A BLUETOOTH-BASED WIRELESS NETWORK FOR DISTRIBUTED SHIPBOARD MONITORING AND CONTROL SYSTEMS

MIDN Kenneth J. Hoover
Professor Antal Sarkady
CDR Charles B. Cameron, USN

United States Naval Academy
105 Maryland Avenue
Annapolis, MD 21402

Henrey Whitesel

Naval Surface Warfare Center Philadelphia
Machinery Research and Development Directorate
Philadelphia, PA

Abstract: A Bluetooth based "power node" has been developed for monitoring and controlling power systems on board US Navy Vessels. For this application a "power node" is defined as an electronic system, which collects information from many sensors and makes appropriate control decisions based on the occurrence of well-defined events. Bluetooth is a low cost, low power wireless standard, which is incorporated on each "power node." This wireless standard allows networking of several "power nodes." An important advantage of this system is that it can be configured for many shipboard applications. The Bluetooth standard uses a spread-spectrum modulation scheme that allows reliable communication between "power nodes" within several sub-networks (piconets) in the same physical location. A robust wireless network that maintains reliable connectivity among nodes even when the communication channels are altered by the opening and closing of watertight doors has been designed. Versatility is achieved by the use of the Motorola MC68HC908JB8 microcontroller in the "power node." This microcontroller is in-circuit-programmable, allowing rapid software changes. The goal of this project is to have tested a working prototype network consisting of several "power nodes" on the *ex-USS America*.

Key Words: Bluetooth; frequency hopping spread spectrum (FHSS); intersymbol interference (ISI); piconet; scatternet; bit errors; bit error rate; attenuation.

Introduction: The Navy is always looking for efficient ways to improve its role in protecting our country. One method to accomplish this task is the reduction of ships' crew sizes. A military made up of an all-volunteer force is very expensive. A large amount of money is spent every year on the training of and salary for sailors and officers. Reducing crew sizes would greatly reduce the cost of the Navy as a whole, provided that this could be done without a loss of efficiency.

Increased automation onboard ship is one way to make smaller crew sizes possible. Many man-hours are consumed by sailors who walk around the ship taking readings and recording the data into logbooks. Much of this time could be eliminated through the use of computer

networks for centralized data logging and display. The use of such a network could also aid in damage control, thereby increasing the safety of the ship.

One such method of increasing automation onboard ship is the use of a wireless network. While a computer network such as the one described above could be implemented using a wired network, wires add a great deal of weight to the ship and decrease the structural and watertight integrity because of the holes which need to be cut into the bulkheads for the wires to pass from compartment to compartment. Along with the structural advantages of a wireless network there are also financial advantages. Both Maintenance and upgrades are less expensive when there are fewer wires to deal with because there are no installation costs. A wireless network is also easily configured and reconfigured: one simply needs to download new software to reconfigure an entire network. The advantages of a wireless network are increased automation, increased structural integrity, decreased weight, decreased installation costs (in both time and money), and increased reconfigurability. The purpose of research described below was to determine the feasibility of a Bluetooth network in a ship-board environment.

This research project uses *radio frequency* (RF) attenuation data obtained by a previous Trident Scholar at the U.S. Naval Academy, ENS Daniel R. J. Estes, completed in May 2001. He performed RF transmission tests using a single sinusoidal sources ranging from 0.8–2.5GHz in frequency on the *ex-USS America*, *USS Ross, USS Carr, USS Leyte Gulf,* and *USS Oscar Austin*[1]. During his research ENS Estes measured the signal attenuation through closed hatches using five different geometries for the transmitter and receiver: open hatch in the direct path from transmitter to receiver, open hatch not in the direct path from transmitter to receiver, closed hatches almost in the direct path from transmitter to receiver, a bulkhead with no open hatches, and closed hatches not in the direct path from transmitter to receiver. He concluded that the maximum RF signal attenuation was approximately 25 dB[1]. Bluetooth devices currently have a receiver sensitivity of -75dBm. This means that using the lowest power Bluetooth transmitter (0dBm) in an environment with no external noise a signal with 75dB of signal path loss is still receivable. This is the equivalent of passing through three closed hatches onboard a ship. It is because of this characteristic of Bluetooth that we believe Bluetooth is capable of operating in a shipboard environment.

