

Amos Kittelson, Todd Denton, Tim Tewelde, Jake Peery: Group 5
EEL4914, 4 Aug 2010

aLife

Project Documentation

DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING



UNIVERSITY OF CENTRAL FLORIDA

EEL4914

Senior Design I

Team 5

Table of Contents

1 -Introduction.....	v
1.1 -Problem Statement.....	v
2 -Purpose.....	8
3 -Research and Component Selection	10
3.1 -Base Station	10
3.1.1 -Embedded Linux Kernel	10
3.1.2 -Android GUI	11
3.1.3 -Software Life-cycle Models	11
3.1.4 -Waterfall Model	11
3.1.4.1 -The V Model	12
3.1.4.2 -Spiral Model.....	13
3.1.4.3 -Agile Methods	14
3.1.5 -Communication Architecture	15
3.1.5.1 -Peer to Peer.....	15
3.1.5.2 -Client-Server	16
3.1.6 -Communication Protocols.....	17
3.1.6.1 -UDP.....	17
3.1.6.2 -TCP	18
3.1.7 -Database Management Systems	19
3.1.7.1 -IBM's DB2 Express-C.....	19
3.1.7.2 -Microsoft SQL Server Express	20
3.1.7.3 -Oracle Database Express Edition	20
3.1.7.4 -MySQL	20
3.1.7.5 -PostgreSQL	21
3.1.7.6 -SQLite	21
3.1.8 -Display.....	21
3.2 -Wireless Sense and Control Modules	22
3.2.1 -US Electric Power Consumption	22
3.2.2 -Appliance Ghost Power Usage.....	24
3.2.3 -Existing Smart Home Devices.....	25
3.2.3.1 -Smart Lighting.....	27
3.2.3.2 -Security Systems and Access Control	29
3.2.3.3 -Appliance Power Monitoring and Shutoff.....	33
3.2.3.4 -Home Theater and Entertainment.....	34
3.2.3.5 -In-home Communication Systems.....	35
3.2.3.6 -Climate Control.....	36
3.2.3.7 -Irrigation.....	37
3.2.3.8 -Pet Care.....	38
3.2.4 -Communication Standards.....	39
3.2.4.1 -ZigBee.....	40
3.2.4.2 -Z-Wave.....	41
3.2.4.3 -X10.....	42

3.2.4.4 -INSTEON.....	42
3.2.4.5 -Bluetooth.....	43
3.2.4.6 -WiFi	44
3.2.4.7 -Conclusions on Wireless Standards.....	44
3.2.5 -MCU.....	45
3.2.6 -ZigBee Transceiver.....	45
3.3 -Remote Client Hardware.....	45
3.3.1 -Apple iPhone/iPod Touch.....	45
3.3.2 -Smart Phones vs Feature Phones and Basic Phones.....	46
3.3.3 -Android Based Phones.....	46
3.3.3.1 -MIPS Based hardware and other embedded devices.....	48
3.3.3.2 -Ford Sync.....	48
3.3.3.3 -MIPS based devices and set top boxes.....	49
3.3.4 -Multi-Tasking and Notifications.....	49
4 -Theory of Operation.....	51
4.1 -Base Station.....	51
4.2 -Wireless Sense and Control Modules	52
4.2.1 -General	52
4.2.2 -ZigBee Wireless Network	53
4.3 -Remote Client	56
5 -Feature and Performance Specifications	57
5.1 -Base Station	57
5.1.1 -Hardware	57
5.1.1.1 -Base Board.....	57
5.1.1.2 -ZigBee Base Station	57
5.1.2 -Software	57
5.2 -Wireless Sense and Control Modules	58
5.2.1 -Hardware	58
5.2.1.1 -MCU	58
5.2.1.2 -ZigBee Transceiver	58
5.2.1.3 -Control and Sensor Related	59
5.2.1.4 -Power Supply	59
5.2.1.5 -Package	60
5.2.2 -Software	60
5.2.2.1 -Main Program	60
5.2.2.2 -ZigBee Protocol Stack	60
5.3 -Remote Client	60
5.3.1 -Hardware	60
5.3.2 -Software	61
5.3.2.1 -aLife Service	61
5.3.2.2 -aLife GUI	61
6 -Design	62
6.1 -Base Station	62
6.1.1 -Hardware	62

6.1.1.1 -NXP LPC3250 Microcontroller62.....	63
6.1.1.2 -LPC3250 OEM Board	65
6.1.1.3 -QVGA Base Board	67
6.1.2 -Software	69
6.1.2.1 -Board Support Package	69
6.1.2.2 -Initial Design Prototype	70
6.1.2.3 -Final Design.....	71
6.1.2.4 -Base Station Class	75
6.1.2.5 -Device Class.....	76
6.1.2.6 -PowerDevice Class.....	77
6.1.2.7 -SecuirtyDevice Class	77
6.1.2.8 -ControlDevice Class	78
6.1.3 -Database	78
6.2 -Wireless Sense and Control Modules	80
6.2.1 -Hardware	80
6.2.1.1 -MCU	80
6.2.1.2 -Control and Sensor Related.....	85
6.2.1.3 -Power Supply.....	86
6.2.1.4 -Package	86
6.2.1.5 -Schematics.....	87
6.2.1.6 -Bill Of Materials	90
6.2.1.7 -ZigBee Protocol Stack	91
6.3 -Remote Client Device	92
6.3.1 -Hardware	92
6.3.1.1 -HTC G1 Dream.....	92
6.3.2 -Software	93
6.3.2.1 -User Interface	93
6.3.2.2 -Android Based Hardware	99
6.3.2.3 -Remote Client Operating Systems and Software	100
6.3.2.4 -Client/Server Communications	100
7 -Design Summary.....	107
7.1 -Wireless Sense and Control Modules.....	107
7.2 -Base Station: Hardware.....	114
7.3 -Base Station: Software.....	116
7.3.1 -Base Station Class.....	116
7.3.2 -Device Class.....	117
7.3.3 -PowerDevice Class.....	117
7.3.4 -SecuirtyDevice Class.....	118
7.3.5 -ControlDevice Class.....	118
7.3.6 -Database.....	119
7.3.7 -Remote Client: Software	120
7.3.8 -Remote Client Operating Systems and Software.....	120
8 -Prototyping	121
8.1 -Parts Acquisition	121

8.2 -Hardware Implementation.....	121
8.2.1 -Wireless Zigbee Base Station and Zigbee Module.....	121
8.2.2 -Base Station Hardware.....	122
8.3 -Software Implementation	122
8.3.1 -Coding.....	122
8.3.2 -Unit Testing	122
8.3.3 -Integration Testing	123
8.3.4 -Operation.....	123
9 -Testing and Evaluation	124
9.1 -Test Plan	124
9.1.1 -Base Station: Hardware Tests.....	125
9.1.1.1 -Component.....	125
9.1.1.2 -Base Board	129
9.1.2 -Software Tests.....	129
9.1.3 -Remote Client tests	132
9.1.4 -ZigBee Modules.....	133
9.2 -Evaluation Criteria	134
10 -Project Management.....	136
10.1 -Team Meetings.....	136
10.2 -Team Organization and Responsibilities.....	136
10.3 -Milestone chart	137
10.4 -Budget and Financing	139
11 -Final Summary	142
12 -Appendices	143
12.1 -Copyright permissions	143

1 - Introduction

1.1 - Problem Statement

In our hectic lives there exists an need: Simplify our lives, please! The aLife (Advanced Living Integration for Education) project will aim to help by taking information from in and around your home, run it through a filter and give your gray matter a break. The system will be smart enough to know when to give feedback to the user and when not too.

Peoples lives are increasingly complicated, especially in managing their homes. You have to remember the shopping list, when to take the kids to soccer practice, whether you locked the front door, if the garage door is closed, when the dog needs more water, to turn off the lights when you're not using them, turn the coffee pot off, keep the AC at the correct temp, and a multitude of other things that eventually lead to some degree of information overload and stress. While there are devices that can increase our standard of living and give us more ways to be lazy (clap on lights), what would be far more useful is a way to make managing our lives more efficient, with respect to time, electrical energy, and

mental energy. A "one stop shop" system of monitoring and control of all of the devices (and even some non electronics) in your home that only presents information to you when pertinent, allow you to set automatic settings for electronics such as lights, and even monitoring power consumption of electronics and disconnect them remotely, saving you money and stress when the electric bill comes. It's like autopilot for your home so you can focus more on living.

For our project, we will develop a prototype system with the potential to do all these things. It will consist of 3 major parts: An in home base station, wireless appliance control modules, and a remote user interface device such as a cell phone. The base station will act as a repository for all of the information about the status of devices in your house. It will pass information back in forth between the user interface and the modules that actually control your home appliances via Ethernet, ZigBee, and USB. The remote modules are designed to be simple, cost effective devices that monitor and control household appliances in a non intrusive way. They are designed to be as flexible as possible so they can be interfaced with a wide variety of common household items. They will each have a ZigBee transceiver for communicating with the base station and basic I/O hardware to perform monitoring and control functions. The user interfaces will be a wall mounted touch screen LCD that is hard wired to the base station, as well as applications that run on any any Android equipped cell phone to allow remote control connection with the base station over the Internet. The user interfaces will present the user with intuitive, concise menus that display information about items in your home and allow the user to change the operation of items in their home. Although there may be several items in your home that are connected, information about them will be integrated and displayed in a central Android app in order to keep the information organized. Some potential applications are:

- Monitor power consumption of appliances within the home and be able to physically disconnect them if they are using too much power (TV, coffee maker)
- Turn lights on and off, or set a lighting schedule if you're on vacation
- Monitor inside/outside temperatures and control heating/AC remotely
- Receive home security system status
- Use bar code scanner to add items to a common and synced shopping list
- Alert user if garage door is open after defined time
- Alert user if someone rings the doorbell
- Alert user if the pets food is running low for too long
- Lock or unlock pet door after pet enters the house after a defined time
- Alert user if entry doors are not locked after a defined time
- Alert user if the oven is on too long
- Turn on lights when client device comes into range
- Occupancy sensing to control energy savings
- Provide weather information from local sensors

- Alert user if sprinklers are running while its raining
- Alert user if client device leaves the range of the server
- Alert user to calendar events
- RFID tracking

There are an abundance of potential applications for managing your home life, and so in this project we will focus more on providing a flexible hardware and software foundation that allows the user to control almost whatever they want. We will pick some basic applications to demonstrate the system, but there is the potential to do much more.

2 - Purpose

Why use Android, when there are a host of other embedded solutions on the market today? Certainly there are other technologies on the market that would cost less to implement while providing an adequate GUI for the user. Other solutions are well established, simple to program and easy to implement projects while providing a wide range of options. Any number of high level languages and GUI design programs can be used with the various options on the market. So, again, why use Android?

Number one, Android is an open source project, bringing with it the open source community that is constantly updating and improving the OS. Open source has the benefit of being vetted, tested and hardened by a wide range of contributors around the world. Linux, an open source project, is a good example of a project that has a proven track record of the benefits capable of being produced by a wide range of people. Internet security is one that Linux has arguably benefited from the most. While Microsoft and Apple are in a constant battle against Internet bad guys, they are caught in a perpetual game of cat and mouse. Patches can take weeks, sometimes months to fix security holes in their operating systems. Each patch must go through a rigorous vetting process to ensure the patch doesn't create new holes or cause other problems. This process is expensive and time consuming. Linux, on the other hand, has been known to have problems fixed within days of the discovery of a vulnerability. With so many devices being attached to the Internet these days, Internet security must be priority number one.

Besides providing quick security updates, the open source community is always attempting to make a product better. For example, while many Android handset owners must wait for their carriers to sent over-the-air updates to their phones to receive upgrades, the open source community is constantly providing updates to Android and making them available to all users. Phones that would have been made obsolete by Android updates have been given new life thanks to the hard work of hundreds of contributors. For example, the HTC G1, the Android phone released, has a slow processor and a small amount memory compared to Android phones on the market just a year later. As Google continued to develop Android its hardware requirements out grew those of the G1. With the help of the open source community, the latest versions of Android are available for installation on the G1, and they run just fine. Using an open source OS in our project helps ensure that lifespan of our project won't be cut short by the natural evolution of the platform.

A project as expandable as ours requires flexibility to grow. After the initial development new ideas might require new hardware, different module interfaces or an updated Internet protocol. Android is an all-in-one solution. It has everything from audio drivers to power management, GPS to WiFi and video camera to barcode reader. No other solution on the market is so complete.

Operating system flexibility is important, as is the application running on it. Android applications are programmed in Java. Java is flexible enough to run across many platforms, independent of the operating system. While Android has many technical significances that motivated our team to use it in our project, Android is relatively new to the embedded platform scene. Part of our motivation is in the development of Android with external hardware, such as the ZigBee based sensor modules we'll be using.

3 - Research and Component Selection

3.1 - Base Station

3.1.1 - Embedded Linux Kernel

The board support package (BSP) running on the LPC3250 was built by Embedded Artists based on NXP's Linux port for the LPC3250. Embedded Artists built the BSP based on the Linux kernel version 2.6.27.8 using the Linux Target Environment Builder (LTIB). It was designed to be as "lightweight" as possible and is only intended to implement the Hardware Abstraction Layer (HAL) for the system and run the Dalvik Virtual Machine (DVM) which Android runs on. The boot sequence consists of the kickstart loader initiating the sequence which then calls the Stage 1 boot loader (S1L), which after completing then calls u-boot version 2009.03 to boot the Linux operating system. The Linux operating system then starts the DVM (Illustration 1) and boots Android, which then is the primary application running on Linux.

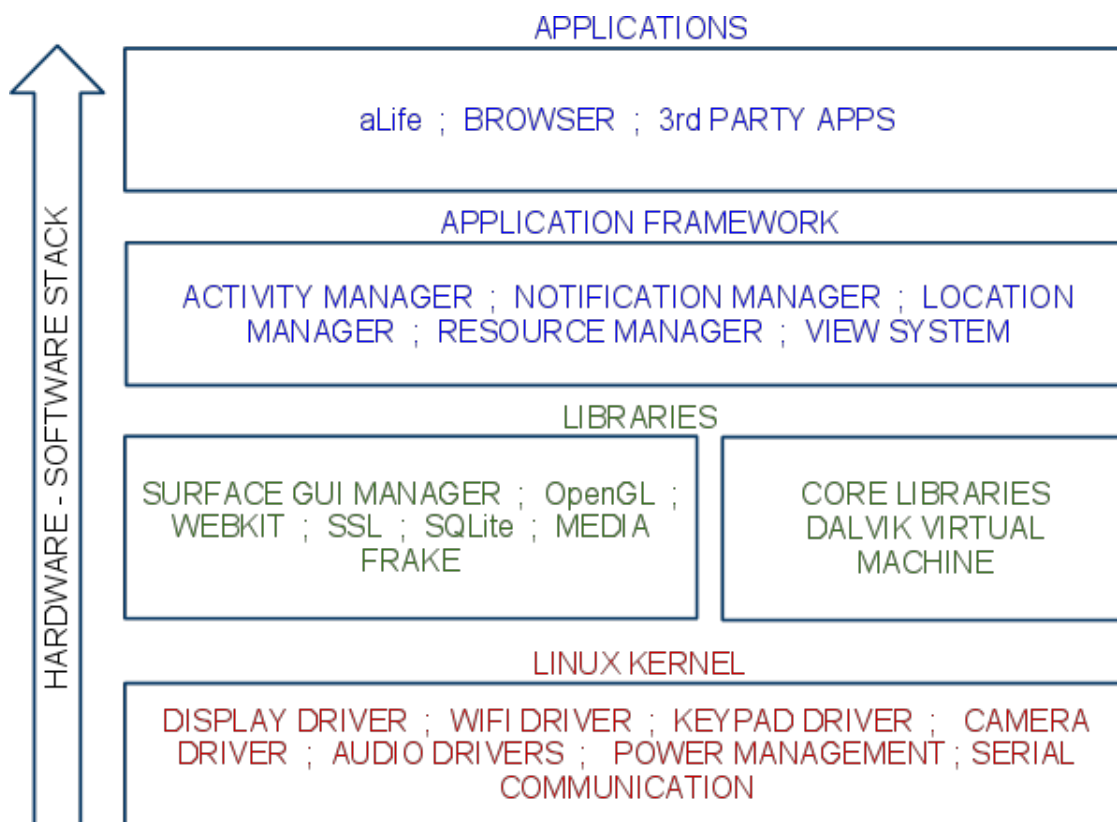


Illustration 1: Android OS Software Stack

The above Illustration gives a representation of what it would look like on boot up. This becomes important later on when we have to incorporate the Linux kernel to support the functioning of the board while the upper application layer,

Android, is used to support the remote and client functionality.

3.1.2 - Android GUI

The Embedded Artists LPC3250 development board runs the Android 1.5 (Cupcake) GUI on Linux through the DVM. We will interface with Android and develop applications with the Android 1.5 Software Development Kit (SDK).

3.1.3 - Software Life-cycle Models

Our project demanded that we create two major software systems: the remote client device software and the base station software. Creating these software systems requires our team to do extensive planning, designing, and unit testing to verify all our required specifications are met. These tasks can be very time consuming if not done in a systematic and efficient manner. For this reason our team decided it would be best to follow a well established software life cycle model for designing, implementing, and maintaining our code.

3.1.4 - Waterfall Model

The Waterfall model¹ was one of the earliest software models to be introduced (Royce 1970) and is still widely used today. The idea behind the model was to break the software development process into sequential stages, as depicted below in Illustration 2, where one developmental stage should be completed before the next one ends. Associated with each stage was a set of milestones and deliverables. These items were used to provide a model of how close a software team was to completion of their project. The Waterfall model is extremely simple and very helpful at providing guidance to software developers.

These are the qualities that made this model a very attractive choice for our team. The well defined structure of the model and due dates provided by the milestones also made it a very natural fit for a senior design project. There are, however, a few downsides to using the Waterfall model that neutralize some of its benefits. The most glaring weakness in the Waterfall model is its inability to deal with change during the development process. The Waterfall model was derived from the hardware world and provided a manufacturing approach to software engineering. This treated the creation of software like the assembling of a car in an assembly line. This approach is only effective if the problem you are trying to address is very well understood and a solution is easy to define. This is not the case for our group. We need flexibility with our software model as we may find it necessary to change our specifications during the development process.

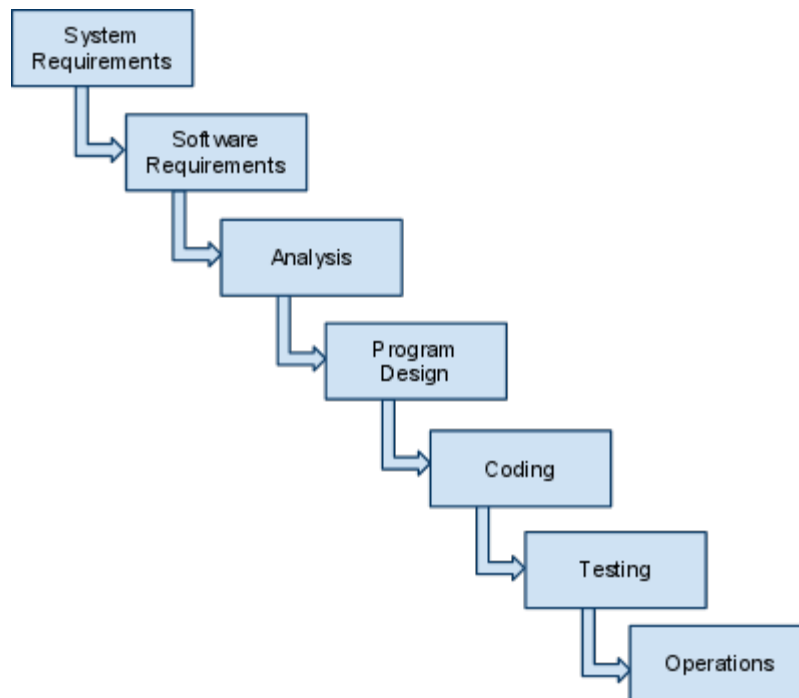


Illustration 2: The Waterfall Model

3.1.4.1 - The V Model

The V Model² is a variation of the waterfall model that better demonstrates the iterative process of software development. The model breaks down the development process into three sections as depicted below in 13: the analysis and design section (left side), the coding (middle), and the testing and maintenance (right side). Each stage of the development process still has clearly outlined objectives for completion yet your progression through the phases may not be linear. Redesign of your projects requirements, system design, or program design, are allowed based on the results of those stage's associated phase of testing. So in essence the V Model demonstrates all the good attributes from the Waterfall model without the issue of inflexibility. Also, the V Model's testing phases place a focus on meeting all design specifications or reworking them if necessary. This is very similar to the approach our team took towards the design of our hardware design making the V Model a very intuitive choice for our software development model.

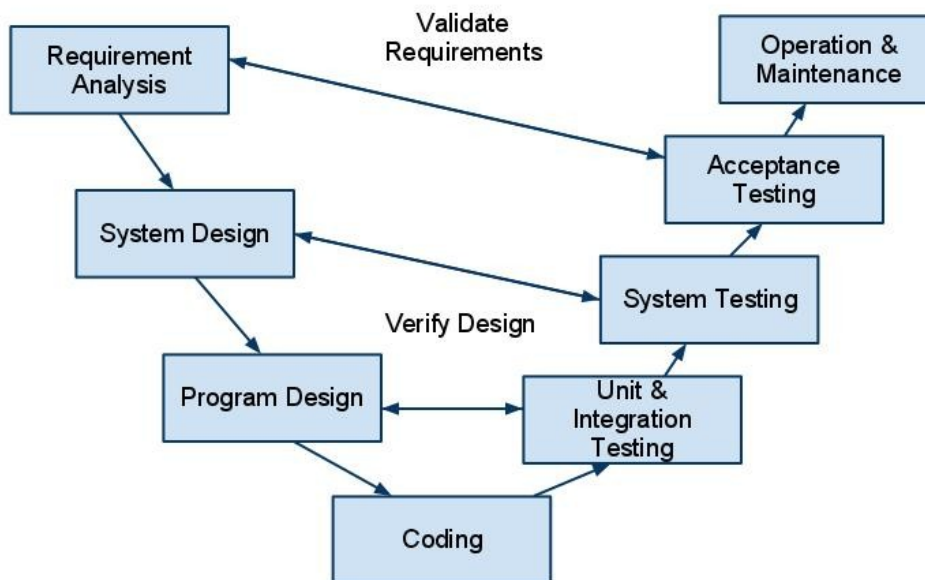


Illustration 3: The V Model

3.1.4.2 - Spiral Model

The spiral model³ is an iterative software model that focuses on prototyping and risk mitigation. This model is generally used on large scale, highly complicated software projects. The idea behind the model is to break the software development process into small, incremental stages where time is taken to evaluate any possible hazards a stage may cause before moving on to the next activity. Several prototypes of the product are made throughout the development process and analysed to see if any changes need to be made to that item which are then reworked in the following stage. This cyclic approach to software creation is where the name Spiral stems from. This meet our teams criteria of being a very well structured and proven software model. Our team also likes the fact that the model was very focused on meeting all specifications, even if they needed to be reworked several times throughout the development process. The downside to the Spiral Model is the length of time and workload demanded to finalize a software product. Due to the scale of our software project and timeline to get our product completed it would not be very practical for our team to spend as much time prototyping and reworking our design. The time taken in the Spiral Model would work better for a product that was going to market where the extra

scrutiny involved in the prototyping stages would be a worthwhile investment as it would generally lead to a product that is well received. This made the Spiral Model an unattractive choice for our team.

3.1.4.3 - Agile Methods

Agile software⁴ development methods are quite different in nature than traditional software models. Most traditional software models make an attempt to create a relatively rigid development structure that progresses in a linear nature. Responsibilities are divided amongst the software team and collaboration doesn't occur until the integration process. The completion of a project is usually marked by a set of deliverables and there is a focus on project management. This is not the case for Agile methodologies. Agile methods promote a speedy, collaborative approach to software development where solutions are created by a self organized cross-functional team. Agile methods believe their should be flexibility in the software development process and that one can satisfy a customer by "early and continuous delivery of valuable software" (Agile Alliance 2001). This approach to software development is very fitting for a senior design project. It deals with the problem of collaborating with a dynamic software team and focuses a lot of it's energy towards the speedy completion of a software project. This lead our team to review two agile approaches, Extreme Programming and SCRUM.

Extreme Programming⁵ tries to improve the quality of software by quickly responding to the changing needs of their customers. This is done by quickly launching their product into production and continually providing small "realease" or "updates" that address any bugs or missing features. Programming can be done in pairs where each individual is responsible for designing, reviewing, coding, and testing their or their partners software. There is a flat management structure and the programming team is discouraged from programming any feature or service until it's actually needed. This methodology would work well for our team. Working in groups without a formal management structure would allow us to leverage each of our team members software strengths. As well, focusing only on the features that will make it to our prototyped model will allow our team to save time and work efficiently. We are, however, concerned will the lack of structure that is provided by extreme programming and it's lack of focus on meeting all required specifications. In order for our team to have a successful senior design project it would be necessary for us to have adequate documentation, clearly defined specifications, and a well tested, complete final product. These downside really make extreme programming an unfeasible option.

SCRUM⁶ has a lot of similarity to Extreme Programming. It's focus is once again on speedy completion of a software product in a very flexible and dynamic work environment. Project's are broken down into several "sprints", usually around 30 days, where a group of prioritized requirements are worked to completion. These

requirements are worked in parallel by multiple self-organized teams who stay coordinated by attending brief daily meetings known as "scrums". Once again there is a flat management structure and the focus is on the changing needs of the customers. This approach meets our team's needs when it comes to completing our project in a speedy, incremental fashion and makes it easy to deal with unexpected changes during the development process. The approach does,, however, faces many of the same downsides of Extreme Programming.

One issue is that we would not be able to meet on a daily basis which may lead to an issue in coordination. SCRUM, like Extreme Programming, doesn't place enough focus on the building of software specifications and documenting the development process. This make it an unattractive choice for a senior design project.

After carefully evaluating all of the previously mentioned software models our team decided to follow the V Model. The V Model is very well structured and allows for redesign, unlike the Waterfall model. The V Model isn't as time consuming as the Spiral Model and doesn't have any of the previously listed downsides of the Agile Methods. We felt the V Model provided the best balance of speed and structure making it the clear winner for our team.

3.1.5 - Communication Architecture

It was decided that there were two possible paradigms we could follow to manage the communication between the Base Station Software and the Remote Client Device(s), the Peer to Peer Model or the Client-Server Model. We needed to make sure that the chosen model would not limit our ability to provide a reliable, secure connection between the devices nor degrade any services provided by our system. As well, we need to make sure the chosen model would not limit our option of communication protocol or database technology.

Whichever of the two models could most effectively meet these needs would be our team's selection.

3.1.5.1 - Peer to Peer

A peer to peer^z architecture is a distributed network architecture where each participating workstation has equivalent responsibilities and capabilities. All participants in the network have the ability to consume or provide resources without the need for centralized coordination instances. This would lighten the additional demands on the systems for adding additional nodes to the network as they would also add additional capacity to our system. This approach would allow our system to distribute the workload between the Base Station Software and the Remote Client Devices making our system more fault tolerant. It would also increase the amount of total users capable of interacting with our system simultaneously. Due to the decentralized nature of the peer to peer architecture our system would be able to have a more robust notification deliver system as we would no longer have a "one to many" scheme. This would allow a peer to notify all other peers of their activities without going through an intermediate centralized

server. The downside to this approach is that it would add complexity to the deliver notification system due to coordination issues. If a flooding, or flushing, approach was taken for delivering notifications then there could be several duplicate notifications provided to the user making our system less reliable. This would lead to the need to have acknowledgement sent back and forth throughout the distributed network adding additional load and complexity to the system.

Also, it may not be necessary to have every peer capable of processing the same request as the Base Station Software. If every node was capable of processing their own request for service it would turn into one large distributed computing system. In order for this implementation to be successful there are issue that would need to be dealt with. These issues include creating an algorithm for efficient world load distributions, synchronization of data and avoiding of race conditions, mutual exclusion, and many other issues that would add time and complexity to our design project. So, although the peer to peer architecture would allow our system to be more adaptive to additional load, more fault tolerant, and less reliant on a centralized server, it would also add complexity with issues such as coordination, workload distribution and mutual exclusion.

