NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GENERIC BLUETOOTH DATA MODULE

by

Ali M Aljuaied

September 2002

Thesis Advisor:	Xiaoping Yun
Second Reader:	Baer Wolfgang

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Generic Bluetooth Data Module.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ali M Aljuaied			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

Sensors are being used in many industrial and military applications. The most common ones are temperature and pressure sensors. Communicating with sensors has long been limited either to wired connections between the sensing element and the control station or to expensive, proprietary wireless communications protocols.

The emerging Bluetooth technology enables fast, efficient sensor communication, which eliminates wired connections and the associated manual tasks of initiating, checking, and changing sensor connections. It will be useful for a broad range of data-acquisition, measurement, control, monitoring, and similar applications. Bluetooth products currently available in the market support many usage models such as printer, headset, etc.

This thesis discusses and investigates the feasibility of interfacing sensors with Bluetooth modules by using off-the-shelf components. A prototype interface board was developed and connected to a Bluetooth module. Testing results showed that it is viable to implement Bluetooth-based wireless sensors for shipboard applications.

14. SUBJECT TERMS Sensors,Computing and Software, Wireless LAN, PAN, Bluetooth.			**15. NUMBER OF PAGES** 178
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

GENERIC BLUETOOTH DATA MODULE

Ali M. Aljuaied
Lieutnant Commander, Royal Saudi Naval Forces
B.S, Pakistan Naval Academy, 1988

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN COMPUTER SCIENCE

MASTER OF SCIENCE IN ENGINEERING SCIENCE (EE)

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Ali M Aljuaied

Approved by: Xiaoping Yun
Thesis Advisor

Baer Wolfgang
Second Reader

Chris Eagle
Chairman (Acting), Department of Computer Science

John P. Powers
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Sensors are being used in many industrial and military applications. The most common ones are temperature and pressure sensors. Communicating with sensors has long been limited either to wired connections between the sensing element and the control station or to expensive, proprietary wireless communications protocols.

The emerging Bluetooth technology enables fast, efficient sensor communication, which eliminates wired connections and the associated manual tasks of initiating, checking, and changing sensor connections. It will be useful for a broad range of data-acquisition, measurement, control, monitoring, and similar applications. Bluetooth products currently available in the market support many usage models such as printer, headset, etc.

This thesis discusses and investigates the feasibility of interfacing sensors with Bluetooth modules by using off-the-shelf components. A prototype interface board was developed and connected to a Bluetooth module. Testing results showed that it is viable to implement Bluetooth-based wireless sensors for shipboard applications.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to acknowledge Prof. Xiaoping Yun for his guidance and patience during work for this research, the author's wife and kids, Almuhanad, Bayder, Ghayda, and Gadeer, and finally the Lab Engineer James Calusdian for all the support he offered.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter covers the motivation and the research objective. It concludes by explaining how the thesis is organized.

A. MOTIVATION

Communication between electronic devices can only be achieved by a set of predetermined rules and standards, such as the Open System Interconnect model (ISO). These standards need to be applied to every aspect of the communication process. Over the last decade electronic communication has evolved significantly from the earliest packet switched data networks to the dedicated high-speed data communication.

New technologies are now emerging that allow wireless communications. The IEEE 802.11b or Wi-Fi standards are becoming accepted as the choice for the networking community as it supports features that enable it to perform handovers between access points. Additionally these standards can effectively become a transparent wireless network, expanding the static wired network. The IEEE 802.11b has data throughput of up to 11 Mbps, works in the unlicensed ISM band of 2.4 Ghz, uses frequency hopping or direct sequence spread spectrum, and has 100 m range coverage, all of which gives it viability against wired networks. This technology is evolving further with the advent of IEEE 802.11a and its competitor HyperLAN2 with even greater data rates where it can support 54 Mbps operating in a different frequency band 5 GHz.This frequency band will help eliminate interference because the band is not yet scattered by users.

Bluetooth technology is an emerging new technology. The need for Bluetooth came from the desire to connect peripherals and devices without cables. The available technology-IrDA OBEX is based on infrared links that are limited to line-of-sight connections. Bluetooth is further fueled by the demand for mobile and wireless access to LANs, Internet over mobile and other existing networks, where the backbone is wired but the interface is free to move. This not only makes the network easier to use but also extends its reach. The advantages and rapid proliferation of LANs suggest that setting up personal area networks, that is, connections among devices in the proximity of the user, will have many beneficial uses. Bluetooth could also be used in home networking

applications. With the increasing numbers of homes having multiple PCs, the need for networks that are simple to install and maintain is growing. There is also the commercial need to provide "information push" capabilities, which is important for hand held and other such mobile devices and this has been partially incorporated in Bluetooth. Blue tooth's main strength is its ability to simultaneously handle both data and voice transmissions, allowing such innovative solutions as a mobile hands-free headset for voice calls, print to fax capability, and automatically synchronizing PDA, laptop, and cell phone address book applications.

The 802.11b and Bluetooth are designed for different roles. There is no point trying to force one technology to do what the other is better suited to do. Bluetooth technology is designed with quick, seamless short-range networks, and features with better power consumption, a small protocol stack, robust data, voice transfer and low cost. These same features that are suitable for Wireless Personal Area Network (WPAN) create a disadvantage in Wireless Local Area Network (WLAN) terms. Therefore the features would be described as slow and limited. Likewise 802.11b is designed for infrequent mobility, IP-based data transmission, medium range and, high data rate making it an ideal choice for WLAN but a poor choice for WPAN, a clumsy, expensive and power-hungry choice in WPAN terms.

B. RESEARCH OBJECTIVE

The ability to pack a large number of transistors into a small area of silicon has made small-embedded devices capable of running complex protocols. Embedded controllers in devices are now capable of being programmed, controlled and used in various smart ways. Thus intelligent devices can now be embedded into the user's work and home areas. Various techniques are available to connect these embedded devices to a LAN or to the Internet, thus forming the so-called "embedded internet." Significant progress has been made in developing small and cheap sensors, which can pick up useful signals from the user environment without user interaction or explicit command. New kinds of electronic tags, which can be built on top of Bluetooth units and can specify what sort of information they exchange and how they communicate, enable interaction among a variety of devices, which have become available. This has opened up the

possibility of creating an "ubiquitous computing" environment. In this environment, the devices are controlled and activated by a combination of intelligent systems and strategically located sensors, which work without explicit user support. The facility to automate depends heavily on the ability of devices to communicate wirelessly with each other with more intelligent central servers, information repositories, sensors and actuators. These uses suggest that a technology like Bluetooth is extremely useful having a significant effect on the way information is accessed and used.

The main objective of this thesis is to consider the advantage of Bluetooth in sending sensors data to a control station. Questions investigated are: How can a Bluetooth module be interfaced with a sensor from off-the-shelf components? What hardware and software is needed for the construction of that unit. What generic functionality does that unit require?

C. THESIS ORGANIZATION

Chapter II of this thesis provides Bluetooth technology background covering some of Bluetooth current applications available in the market; it also covers the different aspect of the protocol layers and some of the techniques used in the technology. This chapter ends with highlighting some of the technology limitations. Chapter III provides the required hardware for constructing the Bluetooth generic data unit by describing all the different hardware types used in Bluetooth products; it also explains how to build the slave unit circuit by using a Futurlec board and the support board. Chapter VI contains an all needed software solution for both the master and the slave programs. Taking into account the generality for building these codes, such as the master code that contains the stack layers, the user can build the application by calling the different methods for the Application Program Interface. Chapter V contains the testing of the hardware and the software and the recommended future improvement. Finally Chapter VI concludes the thesis.

The next chapter covers the theortical part of Bluetooth technology, which covers applications, protocol layers, techniques implemented and, Bluetooth technology limitations.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BLUETOOTH TECHNOLOGY BACKGROUND

In this chapter theortical aspects of Bluetooth technology is covered, which includes types of applications, protocol layers, some of the techniques used and, technology limitations

A. BLUETOOTH APPLICATIONS

Bluetooth technology has satisfied the requirements for wireless communications with low power, cost effectiveness and ad-hoc connectivity between devices. There are numerous numbers of applications requiring the above requirements. The documents of Bluetooth specifications produced by the Special Interest Group (SIG) included profiles, which cover the standard details of all possible models of Bluetooth applications. All the applications can be classified into Data and Voice Access Point applications, cable replacements applications, and ad-hoc networking applications.

1. Data and Voice Access Point Aapplications

Bluetooth facilitates real-time voice and data transmission by providing an effortless wireless connection of portable and stationary communication devices that depend on factors like data rate, connection type, distance and quality of service needed by real-time applications, especially voice applications. The main affecting factor in this class of applications is the power needed by the devices to cover more ranges. Devices supporting these applications are available, but not in quantities required by the market; additionally these devices are expensive.

2. Cable Replacement Applications

In this class of applications, Bluetooth eliminates the need for numerous, often proprietary, cable attachments for connecting practically any kind of communication devices. Connectors are instant and maintainable even when devices are not within the line of sight. The range of each radio is approximately 10 m, but can be extended to 100 m with an optional amplifier. Quite a few devices or add-on to old devices available are in the market to support this class.

3. Ad-hoc Networking Applications

Within this class of application, a device equipped with Bluetooth radio can establish instant connection to another Bluetooth radio as soon as it comes into range without user intervention. There are a lot of devices or adds-on to old devices supporting this class in the market with reasonable prices.

B. BLUETOOTH LAYERS

Bluetooth is defined as a layered protocol architecture. Layering protocols have two nice features. First, they decompose the problem of building a network into more manageable components and, second, they provide a more modular design. Bluetooth protocol stack is divided into four layers according to their purpose and whether SIG has been involved in specifying these protocols. The specifications have developed these protocols more thoroughly.

1. Bluetooth Core Protocols

The core protocols contain specific protocols developed by the Bluetooth SIG and required by most Bluetooth devices. The radio layer is the first layer in this group. It specifies details of the air interface, including frequency, the use of frequency hopping, modulation schemes and transmission power. The different types of power classes used are shown in Table 1. Figure 1 shows the Gaussian Frequency Shift-Keying (GFSK) modulation, which is used to obtain the most efficient use of bandwidth while still maintaining an acceptable error probability. The Baseband layer is the next layer responsible for establishing connection within a small network called Piconet; additionally this layer is responsible for the addressing, packet format, timing and power control.

Power Class	Maximum Output Power (Pmax)	Nominal Output Power	Minimum Output Power[1)]	Power Control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	Pmin<+4 dBm to Pmax Optional: Pmin[2)] to Pmax
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	Optional: Pmin[2)] to Pmax
3	1 mW (0 dBm)	N/A	N/A	Optional: Pmin[2)] to Pmax

Table 1. Power classes used in Bluetooth radio (From Ref. [13].)

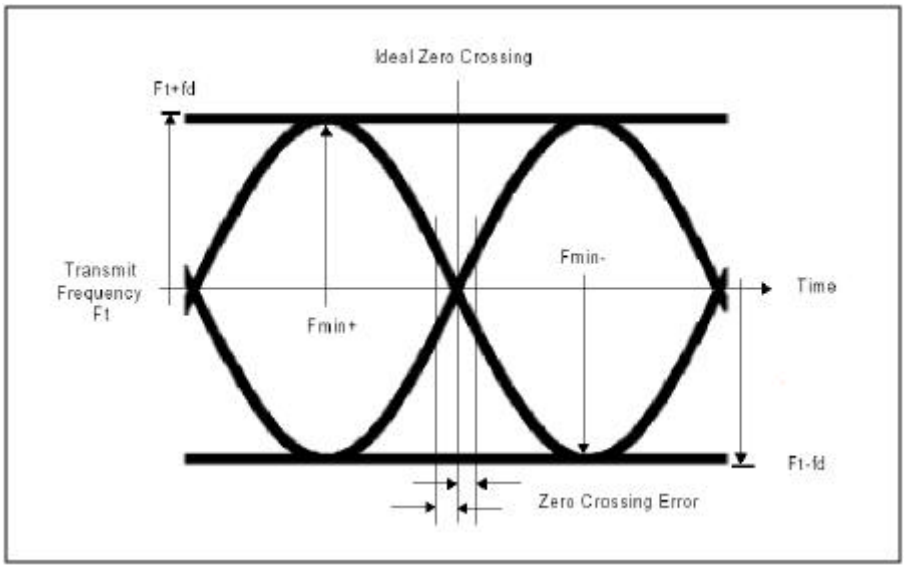

Figure 1. GFSK modulation used in Bluetooth radio (From Ref. [13].)

