Development and operation of a Bluetooth demonstrator

Pedro Duarte, José Alberto Fonseca, Paulo Bartolomeu

Abstract – **Wireless communications in distributed and/or embedded systems is an important research topic today. One of the emerging standards in this domain is Bluetooth. In this paper a demonstrator of embedded devices connected with Bluetooth is presented. The demonstrator is used to show the development technology available at our laboratory. So, besides the description of its operation, the paper includes an overview of the Bluetooth standard and a step by step presentation of the implementation of the modules using one of the current development tools: BlueCore from the company Cambridge Silicon Radio.**

I. INTRODUCTION

Bluetooth is a protocol for wireless communication among small embedded devices which has become popular in the last few years. The research group at the Laboratory of Electronic Systems of IEETA has been working with Bluetooth modules since two years, starting with applications concerning environment monitoring and, currently, working in the wireless transmission of music in MIDI format. In order to illustrate the potential of development it was decided to build a demonstrator that can be used to show the operation of Bluetooth to interconnect two embedded modules or to interconnect embedded modules to a personal computer.

In this paper this demonstrator is described, either in what concerns its development or in what concerns its operation. This paper can then also be used as a user manual of the demonstrator.

The paper is organized in 6 sections. In the next section an overview of Bluetooth is given. The third section describes the architecture of the demonstrator. Section IV includes the description of the operation in a form suitable for a user. Section V gives some implementation details and section VI concludes the paper.

II. AN OVERVIEW OF BLUETOOTH

Bluetooth is an open standard for radio-based communications in the 2.4 GHz Industrial Scientific and Medical (ISM) band targeting low power, low cost, low range and moderate rate applications.

The Bluetooth technology appeared in 1999 as a Bluetooth SIG specification [1][2]. In 2001 Bluetooth specification 1.1 was released [3]. Later, in 2002, the IEEE 802.15 Group adopted this specification (with minor changes) as an IEEE standard, the IEEE 802.15.1 [4]. In 2003, the Bluetooth 1.2 version was officially released [5] and recently, the Bluetooth SIG adopted the new 2.0 Bluetooth specification [6]. Three major companies are supporting the Bluetooth 2.0 specification, namely Cambridge Silicon Radio (CSR) [7], Broadcom [8] and RF Micro Devices (RFMD) [9].

The Bluetooth specification defines the protocol stack shown in Figure 1. This protocol stack can be divided in three logical groups [10]: the application protocol group, the middleware protocol group and the transport protocol group.

The application protocol group consists on the applications (Bluetooth-aware or not) that use the Bluetooth technology.

The middleware protocol group consists on both Bluetooth specific protocols like the serial port emulation (RFCOMM) and other adopted protocols like the Object Exchange Protocol (OBEX).

Finally, the transport protocol group consists of protocols exclusively developed for the Bluetooth technology like Logical Link Controller and Adaptation Protocol (L2CAP) or the Host Controller Interface (HCI).

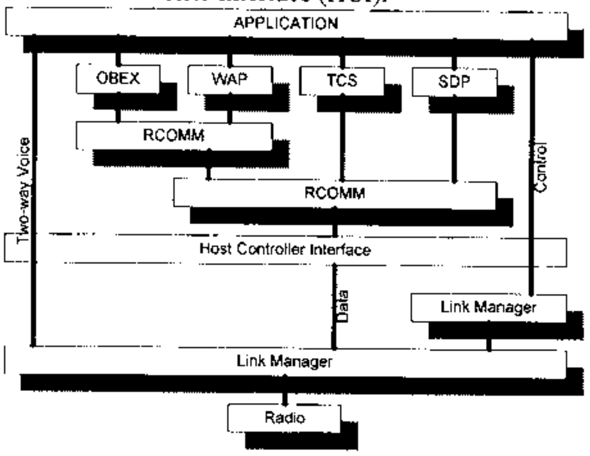

Figure 1 – Bluetooth Stack (Source: Robert Morrow)

The Logical Link and Adaptation Protocol (L2CAP) provides a transparent packet interface to higher layers through protocol multiplexing and packet segmentation (and reassembly). This protocol is also responsible for maintaining the negotiated Quality-of-Service (QoS) when applicable.

The Host Controller Interface (HCI) protocol isn't (as the name suggests) a protocol, but rather an interface allowing a host device to access the lower Bluetooth stack protocols.

The Link Manager Protocol (LMP) is established between two Link Manager entities, to manage the air-interface link properties between them. This protocol manages security, power and bandwidth usage.

