Christmas Lights Animation

System Design Document | Current Version 1.5.0

Prepared By:

Austin Wentz

Jordan Doell

Revision History

Date	*Author*	*Version*	*Comments*
9/12	*Austin Wentz*	*1.0.0*	*Initial version*
12/11/12	*Austin Wentz*	*1.1.0*	*Fleshed out content*
4/20/13	*Jordan Doell*	*1.2.0*	*Added iOS app information*
4/22/13	Austin Wentz	*1.3.0*	*Added sprint reports 4,5,6 in appendix and added additional information*
4/23/13	Jordan Doell	*1.4.0*	*Add iOS app detailed implementation and testing*
4/24/13	Jordan Doell	*1.5.0*	*Added section for adding new songs to the app*

Table of Contents

1.0 Overview

1.1 Scope

This document covers the design specifications of the XMASLA project.

1.2 Purpose

To make an interactive Christmas lights animation product. It will be controllable from an iOS device.

1.2.1 Hardware

- Raspberry Pi or Intel Atom
- Renard 64XC
- SSRez solid state relays

1.2.2 iOS App and Device

The iPhone app will be able to send various commands to the server.

1.2.3 Miscellaneous

- Christmas lights
- Extension cords
- Display case

1.3 Systems Goals

- Choose a song from the iPhone app
- Play music and have the Christmas lights synced to the music
- Control each channel's brightness from the iPhone app to have a fully interactive experience

1.4 System Overview and Diagram

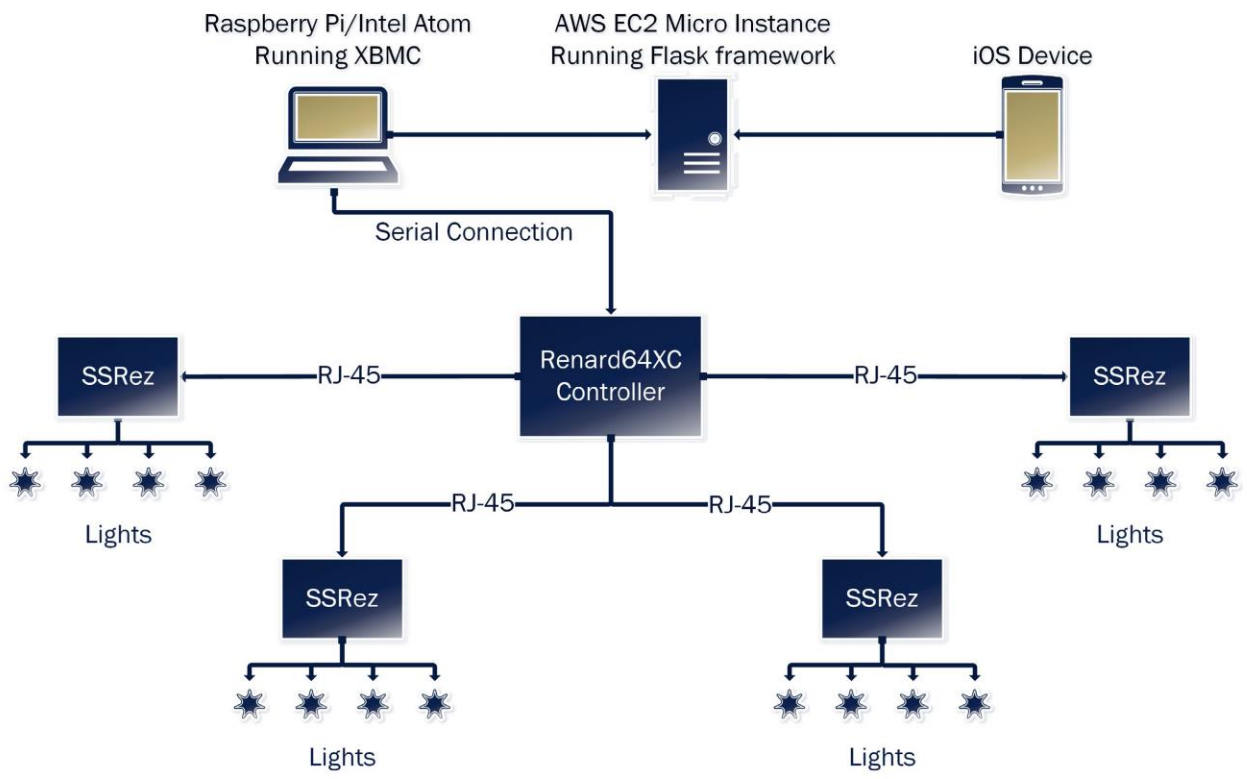


Figure 1 System Diagram

1.5 Technologies Overview

1.5.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. We used the AWS Free Tier so it cost us nothing to run a Linux Micro Instance upon which we ran our web server.

1.5.2 Flask

Flask is a BSD licensed microframework for Python web development. We used this framework for developing the middleman web server.

1.5.3 XBMC

XBMC is an open source (GPL) media center application. It works on Windows, Linux, and OSX. Since XBMC has a built-in Python interpreter, we added our light sequencing software as an add-on for XBMC.

2.0 Project Overview

2.1 Team Members and Roles

- Jordan Doell – iOS development / front end
- Austin Wentz – Hardware / back end

2.2 Project Management Approach

The project is managed using the Agile methodology Scrum. The Scrum Master is Dr. Jeff McGough. Sprints are 3 weeks in length and weekly meetings are held. Trello is used for managing the backlog.

2.3 Terminology and Acronyms

- SSR – Solid State Relay
- Renard – PIC-based dimming controller used to animate Christmas lights

3.0 Requirements

See the System Requirements Document.