The next protocol is link manager responsible for link setup between Bluetooth devices and ongoing link management. This includes security aspects, such as authentication and encryption, in addition to the control and negotiation of baseband packet sizes. The protocol of Logical Link Control And Adoption is the last in this group, which adapts upper-layer protocols to the baseband layer. It also provides both connectionless and connection-oriented services to the upper layer protocol with protocol

multiplexing capability, segmentation and reassembly operations, and group abstractions. This layer permits higher layer protocols and applications to transmit and receive L2CAP data packets up to 64 kB in length. Finally the service discovery protocol that is responsible for device information, services, and the characteristics of the services can be queried to enable establishing a connection between two or more Bluetooth devices.

2. Cable Replacement Protocols and Telephony Control protocol

The cable replacement protocol is called RFCOMM. It presents a virtual serial port that is designed to make replacement of cable technologies as transparent as possible. Serial ports are one of the most common types of communication interfaces used with computing and communication devices. Hence, RFCOMM enables the replacement of serial port cables with minimal modification of existing devices. This protocol is developed by Bluetooth SIG but based on the European Telecommunications Standards Institute ETSI, TS 07.10, and the Version 6.3.0 standard.

3. Telephony Control Protocol

The Telephony Control protocol is a bit-oriented protocol that defines the call control signaling for establishing speech and data calls between Bluetooth devices. In addition it defines the mobility management procedure for handling a group of Bluetooth telephony devices. This protocol is developed by Bluetooth SIG but based on International Telecommunication Union Recommendation Q.931.

4. Adopted Protocols

The Bluetooth strategy is to invent only necessary protocols and use existing standards whenever possible. The following adopted protocols are existing protocols being used by Bluetooth:

The Point-to-Point (PPP) protocol, one of the adopted protocols, is an Internet-standard protocol for transporting IP datagrams over point-to-point link. It is designed to run over RFCOMM to accomplish point-to-point connections.

The TCP/UDP/IP are protocol standards defined by the Internet Engineering Task Force, being used for communication across the Internet. Implementing these standards in Bluetooth devices allows for communication with any other device connected to the Internet. The next protocol, OBEX is a session-level protocol developed by the Infrared

Data Association for exchange of objects. Finally Bluetooth incorporates WAE/WAP in its adopted protocol for the wireless application protocol.

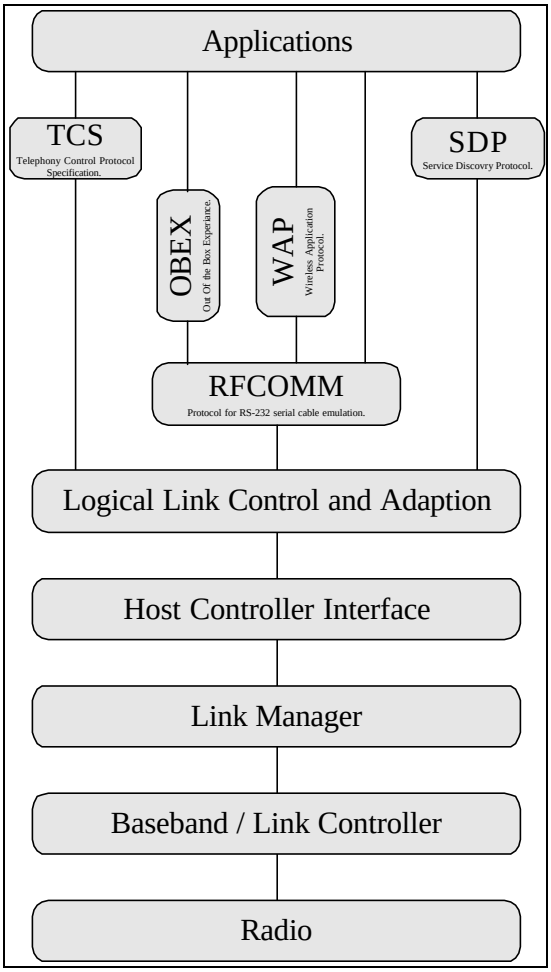

Figure 2. Bluetooth layers (From Ref. [3].)

C. TECHNIQUES IMPLEMENTED IN THE TECHNOLOGY

1. Frequency Hopping

The spread spectrum technique was developed initially for military and intelligence requirements. The essential idea is to spread the information signal over a wider bandwidth to make jamming and intercepting more difficult. Two types of spread spectrum are frequency hopping and, a more recent type, called direct sequence.

In frequency hopping technique, the signal is broadcasted over a seemingly random series of radio frequencies, hopping from frequency to frequency at fixed interval. A receiver hopping between frequencies synchronized with a transmitter picks up the message. This type of technique was chosen to be used in Bluetooth for two purposes; first it resists interference and multi-path effect and, second, it provides a form of multiple access among co-located devices in a different Piconet.

Bluetooth implements this frequency hopping technique by dividing the total bandwidth into 79 physical channels, each with a bandwidth of 1 MHz. This technique occurs by jumping from one physical channel to another in a pseudorandom sequence. The same hopping sequence is shared by all of the devices on a single Piconet. The hop rate is 1600 hops/s, so that each physical channel is occupied for the duration of 625 μs. The device responsible for being the master of the Piconet, which is a function of its address, determines the FH sequence.

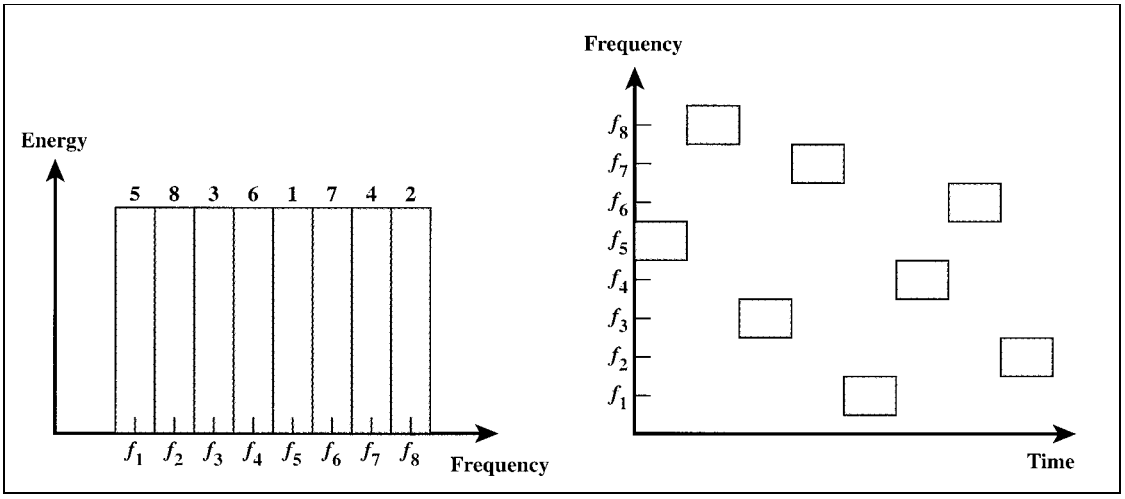

Figure 3. Frequency hopping (From Ref. [12].)

2. Adoption Power Control

Power management applications allow devices to power down for a large part of its duty cycle thus saving energy and prolonging its battery life. However the drawback is that the response time of the application is increased and, if not used correctly, power management can make applications infuriatingly unresponsive.

Bluetooth provides a number of low-power Baseband modes with each one being suited to a different type of application. The general mode is the active mode, which is used when the slave device is listening for transmission from the master device. This mode provides the fastest response time but typically consumes the most power. Sniff mode is used when the slave device becomes active periodically, that is performed in agreement with the master device and the slave device on the interval period. If the slave device receives a packet at the start of the period, it continuous to listen and receive packets; otherwise, it can sleep until the next period. This mode permits reduced power consumption by reducing the duty cycle of the radio; however, it is likely to be less responsive than the active mode.

The hold mode is used when a slave device stop entirely listening for packets for a specified time interval. The master and slave devices agree upon a hold time in this mode; the communication link is disconnected for the amount of time of the agreed interval. During that time the slave device does not listen for packets and may be doing something else, such as establishing a link with other devices or sleeping during the hold time. This mode may be less responsive than the sniff mode permitting a greater power savings than the sniff mode. This, nevertheless, depends upon the hold time duration and upon what the slave device does during the hold time.

The last mode is the park mode where a slave device maintains synchronization with the master device, but it is no longer active like the previous modes. This mode is typically the least responsive of the connected modes since the slave device must make the transition to become an active member before resuming general communications. However this mode permits greater power conservation.

In addition to the Baseband modes mentioned, another power saving feature used is adaptive transmission power. This feature allows slave devices to inform master devices when the transmission power is not appropriate, allowing the master device to adjust its transmission power. This is accomplished through the use of received signal strength. The master device maintains transmission power settings for each slave device it transmits to. This change in transmission power for one slave device does not affect other slave devices in the Piconet.

In addition to this several provisions have been included to save power. For instance the hopping mechanism is rather robust in that master and slave remain synchronized even if no packets are exchanged over the channel for several hundreds of milliseconds. Therefore, no dummy data has to be exchanged to keep synchronization between master and slave.

Furthermore, the receiver can decide quickly whether a packet is present or not. At the beginning of the receive slot, the receiver correlates the incoming signal in a sliding correlator, which is matched to the access code. Since the access code only lasts for a little more than 70 µs, after scanning for about 100 µs the receiver can decide to continue to listen or return to the sleep mode. If the access code is not received within the scan window, apparently no packet was sent, or it was so corrupted that further reception is not logical. If the proper access code is received, the receiver will continue to demodulate the packet header.

3. Piconet and Scatternet

The basic unit of networking in Bluetooth is a Piconet, consisting of a master device and one to seven active slave devices. The device designated as the master makes the determination of the channel that is used by all devices on this Piconet. It makes the determination using its own device address as a parameter while the devices designated as slave devices must tune to the same channel and phase. A slave device may only communicate with the master device and may only communicate when granted permission by the master device.

A device in one Piconet may also exist as part of another Piconet and may function as either a slave or master device in each Piconet. This form of overlapping is called Scatternet. The advantage of the Piconet/Scatternet scheme is that it allows many devices to share the same physical area and makes efficient use of the bandwidth.

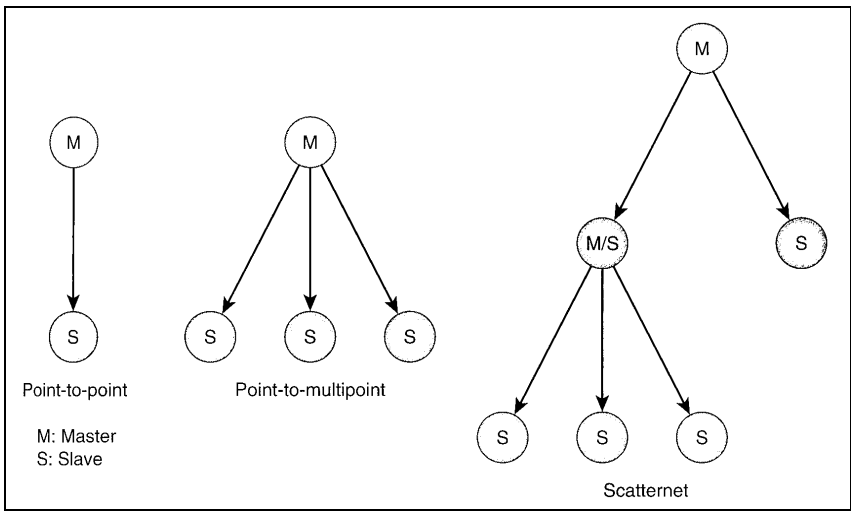

Figure 4. Piconet and Scatternet (From Ref. [12].)

4. Error Correction

Bluetooth makes use of three error correction schemes:

- 1/3 rate Forward Error Correction (FEC)
- 2/3 rates Forward Error Correction (FEC).
- Automatic Repeat Request (ARQ).

These error correction schemes are designed to satisfy competing requirements. The 1/3 rate FEC scheme simply involves sending 3 copies of each bit. The majority logic is used with each received triple of bits mapped into whichever bit is in the majority.

The 2/3 rates FEC is a type of Hamming code. This code can correct all single errors and detect all double errors in each code word. The ARQ scheme relies on other error detection codes, such as Cyclic Redundancy Code CRC. It takes advantage of the code, the fact that a master and slave devices communicate in alternate time slots. When the station receives a packet, it determines if an error has occurred using a 16-bit CRC. If an error is made, the ARQN bit in the packet is put to 1 as a way to indicate to the other station there is an error while requesting to send the packet again.