The Bluetooth Baseband layer specifies all the procedures required to establish a Bluetooth link between two devices. It also specifies the network topology and defines the bandwidth sharing mechanism between devices in a piconet which is defined as the network consisting of a master device and at least one slave. One piconet can accommodate up to seven active slaves.

Baseband defines a Time Division Duplex (TDD) scheme to enable device communication. In this sense the piconet master transmits in specific slots and a slave can only transmit if the master has polled it.

Baseband defines two types of Bluetooth links between devices: Asynchronous Connectionless (ACL) and Synchronous Connection-Oriented (SCO). A piconet supports only one ACL link between a master and a slave while a maximum of three SCO links can be established between them.

ACL links carry best-effort traffic and are suited for asynchronous transmissions. Data integrity is assured by retransmission, sequence and forward error correction (FEC) mechanisms.

SCO links support periodic data transmissions at a 64Kb/s rate in each direction. SCO traffic can't be retransmitted and thus can only recover from errors by using FEC mechanisms. The latest Bluetooth specifications 1.2 and 2.0 define a new type named extended SCO (eSCO) that allows for retransmissions.

The radio layer defines the Bluetooth's radio transceiver characteristics, which operates in the 2.4 GHz licence-free ISM band using a Frequency Hopping Spread Spectrum (FHSS) technique through 79 one-MHz channels. This is performed using a pseudo-random sequence derived from the piconet master's address at a rate of 1600 hops/sec. The specifications 1.2 and 2.0 provide an additional feature named Adaptative Frequency Hopping (AFH) which improves Bluetooth co-existence with other wireless technologies (e.g. Wi-Fi).

Three classes of radio devices are defined: class 1 radios that can transmit up to 100 mw (≈100m range); class 2 radios up to 2.5 mw (≈10m range) and class 3 radios up to 1 mw (≈10cm range).

Additional information concerning the connection establishment can be found in [11]. A deeper overview of the Bluetooth standard can be obtained in [12].

III. ARCHITECTURE OF THE DEMONSTRATOR

A. Bluetooth embedded nodes

The most important parts of the demonstrator are the embedded nodes. Each node is based on a Bluetooth module fabricated by the company Airlogic [13] which we call ABM (from Airlogic Bluetooth Module). This module integrates a core made by the company CSR (Cambridge Silicon Radio) [12] which is called BlueCore2 and includes a specific 16 bits RISC processor called XAP2, a flash memory and all the hardware interfaces required to implement the lower levels of the Bluetooth hardware (figure 2). The ABM module offers also a collection of different digital interfaces that can be used either for configuration or for operation purposes. The module includes also two analog to digital conversion inputs (figure 3).

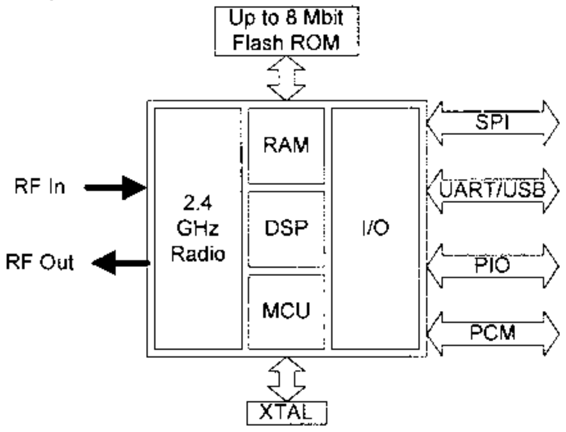

Figure 2 – Internal architecture of the CSR BlueCore2 core

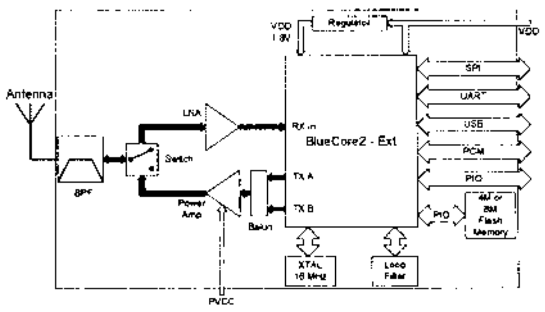

Figure 3 – The ABM - Airlogic Bluetooth Module

The CSR company offers to Bluetooth developers a software tool called BlueSuite that can be used to program the firmware or user application and to configure the ABM (or other CSR) modules. Another tool that is offered is BlueLab that is aimed to the development of integrated applications and/or higher level of Bluetooh stack integration, e.g., a RFCOMM stack interface, a complete SPP profile or a user application on the module. As an example, one could develop a program that reads from one of the A/D inputs and writes the value periodically to an emulated serial port using a small user program and the SPP profile.