4.0 Design and Implementation

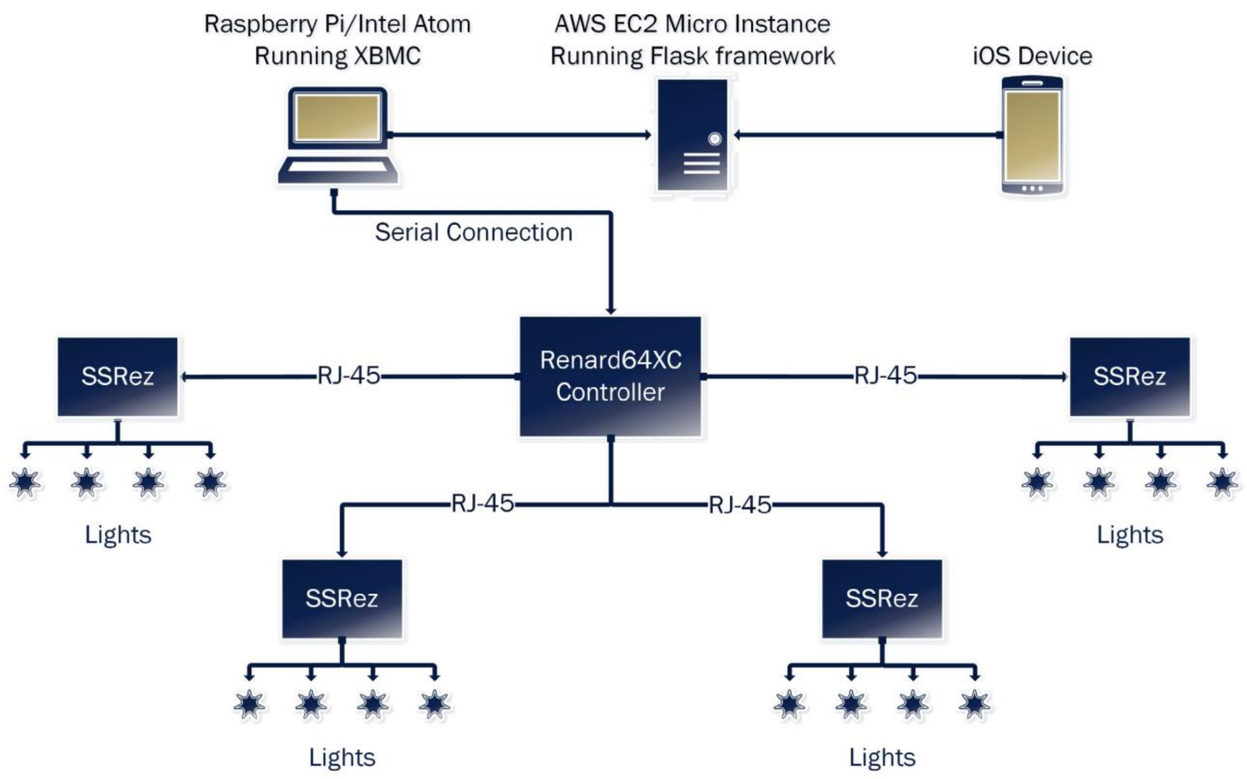
4.1 Hardware

4.1.1 Technologies Used

4.1.2 Component Overview

- Renard64XC
- SSRez
- Raspberry Pi/Intel Atom
- AWS EC2 Micro Instance

4.1.3 Architecture Diagram



4.2 iOS app and Device

4.2.1 Technologies Used

- Macbook Pro for iOS development
- XCode IDE
- iPhone simulator for testing
- JSON Data

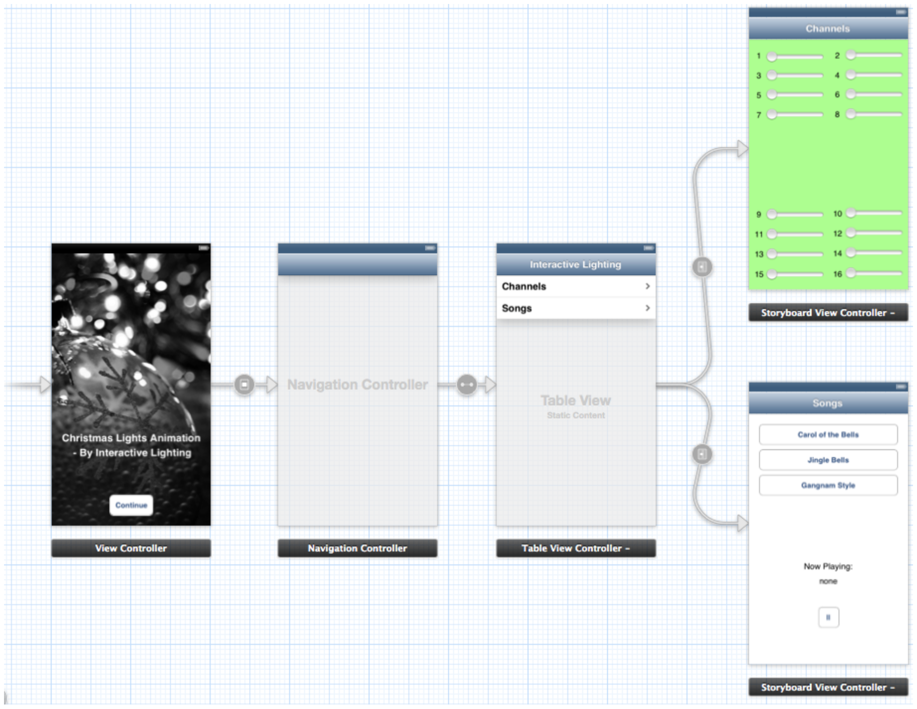
4.2.2 Component Overview

- Send commands to the server
    - Play a song
    - Pause a song
- Change each channel's brightness

4.2.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

4.2.4 Architecture Diagram



4.2.5 Design Details

When the app is started up, there is an initial view that has a continue button. After tapping continue, it goes into a table view with a section for channels and another section for songs. The channel section allows the user to change each individual channel's brightness. For the songs section, there is a button for each song. Tapping on a song button sends a command to play that song. Tapping the pause button pauses the song.

4.2.6 Views

There are 4 main views for the iPhone application.

1. Introductory view when app is opened. This is the picture of lights with a continue button.
2. This is the main view for the app. This is a table view created with a row for channels and a row for songs.
3. This view is to display the 16 different channels. Each channel has its own slider. By sliding a slider, this sends an array of new JSON values to the server. This array holds the values of the 16 channel's brightness levels.
4. The final view is for songs. This view shows a button for each song that is available. Also, there is a now playing section that shows the current song being played.

4.2.7 GUI Components

The GUI(graphical user interface) components were all created in the storyboard designer built into XCode. The event handlers were created in this way also, since it was easier due to the fact that I am still relatively new to iOS programming.

