MavBlue: A Bluetooth Development Kit for Undergraduate and Graduate Research and Education

Steven V. Case
Department of Computer and Information Sciences
Minnesota State University Mankato
steven.case@mnsu.edu

Abstract

The Bluetooth protocols have been developed to support wireless transmission within personal area networks; primarily serving as a cable replacement technology for mobile devices. Minnesota State University, Mankato has taken significant steps to incorporate wireless networking into the educational experience. The campus has initiatives in place to bring a complimentary range of wireless technologies to the campus. A primary motivation for the research identified in this paper is to develop tools and support that can address wireless networking from the PAN (personal area network) perspective. In addition, expanding the wireless initiative to include personal area network allows for research into the "last meter" problem, which has been identified by the Defense Advanced Research Project Agency (DARPA) as one of the most compelling challenges of the next decade (Rabaey, et al., 2000).

For the purpose of this research, Personal Area Networks (PANs) are networks focused on connectivity for the personal use of a single individual. Consequently, PAN solutions are focused more on interconnection of personal devices than on sharing resources, which has been the traditional motivation for other network technologies. Today, the most prominent standard in development that can address the unique requirements of Personal Area Networks is the Bluetooth standard.

Incorporating Bluetooth technology, at its current stage of development, into course work and undergraduate research on wireless data communications imposes certain difficulties. To begin with, Bluetooth-compliant radios and protocol stacks are difficult to obtain and are rather expensive (present pricing for developer's kits ranges as high as $3,000 per radio). Rather than waiting for such equipment to become more readily available, a set of hardware and software tools are being developed that are intended for use within undergraduate research initiatives. This paper details the current design and development of this Bluetooth development environment, which we call MavBlue.

Introduction

Minnesota State University Mankato has taken significant steps to incorporate wireless networking into the educational experience. The campus has initiatives in place to bring a complimentary range of wireless technologies to the campus. Current infrastructure in place at MSU includes WAP, SMS, IEEE 802.11b, and LMDS technology. These

technologies provide the campus community with complementary wireless networking solutions for the WAN (wide area), MAN (metropolitan area), and LAN (local area) domains.

A primary motivation for the research identified in this paper is to develop tools and support that can address wireless networking from the PAN (personal area) perspective and, thus, complete the spectrum of wireless solutions available to the University. As identified by Held, this collection of wireless protocols provides reasonably complete coverage for the leading wireless data communication standards emerging and evolving today [5]. In addition, expanding the wireless initiative to include personal area network allows for research into the "last meter" problem, which has been identified by the Defense Advanced Research Project Agency (DARPA) as one of the most compelling challenges of the next decade [8].

For the purpose of this research, Personal Area Networks (PANs) are networks focused on connectivity for the personal use of a single individual. Consequently, PAN solutions are focused more on interconnection of personal devices than on sharing resources, which has been the traditional motivation for other network technologies. Today, the most prominent standard in development that can address the unique requirements of Personal Area Networks is the Bluetooth standard.

Bluetooth Protocol Stack Overview

Volume 1 of the *Specifications of the Bluetooth System* specifies the protocol stack of Bluetooth, which is shown in Figure 1 [1]. The layers of this protocol stack can be summarized as follows:

- The RF layer, specifying the radio parameters, most closely maps to the physical layers of the International Standards Organization (ISO) model and the IEEE 802 standards.

- The baseband layer, specifying the lower-level operations at the bit and packet levels, most closely maps to the medium access layer of the IEEE 802 standards or the lowest levels of the data link layer of the ISO model. The baseband layer is responsible for forward error correction operations, encryption, circular redundancy check (CRC) calculations, and the automatic-repeat-request (ARQ) protocol.

- The link manager layer most closely maps to the middle levels of the data link layer of the ISO model. The link manager also maps to the upper levels of the medium access layer and the lower levels of the logic link layer when compared to the IEEE 802 standards. The link manager is responsible for specifying connection establishment and release, authentication, connection and release of synchronous connection-oriented (SCO) and asynchronous connectionless (ACL) channels, traffic scheduling, link supervision, and power management tasks.

- The logical link control and adaptation protocol (L2CAP) layer maps to the logical link layer of the IEEE 802 standards. Similarly, the L2CAP corresponds to the service access points of the ISO model. This layer forms an interface between standard data transport protocols and the Bluetooth protocol.

Above the L2CAP layer are a variety of protocols that most closely map to presentation and application layer protocols in the ISO model. For example, the RFCOMM protocol is intended to provide emulation for standard RS-232 cabling whereas the service discovery protocol (SDP) enables one Bluetooth unit to identify the capabilities of other Bluetooth devices within its transmission range.

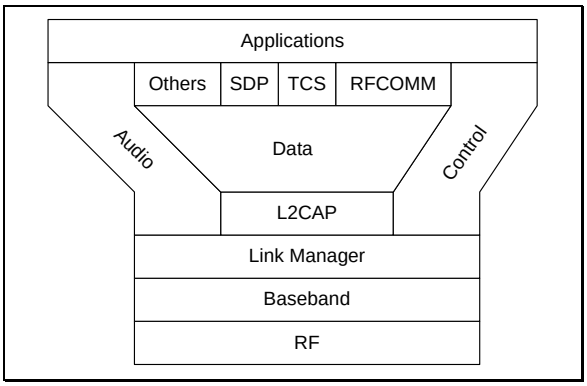

Figure 1. The Bluetooth Protocol Stack

The RF Layer

The RF layer of the Bluetooth protocol specification provides for the physical transmission of bits. This is the lowest layer of the protocol stack and, consequently, provides the least interest to the research detailed in this paper. Nevertheless, some details of the RF layer are worthy of discussion; particularly since the design of the RF layer imposes certain data rate limitations on the use of Bluetooth technology.

