Close to Home (C2H)

Nicholas “Bailey” Godfrey, Marc Garcia, Daniel “DK” Krummen, and Joshua Early

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — In this project we are creating a home automation system. Our system is meant to make energy savings simple and easy for the user of the system and give users peace of mind by giving them control of their homes from anywhere. The system consists of four parts. The application is the users interface, a database communicates between the user and the house, and the hub communicates with the database and the end home devices all in a seamless fashion.**

***Index Terms* — Cellular phones, Databases, Home automation, Microcontrollers, Radio communication, Smart homes, and Zigbee.**

I. Introduction

The Close to Home project is, at its core, an effort to provide home users a way to not only monitor their house to ensure its security, but also control several aspects of the house to save on their monthly power bills, and to provide them with peace of mind. The Close to Home system is one that will be able to be integrated seamlessly into a user's home, and provide them with all of the features of the system in a non-invasive and non-obstructive way. Having complete control over your home will be simple and easy with the Close to Home system using the Android Application that accompanies the system. Paired with the central hub somewhere close to the home's internet router, and a few modules installed throughout the home to communicate with, a user will be able to be the master of their own domain. Our objectives for Close to Home are essentially to provide a system for the home that saves money on a user's power bill every month, while providing them with security and monitoring features at their fingertips. Ultimately, the system will provide more saving benefits than the cost of using the service, as well as providing at least 40% more power savings than upon a housing system that does not have Close to Home installed. We will be providing evidence of this with a scaled-down version of a home and its power uses.

II. Overview

Our group wanted to have a project that would incorporate a large piece of software with at least two different kinds of hardware and an android application. An automated smart house would allow us to implement multiple kinds of hardware (motors, sensors, etc.), design an underlying software to control it all, and implement the whole thing with an android application. We will never have to be individually responsible for one portion of the project, we all know how to incorporate the parts as a whole, and we all have something unique to gain.

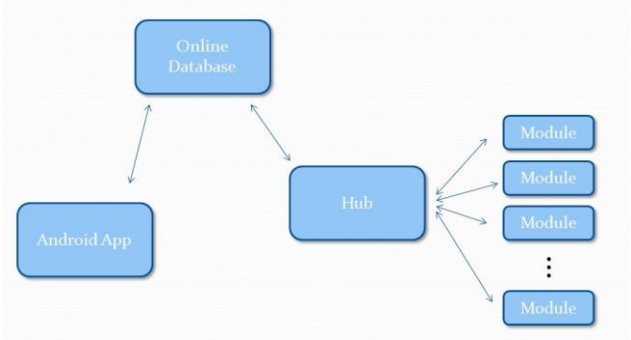


Fig. 1 Close to Home System

We are creating an automated, smart house that will include convenient user control through an Android application. We hope to bring you a better home experience with improved security and energy savings. A central hub will control the smart house, which is still currently in the design process. It will receive commands from both an Android app via a web server, and through an in-house tablet loaded with the app, which can be detached and carried around. The biggest advantage with this design, is being able to fall asleep knowing everything is locked up tight. Each room will be equipped with a break-beam eye system that will monitor the rooms to keep track of which rooms are occupied and which rooms are vacant. From here, user defined settings will determine what, if anything, should be on in the room. This is where we plan to see a majority of our cost savings.

We also know that you are not always at your house, and peace of mind is hard to get when you’re away on vacation, or even gone for the day for work, this is where the phone app comes in. The Android app will give you the same control the in-house control panel would give you, including the ability to lock/unlock doors, monitor your appliances, or even program timers. This will deliver convenient home control to anyone with an Android phone. Various levels of control can be set for each user allowing for a more hierarchical control system.

Imagine the day when you go on vacation and the only thing on is your security system, but you can remotely

access your door lock to let your house sitter in. Never worry about forgetting to lock the front door or shut off the lights when you leave the room, because the automated house will handle that for you. Never worry about your home being hot when you get home from work, you can turn the A/C on just before you head home. The possibilities are as limitless as the sensors and controls we can put on your home. Everyone makes a big deal about checking your a/c, remotely logging into their desktop, or keylessly unlocking their doors, but imagine controlling everything with just one app.

