

Dibs *Light Links*

Group 35

Matthew Bolan
Evan Boyar
Nicholas Cain
Ronaldus Woods

Fall15

12/10/2015

Sponsors: Boeing/Leidos



Table of Contents

1. Executive Summary	1
2. Project Description	2
2.1. Motivation and Goals	3
2.2. Objectives	4
2.3.1. Requirement Specifications.....	6
2.3.2. Related Standards	7
3. Research.....	7
3.1. Existing Similar Projects and Products	8
3.2. Relevant Technology	11
4. Realistic Design Constraints	17
4.1. Door Links fitting under doors.....	17
4.2. Multiple issues with obstacle detection.....	17
4.3. False positive detection	18
4.4. Issues with Wifi stability	18
4.5. User Interface issues	18
4.6. 120 pages.....	19
4.7. Physical component issues	19
4.8. Schedule Conflicts	19
5. Communication Standards.....	20
5.1. Wireless.....	20
5.1.1. Wifi.....	20
5.1.2. Bluetooth.....	21
5.2. Wired	21
6. Microcontrollers.....	22
6.1. Types.....	22
6.2. Tasks	24
6.2.1. Lighting.....	25
6.2.2. Modes of Operation.....	25

7. LED System	29
7.1. LED Bank	30
7.2. LED Control Unit.....	31
8. Human Presence Sensor	35
8.1. Function.....	35
8.2. Limitations	35
8.3. Accommodation of Limitations	36
8.4. Software Tracking.....	38
9. Door State Sensor.....	40
10. Obstacle Presence Sensor.....	44
10.1. Types of Sensors.....	44
10.2. Application of Sensor.....	45
11. Power	45
11.1. Demands of Components	45
11.2. Transmission	47
11.2.1. Requirements of Transmission.....	47
11.2.2. Connector Types.....	52
11.3. Energy Storage and Power Supply	53
11.3.1 Energy Storage Technology.....	53
11.3.2. Rechargeable Battery Types.....	53
11.3.3. Charging Circuits.....	55
12. Feature Set	56
12.1. Personal Accounts.....	56
12.1.1. Device Registration	56
12.1.2. Wireless Notifications.....	56
12.1.3. Visual Notifications.....	56
12.1.4. Nonlocal System Status	57
12.2. Programmable Lights.....	57
12.2.1. Ambient Lighting	57

12.2.2. Timed Lighting.....	57
12.2.3. Sensor Triggered Lighting.....	57
12.2.4. Colored Lighting.....	58
12.3. Automatic Link Ordering.....	58
12.4. Door State Sensor.....	60
12.5. Location Tracking.....	62
12.6. Security Applications.....	63
12.7. Outer Casing.....	64
12.7.1. Crush Resistance.....	64
12.7.2. Water Resistance.....	64
12.7.3. Lightweight.....	65
12.7.4. Easy to Relocate.....	65
12.7.5. Storage.....	65
12.7.6. Conclusion.....	66
13. Physical Design.....	66
13.2. PCB Layouts.....	67
13.3. Functionality Diagrams.....	70
13.4. Design Philosophy.....	73
13.5. Door Link.....	74
13.5.1. Requirements.....	74
14. Application Software.....	75
14.1. Types of Mobile Operating Systems.....	76
14.1.1. Apple IOS.....	76
14.1.2. Google Android OS.....	76
14.1.3. BlackBerry OS.....	77
14.1.4. Ubuntu Touch OS.....	78
14.1.5. OS Selected for Dibs LightLinks System.....	78
14.2. User Interface.....	78
14.2.1. Home Screen.....	79

14.2.2. Configure Screen	81
14.2.3. Settings Screen.....	83
14.2.4. Link/Chain Viewing Screen	85
14.2.5. Mode Screens	86
14.3. Functionality.....	88
14.3.1. Connecting to Dibs LightLinks System Through BlueTooth	89
14.3.2. Navigating Between Modes.....	89
14.3.3. Configuring Modes	90
14.3.4. Viewing and Adding Links/Chains	90
14.3.5. Editing Links.....	91
14.3.6. User Account Information.....	91
14.3.7. Changing Colors of Lighted Pathway	92
14.3.8. Security Functionality	92
14.4. Class Diagram	93
14.5. Communication with Microcontroller	94
15. Server Software.....	96
15.1. Amazon Web Services	96
15.1.1. Cloud Services	96
15.1.2. IoT Services	96
15.1.3. Email Services	97
15.1.4. Personalized Accounts.....	97
15.1.5. Interaction with Android Technology	97
15.2. Building the Dibs LightLinks Server	98
15.2.1. Preliminary Stages of Web Application Development	98
15.2.2. Creating a Bucket for the Web Application.....	119
15.2.3. Creating an Application Server.....	120
15.3. Communication to Microcontroller	131
16. Arduino/Atmega328 Software	131
16.1. Libraries.....	131

16.1.1. Digital I/O	131
16.1.2. Analog I/O	131
16.1.3. Wire.....	132
16.1.4. Serial.....	134
16.1.5. Interface Between Hardware and Libraries	134
16.2. Bootloader	137
17. Project Testing	137
17.1. Hardware	137
17.1.1. Regulated Voltages.....	137
17.1.2. Water Resistance.....	138
17.1.3. Coefficient of Static Friction.....	138
17.1.4. Signal Transmission.....	138
17.1.5. Human Presence Sensor	138
17.1.6. Door State Sensor.....	139
17.2. Software	139
17.2.1 Android Software Testing.....	139
18. Administrative Content	141
19. Future Applications.....	144
20. Conclusion	146
21. Appendix	148
21.1. Permissions	150
21.1.1. PuTTY.....	150
21.1.2. Amazon.....	152

1. Executive Summary

When children are frightened of the dark, they sometimes need to sleep with a nightlight. A night light soothes and grants comfort as it permits them to partially see in the darkness. As people grow older, this fear often goes away, but usually their inability to see in the dark does not. People accept that they cannot see and the need for a night light diminishes. Yet, they still have nighttime activities for which they need light: waking up in the middle of the night with need to use the restroom; turning off the lights before crossing a room or going upstairs; getting prepared for early morning work hours. Often people need a light to pave their way through the darkness. Providing a method of simplistic low level lighting would greatly assist nighttime activity.

To address the problems, a low-level lighting system is proposed that can be used for direction in the dark. The Dibs LightLinks System is a series of blocks with interconnected capabilities. The system is designed to output low level lighting across all connected units with intent to provide a pathway for the user to follow. In addition, sensors inside the devices will detect obstacles in the user's path.

In addition to providing light during nighttime, the Dibs LightLinks System also provides security. According to crimeinamerica.net [1], approximately 72% of all burglaries in America occur while a household member is not at home. The Dibs LightLinks System provides a layer of security in the case a break-in is detected. Using sensors inside the device, the system can passively detect human presence.

The Dibs LightLinks System is composed of three flavors of links and a Link Connector. The three flavors are Door Links, Interface Links, and Standard Links. Standard Links are the most prominent. They contain LEDs and a PIR sensor. Door Links are constructed specifically for finding the position of doors. They are different from other links in that they are a connector linked to a small device that contains a proximity infrared sensor. Interface Links are designed to receive the inputs from the user and inform the rest of the system of what actions to take. They contain the components of the Standard Links in addition to Bluetooth compatibility. Lastly, the Link Connectors are simply m/m wires that connect each Link to the next one.

In addition to the three flavors, the Dibs LightLinks System has 6 built-in modes of operation. The modes of operation are Standard Pathway Mode, Power Saving Pathway Mode, Party Mode, Crowd Mode, Security Mode, and Sleep Mode. When in Standard Pathway Mode, the system lights up, providing the user vision of their path. Power Saving Pathway Mode is similar to the Standard Pathway Mode in function, but detects the user's presence and shuts off lights behind the user's position so as to save

power. Party Mode provides entertainment to the user by switching between colors to provide ambient lighting. Crowd Mode is designed for crowd control, providing the illusion of movement to help direct people in a crowd. Security Mode provides a type of security in lighting up links if human presence is detected and sending a message to the bluetooth module, informing the user of a potential break-in. Lastly, Sleep Mode is the systems off-mode, remaining passive until the user requests otherwise.

The Dibs LightLinks System is designed to be set up in a “Chain” path structure. Each Chain should be set up to be approximately identical and should contain at least one Interface Link. The user should set up the Links to lead them to a predetermined destination. When the user requests the system switch modes, the functionality of the path should change.

The system’s software provides the user with several customizability options. The user control panel comes in the form of an app for the user’s smart phone. The customizability options include various timers for the different modes’ functionalities, pathway lighting color, security lighting color, party color scheme, and various additional mode-dependent options.

The app is designed to be user friendly, granting them control over the entire system, rather than individual links. The default screen enables to user to easily change between modes by tapping their requested mode. If attempting to switch to a security mode, a confirmation popup is displayed that the user must confirm.

Ultimately, the device will offer a wide variety of commercial use and the customizability for the user to make it perform as they need it. If the user so requires it, the device could be used solely for its original purpose, to light up and provide a path at night. If, however, the user would like more security capabilities and the option to alternate the lights for ambience, they can use the device to its full potential.

2. Project Description

When it comes to developing new technologies, it’s virtually impossible to designing a completely new and never-before-seen technology. Every product is the result of millennia of scientific advances and engineers standing on the shoulders of the giants before them. The Dibs LightLinks system is no different. Technologies of the past and present contributed to the motivation for its design and provide guidelines of which to apply.

2.1. Motivation and Goals

Like all projects, the Dibs LightLinks System was the product of standing on the shoulders of giants. In order to think of such a device, motivation and enlightenment were required. In order to streamline it, to bring it to reality, goals had to be set. These goals would grant vision towards a final product.

2.1.1. Motivation:

Utility and versatility are two driving characteristics when designing projects. A prime project would be to provide a tool that can be used in multiple ways, from simple daily activities to more complex occasions. When thinking about potential projects, the LED path designs located in airplanes and theaters were brought to light. The potential for a more versatile system for households or businesses could prove highly beneficial.

The design team's initial plans were to construct very simple interconnecting light blocks that lit up when a button was pressed and turned off after it was pressed again. The original discussion was waking up at night and needing to use the restroom, but not wanting to turn on the light to avoid disturbing others. As the design team brainstormed, they added other features and interfaces, such as obstruction detection to assist the individual in avoiding obstacles and motion detection to help conserve energy when no one is in between the links.

The project has since evolved into a more complex system with greater utility. The name of Dibs was decided upon after the original four different flavors of links we started with, Door, Interface, Battery/Power (which was removed during production), and Standard. The design team added more features to the device, to include Bluetooth capabilities for the user to use an app and to inform them if human presence is detected while in security mode. The project was beginning to flesh itself out.

2.1.2. Goals:

Ultimately, the goal is to develop a device that can be placed down as a makeshift pathway that lights up when the user activates it. The device should come in multiple versions, or flavors, but each one should have the same overarching functionality.

Primary Goals:

- Low level lighting device
- Lightweight and portable

- Safety

Secondary goals:

- Motion detection for security and detection of user location
- Door Position Detection
- Multicolored lights
- Bluetooth
- Multiple Modes of Operation
- Easy to use
- Low-power
- Aesthetically pleasing
- Easy to relocate
- Lightweight
- Crush Resistant

2.2. Objectives

The overall goal of Dibs LightLinks is to take existing technologies and create something new and exciting. To be more specific, the Dibs LightLinks system, plans to achieve this by reaching all the goals listed in **section 2.1**. There are a lot of goals that need to be achieved, but if done so, Dibs LightLinks are sure to be a success in the real world. In order to reach these goals, certain objectives were set. Throughout the course of the semester, there were some deadlines that needed to be met which were the objectives of this semester.

2.2.1. Senior Design Project Idea Assignment

Due August 28

This was the first objective that needed to be met. Coming up with an idea took time for research. Each group member came up with their own individual idea. This allowed brainstorming from each member of the group. Throwing out ideas helped come up with the final idea that became Dibs LightLinks.

The first idea that was agreed on wasn't the Dibs, however. The original plan was to construct a pair of suits. One suit would act as an input and the other as an output. Any movement the input suit would make, the output suit would respond in turn by applying electrical impulses to the muscles of the wearer. This idea, however, was scrapped due to the regulations on human testing for Senior Design projects.

2.2.2. Group Identification Form

Due September 3

This was needed to make sure group members were set in stone. Some trouble arose as the team ended up being originally composed of 5 members, which was not permitted. After lengthy discussion, one member finally realized another group was developing a device that better suited his interest and broke off from the group. After this, the team was immediately set and stone and the document was written and turned in. Once it was turned in, deep discussions came about the development of the overall design and how it both should perform and could perform.

2.2.3. Initial Document - Divide and Conquer **Due September 15**

The Divide and Conquer document helped come up with the specifics of the Dibs LightLinks system. This helped give a general idea of the budget needed, constraints, and parts needed for the design. This also set up a rough plan for how the work needed to be divided up and how the final document would reach the required page count. Further discussion decided on what needed to be in the final project and what would attempt to be implemented in the final project.

2.2.4. Boeing/Leidos Proposals **Due September 24**

This document was a proposal to Boeing/Leidos asking for sponsorship. The proposal contained a brief project description, project specifications and standards, and a detailed budget. Without this, finding funding for the project would require additional effort and also time. After submission, it wasn't long before the project was approved and \$533.74 were provided for project expenses.

2.2.5. Table of Contents Submission **Due October 15**

This document set the path for what was to be expected in the overall Senior Design 1 Document. Careful planning had to be done to meet all the requirements needed for the document and also for how these requirements would be divided up between members to maximize group participation and project efficiency.

2.2.6. Current Draft of Senior Design 1 Document **Due November 12**

The draft document was important for two reasons. The first being that it helped prepare for the final document by getting a good start on the documentation. The second reason of it being so important was the fact that feedback was given by the professor which helped with the process completion of the final document.

For this document, each member was supposed to have at least 15 pages. 3 members didn't even have 10. This made the team realize how far behind they were and stimulated the motivation to work harder and ensure the final paper would be completed within the allotted time frame.

2.2.7. Senior Design 1 Document

Due December 4th

The draft document's final, 120 minimum page design was due at the end of the semester. Due to last minute editing and formatting, the document wasn't turned in on time. The late turn in date was after the weekend and the document was finished in full, edited properly, and turned in upon that day being reached.

2.2.8. Critical Design Review

Due February 12th

The Critical Design Review (CDR) was a presentation given to the Senior Design class to explain our design and progress on our Senior Design project. The presentation was designed to be a preparation for the final presentation.

2.2.9. Final Presentation

Due April 19th

The final presentation was the presentation and demonstration of the Dibs LightLinks System. Here, the final project came together and was presented and demonstrated to an audience of 3. The presentation was an updated adaptation of the presentation used in the CDR. During the demonstrations portion, our project suffered an unexpected malfunction that delayed a proper demonstration to be presented until the following week.

2.3. Specifications and Standards

All projects require specifications. These are the limits of which the device must be capable of performing. Failing to reach these specifications implies physical problems with the system. Additionally, the most optimal way of reaching these specifications is to apply the various standards. Standards are the methodology behind how each system performs relative to others.

2.3.1. Requirement Specifications

In this section, we list our requirement specifications.

- <5 ms turn on delay per link due to detection delay (note: does not apply to Door Links)

- <20 ns signaling propagation delay per link
- <2 s latency with mobile communication
- 2 kg maximum mass
- Each link shall be less than 300 cm by 8 cm by 4 cm, (l, w, h)
- Shall accept and pass on signals sent along any relevant signal lines

Standard Links:

- Shall draw less than 2 W
- Shall sense motion or presence of individual within 2 m

Door Links:

- Shall fit under standard US door (<2.54 cm in height)
- Shall sense open/closed door

Interface Links:

- Shall convert parallel signals to serial signals
- Shall interface with app using Bluetooth

Link Connectors

- Shall relay signals <2 ms

2.3.2. Related Standards

In this section, we list the standards for the technologies we're using.

- Serial Communication - I²C
- Bluetooth communication (802.15.1)
- Internet protocol version four

3. Research

Like all new technologies, the Dibs LightLinks System requires a bit of background research. There will be products that share similar aspects; and their history and design will benefit the construction of the system. Additionally, there will be products that are relevant to the system either by being physically embedded into each Link or simply being used by a device in the system.

3.1. Existing Similar Projects and Products

When developing any technology, there are bound to be similar predecessors or concepts upon which people can build enhanced capabilities or develop a new technology. In developing a lighting-based technology, this is definitely relevant, as artificial light typically has the same function: to provide vision in the dark.

Similar products include the airline ground lighting system (*Figure 3.1.1*) for leading passengers to the exits during an emergency. These features include lights that illuminate all at once when an emergency is declared or when certain conditions are met, such as a loss of power, an emergency door being opened, or any other state of emergency. The functions of these lights is only to turn on. They do not have features such as changing colors or the illusions of moving lights. They have only one color of illumination and one mode of operation.



Figure 3.1.1, Airline Floor Lighting, reprinted in accordance with the Creative Commons License [1]

Pathway lighting (*Figure 3.1.2*) is also used in commercial places, such as hotels and gardens. These are often placed to light up the walk-ways for guests, providing safety and even a peaceful environment. These lights usually turn on, all at once, at dusk and off at dawn. They, too, do not have changing colors and have only one mode of operation: providing lights at night.



Figure 3.1.2, Pathway Lighting, reprinted in accordance with the Creative Commons license [2]

Motion-sensor lights (*Figure 3.1.3*) are widely used for safety purposes. These lights typically use sensors facing the direction of expectant motion and activate when an entity moves within their field of detection. Motion-sensor lighting is found in a wide variety of locations, from public bathroom lights to security lights on houses. However, they only have one purpose, to turn on when triggered.



Figure 3.1.3, Motion Sensor Lights, reprinted in accordance with the Creative Commons license [3]

Strings of colored lights (*Figure 3.1.4*) are often used during the holiday season. They can blink or remain steady. These lights are not triggered by movement. These lights can be set up to “blink” to give an illusion of moving lights. This blinking can be performed by simultaneously turning one light off and its subsequent light on. This illusion of movement can be used in the devices in more entertainment-based modes of operation. These lights, however, aren’t designed to be used as a path and are instead used for the ambience they provide.

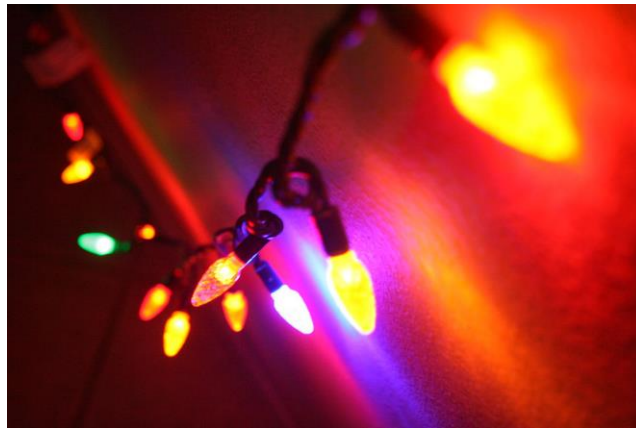


Figure 3.1.4, Holiday Lights, reprinted in accordance with the Creative Commons license [4]

The product most similar in design and usage to the Dibs LightLinks System is the lighted aisle pathway systems in theaters (*Figure 3.1.5*). While these lights are similar to those of airlines and buses, their purpose is less towards emergency evacuation and more towards safe passage to the doorway in the case of someone needing to leave the theater. These lights provide a low-level light that does not interfere with the movie being watched but provides enough light to be followed should someone need to leave. The lighted aisle pathway systems, however, are designed to be permanent fixtures in their environment and provide only one color of light.



Figure 3.1.5, Theater Lighting, reprinted with permission [5]

3.2. Relevant Technology

Similar products can help an individual get ideas and a stepping stool to get started, but they're only for idea expansion. To truly get on the giant's shoulders, one must identify the technology they intend to use. As with any project, it is important to identify the relevant technologies before finding the best possible parts for the project.

3.2.1. RGB LEDs

Light Emitting Diodes (*Figure 3.2.1*), or LEDs for short, are a type of semiconductor that produces light when a certain voltage is applied. The higher the voltage used, the brighter the LED will shine. Using different materials in the semiconductor changes the resulting color of the light.

The LED is a critical component of this system. They permit the devices to provide luminescence in the dark. RGB LEDs will be used as they provide wide gamut of colors which permits multiple modes of operation for the device.



Figure 3.2.1, RGB LED, reprinted in accordance with the Creative Commons license [6]

3.2.2. PIR Sensors

Passive Infrared Sensors (*Figure 3.2.2*), or PIR sensors for short, are useful for motion detection. By observing the infrared spectrum, PIR sensors can detect changes in heat of objects that pass before them. This can be used to detect moving bodies.

These sensors are useful for the security aspect of the system, as they can detect human presence via the person's movement.



Figure 3.2.2, PIR Sensor, reprinted in accordance with the Creative Commons license [7]

3.2.3. Ultrasonic Sensor

In order to detect the location of an obstacle within the path, ultrasonic sensors (*Figure 3.2.3*) will be used. These sensors send out an ultrasonic signal that bounces off of objects. When they receive the sent signal, they use the time it took from send to receive to calculate the distance from the object.

These sensors can be used in order to detect the location of any obstacle in the path of the user and inform them of the object's location by changing the color of the light on the link that detects the object.

The ultrasonic sensor was removed from the final project.

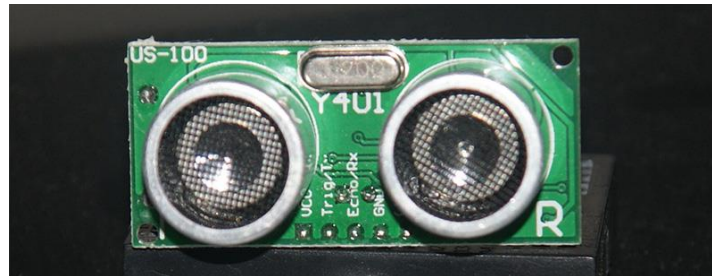


Figure 3.2.3, Ultrasonic Sensor, reprinted in accordance with the Creative Commons license [8]

3.2.4. Proximity Infrared Sensor

The proximity infrared sensor is composed of two infrared diodes (*Figure 3.2.4*) and works similarly to the Ultrasonic Sensor, but uses infrared light instead of ultrasonic sound. This device has less range than the ultrasonic sensor, but is more compact. This means this device can be used to detect closer objects in smaller spaces. These sensors can be used in the door link in order to detect if the door is open or closed.

These sensors will be used on the device for determining the position of doors by the door Links. Being compact and being able to detect object presence make them optimal for sitting under doors.



Figure 3.2.4, Infrared Diode, reprinted in accordance with the Creative Commons license [9]

3.2.5. Wifi

Wifi is a local wireless networking technology that uses microwaves in order to communicate with multiple devices within a small area. Due to the use of microwaves, this technology can be used to communicate through walls. Devices usually connect to Wifi using antenna (*Figure 3.2.5*) that transmit and receive signals.

In order to communicate with the user, Wifi will be a necessary technology. The need to be able to receive user requests and send security information over the internet requires the use of Wifi. The system will connect to Wifi using the Addicore ESP-8266.

WiFi was removed from the final design.



Figure 3.2.5, Wifi Antenna, reprinted in accordance with the Creative Commons license [10]

3.2.6. PVC

Polyvinyl Chloride (*Figure 3.2.7*), or PVC for short, is a plastic polymer with a wide variety of commercial use. From rigid piping systems to flexible wiring, PVC offers a cheap, lightweight, and effective coating to a device or system.

In this product, both flexible and rigid PVC will be used to contain the various components. PVC was chosen over other materials due to various properties being required in the outer casing that will be discussed in section 12.7.



Figure 3.2.7, PVC, reprinted in accordance with the Creative Commons license [11]

3.2.7. Server Technology

Server technology is broken into two sections, software and hardware. They are designed to store and share information, usually on a host to client model. The software is technically the server and it is a program that provides information to responses to other programs from a client. The hardware is the physical device running the server, and is typically referred to as a server even though it may be used for other means.

A server provided by Amazon will be used in order to receive information from a client and send responses back. Essentially, the server will receive the user input from a smartphone app and will send the response to the LightLinks system. The server will also be able to receive signals from the LightLinks system in the case of a security mode being active and send relay the response to the user's smartphone.

Server technology was removed from the final design.

3.2.8. Microcontrollers

Microcontrollers (*Figure 3.2.9*) are used in almost every single electrical device today. They are required for almost every type of user interfacing. Microcontrollers receive an input from the user and output a signal based off of their programming.

Microcontrollers will be very important to this system. Because some users will be connecting to the device through the internet, two microcontrollers will be used. One microcontroller, the Addicore ESP-8266, will be used to connect the system to the local

Wifi. The other microcontroller, the Atmega328P-PU, will control how the rest of the system responds to a given input.



Figure 3.2.8 Microcontroller, reprinted in accordance with the Creative Commons license [12]

3.2.9. Email

Email is a method of information sharing. With humble beginnings as a messaging system to different users of the same computer back in the 1960s, its current name was coined in 1993 when it became a global standard. Modern versions of email work similarly to normal mail; one individual sends a message that another receives and can fetch from their online mailbox. Several popular emailing services include Gmail, Hotmail, AOL, and Yahoo Mail, however, many organizations, companies, and colleges have their own specialized mailing servers.

Email provides an alternate means of the system communicating with the user. While it's not the primary method of communication, it offers a simple, yet effective, alternative.

Email was not used in the final design.

3.2.10. Mobile Technology

Back in 1875, Alexander Graham Bell invented the telegraph. One year later, he invented the telephone, which would go down to revolutionize communications. Since then the telephone has been improved upon drastically, and in 1993, the first smartphone, IBM's Simon, was introduced to the world. Smartphones have gained more and more popularity throughout the years, with 90% of adults in America owning a smartphone in 2014 [13].

Smartphones differentiate themselves from normal phones because they not only provide phone services, but also utilize a PC-like operating system. In additions, smartphones usually offer other features such as a touch screen, applications (known as Apps), and media players. Popular smartphone devices include the iPhone and the Android.

Smartphone technology provides an interesting method of user interfacing. Instead of designing a potentially money-inefficient device to go with the system, an app the average smartphone user can interact with will be designed. A program will be written for Android because it uses Java and can be used by a lot of devices, making it available to a wider audience.

3.2.11. PRESS

A previous (Spring 2015) senior design project had qualities similar to ours. This was the PRESS, or Pressure Reactive Stepping Stones project. PRESS involves a series of pressure-triggered devices used to alert users of activity. Additionally, they lit up in various ways, which is similar to our project.

4. Realistic Design Constraints

Just like in any project, there are a number of constraints that have hindered developmental progress. These constraints range from physically unrealizable factors, to potentially difficulties the device may face in commercial use, to difficulties in writing the paper itself.

4.1. Door Links fitting under doors

In the very first discussion, the point was made that the system needed to be able to pass under a door. This was when it was decided to have specialized Links for different jobs. The Link designed for fitting under the door needed to be small enough to fit under the average American household door, but large enough to be able to relay signals and detect if the door is open or closed. As of the original writing of this paper, the design and size have not been finalized; however, a basic guideline is being followed.

4.2. Multiple issues with obstacle detection

Of all the sensors, the obstacle detection sensor had the most issues. Laser tripwires required static device that were parallel, proximity infrared sensors didn't have the range, and ultrasonic sensor, while being technically able to detect obstacles, has many

problems in its own right. Because the two lines of Links would not keep consistent distance from each other, wouldn't always be completely parallel, and would have different distances depending on how the user chose to set them up, obstacle detection would be an issue. In addition to this, because the system provides light, there may be no need for obstacle detection.

4.3. False positive detection

False positives are always a problem with sensors. A false positive on the proximity infrared sensors would lead to the door being declared as being in the wrong state. A false positive of the obstacle detection sensor would lead to the system informing the user of an obstacle that wasn't there. A false positive of the PIR sensors would be the worst case, informing the user, and potentially the police, of an intruder where there was none.

4.4. Issues with Wifi stability

Communication through wireless always has limitations due to packets of data being sent into the atmosphere. If the Dibs LightLinks System is being used in a highly populated area where there are many Wifi routers operating in the same frequency range, the signal from the user's Wifi router can be greatly decreased. If some data is lost in transmission, problems will arise with the entire Dibs LightLinks System. Switching between modes too quickly could cause the system to produce errors, due to transmission time needed for Wifi. To address the problems of Wifi stability, acknowledgments will be used to ensure communication works to its best capability.

Wifi was removed from the final project.

4.5. User Interface issues

While developing the mobile application, there were many drawbacks that happened during the process. Creating a user interface for a mobile application in Android takes many steps and a bundle of knowledge on the various layouts available for creating such applications. Even with this knowledge, problems arise. Android Studio provides an emulator that was used for developing the user interface. One problem that arose from this is that the user interface may be altered once actually ran on a real device. Some functionality is not available on the emulator such as long pressing items and scrolling list views. Another problem comes with components not showing the way they are supposed to. For instance, the Home Screen had this issue. In the design screen in Android Studio, the screen looked perfectly fine; but when the application was run, the

labels for each mode type on the screen were misaligned and caused the screen to look messed up and choppy instead of the smooth look it was supposed to have.

4.6. 120 pages

One of the larger problems presented is the page count required for this document. The original design was simple, streamlined, and commercializable. What was designed was a simple system that activated when a button was pressed, providing low-level light along a path, and deactivated when the button was pressed again. This, however, was too simple to fill up the required page count for the Senior Design class. In order to counteract this, it was decided to add several attachments and capabilities in order to add more space to fill, eventually ending up with a system that connects to Wifi to interact with a phone app, has 3 different sensor types, and 7 different modes of operation. This truly strained and stretched the original idea, turning what was once a very simple, yet versatile device into a hulking cluster of attachments and possibilities.

4.7. Physical component issues

Over the course of the semester several components that were purchased broke down or didn't work properly. One of the most influential pieces that didn't work was the ESP8266 Wifi module. Due to its erratic nature, the device had to be scrapped as it proved impossible to reliably connect to local Wifi. This led to the complete removal of Wifi from the project. Another problem arose towards the end of development when I2C would not work properly. It was found that the wires used had a high capacitance and sending the information signal down many Links led to a lost signal. Due to this issue, a new design using a data bus was implemented that limited information transfer.

4.8. Schedule Conflicts

Due to the construction of this device being for a class and not a job or entertainment, scheduling conflicts arose between the members of the team. Some members had jobs that kept them from meeting on certain days and all members had classes that limited their available time during other days. Additionally, various other personal matters arose for all the members over the course of the semester that led to unexpected delays or absences from group meetings.

5. Communication Standards

The Dibs LightLinks System will incorporate 2 main modes of communication, wired communication, and wireless communication. With both types of communication, each type has a certain standard to uphold when that type of communication is implemented in each type of technology. These are the following standards of communication that the Dibs LightLinks System follows in its implementation.

5.1. Wireless

The Dibs LightLinks will be using wireless connections to allow outside interaction from other devices such as smartphones and computers. Two major wireless communications that have been considered for the Dibs LightLinks System are Bluetooth and Wifi wireless communications. Both Bluetooth and Wifi communications have different standards to follow under the IEEE standards of wireless communication. the Dibs LightLinks System will follow the standards associated with the respective type of wireless communication.

5.1.1. Wifi

One of the most common standards for implementing wireless local area network (WLAN) computer communication in the 2.4, 3.6, 5, and 60 GHz frequency bands is the IEEE 802.11 standard. The Addicore ESP8266 Wifi module is compatible with 802.11b, 802.11g, and 802.11n modes of transferring data. The Wifi wireless communications are being done in the 802.11n standard because the overall the power consumption of data sent and received in that standard is less than the other compatible options with the ESP8266 Wifi module. The specifications of the ESP8266 Wifi module can be seen below.

Wifi was removed from the final design.

Mode	Typical	Unit
Transmit 802.11b, CCK 1Mbps, $P_{out}=+19.5dBm$	215	mA
Transmit 802.11b, CCK 11Mbps, $P_{OUT}=+18.5dBm$	197	mA
Transmit 802.11g, OFDM 54Mbps, $P_{OUT} =+16dBm$	145	mA
Transmit 802.11n, MCS7, $P_{OUT}=+14dBm$	135	mA

Receive 802.11b, packet length=1024 byte, - 80dBm	60	mA
Receive 802.11g, packet length=1024 byte, - 70dBm	60	mA
Receive 802.11n, packet length=1024 byte, - 65dBm	62	mA
Standby	0.9	mA
Deep Sleep	10	uA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.86	mA
Total shutdown	0.5	uA

Figure 5.1.1.1 ESP8266 Power Consumption [1]

5.1.2. Bluetooth

IEEE developed the 802.15.1 standard that outlines proper implementation for Bluetooth devices. Due to the short range of Bluetooth it was decided that Wifi communications would work better to interact with the Dibs LightLinks System from much further distances instead of just at home uses.

5.2. Wired

The Dibs LightLinks will be using wired communication to relay information between the Atmega328-PU microcontrollers. This will be done using the I²C protocol which is the main wired communication of interest in respect to the project. The way I²C works is by having a single chip (Master) control all other chips (Slaves). This method of communication seemed the best route to go because the Interface Link will be the Link to receive data from the server wirelessly, and then command all the other Links to do whatever the command was given in the packet of data sent. Another reason I²C will be used is because it is fairly simple, needing only two wires to communicate with up to 1008 slaves^[1]. Also acknowledgements are used in this communication method which will be essential to the testing process for detecting errors. A visual representation is given in *Figure 5.2.1* of the connection between a master and slave.

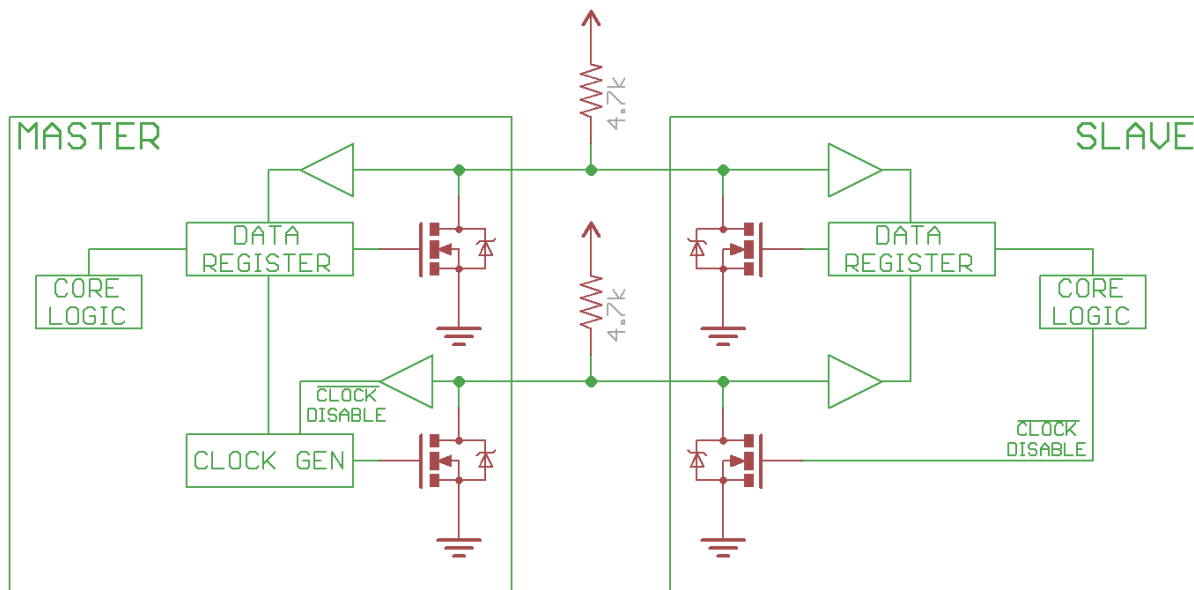


Figure 5.2.1, I^2C Hardware Diagram [1]

6. Microcontrollers

Microcontrollers are everywhere, from cell phones to children's toys. These devices are used anywhere an external source would issue a command or require a change of function. The Dibs LightLinks System is no different, receiving the user input from an app and outputting the result to the system. From bits, to memory, to instruction sets, microcontrollers are very versatile.

