

Home Automation Final Report

December 7, 2000

Project Members:

Chewy Chong
Innes Finch
Anuj Girdhar
Quang Pham
David Williams

Georgia Institute of Technology

ECE4005

Professor: Dr. Henry Owen

TABLE OF CONTENTS

PAGE

A. ABSTRACT	3
B. OBJECTIVE	3
C. OVERVIEW	3
C.1. Control electrical components in home	4
C.2. Home Automation	4
C.3. Java Development Kit	4
C.3.1. X10 Java Development Toolkit	4
C.3.2. Java Application for X10 Control via the X10JDTk	5
C.3.3. Java Servlet for Receiving Web Commands	6
D. PROBLEMS AND SOLUTIONS	6
D.1. Problem: Emulator Solution: Real Palm	6
D.2. Problem: Connect the Palm Cradle to the Firecracker correctly.	7
D.3. Problem: Java	8
D.4. Problem: Lack of programmable DTR line	8
D.5. Problem: Programming of Dragonball registers	8
D.6. CodeWarrior Compiling Problem	9
D.7. Problem: Deciding how to interface the PC with the power line	9
D.8. Problem: Deciding on the structure of the X10JDTk	10
D.9. Problem: Home Surveillance not completed	10
E. TESTING	10
E.1. Used the oscilloscope to check voltage levels	10
E.2. Verify signal from palm to firecracker by indicated by link lights.	11
E.3. Verify from Palmpad remote to GUI on PC to talk to the devices.	11
E.4. Final testing with appliances, Palm and servlet	11
F. LOGISTICS	11
F.1. Timeline Schedule	12
F.2. Order of Completion	12
F.3. Individual Group Member Contribution	13
G. INSTALLATION GUIDE	14
G.1. Installation for the Palm	14
G.2. Installation for the GUI on the computer	14
G.2.1. Adding Java Comm Functionality to your JDE and JRE	15
G.2.2. Adding the Java Servlet Development Kit to your JDE	15
G.2.3. Installing the Tomcat Servlet Web Server	15
G.2.4. Installing the X10 Java Development Toolkit	15
H. MARKETING	16
I. CONCLUSION	17
J. DOCUMENTATION AND SOURCES	18
K. APPENDIX I AND II	19

A. Abstract

With the rising power of technology, we are able to accomplish things at a much quicker rate. We have at the touch of a button access to large amounts of information due to the capability of computers and the Internet. Not only has technology given us more information, but it also has given us the ability to communicate, organize, and manage our time. Electronic mail and Palm Pilots have made time management and organization so much more efficient. Palm Pilots are handheld computing devices that are becoming immensely popular as the means by which everyone manages personal information, accesses and enters corporate data, and mines the richness of the web. X10 has taken this concept of management and control a step further, by enabling you to control electronic devices in your home with a remote or. This concept of controlling electronic devices in your home by a Palm Pilot is implemented in our project.

B. Objective

The objective of this project is to create a series of home automation / control tools based on top of existing X10 devices. These tools include a Palm based X10 controller, a web based X10 controller, and a web based surveillance system.

X10 is an industry standard communication protocol that works over localized electrical wiring (e.g. wiring in the home). A number of home automation X10 modules are already on the market. These modules are placed in between the target device's electrical plug and the electrical outlet. The modules are then able to control the flow of electricity to the attached device enabling us to turn them on, off, dim and brighten lights, amongst other things.

The tools being developed for our project are designed to centralize control of these on-the-market X10 modules. Centralization of control allows for more complicated tasks to be done such as macro lighting (bulk setting of multiple devices) and automation of tasks (timed lighting). The abilities gained by such centralization provide a foundation for a smarter and more efficient home environment.

C. Overview

The main overview for the project is to be able to communicate with different electrical devices within the home wirelessly. The project consists of using the existing X10 systems in two different ways. As one objective, the Base Station will be controlled using the Palm Pilot and a personal computer. Our second objective is to set up a Home Automation System based on the Palm and the computer.