III. The Hub

The chosen developmental board that we decided upon for the hub of our Close to Home system was the Raspberry Pi Model B. This, paired with an Xbee Radio Module interfaced with the Raspberry Pi's GPIO headers allow for the hub of Close to Home to communicate with the various door locks, lights, room sensors, and electronic devices for efficient and timely monitoring and application. Within the following paragraphs, we will explain at length why we chose to utilize the Raspberry Pi as our hub, what it means to use the Raspberry Pi, and a lengthy explanation of all of the different aspects of the Raspberry Pi that made it a good fit for Close to Home.

The Raspberry Pi is a credit-card-sized single-board computer that was initially developed by the Raspberry Pi Foundation in the UK. It was created with the purpose of teaching basic computer science in elementary schools worldwide by providing the teachers with an affordable computer that has access to basic functions and low-level computation. On board the Raspberry Pi is a Broadcom BCM2835 system on a chip which has within it an ARM1176JZF-S 700Mhz processor. Included on this developmental board is a set of 512MB of RAM that provided us with a good amount of system memory for our needs, allowing the Close to Home project to maintain and control several modules simultaneously without falling behind in its scheduling

Raspberry Pi Hardware Specifications	
Processor	700MHz ARM11
RAM	512MB
GPIO	8xGPIO
Floating Point?	Yes
Op. Volt.	5V
UART?	Yes
LAN?	On-Board
Size	3.4” x 2.2”
Weight	45 grams

Table 1: Raspberry Pi Hardware Specifications

The ARM11 processor was perfect for our group's needs, as we needed a developmental board containing a processor that didn't have too much power, in an effort to keep the cost down, but still has the capability to manage the Close to Home network with relative ease. Given the efficiency of ARM processors, and their use of the RISC (Reduced Instruction Set Computing) code, the modest 700Mhz of processing power was ample for our needs. In the event that the processor was becoming overwhelmed with the task at hand, it would have been rather simple to add a small heatsink to the top of the processor, and perform a bit of overclocking to bring the speed up to 1Ghz, but this was not exclusively necessary for the environment that we had set up. The alternative to the RISC architecture is the CISC (Complex Instruction Set Computing) architecture, which is what most modern computers use.

The choice of the processor was also a personal one for many of our group members, as development with ARM processors is becoming more and more prevalent in the job market these days. With the rise of smartphones and other mobile computing, any low power consumption and low instruction set processor is becoming the talk of the town among Silicon Valley. Obviously, if something is becoming popular in modern technology, Computer and Electrical Engineers need to be at the forefront of the field.

Boasting about a previous Senior Design project that involved working with an ARM-based processor looks wonderful on a resume, and we hope that the experience we gain with this project will aid us in our search for work in related fields. Ultimately, we would love to get hired on at a company where we can bring Close to Home to many homes around the world, but that will remain a pipe dream

for now, as we get a working and finished prototype online as a first priority.

The Raspberry Pi's onboard "System on a Chip": The Broadcom BCM2853, came fully equipped with a mini-UART system, which will prove invaluable in our project. Since we will be using the ZigBee trasmission protocol, we will likely need to follow the transmission of data every step of the way, from handshakes to FIFO to flow control and back. Listed online at Broadcom's website is a datasheet for the BCM2853, which wonderfully outlines the mini-UART's operation registers, and has complete instructions for setting the bits high and low for the desired steps of the transmission process. As well, it has a full list of the capabilities of the on-board mini-UART system, and what it is lacking, if anything.

The GPIO headers on-board the Raspberry Pi provided us with nearly unlimited choices as to what kind of external hardware we could add to the developmental board, in order to achieve the Close to Home hub that we envisioned. The layout of the analog and digital headers was intuitive and simple to understand, with information as to what each pin corresponded to readily available on the internet.

Attached to the GPIO headers is a small breakout board that allows access to the GPIO pins in a more convenient format, which was necessary to make the design of the ZigBee transmission radio as clean as possible. Since we didn't want wires running all over the place, we decided upon the Slice of Pi breakout board since it included a very easy to configure interface for the ZigBee Radio Module that it can just simply plug into and get interfacing on the Raspberry Pi's serial bus.