The ABM module constitutes the base of our B2EN module (Basic Bluetooth Embedded Node). B2EN includes a set of buttons to interface with the user, a RS232 level adapter, power conditioning circuitry, connectors and a 7-segment display (figure 4).

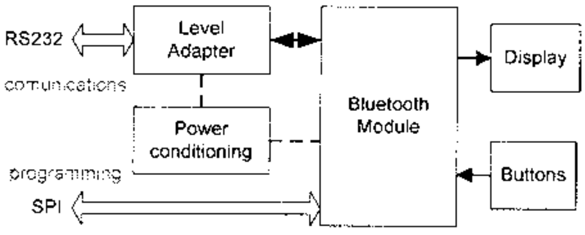

Figure 4 – Block diagram of the B2EN node.

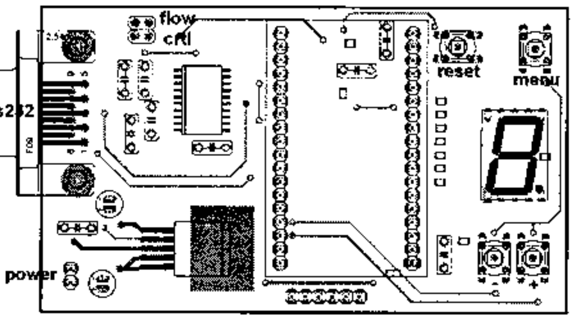

Figure 5 – Layout of the B2EN node.

Due to the modules used, the B2EN module is a Bluetooth Class 1 device with a 100 meters range. It can be powered between 4 and 15 Volts. Currently it is powered with 4 AA batteries.

B. Architecture of the demonstrator

The demonstrator offers two possible interconnections:

a) Interconnection of two B2EN modules which can communicate in different ways (Figure 6).

b) Interconnection of one or several B2EN modules to a personal computer equipped with a Bluetooth interface using the Serial Port profile (Figure 7).

Figure 6 – Interconnection of two B2EN modules

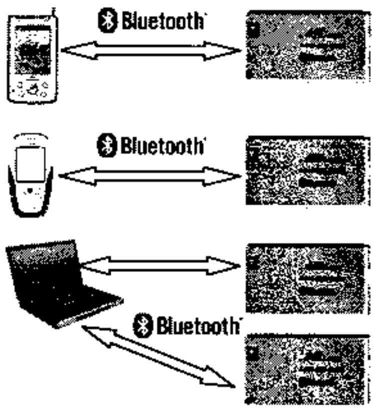

Figure 7 - Interconnection of one or several B2EN modules to a personal computer

IV. OPERATION OF THE BLUETOOTH DEMONSTRATOR

A. Modes of operation of the B2EN modules

Our Bluetooth demonstrator is designed to be used in two types of demonstration: visual demonstration and serial port demonstration. In the visual demonstration the 7-segment display is used to show the operation. In the serial port demonstration one has to connect the serial port interfaces of the B2EN modules.

Each B2EN module has then 3 possible modes of operation (their pairing will be explained latter):

Mode "1" – The B2EN module sends to the 7-segment display every character it has received through the Serial Port Profile. Just the 0-9 and A-F ASCII characters are displayed. All others are ignored. The buttons are not used here for operation.

Mode "2" – The value shown in the display can be incremented or decremented by means of the "+" and "-" buttons. Each time a change is performed, the B2EN module sends the character to the module it is connected to, using the Serial Port Profile.

Mode "3" – The information received at the RS232 port of the node is sent to the other node using the Serial Port Profile. The information received from this profile is sent to the RS232 port.

B. Modes of operation of the demonstrator

With the three previous modes of operation of the B2EN modules, one can settle different modes of operation of the demonstrator.

The two B2EN modules can operate together, using one of them to send an ASCII character when a button is pressed, that is displayed on the other B2EN module. To do this, one of the B2EN must be in mode 1 and the other in mode 2.

Another demonstration consists in emulating a RS232 cable connection, in which a B2EN module is connected to each of the devices formerly connected through a cable. In this case both must operate in mode 3.