D. TECHNOLOGY LIMITATIONS

1. Limited range

Class 3 Bluetooth radios can transmit and receive signals over a 10-meter range of about 30 feet. While this appears to be an adequate range for cable-replacement applications, it may prove inadequate for other, more robust applications. Another range related limitation concerns using Bluetooth to connect to wireless LANs. Still, comparing Bluetooth's 30-foot range to the 150-foot range of HomeRF or 300-foot range of 802.11, one senses a distinct competitive disadvantage inherent to Bluetooth. The potential solution for this range limitation issue is to enhance the Bluetooth standard to allow for higher power, longer range devices.

2. Limited speed

Another limitation of Bluetooth is its data transfer rate. It has a theoretical maximum transmission rate of 1 Mbps. More practically, asymmetric data transmission is accomplished at 721 kbps with symmetric data transmission at just 432.6 kbps. This is fast enough for many applications such as voice transmission, which needs 64 kbps only. On the other hand, 400-700 kbps is fast for sending data for certain applications, but it is not fast enough for others like video and multimedia applications or for the transfer of large files. Comparing these rates with 4,10, and that 11 Mbps rates of IrDA, HomeRF, and 802.11/WiFi, respectively indicates the limitation Bluetooth faces as data transmission technology, even though the technology is mainly built for cable replacement applications.

3. Interference

One of the major limitations with Bluetooth's 2.4 GHz technology is that lots of other devices use the same unlicensed RF band. In general, interference with normal household devices appears to be more of a theoretical problem than a real one. By utilizing relatively fast frequency hopping, most interference with other devices in the 2.4 GHz band is minimized or completely eliminated. The only potential problems appear to be with microwave ovens and with devices connected to a 802.11 wireless network.

In the case of microwaves, the problem occurs because microwaves are generated at the 2.4-GHz frequency bands, which can actually jam transmission from any Bluetooth

device operating within a few feet of an operating microwave oven. In the case of 802.11 networks, a real problem does exist, although it is the 802.11 technologies that is the problem, not Bluetooth. Nevertheless 802.11 does not incorporate frequency hopping, which makes it much more vulnerable than Bluetooth to interference. Additionally, it can be affected by nearby Bluetooth devices.

4. Security

Even with Bluetooth's rather robust security measures, some experts claim Bluetooth technology contains some serious security gaps. Late in 2000, Markus Jakobsson and Susanne Wetzel, researchers from Lucent Technologies Bell Labs, stated that conversation initiated via Bluetooth enabled mobile phones to be bugged and Bluetooth's encryption procedures to be easily defeated.

Their first concern centers on the use of a four-digit customer Personal Identification Number (PIN). While a PIN is only as secure as its user, security can be further compromised when the capability exists to store the PIN in the memory of a Bluetooth device. The second vulnerability identified concerns referred to as "location attack" on the user of a Bluetooth device. This can come about when a third party obtains the device address of a specific Bluetooth device. With this address, a third party can physically trace the device and potentially compromise the security of the device's user. The third concern centers around the Bluetooth encryption key. It is possible for an electronic eavesdropper to utilize a relatively common bugging device to obtain encryption keys from Bluetooth devices used in public places. With this key in hand, the eavesdropper could use a third Bluetooth device to listen in on the conversation between the first two devices.

This chapter covered theortical aspect of Bluetooth technology, different classes of Bluetooth applications, and technology limitations. The next chapter will cover hardware components of Bluetooth devices.

THIS PAGE INTENTIONALLY LEFT BLANK

III. HARDWARE COMPONENTS

This chapter briefly covers the hardware section of the thesis. The first section discusses different Bluetooth hardware approaches while the second section covers the microcontroller used in the thesis. The third section describes the development environment. Finally the last section explains the circuit used to control the Bluetooth module.

A. BLUETOOTH HARDWARE

Choosing software architecture may limit the choice of hardware. There are three forms or approaches, which are available to integrate Bluetooth technology into an application. The first form is single chips/chip set form where a chip made by one of the venders of the SIG contains only Bluetooth core protocol mentioned in the previous chapter. The second form is a module where all protocol stack is contained in that module. The last form is actually half-way between ready product and module where it gives a student or researcher the ability to integrate the technology into their application and the ability to communicate with lower protocols via Host Control Interface (HCI).

1. Single chip/chip set approach

Numerous solutions are currently available from multiple venders. Chip sets come as separate radio and baseband devices in a variety of technologies: silicon-germanium, silicon-on-insulator, and CMOS, or single-chip CMOS device integrating the radio with the baseband. Chip set prices range from $8 to $29, decreasing with large volumes. This option is designed directly onto the product's printed circuit board (PCB). Designing Bluetooth technology directly onto the PCB is the optimum method if PCB end unit cost is the primary design constraint. This approach requires an RF design resource to provide matching networks, filters, amplifiers, and antenna to the transmitter and the receiver paths and requires expensive synthesis and test equipment along with a lengthy qualification process. If a hosted stack configuration is being used then HCI transport must be available and fully functional. Due to the inherent nature of RF strip-lines and micro-strips, a multiplayer PCB is needed to provide the required power planes, ground planes, and associated dielectrics and to separate the digital signal to avoid noise pickup

in the RF and crystal section. Some chips/chip sets cannot support the complete protocol stack; if there is no hosted system, this option will not be the suitable option.

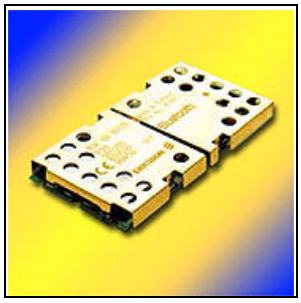

Figure 5. Bluetooth chip by Ericsson has the radio and datalink protocol only (From Ref. [1].)

2. Module Form

The alternative to buying a chip set is to get a "module." These are PCBs complete with RF design and antenna, which will be pre-tested and pre-qualified. This form is far simpler since the primary RF hardware concern is soldering the module onto the application motherboard. Modules are available from numerous sources with a choice of Bluetooth silicon. They are available in class 1 (30 feet coverage) and class 2 (60 feet coverage) and can take several forms. They are currently developed to be integrated with the entire external RF and system components (flash memory, crystal, filters, and amplifiers) into a single device with a predicted size being as small as 5 mm by 5 mm. These modules are all pre-tested and pre-qualified. They are not the right choice if the application requires small devices.

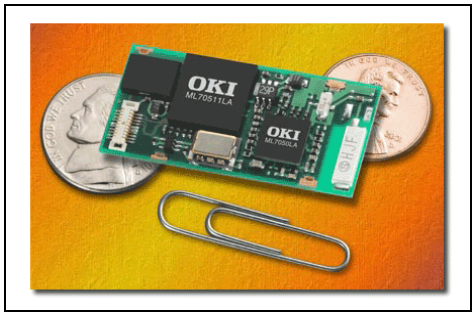

Figure 6. Bluetooth module by OKI contains the complete protocol stack (From Ref. [1].)

3. Researchers and Students Form

This form is between the ready product and self-contained module. They are shielded subsystems designed to be used as add-on peripherals. They feature an embedded CPU, different types of memory, as well as Baseband and radio circuits. The modules offer a generic HCI to the lower layers of the Bluetooth protocol stack while the higher layers of the protocol, as well as applications, must be implemented on the host system. Since the in-system CPU and memory are not available for user-specific implementations, even a minimal standalone Bluetooth node needs an additional host CPU executing applications and the corresponding higher layers of the Bluetooth protocol. The transport layers for communication between the Bluetooth module and the host system are standardized for UART, RS232, and USB. This thesis uses two of these products, the first one made by Troy mainly functions to enable printers with Bluetooth capability and, at the same time, it can work in a Baseband mode by directing it via RS232 communication port. CrossNet makes the second product where it has RS232 connection to the host that will have the upper protocol stack.

Figure 7. Troy Bluetooth module (left) CrossNet Bluetooth module (right)

B. MICROCONTROLLER

In a time when digital electronics is becoming more complex and less accessible to students and low-end circuit developers, microcontrollers have become excellent tools for learning about electronics and programming, as well as providing capabilities to

create sophisticated electronic applications fairly easily and inexpensively. The microcontroller provides a method to learn about digital interfacing and programming, and provides the capability to easily create application that control real-world devices.

There are three types of microcontrollers. Embedded eight-bit microcontrollers in which all the necessary resources (memory, I/O, etc) are available on the chip; in the required application circuit only power and clocking are provided. The second type is 16- to 32-bit microcontroller that relies completely on external memory. Despite the advantage that a microcontroller has with built-in memory, some applications require additional external memory. The last type is a digital signal processor microcontroller; this relatively new type is used for taking sample data from an analog system and calculating an appropriate response. This type is used in applications like active noise canceling or eliminating “ghosting” in video signals.

1. PIC16877

To run the higher Bluetooth protocol layers and applications, the Microchip PIC16F877 Microcontroller was chosen as the host CPU. It is classified as one of the Mid-Range Microcontroller made by the company. The mid-range PICmicro MCU has enough registers to be implemented in I/O peripheral devices as well as enough memory. The characteristics of PIC16F877 are shown in the table below:

Key Feature	PIC16F877
Operating Frequency	DC - 20 MHz
RESET and Delays	Power On Reset, Power up Timer, Oscillator Start-up Timer
Flash Program Memory (ROM)	8K x 14 words
Data Memory (RAM)	368 x 8 Bytes
EEPROM Data Memory	256 x 8 Bytes
Interrupts	14
I/O Ports	Ports A, B, C, D, E
Timers	3
Capture/Compare/PWM Modules	2
Serial Communications	Universal Synchronous Asynchronous Receiver Transmitter (USART)
Parallel Communications	Parallel Slave Port (PSP)
10-bit Analog-to-Digital Module.	8 input channels
Instruction Set	35 Instructions

Table 2. PIC16877 characteristics (After Ref. [7].)

2. Support Circuitry

Before using a Microcontroller in any application, there must be a supporting circuitry to support it in its work. The support circuitry for PIC16F877 Microcontroller consists of a decoupling capacitor, a reset circuit, and an oscillator.

The decoupling capacitor is used to minimize the effect of rapid changes in power demands by devices in the circuit including the PIC16F877, itself. It can be thought of as a filter smoothing out the rough spots of the power supply and providing additional

current for high load situations on the part. During the transition of the circuits from one state to another, the internal (and external) voltages inside the chip will fluctuate, which can cause the microcontroller to lock up, reset, or behave unpredictably in other ways. Due to this reaction, the capacitor is used to filter the voltage both within and without the Microcontroller. The most typically used values for decoupling capacitor are 0.01 to 0.1 µF.

The reset circuit is very simple; often it is just a "pull-up." This simple circuit is frequently enhanced in many applications to ensure that marginal-voltage situations do not cause problem with the operation of the microcontroller. The last part of the support circuitry is the clock. For any computer processor to run, a clock is required to provide timing for each instructional operation. An oscillator provides this clock built into the microcontroller that uses a crystal, ceramic resonator, or an RC network connected to the microcontroller to provide the time base of the microcontroller clock circuitry.

C. DEVELOPMENT ENVIRONMENT.

Hardware development environment was very simple. Tools available in the lab were used in building and testing the circuit. The most important tool used was the Proto-board Pb-503 made by Global specialties. This board had the required functions and tools to build test at the same time. It has two output voltage supply 5 and 13.5 V. The 5 V was used for this circuit. It also has ground connection, which can be wired to the breadboard; it also has 16 LEDs, which can be used to test input and output connections. It contains a variable oscillator that can be connected to an input. The support circuitry for the microcontroller was wired on the board while a serial connection was wired from the microcontroller to the Bluetooth module.

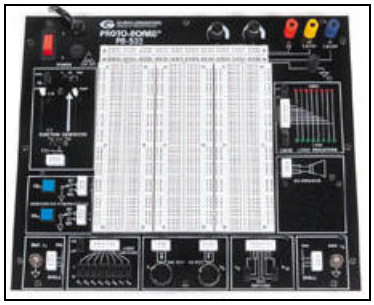

Figure 8. PB-503 lab board

The 2nd useful tool is the PICSTART Plus Development programmer made by the Microchip Company. This programmer is connected to a PC and can program 8, 18, 28, and 40-pin microcontrollers. Using this programmer with MPLAP software, the PIC16F877 was programmed before being used in the circuit. Finally some of the wires and serial port connections and switches were used along with the support circuitry of the microcontroller.

Figure 9. PICSTART Plus Programmer [From Ref. (7).]