6.1. Types

Microcontrollers come in many different types and classifications. A microcontroller can be 8-bit, for very simple operations; 16-bit for more intensive instructions; or 32-bit, for more advanced, and often automatically controlled, devices. Microcontrollers can also utilize memory differently, by either containing embedded memory or using external memory. Microcontrollers can also have different focuses. For example, one microcontroller can be specifically designed to permit the system to access the local Wifi network while another may be used to assist in wired communication.

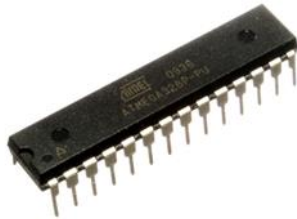


Figure 6.1.1 Atmel328P-PU, reprinted in accordance with the Creative Commons license [1]

For this system, two different microcontrollers will be necessary: one for wireless connection and one to control the lighting systems and sensors. The microcontroller that shall control the system doesn't require too much processing power, so 8-bits shall be fine, if not optimal for the low power consumption. The Microcontroller needs to be able to connect to another microcontroller, control two RGB LEDs, receive signals from sensors, and relay signals to other Links and microcontrollers. More and more specifics can be added to the microcontroller, but there will still be hundreds of options to choose from, so the Atmega328P-PU (*Figure 6.1.1*) microcontroller was decided upon because it satisfied all conditions and is familiar to use.

With the generic microcontroller decided on, only a wireless microcontroller must be determined. This microcontroller doesn't have as many constraints as the Atmega328P-PU, and is therefore more focused on discretion. The microcontroller must be cost efficient, small enough to not take much board space, and most importantly, must be able to connect the Dibs LightLinks System to local wireless networks. Obviously this would leave a lot of choices, so the decision came down to which wireless connection we wanted to implement. The team was in the process of deciding between Wifi communications and Bluetooth communications, as such a popular and affordable microcontroller was picked for both implementations: the Addicore ESP-8266 (*Figure 6.1.2*) for Wifi communications, and the HC-05 (*Figure 6.1.3*) for Bluetooth communications. Both microcontrollers connect other microcontrollers to wireless communications and is small enough to not cause a strain on the board size.

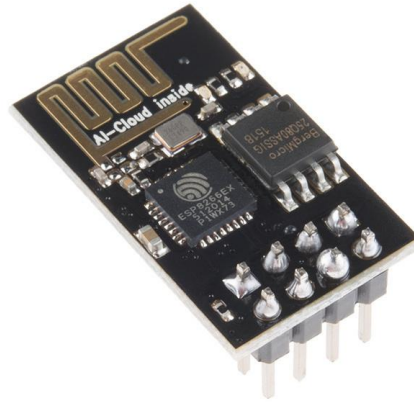


Figure 6.1.2 ESP8266, reprinted in accordance with the Creative Commons license [2]

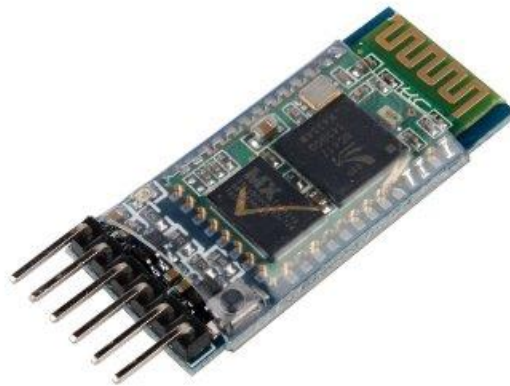


Figure 6.1.3 HC-05, reprinted in accordance with the Creative Commons license [3]

6.2. Tasks

Microcontrollers have a function in any system and that function is to perform tasks. These tasks dictate both how the microcontroller powers certain parts of the system and how the system acts beyond receiving power.

6.2.1. Lighting

The primary focus of the Atmega328P-PU microcontroller is to turn on the RGB LEDs. In order to do this, it provides power to the separate color diodes based on the color requested by the user and, furthermore, alters how the power is provided based on both the mode of operation.

In addition to providing power to the RGB LEDs on its Link, each Atmega328P-PU microcontroller will relay signals it receives from previous microcontrollers and perform the instructions the signals carry. This follows the pretense that instructions are introduced to the circuit by the Addicore ESP-8266.

6.2.2. Modes of Operation

The Dibs LightLinks System has several modes of operation. Depending on how the user wants the system to perform, they can use the app to change from one mode of operation to another. The different modes of operation are the Standard Pathway Mode, Power Saving Pathway Mode, Party Mode, Crowd Mode, Daytime Security Mode, Nighttime Security Mode, and Sleep Mode.

In addition to being able to switch modes, the app permits the user to change the many customizability options of the modes. The customizability options include the color of the lights, the various timers, the clock switching control, what the mode switches to automatically, and the security response should an intruder be detected. Each of these options also has a default mode, which is what the customizability options are set to prior to any changes by the user. The user has the option to revert to default settings at any time.

6.2.2.1. Active

Active modes are modes where the lights turn on and either stay on, or deactivate based off their programming. Modes that activate security lights do not count as active modes.

6.2.2.1.1. Standard Pathway Mode

The Standard Pathway Mode is the basic function of the system. It is for getting from one location to another in a dark environment.

- Preset mono colored lighting along connected links “activates”
- Human Presence sensor “off”

- All lights remain on until mode change
- Security lights remain active until mode is changed
- Remains active until mode changed

User Control:

- Color
- Default (Red)

6.2.2.1.2. Power Saving Pathway Mode

The Power Saving Pathway Mode works similarly to the Standard Pathway Mode; however, it is designed to save power by turning off links that the user has passed.

- Preset mono colored lighting activates along connected links
- Human Presence sensor “on”
- Lights turn on in front of user
- Lights turn off behind the user
- Security lights remain active until mode is changed by user (remains active if timer-based mode changed)

User Control:

- Color
- Default (Red)

6.2.2.1.3. Party Mode

The Party Mode is an entertainment function. It alternates the colors of the lights depending on the user’s input. In addition, the timer can take on the function of changing both when the lights switch colors and when they switch patterns. The patterns available are Cool, Warm, and Custom. Cool pattern consists of cooler colors and is for ambience and relaxation. Warm pattern consists of warmer colors and is for upbeat party settings.

- Variable lighting “active”
- Patterns of operation available
- Switches between colors as selected by user
- Human Presence Sensor “off”
- All lights remain “on” until mode change
- Security lights are deactivated after 10 seconds of entering Party Mode

- Remains active until mode changed
- Timer set for color set change
- A time can be scheduled for Party Mode to switch to Sleep Mode

User Control:

- Color Set (Cool, Warm, Custom)
- Pattern Set (fading, moving, flashing, random)
- Color Timer
- Pattern Timer
- Timer
- Clock Set
- Automatic Mode Switch
- Default (Warm, Fading, 2 seconds, off, 0:00, Sleep Mode)

6.2.2.1.4 Crowd Mode

Crowd Mode is a mode specialized for crowded spaces in order to provide a semblance of direction. This mode is less designed for household applications and more focused on other potentials of use, such as at a public access area or to direct traffic at a party that doesn't use party mode. Crowd Mode works by altering the illumination of lights to simulate their movement.

- All lights "active"
- Human Presence sensor "off"
- Obstacle detection sensor "off"
- Lights set to three separate levels of illumination
- Security lights are deactivated after 10 seconds of entering Crow Mode
- Remains active until mode changed
- Timer set for movement speed
- A timer can be scheduled for Crowd Mode to switch to Sleep Mode

User Control

- Color
- Speed
- Clock Set
- Automatic Mode Switch
- Default (Orange, 0.5 seconds, 0:00, Sleep Mode)

6.2.2.2. Passive

Passive modes are modes where the lights remain in the “off” position. Modes where security lights activate are still considered passive modes.

6.2.2.2.1. Daytime Security

Daytime Security mode is designed to keep the residence secured while the user(s) is not home. This mode will send an alert to the individual’s phone if the sensors are triggered.

- All lights “inactive”
- Human Presence sensor “on”
- Obstacle detection sensor “off”
- Sends Wifi signal to app on user’s phone if human presence is detected
- Can be set to inform the police of potential break-in when human presence is detected
- Security lights “inactive”
- Timer “active”
- Switches to Sleep Mode when time runs out
- Can be scheduled for Daytime Security mode to switch to Sleep Mode at a certain time

User Control:

- Timer
- Clock Set
- Automatic Mode Switch
- Notification
- Inform Police
- Default (0:00, 0:00, Sleep Mode, On, Off)

6.2.2.2.2. Nighttime Security

Nighttime Security mode is designed for silent security while the user(s) is asleep. This mode will simply light up the links that detected human presence in a color set by the user.

- Lights “inactive” unless triggered
- Human Presence sensor “on”
- Obstacle detection sensor off

- Sends Wifi signal to app on user's phone if human presence is detected
- Lights up security lights (alternate colored lights) if Human Presence is detected
- Security lights "active"
- Timer "active"
- Switches to Sleep Mode when time runs out
- Can be scheduled for Nighttime Security mode to switch to Sleep Mode at a certain time

User Control:

- Security Light Color Set
- Timer
- Clock Set
- Automatic Mode Switch
- Default (Red, 0:00, 0:00, Sleep Mode)

6.2.2.2.3. Sleep Mode

Sleep Mode is effectively the "off" setting of the devices. During this time, the device is inactive until the user changes modes.

- Lights "inactive"
- Human Presence Sensor "off"
- Obstacle Detection Sensor "off"
- Timer "inactive"
- Can be scheduled for Sleep Mode to switch to any other mode at a certain time

User Control:

- Clock Set
- Automatic Mode Switch
- Default (0:00, Nighttime Security Mode)

7. LED System

The LED system is the means by which the individual Links affect change in their respective environments. Each Link has its own LED system, which may be controlled by the system as a whole in a centralized way, as a semi-centralized system, or may be controlled in a wholly decentralized manner with each Link acting on its own.

The LED system is comprised of two main parts: the LED bank, which is merely the individual LEDs; and the LED Control Unit, which is the part of the system that interfaces directly with the microcontroller and helps to regulate the voltage and current directed across and through the individual LEDs.

7.1. LED Bank

As discussed previously, the LED bank is comprised of three sets of tri-color (RGB) LEDs. These LEDs allow the links to produce many colors. In order to make these colors, each primary additive color must be controlled independently of the other two, which will be evident in the LED Control Unit section.

Each LED's current and voltage needs must be seen to, and a specific resistor is required. The resistance may be calculated from the formula $R = \frac{V_{CC} - V_{LED}}{I}$, where V_{CC} is the supply voltage, and V_{LED} is the voltage required by the LED, given the setup seen below in *Figure 7.1.1*. The table in *Figure 7.1.2* shows the resistors which must be chosen for each value.

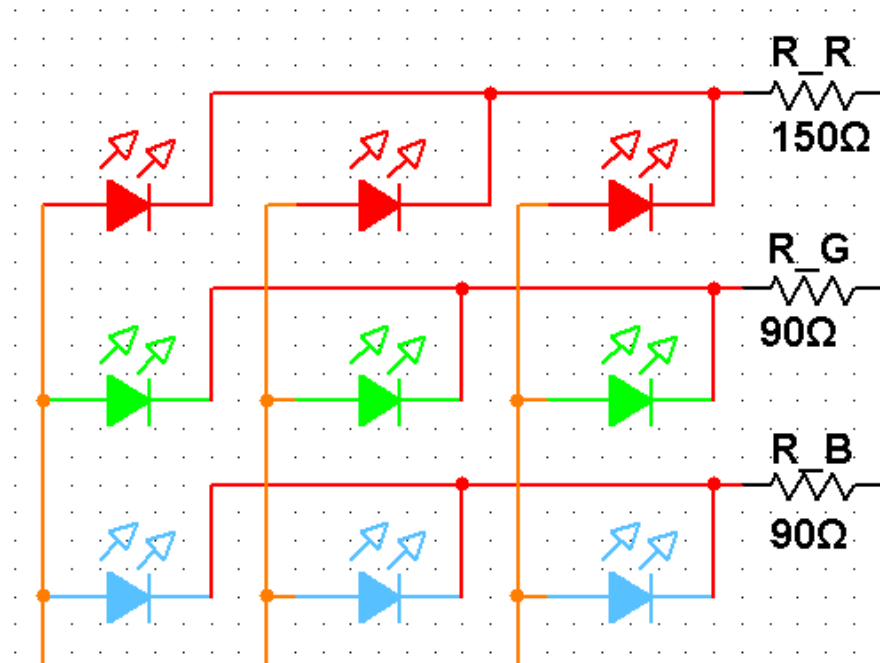


Figure 7.1.1, LED bank

Note that the columns are each a single tricolor LED set.

Table of Resistor Values at Given Input Voltages

V_{CC} (V)	R_R (Ω)	R_G (Ω)	R_B (Ω)
5	150	90	90
3.3	65	5	5

Figure 7.1.2, table of resistor values

7.2. LED Control Unit

In order to discuss the LED Control Unit (LCU), the general way to produce a color should first be discussed. In a red-green-blue (RGB) color production scheme, the intensity of red, green, and blue are varied in order to create different colors.

The LED Control Unit, as seen in *Figure 7.2.1*, is in a sense a simple set of three MOSFET transistors which essentially act as binary enable/disable switches for a given color. This is done in order to reduce the current drawn through the microcontroller. Technically, since the Atmega328's outputs are at a higher voltage than the LEDs require, they could be used to directly power those LEDs. However, this would be more current than really needs to be drawn from the microcontroller, and for various reasons, the lower the current through a microcontroller, the better.

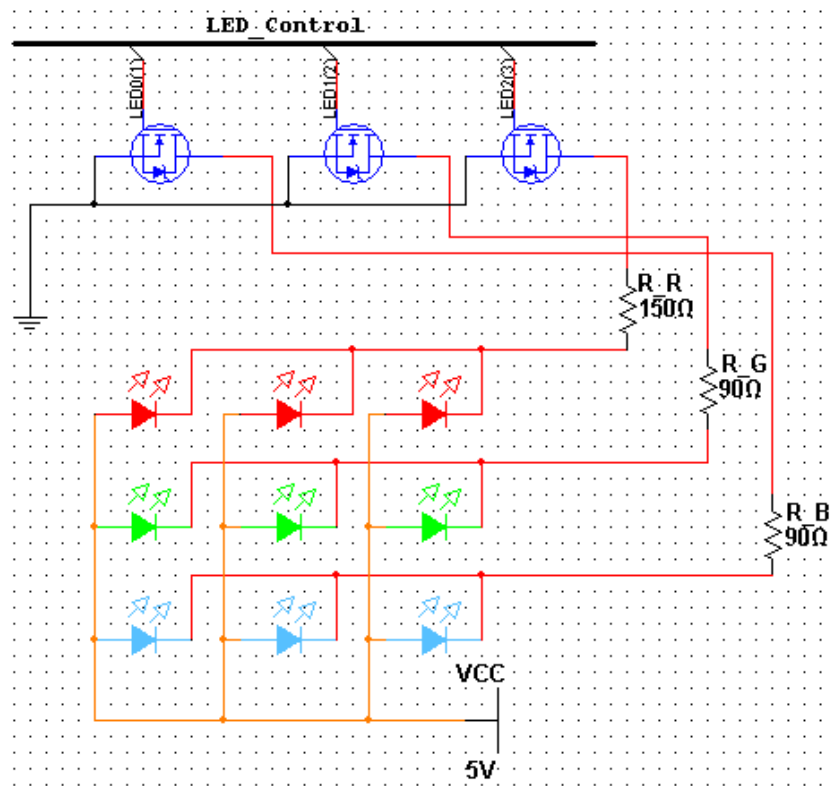


Figure 7.2.1, LED and Controller

At first blush, it might seem odd, then, that the MOSFETs are acting as simple binary gates instead of varying instantaneous current given a varying voltage input. This is because of the way the Atmega328 handles varying current in the analog world. It doesn't work the way a potentiometer might, which is to say varying instantaneous current. Instead, it uses Pulse Width Modulation (PWM) to change the average current. The chip outputs a constant voltage at either a 5 V or 3.3 V, given the board; or 0 V, and the corresponding current. The PWM frequency is 490 Hz on most PWM pins, though pins 5 and 6 have a frequency of about 980 Hz, both of which are well above the frequency that can be seen by a human--persistence of vision fools human eyes and brains into believing there is a constant beam above around 24 Hz, with a 50% duty cycle.

Choosing a proper duty cycle for PWM is something that must be tested experimentally. The Arduino *Analog* library allows for 256 levels of PWM, with 0 being a 0% duty cycle, or off; and 255 being a 100% duty cycle, or on. This is accomplished with eight bits. Theoretically, the maximum number of colors available in this colorspace is (3 colors) *

$2^8 = 768$ colors. However, due to the roughness of calibration, we will only be shooting for $(3 \text{ colors}) * 2^5 = 96$ colors. This is a wide range of colors, and means a single LED color has $(1 \text{ color}) * 2^5 = 32$ different shades, including black.

Now the question comes “is that enough colors?” Given our original design idea of having the ability to fade smoothly from one color to another, it is perhaps a bit lacking, as we can calculate the approximate speed at which a transition from one side of the brightness spectrum to another would be unnoticeable. Using a persistence of vision frequency of 24 Hz and a spread of 32 luminosity values, a fade must be completed in under 1.33 s to be unnoticeable. This is rather fast, but not awfully so.

After actual testing, however, it is clear that it is in fact hard to see transitions occurring at a slower speed, especially above a certain value of brightness. If more fidelity is needed, a hardware or software change must be made.

For a hardware solution, the change needed is to alter the time constant of the LED in question--adding a capacitor in parallel with the LED would serve to add a smoothing effect to the voltage, and as a result the current and the intensity. The time constant, $\tau = RC$. It is the time required to charge the capacitor to about 63.2%. We can find the cutoff frequency, then, through $\text{cutoff frequency} = f_c = 1/2\pi\tau = 1/2\pi RC$. Since our resistor values are already fixed, and the we can choose an f_c to be (the fastest time we might want to change color values)⁻¹, we can then find the capacitance required. From this equation, though, it is obvious that there is a large flaw to doing this in hardware, namely the intensity can only be varied slowly anyway--more quickly, yes, but still somewhat slowly, and because of the capacitor, large jumps in intensity take longer than small jumps. This is non-ideal.

A software solution would be changing the color depth. We could, with one bit being sacrificed to calibrate the exact brightness, have up to $2^{8-1}=2^7=128$ different levels, which would allow a fade time of 5.33 s or less, which is fairly reasonable. It would be fair, though, to disregard most of this and just live with a bit of jerkiness in fading. It's not extremely perceptible to the human eye, especially when other colors are fading as well.

Now, the switching devices used must satisfy four criteria with the answer of “yes”:

1. Can the switching device handle the current?
2. Can the switching device turn on and off in a time far less than the shortest amount of time a PWM signal might be on for?
3. Is the chip's binary voltage output sufficient to fully engage the switching device?
4. (a) Does the switching device have a low resistance when on and (b) a high resistance and therefore a low current when off?

Each of these considerations is important. The first is relatively simple. We merely need to look at Section 11 for the minimum current needed for a color, and multiply that by three to find our required current, which is $20 \text{ mA} * (3) = 60 \text{ mA}$. From that, we simply select a switching device capable of handling 60 mA.

The second is a bit different. Let us calculate the rise time which must not be exceeded. We have the maximum frequency value of 980 Hz, which is the inverse of the distance between the green lines in *Figure 7.2.1* above.

It's not merely that simple, however. As you can see, the duty cycle determines the time on, and that time on is shorter than the period of the PWM. Thinking it through, the formula for the rise time, t_r , assuming zero stable time on would be $t_r = \frac{1}{2^{L-1}} f_{PWM}^{-1} = [2(L-1)f_{PWM}]^{-1}$, where L = number of levels, and f_{PWM} is the frequency of PWM. Since there are up to 256 levels of PWM, $t_r < 2 \mu\text{s}$. This is of course the absolute maximum time allowable. A shorter rise time would be desirable.

The third is similar to the first. We merely need to find a switching device capable of being turned on at or below either +3.3 V or +5 V, depending on what voltage we set the Atmega328 to.

The final requirement is so that there is not much current drawn when the device is off, which would be wasteful, and that there is a low resistance when on, as the source voltage would have to be increased were there a great deal of resistance, and even if there were only a moderate amount, it would again be wasteful from a power dissipation perspective. We shall choose somewhat arbitrary values. For 4a, a Static Drain-Source On-Resistance of less than 0.1Ω is needed, and for 4b, a Zero Gate Voltage Drain Current of less than $1 \mu\text{A}$ is desired.

After all these criteria have been mathematically defined, there is still the question of whether or not a MOSFET is a good fit, which is to ask if there is any single MOSFET capable of satisfying *all four* requirements. As it turns out, all these requirements are satisfied by MOSFETS. Many can handle currents on the order of 10 A, a t_r on the order of a few ns is common, a great many have a threshold voltage of around +1 V - +2.5 V, a Static Drain-Source On-Resistance of around 0.03Ω is not odd, and a Zero Gate Voltage Drain Current in the nA at low voltages is reasonable. Common power MOSFETS satisfy all four of our requirements.

8. Human Presence Sensor

The Human Presence Sensor is located on each Link, save for the Battery Link. Its function is to detect the presence of humans, which will be discussed in greater depth below. It relays this information to the MCU, which then can make certain decisions given the data it receives, and the instructions it gets.

8.1. Function

The Human Presence Sensor (HPU) must satisfy the requirements laid out in **Section 2.3.1**. In essence, the HPU is meant to detect, one or two links at a time, where a person is along the Chain.

There are several different ways of sensing people. Ultrasonic detection, simple visible light sensing, and passive pyroelectric infrared (PIR) detectors are among these ways. Because of their mostly non-directional nature, low sensitivity to environmental shifts, low cost, and the fact that they are the standard non-computer vision method for detecting humans, viz. their use in supermarket door openers, the PIR detector has been chosen.

PIR detectors detect a difference in infrared output between spatial points. More precisely, modern PIR detectors detect a *change* in the difference. This distinction means that there is a short calibration period at the beginning of a PIR sensor's startup process during which all readings must be discarded. Technically, this calibration is an ongoing process, meaning that slow changes in infrared readings do not trigger the detector. This is necessary to account for changes in temperature and sunlight. Both heat and sunlight indirectly and directly produce detected infrared light, respectively, and since the sun shifts its apparent position throughout the day, and the temperature increases and decreases, this compensation is extremely useful. Imagine if burglar alarms went off every time the sun shifted!

8.2. Limitations

As it turns out, we do not exactly have to imagine "if burglar alarms went off every time the sun shifted." This is indeed a problem, since not all changes in sunlight are gradual. Indeed, reflections from cars can suddenly strike a PIR sensor, which is why care must be taken to position the detectors away from reflective surfaces.

We hope to allow the links to distinguish between large humans and non-human animals (NHAs) such as cats and dogs. However, one of the main ways the signals of

NHAs are rejected is by detecting heat given off at about human chest level, and not detecting heat around human shin level, which is to say NHA chest level, or approximately floor level.

Since Dibs LightLinks are designed to be a floor-level product, the first point is not really a problem--sunlight rarely is reflected onto the ground. However, the second point is indeed a problem. Admittedly, neither of the discussed signal rejection methods would filter out large land mammals such as bears; however there are few cases where it would be advantageous to detect burglars and not bears, and in many cases bears are de facto--if not de jure--burglars.

Another issue in a sense is one of the positive qualities of PIR detectors--their low directionality. Indeed, they often detect in a cone measuring 120°, which means more detectors than ought to may detect a human presence.

8.3. Accommodation of Limitations

Still, there must be a method by which humans and NHAs may be distinguished at near ground level. One method clears up the problem as long as NHAs keep a certain distance from the links--blocking a detector's 'sight' strategically. The figure below shows a top view of the sensors.

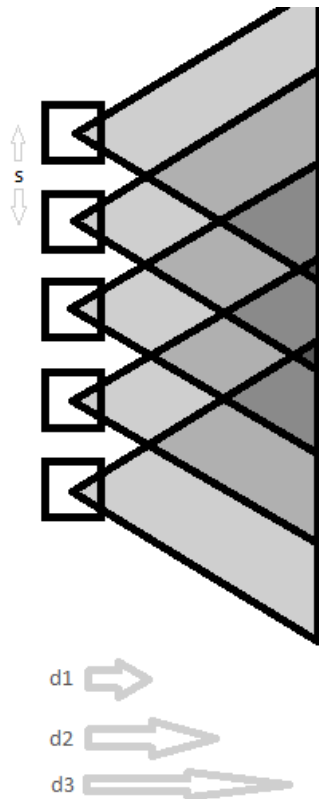


Figure 8.3.1, top view of the sensors

The distance d_n is the distance from the sensor above which a line extending perpendicular to the floor might be detected by at least n sensor(s), where the ideal n is $1 < n < 3$, with the center $n = 2$. Angles will be calculated assuming $n = 2$, and $d_{n2}/d_{n1} = n2/n1$. As is obvious, assuming an infinite detection distance and an infinite number of detectors, the further you get from a detector, the more sensors will be able to detect you, with $n = d_n/d_{n1}$.

The angle, $\Phi = \tan^{-1}\left(\frac{s}{dn}\right)$ where s is the separation between sensors, so given a specific sensor distance, we can also find $dn = s/\tan(\Phi/2)$.

The angle, $\theta = \tan^{-1}\left(\frac{\Delta h}{dn}\right)$, as seen in the figure below, where Δh is the difference in height between the sensor and the target height. There are two important angles to consider: θ_h at d_1 and θ_a at d_1 . They are the minimum and maximum angles at which it should be possible to detect a signal. $\theta_h - \theta_a$ gives us the angular width of the sensor.

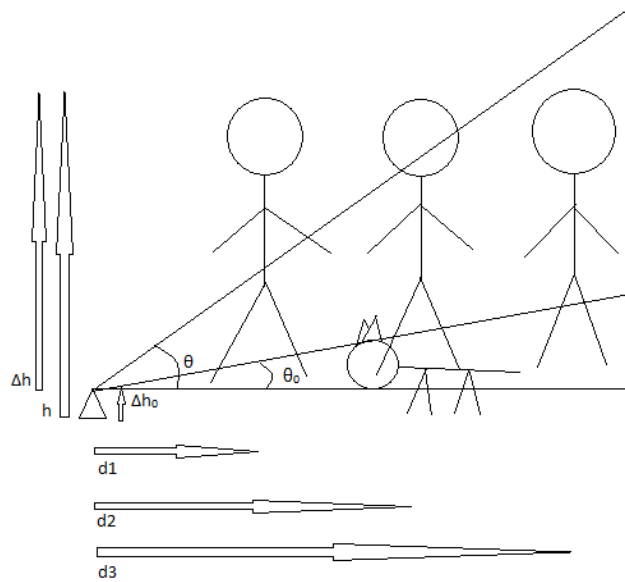


Figure 8.3.2, lateral view of the sensor system

All of this fails to remove NHA signals if the NHA ventures too close to the detector, such that the NHA's distance is equivalent to $d = \Delta(ha)(dn)/\Delta(hh)$. This is not what we'd like. There is one more way to mitigate the problem, which is to reduce the sensitivity of the detector. This has the tradeoff of reducing maximum sensing distance, but works because humans are physically larger than most household NHAs, meaning they they produce more thermal energy, and therefore have a larger infrared signature for the detector to notice.

8.4. Software Tracking

Using the PIR sensors on the Dibs LightLinks System an object's location can be tracked. While tracking the low powered PIR sensors will be constantly running, that way when an object follows the path of LightLinks the PIR sensors on the LightLinks will detect the motion in its vicinity and send a signal along the Chain that at this location there is some sort of object here. This signal will travel along the Chain until the signal encounters its first Interface Link. Once the Interface Link has been reached, the signal is then converted and transmitted wirelessly via Wifi communication to the Dibs LightLinks server. The server then takes this signal and delivers some sort of

notification to the user via the web application or an email stating that there is an object located at Link Number X, where X would be the ID number of the LightLinks. A diagram of this interaction can be seen below.

Software Tracking



9. Door State Sensor

There are multiple ways of sensing whether a door is open or not. Ultrasound, infrared, and a magnetically-tripped reed switch are among these possibilities.

Ultrasound is a bit unwieldy, and requires a surprisingly large amount of energy. The largest issue with ultrasound, however, is its bulkiness. Ultrasonic transducers are not enormously large, but they are just large enough to be difficult to slip underneath a door with their enclosures on.

A magnetically-tripped reed switch is actually a wonderful low power, low size solution, but it would require the user to meticulously line up a magnet with the sensor, and furthermore would mean they would have to attach something to the door. That would be difficult, seeing as doors are notoriously low to the ground. Additionally, where the sensor to move just slightly out of place, the sensor would register that the door was constantly open.

A simple infrared sensor is best for this. The sensor works by bouncing produced infrared light off the bottom of the door. A detector looks for this returning light, and if light is found, indicates that the door is closed. If light is not found, that means it is not bouncing off the bottom of the door.

Admittedly, this does have a major drawback: if the bottom of the door is extremely non reflective to infrared light, the sensor will register the door as being perpetually open. However, almost all common materials are at least slightly reflective to infrared light, including most paint, wood, and plastics. If all else fails, placing a small amount of that ubiquitous household go-to, cellotape, underneath the door would entirely fix the issue for a user.

The close proximity means that even a low albedo in the infrared spectrum will produce a signal of sufficient strength to be detectable. In order to further magnify the signal and convert it from the analog to the digital domain, we can add an operational amplifier (op-amp). The LM358M is a good, multi-purpose digital-output op-amp. In order to fine tune the on-threshold, we can also add a small (10 k Ω) linear potentiometer to the system, as seen in *Figure 9.1a*, shown below.

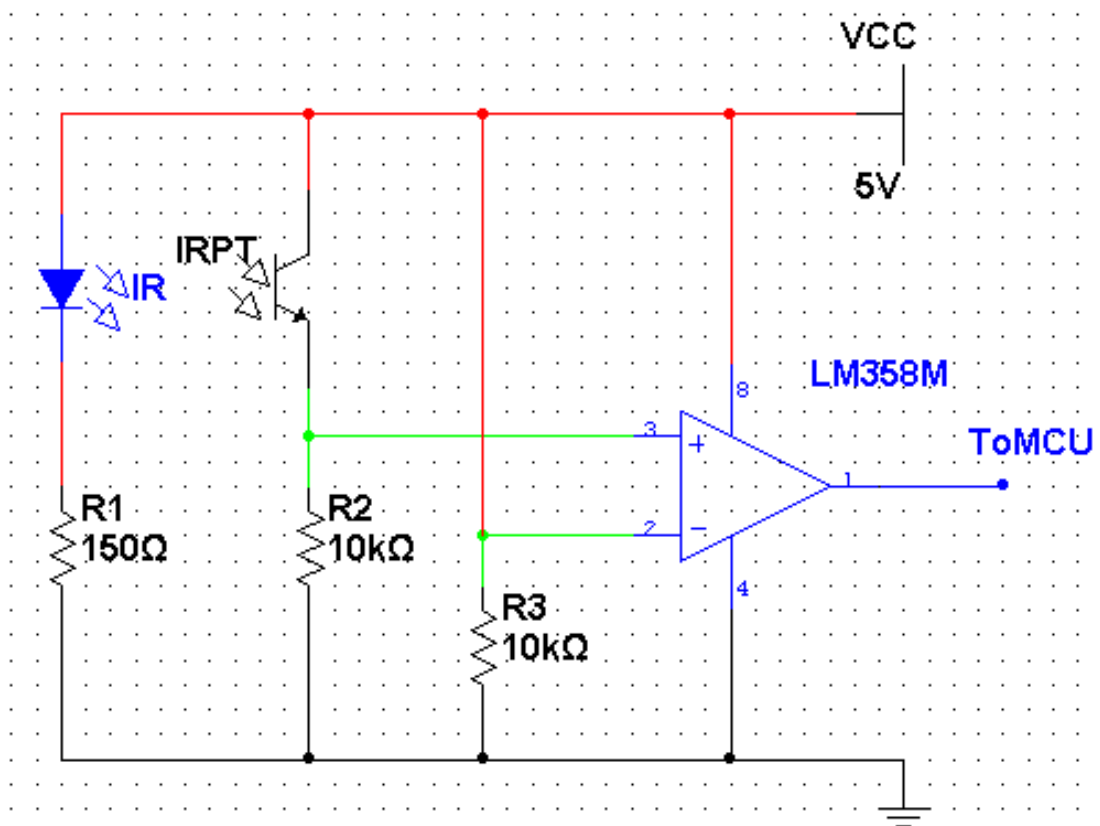


Figure 9.1a, DSS Schematic

However, the circuit (below, *Fig. 9.1b*) is fairly simple, composed of only four components, and is housed within the Door Dongle, which may be attached to the Door Link with a detachable cable.

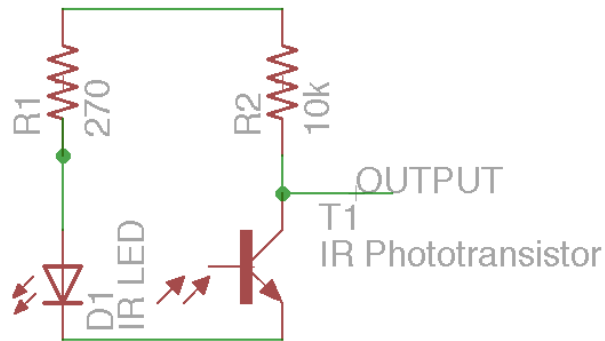


Figure 9.1b, Actual door state sensor

The output voltage is controlled by the IR phototransistor, which varies its resistance with a varying incident infrared light intensity. This output voltage is interpreted by the Atmega328, and is quantized to a 1 or a 0, for open vs. closed.

From the requirements section, The Door State Sensor (DSS) must be able to fit underneath a door. It is obvious that a standard Link would not be able to actually fit underneath a door. In order to accommodate this limitation, a system must be devised to allow sensing to occur without the main Link itself being below.

Acting somewhat like a computer peripheral, we will have a “door dongle” that hangs off the Door Link and has all the relevant sensing components. Seen below in *Figure 9.2* is a standard use case, that is a Chain running along a hallway, with a Door Link positioned next to the door, and a dongle extending into the actual doorway in order to sense the door’s state.