3.1.5.2 - Client-Server

The client-server⁸ model takes a different approach to distributed networking than peer to peer networks. The client-server model is a distributed network architecture where the tasks of the participation workstations are divided between service providers, or servers, and service requesters, also called clients. All service request run through the centralized server which has the responsibility for dealings with such issues as coordination, duplication, synchronization, prioritizing, and mutual exclusion. This approach would allow us to greatly reduce the complexity of the Remote Client Software as it would only need to be capable of sending request to the Base Station and receiving acknowledgements and notifications. Also, due to the centralized approach, only one device is receiving all requests making it easier to know if you are receiving any redundant request and dealing with them according. The downside to this is that the failure rate of our system would become directly proportional to the failure rate of the Base Station or Base Station Software making it the limiting factor to our system's overall reliability. The client-server model does, however, make it much simpler to identify which device in the network may be creating an issues as it divides the roles and responsibilities of all participants into either clients or servers. So, although the system may not be as robust when it comes to avoiding faults, it would make it easy to identify faults and make corrections as necessary. This is not the case for peer to peer networks. Since all nodes in the network have the same ability to service request it may make it difficult to pin point which device amongst the network created a failure making it that much more difficult to correct the issue. Another possible advantage demonstrated by the client-server model is its ease of administration when dealing with service

request. Having only one device deal with data storage and processing greatly reduces the overall complexity of the system and usually allows for easier security integration. Client-server technologies are also a very mature technology that is well documented and easier to administer than peer to peer networking. This is definitely an advantage as it would allow our team to spend less time administering the network and dealing with other portions of our design project. However, what the client-server models adds in simplicity it loses in robustness. Due to the centralized approach of the client-server model susceptibility to server overload becomes a factor. In the peer to peer model the distributed network grew in capacity with the addition of participants where the client-server model does not. These additional loads lead to an issue where an overload of simultaneous request for service are given to the server. This can lead to the server failing and the system being unable to complete service request.

So the decision became apparent that our team would have to choose between which was the lesser of the two evils. We had to decide between making our communication system extremely complicated yet very fault tolerant or easier to administer and implement and face the possible issue of work overload. We decided to go with the client-server model. We chose this because we felt the complexity of the peer to peer model was not worth the fault tolerance it was able to provide. Our network was intended only to connect members of one household making it unlikely that client-server model would have many issues with service overload. As well, the client-server model made the creation of software and dealing of coordination issue much simpler than the peer to peer approach making it clear winner for our team.

3.1.6 - Communication Protocols

Now that our team has settled the matter of what communication architecture to use, the next step was to choose what communication protocol to implement into our design. All communication from the Base Station Software to/from the Remote Client Devices and to/from the Database were going to be through the internet. This left our team with two protocols to choose from, TCP over IP or UDP over IP. Each protocol was available through the Android Platform so it came down to which of the two displayed the most benefits to our project. Our selection would be dependent on the protocols capability to provide secure and reliable connections, high quality of service (QoS), reliable data transfers, and ease of implementation.

3.1.6.1 - UDP

User Datagram Protocol⁹ (UDP) allows applications to send encapsulated IP datagrams without requiring prior communication to set up special transmission channels or data paths. UDP is a simple transmission protocol that focuses on

providing a best effort, connectionless services. UDP segments may be lost or sent out of order and there is no handshake or acknowledgements between the sender and receiver. This allows UDP to be a very fast protocol due to its lack of overhead and its avoidance of the redundancies involved in error detection/correction. What you do sacrifice on is the quality of service provided and reliability of your transfer. Having a fast protocol would definitely be beneficial to our team but it would be unreasonable for our connections to lose sensitive data during transmission. We would not want a service request to be misinterpreted due to loss of data. We could resolve this issue by adding mechanism to increase reliability between the sender and receiver. This would require our team to create error detection and correction algorithms to resolve any issue of lost or corrupt messages.. These mechanism could request retransmission of the message without user intervention. These mechanisms would add complexity and delay to the overall UDP scheme taking away from some of its key benefits. These mechanism are not required, however, and we could take the risk of data loss during our transmissions. UDP is widely used for client-server schemes that answer small queries from a large number of clients. This approach focuses on prompt delivery of messages over accurate transmissions. This would make it important for us to keep our messages very small as it would decrease the chances of us overflowing the UDP receive buffers which minimize packet loss.

3.1.6.2 - TCP

Transmission Control Protocol¹⁰ (TCP) allows applications to provide a reliable end-to-end byte stream over an unreliable internet network. TCP deals with many of the issues UDP ignores such as in-order byte streaming, guaranteeing message deliver, flow control, and connection oriented services. A TCP connection starts with a handshake between the sending and receiving devices. During this process the two sides agree upon flow control elements, data size, sequencing, and any other factors that would affect the transfer of data. Once agreed upon the connection is established between the two devices and kept until all data is reliably delivered in order. This approach stresses accurate delivery of data over timely delivery. This would definitely solve the issues our team had with possible packet loss using UDP but may lead to delays in servicing Remote Client Device requests. This delay will more than likely be negligible due to the relatively small amounts of data being transferred between our devices. Also, since TCP handles error detection and correction in the transport layer it would be unnecessary for our team to add these mechanism in the application layer. This would allow our team to save time allowing us to focus on other parts of our design. There is an issue that does arise with TCP circuit like connection that's established between two communicating devices. TCP connections were modeled after the phone lines where two devices are connected directly together through a virtual circuit. All data is routed through the same path passing through the same intermediate routers. If there is a failure that occurs at a router node during the connection

then the connection is destroyed and the data transmitted is lost. UDP does not set up a connection prior to data transfer relying on each datagram to find its own path to the receiving device. This lack of fault tolerance in TCP transmission could be a concern if there is ever any routing issues that occur during message transmissions. Below is Table 1, a comparison of TCP to UDP.

Item	TCP	UDP
Call Setup (Handshake)	Yes	No
Dedicated physical path	Yes	No
Each packet follows same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Potentially wasted bandwidth	Yes	No
Guaranteed QoS	Yes	No
Large header size	Yes	No
Fault Tolerant	No	Yes
Easy to Implement	Yes	Yes

Table 1: TCP vs UDP

It was decided that we would use TCP connections as opposed to UDP connections. The longer set up times needed with TCP were negligible when compared to the quality of service benefits provided by the protocol. We needed to make sure the communication between the Base Station Software and the Remote Client Devices was reliable and that all requests were received and fulfilled correctly. We feel as if TCP meets these needs effectively. TCP is also supported by the Android Platform making it an easy protocol to implement into our design. This made TCP a clear winner for our team.

3.1.7 - Database Management Systems

There are several options on the market today that can satisfy our team's need for database management. Our intention was to find a suitable option that would be compatible with Android, easy to use, and was low cost. This led to the review of the following software packages.

3.1.7.1 - IBM's DB2 Express-C

IBM DB2 Express-C¹¹ is both a XML database and relational database server management software developed by IBM. It is the free version of IBM's DB2 Enterprise Server Edition, a current leader in the database management market.

According to IBM.com "DB2 Express-C can be setup quickly, is easy-to-use, and includes self-managing capabilities. It also embodies all of the core features of the more scalable DB2 editions, including the revolutionary pureXML technology for powering a new breed of Web 2.0 and SOA based solutions". It is available for Linux (32/64 bit), Windows (32/64 bit), Solaris (64 bit Intel), and Mac OS X (64 bit Intel). These attributes made DB2 a strong choice for our server software.

Not all members of our team use Windows operating system, some use Linux and Mac OS X, both are supported by DB2. The software also meets our low cost requirement and is distributed by a leader in the current market. The software also allows your server to use up to 2 Cores on your computer (1 CPU), up to 2 Giga Bytes of RAM, places no database size limits, no connection limits, and no user limits.

3.1.7.2 - Microsoft SQL Server Express

Microsoft SQL Server Express¹² is a relational database management system developed by Microsoft. It is the free version of Microsoft's SQL Server that's targeted at small scale web server applications. Microsoft's goal was to make the software easy to use, including a robust graphical user interface, allowing for fast deployments of databases. Microsoft uses the same database engine as other, full versions of Microsoft SQL Server. It's only available for Windows (32/64 bit) operating systems, allows your server to utilize 1 CPU, up to 1 Giga Byte of RAM, and up to 10 gb of database storage. This would definitely meet the needs of our design project. We don't expect our database to require a large amount of storage and the easy to implement interface should assist our group as we do not have much database design experience.

3.1.7.3 - Oracle Database Express Edition

Oracle Database 10g Express¹³ is the free version of their popular Oracle Database 10g platform. Oracle's goal was to create an entry-level, lightweight database application that was easy to deploy and simple to administer. The software allows you to take advantage of such features as monitoring database activity and manage database users, storage memory and database objects. It will operate on Linux and Windows operating systems, allows your server to utilize 1 CPU, up to 1 Giga Byte of RAM, and up to 4 gb of database storage. It also comes with a simple graphical user interface and is very easy to deploy on your system. Once again these all meet the current needs of our design project.

3.1.7.4 - MySQL

MySQL¹⁴ is an open source relational database management system. It is the world's most popular open source database and runs on more than 20 platforms including Linux, Windows, Mac OS, Solaris, HP-UX, and IBM AIX. MySQL has support for triggers, cursors, SSL, full-text indexing, and query caching. MySQL does not ship with a graphical user interface for database administration, it does however come with command-line tools. This downside makes MySQL and

unattractive choice. There is third party front-end software available that will provide a graphical interface to assist with administration. This could resolve the issue but is still a more complicated solution than utilizing one of the other prior researched software packages.

3.1.7.5 - PostgreSQL

PostgreSQL¹⁵ is an open source object-relational database management system. According to postgresql.org "it runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others" and it does not impose a storage size limitation on your database. Just like MySQL postgres comes with command-line tools with the option to download additional software to provide a graphical front-end.

3.1.7.6 - SQLite

SQLite¹⁶ is an open source embedded relational database management system. Unlike the other database management software reviewed SQLite is not a standalone process, it is an in-process library that implements a self-contained, serverless SQL database engine. This allows SQLite to avoid inter-process communication making it faster than the more traditional models. SQLite also stores the entire database, including multiple tables, triggers, views, and indices, into one cross-platform file. This allows SQLite to be very lightweight which will work very well for an embedded environment. SQLite is also supported natively by the Android Platform which provided built in API's for managing the database. This will allow our team to host the database locally in the Base Station simplifying the overall design of our system. This definitely made SQLite a great choice for our design project.

After evaluating all the choices available our team decided to use SQLite as our database management system. Every package we reviewed met our needs for being low cost and easy to use, but SQLite had the best native support available from the Android platform. Also, SQLite will allow the Base Station to host the database eliminating the need for additional hardware.

3.1.8 - Display

One of the key features of this project is affordability. We want to design something that is both practical budget wise and performs competitively with things already on the market. There are always pros and cons to any consideration that is being decided upon for your final product, and the same held true for this consideration: LCD vs AMOLED for our screen choice.

LCD is a very commonly used term, and an even more commonly used piece of hardware. It can be found anywhere from televisions, computer monitors, clocks, and the list goes on. For this project the main considerations for the screen was efficiency and cost. The LCD is a very mature product with uses almost anywhere you look. For this reason it was a good obvious choice when trying to decide what screen to pick. The availability and reliability were both good factors when looking at it. However, when looking at it from a performance view there starts to be some downsides to getting one. The LCD itself can't be seen without some type of back lighting or external light shining on it, so the touchscreen we would be considering would have to have some kind of lighting already added into the screen to allow it to be visible to the user. This makes for an increase in the cost, and energy, of the produce without adding anything beneficial from it.

The AMOLED, which stands for Active-matrix organic LED. The OLED part is referring to the type of LED that no longer requires back lighting in it's design and implementation. The use individual LED for each of the pixels that you see on the screen, and this allows for much more in depth colors and a much faster refresh rate on images appearing on the screen. The technology itself is very good when you look at it's lightweight design and low power consumption. The problem comes when you start looking at the prices on some of these AMOLED ¹⁷ devices. Although it is true you could save money if you take into consideration the power that you are saving by using one of the screens it doesn't nearly outweigh the price for buying this thing, which could easily be equal or more than the price of the rest of the project all together.

Once all the research was done on this particular piece, with pricing, efficiency, reliability, and performance all in mind, we chose to go with the LCD TFT screen in our final product. Even though we had to sacrifice some performance in order to maintain cost, there were some definite advantages with choosing the LCD. Not only cost, but AMOLED screens are somewhat prone to degradation overtime, so maintenance would have to be more thorough if we chose to implement that device.

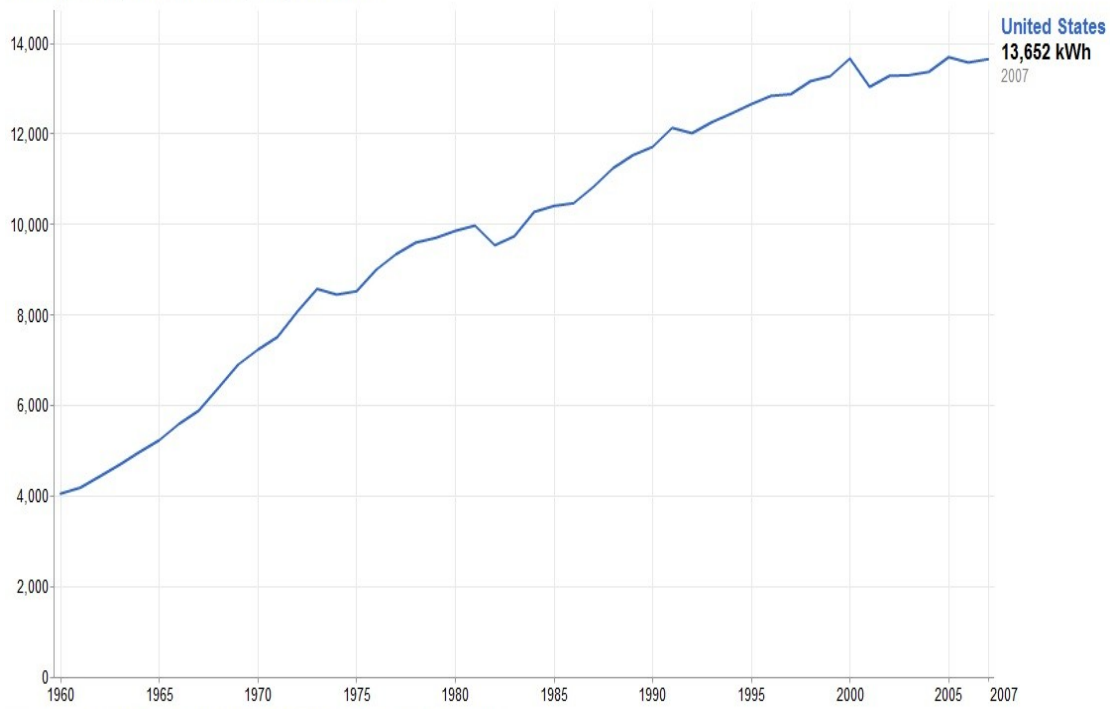
3.2 - Wireless Sense and Control Modules

3.2.1 - US Electric Power Consumption

In only the past few decades human dependence on electrical energy has grown exponentially. Homes that aren't even considered "smart" still have an abundance of electrical appliances of all kinds plugged into them. In fact, our homes have become a universal power and charging station for all of our appliances, tools and toys, from home theater systems, to computers, to cell phones, to coffee makers. As the number of electronic devices in the home has grown, so has home electricity bills. Illustration 4 below shows this increase in electrical power consumption from 1960 to 2007:

Electricity consumption per capita

Electricity consumption in kilowatt-hours per capita. [More info »](#)



Data source: [World Bank, World Development Indicators](#) - Last updated July 26, 2010

Illustration 4: US electrical consumption per capita 1960 - 2007

[18](#)

With the information about total energy consumption per capita now given, we can break down the information and relate it to the elements of the house that consumes the power. The following table, Table 2, relates figures taken from a 2008 survey of total power consumed in America and then gives the percentage of what appliance in the home used that power in relation to total energy consumed.

End-Use	Quadrillion BTU	Billion Kilowatt-hours	Share of Total
Space Cooling	0.77	227	16.50%
Lighting	0.72	212	15.40%
Water Heating	0.43	127	9.20%
Space Heating	0.42	123	8.90%
Refrigeration	0.38	110	8.00%
Televisions and Set-Top Boxes	0.35	101	7.30%
Clothes Dryers	0.26	77	5.60%
Computers and Related Equipment	0.17	49	3.60%
Cooking	0.11	31	2.20%
Dishwashers	0.09	27	2.00%
Freezers	0.08	23	1.70%
Clothes Washers	0.03	10	0.70%
Others	0.89	260	18.80%
Total Consumption	4.71	1379	

Table 2: breaks down electrical power used in the US in 2008 by application¹⁹

According to the US Energy Information Administration, in 2008 the average US home consumed 11,040 kWh, an average of 920 kWh per month²⁰ at a rate of \$0.1136/kWh²¹, for a total cost of \$1,254.44/year or \$104.51/month. Not only is this rate of consumption expensive, but it also has a huge environmental impact due to the fact that most of the electricity in the US is generated by burning coal or oil, or by nuclear reactors. Especially with the current economic, the US could definitely benefit from a reduction in electrical power consumption. It is no surprise that devices that reduce a households power consumption have gone from luxury items for the rich (who can afford the electricity in the first place), to practical tools for the middle and even lower class to use to reduce their cost of living.

3.2.2 - Appliance Ghost Power Usage

Electric plug-in devices use plenty of energy as it is, but what most people don't realize is that they can use power when supposedly turned off. This process is called ghost power consumption. It happens in one of two ways²² - Capacitive

voltage and Inductive voltage. Capacitive voltage happens when you have 2 lines running parallel with each other. If one is carrying an AC current then it will make the voltage on the other line be nearly equal to the one with the current if there isn't any interference from other wires in the vicinity. Even worse, if there are multiple wires around there could be a voltage division effect, meaning your device won't ever receive the proper amount of voltage that it should be from the outlet. Inductive voltage comes from the magnetic field produced by current running through wires. Both of these methods for sending voltage to wires that shouldn't get them are not good and can be reflected in your power bills. That's why power monitoring is so popular these days. If you have an old television that is pulling ghost voltage and costing you money each month can now be tracked and you will be alerted to that fact. There are also methods out there for monitoring power and representing it as a graph so you can monitor how your energy fluctuates monthly, daily, hourly, etc. This makes it a very powerful, and practical tool to be used in any type of home monitoring system. Going into the different ways that this power is monitored, the type like Kill A Watt uses, and that we will use, is the inductive sensing route. That route not only lets you sense and report the power directly, but it also allows you to break the circuit entirely, so you will no longer have any power running to the device and it won't be drawing any more ghost voltage. Some downsides are that now you have "two" power buttons - one at the wall outlet and one on the device, and that tends to become a little troublesome for the user. It also is a bit more invasive, having to physically plug this device into the wall and then plug the monitored device into this one to get an accurate reading. Although there are some difficulties to overcome with this approach, this is definitely a hot topic in today's energy conscious world and worth implementing in aLife.

3.2.3 - Existing Smart Home Devices

The number of electronics devices in the average household continues to go up, to the best way to reduce household power consumption is by cutting back on the power that those appliances use. There are a multitude of smart home devices in all all different price ranges that allow you to reduce your home power consumption, remote control appliances, or provide security. There are companies out there such as Crestron that will wire everything you want in your house up for you, and give you full control over it. And by everything I mean everything: anything you want or could want to control can be wired up to do so. A lot of the things you see them putting together you wouldn't even think of - lighting schemes throughout your whole house, garage doors/front doors, window shades, and televisions just to name a few. These systems provide the user complete control over their own house. The way this is accomplished is through the use of centralized stations that can be placed discretely in your house and have all other devices relay information straight to them. The downside to this system is that there is a lot of wiring between all these externally controlled systems: an example is a light switch and the relay station that

controls them. A few upsides for doing all of this though is that you actually have a company and trained professions come into your house to set it up for you. Most of the time the installation shouldn't take longer than a few weeks, and depending on the company²³ there would be some type of guarantee or at the very least a support option if something were to go wrong. As far as smart home devices go, this way really makes your entire house a smart home since it's all able to be controlled by you. Another good thing is that, again depending on the company, there can be some personalization depending on your needs or others that are living with you to some degree, and that can allow for a more comfortable feel and improve overall satisfaction in the product. However, a downside to this product is one of it's strongest selling points - the user has full control. Not much of this system is automated, and if something is suppose to be done the user has to do it explicitly. This can be a very good thing for people out there who like to be in control of most things, but for others this can be very troublesome because they would have to carry around some type of control panel with them when they wander the house just to turn the lights off to get ready for sleep.

The less extravagant approach is building your system part by part, possibly making homemade devices yourself if you are electronics capable or buying some already made ones. Going this route tends to lead to the more practical items and monitors - energy consumption, appliance control, doors/windows, etc. There are plenty of 3rd party vendors out there willing to sell you individual components that you can hook together to form a fairly robust home network, for example the types of devices you can find on the Black & Decker site that ²⁴ offer these features. One of the obvious downsides to this approach is the fact that you have to do all the instillation yourself, and if your vision of a house project is a unique one there might not be very much support for you as far as Internet research goes. Although more difficult in general to set up and manage, these systems offer a more cost effective solution than the whole house systems. Unfortunately, there is also the downside of how to get the devices to alert you to new messages or important information. From other research we have found that some of the 3rd party items already sit on the ZigBee/Z wave or maybe even both networks already, allowing them to be easily networked together and controlled.

Today it seems that for every piece of hardware in your house there is a "smart home" version of it, be it a door lock, thermostat, or light switch. Since electronic components have become so cheap, it is practical to embed electronics into almost anything, making "smart" home devices. Some systems are practical, such as remote lighting and appliance power monitors, and others are not so much. One of the main goals for the system is to incorporate as many popular and practical smart home features as possible, but still keep the hardware simple and as economical as possible so we will avoid including frivolous features. Including features from multiple device types means that the device types we will incorporate must be basically form factor agnostic, so devices like smart door

locks and smart thermostats won't be feature sets we will include.

After conducting some research on-line, the most common smart home device types are:

- smart lighting
- security systems and access control
- appliance power monitoring and shutoff
- home theater and entertainment
- phone and in home communication systems
- climate control
- irrigation
- pet care

The following subsections provide overview of each of these system types.

3.2.3.1 - Smart Lighting

This is the most basic and most common component of a smart home system. There are products made for post-construction homes which integrate on a plug and play basis, and there are products that are intended to be installed at the time of construction of the home. Since the aLife system is designed as a post-construction system we will focus on lighting systems of this type. There are smart lighting systems that support Insteon, ZigBee, Z-Wave, and X10 communication standards. Additionally, there are three basic levels of functionality in a smart lighting system. Level one (shown in Illustration 5) consists of a system that includes a number of wall mounted control units that the lights plug into and a remote that controls the wall units to switch the lights on, off, or dim. An example system is the SMARTHOME RemoteLinc - Insteon Lamp Control Kit which costs \$135 and features include multiple user dim presets, control multiple lights separately or together with the same remote, and 150 feet of control distance.²⁵

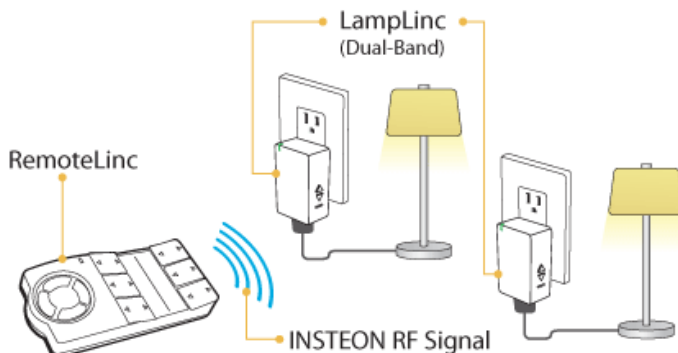


Illustration 5: RemoteLinc Insteon Lamp Control Kit

The intermediate level smart lighting system (shown in Illustration 6 below) consists of a wall mounted units similar to those in a basic system, except they have an infrared sensor and are controlled via a universal TV remote. An example system is the SMARTHOME brand IRLinc Receiver - IR to INSTEON Converter which costs \$100 and features one IR controlled wall unit and a mini remote. This particular system also has the added feature of a built in X10 transceiver that allows control of other X10 based ²⁷ lights throughout the house.

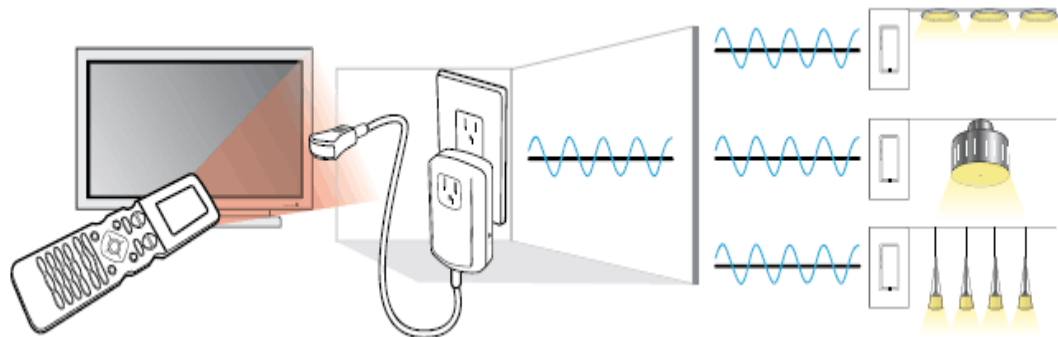


Illustration 6: IRLinc Receiver-IR to Insteon Converter

The high end type system (shown in Illustration 7 below) consists of wall mounted controller with an Ethernet port that connects to a router and allows lighting control with a smart phone, computer, PDA, or any other Internet connected consumer device. An example system is the SMARTHOME SmartLinc - INSTEON Central Controller which costs \$129 for the wall unit only²⁸. This system features stored light timers, device status feedback, and various other minor features.

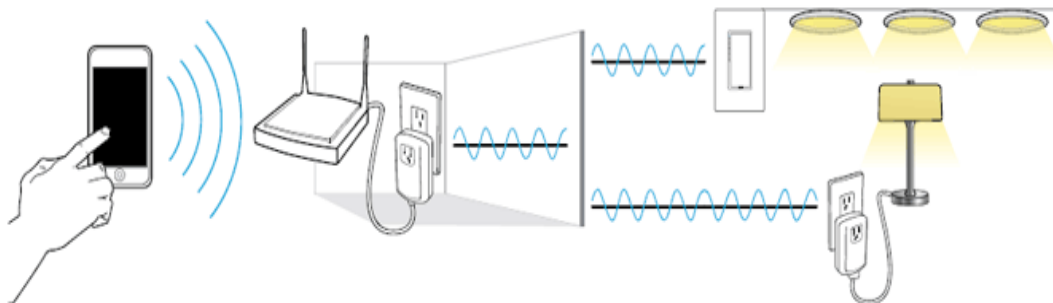


Illustration 7: SmartLinc - Insteon Central Controller

²⁹

There are also wireless light intensity sensors that work in conjunction with smart lighting control systems to maintain a specific level of lighting intensity, which is especially useful when natural light is used along with electric lighting. One such product is the NetVox Z301B ZigBee light sensor that sends on/off control signals to smart light dimmers and supports the ZigBee smart home automation profile.

Smart lighting is probably the most popular smart home feature, so we will include this functionality in our wireless sense and control modules. One of the most appealing features of smart lighting is that it is important not only because it is convenient, but because it also will play an important part in reducing power consumption throughout the home by reducing light levels when not necessary, and can also be a major feature in a home security system by illuminating the home when the residents are away on vacation or when approaching the home at night. We may even add hardware provisions for connecting an external light intensity sensor to further enhance the system. The on/off controller can also extend to other devices and can be used with a power monitor to shut an appliance off when not being used. One consequence of the decision to include the smart lighting feature is that this will require the wireless sense and control devices to be outlet mounted modules. This may prevent other functions to be implemented with this device, however the smart lighting takes priority since it so popular, has applications for other smart home systems, and because existing lighting control systems are very expensive relative to cost of building a system.

Since we are intending on building multi function remote modules in the first place, the added component cost of implementing this type of function will be minimal and we will be able to achieve a much higher feature/cost ratio than what you would get out of a current off the shelf smart lighting system. In essence, we get a lot of bang for our buck with this feature.

3.2.3.2 - Security Systems and Access Control

Security is an important topic today, and that is evident with all of the security systems available. Most of these security systems involve you installing a wall panel, some sensors around your house to monitor movement, windows/doors, etc, and then relate to their network when something is wrong.

ADT is one of the most popular home security system companies. Their systems range from \$300 to \$500 for installation plus around \$40 a month for monitoring.

Their most advanced home security system is the SAFEWATCH PRO RF which features the following:

- 1 Safewatch Pro 3000 Wireless Control Panel:
- 1 Safewatch Pro 3000 Standard Touchpad:
 - All ADT Safewatch® Pro 3000 security systems come with one Standard Touchpad. You can add additional touchpads to place in other areas of your property for additional convenience and to check the status of your system from multiple areas in your home.
- 2 Wireless Door/Window Sensors
 - These devices are recommended for all doors and windows at ground level, windows near trees or shrubs, or doors and windows that are dark or hidden from view.
- 1 PIR Motion Detector - Wireless:

- Can help detect movement in rooms, hallways and on stairs, and helps reduce false alarms. Our sophisticated technology outperforms other motion detectors.
- 1 Quick Key Remote:
 - This handy accessory allows you to arm or disarm your system, open your garage door, or even turn on the lights in your home - all from the palm of your hand. Its unique, one-button operation eliminates the need to remember codes and helps to reduce false alarms.
- 1 Indoor Sounder:
 - ADT Indoor Sounders use loud beeps to notify you to emergency conditions such as fire or intrusion. All ADT systems come with one sounder, but you can add additional sounders.
- 1 Power Supply and 24 Hour Battery:
 - Provides power to your system and an extra high-capacity battery in case of power outage.
- 1 ADT Window Decal and Yard Sign:
 - Warn potential intruders your home is protected by ADT.³⁰

ADT and other security system companies don't actually sell you the hardware, but instead rent it to you, and then charge you the installation fee and mandatory monthly monitoring fee. This means that you don't actually own the hardware and have to pay the installation fee again if you move.