C.1. Controlling the Electrical Devices in the Home

In order to make the Palm communicate with the base station, the first objective is to make it compatible with the X10 devices. It is necessary to reverse engineer the X10 devices and the Palm Pilot to be able to ensure that the pin-outs of the Palm are connecting correctly with those of the X10 devices. It is also necessary to write the software that will ultimately send the commands to the base stations. The software will allow the Palm to communicate with the FireCracker, which is an X10 serial port device. The FireCracker is designed to attach to the serial port of a computer and transmit commands from the computer to the X10 base stations, which in turn control the appliances. Currently, there are X10 modules, as stated previously, that can control these components, such as lights, fans, and stereos, but the number of devices are limited (16 modules). The program will be able to differentiate between houses and devices, thus controlling a total of 256 household appliances.

C.2. Home Automation

Now that we are able to get the Palm to communicate with the X10 devices, it is time to expand the design project to home surveillance using the Internet. The second objective is to setup basic scenarios to help sell the software. Several applications for home security will be developed. A system will be setup using motion sensors and digital cameras that will take pictures and send information to the base station. The information will be sent remotely to a server that can be accessed by the user. These motion sensors and digital cameras communicate any activity to the base station, which will constantly update the web page. To implement this part of the design the group used a web server with Java Servlet technology.

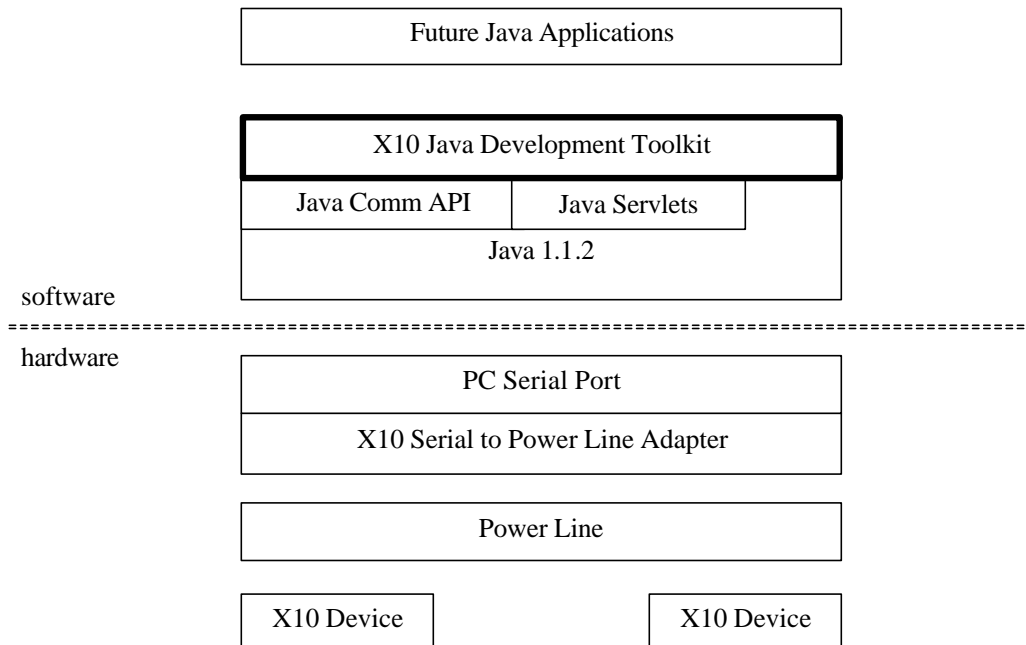
C.3. Java Development Toolkit

Our initial group objective was to write software to control X10 devices from the computer and the web. The language chosen was Java. Since we were planning on writing two different applications for X10 control, focus was placed on developing a robust set of Java tools to control X10 devices. This set of tools would provide X10 control to other Java applications that required X10 functionality. The resulting applications written to provide X10 control from the computer and the web utilized our X10 Java Development Toolkit (X10JDTk)

C.3.1. X10 Java Development Toolkit

The X10JDTk was written specifically to assist future developers of Java X10 applications. The toolkit was abstracted out such that the programmer does not have to directly deal with intricacies of sending a X10 command to a X10 device. In other words, the programmer can simply send a command to the toolkit knowing it will result in the X10 command being sent out on the power line.