Figure 9.2, Door and Dongle

Here we see a Chain running along a room, with a dongle extending into the door well. The sensor is currently reading “1”, or “closed,” since the optical transistor is receiving a high light level, and is therefore in its low current mode. This is transmitted to the MCU.

In actuality, the dongle should be placed as close to the fulcrum of the door as possible. This is to prevent an individual from tripping on the dongle. Given a minimum angle at which the door is considered open, the dongle must be centered such that no part of the door is above the sensor.

Proximity to the fulcrum is a double-edged sword. The closer the dongle gets, the greater the angle, and therefore the more the door must open in order for the sensor to

register the door as opening. Because of this, it must be the responsibility of the user to test whether or not the door sensor is working properly.

10. Obstacle Presence Sensor

In order for the Dibs LightLinks System to inform the user of the presence of obstacles in their path, a sensor is required. This sensor will be used to detect the location of obstacles so the microcontroller can decipher which Links to activate so the user can be informed of the obstacle's presence.

10.1. Types of Sensors

The Obstacle Presence Sensor (OPS) must satisfy the requirements laid out in **Section 2.3.1**. The OPS is a safety-based function of the LightLinks system integrated to detect any obstacles in the path of the user and inform them of it via alternate colored lighting.

There are many methods of sensing an object's location. Laser tripwires, Ultrasonic Sensors, and Infrared Sensors are three such methods, each with their own advantages and disadvantages. A method of obstacle detection that is compact enough to fit on a single Link, reliable enough to be able to consistently detect obstacles regardless of path structure, and has enough range to be able to detect obstacles in any given path is required.

Laser tripwires would easily be able to detect if an object is in the path, but not its location. Due to the structure of the pathing, only whether or not an obstacle lies in the path and not its location is needed, so this will not be a problem. Laser tripwires, however, suffer many relevant problems. The requirement of the paths to be parallel, the amount of lasers required to provide adequate feedback, and power consumption deem this a poor choice for an OPS.

Infrared Sensors would come in a pair, one that sends an infrared signal and another that receives a signal. These would require less power and fewer sensors than laser tripwires, but would have a much shorter range. Additionally, due to their compact size, they would be optimal sensors for the door link.

Ultrasonic Sensors, like the infrared sensors, would also come in a pair. One would transmit an ultrasonic signal and the other would receive the signal. Depending on the time it took to receive the signal, the system would be able to determine the approximate location of an obstacle. Unlike infrared sensors, the range of ultrasonic sound is well beyond what is required in this system. These sensors are optimal for

obstacle detection in the system. Ultimately, however, ultrasonic sensors weren't used in the final design.

10.2. Application of Sensor

The Ultrasonic Sensor would be located only on Standard Links. There would be a single pair of sensors in the center of each Link lengthwise, but only on one side. The user would set up the system such that all ultrasonic sensors were pointed inward in order to detect obstacles in the pathway. Because of this, two different types of Standard Links should be manufactured, one for each orientation of ultrasonic sensor.

The ultrasonic sensors would emit sound at ultrasonic frequencies towards the inside of the path. If there are any obstacles located within 4 feet of the sensors, the object will be detected as an obstacle in the path. This range can be adjusted in the user interface. If an obstacle is detected, the Links on either side of the obstacle will change color to provide as a warning indicator for the user.

11. Power

In this section, power requirements will be discussed. Each component in the system has its own needs, needs with both minimum power required and maximum power which may be handled.

11.1. Demands of Components

Among the major components of this project, there are a few devices which require particular, specific voltages and currents, and one which requires an extraordinary level of stability in terms of voltage. The major components are the microcontroller, the PIR sensor, the LED system, and the Wifi module.

Let us first turn our attention to the microcontroller. It has a great deal of flexibility in its requirements. As seen in the tables below, *Figure 11.1.1* and *Figure 11.1.2*, the measured power consumption is influenced a great deal by the voltage chosen and the clock speed chosen. Note that this is actually not exactly a measure of the microcontroller alone; instead, it also takes into consideration the consumption of the external oscillator (if applicable) and the voltage regulator chosen.

Oscillator	16Mhz (external)	8Mhz (internal)
Voltage: 3.3 (V)	3.3	3.43
Current (mA)	6.6	3.6
Power (mW)	21.8	12.3
Voltage: 5 (V)	4.98	5.2
Current (mA)	16.43	12.2
Power (mW)	81.8	63.4

Figure 11.1.1 data courtesy of Igor Ramos [1]

The PIR device has its own power needs Its requirements are also rather relaxed, needing a voltage source of between 5 V and 12 V, inclusive, and draws a current at around 1.5 A. I was unable to find precise amperage at the correct voltage. As such, actual testing on hardware is required for a greater deal of accuracy. However, if the 1 V regulator is bypassed, the current is 1.6 mA, which I suspect is the same as its normal operating amperage.

The LED system of each Link is composed of three sets of tri-color LEDs, along with a controller that is connected to the main microcontroller. The LEDs themselves can draw 20 mA per color, meaning all three colors at full intensity may draw up to 60 mA, and all three tri-color LEDs together require 180 mA. Interestingly, the different colors need a different forward voltage: green and blue need 3.2 V, while red only needs 2 V. This is accommodated by the controller.

The final of our devices is the most finicky and strict. It is the Wifi module, the ESP8266, which by all accounts requires an extremely stable voltage source to operate properly, which means it needs to have its own isolated voltage regulation at 3.3 V. This is to say even if the Atmega328 is running at 3.3 V as well, it is insufficient to allow both 3.3 V components to draw from the same 3.3 V supply. This need for stability is a common feature of wireless communication devices, and choosing a different one would not remove the limitation. It draws a maximum of 170 mA.

11.2. Transmission

The electrical power necessary to drive the Links must make its way from the Battery Link to the rest of the Links, as all other Links are unpowered. This section will discuss the constraints and requirements of the transmission of this power through the Links, including through the Link Extenders.

11.2.1. Requirements of Transmission

There are a few especially important considerations when choosing a transmission system, namely:

- Distance power must be transmitted
- Voltage required
- Voltage tolerance
- Amperage required

The first requirement is intrinsically related to the way the system is meant to be used, and is less constant than in most other devices, save perhaps Christmas lights. The Links actually mirror Christmas lights in another interesting way. When these holiday decorations first came on to the market, they were flawed in a surprising way. Christmas lights were on the line in series, not parallel, meaning that if one bulb went out, they all went out, and their light output dropped somewhat noticeably as you moved away from the source. After a strangely long time, this was corrected, and now the lights are arranged in series. Similarly, the links are also in series, which allows them to all be at as close to the same input voltage as possible. Despite that fact, there will be some voltage drop.

A simple addition of all input currents in a Link gives a rough estimate of the maximum current that may be drawn by a single link. As previously mentioned, Chains of links can be of an arbitrary length. Of course, practical limitations dictate a realistic length, and therefore a maximum number of links. These limitations are (1) the maximum current that may be supplied by the power source and (2) the maximum current that may be carried by the lines and internal traces. Technically, voltage drop from the internal resistance of the wires should be a consideration given that there is only a certain input voltage tolerance. However, the voltage drop will be a negligible consideration when compared to the current drawn, given the fact that most links will be drawing a considerable quantity, and to compensate for that, wires and internal traces will need to be rather thick, and therefore have a low resistance, which means a low voltage drop.

So now, given these considerations, we must make our decision: how many links may comprise a single Chain? A good starting point is to consider what the maximum supply current is, and work backwards from there. The aim of a commercial version of the Dibs LightLinks System is to have a battery backup system capable of powering a reasonable number of links for a reasonable amount of time. This will be discussed in **Section 11.3**, Storage. For now, however, we must pick a value for output amperage, divide that by the amperage drawn by the most current-thirsty link, and use the floored value of that as the maximum number of links in a Chain. Suppose 10 A may be supplied, and a Link needs 600 mA. That would mean there could be sixteen links in a Chain. From the amperage, we can calculate the type of wire that must be used in LEs, as well as the thickness of the power supply traces within a standard link. There are two main ways of calculating wire thickness (and from that, it's simple to see what gauge wire corresponds to the calculated thickness).

The first is far simpler, and is applicable to LEs, and is in a sense the idealized case for conventional links were they not drawing current. You decide the maximum voltage drop allowed, and use that. Looking at Ohm's Law, $R=V/I$, we use the voltage and current we've decided on to calculate the resistance of the wire. Resistance of a wire R , is related to its resistivity ρ , length L , and its area $A = \pi r^2$, where r is the radius of the wire, and what we are trying to find. Combining these equations, we see that $r \geq \sqrt{\frac{\rho LI}{\Delta V \pi}}$. The resistivity of copper is approximately $1.68 \times 10^{-8} \Omega\text{-m}$. Using sample numbers of 1 V maximum drop, and 50m of length, as well as the sample current above, that gives us a radius of $\sqrt{\frac{1.68E-8 * 50 * 1}{3 * 3.14}} \leq 0.000299$ m. The AWG can be found with $AWG = -39 \log_{92}(2r/127) + 36 = -39 \log_{92}(2 * 0.000299 / .000127) + 36 = 22$ gauge wire. The AWG table is also handy, and is presented below in *Figure 11.2.1*. Note that it asks for *diameter* in mm, instead of *radius* in m.

AWG	Conductor Diameter (mm)		AWG	Conductor Diameter (mm)
0000	11.684		15	1.45034
000	10.40384		16	1.29032
00	9.26592		17	1.15062
0	8.25246		18	1.02362
1	7.34822		19	0.91186
2	6.54304		20	0.8128
3	5.82676		21	0.7239
4	5.18922		22	0.64516
5	4.62026		23	0.57404
6	4.1148		24	0.51054
7	3.66522		25	0.45466
8	3.2639		26	0.40386
9	2.90576		27	0.36068
10	2.58826		28	0.32004

11	2.30378		29	0.28702
12	2.05232		30	0.254
13	1.8288		31	0.22606
14	1.62814		32	0.2032

Figure 11.2.1

The second way has more to do with deciding on the maximum allowable heat dissipated along a trace, which makes more sense when dealing with device internals. Using the IPC-2221 specification, we are able to determine the dimensions of our trace. The decisions which factor into the trace design are the width, the thickness, the maximum allowable temperature rise, and the maximum current. This is given by the equation $I = k\Delta T^{0.44} A^{0.725}$, where $k=0.048$ for outer layers, and 0.024 for inner layers, I is the current in A, ΔT is the maximum allowable temperature rise in K, and A is the cross-sectional area in mil^2 . Solving for the width, w , which we get from the cross

sectional area $A = w*d$, where d is depth, $w = \frac{I}{k\Delta T^{0.44} d^{0.725}} \frac{1}{0.725}$. For an outer trace, choosing a reasonable ΔT of 10 K, a reasonable thickness of 1 mil, and the current of 10 A chosen above, we can get a width of approximately 390 mil, or around 1 cm. That's a rather wide trace, which makes sense given the high amperage that the trace must accommodate without issue.

Now that we have the type of wire and trace used in the ELs and the conventional links, respectively, there's one more connection that needs to be considered: that is the connections between links. Again, there are various considerations to be made. The first of which is essentially "is a connector more like an EL's wire, or more like a conventional link's trace?" The answer is that it's more like an EL's wire in that it needs to be essentially invisible from a voltage drop perspective. From that it can be reasoned that the cross-sectional area of a connector must be approximately equal to that of the wire, which given our variable choices above works out to around 435 mil^2 . Interestingly, the trace area and the wire area work out to be approximately the same, around 400 mil^2 each!

Taking a look at the way most connectors work, that's hardly a consideration worth making since most consumer-scale connectors are certainly large enough. This brings us to the other consideration. Since these links must be water resistant, and their power

systems are not self-contained, naturally their connectors cannot be exposed to the elements. Doing so could be disastrous from both a device integrity point of view and from a power consumption point of view. Water resistant connectors appear to be a must. How important are they for reducing power consumption exactly? Drinking water has a resistivity of around $100 \Omega \cdot \text{m}$. Since $R = \frac{\rho l}{A}$, given that the area is equivalent to $2.809 \times 10^{-7} \text{ m}^2$, and choosing a reasonable separation of $2 \text{ cm} = 0.002 \text{ m}$, the resistance between the two connectors on one side of a Link is $712 \text{ k}\Omega$. At a voltage separation of 9 V , that's a dissipation of only 0.11 mW , and a current of only $0.13 \mu\text{A}$.

At first blush, it would seem that there's no reason, then, to worry about any leakage through water. However, there's one important consideration that has nothing at all to do with power transmission itself that nevertheless puts certain restrictions on what can be done with respect to power transmission. That consideration is the signal that must be transmitted in close proximity to the voltage lines. Despite the low current, there is still a voltage gradient stretching between the terminals of the voltage line connectors. This gradient is not guaranteed to be perfectly constant, and given that a digital signal may be interfered with, some consideration must be given to isolating that signal from outside interference.

In the end, the capacitance of the wires caused I²C communications to fail entirely. Each wire had a capacitance of approximately 108 pF , with a full per-link capacitance of approximately 390 pF . This was not originally listed as a consideration when picking pull-up resistors, but after consideration of the frequency of data transfer signals, it began to look more and more like the cause of communication failures. We consulted a Texas Instruments document outlining the way pull-up resistors are chosen, and noted the requirements. At 5V , we require resistors of at least $1.5 \text{ k}\Omega$. However, the capacitance of the wires matters as well, which limits the maximum value of the pull-up resistors. This maximum varies according to the curve in *Fig. 11.2.2*, below.

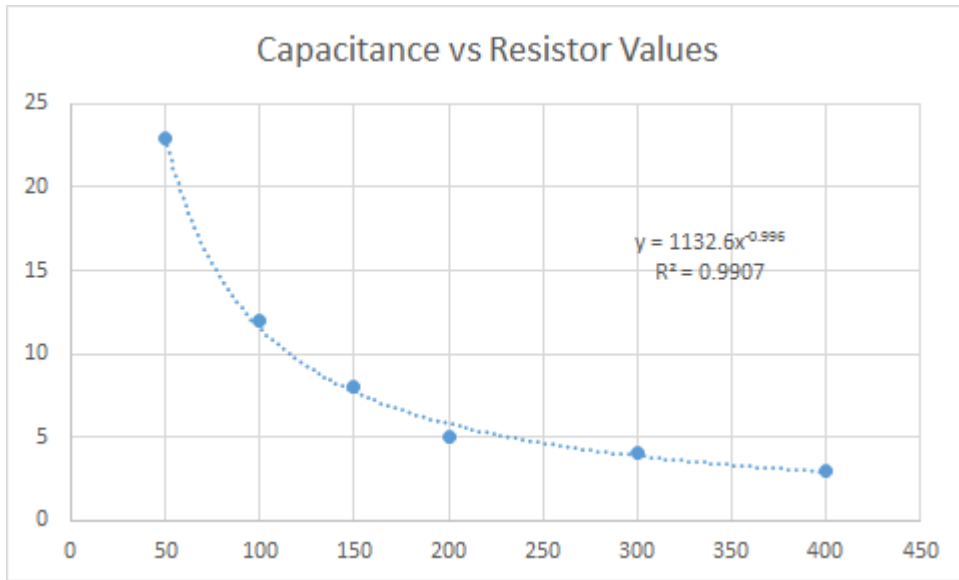


Figure 11.2.2, *Capacitance vs Resistor Values*

This puts a further limit on the number of Links possible: slightly more than one, which is, of course, useless for communication.

11.2.2. Connector Types

In regard to the above concern over water, we must waterproof the connections being made. In all, there are five connections: serial data (SDA), serial clock (SCL), Link ordering line (LOL), positive supply voltage (Vcc), and ground (GND). Vcc and GND connector requirements have been discussed above, and must be met. The SDA, SCL, and LOL connectors need only to supply 3.3 V or 5 V at a low amperage.

The connectors must also be able to work with and not be blind to the human element. They must be directional so that a human being not intimately familiar with the design is able to connect the links properly without fail. Doing so improperly could easily damage components, or cause serious harm to the user or other persons in the vicinity.

To use the vernacular, the connections must be “idiot proof.” Bearing in mind that the links may be rotated about the center around the vertical axis, this means that connectors must either be male-type or female-type on both ends. In order to reduce

the risk of accidental breakage, the links should be female at both ends, and the connectors ought to be male at both ends.

Waterproof connectors generally come in two, three, and four-circuit models. Since we have three data lines (SDA, SCL, and LOL) and two power supply lines (GND and Vcc), we will split the connectors in two, utilizing a two-circuit connector as well as a three circuit connector on either end, the Mizu-P25™ Miniature Waterproof Connectors.

11.3. Energy Storage and Power Supply

For each Link to function, it must be powered. In order to do this, there must be a well of energy available. This reservoir may be comprised of both a battery and an electrical mains connection. It either is connected to both or just to the battery, since the Links are designed to be used in an outdoor environment.

11.3.1 Energy Storage Technology

The first question that must be considered is if it's even practical to use stored electricity. Will the voltages be tolerable, will enough current be supplied, will power be supplied for a reasonable amount of time? If so, what battery technology--if indeed it is battery technology that will be used--is best for the application?

For certain applications, a capacitor bank is a reasonable choice. If an extremely large current must be drawn in a very short amount of time, there isn't really another practical choice. For this application, however, we need only a somewhat large current at a low voltage for a long time, which indicates that a battery is the proper choice.

Due to the closed nature of the devices, it also makes sense to have sealed, non-replaceable batteries, and that as a result these batteries must be rechargeable.

11.3.2. Rechargeable Battery Types.

There are a few main types of rechargeable battery. The most common among these are the lead-acid, alkaline, nickel-metal hydride, and lithium-ion polymer (LiPo) based batteries. They each have their own costs and benefits, which must be weighed when choosing the battery technology to use.

In any engineering application, there's one major constraint that must be evaluated: cost. Alkaline batteries are the least expensive per Watt-hour of energy of the

technologies listed above (though certain more exotic lead-acid batteries can deliver slightly more on occasion). They deliver 7.7 W-h/\$. This is a nice amount of energy.

That leads into the second consideration: energy density. There are two main ways of measuring energy density, that is energy per unit volume, and energy per unit mass. Since our application is mainly one that deals with ground-level, immobile devices, mass is essentially a non-factor. Because of this, Watt-hours per kilogram is what we will use. LiPo batteries are far more efficient than the alkaline batteries, at around 150 W-h/kg as compared to a rather dismal 85 W-h/kg.

We also would like our batteries to have a low effective series resistance (ESR). LiPo batteries have a much lower ESR than alkaline batteries at around 12 mΩ and 120 mΩ, respectively. The reason we want a low ESR is to reduce the power dissipated through them. This adds up, since the batteries are arranged in different configurations based on their natural voltages.

If we wish to have a voltage of 9V out, we need to design our battery banks in a way that allows this. LiPo batteries have an inherent voltage of 3.7V, and alkaline batteries have an inherent voltage of 1.5 V. That means we would need 3 LiPo batteries in series, or 6 alkaline batteries in series, which gives an ESR of 36 mΩ and 720 mΩ, respectively. Due to the property of impedance matching, we want our internal resistance (resistor sum of the ESRs) to be as close to the load resistance as possible in order to most efficiently transfer power to the system without a loss.

This gives us the rather interesting challenge of measuring the resistance of the entire system, which is full of various Link types, as well as different Link numbers. Compounded on that is the fact that it is possible that we may also add batteries in parallel, changing the ESR.

In summary, the power dissipated by the battery is $\frac{E^2 r}{(r+R)^2}$, where E is the electromotive force of the battery, r =the ESR of the batteries, and R=the load resistance. Given a varying load resistance, the best we can do is note that if we hope to decrease the power wasted by the battery, we must attempt to lower the ESR of the batteries as far as possible. Because of this, it seems that the LiPo battery type is, once again the best.

As such, we have decided on using the LiPo battery technology. It offers the best in most categories, and is a reasonably reliable modern technology.

11.3.3. Charging Circuits.

In order to charge the battery, we will use a premade charging circuit, the LP2951, which will be powered from a 12 VDC wall-adapter, the “Super Power Supply” 12 VDC adapter.

Since we need to be able to reach 9 V at the battery, we are using a 3-cell (technically 3*k, k=1, 2, 3...; since the cells can be layered in parallel to increase current output) charging circuit.

The LP2951 automatically allows the input power source to be removed or not. This is because of the voltage biasing produced by the transistor, as seen below in *Figure 11.1*.

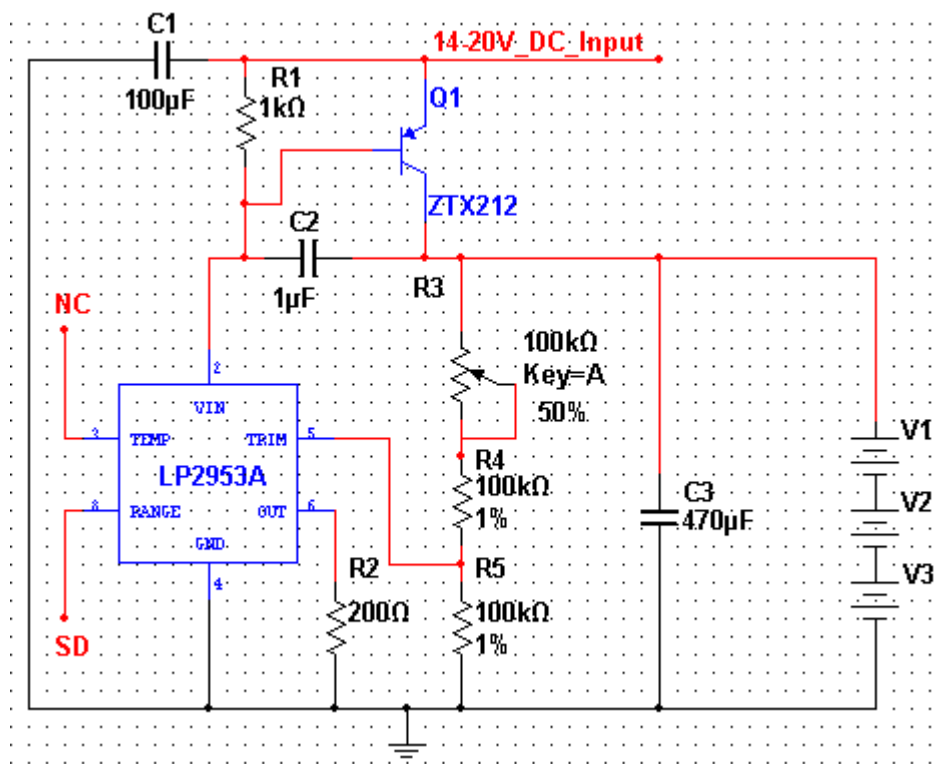


Figure 11.1, Charging Circuit

This added difficulty, as well as the fact that the project pivoted mainly to an indoor system, the battery was rejected in favor of a 4 A wall charger.

12. Feature Set

The Dibs LightLinks system offers a great deal of variety to its use. While one simple use may be to light a walkway, another use could be a security system, a party light, a notification and alert system, and plenty more. The Dibs LightLinks System comes with many features and possibilities.

12.1. Personal Accounts

The Dibs LightLinks System will offer access to an account tailored to the user themselves. This account can be accessed both through a web page as well as a mobile application. With this account the users can receive message alerts from the Dibs LightLinks System, give Dibs instructions from afar, as well as register the device on the server.

Personal Accounts were not added to the final project.

12.1.1. Device Registration

Once a device has been properly registered on the server, users can use the web application to send different instructions to the Dibs LightLinks system. These instructions can consist of functions such as activating security mode, changing the color of the links, activating night mode, and plenty more features.

12.1.2. Wireless Notifications

Since Dibs can communicate with other devices via a network, it can inherently interact with other devices via a wireless internet connection. This allows the Dibs LightLinks System to send notifications straight internet capable devices.

12.1.3. Visual Notifications

Every link that is a part of the Dibs LightLinks System comes equipped with a PIR motion sensor. As users move past the links the Dibs LightLinks System keeps track of the last known link that has been past. When the user gets some sort of notification such as an email the last visited Link begins to flicker to visually notify the user that something thing new is in their inbox. This feature is meant for the moments that the

user may not have some sort of device on their person to check their messages or emails.

12.1.4. Nonlocal System Status

Whether someone is at home, at work, or generally on the go they can always know the status of the Dibs systems via a smartphone or a computer. Using the application a user can check how their links are connected and if there has been any sort of problems that have caused the links to not function correctly.

12.2. Programmable Lights

There are multiple possibilities for how exactly to use the lights. The lights could be programmed to anything as simple as turning on and off to flickering in certain timed patterns with different colors as some sort of security alert. All of the functions of the LED lights can be controlled by the web application that is used.

12.2.1. Ambient Lighting

This is one of the defining features of the Dibs LightLinks System. The ambient lighting that the LEDs produce are meant to light up pathways through the dark without being too disturbing or alarming to the users. The Low powered ambient light setting is a key factor for Dibs Sleep mode because it provides light that is calm and welcoming rather than just bright and overbearing.

12.2.2. Timed Lighting

The LEDs of the Dibs LightLinks System can be set to correspond to a certain time of day. The lights can also turn on or off depending on a specific timed pattern the user may have set via the application.

12.2.3. Sensor Triggered Lighting

Since every link in the Dibs LightLinks System has a PIR sensor attached to it, it is very possible to have the links turn on or off based on the most recent motion that the link detected. This is one of the key features in Sleep Mode because the lights can actually follow the user's path that is set based entirely on motion.

12.2.4. Colored Lighting

The LEDs have access to different color choices that the links can emit. These different colors be seen in a rapid and random order during party mode, or in a constant color for when certain sections need to be lit with different colors, which includes being completely lit with white LED lights as well. The amount of different colors is determined by how many bits the Dibs LightLinks System is dedicating to the LEDs of each link. This bit amount is 5 bits. Since 5 bits are being dedicated to the coloring of the LED lights, there can be a total of 3 colors * 2^5 different possible values for each colors which totals to 96 different colors that the Dibs LightLinks System can emit.

12.3. Automatic Link Ordering

A certain number of functions of the LightLinks requires a using a centralized system, as opposed to merely a decentralized system that might only make use of local sensing. There must then be a method of determining the order in which the links are arranged in a chain.

One possibility that was considered was manually entering the order of the links. A person would input the string of numbers designating each Link into the smartphone application, and then manually order them. This is a time consuming process.

A better method would remove the need to enter the numbers of links by hand, which may be accomplished with a QR-code input. That is to say, each link would have a scannable QR-code on it that has its semi-unique (there are 120 numerical possibilities--see Section 6.1.3 for details as to why the maximum possible number is 120) number encoded. The user would then scan the Link's QR code in the Link Entry section of the smartphone application, and the application would automatically log that number for later use.

Needing to enter them in some manner, however, is a non-negotiable requirement of the I²C library. Each component needs to be individually addressable, and the addresses may not be self-selected, which is to say that there is no room for collisions in addressing, though I²C can detect collisions in transmission. For more details, again reference Section 6.1.3.

That better method would also allow for automatic Link ordering. In this section, we will describe the algorithm we invented to allow this. It is all initiated by the user, once each Link in a single Chain's ID has been sent to its IL. The process begins when the user hits the "Order" button on the smartphone application. Doing so prompts the IL which is

the master to tell all of its slaves, which are all of the other Links, to set their announcement pins (APs) to HIGH, and listen on their listening pin (LP). Each slave then tells the master it has completed this over I²C. If the master doesn't hear an ACK from each and every one of its slaves within the timeout period, the master tries again. After the second timeout if nothing is heard, the master tells the slaves to stop, and fall back to decentralized mode. The master then sends an error to the smartphone application.

If, however, the master receives an ACK from each Link, the master tells each Link to begin ordering themselves. During the ordering process, which will be explained further below, a Link sends its ID and its proposed order number on the chain. The master then sends the ID and number to each of the other links. If there are any conflicts, the master resets the ordering process. If this happens again, the master sends an error to the smartphone application.

The actual ordering works as such: imagine five blind people in a line, Alice, Bob, Carter, Dana, and Eve, who wish to know what order they're standing in. We sighted people can see they're in the order Eve, Bob, Alice, Dana, Carter (EBADC). Dana is the leader of this merry group, and tells the others to each reach up and touch the shoulder of the person in front of them. They then sound off once they've completed the activity of reaching. Reaching, not feeling that they've touched someone because admittedly, this analogy breaks down because the person in front should notice they aren't touching anyone. Assume, perhaps, that they've all got insensate, injured hands as well.

Dana then asks whoever is not currently being touched to please speak up. Carter says his name and that he believes he's at position 0 since he's not being touched, and is therefore the caboose. No one contests this, and Dana tells Carter to let go of the person in front of him. Carter acknowledges this. Dana then asks the next person to please speak up if they're not being touched. The next person is herself, since she felt Carter let go, but she tells them to do this anyway, just in case there's a fault in their line. She announces her name and claims to be in position 1, and hears no contest from anyone. Next she tells herself to let go of the person in front of her. This continues down the line, until Bob, who is in position 3 announces his position. At that point, Eve has not spoken, but Dana knows where Eve is by process of elimination. Dana then announces that the ordering process is over, and that the order is Eve, Bob, Alice, Dana, Carter.

The explanation above can be translated into electronics in this fashion: substitute names for IDs, people for links, speech for I²C communication, leader for master, touching for holding the AP HIGH, letting go for holding the AP LOW, and feeling for listening on the LP. This process is illustrated below in *Figure 12.3.1*.

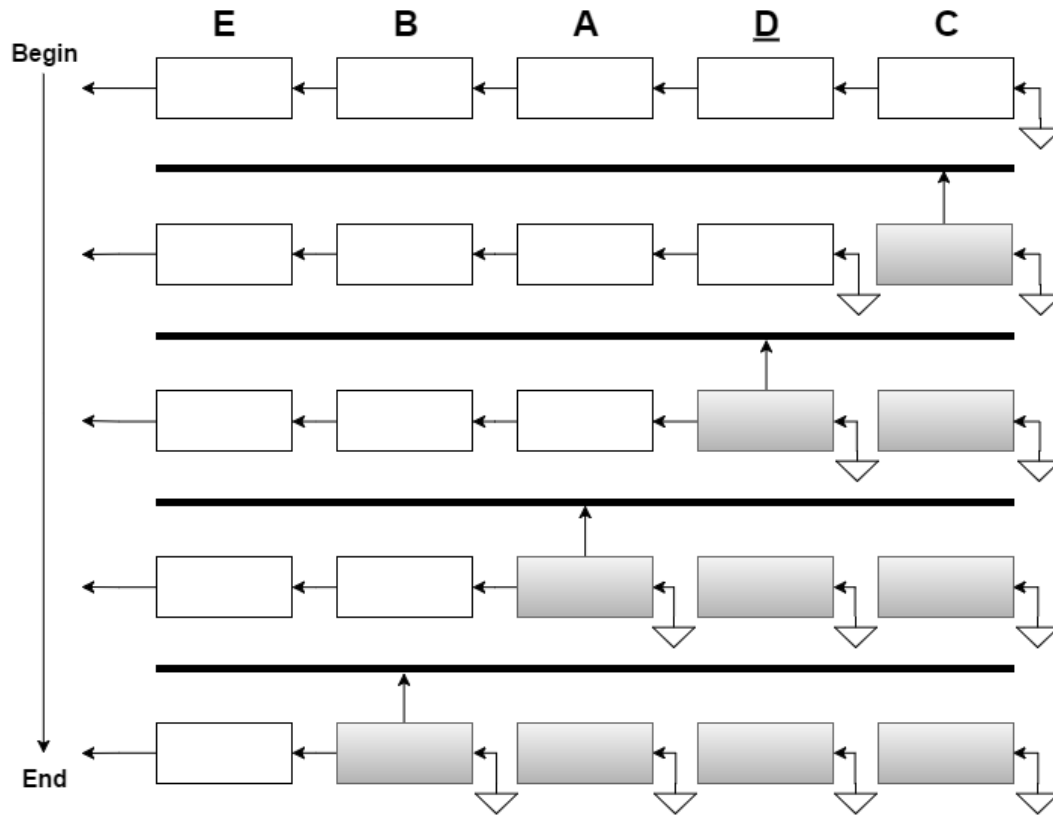


Figure 12.3.1, Link ordering process

Due to non-functioning I²C communications, this method of ordering was dropped in favor of hard-coding.

12.4. Door State Sensor

On the Door Links of the Dibs LightLinks system, there are Proximity Infrared Sensors that are used to detect if the door has been opened or not. The Proximity Infrared Sensors act as a type of tripwire for the door to activate. While the door is closed, the Proximity Infrared Sensors on Door Link face parallel to each other along the Dibs LightLinks pathway. One of the parallel sensors sends an infrared signal to the other parallel sensor. This signal persists until an object breaks communication between the sensors. In this case what would break the sensor communication is the door opening. Once the door opens and the signal is broken, the Door Link sends a signal down the Chain of LightLinks. This signal travels until it reaches an Interface Link. Once the

Interface Link is reached, the signal is transmitted over Wifi communications to the Dibs LightLinks server. After the signal reached the Dibs LightLinks server, the server sends a notification to the user through their personalized account. A flowchart of this transferring of information can be seen below.

