Remote Data Acquisition using Bluetooth

David R. Loker, P.E.
Collin G. Frampton, Titan J. McElhaney,
Jonathan R. Mook, Anthony M. Sansone
Penn State Erie, The Behrend College

Abstract

In this paper, a remote data acquisition project using Bluetooth technology is presented for a senior technical elective telecommunications course in the Electrical Engineering Technology Baccalaureate Program at Penn State Erie, The Behrend College. There are several noteworthy characteristics of this project. First, the project used two Bluetooth modules which were each interfaced to a computer through an RS-232 port. Second, LabVIEW was used as the software development environment for communicating with the Bluetooth modules and for automatically uploading and downloading information from a data collection process.

The system requirements for the remote data acquisition project consisted of a local computer and a remote computer, simulating an environment for remotely collecting data. Each computer contained a Bluetooth module that was configured as either a master or a slave. The master initiated the establishment of a communication link with the slave, and the slave responded to the master. The local computer executed a LabVIEW program that configured its Bluetooth module as a master. The remote computer, which contained a data acquisition board for data collection, executed a LabVIEW program that configured its Bluetooth module as a slave.

The purpose of the local computer was to transmit control information for the data collection process to the remote computer. This information included the number of samples, the sample rate, and the acquisition mode. The purpose of the remote computer was to acquire the proper number of samples at the appropriate sample rate and to automatically upload the data to the local computer. Once the remote computer uploaded the data, it was downloaded at the local computer and displayed on a graph.

Results from both the local and remote computer programs are presented in order to verify that the functional requirements of this design project were satisfied. Results from a student assessment form are also presented. Conclusions and recommendations regarding the educational benefits of using this design project within a Baccalaureate Electrical Engineering Technology program are discussed.

I. Introduction to the Project

A senior technical elective course in telecommunications is offered as part of the Baccalaureate degree in Electrical Engineering Technology at Penn State Erie, The Behrend College. There are two prerequisites for this course. The first prerequisite is a junior level communications systems course which emphasizes an introduction to analog communication techniques. The second prerequisite is a junior level measurements and instrumentation course which introduces

LabVIEW.[1] LabVIEW is a graphical software programming language that was used for data acquisition. The senior telecommunications systems course emphasizes voice and data communications techniques. There are approximately seven laboratory experiments covered within this course. Each of the experiments utilizes LabVIEW and typically requires two weeks to complete. As part of the course requirements, students complete a group project in place of taking a final exam. Students select the team members and the project. Four team members chose to work on this remote data acquisition project, and they are the co-authors of this paper.

A detailed listing of the requirements for this project is shown below.

- Automatic collection and display of data acquired by a data acquisition board.
- Data transfer using Bluetooth technology.
- Software development using LabVIEW.
- Oral presentation of results.
- Demonstration of results.
- Report documenting results.

The overall project grade was worth 20% of the final grade for the course. The oral presentation was worth 25% of the project grade and the demonstration of the project and report were worth 75% of the project grade.

The remote data acquisition (DAQ) project consisted of a local computer and a remote computer. Each computer contained a Bluetooth module that was configured as either a master or a slave, and each computer included an RS-232 port for interfacing to each module. LabVIEW was used for communication with the Bluetooth modules through the RS-232 port. A LabVIEW program was executed on the local computer that configured the Bluetooth module as a master. A second LabVIEW program was executed on the remote computer that configured its Bluetooth module as a slave.

The user on the local computer selected one of three possible operational modes for the data acquisition process: single acquisition mode for collecting one set of data, continuous acquisition mode for continuously collecting sets of data, and stop for terminating the communication link. For each set of data points, the user specified the number of samples and the sample rate for collecting the data. For the continuous mode, the user also selected the time delay between each set of collected data points.

The purpose of the remote computer, which contained the DAQ board, was to acquire the proper number of samples at the appropriate sample rate and to automatically upload the data to the local computer. If continuous data collection was selected by the user at the local computer, the remote computer delayed a specific amount of time and repeated the data collection and uploading process. Each time the data was uploaded, the local computer downloaded and displayed the data on a graph.

II. Introduction to Bluetooth

Bluetooth received its name after a tenth-century Viking king who united and controlled Denmark and Norway.[2] This name was used since the objective of Bluetooth technology was to unify the telecommunications and computing industries.

Bluetooth wireless technology is a global standard for a low-cost, low-power, short-range radio that eliminates cables between stationary and mobile devices. It has created the idea of a Personal Area Network (PAN) that allows many different types of devices to interact with each other, as well as devices from many different types of manufacturers. It offers fast and reliable digital transmission of both voice and data over the globally available 2.4 GHz license-free Industrial, Scientific, and Medical (ISM) band. In order to avoid interference from other signals, the Bluetooth devices hop synchronously to new frequencies while maintaining a reliable communication link. The technique is called frequency hopping spread spectrum (FHSS), and it is accomplished by using a pseudo random sequence to determine the proper frequency hop channel number. Frequency hopping occurs at a rate of 1600 hops per second, and there are a total of 79 channels, with each channel occupying a bandwidth of 1MHz.[3] The maximum data rate is 723.2Kbps with a maximum transmission range of approximately 100 meters.

When Bluetooth modules come together, they form a network automatically, where each Bluetooth module functions as either a master or a slave. Since each Bluetooth module has a unique device address, the master can communicate directly with each slave. The master sets the frequency hopping sequence and the slave synchronizes to the master in time and frequency. A collection of one master and one or more slaves forms a piconet. The Bluetooth specification permits a piconet of up to seven slaves, where each slave is only communicating with the shared master.[4] Larger networks are formed by linking together piconets to form a scatternet, where some devices are members of multiple piconets.

III. Bluetooth Module

The Bluetooth modules used for this project were from Ericsson.[5] The Bluetooth module is functionally broken into three major sections: the baseband controller, flash memory, and the radio. The Bluetooth radio defines the requirements of the Bluetooth device operating in the 2.4GHz ISM band. The baseband controller includes firmware for the host controller interface (HCI), which handles the communication with an external host (e.g., computer). The host and the baseband controller can communicate with each other for the transmission and reception of data through a Universal Asynchronous Receiver Transmitter (UART) or a Universal Serial Bus (USB). The unique characteristics of these Bluetooth modules include a maximum data rate of 460 Kbps over the UART interface and a radio frequency (RF) output power of 0dBm for a maximum range of approximately 10 meters.

III.A. Hardware Configuration

One Bluetooth device was included with the Application and Training Tool Kit and mounted directly on a printed circuit (PC) board.[6] This vendor was also used to purchase additional Bluetooth modules. The second device was mounted on a protoboard by using a prototype adapter.[7] Each module was interfaced to an RS-232 port on each computer. The PC board already contained an interface for an RS-232 port. Thus, it was connected directly to the

computer. However, for the protoboard Bluetooth module, an RS-232 transceiver was needed to convert the signals from the Bluetooth module to RS-232 logic levels for interfacing to the computer.[8-9] The wiring schematic is available from the datasheets for the Bluetooth module. Based upon the schematic, four signals were provided for the UART interface. The Transmit Data (TXD) and Receive Data (RXD) from the RS-232 port on the computer were connected to the RXD and TXD on the Bluetooth module, respectively, through the RS-232 transceiver, for data flow. For flow control, Request to Send (RTS) and Clear to Send (CTS) from the RS-232 port on the computer were connected to the RTS and CTS on the Bluetooth module, respectively, through the RS-232 transceiver. Information about flow control for the RS-232 port is available in the literature.[10] Also, the grounds for the computer RS-232 port, Bluetooth module, and RS-232 transceiver were all connected together.

III.B. Host Controller Interface (HCI)

The Bluetooth modules operate on an HCI, which provides a uniform interface method for the Bluetooth modules. The HCI handles all of the communication with an external host, such as a computer. From the HCI specification, all information is in binary and sent in packets with hexadecimal little-endian formats. In this format, the least significant byte is transmitted first.[4]

The different groups of commands supported by HCI are: Link Control commands, Link Policy commands, Host Controller & Baseband commands, Informational commands, Status commands, and Testing commands. The Link Control and Policy commands enable the host to establish connections with other Bluetooth devices in its vicinity. The Host Controller & Baseband commands, along with the Informational and Status commands, enable the host to access various registers in the baseband controller. The Testing commands enable the host to test various functions within the Bluetooth module.

There are four main types of HCI packets: Command, Event, Asynchronous Connection-Less (ACL) link, and Synchronous Connection-Oriented (SCO) link. The format for the Command packet is shown in Figure 1. The maximum size of each Command packet is 255 bytes. The OpCode is two bytes and consists of a group field (OGF) which is 6 bits and a command field (OCF) which is 10 bits. Together these fields uniquely identify the specific groups of commands and the command within a group. Each command includes a set of parameters associated with it. The specific parameters and the size of each parameter are included in the Command packet. The 1-byte parameter total length identifies the total number of bytes contained within the parameter list.

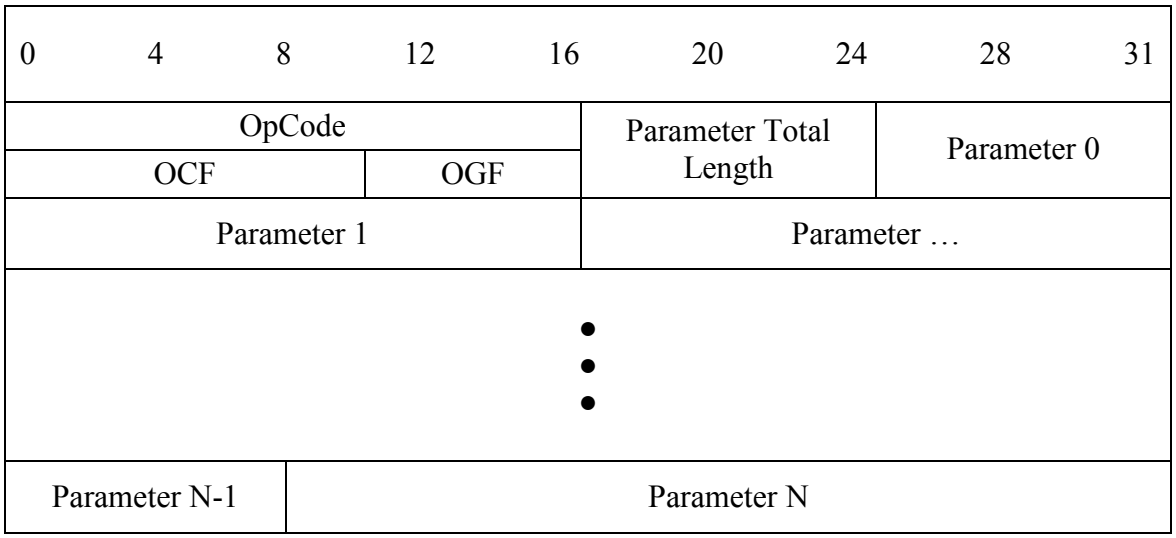

Figure 1. Command Packet Format

The result of each command is reported back in the form of an Event packet. The format for the Event packet is shown in Figure 2. The maximum size of each Event packet is 255 bytes. The 1-byte event code specifies the packet as an Event packet. The 1-byte parameter total length identifies the total number of bytes contained within the event parameter list. Most of the commands receive a Command Complete event after a command is properly completed.

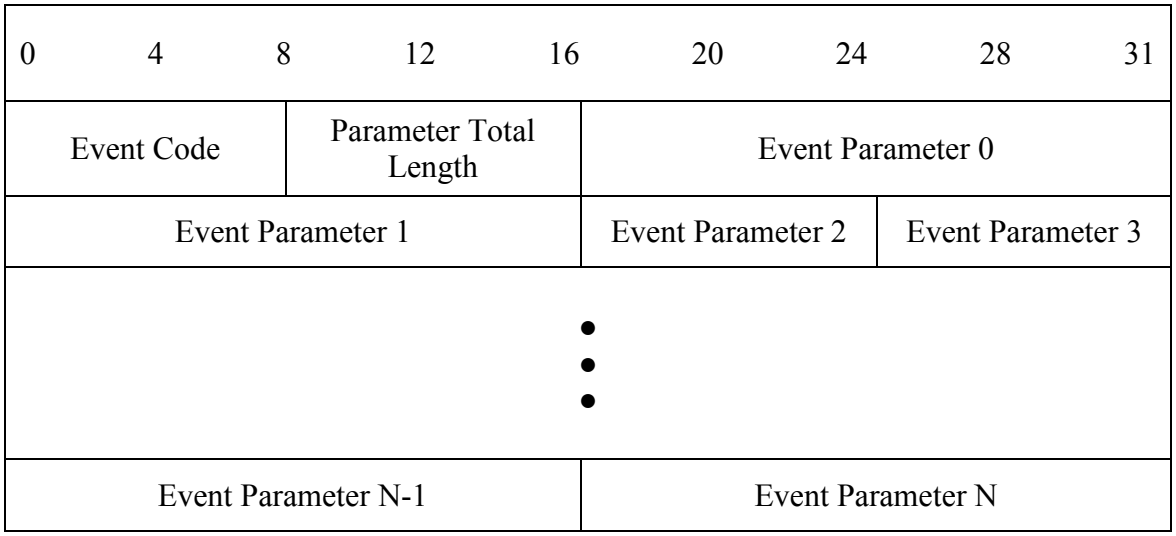

Figure 2. Event Packet Format

The third type of packet is the ACL packet, as shown in Figure 3. This packet contains the data that is exchanged between the Bluetooth modules. The 12-bit connection handle uniquely identifies a communication link with another Bluetooth module. The 2-bit packet boundary (PB) flag identifies whether the packet is a continuing fragment or the first packet of a higher layer

message. The 2-bit broadcast flag (BC) identifies whether the packet is for point-to-point transmission or for broadcasting to all slaves.

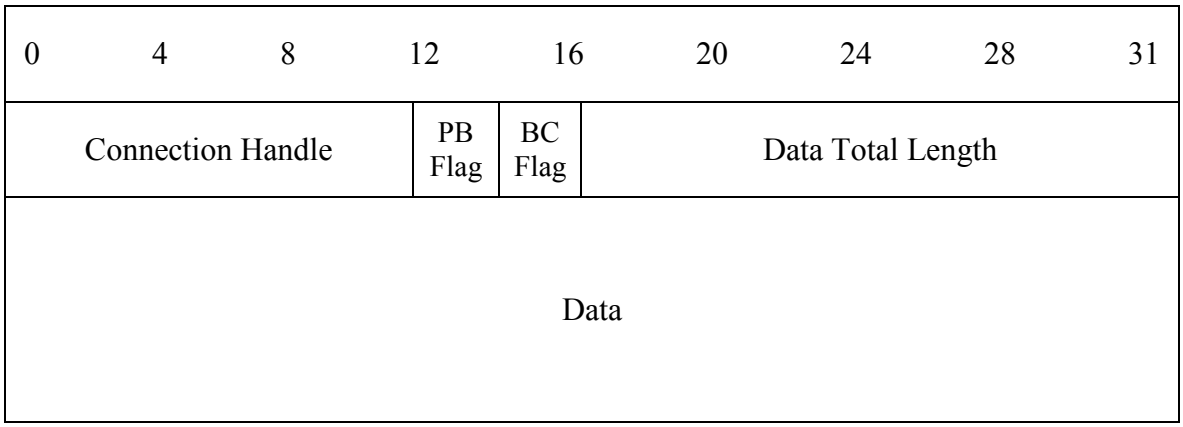

Figure 3. ACL Packet Format

SCO packets are used for transmitting voice data and will not be covered in this paper. Because these packets are sent through the UART interface, which is a common interface for all of the packets, a 1-byte packet indicator is sent to identify the type of packet. This identifier precedes the packets shown in Figures 1-3. The specific 1-byte indicator is shown in Table 1.

Table 1. HCI Packet Type Indicator

HCI Packet Type	**HCI Packet Indicator (in Hex)**
Command	01
ACL Data	02
SCO Data	03
Event	04

IV. Software Development for Bluetooth Communication Link

Bluetooth modules communicate in a master and slave configuration. Through a sequential process the devices are initialized, and then the master sends out an inquiry command to search for other devices within its range. When a slave responds, a connection request and accept connection are made. With a functional communication link between the devices, data can be transmitted in the ACL packet format shown in Figure 3. At the end of a communication session, a disconnect command is sent to close the connection. The flowchart for this process is shown in Figure 4.

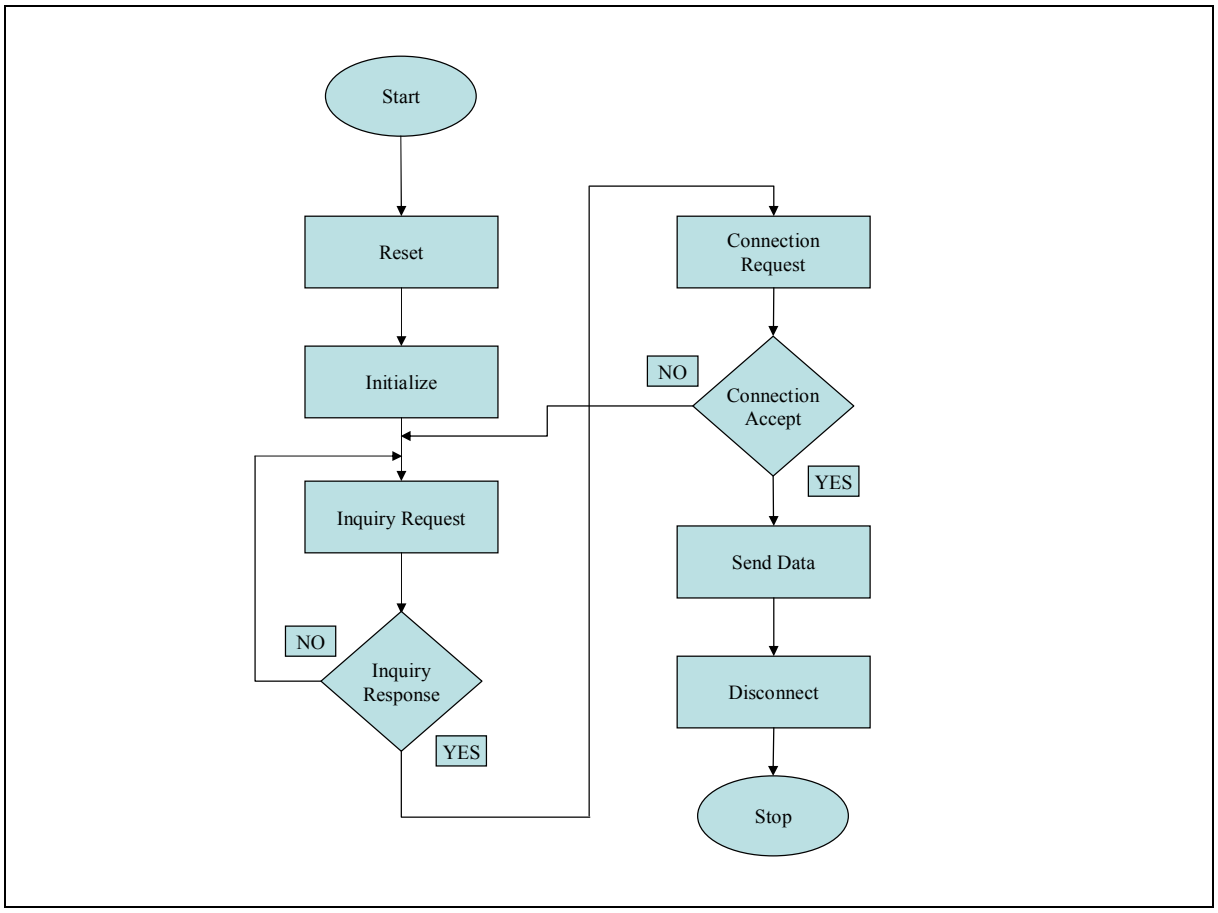

Figure 4. Flowchart for Bluetooth Communication Link

A detailed listing of the sequence of events for the establishment of a Bluetooth communication link between 1 master and 1 slave, described as a point-to-point connection to form a piconet, is shown in Table 2. The Bluetooth module on the protoboard was configured as the master and the module on the PC board was configured as a slave.

The initialization routine included numerous HCI commands for setting the desired parameters of the Bluetooth module. The commands were sent to each device independently, and a Command Complete event was received for each command. The first step was to reset the device to the factory default values. Next, a Read Buffer Size command was sent to determine how much buffer space is available in the Bluetooth device. The response from the command indicated a maximum of 672 bytes in each ACL data packet and a maximum of 8 ACL data packets can be stored in the data buffer for each Bluetooth module. The next step was to set the Host Controller to Host Flow Control. Once the host knows the maximum number of bytes per packet and the maximum number of packets, it can segment the data to not exceed these values. The Host Buffer Size was set next, which notifies the host controller of the buffering capabilities of the host. Although this was not required for this application, it is necessary for some situations where the host has limited processing capability, such as in mobile phone handsets.

Table 2. Detailed Sequence of Events for Bluetooth Communication Link

	MASTER (Proto Board - 5302 1737 8000)		**SLAVE (Trainer Board - 8F65 3137 8000)**	
	Send	Receive	Send	Receive
Initialization	Reset 0103 0C00	Command Complete 040E 0408 030C 00	Reset 0103 0C00	Command Complete 040E 0408 030C 00
	Read Buffer Size 0105 1000	Command Complete 040E 0B08 0510 00A0 0200 0800 0000	Read Buffer Size 0105 1000	Command Complete 040E 0B08 0510 00A0 0200 0800 0000
	Set Host Controller to Host Flow Control 0131 0C01 00	Command Complete 040E 0408 310C 00	Set Host Controller to Host Flow Control 0131 0C01 00	Command Complete 040E 0408 310C 00
	Set Host Buffer Size 01 330C 0700 A000 0100 0000	Command Complete 040E 0408 330C 11	Set Host Buffer Size 01 330C 0700 A000 0100 0000	Command Complete 040E 0408 330C 11
	Write Connection Accept Timeout 0116 0C02 A01F	Command Complete 040E 0408 160C 00	Write Connection Accept Timeout 0116 0C02 A01F	Command Complete 040E 0408 160C 00
	Write Page Timeout 0118 0C02 0040	Command Complete 040E 0408 180C 00	Write Page Timeout 0118 0C02 0040	Command Complete 040E 0408 180C 00
	Write Scan Enable 011A 0C01 03	Command Complete 040E 0408 1A0C 00	Write Scan Enable 011A 0C01 03	Command Complete 040E 0408 1A0C 00
Inquiry	Inquiry Request 0101 0405 338B 9E04 08	Command Status 040F 0400 0701 04		
		Inquiry Result (**Address - Connection Handle**) 0402 0F01 **8F65 3137 8000 0100** 0000 0000 **XXXX**		
		Inquiry Complete 0401 0100		
Connection	Create Connection (**Address - Connection Handle**) 0105 040D **8F65 3137 8000** 0800 **0100 XXXX** 01	Command Status 040F 0400 0705 04		
			Accept Connection Request (Sent w/in 4.92s) 0109 0407 **5302 1737 8000** 01	Connection Request 0404 0A **5302 1737 8000** 0000 0001
				Command Status 040F 0400 0809 04
		Connection Complete 0403 0B00 **0100 8F65 3137 8000** 0100		Connection Complete 0403 0B00 **0100 5302 1737 8000** 0100
		Command Status 040F 0400 0800 00		
Data	Write Packet (4 Bytes - 00 00 01 01) 0201 2004 0**000 0001 01**			Receive Packet (4 Bytes - 00 00 01 01) 0201 2004 0**000 0001 01**
Close	Disconnect 0106 0403 0100 16	Command Status 040F 0400 XX06 04		
		Disconnect Complete 0405 0400 0100 13		Disconnect Complete 0405 0400 0100 13

The next command was the Write Connection Accept Timeout, which told the module the maximum amount of time to wait for a connection accept response from the host. Associated with this was the Write Page Timeout, which defined the maximum time the local device will wait for a baseband page response from the remote device at a locally initiated connection attempt. The final stage in the initialization routine was the Write Scan Enable command. The scan enable parameter controls whether the device periodically scans for page requests and inquiry requests from other Bluetooth devices. Both scans were set to be enabled. The devices were now initialized and ready for an inquiry command.

The purpose of the inquiry command was to discover other Bluetooth devices that are waiting to communicate. The device configured as the master sent out an Inquiry Request command and received three codes in return. The first event received was the Command Status to signal the start of an inquiry command. An Inquiry Result event was received with the address of the remote device, the connection handle for that particular connection, and a clock offset (identified as xxxx in the table since its value varies for each connection) required for determining a frequency hop sequence. These values were used in the next stage for the master to connect to the remote device. The final response was an Inquiry Complete event.

For the master to connect to a remote device to establish a piconet, the Create Connection command was issued by the master with the address of the remote device, the connection handle, and the clock offset, all of which were obtained from the Inquiry Result event. A Command Status event was received to verify the beginning and end of the connection attempt. Between these events, a Connection Complete event was received verifying a communication link has been established with a remote device. A piconet was established between 1 master and 1 slave, and the devices were now ready for communication. Test data was used to verify that the slave received the same data as what was sent by the master. Finally, a Disconnect command was sent by the master to terminate the communication link with the slave.

Since these packets of information were communicated through the computer RS-232 port and the Bluetooth device, a LabVIEW terminal program was created for initial testing. Information on the development of a LabVIEW terminal program is available in the literature.[11] The terminal program was executed on both the local and remote computers to verify that a successful communication link was established between a master and slave. Once this was established, two separate programs were created for the local and remote computers in order to automatically create the piconet and transfer data.

V. Software Development for Remote Data Acquisition System

There were two programs written, one for the local computer and one for the remote computer. The flowcharts for each are shown in Figures 5 and 6. The program on the local computer began by configuring its Bluetooth module as the master and automatically creating a communication link with the slave. The sequence of events is detailed in Table 2. Then it read the DAQ control parameters from the user inputs on the front panel. This data was framed into an HCI packet and transmitted. At this point, the program waited for data from the slave. Once the program received the returned data, it extracted the DAQ control parameters and the data samples from

the packet. The waveform was then displayed on the front panel. If the program was set for continuous run, the process repeated.

The program on the remote computer configured its Bluetooth module as a slave and waited for a connection request from the master, as detailed in Table 2. Once this was received, the connection was accepted. Then it received the packet from the master and extracted the DAQ control parameters. This information was used to acquire data from the DAQ card. The DAQ control parameters were added to the data acquired, and a packet was created. This packet was transmitted to the master. If the operational mode was set to continuous run, the program re-looped to wait for DAQ control parameters to be sent from the master. If the program was not set to run continuously, the program waited for a disconnect event indicating that the master had disconnected the communication link.

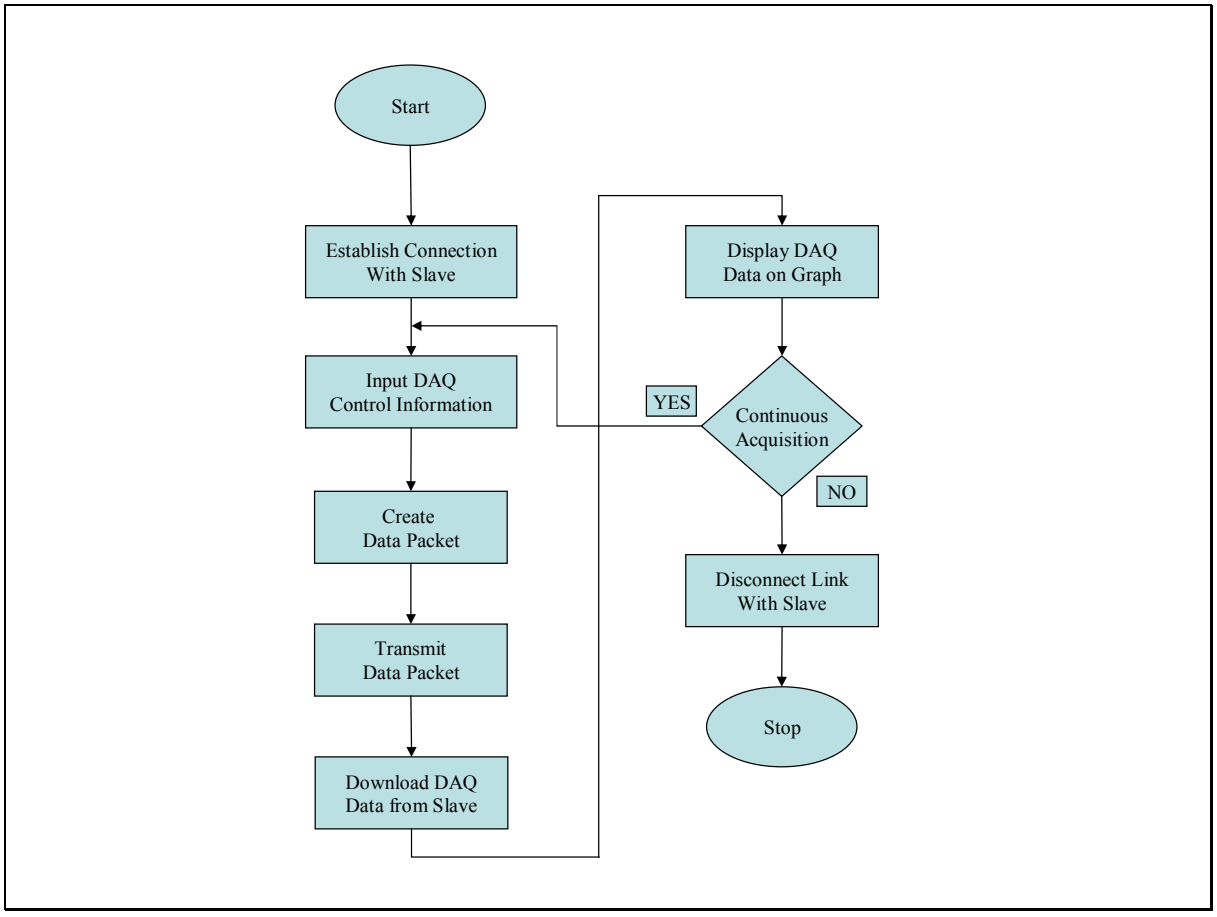

Figure 5. Flowchart for Local Computer

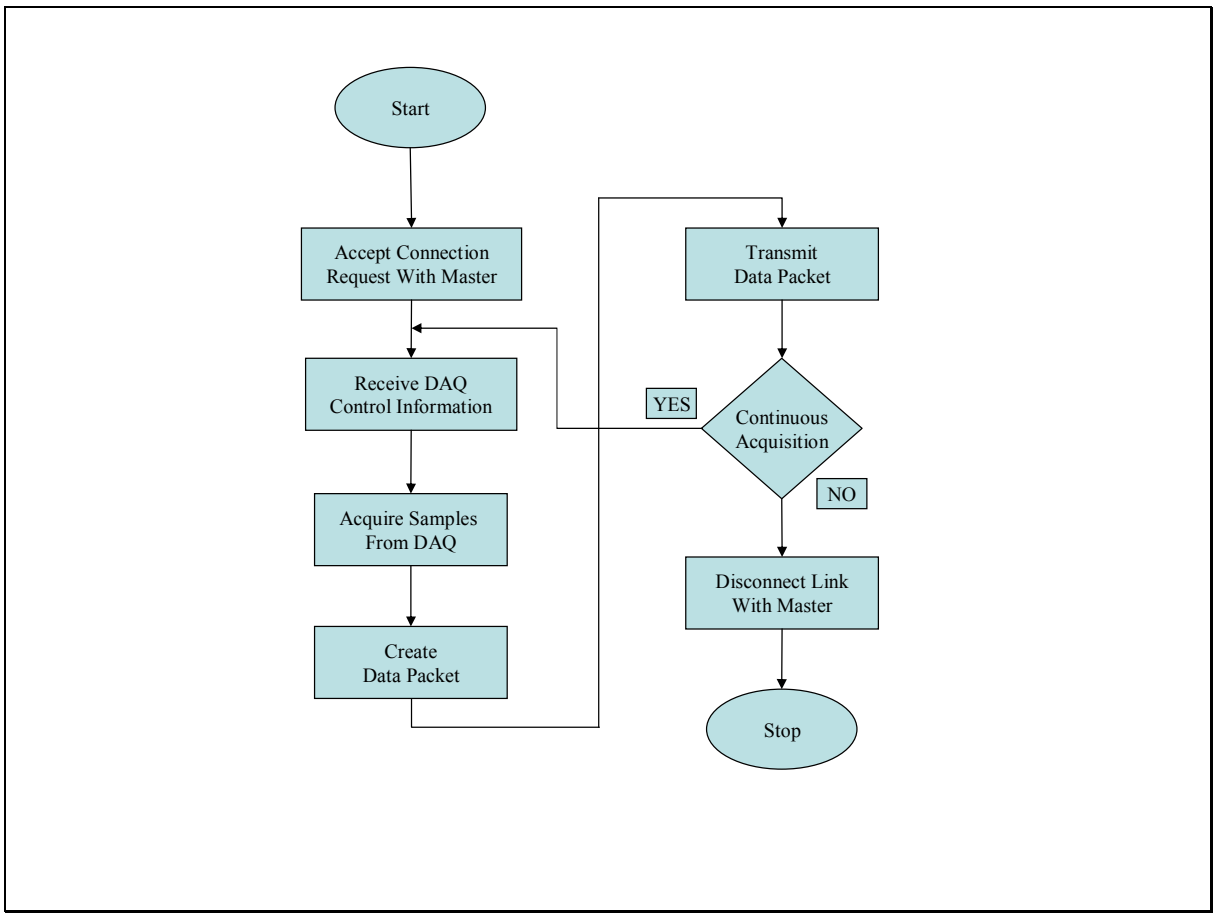

Figure 6. Flowchart for Remote Computer

VI. Results from Data Acquisition System

Figure 7 shows the front panel of the LabVIEW program, designed by the student team, for the local computer. This program configured its Bluetooth module as the master, had the acquisition mode set for continuous operation, and the DAQ control parameters set for collecting 28 samples at a rate of 10KHz with a time delay of 250mS between each set of samples. After the data was collected and uploaded by the remote computer, the data was downloaded on the local computer and displayed on a graph. In addition, the DAQ control parameters that the remote computer used to acquire the data were also displayed. This confirmed that the proper parameters were used for data collection.

The front panel of the LabVIEW program shown in Figure 8 was also designed by the student team, and it was executed on the remote computer. This program configured its Bluetooth module as a slave. A sinusoidal waveform was connected to a channel on the DAQ board. Once the program received the DAQ control parameters from the master, it used these parameters to acquire the data. After displaying the data on a graph, it uploaded the data to the local computer.

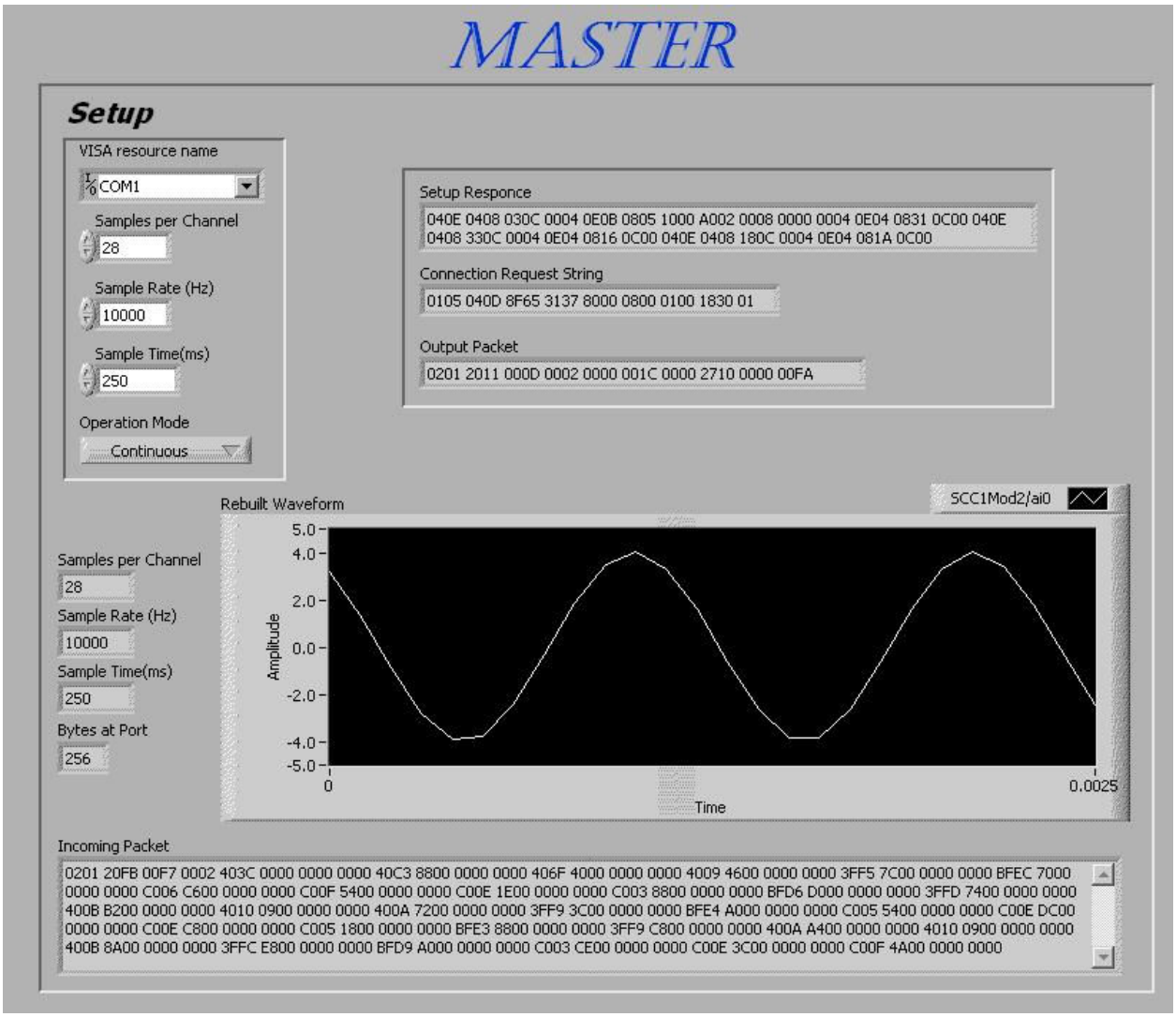

Figure 7. Results from Program on Local Computer

Figure 8. Results from Program on Remote Computer

VII. Results from Student Assessment Form

Each of the four members of the team was asked to complete an assessment form to evaluate this project. Table 3 includes the questions asked on the form and the responses from the students.

Table 3. Student Assessment Results

Questions	Responses
Approximate time in hours put into this project	40-50 → 4 students
Level of Difficulty	Very hard → 1 student Hard → 3 students
Areas of Difficulty	Bluetooth Instructions → 4 students
Appropriateness of Project for this Course	Excellent → 4 students
The Number of Students on This Project for the Level of Effort Required	About Right → 4 students
Level of Interest in Bluetooth and Wireless Technology as a Result of This Project	Significantly Improved → 1 student Improved → 2 students Same → 1 student
Suggestions for Improvement	Focus more on applications by using a pre-coded set of Bluetooth instructions to alleviate problems in writing code using HEX → 1 student Send more data by having the slave program break data into smaller packets and master program reassemble the data. (This was not required for this project in order to simplify the project requirements) → 2 students
Would you prefer to have this project covered during lecture and included as a lab rather than as a project?	No → 4 students (The students indicated that there is too much material for one lab)
Other Comments	Utilize other devices to interface with the Bluetooth device, such as a cell phone or a PDA. (No other comments provided)

VIII. Conclusions

In this paper, results were shown to successfully demonstrate a remote data acquisition system using Bluetooth. Also, LabVIEW was used as the programming environment for implementing the software requirements.

Results from the student assessment form were very favorable. Each student felt that this was an excellent project for the course, and most students indicated their level of interest in Bluetooth and wireless technology either significantly improved or improved as a result of this project.

All students agreed that this project was either hard or very hard, with the Bluetooth instructions as the primary area of difficulty. They also indicated that the project was not trivial, but in the end they felt that the project was gratifying to implement a fully working Bluetooth data acquisition system.

IX. Recommendations

One recommendation would be to provide the students with some text instructions already pre-coded to generate the appropriate HEX codes. This would help the students have more time to focus on applications.

An additional recommendation for this project would be to replace the remote computer with a microcontroller that has a serial port for interfacing to a Bluetooth device. Software development for the microcontroller could be achieved by using assembly language or a higher-level programming language. However, it may also be necessary to provide pre-coded instructions to help reduce the level of difficulty and time needed for programming the microcontroller.

Finally, this project could also be expanded into a capstone design project by utilizing the above recommendations and many Bluetooth devices to form a scatternet. Software development could also include SCO packets for transmitting voice data. Then, various network protocols could be investigated for traffic management.

Bibliography

1. National Instruments, LabVIEW, Version 7.1.
2. Web page http://www.palowireless.com/infotooth/.
3. Bray, J. and C. Sturman, *Bluetooth: Connect Without Cables*, Prentice Hall, 2001.
4. Bluetooth Specifications, web page https://www.bluetooth.org/spec/.
5. Ericsson Microelectronics AB, part # ROK 101 007.
6. Web site http://www.comtec.teleca.se/.
7. Web site http://www.emulation.com/bluetooth/.
8. Maxim Integrated Products, Inc., part # MAX3232ECPE.
9. Analog Devices, Inc., part # ADM3203AN.
10. Tomasi, W., *Electronic Communications Systems: Fundamentals Through Advanced*, Prentice Hall, 2004.
11. Loker., D. Remote Data Acquisition Using LabVIEW. Proceedings of the Annual Meeting, American Society for Engineering Education. (Reference session 2259, file 00859.PDF on CD-ROM). Albuquerque, N.M., 7 pages. (2001).

DAVID R. LOKER

David R. Loker received the M.S.E.E. degree from Syracuse University in 1986. In 1984, he joined General Electric (GE) Company, AESD, as a design engineer. In 1988, he joined the faculty at Penn State Erie, The Behrend College, in the Electrical Engineering Technology program. His research interests include wireless sensor networks, communication systems, and instrumentation systems.

COLLIN G. FRAMPTON

Collin G. Frampton, a senior at Penn State Erie, The Behrend College, is majoring in Electrical Engineering Technology. In 2001, he received an honorable discharge from the U.S. Coast Guard where he served as an Electronics Technician 2nd Class. His research interests include control systems and relevant programming.

TITAN J. MCELHANEY
Titan J. McElhaney is currently a senior at Penn State Erie, The Behrend College majoring in Electrical Engineering Technology. His interests are control systems and programmable logic controls.

JONATHAN R. MOOK
Jonathan R. Mook received a certificate in Electrical Engineering from Crawford County Area Vocational Technical School in 2001. He received an Associates Degree in Electrical Engineering Technology in 2003 from Penn State Erie, The Behrend College. He is expected to receive a B.S.E.E.T. degree in May 2005.

ANTHONY M. SANSONE
Anthony M. Sansone received his A.S.E.E.T. from Penn State Erie, The Behrend College in May of 2003. He will be completing his B.S.E.E.T. in May 2005.