There are also networks out there that allow you to be in control of your own security network that you set up yourself. Illustration 8 below is an example of Vera2, a Z-wave controlled hub that allows users to access their security systems through wi-fi in their homes.



Illustration 8: Example of the Vera system GUI³¹

This is the interface for the Vera system, and as we can see there are many features to the system. On this page we have access to a fan and a lamp for direct control, as well as other options on the left side for accessing any number of things. This particular device is only the hub for all other external devices in your home, and it alone can run upwards of \$250.00³². You can buy sensors to add onto this device separately. Some of those, depending upon what you get (cameras, sensors, etc) can be just as expensive per component.

For home security you don't always have to get these extravagant systems, but individual pieces to fit any type of need that you might have. One example of that is the Brinna PeepHole viewer, shown below in Illustration 8.



Illustration 9: LCD peep-hole device [33](#)

This device can be placed at your door instead of your normal peephole and produces the image of whoever is outside on the LCD screen. This particular device runs at around \$100.

For every component that makes up the ADT and similar security systems, there are wireless "smart home" versions of those products available. In fact, by integrating individual components you can create a custom security system that is tailored to your needs and has more functions than the ADT system. Example wireless ZigBee and Insteon based components are:

- Motion/occupancy sensor
- Remote controlled deadbolt
- Alarm sounder
- Window/door sensor
- Glass break sensor
- Smoke/carbon monoxide detector
- Flood sensor
- Smart security lighting system
- Keyfob system access remote
- Personal panic button

Security systems are arguably the most important systems in a smart home because they are much more than a luxury, they can potentially save lives of those in the household and protect their property. Systems like those implemented by ADT and Brinks are more traditional and have been around for several decades, however with the emergence of smart home devices and in home RF standards, an average person with minimal electronic knowledge can

assemble an "a la carte" style security system that rivals or exceeds the capabilities of a traditional system. We would like very much to include security system features in our hardware, unfortunately this is not possible because security system sensors and control units are very form factor specific. For example PIRs require a specific IR sensor geometry, smart door locks must be in the shape of a door lock, and window/door sensors require two separate parts. It would not be possible to implement several of these sensors into one reasonably sized package, and besides that, we have already decided on implementing smart lighting features which requires a wall outlet mounted package. Even though we will not be able to implement security system sensors in our modules, we will use a wireless protocol that will allow the relatively cheap 3rd party security system sensors to work on our wireless network in conjunction with our modules. One benefit to the smart lighting feature is that lights can be used as a part of a security system to illuminate the home before the user enters the house, or by keeping lights on timer when the user is on vacation. Another way we can use our modules to assist a security system is by including a relay on the modules that can be used to bypass a garage door opener near the garage door motor, giving the user the ability to open the garage door and turn on lights in the garage remotely upon approaching their home at night, or close the garage door remotely if the user left it open on accident while away from their home.

3.2.3.3 - Appliance Power Monitoring and Shutoff

Appliance control through power monitoring and shutoff is probably the most expandable and customizable part of the smart home system. The control comes from where the appliance plugs in to the wall. For the majority of implementations companies use wall plugs that allow appliances to be plugged into them. This allows for power drawn by the appliance to be measured directly, and offers a simple solution for cutting power to the device.

The greatest strength of devices of this type is that they provide real savings on electricity bills, so they appeal to frugal homeowners (especially in the current weak condition of the economy), and those who like being "green" and reducing their carbon footprint.

These devices tend to be either a per-outlet unit or an integrated entity of wall components. They tend to not be very expensive, \$20.00, but the higher priced models can offer features like 3-pronged connections, noise filters, among other features.³⁴ These are also the most directly controllable device in a smart home from outside your home. They are built on paradigm of sense and control, due to the user receiving some type of alert about a unit pulling too much power, as determined by the user and appliance it is connected to, and immediately providing a solution to solve the problem. Although there are more examples of devices other than wall plugs, since most things that you would want to sense power from and have the option to turn off are plugged directly into an outlet, they tend to dominate the market in use over alternative products.

An example of this type of system is the popular P3 Kill A Watt outlet plug in power monitor and control device (shown in Illustration 10). It costs around \$25 and features:

- Large LCD that displays power consumption statistics and metrics
- Displays power usage in volts, amps, watts, VA, Hz
- Accurate to 0.2%
- Cumulative kilowatt-hour monitor
- Appliance cost forecasting



Illustration 10: P3 Kill A Watt module

Appliance power monitors are probably the most practical devices in a smart home system because they actually have the power to pay for themselves and then some. They are especially popular today because of the movement to reduce our individual power consumption and carbon footprint. Therefore, we will make this one of the primary features of our modules. There is also the convenience of the form factor for this type of device being very similar to that of a smart lighting controller. We will actually leverage the smart lighting controller on/off hardware to switch off other appliances when the system deems that they are not in use but still are drawing an appreciable amount of ghost power.

3.2.3.4 - Home Theater and Entertainment

Just as there are universal remotes available to control your TV, DVD player, DVR, and stereo, there are similar remotes that do all of that plus control smart home devices. One such device is the Control4 System SR-250 remote control

(shown in Illustration 11) which not only controls your home theater system components, but also compatible ZigBee smart home devices.



Illustration 11: Control4 System SR-250 Remote Control³⁵

Home theaters and entertainment systems are probably the most popular types of home electronics in general, however they are of questionable tangible value in the home because they are basically a normal universal remote but with additional controls for wireless home network standards. Most users will already have a remote, and we will implement smart device remote control with a cell phone instead. Furthermore, universal smart remotes for controlling these systems are very form factor specific since they are best suited to being in traditional remote control form. Since we have already decided that our modules will be wall outlet mounted, we will not include this type of functionality in our system.

3.2.3.5 - In-home Communication Systems

There are various types and levels of in home communications systems:

- Doorbell chimes
- Audio and video Intercoms for placement throughout the house as well as at outside doors
- Even notification systems that integrate with other sensors and notify you via text message or phone call when there is an alarm event in the home

The Smart Home I/O Linc Insteon doorbell and telephone alert system (shown in Illustration 12) costs \$90 and provides user notification for up to 2 doorbells and one telephone line. This type of system is valuable because it can notify the user when someone is at the door when the user is out of hearing range of the doorbell or if you are in the backyard or garage and can't hear the phone. These notifications can be sent to a home base station, auxiliary chimes/sounders throughout the home, or to a cell phone.



Illustration 12: I/O Linc Insteon doorbell and telephone alert system

There are post home construction intercom devices that provide easy audio communication throughout the house as well as at the front door, however because of the heavy power demands of streaming audio they all draw their power from wall outlets and rely on built in home wiring for communication and do not interface with standard wireless standards. One of the major requirements for this project is to connect all devices through common smart home wireless standards, therefore we will not examine those types of systems for this project.

A remote doorbell device could be useful in the home, however it also has the problem of requiring a form factor which is incompatible with the rest of the features we will build into our system, therefore we will not include this functionality in our design. However, our in home wireless network will use a common smart home standard, so 3rd party devices of this type can be used with our aLife system.

3.2.3.6 - Climate Control

One of the key reasons why people like home automation is that it gives them a sense of control. One of the most valuable things to control are things that are costing you money, in this case in your house. Research into the area of power consumption will inform you that a great majority of a home owner's power bill comes from the home AC unit. This provides for a great need to control the thermostat to make sure that it isn't wasting energy, and costing the user money.

The range of control that you can have over a thermostat can vary from a programmable one to a remote accessible model. Programmable thermostats typically offer a temperature range that it will become operational, and some even offer different profile modes for days of the week. This offers a proactive choice for the user to have to help regulate the house. Unfortunately this doesn't help account for anything that might be happening currently, and also makes it very hard to track progress in saving money and energy on any type of continual basis since your only real source of information comes with the power bill at the beginning/end of each month

The type of thermostats that allow for direct control outside of the home are a bit more complicated and pricey. They tend to offer access to the popular home networking systems on the market today: Z-wave, ZigBee, Insteon and X10., and offer these items and part of a "whole home" type integration scheme. The amount of information sent back from these types can range from displaying the current temperature in your home to allowing you to change the settings from the Internet, and anywhere in between. They offer more meaningful data since you are able to check how long the unit has been running and allow for a more dynamic and informed setting of your home needs.

A good example of a "smart" thermostat is the VENSTAR INSTEON Remote Control Thermostat which costs \$200 and features:

- Remote control via any standard INSTEON wireless controller
- programmable temperature presets
- energy saving mode
- Reports changes in temperature, temperature set points, mode, and fan mode
- Supports +1, -1 degree on incremental bright/dim commands

Smart climate control systems are another valuable type of system with the potential for paying for itself by reducing the amount of energy used by the home.

It would compliment an array of appliance power monitors and smart lighting controllers, however it requires a form factor which is incompatible with our system goals. Thermostats must interface directly with heating/AC units, usually through pre-installed wiring at an eye level area on a major wall, and not down near the floor like where you find power outlets. That being said, it may be possible to adapt the base station to install in a standard thermostat location and include the necessary hardware for the module to operate as a smart thermostat.

We may decide to go that route later in the project, but we will opt to not pursue this at the initial design phase.

3.2.3.7 - Irrigation

Irrigation as a feature set of a smart home system isn't a complicated thing to accomplish. Other than special days of the week that you can water on and

current weather conditions, most of that operation can be done remotely and automatically without any direct user input. There are many different types of irrigation systems, and a good example of what this type of system requires is seen in the INSTEON 8-Zone Sprinkler Controller, shown below in Illustration 13.

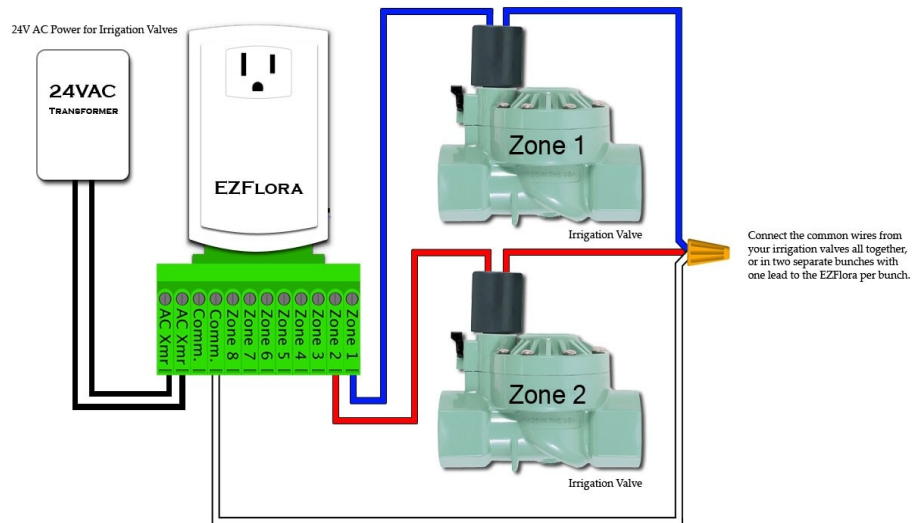


Illustration 13: Insteon 8-Zone Sprinkler Controller³⁷

The above figure shows the layout of what this particular component does. It has a wall adapter that it plugs into, and several configurable pins beneath it. Once you have set your sprinkler heads throughout the yard you run the wires back to this station, either as a solitary zone or multiple zones depending upon the user requirements, and then can activate them as seen fit. This control unit either receives commands from your computer or some stand alone device that you have downloaded the software for.

Irrigation is a relevant feature of a smart home and can help reduce waste water usage which is especially important in dry states as well as help maintain landscaping, but it is more of a secondary type of system. It is not practical to build in a complete irrigation system controller into our modules, however many irrigation systems have an external control loop, so we will be able to control those types of systems with the on board software controlled relay we will build into the remote modules.

3.2.3.8 - Pet Care

Having a highly integrated smart home that allows you to control almost everything remotely is no good if you still have to be home frequently to feed your pet or pay someone to watch your pet. PETCO sells automatic pet water bowls for between \$40 and \$70, and automatic ³⁸food dispensers such as

the PetMate Le Bistro Portion Control Automatic Feeder from \$35 to \$150 depending on the size of the system. The water fountains feature constant water circulation and carbon filtration. The automatic feeders feature variable portion size and programmable feeding times. These systems can even be combined with a smart home appliance remote controller such as the SMARTHOME INSTEON ApplianceLinc (shown in 39) which costs \$35 and allows the user to enable/disable the feeders or change the feeding times remotely.



Illustration 14: Insteon ApplianceLinc⁴⁰

As we have seen the smart home devices start as high as whole houses and go as small as wall outlets and light switches. This is what makes choosing aLife such a rewarding and challenging project. We are building into a field that has just recently started to become popular and affordable to most people, but we are building between the two established markets. We offer all of the same pre-existing smart home devices that any of these other companies offer while keeping it at a small enough level implementation wise and cost wise to stay competitive with the component by component people.

Automated pet care systems are in interesting product type that doesn't normally come to mind when you think of a smart home system. Unless you have a pet and are gone from your home most of the time, this type of system is secondary in priority and usefulness. Therefore, we will not include functionality for active pet care devices in our system. It is possible to do some kind of control with the appliance power on/off features of our modules and the addition of a dedicated irrigation app, but for now we will not implement that feature.

3.2.4 - Communication Standards

One of the most important goals of this project is to make an installation of the

aLife system non destructive to the home infrastructure and generally non invasive and easy to install. Wireless communications meet this goal the best, however there are a number of existing wireless standards that are optimized for smart home applications. Below is a summary of each standard with their respective strengths and weaknesses. Following the standard summaries, we will conclude with the selection of a particular standard that best meets the goals of our project.

3.2.4.1 - ZigBee

ZigBee provides a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4-2003 standard⁴¹. The standard was developed specifically for Wireless Home Area Networks (WHANs) for use in smart home devices such as thermostats, light switches, lamps, power monitors, etc. The networks operate in the ISM band, typically around 2.4GHz at data rates up to 250kb/s and at a transmission ranges from 10m to 75m depending on the environment. The ZigBee specification is intended for relatively low data rate, secure mesh networks. It was designed to be simpler and less expensive than other WHANs, and also to be very low power so that small battery powered devices can use the standard and still achieve long battery life. The ZigBee standard is managed by the ZigBee Alliance, an association of companies that have developed standards promote interoperability between ZigBee devices. The Radio Frequency for Consumer Electronics consortium worked with the ZigBee Alliance to develop the ZigBee RF4CE standard which provides a universal communications protocol for home audio, visual, and control products. That way different products from different manufacturers that adhere to the RF4CE standard can integrate their devices into one unified mesh network. The RF4CE standard includes various device profiles, two of the most important of which are the Home Automation profile and the Smart Energy profile. The Home Automation profile specifies a protocol for managing displays, thermostats, lighting occupancy sensors, security systems, and other similar devices. The Smart Energy profile specifies a communication protocol for monitoring and controlling devices such as smart lighting systems and climate controls.

A ZigBee network is essentially an ad-hoc wireless mesh network. There are three basic types of devices in a ZigBee network: ZigBee coordinator (ZC), ZigBee Router (ZR) and ZigBee End Device (ZED). Every ZigBee network one and only one ZC, as it's the device that establishes the ZigBee mesh network. It is also used as a network bridge to other networks. A ZR can serve an application function as well as serve as an intermediate router to pass data between other devices. A ZED functions as an application device only and will not relay data between other devices, therefore it requires the least amount of memory and is the cheapest type of device to manufacture. If the network is beacon enabled, ZED can remain asleep most of the time and only wake up when polled by the ZC. Networks can also either be beacon enabled or non beacon enabled. In non beacon enabled networks, ZR devices are on all the

time, listening to all activity on the network, while ZEDs may be asleep. While this enables faster response times by devices, they also consume more power.

In beacon enabled networks, ZR devices periodically send out beacons to confirm to the other nodes that it is present in the network, however ZR and ZED devices can sleep in between beacons. This reduces the amount of power consumed by the devices, however it may slightly increase the response time of nodes to polls by the ZC since they must first wake up. In both modes, the protocol is optimized to minimize transceiver activity and keep power consumption to a minimum, and wake up latency is around 15ms which is much less than other standards such as Bluetooth.

There are a number of positive aspects to the ZigBee standard. First of all, it was designed from the ground up for smart home applications. The RF4CE standard and various device profiles are very useful in that it ensures compatibility between different products from different manufacturers for different application types, making development of an integrated system easy. The standard is open source, and also uses portions of the EM spectrum that is free to use for private applications, and getting rights to use the technology for commercial uses is relatively inexpensive. Perhaps one of the most important features is that the standard was designed to be low power, which is already an important factor in device designs, and will be even more so in the future. There are currently a number of ZigBee based smart home devices available for various applications such as door locks, thermostats, and window and door sensors. There are also a number of inexpensive MCUs available with built in ZigBee transceivers and free protocol stacks.

The standard is not perfect, however. The biggest limitation is that there are not nearly as many smart home devices available for ZigBee as there are for Insteon and X10. ZigBee also operates in the same portion of spectrum as WiFi, so sometimes there are interference issues between those systems. And although the spectrum licensing is free for private applications, companies developing devices for commercial applications must pay licensing fees.

3.2.4.2 - Z-Wave

Z-wave is another type of wireless communication standard that works much like a ZigBee or Wifi connection does. It operates in the 900 MHz range, and asserts that this allows for better data communication. The ZigBee and Wifi networks both operate on the 2.4 GHz⁴² ranges, and sometimes there could be a conflict of data transmission if both are running at the same time in the same area. Z wave is also more structured to an at home network than ZigBee. Some of the downsides are that it's a newer technology and not many partners in the Z-wave alliance⁴³ are making this type of technology, so it's harder to locate 3rd party vendors or items that allow for "plug-and-play" type functionality that we want to demonstrate with our project. Other considerations such as power and performance seem to be a bit better with the Z wave. Companies like Black &

Decker, specifically Kwikset as a member of B&D, have joined both the ZigBee and Z wave alliances, allowing any module that operates on that type of network to be able to interact with a 3rd party device.⁴⁴ This makes it very generic and easy implementable, which is a key selling point for this type of a project.

3.2.4.3 - X10

X10 is the most popular smart home device communication standard and has been around since the late 1970s. X10 is not a truly wireless standard, instead it relies on transmitting a modulated digital signal over existing home power wiring. X10 devices are usually plug into the wall where a lamp or other appliance is installed. Typically, a 120kHz carrier signal is used and one bit is transmitted at the zero crossing of the 60Hz power signal. The digital signal consists of an address field and a command field that is sent from a controller device to a controllable device. Commands can be as simple as "turn on" or "turn off", or the maser can poll the slave device for sensor data such as temperature sensor readings.

The strengths of X10 are that it is the most common smart home device communication standard and there are an abundance of X10 based devices available on the market that can do just about anything you can think of in the home. It also uses pretty simple, cheap hardware so it will keep the overall cost of devices down. It also does not compete with any other in home communication standards like ZigBee does with WiFi, for instance.

There are also a number of important drawbacks and limitations to the X10 standard. Since X10 communication signals propagate over in home power wires, it is at the mercy of the topology of the in home wiring and devices must be placed at wall outlets. Often times, there are transformers that bridge different portions of the household wiring network which will greatly attenuate or block all together the X10 signal. Residual Current Devices designed to prevent electrical shocks also attenuate X10 signals that pass through them. Also, certain types of power supplies used in modern electronic equipment (such as computers, television sets, and satellite receivers) have built in capacitors that provide a low impedance path from the line to neutral, effectively diverting the high frequency X10 signal away from the hot line. Since X10 only transmits 120 bits per second, it is also a very slow protocol, and there is also no contention handling in the standard, so if two devices try to talk at the same time there will be a collision and data will be lost.

3.2.4.4 - INSTEON

Insteon is a common standard for connecting smart home devices. It is similar to the X10 standard and was intended to address the inherent limitations of X10 but also be backwards compatible with X10 devices. It is a two medium communications standard that uses a home's built in power wiring to communicate with other devices in a manner similar to X10, but also uses a RF

transceiver to communicate with other devices wirelessly. In an Insteon network, all devices are peers so they can transmit, receive, and repeat any message on the network. Some improvements over X10 is that all Insteon devices act as repeaters so the entire network gets a transmitted message, and an acknowledgement based protocol is used. If a device does not receive an acknowledgement after it transmits a message, it retransmits until it succeeds. The wired based protocol also uses phase shift keying and synchronized message repeats. The 60Hz power signal is used to coordinate message repetition so that even with collisions, the devices are able to resolve the original messages.

Insteon based smart home devices are becoming ubiquitous and seem to be replacing the X10 standard. The hardware required for an Insteon transceiver is more complicated than an X10 transceiver, however it is still cheap enough to allow devices to be made inexpensively. Insteon also achieves higher data rates than X10 and more reliable communications. The RF signals can also penetrate walls in a home and allow devices to be placed away from power outlets.

Insteon is a major improvement over X10 and is a suitable standard for modern devices, however it does have limitations. One important limitation is that most Insteon devices either operate in the wired mode or wireless mode but rarely both. Also, the wired based protocol is susceptible to high frequency interference on the power lines, and no troubleshooting tools are available to setting up Insteon networks as there are with X10 so corrections must be done trial and error.

3.2.4.5 - Bluetooth

Bluetooth is a very popular networking system these days. Almost all people with phones have a Bluetooth type of card installed in them, and there are many devices already out there that implement them, like the hands free headsets for cellphones that seem to be growing in popularity. As a network medium, Bluetooth has the connotation attached with it of being very reliable and secure. This is due to the built in 64 and 128 bit encryption systems established in the protocol. This makes it ideal for a security type of home networking system, and would add some type assurance to our product. However, there is a decently long list of downsides to using Bluetooth.⁴⁵ Bluetooth as a product was not meant to be used at distance: it was meant to eliminate the need for two nearby technologies to have to be cabled together. Also, since they both operate at the 2.4 GHz frequency, there is some similarity about data transmission, but that's about where the similarity ends. Bluetooth can only transmit at a maximum speed of 1 Megabit per second, which is nowhere near fast enough especially if you have a network extended throughout your entire house. The protocol stack is also 250Kbytes, which is roughly 10 times larger than that of the ZigBee network. Also, the most killer aspect of this network is the connection rate. ZigBee boasts a connection speed of 30 milliseconds, whereas Bluetooth can take upwards of

10 seconds to recognize a new device.

3.2.4.6 - WiFi

Wi-Fi is probably the most common and widely used wireless standard by far. It has revolutionized the way that we communicate and has been a front runner in the wireless communications field. It uses the IEEE 802.11 specifications, and sets up a point to hub connection for accessing the network. The way aLife would work if we implemented it using this method would be to have the base station as a router and have all the nodes connect to it much like you connect your computers or laptops to a wireless network at your home. Some of the downsides⁴⁶ of that are the same we have all experienced - slow connection speed and no real security. The data rates are a bit slower than ZigBee - 11 and 54 Mbits/sec are the two going rates - and Wifi has no built in security to speak of. The connection itself can be troublesome also, as it can sometimes require up to 3-5 seconds to locate the hub and connect to the Internet. Lastly with the constant connection being established to the "router", there is a very large power consumption as compared to the rest of the networks that we discussed so far in this section. One of the best things going for Wifi however is it's network size, which is nearly the same as the ZigBee one. It's also a widely used a known communication medium and would make for a very easy configuration and implementation of that device.

3.2.4.7 - Conclusions on Wireless Standards

The main requirements of the communication network by the aLife system are that the the network be nondestructive to the home, achieve reliable communications, relatively high overall network bandwidth compared to the bandwidth requirements of an individual device, have cheap transceiver hardware, facilitate low power system operation, and have sufficient information freely available for us to easily implement a robust system.

X10 is a very popular industry standard that would be cheap to implement, however it is antiquated and does not provide reliable enough communications, we did not select that standard. Z-Wave in theory is a good, modern standard with appropriate performance for our project, however it is closed source which would prevent us from being able to get enough information about the protocol to implement a true Z-Wave network so it will not work for us. Bluetooth is an inexpensive technology that has been widely developed, however it does not have the transmission range or network scalability that we require for our application, so it is out. WiFi networks are ubiquitous, can support very high data rates and good transmission ranges, and have free open source protocol stacks, however it requires expensive hardware and uses a lot of power, so it will not work for us. Insteon is a very appealing standard because it was designed for our application. It is relatively cheap to implement, has good transmission ranges and data rates, and has the added bonus of being compatible with X10 devices. There are also many devices available for Insteon which would allow us

to construct a cheap, robust system. However the main drawback to Insteon is that it has limited low power capabilities, and development tools and protocol stack availability is limited. ZigBee has all of the benefits of Insteon, plus more.

It has low power operation features, reliable communications, and the protocol is more robust in terms of controlling devices at a system wide profile level. There are also a number of cheap, powerful MCUs available with ZigBee transceivers built in as well as free powerful protocol stacks. The only real drawback to ZigBee is that it is not as widely used in smart home devices as X10 and Insteon based products, however it has a better foundation and more potential than the other standards. If anything, it would be a plus to use ZigBee because we can further the proliferation of a superior technology. It is for all these reasons that we select ZigBee as the wireless communications standard for our wireless sense and control modules.

3.2.5 - MCU

One of the goals for the wireless sense and control modules is to keep them as small and cheap as possible, while maintaining a flexible and diverse feature set. The modules don't require much processing power, so we selected an 8 bit MCU. In order to reduce the number of components on the boards and keep the design simple and cheap, we selected the Freescale MC13213 because also has a ZigBee transceiver built in, we are familiar with Freescale's free CodeWarrior IDE. Fortunately, we were also able to get 8 parts for free on sample.

3.2.6 - ZigBee Transceiver

The ZigBee transceiver in the MC13213 is a fully IEEE 802.15.4 standard-compliant MAC called the Simple Media Access Controller (SMAC) and is intended for simple MCUs. It has a typical RAM footprint of 3kB. As the name implies, it constitutes the MAC layer in the ZigBee stack. Freescale provides a free ZigBee protocol stack called the BeeStack which is designed to work with this transceiver and MCU. The transceiver requires an external oscillator, antenna, and various passive devices in order to implement the PHY layer and enable communication.

3.3 - Remote Client Hardware

3.3.1 - Apple iPhone/iPod Touch

The Apple iPhone/iPod Touch have the appropriate hardware to run the aLife remote client software. Since the iPhone and the iPod Touch can run the same applications, the hardware will be referred to as just the iPhone. The iPhone hardware configurations aren't as diverse as devices running Android. The minimum operating system for receiving push notifications on the iPhone is version 3.0. All iPhones will run all versions of the iPhone operating system so there aren't any iPhone hardware requirements to specify.

The iPod Touch doesn't have a cellular modem and thus isn't required to stay

awake at all times to receive notifications over the Internet. According the Apple support article HT35676, when the iPod touch screen is on and the device has a Wi-Fi connection then push notifications can be received at any time. But, if the iPod touch screen is asleep, then it will only check for notifications every 15 minutes. This limits the effectiveness of the iPod Touch to alert users to critical notifications.

3.3.2 - Smart Phones vs Feature Phones and Basic Phones

Smart phones are typically defined as phones that run a common operating system on different variations of hardware. iOS, Android, Windows Mobile and Symbian are examples of operating systems that would run on a multitude of different phones. They are expected to be able to run third party software.

Feature phones typically have a medium sized screen and a limited, unspecified, operating system that is customized for the phone. The phone will most likely have some built in applications, like email or Facebook, but may or may not run third party applications. Basic phones with their smaller screens may or may not run third party applications. If the feature phone or basic phone does run third party applications, they are most likely written in Java or BREW and applications do not run in the background. Since the aLife system requires real time communication with the aLife server, a feature or basic phone presents a problem.

One way to over come the real time shortcoming is to implement a system that uses SMS. The aLife server could send the user an SMS with information about the event. For example, and example SMS could be, "Hello, its after 10:00pm and your garage door is open." Upon receipt, the user could then launch the aLife remote client application and take action as necessary. This would allow a user with a less capable phone to still receive aLife notifications and take action as needed.

Since all phones send and receive SMS messages, then even the most basic phone could still be used to communicate with the aLife system using strictly the SMS service. For example, the aLife server could send an SMS such as, "Hello, its after 10:00pm and your garage door is open. Reply with "close" to close it." Another example: "Hello, the house temperature is 90 degrees and the system is off. Reply "Cool 80" to turn on the system and set to 80 degrees." This would allow all cellphones to communicate with the aLife system.

3.3.3 - Android Based Phones

One of our project members owns a HTC G1 Dream, one of the original Android handsets available on the market. It will be the main device used for testing the aLife remote client user interface. Currently, Google officially offers software upgrades up to Android version 1.6 in its stock configuration. Thanks to the open source nature of Android and the incredibly active Android open source community, the phone has been able to be upgraded to Android 2.1 The phone

specifications are as follows.