Figure 1. The X10JDTk Stack



The X10 transmitter used in this project converts PC serial signals into X10 commands directly on the power line. This transmitter has two connectors. On one end, the transmitter is attached to a serial port on the computer. On the other end, it is plugged directly into a regular power outlet. The X10JDTk sets and maintains the necessary serial port properties (baud rate, stop bits, etc) and sends the appropriate commands to the transmitter to accomplish a requested X10 task.

The X10JDTk can send and receive X10 commands from the power line. It has a buffer which stores all the X10 commands on the power line not sent by itself. For example, a wireless transmitter such as the Palm X10 device may be used to send commands to a given X10 device. These foreign commands are picked up (but not removed from the power line) by the toolkit and stored. Since the foreign commands are stored, the programmer can write applications that use this information to become a X10 device themselves. For example, a programmer can write an application that takes input from the X10JDTk. Using a wireless transmitter, I can tell an application to perform a specific task via X10 commands. A X10 protocol can even be written based completely on X10 ON, OFF, DIM and BRIGHTEN signals.

C.3.2. Java Application for X10 Control via the X10JDTk

This stand-alone Java application was used to test the functionality of the X10JDTk. Since the toolkit itself does not have a front end, it was difficult to test it without this application. All functionality of the toolkit can be accessed using this application. X10

commands are typed out in the application. When the user hits enter, the requested X10 command is sent to the transmitter. Consequently, the transmitter places the appropriate X10 signal onto the power line.

When foreign X10 commands are detected on the power line, the transmitter copies the command and sends it to the toolkit. The toolkit notifies a listener that a command has been received and places that command in a buffer. Any applications set to listen in on these events is then able to read the foreign command from the toolkit's buffer. In the stand alone Java application, the foreign commands are shown in the receive window.

C.3.3. Java Servlet for Receiving Web Commands

A Servlet is the Java equivalent of a CGI program. Commands are sent to this program via the web and the program performs a specific task. Think of it as another method of receiving input. The Java Servlet requires a web server that is JSP / Servlet aware. Sun and Apache have JSP/Servlet plug-ins for web servers such as Apache, Netscape and IIS.

The web application developed consists of two parts. The first part was the Servlet itself. It had to be able to interpret HTTP POST parameters and convert them to X10JDTk commands. The second part was the HTML front end of the web app. Focus was placed primarily on the Servlet since we thought it could be used as part of the X10JDTk. Once completed, programmers can use this Servlet to develop web applications of their own. Consequently, the HTML web front end was designed to illustrate the basic functionality of the Servlet.

D. PROBLEMS AND SOLUTIONS

This section of the report explains some of the problems that arose during the design process and the methods that we used to overcome them. This is to be converted later into a Frequently Asked Questions guide in the user guide for our product when marketed.

D.1. Problem: Emulator, Solution: Real Palm

At the onset of the project our biggest problem was that none of the group members was the owner of a Palm Pilot. Our initial solution for this was to use the Palm emulator that was downloadable from the Palm Pilot web site for software developers. We used the ROM emulator to try and communicate with the X10 base stations using the Firecracker. We ran into some problems here since the emulator would not communicate with the Firecracker. At this time, one of our group members was able to procure a Palm VII, and the Georgia Tech ECE department was able to provide us with another one. On working through some other issues with pin connections (discussed below), we found that we were able to overcome this problem by using a Palm Pilot rather than the emulator.

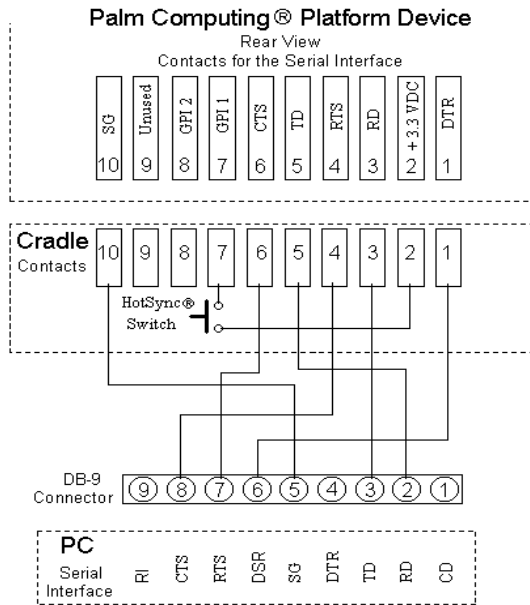
D.2.Connect the Palm Cradle to the Fire Cracker correctly. Solution: had to solder correct pins since there was not a 1 to 1 correlation.