UISlider

- Simple slider for iOS interfaces
- Holds value for a channel's brightness

- Able to be tapped and slid around

UIButton

- Simple button for iOS interfaces
- Able to be tapped
- Displays song text

UILabel

- Simple labe for iOS interfaces
- Displays text

4.2.8 Sending JSON Data

Sending brightness values for each channel:

- To send out the data for each channel, a simple array was created that holds all the values from the 16 sliders in view 3 of the app.
- This array is then converted to an NSArray and packaged into a NSdictionary.
- The resulting NSDictionary is placed in a NSData object.
- Next, a NSMutalbeURLRequest is created with some attributes.
- Finally, a NSConnection is created using the request which sends the data to the server.
- Sample Data:
    - {"values": "0", "0", "0", "0", “0”, …. “0” }

Sending song information to play:

- To send out the data for a song, I extract the name of the song from the sender’s button.
- This string is then placed into a NSdictionary.
- The resulting NSDictionary is placed in a NSData object.
- Next, a NSMutalbeURLRequest is created with some attributes.
- Finally, a NSConnection is created using the request which sends the data to the server.
- Sample Data:
    - {"play": "Carol of the Bells”}

Sending song information to pause:

- To send out the data for a song, I extract the name of the song from the sender’s button.
- This string is then placed into a NSdictionary.
- The resulting NSDictionary is placed in a NSData object.
- Next, a NSMutalbeURLRequest is created with some attributes.
- Finally, a NSConnection is created using the request which sends the data to the server.
- Sample Data:
    - {"pause": "Carol of the Bells”}

4.2.9 Adding New Songs

To add a new song to the app, there are a few simple steps.

- Open up the project in XCode
- Open up the storyboard in the designer interface
- Click on an existing button
- Copy and paste for a new button and arrange where needed
- Change the name of the text on the button to the song name

4.3 XBMC Client Add-on

4.3.1 Technologies Used

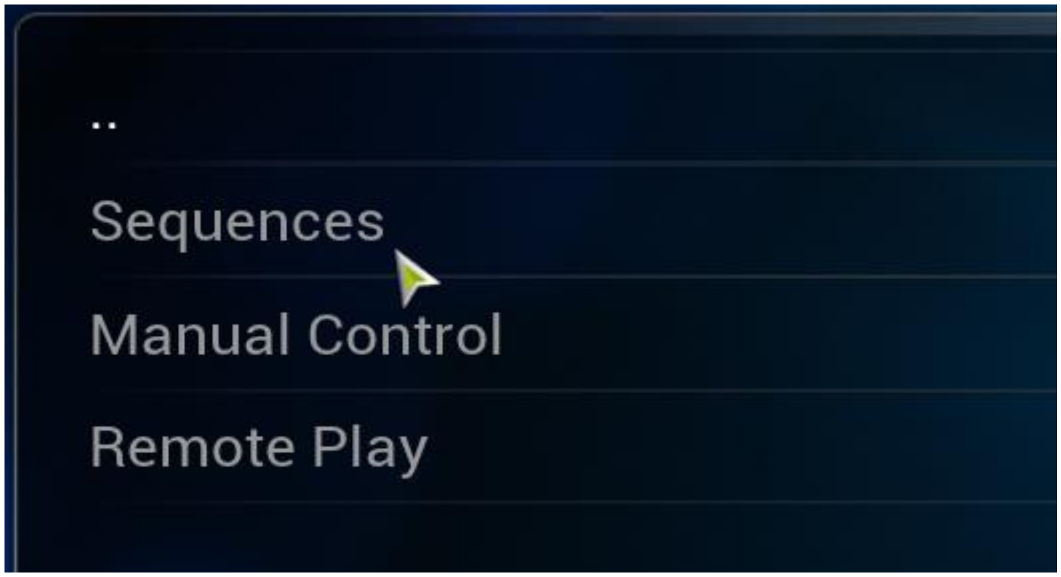
- Personal Computer for development and testing
- School laptop for development and testing
- Raspberry Pi for testing
- Intel Atom computer for presentation and testing
- Sublime Text 2 to develop add-on

4.3.2 Component Overview

- XBMC
- Interactive Lighting Add-on
    - sequences
        - List of available sequences to play
    - manual
        - allows manual control of the lights from the iOS app
    - remote
        - allows iOS app to play a sequence remotely

4.3.3 Design Details

The client add-on has three different modes. These modes are accessed via the main menu as pictured below.



4.3.3.1 Sequences

The sequence mode is for playing sequences directly from the client.

4.3.3.2 Manual Control

The manual control mode is for manually controlling the Christmas lights from the iOS app.

4.3.3.3 Remote Play

The remote play mode is for playing songs remotely via the iOS app.

5.0 System and Unit Testing

This section describes the approach taken with regard to system and unit testing.

5.1 Overview

Testing was done both in isolation (unit) and as a whole (system). Testing was done on the hardware, iOS App, client add-on, and on the web server.

5.2 Test Setup and Execution

5.2.1 Solid State Relay Testing

Testing was done on each of the eight SSRez. The SSR's were connected to the Renard controller and test lights were attached to the SSR's and then a jumper wire was used to test each channel according to the following table.

Renard Channel	Optoisolator	SSR Channel	Connect Jumper Wires between

1	U1	1	U6 IC socket pin 3 and pin 14
2	U2	2	U6 IC socket pin 13 and 14
3	U3	3	U6 IC socket pin 12 and 14
4	U4	4	U6 IC socket pin 11 and 14

5.2.2 Renard Controller Testing

The Renard 64XC was tested independently from the SSR's by using the onboard LED lights.

Channel	Output Jack	Test Result
1	J12	Pass
2	J12	Pass
3	J12	Pass
4	J12	Pass
5	J8	Pass
6	J8	Pass
7	J8	Pass
8	J8	Pass
9	J11	Pass
10	J11	Pass
11	J11	Pass
12	J11	Pass
13	J7	Pass
14	J7	Pass
15	J7	Pass
16	J7	Pass
17	J10	Pass
18	J10	Pass
19	J10	Pass
20	J10	Pass
21	J6	Pass
22	J6	Pass
23	J6	Pass
24	J6	Pass
25	J9	Pass
26	J9	Pass
27	J9	Pass
28	J9	Pass
29	J5	Pass
30	J5	Pass
31	J5	Pass
32	J5	Pass
33	J21	Pass
34	J21	Pass
35	J21	Pass
36	J21	Pass
37	J16	Pass
38	J16	Pass
39	J16	Pass
40	J16	Pass
41	J20	Pass
42	J20	Pass
43	J20	Pass
44	J20	Pass
45	J15	Pass
46	J15	Pass
47	J15	Pass
48	J15	Pass
49	J19	Pass
50	J19	Pass
51	J19	Pass
52	J19	Pass
53	J14	Pass
54	J14	Pass
55	J14	Pass

PIC Slot	Firmware Diagnostic Test
U7	Pass
U8	Pass
U9	Pass
U10	Pass
U11	Pass
U12	Pass
U13	Pass
U14	Pass

56	J14	Pass
57	J18	Pass
58	J18	Pass
59	J18	Pass
60	J18	Pass
61	J13	Pass
62	J13	Pass
63	J13	Pass
64	J13	Pass

5.2.3 Christmas Light and Extension Cord Testing

The Christmas lights and the extension cords were each individually tested by plugging them in.