- 528Mhz Qualcomm 7201 processor
- 96MB Internal RAM
- 128MB Internal ROM
- OS: CyanogenMod 5.0.8-DS; Android 2.1

According to Andy Rubin of Google, the minimal requirements for all current versions (up to 2.2 as of this writing) of Android are 32 megabytes of RAM, 32 megabytes of flash, and a 200 megahertz online processor.⁴⁷

All major carriers in the United States are offering Android handsets. Currently, most phones that are available to purchase have hardware capable of running Android 2.1 (Eclair), while many are coming preloaded with Android 2.2 (Froyo). Android 2.1 brought about a significant user interface speed increase along with application speed performance boosts. The aLife user interface requires a fluid interface and quick response to keep the user's experience enjoyable.

The Android operating system is also available on Mobile Internet Devices (MID), sometimes termed tablets. These small devices normally are used as personal information devices, used for accessing content on the Internet and taking advantage of location based services. MID's are generally larger than consumer cellphones and aren't considered as portable as a cellphone would be. MID's are usually devices with screen sizes larger than 5" measured diagonally. They usually contain WiFi radios for Internet access with some offering 3G modems built in. MID's that run the Android operating system 2.1 or greater will be able to run the aLife remote client software like any other capable Android cellphone. The connectivity of the device will determine its usability in concert with the aLife central control unit. Since the aLife remote client software requires a Internet connection for all communications between the aLife central control unit, MID's that are limited to Wi-Fi connectivity will be limited to area's with WiFi coverage and must maintain the Wi-Fi connection. The lack of a cellular modem does not mean its not a device that shouldn't be considered for the aLife remote client software. MID's can be purchased for sub-\$200 prices. This makes it an attractive option for a device that is dedicated to a house or even a room.

An example of an MID is the Dell Streak (48). It has a 5" 800 x 480 capacitive touchscreen display, 1Ghz Qualcomm QSD8250 Snapdragon ARM processor, 512MB RAM, 512MB ROM, Wi-Fi and 3G GSM connectivity running Android 2.2.⁴⁸ The hardware specifications meet the minimum requirements for the aLife remote client software and would make it an excellent candidate for aLife use around the home or office. Its cellular modem ensures a constant connection to the Internet to receive aLife notifications and check on environmental and security conditions. Since the aLife remote client software is a service that runs in the background and Android supports multi-tasking, the Dell Streak can be used for viewing media content or surfing the web while still having the ability to

receive real time updates the the aLife central control unit.



Illustration 15: The Dell Streak (Courtesy of Dell Inc.) [49](#)

3.3.3.1 - MIPS Based hardware and other embedded devices

While the aLife system is designed for use primarily with smart phones, the remote client software has the capability to run on many software platforms besides those you would find in popular cellphones or MID's. Dozens of hardware and software platforms exist that have features that are attractive for use in embedded systems. These platforms can move the aLife remote client software out of the hand held device sphere, and into the rest of our environment. Cars with embedded platforms, appliances with built in touch screens and stationary touchscreen devices that install in the wall can all be used to interface with the aLife system. If the platform has a screen and a user interface, then it has the potential to also be a aLife remote client device.

3.3.3.2 - Ford Sync

The Ford Sync is a in-car communications and entertainment system installed in some new models of Ford cars. The system is based on an ARM 11 processor and runs the Microsoft Auto platform.⁵⁰ Different user interfaces are available, including an interface that is primarily text-to-speech based with a 2 line dot matrix display for visual display. In 2011 Ford will debut a technology that allows applications on a smart phone to interact directly with the Ford Sync system, allowing the smart phone to display text on Ford Sync display, convert text to speech and receive voice commands to control the smart phone application.⁵¹ This will allow the aLife user a hand free interface to the aLife remote client software while the smart phone is connected to the Ford Sync.

3.3.3.3 - MIPS based devices and set top boxes

The MIPS (Microprocessor without Interlocked Pipeline Stages) is a reduced instruction set computer instruction set architecture.⁵² The MIPS architecture can be found in many embedded devices like cable boxes, satellite boxes, flat screen TV's and mobile devices. Android itself is currently being ported to the MIPS architecture.⁵³ These architectures are all viable candidates to the aLife remote client software. Future versions of the aLife system could be based on MIPS hardware. As long as the hardware can run Android, then it can run the aLife software.

3.3.4 - Multi-Tasking and Notifications

Notifications are a critical component of the aLife system. Since the system is designed to run silently until a notification is sent to a user it is necessary for the receiving device to be able to receive those notifications at all times. If an important notification such as a security alert or a doorbell event are delayed by even 5 minutes then the system loses some of its effectiveness. Notifications on the iPhone are handled differently than on phones that run a multi-tasking operating system. The iPhone uses something called the Apple Push Notification Service. This is a service that is owned and operated by Apple and uses a WiFi or cellular data network connection to send push notifications to the iPhone operating system without the receiving application running in the background. The only requirement is that the application is run one time on the iPhone to setup the push notification identifiers in the iPhone. Once the push notifications have been setup in the iPhone then as long as the phone has an Internet connection then notifications can be received in the background.

Receipt of a notification can be indicated on the iPhone in three different ways:

- Sounds: Plays an audible alert
- Alerts: Displays an alert on the screen
- Badges: Displays an image/number on the application icon on the main menu

To weigh the pros and cons of each type of notification we must compare them to the severity of the notification and the current state of the iPhone. If the user is browsing the web on the iPhone and the aLife system send a notification indicating that the house security system has been tripped and the aLife remote client software is set to display a badge on the aLife icon, then the user won't know about the security notification until the main menu on the iPhone is accessed and the page with the aLife icon is displayed. Similarly, if the aLife remote client software is programmed to play an audible alert but the user is playing an sound intensive game on the iPhone, the alert may go unnoticed, or ignored until the user decides to check what the notification was. In the event of a notification requiring the immediate attention of the user this is undesired.

Notifications with high levels of importance should display a pop-up alert and play a sound to alert the user whether they're playing a game, browsing the web or carrying the phone in their pocket.

Notifications from the aLife system that don't require immediate attention could use less subtle notifications on the iPhone, but some dissection of notifications are still necessary. For example, if the notification is that the air conditioning has been running for an hour but the temperature in the house is continuing to rise, then we must decide how important this information is. The data suggests that there is either a problem with the air conditioning or that enough outside air is getting inside so that the air conditioning can't keep up. Does this warrant a pop up that interrupts the iPhone user or would an audible alert for the user to check at their convenience be sufficient? How about a door bell notification? This type of alert doesn't carry the same significance of a security alarm, but it does hint of the need for immediate attention.

4 - Theory of Operation

4.1 - Base Station

The Base Station Software is intended to be an "always on" process that works with little to no user intervention. The system is responsible for responding to service requests sent from a remote client devices in a reliable and time efficient manner. The software will also have the responsibility of periodically polling certain controlled devices and sending their information to it's specified table in our database. The software must also provide the user a basic security mechanism, basic log in and log out functionality, which will allow access into our system. The only feature that will be enabled to the user through the base station software will be the ability to add a device to the ZigBee network. Any additional services provided to the user will be done by the remote client device. The Base Station must also be able to accept full duplex TCP socket connection requests from one, or up to five users, with a high degree of reliability. After connecting with a remote client device the system must be able to handle any service request sent from the said device on a first come, first serve basis. Any duplicate request, simultaneous request, or unsuccessful requests must be handled accordingly and a notification must be sent to said remote devices if the issue was not resolved. The Base Station software is also responsible for sending notifications to one, or all, remote client devices. These notifications are based on a user's specific notification settings.

The Base Station is designed to be a wall mounted unit that is always active and able to respond to requests from both the remote client and the ZigBee network. It will also have a built in GUI that will allow the user some access for changing and adding devices to the ZigBee network from the Base Station. In processing requests from the ZigBee network side of the system, it will provide a way to determine upon receiving a signal from what device it came from and to alert the user of this by sending a notification to the remote device with this information. From the remote device it will be able to receive commands on the software side of the Base Station and then to transfer those commands into instructions for the specified ZigBee network device to decode and follow. The hardware will allow for multiple connections to be made for remote devices and for multiple sensors to be set up on the ZigBee Network. It will also have the means of communicating through the Internet and updating the database about notifications being received, transmitted, or handled according to what the notification was sent about.

Communications between the LPC3250 Android base station and the remote ZigBee devices will be channeled through a ZigBee base station which will act as the ZigBee controller for the entire wireless network. The ZigBee base station will be a separate board from the LPC3250 base board and will be the same PCB as the wireless modules, but with only the MCU and ZigBee related hardware populated. This board is only intended to provide ZigBee network

coordinator functions. The ZigBee coordinator board and the Android base board will have soldered connections for serial communications and power for the ZigBee board. The wireless base station will essentially act as a multiplexor/demultiplexor for communications between Android base and the remote wireless devices. When the Android base station needs to send data to the wireless modules, it will send the data in standard ZigBee protocol to the ZigBee base station through the serial connection, and the ZigBee base will then forward the data over the wireless network to the intended module. When the ZigBee base station receives data from a target wireless module that is intended for the Android base station, it will simply forward the raw received data to the Android base station via the serial connection.

4.2 - Wireless Sense and Control Modules

4.2.1 - General

The wireless ZigBee network centered at the base station serves as the communications infrastructure for the in home portion of the system. There are three primary types of devices that can be incorporated into the aLife ZigBee network:

- aLife native wireless sense and control units (here-on referred to as wireless sense and control modules) that operate with built in hardware only
- aLife wireless sense and control modules that are interfaced with peripheral hardware
- 3rd party, off the shelf ZigBee compliant smart home devices

The purpose for all of the modules are to serve as a sensor and/or control device for the main system. They are the entry point for all home status information, and the end points for control commands. For instance, these devices will monitor the actual current being consumed by an appliance, physically connect or disconnect an appliance, or have a wired connection to a sensor. The native aLife modules and 3rd party modules will operate with the same wireless protocol, so the interface with the base station is identical. Example 3rd party devices are ZigBee thermostats, window and door sensors, and electronic door locks. This allows a high degree of flexibility and expandability in the overall system, allowing the user to select any compliant ZigBee device and add it to the system for their specific application.

The wireless sense and control modules are designed to have simple, but flexible hardware that can allow the modules to be used in numerous applications while keeping the cost the modules low. The modules will all have the identical on board hardware, and when they are added to the aLife system the user will configure them for a specific application through the base station interface. The modules can do the following functions when operating with built in hardware only:

- Home appliance power consumption monitoring and electrical disconnect
- Smart lighting control that allows remote on/off/dim control
- Temperature sensing at the location of the module

The modules also have I/O ports that enables interfacing with other peripheral hardware in order to perform the following functions:

- Garage door position monitoring and control
- Remote home appliance power consumption sensing with an external current sense loop
- Remote temperature sensing with an external temperature sensor.
- In addition, there are extra digital and analog I/O types for future applications

In operation, the wireless modules would essentially act as slave devices to the master base station. If configured as a sensor, the module would continuously monitor the sensor data and report the data back to the base station when polled (ex for power monitoring), or if there's an interrupt even that requires immediate notification to the base station (ex window or door open sensor). If configured as a control (output) device, they would wait for instructions from the base station and maintain a constant output state until the base station commands them to do otherwise.

4.2.2 - ZigBee Wireless Network

Both the remote wireless sense and control modules and the ZigBee base station will use the Freescale BeeStack Consumer RF4CE ZigBee protocol stack in conjunction with the MC13213 built in ZigBee MAC, PHY, and transceiver (stack shown in Error: Reference source not found).

Illustration 16: Freescale MC13213 ZigBee communications stack

"The BeeStack Consumer layer implements an interface between the application and the IEEE.802.15.4 MAC layer. The BeeStack Consumer layer conceptually includes a management entity that provides the service interfaces through which layer management functions may be invoked. This management entity is also responsible for maintaining a database of managed objects pertaining to the BeeStack Consumer layer. The BeeStack Consumer layer provides a number of services, accessed through the BeeStack Consumer API. The BeeStack Consumer services have the following characteristics:

- Are started using BeeStack Consumer API function calls
- Communicate to application the execution status using confirm messages
- Inform the application layer about asynchronous network information arrival using indication messages ⁵⁴

The following image, 39, representation the functionality of the BeeStack for the consumer.

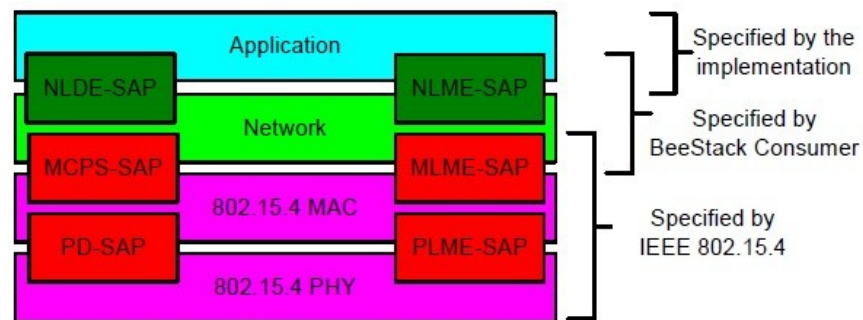


Illustration 17: BeeStack Consumer layer interfaces ⁵⁵

The BeeStack Consumer protocol defines two types of devices to implement a wireless Personal Area Network (PAN): controller nodes and target nodes. This is different than the ZigBee coordinator/router/end device setup described in the research portion of this paper. In this protocol, there can be multiple controller and target nodes. The functions of the two node type are:

Controller node - This device contains a subset of the BeeStack Consumer features. The controller node type is used in remote control applications. This node does not start a PAN. During the pairing process, it receives a Pan Id and Short Address from the target node it has paired with. These values are used in future communications with the paired target node. ⁵⁶

Target node - This device contains a subset of the BeeStack

Consumer features. The target node type is used in Consumer Electronics device applications (e.g. TV's, DVD's, etc.). It is intended to be used in backbone powered devices. Starting a target node always starts a new PAN.⁵⁷

A target node has full PAN coordinator capabilities and can start a network in its own right. Both types of node can join networks started by target nodes by pairing with that target. Multiple RC PANs form an RC network and nodes in the network can communicate between RC PANs. To communicate with a target node, a controller node first switches to the channel and assumes the PAN identifier of the destination RC PAN. It then uses the network address, allocated through the pairing procedure, to identify itself on the RC PAN and thus communicate with the desired target node."⁵⁸

An example network is shown in 55 between a TV, DVD player, and CD player and their respective remote controls. The aLife ZigBee network will work in a similar fashion where the remote sensor and control devices will act as target nodes and the ZigBee base station will act as the lone controller node.

Illustration 18: Example BeeStack Consumer Mesh Network

4.3 - Remote Client

The remote client will be the main interface to most functions of the aLife system. The remote client software will be passive, running as a service in the background on the client device. The service will maintain a connection with the aLife base station and update its own IP address in the base station database, so the database can push notifications to the client. The client will display notifications and give the user the option to respond or ignore the event that caused the notification.

The remote client software will also provide the ability for users to view the status of their devices. Client software will also be the interface for adding devices to the aLife system. In the menu the user will be given the option to put the aLife system into discovery mode, much like Bluetooth, and add a ZigBee device to the system. The client software will detect the type of device being added to the system and will present the user with options to configure the device.

5 - Feature and Performance Specifications

5.1 - Base Station

5.1.1 - Hardware

5.1.1.1 - Base Board

For this project to work, we needed to design a hardware platform that would allow us to implement all the features of aLife.

The following describes the list that we needed to incorporate to ensure that the functionality of the Base Station fulfilled all the roles we needed it to do.

Functionality:

- Needs to be powered off a wall outlet or a USB connection
- Have enough memory to allow for the Android operating system to be run smoothly
- Must have a LCD screen controller built in and allow for touch screen input to the system
- Must allow for some type of internet connection built in
- Have some type of restart
- Have some type of speaker system built in
- Connections
 - Allow for USB connections
 - Allow for a RS232 connection to be made
 - Allow for a USB-to-serial connection

5.1.1.2 - ZigBee Base Station

The ZigBee base station will be same PCB as the remote sensor and control modules and will be hardwired to the Embedded Artists LC3250 board, see the feature and performance specifications for those modules below.

5.1.2 - Software

The specifications for the base station software were decided upon after extensive requirement analysis. Taking into consideration time constraints and scalability, our team deemed the following list of specifications necessary.

Functional:

- Base Station must be able to receive status information from ZigBee devices
- All users must be categorized as either generic user or administrative user
- All users will have the ability to unregister their login credentials
- Only administrative user can unregister other users login credentials

- Base station must be able to add or remove a device from the ZigBee network
- Base station must be able to accept up to 5 TCP connections from remote client devices
- Request for service will be dealt with on a first come first serve basis
- Multiple request to service the same notification must only be serviced once
- Users must be notified of any unsuccessful service requests
- Base station must poll all devices and update any relevant status information at least every 5 minutes.

Security:

- A login and password must be created for all users
- Base Station must allow a user to create a new login
- A user can have up to 5 unsuccessful attempts to login prior to being locked out for up to 5 minutes.
- Only registered remote client devices may have service request fulfilled by the base station

Notifications:

- All notifications must have a unique notification ID
- All notifications set up by a user must be delivered by the base station
- Users shall only receive notifications that they are registered to receive

Database:

- Base station must be able to make a connection to SQLite database
- Base station must be able to add/remove/update a row in any of the database's tables

5.2 - Wireless Sense and Control Modules

5.2.1 - Hardware

5.2.1.1 - MCU

- BDM interface header for debugging
- UART header and through hole solder points for debugging and testing
- Bus clock operation at least 16MHz

5.2.1.2 - ZigBee Transceiver

- Transmission range of at least 30m
- Able to transmit $\geq 200\text{kb/s}$ at 30m

- Antenna area \leq 2 square inches of PCB space

5.2.1.3 - Control and Sensor Related

Each ADC input will be paired with a digital input and will share a terminal. Likewise, each analog output will be paired with a digital output and will share a terminal. All inputs and outputs will have transient voltage suppressors for surge protection.

- Various MCU I/O and power supply ports:
 - (2) analog to digital inputs, 3.3VDC max
 - (2) analog line level output (via integrated PWM), 3.3VDC, 50mA max
 - (2) digital line level outputs, 3.3VDC, 50mA max
 - (2) digital line level inputs, 3.3VDC max
 - (1) software controlled relay switched send/receive loop
- (1) 3.3VDC rail, 300mA max (PTC limited)
- (1) ground
- 120VAC pass through
- One standard 3 prong 120VAC male plug on the back of the unit for module power and pass through to the front of the module
- One standard 3 prong 120VAC female socket on the front of the unit that is powered by the male prong in the back for appliance power control and sensing
- 120VAC pass through current sensing
- Series current sense resistor through one AC line
- Current sensing is accurate to $\pm 5\%$ of actual current consumption
- Maximum load of 1000W supported
- 120VAC pass through lighting control or appliance power cutoff/limiting
- On/Off/Dim control, with software controlled dimming in $\leq 3\%$ increments
- Temperature sensor
- ≥ 11 bits of measurement accuracy
- Measured temperature is accurate to $\leq \pm 2$ degrees Celsius of actual temperature
- Temperature operating range of -40 degrees C to +125 degrees Celsius

5.2.1.4 - Power Supply

Power for the modules will be derived from a standard 120VAC wall outlet which will feed a full wave bridge rectifier and then a 3.3VDC high efficiency switching power supply. The power supply performance requirements are:

- $3.4\text{VDC} > V_{out} > 3.2\text{VDC}$
- Ripple voltage: $< 50\text{mV}$
- Minimum current supply capability: 500mA

- Power supply efficiency >85%
- System wide power requirements:
- <50mA total IC current draw (not including relays and external components)
- <100mA total system current draw (not including external components)

5.2.1.5 - Package

The modules are designed to be wall outlet mounted and all hardware contained within a relatively small package that does not obstruct the second wall outlet. The specific dimensions of the chassis is not known at this time because it depends on the size of the PCB, however the general requirements are as follows:

- 1 male 3 prong 120VAC plug on the back of the chassis
- 1 female 3 prong 120VAC plug on the front of the chassis
- An 8 position screw terminal block for sensor and control I/O on one side of the chassis
- Chassis dimensions not to exceed:
- length of PCB plus 1 inch (parallel plane to wall)
- width of PCB plus 1 inch (parallel plane to wall)
- 2 inches deep (perpendicular plane to wall)

5.2.2 - Software

5.2.2.1 - Main Program

- The firmware will be programmed in C using Freescales' free CodeWarrior IDE
- Total program size (including BeeStack Consumer ZigBee protocol stack) <= 60kB
- Total RAM usage (including BeeStack Consumer ZigBee protocol stack) <= 4kB

5.2.2.2 - ZigBee Protocol Stack

- Run Freescale BeeStack Consumer (RF4CE) protocol stack
- Follow standard ZigBee device profiles (ex Home Automation, Smart Energy) for command protocol

5.3 - Remote Client

5.3.1 - Hardware

- Hardware capable of running Android 2.1 or later
- Must be capable of maintaining a constant internet connection at all times

5.3.2 - Software

5.3.2.1 - aLife Service

- Automatically starts when the client device is turned on
- Maintains local IP address information in base station database
- Maintains open an socket for base to client notifications

5.3.2.2 - aLife GUI

- Option to disable all communications with the server
- Option to mute all notifications
- Presents notifications in the Android notification window
- Allows user to ignore or respond to events
- Gives the user a list of devices on the system
- Allows the user to see a device history and status
- Allows user to control devices

6 - Design

For the hardware requirements of this project, we needed to make sure that we met the basic system requirements for running android on the base station before we started with any other considerations. ⁶⁰ [61](#)

6.1 - Base Station

6.1.1 - Hardware

The hardware design in this paper begins from the microcontroller of the Base station and will continue to add components until it has reached the complete functionality of the Base station module. Table 3 gives hardware specifics for the microcontroller.

6.1.1.1 - NXP LPC3250 Microcontroller⁶²

Processor	ARM926EJ-S core with speed up to 266 MHz
Flash	None
RAM	256 KB
Instruction cache	32 KB
Data cache	32 KB
Timers	<ul style="list-style-type: none"> • One 32-bit general purpose high-speed timer • Six enhanced timers/counters • One 32-bit millisecond timer driven from RTC clock
PWM	2 single output PWMs and 1 motor control PWM
ADC	10-bit, 400 kHz multiplexing from 3 pins
Serial interfaces	7xUART, 2xI2C, 2xSPI, 2xSSP, 2xI2S
USB	<ul style="list-style-type: none"> • Fast-Speed USB 2.0 interface • Device, host (OHCI compliant), or On-The-Go (OTG)
Ethernet	10/100 Ethernet MAC with dedicated DMA controller
SD/MMC	Secure Digital (SD) memory card interface
Other	<ul style="list-style-type: none"> • Vector Floating Point Unit • Two NAND Flash controllers. Single-level and multi-level • External memory controller for DDR and SDR SDRAM and static devices • 24-bit Color LCD controller • Touch screen controller • Keypad interface • 0.9 V low power mode

Table 3: Hardware specifications for the microcontroller

With the specs above defined, we can then move on to a logical description about how the device will operate. Illustration 19 gives a block diagram layout of the microcontroller and explains the functionality of the hardware piece.

Illustration 19: Functional description of the NXP LPC3250

With the microcontroller defined, we can then move on to integrating that with the rest of the board. The next hardware section includes the microcontroller placed into the LPC3250 OEM Board. Table 4 outlines the specifics about this device.

6.1.1.2 - LPC3250 OEM Board

Processor	NXP's ARM926EJ-S LPC3250 microcontroller in BGA package
External Flash	128 MB NAND FLASH (1Gbit) 4 MB SPI-NOR FLASH (32Mbit)
Data Memory	64 MB DDR SDRAM + 256 KB internal 16-bit data bus to DDR SDRAM
Ethernet	100/10M Ethernet interface based on National DP83848 Ethernet PHY
Clock Crystals	13.000 MHz crystal for CPU
Dimensions	66 x 48 mm
Power	3.15V-3.3V powering
Connectors	200 pos expansion connector (as defined in SODIMM standard), 0.6mm pitch
Other	<ul style="list-style-type: none"> • 256 Kbit I2C E2PROM for storing non-volatile parameters • 5 LEDs • Buffered 16-bit databus

Table 4: Functional description of the LPC3250 OEM Board

Since the LPC3250 OEM Board mainly adds a way to interface the microcontroller to the Base Station and maintains the same general functionality as the microcontroller itself, there is no need to make a block diagram for it. Illustration 20 is what the LPC3250 OEM Board will look like with the microcontroller attached.

Illustration 20: Picture of the LPC 3250 OEM Board

[65](#) [66](#)

With the LPC3250 OEM Board in place, we can now integrate into the final part of the hardware component. The QVGA Base Board will act as the main interfacing board for the microprocessor and the interface between the user and the ZigBee network. Table 5 gives a description of the Base Boards hardware.

6.1.1.3 - QVGA Base Board

Display	<ul style="list-style-type: none"> • 3.2 inch QVGA TFT color LCD with touch screen panel
Connectors	<ul style="list-style-type: none"> • 200 pos SODIMM connector for OEM Board • Expansion connector with all LCD controller signals, for custom displays • Expansion connector with all cpu signals • Ethernet connector (RJ45) • MMC/SD interface & connector • JTAG connector • Pads for ETM connector
Interfaces	<ul style="list-style-type: none"> • USB OTG interface & connector • USB host interface & connector • RS232 interface • IrDA tranceiver interface
Power	<ul style="list-style-type: none"> • Power supply, either via USB or external 9-15V DC • 0.3F capacitor backup for RTC and LED on ALARM output
Expansion	<ul style="list-style-type: none"> • Expansion connector with all LCD controller signals, for custom displays • Expansion connector with all cpu signals
Other	<ul style="list-style-type: none"> • 5-key joystick • 3 axis accelerometer • Push-button key and LED on P2.10 • 4 push-button keys via I2C • 8 LEDs (via I2C) • 1 Analog inputs • USB-to-serial bridge on UART #0, and ISP functionality • Reset push-button and LED • Speaker output (DAC) • 240x150 mm in size

Table 5: Hardware specifications for the Base Board

With the hardware fully assembled, we have now finished the hardware design portion of this project. Illustration 21 is of the QVGA Base Board with both the LPC3250 OEM Board in place.

Illustration 21: Embedded Artists LPC3250 Base Board

[67](#)

The hardware described above will function in a particular manner to achieve the overall goal of this project, which is to prove a smart home type wall unit that will interface with the rest of the other components of the project. The following picture illustrates the general idea about what the purpose of the base station is and what type of functionality it will generate.

The Base Station should meet all of those particular criteria listed below in Illustration 22. Some of the naming conventions we used for labels in the picture are:

Illustration 22: Functional Specifications of the Base Station

The figure above describes the functionality we plan to implement with the Base Station device in the aLife project.

6.1.2 - Software

The following section discuss the design elements for the base station software.

6.1.2.1 - Board Support Package

The Embedded Artists NXP LPC3250 demo board will be the host client for the base station software. This would require the demo board to have a working version of Android 1.5 platform. In order to meet these needs the demo board must be equipped with the correct support package. The items needed in the support package are listed below.

The support package will include:

- Android: version 1.5 (Cupcake), ported by Embedded Artists
- Linux kernel: version 2.6.27.8, ported to the LPC3250 by NXP
- Bootloaders:
 - Kickstart
 - S1L: October 4, 2009 or later build
 - u-boot: version 2009.03
- Root file system: JFFS2

These items will be installed on the demo board prior to installing any versions of the base station software.

6.1.2.2 - Initial Design Prototype

Our team started on the initial design of our base station software very early in our development process. Below is Illustration 23, our initial prototyped design.

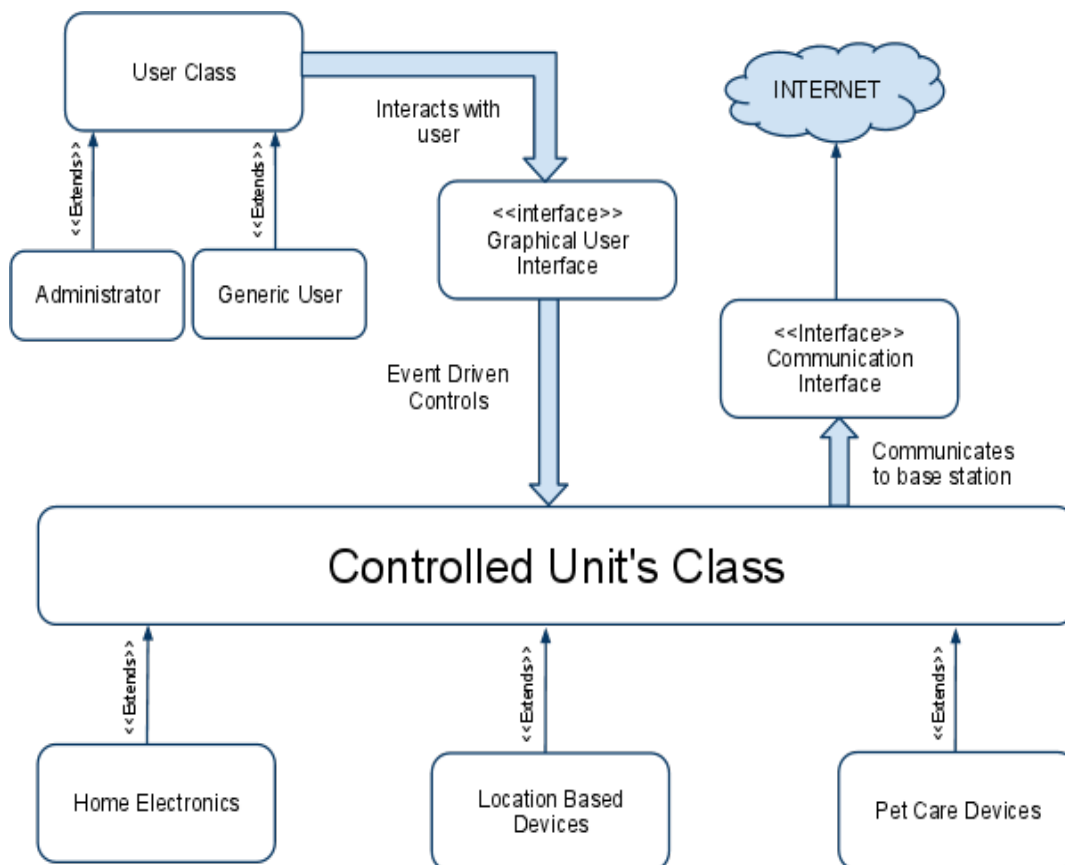


Illustration 23: Initial Base Station Software Design

The idea behind the model was to make the base station software as similar as possible to the remote client software. The base station software was to have a graphical user interface that could add or remove a device, set up notifications, review notification history, and monitor one of our controlled devices. After discovering the limited hardware resources we would have with the embedded artist demo board and the choice to use a client-server communication model, we found that the base station would only need basic functionality as its responsibility lied in continuously polling the ZigBee devices and fulfilling remote client device requests for service. This led our team to trim down on the features provided by the base station and provide a more accurate design model.