The goal of Bluetooth is to devise a ubiquitous, ad hoc radio system. Therefore the choice of RF spectrum to use is constrained by government regulations on the use of the RF spectrum. In order for Bluetooth to become ubiquitous, it must incorporate a radio solution that does not require any special licensing. Therefore, the radio must operate in an unlicensed spectrum, typically referred to as the Industrial, Scientific, Medical (ISM) band. The Bluetooth RF layer operates in the ISM band located at 2.5 GHz.

The selection of the 2.4 GHz ISM band provides approximately 80 MHz of bandwidth for the RF layer. As with any data network, it is necessary to provide multiple, concurrent access to this physical media. The Bluetooth solution is to divide the spectrum into separate RF channels, each with a 1-MHz allocation, and to use frequency hopping within the available channels.

The Baseband Layer

The baseband protocol is responsible for establishing the physical link between two Bluetooth radios. The Bluetooth standard partitions radios as masters and slaves. As such, the baseband layer provides a variety of services, including connection and connectionless services, error detection and correction, flow control, hop management, address management, encryption, and authentication. Clearly, the baseband layer provides many of the same capabilities as the data link layer in the ISO protocol stack and the same services as the media access control (MAC) layer in the IEEE 802 standards.

The Bluetooth protocol at the baseband layer uses a combination of frequency modulation and time division modulation. To begin with, the radio spectrum is partitioned into 79 or 23 RF channels, depending on the national licensing issues. The system then uses a pseudo-random frequency hopping sequence to transmit data using the available physical channels. The hopping sequence allows logical channels to be constructed from the physical channels, with each logical channel using a unique hopping sequence. Each logical channel has one master radio and one or more slave radios. The master radio's clock and address are used to uniquely select a hopping sequence. Each logical channel is called a *piconet*. Multiple piconets with overlapping coverage areas form a *scatternet*. Figure 2 provides an illustration of possible example configurations of master and slave radios into a piconet and a scatternet. In the figure, the three notebook computers have master radios and the phone and printers have slave radios. Three piconets exist. However, two of the piconets have overlapping coverage areas. The printer that exists within the overlapping coverage enables two of the piconets to combine and form a scatternet.

It is important to realize that a radio cannot serve as the master in more than one piconet as there would be no way to create a unique hopping sequence for each of the piconets.

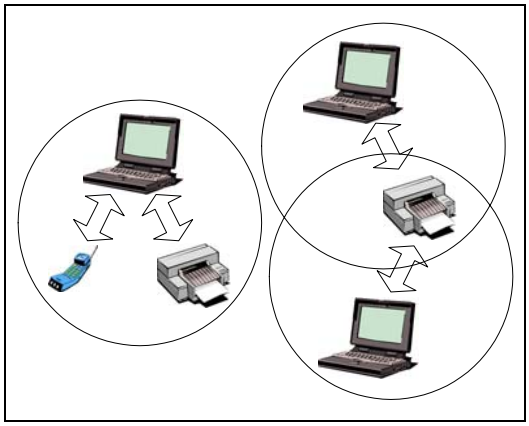

Figure 2. Bluetooth Piconet (left) and Scatternet (right) Configurations

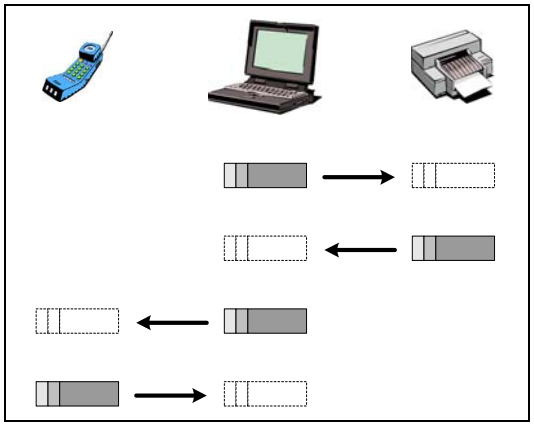

Figure 3. Bluetooth's use of Time Division Duplex and Its Timing

Within each logical channel, time division multiplexing is used as each channel is divided into time slots of 625-μs duration. Time slots are numbered using a 27-bit sequence number, thus providing a cycle length of 2^{27}. Finally, alternating transmission direction supports duplex operation. The master transmits to one (or more) slave radios on the even-numbered slots and then the slave addressed during that slot can transmit to the master on the next slot. Finally, transmission is packet-based with packet lengths of up to five time slots.

The concept of time division duplex and its timing is illustrated in Figure 3. In the illustration, a master radio (the notebook computer) communicates to two slave radios within its piconet. Four time slots are illustrated. The first two slots allow the master radio to communicate to the slave radio in the printer. In the third and fourth time slots, the master radio communicates with the slave radio in the wireless phone.

The Link Manager Layer

The link manager layer implements the Bluetooth link manager protocol (LMP), the next level up from the baseband protocol. The link manager layer is used for link set-up, security and control. Thus, the link manager creates connections between the master and slave. In addition, the link manager is responsible for negotiating parameters for data encryption. The Bluetooth protocols allow negotiation of encryption keys, key size, and the dynamic enabling and disabling of data encryption, polling intervals, and power management parameters.