5.2.4 iOS App Testing

The iOS app was tested initially by running it in the simulator and checking that the views were all loading properly. After the kinks were worked out, the views all transition back and forth correctly.

For the connection, me and Austin got together and I began sending several different commands to the server. He was able to see each command that was sent from the app, so the app is sending the JSON data properly to the server.

5.2.5 Client Add-on Testing

The add-on was tested for both correctness and usability.

5.2.6 Complete System Testing

Testing was done on the system as a whole.

6.0 Development Environment

6.1 Development IDE and Tools

Sublime Text 2 was used to develop the server and backend. Xcode was used for iOS development.

6.2 Source Control

Github is used for source control.

6.3 Dependencies

Describe all dependencies associated with developing the system.

6.4 Build Environment

Since Python is an interpreted language, no build environment was necessary.

7.0 Release | Setup | Deployment

This section should contain any specific subsection regarding specifics in releasing, setup, and/or deployment of the system.

7.1 Deployment Information and Dependencies

7.1.1 Client Dependencies

- pySerial
- httplib2
- xbmc
- vixen – for sequencing purposes

8.0 End User Documentation

See iPhone App User Guide and Interactive Lighting Client User Guide.

Appendix I: List of Figures

Appendix II: Supporting Information and Details

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

Appendix III: Progress | Sprint Reports

This section will contain a complete list of all of the period progress and/or sprint reports which are deliverables for the phases and versions of the system.

III.1 Sprint 1 Progress Report

Sprint Report 1

Team Members:	Austin Wentz and Jordan Doell
Date:	October 5, 2012
Class:	Senior Design
Subject:	Sprint 1 Report
Sponsor:	L-3: June Alexander-Knight

Sponsor Description:

L-3 Communications is a world class defense contractor. They play a huge role in the defense industry for the United States government. June Alexander-Knight graduated from SDSMT and since then, works for L-3. She has also been a strong supporter of SDSMT students and graduates.

Sponsor's Problem/Goal:

Sync Christmas lights to music using a Linux board and controller, and control the system using an iPhone app.

Customer Needs:

- Linux board to control lights
- SSR's to power on and off the strands of lights
- iPhone app to do sequences or play music
- Use sequencer software to program light show

Project Environment

Project Boundaries

- The project will have two separate environments: mobile device environment and Christmas lights controller environment

- The project’s mobile environment will be focused on iOS devices
- The project’s controller environment consists of Raspberry Pi, PIC microcontrollers, and additional circuitry to control the lights
- Communication between environments will be done over TCP/IP via JSON
- The mobile environment will be developed in Objective-C
- The controller environment will be developed in Python, and also in Clojure
- No code will need to be written for the PIC microcontrollers

Project Context

Technical Environment

The technical environment can be split into three parts: mobile device, high-level controller, and low-low-level controller.

Mobile Device

The iPhone is used as the mobile device. Development will be done on a Mac mini.

High-Level Controller

The Raspberry Pi is used as a high-level controller. It will receive commands from the mobile device, perform any required processing on command data, and send the commands to the low-level controller. The Raspberry Pi uses a Debian-like flavor of Linux. Development will be done in Linux and Windows.

Low-Level Controller

To directly control the Christmas lights, we are using a popular do-it-yourself light dimmer scheme called Renard. In particular we are using the Renard 64 XC design. No development needs to be done on the low-level controller.

