Will detect any bluetooth device on the surrounding area.
- Once root detects another iMote, sends a message.
- The iMote that receives the incomming message will show a green ligth.

**Build and loading**

Open a cygwin window and from `/opt/tinyos-1.x/contrib/imote/apps/Connection` folder and type `make install imote`, and follow the window instructions.

**Conclusion**

The root imote detects the remote iMote, establishes connection, sends a message, but the remote iMote does not receive the message.
3.4.2.3 PingPong demo

The purpose of this demo is to know if the bluetooth communication works. This demo is provided in the iMotes development kit. The main objective is check if bluetooth communication really works.

The code has been updated for running on iMote 86505 and 86460.

Build and loading

Open a cygwin window and from /opt/tinyos-1.x/contrib/imote/apps/Connection folder and type make install imote, and follow the window instructions.

Conclusion

The master iMote when detects the slave one sends a message, but the slave does not receive it.

3.5 References

Chapter 4 - Mobile platforms

4.1 Introduction

In this chapter we will see the hardware that we will use for developing this project. Afterwards we will see the available platforms for developing applications for this terminal. And it will be developed a sample application on the chosen application.

4.2 Mobile device

The mobile device use has been the Nokia 6630. The characteristics [1] in which we are interested are:

Operating System:
Symbian OS v8.0a

Developer Platform:
S60 2nd Edition, Feature Pack 2

Java Technology:
FileConnection and PIM API (JSR-75)
Nokia UI API
CLDC 1.1
Bluetooth API (JSR-82 No OBEX)
Mobile 3D Graphics API (JSR-184)
MIDP 2.0
Mobile Media API (JSR-135)
Wireless Messaging API (JSR-120)

Any mobile that fulfills (or improve) those characteristics can be used as well. Take a look that Bluetooth
with Obex protocol can not be used, but for the purposes of this project it does not matter.

4.3 Development platforms for Mobile Devices

The most known available platforms for mobile devices are Symbian [2] and J2ME [3]. Symbian it is a platform that only works on Mobile devices, its programming language it is C++ with several limitations. Supports bluetooth programming for OBEX, SDP, RFCOMM, L2CAP and HCI layers. Symbian application has all the advantages and disadvantages as native applications.

J2ME if implements JSR082 can also work with bluetooth. The main advantage is that any java application that works for J2ME can work with J2ME. Its programming language is Java. If J2M2 supports JSR082, then supports bluetooth programming for SDP, SPP L2CAP and OBEX.

Keep in mind as well that both platforms can exists on the same mobile its architecture is the following.

For developing Midlets and Symbian applications take look the articles: Getting started with J2ME and Getting started with Symbian both placed on the annex.

4.3.1 Symbian and J2ME

For developing a bluetooth application is not mandatory work with one specific platform. Nevertheless J2ME has great advantages. As we saw in the previous chapter we can develop J2SE applications that works with bluetooth protocol. So we can build (and debug) an application with J2SE.

4.3.2 The J2ME Application

The aim of this application is to establish a bluetooth communication between a J2ME mobile and a PCWin32 laptop. In this Client-Server application the PCWin32 will act as server and the J2ME will act as client.

The PCWin32 client will:
- Will connect with the J2ME mobile server
- Send/receive data
The J2Me server will:

- Wait for customer connection.
- Send/receive data

Build and loading

From eclipse import the project from C:\projects\build\ClientSPP.j2me and run the project.

Conclusions

The client detects the J2ME mobile establishes the connection, sends the data “I’m the server”. But no data is received from the customer and from the mobile side the connection is performed but no-data is received.

4.4 References

Chapter 5 - Benchmark Applications

5.1 Introduction

From the results obtained on each platform and following the methodology presented in the first chapter we will develop two sample applications that will interconnect the three platforms.

5.2 Summary of tests performed

In the following table are summarized the test results performed on the different platforms.

<table>
<thead>
<tr>
<th>Test description</th>
<th>Platform</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP (avLink-Toshiba)</td>
<td>PCWin32</td>
<td>Successful</td>
</tr>
<tr>
<td>SDP (Atinav-Ezurio)</td>
<td>PCWin32</td>
<td>Successful</td>
</tr>
<tr>
<td>SDP (BlueCove-Ezurio)</td>
<td>PCWin32</td>
<td>Failed</td>
</tr>
<tr>
<td>Serial Communication</td>
<td>PCWin32 / iMote</td>
<td>Successful</td>
</tr>
<tr>
<td>SPP Client-Server Communication (Avlink-Toshiba and Atinav-Ezurio)</td>
<td>PCWin32 / PCWin32</td>
<td>Successful</td>
</tr>
<tr>
<td>SDP</td>
<td>iMote</td>
<td>Limited</td>
</tr>
<tr>
<td>PingPong demo</td>
<td>iMote</td>
<td>Failed</td>
</tr>
<tr>
<td>SDP</td>
<td>J2ME</td>
<td>Successful</td>
</tr>
<tr>
<td>SPP Client-Server (Atinav-Ezurio and Nokia 6630)</td>
<td>PCWin32/J2ME</td>
<td>Failed</td>
</tr>
</tbody>
</table>

