Kreative Outlets

Team Members:

Travis Crist - Computer Engineer

Jason Peressini - Computer Engineer

Steve Clark - Electrical Engineer

Advising Professor:

Hugh Smith

California Polytechnic State University

Senior Project

March 19, 2012

Abstract

With the widespread use of smart phones and the advancement of WiFi technologies, people now have the Internet at their fingertips at all times. Home automation and control has the possibility of becoming the next big technology that can utilize the abundant availability of the Internet. This paper documents the findings and results of a project aimed at developing the infrastructure needed for a home automation system that is accessible anywhere. We implemented a system that utilized a web enabled ethernet connection and RF wireless technology to control AC outlets and provide the user with feedback information about the status of their home. This system included the development of a microprocessor based control, sensor, and base nodes and a web based interface.

Contents

List of Figures

1 Acknowledgements

Dr. Hugh Smith, Cal Poly
Dr. John Oliver, Cal Poly
AVRfreaks.net
Mark Peressini, Science Applications International Corporation

2 Introduction

The goal of the KreativeOutlets senior project was to create a basic home automation and control system. Home automation has the potential to be very useful and thanks to the help of smart phones, control can be very convenient for many homeowners. The system that we designed consists of a web-based user interface through which the user can control an AC outlet's power and receive sensor data from our wireless nodes placed throughout their home.

Our system consisted of four functional modules: Base Station, Control Node, Sensor Node and Webserver. The Base Station, Control Node and Sensor Node communicated with each other via a RF Zigbee network. In order to create a Zigbee network we utilized Atmel Zigbit modules mounted on PCBs that we designed. The Base Station was connected to the Webserver over a TCP connection. The high level data flow can be seen in **Figure 1**. What follows below will be a summary of the responsibilities and functionality of each module.

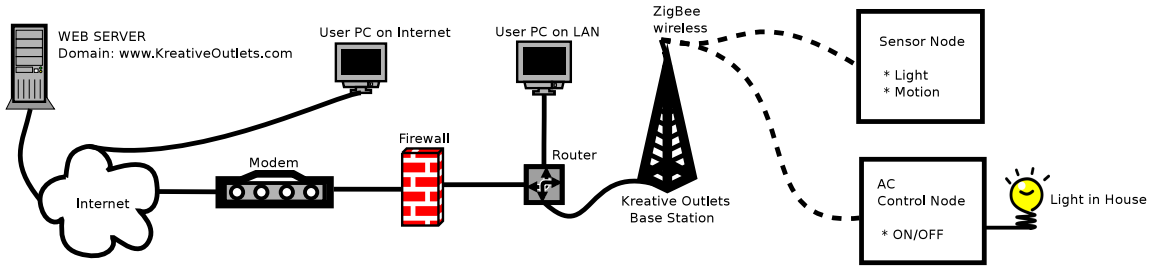

Figure 1: High Level Diagram

The Control Node, as seen in **Figure 2** below, consists of an Atmel Zigbit module connected to a 120 VAC relay. The node is plugged into an AC outlet which it draws power from. It provides a relay controlled AC outlet plug for users to plug appliances in. The Zigbit module allows the Control Node to wirelessly communicate with the Base Station using an RF connection. It periodically sends status updates to the Base Station which indicate the current state of the relay (on or off). When the Control Node receives a command from the Base Station, it responds accordingly by turning on or off the relay.

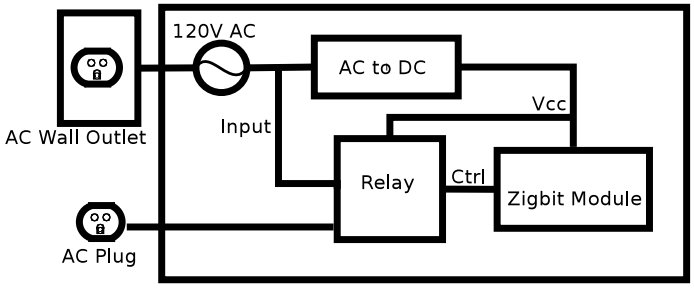

Figure 2: Control Node

The Sensor Node, as seen in **Figure 3**, utilizes an Atmel Zigbit, a motion sensor and a light intensity sensor. The node is battery powered and is able to enter a low power state (sleep) in order to save power. The Zigbit module allows the Sensor Node to communicate with the Base Station using RF. The node wakes up periodically and sends the data collected from the sensors to the Base Station. After the data is sent, the Sensor Node goes back into sleep mode.

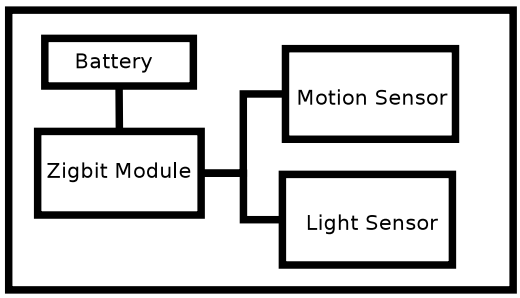

Figure 3: Sensor Node

The Base Station, as seen in **Figure 4**, consists of an Atmel Zigbit module and an Ethernet module. The device is powered by an AC wall outlet and requires an Ethernet connection to an internet enabled network device. The Base Station is responsible for creating and maintaining the Zigbee RF network and keeps track of the nodes that are connected to it. The Base Station is also responsible for establishing a TCP connection with the Webserver as well as maintaining it by sending a blank TCP keepalive packet. If the TCP connection is ever dropped, the Base Station immediately trys to re-establish the connection. The device acts as a middle-man and accepts data from Sensor Nodes and Control Nodes and forwards the data to the Webserver and vice versa.

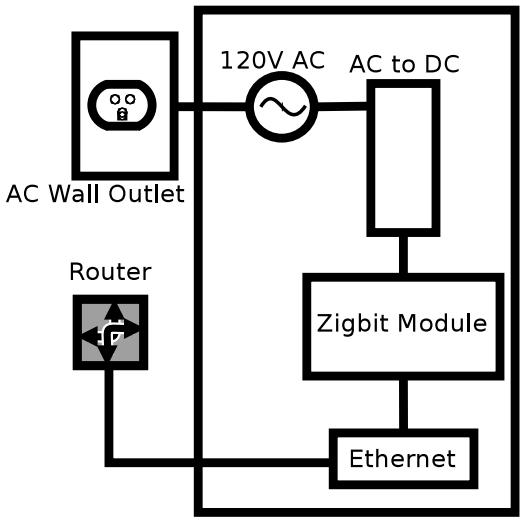

Figure 4: Base Station

The final component is the Webserver. For the website we chose the domain name KreativeOutlets.com. To host the server we used an Amazon EC2 server running Ubuntu 10.04. Amazon EC2 is a true virtual computing environment, it allows one to use web service interfaces to launch virtual machine instances with a variety of operating systems, load them with our application environment, manage our networks access permissions, and run our virtual machine images.[22] The website has a user management system so that the user can log in to a unique homepage.

Once logged in the user will be able to add Base Stations to their account. After a Base Station is added to the account the status page will automatically populate with the correct Sensor and Control Nodes that are attached to the Base Station. The website shows the status of the Control and Sensor Nodes and allows users to turn the Control node On and Off. It also allows users to setup notifications which email the user when an action occurs based on a sensor.

Along with the notifications the Webserver implements an automation system which allows the user to perform an action on the Control Node based on the motion or light sensor status from the Sensor Node. This allows the end user to implement features like turning on a light when there is motion or turning off a light when there is no motion. To implement the website HTML5, CSS3, PHP, Javascript, MySQL, Twitter BootStrap, and UserCake were used.

3 Project Definition

3.1 Features

- Easy to use web interface.
- Turn On/Off any device that uses an AC outlet.
- Know if your home has been invaded.
- Provide this functionality from any webenabled device.

3.2 Functionality

- Turn On/Off lights from any web accessible location.
- Check the status of the lights in your home.
- Check if motion has occurred in your home.
- Home Automation Scheduling.

3.3 Specifications

1. Base Station sends status update packet using TCP connection to the website continuously updates the status of the house.
2. Website can send commands using the TCP connection to the Base Station.
3. Sensor Nodes are wireless and battery powered, they will communicate with the Base Station via RF using Zigbee protocol with BitCloud as our software stack.
4. Control Nodes are wireless and AC powered, they will communicate with the Base Station via RF using Zigbee protocol with BitCloud as our networking stack.
5. Website will have a GUI interface to control the devices and create home automation schedules.
6. Website accessible by any browser capable of HTML5, CSS3, PHP5.3, and Javascript.

3.4 Requirements

1. Base Station that can communicate with the end user via the world wide web. It can receive commands and provide the user with sensor information.
2. Sensor Nodes provide data to the Base Station about the percent of light in the room, and the time of the last motion in the room.
3. Control Nodes control electrical outlets throughout the home, triggered by requests from the Base Station.
4. Sensor Nodes report sensor status and are physically portable.
5. Website that allows users to login and manage their Base Station from anywhere with web access.