Current Systems Overview

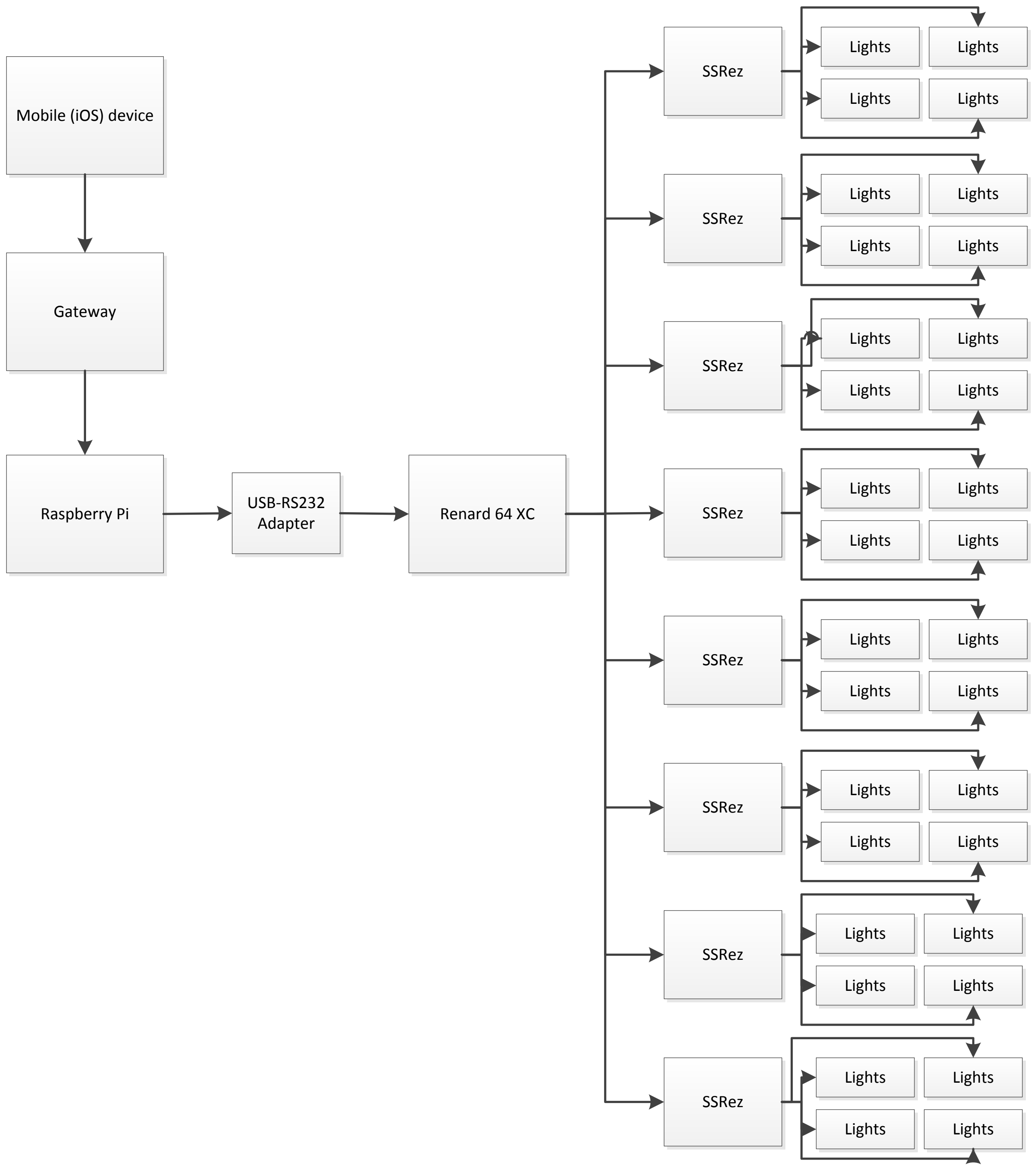


Figure 2: System Overview

Product Deliverables

No product deliverables at this point.

Future Product Deliverables

- Functional prototype
- Source code
- User manual / documentation
- Requirements document
- Design document

Backlog

Completed

- Purchase and configure single board computer (SBC) to act as high –level controller
- Purchase SSR pcb kit, SSR heat sinks, and Renard microcontroller pcb kit
- Analysis and research for design and requirements for project
- Start learning iOS development

Remaining

- Develop interface between Raspberry Pi and Renard 64XC
- Implement Renard serial protocol
- Develop prototype which switches lights on and off using predefined sequence
- Assemble additional circuitry (SSR and Renard kits)
- Purchase Christmas lights
- Purchase extension cords
- Program and configure Raspberry Pi to act as midi sequencer for lights
- Develop and implement iPhone app which controls the Christmas lights

Potential Issues

- Difficulty in assembling additional circuitry correctly
- Troubleshooting issues with SSRs and Renard microcontroller
- Safety issues when dealing with high voltage power sources
- Possible issues with iOS development

III.2 Sprint 2 Progress Report

Sprint 2 Report

Team Members:	Austin Wentz and Jordan Doell
Date:	November 1, 2012
Class:	Senior Design
Subject:	Sprint 2 Report
Sponsor:	L-3: June Alexander-Knight

Backlog

Completed

- Purchase and configure single board computer (SBC) to act as high –level controller
- Purchase SSR pcb kit, SSR heat sinks, and Renard microcontroller pcb kit
- Analysis and research for design and requirements for project
- Start learning iOS development
- Assemble additional circuitry (SSR and Renard kits)
- Implement Renard serial protocol
- Purchase Christmas lights
- Purchase extension cords
- Develop prototype which switches lights on and off using predefined sequence
- iPhone app prototype

Remaining

- Design display case for electronic components
- Have the display case made and assembled.
- Program and configure Raspberry Pi to act as midi sequencer for lights
- Develop and implement iPhone app which controls the Christmas lights

iOS Application progress:

Jordan Doell

During Sprint 2, I have been continuing to learn Objective-C and iOS application development. I found and have been watching a podcast that covers iOS development and Objective-C. Also, James has been lecturing to me and Josh about iOS and some of the components we will need for the project. We still have a few more lectures to go, but we are making progress.

App Prototype:

I have gained enough knowledge of iOS so far to make a simple prototype. It is nonfunctional so far, but gives a little direction to where we are headed with the app. Below are some screenshots of the different views in the app.