Table 5: Summary of tests performed

From those results I will present two final applications: Remote iMote Monitor that will manage information that is inside the iMote and Silence Presence Control that is a bluetooth scan that detects all bluetooth devices on the surrounding area.

5.3 Silent Presence control

The first final application developed will be a silence presence control. The aim of this pervasive application is scan for all the surrounding bluetooth devices and check if this bluetooth device is allowed to be in the area.

One application could be for checking the access in a restricted room. All staff from the company will carry a passive bluetooth device. An iMote will be placed on a wall and will be visible from each point of the room. The authorized people when will be inside the room will know by the iMote led color if there is someone missing, or someone not authorized inside the room or if all the authorized staff is in the restricted area.

If all the authorized staff is inside the room, everybody, will see that the iMote led is set to green, and they will understand all the authorized staff is in the room.

If there is one person that is missing the iMote led is set on yellow, and everybody will understand that there is someone missing. But if the iMote led is set on red, it will be understood that there is someone not
allowed to be inside the room.

Another utility for this application is check that all the bluetooth devices which I will work with are in the surrounding area and do not exist other foreign bluetooth devices that could disturb. In the conclusions chapter in the section of extensions are explained several applications that could be done based in Silence Presence Control.

The algorithm that Silence Presence is the one exposed in the following flow chart.
The application will be detecting all the time the surrounding bluetooth devices and if there is one device that it is not in its database of known bluetooth devices, it will light up the iMote red led. If there is one device that is not in the surrounding area and its on the customer database; the server will light up the iMote yellow led. If there are all the devices that are on the customer database in the surrounding area, the server will light up the iMote green light.

5.3.2 The GUI

The application GUI consists in a tool bar with two buttons for start scanning and close the application. In the middle form there are two list, the left list shows the bluetooth address and alias of the bluetooth devices found in the surrounding area. On the right list appears all the bluetooth devices to be found. On the bottom area appears a log window that is showing in all the moment what is going on.
5.3.3 Execution Processing

Now are described the three possible scenarios that can occur during the application execution:

Scenario I

When Silence Presence control starts up, it orders to the iMote to light on the yellow led. And Begins to scan on the surrounding area for known bluetooth devices. While not all known devices have been found the iMote led does not change color.

Scenario II

But if during the inquiring phase (scenario I) is detected a bluetooth device that is not registered in the Silence Presence Control database, it is ordered to the iMot to light on the red led.
Scenario III

When all known devices have been found the green iMote led is light on. On the right list appears the bluetooth addresses of the devices to be found, and on the other side appears the list of bluetooth addresses to be found.

Illustration 27: Scenario II, One unexpected bluetooth device has been found.

Illustration 28: All devices have been found and the green light is on.
5.4 Remote iMote monitor

The second final application developed will be an iMote monitor. The aim of this application is retrieve information from the iMotes that controls a pervasive system and reconfigure them for a new behavior.

If our pervasive system is placed in a building plant and there is an iMote that controls each room. An administrator, with the iMote Monitor loaded in his computer, will retrieve information and parametrize the behavior from the iMotes.

5.4.1 Application architecture

Due to the limitations that we have found for exchanging information with the iMotes via bluetooth protocol, and the success for connecting two PCWin32 computers via bluetooth protocol. What we have done is attach the iMote via Serial Interface (COM2) to a PCWin32 computer for solving the limitations that the iMote has with bluetooth. The iMote Monitor server side will be placed on the PCWin32 and will play the server role and will offer the services that provides its attached iMote. The imote and the PCWin32 computer will be a unique element.

From the other side, the administrator will carry another PCWin32 computer with the iMote Monitor client application, and will ask (via bluetooth) for the services that supplied by the iMote. But, what happens if we have several iMotes scattered in a closest area? The first thing that the iMote Monitor client will do is detect the iMote to control.

The services offered by the PCWin32 computer attached to the iMote are change the iMote light color to
green, magenta or red. And Retrieve the internal counter value.

The following flow chart pict the flow diagram that follows the iMote Monitor Server (left) and the iMote Monitor Client (right).