ENS Estes concluded from his research that wireless transmission is possible onboard ships. However, he used only a single unmodulated sinusoidal carrier frequency in his tests. A modulated signal requires much more bandwidth than an unmodulated carrier.

For example look at figure 1. This figure shows four plots. The top two plots are the time-domain representations of a sinusoidal signal and a square pulse. The bottom two pictures are the frequency-domain representations of these two signals. The sinusoidal signal occupies only one discrete frequency, whereas the square pulse requires considerably more bandwidth because it is made up of many harmonically related sinusoids (a *Fourier Series*) which are responsible for the sharp edges of the pulse. Though actual digital transmission signals do not use square pulses, because of the large bandwidth associated with them, the signals do still occupy much more bandwidth than a single sinusoid. Our research and another series of shipboard wireless testing are therefore intended to extend ENS Estes' work to a broader frequency range.

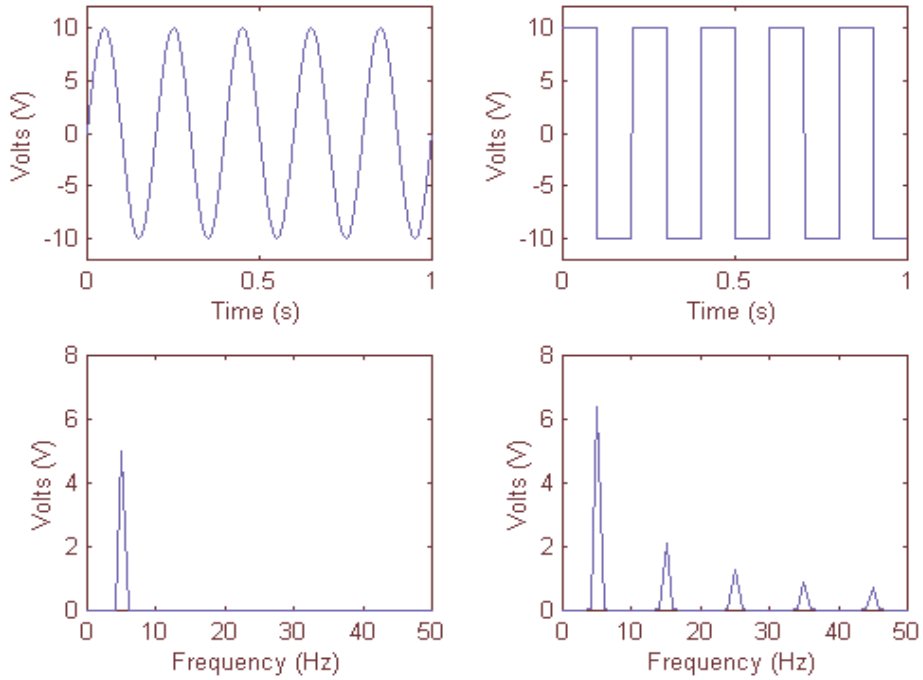

Figure 1: Sinusoid and Pulse with Fourier Transforms

Because signal interference may affect the phase component of a signal along with its magnitude, it is important to conduct a series of tests similar to those done by ENS Estes but using data transmission instead of a single sinusoid. When data are transmitted and attenuated through objects such as bulkheads and hatches, *Intersymbol Interference* (ISI) may occur.