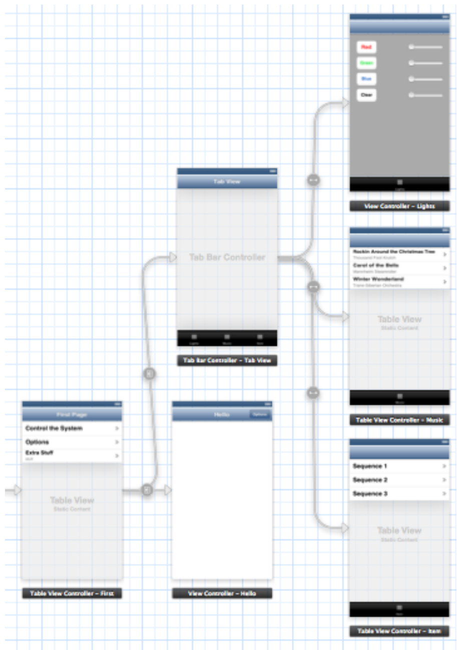


Fig. 1: Overall storyboard for the prototype

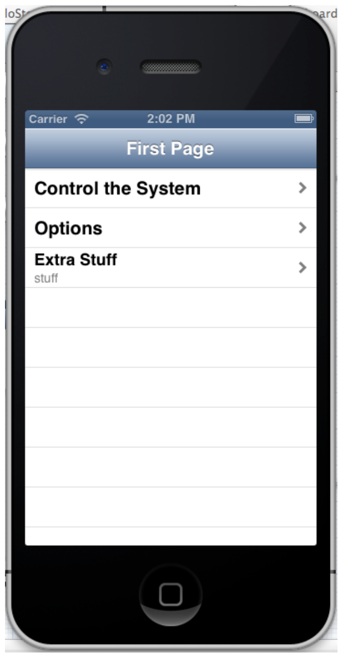


Fig. 2: Main page of the app

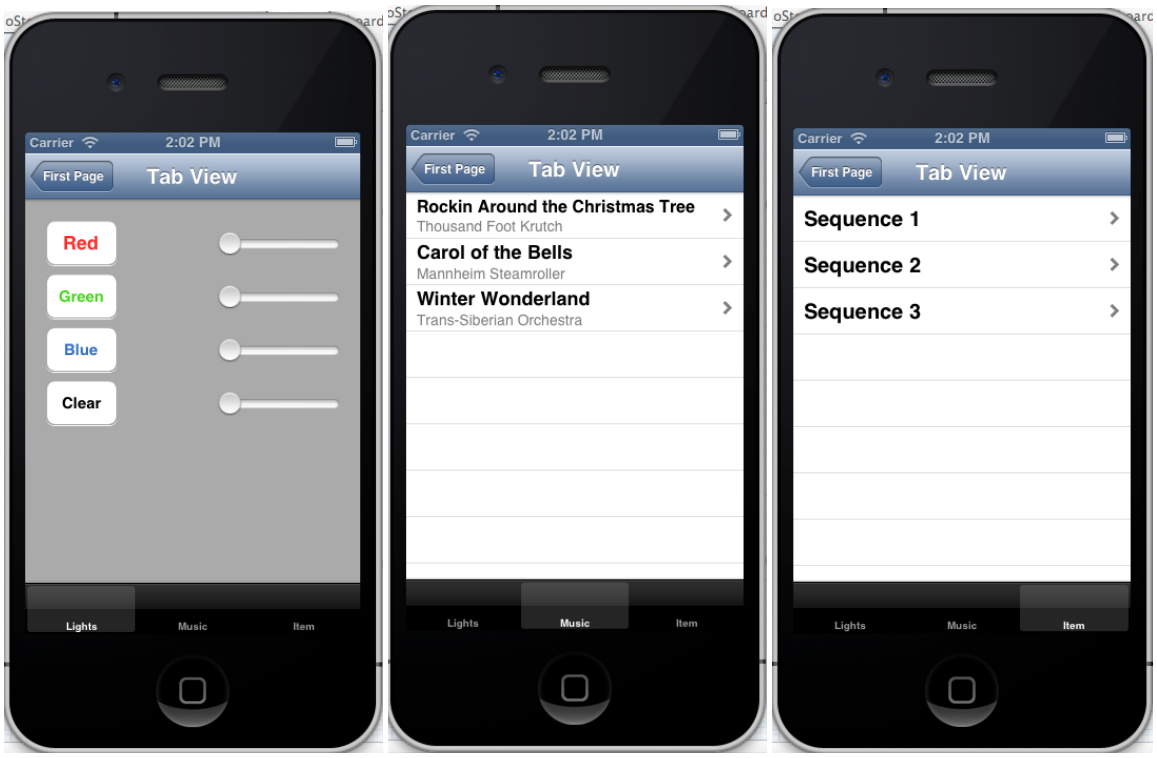


Fig. 3,4,5: Lights tab, Music tab, and Sequences tab

Christmas Light Controller Progress
Austin Wentz

Considerable progress has been made on the hardware front. The Renard 64XC and the 8 SSRez's are now soldered and thoroughly tested. In total, the soldering took 20-25 hours. Testing took another 5 hours to complete. With the hardware assembled, I put together a simple prototype which turns lights on and off using a predefined sequence. Several short videos are available to demo the prototype.

Display Case

I have also been working on a design for a display case which houses the hardware. The dimensions of the case will be 16.5 inches x 16.5 inches x 12 inches. Here are some initial requirements for the case:

- Safety features – Renard 64XC and SSRez's will only be powered when lid is closed.
- Locking mechanism to prevent theft
- Made of acrylic
- Fan for keeping SSR's cool
- Cord management

III.3 Sprint 3 Progress Report

Sprint 3 Report

Team Members:	Austin Wentz and Jordan Doell
Date:	December 7, 2012
Class:	Senior Design
Subject:	Sprint 3 Report
Sponsor:	L-3: June Alexander-Knight

Backlog

Completed

- Put Christmas lights on house for demo
- Film demo of Christmas lights blinking in sync to music
- Design display case for electronic components
- Have the display case made and assembled (went with pre-assembled case)

Remaining

- Program and configure Raspberry Pi to act as midi sequencer for lights
- Develop and implement iPhone app which controls the Christmas lights

Christmas Light Controller Progress

Austin Wentz

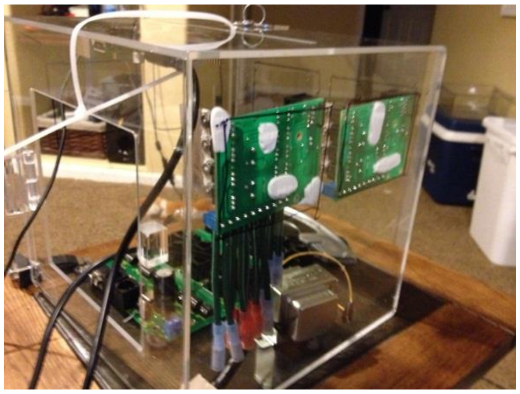
We made substantial progress on the Christmas light controller during Sprint 3. A display case was purchased to house the embedded hardware, I put up Christmas lights on my house, and a demo was filmed of the Christmas lights blinking in sync with music.

Display Case

Designing a display case and having it assembled was taking more time than originally anticipated, so we went with a premade temporary solution. An 8” x 8” x 8” acrylic display case, originally designed to be a ballot box, was purchased and modified. Before and after photos are shown below:

Before

After



Christmas Lights

iPhone App

Jordan Doell

Prototype

The prototype app GUI was primarily made during sprint 2, but some of sprint 3 was spent researching some more of how we want the app to look. Hopefully during Christmas break, some more progress will be made.

Also, more time was spent learning a little more about iOS. I'm still getting through the podcast class, so hopefully the app will begin gaining some functionality soon. James is putting the final touches on his framework, so as soon as he gets that finished, that will be put into the app as well. The communication between the iPhone and base station will probably be the biggest challenge to get working.

III.4 Sprint 4 Progress Report

Sprint 4 Report

Team Members:	Austin Wentz and Jordan Doell
Date:	February 8, 2013
Class:	Senior Design
Subject:	Sprint 4 Report
Sponsor:	L-3: June Alexander-Knight

Backlog

Completed

- Configure EC2 server to act as middleman between iPhone app and xmas lights
- Develop RESTful web service to allow iPhone to send commands and Raspberry Pi to get commands
- Begin connecting iOS framework to UI

Remaining

- Develop and implement iPhone app to interface with controller to control lights and music
- Program and configure Raspberry Pi to act as midi sequencer for lights
- Implement JSON-RPC on Raspberry Pi
- Implement JSON-RPC on EC2 server
- Connect iOS framework to UI
- send JSON from iPhone to server

Christmas Light Controller Progress

Austin Wentz

Configuring EC2 Server as "Middleman"

To avoid issues with firewalls, network configuration, and etc. we decided to use a server as a middleman for communication between the iPhone and the Raspberry Pi. We went with an Amazon EC2 server running Ubuntu. When the user wishes to send or receive information, the iOS app will send notifications or requests to the EC2 server. The Raspberry Pi will run an application which will periodically query the EC2 server for any new information. Figure 1 shows a diagram of this configuration.