Having obtained a Palm Pilot, our first objective was to establish reliable communication between the Palm and the Firecracker. Our initial attempts to attach the serial port Firecracker to the output pins of the Palm failed to set up the necessary communications. We obtained the pin-outs of the cradle and the Firecracker, and the protocol used by the Firecracker to send commands to the base station. On analyzing the pin-outs, we found that the problem was that there was no one to one correlation between the two sets of pins. In fact the Palm did not have a programmable DTR line which was required by the Firecracker. We had to open up the cradle and solder temporary connections between the cradle and the Firecracker to find out the best way to connect the Palm so that when using the HotSync on the Palm, we could communicate with the X10 base station. We were also able to later connect the pins directly from the output of the cradle without having to solder every time. At this point we also realized that a problem we would have later would be the lack of the DTR line.

Table 1. The right pin-connections.

DB9 – Cradle	DB9-FireCracker
Pin 2-TXD	Pin 4-DTR
Pin 5-GND	Pin 5- GND
Pin 8-RTS	Pin 7-RTS

Figure 2. The original incorrect connections we tried



D.3. Problem: Java, Solution: Native Code better since the only form of translation we need is what Codewarrior does instead of having to use Java API's

Having established communication, our next move was to start writing the software that would actually send the commands outlined in the Firecracker protocol. Our initial impulse was to find a Java compiler for the Palm since all of us were familiar with Java, and none of us had any knowledge of either C or CodeWarrior, which is the Palm's native compiler. We tried two different Java compilers for the Palm, Jump Java and Ghost Machine. However, we were unable to get either to compile even the most simple of programs. We tried to contact the owners of the two compilers to find out if we were doing something wrong, but got no response. Having failed in our attempts to find a Java compiler, we turned to CodeWarrior. CodeWarrior is a C compiler which produces '.prc' files which are Palm executable. At this time the group members also started to learn C in order to be able to produce the software.

D.4. Problem: Lack of programmable DTR line, Solution: Use the TXD line instead

As mentioned above, one of the biggest problems was that the Firecracker used the DTR line to communicate with the Palm, however, the Palm does not have a programmable DTR line. We decided that since we were going to have to program the lines to send the requisite commands, we could try using the TXD line instead to send the same commands that the DTR line sent out. So in order to connect the Firecracker to a Palm Pilot we had to connect the Palm Pilot's TXD line in place of the DTR line going to the Firecracker. We left the DTR line unconnected, such that we had only the GND, TXD (going to where DTR is supposed to be), and RTS going to the Firecracker.

D.5. Problem: Programming of Dragonball registers

Another problem we had with the Java API was that the TXD and RTS lines could not be manipulated. So we had to go through the Dragonball registers to control the commands sent out by the software. There are three memory-mapped bytes that are used to control each port; select, dir, and data. The select register is used to change whether a port (or certain bits of a port) is connected to internal peripherals, or if it is just a general purpose I/O pin. The dir register controls whether it is an input pin or an output pin. And the data pin is used to read/write bits to the port.

The TXD and RTS lines are normally connected to the UART (select register bits are 0). With the select register, however, we choose to make them into general-purpose I/O pins and we can then set them high or low at will using the following code.

```
/* detach RTS from UART and make it into a GP I/O */  
SET_BIT(pRegisters->portMSelect, UART_RTS_LINE);
```

To make the pins into latching output pins, we then used the dir register.

```
/* set RTS to be an output */  
SET_BIT(pRegisters->portMDir, UART_RTS_LINE);
```

Then to set the pin high or low we used the data register.

```
/* make the pin high */  
CLEAR_BIT(pRegisters->portMData, UART_RTS_LINE);  
  
/* make the pin low */  
SET_BIT(pRegisters->portMData, UART_RTS_LINE);
```

The same thing is done with TXD, and then we just reconnected the pins to the UART by using the select register.

```
/* set RTS back to normal */  
CLEAR_BIT(pRegisters->portMSelect, UART_RTS_LINE);  
/* set TXD back to normal */  
CLEAR_BIT(pRegisters->portGSelect, UART_TXD_LINE);
```

D.6. CodeWarrior Compiling Problem

This problem actually pushed us back a little in the timeline for completing the Palm Pilot part of the project. We had several problems getting CodeWarrior to compile our software. David eventually called the people at CodeWarrior and found out this was due to the fact that we were importing our resource file incorrectly. What the resource file does is allow us to set up our GUI which is what happens in all C based programs. We defined our GUI in our .rcp file and were trying to bring it in using standard C methods. CodeWarrior does not allow so so we installed a program which brings in the .rcp file, changed the configuration to Codewarrior to allow these files, and downloaded the new SDK so we could use the proper header to control the Palm. After a little more tweaking with code we solved the problem. Anytime an error such as

“Rsrc line 1000 not found” is encountered make sure Codewarrior is bringing in your resource correctly.

D.7. Problem: Deciding how to interface the PC with the power line

This was not really a problem. It was more like a challenge. Due to limited amounts of funding, a large amount of research was required to find a cost effective way of bridging the PC and X10 devices. We already had a X10 wireless transmitter (FireCracker) but the required serial signals that the PC needed to send to the transmitter was beyond our programming abilities. Also, the wireless transmitter did not allow us to receive foreign X10 signals on the power line.

The transmitter we finally agreed on solved both our problems. Not only could it receive foreign X10 signals, the programming that required interfacing it to Java was within comprehension. The transmitter is composed of two separate devices. The part that attaches to the PC simplifies the required serial commands to transmit a X10 command. The part that attaches to the power line converts serial signal directly to X10 commands on the power line. The device was found to cost about \$100, which was well within our budget.

D.8. Problem: Deciding on the structure of the X10JDTk

It was difficult abstracting out the structure of the toolkit. We did not want the programmer using the toolkit to have to worry about the intricacies of interfacing Java to the adapter yet we did not want to restrict the programmer's ability to control different aspects of the toolkit. So a balance needed to be founded. We finally agreed upon a structure that balances the simplicity of use and power. X10 commands were to be taken in directly as text.

D.9. Problem: Home Surveillance not completed

The last problem we encountered was that we were unable to setup the Home Surveillance system in time for this presentation. We have the product ready to go, since it is an extension of the servlet that has been set up. We have only to complete debugging the application and set up the web page for displaying the pictures from the camera. We feel that given an extra week, we could complete this application. Regardless, documentation for installation of this application is included in this report and in the users guide provided with the product.

E. TESTING

This section of the report explains our procedures for testing the final products. Using the following tests we were able to verify that our products were sound, and were ready for the market.

E.1. Used the oscilloscope to see the voltage level change when a Hotsync button was pressed on the Palm which went through the firecracker

In order to ensure that we had sufficient voltage flowing from the Palm to the Firecracker, we connected the Palm to an oscilloscope and used the HotSync button on the Palm to measure the voltage level. The voltage was found to be 3.3V, which was sufficient for our product.

In addition, we were going to verify that the frequency of the X10 devices did not conflict via EMC testing located in Norcross, Ga. The IEEE post out there granted us

permission and services valued at \$2000, but we did not have enough time to be able to use this service. This will be added in future modifications.

E.2. Verify signal from Palm to Firecracker by indicated by link lights.

After soldering the wires from the Palm to the Firecracker, and ascertaining the correct connections, we decided to just connect the Firecracker directly to the cradle instead of soldering every time we needed to use it. In order to make sure that we had the connections going the correct way, we purchased an RS232 link indicator from Radio Shack. This device has link lights that light up when a signal is detected on the lines of the serial port devices. We connected this device between the Palm and the Firecracker and made sure we had the correct signals and therefore the right pins connected.

E.3. Verify from Palmpad remote to GUI on PC to talk to the electrical devices.

We found that the best way to check whether the commands that were being sent to the X10 devices could be sent through the computer was to read the commands on the GUI. Hence, we set up the GUI such that when the Palmpad remote (provided with the X10 devices) was used to turn a device on or off, the commands sent through the power line were also sent to the output on the screen. Thus we were able to verify that we could in fact use the GUI as we had designed it to send the commands we wanted.

E.4. Final testing with appliances, Palm and servlet

The last test we performed was actually setting up all the appliances using the X10 devices in the CoC. We connected the appliances and tested that our software worked using both the Palm and the servlet software that was written. We found that on connecting all the appliances, our product worked without a problem.

F. LOGISTICS

The following is the schedule we followed, note that we added items to it as we went through the project and projected timelines for the added items. Along with the projected timelines, we also have the dates we actually started and completed the items.

F.1. Timeline Schedule**Schedule and Progress**

Projected	Start Date	Projected Task	Completed
09/14/00	09/12/00	Figure out what codes need to be sent to Firecracker to control our transmitter	09/12/00
09/17/00	09/12/00	Figured out Java would not work on Palm emulator so we just switched to Code Warrior instead of trying to test on an actual Palm	09/20/00
09/19/00	09/16/00	See if we can detect signals from a desktop to our firecracker module using a home built circuit board and some LEDs	09/22/00
09/21/00		Obtain an actual Palm	09/20/00
10/01/00	09/12/00	Connect the Palm Cradle to the Firecracker correctly	09/19/00
10/01/00	09/12/00	Verify that signals can be sent from Palm to firecracker using RS232 adapter which can detect the TXD, DTR, and RST line	09/19/00
10/09/00	09/29/00	Write Design Review dictating is our project still feasible	10/10/00
10/9/00	11/3/00	Use CodeWarrior to control specific X10 modules. Allows us to program in the Palm's native code. Have to write front end GUI also. Big lag at this point due to problems described above.	11/25/00
10/17/00	11/15/00	Begin Home Surveillance via the Web In Earnest See conclusion below.	Incomplete
10/17/00	10/20/00	Find a computer that will host the server, has Internet access, can run Java Servlets and be able to receive X10 camera transmissions	11/07/00
10/19/00	11/04/00	Program the servlet on the server	11/15/00
10/25/00	10/31/00	Write front end GUI for the servlet	11/20/00
11/04/00	11/05/00	Use Motion Detector and Xcam2 to capture images and send them to server wirelessly for saving to a file.	12/03/00
11/05/00	11/15/00	Begin wrap up of project	12/04/00
12/06/00	12/04/00	Write project final report and create presentation.	12/06/00

Table 2. Timeline**F.2. Order of Completion**

As seen from the above timeline and actual completion dates, we ran into some problems and did not always meet our projected dates. However, we accounted for the same and solved the problems created by lack of time by moving up certain aspects of the project. For instance we started the GUI implementation simultaneously while working on the software for the Palm. At about the time that we realized Java was not going to work on the Palm, some of us worked on learning C before we even completely ruled Java out and decided on CodeWarrior. Overall, we were able to complete the project on time and make it marketable.

F.3. Individual Group Member Contribution**Chewy Chong:**

1. Structured the Development kit
2. Wrote the Development kit
3. Wrote the two applications for the Development kit

Innes Finch:

1. Assisted in Codewarrior compiling problem.
2. Wrote design goals for critical design review.
3. Put together the categories for final report.
4. Assisted in the proposal presentation (finding the necessary pictures)
5. Assisted in Hotsync of Palm Pilot
6. Assisted in all of the testing that we did, except for the servlet
7. Searched for necessary downloads, such as emulator, Codewarrior, etc.

Anuj Girdhar:

1. Set up the web page for the group to use complete with facilities for uploads etc.
2. Found the pin-outs for Palm and Firecracker.
3. Helped with figuring out problems with wiring of cradle to Firecracker, soldering of pins, and final pin configuration.
4. Helped with testing of Palm to X10 base station configuration through RS232, and testing of voltage levels of Palm.
5. Helped to put together presentation for proposal
6. Helped to work on trying to use Jump Java and Ghost Machine to compile projects for the Palm.
7. Learnt C to use for CodeWarrior, helped to write the modules for the software.
8. Worked on getting CodeWarrior to compile.
9. Compiled report for Critical Design Review.
10. Helped in final testing for the Palm to base station control of electrical appliances.
11. Compiled Final Project report.

Quang Pham:

1. Searched the hardware: x10, Palm pilot, costs
2. Register at palm.com to get the ROM, learn the emulator
3. Did program in the Jump software, and Java
4. Go hunting for the Palm 5, and Palm 7 (got one)
5. Find the pin connections, soldered from cradle to FireCracker
6. Learned CodeWarrior and testing the devices

David Williams:

1. Assisted in figuring out what pins to use to go from the Palm serial interface to the Palm since the one on Palm website were not right.

2. Bought signal detector from Radio Shack to show what signals are being used for Palm i.e. TXD, DTR, RTS
3. Ordered Codewarrior from Metroworks
4. Bought female to male adapter from Radio Shack to signals could run from firecracker to serial interface
5. Took all pictures for and created the PowerPoint presentation for the Design Review
6. Talked to EMC people about testing our project for frequency violations. Secured a promise to do that but did not get a chance to use it because of time violations
7. Took course online from www.metrowerks.com that assisted me in learning how to use Codewarrior
8. Helped write the C code that enabled us to control the Palm. Also figured out to use Codewarrior to compile properly for the Palm by adding a resource type to the compiler. Customized the Codewarrior so it would compile for our program that took a lot of research and interaction with people from Metrowerks. Also had to download new SDK for Codwarrior
9. Assisted in all preliminary, intermediate, and final stages of testing and documentation

G. INSTALLATION GUIDE

The installation guide is two-fold. The first part explains how to install the software on the Palm. The second explains the setting up of the servlet and the GUI based applications for the computer.

G.1 Installation for the Palm

The product will ship complete with compiled Palm Pilot usable file. The code that is used to develop the CodeWarrior compiled file will also be available. In order to modify the software, a user has only to create a new project within CodeWarrior and then modify it as needed. In order to install the compiled file 'Group1.prc', the user must open the Palm Desktop on his/her computer. Under the install menu, simply add the file 'Group1.prc' to the files to be added at the next Hotsync. After that, it is simply a matter of Hotsync'ing the Palm and the file will be added to the Palm Pilot complete with GUI. In order to connect the appliances to the X10 devices, simply plug the devices into the outlets on the X10 modules and set the appropriate house and appliance numbers. For additional information on connecting the appliances, refer to the documentation on the X10 web page.

G.2. Installation for the GUI on the Computer

This document will detail the steps required to install the X10JDTk. The following items are required for the toolkit. You will need to install the them before using the toolkit. Each of them will have their own installation guide.

Item	Location
JRE / JDK 1.2 or above	http://java.sun.com
Any Java Development Environment - JBuilder 4.0 (preferred)	<cdrom>:\x10jdk\jb4fndwin.zip
Java Comm API 2.0 or above	<cdrom>:\x10jdk\javacomm20-win32.zip
Java Servlet Development Kit	<cdrom>:\x10jdk\jsdk2_1-win.zip
Jakarta-Tomcat Servlet Web Server	<cdrom>:\x10jdk\Jakarta-tomcat.zip

G.2.1. Adding Java Comm Functionality to your JDE and JRE

Since the X10 transmitter being used requires a serial connection on the PC, you will need to add the Java Comm API to your Java development and runtime environments (JDE and JRE respectively). The installation guide found in the Java Comm API will detail how to add Comm functionality to your JDE and JRE.

Important: The X10JDTk will not work without Comm functionality!!

G.2.2 Adding the Java Servlet Development Kit to your JDE

No one ever gets it right on the first try. Especially when you are programming in Java. By adding the JSDK to your JDE, you will be able to test out your Servlets while you code. The JSDK provides the necessary Servlet libraries for *code insight* abilities within your JDE. Also, the JSDK provides limited web serving abilities to test your Servlets directly on the web. Refer to the installation guide found in the JSDK installation package for installing the JSDK.

Notice: The JSDK is not necessary for the X10JDTk to function.

G.2.3 Installing the Tomcat Servlet Web Server

The Tomcat Servlet web server is a standard static content server add-on module. It allows regular web servers to offer Java Servlet functionality. Tomcat has limited static content serving abilities and is capable of regular web server functions (but not as fast). Refer to the installation guide found in the Tomcat installation package for installing the JSDK.

Notice: Tomcat is recommended for anyone attempting Servlet development and is not required for X10JDTk to function.

G.2.4 Installing the X10 Java Development Toolkit

The toolkit source files (.java) and class files (.class) have been compressed into a file called x10jdk.zip. The compressed file contains two subdirectories. One of them (HAPGateway) contains the code for the toolkit and the other (HAPServlet) contains the code for the toolkit's Servlet interface. All the source files contain documentation and comments.

H. MARKETING

The cost of our product is broken up into two parts. The cost of manpower hours put in by the group members, and the cost of the products that we had to buy in order to make our product work. The following tables show the costs of each.

Group Name	Average Hours Per Week	Total Hours for 15 Weeks
Chewy Chong	10	150
Innes Finch	8	120
Anuj Girdhar	8	120
Quang Pham	8	120
David Williams	8	120
Cost of each employee	Approx \$15 per hour each	Total cost for design= \$ 9,450

Table 3. Average Manpower Cost

Product	Cost of Product	Number Needed
X10 Firecracker	\$90 (Free for us)	1
Palm Pilot	\$400	1
ROM for emulator	(Free)	1
25 to 9 pin, male/female adaptors	\$5	4 (2 for Consumer)
CodeWarrior	\$119	1
Hawkeye Motion Sensor	\$20	1
Xcam Wireless Camera	\$80	1
RS232 devices	\$7	2
X10 Serial Port Interface Connector	\$100	1
Total Cost	To Developers	\$746
	To Consumers	\$707

Table 4. Cost of appliances bought.

Overall, we estimate the cost of developing our product to be approximately \$10,000. We are planning on providing the developers kit free of charge for all educational purposes, with a clause on receiving royalties on all products developed under it. In addition, for all commercial purposes, we plan on charging approximately \$1000 for unlimited use of our product. With these costs in mind, we feel that we should be able to pull ourselves out of the red within a few months.

I. CONCLUSION

In conclusion, we feel that our product is completely sound, and has great market value. The advantage of our product is that there are no known competitors out there. The company that manufactures the X10 devices also produces a remote device called the Palmpad, and Smarthome come closest to matching our project. However, the advantages we have over them are twofold. First, we are able to run all devices using the Palm Pilot; this is an obvious advantage, as users are more likely to carry a Palm around with them, than a Palmpad remote (often in addition to a Palm they may already be carrying). Secondly, our project allows for greater development of products. The code is open source and we are providing for development by others for application in several ways. With our software, we could develop it for more complicated applications such as TV control, VCR setup and usage, stereo control, and even control of applications on the computer. In particular, we will provide installation instructions on setting up Home Surveillance. Even though we were unable to complete that application in time for this presentation, we have the system almost complete and ready to go within a week. Hence we feel that there is no better product available in the market.

J. DOCUMENTATION AND SOURCES

The following are the resources used by us in completing this project.

1. <http://www.x10.com> – web page for the X10 devices
2. <http://www.palmos.com> – Palm Pilot resource
3. <http://www.3com.com> – 3Com’s web page
4. <http://www.palmos.com/dev/tech/palmhardware/> - Link to the pin-outs for the Palm
5. <http://www.theprescotts.com/firecracker/> - Java “Driver” for the FireCracker (Serial to RF Device)
6. http://www.x10.com/manuals/cm17a_proto.txt – FireCracker Serial Communications Specification
7. <http://java.sun.com/products/javacomm/> - Java Serial Communications API
8. <http://www.hewgill.com/pilot/index.html> – Java Jump for the Palm (Java -> Native Palm Program)
9. <http://www.cs.utah.edu/~mcdirmid/ghost/> - Ghost Machine used to run JAVA on small devices such as: Palm Pilot, mobile phone.
10. <http://www.palmos.com/dev/tech/tools/emulator/> - This is the website for the copilot or pilot emulator:
11. <http://www.palmos.com/dev/tech/palmhardware/> - Hardware schematics for the cradle to 9-pin serial port for the Palm
12. www.sun.com – JavaComm API and API documents found on Sun’s website
13. www.smarthome.com/1160.html – X10 Serial Port Interface Connector
14. Java Servlet API and Java Servlets book by O’Reilly
15. Documentation that came with the Smarhome adaptor specifying its use
16. PalmOS Programming for Dummies by Liz O’Hara and John Schettino

K. APPENDIX I
PALM PILOT CODE
GROUP1.prc

K. APPENDIX II
JAVA SERVLET CODE