6.1.2.3 - Final Design

At this point in the development process our team had completed the Base Station Software's functional and design requirements and, following our software model, began to work on the base stations software's system design.

We did not want to overload the base station with unnecessary functionality, so we started the final design process by creating an event table and use case diagram outlining the outcomes to all external and internal stimulus to our system. They are displayed in Table 6 and Illustration 24 respectively.

Event Name	External Stimuli	External Response	Internal Data and State
Log In / Log Out	The user logs into the system using their log in credentials	User is directed to home screen on successful log in; else directed back to log in screen	The system queries our database for log in credentials and returns access information.
Add a Device	The user touches the "Add Device" button on LCD Display	The system prompts the user for new device information	System wait for user submission. Device added to database and ZigBee network
Poll ZigBee Power Sensors	No external stimulus; Periodic task	No external response; Information handled within system	System sends request to ZigBee Base station through serial communication; Results are interpreted and submitted to database
Request for Socket Connection	Request for connection from a remote client device	Successful connection acknowledgement sent to device; else connection timeout	A duplex TCP connection is established between the Base Station and Remote Client Device
Remote Client Service Request	Request for service from a remote client device	User is prompted with a request fulfilled acknowledgment; else error message prompted	Service command is sent to ZigBee Base station; Base station receives acknowledgement and submits to user; Notification table updated in database

Table 6: Base Station Event Table

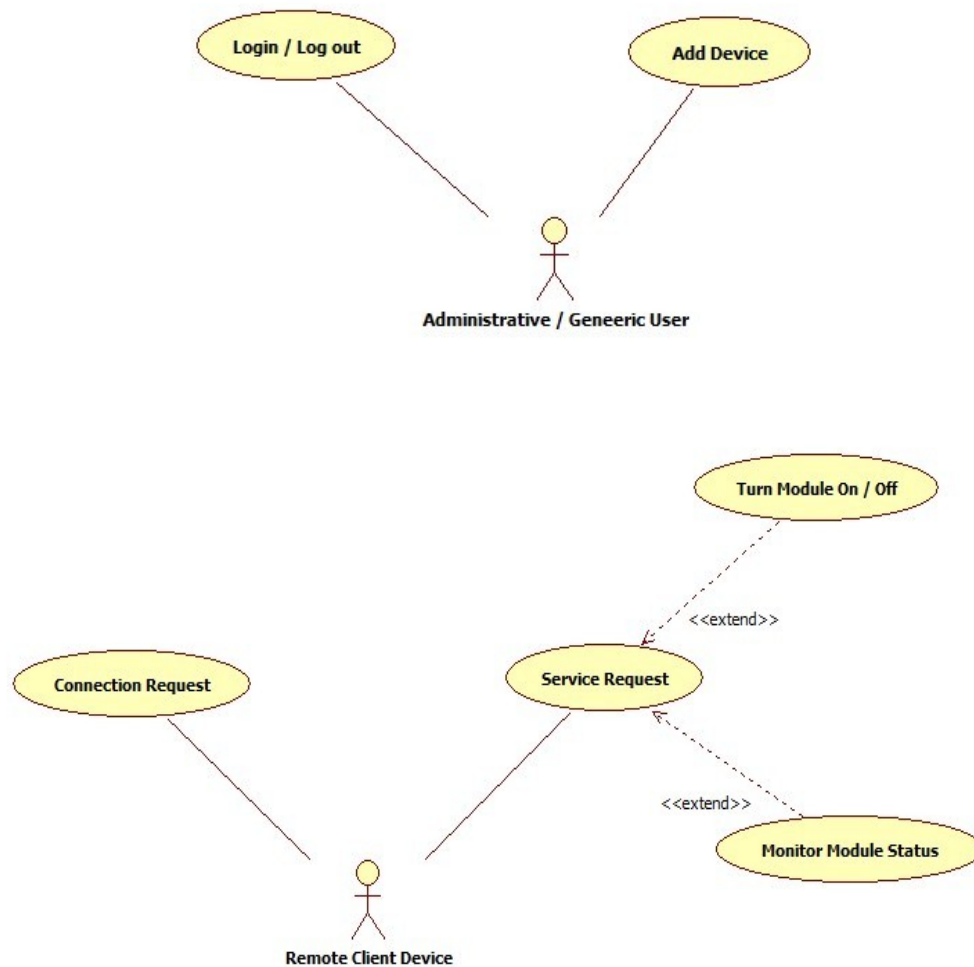


Illustration 24: Base Station Software Use Case Diagram

The use case diagram and event table reinforce our theme of simple, functional features. The base station software will allow a user the ability to log in and out of the system and add/remove a device from the ZigBee network. All other functions of the base station software will be not be visible to the user. These additional features include the ability to poll any connected ZigBee device for status information, the ability to accept a TCP connection, and the ability to fulfill a remote client request for service.

After understanding the design of our software's functional features, the next step in our design process was to understand how our software system should respond to interactions between users, either directly or via remote client devices. Below is four sequence diagrams demonstrating critical interactions our system must be able to handle. The first is a new user registration, the second is adding a new device, the third is a remote client service request, and the fourth is

multiple users responding to a notification request. They are listed as Illustration 25, Illustration 26, Illustration 27 and Illustration 28 respectively.

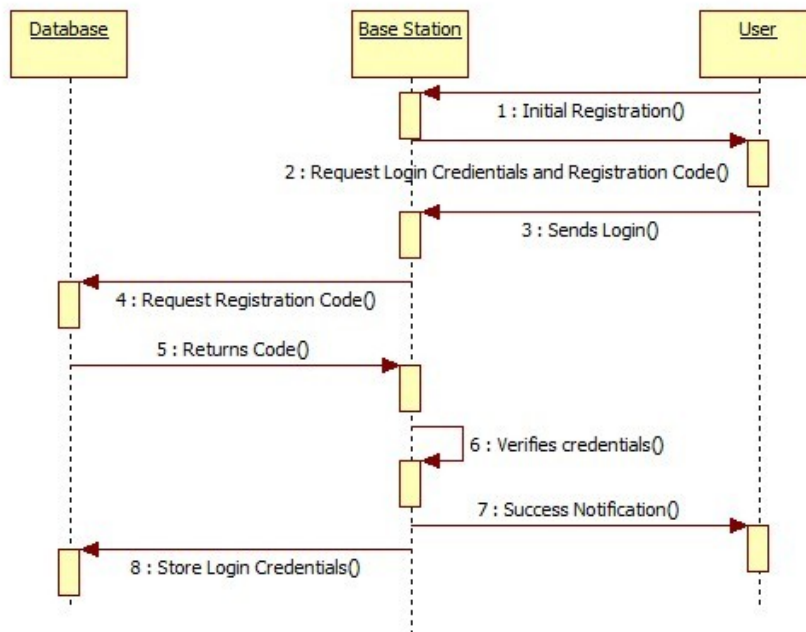


Illustration 25: New User Registration

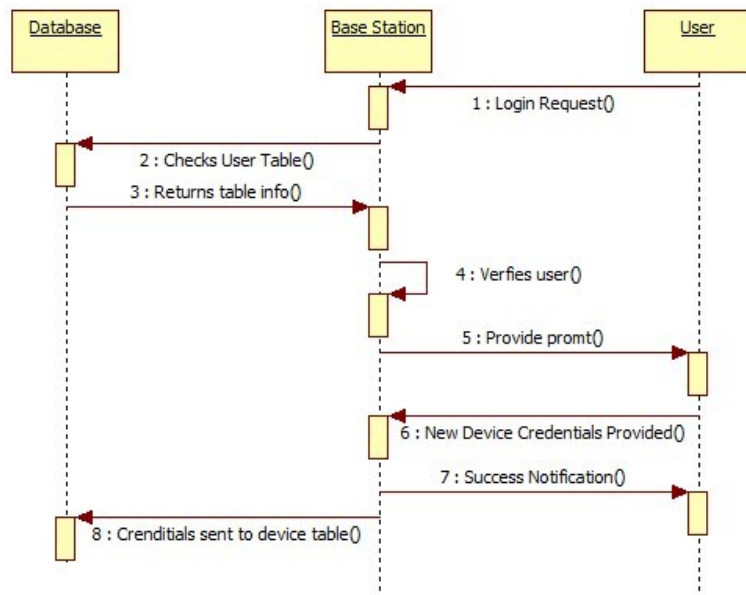


Illustration 26: Add a New Device

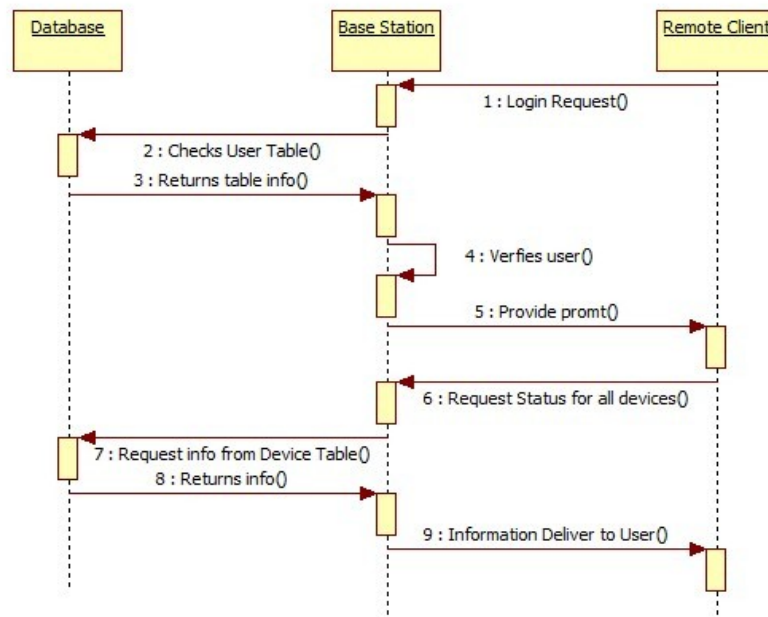


Illustration 27: Fulfill Service Request

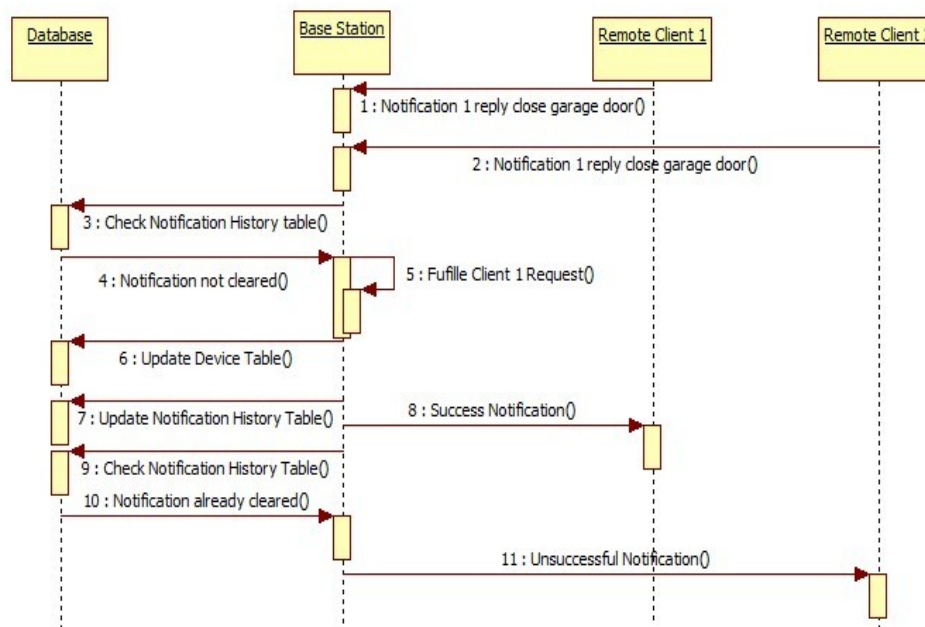


Illustration 28: Multiple Reply to Notification

The sequence diagrams gave our group insight on how the base station software interacts with remote users and the database. We found that most scenarios called for socket connections to be established and the base station to be queried. This demonstrated a need for a handler for both the database and socket connections to allow easy communication between those devices. We also found the need for a generic devices class which extends more specific devices. These will be used to create device object and hold status information concerning any active device. We would also need a main class that will create our GUI and fulfill any autonomous services that are required of the base station.

6.1.2.4 - Base Station Class

The base station class will be the responsible for creating the GUI, interacting with Users, and fulfilling any autonomous services request required of the base station. The handlers for the database and socket connections available directly through the Android API which will not require our team to create any auxiliary classes. The class fields and methods are explained below.

Fields:

- PowerNotif (Boolean): Field tells whether a powered device notification is set up
- SecurityNotif (Boolean): Field tells whether a security device notification is set up

- ControlNotif (Boolean): Field tells whether a control device notification is set up
- ActiveDevices(Device []): Array of active devices in the network
- RequestedService (String): Integer representation of a request

Methods:

- BaseStation (): Constructor to create the GUI interface for our system.
- getActiveDevices(): Takes in no parameters. Returns all active devices in database.
- addDevice(Device X): Adds specified device to network and database. No return.
- removeDevice(Device X): Removed specified device from network and database. No return.
- setNotifications(): Takes no parameters. Sets all initial notifications booleans. These will dictate which notifications will be checked for continuously throughout the program.
- pollDevices(ActiveDevices): Takes in all active devices and updates their information in the database tables. Returns nothing.
- socketParser (ByteStream): Takes in a socket byte stream and returns the requested service in string format.
- requestService (Device X, RequestService): This method will take in a device and service request and will fulfill the said service. Returns nothing.

6.1.2.5 - Device Class

The devices class is the abstract parent class of all specific controlled devices. It will be extended by a PowerDevice class, SecurityDevice class, and ControlDevice class.

Fields:

- ID (Int) : Device's ID
- Name (String): Name provided by user for device
- Status (Boolean): Provides status of device, on or off

Methods:

- getID(): Takes in no parameters. Returns an Int, the value of the devices ID.
- setID(Int ID): Takes in an Int and updates ID field. Does not return anything.
- getName(): Takes in no parameters. Returns a String, the value of the devices Name.

- setName(String name): Take in a String and update Name field. Does not return anything.
- getStatus(): Takes in no parameters. Returns Status of device
- setStatus(Boolean Status): Takes in a boolean and updates Status of device. Does not return anything.

6.1.2.6 - PowerDevice Class

The PowerDevice Class will be a factory class for power monitoring device objects.

Fields:

- TotalWatts (Int): Provides total watts used by device
- SetWatts (Int): Provides user defined limit for watt usage
- CurrentTemp (Int): Provides current temperature information
- SetTemp (Int): Provides user defined limit for temperature
- PollTime (Int): Time the information was polled
- SetTime (Int): Set time for notification purposes
- Timeframe (Int): Timeframe for power information i.e. 7 days or 30 days

Methods:

- PowerDevice(): Constructor to create object. Will have generic and specific constructors.
- turnOn(): Takes in no parameters. Sends message to ZigBee device to turn on
- turnOff(): Takes in no parameters. Sends message to ZigBee device to turn on
- getTemp(): Takes no parameters. Polls ZigBee device for temperature.
- setTemp(Int Temp): Sets ZigBee device with provided temperature. Does not return anything.
- getWatt(): Takes no parameters. Polls ZigBee device for wattage.
- getTimeframe(): Takes in no parameters. Returns the timeframe field.
- setTimeframe(Int Timeframe): Sets the timeframe field. Does not return anything.

6.1.2.7 - SecurtyDevice Class

The Security Device Class will be a factory class for security monitoring device objects.

Fields:

- SetTime (Int): Provides set time for any notifications

Methods:

- SecurityDevice(): Constructor to create object. Will have generic and specific constructors.
- turnOn(): Takes in no parameters. Sends message to ZigBee device to turn on
- turnOff(): Takes in no parameters. Sends message to ZigBee device to turn on
- isOpen(): Takes in no parameters. Returns true if ZigBee device is open.
- IsLocked(): Takes in no parameters. Returns true if ZigBee device is locked.
-

6.1.2.8 - ControlDevice Class

The ControlDevice Class will be a factory class for controlled device objects.

Methods:

- ControlDevice(): Constructor to create object. Will have generic and specific constructors.
- turnOn(): Takes in no parameters. Sends message to ZigBee device to turn on
- turnOff(): Takes in no parameters. Sends message to ZigBee device to turn on

6.1.3 - Database

The SQLite database will be utilized to store information about current users, devices, power history, notification history, and notification set ups. Each of these elements will be have their own table in our database with their own specified fields. Listed below is our database structure.

Users:

- ID (Int)
- Password (String)
- Name (String)
- Permission (Boolean)
- IP Address (String)
- Fail Attempts (Int)
- Failed Time (Int)
- Success Time (Int)

Devices:

- ID (Int)

- Device Type
- Name (String)
- Address (String)
- Field1 (Various Types)
- Field2 (Various Types)
- Field3 (Various Types)
- Field4 (Various Types)
- Field5 (Various Types)
- Field6 (Various Types)
- Field7 (Various Types)
- Field8 (Various Types)
- Field9 (Various Types)
- Field10 (Various Types)
- History1 (Various Types)
- History2 (Various Types)
- History3 (Various Types)
- History4 (Various Types)

Power History:

- Device ID (Int)
- Power Usage (Int)
- Time of Day (Int)

Notification History:

- Notification Type (Int)
- Notification Level (Int)
- Notification ID (Int)
- User ID (Int)
- Device ID (Int)

Notification Set Up:

- ID (Int)
- Description (String)
- Users (String)
- Set Time (Int)
- Set Temp (Int)
- Set Energy (Int)

6.2 - Wireless Sense and Control Modules

6.2.1 - Hardware

6.2.1.1 - MCU

- Low voltage MCU with 40 MHz low power HCS08 CPU core
- 60K flash memory with block protection and security and 4K RAM
- Low power modes (Wait plus Stop2 and Stop3 modes)
- Dedicated serial peripheral interface (SPI) connected internally to 802.15.4 modem
- One external 4-channel (5-channel internal) 16-bit timer/pulse width modulator (TPM) module and one external 1-channel (3-channel internal) 16-bit timer/pulse width modulator module, each with selectable input capture, output capture, and PWM capability.
- 8-bit port keyboard interrupt (KBI)
- 8-channel 8-10-bit ADC
- Two independent serial communication interfaces (SCI)
- Multiple clock source options
- Internal clock generator (ICG) with 243 kHz oscillator that has +/-0.2% trimming resolution and +/-0.5% deviation across voltage.
- Startup oscillator of approximately 8 MHz
- External crystal or resonator
- External source from modem clock for very high accuracy source or system low-cost option
- Inter-integrated circuit (IIC) interface.
- In-circuit debug and flash programming available via on-chip background debug module (BDM)
- Two comparator and 9 trigger modes
- Eight deep FIFO for storing change-of-flow addresses and event-only data
- MC13211/212/213 Technical Data, Rev. 1.8
- Tag and force breakpoints
- In-circuit debugging with single breakpoint
- System protection features
- Programmable low voltage interrupt (LVI)
- Optional watchdog timer (COP)
- Illegal opcode detection
- Up to 32 MCU GPIO with programmable pull-ups ⁶⁸

Illustration 29 gives a Block Diagram of the functionality of the MC13213 MCU device.

Illustration 29: Figure X - MC13213 block diagram

With the functionality defined, we can now go into the physical aspects of the MC13213 MCU device. Illustration 30 gives the pin layouts for the MC13213 MCU.

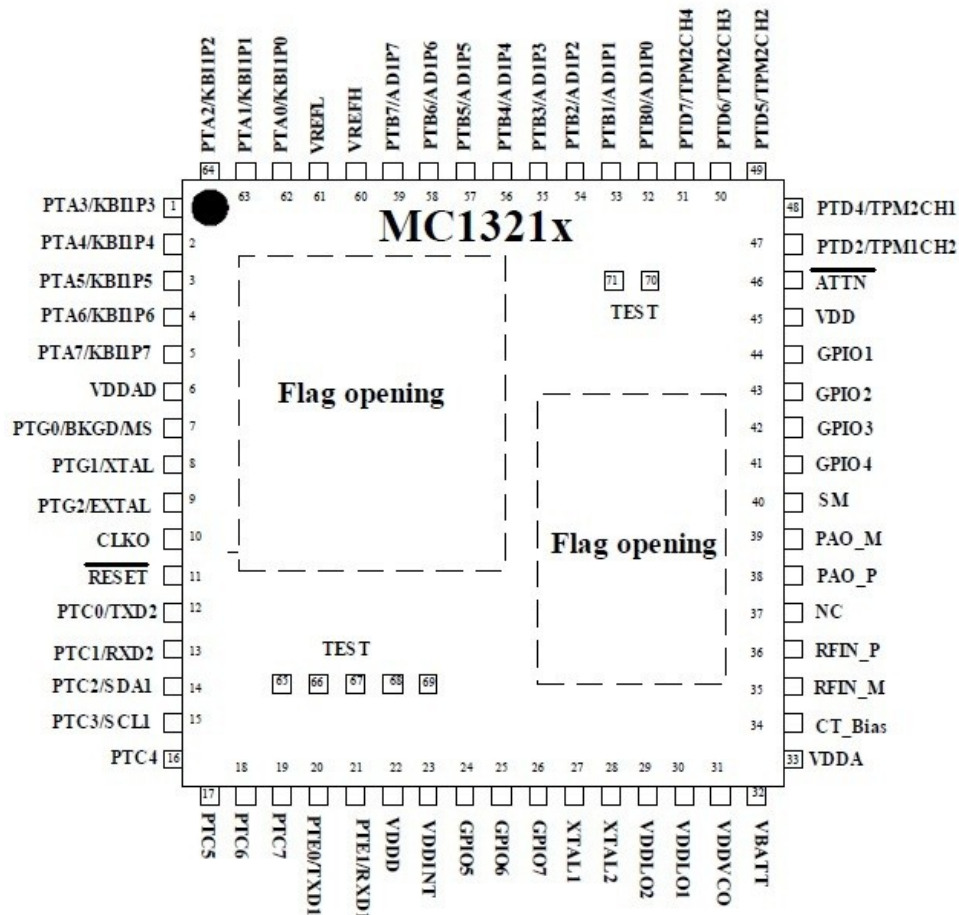


Illustration 30: MC13213 Pinout

[69](#)

With the physical and functional descriptions of the MCU covered, we can then go on to describe the other component in the wireless sense and control module, the ZigBee transceiver.

ZigBee Transceiver

- Fully compliant 802.15.4 Standard transceiver supports 250 kbps O-QPSK data in 5.0 MHz channels and full spread-spectrum encode and decode
- Operates on one of 16 selectable channels in the 2.4 GHz ISM band
- -1 dBm to 0 dBm nominal output power, programmable from -27 dBm to +3 dBm typical
- Receive sensitivity of <-92 dBm (typical) at 1% PER, 20-byte packet, much better than the 802.15.4 Standard of -85 dBm
- Integrated transmit/receive switch
- Dual PA output pairs which can be programmed for full differential single-port or dual-port operation that supports an external LNA and/or PA.
- Three low power modes for increased battery life

- Programmable frequency clock output for use by MCU
- Onboard trim capability for 16 MHz crystal reference oscillator eliminates need for external variable capacitors and allows for automated production frequency calibration
- Four internal timer comparators available to supplement MCU timer resources
- Supports both packet data mode and streaming data mode
- Seven GPIO to supplement MCU GPIO

Illustration 31 gives the Block diagram functionality for the MC13213 ZigBee transceiver.

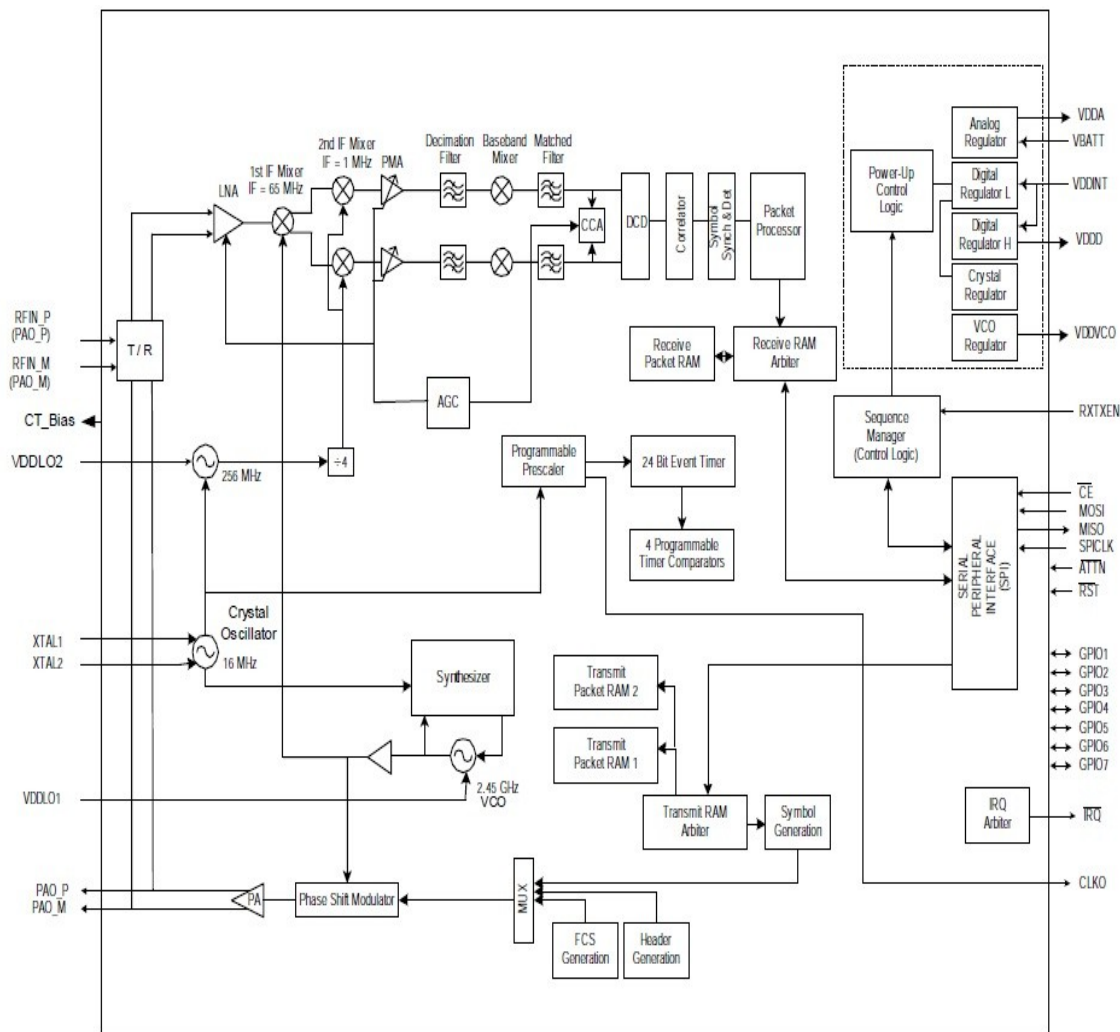


Illustration 31: MC13213 ZigBee transceiver block diagram⁷⁰

With that functionality defined, we can now define the functionality of the device. Illustration 32 describes the functionality of the aLife wireless module.

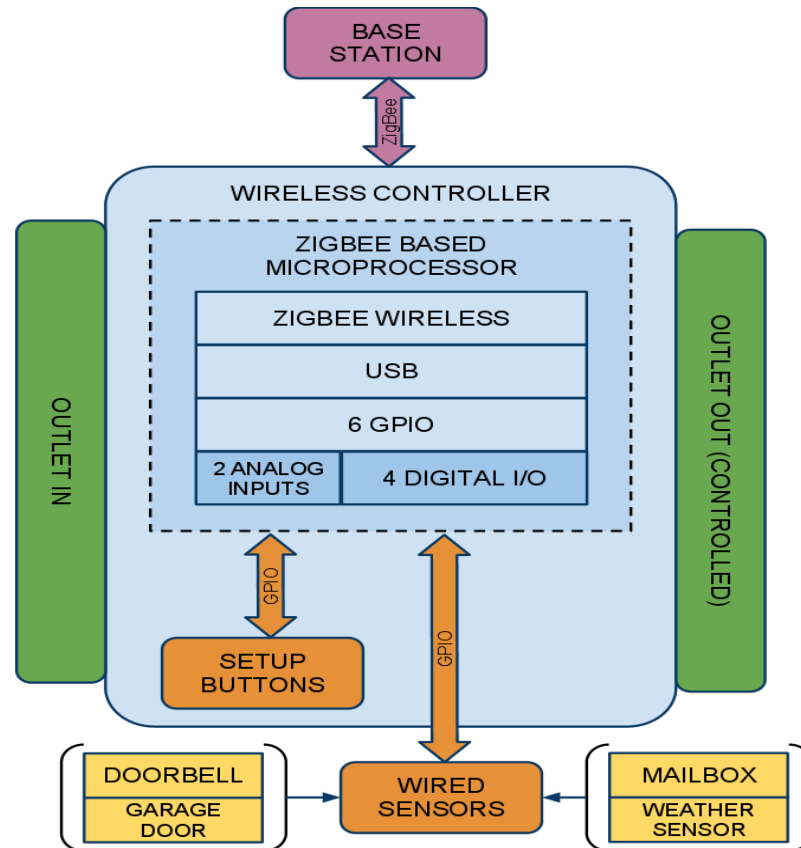


Illustration 32: Functional Diagram for a Wireless Module

The figure above describes all of the hardware and functionality components that will be implemented in the ZigBee wireless module.

6.2.1.2 - Control and Sensor Related

Lighting control circuit

- Basic diac/triac phase based dimmer circuit.
- MCU I2C interface with 100k Ohm 100 step digital potentiometer for dim control
- Disconnection is done via MCU controlled 10A , 240VAC relay

120VAC Current sensor IC

- 10A max handling
- <1.3mOhm series resistance, <2nH inductance
- +/- 5% accuracy
- 0-2VDC result output, interfaced with MCU A/D input

Temperature sensor

- Readings via built in sensor or external sensor interface
- +/- 2 °C accuracy
- Temperature range of -40 °C to +125 °C
- 11-bit, 0.125 °C resolution
- I2C interface with MCU

Digital and Analog I/O

- 2 MCU digital PWM outputs, integrated through op amp LPF. Duty cycle of PWM determines voltage level (0% or 100% generate digital output), period of PWM and LPF crossover frequency determines maximum digital data rate
- 2 External inputs: connect to digital input and ADC input in parallel so same input port can accept analog or digital signals
- 1 relay switched input/output loop, controlled by MCU. Relay is 10A max, 240VAC max.

6.2.1.3 - Power Supply

The wireless sense and control modules draw their power from a 120VAC outlet. The power subsystems are:

- 120VAC to 9VAC transformer
- 9VAC full wave bridge rectified to create 7VDC rail (used by relays and 3.3VDC switcher)
- 7VDC input into 300mA 550kHz switching power supply, output set to 3.3VDC (powers all components except relays)

6.2.1.4 - Package

The dimensions for the the modules packaging have not been determined since the dimensions of the PCB is not yet known. In general, it will be a plastic rectangular project, similar to those commonly found at RadioShack.

6.2.1.5 - Schematics

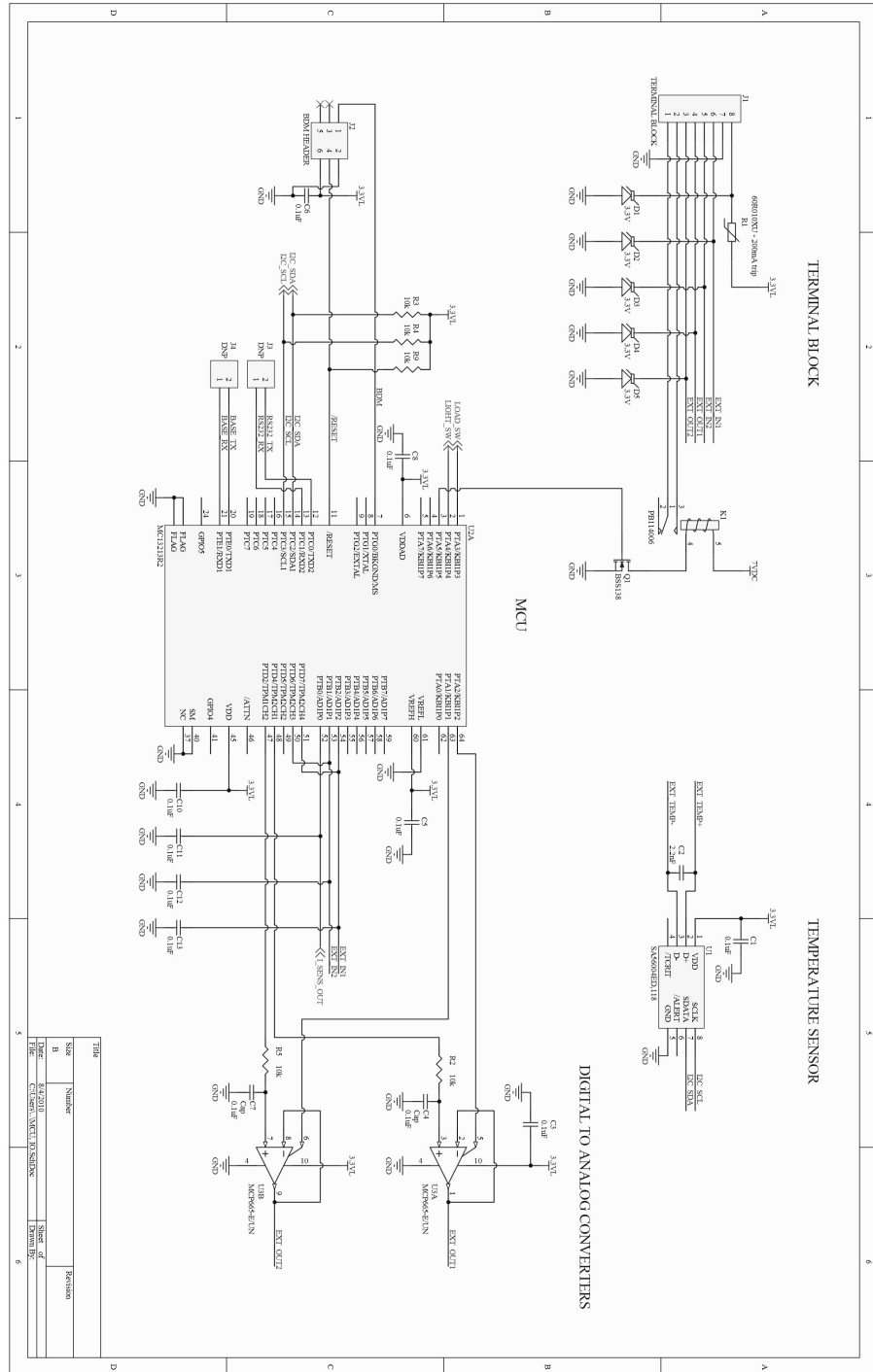


Illustration 33: MCU, Temperature Sensor and Terminal Block Schematics

Illustration 34: Light Dimmer Schematic

Illustration 35: ZigBee Transceiver Schematic

6.2.1.6 - Bill Of Materials

6.2.1.7 - ZigBee Protocol Stack

Freescale provides a free ZigBee RF4CE protocol stack specifically designed for the MC1323 MCU called the BeeStack Consumer. The BeeStack has the following features:⁷¹

- Based on the IEEE 802.15.4 Standard
- Supports ZigBee 2006 Specification
- Supports star, mesh and tree networks
- Advanced Encryption Standard (AES) 128-bit security
- Supports the ZigBee Home Automation Profile
- Supports the ZigBee Smart Energy Profile
- Supports application profiles that define standardized command sets for multi-vendor interoperability
- Supports vendor specific extensions to standard application profiles for vendor specific customizing
- Supports AES-128 bit encryption
- Provides a mechanism for secured key generation
- Specifies various power saving modes
- Provides a simple mechanism to pair devices (such as a remote to a TV)
Ensures only authorized devices are able to communicate (a user's remote will not turn their neighbor's TV on or off)

6.3 - Remote Client Device

6.3.1 - Hardware

6.3.1.1 - HTC G1 Dream

Table 7 gives the specifications for the HTC G1 Dream phone.

Processor	Qualcomm® MSM7201A™, 528 MHz
Platform	Android™
Memory	ROM: 256 MB RAM: 92 MB
Display	3.2-inch TFT-LCD flat touch-sensitive screen with 320 x 480 (HVGA) resolution
Network	HSPA/WCDMA: 2100 MHz Up to 7.2 Mbps down-link (HSDPA) and 2 Mbps up-link (HSUPA) speeds Quad-band GSM/GPRS/EDGE: 850/900/1800/1900 MHz (Band frequency, HSUPA availability, and data speed are operator dependent.)
Device Control	Trackball with Enter button
Keyboard	Slide-out 5-row QWERTY keyboard
GPS	GPS navigation capability with Google Maps™
Connectivity	Bluetooth® 2.0 with Enhanced Data Rate Wi-Fi®: IEEE 802.11b/g HTC ExtUSB™ (11-pin mini-USB 2.0 and audio jack in one)
Camera	3.2 megapixel color camera with auto focus
Audio	Built-in microphone and speaker Ring tone formats: AAC, AAC+, AMR-NB, MIDI, MP3, WMA, WMV 40 polyphonic and standard MIDI format 0 and 1 (SMF)/SP MIDI
Expansion Slot	microSD™ memory card (SD 2.0 compatible)
AC Adapter	Voltage range/frequency: 100 ~ 240V AC, 50/60 Hz DC output: 5V and 1A
Special Features	Digital Compass, Motion Sensor

Table 7: HTC G1 Specification Table⁷²

6.3.2 - Software

6.3.2.1 - User Interface

The aLife remote client software will be designed to give the user an intuitive, responsive and fluid interface to the aLife system. The interface design will mix text, icons, scrollable lists and gestures for the user to interact with. The user interface look similar across platforms. The iPhone remote client software should have the same look and feel of an Android version of the remote client software. Since the scope of this project is focusing on Android, the following user interface specifications will focus on Android. Remote client software build on other platforms in the future will use the Android application as a design standard.

The Android operating system has a built in user interface element called the menu bar. The menu bar is a header that contains icons that display system information like WiFi signal strength, cell tower signal strength, battery level, time and alarms. System icons in the menu bar are right justified (see Illustration 36).



Illustration 36: Android Home Page

Applications may also place icons in the menu bar which will be left justified (Illustration 37). Application icons can be used to represent things like new emails, sync progress, new instant messages, appointments, or an aLife notification. When the aLife remote client software receives a notification from the aLife central control unit it will be displayed in the upper left corner of the Android user interface, no matter what application is in the foreground. In these example screen shots you can see the menu bar with no notification icons in the upper left

of the screen. In another there is a mail icon, notifying the user that new emails are available for review. Finally, the with the menu bar dropped down the user can see more information (Illustration 38), like the email account and the number of new emails. Selecting (pressing) the notice takes the user to the email application. The user can also press the clear button to dismiss the notification.



Illustration 37: Notification Icon

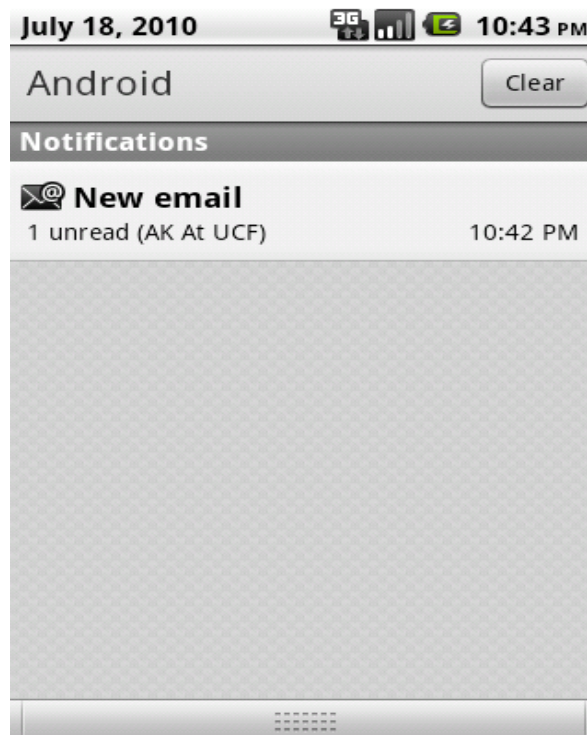


Illustration 38: The Notifications Window

The aLife remote client software will display an icon in the top left of the menu bar to indicate that there is an notice for the user to review. The user will pull the

menu bar down to review what the event is related to. Selecting the notification will take the user to the aLife remote client application where the user will be presented with more detailed information and options. Examples of notifications are energy usage, temperature issues, security and lighting. The aLife notification icon will be supplemented with overlay graphics that indicate what type of notification is waiting for the users review. If more than one notification is pending then the aLife icon will be over-layed with a plus symbol to indicate multiple notifications. The icon will have different graphic overlays to indicate security, energy and temperature notifications.

When the user has pulled down the menu bar and pressed on an aLife notification they will be taken to the aLife remote client application information and control page (See figure X). These pages are designed to give the user feedback and, if possible, control.

For example, if the aLife system is set to make sure the house is secure at 10:00pm and the garage door is open then the user will receive a security notification in the top left the Android user interface. And audible tone may also sound as described in the notifications subsection of the research section of this document. When the user pulls down the menu bar the notification window will indicate a security issue with the garage door. When the user selects the notification the user will be presented with a screen similar to the one in Illustration 39. The aLife remote client software will tell the user that the garage door is open and that the reason the notification was presented is because it is set to do so if the time is 10:00pm and the garage door is open. The user will have the option to close the garage door, set the system to ignore the garage door for the rest of the night or never remind the user in future.

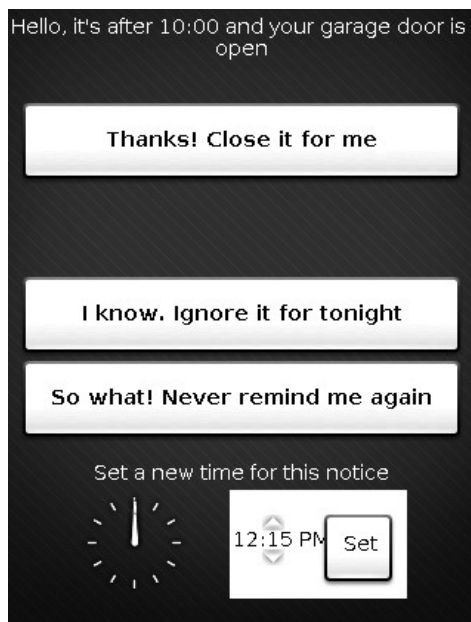


Illustration 39: Example Notification with Garage Door Control

Similarly, the aLife system may inform the user to an unsafe condition with the front or back doors (Illustration 40).

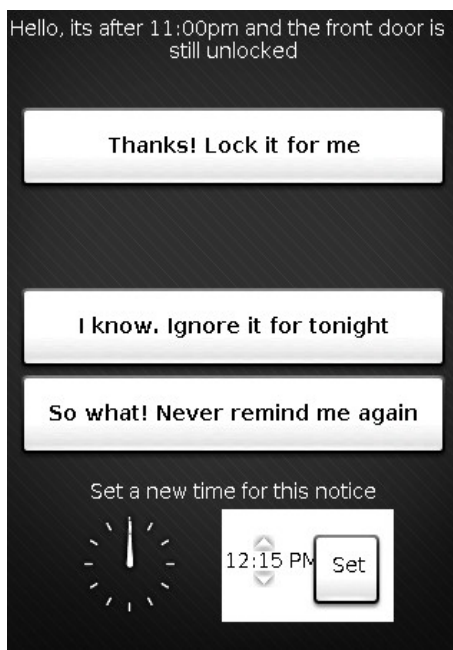


Illustration 40: Example Notification with Door Lock Control

In addition to security notifications, the aLife system may also alert the user to energy saving conditions, such as when the temperature outside is similar to the temperature inside. If temperature outside is lower than the temperature inside and the HVAC system is working to cool the house, then the system could alert the user (Illustration 41) to a money saving opportunity.



Illustration 41: Example Notification with Temperature Variables

The remote client software menu system will be easy to follow and require minimum about of pages flips or button presses to navigate. Devices currently added to the system will be listed in a scrollable list, followed by additional program options. The list will have inertial scrolling so allow for easy 'flicking' through the list. Pressing the 'back' button on the device returns the user to the previous menu.

Selecting a device will display a basic graph showing the device history. History length will be determined at testing. Devices that have an energy history will show their energy graph. Devices that have on/off states will show their on/off history. Below the history will be buttons related to individual devices (Illustration 42). Button behavior will be based on the control-ability of the device and its options.

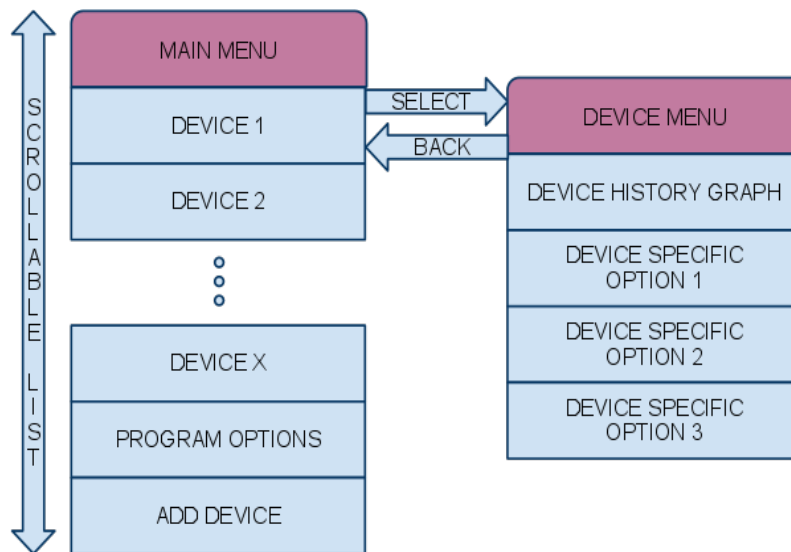


Illustration 42: Device Menu Tree

The program options page (Illustration 43) will give the user general the ability to control whether or not to maintain a constant connection with the server or periodically check the server for updates to save battery life. The user will also be able to mute just the aLife audio feedback, control if the service starts with the phone, and input user name data.

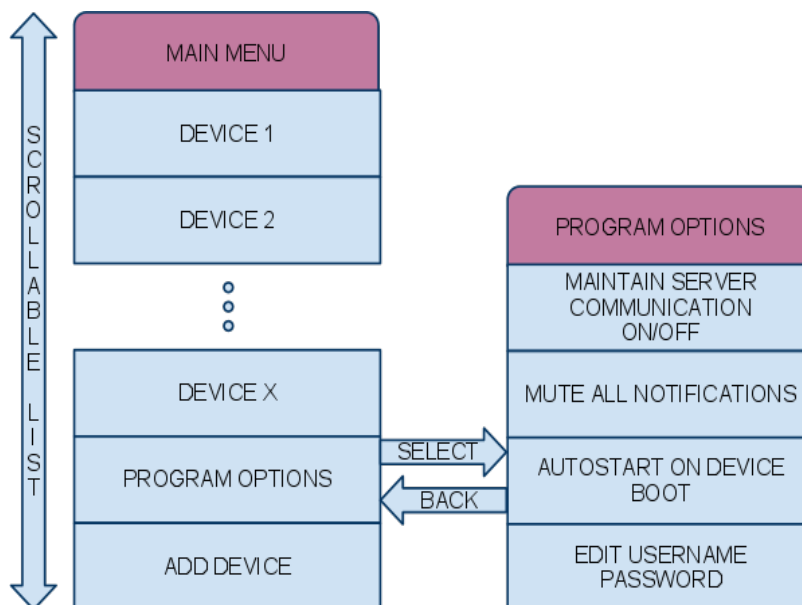


Illustration 43: Program Options Menu Tree

Adding a ZigBee device to the aLife system is simplified by the GUI we designed.

Once the device is recognized by the system, the GUI will automatically present the user with the options that are relative to that device. See Illustration 44.

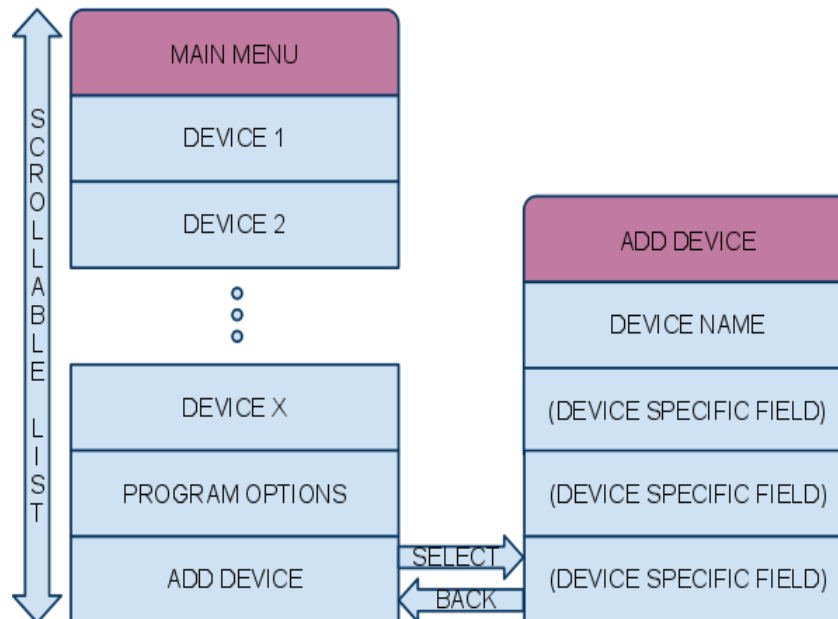


Illustration 44: Add Device Menu Tree

6.3.2.2 - Android Based Hardware

One of the reasons our group gravitated towards the Android Operating System was due to its wide-scale availability in many popular consumer electronic devices. This would allow us to leverage equipment that users already owned and were accustomed to using, such as a cell phones or tablet computers. Due to this, we decided to outline general specifications required of any potential remote client device along with a more detailed specifications list for the particular remote client device we will be using with our prototyped system.

There are three general purpose specifications required of any potential remote device. First, the hardware for the remote client device must, of course, be capable of running Android 1.6 platform or later. Second, the hardware device must provide the user a communication medium, such as a keyboard or touchscreen LCD, to interact with the aLife software system. Third, the device must be capable of accessing the Internet as that is our current means for sending/receiving request to/from the Base Station.

For our prototyped system, we would like to have our remote device fit comfortably in your hand having dimensions no larger than 5.00 in x 5.00 in x 1.00 in and weight less than 8 ounces. The device must have a replaceable rechargeable battery supply along with the ability to run off of an AC power cord.

6.3.2.3 - Remote Client Operating Systems and Software

aLife's software system was fashioned with simplicity and convenience in mind. We wanted any member of a family, young or old, to feel comfortable using our system with little to no learning curve. We also wanted to avoid having the system feel too simplistic and lose value with our clientèle. Our focus was then to make the software robust, simple, and quick. In order to meet these goals we needed our software system to be able to run on any Android enabled device running Android 1.5 platform or later. We needed the software system to provide secure data communications with the base station over the Internet and respond to user request withing a timely manner (under 2 seconds under nominal conditions). Our system will need to be equipped with a mechanism for authenticating log-ins for different user profiles. We need our software to provide help menus to our users to assist with specific tasks and general operation. We need our system to be able to provide notifications to our users whether it's the active process or just a background process.

6.3.2.4 - Client/Server Communications

The aLife server will communicate event notifications over TCP/IP using a custom protocol. The protocol establishes differences between priority levels, notification type (power, temperature, etc.), and device controllability. Table 8 specifies the the types of notifications the server will handle, the more specific notification and control options and the command and response types.

Notification Type	Event Name	Server -> Client Protocol Number	Client -> Server Protocol Number
Power Monitoring	Device Specific with Power Control	1	1
Power Monitoring	Device Specific without Control	2	3
Power Monitoring	Whole House Power Information	3	3
Temperature Monitoring	Indoor Temperature with HVAC Control	4	2
Temperature Monitoring	Indoor Temperature without Control	5	3
Lighting	Device Specific with Power Control	6	3
Security	Door Lock with Control	7	3
Security	Window Sensing	8	3
Security	Motion Sensing	9	3
Security	Garage Door with Control	10	3

Table 8: Server to Client Notification Types

Protocols are transferred over TCP/IP between the server and client. Different event notifications require different GUI's that are tailored to the event. See Table 9, Table 10, Table 11. The GUI is designed to be transparent enough to allow for additional devices be implemented in the system with minimum reprogramming required for the client. Basic GUI pages will be constructed to cover several event cases, with the headings and button field information coming from the server for each event. The following two tables describe the protocol the aLife server will use to communicate event notifications to the client.

When the client software receives a notification, it will be listed in the Android notification window. Selecting the notification in the GUI causes a page flip to a

GUI designated for the event type. The event type determines if the user will be presented with control options, temperature or time set points, power control or silencing options.

Type 1 (Power)	Type 2 (Power)	Type 3 (Power)	Type 4 (Temperature)	Type 5 (Temperature)
Notification Type	Notification Type	Notification Type	Notification Type	Notification Type
Notification Level	Notification Level	Notification Level	Notification Level	Notification Level
Notification ID#	Notification ID#	Notification ID#	Notification ID#	Notification ID#
Short Heading Text	Short Heading Text	Short Heading Text	Short Heading Text	Short Heading Text
Heading Text	Heading Text	Heading Text	Heading Text	Heading Text
Button 1 Text			Button 1 Text	Current Temp
Button 2 Text			Button 2 Text	
Button 3 Text			Button 3 Text	
Current Set Time			Current Temp	
			Current Set Point	
			Current HVAC Mode	
<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>
Display State with Control Options	Display History with Notification Options	Display History with Notification Options	Display State with Control Options	Display History with Notification Options

Table 9: Server to Client Notification Protocols 1-5

Type 6 (Lighting)	Type 7 (Security)	Type 8 (Security)	Type 9 (Security)	Type 10 (Security)
Notification Type	Notification Type	Notification Type	Notification Type	Notification Type
Notification Level	Notification Level	Notification Level	Notification Level	Notification Level
Notification ID#	Notification ID#	Notification ID#	Notification ID#	Notification ID#
Short Heading Text	Short Heading Text	Short Heading Text	Short Heading Text	Short Heading Text
Heading Text	Heading Text	Heading Text	Heading Text	Heading Text
<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>	<u>Page Flip</u>
Display History with Notification Options	Display History with Notification Options	Display History with Notification Options	Display History with Notification Options	Display State with Control Options

Table 10: Server to Client Notification Protocols 6-10

Protocol Data	Size	Description
Notification Type	1 Byte	Defines the rest of the data fields in the protocol and the GUI presented to the user
Notification Level	1 Byte	Defines how the client device alerts the user to the receipt of the notification
Notification ID#	1 Byte	Used to keep track of notification events. When the user responds with a command, the client will respond with the notification ID# that initiated the notification
Short Heading Text	<32 Characters	The space for text in the Android notification window is limited so here we'll provide the event type. Ex: "Garage Door Open" or "HVAC Temperature Problem."
Heading Text	<128 Characters	Once the user is in the aLife client application there is more screen space to present information to the user. Ex: "Hello, it's after 10:00PM and your garage door is still open."
Button 1 Text	<64 Characters	Button 1 will display control options. Ex: "Close it for me." or "Turn it off for me."
Button 2 Text	<64 Characters	Button 2 will display a snooze option text
Button 3 Text	<64 Characters	Button 3 will display an ignore option text
Current Temp	1 Byte	Current temperature indoor temperature. Outdoor temperatures will be displayed in the heading text.
Current Time/Temp Set Point	1 Byte	The current set point the HVAC is at
Current HVAC Mode	1 Byte	The current HVAC Mode: 1 - Off 2 - Cool, Fan: Auto 3 - Cool, Fan: On 4 - Heat, Fan: Auto 5 - Heat, Fan: On

Table 11: Server to Client Protocol Definitions

Every notification event will have a response option available (Table 12). A response to a notification is not necessary. The user has the option of clearing the notification out of the Android notification window without opening the remote client software. The response type is determined by the notification type. The response type is designed to be generic enough to only require three different types, as defined in the table below. The remote client software will transmit all fields of information back to the aLife server. In the case of a time or temperature set point, the remote client software will transmit the set point back to the server. The server will determine if the set point has been changed and needs updating. If the "set" button was pressed, the button number response will be zero.