5.4.2 The GUI's

The customer GUI has a tool bar for start and close application. The Bluetooth server frame shows the server URL where the service holds on and what kind of services are offered by the server.

Illustration 31: Application server (left) and customer (right) flow diagrams

Illustration 32: iMote Monitor client GUI
On Bluetooth key device if the text box prints out the key device bluetooth address. If the textbox background is red, the key device has not been found and the application can not continue. If the textbox is green the key device has been found and the customer can inquiry for the server services.

On the iMote frame appears the last iMote counter value and the last iMote color requested to the server. The progress bar if for showing the user that the application is alive. At the bottom frame is shown all logging information.

The server GUI has a toolbar for start and close the application. On the iMote frame is shown the last iMote counter value. The progress bar is for showing that the application is alive. The bottom frame shows all logging information.

### 5.4.2 Execution Processing

The left window is the Server GUI, on the bottom of the window appears the communications log and the requests that receives. The right window is the Client GUI, it shows the bluetooth addres from the bluetooth key device; the URL and the Services that are offered by the Server. Finally, the color that is requesting, in
this case is yellow.

Illustration 35: Scenario I. The customer request to turn on the green light.

Now, the customer ask for light on the green light.

Illustration 36: Scenario III. The customer request to turn on the red light.

And finally the color requested is red.
Chapter 6 - Conclusions

6.1 Introduction

In this chapter I describe the conclusions of this projects. The conclusions are divided in two main blocks the methodology used during the project, and the technologies used.

6.2 Methodology

It is a great methodology perform several test on the different platforms that checks the different issues that are related with the PFC. Depending on the results, you can define with more precision the application/s that will represent the PFC. Another great advantage is that you can detect wrong developments before starting the final application.

A great practice is documenting while the Configuration Management tasks are performed and make regular backups from the system.

6.3 Technologies

6.3.1 Intel iMotes and TinyOs

There is a lot of information about TinyOS. But regarding Intel iMotes there is not information at all, only a few presentation documents that introduce you on this kind of motes.

Regarding to the Configuration Management is very unestable for several reasons: It is very complicated uninstall it, the toolchain linker is under license and the build process is under cygwin.

6.3.2 Mobile platform

On the web there are a lot of well written documentation that talks about configuration management and sample applications for J2ME and Symbian.

The election of the mobile to use is very important, because the way that you download the applications into the terminal is quite different from one terminal to another.

6.3.3 PCWin32

The main difficulty is make the matching between the bluetooth physical interface and the java jsr82 library that deals with.

Another other limitation is that there is only one Open Source library for win32 and it is quite limited. There are other implementations that are under 15-day evaluation.
6.4 Future extensions

With a top-down methodology and all the battery of tests performed you have a clear picture from the state of each technologies. From my point of view the coverage of the project is quite correct, nevertheless some of the test were not successful due to the fact that the iMotes hardware provided were a prototipe and some of jsr82 libraries were a beta version.

As a extension for obtaining a realiable communiaction between the different platforms I will propose repeat the same study but with other TinyOS motes and with other stable libraries for pcwin32 that implements jsr082. If with those changes the test were successful, I would propose tho remove PCWin32 platform and communicate directly J2ME platform with iMote-TinyOS platform.

From Silence Presence Control can be created derivated application that I call “Preference Traffic Ligth”. For this application we will install a bluetooth scaner in a traffic light and the iMotes, or other bluetooth device will be installed in public transport vehicles. When the public transport vehicle approaches the traffic light will detect its presence and will try to let him pass.
Chapter 7 - Project Management

7.1 Introduction

In this chapter are described the time distribution during the project, the costs of the project and the conclusions to take in count if in future you want to use the same technology.

7.2 Planning

For this project the accounting will be in days, we will consider that a journey (1 day) has 8 working hours. This fact is quite important when the project is performed in (or with) other countries. E.g. In some french and german companies the 1 journey are 7 hours.

It is also very convenient keep in mind the staff dedication. In the case of this project we had those non-working days due for reasons like: personal holidays, bank holidays and other projects dedication.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Ini. Date</th>
<th>End. Date</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holidays</td>
<td>12.8.06</td>
<td>27.8.06</td>
<td>10</td>
</tr>
<tr>
<td>Bank day</td>
<td>11.9.06</td>
<td>11.9.06</td>
<td>1</td>
</tr>
<tr>
<td>Bank day</td>
<td>12.10.06</td>
<td>12.10.06</td>
<td>1</td>
</tr>
<tr>
<td>Bank day</td>
<td>1.11.06</td>
<td>1.11.06</td>
<td>1</td>
</tr>
<tr>
<td>Christmas holidays</td>
<td>23.12.06</td>
<td>1.1.07</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 6: Non-working days summary

On the following Gant chart is shown the project plannification based on time.
7.3 Roles

There are three roles on this project: Project Leader, Software Designer and Software programmer. Depending on the complexity of the project those roles can be performed for only one person, or a person can perform two roles, or even several people can deal with a unique role. For this project the all the roles will be collected in one person.

The Project Leader role will be the contact person who will collect all the requirements related with the cost in time and in money from the customer. It will control that the project is on time and in the budget.

Software designer will be responsible for collecting all technological requirements from the customer. He will organize the structure of the software and will decide which tests perform in each platform.

Software programmer is the person who will perform all the configuration management, will implement the test and the final applications.

In the following table are described the cost of each role. The of each role have been extrapolated from [1].

<table>
<thead>
<tr>
<th>Role</th>
<th>Price (hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Leader</td>
<td>100 €</td>
</tr>
<tr>
<td>Software Designer</td>
<td>80 €</td>
</tr>
<tr>
<td>Software Programmer</td>
<td>65 €</td>
</tr>
</tbody>
</table>

*Table 7: Role costs*

7.3 Cost

In the following table shows the time in days and in PM (person-month) spent in each task.
From the cost in days of each role we can solve out the human resource cost.

<table>
<thead>
<tr>
<th>Role</th>
<th>Cost day (€)</th>
<th>Days (Days)</th>
<th>Total (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Leader</td>
<td>800</td>
<td>2</td>
<td>1.600</td>
</tr>
<tr>
<td>Software Designer</td>
<td>640</td>
<td>6</td>
<td>3.840</td>
</tr>
<tr>
<td>Software Programmer</td>
<td>520</td>
<td>83</td>
<td>43.160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>48.600</strong></td>
</tr>
</tbody>
</table>

**Table 8: Staff cost**
7.3.2 Material

The following tables shows the cost of the resources employed in this project.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop Toshiba</td>
<td>830</td>
</tr>
<tr>
<td>Intel iMotes</td>
<td>3.000</td>
</tr>
<tr>
<td>Nokia 6630</td>
<td>150</td>
</tr>
<tr>
<td>Ezurio PCMCIA bluetooth device</td>
<td>45</td>
</tr>
<tr>
<td>Toshiba USB Bluetooth device</td>
<td>35</td>
</tr>
<tr>
<td>Trust USB Bluetooth device</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atinav Bluetooth library</td>
<td>650</td>
</tr>
<tr>
<td>Aventa Bluetooth library</td>
<td>350</td>
</tr>
<tr>
<td>BlueCove Bluetooth library</td>
<td>LGPL</td>
</tr>
<tr>
<td></td>
<td>5.090</td>
</tr>
</tbody>
</table>

Table 9: Economic cost

7.5 Summary

The total costs of the project is:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff</td>
<td>48.600</td>
</tr>
<tr>
<td>Material</td>
<td>5.090</td>
</tr>
<tr>
<td></td>
<td><strong>53.090</strong></td>
</tr>
</tbody>
</table>

Table 10: Total cost

Up to this point we have justified all cost, but what the customer wants to hear is:

The project will begin on 1.9.2006, will finish on 2.1.2007, and it will cost **52.090 €**

7.6 References

[1] www.infojobs.es
Annexes
The aim of this annex section is to provide several essential documents for developing the project, but they are not part of the main study line. The articles you will find at the continuation are independent from each other. The main subject is the configuration management of the 'X' development environment on platform 'Y'.

At the following is described each article's motivation:

- Bluetooth addresses list. In this article, the Bluetooth address of all devices used in this project is written. It is very useful when detecting foreign Bluetooth devices that can disturb our tests.

- Getting started with aveLink BT SDK for Java. This article is a guide that explains step by step how to install the aveLink library and how to develop any Java application that uses Bluetooth functionality.

- Getting started with Aventana. This article is a guide that explains step by step how to install the Aventana library and how to develop any Java application that uses Bluetooth functionality.

- Getting started with iMotes and TinyOS. This article is a guide that explains how to install iMotes and TinyOS development environment and how to develop applications for this platform.

- Getting started with Symbian. This article is a guide that explains how to install the Symbian development environment and how to develop applications for this platform.

- Getting started with J2ME. This article is a guide that explains how to install J2ME development environment and how to develop applications for this platform.
Bluetooth addresses list.

In the following section are annotated the alias and address of the bluetooth devices used in this project. This is useful on uncontrolled environments, where we can not control the bluetooth devices in the surrounding area.