Door State Sensor

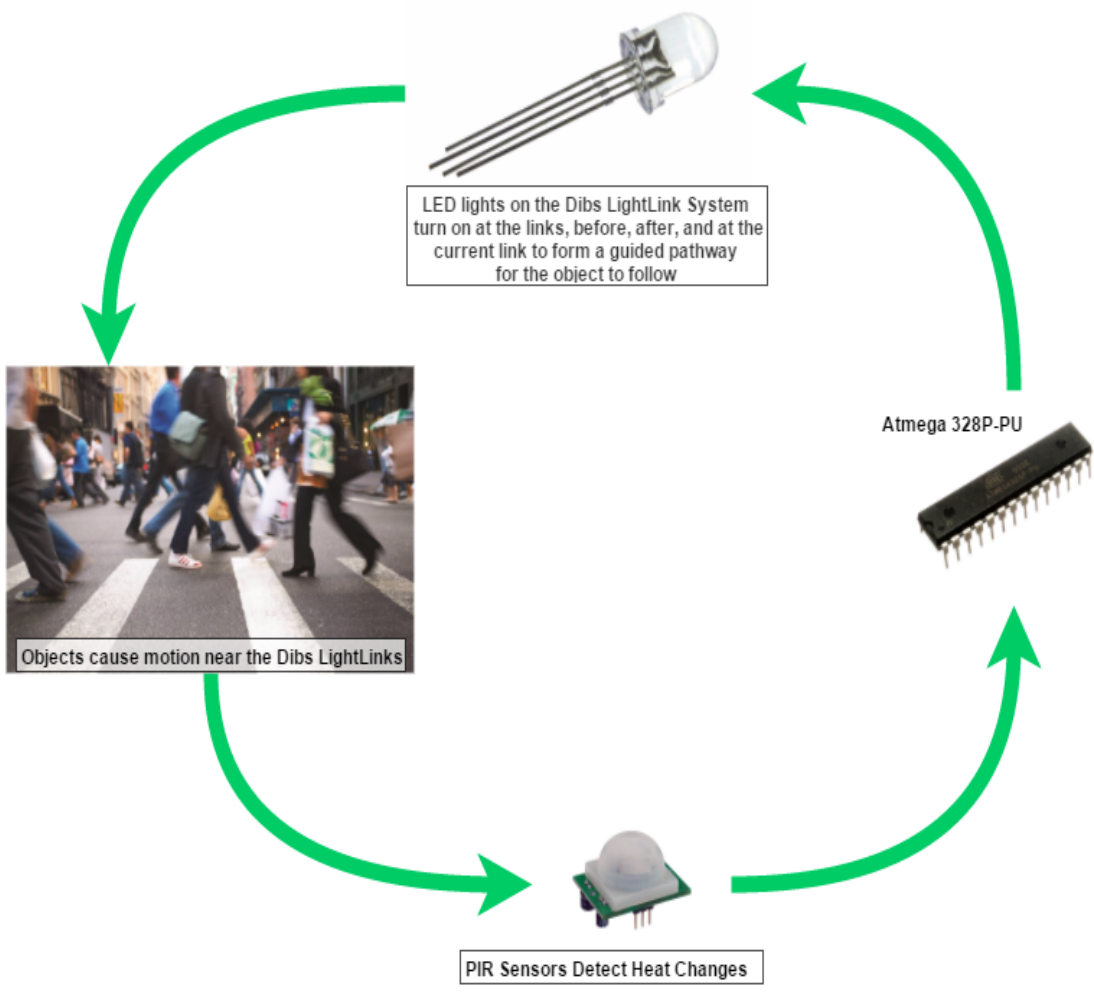


Figure 12.4.1 Dibs LightLinks System Door State Sensor diagram

12.5. Location Tracking

The Dibs LightLinks System has the capability to track an object's location using the PIR sensors on each of the links of the Dibs LightLinks system. The PIR sensors track the radiation given off by objects. This means that if an object appears to be emitting radiation then the PIR sensor will detect its presence. These PIR sensors are the driving force behind the Dibs LightLinks system's Location Tracking Mechanism. As an object moves, the PIR sensors take note that an object is moving and sends a signal down the chain of LightLinks that there is an object in this vicinity. This signal travels to the Atmega328 microcontroller signaling that one of the PIR sensors has detected an object. Depending on which mode the Dibs LightLinks System has been set to, the microcontroller could send a signal to turn that specific Link's LEDs on or the Atmega328 could send a signal to an Interface Link to transmit some sort of alert to the user stating that at Link ID Number X there is some sort of motion happening. A diagram of how the signals are transmitted is shown below.

Location Tracking



12.5.1 Dibs LightLinks System Location Tracking Diagram

12.6. Security Applications

Considering that the Dibs LightLinks System is capable of tracking an object's location, sensing the state of a door using a Door Link, sending notifications wirelessly from the Dibs LightLinks System, to the Dibs LightLinks server, to the user via their own

Personalized Account, it stands to reason that the Dibs LightLinks System actually has some loosely implemented security applications. As such, the Dibs LightLinks System has a security mode that it can be set to. In this mode, anything that would cause a signal to be sent along the Chain of LightLinks (such as the Door Link sending a signal that the door has been opened, or each of the PIR sensors sending a signal that some sort of motion has occurred in that specific Link's location) would send an alert signal to an Interface Link. The Interface Link would then transmit this signal wirelessly to the server and the server would then send a notification to the user through the mobile application, the web application, and if the user so chooses through the user's email as well.

12.7. Outer Casing

The microcontrollers, wires, RGB LEDs, board, and other internal components are weak and susceptible to external influence. To circumvent this, a casing is needed. This case must prevent the insides from being crushed or exposed to the elements. Additionally, the casing must keep the individual devices of the system lightweight and easy to move.

12.7.1. Crush Resistance

The Dibs LightLinks System must be crush resistant. Due to its designation as a pathway, the user accidentally stepping on it should not cause any damage to the internal structure. This requires the external container to be rigid whilst being shaped in such a way that it can hold the internal devices in an efficient manner.

There are many materials that can be used in the composition of the shell, but aluminum and Polyvinyl Chloride (PVC) appear to be two of the best materials for this purpose. Aluminum and PVC are both lightweight materials that can be molded into a sturdy enough casing to be used.

12.7.2. Water Resistance

The Dibs LightLinks System was originally intended to be water resistant. This was because it had been meant to also have recreational purposes that might have lead to placement outdoors. Because of this, the device would have had to have been capable of retaining proper functionality in the rain or when other sources of water drenched it.

Water resistance, however, doesn't cause too much of a strain on what material the casing is made out of. There is a vast range of materials that are water resistant and

numerous methods of waterproofing non-water resistant materials. Aluminum and PVC, coincidentally, happen to both be water resistant, and are therefore good materials for a water resistant casing.

However, again, the Links are no longer outdoor devices, so a simple low sensitivity to the kinds of water found indoors was needed, and a mostly-non-porous project enclosure was used.

12.7.3. Lightweight

The Dibs LightLinks System is optimized for household use. Because of this, the components should be lightweight so the user can easily place them without any assistance. In addition, lightweight materials can be transported easily for both relocation purposes and marketing purposes.

Lightweight, like water resistant, isn't too much of a strain on the materials used. Unlike water resistance, there aren't really methods of making heavyweight material lighter, so we're more restricted to using light materials. Just like crush resistance and water resistance, aluminum and PVC prove to both be optimal materials for this constraint.

12.7.4. Easy to Relocate

The Dibs LightLinks System is not designed to be a permanent fixture. While certain households may use it in such a manner, it shall be designed to be portable. This grants it the versatility to be taken apart and set up in a different layout if the user intends to walk a different route. This also offers the capability to change the path for a different use of "Party Mode" or for outdoor events.

The individual devices themselves do not need to be flexible, as they would be lightweight. This means that each individual link needs to be rigid, but still easy to relocate. Once again, using aluminum or PVC would fulfill this requirement, as they can be rigid structures that are light enough to be moved.

12.7.5. Storage

The Dibs LightLinks System is not designed to be a permanent fixture. While certain households may use it in such a manner, it shall be designed to be easily stored. Ideally, the less space it takes up in storage, the more functional it is. The device should also be able to be stored for lengthy periods of time without suffering any loss of function.

12.7.6. Conclusion

In order to satisfy all the constraints provided for the casing, two materials have proven optimal. Aluminum and PVC are crush resistant, water resistant, lightweight, and easy to relocate

Aluminum, however, has a special property when dealing with certain frequencies of light that may hinder the sensors on the device. It reflects infrared radiation. Because of this, aluminum could cause false positives to occur on the PIR and Proximity Infrared sensors, and would therefore not be the best option for an external casing. PVC, however, does not suffer from this issue, as it has a high infrared absorption rate, and is therefore the optimal material for the casing.

13. Physical Design

The Dibs LightLinks System is composed of four distinct flavors of Links, Door, Interface, Battery, and Standard, and one Link extender, dubbed the Connector. Each one has its own job to perform for the system, and therefore must be physically different.

13.1. Block Diagram

This block diagram is an updated version, and shows the exchange of the Wifi Unit (WFU) for the Bluetooth Unit (BTU).

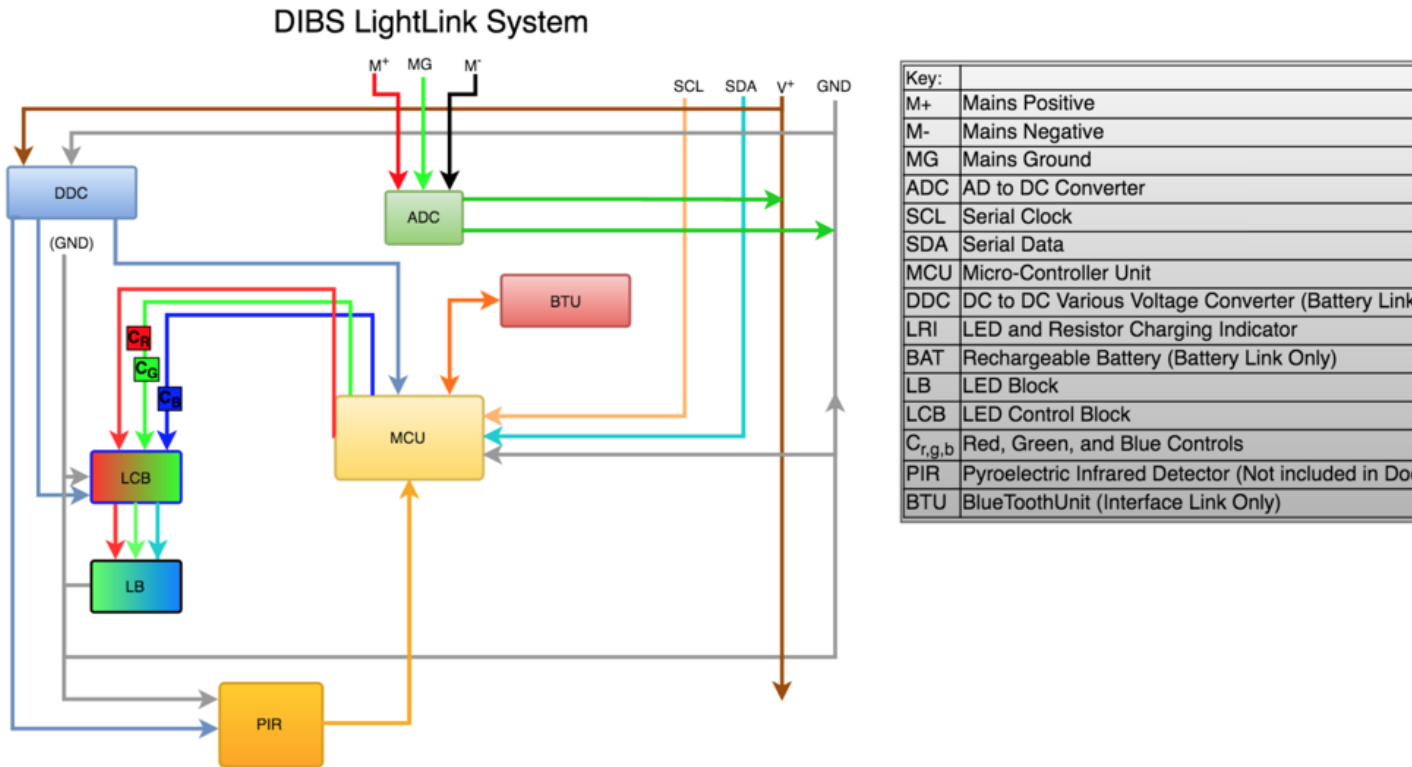


Figure 13.1.1, Main Block Diagram

13.2. PCB Layouts

The first PCB board is the general board. It contains excess elements which will not be used in every link. However, its generality allows for easier duplicate printing. It is seen below in *Figure 13.2.1*.

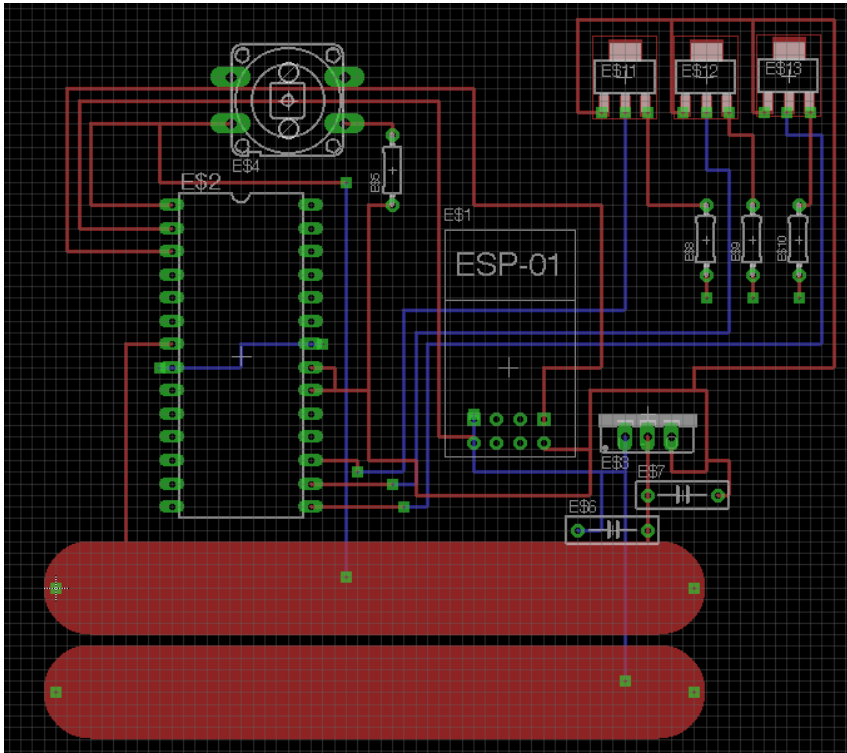


Figure 13.1.1a, Original Board

Given the modular nature of the Dibs LightLinks system, as well as the general principle of cost savings through high-volume production, only one main board was designed. This is to say that each board has excess connections and traces designed so as to make it possible to convert into any board. This pluripotency is very useful, and is not seen directly in the board below (*Fig. 13.1.1.b*), but note the lack of stabilizing capacitor on the voltage regulator. This is because each and every board has at least the ground and vcc connections of either the door link sensor (IRVCC, IRGND; left), or the Bluetooth module (VCC, GND; right) free, which saves space.

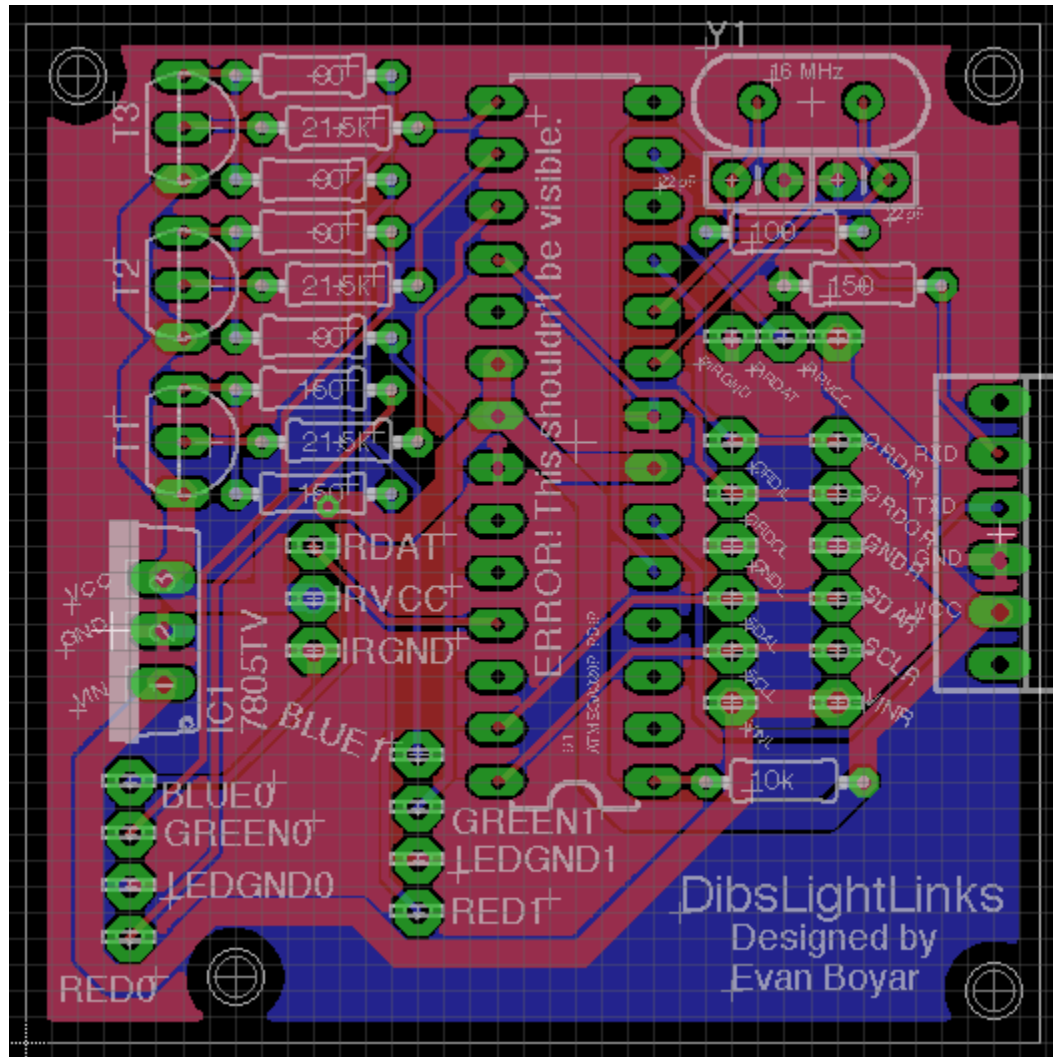


Figure 13.1.1b, Final Board

Also note the presence of the Bluetooth module connections even on the Links which do not have a Bluetooth connection--which is to say all Links save the Interface Link.

Each Link is connected to all other Links in the chain via the Link Extenders (LEs). The connections to those Les are soldered directly to the board. Those solder points are the two rows of six through holes which may be seen to the right of the Atmega328.

The other board is the battery charging circuit, shown below in *Figure 13.2.2*. It is a freestanding board, separate from the main one, and is only present in the one Link that involves charging--the Battery Link.

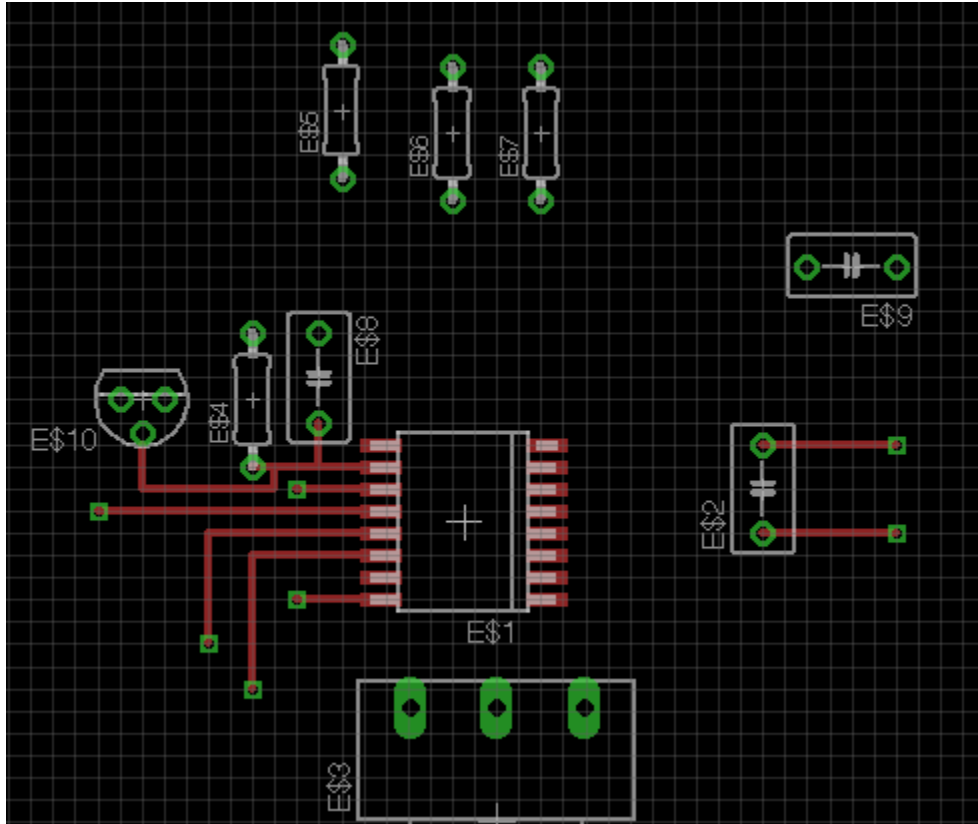


Figure 13.2.2, Charging Board, unused

13.3. Functionality Diagrams

In the early stages of development, functionality and concept diagrams were drafted to show the flow of data and signals throughout the Dibs LightLinks System. There are also slightly more detailed functionality concepts diagramed for each type of the five LightLinks below.

In *Figure 13.3.1*, the inputs and outputs are shown for the overall Dibs LightLinks System.

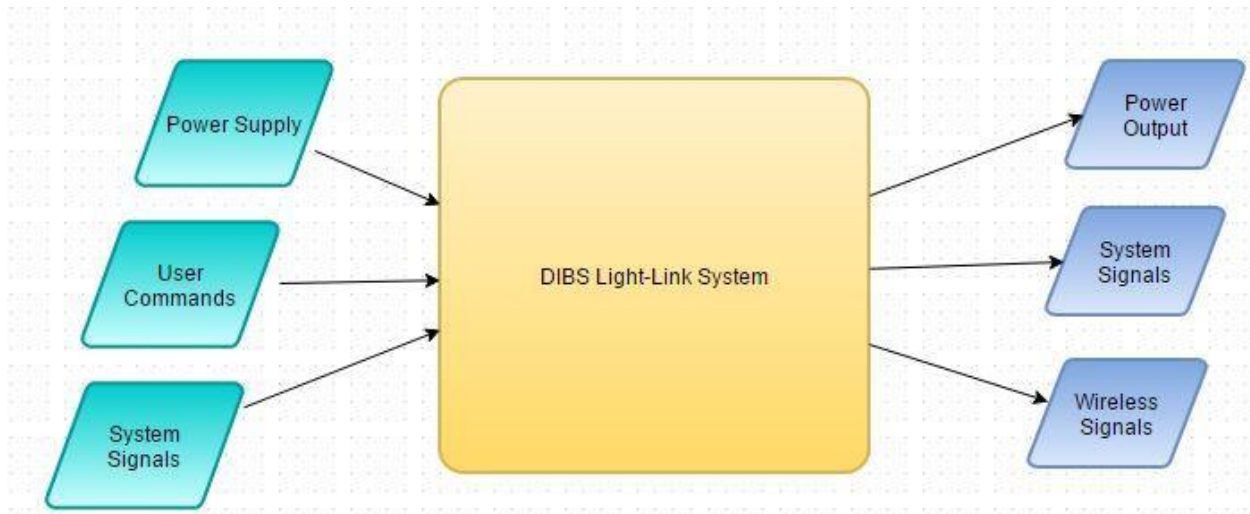


Figure 13.3.1, System Diagram

In Figure 13.3.2, the input and output signals for the Door Link are shown.

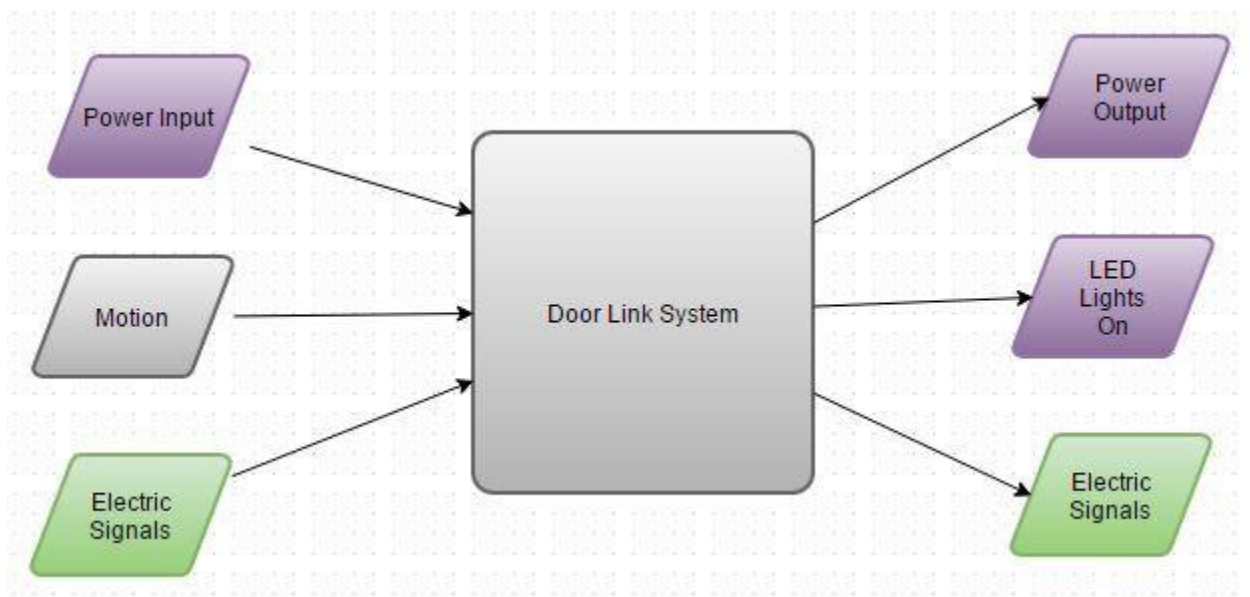


Figure 13.3.2, Door Link System Diagram

In the Figure 13.3.3, the different inputs and outputs are shown for the Interface Link. This Link will be the most complicated of all the Links. It will be the Link controlling all other Links.

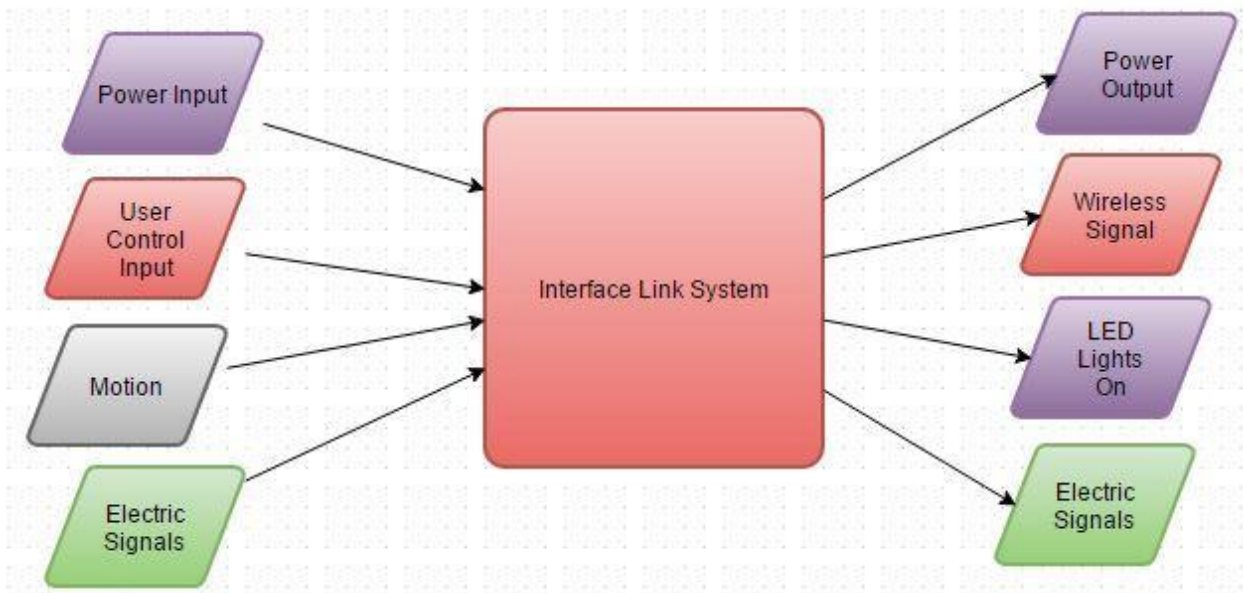


Figure 13.3.3, Interface Link System Diagram

In Figure 13.3.4, the inputs and outputs are shown for the Battery Link

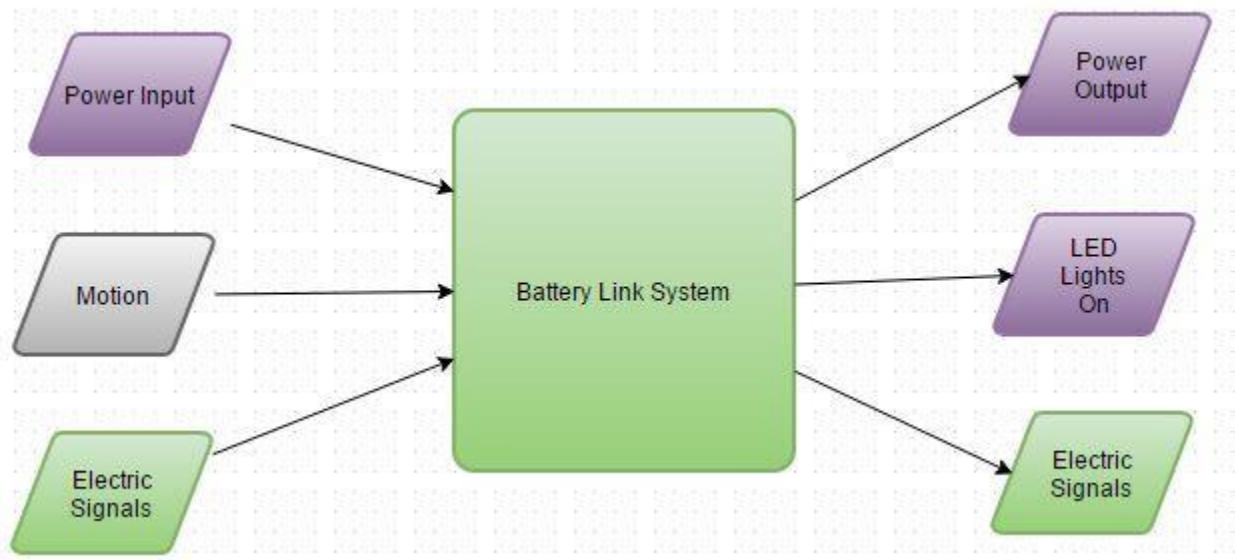


Figure 13.3.4, Battery Link System Diagram

In *Figure 13.3.5*, the inputs and outputs are shown for the Standard Link, which is the most basic of all the Links.

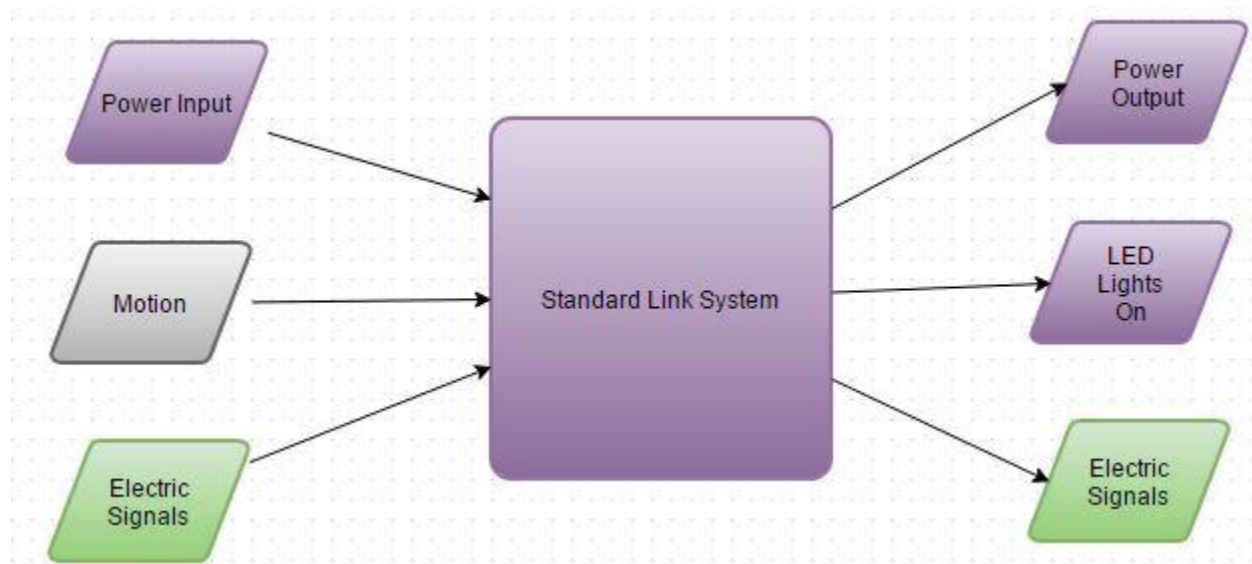


Figure 13.3.5, Standard Link System Diagram

13.4. Design Philosophy

The Dibs LightLinks System is designed for commercial use and optimized for everyday household application. Because of this, the physical structure of the Links needs to follow strict guidelines. These guidelines are to provide security, safety, and stability.

In order to provide security, a system must be able to keep its user safe from harm either by removing threats or informing the user. In the case of the Dibs LightLinks system, the user shall be informed of a potential threat. In order to do this, the sensors that can detect intruders shall be oriented in such a way that grants them optimal lines of detection.

Because the Dibs LightLinks System is to be set up on the floor, safety is a massive issue. The device's primary designation is to light up and assist the user in getting from one point to another safely and without being hindered by obstacles. If the device stands out too much or doesn't have a user friendly shape, the user may accidentally trip on it. To this end, the individual devices must be small enough to not be tripped on and in a shape that stepping on them would not cause damage to the device or the user.

The stability of this system is defined as its ability to endure active use. In order for it to be stable, it needs to be able to withstand being stepped on, constructed and deconstructed, and used constantly. To provide this, the external structure should abide by previously set conditions and the internal structure must be designed to be compact and power efficient.

Overall, the system needs to be operable under all normal household conditions and commercial non-household use should also be taken into account, such as party applications and office settings.

13.5. Door Link

The Door Link is designed for placement in doorways or in locations where a door may be opened. So as to avoid conflict with the door's movement, the Door Link is to be small enough to fit under the door without compromising the system or providing a potential hazard. Its design is different from the other links in that it is designed to retain the connection under a door and provide feedback on the door's current position.

13.5.1. Requirements

In the final production of the devices, certain requirements have been altered. Originally, we were interested in a device of dimensions not exceeding 300 cm by 4 cm by 4 cm, (l, w, h). However, the final device ended up having slightly different dimensions in order to have a more eye-catching shape, sacrificing a thin width for a low height, which was a more desirable quality. Additionally, the shift from having outdoor functions to mainly indoor use meant that water resistance and batteries were no longer needed, obviating the specifications relevant to those applications.

Moreover, the move from WiFi to Bluetooth communications means that the links no longer connect to the Internet. Instead, they connect directly to the smartphone through a Bluetooth-application interface, meaning that WiFi standards no longer are relevant.

However, the following specifications were met:

Specification	Original	Actual	Passed?
Turn-on delay per link	<5 ms due to detection delay	3.1 ms	Yes
Signal propagation delay	<20 ns	~5 ns	Yes
Mobile interface delay	<2 s	~0.3 s	Yes
Maximum mass	<0.23 kg	0.20 kg	Yes
Power	<2 W	0.39 W	Yes
Motion sensing range	>2 m	5 m	Yes
Door sensor height	<2.54 cm	1.9 cm	Yes
Door sensor triggering	Open vs closed	Open vs closed	Yes
Signal conversion	Parallel to serial	Parallel to serial	Yes

Figure 13.1, Specifications table

14. Application Software

In order for Dibs LightLinks System to be fully functional, application software must be implemented to give the user a way to interact with the system. There are many different types of application software out in this day and age, so deciding on which kind to use for development was more than just a quick decision. Research was conducted on mobile operating systems to find the best system to meet the demands needed for the Dibs LightLinks System software. Once a mobile device was chosen to be used as the application, the user interface had to be built along with all of its functionalities. In the sections below, more detail is given on how each of these specific tasks was handled.