Other than handling link set-up, perhaps the most critical responsibility of the link manager layer is to manage the various states in which each radio can operate. Many of the operational states are intended to provide extremely low-power modes of operation in order to maximize battery life for mobile devices.

The initial (or default) state of a Bluetooth radio on power-up is *standby*. This state operates the radio in low-power mode with only the native clock running. In the *connected* state, a connection has been established and packets may be exchanged between the master and the slave radios. To transition to a *connected* state from a *standby* state, the radio must follow certain procedures as defined in the substates *page*, *page scan*, *inquiry*, and *inquiry scan* [12][13].

The Logical Link Control and Adaptation (L2CAP) Layer

The L2CAP layer most closely resembles the data link layer of the OSI seven layer model. The L2CAP layer is responsible for protocol multiplexing, segmentation and reassembly of application level packets, and provides additional QoS capabilities.

The baseband protocol allows for the reliable delivery of up to 2,745 bits of user data; just 383 bytes of data. The L2CAP layer provides for reliable delivery of application packets up to 64KB in size. Clearly, in order to do so, the L2CAP layer must manage the segmentation and reassembly of these large packets into 383 byte packets.

In addition to segmentation and reassembly of application level packets, L2CAP provides support for another layer of logical channels within the piconet's logical channel. The L2CAP logical channels are analogous to sockets within the Internet's transport layer.

Finally, the L2CAP layer provides additional QoS support such as negotiation of flow specification (similar to that specified in RFC 1363), which is exchanged with the remote device during channel configuration.

Java and JSR-82

Our implementation of the MavBlue software have been developed in Java. The decision to use Java was based on two primary considerations, portability and ease of learning. The introductory programming courses at Minnesota State University Mankato are Java based. Therefore, by implementing th MavBlue environment in Java, undergraduate majors are able to become involved in the research once they have completed their CS1 and CS2 courses. In addition, this approach allows networking of a collection of heterogeneous nodes without undue effort to port the software to those heterogeneous environments.

The implementation requires host platforms to support either J2ME or J2SE with javax.com. The primary development is done using J2SE on Windows-based workstations. However, to the extent possible, the software is also ported to the Imsys platform, which is based on J2ME. Details on J2ME are available in [11] and [9].

When the development of the MavBlue project was initiated, no standard Bluetooth APIs existed. The MavBlue project proceeded to create a project-unique API for each of the Bluetooth protocols. Each protocol was implemented as a package, all contained within the edu.mnsu.bluetooth domain. For example, the Bluetooth L2CAP protocol was implemented as the edu.mnsu.bluetooth.l2cap package.

Since the start of the MavBlue project, a standard Java API for Bluetooth has been developed. The standard was developed by the Java Community Process (JCP) and is available as JSR-82. The MavBlue project is now in the process of revising the project's implementations in order to be consistent with JSR-82. However, it is significant to note that JSR-82 was developed for J2ME, not for J2SE. As such, JSR-82 is heavily based on the generic communication framework (GCF) used by J2ME to perform input and output. Unfortunately, the J2SE environment does not support GCF. As a result, Bluetooth channels within the MavBlue environment are encapsulated as a MavBlue specific class rather than using the JSR-82 model. Details on JSR-82 are available in [6] and [5]

MavBlue Development Kit Components

The MavBlue development kit consists of an integrated set of hardware and software that provides students and faculty with the ability to perform research into the application of Bluetooth technology. The development kit consists of three distinct components:

1. The MavBlue Bluetooth Protocol Stack. This is a Java implementation of the Bluetooth protocols. At present, the HCI, L2CAP, and SDP protocols are supported.

2. The MavBlue Bluetooth Module. This is a Blueooth adapter designed to interact with a host computer via the HCI UART interface. The current MavBlue module is available in two configurations: a RS-232 module that uses a standard DB-9 connector and a SIMM module for embedded environments.

3. The MavBlue Bluetooth User Interface. This is a graphical user interface to configure the protocol stack and to configure a Bluetooth piconet. The MavBlue Bluetooth user interface serves as the JSR-82 Bluetooth Control Center (BCC).

The remaining subsections provide additional details on these three components of the MavBlue development kit.

MavBlue Bluetooth Protocol Stack

Allowing undergraduate and graduate students to perform research into wireless communication requires the students to have access to the underlying network protocols. In the case of Bluetooth, as a minimum, source code to the implementation of the HCI and L2CAP layers is required. In our specific environment, the implementation needed to be portable to the Windows, TINI, and SNAP environments.

No publicly available implementation was located and licenses for commercially developed stacks were prohibitively expensive. Consequently, a Java-based implementation of the HCI and L2CAP layers was developed as part of a related research project at MSU. The source code for the implementation is available from the author in order to encourage further research into Bluetooth and personal area networking.

The implementation of the Bluetooth protocol stack was developed with the objective of supporting real-time, embedded systems. As such, the development followed the methodology and techniques established by Bruce Douglass for using UML to develop efficient objects for real-time, embedded systems [2]. Each protocol layer was encapsulated into its own package following the *Microkernel Architecture Pattern* identified by Douglass as a common layered architecture for communication protocols. Figure 4 uses UML notation to provide a static structure diagram of the architecture used to implement the HCI layer. Similarly, Figure 5 provides the static structure diagram for the L2CAP layer.