<table>
<thead>
<tr>
<th>Physical Device</th>
<th>Alias</th>
<th>Bluetooth Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imote Id: 86470.</td>
<td>IMOTE:Tibidabo</td>
<td>4b5f42886470</td>
</tr>
<tr>
<td>Imote Id. 86525</td>
<td>iMote86525 -INTRUSO</td>
<td>4b5f42886525</td>
</tr>
<tr>
<td>Imote Id. 86501</td>
<td></td>
<td>4b5f42886501</td>
</tr>
<tr>
<td>Imote Id. 86505</td>
<td></td>
<td>4b5f42886505</td>
</tr>
<tr>
<td>Imote Id. 86521</td>
<td>iMote86521 -prueba4- #86505</td>
<td>4b5f42886521</td>
</tr>
<tr>
<td>Imote Id. 86460 (ROOT)</td>
<td>iMote86460 -prueba2- #86460</td>
<td>4b5f42886460</td>
</tr>
<tr>
<td>Nokia 6630</td>
<td>Olga T.</td>
<td>00174bccb0c0</td>
</tr>
<tr>
<td>Nokia 6630</td>
<td>Javier C</td>
<td>00174bccd4f2</td>
</tr>
<tr>
<td>Bluetooth PC - Toshiba</td>
<td></td>
<td>000a94f52bb5</td>
</tr>
<tr>
<td>Bluetooth PC – Trust</td>
<td></td>
<td>000E14579A5</td>
</tr>
<tr>
<td>Bluetooth PC – Ezurio</td>
<td></td>
<td>008098c46dbc</td>
</tr>
</tbody>
</table>

Table 11: Bluetooth address list
Getting started with aveLink BT SDK for Java

The aim of this document is explain all the things that are not covered in the installation documentation of the library.

1. Hardware

Toshiba USB bluetooth adaptor

2. Library

First of all it is necessary being registered no be able to download the library. You can do at http://www.avelink.com/asp/indexB.asp?suite=B. It is a renovable trial version for fourteen days.

Uncompress it can execute Install.bat for installation. The installation folder will be C:\tools\Atinav. Follow very carefully what its explained C:\tools\Atinav\J2SE USB\ReadMe.txt.

3. Test project

Build an eclipse project with the following characteristics.

![Illustration 38: Eclipse project propery window](image)

Take a look that output folder will be: /build
It is quite possible that the CLASSPATH variable could not be initialized with a folder name, in this case initialize to any .jar file that holds from the folder you want deal with. Close the form. And now update the CLASSPATH value go toWindow|Preferences...|Java|Build Path|Classpath variable and edit the classpath value with the folder that contains the library and the license

Finally execute:
In the console window are printed out the addresses of all surrounding Bluetooth devices.

4. Problems found

4.1 Issue 1

Problem: License expired
Solution: Download it again! And replace the Certificate file

4.2 Issue 2

Problem:

Solution:
Read very carefully the library installation described (in my case) on the file `C:\tools\Atinav\J2SE USB\ReadMe.txt`. And copy the file `C:\tools\Atinav\J2SE USB\aveLinkBT\BIN\usb.dll` into `C:\windows\system32`

### 4.3 Issue 3

Problem:
Cannot get local device: `javax.bluetooth.BluetoothStateException`: 1021001 : 10131421093105 : License file not include the of `Certificate.crp` in CLASSPATH.

Solution:
CLASSPATH must contain the folder where is placed the file that contains the license.

### 4.4 Issue 4

Problem:
Please contact support@atinav.com or visit the web site [http://www.atinav.com](http://www.atinav.com)

Solution:
Download again the library and replace the license file `Certificate.crp`
4.5 Issue 5

Problem:

java.lang.NoClassDefFoundError: com/intel/bluetooth/test/StandaloneTest
Exception in thread "main"

Solution:

It can not be generated the .class files, so the .class files can not be executed. Type from command line:
C:\projects\final\TestLibrary.ATinav\build\ant compile
Afterwards execute the project from eclipse.

4.6 Issue 6

Cannot get local device: javax.bluetooth.BluetoothStateException: 1021001 : 10 101 : BCC properties file not found

Be sure that the in the folder \lib from your project is placed the file bcc.properties. You can find it on the folder where originally was installed the library C:\tools\ATinav\J2SE USB\aveLinkBT\LIB

4.7 Issue 7

The aveLink library installation, wipes out the installation of another USB bluetooth dongles.

4.8 Issue 8

Problem:

USB dll not loaded: no USB in java.library.path

Solution:

Copy "C:\tools\ATinav\J2SE USB\aveLinkBT\BIN\USB.dll" into c:\windows\system32
Getting started with Aventana

1. Hardware

Ezurio pc-mcia bluetooth

2. Software

Widdcom 4.0.1.700 provided with the USB Bluetooth adapter. Aventana. You can download it http://www.avetana-gmbh.de/avetana-gmbh/produkte/jsr82.eng.xml. It is a demo-version for 15 days. When you ask for the library you have to introduce the 3 bluetooth addressess of the usb bluetooth adapters that you are going to use.