14.1. Types of Mobile Operating Systems

Before beginning to make an application to work with the Dibs LightLinks system, much research was conducted on the different kind of mobile phones and the operating systems that run on them. In order to get a better idea of which operating system would best suit the needs of the application, a comparison of the many different mobile operating systems out there was conducted. These systems include Apple's IOS platform, Google's Android platform, BlackBerry's OS, and Ubuntu Touch OS.

14.1.1. Apple IOS

Apple IOS is the second most popular operating system for mobile applications with around 35% of total users having an Apple operating system^[1]. Although it is second in terms of number of users using this OS, it is number one in profits as of the first quarter of 2015 with its profitability being around 70% more than that of Android^[2]. It is developed by Apple and strictly for their products. IOS is used on Apple iPhones but not just the iPhone, as it also is used on the iPod Touch, iPad, and even the new versions of Apple TV.

When this platform first came out, no third party applications were allowed to be downloaded on the OS for security purposes. However, like mostly anything that has security protocols, hackers found a way to get around this issue and take away the security protocols preinstalled on the Apple OS. Someone doing this to their phone would be doing what is called "jailbreaking" their iPhones. Since this was being done too much and messing with the security of the IOS software, Apple decided to put out a release that finally allowed third party applications to be downloaded.

IOS has gone through many version updates, with the current version of IOS being IOS 9. IOS also has a feature called SIRI in which a user can ask it to do something for them and it will try to handle the request. This has helped with its continuing popularity. As far as programming for applications in IOS, there is a few Integrated Development Environments (IDEs) that make programming for applications a bit simpler than trying to do everything from command line. The most popular IDE for programming in IOS is Apple's Xcode software. This allows application developers to program in either Objective-C or Swift programming languages.

14.1.2. Google Android OS

Google's Android OS is the most popular OS being used in the world with about 57% of mobile users having this OS^[1], mainly due to its cost effectiveness. There are over

24,000 Android devices out in the internet of things as of August 2015^[3], with many devices at low costs compared to its rival Apple. Most of Android's software is open source, which makes finding documentation about how its OS works less task demanding compared to Apple OS which is proprietary information.

The interface is very user friendly and has many options for customization; unlike the iPhone which has a strict layout with minimal customization settings unless the user does a jailbreak on its OS which is not recommended by Apple. Even with Android's many customization settings, the Android application market (known as the Google Play Store), also allows third-party applications to be downloaded which can further customize the look of the Android system. These applications can even allow Android to take on the look of another operating system such as Windows Phone.

Android Applications are written in the object oriented programming language known as Java, which is a high level language that has many IDEs available to make programming for applications much simpler compared when mobile applications first came out. The most user friendly IDE currently available is Android Studio, which was developed for the specific purpose of creating Android Applications.

14.1.3. BlackBerry OS

BlackBerry released a mobile platform for smartphones called BlackBerry 10.0 which was announced on January 30th, 2013^[4]. This operating system is proprietary and only made to run on devices developed by BlackBerry. The first phone made to run the BlackBerry OS on smartphones was the Z10^[5]. Since this smartphone, there have been a line of devices from BlackBerry to run this OS. These devices include the Z10, Z30, Z3, Q10, Q5, P'9982, P'9983, and Leap^[6].

Since the OS is proprietary, there is not much documentation on the way it is designed, therefore programming applications for this OS is limited. This is the underlying reason why the applications available on BlackBerry World are scarce compared to the Apple IOS store and Google Play Store. There is an IDE for programming applications for BlackBerry called Momentics^[7]. The programming languages used for these applications are C++, C, or QML. There is however a new BlackBerry Phone which runs Android OS instead of BlackBerry's proprietary OS. This device is called the BlackBerry Priv^[8].

14.1.4. Ubuntu Touch OS

The mobile OS, Ubuntu Touch, is linux based. It was released on January 21st, 2013 for mobile developers^[9]. This was just a preview though and not a stable version. A more stable version, version 14.09, was developed and released to manufacturers in September 2014^[10]. Since the OS runs off of a Linux kernel, it can be downloaded onto Android devices to be ran as the OS.

Currently there are only three smartphones that come preinstalled with Ubuntu^{[11][12][13]}, all of which are not sold in the United States mobile market. Since Ubuntu is open source just like Android, developers can dig deep into the source-code and make changes as seen fit to benefit their needs and wants on their device. The goal of this OS was to make it convergent, which means to make it compatible on all devices such as smartphones, tablets, laptops, TVs, and even desktop computers. Ubuntu comes with the basic core applications that many other operating systems have as well such as: calculator, file manager, calendar, media viewer, and mail.

14.1.5. OS Selected for Dibs LightLinks System

After comparing the different operating systems for mobile devices, the decision was made to go with Google's Android device. The reason being such a wide variety of devices run Android OS, giving a wider user space and also because the development of the application is written in java programming language, which is one of the languages that is easy to use due to it having many prebuilt libraries that can be accessed. There are also many Application Program Interfaces available for connecting Android devices to other devices on the same Wifi network which will prove helpful in the overall development process.

Also with all the functionality available in Android Studio, including tips and help documentation, developing the application in Android seemed the most plausible. For the application side of the project, an Android device will be used to communicate with the Dibs LightLinks System lighted pathway. This consists of a mobile application that allows the user some freedom to personalize the LightLinks to their own specific wants and needs.

14.2. User Interface

The user interface for the mobile application was built in Android Studio. The user interface is where the human and machine interact; in general, this would be the application (the users) communicating with the LightLinks (the machine). In order to

have a nice user interface, the design was made as simple as possible, making clear all the options the user has to configure the LightLinks.

There are many different forms of user interfaces. The one that the Dibs LightLinks System will be using is the Graphical User Interface, or GUI for short. This gives the user the ease of use, while taking care of all the complex communication between the user and machine behind the scenes. Android software provides a variety of pre-built user interface components that a programmer can select from to begin building a user interface. The Dibs LightLinks System user interface will consist of 7 screens, with various screen layouts that will allow for a smooth running user experience. The User Interface will consist of:

- A main screen that will be the Home Screen
- A Configure screen
- Five different Mode configuring screens

14.2.1. Home Screen

This is the first screen that the user will see when they open the application. The user will be able to navigate to the different modes of operation as mentioned in **section 6.2.2** through this screen along with choosing to configure them. The two types of modes, Active and Passive, divide the modes. There are three Active modes and three Passive modes. To the left side of the screen there are the three Modes: Standard Pathway Mode, Party Mode, and Sleep Mode, going from top to bottom. To the right side of the screen are the three Modes: Power Saving Pathway Mode, Crowd Mode, and Security Mode, going from top to bottom.

This screen uses a Relative Layout, which allows components of the screen to be placed relative to one another to achieve the overall look of the User Interface. Android Studio has a nice way of seeing what a screen is made up of. This is known as a component tree in which all the components on the screen are shown in a hierarchy in the order they are placed. The component tree shown in *Figure 14.2.1.1*, shows the different layouts used in this screen along with the other User Interface components that were used to create this screen.

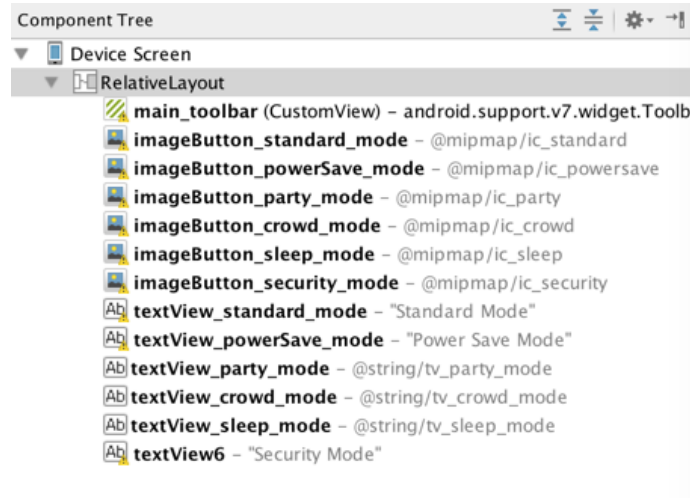


Figure 14.2.1.1

As displayed in the Home Screen component tree, there are 13 components used for this screen configuration. The toolbar is across the top of the screen, with everything else below it. As with the relative layout, all components are relative to one another. The Standard Mode button is below the toolbar and to the left of the screen. The Power Save Mode button is below the toolbar and to the right of the Standard Mode button. Beneath these buttons are text fields, with each text field showing the respective modes name. The Party mode button is below the Standard Mode text view and the Crowd Mode button is beneath the Power Save Mode text field. Text fields for Party Mode and Crowd mode are beneath their respective buttons. The Sleep Mode button and Security Mode button are beneath the Party and Crowd text fields, in which these modes text fields are beneath them. This allows for a clean looking User Interface without having to do any complicated nesting of layouts.

Originally, the Home Screen was developed using a Linear Layout, in which many more Linear Layouts were used to create the same look used with the Relative Layout. But after further research, it was found out that it is usually not a good idea to use nested layouts whenever an overall layout is deep^[14]. This means it has many levels of layouts just to achieve one complete layout. Nesting layouts usually hinders performance because it puts an overload on the drawing of screen components taking place in the Android Application. As shown in *Figure 14.2.1.2*, there is a screenshot of the user interface for the Home Screen, which may be subject to change, further into the development process.



Figure 14.2.1.2, Home Screen

14.2.2. Configure Screen

The configure screen will look just like the Home Screen with the exception that it will say “select a mode to configure” at the top of the screen. As shown from *Figure 14.2.2.1*, the component tree has a very similar design as the Home Screen’s component tree, with the only difference being a text view added to the top of the layout instead of a toolbar since it will not be needed in the configure screen. This puts the text above all of the modes to choose from, which is what was needed to achieve the desired look of this screen.

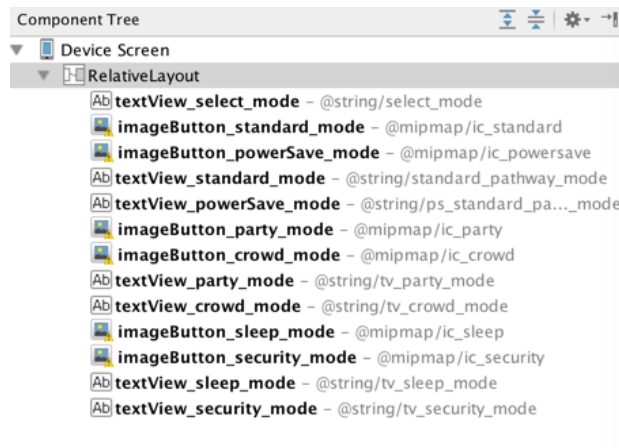


Figure 14.2.2.1, Configure Screen Component Tree

The screenshot of the Configure Screen is shown in *Figure 14.2.2.2*. This screen will be used to let the user know they have chose to configure a mode. Depending on which button is pressed, the user will be either selected with the color picker if color is the only thing that can be configured in the mode selected, or they will be taken to another screen where they can customize all the different things the mode is capable of customizing. Once a mode has been configured and the user hits “Update” on that modes configure screen, the user will be brought back to this screen in which they can select another mode to configure if they want or they can go back to the Home Screen with the simple click of the back button on their device.



Figure 14.2.2.2, Configure Screen

14.2.3. Settings Screen

The settings screen is where the user can go to make changes to the application and the LightLinks system overall. It is a fairly straightforward design as shown in *Figure 14.2.3.1*, the component tree shows there is only one Linear Layout of vertical orientation with four buttons inside. Each one of these buttons will provide some function to happen in the application. After consideration, user login was taken out of the application and the settings page was decided not to be used since there were other ways to get to change the colors and edit the modes.

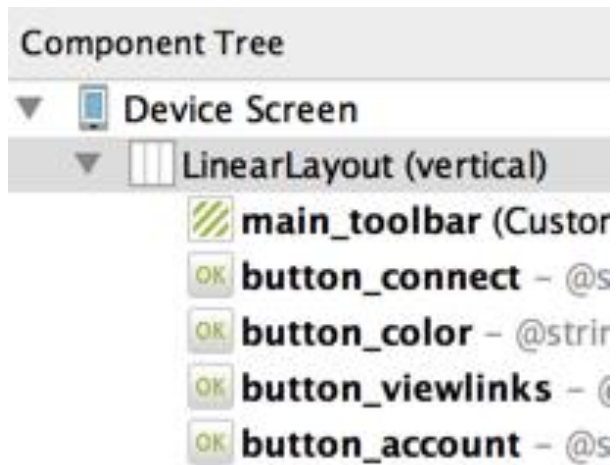


Figure 14.2.3.1, Settings Screen Component Tree

The screenshot of the Settings Screen is in Figure 14.2.3.2, and shows the design of the screen.

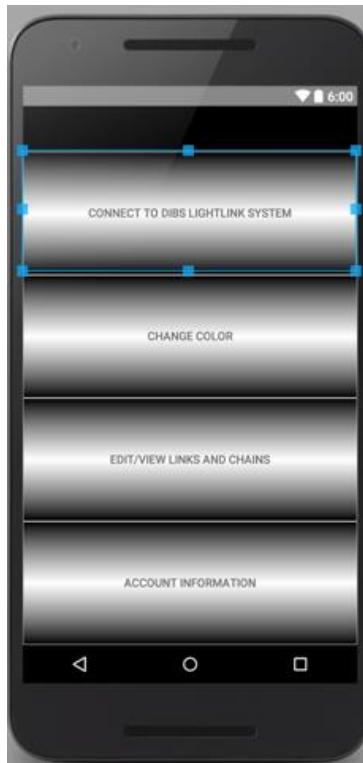


Figure 14.2.3.2, Settings Screen

14.2.4. Link/Chain Viewing Screen

This screen will be where the user gets a visual representation of the Dibs LightLinks System they have created. As mentioned in **Section 1**, Links that are connected together make up a Chain. This Chain is what the application will show in a listview. If for some reason, only one Link is in the Chain, then this simply becomes just a Link.

This screen will show the Link/Chains that are all running in the Dibs LightLinks System and has a button to automatically order Links in a Chain. There is also a button for manually adding Links to a Chain. The component tree in *Figure 14.2.4.1* shows what was used to make this screen. As shown in the figure, the design is pretty simple with only the one Linear Layout of vertical orientation as the parent layout, with a toolbar, two buttons, and a scroll view inside of it.

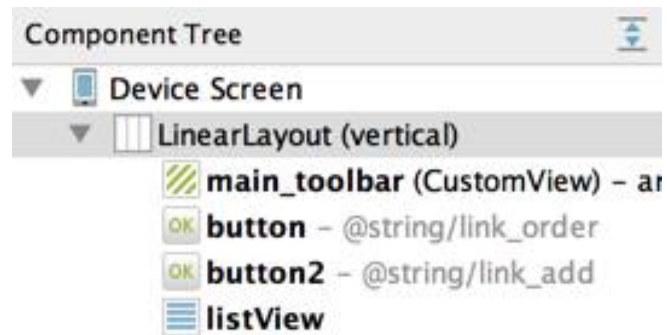


Figure 14.2.4.1, Chain View Screen Component Tree

The screen will look different depending on how many Chains is detected in the system. If none are detected, there simply will not be a list view present. The screenshot for this screen is in *Figure 14.2.4.2*, as simple inspection shows, the list view is a generic view because it uses an Adapter, specifically an ArrayAdapter, which handles the information stored in an array to populate the view.

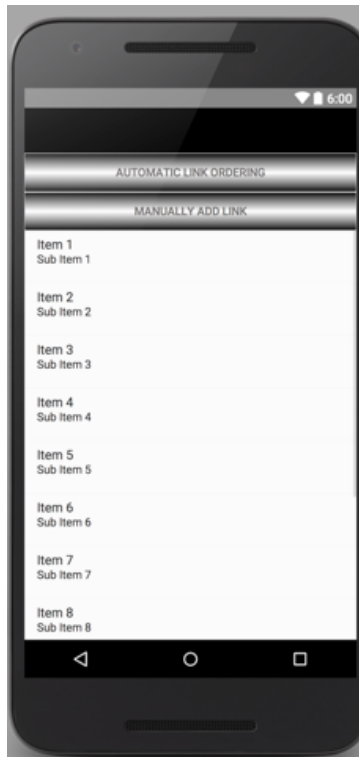


Figure 14.2.4.2, Chain View Screen

Due to difficulties in the wire configuration of the project, the chain viewing screen was not added to the final project because communicating to each link for order configuration was made impossible.

14.2.5. Mode Screens

There are a total of 6 different modes that Dibs LightLinks System can function in. As mentioned in **Section 6.2.2**, these modes all have different default settings such as light colors.

All of these screens follow the same simplistic design. There is a text field with a description of what the item the user is trying to configure is i.e Color, timer, etc. Then to the right of this is either another text field that is editable for user input, or a drop down menu with items to select. At the bottom of the screens there is an update and cancel button.

To give a general idea of what each screen looks like, the most complex of these mode screens is shown in *Figure 14.2.5.2*, the Party Mode Configure screen. This is the mode that has the most user control settings, therefore all other mode screens will look similar to this screen but with less options for configuring the mode. With modes that only are capable of changing the color, instead of going to a configure screen, the application will simply show a dialog to allow the user to pick the desired color. To show the components of this layout, the Party Mode component tree in *Figure 14.2.5.1* has been included. By inspection of this figure, it should be clear that the design is simple, with just one Relative Layout needed.

Notice there is a new component used in this screen that has not yet been discussed, the Spinner. The Spinner component is used to make drop down menus with predetermined items. It functions in the same way as the listView component does by using an Adapter to fill the view. The Spinner has been customized to take on the desired font size and text color of the application theme.

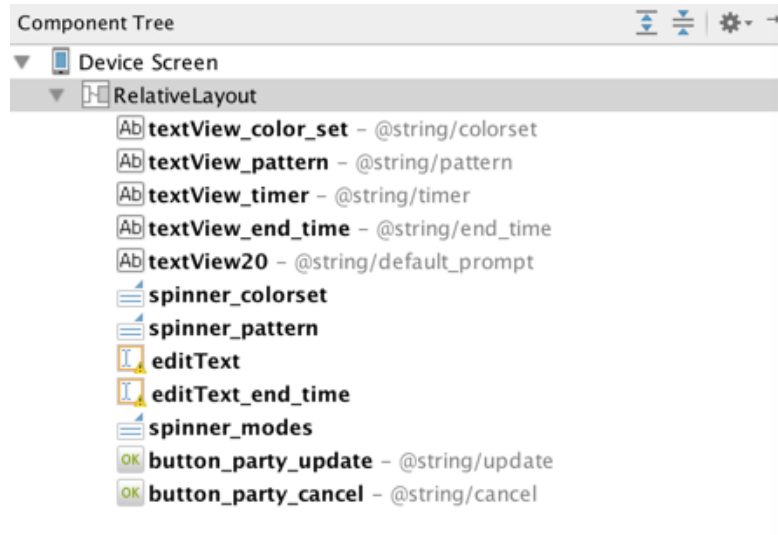


Figure 14.2.5.2, Party Mode Component Tree



Figure 14.2.5.1, Party Mode Screen

Now that the user interface has been discussed in detail on how each screen was made, it is time to talk about how all of these screens function together to make a smooth-running application.

14.3. Functionality

Since an Android application will be used for the Dibs LightLinks System project, its functionality will be based on direct manipulation. Direct manipulation is when a user is presented object to interact with and based on what they do with these objects, different events happen. The main features of the application include:

- Connecting to Dibs LightLinks System through bluetooth
- Navigating between the six different modes
- Configuring these modes to the user's personalized liking

- Viewing Chains and/or Links
- Editing user account information

Further detail on each of these features will be reviewed for the next few pages.

14.3.1. Connecting to Dibs LightLinks System Through BlueTooth

When the application is opened for the first time, it will not know which microcontroller to connect to. Since the Dibs LightLinks System is controlled by an Arduino microcontroller, there will need to be a connection between the microcontroller and the mobile application. The Interface Link will be the Link with the master microcontroller that will let all other Links know what to do, therefore a connection must be made to this Link through bluetooth.

This is achieved by having the HC-05 bluetooth module as part of the Interface Link. Data will be sent through bluetooth and the bluetooth module will process this data and deliver it to the Atmega328-PU microcontroller. The application is designed for easily connecting to the Dibs LightLinks System. The user would click the icon that looks like a connection sign in the toolbar at the top of the screen. The user will be shown all available bluetooth devices around and then they would have to select the “HC-05” that is displayed in the list. If this is the first time connecting to the device, the phone and bluetooth module will have to be paired to on another.

If a connection was established a message will come across the screen for a short period of time showing “Connected to the HC-05”; if no connection was made, an error message will come across the screen showing “Could not connect to bluetooth”.

14.3.2. Navigating Between Modes

When the application is first opened, the user will be viewing the Home Screen as mentioned in **Section 14.2**, which presents the six different modes of operation of the Dibs LightLinks System. This is where the main functionality of the application takes place.

The default mode that is started when the application is opened for the first time is Standard Pathway Mode. If a user wants to switch to a different mode, all they have to do is click on the mode of interest in which a prompt will come up asking the user if this is the mode they want to switch to. If the user clicks “ok” then the mode will change to the new one that was just selected.

However, if the user accidentally clicked this mode, they can hit “cancel” and the mode will not be changed. When a new mode is selected, its default settings as specified in **Section 6.2.2** will be used until the user configures them to their own personalized liking through the Configure Screen.

14.3.3. Configuring Modes

If a user is not satisfied with the default settings of the mode they selected, or even if they just want to play around with the mode capabilities, they can choose to configure the modes by going to the Configure Screen. This screen can be reached by clicking the wrench icon on the toolbar at the top of the screen. Once the user is in the Configure Screen, they can select a mode to configure by clicking on that mode button. When the user clicks the mode button, they will be taken to that mode’s screen in which all of the user controls are displayed. These user controls will vary depending on which type of mode is selected.

Note that some functionalities of modes are not able to be configured such as the sensors being on or off and lights being on or off. In order to give each mode its own distinction between others, some settings must stay unchanged.

Due to constraints on the wires used for the system, configuration of modes was not capable on the actual LightLinks, but was tested and working in the prototype stages of development.

14.3.4. Viewing and Adding Links/Chains

When the application is first connected to Dibs LightLinks System, they will be taken to the screen where Chains and Links can be viewed. This screen will not show any Chains yet because only connection has been established to the Interface Link. Therefore the only thing that will be in the list view will be the Interface Link.

Seeing how a user might create a long lighted pathway with many LightLinks, manually inputting Links can become tedious. For this reason, a better way of adding the LightLinks connected in a Chain all at once into the list was implemented into the application. Automatic Link Ordering will be the desired method for displaying Links in a Chain. The way this is supposed to work is given in **Section 12.3**, which gives a great visual representation as well as a textual explanation. In order to find if there are any Chains, the user can click the “Automatic Link Ordering” button in which the Interface will send out a signal to all of its connected Links and determine how many other Links

are connected and which order they are in based on the signal it receives back. Once this happens, the list view will show the Chain that the Interface Link has control over.

If for some reason the automatic Link ordering fails, the user will be asked to manually input the Links that are connected. A screen will pop up to ask for the links name and serial number. After the user finishes inputting information, they hit the “ok” button and return to the Chain viewing screen of the application in which the new links just added should show in the list now. If there is an error during this process, it will notify the application, which in turn will notify the user.

This functionality was taken out of the application due to wire constraints on the system.

14.3.5. Editing Links

The scrollable list view will not only display the LightLinks Chain, but also allow the user to edit each one individually by long pressing the intended Chain a user wants to edit. When the user long presses the field, it will give them the option to edit or if this was done by mistake, the user can get out of this by pressing elsewhere on the screen. Another implementation was considered for editing which would entail swiping left on an item, in which the edit icon would show up. This could be the way it is done in the final product. But for now, long pressing is the way to go.

By default the Chains will be named after their Interface Link which will be a number. If a user wanted to change the name of the Chain to something different, this could be achieved by the long pressing of the Chain.

Note: This functionality was taken out of the system due to wire constraints on the system.

14.3.6. User Account Information

The mobile application will have the capability to create a user account that will be linked with the web server Dibs LightLinks System will use as mentioned in **Section 15**. Therefore the application will be linked to the server and the Interface Link. In order to allow full functionality of the security modes, a web server must be built to communicate to the user when they are away from home. The user can create an account on the web server and then use this same account in the mobile application. To edit this account, the user can go to the Settings Screen and select “Edit User Information” This will bring up the general account information such as username, password, phone number, and email. The user can then change their password and other information directly through the mobile application and it will update the information on the server as well.

Note: This functionality was taken out of the final design due to not having a user login process anymore.

14.3.7. Changing Colors of Lighted Pathway

A neat feature that is being implemented into the Dibs LightLinks System is the ability to change colors of the LEDs. To do this from the application, the user will go through settings and select “Change color” which will then give the user options on the colors available. Another option would be to select the configure icon in the toolbar and choose a mode to edit its color from its configuration screen. Since Dibs LightLinks System uses Red, Green, Blue (RGB) LEDs for its design, a signal will be sent from the application to the LightLinks which will differ depending on the color the user wants.

Note: Due to wire constraints on the system, changing colors of the lights is only possible on the prototype circuit on the breadboard. All modes will be using their default colors.

14.3.8. Security Functionality

There are two types of security modes in the Dibs LightLinks System, Daytime and Nighttime. Both of these modes are made to notify the user if someone is detected in the pathway. This includes a human being detected by the PIR sensors as well as if the Door Link detects a change. If someone is detected while these modes are the running modes, the Dibs LightLinks System will notify the user. This can be a very great functionality that users will likely take advantage of; especially if they have no other security system in their homes.

Notifying the user is done by first having the Dibs LightLinks System send a signal to the web server. The web server will then send a signal to the mobile application, in which the user will be notified that someone is in their pathway. When the user gets this notification, they have three options to choose from on what to do.

The first option would be to notify the authorities, which should only be used if the user knows for a fact that no one is at home. The next option would be to try to scare the person away by activating the security lights to blink red. The final option would be to do nothing at all, simply ignoring the notification.

Note: After further consideration, these two modes were decided to become just one mode called “Security Mode” in which the lights will be off and will blink red whenever someone is detected in the pathway.

14.4. Class Diagram

The mobile application requires a few main classes to allow functionality. Since there are multiple modes to play with in the application, a Mode class will be used in which all the data for the mode will be stored. Also since Dibs LightLinks System has user accounts being used, a User class will be used to store all the information of a specific user.

These classes are shown in *Figure 14.4.1*. As clearly shown, the Mode class is used to create an object for each one of the seven modes of operation. These modes each have their own methods which can manipulate the Dibs LightLinks System.

The User class will only have one method, `updateInfo`, which will be used when the user makes any changes to their information in the application and then hits the “update” button. Additional classes may develop as the application becomes more complex to handle all the features.

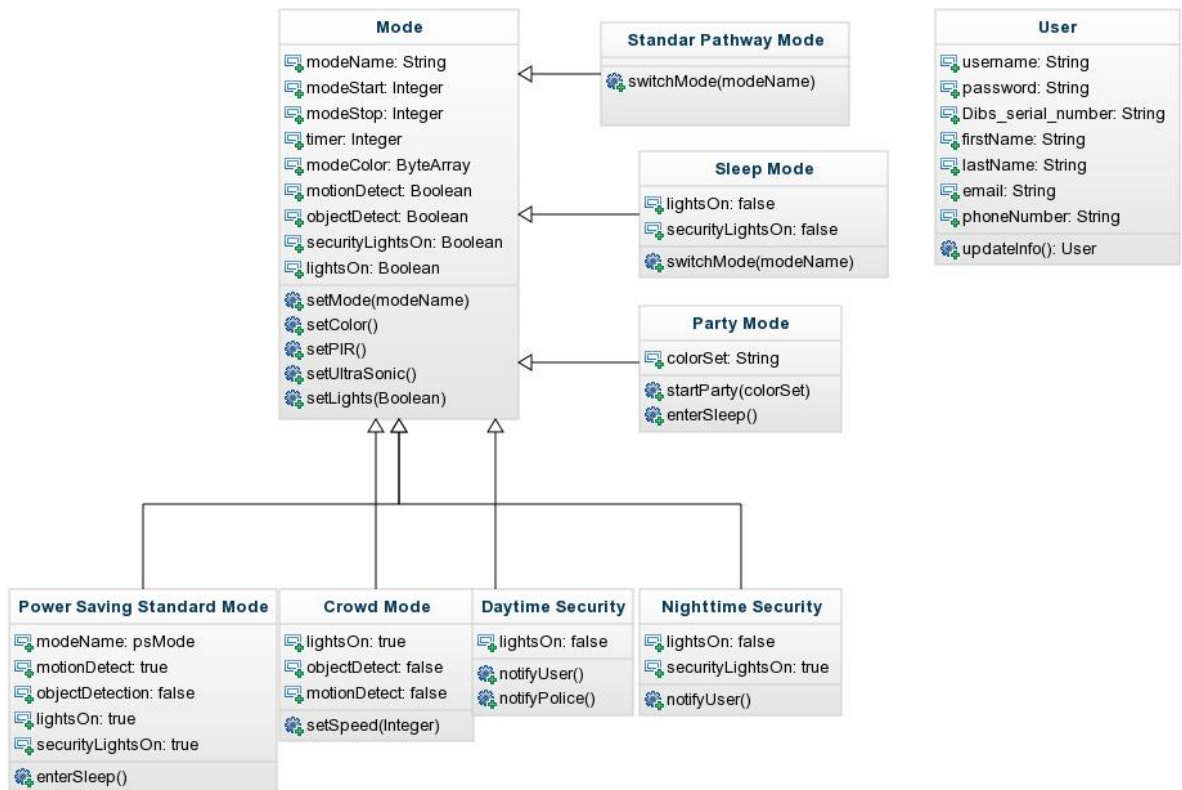


Figure 14.4.1, Class Diagram

14.5. Communication with Microcontroller

The way the application will communicate with the microcontroller will be through Wifi using serial communication. There is a three step process in this communication. The first thing that has to happen is establishing a bluetooth connection on both the mobile application and the Interface Link. Then what happens when a button is pressed in the application is a signal is sent over bluetooth to the bluetooth module. The bluetooth module sends receives data from the application and sends it to the Interface Link.

The communication between everything is displayed nicely in *Figure 14.5.1* to give a visual representation of what was just discussed. Note that the blue arrows indicate the

communication route from the application to the microcontroller and the green arrows indicate the communication route from the microcontroller to the mobile application.

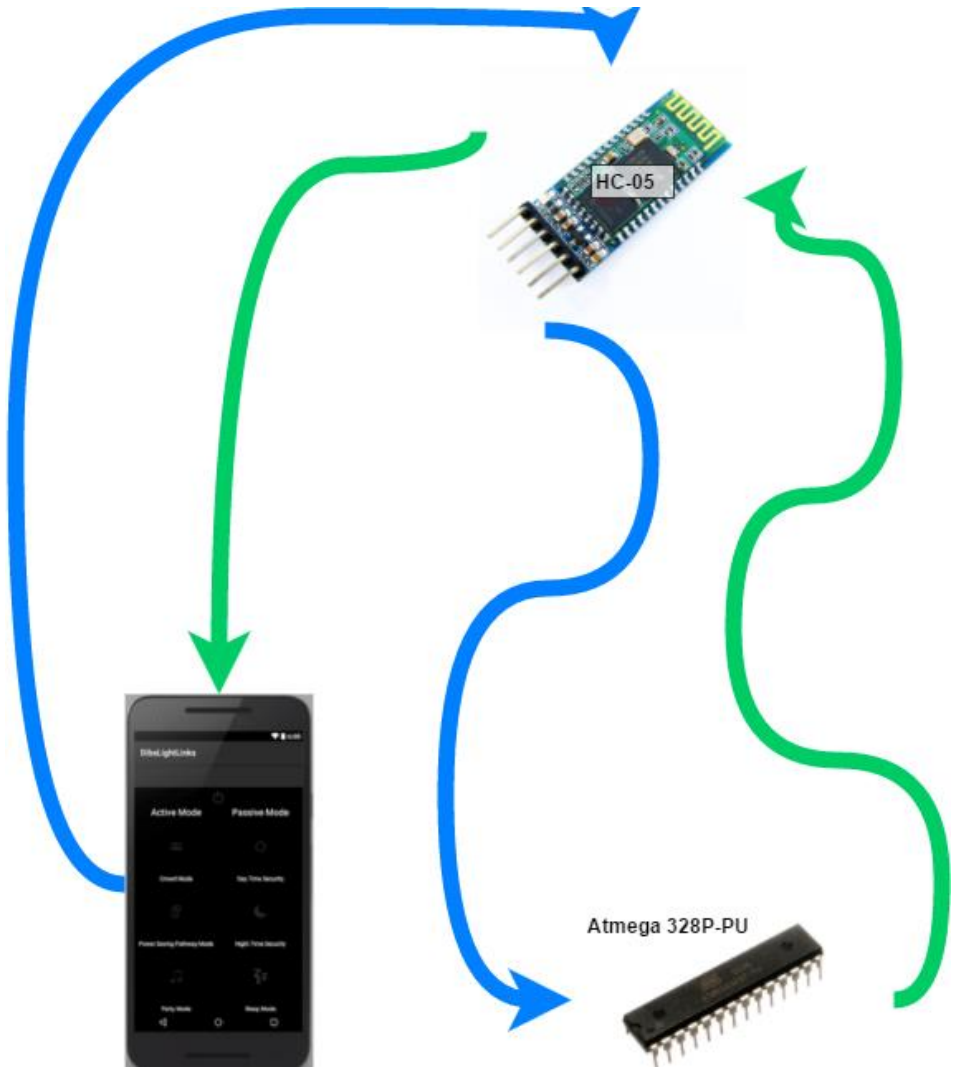


Figure 14.5.1, Communication Diagram

15. Server Software

In order for the Dibs LightLinks System to be able to perform a good portion of its inherent functionality it has to be able to communicate over wireless channels to other devices via the internet. It is for this reason that a web server is being developed for the Dibs LightLinks System to interact with in order to send and receive information, may it be in the form of alerts, notifications, instructions, and much more. In order to make a web server an internet provider is needed. After some researching it seems that Amazon Web Services offered the best package for the Dibs LightLinks requirements.

15.1. Amazon Web Services

Amazon Web Services is a collection of remote computing services that make up a cloud computing platform offered by Amazon.com. There is an abundance of different services that AWS offers such as cloud computing, cloud storage, database setup and storage and plenty more. The decision to go with Amazon Web Services was brought upon by the fact that the services were both inexpensive and user friendly.

This was not used in the end because Wifi was exchanged for Bluetooth.

15.1.1. Cloud Services

There are multiple cloud services that Amazon Web Services provides. The main cloud service that the Dibs LightLinks System uses is the Amazon Elastic Cloud Compute service, or EC2 for short. EC2 allows users to develop instances of a web server on Amazon's cloud. Another useful Amazon Web Service that the Dibs LightLinks System uses is the Amazon Simple Storage Service, or S3 for short. S3 gives its users access to cloud storage space to be used to store user information and web application files.

15.1.2. IoT Services

Amazon Web Services has a built in Internet of Things (IoT) service that supports connecting different devices to the Amazon cloud. This service is used to facilitate the communication between the Dibs LightLinks system's Wifi module and the web server. This service will also play an important role in the interaction between the server and the mobile application available for Android.

15.1.3. Email Services

Amazon Web Services has email services that allow the server to send outgoing emails. This will allow the server to send the Dibs LightLinks System different email notifications when necessary. These emails could also be used for alerts while the Dibs LightLinks System is in security mode.

15.1.4. Personalized Accounts

Another useful feature of the Amazon Web Services is the multiple database services it offers. The main Amazon Web Services database service that the Dibs LightLinks systems uses is the Amazon SimpleDB service. This service provides simple database functionality to be used for storage management. The databases will be used with the web applications to store and manage the Personalized Accounts associated with the Dibs LightLinks system.

15.1.5. Interaction with Android Technology

The Dibs LightLinks System is meant to be accessible and modifiable through web pages on a computer as well through mobile applications on Android mobile devices. Both the PC web applications and the Android Mobile applications will be using wireless communications in order to interact with the Dibs LightLinks server.

15.1.5.1. Server to Android

Amazon Web Services offers a service that allows mobile applications to be tested on a server via the Amazon Mobile Hub service that allows the building, testing, and monitoring of mobile applications. This service will allow for simplified integration with the mobile application on the Amazon Web Server. This feature was ultimately not used.

15.1.5.2. Android to Server

Mobile technology can be connected to the server via the IoT services aforementioned. This will allow for mobile devices to send data to the Amazon Web Server more fluently. Using an Android mobile device the user will also be able to modify the Dibs LightLinks System at will. This means that at any time the user could switch their Dibs LightLinks System to Security Mode to help keep their home safer, Party Mode to liven up a scene at a moment's notice, or even Sleep Mode when it's time to turn in for the night and head to sleep. This feature did not get featured in the final project.

15.2. Building the Dibs LightLinks Server

The Dibs LightLinks System has a multitude of features that are attainable through the different types of communication signals to and from the Interface Link in the system. One of the types of communication used in the Dibs LightLinks System is wireless communication. In order for the Interface Link to have Wifi functionality an Addicore ESP8266 ESP-01 Wireless Transceiver module is being connected to the Dibs LightLinks's microcontroller. Once the Dibs LightLinks System has been equipped with Wifi capabilities the next part would be what the Wifi module actually connects to, which in the Dibs LightLinks System is a personalized web server being design with Amazon Web Services. With the tools provided by Amazon Web Services, a web application is going to be developed with the following properties:

- Personalized accounts that the user will use to login and register their Dibs device
- Database functionality that will be used to store each personal account
- A file system for users to upload any sort of files they wish to their account
- Email services to notify users of intrusions while security mode is activated
- Internet of Things capabilities for mobile devices to use for connection to the server

In order to ensure that the web server is implemented correctly, Amazon Web Services provides a very detailed process to follow when a user is building their first web application on their servers.

15.2.1. Preliminary Stages of Web Application Development

Before a new user can begin to develop a new web application, Amazon Web Services instantiated a few recommended preliminary steps to follow beforehand. These steps were put in place to ease the process of developing the web application, as well as making it possible to actually access the services that Amazon Web Services has to offer.

15.2.1.1. Creating an Amazon Web Services Account

The first preliminary step to making a web application with Amazon Web Services is to actually make an account to use Amazon Web Services. In order to use any of the services that Amazon Web Services provides, a user has to make their own personal account to sign into the Amazon Web Services console. This part is fairly straight forward, in whichever web browser navigate to <http://aws.amazon.com/> and click the 'Sign In to the Console' button.

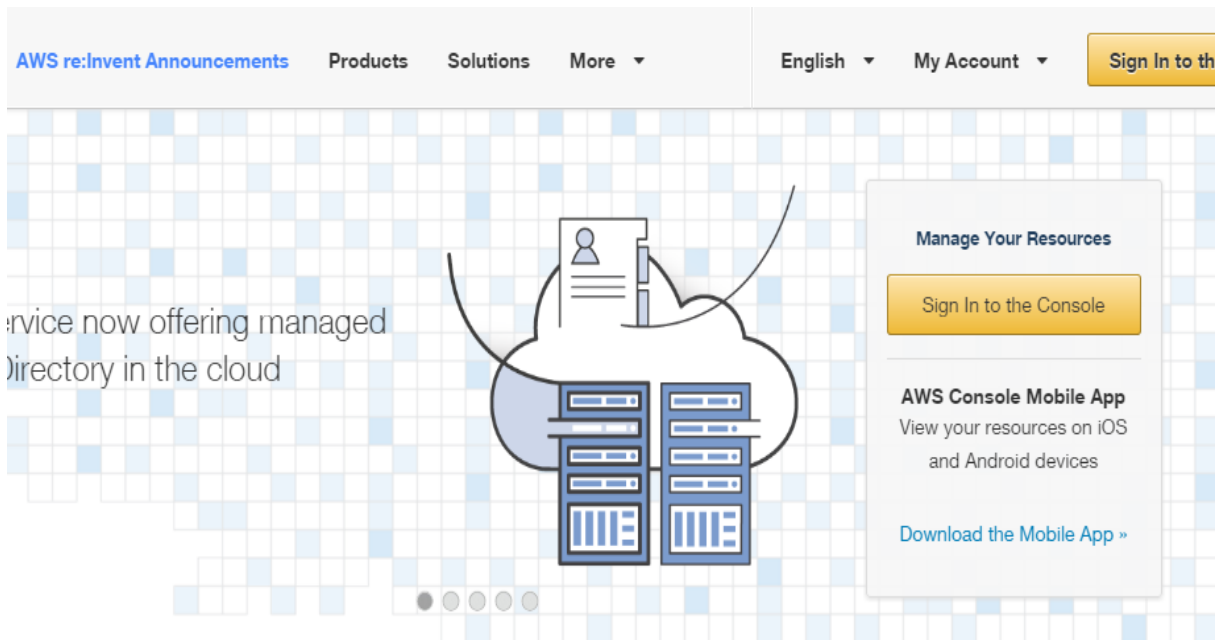


Figure 15.2.1.1.1, AWS Welcome Screen [1], Courtesy of Amazon.com Inc.

Once the 'Sign In the Console' button is clicked, Amazon Web Services will redirect the page to a login in screen. Enter any personal email information and select the 'I am a new user' option. Click the 'Sign in using our secure server' button and follow the instructions that Amazon Web Services provides for the creation of a new account.



Sign In or Create an AWS Account

What is your e-mail or mobile number?

E-mail or mobile number:

- I am a new user.**
- I am a returning user
and my password is:**

[Forgot your password?](#)

Figure 15.2.1.1.2, AWS Login Screen [1], Courtesy of Amazon.com Inc.

15.2.1.2. Creating an IAM user

After an account has been created to gain access to the Amazon Web Services Console, the next preliminary step is to create an IAM user through Amazon Web Services. Services in the Amazon Web Services require that credentials be provided when users want to access them. This method is in place so that the service can determine whether a user has permission to access its resources. Users can create access keys for their Amazon Web Services account to access the command line interface or API. Amazon Web Services recommends the use of the Amazon Web Services Identity and Access Management (IAM) in order to protect their Amazon Web Services resources from unauthorized access. To create an IAM user the first step is to sign in to the AWS Management Console and open the IAM by using the navigation bar to navigate to <https://console.aws.amazon.com/iam/>. Once the IAM console is open, look in the Navigation Pane on the left side of the page. Click 'Groups' then click the 'Create New Group' button in the upper left corner.

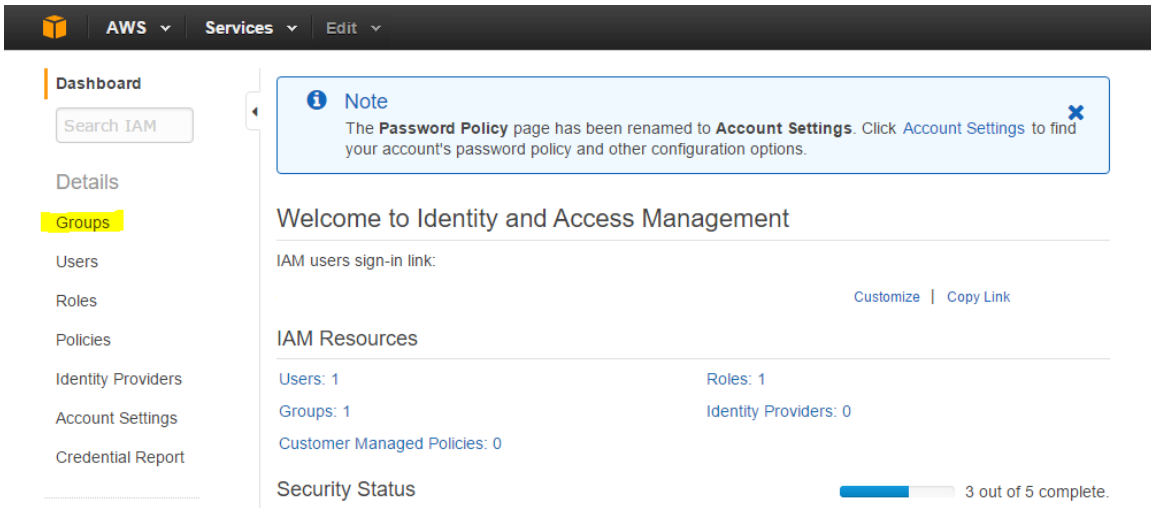


Figure 15.2.1.2.1, AWS IAM Dashboard: Groups Highlighted in Yellow on the Left [1], Courtesy of Amazon.com Inc.

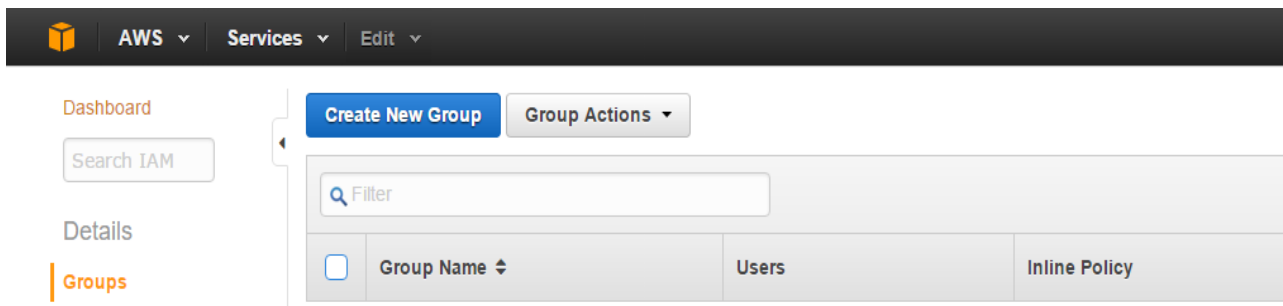


Figure 15.2.1.2.2, AWS IAM Groups Section [1], Courtesy of Amazon.com Inc.

After the 'Create New Group' button has been clicked, type a 'Group Name' to use for the group being created, then click the 'Next Step' Button in the lower right corner.

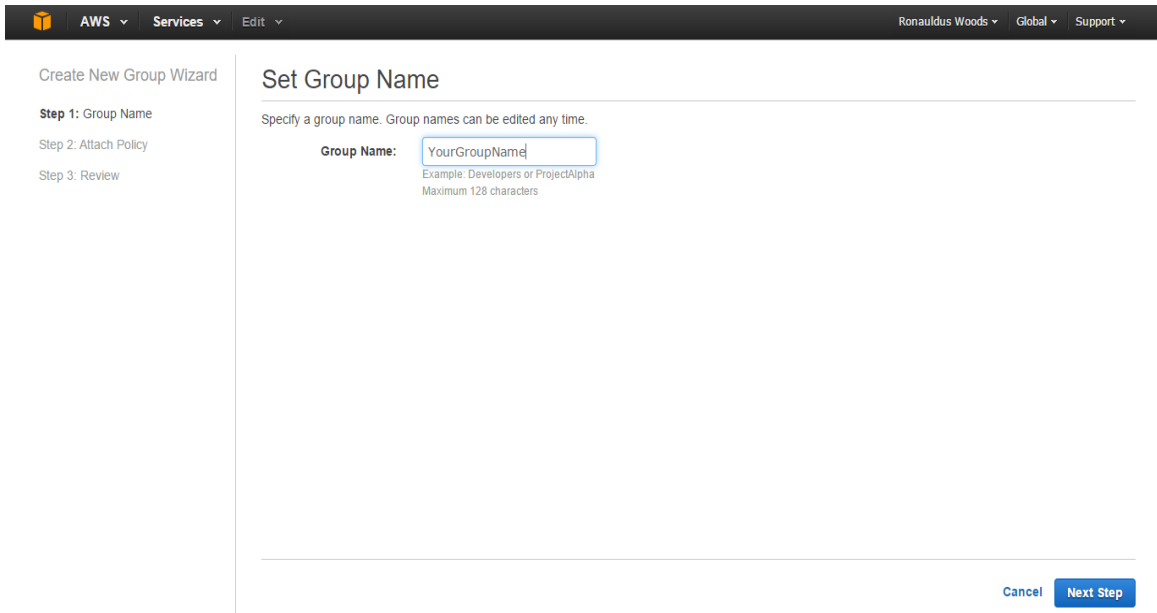


Figure 15.2.1.2.3, AWS IAM Set Group Name section [1], Courtesy of Amazon.com Inc.

After the group name has been selected, a policy has to be added to the group. In the list of policies, check the box next to the 'AdministratorAccess' policy then click the 'Next Step' button in the lower right corner.

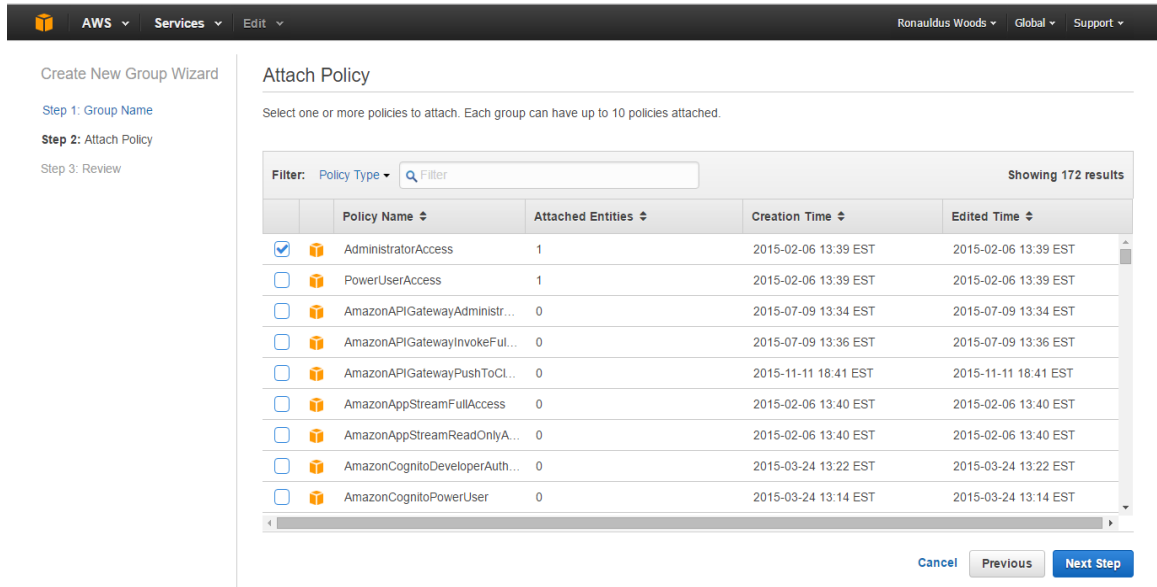


Figure 15.2.1.2.4, AWS IAM Attach Policy section [1], Courtesy of Amazon.com Inc.

After the policy has been attached to the group, the next step of creating the group is to confirm that everything is correct. Review all the information is correct for the group, then click the 'Create Group' button in the lower left corner.

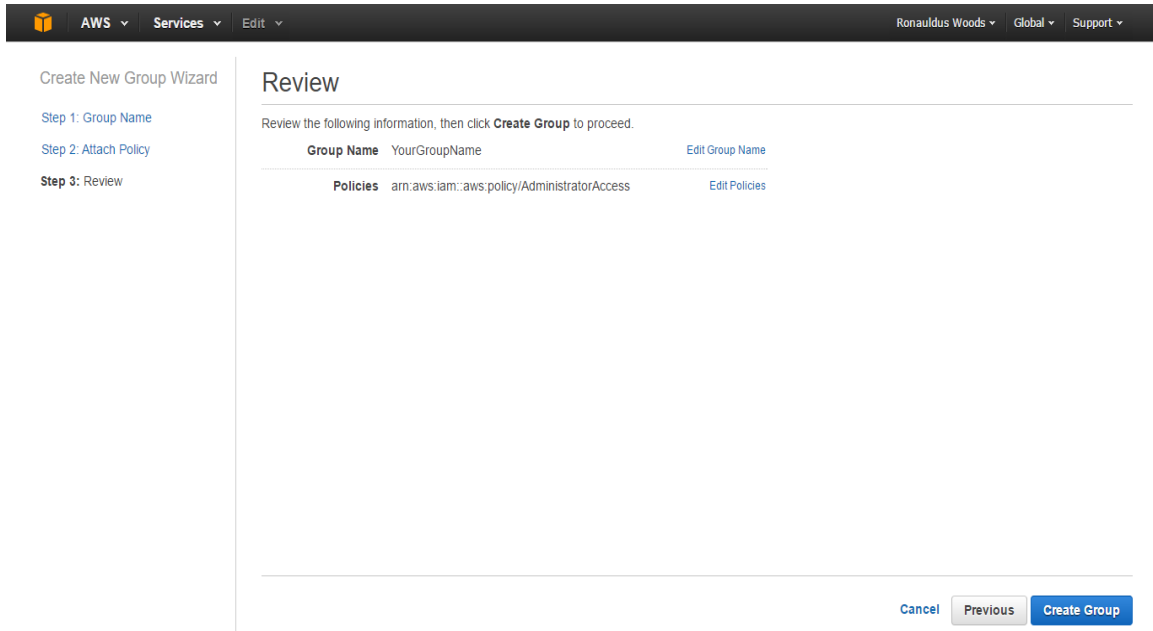
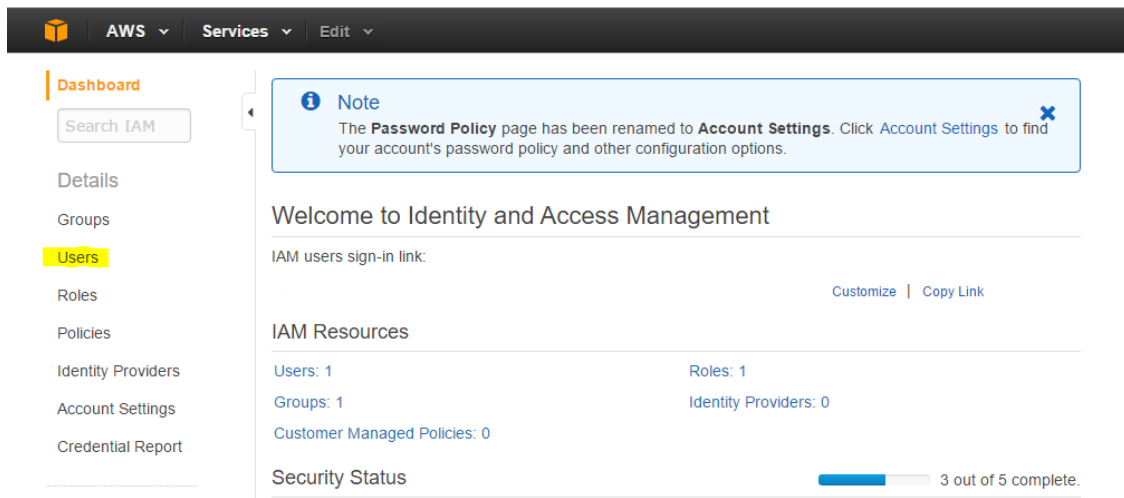


Figure 15.2.1.2.5, AWS IAM Group Review section [1], Courtesy of Amazon.com Inc.

Now that an IAM group has been created, IAM users need to be created and added to the group. Click 'Users' in the Navigation Pane, then click 'Create New Users' in the the upper left corner



15.2.1.2.6, AWS IAM Dashboard: Users Highlighted in Yellow on the Left [1], Courtesy of Amazon.com Inc.



15.2.1.2.7, AWS IAM Users section [1], Courtesy of Amazon.com Inc.

Once 'Create New Users' has been clicked, select the user names that will be used and uncheck the box next to 'Generate an access key for each user' then click the 'Create' button.

Create User

Enter User Names:

1. User0
2. User1
3. User2
4. User3
5. User4

Maximum 64 characters each

Generate an access key for each user

Users need access keys to make secure REST or Query protocol requests to AWS service APIs.
For users who need access to the AWS Management Console, create a password in the Users panel after completing this wizard.

Cancel Create

Figure 15.2.1.2.8, AWS IAM User Name section [1], Courtesy of Amazon.com Inc.

After the users have been created, click on one of the users from the list. Once the page opens, click the 'Groups' tab if it isn't already selected, then in the Groups tab, click the 'Add User to Groups' button. After the 'Add User to Groups' button is clicked, check the box of the group that the user will go into, then click the 'Add to Groups' button in the bottom right corner.

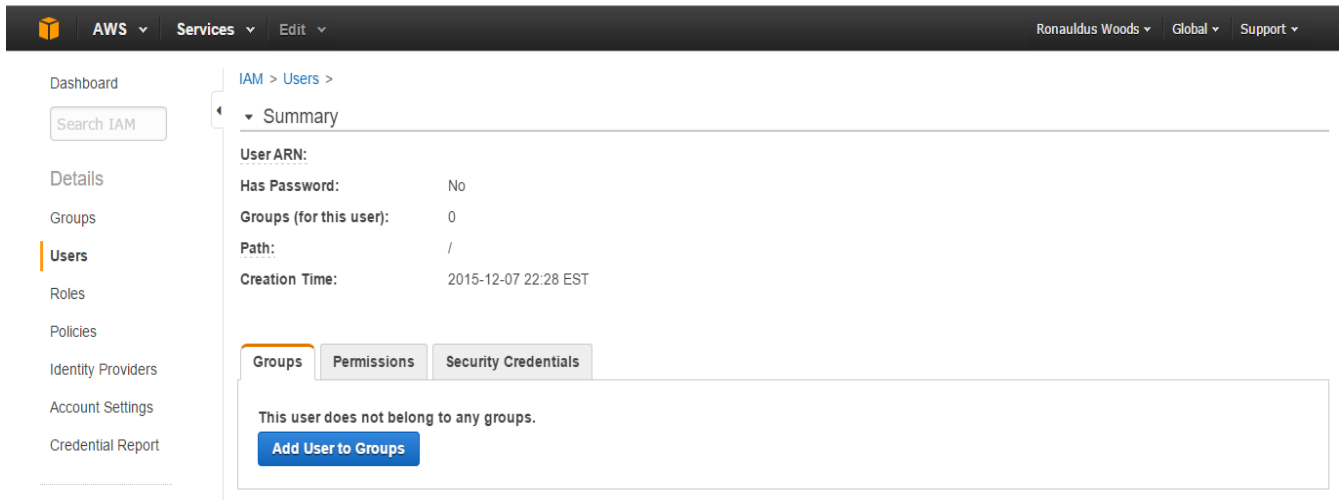


Figure 15.2.1.2.9, AWS IAM Specific User section, Groups Tab [1], Courtesy of Amazon.com Inc.

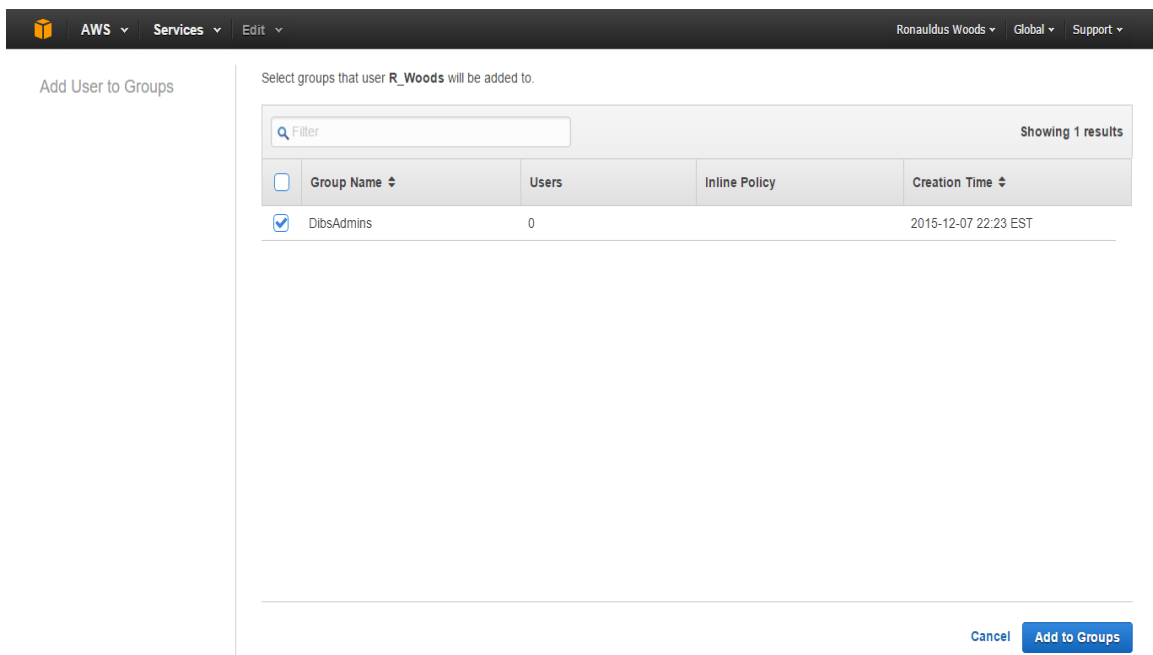


Figure 15.2.1.2.10, AWS IAM Add User to Groups section [1], Courtesy of Amazon.com Inc.

After the user has been added to the group, click on the 'Security Credentials' tab. In the 'Sign-In Credentials' section then click on the 'Manage Password' button.

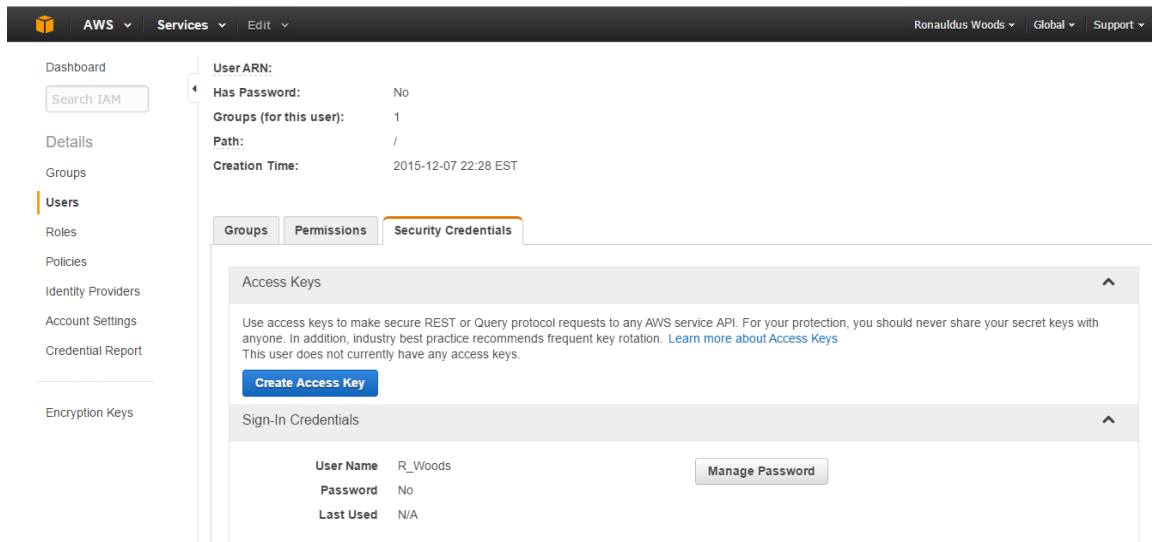


Figure 15.2.1.2.11, AWS IAM User Section, Security Credentials Tab [1], Courtesy of Amazon.com Inc.

On the new page select the option to 'Assign a custom password' and type the password inside the 'Password' and 'Confirm Password' boxes. Leave the checkbox next to 'Require user to create a new password at next sign-in' unchecked if the user is meant to keep the same password when they login. After the password has been entered into both text boxes click the 'Apply' button in the bottom right corner.

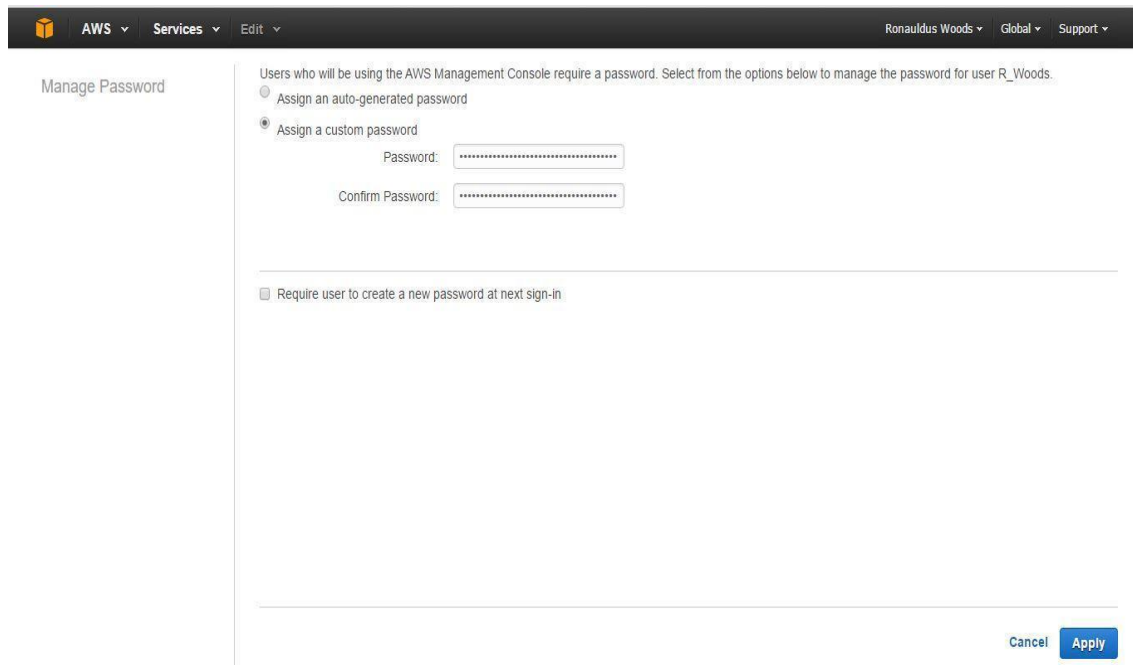


Figure 15.2.1.2.12, AWS IAM Manage Password section [1], Courtesy of Amazon.com Inc.

15.2.1.3. Creating a Key Pair

The next preliminary step in creating a web application using Amazon Web Services is to create a Key Pair for added security. Amazon Web Services uses public-key cryptography to secure the login information for the EC2 instances on the cloud. The user specifies the name of the key pair when they launch their instance, then the user provides the private key to obtain the administrator password for their Windows instance so they can log in using RDP. To get started with creating a Key Pair, navigate to <https://console.aws.amazon.com/ec2/> to open the EC2 console. When the page opens, check the 'region selector' in the top right corner and make sure it is on 'US West (Oregon)'. Once the region is correct, click on 'Key Pairs' in the navigation pane on the left of the screen.

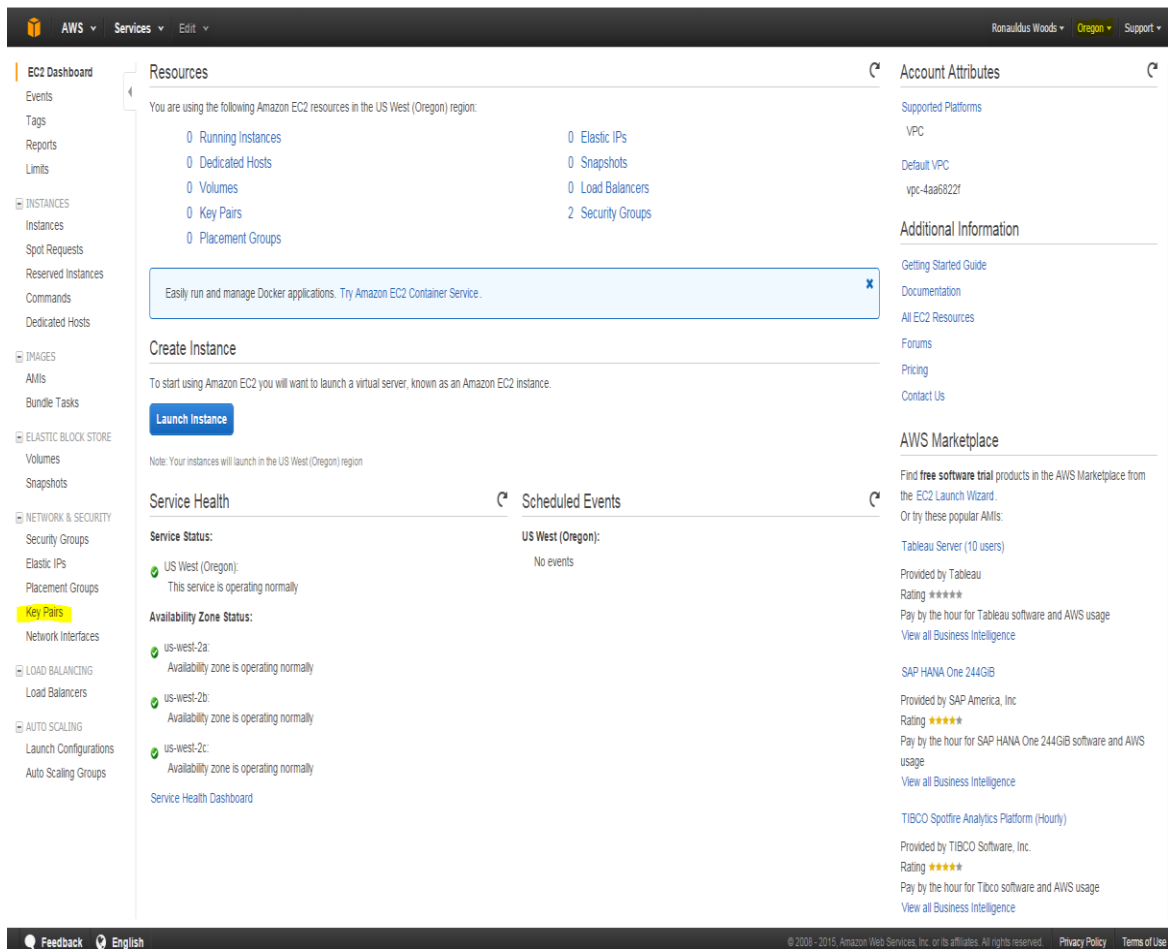


Figure 15.2.1.3.1, AWS EC2 Dashboard, Key Pairs and Oregon, Key Pairs Highlighted in Yellow to the Left, Oregon Highlighted in Yellow in the Upper Right Corner [1], Courtesy of Amazon.com

After clicking on 'Key Pairs' click on the 'Create Key Pair' button. After clicking the 'Create Key Pair' button, enter a name for the Key Pair in the text box and click the 'Create' button. When the 'Create' button is clicked a file is downloaded by the web browser automatically. The .pem file should be saved somewhere safe as that moment would be the only moment to save the .pem file.