Response Type 1	Response Type 2	Response Type 3
Notification ID#	Notification ID#	Notification ID#
Button # Pressed	Button # Pressed	Button # Pressed
New Set Time	New Temp Set Point	
	New HVAC Mode	

Table 12: Client to Server Response Protocols

Notifications are a critical component of the aLife system (Table 13.) Since the system is designed to run silently until a notification is sent to a user it is necessary for the receiving device to be able to receive those notifications at all times. Notifications with high levels of importance should display a pop-up alert and play a sound to alert the user whether they're playing a game, browsing the web or carrying the phone in their pocket. Notifications from the aLife system that don't require immediate attention could use less subtle notifications.

Notification Type	Notification Name	Notification Select Page Flip
1	Device Power Monitoring and Switching	Graph with Power Control
2	Device Power Monitoring	Graph
3	Whole-House Power Monitoring	Whole House List
4	Indoor Temperature with Control	Graph with control buttons
5	Indoor Temperature	Graph
6	Lighting	Device History with Power Control
7	Door Security	Device History with Lock Control
8	Window	View Device List
9	Motion Sensor	View Device List
10	Garage Door	View Device Graph with Control

Table 13: Client Notification Page Flip List

Notification Level	Notification Name	Notification Action
1	Urgent	Audible alert with popup
2	General	Audible with icon
3	Passive	Icon only

Table 14: Server to Client Notification Levels

7 - Design Summary

7.1 - Wireless Sense and Control Modules

The wireless sense and control modules generally fit into the overall aLife system as shown in illustration below:

The wireless modules will form a mesh network throughout the home to include all modules so that they all have a data route back to the base station. An example mesh network is shown in the illustration below.

Illustration 2: (Wireless network pic here - illustration 18)

Each wireless sense and control module has an MCU with a ZigBee transceiver built into it that runs the Freescale BeeStack Consumer RF4CE protocol stack. The stack interfaces with the ZigBee PHY, and MAC, as well as the MCU application layer as shown below.

Illustration 3: (zigbee protocol stack pic here - illustration 16)

The hardware to implement the MCU, ZigBee transceiver, power supply, and sensor and controls is in the schematics below, and the associated bill of materials below that.

Illustration 4:

Illustration 5:

7.2 - Base Station: Hardware

The Base Station hardware can be defined by two big components. The first being the LPC3250 OEM Board. The following Illustration is of the LPC3250 board.

Illustration 8: LPC3250 OEM Board

The functional requirements implemented by this can be found in the following Illustration. This demonstrates the functional block diagram for the microprocessor on the OEM Board.

Illustration 9: Functional diagram of microprocessor

The other main component of the hardware is the QVGA Base Board. The following Illustration is a picture of what it looks like with the LPC3250 OEM Board in place.

Illustration 10: QVGA Base Board picture

The following illustration is the functionality that this base board will bring to the project.

7.3 - Base Station: Software

The base station software will consist of the following classes, methods, and data structures.

7.3.1 - Base Station Class

The base station class will be the responsible for creating the GUI, interacting with Users, and fulfilling any autonomous services request required of the base station. The handlers for the database and socket connections available directly through the Android API which will not require our team to create any auxiliary classes. The class fields and methods are explained below.

Fields:

- PowerNotif (Boolean): Field tells whether a powered device notification is set up
- SecurityNotif (Boolean): Field tells whether a security device notification is set up
- ControlNotif (Boolean): Field tells whether a control device notification is set up
- ActiveDevices(Device []): Array of active devices in the network
- RequestedService (String): Integer representation of a request

Methods:

- BaseStation ():Constructor to create the GUI interface for our system.
- getActiveDevices(): Takes in no parameters. Returns all active devices in database.
- addDevice(Device X): Adds specified device to network and database. No return.
- removeDevice(Device X): Removed specified device from network and database. No return.
- setNotifications(): Takes no parameters. Sets all initial notifications booleans. These will dictate which notifications will be checked for continuously throughout the program.
- pollDevices(ActiveDevices): Takes in all active devices and updates their

information in the database tables. Returns nothing.

- socketParser (ByteStream): Takes in a socket byte stream and returns the requested service in string format.
- requestService (Device X, RequestService): This method will take in a device and service request and will fulfill the said service. Returns nothing.

7.3.2 - Device Class

The devices class is the abstract parent class of all specific controlled devices. It will be extended by a PowerDevice class, SecurityDevice class, and ControlDevice class.

Fields:

- ID (Int) : Device's ID
- Name (String): Name provided by user for device
- Status (Boolean): Provides status of device, on or off

Methods:

- getID(): Takes in no parameters. Returns an Int, the value of the devices ID.
- setID(Int ID): Takes in an Int and updates ID field. Does not return anything.
- getName(): Takes in no parameters. Returns a String, the value of the devices Name.
- setName(String name): Take in a String and update Name field. Does not return anything.
- getStatus(): Takes in no parameters. Returns Status of device
- setStatus(Boolean Status): Takes in a boolean and updates Status of device. Does not return anything.

7.3.3 - PowerDevice Class

The PowerDevice Class will be a factory class for power monitoring device objects.

Fields:

- TotalWatts (Int): Provides total watts used by device
- SetWatts (Int): Provides user defined limit for watt usage
- CurrentTemp (Int): Provides current temperature information
- SetTemp (Int): Provides user defined limit for temperature
- PollTime (Int): Time the information was polled

- SetTime (Int): Set time for notification purposes
- Timeframe (Int): Timeframe for power information i.e. 7 days or 30 days

Methods:

- PowerDevice(): Constructor to create object. Will have generic and specific constructors.
- turnOn(): Takes in no parameters. Sends message to Zigbee device to turn on
- turnOff(): Takes in no parameters. Sends message to Zigbee device to turn on
- getTemp(): Takes no parameters. Polls Zigbee device for temperature.
- setTemp(Int Temp): Sets Zigbee device with provided temperature. Does not return anything.
- getWatt(): Takes no parameters. Polls Zigbee device for wattage.
- getTimeframe(): Takes in no parameters. Returns the timeframe field.
- setTimeframe(Int Timeframe): Sets the timeframe field. Does not return anything.

7.3.4 - SecurityDevice Class

The Security Device Class will be a factory class for security monitoring device objects.

Fields:

- SetTime (Int): Provides set time for any notifications

Methods:

- SecurityDevice(): Constructor to create object. Will have generic and specific constructors.
- turnOn(): Takes in no parameters. Sends message to Zigbee device to turn on
- turnOff(): Takes in no parameters. Sends message to Zigbee device to turn on
- isOpen(): Takes in no parameters. Returns true if Zigbee device is open.
- IsLocked(): Takes in no parameters. Returns true if Zigbee device is locked.

7.3.5 - ControlDevice Class

The ControlDevice Class will be a factory class for controlled device objects.

Methods:

- ControlDevice(): Constructor to create object. Will have generic and specific constructors.

- turnOn(): Takes in no parameters. Sends message to Zigbee device to turn on
- turnOff(): Takes in no parameters. Sends message to Zigbee device to turn on

7.3.6 - Database

The SQLite database will be utilized to store information about current users, devices, power history, notification history, and notification set ups. Each of these elements will have their own table in our database with their own specified fields. Listed below is our database structure.

Users:

- ID (Int)
- Password (String)
- Name (String)
- Permission (Boolean)
- IP Address (String)
- Fail Attempts (Int)
- Failed Time (Int)
- Success Time (Int)

Devices:

- ID (Int)
- Device Type
- Name (String)
- Address (String)
- Field1 (Various Types)
- Field2 (Various Types)
- Field3 (Various Types)
- Field4 (Various Types)
- Field5 (Various Types)
- Field6 (Various Types)
- Field7 (Various Types)
- Field8 (Various Types)
- Field9 (Various Types)
- Field10 (Various Types)
- History1 (Various Types)
- History2 (Various Types)
- History3 (Various Types)
- History4 (Various Types)

Power History:

- Device ID (Int)

- Power Usage (Int)
- Time of Day (Int)

Notification History:

- Notification Type (Int)
- Notification Level (Int)
- Notification ID (Int)
- User ID (Int)
- Device ID (Int)

Notification Set Up:

- ID (Int)
- Description (String)
- Users (String)
- Set Time (Int)
- Set Temp (Int)
- Set Energy (Int)

7.3.7 - Remote Client: Software

The remote client software is described in the following section.

7.3.8 - Remote Client Operating Systems and Software

aLife's software system was fashioned with simplicity and convenience in mind. We wanted any member of a family, young or old, to feel comfortable using our system with little to no learning curve. We also wanted to avoid having the system feel too simplistic and lose value with our clientèle. Our focus was then to make the software robust, simple, and quick. In order to meet these goals we needed our software system to be able to run on any Android enabled device running Android 1.5 platform or later. We needed the software system to provide secure data communications with the base station over the Internet and respond to user request withing a timely manner (under 2 seconds under nominal conditions). Our system will need to be equipped with a mechanism for authenticating log-ins for different user profiles. We need our software to provide help menus to our users to assist with specific tasks and general operation. We need our system to be able to provide notifications to our users whether it's the active process or just a background process.

8 - Prototyping

The goal of the prototyping section is to discuss our team's strategy for implementing our design into a working prototyped model. This section will include information about PCB layouts, PCB manufacturing, parts acquisition and component placement, and software implementation strategies.

8.1 - Parts Acquisition

The overall strategy our team is going to employ for acquiring parts is to find the lowest priced parts we possibly can. We will continue to make cost our number one priority when choosing a vendor unless time constraints will delay the production of our final product. In order to successfully employ this strategy our team is going to need to start acquiring parts as soon as possible and utilizing as many resources as we can to compare vendors. Our team expects to get most of our parts via online vendors due to the relatively cheap prices offered and large selection of components. All items currently on loan to our group are already in our possession and do not need to be sought after during this timeframe. The purchasing of the hardware equipment will be the responsibility of our hardware designer. We expect to have all parts purchased prior to any major component installation based on the dates dictated by our milestone charts.

8.2 - Hardware Implementation

There are three major hardware components that need to be built for our project, the base station hardware, the wireless Zigbee modules, and the wireless Zigbee base station. Due to the design decisions made during the development process the hardware components needed to implement the wireless Zigbee base station are the same as those needed for the wireless Zigbee module. For this purpose we will group our implementation strategy for both of those items.

8.2.1 - Wireless Zigbee Base Station and Zigbee Module

Our team expects to spend the most of hardware development time building our wireless Zigbee modules. For this reason we wanted to have as much manpower available as we could to assist with the labor. We decided to have all software development done in the first third of our second semester so we could focus the rest of our time implementing our hardware and integrating it with the rest of our system. We expect to have all parts necessary for the build on hand prior to configuring our modules and roles assigned to each individual builder. Our plan is to have all hardware built and tested by the due dates set out in our milestone chart. The stage of development will be considered complete when the hardware modules meet all required specifications as outlined in the document.

8.2.2 - Base Station Hardware

The Embedded Artist Demo Board is the only hardware necessary to implement the Base Station hardware. We received the board on loan from InfrSAFE, Inc with all parts pre-assembled. The android port has already been compiled and configured to work on the unit. Our team does not expect to spend any time assembling or configuring the hardware for the base station.

8.3 - Software Implementation

There are two major software systems that will need to be built for our project, the base station software and the remote client software. Each system's functionality has been discussed in detail in the paper's design section but no strategy has been provided on how we will implement our design. The provided sections below will discuss our team's strategy for coding, unit testing, integration testing, and operating our system.

8.3.1 - Coding

At the end of Senior Design I our team expects to have all the design and analysis completed for both our major software systems. The natural progression from there, following our software model, would be to immediately start the coding phase. Our milestones chart indicates that we wish to have both the base station software and remote client software completed by the first third of the semester. In order to meet these needs we will need to break the software development responsibilities up between all our software engineers and work on our respective modules in parallel. The software design clearly outlines all of our modules preconditions and postconditions making it easy for our team to take this modular approach to coding. We will coordinate our current progress during team meetings while setting deadlines for completion of units. If a software engineer were to complete their section of work they shall move on to other, uncompleted units until both software packages are complete which will signify the end of this stage of development.

8.3.2 - Unit Testing

Both the remote client software and base station software should be completed by the first third of the semester. Following the guidelines of our milestone chart the unit testing shall be completed by the first third of the semester of Senior Design II as well. Unit testing responsibilities will be divided amongst all of our software engineers. Each module created by a software engineer needs to be individually tested to verify all preconditions and postconditions are met. If a test unit fails it is the responsibility of the software engineer to troubleshoot the issue and specify whether a redesign in specifications or design is necessary. If a change is needed then the issue will be brought to the entire software team to discuss and rework. This stage of development will be considered completed when all modules for the base station software and the remote client software have successfully passed all unit tests.

8.3.3 - Integration Testing

The integration testing shall take place after all unit testing has been completed. Following the guidelines of our milestone chart all integration testing shall be done by the first third of semester of Senior Design II. The integration testing will involve our software engineers following the guidelines of our test plan on both the remote client device and base station software. All faults will be categorized based on our evaluation criteria and will be debugged by the individual who discovered the bug. It is the responsibility of the software engineer who is running the test to specify whether a redesign in specifications or design is needed due to a test failure. If a change is needed then the issue will be brought to the entire software team to discuss and rework. The completion of this stage will be denoted by the successful passing of all test cases for both the base station software and the remote client device software.

8.3.4 - Operation

The operation stage takes place when the remote client software, base station software, and aLife hardware components are complete and have been integrated together. Following the guidelines of our milestone chart this stage shall be completed by the end of Senior Design II. This stage demonstrates our system's ability to meet all functional specifications described throughout our document. Every member of our team will be responsible for testing our software's ability to integrate with other units of our project. If there is an issue that is discovered it is up to that team member to decide if redesign is necessary in the specifications or design. If a change is deemed necessary then the issue will be directed to the appropriate work group and will be reworked. This completion of this stage will be signified by a successful demonstration of our product to our Senior Design Panel.

9 - Testing and Evaluation

9.1 - Test Plan

aLife is a complicated system with many hardware and software components to consider when selecting a testing plan to follow. With the built in modularity of the three subsystems of the project and the need to interconnect them at the final product, we decided to go with an integration style testing procedure. With this type, it offers us a chance to first do some basic tests on the individual systems before putting the final project together. To further help us in this goal we decided to go with the Bottom-Up Integration Testing scheme. The following figure shows a representation of what a Bottom-Up implementation looks like.

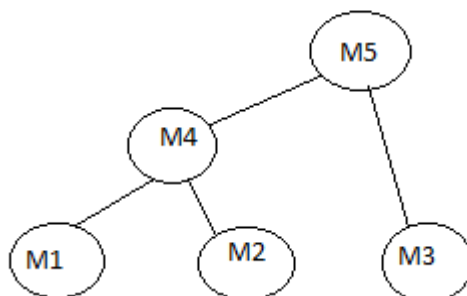


Illustration 45: Diagram of a Bottom-up Integration Test

This style of testing, as seen in Illustration 45, begins at the most basic levels of the system, and starts testing there. In this example they start with M1, M2, and M3, and in aLife we will start with the 3 basic components of the system: the Base Station, the remote client, and the ZigBee network. Once all of the tests, run until a satisfactory result is obtained, we will start integrating the systems together. Once 2 or more sections are integrated, we will run the tests for each of the individual sections on this newly integrated part and test for correctness. This method allows for easy testing for a very modular project, like aLife is, and it allows us to focus on the very fundamentals of what our project is oriented to do. It makes making test cases easier since there is a lot of re-use of them and it allows for a very thorough test once the entire project is put together. The following tables are the guidelines that we are going to follow for implementing each of the different systems for aLife.

For hardware testing using this model, it would be very tedious to check each component individually on the board and then reassemble it in order to test a completely different unit being integrated into the project. For this reason, this particular testing scheme will deal with testing each of the individual specifications of the hardware components. Once that test has been established as a "passing" score, we will put the hardware pieces together and do an overall hardware systems check. This will allow us to continue to use the Bottom-Up model, as we will keep checking each of these requirements each time a new part is added, and we also don't have to tear apart our whole board when there is new testing to be done.

9.1.1 - Base Station: Hardware Tests

9.1.1.1 - Component

Component	Test Name	Description
NXP LPC3250 Microcontroller	Power	Test that the microcontroller works at 1.2V by supplying that much power to it
	Timers	Run a test on the timers to make sure they are working correctly
	Memory	Power up the microcontroller and see if you can process anything by using the memory
	Serial Interfaces	Connect the device to the appropriate interface to make sure the serial interfaces are working as intended
	USB	Hook up a USB device and make sure that the controller can interface with it
	Ethernet	Test whether the microcontroller can interface properly with the internet
	Memory Card	Test to see if the memory card interface by putting in the appropriate SD cards and checking the results
	LCD controller	Test that the microcontroller has the LCD controller working
	Keypad interface	Hook up a keypad to see if the interface is working
	External memory controller	Test by installed the appropriate DDR and SDRAM and making sure it works
	Low Power Mode	Turn the device to .9V to make sure that low power mode is working properly

Table 15: Microcontroller tests

In Table 15 above, we test each of the individual features of our chosen microcontroller. Once we have determined that all of those features are working correctly, we will then place it in the OEM Board. With that in place, we will begin

to test the board and the microcontroller together, as seen in the following table. Since we have to integrate all of the hardware pieces together as we are building the base board, some of the test for each of the components will overlap.

Component	Test Name	Description
LPC3250 OEM Board	Power	Supply the board with 3.15V-3.3V and verify that is operational
	External Flash	Connect the recommended 128 MB NAND FLASH and test that it is working
	Data Memory	Test that the 64MB SDRAM is working properly by having the board store some type of information.
	Connectors	Verify that the expansion connectors are correct and that they will interface with the Base Board
	LED	Run a test on the board to verify that it can output some results to the LED's
	Databus	Verify that the databus will transmit data to the destination

Table 16: LPC3250 OEM Board tests

From the table above, we have thoroughly measured the working specifications of the OEM Board. With that component working, we will move on the final state of integration of the Base Board component. Since this is the final step of the hardware integration we want to make sure that systems vital to the performance of this piece are retested in this integrated environment. The following table outlines the tests that will be performed on the Base Board component.

Component	Test Name	Description
QVGA Base Board	Power	Test that the device can draw it's power from either a USB port or directly from 9-15V DC
	LCD	Test that the LCD screen is operational and make test the LCD controller in the microprocessor is working
	OEM SODIMM connector	Plug the OEM board into the Base Board to verify the connectors are working
	Ethernet	Connect an Ethernet cable to make sure the interface is working properly
	USB	Connect a USB device and verify the Base Board can act as both a master and Slave to the external device
	RS232	Make a connection to the RS232 interface and verify that is working properly
	IR transceiver	Test the IR transceiver by allowing it to transmit and receive IR signals
	Power Supply	In accordance to the voltage requirements in the earlier test, make sure this falls within acceptable ranges
	5-key joypad	Enable the joypad and verify it works through use of the LCD
	Accelerometer	Verify the 3 axis is operational by use of a specific LED lighting up when the Base Board is turned
	Analog input	Verify the analog input is working correctly by connecting the proper input and verify the input date is correct
	USB/Serial Bridge	Test that the USB to serial bridge is operation by connecting a USB device and having it output data to the serial bus
	Reset button	Test that the reset button will reset the Base Board and any integrated components on board
	Speakers	Test the speakers by having the Base Board output some type of audio sound

Table 17: QVGA Base Board tests

With these of testing cases completed, we now have a working base board model that we can then use in the rest of the integration testing. As before stated, it would be very inconvenient for the previous 3 table's worth of testing procedures had to be replicated whenever one of the 3 subsystems was added into the integrated part of the project. For that reason, the following table will be all of the tests we chose to implement.

9.1.1.2 - Base Board

Test Name	Description
Plug in the base station	Turn on the base station and make sure all of the components are on
Check Embedded Artists board	Make sure that the embedded artist board is functioning correctly.
Drivers	Test to see if the drivers for the board is properly controlling
Test Memory	Make sure all the memory on the board, built in or external, is fully functional
Ports	Check to make sure that all output ports on the device are working
Ethernet	Test that the board is properly receiving Internet
USB	Make sure that the built on USB port allows for devices to be connected to it
Data Communication	Test that the output ports of the base board and the USB port are all able to send and receive information correctly
LCD Screen	Make sure the LCD screen is properly powered and able to transmit any relevant data to it's screen
Reset	Allow for testing of the Reset button to make sure that the board can properly be restarted
5-key Joypad	Make sure the built on joystick on the board is working properly
Speakers	Test that the speakers are turned on and allowing for noise to be emitted

Table 18: Integration tests - Base Board

The above list represents the core features that we need working for our Base Board to be functional and our project to be able to function properly.

9.1.2 - Software Tests

Below is Table 19 - Table 22, the Base Station's Software's test cases. The provided test cases will verify the base station software is working as expected.

Test	Test Objectives	Test Description	Expected Results
Create New User / Remove a User	Verify the base station is able to add new users and remove users	Register a new user option selected. A valid user login and password are provided. After successfully registering the user will be removed from the system	New user credentials will be accepted by the system and a new entry will be added to the user table. The user will then be unregistered and removed from the table.
Login/Log out	To verify a user can log in and out of the base station software	Login option selected. A valid user's login and password are provided. After successful login the log out option is selected.	A user is successfully granted access into our system. After logout option selected the user is successfully logged out of the system.
Multiple User Request	Multiple users will send request to the user to verify all request are fulfilled on a first come first serve basis	Multiple users will submit the same request to the same ZigBee device. Multiple user will respond to the same notification at the same time.	The base station will respond to the request on a first come first serve basis. All unsuccessful request to have a response notification sent to the user.

Table 19: Software Tests - part 1

Test	Test Objectives	Test Description	Expected Results
Modify Database Table	To verify that the system can add, remove, or update all table in the database	A method call to the database handler class will be made to add a row, delete a row, and modify a row in every table in the database	All data gets inserted successfully, all data get deleted successfully, and all data gets modified successfully.
Connect with Remote Device	To verify we can make successful duplex TCP connection's between the base station and remote client devices	A connection attempt will be made with one remote client device. Every successful connection a new device will attempt to connect up until 5 total devices are connected	All remote client devices shall successfully connect to the base station
ZigBee Device Check	To verify the base station can communicate will all ZigBee devices currently on the network	A call will be made to each ZigBee device connected to the network and all that ZigBee device's functions will be test	The base station will be able to successfully communicate with the ZigBee device and all commands will work as expected

Table 20: Software Tests - part 2

Test	Test Objectives	Test Description	Expected Results
Notification Delivery Test	This test is to verify only users set up for notifications receive notifications and that each notification has a unique ID	We will feed the system simulated information that will make a notification condition true. The base station will cross reference the notification set up table and only send notifications to user who are registered. The notification history table will be updated with notifications sent out. This will be done for all notification types.	Only those user who requested notifications will receive them. All notifications will be sent out with a unique notification ID.
Create Database	To verify that a database can be created by the base station	A method call to the database handler class will be made to create a new database	A new SQLite database file will be created
Locked Out	To verify security mechanism to lock out user who place in incorrect login credentials at least 5 times are locked out	Five login attempts will be made with a non valid user login and password	A user should be prompted of login error and not be able to login to the system for at 5 minutes.

Table 21: Software Tests - part 3

Test	Test Objectives	Test Description	Expected Results
Add / Remove ZigBee Device	To verify the base station can add or remove a ZigBee device from the ZigBee network	An attempt will be made to add a new ZigBee device to the network. After successfully adding the device to the network an attempt will be made to remove the device.	The device will be successfully added to the network and then successfully removed from the network.
Self Polling	To verify the base station is able to poll all ZigBee devices withing five minutes and update their status information in our database	The base station will poll all active ZigBee devices on our network for their status information and send that data to our database	The base station will successfullly receive the status information and successfullly add it into our database
Single User Request	A single user request will be made to every ZigBee device on the network to verify all capable functions can be fulfilled	The user will request all available service from every ZigBee device on the network	All request will be fulfilled sucessfullly

Table 22: Software Tests - part 4

9.1.3 - Remote Client tests

Table 23 is the Remote Client Software test cases. The provided test cases will verify the base station software is working as expected and meets all required specifications.

Test	Test Objective	Test Description	Expected Results
Turn on	Be able to turn on the Remote Client	Physically turn on the device and see if it functions properly	The device should be able to turn on properly
Log on/off	Be able to log in or out of your profile at the Base Station	Have the Base Station on and be able to log into a profile from the remote client	There should be no problem logging in and out of a profile
Test Notifications 1	Be able to receive notifications	Have the Base Station send a notification to the remote client	The remote client should be receiving notifications
Test Notifications 2	Be able to send notifications	Send a notification back to the Base Station and verify that it has been received	It should also be able to send them out
View network	Be able to view active ZigBee nodes	Access the network of active ZigBee devices from the Remote Client	The user should be able to see the network of active nodes
Access a Node	View information being collected from a node	Select a ZigBee module and be able to see information regarding its status or anything else it reports	The user should be able to read information regarding a particular node
Add/Remove a node	Be able to add or remove a ZigBee device from the remote client	Use the Remote client to add a ZigBee device to the network, then use it to remove one	The Remote Client should be able to add/remove with no problems
Notification interaction	Be able to interact with incoming notifications if they allow for it	If a notification is able to be interacted with	The notification should work correctly upon receiving a command
Edit Profile	Be able to edit some settings on your use profile	Change something in your profile and make sure that the change saves	The profile information that we edited should be correct
View History	If a node records history, be able to view its past performance or data	View the history of a device from the Remote Client	The history of a device should be properly displayed

Table 23: Remote Client Tests

9.1.4 - ZigBee Modules

Below is Table 24 and Table 25, the ZigBee Module test cases. The provided test cases will verify the ZigBee module is working as expected and meets all required specifications.

Test Name	Test Objective	Description	Expected Results
Enable Device	Make sure the ZigBee module is able to be connected to the network	We will supply power to a ZigBee Module and attempt to detect the device	The ZigBee device should be detected by the Base Station
Add Device	Add a ZigBee Module to the ZigBee network	Integrate the current module into the network	The module should be correctly added to the network
Power	Verify that the device is getting a proper power supply	Test that the voltage supply is between 3.4V DC and 3.2V DC	The power should be within the proper range
Ripple	Determine if the ripple voltage is within parameters	Test that the ripple voltage is < 50mV	The ripple voltage should be at a nominal level
Power Monitoring	Test whether the device can accurately detect power	Have the module plugged into a wall and measure the power drawn. Compare that with the information the module is recording	The module should be within a reasonable range of correctness: >85%

Table 24: ZigBee Modules Tests part 1

Dimmer	Make sure the light dimmer feature is working correctly.	Have a light connected to the module and look to see if the light dims as you adjust the settings.	The module should allow some control of dimming functionality to the light
I/O	Verify that the I/O lines of the ZigBee device is working	Measure the analog and digital separately to verify they are outputting the correct information	All of the output ports should be functioning correctly
Transmission range	Test the transmission range of the device	Measure that the ZigBee module can transmit upwards of 30 meters away	The ZigBee should be able to transmit at maximum range, or within 2 meters of it
Data rate	Test the data rate of the system	Verify that the module can transmit data at speeds $\geq 200\text{kb/s}$ by collecting the output and clocking the speed	The output should be at least 200 kb/s to pass the test
Disable Device	Be able to disconnect the device from the network	Be able to disconnect the device from the network, either physically or remotely	Test that the device will correctly disconnect from the ZigBee network
Power Down	Turn off the device	Test that the device can be properly shut down	The device should be able to be powered off
Correct Data	Verify the correctness of the data being transmitted from the module	Depending upon the application this module was created for, determine correct output logically and compare that to the physical output being received	The device should be able to output information, and that information should be correct in the context in which the device is sending it

Table 25: ZigBee Modules Tests part 2

9.2 - Evaluation Criteria

Testing is a very integral part of any project. Since aLife incorporates so many types of hardware and software features working with each other, thoroughly testing each of the parts will be a challenge. In order to ensure proper testing is done, we need to define what is and what isn't going to be considered a fault and what category that type of fault falls under. This will allow for easier individual testing, and will help ensure that any integrated system tests towards the conclusion of our project are made easier.

Software faults

Doesn't run at all	Only runs partially	Runs but no output	Incorrect output	Correct Output, but Slow Performance
1	2	3	4	5

Table 26: Hardware faults categories

Doesn't turn on	Not reliable 100% of the time in it's functionality	Turns on but no output	Incorrect output on the ports	Slow processing time
1	2	3	4	5

Table 27: Software faults categories

The above tables list out specific criteria that will be looking for in our project in order to identify any faults that might have occurred. The label fields at the top are a description of what type of fault has occurred, while the number below is corresponds with a ranking of how "bad" that particular fault is. With this list we can then work towards testing each of the individual sections and allow corrections to be made according to urgency.