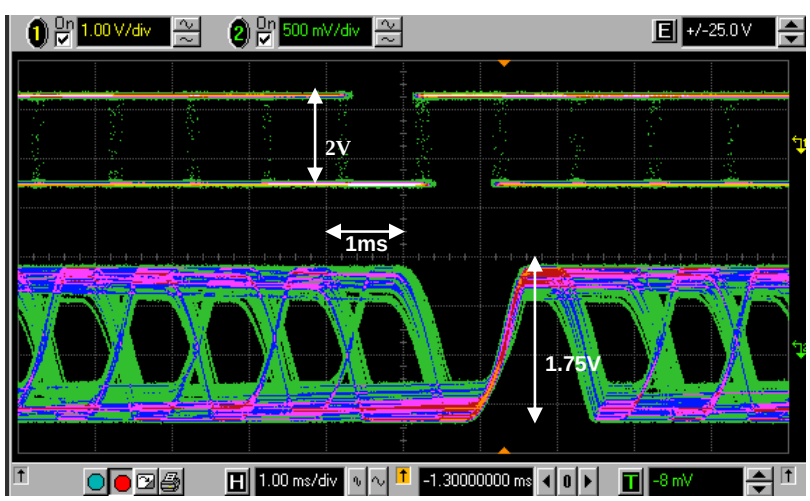

Figure 2: Intersymbol Interference

ISI occurs when a signal comprised of distinct time–domain symbols or pulses is distorted such that those pulses begin to "bleed" into one another. This "bleeding" leads to receiver errors in the reconstruction of binary digits (bits). Figure 2 shows an example of the effects

of ISI. The figure, taken from an oscilloscope, shows the original signal and the signal after passing it through a filter which caused ISI to occur. This figure shows many traces of the same signals. The top signal is the original signal. It is a series of random square pulses two volts in amplitude. By passing the initial top signal through a filter the bottom signal was created. The filter attenuated the signal and created some ISI within the signal. From this figure you can see that ISI can be a problem for digital communication because it distorts waveforms.

Having seen the affects of ISI on the signal itself, what are the effects of ISI on the actual data sent? The most likely effect of ISI on digital data is single or multiple *bit errors*. Small numbers of bit errors have little or no effect on some types of data (e.g. voice and video), however a single bit error in other data (e.g. files and control information) may require retransmission of that data. This ongoing research project is investigating the frequency of bit errors for a Bluetooth wireless network arranged in geometries similar to those tested by ENS Estes and using several different data transmission rates. The practicality of a shipboard Bluetooth network will be determined from the collected data.

Bluetooth: The Bluetooth specification defines a short-range, low-power consumption, radio frequency (RF) technology for digital, wireless communication. Bluetooth is aimed at replacing the serial cables which are used on a day to day basis with many electronic devices, such as the cables connecting mice and keyboards to personal computers. Bluetooth is also capable of coding, decoding, and transmitting voice data making it applicable to cellular telephones and wireless headsets.

Bluetooth transmits in the unlicensed frequency band (2.4–2.48 GHz). Because this frequency range is used by many other devices (e.g. cordless telephones and microwave ovens), Bluetooth devices must address the problem of interference with these other devices. To accomplish this task, Bluetooth incorporates a *Frequency Hopping Spread Spectrum* (FHSS)[2] scheme. The frequency hopping pattern is chosen by a *pseudonoise* (PN) code generator[3], using a portion of the device's Bluetooth address as the seed for the PN code generator[4]. Bluetooth changes transmission channels at a rate of 1600 hops/s, creating 625 μs time slots. For example, if a user wants to transmit 1kbyte of data, instead of sending all of the information on a single channel where a burst of noise could destroy the entire transmission, the data will be segmented and each segment transmitted on a different channel. Therefore, if a burst of noise on a specific channel wipes out that information the majority of the data will be preserved. The fact that data were lost will be known, so the lost data may be retransmitted. Security is also built into the system because of the pseudorandom pattern used to alternate between transmission channels. Only the transmitting and receiving device will know what the next transmission channel is because only they will be synchronized before any transmissions take place. Along with FHSS, Bluetooth also uses forward error correction (FEC) and cyclic redundancy checks (CRC) for error detection and correction.

A Bluetooth network is made up of master and slave devices. The designation of a device as a master or slave occurs as the network is being established. This designation is in no way influenced by the capabilities of the device: it merely designates which device will control network operations. The master of a given network is simply the device which initiates the establishment of the network through and *inquiry*. The master device determines the

frequency-hopping pattern and synchronizes the PN generators of other devices, assigns network addresses to other devices, and handles all communication procedures between devices. The master controls the network.

The most basic Bluetooth network is a single point-to-point link. A point-to-point link is a communication link from one remote device to another remote device. We use this type of network as an example of how to establish a Bluetooth network.

Bluetooth network setup is accomplished completely by the software and hardware within a given Bluetooth device. No input from the user is necessary. The first step to establishing a network is for one device to execute an inquiry. During an inquiry the device is sending signals and waiting for replies to determine if there are any other Bluetooth devices within range. When a second Bluetooth device receives this inquiry signal it will respond with with, among other information, its Bluetooth Device Address (BD_ADDR). Upon receipt of this response the inquiring device becomes master of the network and will use the BD_ADDR of the second device to contact that device until a communication link is established. However, at this point the responding device may initiate a master-slave switch, after which it would be the master.

This leads to the next step: creating a link between devices. Bluetooth provides two type of communication links between devices: Asynchronous Connection-Less (ACL) links and Synchronous Connection-Oriented (SCO) links. SCO links are reserved for voice traffic so we will not go into detail about them. ACL links are used for data traffic. They can be point-to-point or point-to-multipoint (broadcast) transmissions and most ACL communications can be retransmitted in case of errors. In order to establish an ACL link the master will enter a *page mode* in which it will send connection requests to devices for which it has the BD_ADDR. In a two-device network, that would be the only device which responded to the master's inquiry. The slave device then has the option of accepting or rejecting the connection request. Assuming that the slave accepts the connection request, it will then transmit more information about itself to the master. The master will use this information to establish rules for transmissions on this link, such as data transmission rate or encryption type. Once an ACL link is established data transmission between master and slave can take place[5].

The basic multi-slave Bluetooth network is a *piconet* (Figure 3). A piconet is defined as a network consisting of a master and up to seven slaves. The master and slaves of a piconet form a star topology. All transmissions must go through the master. For example if Slave One wants to communicate with Slave Two, Slave One must first send the data to the master and then the master will send the data to Slave Two.

The procedures to establish a link are the same in a piconet as described above for a point-to-point connection. The only difference is that an inquiry will result in several devices responding instead of only one. When the master is ready to create a link it can choose with which device(s) it wants to create the connection(s). Again, as with the point-to-point network, the device which initiates the inquiry becomes the master of the piconet.

Current Progress: The first step to creating a Bluetooth network is to create an interface or method of communication from the Bluetooth device to a host. A host is the electronic

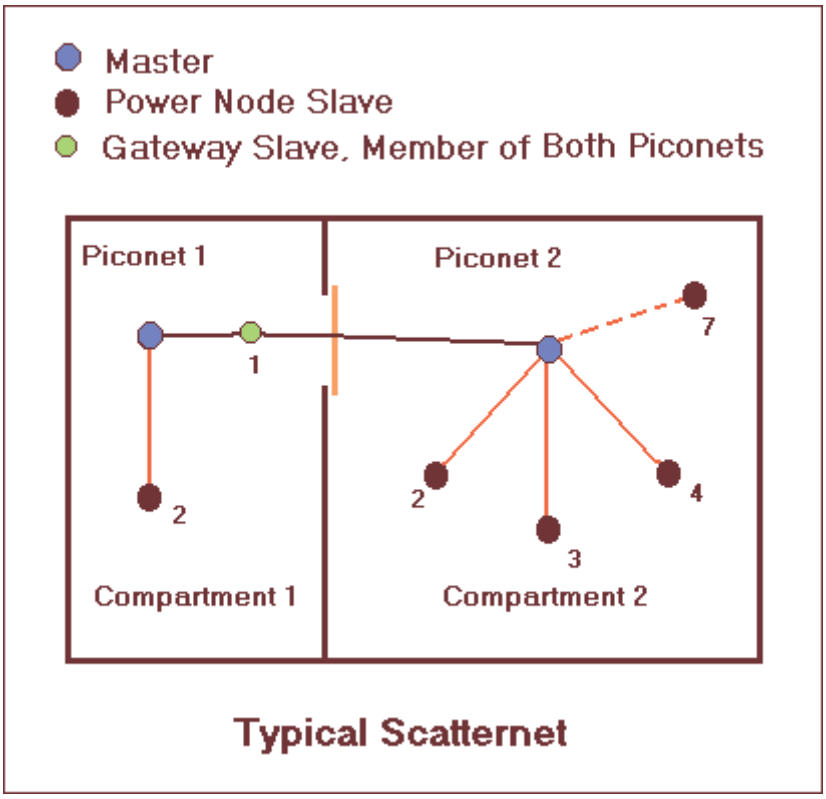

Figure 3: Bluetooth Network

device which will be running Bluetooth applications (e.g. a computer, cell phone, or Palm Pilot). A host and a Bluetooth transceiver interfaced together make up a Bluetooth master or slave. The majority of the transmission procedures and link policy controls for Bluetooth are taken care of by the *firmware* included in the Ericsson Bluetooth module which we use. Firmware is code which is written in read-only memory (ROM), normally within a microchip. The protocols and code we deal with the *Host Controller Interface* (HCI) on the Bluetooth module.

The HCI is the Bluetooth component that provides a standard interface for a host to communicate with a Bluetooth transceiver through a Universal Serial Bus (USB) or RS-232 serial communication port connection. A host passes commands and data to the HCI. The HCI then interprets the commands and passes instructions to the *Logical Link Control and Adaptation Protocol* (L2CAP) layer, which is software embedded in the Bluetooth module controlling the actual wireless link and data transmission.

Programs: In order to begin writing Bluetooth programs we first had to define all of the opcodes and event codes used by the Bluetooth HCI in a header file. Bluetooth HCI opcodes are two-byte binary HCI commands. Opcodes are transmitted from the host to the HCI where they are interpreted and acted upon. Bluetooth HCI event codes are one byte binary codes representing several HCI events. An event is an action taking place in the Bluetooth transceiver about which the HCI must notify the host. An example of this is when a command is passed to the HCI. A command-complete-event is returned stating the success or failure of the requested command. In the header file all of the Bluetooth HCI

opcodes and event codes are associated with a readable name. For example: the symbol `HCI_INQUIRY` represents the hexadecimal code `0x0401`.

We wrote a program in C++ to create an *HCI command packet* for each of the HCI commands the program uses. As stated earlier, packets of binary information are sent between the Bluetooth HCI and the host. These packets are very specific in their format in order to create a standard pattern for the bits being transmitted. The Bluetooth HCI uses three general packet types: *HCI command packets, HCI event packets,* and *HCI data packets*. HCI command packets contain opcodes and parameters for HCI commands sent from the host to the HCI. HCI event packets contain the event code of Bluetooth events and any information pertinent to that event which must be passed to the host. HCI data packets contain data transmitted by or to a remote device along with information which identifies from which device the data came or to which it is going.

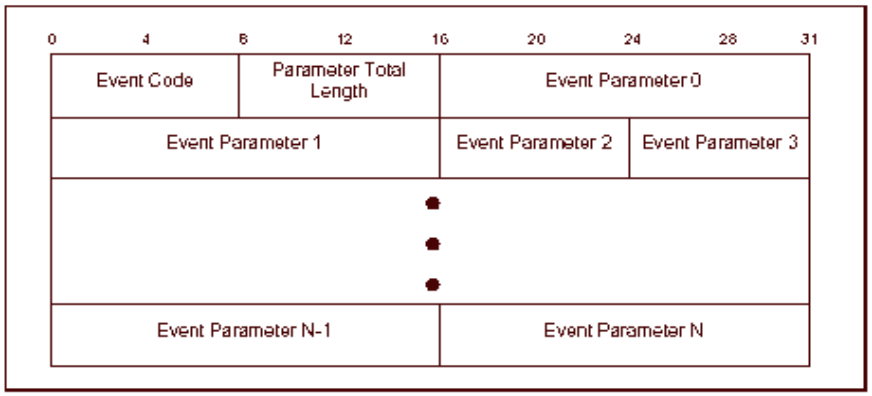

Figure 4: Bluetooth HCI Event Packet[4]

Figure 4 shows the structure of an HCI event packet. The first byte in the packet is the event code. This is used to determine what event has occurred. The second byte gives the total parameter length in bytes. This part of the packet is used to determine the end of one packet and the beginning of another. The rest of the packet contains the parameters for the given event. Each event or command has its own specific parameters and they are placed in a specific order in the packet.

We wrote a program, BT_Net v.1, to send an HCI command to the Bluetooth module and successfully interpret the returned event packet. This program has two functions. The first function creates a command packet requesting the Bluetooth device address of the module and the other creates a command packet requesting the buffer size of the module. Since data arrives in packet format vice a string of characters, it is necessary to determine what type of event has been received and what command created it in order to correctly interpret the packet. Once the type of event and, if necessary, the command that created the event has been determined, the results are displayed in three lines. The first line informs the user that the command was completed successfully or failed, the second line explains what type of data is contained in the third line, and the third line displays the actual data obtained from the command (Bluetooth device address or buffer size).

The program BT_Net v.2 was written to create a point-to-point link between two PC's for character data transfer. This program was successful in that a point-to-point link was

established and control information transfered, however data transfer does not execute in the current version. BT_Net v.2 is based to a large extent on BT_Net v.1, but is much more comprehensive.

Upon execution BT_Net v.2 will initialize the Bluetooth module for use. There is a series of commands which must be executed before an inquiry can be conduced and a link created. A brief description of these commands is contained in Table 1 and listed in order of execution.

Command	**Description**
Reset	Resets the Bluetooth module.
Read Buffer Size	Informs the host of the size of the data buffer in the Bluetooth module.
Host Buffer Size	Informs the Bluetooth module of the size of the host's data buffer.
Write Scan Enable	Enable the Bluetooth module to scan for inquiries and connection requests from other Bluetooth devices.
Write Authentication Enable	Informs the Bluetooth module if link authentication will be used or not.
Set Event Filter	Used to filter unwanted events and automate the acceptance of a connection request.
Write Connection Accept Timeout	Sets the amount of time the Bluetooth module is allowed to wait for the remote device to accept a connection request before automatically terminating the request.
Write Page Timeout	Sets the amount of time the Bluetooth module is allowed to wait for a remote device to respond to a connection request.

Table 1: Bluetooth Initialization Commands[6]

The key differences between BT_Net v.2 and BT_Net v.1 are:

- In BT_Net v.2 many more events are expected and thus more those events are handled by extending the "switch" statements that exist in BT_Net v.1.
- BT_Net v.2 takes into consideration the chance occurrence of a segmented packet being received. This arises because of the polling scheme implemented to check the serial port for incoming data.

The chance occurrence of a segmented packet is addressed by checking the length of the data read from the serial port and comparing it to the expected length. The expected

length is determined by checking the "total parameter length" section of each received packet and computing a total. In the event that these two numbers do not match, the last packet is moved to the front of the host's data buffer and the next time data is read from the serial port the new data, containing the remainder of the segmented packet, is placed immediately following the segmented data already received. This does not take place until the previous unsegmented packets have been read and processed.

The only foreseeable problem with the BT_Net v.2 code may occur if a packet is segmented before the "total parameter length" byte in a received packet. This would lead to an inaccurate computation of the expected packet length. Should this problem arise it is solvable, but we have neglected it to date.

The next program BT_Net v.3 will add functions to analyze bit error rates to the current BT_Net v.2 program. To test bit error rates a known bit pattern will be transmitted. As this data is received at a remote node it will be compared to the expected bit pattern and any deviations from the expected pattern will be recorded and written to a file. Another measurement that will be made with this program is the Receive Signal Strength. The Bluetooth HCI contains a command which reads the current signal strength (in dBm) at the receiver. Using signal strength measurements we will be able to validate the shipboard attenuation values obtained by ENS Estes.

When the programing is complete this network will be taken onboard *ex-USS America* at the Philadelphia Naval Yard for testing. During these tests onboard the *ex-USS America* attenuations (signal strengths) and bit error rates will be measured and analyzed from different geometries in different spaces using several transmission rates, with and without error correction. From these measurements it will be determined whether the use of Bluetooth onboard ships is practical.

References

[1] D.R.J. Estes, ENS, *A Trident Scholar Project Report: Assessment of Radio Frequency Propagation in a Naval Shipboard Environment*, United States Naval Academy, Annapolis, MD, 2001.

[2] R.L. Peterson, R.E. Ziemer, D.E. Borth, *Introduction to Spread Spectrum Communications*, New Jersey: Prentice Hall, 1995.

[3] Ferrel G. Stremler, *Introduction to Communication Systems*, Third Edition, New York: Addison–Wesley Publishing Company, 1992.

[4] Bluetooth SIG, *Specification of the Bluetooth System*, Vol. 1, 2001.

[5] R. Morrow, *Bluetooth Operation and Use*, New York: McGraw–Hill, 2002.

[6] Ericsson, *ROK 101 008 Bluetooth PtP Module: Specification Sheet*, Kista-Stockholm, Sweden, 2000.