D. CIRCUIT BUILDING

1. Initial Control Circuit

The initial circuit purpose was to test the PIC16F877 Microcontroller. The PIC16F877 requires a single +5 V power supply, which was provided by PB-503. There are two pins for power supply and two pins for ground, which are for reducing the voltage drop between the Vcc terminals of the chip and the Vcc conductors within the chip itself. The clock used was 16 Mhz, which was reduced by a MM74HC191 divider to 4 Mhz and connected to pin 13 of the microcontroller to provide chip clocking. A DIPswitch was used as a reset switch connected to pull-up resistor and to pin 1 of the microcontroller to provide resetting function to the chip. Port A0 was defined as an analog input port and port A1 was defined as a digital input port. Port B was defined as an output port. The microcontroller is connected serially to the Bluetooth module via a MAXIM RS232 transceiver chip.

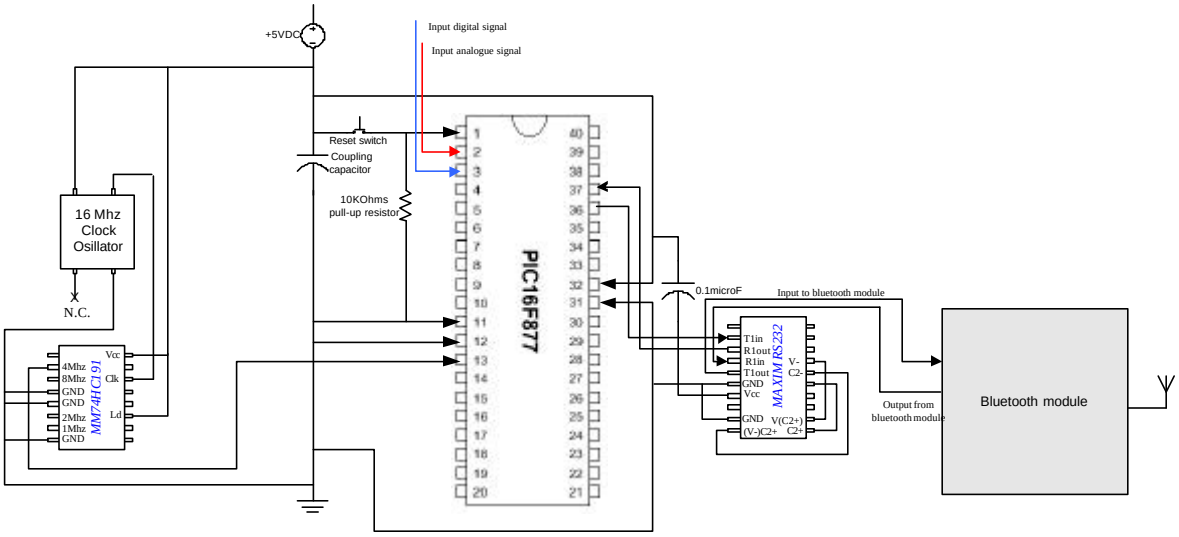

Figure 10. PIC 16F877 connected to Bluetooth module

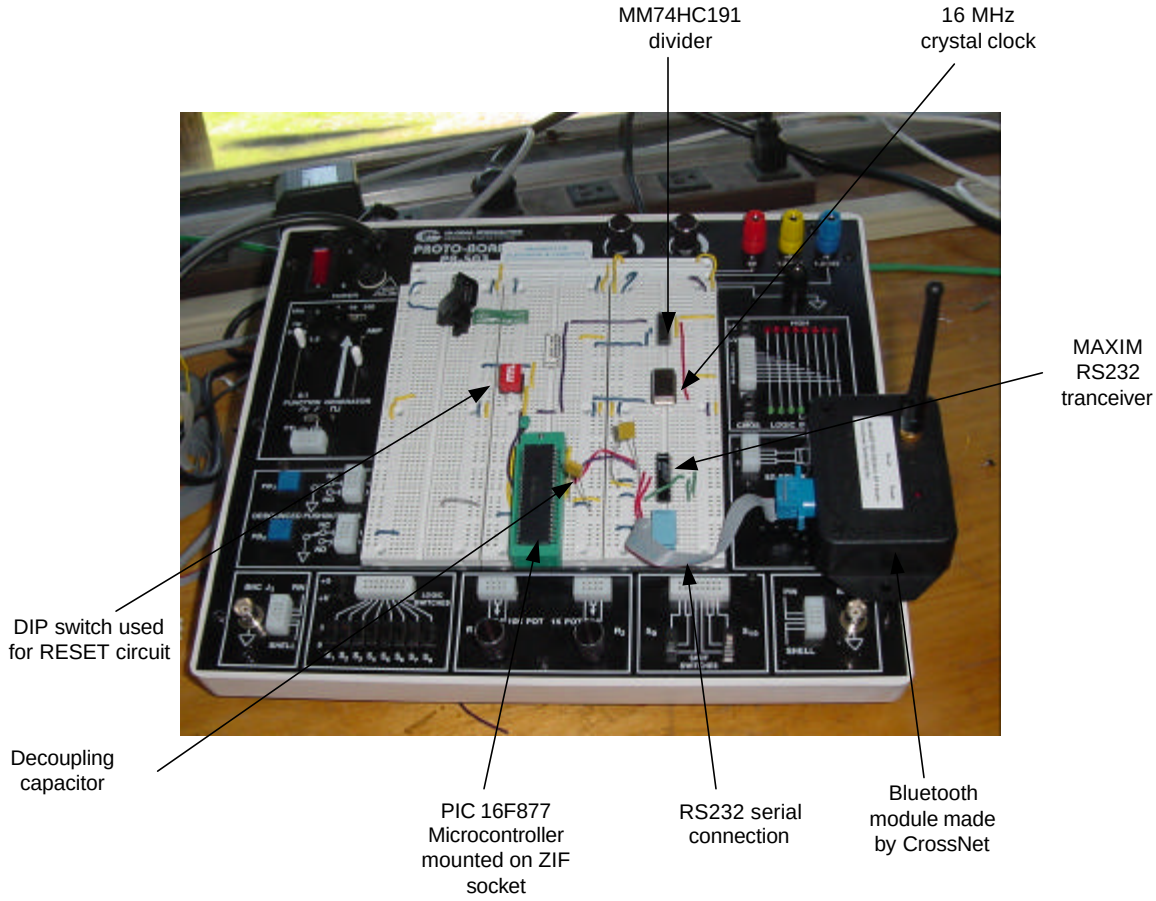

Figure 11. Initial circuit on PB 503

2. Futurlec Control Circuit

Instead of making a new circuit board for the initial circuit, Futurlec Company is manufacturing a small board containing PIC16F877 with all the required circuitry on the board. The board was very cheap at only $27. It has PIC16F877 with a 4 Mhz crystal clock and resetting switch. The board also has four types of connection to the outside world: RS232 and RS485/RS422 for serial control communication and ETT CON 34 pins that give access to the different ports of the microcontroller. The last connection was for LCD. The board also has extension slots for an extra memory. This board was used for the final testing and implementing of the circuit.

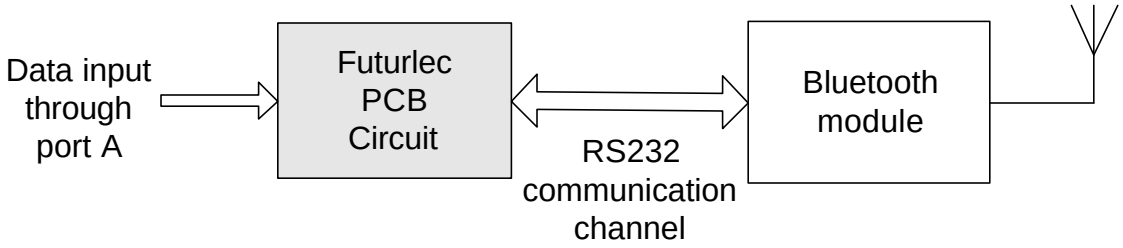

Figure 12. Futurlec PCB connected to Bluetooth module

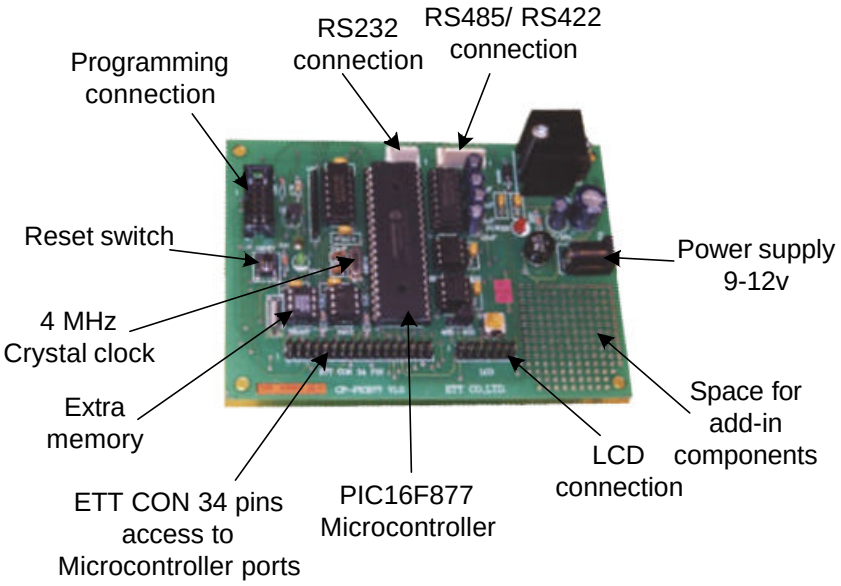

Figure 13. PIC16F877 in PCB manufactured by Futurlec

3. Attached Board

To make the Futurlec board serve the function of the thesis, another board was attached to it carrying all the extra-added components not in the Futurlec board. The attached board is carrying a temperature sensor and RS232 chip, which have the handshake RTS and CTS pins on it. In addition a digital switch was added to act as digital input to the microcontroller. The Bluetooth development kit made by Ericsson was connected to the attached board via a RS232 connection, getting its power supply through the attached board Figure 14. For future use and modifying, a space on the attached board is setup to add more components, such as DAC capability, and more sensors. The complete boards and the Bluetooth module are shown in Figure 15.

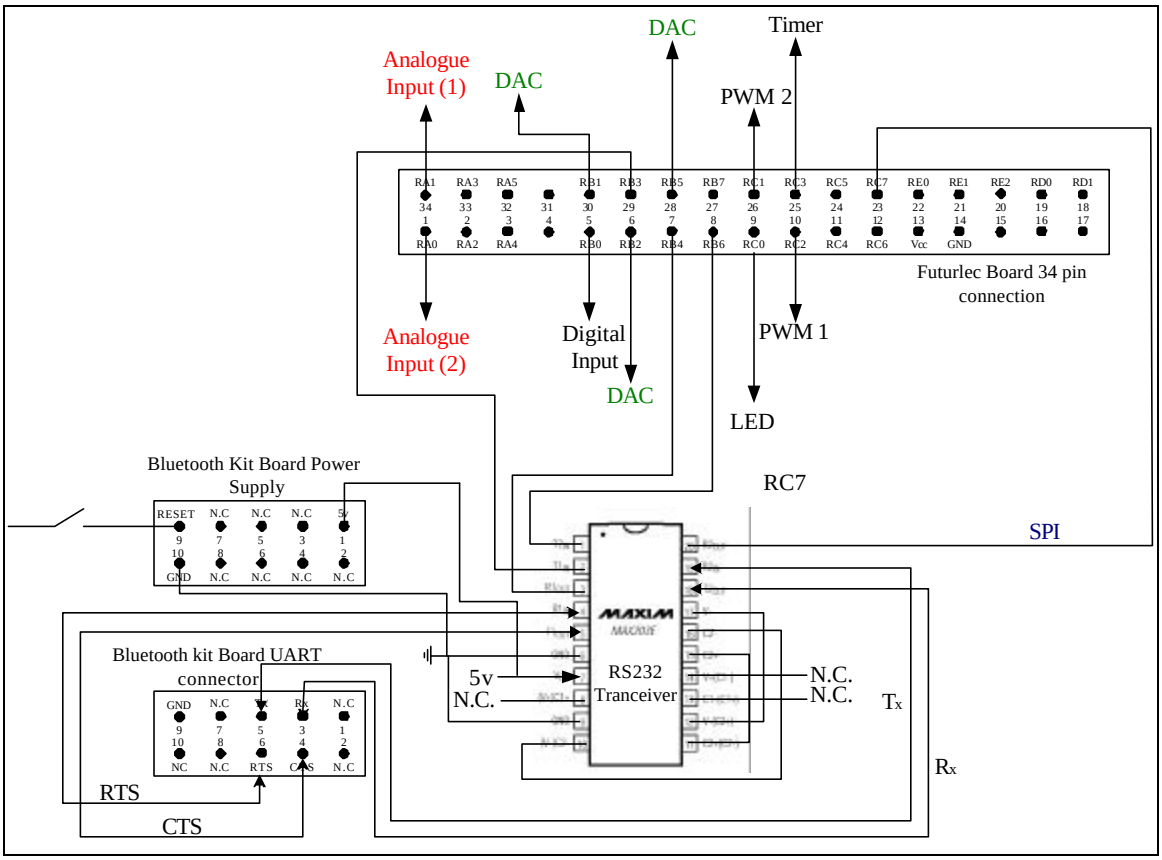

Figure 14. Replacing Futurlec board RS232 By RS232 203 and Bluetooth kit connection