Using the Raspberry Pi's serial port to communicate over ZigBee turned out relatively simple once we discovered that there was a wealth of Python scripts and code examples that helped us learn more about low-level communication with the Raspberry Pi. Initially we intended to use Java for the actual coding backbone on the Close to Home Hub, but Python proved more efficient and more easily interfaced with the serial communication that we would be using to make the Hub talk to the other Modules, such as the lock and light modules. As well, Python had easy to import libraries for the PHP scripts that would be run remotely on the Database Server. This allowed the Hub to communicate rather effortlessly between the in-wall modules and the online database, and consequently, the Android application.

IV. Electronic Modules

The electronic modules are what we call the individual components that will receive a signal from the central hub. These are each individually designed to handle the different tasks for each goal of the project. While each module is designed separately, there is one major factor that they all share, the microcontroller. For this project we chose the MSP430G2553. The MSP430 series is an touted as an "Ultra Low-Powered Microcontroller", and as such draws very little power when in standby mode. The following are the parametric for the MSP430:

Voltage Range	1.8V to 3.6V
Power Consumption:	
Active:	250 uA
Standby	.7 uA
Off (RAM Retention)	.1 uA
Memory:	
Flash	8KB
RAM	256KB

Table 2: MSP430F2121 Parametrics

Another feature that each module shares is the wireless communication. We chose to use the Zigbee protocol for our wireless communication, and will be using the XBee to implement this. We chose the Zigbee protocol as it is a low power protocol, and will allow us to save the appropriate amount of energy costs that we are looking to obtain. The Xbee will be used, as it is easy to implement based on the memory constraints we have with the MSP430, because it uses the standard UART communication protocol.

Aside from the microcontroller and wireless communication, each module has unique features that separate them from one another.

A. Lock Module

The lock module will be the module that attaches to the door and locks and unlocks the door. We are not looking to redesign a standard mechanical lock, so we will be modifying a standard lock for our personal uses. The first thing to talk about here is the servo that we will be using. The servo is the FS105B, and has variable torque outputs based on the input voltage. We want to be able that the servo will be able to turn any lock it is attached to. The rating for the FS105B will be 49 in*grams at 6 volts, which will be more than enough torque to turn the standard home locks. We also believe that it is very important that we are able to physically override the lock. As such, the

keyhole and thumb twist will be available. This of course makes this system more sensitive to state changes, so we cannot simply keep track of the state changes through the MCU alone. For each of these modules we will be using some sort of state detection device. For the lock module, we will be using a simple micro switch that is depressed when the door is unlocked and vice versa when the door is locked.

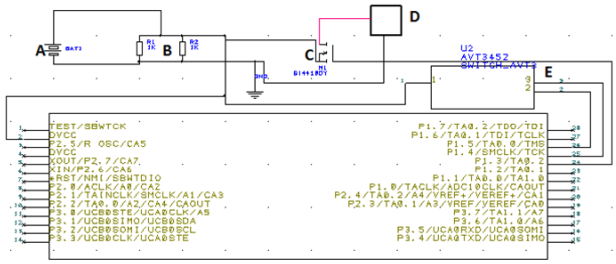


Fig. 2 Lock Module Schematic

B. 120-Volt Module

The 120-Volt module is what we call the module that we will plug small household appliances such as lamps, desk fans, entertainment systems, and any other electronic that is designed to work with the basic wall outlet in the United States. The module will have a male and female port for a standard 3-wire port, and will accept two or three pronged plugs. The main goal of this module is to accurately determine if the appliance that is plugged in is on or off, and be able to adjust the state of the appliance as desired by the user. In order to drive the appliance on or off, we will be using a 3V relay, which we drive using a transistor who will be switched on by a pin from the MSP430. The MSP430 can be powered with a voltage between 1.8 and 3.3 volts, which will be enough to both power the MCU and drive the relay. The ability to tell whether the device is on or off, comes from a resistor of known value plugged into the MSP430's analog to digital convertor. Essentially, the analog to digital convertor reads in a voltage, and compares it to a reference voltage on the chip, giving it a value. If we know the value of the resistor, and the voltage across the resistor, we can easily tell if there is current being drawn across the connection, which would indicate that the appliance is on or off. The power supply for this module will simply be the wall power that is drawn by the board when it is plugged in, rectified, and dropped to a safe voltage for our MSP430.

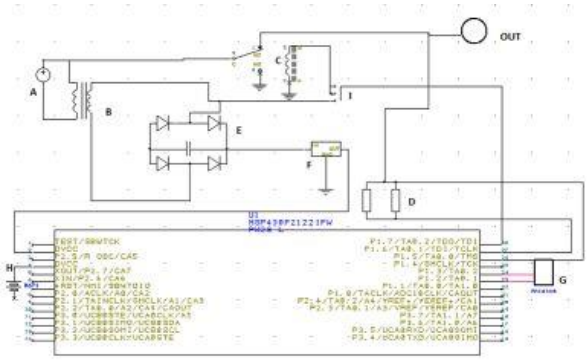


Fig. 3 120-Volt Module Schematic

C. Wall Module

The wall module is what we call the module that we will be using to turn any overhead lighting and ceiling fans on and off. This module is essentially identical to the 120V module, except that there will obviously be no female or male ports. We will still be using the same relay, and the same state detection through the analog and digital converter onboard the MSP430. Since a wall switch works like a typical switch, we can easily tie the relay into this, and disrupt the switch if so desired. We will ideally make simply plug and play type switch that can be inserted into the appropriate frame and socket.

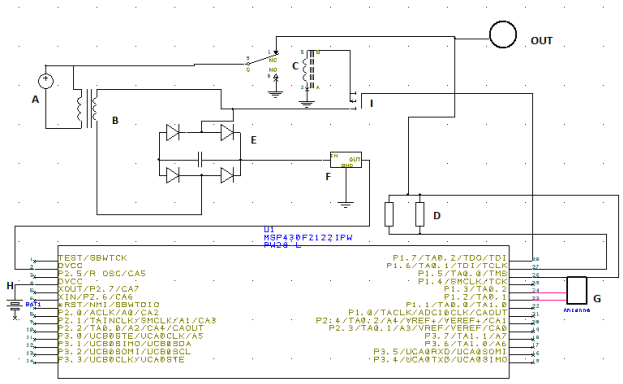


Fig. 4 Wall Module Schematic

D. Occupancy Module

Finally, we have the occupancy module. This will be the module that will be placed at the doorway to any of the rooms that are of interest to our system. For the purpose of our presentation they will be visible red lasers for visual purposes, but can be any variety of lasers to send or receive signals. We have designed this module to run off of 2 AA batteries as we are trying to keep the form factor of this device small. We will have the laser beam set across the doorframe from the receiver. The receiver being

a photocell, whose resistance drops dramatically when it is hit with the laser. Because a single laser beam will not be enough to be able to determine if someone is entering or leaving a room, we have decided to use a double beam system. The idea behind this system is that depending on which beam is broken, we can decide which direction a person is going. If someone is entering, it simply increments a counter and reports that the room is occupied, if someone is leaving, the counter is decremented, and when the counter reaches zero, we can safely report that the room is vacant. This information will be used to control lights, appliances, fans, and any other electronic in the room, so that energy is not wasted if the room is vacant.

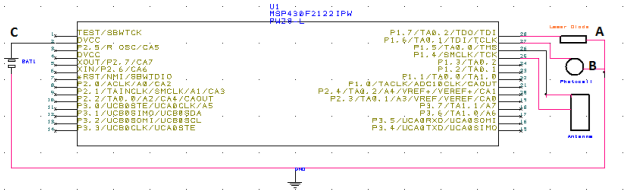


Fig. 5 Occupancy Module Schematic

V. ANDROID APPLICATION

In the design of the application, our main goal is for simplicity and ease of use. The entire purpose for the project is to make the users interaction with their home easy and natural. For this to be possible we had to take into account the tasks our application will be taking care of around the users home. Those tasks include things like turning on or off a light by the flip of a switch. This is already a relatively easy task, therefore we had to figure out how our application will be more beneficial than doing the task itself. This means that our application must be easy to navigate with as little navigation to perform the desired task. Furthermore, the application must be lightweight in performance. If it takes longer for the application to load than it takes the user to accomplish the task manually, our application becomes less valuable to the user. In fig. 6 we see how we tried to streamline the application to be easy to read and navigate as the home screen is very basic, uncluttered, and simple.