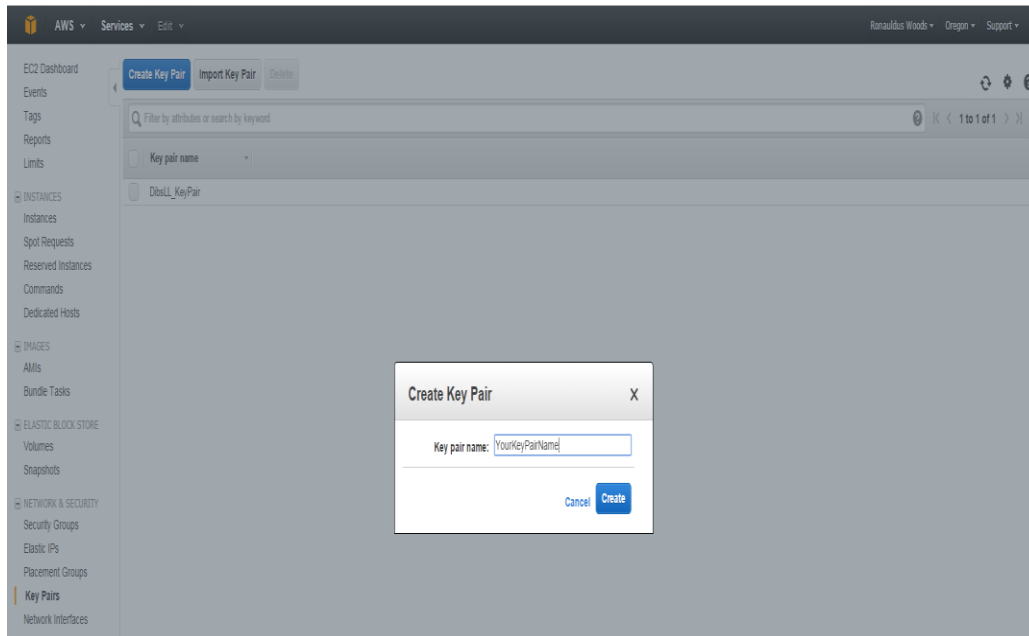


Figure 15.2.1.3.2, Key Pairs Section [1], Courtesy of Amazon.com

For the next step differs depending on the OS of the computer. For Dibs LightLinks System the web application is developed using a Microsoft Windows Operating System. For this reason, the next step requires that PuTTY be downloaded to proceed. If PuTTY is not already on the system, navigate to <http://www.chiark.greenend.org.uk/~sgtatham/putty/> for download and installation instructions. After PuTTY has been installed run the PuTTYgen.exe file. When PuTTYgen opens, look for the section labeled 'Type of key to generate' in the 'Parameters' box and select the SSH-2 RSA section. Then click the 'Load' button.

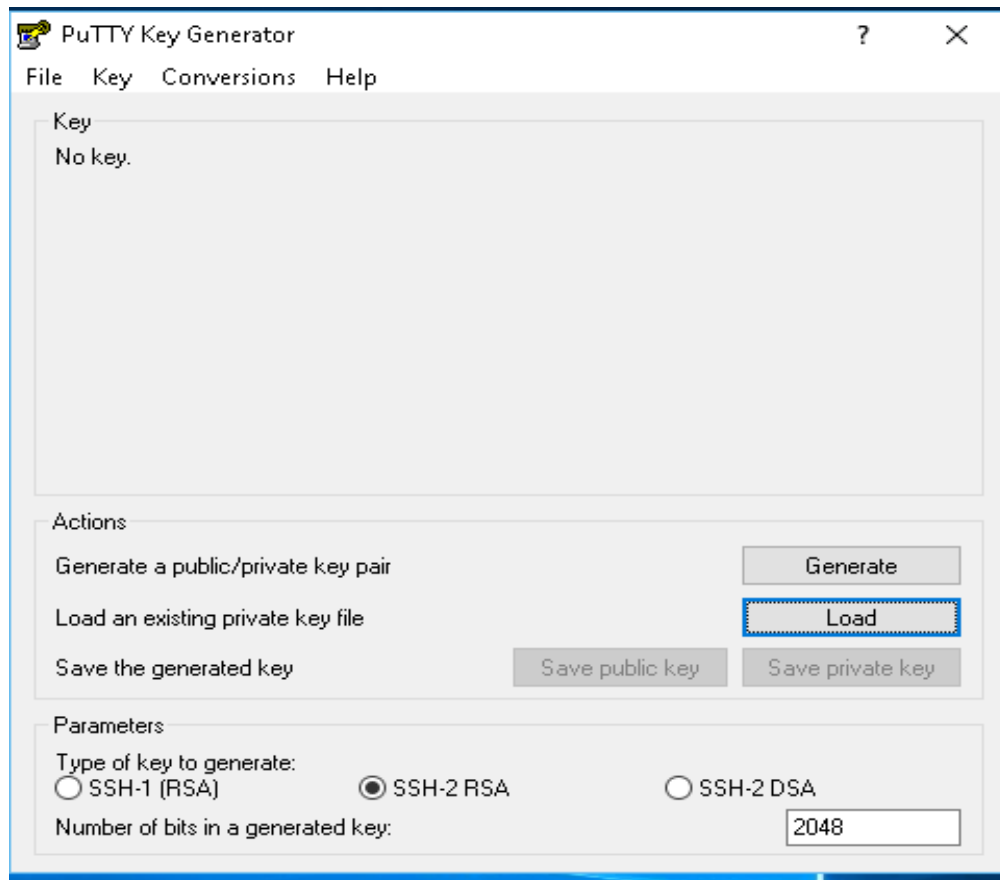


Figure 15.2.1.3.3, PuTTYgen [2], reprinted in accordance with the PuTTY license

Navigate through the file directory to locate where the .pem file is save. Select the private key file and click 'Open' and 'OK' afterwards when the dialog box appears. Back on the PuTTYgen application. Click the 'Save private key' button. When the warning about a passphrase pops up click 'Yes.' Save the .ppk file somewhere safe in the file directory and then the Key Pair creation is done. After creating a key pair there is only one more final preliminary step before beginning to develop the web application through the Amazon Web Application Services.

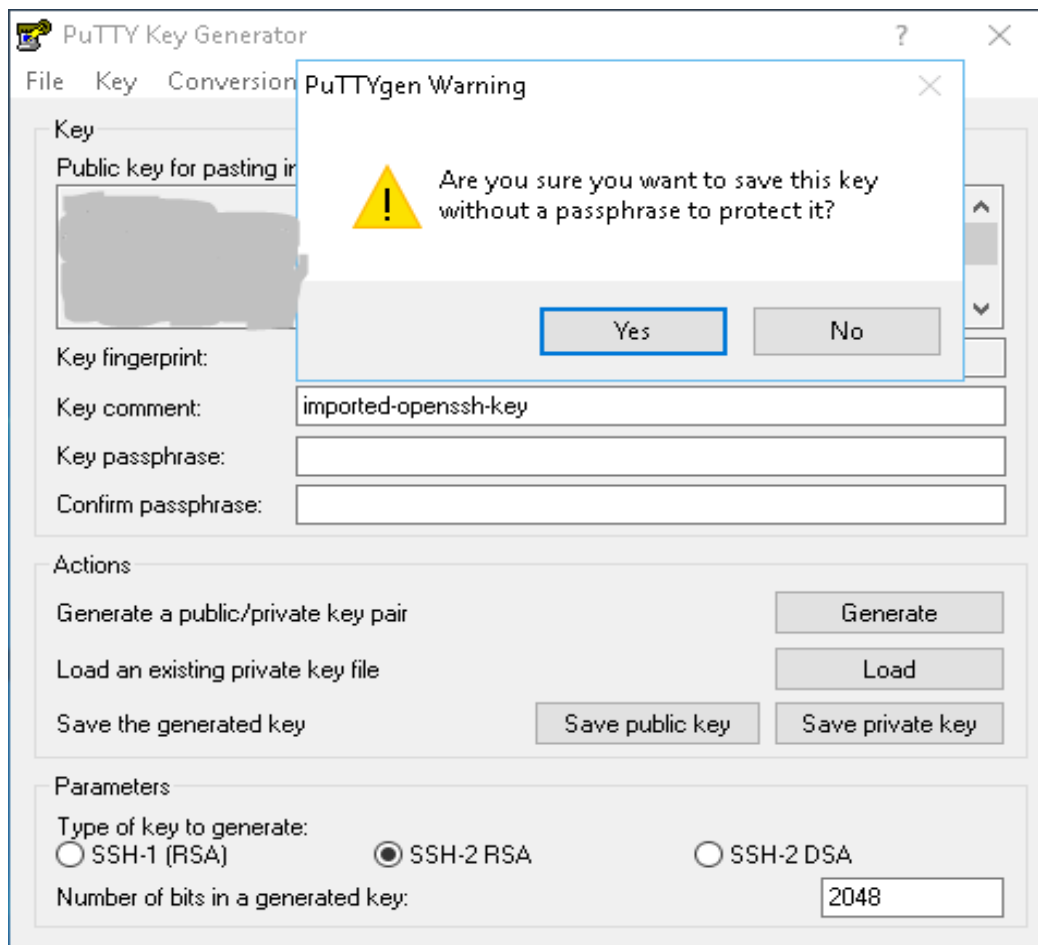


Figure 15.2.1.3.4, PuTTYgen after Load and Save private key [2], reprinted in accordance with the PuTTY license

15.2.1.4. Configure a Virtual Private Cloud

The final preliminary step to building a web application with Amazon Web Services is the step that consists of configuring a Virtual Private Cloud (VPC). To start configuring a VPC navigate to <https://console.aws.amazon.com/vpc/>. When the page opens, ensure that in the top right corner the 'US West (Oregon)' region is the selected region. After the correct region is selected click on 'Your VPCs' in the navigation on the left side.

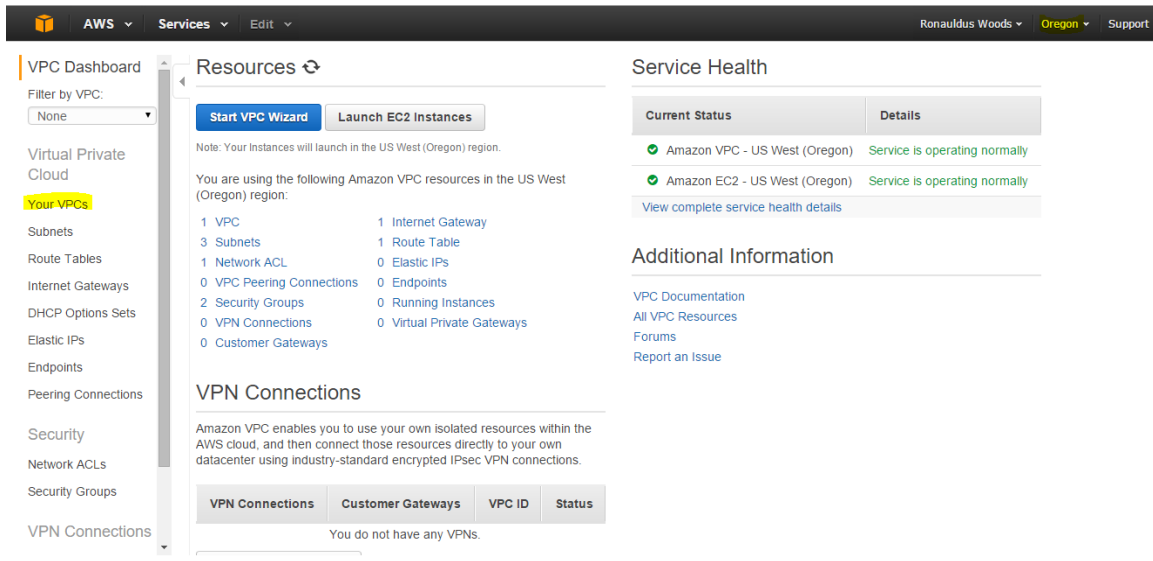


Figure 15.2.1.4.1, AWS VPC Dashboard: Your VPCs Highlighted on the Left [1], Courtesy of Amazon.com Inc.

Here there should be a default VPC available. The user can choose to use this VPC to set up a web application or create a new VPC. If the user wishes to use the default VPC they can and skip the rest of the steps involving using the 'VPC Wizard.' For the Dibs LightLinks System however, a new VPC is going to be created. Navigate back to the VPC Dashboard and click on 'Start VPC Wizard' in the upper left corner as denoted in Figure 15.2.1.4.1. When the VPC Wizard opens, take care to make sure that 'VPC with a Single Public Subnet' is selected then click the 'Select' button.

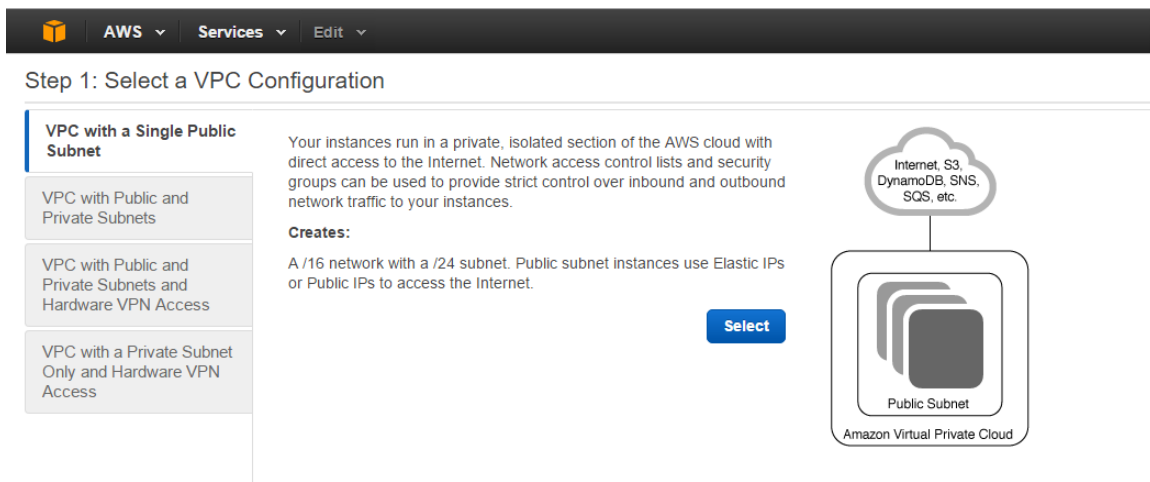


Figure 15.2.1.4.2, AWS VPC Wizard [1], Courtesy of Amazon.com Inc.

When the next page opens, enter a personal name into the 'VPC name' text box. In the 'Subnet name' box, change the name from 'Public subnet' to 'Public subnet 1.' Leave the rest of the default settings the same then click the 'Create VPC button in the lower right corner of the page then 'OK' button on the 'VPC Successfully Created' page on the right side of the screen.

The screenshot shows the 'Step 2: VPC with a Single Public Subnet' configuration page in the AWS Management Console. The page includes the following fields and options:

- IP CIDR block:** 10.0.0.0/16 (65531 IP addresses available)
- VPC name:** YourVPCName
- Public subnet:** 10.0.0.0/24 (251 IP addresses available)
- Availability Zone:** No Preference
- Subnet name:** Public subnet 1
- Add endpoints for S3 to your subnets:** Subnet: None
- Enable DNS hostnames:** Yes (selected)
- Hardware tenancy:** Default

At the bottom right, there are three buttons: 'Cancel and Exit', 'Back', and 'Create VPC'.

Figure 15.2.1.4.3, AWS VPC Wizard Step 2 [1], Courtesy of Amazon.com Inc.

The screenshot shows the 'VPC Successfully Created' confirmation page in the AWS Management Console. The page includes the following elements:

- Header:** VPC Successfully Created
- Message:** Your VPC has been successfully created. You can launch instances into the subnets of your VPC. For more information, see [Launching an Instance into Your Subnet](#).
- Buttons:** OK
- Navigation Pane (Left):** VPC Dashboard, Filter by VPC: None, Virtual Private Cloud

Figure 15.2.1.4.4, AWS VPC Confirmation Page [1], Courtesy of Amazon.com Inc.

In the Navigation Pane on the left side of the screen, click on the 'Route Tables' option. When the page opens, select the Route Table that has a 'Yes' in the Main column. Click on the 'Name' of the Route Table, type 'Main' and press enter. On the other Route Table click on Name again, type 'Custom' and press enter.

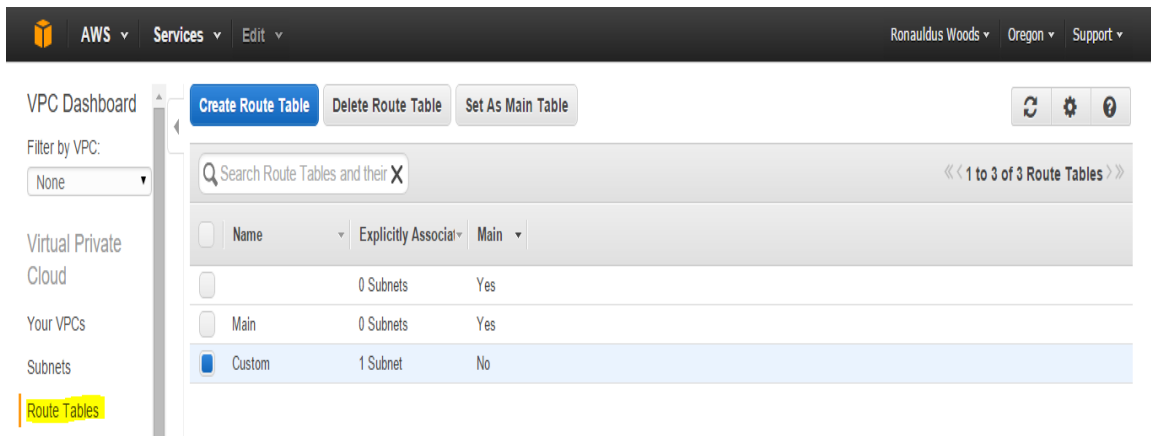


Figure 15.2.1.4.5, AWS VPC Route Tables (Highlighted on the Left) section [1], Courtesy of Amazon.com Inc.

After the Route Tables have been named a subnet needs to be created for the VPC. Click on 'Subnets' in the Navigation Pane on the left side of the screen. When the page opens click the 'Create Subnet' button in the upper left corner of the screen.



Figure 15.2.1.4.6, AWS VPC Subnets(Hightlighted on the Left) section [1], Courtesy of Amazon.com Inc.

When the new window pops up, enter the name 'Public subnet 2' in the 'Name tag' box. Select the VPC that was created earlier from the 'VPC' drop down menu. In the 'Availability Zone' drop down menu, select the second Availability Zone from the list. In the 'CIDR block' enter '10.0.1.0/24' then click the 'Yes, Create' button.

Create Subnet ✕

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag ⓘ

VPC ⓘ

Availability Zone ⓘ

CIDR block ⓘ

Figure 15.2.1.4.7, AWS VPC Subnet section: Create Subnet [1], Courtesy of Amazon.com Inc.

From the list of subnets select the subnet named 'Public subnet 2' then click on the 'Route Table' tab near the bottom of the screen. With the 'Route Table' tab selected click on the 'Edit' button just below the 'Summary' tab.

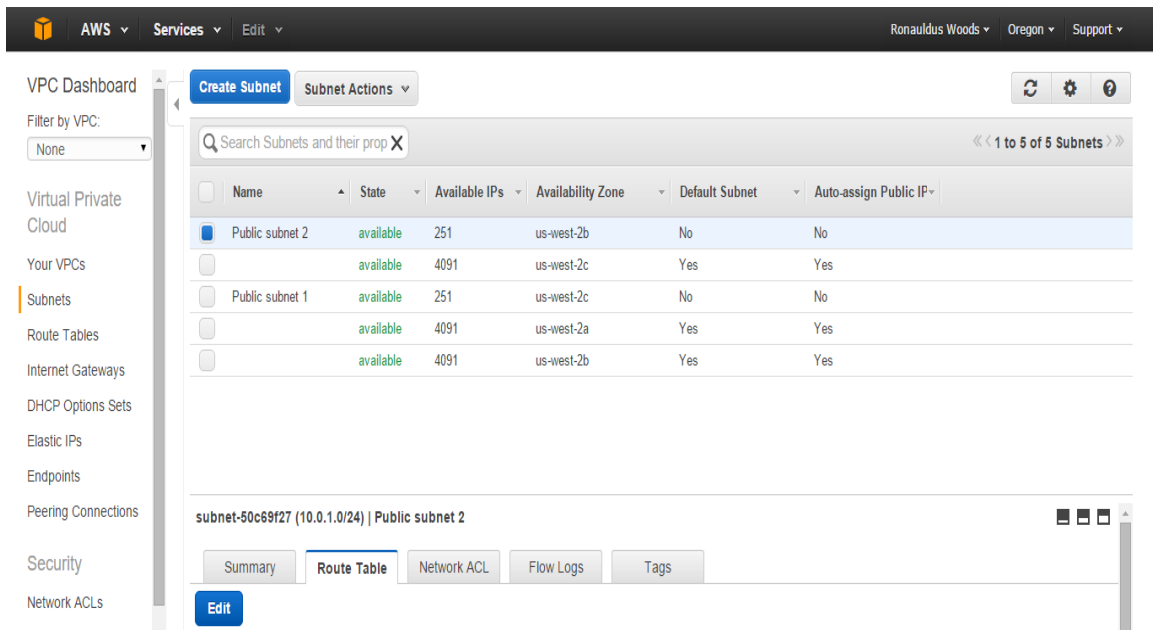


Figure 15.2.1.4.8, AWS VPC Subnets section: Public Subnet 2 with Route Table Tab selected [1], Courtesy of Amazon.com Inc.

Select the Route Table named 'Custom' from the 'Change to' drop down menu then click 'Save' button just below the 'Summary' and 'Route Table' tabs.

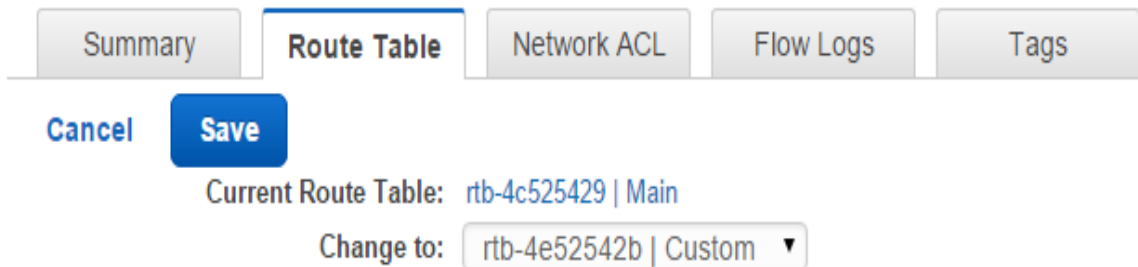


Figure 15.2.1.4.9, AWS VPC Subnets section: Editing the Route Table [1], Courtesy of Amazon.com Inc.

Now that the public subnets have been created the next step is to create private subnets. Click the 'Create Subnet' button again to create another subnet. This time in the 'Name tag' box enter 'Private subnet 1' as the name. Select the first 'Availability Zone' from the list of options. In the 'CIDR block' box enter '10.0.2.0/24' then click the 'Yes, Create' button. Click the 'Create Subnet' button once more to create another

subnet. This time in the 'Name tag' box enter 'Private subnet 2' as the name. Select the second 'Availability Zone' from the list of options. In the 'CIDR block' box enter '10.0.3.0/24' then click the 'Yes, Create' button.

Create Subnet ✕

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag ⓘ

VPC ⓘ

Availability Zone ⓘ

CIDR block ⓘ

Figure 15.2.1.4.10, AWS VPC Subnet section: Create Private subnet 1 [1], Courtesy of Amazon.com Inc.

Figure 15.2.1.4.11, AWS VPC Subnet section: Create Private subnet 2 [1], Courtesy of Amazon.com Inc.

Now that all the preliminary steps to creating the web application have been completed, the actual development of the web application can begin.

15.2.2. Creating a Bucket for the Web Application

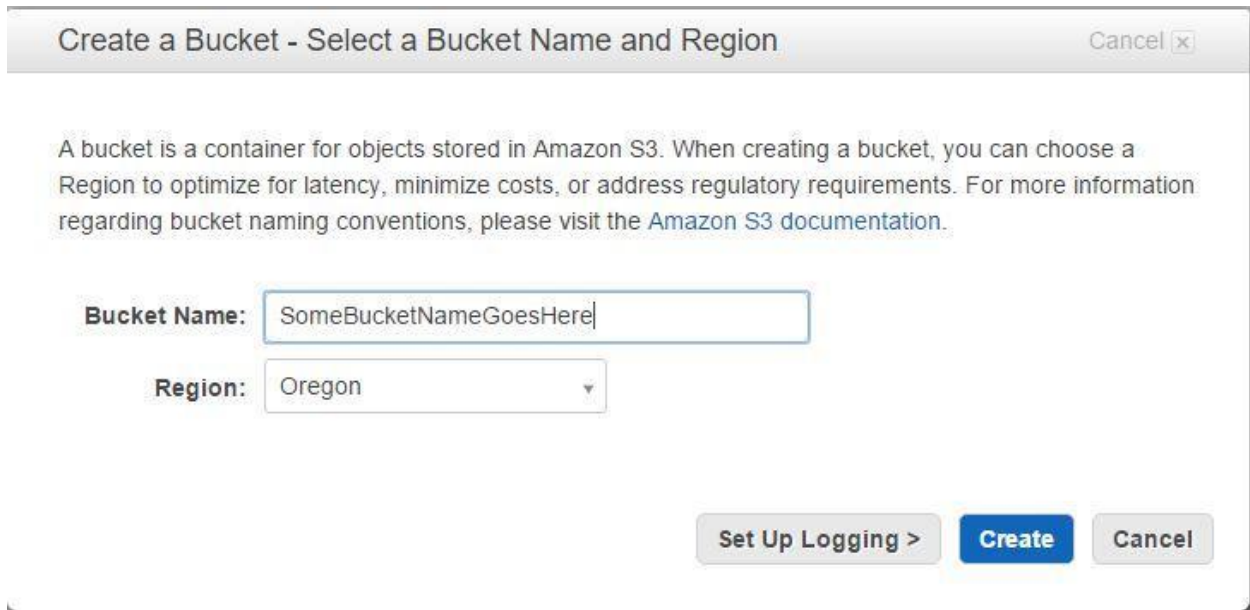
The first step is to create something that Amazon Web Services denotes as a 'Bucket'. Buckets are used to store the files that are going to be used by the web application. In order to start the creation of a Bucket users need to navigate to the Amazon S3 console by going to <https://console.aws.amazon.com/s3/>. Once the console is open, click on the 'Create Bucket' button in the top left corner.



Figure 15.2.2.1, AWS S3 console [1], Courtesy of Amazon.com Inc.

When a Bucket is created it is recommended that the name follow the standard DNS naming convention. Once the name is selected the region of where the bucket is stored in the cloud should be picked. The closest region to the location that will be used the

most by the Dibs LightLinks System is the Oregon region. Once everything has been selected click the 'Create' button in the lower right corner.



Create a Bucket - Select a Bucket Name and Region Cancel

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the [Amazon S3 documentation](#).

Bucket Name:

Region:

Set Up Logging > Create Cancel

Figure 15.2.2.2, AWS S3 Bucket Creation [1], Courtesy of Amazon.com Inc.

15.2.3. Creating an Application Server

After the Bucket has been created the next step is to use the Amazon EC2 service to create a virtual server known as an EC2 instance to run the web application. To start developing these virtual servers open the EC2 console at <https://console.aws.amazon.com/ec2/>. After the EC2 dashboard opens, the next task is to create a Security Group for the Amazon EC2 Instance. In the upper right corner of the screen make sure that the 'US West (Oregon)' region is selected from the list of options. Look to the left hand side on the navigation pane and scroll down until 'Network and Security' appears and click on the 'Security Groups' option.

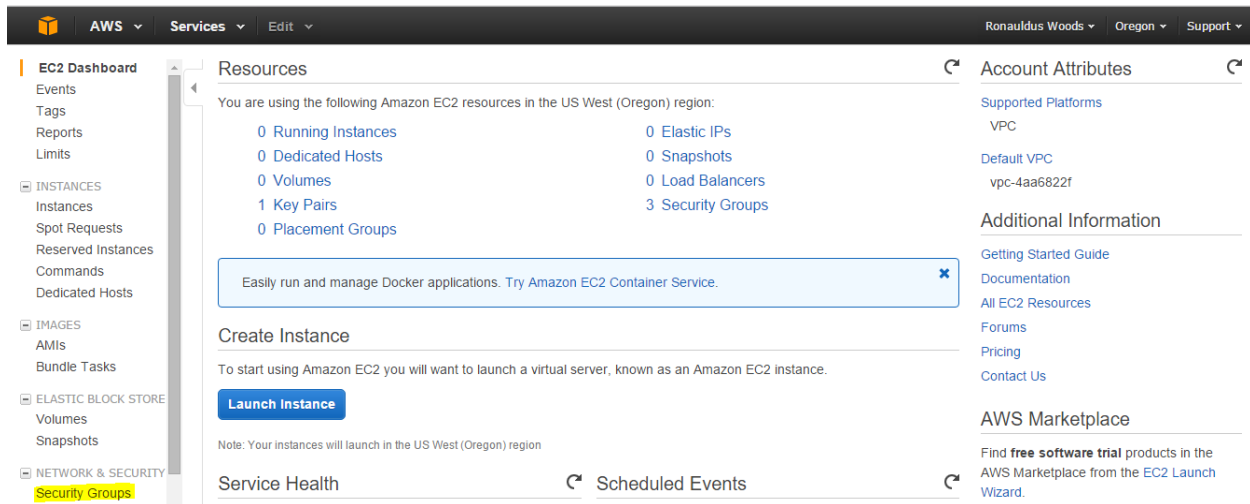


Figure 15.2.3.2, AWS EC2 Dashboard, Security Groups Highlighted on the Left [1], Courtesy of Amazon.com Inc.

In the 'Security Groups' section, click the 'Create Security Groups' button in the upper left corner to start with the security group creation. In the new window that pops up, enter some name for the security group, for testing purposes the name was set as 'WebServerSG' for the Dibs LightLinks System. Enter a description for the new security group and choose the VPC that was created earlier in **Section 15.2.1.4** for the the security group. Along with the security group being created, rules for how the server is accessed have to be added to the security group. Click on the 'Inbound' tab then click the 'Add Rule' button to add rules. Select 'RDP' from the 'Type' list. Then in the 'Source' section, select 'Custom IP' and choose an IP address range. 0.0.0.0/0 can be used as an IP address range but this range is very unsafe and allow outside users to access the server via RDP. A better option would be to use a personal IP address. Once an IP address range has been selected click the 'Add Rule' button once more and select the 'HTTP' type from the type list, then click the 'Create' button.

Create Security Group [X]

Security group name ⓘ WebServerSG

Description ⓘ Dibs LLS Security Group

VPC ⓘ vpc-d17445b4 (10.0.0.0/16) | DibsLL VPC
* denotes default VPC

Security group rules:

Inbound Outbound

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
RDP	TCP	3389	Custom IP An IP Address [X]
HTTP	TCP	80	Anywhere 0.0.0.0/0 [X]

Add Rule

Cancel Create

Figure 15.2.3.3, AWS EC2 Security Groups Creation [1], Courtesy of Amazon.com Inc.

Once an EC2 instance has been created, the following procedure is adding the rules to inbound connections. Make sure the inbound tab for their security group is selected in the mid-left of the dialog box. Click 'Add Rule' and then select 'RDP' from the 'Type' list. Under 'Source,' select 'Custom IP' and enter the public IP address range that is desired, which for testing is could be a personal computer's IP addresses. Click on 'Add Rule' again and under 'Type' select 'HTTP.'

After adding the rules for the inbound connections, the next step is to create an 'IAM Role' to manage Amazon Web Services credentials for software running on the EC2 instance. Open the IAM console (feel free to follow the steps in **Section 15.2.1.2** to find out how if need be) and click on 'Roles' navigation pane. After 'Roles' has been clicked, click the 'Create New Role' button.



Figure 15.2.3.4, AWS IAM: Roles Section [1], Courtesy of Amazon.com Inc.

When the page opens, type a 'Role Name' into the respective text box and click the 'Next Step' button in the bottom right corner.

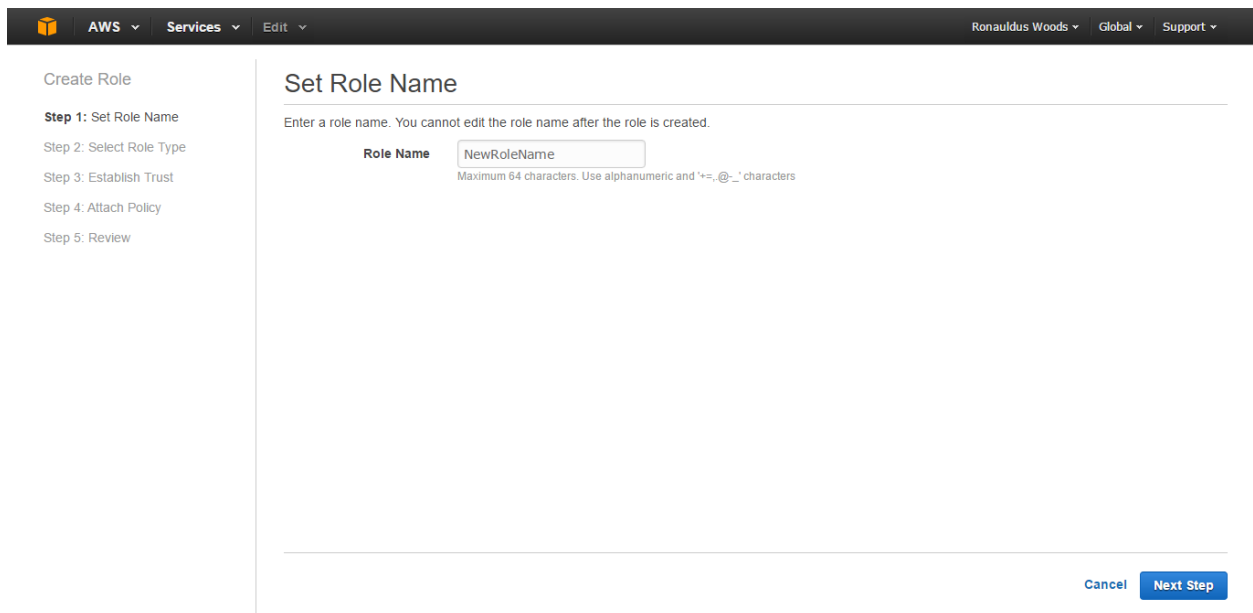


Figure 15.2.3.5, AWS IAM: Roles Section [1], Courtesy of Amazon.com Inc.