4 Design Documents

4.1 Use Cases

4.1.1 Case 1

You have a friend over and you want to set the mood. The problem is, you are quite comfortable on the couch and the light switch is on the other side of the room. Luckily, your smartphone is within reach. Using the KreativeOutlets web interface you access the status page and turn off the light.

4.1.2 Case 2

You're in a hurry one morning for work. You rush out the door forgetting that you left the toaster oven on. halfway to work when you realize this and you enter panic mode in fear that your house might burn down. What do you do? You access the KreativeOutlets web interface via your smartphone and shut off the toaster to save your home.

4.1.3 Case 3

You just left town for the week and can't remember if you left the lights on in your house. This will cost a fortune, luckily you have the KreativeOutlets web interface to check which rooms lights are on. Using the interface you turn off the lights in your home, leaving on only the ones you need.

4.1.4 Case 4

You are living with a few college roommates and you are on a tight budget. For some reason, one of your roommates always leaves his lights on when he leaves the house. With a few simple configurations on the KreativeOutlets web interface you can automate his lights to turn off when he is not there. Combining the motion sensor data and the control nodes statuses, the Base Station can shut these lights off for him.

4.1.5 Case 5

For some reason your cable modem needs to be reset almost once a day. Unfortunately its in a very hard to reach place. To remedy this situation you decided to plug the AC outlet of the cable modem into the Control Node. Now in order to reset your internet connection all you need to do is access the status page to power cycle the cable modem.

4.2 Gantt Chart

The **Figure 5** is the Gantt Chart of our project. It shows the progression of the project from the start of implementation to completion.

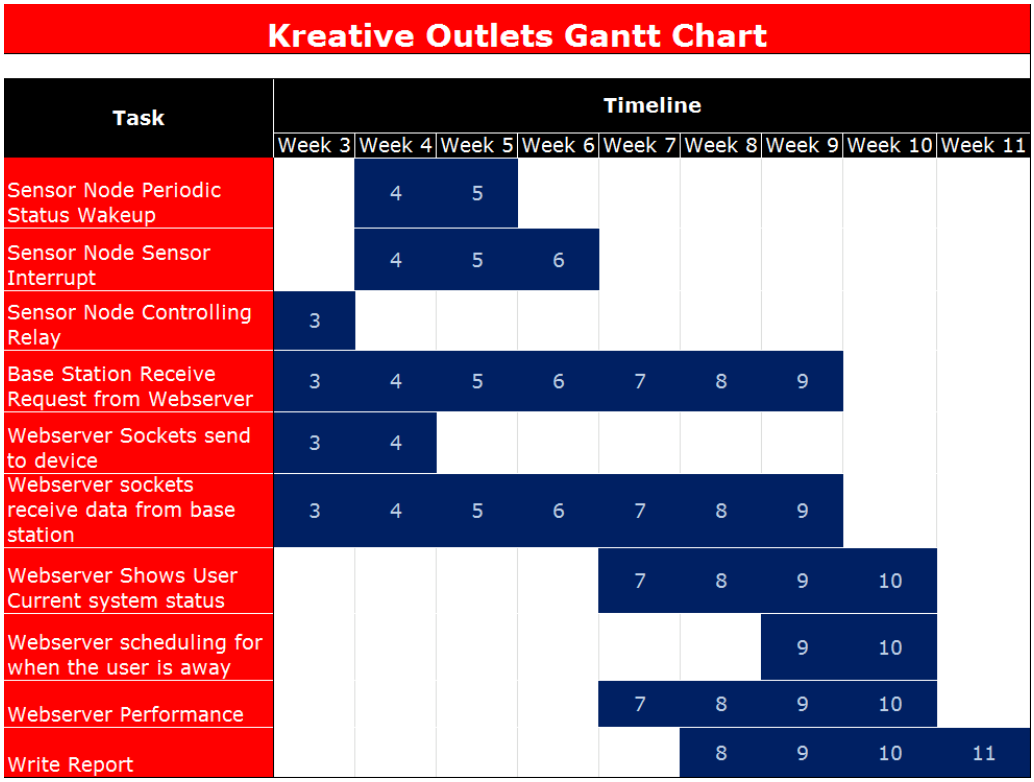

Figure 5: Gantt Chart

5 Implementation

5.1 Considerations

This section details the hardware and software decisions we made and why we made them. A narrative then follows describing the prototyping and testing process. The narrative includes major roadblocks in the implementation process.

5.1.1 Hardware

RF Interface - We decided to use Atmels Zigbit modules to handle our RF data transmissions. The Zigbit module consists of and ATMega1281 microcontroller paired with a

2.4GHz AT86RF230 transceiver and comes packaged with dual surface mount antennas as shown in **Figure 6**. The ATmega1281 microcontroller was programmed with our device specific application code. The transceiver contains all required features to create a Zigbee compliant network. Atmel also provides the Bitcloud SDK which is a software stack that contains all the necessary software layers to configure and manage a Zigbee compliant network. The Bitcloud stack utilizes a small event driven operating system featuring a task handler which makes it simple to add application level processes. The task handler also made it possible for us to not use any additional external microcontrollers. The Zigbit module boasts ultra low power consumption along with a good transmission range.

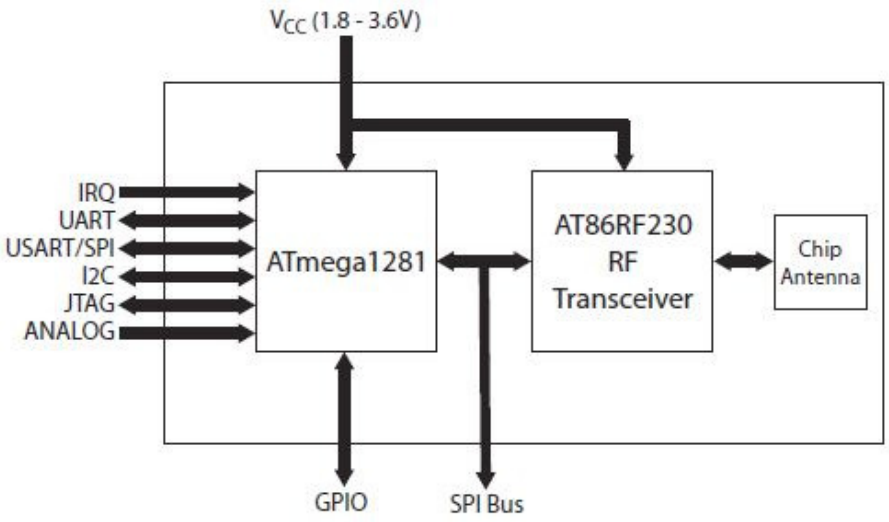

Figure 6: Zigbit Block Diagram

Ethernet Interface - Our implementation of this system required that we connect to an external Webserver in order to send and receive data. We decided to use the TCP network protocol because it features a reliable connection. The module we decided to use was the Lantronix XPort module. This module included a TCP/IP stack. With proper configuration, this module acted as a transparent serial-to-tcp tunnel. We communicated with the XPort using our Zigbits USART pins and were able to write simple commands to connect to our Webserver and establish a TCP connection. Once connected, sending/receiving data was achieved by simply writing/reading data on the USART pins. So to send data to the Webserver we simply wrote data to our USART Tx pin and to read incoming packets from our Webserver we would just read the data on the USART Rx pin. The pin connections are shown in the **Figure 7**.

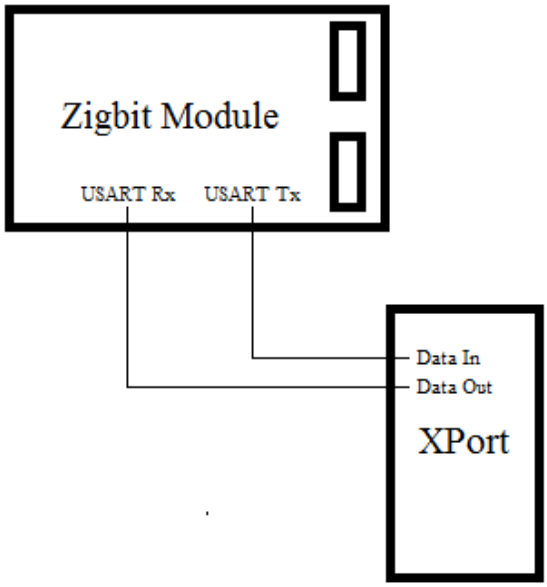

Figure 7: Xport Pin Connection [3]

Relay - We required a 120VAC relay that could be driven by our microcontrollers 3.3V GPIO interface. Among the different types of relays, we found ones catering towards our application to control them via microcontroller. Opto-isolation circuitry is a type of buffering circuitry that protects the microcontroller from the high voltage 120VAC. We determined that purchasing a relay board with this feature was a smart decision to prevent damage to the Control Node.

The two options were narrowed down to the RBX-1 3-Channel Opto-isolated Relay Board by Probotix and the ET-OPTO RELAY4 by ETT. We decided to use the ET-OPTO RELAY4 for two reasons. The RBX-1 requires a 5V control signal which would need additional external circuitry to interface with the microcontrollers 3.3V GPIO output whereas the ET-OPTO RELAY4 only requires as low as a 2V control signal.