Fig. 6 This is a sample of what the home screen of the application looks like after a user is logged in.

A list of tasks the Android application will perform include viewing room occupancy, switching lights off and on, locking and unlocking doors, and switching off and on appliances that run on 120V outlets.

Our target platform for he Android application is Android version 4.0 Ice Cream Sandwich Which is a Java based platform with a powerful public API. It is developed in the Eclipse IDE that allows powerful debugging and testing tools that seamlessly integrates with the Android software development kit.

We designed the Close to Home application to be highly reusable as it is considered to be a real time application that the user can access anytime anywhere. Unfortunately there are limitations that can cause our application to be not as reusable as we would like it to be but are justifiable tradeoffs to the overall design. Our application will only run on devices running operating system running 4.x versions of Android and higher. Also, the application will only run with internet connection. We concluded that 4.x and higher versions were the current market standard that give us all the tools we needed to perform all the tasks we need for the system. Internet connection is necessary for communication with the MySQL database as it is the hand-shaker between the users application and the rest of the system.

As far as portability, the current application is easily downloaded to any Android device running the supported version of the Android OS. Our vision is to have a fully modular and dynamic application that will adapt to any home fitted with our system. The current prototype is a little more statically designed requiring some initial installation and set up in the code itself.

A. Android to Database Communication

Android phone to database connection is accomplished using an HTTP protocol that call php scripts. These scripts perform basic CRUD (Create, Read, Update, Delete) operations on the MySQL database through HTTP GET and POST methods. The Android Java code uses JSON responses to parse data that the Android Java code can use to display the data to the user on their Android device.

B. High Level Design

In figure fig. 7 we see how the Android application interacts with the rest of the system. The user logs into the system which is verified on the database. Once the user is logged in, they have access to the homes device functions such as lights and locks. Any updates to the devices are sent to the database where the main hub will see them and relay that data to the devices themselves.

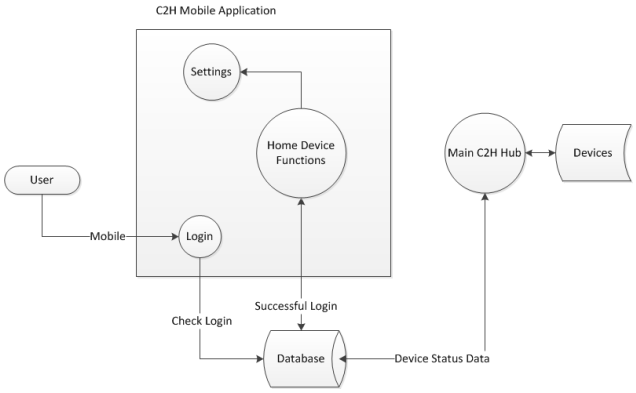


Fig. 7 This is the high level design of the Close to Home (C2H) mobile application's interaction with the rest of the system

C. Low Level design

The application consists of two file types. The first file type is the Java files. They run the program on the device and manage all HTTP request. There is also a JSON parsing class that parses all the data received from the database for the application to use. All parsed data is displayed using xml files that format what is shown to the user.

VI. Server Implementation

Our project relies on the Android application being able to communicate with our central hub. There are many options we could have chosen to bridge this communications, but we considered the future use of multiple Android applications concurrently and decided to go with an approach that would relay communications in a timely manner and utilize one synchronized database.

Currently our web-server is being run off of a laptop and being broadcast locally to the Android application and central hub. We have not implemented any form of encryption or other security measures since this is simply a prototype. We can move everything over to an actual http server and implement appropriate encryption once we leave the prototyping phase.

We decided to utilize MySQL for our database implementation. This allows our central hub and Android application(s) to perform tasks simultaneously without interruption or running the risk of becoming out of sync. Our database design is very simple. We are utilizing two different tables: one to keep track of our modules' status and one to relay commands from the Android application(s) to our central hub. Each module represents a row in the status table and the command table. This allows our central hub to keep track of everything individually and our Android application(s) to keep track of everything at once.