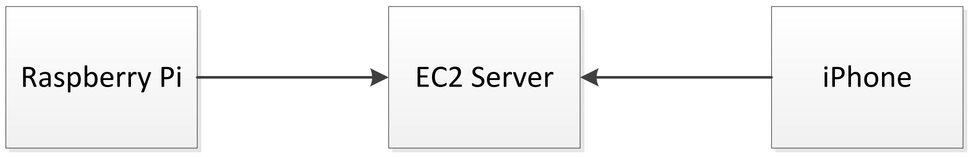


Figure 3

Define Interactive Lighting JSON-RPC Interface

JSON-RPC is a remote procedure call protocol encoded in JSON. It is a very simple protocol, defining only a handful of data types and commands. JSON-RPC allows for notifications (info sent to the server that does not require a response) and for multiple calls to be sent to the server which may be answered out of order.

JSON-RPC will be used to send commands to the lights and to retrieve lists of songs and light sequences available to be played. The interface is defined as follows:

```
//play the song identified by songId
//json example: {"jsonrpc": "2.0", "method": "playMusic", "params": [3]}
void playMusic (int songId)

//run the light sequence identified by lightId
//json example: {"jsonrpc": "2.0", "method": "runLights", "params": [2]}
void runLights (int lightId)

//play the song identified by songId and run the light sequence identified by lightId
//json example: {"jsonrpc": "2.0", "method": "playMusicWithLights", "params": [3,2]}
void playMusicWithLights (int songId, int lightId)

//request a list of valid songs and songId's
//json example: {"jsonrpc": "2.0", "method": "getMusicList", "params": [], "id": 1}
char* getMusicList ()

//request a list of valid light sequences and lightId's
//json example: {"jsonrpc": "2.0", "method": "getLightList", "params": [], "id": 2}
```

```
char* getLightList ()

//request a list of valid song/light combinations
//json example: {"jsonrpc": "2.0", "method": "getMusicLightList", "params": [], "id": 3}
char* getMusicLightList ()
```

Developing RESTful Web Service

A web application/service is now needed to run on the EC2 server. The service needs to perform two main tasks: accept new information from the iOS application and allow the Raspberry Pi to retrieve this information. This is done through a RESTful web service.

iPhone App Progress

Jordan Doell

I got the framework from James. Now I am beginning to connect the framework to the user interface I had before. Once I get that done, I can start trying to send JSON over to the server that the Raspberry Pi will talk to.

Most of my time spent during this sprint was looking through the framework we got from James. I had to go through and try to understand what all is going on and how to use it. Now I am beginning to connect up the interface. Also, I have been researching about JSON. I haven't used JSON before so I wanted to be a little more familiar with it. As soon as the interface is done, I can start trying to send some JSON data to the server that Austin has been working on.

Meetings

Since the sprint started, we have met a few times after Senior Design class on Tuesdays and Thursdays. Jordan also met with James and Josh when he got the framework and James described it briefly.

III.5 Sprint 5 Progress Report

Sprint 5 Report

Team Members:	Austin Wentz and Jordan Doell
Date:	March 15, 2013
Class:	Senior Design
Subject:	Sprint 4 Report
Sponsor:	L-3: June Alexander-Knight

Backlog

Completed

- Implement client code on Raspberry Pi/laptop
- Refine RESTful web service on EC2 server
- Send JSON from iPhone to server
- Connect iOS framework to UI

Remaining

- Test web service
- Test client code

Christmas Light Controller Progress

Austin Wentz

EC2 Web Service Implementation

For the web service, we are using Flask which is a microframework for Python web development. Song and light sequencing information is stored in a sqlite database. Retrieval and adding/updating information is done through GET and POST commands. For example, to retrieve a list of available songs to play, just send a GET request at the /songs URL.

Client Application Implementation

The client application is implemented in Python also. The application has several different modes. One mode is to change the brightness of the lights based upon values on the web service. The values can be modified via mobile devices. Another mode is to receive commands to play songs and/or light sequences.