5.1.2 Software

Embedded Firmware - Our embedded code was written using the Atmel AVRStudio 5 IDE. The IDE supported compiling, building our HEX firmware images, and programming our HEX files to the Zigbit Module. Our code linked the Bitcloud SDK libraries that were provided by Atmel. The Bitcloud SDK provides an embedded software stack which allowed

us to create user applications that configured and controlled Zigbee standard compliant networks. Bitcloud contains a comprehensive set of APIs used to control the various software layers shown in **Figure 8**.

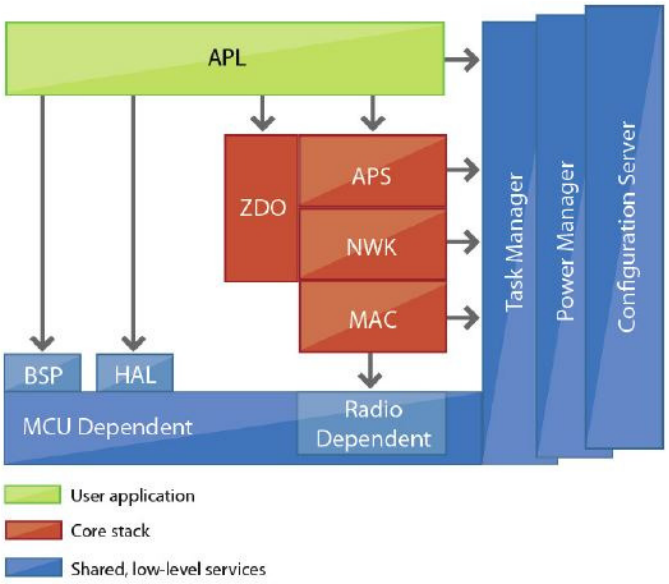

Figure 8: BitCloud Software Layers [2]

5.2 Implementation Process

5.2.1 Webserver

We used Amazon EC2 instance for the Webserver since Amazons servers are reliable and its a Virtual Machine that we have complete control of so we can open any ports that we would like for creating and establishing the TCP connection. The following were installed on the Webserver for the creation of the website:

OS - Ubuntu 10.04, the latest Long Term Release for stablity.[14]

HTML5 & CSS3 - Used to display web pages on the internet.

PHP5.3 - Server side code to generate HTML5 pages based on a collected data.[11]

PHPMyAdmin - Used to manage the MySQL database.[12]

Postfix - Used to send activation and notification emails via PHP commands.[15]

Tiwtter Bootstrap - A CSS/Javascript framework, used throughout the site. It uses a floating layout which allows the website to react to the end users browsing device

so if they use a mobile device the entire site is resized and visible. Bootstrap also contains javascript buttons and tabs which were used in the site design.[7]

UserCake - An open source PHP user management system, allowed us to focus on the actual task of creating the website without needing to create our own user management system.[16]

5.2.2 Hardware

We received the Atmel Zigbit surface-mount chips and designed a simple PCB breakout board using a PCP layout application called Eagle. We had the boards fabricated and the chips mounted as a favor by Mark Peressini. We experienced issues loading the firmware onto the Zigbit with the ISP Programmer, so we borrowed an Atmel STK600 from Dr. Oliver in order to load the firmware using JTAG. Further down the line we experienced issues with the copper traces lifting off the PCB. We then designed and ordered new, more complex PCBs from Sunstone Circuits. The new PCBs contained silkscreen and solder mask as well as thicker traces and proved to be much more robust. For the new design, we grouped together the programming pins and power pins to make it easier to load the firmware using the STK600.

We received the ET-OPTO RELAY4 and its accompanying documentation which was quite confusing. We attempted to drive the relay control with a DC power supply but the relay would not switch when we applied voltage. We then resorted to the manual which contained a schematic. We decided to try bypassing the input BJT **Figure 9**. because we did not think the relay was switching on. This quick fix caused the relay to function correctly. We hardwired a GPIO pin from the Controller Nodes microcontroller to the input control of the relay.

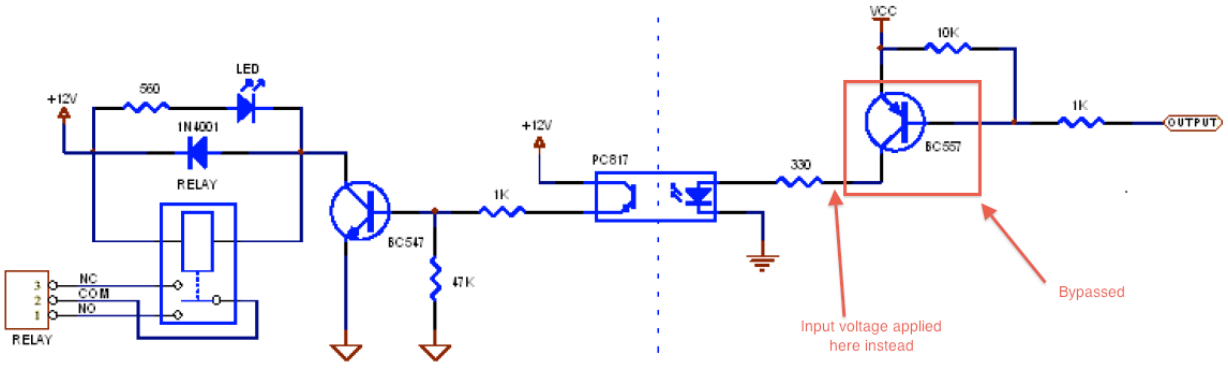

Figure 9: Input BJT as shown in the ET-OPTO RELAY4 Manual [6]

For the majority of the development process, we utilized power from the STK600 and lithium ion batteries to power the Zigbit and the XPort. Once the prototypes for the nodes were finalized, it was time to utilize our own power schemes. The Base Station requires a regulated DC supply of 3.3V to power the Zigbit module and the XPort as shown in **Figure 10**.

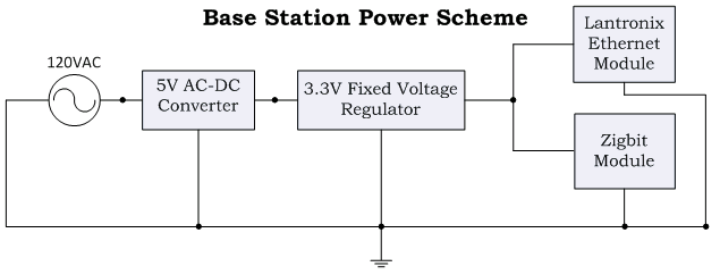

Figure 10: Base Station Power Scheme

Initially we had a 12VDC supply to drop down to 3.3V via an LM317 adjustable voltage regulator. This proved to be a very inefficient solution due to all the power lost in heat through the regulator and we discovered that our regulator was overheating. The overheating caused a significant current drop and prevented our XPort from functioning. We then acquired a 5VDC supply and utilized a LD33V fixed 3.3V regulator with a heatsink attached. This proved to be a cool and effective solution to our problem. The Sensor Node requires 3.3V for the motion sensor, light intensity sensor, and Zigbit module as shown in **Figure 11**. To power the Sensor Node we used a 3.6V battery in order to make it an entirely wireless, mobile unit.

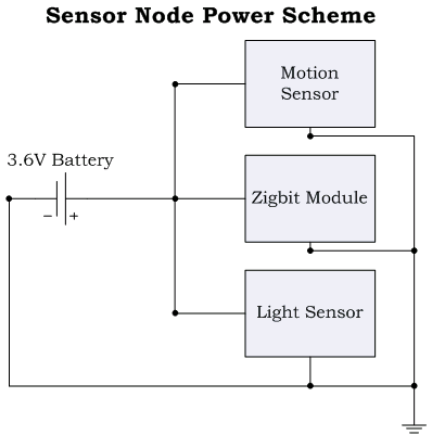

Figure 11: Sensor Node Power Scheme

The Control Node requires 12V for the relay and 3.3V for the Zigbit module as shown in **Figure 12**. To attain the 3.3V we used a 12VDC power supply and fed it into an LM317 voltage regulator.