3. Installation

Do not plug the usb bluetooth adapter, wait until widdcomm installation asks it.

4. Execution

There is a project called Inquiry.Aventana that detects the nearer bluetooth devices. This project it is placed on C:\projects\build\Inquiry.Aventana. You can import it from Eclipse. Take note that the Aventana library downloaded it is placed on lib folder.

5. Problems found

5.1 Issue 1

If the bluetooth address from the adapter that you are currently using was not in the form when you requested the library, the library will not work.

avetanaBluetooth version 1.3.11

Cannot get local device: javax.bluetooth.BluetoothStateException: license not valid for address 00-80-98-c4-6d-bc
Running on Version : 1.4.2 Build 18
Local Address 00:80:98:C4:6D:BC
Local Name YOGURSIN
Valid until 28.11.2006

5.2 Issue 2

If the library goes off, the the system date to any date when the library works. (e.g. 04.02.07)
Getting started with iMotes and TinyOS

1. VCP (Virtual Com port)

VCP is a tool that emulates a USB connection as a Serial Com (RS232) connection. The instructions for installing it are in http://www.ftdichip.com/Documents/InstallGuides/Windows_XP_Installation_Guide.pdf

Once installed, you have to plug the serial cable between the iMote and the laptop. On the device manager you will see the new serial port.

![Device manager window](image1.png)

*Illustration 43: Device manager window*

As default the TinyOS development environment uses serial com2 for updating the firmware iMotes. For changing it go to USB Serial Port (Com6) properties.

![Serial configuration window](image2.png)

*Illustration 44: Serial configuration window*
Advanced options let you modify with which COMx you want work with.

2. Tiny OS

You can download TinyOs 1.1.17 from [http://www.tinyos.net/dist-1.1.0/tinyos/windows/tinyos-1.1.7-installer-2](http://www.tinyos.net/dist-1.1.0/tinyos/windows/tinyos-1.1.7-installer-2). Once downloaded, execute `C:\projects\svn\Tools\TinyOS\1.1.7\setup.exe` and follow the instructions.

The installation folder **MUST BE** `c:\tinyos`, **AND CAN NOT BE OTHER ONE**. Click next for continue.
Click on Yes.

Click next.

Illustration 47: TinyOs installer screenshot.

Illustration 48: TinyOs installer screenshot.
Click on Continue. Finally cygwin will be installed.

For checking the installation has been performed correctly open cygwin window (execute C:\tinyos\cygwin\cygwin.bat) and type the toscheck command. Si todo ha ido bien deberá mostrar la siguiente pantalla.

Illustration 49: Cygwin window

Do not close this cygwin window for install the rpm's.

3. RPM's

Now you must to install 3 rpms. In a cygwin window go to the folder /cygdrive/c/projects/svn/tools/tinyos/rpm:

- NesC, type: $ rpm -Uvh nesc-1.1.2a-1.cygwin.i386.rpm
- imote platform, type: $ rpm -ivh contrib-imote-1.1.7-1.cygwin.i386.rpm
- gcc cross compiler, type: $ rpm -ivh arm-thumb-elf-gcc-3.2-1.cygwinh.i386.rpm

If there is any problem installing the rpm's include the flag --ignoreos to the command.
For checking that the installation has been performed successfully go to the folder /opt/tinyos-1.x/contrib/imote/apps/blink and type the command make install imote.
If appears the previous screen do not worry, because the toolchain ARM ADS 1.2 Toolchain (el linker) will be missing.

### 3.1 Issue 1

**Solution:**

Copy the dll `C:\projects\svn\Tools\TinyOS\cygwin\cygwin1.dll` to `C:\tinyos\cygwin\bin`. If exist the dll `C:\tinyos\cygwin\bin` it is very convinient back it up. Close the cygwin window that caused the problem and open a new one.

### 4. Uninstall

For uninstall TinyOs remove the folder `c:\tinyos`. And remove from the system registry any key that contains the values: cygwin and `c:\tinyos`. The last operation is very dangerous and can cause serious damage on the system.

### 5. ADS toolchain

Execute "C:\projects\Tools\ARM ADS 1.2 Toolchain\ARM ADS1.2\ARM ADS1.2\SETUP.EXE" for installing ARM toolchain (the linker)

Click Yes for continue.
You must install the tool on `\Program Files\ARM\ADSv1_2` and click Next.
Illustration 54: ADS Toolchain installer screenshot

Illustration 55: ADS Toolchain installer screenshot
Select the license file and click next.
Finally for checking that the installation has finished successfully. Open a cygwin window go to the folder `/opt/tinyos-1.x/contrib/imote/apps/blink` ant type the `$ make install imote`. When appears the following screen:

Illustration 58: ADS Toolchain installer screenshot

Illustration 59: ADS Toolchain installer screenshot

Finally for checking that the installation has finished successfully. Open a cygwin window go to the folder `/opt/tinyos-1.x/contrib/imote/apps/blink` ant type the `$ make install imote`. When appears the following screen:

Illustration 60: The firmware is ready to be loaded
Reset the iMote, in order to load the firmware on it.

Illustration 61: The firmware with the new application has been loaded.

6. Problems found

6.1 Issue 1

Problem:
The VCP is not configured for being COM2.

Solution:
Read VCP installation for COM2 or modify the makefile to configure the new COMx.

6.2 Issue 2

Problem:
Error L6002u.

Solution:
Install tinyOS in the folder `c:/tinyOS`

6.3 Issue 3

Problem:
You have installed the tool chain but you have even problems with the license.

Solution
Install the ARM Development suite (evaluation cd) on the same folder.
Getting started with Symbian

1. Introduction

The aim of this section is explain, step by step, how to install Symbian Development Environment and how to develop an execute a sample application.

2. Hardware

The computer where we will develope the application will be a PCWin32. Any laptop or desktop available in the market (year 2006) will be enough for developing applications for Symbian platform.

The mobile where we will execute the applications will be the NOKIA 6630. The operating system that carries this mobile is Symbian OS 8.0a. You can find the mobile specification at http://forum.nokia.com/devices/6630. It can be used other S60 o S80 terminal with Symbian OS version equal or greater.

3. Software

In this section is described how to install auxiliar tools needed for developing on Symbian platform.

3.1 Perl

It is mandatory install Perl in the computer where we are going to develop for Symbian platform. This tool is used for being able to install SDK S60, some tools are perl scripts.

3.1.1 Download

For download Active perl just follow this link. http://www.activestate.com/Products/ActivePerl/? x=1. The version that I have used has been 5.8.8. For knowing which is the version that we have to install, install SDK60 and during the installation process it will tell you.

3.1.2 Installation

Install Active Perl with the default parameters. If the installer ask you for include Perl executable in the system variable PATH do it. If not we will have to do it manually.
3.1.3 Check the installation

For checking the installation open a cmd window and type Perl -version. It should have to appear the Perl version and its license information.

![Perl installation screenshot.](image)

Illustration 62: Perl installation screenshot.

3.2 JRE 1.5

It is mandatory as well have installed JRE v1.5.0 (Java Runtime Environment) for instal S60 SDK. Some tools included in the SDK are implemented in Java.

3.2.1 Download

You can download JRE v1.5.0 follow this link [http://www.sun.com/]().

3.2.2 Instalation

Install JRE with the default installation parameters.
3.2.3 Checking the installation

For testing that the installation has been successfully installed; open a cmd window and type: java -version. It should have to appear information about JRE product version.

3.3 SDK S60 para C++

The SDK has the tools for developing Symbian applications (compilers, linkers, emulators,...) for S60 platform. Depending on the mobile operating system you should have to work with different SDK version. Check the mobile specification for knowing which SDK version you have to download.

<table>
<thead>
<tr>
<th>Symbian OS</th>
<th>SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 7.0</td>
<td>1st Ed.</td>
</tr>
<tr>
<td>8.0a</td>
<td>2nd Ed. FP1</td>
</tr>
<tr>
<td>8.1</td>
<td>2nd Ed. FP2</td>
</tr>
<tr>
<td>9.1</td>
<td>3rd Ed</td>
</tr>
</tbody>
</table>

In case of Nokia 6630 mobile you have to download SDK 2nd Ed. FP2.

3.3.1 Download

For downloading the SDK is mandatory being registred on Nokia Forum (it is free) [http://forum.nokia.com]. Once registered download it from [http://forum.nokia.com/info/sw.nokia.com/id/4a7149a5-95a5-4726-913a-3c6f21eb65a5/S60-SDK-0616-3.0-mr.html].

3.3.2 Instalation

Install JRE (Java Runtime Environment) with the default parameters. The installer will ask us for the net device that will be used by the simulator, in case that the application performs any internet access.
Afterwards will asks about the default SDK that you want to use. For Nokia 6630 mobile we will use S60_2nd_FP2_CW. Finally restart the computer.