In order to cut down on the number of individual programs needed for server communications, we chose to utilize PHP to give access to our database to the Android application(s) and central hub. PHP allows us to have multiple live connections with our database and allows us to use the same set of programs for our Android application and central hub. This cut down drastically on our time for implementation and makes operations more efficient.

Our PHP programs are designed to give the ability to query and update the database to our Android application and central hub. There are 11 PHP programs designed for the status table and 15 PHP programs designed for the commands table. Adding more peripherals to our project will require the addition of more PHP programs. However, once the project leaves prototyping, many of the PHP programs will be combined in order to cut down on unnecessary server communications.

The 11 PHP programs for the status table are made up of six query programs and five update programs. The five update programs are mainly there to allow the central hub to change the statuses of the hardware peripherals. Five of the six query programs are mainly there to allow the Android application to check the status of any one of the hardware peripherals. The sixth query program is there to allow the central hub and Android application to check the status of all of the hardware peripherals simultaneously.

The 15 PHP programs for the commands table are made up of ten update programs and five query programs. Five of the update programs are there to allow the Android application to relay a command to the central hub by

changing a status on the database. The five query programs are there for the central hub to be able to see if any of the statuses have been changed by the Android application. The other five update programs are there for the central hub to change the status back after performing the appropriate command.

The overall functionality of our project are as follows. Whenever the central hub detects a change in status for one of the hardware peripherals, it executes the appropriate PHP program to change the status listed in the database. This means that whenever the Android application executes one of the PHP programs to query one of the hardware peripherals status, that the status it is receiving is live. Whenever the Android application wants to change the status of one of the hardware peripherals, it executes the appropriate PHP program to change the status listed in the database. Whenever the central hub detects this change in status, it changes the appropriate design peripheral and then calls on the appropriate PHP program to change the status back.

VII. Testing

The testing will be done as a group. This will be done to eliminate the possibility for error and to ensure that one person is not responsible for a bulk of the testing. The testing of individual components will be divided up evenly. There will also be someone assigned to check the results of each testing procedure. This is to ensure that valid test results are admitted in order to guarantee that each individual component works. Otherwise when we combine all of the individual components, we would not know what the error was. This method of testing allows us to guarantee that any errors in combining the individual components are not from faulty components, but instead from our faulty arrangement of the components.

The testing of the complete system will require a more comprehensive, rigorous test routine that each of us must participate in. At this stage all of our individual efforts will be combined into one complete system. We will each be responsible for testing the aspect of the project that we designed, but it will involve all of us testing our various portions at the same time. This will allow us to guarantee that the system works flawlessly as a whole and does not break under any use cases.

The testing of the complete system will entail: checking the accuracy of hardware status on the smart hub and the Android application, checking the ability to issue a command to any of the hardware components from the Android Application, checking the ability for the smart hub to automate itself, and checking the ability of the system to handle concurrent users.

Bailey Godfrey and Marc Garcia will be tasked with going through all of the use case scenarios for the Android application separately on different instances of the Android application. They will also be responsible for setting off multiple sensors. This will guarantee that the system can handle multiple users checking hardware statuses and issuing commands at once while verifying that the home automation aspects work too. In order to verify that everything happens in a timely manner, Joshua Early will be tasked with monitoring the MySQL server and the Apache web server traffic. Checking the web connected elements will allow us to verify that the Android applications are in fact receiving real time status updates and executing their commands in real time. Daniel Krummen will be responsible for monitoring activity on the smart hub. Any activity on the smart hub must represent movements or commands made by Bailey or Marc and must be passed on to the web connected elements, MySQL and the Apache web server.

VIII. Conclusion

In summary, this project was a wonderful experience for all team members involved. We quickly learned the importance of every week as things got closer and closer to our deadlines. It truly showed that you get out what you put in, and the Close to Home project is 100% composed of our group efforts and determination. We hope that in the future we can list our Senior Design project on all of our individual résumés. It will hopefully help us springboard our career into the field of either integrated circuits, software engineering, or microprocessor engineering. We are very thankful for our professor, Dr. Richie for giving us this opportunity to show our stuff in a group setting in order to create a solid foundation for our future. We also would like to thank Duke Energy for providing us with the funding that we needed to make our dream of the C2H system a reality.