The other mode of operation is to substitute one of he modules by a Bluetooth enabled PC or PDA. In this case the PC/PDA will receive the ASCII character sent by the B2EN module, or send a character that will be displayed on the B2EN display. For this demonstration the B2EN module must be programmed in Mode 1 and 2 respectively.

The other possibility is to use the B2EN configured as a Serial Port emulation. Mode 3 is then required. The PC/PDA is able to access the device connected to the B2EN module as if it was connected to it with a cable. When using a PC/PDA most of the software developed to work with a hardware UART can be used to conduct the tests.

B. Configuration of the B2EN modules

When configuring the operation mode, it is also necessary to configure the Bluetooth role of the B2EN module and choose the ASCII character that will be used in the paring process as part of the pin. It must also be chosen which, from the communicating devices, will be acting as Master or Slave of the Bluetoth piconet (see section 2).

The configuration process is as follow (see figure 5 to obtain the buttons' position):

Press and release, simultaneously the three buttons <Menu>, <+> and <->.

The display will show the "3" symbol.

Use the <+> and <-> buttons to configure the mode of operation of the B2EN module, according to the previous description.

Press and release the <Main> button.

The display will show the "M" symbol.

Use the <+> and <-> buttons to configure the Bluetooth role of the B2EN module as Master or Slave.

Press and release the <Main> button.

The display will show the "0" symbol.

Use the <+> and <-> buttons to choose the ASCII character that will be used as part of the pin during the pairing process. The possible values are "0 / 1/ 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9 / A / b / C / d / E / F".

Press and release the <Main> button.

Press and release the <Reset> button.

The module will now start working.

D. Operation procedure of the B2EN modules

After the configuration process takes place and before a connection can be established the module must be paired with another device. The pairing is used in order to create a common link key. The pin code used in this process is "pass_X", where "X" is the ASCII character selected during configuration. After a successful pairing the address and link key associated with the paired peer device will be stored and the process will not be repeated until a new configurations takes place. The pairing procedure is executed automatically by the B2EN module software.

Once the device has been paired, a connection can be established between the two devices to implement the desired operation mode. This will be also done automatically by the B2EN module software.

After a configuration, the ASCII character chosen will flash in the display while the pairing takes place and while there isn't a connection established. After a successful pairing the relevant information will be saved. Then a connection establishment is tried. In the future, after any "power cycle" or reset, the modules will automatically establish a connection with the device with which it was paired.

When there is a connection established it will be shown in the display a lighted character that can be the character / sent or received by the module (in mode 1 and 2) or the pin code ASCII character (in mode 3).

V. IMPLEMENTATION DETAILS

A. Firmware choice

To this application the best suited Bluetooth profile seams to be the serial port profile, because it is intended to transmit a few bytes or a stream of bytes. This is the same that happens in the case of using a serial port and cable.

According with the serial port profile the layers and entities in this profile are the ones shown in figure 8.

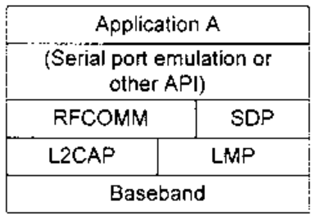

Figure 8 – Protocol model

The Baseband, LMP and L2CAP are the Bluetooth protocols correspondent to the OSI layers 1 and 2. RFCOMM is the Bluetooth adaptation of GSM TS 07.10, providing a transport protocol for serial port emulation. SDP is the Bluetooth Service Discovery Protocol.

The port emulation layer shown is the entity emulating the serial port, or providing an API to applications.

The better way to obtain a fully integrated system is to use a RFCOMM firmware module and to develop the rest of the stack/application in BlueLab [15]. In terms of hardware the two types of modules, HCI and RFCOMM are similar. The difference is in the firmware that they use. In the first type (called by CSR the "HCI firmware build", i.e., the ensemble compiled and ready to use) the module provides the layers up to, and including, the Baseband and LMP of the simplified stack schematic, while the "RFCOMM firmware build" includes also the L2CAP, SDP and RFCOMM layers.

B. Bluetooth Stack

The Blueetoth Stack is implemented using a scheduler to co-ordinate the stack layers execution. The scheduler provides a single-threaded co-operative non-preemptive multi-tasking environment. A direct consequence of the use of this simple scheduler is that there can be no blocking calls. A message passing method is used where a message is sent to a particular task via the scheduler, using a queue identifier as the destination. The scheduler executes the tasks that have messages pending. It is up to the software module to pull the message or messages from the queue and to act on them. The result of this is that the Bluetooth protocol stack becomes message driven.

C. Application

CSR has provided in the BlueLab development tool some libraries that sit on top of the Bluetooth Stack or use the module hardware to provide useful results. The most important are the Event library, the Persistent Store library, the Stream library, the Message and Scheduler libraries, the Buttons library and the Connection Manager library.

This last one is very useful to implement the Serial Port Profile because it is intended to allow simple, low-traffic RFCOMM connections between a pair of devices. It uses MessageQueue 0 for incoming messages and posts outgoing messages to MessageQueue 1.

The MessagesQueues are a way of passing messages between tasks/libraries. The tasks that have pending messages will be executed.

In what concerns our application, it consists essentially of a main program and a main task. The main program performs some initialization and launches the scheduler supplied by the tool suite. The events related with Bluetooth are handled by task 1. Task 1 executes whenever there are messages in MessageQueue 1. These messages are posted by the Connection Manager. The task produces messages posted to MessageQueue 0, thus triggering the execution of the Connection Manager. These messages are often requests that are answered with Indication and Confirm messages.

The message sequence passed between the Connection Manager and an application to execute the basic functions related to the connection establishment are the following:

Initialization of the Connection Manager:

- CM_INIT_REQ (application to connection manager)
- CM_INIT_CFM (connection manger to application)
- CM_OPEN_REQ (application to connection manager)
- CM_OPEN_CFM (connection manger to application)

Starts the Connection Manager and drives it to a pre-established state.

Inquiry:

- CM_INQUIRT_REQ (application to connection manager)
- CM_INQUIRY_RESULT_IND (connection manger to application)
- CM_INQUIRY_COMPLETE_CFM (connection manger to application)

Used to discover devices within the range.

Pairing as master or slave:

- CM_PAIR_REQ as master/slave (application to connection manager)
- CM_PIN_CODE_REQ (connection manger to application)
- CM_PIN_CODE_RES (application to connection manager)
- CM_PAIR_CFM pairing not finished (connection manger to application); the link key is received if it is the master
- CM_PAIR_CFM pairing complete (connection manger to application); the link key is received if it is a slave

Executes the pairing with a device to obtain the link key, being master or slave of the piconet

Add paired device to security manager, to avoid pairing procedure

- CM_ADD_SM_DEVICE_REQ (application to connection manager)

This is done when the device is starting operation with available information from a previous pairing (link key and Bluetooth address already available).

Connect as master

- CM_CONNECT_AS_MASTER_REQ (application to connection manager)
- CM_CONNECT_CFM connection complete (connection manger to application)

Connect as slave

- CM_CONNECT_AS_SLAVE_REQ (application to connection manager)
- CM_CONNECT_CFM connection complete (connection manger to application)

With this basic steps and a few more code the application that runs on the B2EN modules was created. In figure 9 it is shown the state machine that establishes the Bluetooth connection.

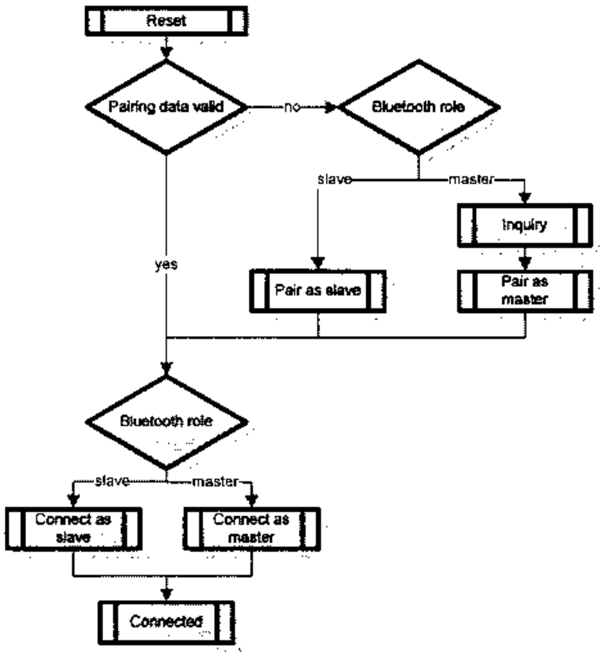

Figure 9 -- Bluetooth application state machine

As it can been seen, after a device is connected, if it has valid pairing information it will immediately establish a connection with the device with which it was paired. If it hasn't been already paired the slave will enter a state where it can receive pair requests, while the master will discover the available devices in its range and try to pair with them. During the pairing a Pin Code is used. This code is being used to limit the number of devices allowed for pairing. If both devices use the same pin the pairing will be done. In the other cases, the slave will keep accepting pairing requests and the master will return to execute a new inquiry to try to find a device that is using the same Pin Code.

Once the connection is established the functionalities of the Stream library are used to send and receive information through the created RFCOMM connection. When the device is working as RS232 port emulation the UART Source Stream is connected to the RFCOMM Sink Stream and the UART Sink Stream is connected to the RFCOMM Source Stream.

When the device is in mode 1 it uses an event generated when data arrives to a source to display the received data in the display. When it is in mode 2 it uses another event that is generated when a button is pressed to send the ASCII character to the RFCOMM sink. These functionalities are provided by the Event and Button libraries, jointly with the Stream and Scheduler libraries.

The configuration procedure is implemented using the Button library event and a state machine to drive the user through the configuration options. It uses the Persistent Store library to save the configurations in non volatile memory. This same library is also used after a successful pairing to save the link key, avoiding the need to repeat the pairing procedure in the future, thus accelerating the connections.

D. Development process

The code is written in C and is compiled with the compiler and linked with the libraries provided by CSR in the BlueLab development tool. The compiler and linker run under CygWin, a LINUX like emulator. The source code of some of these libraries is also available, so the user can modify it if necessary.

During the development the code can be compiled to a form suitable to be loaded in a simulator that is provided with the tools suite. This is done for debugging purposes. A Bluetooth module connected to the PC provides the lower level functionalities of the simulator.

After the code is in a more advanced development stage it can be downloaded to the target modules through a SPI interface that connects to the parallel port of the PC.

This SPI interface can also be used to work with the Persistent Store Keys that hold the configuration and that are used by the hardware and Bluetooth stack.

VI. CONCLUSIONS

In this paper a demonstrator of the operation of Bluetooth modules developed at the Electronic Systems Laboratory of IEETA was presented. This demonstrator can be used to show Bluetooth connections between two Bluetooth modules or between the modules and a Personal Computer. Besides the visible demonstration using buttons and 7-segment displays, the demonstrator provides a functionality similar to the Serial Port Profile. The demonstrator Bluetooth modules, called B2EN – Bluetooth Basic Embedded Nodes, were developed using Airlogic modules and the CSR BlueLab development tool suite. A short overview of the development process was also presented in the paper.

REFERENCES

[1] Bluetooth SIG, "Specification of the Bluetooth System 1.0", July 1999.

[2] Bluetooth SIG, "Specification of the Bluetooth System 1.0B", December 1999.

[3] Bluetooth SIG, "Specification of the Bluetooth System 1.1", Specification Volume 1, February 2001.

[4] IEEE Standard for Information technology—Telecommunications and information exchange Between systems—Local and metropolitan area networks—Specific requirements, "802.15.1 -Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)", June 2002.

[5] Bluetooth SIG, "Bluetooth 1.2 Core Specification", November 2003.

[6] Bluetooth SIG, "Bluetooth 2.0 Core Specification", November 2004.

[7] ambridge Silicon Radio, http://www.csr.com, April 2005.

[8] Broadcom, http://www.broadcom.com, April 2005.

[9] RF Micro Devices, http://www.rfmd.com, April 2005.

[10] Chatschik Bisdikian, "An Overview of the Bluetooth Wireless Technology", IEEE Communications Magazine, December 2001, pp. 86-94.

[11] Jaap C. Haartsen, "The Bluetooth Radio System", IEEE Personal Communications, February 2000, pp. 28-36.

[12] McDermott-Wells, P., "What is Bluetooth?", Potentials, IEEE ,Volume 23, Issue 5, Dec. 2004-Jan. 2005, pp. 33-35.

[13] Robert Morrow, "Bluetooth Operation and Use", McGraw-Hill, 2002.

[14] European Telecommunications Standards Institute (ETSI), "3GPP TS 07.10 version 7.2.0 - Digital cellular telecommunications system (Phase 2+) Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol", 1998.

WEB PAGES REFERENCES

[13] Airlogic Co. Ltd, http://www.airlogic.co.kr/

[14] Cambridge Silicon Radio, CSR http://www.csr.com/home.htm

[15] BlueLab SDK for single-chip BlueCore Applications http://www.csr.com/development/bluelab.htm

[16] BlueCore2, http://www.csr.com/products/bc2range.htm