The lower the score of a particular part, for example the OEM board, the worse condition it is in. For the aLife to be a reliable and efficient system, we need these tests to be conducted and passed within a very high margin. Although some of the categories don't apply in all cases to all modules being tested, the tester can interpolate the region in the scale where the part functioned incorrectly. This will allow us to quantify our test results and give us a standard, testable basis for retests and comparisons between modules. Since we want to keep accurate results and allow some type of comparisons between the hardware and software components of this project, the tests will be formatted in the following way. For each module, the tester will find the proper list of testing cases that are required to thoroughly test that piece. Once the tests have been completed, the numbers you gave each of the sections indicating where they ranked on the above Hardware/Software scale is written down and added. Since 5 is the "best" score that you can give any test area, we will find the ratio between what we scored the module and what the total score was. If it was anywhere under 90%, there will be a discussion and research on where the module was struggling at. In the case of multiple modules being integrated together failing to produce the intended percentage, then both will be taken back apart and improved individually before retrying the integration testing.

10 - Project Management

One of the most important elements to having a successful senior design project is team management and organization. Having four individuals with a great foundation in math, science, and engineering principles will only get you so far, it takes good team communication and a well organized plan of execution in order to facilitate a productive work environment. This, as our team found, is a principle that is a lot easier to understand than it was to implement. Issues such as scheduling coordination, distributing balanced workloads, compromising on design ideas, and dealing with last minute adjustments were not easy issues to address and could have easily caused a disruption in our work flow or, even worse, led to an unsatisfactory final product. This was obviously an unacceptable outcome for our team which lead us to focus a lot of our early energy on establishing routine meetings, staying organized, and setting clear milestones for project completion.

10.1 - Team Meetings

Meetings were a pivotal tool used by our team to stay up to date with each individuals current progress and keeping everyone focused on meeting our desired objectives. Our team set a goal to have at least one meeting a week with all group members attending. Most of our meeting were done virtually through Skype (Beta) Version 5.0.0.105 using the video conferencing feature with in person meetings only held on an as needed basis. This helped to alleviate a lot of the coordination issues we were experiencing with everyone's schedules. We also incorporated an open forum style of discussion where any team member could provide their input on the topic at hand or voice their concern about a specific issue. Although this was an effective strategy to initiate discussion it became troublesome whenever we allowed ourselves to get off topic. To combat this issue we decided to start each meeting by creating objectives we would like to complete prior to the conclusion of the meeting. This helped our group stay organized and kept all members focused on the task at hand. At the conclusion of our meetings we would discuss what milestones we would like to achieve prior to our next meeting. This allowed our team to take a progressive approach to completing our design project and is a practice we plan to continue going into our second semester.

10.2 - Team Organization and Responsibilities

Below, in Table 28, is a list of our team's departmental responsibilities and a description of each title listed. It was our belief that no one person should lead our team for the entire duration of our project. We felt the best solution would be to cycle that roll throughout both semesters between all group members.

Amos	Project Manager (1st half/1st Sem), Software Engineer, Researcher, Assembler, Software Unit/Integration Tester
Tim	Project Manager (2nd half/1st Sem), Software Engineer, Researcher, Assembler, Software Unit/Integration Tester
Todd	Project Manager (1st half/2nd Sem), Software Engineer, Researcher, Assembler, Software Unit/Integration Tester
Jake	Project Manager (2nd half/2nd Sem), Hardware Designer, Embedded Software Engineer, Researcher, Assembler, Hardware Tester

Table 28: Team Member Responsibilities

Project Manager: Directs, coordinates, and exercises functional authority over all other active engineers in the group. Is also responsible for leading team meetings and providing the final word on team disputes.

Software Engineering: Responsible for the design, creation, and maintenance of aLife's multiple software systems.

Hardware Designer: Responsible for layout design and implementation of custom hardware components.

Researcher: Responsible for creating research topics for documentation and researching said topics.

Assembler: Assists in the assembling of aLife's hardware components.

Hardware Tester: Is responsible for following following all test cases outlined in our Hardware Test Plan and reporting any issues found. Also responsible for making sure the system meets all outlined specifications.

Software Unit/Integration Tester: Is responsible for following the test plan created to test all software modules and report any issues found. Also responsible for making sure the software system meets all outlined specifications.

10.3 - Milestone chart

Due to the large scale nature of this project and the unique group dynamics involved, it was very important for our team to have direction throughout each semester of our design project. For this reason we established a Milestone chart for Senior Design I and for Senior Design II. The milestones listed below incorporate most of the important objectives needed for project completion but is not all inclusive. Any additional milestones added to our team's agenda will be discussed during team meetings and scheduled for completion at that time.

Below is Table 29 and Table 30, Senior Design I and Senior Design II milestone charts respectively.

Milestone	Due Date	Status
Form a Project Group	End of Second Week	Completed
Choose a Project	End of Third Week	Completed
Create a Meeting Schedule	End of Third Week	Completed
Create Roles for Team Members	End of Fourth Week	Completed
Functional Analysis	End of Fourth Week	Completed
Project Identification Document	6/28/10	Completed
Table of Contents	7/04/10	Completed
Distribute Roles for Documentation/Design	7/04/10	Completed
Research	Periodic / End of Semester	In Progress
Have 40 pages documented	7/19/10	Completed
Have 80 pages documented	7/26/10	Pending
Have 120 pages documented	Last Day of Class	Pending

Table 29: Senior Design 1 Milestone Chart

Milestone	Due Date	Status
General Purpose Refresher Meeting	First Week	Pending
Redefine Design Roles if necessary	First Week	Pending
Complete Low Level Design of all Hardware	End of Third Week	Pending
Complete Low Level Design of all Software	End of Third Week	Pending
Update Research and Documentation	End of Third Week	Pending
Complete Base Station Software	First Third of Semester	Pending
Complete Remote Client Software	First Third of Semester	Pending
Complete Database Design	First Third of Semester	Pending
Complete Software Unit Testing	First Third of Semester	Pending
Complete Software Integration Testing	First Third of Semester	Pending
Update Research and Documentation	First Third of Semester	Pending
Complete Base Station Hardware	Second Third of Semester	Pending
Complete ZigBee Base Station Hardware	Second Third of Semester	Pending
Complete Remote ZigBee Controller	Second Third of Semester	Pending
Complete Hardware Testing	Second Third of Semester	Pending
Update Research and Documentation	Second Third of Semester	Pending
Complete Entire System Integration	Last Third of Semester	Pending
Make sure systems meets all required specifications	Last Third of Semester	Pending
Finalize Prototype and Demonstrate	Last Third of Semester	Pending
Finalize Research and Documentation	Last Third of Semester	Pending
Complete Presentation Slides	Last Third of Semester	Pending
Submit Project to Panel and Demonstrate	Last Day of Class	Pending

Table 30: Senior Design 2 Milestone Chart

10.4 - Budget and Financing

One of our team's motivations at the onset of this project was to keep this system as low cost as possible. This made it important for us to only integrate high value, low cost components into our design or to try to get as many donated parts as possible. Our early goal was to have all purchased equipment equal no more than \$600 with all cost being evenly distributed between all group members. Well felt that this goal was not too ambitious and would still allow us to have a full featured final product. We still, however, had to be diligent in our

decision making as we did not want to spend any of our money unnecessarily. So the key to success in meeting our budgeting goals was to find as many free parts as we could.

Luckily our group was able to find a company that was willing to provide us a few key parts. InfrSAFE, Inc loaned us, free of charge, the Embedded Artists NXP LPC3250 demo board which includes all hardware listed for the base station except for the Freescale MC13213 processor and antenna. It is the same device as the remote ZigBee control units listed below but with some components depopulated. This greatly reduced the out of pocket cost of the base station hardware. This left us with finding parts to accommodate the needs for our ZigBee remote control units, ZigBee base station, and our Android enabled remote client device.

One of our team members volunteered the use of his Google G1 cell phone to use as our Android enabled remote client devices. This device was the first cell phone to be manufactured directly by Google and host the Android operating system. The device was very familiar to our group and had a proven history of running user built android applications. This was an additional item we were able to acquire free of charge.

The ZigBee remote control units and ZigBee base station account for the majority of our project's budget. The layout for the modules were created by our team's hardware designers and all components were purchased individually. We did not have any additional cost added to our budget for assembling the units but we do expect to have pay for PCB production.

Listed below in Table 31Error: Reference source not found is the estimated cost of all components needed for the completion of our project. All prices listed are based off what we expect to pay for the item as we have yet to complete our parts acquisition.

Item Description	Unit Price	Quantity	Total
Base Station			
Embedded Artists NXP LPC3250 demo board	\$0.00 (on loan)	1	\$0.00
Wireless Sense and Control Units / Wireless Base Station			
PCB Components	\$ 15.00	4	\$60.00
PCB – Bare Boards	\$10.00	4	\$40.00
Chasis	\$5.00	3	\$15.00
Board Assembly Labor	\$5.00	4	\$20.00
Additional Items			
Google G1	\$0.00 (on loan)	1	\$0.00
Total			\$135.00

Table 31: Project Budget

11 - Final Summary

We started this project with the goal of creating a system that would help simplify peoples lives. This novel concept grew into an idea, this idea turned into a project, the project became a design, and soon the design will be realized into an actual product. The entire process was very long and full of lessons learned we hope we can take with us. Two, in particular, come to mind that helped define our team's approach to creating a successful design project. The first lesson learned was the importance of keeping our group coordinated and oriented towards task completion. The dynamics involved with working with four full time students can cause many issues with scheduling and balancing priorities. We addressed these issues early by placing a focus on attending regular team meetings and keeping opens lines of communication between our group. Also, our team learned relatively early the importance of understanding what we were capable producing. We had very ambitious goals for our project in the early stages and not all of our expected functionality could make it to our final design.


This is a natural repercussion of the design process and it not always a bad thing. Comparing different solutions and refining our choices allowed our team to have a very sound and scalable project. We feel it was this deliberate approach to problem solving that allowed us to successfully research and design our project and we plan to continue this approach when implementing the second phase of development.

12 - Appendices

12.1 - Copyright permissions

Hotmail - zynergy07@knights.ucf.edu - Windows Live

http://co103w.col103.mail.live.com/default.aspx?wa=wsignin1.0



University of
Central Florida
by Windows Live™

[Mail](#) | [Office](#) | [Photos](#) | [More](#) |

zyr

Hotmail

Inbox (333)

Folders

- Junk
- Drafts (2)
- Sent
- Deleted (3)**
- [New folder](#)

Quick views

- Flagged
- Photos (6)**
- Office docs (14)**
- Shipping updates

Messenger

[Sign in to Messenger](#)

New | Reply | Reply all | Forward | Delete | Junk | Sweep ▾ | Mark as ▾ | Move to ▾ |

Re: request permission to use photo

This message is part of a conversation. [Show conversation](#)

JJ Robinson [Add to contacts](#)

To: zynergy07@knights.ucf.edu

8/03/10
[Actions](#)

From: **JJ Robinson** (jrobinson@control4.com)

Sent: Tue 8/03/10 10:03 PM

To: zynergy07@knights.ucf.edu

[Back to message](#) ↶

[Options](#) ▾

Jacob,

You have our permission to use that photo as long as you attribute it "Photo courtesy of Control4."

jj robinson

On 8/3/10 2:50 PM, "zynergy07@knights.ucf.edu" <zynergy07@knights.ucf.edu> wrote:

Mr Robinson,

Can I please have permission to use a photo of the SR-250 remote control in a paper for a project at UCF?

Thank you,

Jacob Peery

Home

Contacts

Calendar

© 2010 Microsoft | [Terms](#) | [Privacy](#) | [Advertise](#)

[Help Center](#) | [Feedback](#) | [English](#)



Mail (333) Office Photos More |

zyr

Hotmail

New | Reply Reply all Forward | Delete Junk Sweep ▾ Mark as ▾ Move to ▾ |

Inbox (333)

Folders

- Junk
- Drafts (2)
- Sent
- Deleted (3)**
- [New folder](#)

Quick views

- Flagged
- Photos (6)**
- Office docs (14)**
- Shipping updates

Messenger

[Sign in to Messenger](#)

Re: Request for permission to use images

[Back to message](#) Options ▾

This message is part of a conversation. [Show conversation](#)

Maria Hall [Add to contacts](#) 8/03/10
 To zynergy07@knights.ucf.edu [Actions](#)

From: **Maria Hall** (maria.hall@embeddedartists.com)
 Sent: Tue 8/03/10 5:57 PM
 To: zynergy07@knights.ucf.edu

Dear Jacob,

You may use all the pictures and the information about the boards that you find on our public web pages (if you need to publish information from our support site please let me know). We are very interested in your project! Could you perhaps send us a copy? We will soon have a university page on our web site. Perhaps we can publish some information about your project there?

Best regards, Maria

2010/8/2 <zynergy07@knights.ucf.edu>

Hi Maria

My name is Jacob Peery and I am a student at UCF in Orlando, FL USA and I am using the LPC3250 demo kit in a project. As part of the project, I must write a paper documenting the design process and the components used, and I was wondering if I can have your permission to include pictures of the LPC3250 OEM board and base board along with a copy of the feature and spec tables for those boards?

Thank you very much,

Jacob Peery

--

Maria Hall, Managing Director

Embedded Artists AB
 Södra Promenaden 51
 SE-211 38 Malmö, SWEDEN

Mobile: +46 (0)739-83 86 25



Mail (333) Office Photos More |

zyr

Hotmail

New | Reply Reply all Forward | Delete Junk Sweep ▾ Mark as ▾ Move to ▾ |

Inbox (333)

Folders

- Junk
- Drafts (2)
- Sent
- Deleted (3)**
- [New folder](#)

Quick views

- Flagged
- Photos (6)**
- Office docs (14)**
- Shipping updates

Messenger

[Sign in to Messenger](#)

RE: Request for permission to use images

[Back to message](#) Options ▾ ↑

This message is part of a conversation. [Show conversation](#)

North Andrew-B07863 [Add to contacts](#) 8/02/10
 To zynergy07@knights.ucf.edu [Actions](#)

From: **North Andrew-B07863** (B07863@freescale.com)
 Sent: Mon 8/02/10 8:05 PM
 To: zynergy07@knights.ucf.edu

Jacob – yes, that's fine. Thanks.

Andy North
 Corporate and Microcontroller Public Relations
 Freescale Semiconductor
 Office: 512-996-4418
 Mobile: 512-203-7169

From: zynergy07@knights.ucf.edu [mailto:zynergy07@knights.ucf.edu]
Sent: Monday, August 02, 2010 3:03 PM
To: North Andrew-B07863
Subject: Request for permission to use images

Hey Andy,

My name is Jacob Peery and I am using the MC13213 MCU for a project at the University of Central Florida. As part of our project we have to document the design process and provide details on the components we used. I was wondering if I can have your permission to use images from the MC13213 datasheet in our report?

Thank you very much,

Jacob Peery

- Home
- Contacts
- Calendar



Mail (333) Office Photos More |

zyr

Hotmail

New | Reply Reply all Forward | Delete Junk Sweep v Mark as v Move to v |

Inbox (333)

Folders

Junk

Drafts (2)

Sent

Deleted (3)

New folder

Quick views

Flagged

Photos (6)

Office docs (14)

Shipping updates

Messenger

Sign in to Messenger

RE: request permission to use photo

Options Back to message

This message is part of a conversation. [Show conversation](#)

Kristoph Kot [Add to contacts](#) 8/03/10
To zynergy07@knights.ucf.edu [Actions](#)

From: **Kristoph Kot** (Kot@p3international.com)
Sent: Tue 8/03/10 8:58 PM
To: zynergy07@knights.ucf.edu (zynergy07@knights.ucf.edu)

Hi Jacob,

Thank you very much for your inquiry. We appreciate your interest in the P4400 Kill A Watt.

Yes, you may use an image or images of the P4400 Kill A Watt unit for your project at UCF.

If you need any additional assistance of images of the product please do not hesitate to cont me.

All best,
Kris

Kristoph Kot
Account Specialist
P3 International
T: (212)346-7979 Ext. 8276
F: (212)346-9499
kot@p3international.com

From: zynergy07@knights.ucf.edu [zynergy07@knights.ucf.edu]
Sent: Tuesday, August 03, 2010 4:54 PM
To: Kristoph Kot
Subject: request permission to use photo

Hi

Can I please have permission to use a photo of the Kill A Watt module in a paper for a project at UCF?

Thank you,
Jacob Peery

Home
Contacts
Calendar



Mail (333) Office Photos More |

zyr

Hotmail

New | Reply Reply all Forward | Delete Junk Sweep ▾ Mark as ▾ Move to ▾ |

Inbox (333)

- Folders
- Junk
- Drafts (2)
- Sent
- Deleted (3)**
- [New folder](#)
- Quick views
- Flagged
- Photos (6)**
- Office docs (14)**
- Shipping updates
- Messenger
- [Sign in to Messenger](#)

RE: permission to use product images

[Back to message](#) Options ▾ ↑

Mike Scharnagl [Add to contacts](#) 8/02/10
To zynergy07@knights.ucf.edu [Actions](#)

From: **Mike Scharnagl** (mscharnagl@smarhome.com)
Sent: Mon 8/02/10 7:47 PM
To: zynergy07@knights.ucf.edu

Jacob –

You have permission to use the images provided we are referenced as the source.

Regards,
Mike

Mike Scharnagl
Director of Marketing
Smarhome, Inc.
Home Automation Superstore

<http://www.smarhome.com>

Tel: (949) 221-0037 ext 152
Fax (949) 221-9241

Smarhome Forum [Home Automation Forum](#)
Twitter [SmarhomeInc](#) & [INSTEON](#)
Facebook [Become a Fan!](#)
YouTube [Watch Our Videos](#)
Subscribe to the [Smarhome Newsletter](#) to receive exclusive deals

Smarhome's Core Value #1: We deliver a customer experience that earns our customers' loyalty and confidence

From: zynergy07@knights.ucf.edu [mailto:zynergy07@knights.ucf.edu]
Sent: Monday, August 02, 2010 12:38 PM
To: SmartLabs PR
Subject: permission to use product images

Hi Mike

- Home
- Contacts
- Calendar

Illustration Index

Illustration 1: Android OS Software Stack	10
Illustration 2: The Waterfall Model.....	12
Illustration 3: The V Model.....	13
Illustration 4: US electrical consumption per capita 1960 - 2007	23
Illustration 5: RemoteLinc Insteon Lamp Control Kit	27
Illustration 6: IRLinc Receiver-IR to Insteon Converter	28
Illustration 7: SmartLinc - Insteon Central Controller	28
Illustration 8: Example of the Vera system GUI31	31
Illustration 9: LCD peep-hole device	32
Illustration 10: P3 Kill A Watt module	34
Illustration 11: Control4 System SR-250 Remote Control35	35
Illustration 12: I/O Linc Insteon doorbell and telephone alert system.....	36
Illustration 13: Insteon 8-Zone Sprinkler Controller37	38
Illustration 14: Insteon ApplianceLinc40	39
Illustration 15: The Dell Streak (Courtesy of Dell Inc.) 49.....	48
Illustration 16: Freescale MC13213 ZigBee communications stack	53
Illustration 17: BeeStack Consumer layer interfaces 55.....	54
Illustration 18: Example BeeStack Consumer Mesh Network	56
Illustration 19: Functional description of the NXP LPC3250.....	64
Illustration 20: Picture of the LPC 3250 OEM Board.....	66
Illustration 21: Embedded Artists LPC3250 Base Board.....	68
Illustration 22: Functional Specifications of the Base Station	69
Illustration 23: Initial Base Station Software Design	70
Illustration 24: Base Station Software Use Case Diagram	72
Illustration 25: New User Registration	73
Illustration 26: Add a New Device	74
Illustration 27: Fulfill Service Request	74
Illustration 28: Multiple Reply to Notification	75
Illustration 29: Figure X - MC13213 block diagram.....	81
Illustration 30: MC13213 Pinout	82
Illustration 31: MC13213 ZigBee transceiver block diagram70	84
Illustration 32: Functional Diagram for a Wireless Module	85
Illustration 33: MCU, Temperature Sensor and Terminal Block Schematics.....	87
Illustration 34: Light Dimmer Schematic.....	88
Illustration 35: ZigBee Transceiver Schematic.....	89
Illustration 36: Android Home Page	93
Illustration 37: Notification Icon	94
Illustration 38: The Notifications Window	94
Illustration 39: Example Notification with Garage Door Control.....	96
Illustration 40: Example Notification with Door Lock Control	96
Illustration 41: Example Notification with Temperature Variables	97
Illustration 42: Device Menu Tree	98

Illustration 43: Program Options Menu Tree	98
Illustration 44: Add Device Menu Tree	99
Illustration 45: Diagram of a Bottom-up Integration Test.....	124

Table Index

Table 1: TPC vs UDP.....	19
Table 2: breaks down electrical power used in the US in 2008 by application19.....	24
Table 3: Hardware specifications for the microcontroller.....	63
Table 4: Functional description of the LPC3250 OEM Board.....	65
Table 5: Hardware specifications for the Base Board.....	67
Table 6: Base Station Event Table	71
Table 7: HTC G1 Specification Table72.....	92
Table 8: Server to Client Notification Types.....	101
Table 9: Server to Client Notification Protocols 1-5.....	102
Table 10: Server to Client Notification Protocols 6-10.....	103
Table 11: Server to Client Protocol Definitions.....	104
Table 12: Client to Server Response Protocols.....	105
Table 13: Client Notification Page Flip List	106
Table 14: Server to Client Notification Levels.....	106
Table 15: Microcontroller tests.....	125
Table 16: LPC3250 OEM Board tests.....	127
Table 17: QVGA Base Board tests.....	128
Table 18: Integration tests - Base Board.....	129
Table 19: Software Tests - part 1.....	130
Table 20: Software Tests - part 2.....	130
Table 21: Software Tests - part 3.....	131
Table 22: Software Tests - part 4.....	131
Table 23: Remote Client Tests.....	132
Table 24: ZigBee Modules Tests part 1.....	133
Table 25: ZigBee Modules Tests part 2.....	134
Table 26: Hardware faults categories.....	135
Table 27: Software faults categories.....	135
Table 28: Team Member Responsibilities.....	137
Table 29: Senior Design 1 Milestone Chart.....	138
Table 30: Senior Design 2 Milestone Chart.....	139
Table 31: Project Budget.....	141

Bibliography

¹ Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

² Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

³ Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

⁴ Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

⁵ Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

⁶ Shari Pfleeger, *Software Engineering Theory and Practice 4th Ed* (Prentice Hall 2009)

⁷ <http://en.wikipedia.org/wiki/Peer-to-peer>

⁸ http://en.wikipedia.org/wiki/Client%E2%80%93server_model

⁹ Andrew Tanenbaum, *Computer Networks 4th Ed* (Prentice Hall 2003)

¹⁰ Andrew Tanenbaum, *Computer Networks 4th Ed* (Prentice Hall 2003)

¹¹ <http://www-01.ibm.com/software/data/db2/express/>

¹² <http://www.microsoft.com/express/database/>

¹³ <http://www.oracle.com/technetwork/database/express-edition/overview/index.html>

¹⁴ http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL

¹⁵ http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL

¹⁶ <http://www.sqlite.org/>

¹⁷ <http://en.wikipedia.org/wiki/Amoled>

¹⁸ http://www.google.com/publicdata?ds=wb-wdi&met=eg_use_elec_kh_pc&idim=country:USA&dl=en&hl=en&q=us+electricity+consumption+trends

¹⁹ http://www.eia.doe.gov/ask/electricity_faqs.asp#electricity_use_home

- ²⁰ http://www.eia.doe.gov/ask/electricity_faqs.asp#electricity_use_home
- ²¹ <http://www.eia.doe.gov/emeu/aer/txt/ptb0810.html>
- ²² http://en.wikipedia.org/wiki/Phantom_voltage
- ²³ <http://www.crestron.com/>
- ²⁴ <http://www.blackanddecker.com/>
- ²⁵ specs taken from <http://www.smarthome.com/2490A1/RemoteLinc-INSTEON-Lamp-Control-Kit-Silver-Remote/p.aspx>
- ²⁶ image copied from <http://www.smarthome.com/remote-lighting-control.html>
- ²⁷ <http://www.smarthome.com/2411R/IRLinc-Receiver-IR-to-INSTEON-Converter/p.aspx>
- ²⁸ <http://www.smarthome.com/2412N/SmartLinc-INSTEON-Central-Controller/p.aspx>
- ²⁹ <http://www.smarthome.com/remote-lighting-control.html>
- ³⁰ Table referenced
from [http://www.adt.com/wps/portal/adt/for_your_home/products_services/security_systems/store?
ru=http://prod.commerce.adt.com/webapp/wcs/stores/servlet/ProductDisplay&storeId=10101&langId=-1&parent_category_rn=10112&productId=11421&catalogId=10101&commerceParams=ru.storeId.langId.URL.categoryId.debug.productId.parent_category_rn.catalogId&categoryId=10112](http://www.adt.com/wps/portal/adt/for_your_home/products_services/security_systems/store?ru=http://prod.commerce.adt.com/webapp/wcs/stores/servlet/ProductDisplay&storeId=10101&langId=-1&parent_category_rn=10112&productId=11421&catalogId=10101&commerceParams=ru.storeId.langId.URL.categoryId.debug.productId.parent_category_rn.catalogId&categoryId=10112)
- ³¹ <http://www.smarthome.com/images/1370side2big.jpg>
- ³² <http://www.smarthome.com/1370/Vera2-Z-Wave-Web-Enabled-Automation-Controller/p.aspx>
- ³³ <http://www.smarthome.com/5041/Brinno-PeepHole-Viewer/p.aspx>
- ³⁴ <http://www.smarthome.com/1626-10/FilterLinc-10-Amp-Plug-In-Noise-Filter/p.aspx>
- ³⁵ http://www.control4.com/products/5/54/control4_system_remote_control_sr_250/
- ³⁶ <http://www.smarthome.com/24950A6/I-O-Linc-INSTEON-Doorbell-and-Telephone-Ring-Alert-Kit/p.aspx>

³⁷ <http://www.smarthome.com/31270/INSTEON-8-Zone-Sprinkler-Controller-Lawn-Irrigation-System/p.aspx>

³⁸

[http://www.petco.com/Shop/petco_ProductList_PC_productlist_Nav_179_N_22+4294956560+30+4294965683.aspx?cm_mmc=GooglePKW-_60-DOG_BOWLS-FEEDERS_AUTO-_-\[auto+pet+feeder\]-_-xxx](http://www.petco.com/Shop/petco_ProductList_PC_productlist_Nav_179_N_22+4294956560+30+4294965683.aspx?cm_mmc=GooglePKW-_60-DOG_BOWLS-FEEDERS_AUTO-_-[auto+pet+feeder]-_-xxx)

³⁹ <http://www.petco.com/product/106284/Petmate-Le-Bistro-Portion-Control-Automatic-Feeder.aspx>

⁴⁰ <http://www.smarthome.com/2456S3/ApplianceLinc-Relay-INSTEON-Plug-in-Appliance-Control-Module-3-pin/p.aspx>

⁴¹ <http://en.wikipedia.org/wiki/ZigBee>

⁴² <http://www.z-wave.com/modules/AboutZ-Wave/>

⁴³ http://mobiledevdesign.com/hardware_news/zigbee_zwave_battle_1008/

⁴⁴ <http://www.newsguide.us/technology/electronics/Kwikset-Expands-Home-Connect-Product-Line-and-Announces-New-Certification-at-CES-2010/>

⁴⁵ <http://www.differencebetween.net/technology/difference-between-zigbee-and-bluetooth/>

⁴⁶ http://www.stg.com/wireless/ZigBee_comp.html

⁴⁷ http://news.cnet.com/From-Dangers-realm-come-Androids-makers/2008-1039_3-6218126.html

⁴⁸ Source: <http://gdgt.com/dell/streak/specs/>

⁴⁹ Source: <http://content.dell.com/us/en/corp/d/corp-comm/image-gallery-tablets.aspx>

⁵⁰ http://en.wikipedia.org/wiki/Ford_Sync

⁵¹ <http://mashable.com/2010/04/20/ford-sync-applink/>

⁵² http://en.wikipedia.org/wiki/MIPS_architecture

⁵³ <http://www.mipsandroid.com/projects/show/mips-android>

⁵⁴ Paragraph copied from Freescale BSCONRM Beestack reference manual page 13

- ⁵⁵ *Freescale BSCONRM Beestack referenc manual page 13*
- ⁵⁶ *Freescale BSCONRM Beestack reference manual page 13*
- ⁵⁷ *Freescale BSCONRM Beestack reference manual page 13*
- ⁵⁸ *Freescale BSCONRM Beestack reference manual pages 11-12*
- ⁵⁹ *Freescale BSCONRM Beestack reference manual*
- ⁶¹ *<http://www.talkandroid.com/android-forums/android-hardware/2-android-minimum-hardware-requirements.html>*
- ⁶² *Table copied from EA website (See permissions)*
- ⁶³ *Table copied from EA website (See permissions)*
- ⁶⁴ *Image copied from NXP LPC3250 product page*
- ⁶⁵ *Table copied from EA website (See permissions)*
- ⁶⁶ *Image copied from EA website (See permissions)*
- ⁶⁷ *Image copied from EA website (See permissions)*
- ⁶⁸ *Table copied from MC13213 datasheet*
- ⁶⁹ *Copied from Freescale MC13213 datasheet*
- ⁷⁰ *Image copied from MC13213 datasheet*
- ⁷¹ *Feature list copied from Freescale MC1321X datasheet*
- ⁷² *Specification copied from <http://www.htc.com/www/product/dream/specification.html>*
- ⁷³ *<http://www.freetutes.com/systemanalysis/sa9-bottom-up-integration.html>*