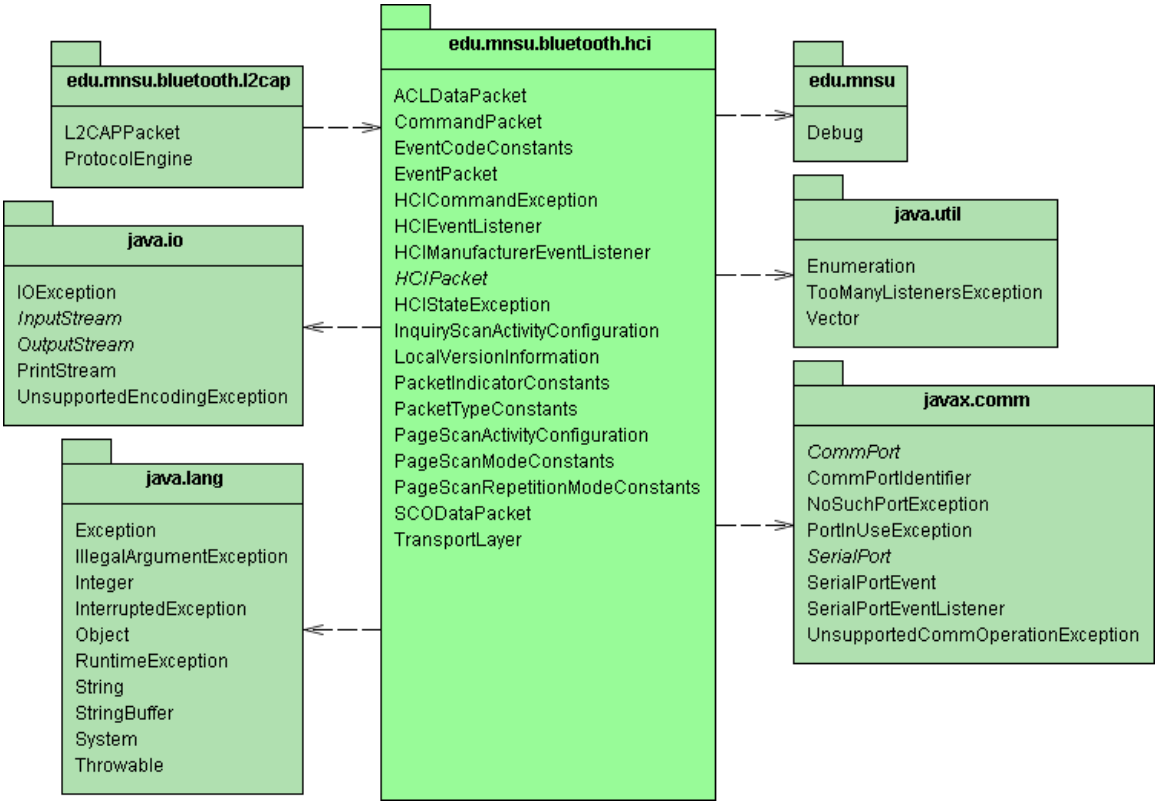

Figure 4. UML Structure of the HCI Package

The implementation of the protocol layers was performed using the *iterative prototyping* methodology identified by Douglass. Iterative prototyping is an implementation strategy that relies on implementing vertical slices through the layered architecture. This strategy is proven to be less risky and less expensive than the traditional approach of simply translating the layered architecture into a layered implementation [2]. This approach is also proving effective for student research as it allows students and faculty to focus on the functionality required to support their research.

A significant portion of the design and implementation of the HCI and L2CAP layers addresses the functionality required to discover and connect Bluetooth devices into a piconet (i.e. the inquiry and paging processes) as well as the functionality required to establish ACL connections between Bluetooth devices.

Once the piconet has been established, the system is assumed to be in a state that requires real-time performance. Within this state, the protocol stack must provide real-time services for sending and receiving data between Bluetooth devices. The MavBlue implementation uses the *Channel* class to provide the abstraction of an ACL connection. As such, the *Channel* class provides the application-level interface to the L2CAP layer for sending and receiving application-level data. The *Channel* object provides a *write()* method for sending data and a *read()* method for receiving data. The design and

implementation of the *write()* and *read()* methods used the guidelines established by Douglass to design a solution suitable for real-time systems.

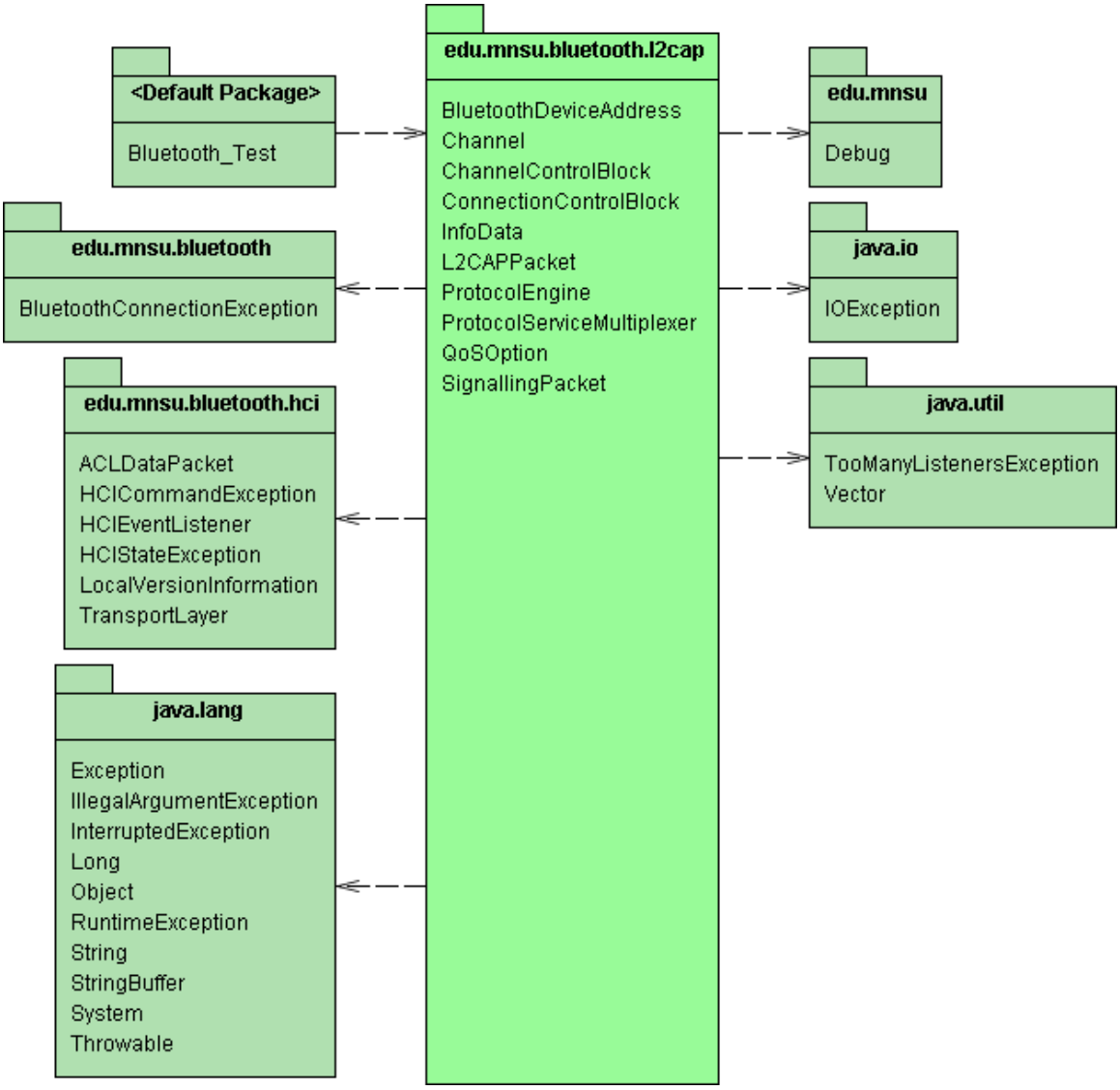

Figure 5. UML Structure of the L2CAP Package

MavBlue Bluetooth Module

Although commercial Bluetooth adapters are now available at reasonably affordable prices, such adapters are limited in usefulness for research. The problem with these adapters is that the end-user is not provided with low-level access to the adapter and the Bluetooth protocols. Instead, it is assumed that the user will access the Bluetooth device in a manner consistent with one of the Bluetooth SIG's profiles. This almost always

implies that the user sees the Bluetooth device as either a virtual serial port or as a virtual Ethernet port.

Bluetooth development kits provide developers and researchers with full access to the Bluetooth hardware and Bluetooth protocols. Unfortunately, the commercially available development kits are too expensive for most undergraduate and graduate research projects. Typical pricing for commercial development kits range from approximately $1500 (for hardware and software, but no source code) to $12000 (for embedded development kits with hardware and software).

The MavBlue development environment includes Bluetooth adapters developed at Minnesota State University. The Bluetooth adapters are based on a common schematic that relies on the HCI UART interface. This interface was selected as it is easily adapted to RS-232 interfaces. This allows the adapter's design to be adjusted to various physical layouts depending upon the needs of the researcher. The current schematic for the MavBlue Bluetooth module is provided in Figure 6.

The schematic for the MavBlue module is based on a design from NSM Technologies. The adapter was originally developed to support the Maxell MBM02 Bluetooth chip. However, production versions of that chip are no longer available. The schematic has been modified to now use the Bluetronics Bluetooth module.

The MavBlue module schematic has currently been adapted to two different physical layouts. One layout is based on low-cost production and uses a standard 9-pin RS-232 interface to connect to the host computer (see Figure 7). The second layout is based on a 72-pin SIMM form factor and allows the module to be integrated with TINI and SNAP embedded platforms (see Figure 8).

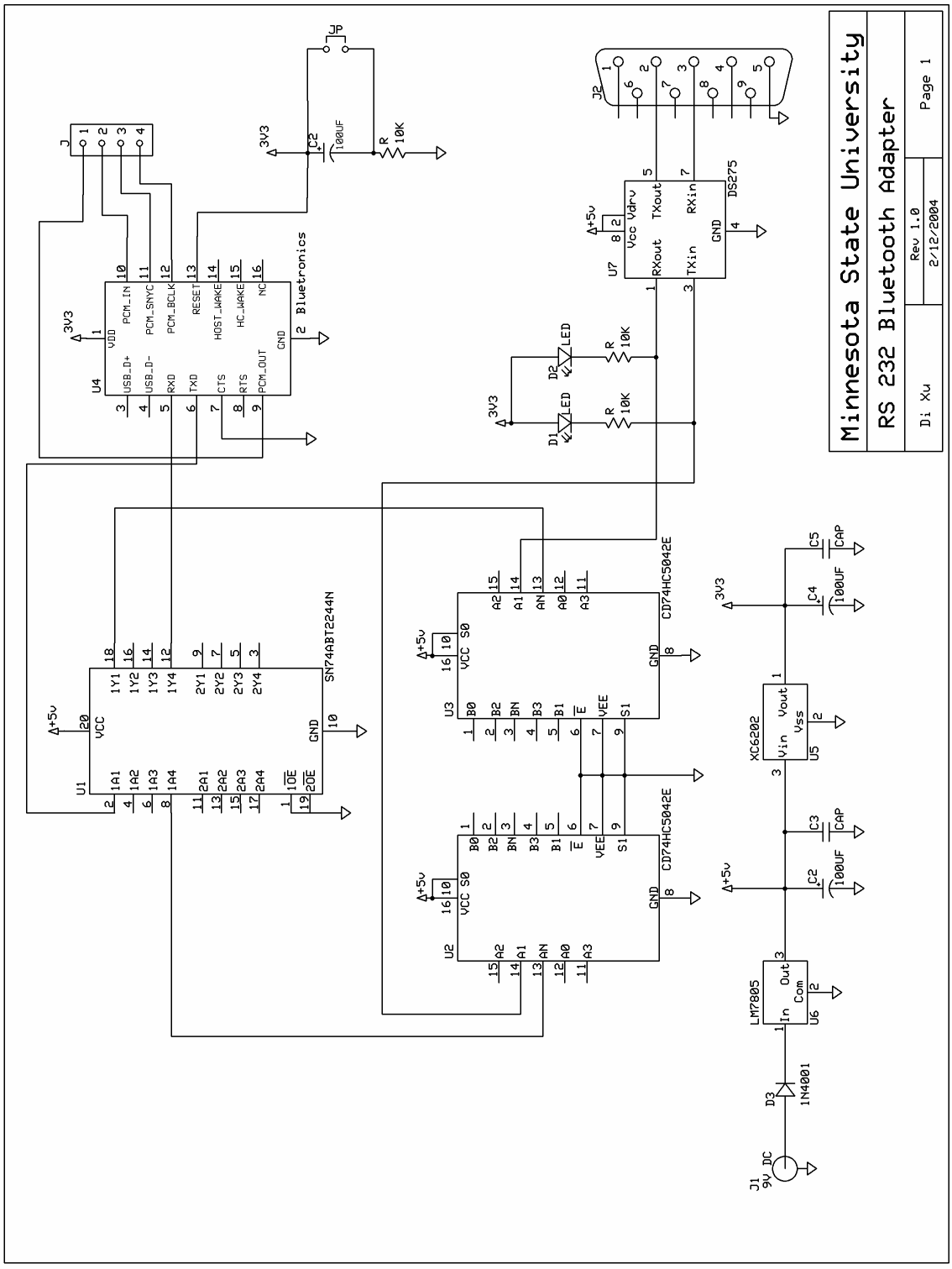

Figure 6. MavBlue Bluetooth Module Schematic

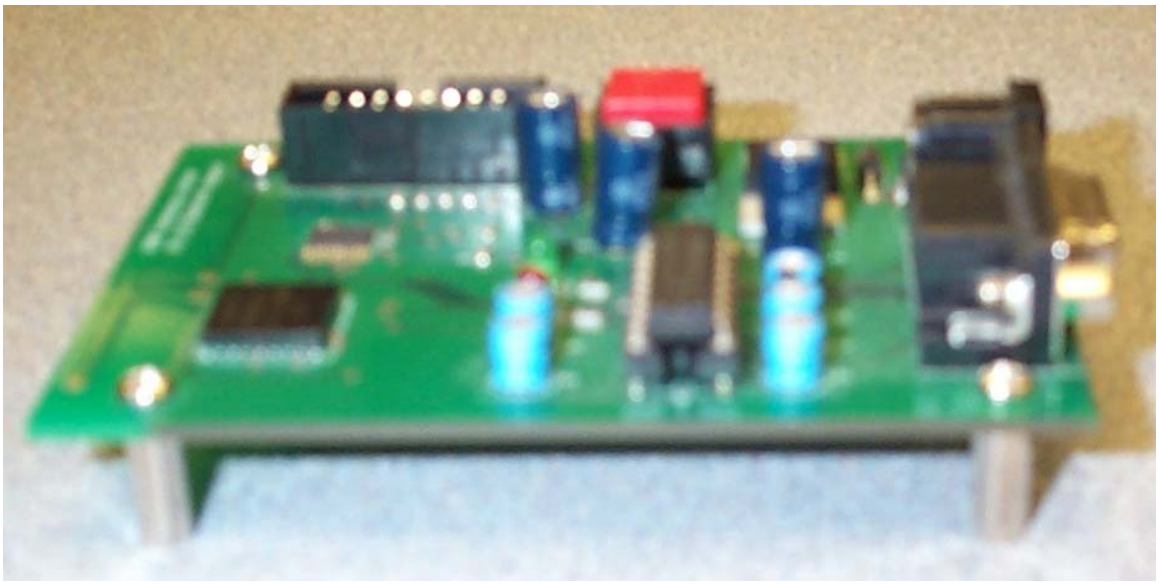

Figure 7: An Example 9-pin RS-232 Bluetooth Configuration

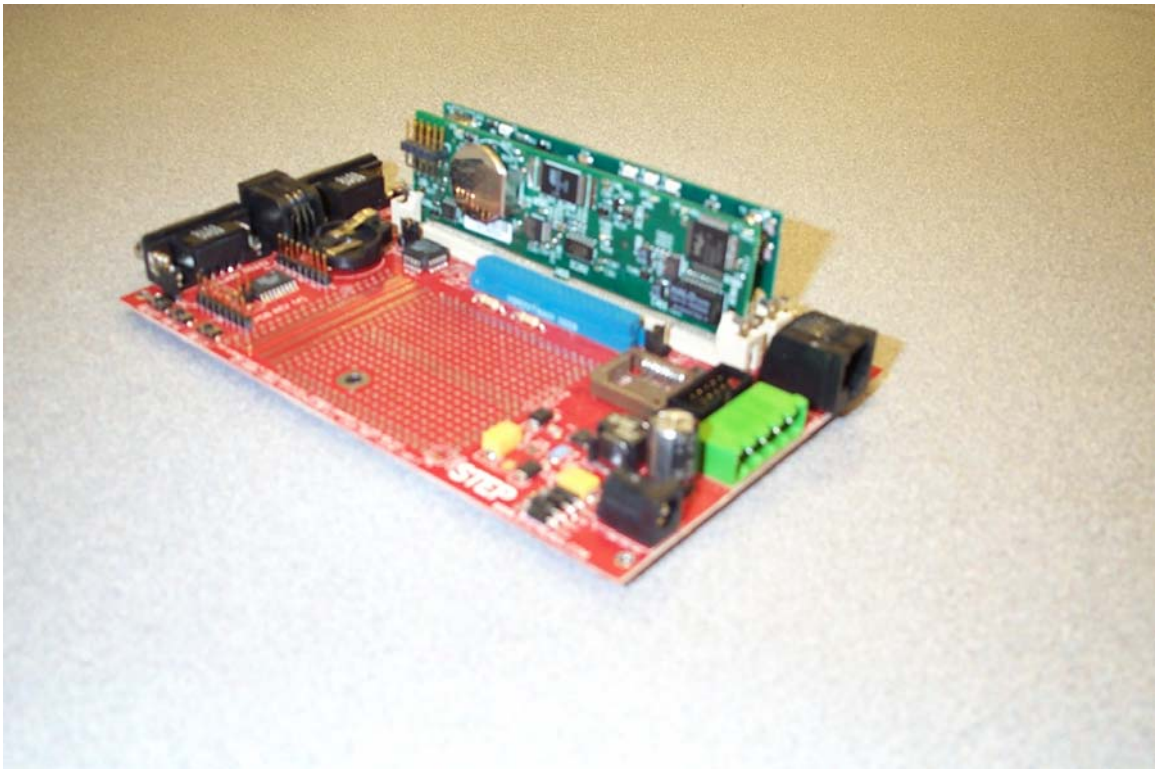

Figure 8: An Embedded SNAP Development Platform

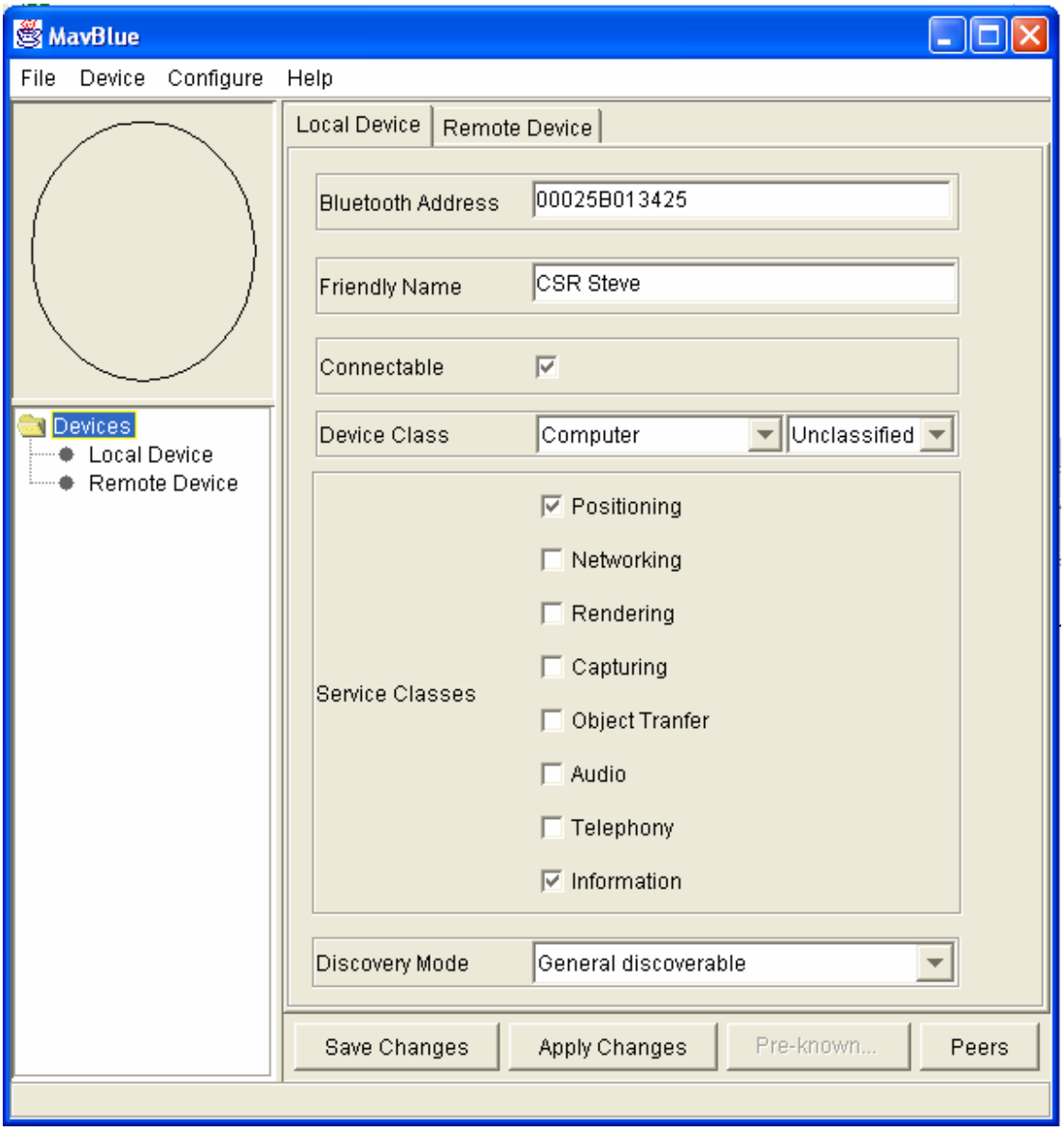

Figure 9. MavBlue Primary User Interface

MavBlue User Interface

The MavBlue user interface is illustrated in Figure 9. The user interface is divided into three primary regions; a graphical view, a tree view, and a tabbed view. Each region provides a differing perspective on the Bluetooth network. The graphical view provides a visual indication of the devices participating in the same piconet as the local device. In the example, the piconet is currently empty. The tree view provides a view of the local and remote devices. It is the intent of the tree view to list the devices along with the services provided by those devices. However, this functionality is not yet implemented.

The tabbed view provides a detailed listing of the local device and all known remote devices. In addition to providing the user with information pertaining to known devices and the current configuration of the piconet, the MavBlue user interface also provide the functionality of the JSR-82 Bluetooth Control Center. Additional details on the design and implementation of the MavBlue user interface can be found in [7]

Conclusions and Future Enhancements

The MavBlue development kit provides students with complete accessibility to Bluetooth hardware and software in order to Bluetooth technology. In order to be of greater benefit to future researchers, the MavBlue development kit should be extended with the following capabilities:

- Dynamic invocation of applications from within the MavBlue environment. The MavBlue environment should allow the developer to run Bluetooth applications from within the MavBlue environment. The applications would be invoked on the local host computer and becomes services available from the local device.

- Simulated radios using multicast sockets. In many research activities, it is necessary to simulate the existence of additional devices. Copies of the MavBlue environment could be initiated on remote workstations that are networked on a standard TCP/IP network. The Bluetooth protocol stack can simulate Bluetooth packet communication by transmitting baseband packets as UDP/IP multicast messages.

- The current MavBlue environment are limited to the HCI, L2CAP, and SDP protocols. The MavBlue environment should be extended to support all of the Bluetooth protocols. This will enable the user interface to automatically configure the local device to a specific Bluetooth profile.

- The current MavBlue implementation uses the standard Java XML bindings to access the persistent storage. The Java XML binding is specific to J2SE and J2EE, but not supported for J2ME as the binding is too memory intensive for J2ME devices. The MavBlue user interface should be modified to use a less resource intensive XML binding such as MinML.

- The current MavBlue Bluetooth module is limited to UART communication with the host computer. On many architectures, this effectively limits the host communication to 128Kbps and, therefore, does not allow the host computer to fully utilize the available Bluetooth data rate. The MavBlue Bluetooth module should be enhanced to support both UART and USB communication with the host computer.