### 3.3.3 Test the installation

For checking that the installation has been performed successfully. Open a cmd window and type: devices. It should have to appear the chosen SDK.

![Illustration 64: Network devices installed.](image)

Illustration 65: Default Symbian SDK installed

### 3.4 Nokia PC Suite

Each mobile has its own proprietary tools for downloading applications and files into the mobile. In case of Nokia 6630 is Nokia PC Suite. So install Nokia PC Suite with the default installation parameters.

When the mobile is being connected through the USB interface. The computer will detect the new hardware
and will ask us permission for installing the mobile drivers. Windows will look up automatically the drivers on the CD where is included Nokia PC Suite. This operation is performed several times, one for driver that the system needs.

Once the drivers are successfully intalled we will see in the explorer something like this:
3.5 Code Warrior

CodeWarrior is one of the IDE (Integrated Development Environment) for developing symbian applications.

3.5.1 Downloading


3.5.2 Installation

Install CodeWarrior with the default installation parameters. Once installed, restart the computer.

4. Sample application

The following sample applications will help us to understanding how a Symbian application works and checking if all the development environment was successfully installed.
4.1 helloworld basic

Open Code Warrior (C:\Archivos de programa\Nokia\CodeWarrior for Symbian v3.1\Bin) and go to menu: File | Import project from .mmp file.

Next, we will introduce the location of a mmp file
C:\Symbian\8.0a\S60_2nd_FP2_CW\Series60Ex\helloworldbasic\group\helloworldbasic.mmp. And click on next button.

Finally click on Next button. Now it will appear all sources loaded.
Up to this point we can generate several targets. For debugging and generate a release application, we will only be interested in WINSCW UDEB and ARMI UREL.

- WINSCW UDEB. For debugging the project with a simulator.
- THUMB UREL. For generating a release version, ready for being downloaded into the mobile.

### 4.2 Debuging

Select target WINSCW UDEB and build the application (F7).

Now we will add a breakpoint in the routine `void CHelloWorldBasicAppUi::HandleCommandL(TInt aCommand)` that is on the source file `helloWorldBasicAppUi.cpp`. 
And now click on debug button:

Illustration 71: Breakpoint set

Illustration 72: Launch application

Illustration 73: Mobile simulator

It will appear (passed several seconds) the simulator with the application loaded.

Select the application HW and click on OK button. Afterwards click on Option buttons and select the option Hello. It will be triggered the CodeWarrior debugger.

4.3 Release
The target THUMB UREL is used for generating the release version that finally will be installed on the mobile. For build the application (F7).

For generating the package that will be finally downloaded into the mobile, we will use the tools that offers us the Symbian SDK. So open a cmd window go to the folder C:\Symbian\8.0a\S60_2nd_FP2_CW\Series60Ex\helloworldbasic\sis and type makesis helloworldbasic.pkg

Illustration 74: Application installer creation.
Getting started with J2ME

1. Introduction

The aim of this section is explain, step by step, how to install J2ME Development Environment and how to develop an execute a sample application.

2. J2SE

For installing J2ME is mandatory that J2SE would be installed previously. Follow the instrucion for installing JRE in the article GettingStarted de Symbian.

3. J2ME

Execute C:\projects\svn\Tools\j2me\sun_java_wireless.toolkit-2_5-beta-windows.exe. And install it with the following installation parameters.

Destination Directory
C:\tools\WTK25
Program Folder
   Sun Java Wireless Toolkit 2.5 Beta
JVM Folder
   C:\Archivos de programa\Java\jre1.5.0_07

4. ANT

Ant is a make tool. You can download it from http://apache.rediris.es/ant/. Uncompress the .zip file into the folder c:\tools\ant and add the following system variables ANT_HOME=c:\TOOLS\ANT and append Path with %ANT_HOME%\BIN

For checking the installation, open a cmd window and type ant -version. It should have appear the ant version.

5. DayTimeApp sample application

Open a cmd windows and go to the folder C:\projects\svn\build\HelloWorld.j2me
\build and type ant run. It should have to appear the simulator with the midlet loaded.
6. Problems

6.1 Issue 1

It can be very probable the size of the midlet is not updated, update the attribute MIDlet-Jar-Size in the file C:\projects\svn\build\HelloWorld.j2me\res\DateTimeApp.jad. With the size (in bytes) of the file C:\projects\svn\build\HelloWorld.j2me\build\bin\DateTimeApp.jar.

Ford downloading the midlet in to the mobile click on C:\projects\svn\build\HelloWorld.j2me\build\bin\datetimeApp.jar. And Nokia PC Suite will be triggered for download the application, finally you will have to install the application from the mobile. Finally, select the application from the mobile menu application.