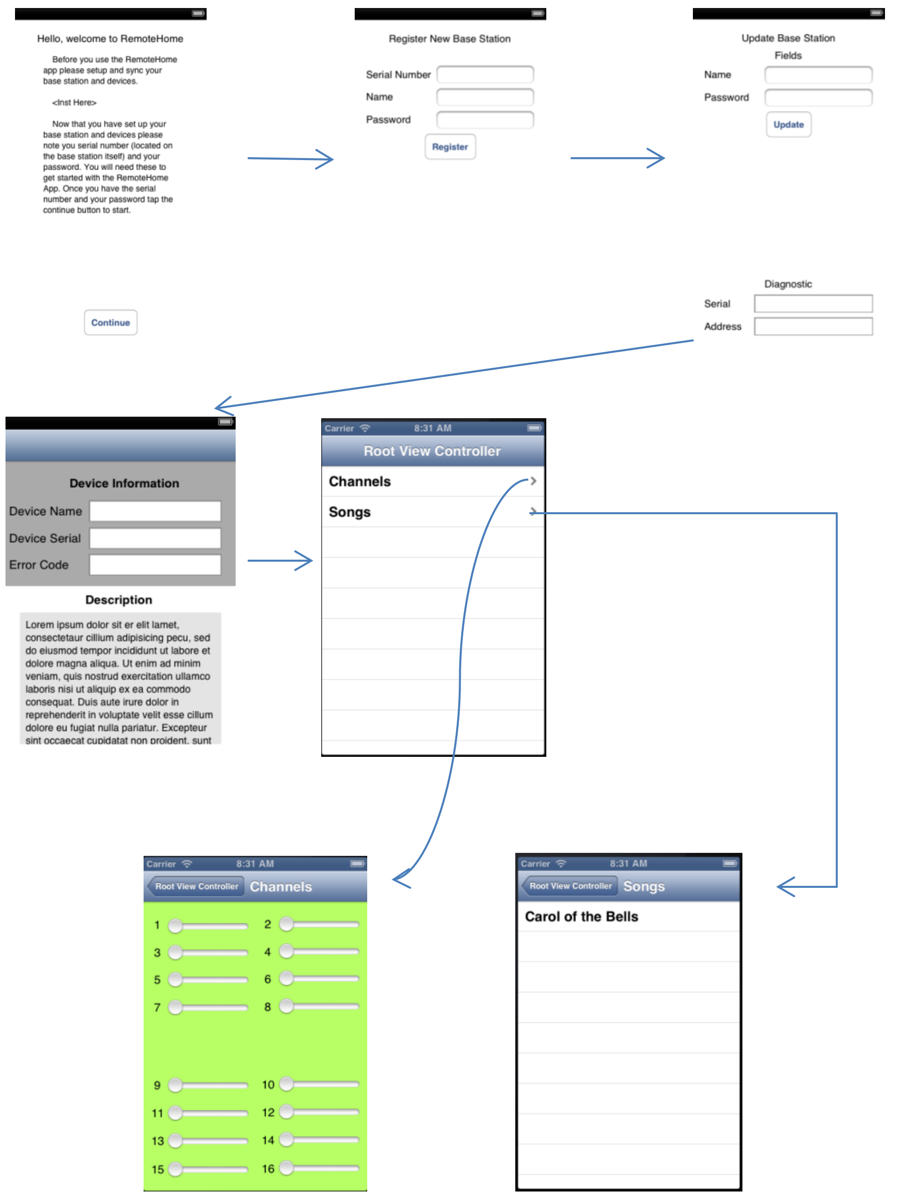
iPhone App Progress

Jordan Doell

For the app, I have included Figure 1 below. I updated the layouts a little bit to simplify things. I am still working on getting the app to flow from James' framework to my interface that I have made. Figure 1 shows what it should eventually look like. All of the first 4 views won't have to be brought up each time the app starts up once the base station gets added initially. The main view with channels and songs is pretty self-explanatory. Once in the channel view, it should send data to the server when a slider has been changed. It will send an array of all the sliders values to the server. For the songs, they will be hard coded for now. When a song is tapped, it should send data to the server letting it know which song to begin playing.

I am planning to meet with James or Josh next week to hopefully get everything figured out since I am a little behind. Most of my time this sprint has gone to researching how to accomplish what I need to do in Xcode. Austin has been working on the JSON data for me to send to the server, so once he gets that figured out and I get the layouts connected, hopefully we can begin some testing.

Figure 1:



Hello, welcome to RemoteHome

Before you use the RemoteHome app please setup and sync your base station and devices.

<Inst Here>

Now that you have set up your base station and devices please note you serial number (located on the base station itself) and your password. You will need these to get started with the RemoteHome App. Once you have the serial number and your password tap the continue button to start.

Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt

III.6 Sprint 6 Progress Report

Sprint 6 Report

Team Members:	Austin Wentz and Jordan Doell
Date:	April 12, 2013
Class:	Senior Design
Subject:	Sprint 6 Report
Sponsor:	L-3: June Alexander-Knight

Backlog

Completed

- Some Testing
- Presequenced the songs and made midi files with Vixen
- Finished integrating lights into XBMC
- Created display for Design Fair

Remaining

- Testing

Christmas Light Controller Progress

Austin Wentz

During sprint 6 I finished integrating the light sequencing into XBMC. Sequences and their associated songs can now be played within XBMC. Since we are using Vixen to create the sequences, I wrote a utility program that converts the sequence from Vixen's format into our own format used within our plugin for XBMC.

iPhone App Progress

Jordan Doell

I changed up the UI for the app in between sprints 5 and 6. Also, it sends out commands to the server now. This uses the JSON data format. The storyboard is shown in figure 1.

Most of the other time spent during this spring was for some testing and preparation for the design fair. This included figuring out what to do for the poster and designing it. The poster is almost complete. There are a few changes that need to be made but that shouldn't take too long. Also, we are still figuring out how we want the design to look at the fair, like where the lights should go and how to organize our booth.

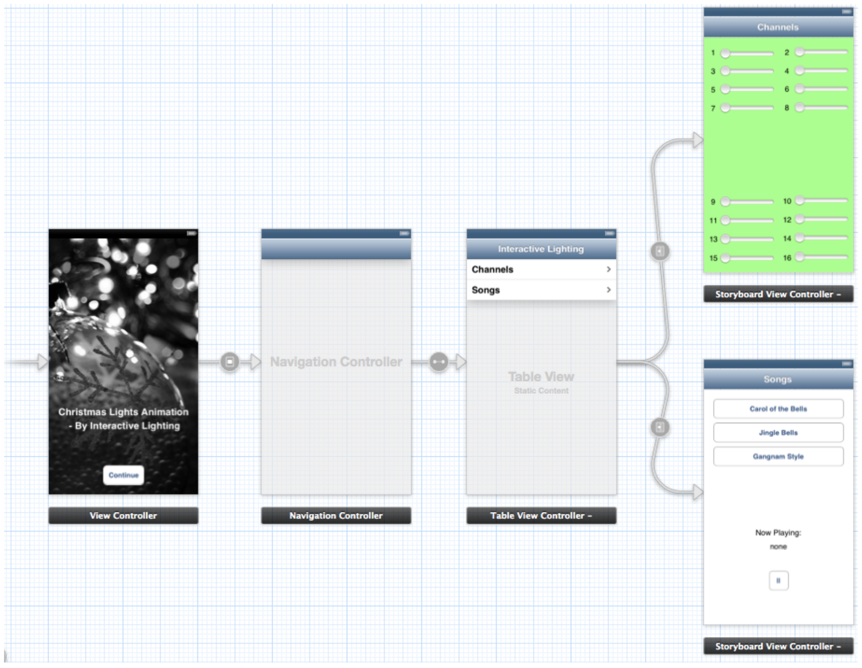

Channels
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Storyboard View Controller -
Christmas Lights Animation
- By Interactive Lighting
Continue
View Controller
Navigation Controller
Navigation Controller
Interactive Lighting
Channels
Songs
Table View
Static Content
Table View Controller -
Songs
Carol of the Bells
Jingle Bells
Gangnam Style
Now Playing:
none
Storyboard View Controller -

Figure 1: