Bluetooth Transaction System

Project Overview

Martin Ferus

Table of Contents

Overview

The goals of my project were to obtain a Bluetooth module that can be connected to the Z8 and communicate with it. I planned to create an example POS (Point Of Sale) system which would allow a simple payment transaction to be completed from a cell phone. I planned to show how the system could keep track of transactions and have access to the user's funds information which would simulate a money account.

I was able to achieve these goals in my project. I did have some issues with implementation of some of the original architecture ideas which I overcame with alternate solutions. These are further described in the document.

Project Information

The payment system I created simulated a simple point of sale system which allows a customer to pay for a purchase using their cell phone. In order to keep the project manageable, the store only sells 3 items: Milk, Socks and Light bulbs. Any transaction using this system involves two actors. A client shops for items in the imaginary store and brings them to the cashier. A cashier enters info into the terminal about the objects which the customer is buying and their quantity. The terminal communicates with the client's cell phone via a Bluetooth, informing the customer which items they are purchasing and the price for each item. The customer can then choose to pay for the items. At this point a payment transmission is sent to the POS system and the total is deducted from the remaining available balance and the total is set to zero. The balance amount simulates a link to the customer's bank or credit card account which would be accessed in a real transaction.

I originally planned to have the Z8 act as the transaction system and only connect to the PC to update the balance amount. The problem I ran into with this set up is that I was not able to connect both the Bluetooth module and the serial connector to the PC simultaneously. The challenges section below discusses this problem in more detail. After I tried several ways to overcome this problem which were unsuccessful, I decided that I would still be able to accomplish my goals and there were two possible solutions. The first solution involves creating the transaction system to be operated from the PC completely and use the Z8 as a way to communicate the between the PC and the Bluetooth module. In this configuration, the cashier would interact with the computer system via a visual interface. The second solution I created was to contain the complete transaction system on the Z8 board, completely eliminating the PC. In this version, the cashier could interact with the system using the on-board switches and reading the LED display. For each of these systems, the customer's cell phone client would work in exactly the same way.

In the following section I will describe the necessary components for the implementation of each version and the steps required to get it working. I will discuss the hardware configuration and software development process in the 'Development' section.

Project Versions

Version 1 – PC Based Transaction System

The PC based point of sale system uses a personal computer as a central server for processing transactions. A cashier program is simulated on the PC and it can interact with a client cell phone.

- **Hardware:**

The following hardware was used to run this system.

- **PC -**

  A personal computer with the windows operating system is required. This system must have an available serial port.

- **Z8 Evaluation Board**

  The Z8 Encore! Microcontroller evaluation board is used for this project. In this version the Z8 plays a minor role but it has only been tested on this board.

- **Parani-ESD 100 Bluetooth module**

  The ESD 100 is a Bluetooth module I purchased from Lemos International. It has a range of up to 100 meters and supports the SPP (Serial Port Profile) which enable it to emulate a serial connection over the Bluetooth link.

- **Bluetooth Enabled J2ME cell phone**

  The cell phone is required to run the client application. I used a Sony Ericsson K530i. Any compatible cell phone can work for this project but it must have built-in Bluetooth and support J2ME with MIDP 2.0 profiles, a CLDC 1.0 configuration and the Bluetooth libraries for J2ME (JSR 82). It must also allow third party access to the Bluetooth module from within the J2ME framework and especially to allow interaction using SPP.

- **Wiring**

Because this is an academic project, the wiring of the Bluetooth device to the Z8 was done by simply using wires to connect the UART, VDD and GND pins on the Z8 directly to the pins on the Bluetooth module. In any practical implementation, the module and Z8 microcontroller would be most likely placed together on a circuit board. Additionally, a serial cable is required to connect the Z8 board to the PC and a power supply is needed for both the PC and the Z8 board.

- **Software**

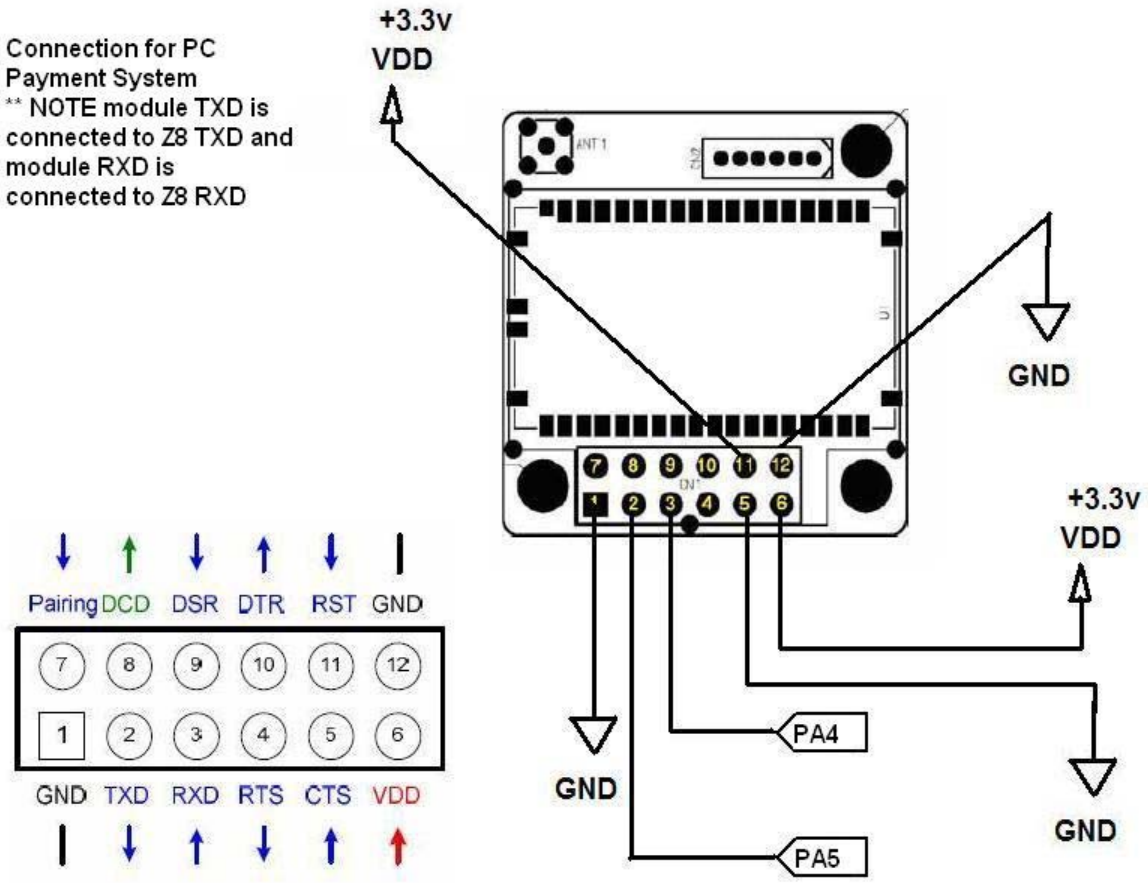
Each component has some required software to operate the transaction system. The Z8 needs to be pre-programmed with the appropriate software which will allow the communication between the Z8 and the PC. The PC will need to have the Windows operating system and a Java Virtual Machine installed. The custom software it will be running is the program the cashier will interact with directly and it is a Java SWING

based application. Although it is a Java Program, Windows is required for this version because of components which are used for serial communication. It is possible to configure this application to run on Linux, UNIX or Solaris OS easily without changing any of the components but for this version only the Windows option is enabled. The cell phone will also need to have the client J2ME application installed. The application should work well on any phone that matches the requirements for the phone mentioned above. Bluetooth communication should be enabled on the cell phone after the software is installed.

- **Setup**

The following instructions detail the process of setting up the transaction system if all the required components are available. (Version 1)

1. Load the appropriate software onto the Z8 board if it is not already. The appropriate code is located in TOPC_COM.zip. Consult the Z8 manual for detailed instructions.
2. Install the Java SWING application on the PC:
   a. The code is located in SWING_POS.zip
   b. Unzip in any directory
3. Use the serial cable to connect the PC to the Z8 board.
4. Connect the Parani Bluetooth module to the Z8 board. Refer to the diagram below for detailed pin instructions.



5. Note that for this implementation, the RXD pin of the Bluetooth module connects to the RXD pin on the Z8 board and the TXD pin of the Bluetooth module connects to the TXD pin on the Z8 board. This is to ensure direct communication between the Bluetooth module and the PC.
6. Install the Software on the cell phone.
   a. Unzip BTPayment.zip into any directory.
   b. Copy the files from the /bin directory to the cell phone. (refer to your cell phone manual for instructions on installing J2ME applications.)
   c. Ensure the Bluetooth capability is enabled on the cell phone.
7. Make sure that the Z8 board is connected to the power source.

- **Operation**

The following instructions detail the process of operating the transaction system and performing a sample transaction.

1. Start up the Java SWING application on RunPOS.bat.
2. On your cell phone, ensure that the cell phone has located the Bluetooth module. It should be called “BluetoothPayment”. If it is not located, you can manually search for the device.

3. Start the application on your cell phone from the location it is installed in. The application should be called “BTPayment”.
4. To simulate a cashier scanning items, click on any of the three buttons on the SWING application. Each click will increase a total on the screen and transmit an item price to the cell phone.
5. To pay for the items so far using the phone, select the option “More” and then select “Pay”. This will transmit a payment authorization to the PC and deduct the total from the available funds balance and update the total and balance values on the screen. The remaining balance will also be transmitted back to the phone.
6. To obtain the remaining balance at any time using the phone, select the “More” option and then select “Info”. This will request the balance from the PC and transmit it back to the phone.
7. At any time, if you wish to clear the screen on the phone, simply select “More” and then “Clear”. This will clear the screen.
8. The PC program is set up so that it can transmit messages to the customer’s phone. Simply enter the text you wish to send into the textbox on the screen and press the “Send” button. This will transmit the text in the textbox to the customer’s phone.
9. If a payment transaction is processed which is more than the remaining balance, the PC sends the phone a message indicating “Insufficient Funds”

Version 2 – Z8 Based Transaction System

The Z8-Based point of sale system is almost identical in operation to the PC version. The main technical difference is that it is operated completely from the Z8 evaluation board without the need for a PC. The majority of the components and operation is the same with some key exceptions. In interest of brevity, I will only highlight the differences and everything else can be assumed to be identical to that of version 1.

- **Hardware**

The main difference in Version 2 is the lack of a PC. The PC is not required and in this version the Z8 plays the central role as the main transaction processing server. All other hardware requirements are identical.

- **Wiring**

The wiring needs are the same except for the need for a serial cable. Because the PC is not used, the serial cable is not necessary for operation. It may be needed to initially set up the software on the Z8 unless it is already done.

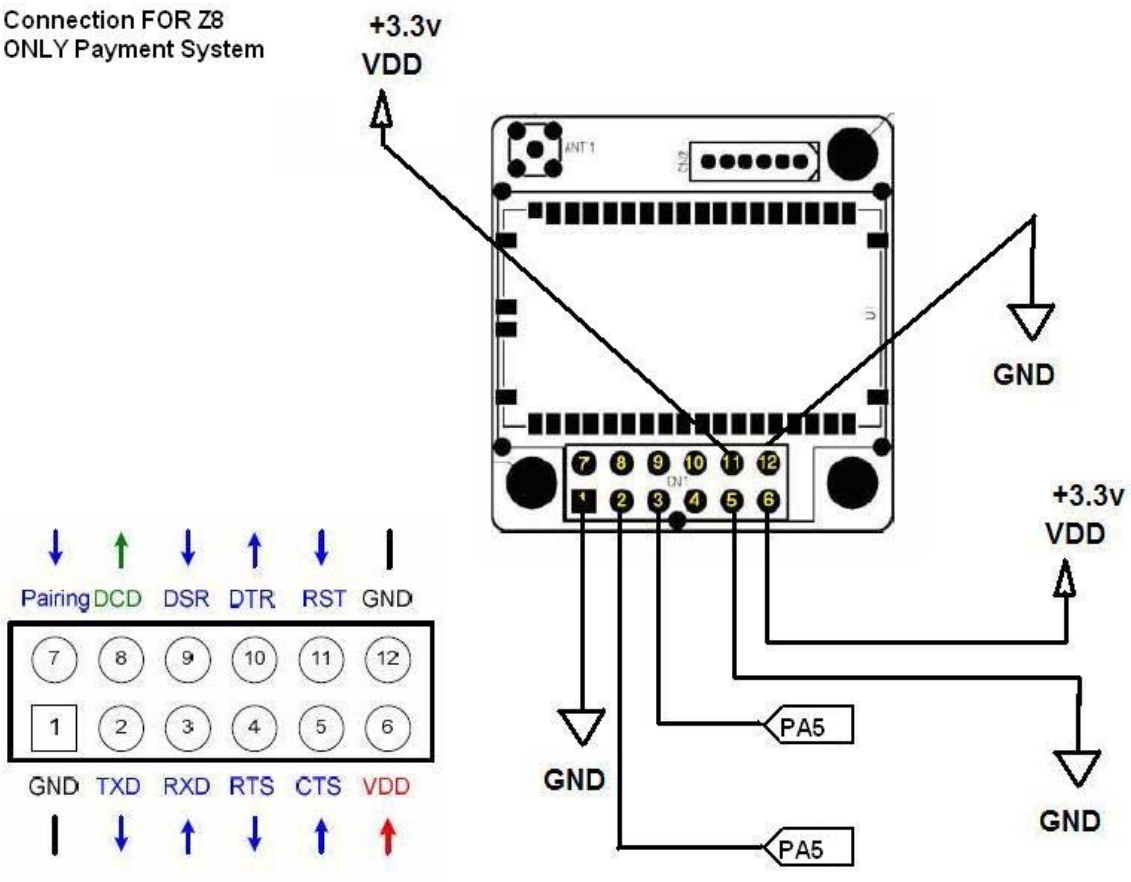
- **Software**

In Version 2, the main difference is that the PC Java SWING application has been ported to the Z8. The Z8 handles all of the transaction processing and communication with the Bluetooth module. The cell phone software is completely compatible with both versions and needs no changes for this version.

- **Setup**

The following instructions detail the process of setting up the transaction system if all the required components are available. (Version 2)

1. Load the appropriate software onto the Z8 board if it is not already. The appropriate code is located in Z8_ONLY.zip. Consult the Z8 manual for detailed instructions.
2. Connect the Parani Bluetooth module to the Z8 board. Refer to the diagram below for detailed pin instructions.



3. Note that for this implementation, the RXD pin of the Bluetooth module connects

to the TXD pin on the Z8 board and the TXD pin of the Bluetooth module connects to the RXD pin on the Z8 board. This creates a regular UART connection between the Parani Bluetooth module and the Z8 evaluation board.

4. Install the Software on the cell phone.
   a. Unzip BTPayment.zip into any directory.
   b. Copy the files from the /bin directory to the cell phone. (refer to your cell phone manual for instructions on installing J2ME applications.)
   c. Ensure the Bluetooth capability is enabled on the cell phone.
5. Make sure that the Z8 board is connected to the power source.

- **Operation**

The following instructions detail the process of operating the transaction system and performing a sample transaction.

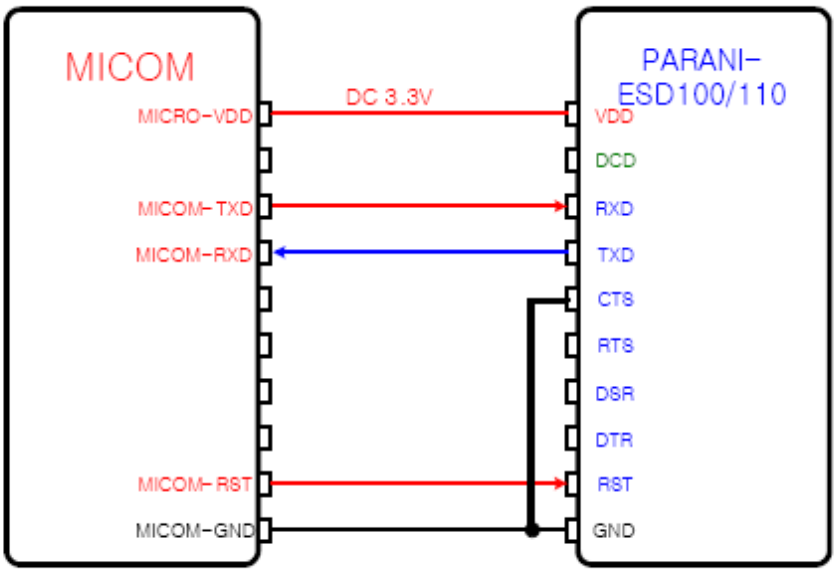
1. On your cell phone, ensure that the cell phone has located the Bluetooth module. It should be called “BluetoothPayment”. If it is not located, you can manually search for the device.
2. Start the application on your cell phone from the location it is installed in. The application should be called “BTPayment”
3. To simulate a cashier scanning items, Press one of the switches on the Z8 board. SW1 corresponds to Light bulbs, SW2 corresponds to Socks and SW3 corresponds to Milk. The total will be scrolled across the LED display. The balance is tracked internally.
4. To pay for the items so far using the phone, select the option “More” and then select “Pay”. This will transmit a payment authorization to the Z8 and deduct the total from the available funds balance and update the total value on the LEDs. The remaining balance will be transmitted back to the phone.
5. To obtain the remaining balance at any time using the phone, select the “More” option and then select “Info”. This will request the balance from the Z8 and transmit it back to the phone.
6. At any time, if you wish to clear the screen on the phone, simply select “More” and then “Clear”. This will clear the screen.
7. If a payment transaction is processed which is more than the remaining balance, the Z8 sends the phone a message indicating “Insufficient Funds”

Development

The following section describes the development details of each component of the transaction system. The tools and process needed for development and configuration will be described as well as details of the software developed.

Parani ESD-100

The Parani ESD-100 Bluetooth module is a small device designed to communicate with other Bluetooth devices using the SPP (Serial Port Profile). It can emulate serial communications between the host and another Bluetooth enabled device. This module can be connected to a host using a UART connection. It supports hardware flow control but I chose not use it for my project because it was unnecessary given the scope. To support the connection, there are 12 pins on the module. (Please refer to page 29 of the Parani ESD-100 user manual for detail of the pin configuration. It is located in /DOCUMENTATION/BT_MANUAL.pdf) The following diagram shows which pins should be connected for the configuration for this project.



The Parani ESD-100 uses AT commands for configuration and operation. Module can receive commands from the host it is connected to. In order to communicate with the module, a PC can use terminal software to send and receive commands. To connect a PC to the module using the Z8, it needs to be connected in the same way as described above for project version 1. The wiring diagram has been included above. To connect the PC to the Z8 simply use the serial cable.

The serial port settings can also be changed using the AT commands. For this project, the default configuration will be used. This configuration is:

Baud Rate: 9600
Data Byte: 8
Parity: No Parity
Stop Bit: 1
No Hardware Flow Control.

In addition to configuration of the serial port communication, the module will operate in one of 4 modes. The module is set to mode 0 by default. This mode is mainly for configuration and the device accepts AT configuration commands in this mode. It does not broadcast itself as a Bluetooth

node in this mode. In this mode, the following commands need to be issued to ensure the Bluetooth module is configured correctly for the project.

AT+UARTCONFIG,9600,N,1,0
This command configures the baud rate to 9600, sets the parity to no parity, sets the stop bit to 1 and indicates a 0 to disable hardware flow control.

AT+BTMODE,3
This command configures the Bluetooth module to operate on mode 3. Mode 3 enables the module to act as a server and broadcast it's availability to any connections. This is the mode used for the project. The two other available modes are more restrictive and limit the availability and role of the module.

After these changes are made, the settings are stored on internal flash memory and the module is ready for operation for the project for either version.

Z8

The Z8 evaluation board plays a different role for each version of the project. In version 1, the Z8 acts as a way to enable serial communication between the PC and the Bluetooth module. The software for this version is very simple. It's only function is to initialize UART0 on Port A to operate on the settings compatible to the Bluetooth module. After the initialization, there is a while loop that will run indefinitely and has no operations. The serial port communication between the PC and the Z8 board is "echoed" on the UART pins on the board. When connected as described above, the transmit pin which is connected to the Bluetooth module's transmit pin, is connected to the serial cable's receive pin. This creates a way that all Bluetooth transmissions are received by the PC's serial connection. Similarly, all transmissions from the PC are received by the Z8 receive pin and since the UART receive pin on the Z8 is connected to the receive pin on the Bluetooth module, the module receives all the communication that the Z8 receives. Once this software and connection are made, the Z8 acts as merely a communications conduit and all operation of the module and transaction system is handled by the PC.

In version 2, the Z8 plays the central role of the transaction system. The Bluetooth module is connected the UART0 on Port A to communicate directly with the Z8. In this configuration, the pins are connected as expected. The transmit pin on the Z8 is connected to the receive pin on the Bluetooth module and vice versa. The software on the Z8 is also more robust. There UART port is initialized to operate on a configuration compatible with the Bluetooth module. The program operates in a round-robin fashion. A while loop is responsible for polling for button presses and for displaying the appropriate scrolling total value. Global variables maintain the value of the total and the remaining balance. All output to the Bluetooth module is transmitted through UART0 and a timer is used to retrieve pending data that is being received from the Bluetooth module. The software accepts two types of input from the cell phone to process transactions.
<PAYMENT> : this is the string used in this protocol to identify that the connected customer has accepted the total amount and authorized a payment transaction. When this is received, the total is deducted from the remaining balance. If the funds are inadequate, a response is send to the customer indicating this; otherwise the remaining balance is sent.
<INFO> : this string is used in this protocol to identify that the connected customer has requested to know the remaining balance. The remaining balance is then transmitted to the customer.

All Z8 software is developed using the Zilog IDE and written in the C language. Once a Z8 has been programmed with either version of the software, it is ready for operation.

Cell phone

The cell phone client application was developed for a J2ME platform. The cell phone needs to meet all of the requirements as listed above. In order to develop the software, a PC with a JDK 1.4 or higher is needed. Additionally, the Sun Java Wireless Toolkit needs to be installed. The toolkit simplifies the building and deployment of J2ME applications. Once installed, a project can be set up with the appropriate specifications needed and it can be used to build and package the application.
To develop the software for this project the following settings need to be used in the project.
MIDP 2.0
CLDC 1.0
Bluetooth for J2ME (JSR-82) support.
Permissions need to be given to the following packages:
javax.microedition.io.Connetor.bluetooth.client
javax.microedition.io.Connetor.bluetooth.server

These settings will enable the application to access resources on the phone which it will need for Bluetooth communication. These settings are set by the toolkit in the MANIFEST.MF file which is part of the application deployment.

The J2ME program uses some standard UI controls and menu items. It uses a Bluetooth SPP URL to identify the Bluetooth module which it will communicate with and opens a StreamConnection object to that device. InputStream and OutputStream objects are used to send and receive data using the Bluetooth connection. The details of handling the Bluetooth stack are encapsulated by these Java objects. A thread is started during the startup of the application which polls for data input and displays any received text in the appropriate place on the screen. The preprogrammed commands of <PAYMENT> and <INFO> are sent whenever the corresponding user input is detected.

Once the program is written, it can be compiled by the program and packaged appropriately. The SJWT will generate the necessary JAR and JAD file as well as the MANIFEST.MF file. These three files need to be deployed to the phone. Once the application is successfully installed on the phone, it is ready to interact with the payment system and is compatible with both version 1 and version 2 of the project.

A copy of the source as well as the generated deployment files can be located in BTPayment.zip

PC Payment System

The development of the PC based transaction system for version 2 contains two parts. One part is a custom transaction system developed for the project which handles the graphical user interface and the transaction processing. The other part is an open source library for Java which provides an API to allow embedded serial communication in Java programs. This library is called “Simple Serial”. It technically allows communication on Windows, Linux / UNIX and Solaris operating system but for this project, I only used the windows capabilities. The library consists of Java source code and windows compatible .DLL files which allow the JVM to communicate with native windows serial port control. The API provides an object that is used to send and receive data between the program and the serial connection. The program uses SWING GUI components to present the user with a nice graphical interface which is created to simulate a cash register program. The interaction between the user and the interface triggers events which simulate cashiers actions. There are 3 buttons which correspond to each item a customer can purchase. Pressing each button increases the total amount which is displayed in a textbox on the screen and transmits the price to the customer using the serial connection. There is also a textbox which displays the current available balance and an input box which allows the cashier to enter

text to send to the customer. There is a SWING thread which is running in the background to check the serial port for input and whenever there is any input, it is evaluated and the appropriate action is taken. The commands received can be either <PAYMENT> or <INFO> and are processed the same way as on the Z8 version described above. For additional info and debugging, certain operations are logged on the console window.

After the program is developed, the classes are compiled into class files and included in the same directory as the simple serial objects. A .bat file was created to run the java command to execute the main method in the application. Once the program is running, it is ready to communicate with the Z8 over the serial port on COM1, which is the default serial port. This setting cannot be changed in this version of the program but it is technically possible with little effort.

The source code as well as the compiled objects can be located in SWING_POS.zip

Challenges

I faced several challenges during the development of this project which I either overcame or otherwise resolved with an alternate solution. Each component presented its own set of difficulties and challenges. These are detailed below.

Parani ESD-100

The Bluetooth module provided significant challenges. The biggest problem was trying to communicate between the Z8 and the module. The first obstacle was getting the module connected correctly. I looked over the instructions and it was not evident to me that there should be two GND connections made. In addition, the diagram in the manual confused me and this confusion lead me to connect the VDD and GND pins in reverse, which resulted in short circuiting the module. I was not aware of this and continued trying to get it to work in many possibly configurations. At times I was discouraged and worked on other components of the project, which thankfully could be worked on independently.

After some help and evaluation of the module using the oscilloscope, I determined that it had been rendered unusable. The oscilloscope showed a clean transmission from the Z8 to the module, but the response from the module was a very irregular wave-form and its range of voltage was significantly lower than expected: 0 to about 1.5 volts and the expected high was around 3.3 volts.

After this determination, I was able to purchase a second identical module and have it shipped quickly. I received it on the last Friday before the deadline and was able to focus on getting it to work for about 4 days.

The following issues were less serious. The next biggest problem was to configure the module. Because I was unable to connect both the module and the PC serial connection simultaneously (this problem described later) I was unable to easily send commands and view the response or to even tell if the communication was occurring correctly. I attempted several ways to try to connect the module directly to the PC using the Z8. I tried to set both UART pins as simply GPIO input but that did not work. I considered connecting the module directly to the serial cable using available wires using a serial cable pin diagram as a reference but decided not to since the wires did not fit correctly and I was unsure if it would operate correctly. Additionally, I did not want to damage a second module because of the cost and time constraints. I finally was able to connect using the reversed TXD – RXD pin connection method and configure the module correctly using hyper terminal.

Z8

The main issue I had with the Z8 in this project was trying to use both UART ports to communicate with the PC and the Bluetooth module simultaneously. The goal of the original program was to operate the transaction system from the Z8 as in version 2 but the balance would be stored on the PC and would be updated from the Z8 after a transaction. I intended to connect the serial cable to UART0 on Port A and the module to UART1 on Port D. I wrote some code for the Z8 which would perform a communication between the PC and the Bluetooth module by passing all input from UART0 as output to UART1 and vice-a-versa. The Z8 would be able to perform some operations in between if needed. I was unsuccessful in making this type of connection.

I tried several possibilities to troubleshoot this issue. First, I eliminated reliance on kbhit() to read input because I thought it was only configured to read UART0. When looking at the code for kbhit() in the Zilog libraries, I was able to confirm that it only read input from the current default UART but I was unable to determine how to set which UART is the default one. Once I used more low level methods to read inputs from both UARTS, I was still unable to read any input from UART1 although UART0 worked ok with this method. I thought that perhaps the code which handled the polling of the switches was interfering since it also used Port D. After removing the button code form the project, I was still unsuccessful in communicating with the Bluetooth module. In the interest of time, I decided to split these two features into separate versions of the project so that I can showcase all of the features and capabilities that I wanted. I suspect that there was an error in the configuration of Port D for UART1 or the way which I was passing data between the two UARTs was incorrect. I have included the code for the attempt to connect both simultaneously. It can be found in DUAL_PORT.zip

An additional issue which I had with the Z8 module was the completeness of data when receiving from the Bluetooth module. After debugging, I determined that only some of the characters which were being sent were registered by the program. I narrowed the issue to the method which I used to poll the UART input. Originally I checked the input during the main while loop execution and additional processing delays caused too much of a lag between each poll and characters were skipped. To overcome this, I changed the polling to occur on a timer which generated an ISR which had priority over the while loop execution. I also set the timer to generate an interrupt at a faster rate than the baud rate of the Bluetooth module (9600) to ensure no characters were missed. I was not sure if I can configure the GPIO pins to generate an interrupt since the pins were in use for a UART connection but I thought that an interrupt driven method would probably be best. The timer method did work reliably for the scope of the project since only one customer interacted with the module at any given time.

Cell phone

The development of the software for the cell phone provided some challenge. The biggest challenge was the initial learning curve. I am an experienced Java programmer but never worked with J2ME and I knew nothing about Bluetooth communication. After reading much documentation online, I began to experiment with J2ME programming and development tools. I tried to develop examples manually by generating the required files by hand and compiling applications from the command line. I also attempted to work with a J2ME eclipse plug-in called EclipseME. This plug-in extends the eclipse environment to allow the development of J2ME applications. I found this to be a very robust and feature rich development environment but it was overly complex for the short amount of time I had to work on my project and I decided not to use up time learning how to use it. I would consider this for any long-term J2ME development projects. I finally discovered the Sun Java Wireless Toolkit which is a tool that helps with the initial configuration of a project and with building and packaging an application for deployment. It also provides a phone emulator to test applications on. Although I had problems with the emulator, this tool proved to be the best for the job. I developed the applications in TextPad and used the toolkit to build them.

In addition to this, I spend a considerable amount of time to figure out how to establish a

Bluetooth connection within the phone. Several factors need to be correctly configured for the phone to work. The project needs to be build with the appropriate permissions enabled and the phone needs to have the correct Bluetooth settings turned on. The method to establish a simple SPP connection to the phone is not clearly described in any online documentation that I could find and I had to scrap together some ideas from other projects and use trial and error.

The last issue to overcome was to try to ensure that threading works correctly to poll for input. Unlike the PC version described below, the input can only be read one character at a time and I had to create some code to read a whole line which would be displayed on the screen. This was a noticeable difference between the regular Java and J2ME version because the stream objects in regular java provide methods to read a line of text at a time. I was able to learn how to do all of these things in J2ME and in the end, I think that it is a great platform for development after the initial hurdle of configuring the environment correctly.

PC Payment System

The PC proved to be slightly less challenging because of my familiarity with the environment. The biggest obstacle was to create a program which can communicate using the serial port. Because of my familiarity with Java, I began to look for a way to accomplish this task. Sun provides a downloadable package called the "Java Communications API". This is a set of classes which extends Java through the javax.comm package and provides a set of objects which can be used for communication using RS-232 hardware. This is a very robust and easy to use package and is perfect for this project with a small exception. The Windows support has been discontinued because of the frequent and drastic changes to the Windows native serial port interfaces. Sun decided that in order to maintain reliability of their software, they would simply stop supporting that OS. Fortunately, there are archived copies of older releases of the package which include Windows support. I used one of these older versions in my program.

The rest of the application was less complex. I am familiar with working with Java APIs and in development of SWING applications for the desktop. The only problematic issue was how to listen for input. I resolved this problem in the same way as in the cell phone version, by creating a thread which runs in the background and polls for input. The main difference is that rather than using regular threads, SWING applications use a special worker thread.

The API provided many options which would allow the serial connection to be configured for each computer according to system setting, operating system and available ports. I chose to limit the complexity of the application to only connect to the default windows configuration of serial communication on port COM1. Given more time, the application could easily be made portable.

Future Considerations

There are several ways to improve this project from both the academic and practical perspective. The first update I would try to make is to make the Bluetooth module connect to the Z8 simultaneously with the PC. Separating the tracking of the balance from the transaction system itself, better illustrates the separation between the transaction system and the "account" which stores the value. If this separation existed, the transaction system would not need any knowledge of how the balance is deducted and can be extended to work with any compatible system. This would require some sort of protocol for exchanging transaction info between the system and the account interface. With such a protocol in place, any account processing system can be made to work with a Z8 based Bluetooth transaction system as long as it conformed to the protocol. This would enable compatibility with credit card processing machines, stored value systems and others.

For a practical implementation of the system, hardware flow control would need to be used

to ensure data transfer reliability, especially since it can be assumed that more than one customer would be able to connect to the system.

An added feature can be built in to ensure correct communication between customers and the system. The Bluetooth stack manages connections but it does not know anything about users and the data. A protocol can be developed which provides a way to uniquely identify users and possibly establish and maintain user sessions using handshaking and confirmation. This would ensure that all customers concurrently connected to the system can perform operations separately without interfering with other customer's transactions. This may require threading capability on the Z8.

Security is an issue that was not addressed in this project at all. For a secure transaction system, appropriate measures would have to be placed on all the components. The cell phone could establish an encrypted connection with the Bluetooth server. A more secured profile can be used other than SPP. The connection between the Z8 and the PC would likewise need to be secured, probably using encryption. This area would be one of the most in need of research and development for a working transaction system.

Conclusion

Bluetooth communication along with J2ME is rapidly becoming more ubiquitous and widely available. More and more applications are developed which harness these capabilities to enhance the users' experience. Hopefully in the future, Bluetooth transaction systems will eliminate many hassles in shopping experiences. It may be possible to envision supermarkets without checkout lines or bars and restaurants where the employees do not need to worry about collecting and processing payments and can focus on the needs of their customers. The possibilities are vast but there are several obstacles to overcome. After working on this project, I can see that any good system will need to handle security well and be able to differentiate between users and prevent them from being able to affect each other's transaction.

Links and References

The following section lists the links to obtain the various software I used for development and references to documentation.

The Sun Java Wireless Toolkit:
http://java.sun.com/products/sjwtoolkit/

The Java Communications API (Newest Version)
http://java.sun.com/products/javacomm/

The Java Communications API Archive version
http://web.media.mit.edu/~benres/simpleserial/

Parani ESD-1000 documentation and software page.
http://www.sena.com/support/downloads/#sd

Java.net reference for Bluetooth information.
http://today.java.net/pub/a/today/2004/07/27/bluetooth.html

Technical Specifications of the Cell phone I used.
http://www.mobiledia.com/phones/sonyericsson/k530i.html