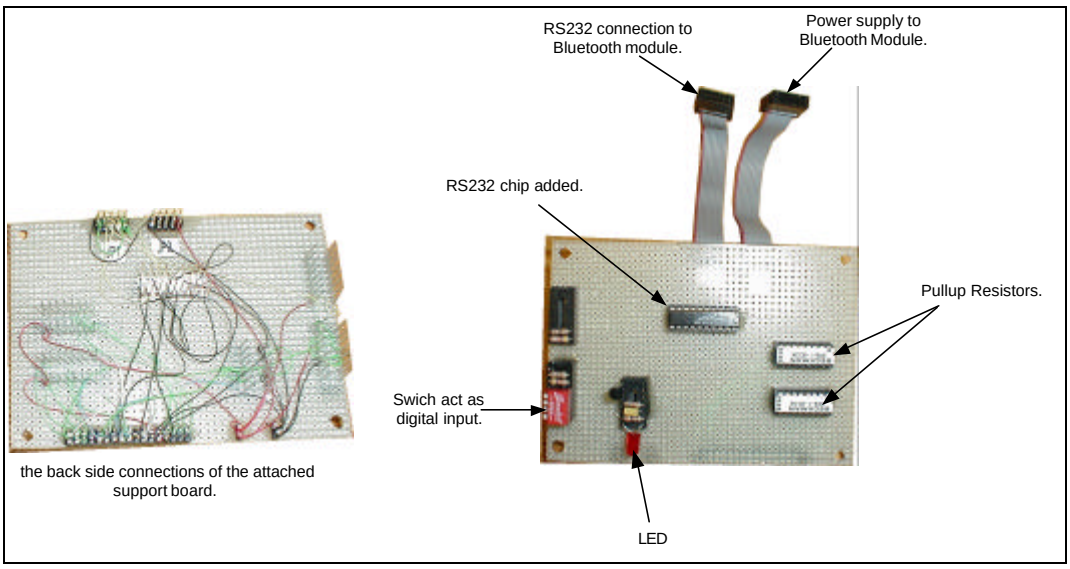

Figure 15. The support board with required added components

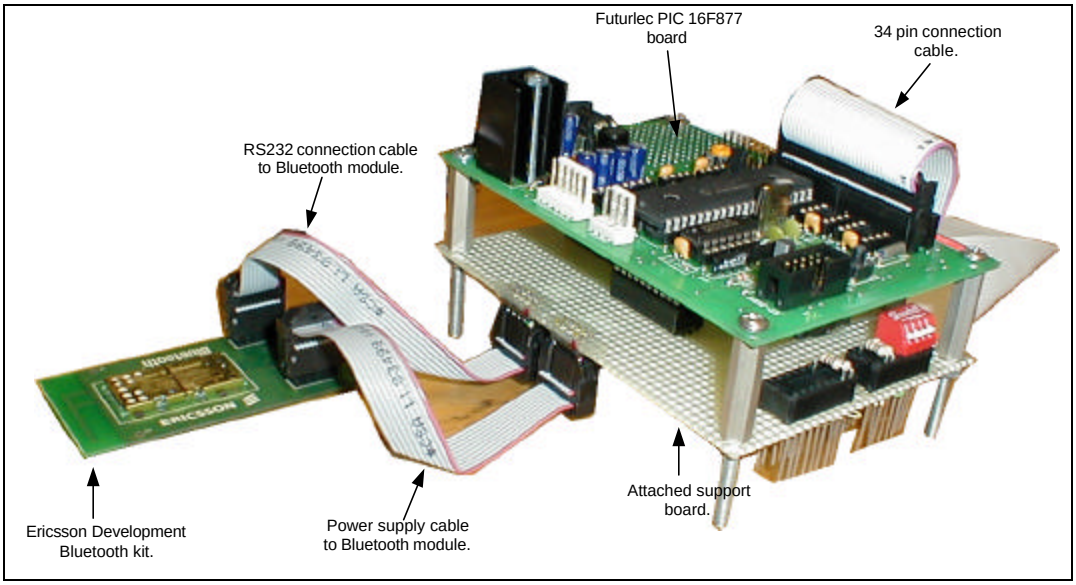

Figure 16. The complete Generic Bluetooth Data unit with Ericsson Bluetooth Development kit attached to support board

In this chapter the hardware components needed to interface sensor with Bluetooth module was discussed. The next chapter covers programs needed to run on the microcontroller to manipulate dtata received from sensor.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOFTWARE STACK

The software solution is divided into two parts; the first part was installed in the microcontroller and is called the host or slave program. The second part is installed on a PC having windows 2000 as an operating system and is called the controller or the master program. The logic for the Bluetooth connection and control of the microcontroller is programmed into the master program, which makes it very generic allowing it to be attached to different applications. The Bluetooth package type chosen for connection is the DM1 package shown in Table 3. This package can transfer data between 0-17 bytes with full error coding at the symmetric max rate of 108.8 kbps over the air.

Type	Symmetric	Asymmetric	
DM1	108.8 Kbps	108.8 Kbps	108.8 kbps
DH1	172.8 Kbps	172.8 Kbps	172.8 kbps
DM3	258.1 Kbps	387.2 Kbps	54.4 kbps
DH3	390.4 Kbps	585.6 Kbps	86.4 kbps
DM5	286.7 Kbps	477.8 Kbps	36.3 kbps
DH5	433.9 Kbps	723.2 Kbps	57.6 kbps

Table 3. Different packages and their speed. (From Ref. [6].)

A. SLAVE PROGRAM

1. Program description

The slave program was implemented in ANSI C and then compiled to PIC microcontroller machine language by using HI-TECH compiler, and then downloading into the flash memory of the PIC 16F877 microcontroller. The program contains five different files compiled together. They all have well-separated functions and provide each other with hardware independent services. The software is general and can control many types of processes. The data measured at the input are sent over Bluetooth communication in one package at user defined sample period between

6-255 milliseconds. The microcontroller is interrupt-driven and sets different status flags depending on the interrupt type. The different interrupts can be individually shut off, but the status flags are still raised.

The program uses different status flags to keep track of internal and external actions. An interrupt subroutine is called for every generated interruption. The subroutine investigates the interrupt and sets different status flags depending on the origin. The main thread is polling a general event flag that is raised during an interrupt, such as the receiving of a data package. After the general event flag is raised, the main thread investigates other flags to establish the reason of the event, and taking appropriate action. Actions that are taken by the master thread are done by a polling method; an example is the transmission of data to the Bluetooth module, which is done byte by byte. A register is loaded with the byte and a special status flag is polled. When the flag is changed the register has sent the byte and is ready to transmit the next one. An interrupt that occurs during execution in the interrupt subroutine is lost, so it is essential to keep the interrupt subroutine as small as possible.

2. Program files

The program consists of four source files with each file having specific functions and one header file (see codes in Appendix (A)). Global.h, the header file, contains all the required global variables, definitions of functions and macros of the program. Other files include this file. The Main.c source file, the starting point of the program contains the main function and the initialization code of PIC 16F877, such as assigning ports and setting the sample period of the ADC. It also creates the different status flags and most global variables used in the program. Finally this file with the help of the HCI.c file initializes the Bluetooth kit.

Next, HCI.c the file contains all Bluetooth kit hardware dependencies. It initializes the Bluetooth kit by calling different functions defined in the Bluetooth specification manual. The calls are blocked and are not returned until the Bluetooth kit has answered with an event indicating the success of the command. The file also provides the service of sending data over a L2CAP channel. A function call for sending a data package is blocked and the call is returned when the package is transmitted.

The Isr.c file handles all PIC 16F877 microcontroller interrupts; based on the type of interrupt, the file sets the right status flag. The decision of what action to take based on the interrupt type is left to the application. The Application.c file has all the required logic based upon different interrupts. It checks the status flags in priority order with the goal of minimizing the risk of overwriting old interrupt data. Figure 17 shows the program files and program starting point.

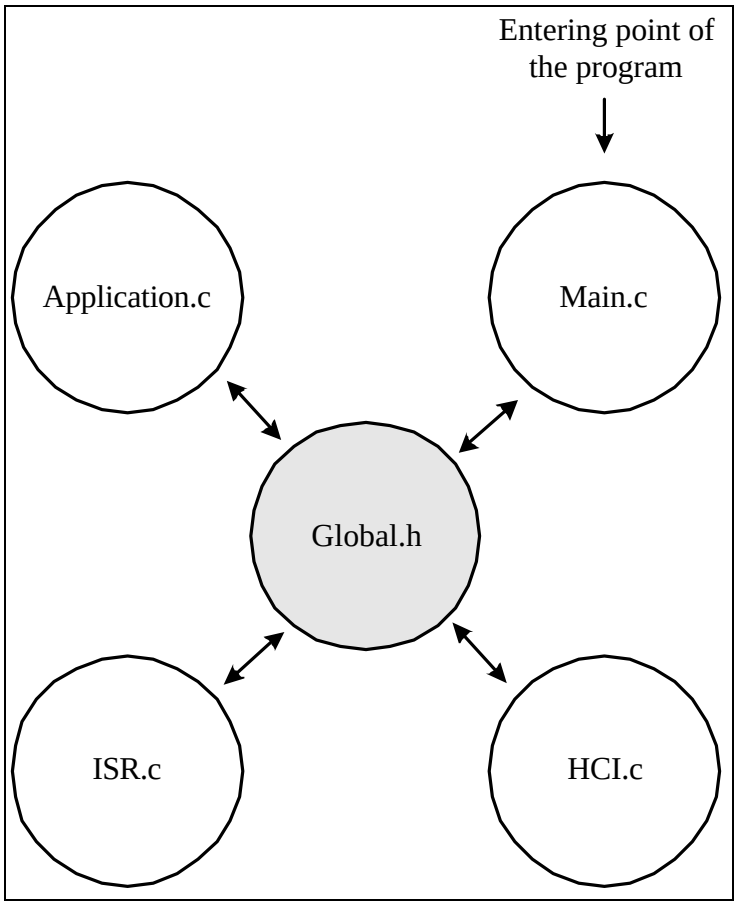

Figure 17. Different slave files and their relationship

The system clock in the microcontroller is running at 4 MHz with the time for executing one instruction being one microsecond. The main mechanism for synchronizing all external events is much slower than the system clock. The interrupt routine is short and can frequently execute the interrupt before the next occurrence. Multiple interrupt flags can be set. The received byte from the Bluetooth module is stored

internally in a two-bytes large queue First In First Out (FIFO). These bytes are then copied to the receiver buffer in the interrupt routine. The interrupt routine sets a status flag when it has detected a whole package.

B. MASTER PROGRAM

The software was developed in Java, which has the advantage of being machine independent, which is easier compared to C++. The software was built to match the slave program packet structure, and to be generic, which means it has an Application Program Interface (API) that can be used by an user-built application. The program was built in classes allowing each class to serve as a Bluetooth layer. All the classes mentioned here refer to the data part of Bluetooth; voice is not included and is beyond the scope of this thesis.

1. Program Description

The program is designed as a Bluetooth stack with its different layers. The lower layers provide different services to the above layers. The benefit of dividing the program into different separated layers is to make the task easier when adding more functionality. Different applications access the stack through an API. The API offers all functionality needed for hardware independent wireless communication. The ability to access the stack in a well-defined way makes it easy in developing a new application, as well as modifying existing programs to gain the functionality of wireless communication.

All received packets are reassembled from the byte stream and then scheduled in a Java thread. The thread copies the data contained in the packet from the general receiver buffer, into its own memory space. The data is maintained in a vector without any other copying needed during stack execution. This is done using the vector as a parameter together with a pointer to the first byte. The Above layers simply peel off its own data by adjusting the pointer. Sending data is done the same way. The top layer reserves all memory needed in a byte vector and copies its data into the vector. A pointer is used to keep track of the first byte. No extra memory allocation and vector copying is needed.

2. Program Files

According to Bluetooth specifications, there are three types of HCI packets to communicate with a Bluetooth module: the HCI Command Packet, used by the PC to send commands to the Bluetooth kit; the HCI Event Packet, used by the Bluetooth kit to notify the PC when events occur or when a command was executed, and the HCI Data Packet, used to exchange data between the PC and Bluetooth kit. More information on the structure and the different field of these packets can be found in Ref (1).

The program in the master consists of four Java class files and one Java interface file. These files act as Bluetooth layers, the application part is not included here and left for a user to develop according to the desired application. The layers here are specifically used in a general sense. The 1st file is the RS232.java class file, which establishes the layer of communication with a serial COM port and presents the first layer between the Bluetooth module and the stack in the master. The bytes stream that is received from the port is reassembled into data or event packets, and sent to the HCI.java class, which acts as the Host ControllLayer for the Bluetooth stack. On the other hand, data or command packets that are received from the HCI layer are transmitted over the serial COM port to the Bluetooth kit lower layers.

The HCI layer adds the general functionality of communication with the Bluetooth module. The Bluetooth module is accessed through this layer. The layer divides the incoming packets into two categories; data packets are directed or sent to a L2CAP layer and events are sent to the ControllLayer. The L2CAP (Logical Link Control and adaptation Layer) adds functionality to send data packets over Bluetooth channels; it supports protocol multiplexing, allowing multiple protocols and applications to share the air interface. It also enables segmentation of large packets used by higher layers into small packets for baseband transmission and the corresponding reassembly of those packets when received. The ControllLayer performs multitask functions, provided to the application by the API, such as initializing the Bluetooth module. For more information on the definition of functions used in the HCI class or the function of the other classes refer to Ref (1).

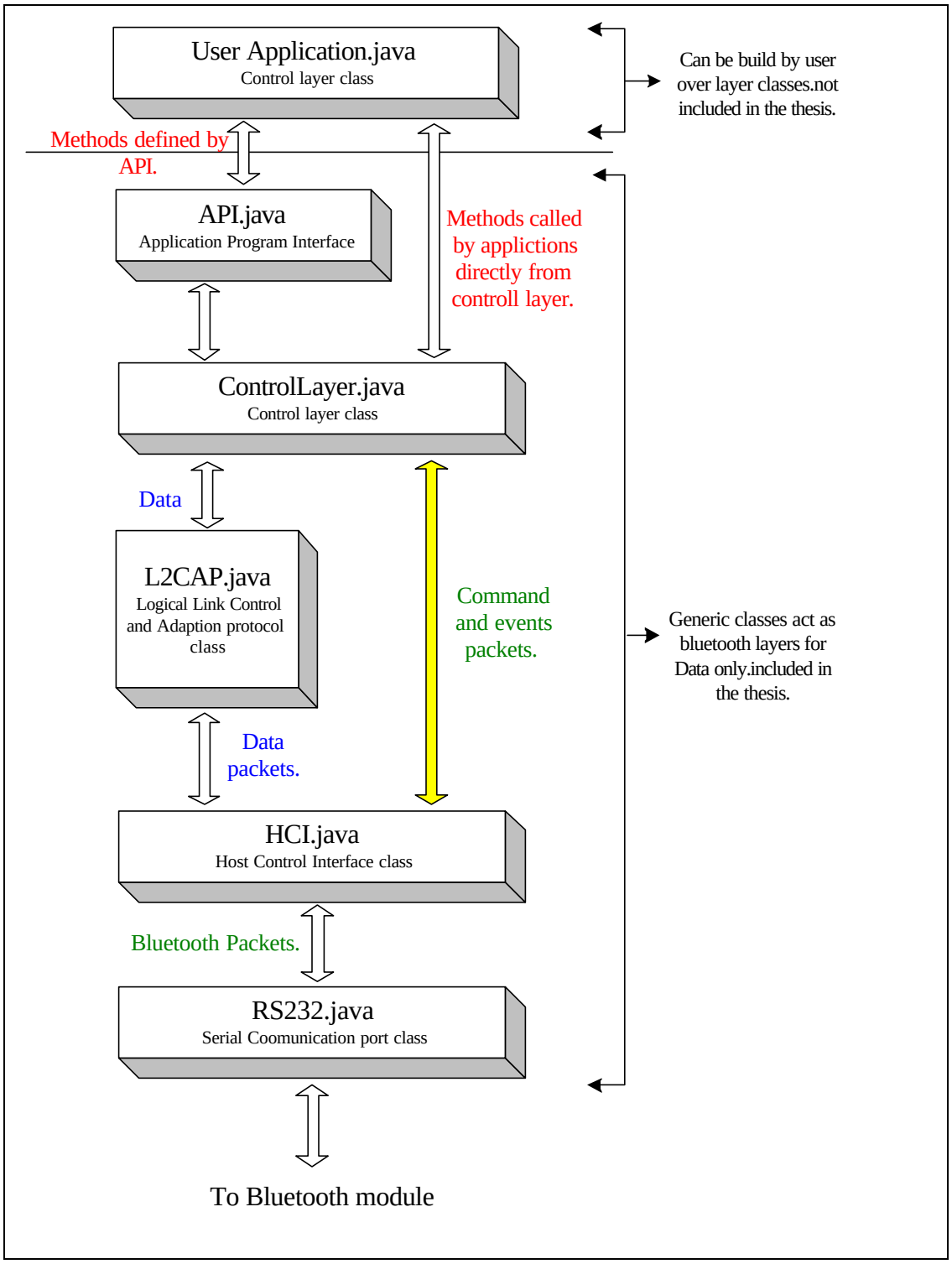

Figure 18. Different classes of Bluetooth data layers and data flow in master

The API is a Java interface that is implemented by user application. It provides well-defined methods between the layers and the user application. Methods called by the API are shown in Table 4. User application can also call other methods in the ControllLayer directly without going through the API for more flexibility in accessing Bluetooth functions; these methods are shown in Table 5. Figure 12 shows the different classes used and their relationship:

Method	***Explanation***
void receiveData(byte[] data, int pos)	This method is called each time a data packet is received. The int argument pos is used as a pointer points to the first data byte in the vector data.
void receiveEvent(byte[] event)	This method is called each time an event or command packet is received.
void BTFound(byte[] address)	This method is called for each remote device found during the inquiry phase. The vector contains the unique Bluetooth address of the remote device.
void connectionIsClosed(byte[] btAdress)	This method is called when the remote device closes the connection.
void stackInformation(String t)	This method is used for debugging purposes and knowledge of the work being done in the layers. It carries two types of messages; information messages start with I-> and error messages start with E->.
void receiveRSSI(byte value)	This method is called when the user wants to know the radio signal level.

Table 4. API methods accessible by user applications (After Ref. [7].)

Method	*Explanation*
boolean init()	This method blocks and returns true if Bluetooth initializing was successful; otherwise, it is false. If the Bluetooth module is not properly attached to the serial port, or if the module is not in reset mode, the thread will never return. This method can be called multiple times during initializing, returning all threads in case of success.
boolean inquiry()	This method blocks and returns true if Bluetooth inquiry was a success; otherwise, this is false. For each found remote Bluetooth device the layers calls Btfound.
byte[] creatConnection(byte[] adress)	This method blocks and returns the channel identifier (CID). The CID is used to specify the unique data channel. The return vector is two bytes long and will, if a channel was established, have a value not equal to zero. The input argument is the Bluetooth address of the remote device.
void sendData(byte[] CID, byte[] data)	This method sends data payload over the created channel. If any error occurs while transmitting the data, the method stackInformation will be called with an error message.
void closeConnection(byte[] btAdress)	This method closes the connection with a specific

	remote device.
void readRSSI(byte[] btAdress)	This method measures the radio signal strength of the connection.
reset()	This method invokes software resetting the Bluetooth module.

Table 5. Methods of controllLayer class, directly accessible by user application. (After Ref. [7])

C. COMMUNICATION PROTOCOL

The protocol for data packets allows a user to update all outputs or receive all input values in one packet. The solution fits in one Bluetooth packet, thereby maximizing the data speed and decreasing the complexity of the program. The bytes are transmitted with LSB (Least Significant Byte) first. The general structure of the data packet is shown in Figure 13. The payload is user defined. The head is added to the packet to be used by the layers and is of no interest to the API.

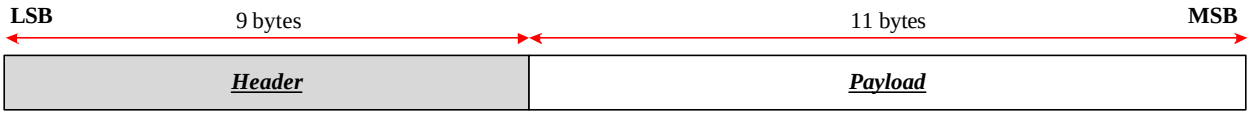

Figure 19. General structure of the Bluetooth data packet

The head is illustrated in Figure 14 which have parameters defined in Ref (1). Table 6 shows the break down of the packet header fields.

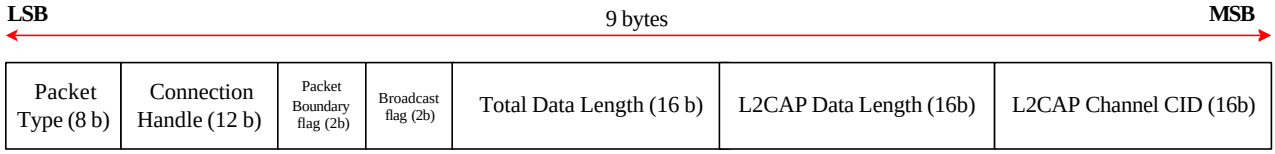

Figure 20. General structure of the packet header of the data packet

<table>
<tr><th>Field</th><th colspan="2">Explanation</th></tr>
<tr><td rowspan="5"><u>Packet type</u>, there are four types of packets specified on the right. The packet type used here is 0x02.</td><td><u>Value</u></td><td><u>Packet type</u></td></tr>
<tr><td>0x01</td><td>Command Packet</td></tr>
<tr><td>0x02</td><td>Data Packets on an ACL connection</td></tr>
<tr><td>0x03</td><td>Voice packet on a SCO connection (not used for this thesis</td></tr>
<tr><td>0x04</td><td>Event packet</td></tr>
<tr><td colspan="3"><u>Connection handle</u>: this is the channel between two Bluetooth devices and the data field is 12 bits long</td></tr>
<tr><td rowspan="5"><u>Packet Boundary flag (PB):</u> this used to fragment L2CAP packets but it is not used here, the field is 2 bits long and is always set to 0x02.</td><td><u>Value</u></td><td><u>Parameter meaning</u></td></tr>
<tr><td>0x00</td><td>Reserved for future use</td></tr>
<tr><td>0x01</td><td>Continuing fragment packets of higher layer message</td></tr>
<tr><td>0x02</td><td>First packet of higher layer message (i.e. start of an L2CAP packets)</td></tr>
<tr><td>0x03</td><td>Reserved for future use</td></tr>
<tr><td rowspan="5"><u>Broadcast Flag (BC)</u>: the Bluetooth kit used here is made by Ericsson for research purposes and doesn’t support Piconets or broadcast, the field is 2 bits long and is always set to 0x00.</td><td><u>Value</u></td><td><u>Parameter meaning</u></td></tr>
<tr><td>0x00</td><td>No broadcast only point-to-point connection</td></tr>
<tr><td>0x01</td><td>Active broadcast. Packet is sent to all active slaves</td></tr>
<tr><td>0x02</td><td>Piconet broadcast. Packet is sent to slaves including slaves in “Park Mode”</td></tr>
<tr><td>0x03</td><td>Reserved for future use</td></tr>
<tr><td colspan="3"><u>Total data Length</u>: the length is specified in bytes including the L2CAP data length and channel fields</td></tr>
</table>

L2CAP data Length: payload length in bytes
L2CAP channel (CID): the CID channel of the connection

Table 6. Parameter values and descriptions in packet header (After Ref. [6].)

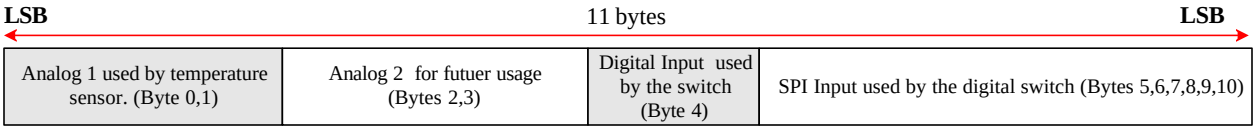

Figure 21. Bytes contained in the payload field from slave to master

Figure 15 shows packet payload from slave to master and contains Analog1, which has 10 active bits, the bytes are sent LSB first and specify the input voltage on pin 34 (see Figure 10). Analog2 is pin 1 of the 34-pin connection. It functions the same way as analog1, but it is not used in this thesis and can be used for another analog input, such as pressure gauge. Digital Input is pin No 5, which is the manual switch; it functions as digital interrupt triggered on the raising edge. The detection of the raising edge will set this byte to 0x01 for one packet. In addition, the SPI input has 6 Bytes reserved for any serial data received from any peripheral interface, which is not used here but can be added by assigning one of the input pins.

This chapter covered programs needed for slave and master unit of Bluetooth devices. The next chapter is covering testing the prototype unit and the results obtained.

THIS PAGE INTENTIONALLY LEFT BLANK

V. TESTS AND FUTURE IMPROVEMENTS

This chapter covers the approach chosen to perform prototype unit testing and the results obtained from the tests.

A. CURRENT STATUS OF IMPLEMENTATION

This thesis covered building a generic data Bluetooth module capable of receiving analog and digital data, and manipulating and transmitting it wirelessly via Bluetooth to another unit with Bluetooth connected to it. There are two parts covered: the first part is called the slave, composed of hardware and software, and the second part is called master, which has software only.

The hardware of the slave consists of two circuit boards and a Bluetooth development board. The first board is called the Futurlec board, which has PIC 16877 Microcontroller and all the required support circuitry, such as the clock. This board was purchased ready from the manufacturer for any general control function. The second board is called the support board; it was wired in LAB and was built for two reasons. The first reason is to carry the temperature sensor, the digital switch acting as digital input, all required pull-up resistors, the connection to the Futurlec board, and connections to the Bluetooth module. The second reason is because the Futurlec board RS232 transceiver chip does not have RTS and CTS, which makes it incompatible with an Ericsson development kit. Therefore instead of this chip, another RS232 transceiver chip was wired on the support board to make it compatible.

The slave has a program coded in ANSI C language found in Appendix (A). It consists of five files with a defined function of receiving data from the assigned pins on the 34-pin connection on the support board. The program then manipulates via the microcontroller registers and constructs it as a Bluetooth data packet, which is sent to the Ericsson Bluetooth development kit via RS232 connection. The program is responsible for command and event packets being sent to the Bluetooth module based on the different interrupts being received. After the program was coded it was compiled by a HI-TECH compiler, which converted the C code to assembly language of the microcontroller. The generated code is in Appendix (B).

The second part is called master and it consists of an Ericsson Bluetooth module connected to a PC with a program coded in Java acting like Bluetooth stack. The program consists of classes with each class acting as a layer of the stack and serving the layer above it. The RS232 class receives and sends data, command and event packets through PC serial port, which has an Ericsson Bluetooth development kit connected to it. The HCI class function as an interface between upper layers and lower layers; the L2CAP class segments data into Bluetooth packets and passes it to the HCI or to the ControllLayer. ControllLayer manages the link and sends control command received from the application layer. The program has an API with defined methods, so any future work application can call these methods and use them.

B. CONDUCTING TESTS.

Due to a shortage of time and the availability of only a single Ericsson Bluetooth development kit, the testing was divided into parts. Each part covered a component. The slave was divided into three parts; the Futurlec board part, the support board part, and the Bluetooth module part. Programming the microcontroller with simple ADC program and installing it on the board tested the Futurlec board part. The support card part was connected to the Futurlec board and to a hyper terminal connection on a PC, assuring that the temperature sensor was passing data to the boards. The third part tested the connection between the boards and the development kit; to do this RS232 serial box breaker shown in Figure 23 was used.

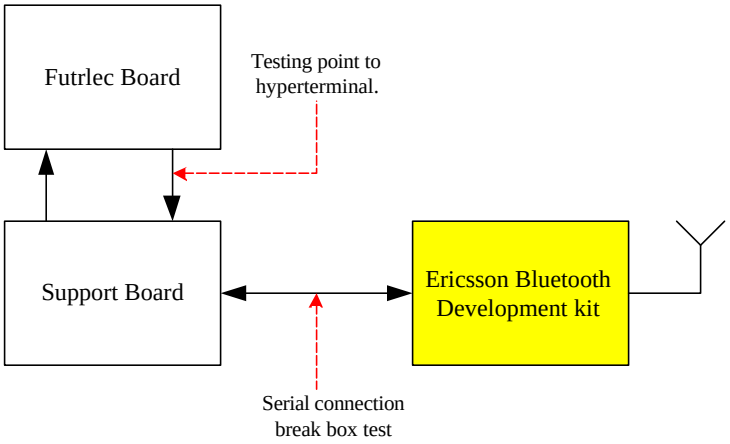

Figure 22. Testing points

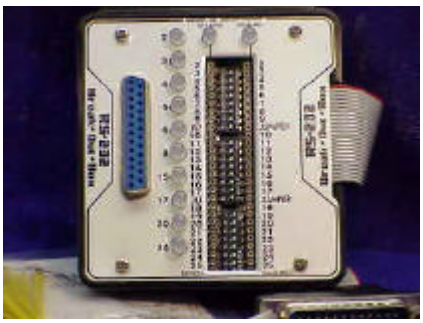

Figure 23. RS232 Break Out Box

The testing of slave program was done via simulator that came with the MPLAB program, which enabled viewing the different registers and RAM, and ROM of the microcontroller during program execution. The master program was compiled using JBuilder Ver 7.0. Since no GUI was built for this program, building a GUI can be future work.

C. TESTS RESULTS

The testing result obtained by the hyper terminal is shown in Figure 24. It indicates that data transfer occurred between the boards and the serial connection going to the Bluetooth module. The testing of the connection between the Bluetooth module and the board using the serial connection breaker indicates communication in the four channels TD, RD, RTS, CTS. The simulator testing of the slave program is shown in Figures 25,26,27. Tracing the program counter and values in the different registers shows that the program is running. The simulator results when injecting interrupts through the simulated input pins showing that the program reacted as expected in the specified registers.

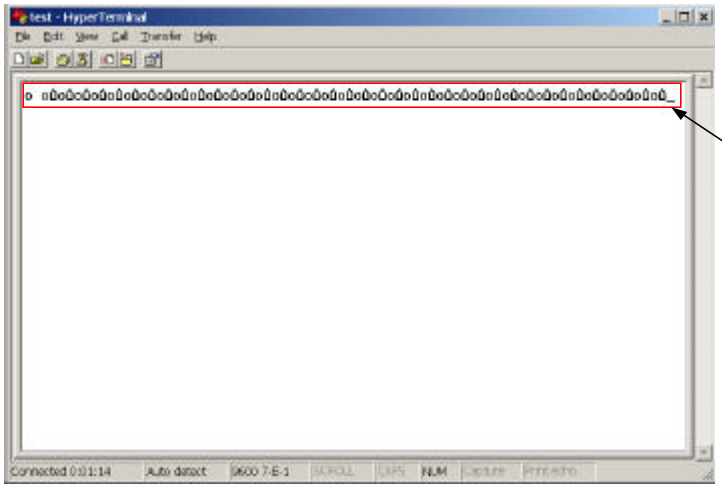

Figure 24. Data from combined boards via RS232 hyper-terminal

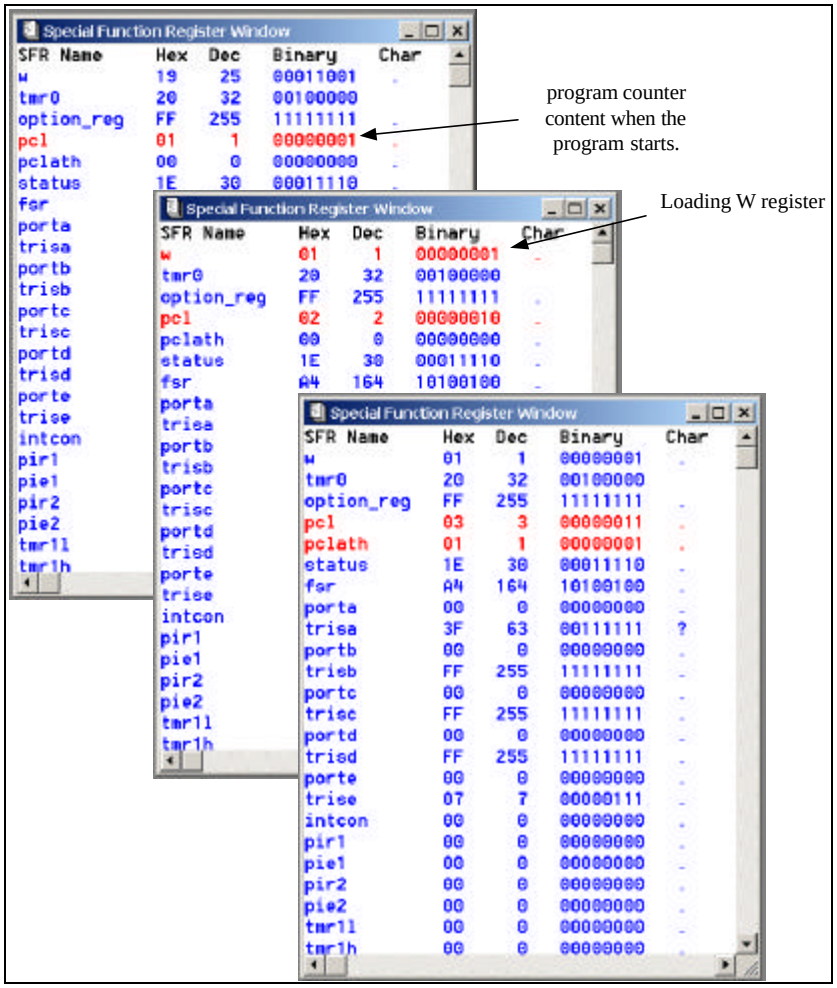

Figure 25. Three snapshots of microcontroller registers in MPLAB simulated slave program

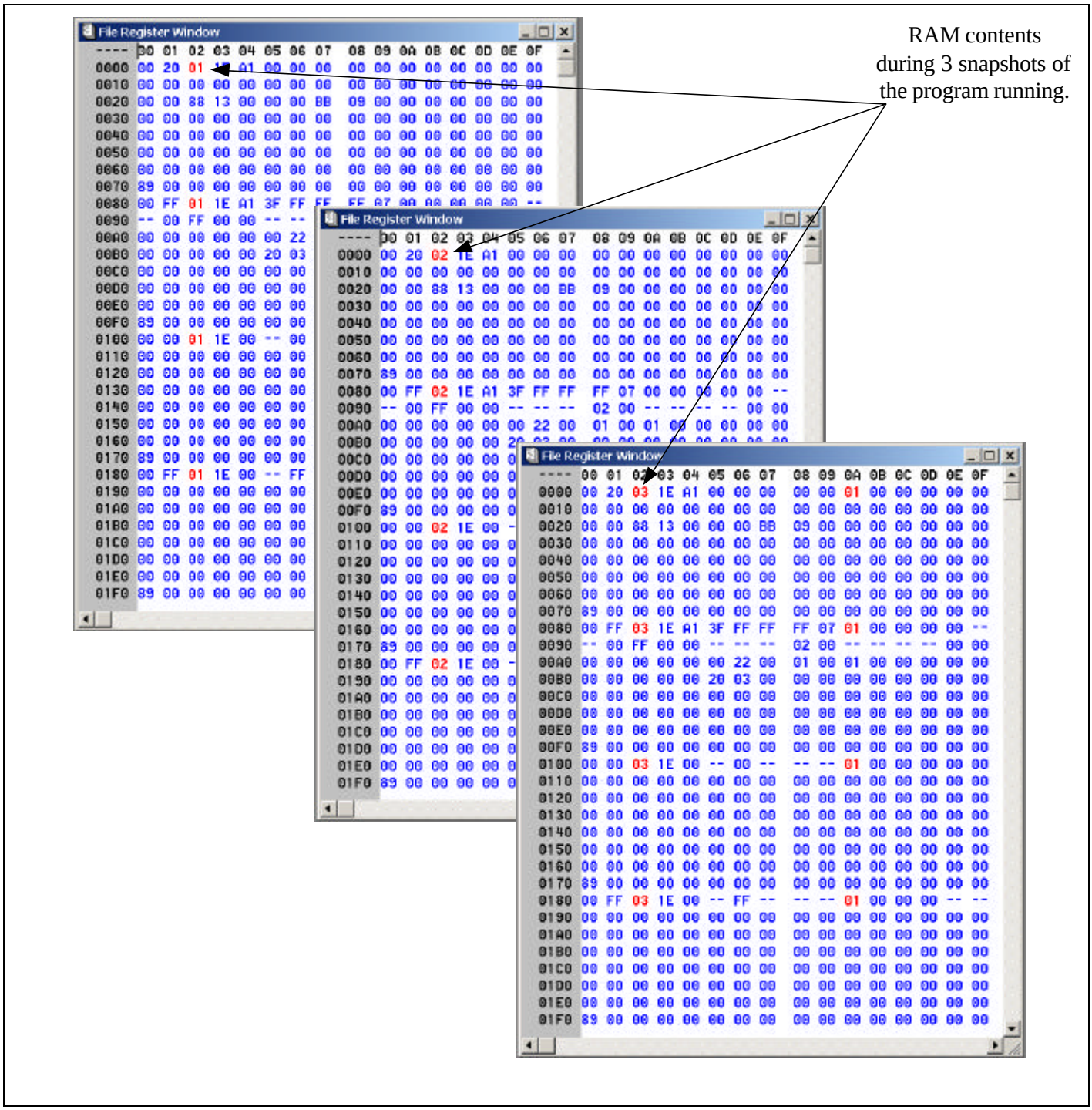

Figure 26. Three snapshots of the microcontroller RAM in MPLAB simulated slave program

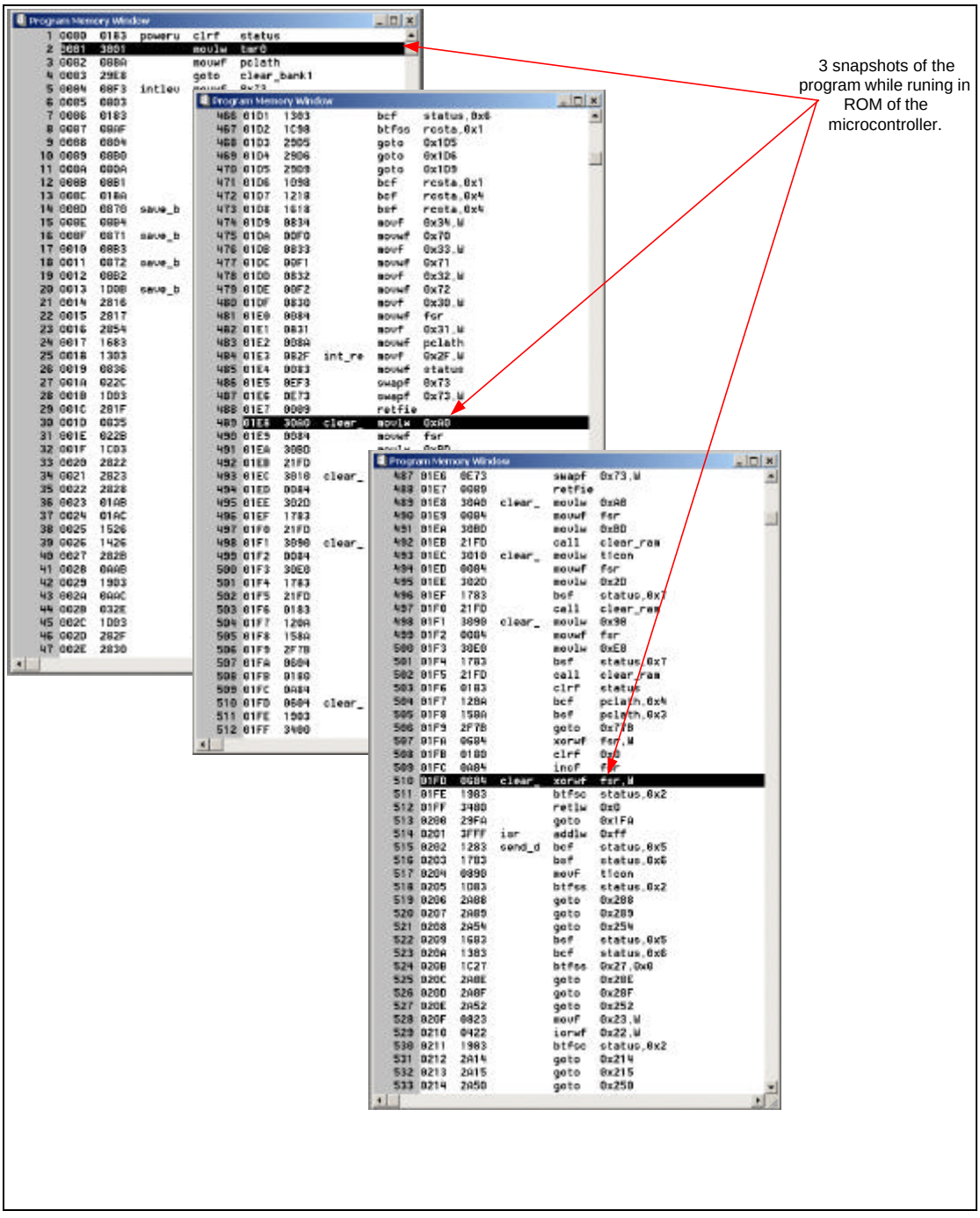

Figure 27. Three snapshots of the microcontroller ROM in MPLAB simulated slave program

The unavailability of a packet capturer and analyzer as the one shown in Figure 22,23 stopped the testing of the Bluetooth development module transmission. A second attempt to receive the signal via a spectrum analyzer with an antenna connected to it for 2.4 GHz was unsuccessful, due to the low power and the fast frequency hopping radio of the module. The single Bluetooth module interrupted testing the communication between master and slave; other Bluetooth products available in the labs can not receive from the development kit due to an undefined profile for the development kit.

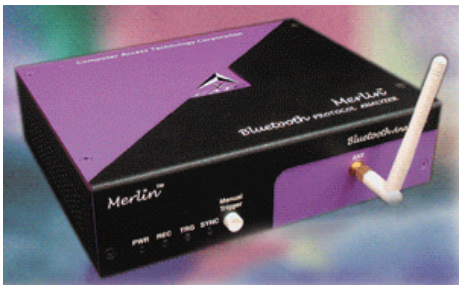

Figure 28. Merlin capturing Bluetooth packet analyzer by CATC (From Ref. [15].)

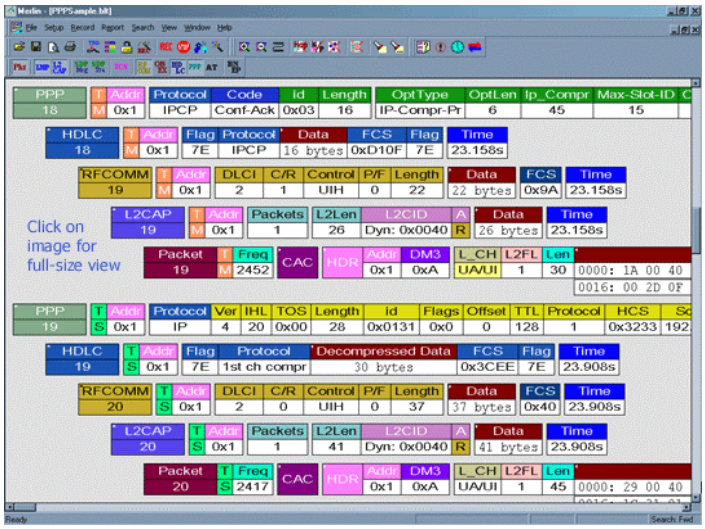

Figure 29. Bluetooth packet captured by Merlin protocol analyzer (From Ref. [15].)

In this chapter the prototype unit testing and tests result obtained were discussed. The next chapter covers conclusion and future improvement to the prototype unit.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

This chapter contains summary and conclusion. It concludes with suggested future improvement to the prototype unit.

A. SUMMARY

Bluetooth technology is focused on replacing cables used to connect different mobile devices. The technology is based on very small, low-cost, lightweight radios that easily form ad-hoc, secure links between various devices, allowing personal connectivity. The ability of Bluetooth to send and receive data can be beneficial in many areas; one of the areas is connecting sensors to a Bluetooth module and receiving that data in a remote or mobile unit enabled with another Bluetooth unit.

In this thesis a Generic Bluetooth Data unit has been constructed from available and low-cost components, Generality was in mind when this unit was built from both hardware and software prospective. The unit is capable of receiving a signal from any defined port of its microcontroller input ports, which it either digitizes or passes via a Bluetooth module. The unit is capable of receiving two analog inputs, one digital input and one SPI input; also after adding DAC to the support board it is capable of outputting an analog signal via two outputs. The software was coded in two parts. One is downloaded into the microcontroller memory and is called slave program. The program was tested by simulation, and worked successful. The second part of the software was coded and loaded on a PC where it needs an application that uses the defined API methods. Any users can build their own application using Java over the API and using the different classes as layers. The thesis used an Ericsson Bluetooth development kit, for research and students purposes. Other modules from different venders may be used as long as the HCI functions of that module are revised to match the functions in this thesis. If they are not matched, changes and modifications can be added to the code functions to match the new module.

B. FUTURE IMPROVEMENT

The term generic in the name of the thesis was taken into account when building the support card; in other words, the card is capable of having two analog inputs, two

analog outputs, two pulse width modulation outputs, one SPI input, one digital input and clock output. Improvements can be added by using all these inputs and outputs in different applications, such as adding more sensors or a Serial Peripheral Interface by connecting them to the specified pin in the 34-bin connection.

The slave program was coded in ANSI C in Appendix (A). It can be modified by adding more functions or eliminating some of the functions as required; the compiled version of the code in PIC assembly language is also in Appendix (B) for any user wanting to track or follow the logic of the program in registers. The master program is in Java in Appendix (C) and can be used by user applications by calling the functions defined in the API; the HCI functions is standardized according to Bluetooth specifications 1.0 and is suitable for an Ericsson development kit. Other venders have defined their functions with different names. When using any Bluetooth with this program the user must revise all the function names and match those names with the HCI functions for the appropriate products.

APPENDIX A – SLAVE PROGRAM IN ANSI C

The code is available in electronic format. To get a copy, contact:

Prof Xiaoping Yun at the Naval Post graduate School.

xyun@nps.navy.mil

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B – SLAVE PROGRAM IN MICROCONTROLLER ASSEMBLY

The code is available in electronic format. To get a copy, contact:

Prof Xiaoping Yun at the Naval Post graduate School.

xyun@nps.navy.mil

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C – MASTER PROGRAM IN JAVA

The code is available in electronic format. To get a copy, contact:

Prof Xiaoping Yun at the Naval Post graduate School.

xyun@nps.navy.mil

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. The official Bluetooth website, "*Specification of the Bluetooth System.*" [http://www.bluetooth.com/developer/specification/]. 22 February 2001.
2. Benson, David, *Easy PIC'n, A Beginner's Guide to Uusing the PIC Microcontroller*, Square One Electronics, Kelseyville, CA,1999.
3. Bray, J., and Sturman, Charles F. *Bluetooth connect without cables*, Prentice Hall, Inc., 2001.
4. Stallings, William, *Wireless Communications and Networks*, Prentice Hall, Inc., 2001.
5. Stevens, Roger, *Serial PIC'n, PIC Microcontroller Serial Communications*, Kelseyville, CA, May 1999.
6. Nilsson, P. and Brodin, J."*Implementing a Wirless I/O Unit Using Bluetooth*" .Department of Automatic Control, Lund Institute of Technology, Sweden, November 2000.
7. Miller, B. and Bisdikian, C. *Bluetooth Revealed*, Prentice Hall, Inc., 2001.
8. Muller, N. *Bluetooth Demystified*, McGraw-Hill Companies.,2001.
9. Microchip, *PIC16F87X Data Sheet*, Microchip Technology Inc., 2001.
10. Computer Access Technology Corporation ATC website, ["http://www.catc.com//products/merlin.html"]. 26 September 2002.
11. Michael, P., and Myke P. *Programming & Customizing PICmicro Microcontrollers*, McGraw-Hill, Companies., December 2000.
12. Bray, J., and Senese, B., and McNutt, G., and Munday, B., and Kammer, D. *Bluetooth Application Developer Guide*, Syngress Media Inc., December 2001.
13. Miller, M. *Discovering Bluetooth*, Sybex Inc. August 2001.
14. Dreamtech Software Team. *WAP, Bluetooth, and 3G Programming: Cracking the Code*. John Wiley & Sons Inc., November 2001.

15. The official Bluetooth website, "*Profiles of the Bluetooth System.*" [http://www.bluetooth.com/developer/specification/].22 February 2001.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Professor Xiaoping Yun, Code EC/Yx
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA

4. Professor Baer Wolfgang
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA

5. BGEN. Nasir Alsubaie
   PIC Manager
   Private Information Centre
   Kingdom of Saudia Arabia

6. COL. Khalid Alhabdan
   PIC Depoty Manager
   Private Information Centre
   Kingdom of Saudia Arabia

7. LtCdr Ali M Aljuaied
   Private Information Centre
   Kingdom Of Saudia Arabia

THIS PAGE INTENTIONALLY LEFT BLANK