Acknowledgement

The authors wish to thank Dr. Samuel Richie for providing us with this opportunity to work in such a tightly knit group setting, and for ensuring that we were on task throughout the entire design and developmental process. We would also like to thank Duke Energy for fully sponsoring our project and letting us make the Close to Home System a reality despite any cost roadblocks that

may have arisen. Thank you as well, to the members of our Senior Design Review Committee for taking the time out of their busy schedules to watch our presentation and for reviewing this document to the fullest extent that they did.

REFERENCES

[1] Wei-Meng Lee, "Beginning Android Application Development", John Wiley & Sons *2011*

[2] Selecting XBee Radios and Supporting Software Tools (http://jeffskinnerbox.wordpress.com/2012/12/22/selecting-xbee-radios-and-supporting-softwaretools/)

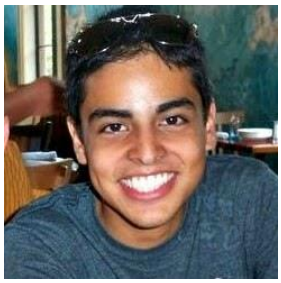
[3] Freeing UART on the Pi (http://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi/freeing-uart-on-the-pi)

[4] Bouvy, Michael, "Raspberry Pi + XBee: UART / Serial howto" (http://michael.bouvy.net/blog/en/2013/04/02/raspberry-pi-xbee-uart-serial-howto/)

Nicholas Godfrey is currently pursuing a bachelors in Electrical Engineering at the University of Central Florida. He is a devoted member of the Theta Tau fraternity, and is highly skilled with electrical circuits. Given his unique expertise with circuits, at least among a group of Computer Engineers, he chose to take on the role of developing the in-house wall modules. He designed simple circuits that would receive power from the house's grid, convert it accordingly, and then use them on a small board equipped with an MSP430 Microprocessor. From there, the data pertaining to the observed module would be transmitted via ZigBee to the central Hub of communications for the Close to Home system. Since Nicholas had to develop the circuit from scratch, his expertise in step-down circuits, and breadboard prototyping was a brilliant asset to the Senior Design team, and we could not have done it without him.

Joshua Early is currently pursuing a bachelors in Computer Engineering at the University of Central Florida. He is a hard worker with an affinity for mobile and object-oriented software development, so he decided to devote his work-hours, alongside Marc Garcia, to the development of the Android Application that would be the control system for the end-user in the Close to Home system. Getting the Android Application to interface directly with the Close to Home Hub, and then to interface with a Server Application of Marc's design, was no easy task. Ensuring that all devices were speaking the same language, and working in unison was something that took hours of labored testing as well as heavy trial and error. However, the final product will yield an Android Application that will be convenient to use, and will provide full control and vision over your home while you're in it, or if you're away.

Daniel Krummen is pursuing a bachelors in Computer Engineering at the University of Central Florida. Daniel expressed an interest early on into the field of microcontrollers and lower-level programming, so he decided to work on the central Hub of the Close to Home system. These system resources will prove invaluable when computing the various kinds of calculations necessary to schedule the Close to Home mainframe. The Close to Home system will feature a fully customizable method of automating your house's appliances to ensure that there are rarely lights on in an unoccupied room, or a television running with no one watching. Daniel plans to incorporate either built-in profiles for recommended house monitoring settings, or providing the client with a way to customize their own configuration via the Android Application.



Marc Garcia is pursuing a bachelors in Computer Engineering at the University of Central Florida. Given Marc's expertise with Object-Oriented and higher level programming, he and Josh worked together to create the most aesthetically appealing Android Application for the end user, as well as making sure that, along the way, the application always had the goal of being user-friendly in mind. Marc had a great eye for design, and was the specialist when any of the project needed a professional polish on it. For instance, the GUI of the Android Application was the product of Marc's design, and making sure that it ran smoothly on nearly all Android platforms was another of Marc's goals.