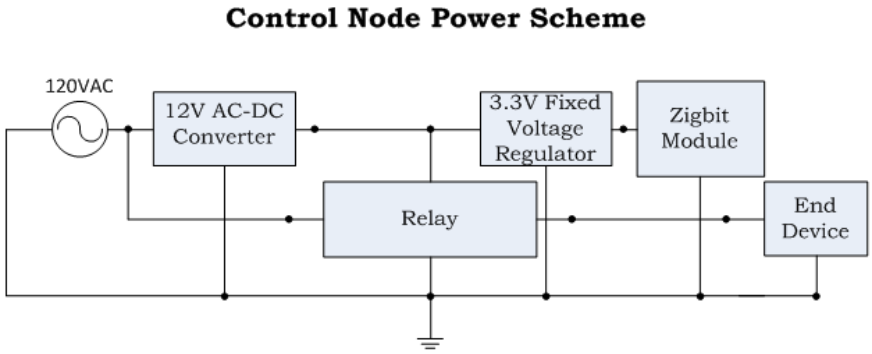

Figure 12: Control Node Power Scheme

5.2.3 Software

5.2.4 Embedded Firmware

We decided to use the Bitcloud SDK in order to implement our wireless Zigbee network. Any application built using Bitcloud is guaranteed to be Zigbee compliant. The first step before programming was to research the Zigbee protocol. The Zigbee specification is represented in a 600 page datasheet but is also summarized at wikipedia.com[20] and zigbee.org.[19]

A Zigbee network contains exactly one Coordinator node and a nearly limitless number of End Devices and Router nodes. The Coordinator is responsible for setting up and managing the network. Since every deployment of our system requires a Base Station, this was the natural choice to assign the role of Coordinator. Since the Sensor Nodes and the Control Nodes needed to send/receive data from the Base Station, they were Configured as End Devices. End Devices are able to enter a very low power state called "Sleep". Since our Sensor Node was battery powered it was important for it to conserve power. The Router node of a Zigbee network doesnt sleep and is usually used to pass on data from other nodes, so we decided that none of our devices would be assigned this role. In a future implementation it would be possible to create Router nodes that would boost the range of the system.

We created a simple packet structure that each node in the Zigbee network used to communicate. The packet diagram is shown below in **Figure 13**. The header and footer sections are used by the Bitcloud stack. The payload section contains the packet that we designed. The Base Station ID is filled in by the Base Station and contains a number that is unique to one Base Station. The Node ID contains the ID of the source node or the destination node depending on which opcode is used. The opcode defines the functionality and type of the packet. We had four different packets which were, motion sensor status, light sensor status, control node status, and control node command. For each of the status packets the data field contained the corresponding sensor value. For the control node command packet the data field contained a '1' or a '0' indicating to the sensor node to turn the light on or off, respectively.

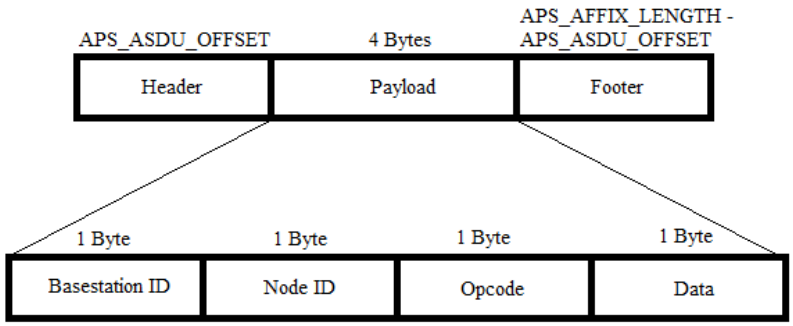

Figure 13: RF Packet Diagram

The next step was to learn how to program using the Bitcloud libraries. The Atmel Bitcloud Developer Guide contained all the information needed to utilize the various Bitcloud APIs. Bitcloud has a very steep learning curve and it was important to read and re-read this document several times in addition to referencing it. The developer guide discusses how the Bitcloud OS works as well as a basic user application template that includes all function definitions required by the Bitcloud stack. The Bitcloud OS is technically not a full blown OS. It is an event-driven system that designates tasks to the lower software stack layers. Upon completion of each task, a user defined callback function is invoked. The user application is able to change the state of execution and post a task which allows the underlying stack to handle the request whenever possible. The best programming style, as recommended by the developer guide, to utilize this system was to implement our application as a state machine.

Each of our devices shared the common state machine responsible for the network joining procedure. This state machine is shown in **Figure 14**. The task handler code that implements this state machine is located in the common.c file. Upon successful network join, tasks were delegated by each particular devices task handler. Each device-specific task handler was also implemented as a state machine, these are shown in the diagrams below.

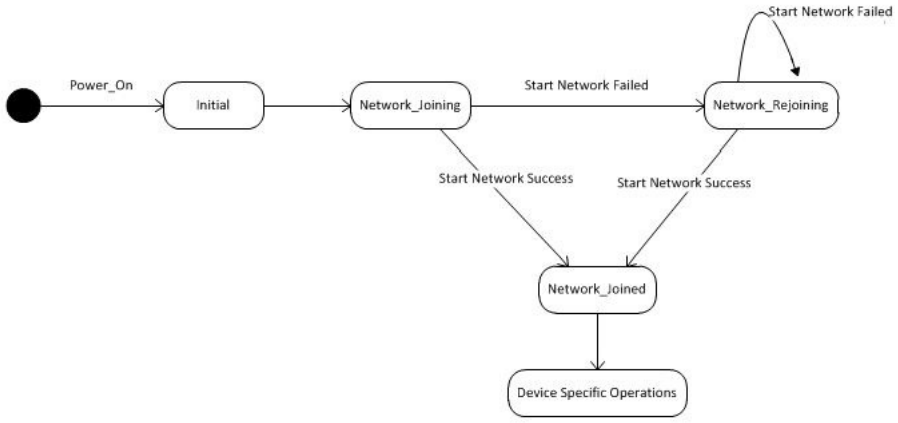

Figure 14: Network State Machine

The Network State Machine shows the process that the Nodes use to join the Zigbee network as seen in **Figure 14**.

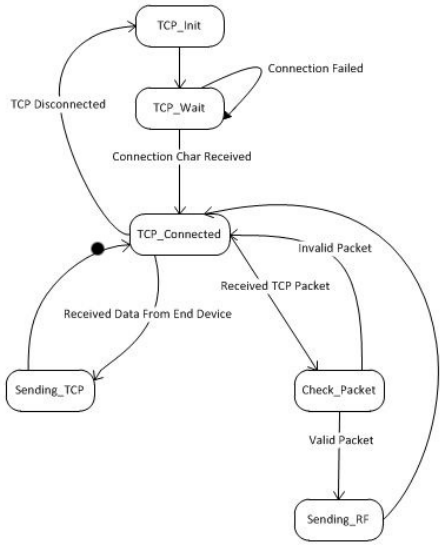

Figure 15: Base Station State Machine

The Base Station State Machine as seen in **Figure 15** shows the software process that the Base Station used to connect to the Webserver.

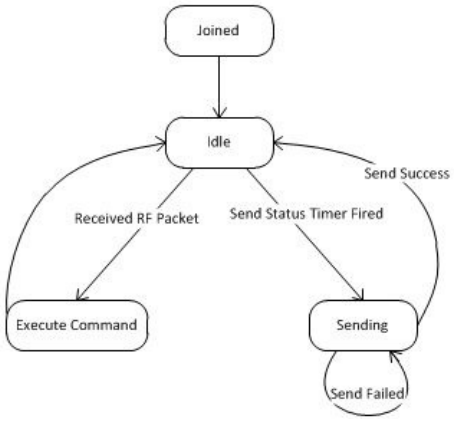

Figure 16: Control Node State Machine

The Control Node State Machine shows the process that the Control Node uses to communitate with the Base Station via the Zigbee network as seen in **Figure 16**.

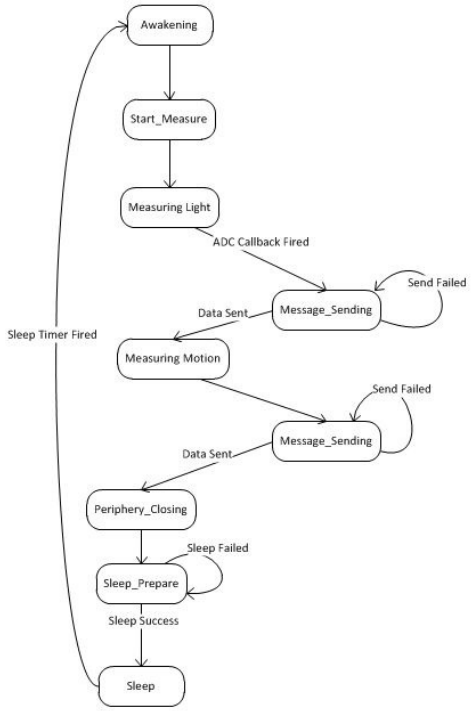

Figure 17: Sensor Node State Machine

The Sensor Node State Machine shows the process that the Sensor Node uses to communicate with the Base Station via the Zigbee network as seen in **Figure 17**.

5.2.5 Website

Initially for the website we tried to use an old netbook as the Webserver and the service no-ip.com which provides a service that automatically routes the domain name to the dynamic IP of the Webserver. That way when Charter changes the IP it was accessible. This idea did not work. The no-ip.com service was very slow to resolve the DNS name and render the pages. Due to the slowness of the machine and the no-ip.com services delays we abandoned this approach for the project.

Due to these issues we looked into other hosting options. We discovered that Amazon now has their virtual server EC2 service with a free 6 month trial. This was perfect for our project since on the Amazon EC2 server we had full control. We could load any OS, open any port and setup the permissions however we would like. Also the response time improved dramatically. Amazons servers were much better suited for the project.

Once the server hosting platform had been decided, we needed to load an OS onto it. We decided to use Ubuntu 10.04 since it is Ubuntu's latest Long Term Release, which

usually is the most stable and supported version. We installed Ubuntu 10.04 and then setup an SSH key access to the server. From there we installed Apache, MySql, PHP, PHPMyAdmin, and setup a basic website. The Webserver was now up and running ready to host content.

To transfer file to the server we used SFTP over SSH with the put and get commands to push and pull files from the server. This was very inefficient, so we installed Dropbox on the server via the command line. Then we setup a simlink from the apache www folder where the website is supposed to be, and linked it to a folder in Dropbox. This allowed us to save the files locally on our PCs and then they would automatically be uploaded to the server via the Dropbox service. This really helped speed up development work since we could work locally in a text editor and anytime we saved the website would update.

Next it was time to build the website. First we found a user management system called UserCake. Its a open source PHP user management system. This gave us a great start since we already had a login system setup and didnt need to debug it. After the login system was established we focused on creating the PHP TCP socket server to communicate with the Base Station. This proved to be a timely task.

Before creating the socket server we needed to establish a communications protocol between the Webserver and the Base Station. We decided to use 4 bytes for each data packet as seen in **Figure 18**. The first byte consisted of a unique Base Station ID that the packet was sent or received from. The second byte held the node id which is sending or receiving data. The third byte was the opcode which allowed us to distingusih between the Control and Senor Nodes and the different operations on each. Finally the fourth byte consisted of data from the Control and Sensor Nodes, its used to send and receive data from the server to the Nodes.

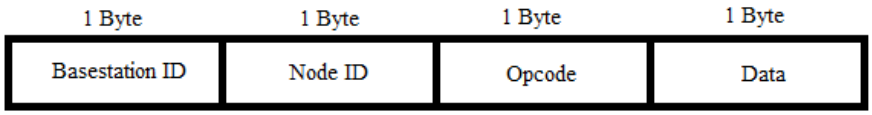

Figure 18: TCP Packet Diagram

For the TCP socket server we used PHP since it was already configured to manage the MySQL database. The MySQL database provides the interface through which the TCP socket server and the website communicate. For the first attempt at the TCP socket server we used an array to keep track of clients, however if any client had an error and crashed the entire TCP socket server would shut down. This was a major issue so we revised the server. To remedy this the new server now upon accepting a new connection forked off a new child process to handle the connection. This process handles the connection until it ends, or an error occurs. Once we implemented forking the TCP socket server became much more reliable and could handle many connections simultaneously. See **Figure 19**.

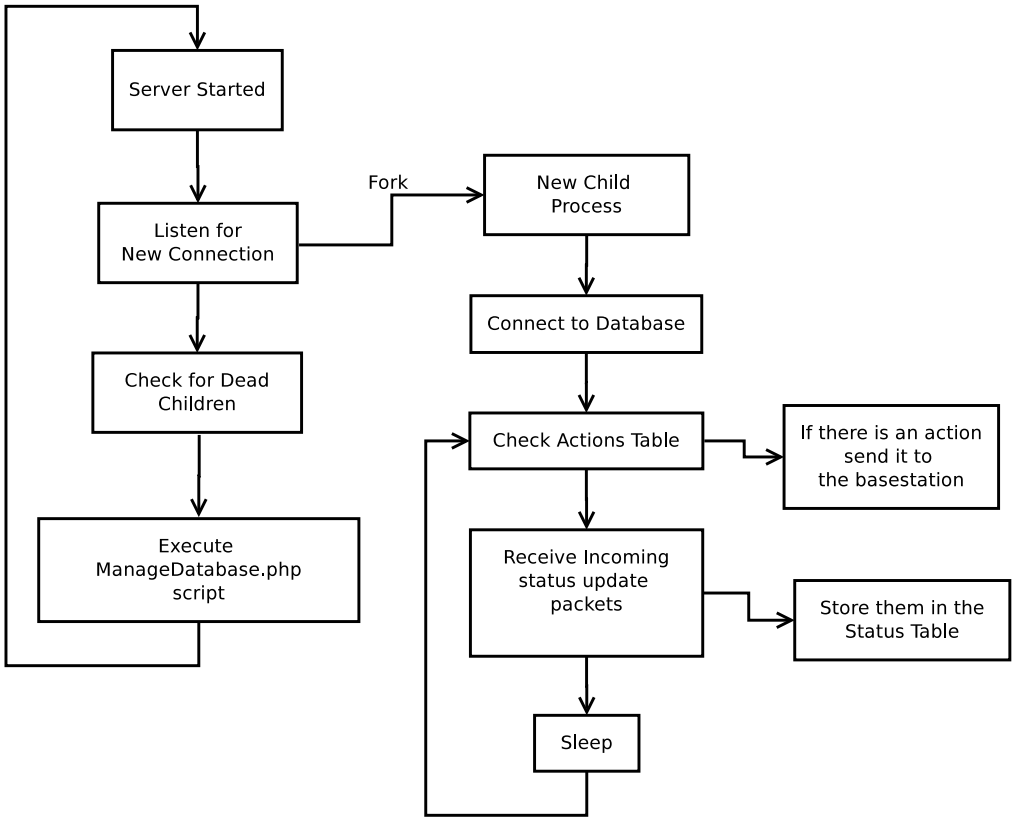

Figure 19: PHP TCP Socket Server

After the TCP socket server was stable we needed to manage the database. For this task we create the script ManageDatabase.php which is executed each time the server waits for a new connection. This script runs once and then exits. It manages the database for all of the website functions. By executing it once a loop if we changed the ManageDatabase.php script on the next loop the new changes would be used and we did not need to shutdown the server. This allows the ManageDatabase.php script to be changed at anytime without affecting the users functionality.

The database ManageDatabase.php script checks the status table, where all status updates are stored. From the status table it cross references the nodes table, if the node has not been seen before it is added to the nodes table with a unique_node_id. The rest of the script handles the automation, notifications, and the propagation of status updates into the userstatus table. The userstatus table contains the latest user status's which are shown on the status page.

The next step was to design the website, we used Twitters BootStrap javascript framework for the layout and javascript buttons. It allows the site to have a fluid design so

that when on a mobile device the site can adapt to the correct screen size. we continuously implemented a feature, tested, implemented and tested until all of the features were complete. Finally the website was complete when the user clicks on the On/Off button an action is placed into the actions table. If the Base Station is on an active TCP connection then the thread handling the connection will check the actions table, see the action, and send it to the Base Station. Upon sending the action it is deleted from the actions table.

For the status page we dynamically generate a tab for each Base Station and a bar for each node on the Base Station by using PHP. To get live updates on the status page we used javascript with an ajax call to update the page every 10 seconds with the latest statuss from the userstatus table. Finally we implemented the automation and notifications. For the notifications we installed postfix, and set it up with PHP as our mail server. Finally we started testing the Webserver and worked out the bugs.

5.2.6 Bill of Materials

This project cost a total of $154.90. This cost would only decrease with a high production volume therefore this total cost figure acts as an indicator of potential retail value per base system.

KreativeOutlets Bill of Materials				
Component	**Model Number**	**Quantity**	**Price Per Unit**	**Cost**
Microcontroller Tranceiver Module	556-ATZB-24-A2 Atmel Zigbee/802.15.4 Modules &	3	$29.00	$87.00
Ethernet Module	XP1001001-04 Xport XE	1	$58.00	Free Sample
PCB Fabrication	Sunstone	3	$19.00	$19.00
12V ACDC Adapter	CH-1205	1	$6.00	$6.00
5V ACDC Adapter	EPA-201DA-05	1	$6.00	$6.00
3.3V Fixed Voltage Regulator	LD33V	1	$3.00	$3.00
Adjustable Voltage Regulator	LM317	1	$3.00	$3.00
3.6V Lithiom Ion Battery	Tadiran	1	$7.00	$7.00
Push-button Switch	Radioshack 275-617	2	$3.19	$6.38
LEDs	N/A	9	$0.50	$4.50
Resistors	N/A	9	$0.10	$0.90
Perf Board with traces	N/A	1	$6.00	$6.00
Perf Board without traces	N/A	1	$5.00	$5.00
Webserver Hosting	Amazon Hosting	3 months	N/A	$1.12
Total				**$154.90**

Figure 20: Bill of Materials

6 Setup

Before you can configure your KreativeOutlets system, you need to ensure that you have an established internet connection in your home. Plug your Base Stations power cable into an electrical outlet and press the red power button on the top of the enclosure to power it on. Connect a CAT5 ethernet cable between the ethernet port on the Base Station and your network router. Plug your Control Node in an electrical outlet and press the red power button on the top of the enclosure to power it on. Plug in an end device that you would like to control such as a lamp or television. Place the Sensor Node in a room location that has the best birds eye view of the area for most accurate operation. Verify power for all devices indicated by the lit LEDs contained within each enclosure.

Figure 21: Home Page

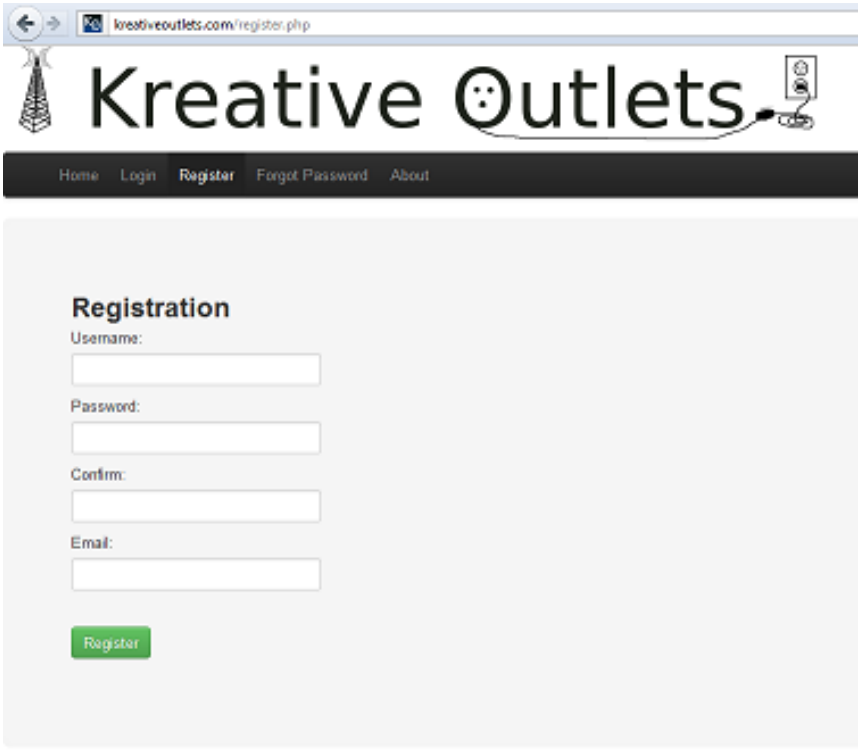

Figure 22: Registion Page

Create Kreative Outlets Account:

1. Open an internet browser
2. Navigate to http://www.kreativeoutlets.com/
3. Click the Register header
4. Key in the following:
    - Username
    - Password
    - Verify password
    - E-mail address
5. Click the Register button

6. Go to your e-mail inbox and follow the link in the confirmation e-mail
   (This will activate your account with KreativeOutlets) Once logged in you are able to activate/remove nodes, view live sensor status updates, add notifications, and add automations.

Activating Nodes:

1. Click the Settings header
2. Select your current time zone and click the Set Time Zone button
3. Click the Add/Remove Base Station link under Settings
4. Key in the number 1 if this is the first Base Station in your home
5. Click the Add Base Station button

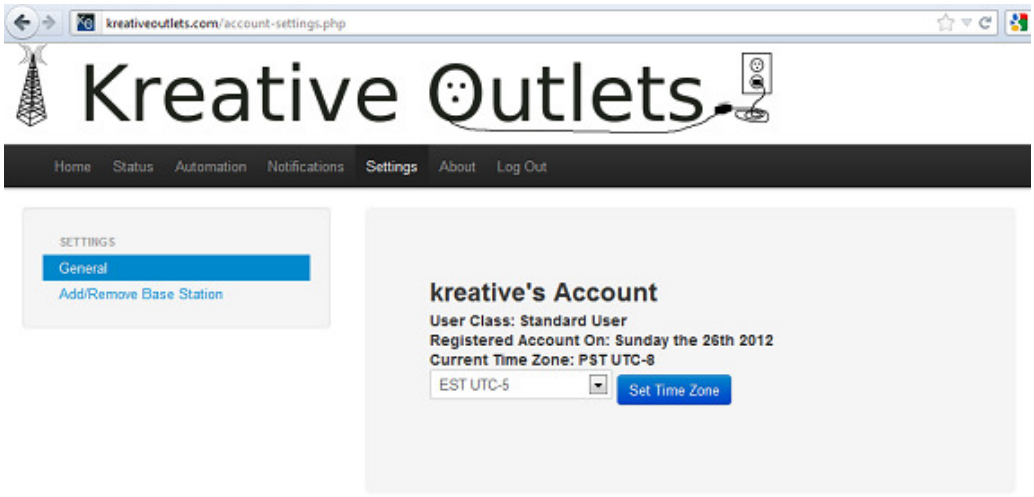

Figure 23: Account Settings Page

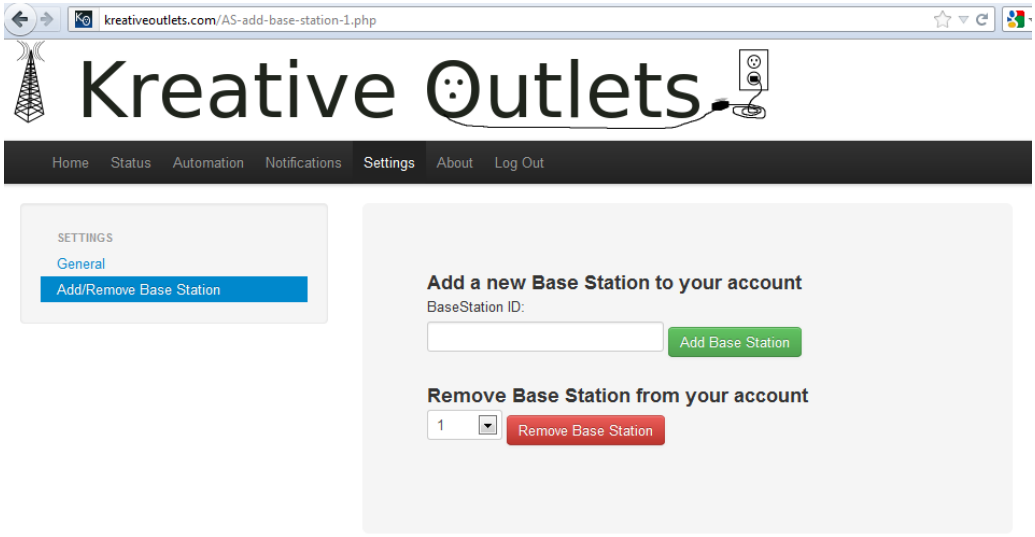

Figure 24: Add or Remove Base Station(s)

Removing Nodes:

1. Click the Settings header
2. Click the Add/Remove Base Station link under Settings
3. Select Base Station ID number from drop-down menu
4. Click the Remove Base Station button.

Add Light Notification:

1. Click the Notifications header
2. For Light % choose the following:
   - >
   - 70%
   - Choose Light Node #4
3. Click Add Light Notification button
   (This will send an e-mail notification if the light sensor detects light percentage greater than 70% indicating that someone may have turned a light on in your house when no one should be home)

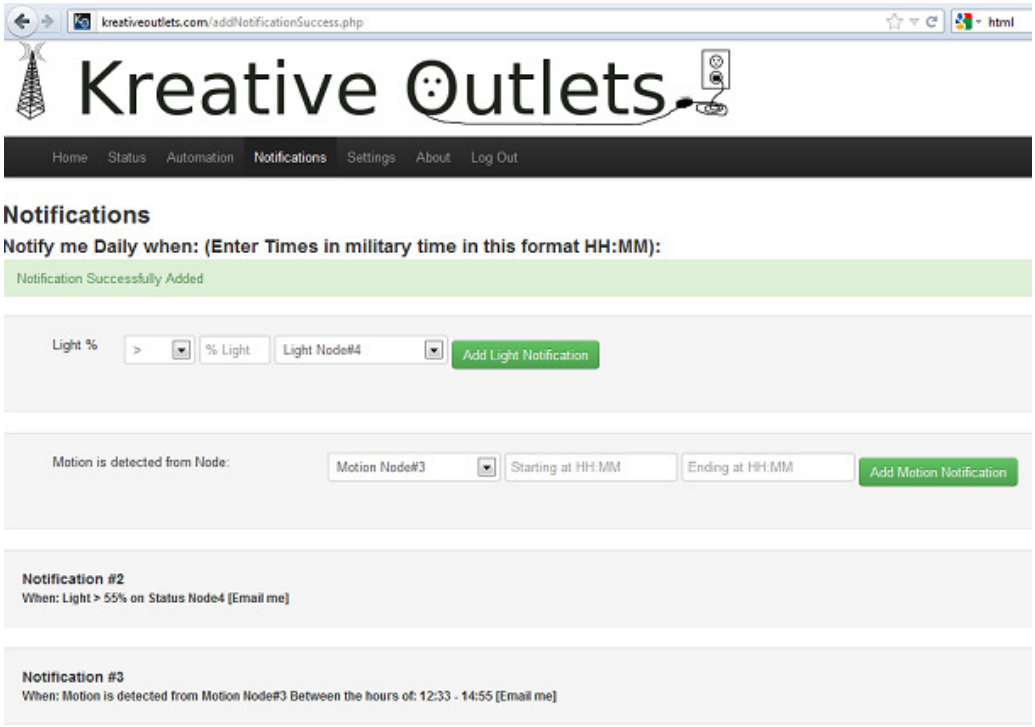

Figure 25: Add Notifications

Add Motion Notification:

1. Click the Notifications header
2. For Motion is detected from Node choose the following:
   - Motion Node#1
   - 08:00
   - 18:00
3. Click the Add Motion Notification button
   (This will send an e-mail notification if the motion sensor detects movement in your home when no one should be home while you are at work from 8am to 6pm)

Add Light Automation:

1. Click the Automation header
2. For Light % choose the following:

- >
- 70%
- Choose Light Node #4
- Turn Off
- Control Node #1

3. Click Add Light Automation button
   (This will turn off the lamp connected to Control Node#1 when Light Node#4 detects that the light level in that particular room is sufficient without the lamp. Conversely, you can create an automation to turn on that lamp if the natural light level is below a chosen threshold.)

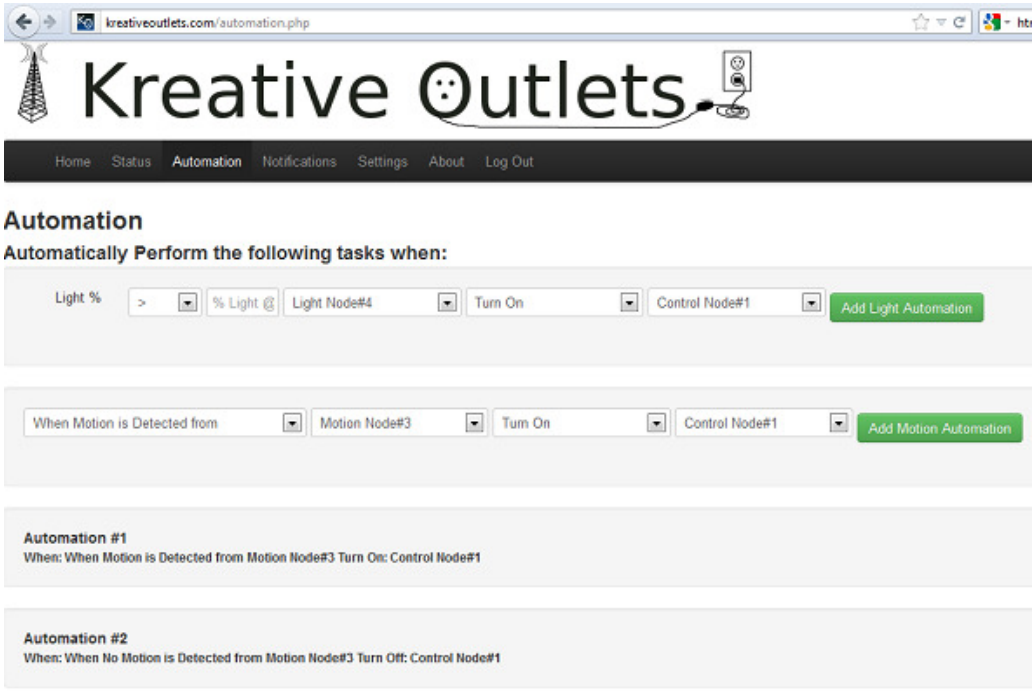

Figure 26: Automation

Add Motion Automation:

1. Click the Automation header
2. For Motion is detected from Node choose the following:
   - When Motion is Detected from

- Motion Node#3
- Turn On
- Control Node#1

3. Click the Add Motion Automation button
   (This will turn on the lamp connected to Control Node#1 when Motion Node#3 detects that there is motion in that particular room indicating someone has entered the room. Conversely, you can create an automation to turn off that lamp if there is no motion in that room indicating that the room is vacant.)

Manual Control:

1. Click the Status header (A page with a tab for each of your systems activated Base Stations should appear)
2. Click the ON/OFF buttons (This powers ON/OFF the end device attached to your Control Node. The Status indicator to the right indicates whether the device is already ON or OFF)

Figure 27: Status Page

Status Indicators:

1. Click the Status header (Motion Detected displays whether motion has been detected recently or not. Light Intensity displays the last data packet received for the light sensor indicating the light intensity percentage as well as a date/time stamp)

7 Final Product

The Base Station can be seen in **Figure 28**.

Figure 28: Base Station

The Sensor Node can be seen in **Figure 29**

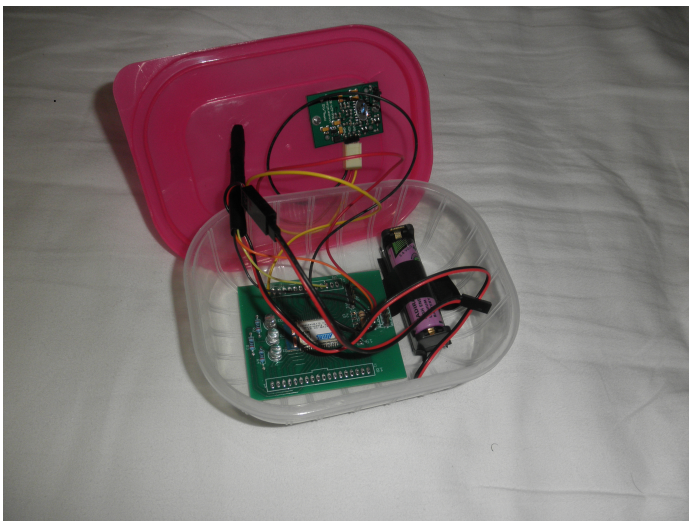

Figure 29: Sensor Node

The Control Node can be seen in **Figure 30**

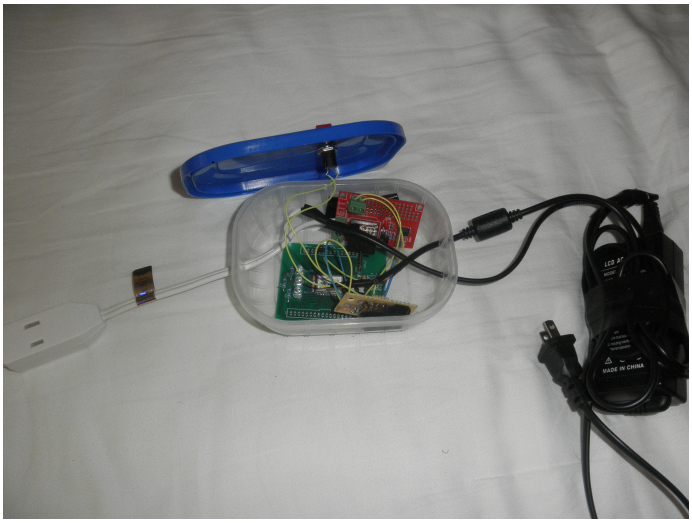

Figure 30: Control Node

8 Similar Products

- Z-Wave[18]
- X10 Home Automation[17]
- ONE-NET[9]
- INSTEON[4]

9 Glossary of Terms

API - Application Programming Interface

BitCloud - Software stack that which provides all the necessary low-level layers needed to create Zigbee compliant networks.

CAD - Computer Aided Design

CSS3 - Cascading Style Sheets, a style sheet language used to format the layout, colors, and fonts of a webpage.

GPIO - General Purpose Input/Output pin

HTML5 - HyperText Markup Language

IDE - Integrated Development Environment

IP address - An Internet Protocol Address is a numerical address designating the computer network a device has joined.

ISP - In System Programmer

Javascript - A prototype-based scripting language.

JTAG - Joint Test Action Group, set of pins on our microcontroller used to program and debug our embedded firmware.

Opto-isolation - Electronic protection circuitry that utilizes an LED and a phototransistor to isolate the low voltage side from the high voltage side.

PCB - Printed Circuit Board

PHP5.3 - General-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

RF transceiver - A module capable of transmitting and receiving data on the Radio Frequency band.

Relay - A switch that is flipped when an electrical signal is applied and returns to its initial state when the electrical signal is removed.

SDK - Software Development Kit

TCP - Transmission Control Protocol provides an efficient reliable means of transferring/receiving data packets from a program on one computer to a program on another computer.

ZigBee - A specification for a collection of high-level communication protocols. The protocol is optimized for low data-rate, low power consumption, RF networks.

References

[1] "Arduino, xport, php and the internet," March 2012. [Online]. Available: http://www.glacialwanderer.com/hobbyrobotics/?p=15

[2] "Atmel bitcloud quick start guide," March 2012. [Online]. Available: www.atmel.com/Images/doc8200.pdf

[3] "Atmel diagram," March 2012. [Online]. Available: http://www.atmel.com/Images/doc8226.pdf

[4] "Atmel diagram," March 2012. [Online]. Available: http://www.insteon.net/

[5] "Avr freaks," March 2012. [Online]. Available: http://www.avrfreaks.net/

[6] "Bjt diagram," March 2012. [Online]. Available: http://www.olimex.cl/pdf/ET-OPTO\%20RELAY4\%20Manual.pdf

[7] "Bootstrap api and site," March 2012. [Online]. Available: http://twitter.github.com/bootstrap/

[8] "Ethernet shield," March 2012. [Online]. Available: http://www.ladyada.net/make/eshield/examples.html

[9] "One-new," March 2012. [Online]. Available: http://www.one-net.info/

[10] "Photocells," March 2012. [Online]. Available: http://www.ladyada.net/learn/sensors/cds.html

[11] "Php api and site," March 2012. [Online]. Available: http://www.php.net/

[12] "phpmyadmin, manage database," March 2012. [Online]. Available: http://www.phpmyadmin.net/home_page/index.php

[13] "Pir motion sensors," March 2012. [Online]. Available: http://www.ladyada.net/learn/sensors/pir.html

[14] "Ubuntu," March 2012. [Online]. Available: http://www.ubuntu.com/

[15] "Used for sending email," March 2012. [Online]. Available: http://www.postfix.org/

[16] "usercake the fully open source user management script," March 2012. [Online]. Available: http://usercake.com/

[17] "X10 home automation," March 2012. [Online]. Available: http://www.x10.com/homepage.htm

[18] "Z-wave," March 2012. [Online]. Available: http://www.z-wave.com/modules/ZwaveStart/

[19] "Zigbee alliance," March 2012. [Online]. Available: http://www.zigbee.org/

[20] "Zigbee information," March 2012. [Online]. Available: http://en.wikipedia.org/wiki/ZigBee

[21] "Zigbit modules mcu wireless," March 2012. [Online]. Available: http://www.atmel.com/products/microcontrollers/wireless/modules.aspx

[22] Amazon, "Amazon elastic compute cloud (amazon ec2)," March 2012. [Online]. Available: http://aws.amazon.com/ec2/

Information about Amazons EC2 Service

Analysis of Senior Project Design

Please provide the following information regarding your Senior Project and submit to your advisor along with your final report. Attach additional sheets for your responses to the questions below.

Project Title: KreativeOutlets

Quarter / Year Submitted: Winter 2012

Student: (Print Name) Travis Crist (Sign) _____

Student: (Print Name) Steve Clark (Sign) _____

Student: (Print Name) Jason Peressini (Sign) _____

Advisor: (Print Name) Hugh Smith (Initial) _____ Date: _____

- **Summary of Functional Requirements**

  Describe the overall capabilities of functions of your project or design. Describe what your project does. (Do *not* describe how you designed it.)

- **Primary Constraints**

  Describe significant challenges or difficulties associated with your project or implementation. For example, what were limiting factors or other issues that impacted your approach? What made your project difficult? What parameters or specifications limited your options or directed your approach?

- **Economic**

  o Original estimated cost of component parts (as of the start of your project)
  o Actual final cost of component parts (at the end of your project)
  o *Attach a final bill of materials for all components*
  o Additional equipment costs (any equipment needed for development?)
  o Original estimated development time (as of the start of your project)
  o Actual development time (at the end of your project)

- **If manufactured on a commercial basis:**

  o Estimated number of devices to be sold per year
  o Estimated manufacturing cost for each device
  o Estimated purchase price for each device
  o Estimated profit per year
  o Estimated cost for user to operate device, per unit time (specify time interval)

- **Environmental**

  Describe any environmental impact associated with manufacturing or use.

- **Manufacturability**

  Describe any issues or challenges associated with manufacturing.

- **Sustainability**

  o Describe any issues or challenges associated with maintaining the completed device or system.
  o Describe how the project impacts the sustainable use of resources.
  o Describe any upgrades that would improve the design of the project.
  o Describe any issues or challenges associated with upgrading the design.

- **Ethical**

  Describe ethical implications relating to the design, manufacture, use or misuse of the project.

- **Health and Safety**

  Describe any health and safety concerns associated with design, manufacture or use.

- **Social and Political**

  Describe any social and political concerns associated with design, manufacture or use.

- **Development**

  Describe any new tools or techniques used for either development or analysis that you learned independently during the course of your project.

- **Summary of Functional Requirements**

  We designed a system that we call a "Home Automation System". It provides the user with the capability to turn on or off AC outlets in their house that they have configured with the system. Along with being able to control their outlets on the fly, they are also able to automate the control. The user can set up calendar based automation using time and date as well as setting up sensor based automation. Our system contains a motion sensor and light intensity sensor, both of which can be used to control the status of the user's outlets. Another feature of our system is that it allows the user to check the light intensity status as well as motion status in order to provide some security monitoring.

- **Primary Constraints**

  To complete this project we first had to decide on what kind of network to use between our Nodes. We decided to use a ZigBee star network for its low power and performance. Other considerations that we took include making the Basestation Node a dummy Node, all it does is forward packets from the Webserver to the Sensor/Control Nodes and it forwards packets from the Sensor/Control Nodes back to the webserver. By making this a dumb Node we can add new Nodes with new feature very easily without the end user needing to replace a Basestation. Another hurdle was keeping the Basestation connected to the Webserver. This was solved by sending a keep alive packet about ever 5 seconds to the Webserver. One of the most difficult challenges of the project was handling multiple clients via a TCP socket server. Many hours went into debugging and testing the TCP socket server code so that it would run all day without any issues or performance hits on the server. Implementing the ZigBee RF network proved to be a very time consuming process because learning the Bitcloud stack was not easy. The Bitcloud stack was difficult to learn but once we did, it proved to be a very useful and powerful stack.

- **Economic**
  - Original estimated cost of component parts: $100
  - Actual final cost of component parts: $154.90
  - Additional equipment costs: Logic Analyzer $150
  - Original estimated development time: ~20 weeks
  - Actual development time: 20 weeks

- **If manufactured on a commercial basis:**
  - Estimated number of devices to be sold per year
  - Estimated manufacturing cost for each device
  - Estimated purchase price for each device
  - Estimated profit per year
  - Estimated cost for user to operate device, per unit time (specify time interval)

- **Environmental**

  Our project has very little environmental impact. Our devices use a small amount of electricity but they are also capable of being configured in a way that would allow the user to actually save electricity by only using power (lights, heater, etc.) when needed. There would be a small amount of environmental impact during the manufacturing of our electronic components as well as our printed circuit boards and packaging needed to contain our electric components.

- **Manufacturability**

  The manufacturing capability needed for our project is fairly minimal. Our semiconductor based digital circuits require manufacturing which is already being provided by several companies. We would need some manufacturing for our PCBs for which there are also several companies that can provide this service. It would be nice to have some heavy duty plastic enclosures manufactured in order to provide protection for our digital circuits.

- **Sustainability**

  We designed our system with future advancement in mind. Most of our logic that would need to be changed in order to add new features is located server-side. This allows us to make changes and add features without disrupting current users performance. Our hardware is set up in a modular way that would allow us to sell new products that could be added to an already existing system. One device in our system requires the use a battery. This firmware for this device could be designed in a way that would allow more efficient use of the battery and not require as many recharges/replacements. Our power schemes could also be updated to use more complex circuitry in order to provide more efficient power use for our whole system.

- **Ethical**

  There is the possibility that somebody could spoof our system and use it to control another user's house. We didn't put any explicit security features to prevent this from happening. In future revisions it would be very possible to implement theses security measures.

- **Health and Safety**

  The only health and safety consideration needed for our product is the possibility of electric shock. One of our system components must be place between the power of an AC outlet which is 120VAC and can provide enough current to cause death in the case of accidental misuse. Our product is electrically isolated and shrink tubing has been provided to minimize this risk. A possible revision would include a safety circuit that would blow a fuse in the case of accidental human exposure.

- **Social and Political**

  Our product relies on the availability of the Internet for public use. The Internet has recently become the target for political regulation and it is possible for laws to be put in place that would render our product unusable. Our system also uses an RF network and the frequency band that our network uses is controlled by the FCC. We wouldn't foresee this every being a problem however.

- **Development**

  In the development of this project we learned how to interface a basic electronic circuit web front end. This involved learning how to use a TCP connection effectively to communicate between the website and the electronic circuit. Through this process we learned the importance of establishing communications protocols for the network, how to build a website with an interactive control panel, how to create an RF network between electronic devices, and most importantly the importance of good team communication. Through this project we developed a base network that could be built upon to create a home automation system capable of doing many tasks.

Bill of Materials:

KreativeOutlets Bill of Materials				
Component	**Model Number**	**Quantity**	**Price Per Unit**	**Cost**
Microcontroller Tranceiver Module	556-ATZB-24-A2 Atmel Zigbee/802.15.4 Modules &	3	$29.00	$87.00
Ethernet Module	XP1001001-04 Xport XE	1	$58.00	Free Sample
PCB Fabrication	Sunstone	3	$19.00	$19.00
12V ACDC Adapter	CH-1205	1	$6.00	$6.00
5V ACDC Adapter	EPA-201DA-05	1	$6.00	$6.00
3.3V Fixed Voltage Regulator	LD33V	1	$3.00	$3.00
Adjustable Voltage Regulator	LM317	1	$3.00	$3.00
3.6V Lithiom Ion Battery	Tadiran	1	$7.00	$7.00
Push-button Switch	Radioshack 275-617	2	$3.19	$6.38
LEDs	N/A	9	$0.50	$4.50
Resistors	N/A	9	$0.10	$0.90
Perf Board with traces	N/A	1	$6.00	$6.00
Perf Board without traces	N/A	1	$5.00	$5.00
Webserver Hosting	Amazon Hosting	3 months	N/A	$1.12
Total				**$154.90**