References

[1] Bluetooth SIG. (2001) *Specification of the Bluetooth System – Version 1.1*, Volumes 1 & 2. February 2001.

[2] Douglass, B. (2000). *Real-Time UML: developing efficient objects for embedded systems.* Reading, MA: Addison-Wesley Longman, Inc.

[3] D. Geary. (1999) *Graphic Java 2, Volume 2: Swing (3rd edition)*. Prentice Hall PTR, 1999.

[4] Held, G. (2001). "Data Over Wireless Networks". McGraw-Hill. New York, NY, 2001

[5] B. Hopkins, & R. Antony. (2003) *Bluetooth for Java*. New York: Springer-Verlag, 2003.

[6] C. Kumar, P. Kline, & T. Thompson. (2004) *Bluetooth Application Programming with the Java APIs*. San Francisco: Morgan Kaufmann Publishers, 2004.

[7] Loi, P., & Case, S. (2004). *Development of the User Interface for the MavBlue Development Environment*. Midwest Instruction and Computing Symposium, April 16-17, 2004. Morris, MN.

[8] Rabaey, J., Ammer, M., da Silva, J., Patel, D., & Roundy, S. (2000). "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking". Computer. July 2000. pp. 42-48.

[9] R. Riggs, A. Taivalsaari, & M. VandenBrink. (2001) *Programming Wireless Devices with the Java 2 Platform, Micro Edition*. Boston: Addison-Wesley, 2001.

[10] M. Robinson, & P. Vorobiev. (2003) *Swing, second edition*. Greenwich, CT: Manning Publications Co., 2003.

[11] K. Topley. (2002) *J2ME in a Nutshell*. Sebastopol: O'Reilly, 2002.

[12] J. Y. Wilson, J. A. Kronz. (2000) *Inside Bluetooth Part I*. Dr. Dobb's Journal. 25(3). pp. 62-70, 2000.

[13] J. Y. Wilson, J. A. Kronz. *Inside Bluetooth Part II*. Dr. Dobb's Journal. 25(4). pp. 58-64, 2000.