On the 'Select Role' page under 'AWS Service Roles' select 'Amazon EC2'.

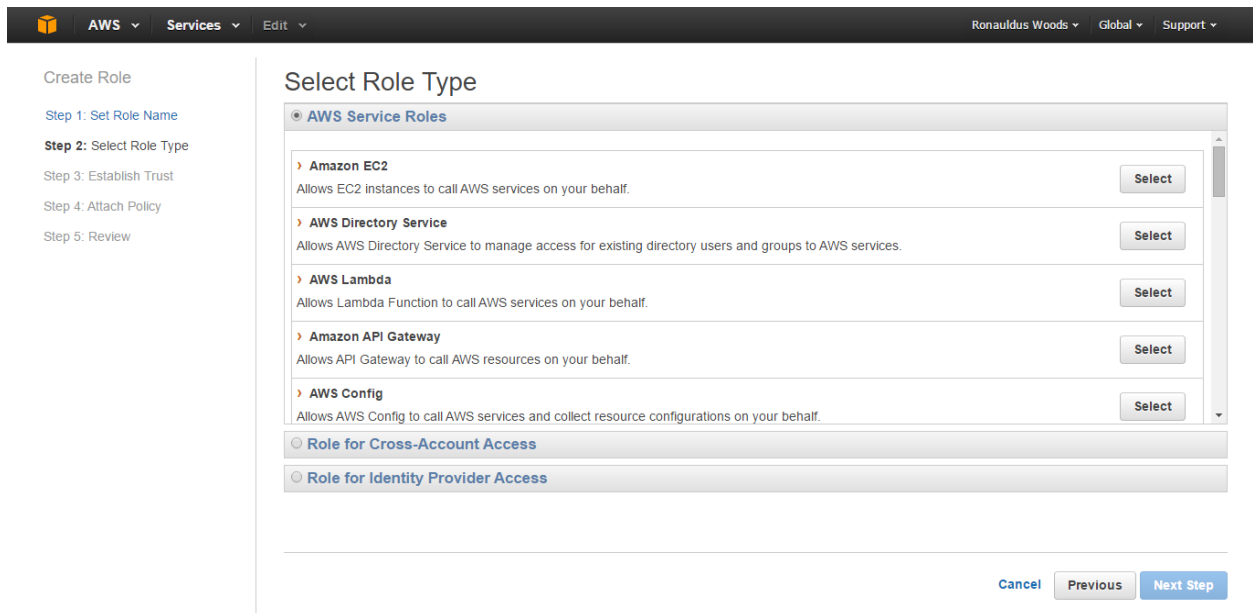


Figure 15.2.3.6, AWS IAM: Select Role Type [1], Courtesy of Amazon.com Inc.

On the 'Attach Policy' page select the 'PowerUserAccess' policy then click the 'Next Step' button.

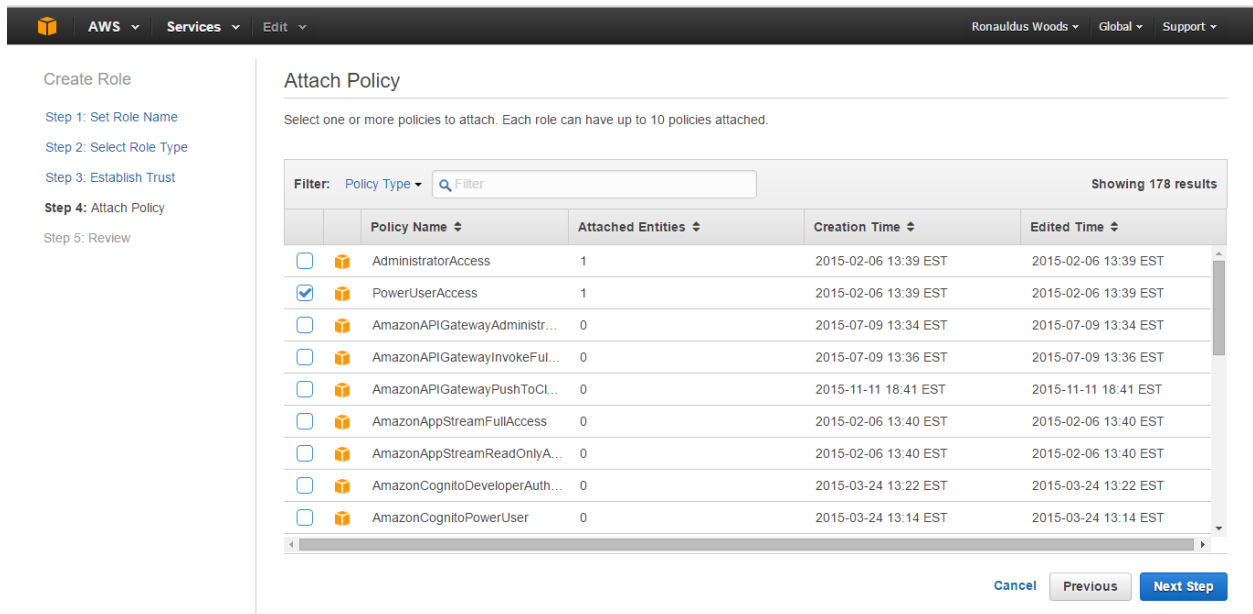


Figure 15.2.3.7, AWS IAM: Select Role Type [1], Courtesy of Amazon.com Inc.

Review the information then click 'Create Role' button in the bottom right corner.

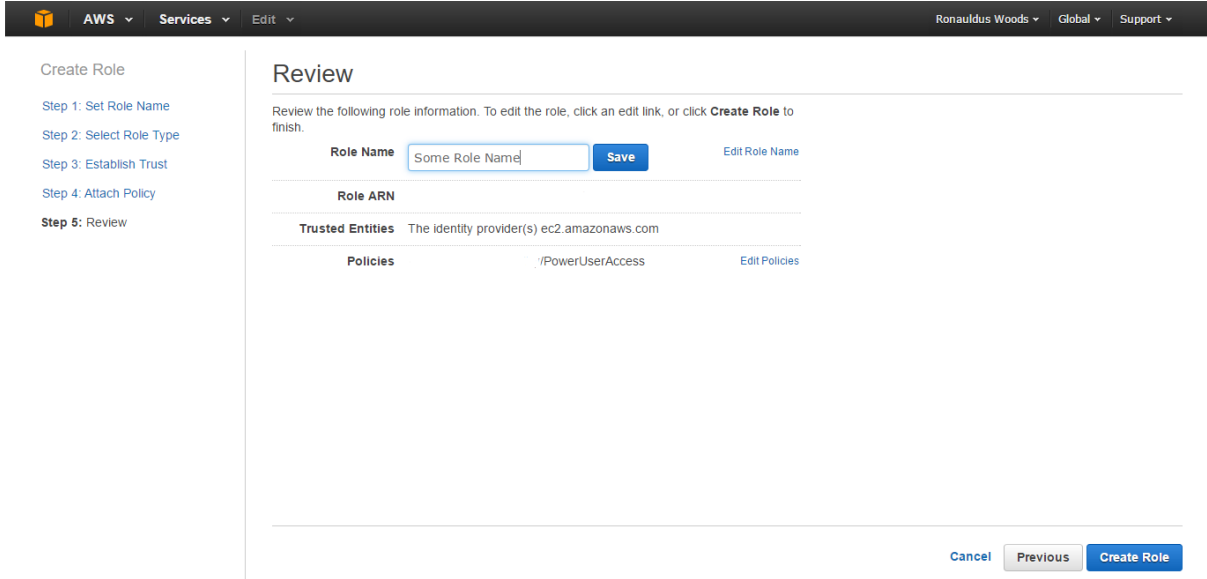


Figure 15.2.3.8, AWS IAM: Role Review [1], Courtesy of Amazon.com Inc.

Now that the IAM role has been created the next step is to launch an EC2 instance. Open the 'Amazon EC2 Console' (refer to earlier in this section for a reference on how to do this if need be) and be sure to verify that the navigation bar has the 'US West (Oregon)' region selected in the top right corner of the page. In the navigation pane on the left hand side of the screen click on 'Instances' then click the 'Launch Instance' button.

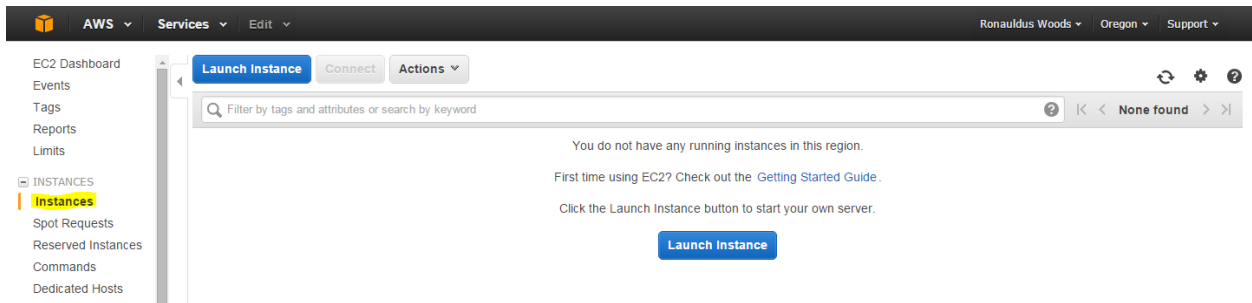


Figure 15.2.3.9, AWS EC2: Launch Instance, Instances Highlighted in Yellow on the Left [1], Courtesy of Amazon.com Inc.

When the 'Choose an Amazon Machine Image' page opens click the 'Free Tier Only' check box to ensure that any extra charges aren't applied in the testing stages of the web server. Since the Dibs LightLinks System is primarily being developed using computers running on Microsoft Windows Operating Systems, the Dibs LightLinks System will use is the Microsoft Windows Server 2012 R2 Base Amazon Machine Image (AMI). Click the 'Select' button for the respective AMI.

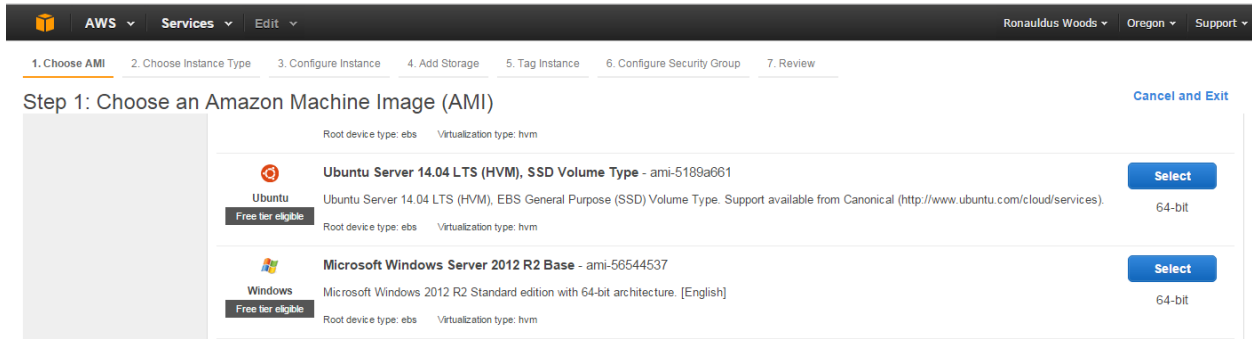


Figure 15.2.3.10, AWS EC2: AMI Selection [1], Courtesy of Amazon.com Inc.

On the 'Choose an Instance Type' page, keep the 't2.micro' instance type selected to remain in the free tier of Amazon Web Services, then click the 'Next: Configure Instance Details' button in the bottom right corner of the screen.

Step 2: Choose an Instance Type
 Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: **All instance types** | **Current generation** | [Show/Hide Columns](#)

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input checked="" type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	m4.large	2	8	EBS only	Yes	Moderate
<input type="checkbox"/>	General purpose	m4.xlarge	4	16	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.4xlarge	16	64	EBS only	Yes	High
<input type="checkbox"/>	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

Figure 15.2.3.11, AWS EC2 Instance Type Selection [1], Courtesy of Amazon.com Inc.

When the page opens, select the VPC that was created in **Section 15.2.1.4** for the 'Network' dropdown menu. Then from the 'Subnet' dropdown menu select on of the public subnets that was created in **Section 15.2.1.4**. Set 'Auto-assign Public IP' field to 'Enable' to ensure that instance can receive a public IP address or a public DNS name. Select the 'IAM role' that was created earlier in this section from the respective drop down menu, then click the 'Review and Launch' button in the bottom right corner of the screen.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances [Launch into Auto Scaling Group](#)

Purchasing option Request Spot instances

Network [Create new VPC](#)

Subnet [Create new subnet](#)

Auto-assign Public IP

Domain join directory [Create new directory](#)

IAM role [Create new IAM role](#)

Shutdown behavior

Enable termination protection Protect against accidental termination

Monitoring Enable CloudWatch detailed monitoring
[Additional charges apply.](#)

Tenancy [Additional charges will apply for dedicated tenancy.](#)

Network interfaces

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

Figure 15.2.3.12, AWS EC2 Instance Detail Configuration [1], Courtesy of Amazon.com Inc.

On the 'Review Launch Instance' page click 'Edit security groups' on the right hand side under the 'Security Groups' section.

▼ Security Groups [Edit security groups](#)

Security group name launch-wizard-1
Description launch-wizard-1 created 2015-12-14T16:08:23.077-05:00

Type	Protocol	Port Range	Source
RDP	TCP	3389	0.0.0.0/0

Figure 15.2.3.13, AWS EC2 Edit Security Group Before Launch [1], Courtesy of Amazon.com Inc.

On the 'Configure Security Group' page, click the 'Select an existing security group' option and select the security group that was created earlier in this section. Click the 'Review and Launch' button in the bottom right corner once more. Back at the 'Review Instance Launch' page click the 'Launch' button in the bottom right corner.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-a44573c0	default	default VPC security group	Copy to new
<input checked="" type="checkbox"/> sg-42d8ea26	WebServerSG	Dibs LLS Security Group	Copy to new

Inbound rules for sg-42d8ea26 (Selected security groups: sg-42d8ea26)

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
RDP	TCP	3389	

[Cancel](#) [Previous](#) [Review and Launch](#)

Figure 15.2.3.13, AWS EC2 Configure Security Group [1], Courtesy of Amazon.com Inc.

Step 7: Review Instance Launch

Review the details of your instance launch before you start. You can change any of the details, such as **AMI** or **Instance Type**, before you launch your instance and complete the launch process.

▼ **AMI Details** [Edit AMI](#)

Microsoft Windows Server 2012 R2 Base - ami-56544537
 Free tier eligible Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]
 Root Device Type: ebs Virtualization type: hvm

▼ **Instance Type** [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

▼ **Security Groups** [Edit security groups](#)

Security Group ID	Name	Description
sg-42d8ea26	WebServerSG	Dibs LLS Security Group

All selected security groups inbound rules

Security Group ID	Type	Protocol	Port Range	Source
sg-42d8ea26	HTTP	TCP	80	0.0.0.0/0
sg-42d8ea26	RDP	TCP	3389	

[Cancel](#) [Previous](#) [Launch](#)

Figure 15.2.3.14, AWS EC2 Instance Pre-Launch [1], Courtesy of Amazon.com Inc.

In the 'Select an existing key pair or create a new key pair' window that pops up, select to 'Choose an existing key pair' and choose the key pair that was created in **Section 15.2.1.3**. Click the acknowledgement then click the 'Launch Instances' button in the bottom right corner of the window.

The screenshot shows a dialog box titled "Select an existing key pair or create a new key pair" with a close button (X) in the top right corner. The dialog contains the following text: "A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance." Below this is a note: "Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#)." There are two dropdown menus: the first is labeled "Choose an existing key pair" and the second is labeled "Select a key pair" with "DibsLL_KeyPair" selected. A checkbox is checked with the text "I acknowledge that I have access to the selected private key file (DibsLL_KeyPair.pem), and that without this file, I won't be able to log into my instance." At the bottom right, there are two buttons: "Cancel" and "Launch Instances".

Figure 15.2.3.15, AWS EC2 Key Pair Selection Before Launch [1], Courtesy of Amazon.com Inc.

At this point the web server creation has been created. Navigate back to 'Amazon EC2 Console' and click 'Instances' in the navigation pane to find out the status of the instance. Once the instance says 'running' it is ready for use. The final overall product utilized Bluetooth rather than Wifi.

15.3. Communication to Microcontroller

Communications were meant to be between the microcontroller and the Amazon Web Server via the Addicore ESP8266 ESP-01 Wireless Transceiver that is connected to the microcontroller of the Dibs LightLinks system. After careful consideration it was concluded that Bluetooth better fit the needs of the system with the HC-05 Bluetooth module.

16. Arduino/Atmega328 Software

The Atmega328 MCU is the MCU used in a common Arduino model, the Arduino Uno, and because of this, this MCU is compatible with the Arduino software suite. This suite is comprised of the Arduino IDE, which is installed on a traditional computer, and the Arduino bootloader, which is installed directly on the MCU.

16.1. Libraries

We will be programming the Atmega328 using the Arduino integrated development environment (IDE). As such, we have certain libraries available to us: the Digital I/O library, the Analog I/O library, and the Wire library.

16.1.1. Digital I/O

The Digital I/O library will be used in the automatic Link ordering process. It has three main functions: `pinMode()`, `digitalWrite()`, and `digitalRead()`.

The first sets a pin's mode, which is to say if the pin is in an input, output, or input pullup state. It's syntax is as follows: `pinMode(pin, mode)`, with `pin` being a legal digital pin, and `mode` being "INPUT," "OUTPUT," or "INPUT_PULLUP." All of this must be defined in the setup loop.

The last two do what you might imagine: they either write or read digitally. The syntax is `digitalWrite(pin, value)` or `digitalRead(pin, value)`, with `pin` being a legal digital pin, and `value` being either "HIGH," which is either 3.3 V or 5 V, or "LOW," which is 0 V.

16.1.2. Analog I/O

The Analog I/O library will be used in analog interfaces, such as in the LED system for output, and possibly in the door dongle input if it turns out in testing that a simpler circuit (one excluding the op-amp) provides reliable data. Like the Digital I/O library, it has

three functions. However, analog pins can handle input or output, and as such are bidirectional, so there is no analog analogue to the `pinMode()` function.

The first function, `analogReference()` allows one to configure the reference voltage used for analog input. This function will likely go unused. However, for completeness, the syntax is `analogReference(type)`, where `type` is "DEFAULT," "INTERNAL," "INTERNAL1V1," "INTERNAL2V256," and "EXTERNAL."

The next two functions, `analogRead()` and `analogWrite()` work similarly to their digital cousins. They have precisely the same syntax, with one alteration. Since `analogWrite()` merely fakes an analog output by using PWM, `analogWrite()`'s value is a duty cycle from 0 to 255, inclusive, which are always off and always on, respectively. That duty cycle is a simple division--that is to say that $\text{value}/255$ is equivalent to the percent cycle on. For example, a value of 174 would give a duty cycle of $174/255$, or about 68.2%. This has the interesting side-effect of making it impossible to have a perfect 50% on, 50% off square wave cycle. The closest possible value ($127/255$ and $128/255$) is 0.2% off or so, since only integer values are allowed.

16.1.3. Wire

The wire library was intended to provide nearly all the internal input/output communication among links and modules. Specifically, we intended to be using the I²C communication standard. The way this works in practice is that there are two types of components: master and slave.

The master calls the shots, so to speak. A master device tells the slave device what to do, whether it is to listen for or to send data. Slave devices do their best to accommodate the master device's wishes. Devices are not allowed to be both masters and slaves, and cannot change this designation, as the names would suggest.

The I²C communication library is perhaps the communication library that uses abstraction most. Pins and values aren't really directly referenced, except for in an obfuscated setup process. To the programmer, the library allows one to begin at the device level.

In the setup loop, one uses the `Wire.begin(address)` function to join the I²C bus. A master device need not add an address, though it is allowed. Slave devices give themselves a name, which should be unique along the Chain, in the same way that a numerical house address in the real world is unique along the street it's on. The Link is given an address identical to the number written on its outer casing, and is pre-programmed in.

The implementation of I²C for the Atmega328 places a hard limit on the number of addresses available, since addresses must be exactly 7 bits long. Additionally, the first eight addresses (0-7) are reserved. This means there are 2⁷-8, or 120 unique addresses. This is plenty, given that the maximum number of links in a Chain is an order of magnitude lower than that.

The functions available in the Wire library are as follows: `Wire.begin()`, `Wire.requestFrom()`, `Wire.beginTransmission()`, `Wire.endTransmission()`, `Wire.write()`, `Wire.available()`, `Wire.read()`, `Wire.onReceive()`, `Wire.onRequest()`, and each of these will be discussed in some detail below.

The `Wire.requestFrom()` function is used by the master device to request bytes from a given slave device. The syntax is `Wire.requestFrom(address, quantity, stop)`, with `stop` being optional. `Address` is the address of the slave device, `quantity` is the number of bytes to request, and `stop` may be either `TRUE` or `FALSE`, where `true` sends a stop message after the request, and `true` instead continually sends a restart message afterward.

The `Wire.beginTransmission()` is another master function, with a syntax of `Wire.beginTransmission(address)`, where `address` is the address of the slave device being targeted. Similarly, `Wire.endTransmission()` ends the transmission.

`Wire.write()` writes data from a slave device or queues bytes to be transmitted from a master device to a slave device. Its syntax is `Wire.write(value)`, `Wire.write(string)`, or `Wire.write(data, length)`. `Value` is a value to be sent, `string` is a string to be sent, and `data` is data to be sent. The number of bytes of data is `length`.

`Wire.available()` is another master function. It returns the number of bytes that may be retrieved by `Wire.read()`, which reads a byte transmitted from a slave device to a master device, and returns the next byte received.

`Wire.onReceive()` can call a function when a slave device receives a transmission from its master. The syntax is `Wire.onReceive(handler)`, where `handler` is the name of the function. In the same vein, `Wire.onRequest()` has a syntax of `Wire.onReceive(handler)`, and registers a function to be called when a master requests data from a specific slave device.

There are designated I²C pins on the Atmega328. Those pins are A4 for SDA, and A5 for SCL, which can be seen in the pin list which can be seen above in **Section 16.1.5**, Interface Between Hardware and Libraries.

Once again, however, the high wire capacitance entirely removed this capability.

16.1.4. Serial

Serial communication will be used solely in the interface link. It will be used to communicate between the ESP8266 and the Atmega328. Serial communication is accomplished with two lines, a transmitter (TX) and a receiver (RX). The wires cross, which is to say that the TX of one device becomes the RX of the one it's connected to, and vice-versa. For TX, think mouth, and for RX, think ear.

There are designated serial pins on the Atmega328. Those pins are D0 for RX, and D1 for TX. Because of the designation, connected devices can send or listen concurrently--there is no possibility of a collision. This allows for easy transfer of data in a timely fashion.

The serial library is rather extensive. Additionally, there are quite a few sub-libraries that use functions from the main serial library, namely ones for the ESP8266. One such is the WeeESP8266, available on [GitHub](#) [1].

This library intended to be used to transfer data to and from the ESP8266, which in turn will communicate with a Wifi network. This, however, was used only to speak with the Bluetooth unit, which directly communicates with the personal smartphone application user within a modest distance (~20 m).

16.1.5. Interface Between Hardware and Libraries

Each software library relies on the concept of pin-mapping, which is what translates software pin variables into the hardware pins. The table below, *Figure 16.1.5.1*, lists all hardware pins on the left side, and relates them to their software uses. The second table below, *Figure 16.1.5.2*, lists hardware pins on the right side, and relates them to their software uses.

Atmega Pin	Arduino IDE Pin	Link	Use
1	Reset	All	Reset
2	D0 (RX)	Interface	RX
3	D1 (TX)	Interface	TX
4	D2	Door (possible)	Input (possible)
5	D3 (PWM)	Unused	Unused
6	D4	Unused	Unused
7	Vcc	All	Vcc
8	GND	All	GND
9	crystal	All	crystal
10	crystal	All	crystal
11	D5 (PWM)	Unused	Unused
12	D6 (PWM)	Unused	Unused
13	D7	Unused	Unused
14	D8	Unused	Unused

Figure 16.1.5.1, Pin Mapping Left Side

28	A5	All	SCL
27	A4	All	SDA
26	A3	Unused	Unused
25	A2	Door (possible)	Input (possible)
24	A1	Unused	Unused
23	A0	Unused	Unused
22	GND	All	All
21	Aref	All	All
20	Vcc	All	All
19	D13	All	Ordering IN
18	D12	All	Ordering OUT
17	D11 (PWM)	All	Red
16	D10 (PWM)	All	Green
15	D9 (PWM)	All	Blue

Figure 16.1.5.2, Pin Mapping Right Side

The blank rows under the Link and Use columns indicate that the pin is unused. Unused pins are not attached to anything in the hardware domain, and similarly are not referenced on the software side.

16.2. Bootloader

In order to allow the Atmega328 to work with the Arduino ISP, the Arduino bootloader must be flashed to said microcontroller. This is accomplished by using the process outlined in the Arduino.cc tutorial "[From Arduino to a Microcontroller on a Breadboard.](#)"

This process uses a full Arduino as an in-system programmer. However, this is the only actual Arduino used in the project, and it will later be used to upload completed programs, or *sketches*, to the many microcontrollers.

17. Project Testing

In order for the project to be successful, extensive testing must be done on every component in the Dibs LightLinks system, from the hardware, to the software. In the preceding sections, an overview of how testing will be conducted for both the hardware side and the software side will be talked about.

17.1. Hardware

In this section, the testing of hardware devices and components, as opposed to the functioning of the software will be discussed. However, in certain circumstances, testing requires the use of software in order to interpret the hardware data streams. An example of this is in the PIR sensor, which relays its information to the Atmega328, which in turn will send serial data back to a computer for visual inspection.

17.1.1. Regulated Voltages

There are a set of regulated voltages output from various points in the design. They will be tested with a digital multimeter in situ, and judged according to the table below in *Figure 10.1.1.1*.

Nominal Voltage	9 V	5 V	3.3 V
Max Voltage	9.5 V	5.2 V	3.2 V
Min Voltage	8.5 V	4.8 V	3.4 V

Figure 17.1.1.1, Passing Regulated Voltages

If the measured voltages are within these tolerances, inclusive, the device passes.

17.1.2. Water Resistance

The device shall be submerged underneath water of a depth of 5 cm for a period of 30 minutes. The device shall then be opened and inspected visually for signs of internal water leakage. If no leakage is found, the device passes.

17.1.3. Coefficient of Static Friction

The device shall lie on a dry, horizontal plank of planed pine wood. A force gauge shall first be used to measure the weight of the device. The force gauge shall then be used to measure the maximum force applied to the device as it is moved from being stationary to a small velocity. The transition point between stationary and moving will be the point at which the coefficient of static friction may be inferred.

The coefficient of static friction is found by dividing the magnitudes of this force value by the weight of the device. If the coefficient of static friction is found to be greater than 0.65, the device passes.

17.1.4. Signal Transmission

The continuity tester of a digital multimeter shall be used to test the input and output data and power lines of each device. If continuity is found in each case, the device passes.

17.1.5. Human Presence Sensor

The device must reject a warm body at 2 m that is under 1 m high to pass. A human shall get on their hands and knees and emulate a ground-dwelling animal, moving at a velocity of no more than 5 m/s and no less than 0.5 m/s at that 2 m distance, and being sure to keep their body no higher than 1 m tall. The device must detect a warm body at 2 m that is above 1.5 m high to pass. A human shall walk at a velocity of no more than 5 m/s and no less than 0.5 m/s at that 2 m distance, being sure to keep their body no lower than 1.5 m.

The Human Presence Sensor will be tested detached from a Link. It will be attached to a fully-built, standard Arduino, which in turn will be connected to a standard computer via USB. The USB connection will send state information back to the computer via serial USB.

The Human Presence Sensor requires a period of time in a constant environment to acclimate and begin to output useful information. This translates to leaving the sensor in an empty (devoid of humans and other animals) room for a period of no less than 4 minutes.

17.1.6. Door State Sensor

The sensor shall detect a metal door, a plastic door, and a wooden door being closed if they are indeed closed to pass. The sensor shall detect these doors being open if they are indeed open to pass.

The Door State Sensor will also be tested detached from a Link. It will be attached to a fully-built, standard Arduino, which in turn will be connected to a standard computer via USB. The USB connection will send state information back to the computer via serial USB.

17.2. Software

The testing for the software side of the Dibs LightLinks System will be conducted through stress testing the system in different modes to ensure proper functionality. The Android application will be testing its functionalities in each of the seven different modes. More on this will be discussed in **Section 17.2.1**.

The server side of software will be tested using the security modes in order for the notifications to be sent when someone is detected in the path. More detail will be put into this in the section below, **Section 17.2.2**.

17.2.1 Android Software Testing

The Android Studio IDE, which is what was used to create the mobile application, has a great way of testing the application. It comes with an Android Virtual Device (AVD) manager which allows emulators for multiple Android devices to be downloaded to the computer for running the application. The virtual device used for testing of the Dibs LightLinks System was the Nexus 5, as shown in the screenshots of the user interface in **Section 14.2**.

In order to show how testing will work for the application side, *Figure 17.2.1.1* displays the Android Studio application with a few components highlighted. At the top of the screen, there is a green button that looks like a play symbol (surrounded by red). This symbol is used to start running the application on the AVD. When the application is

started, the Event Log at the bottom right of the screen (surrounded by green) shows what is happening with a time step beside each event. This will let the application developer know if anything faulty happens such as the build not completing or if the application exits unexpectedly. To the bottom left of the screen, the Android Monitor (surrounded by blue) is where actual debugging will happen. Notice there are five different tabs to choose from, in which all different levels of the application can be accessed and logs can be placed in the java code to document when things happen in the application. When an error occurs, it will be displayed in this monitor, in which the debugging process will take place.

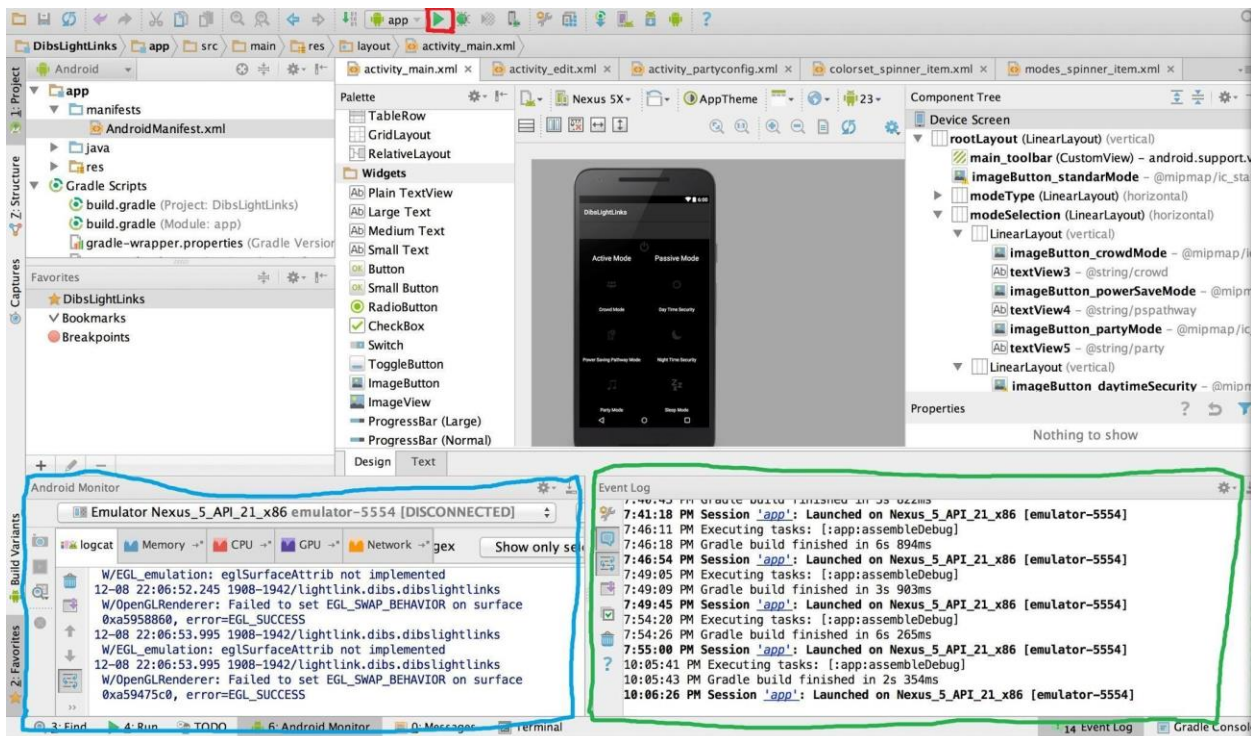


Figure 17.2.1.1, Android Studio Application Software

Testing will be conducted on each of the seven modes. This will be achieved by first selecting a mode and testing it out on its default settings. If no errors occur, then the mode will be configured to personalized settings. For each mode, all of the different possibilities of user controls must be tested individually to ensure proper functionality. As the development of the application becomes more complete, a real Android device will be used to test the application instead of the Android Studio AVD emulator.

18. Administrative Content

The Dibs LightLinks System was a group project, and like any group project, it required structure and teamwork to finish. The team met multiple times in order to discuss every aspect of each device. Many decisions had to be made to divide up the work and ensure the project's success.

The team consists of two Computer Engineers and two Electrical Engineers. The Computer Engineers are Ronauldus Woods and Nicholas Cain. The Electrical Engineers are Evan Boyar and Matthew Bolan. In order to divide the work equally, the Computer Engineers were assigned the web services, software, and programming content and the Electrical Engineers were assigned the physical structure, board designs, and electrical schematics.

When the team met for the first time, they set several standards in motion. It was decided upon that they would meet every week. If they were unable to meet in person due to schedule or location constraints, the group would meet via Google Hangouts. In order to share the work they had developed, the group would use Google Drive and Google Docs. These methods were easy, effective, and free.

Each member of the team was assigned 30 pages to complete. These pages were divided between the members both based on their classification of engineer and their specialty within their field. Upon the completion of their 30 pages, the members reconvened to finish writing sections of the paper they agreed to work on together. After finishing these sections, they pooled all their documents into one and formatted it to the required formatting specifications.

Component	Quantity	Price	Total
LED Lights	60	\$1.50	\$90
.020" x 21" x 51"	5	\$10	\$50
ATMEGA328PU MCU	12	\$3.02	\$36.24
PIR Sensors	12	\$1.90	\$22.80
Printable Circuit Boards	12	\$9.00	\$108.00
AC/DC Converter	12	\$15.00	\$180.00
Breadboard	1	\$15.00	\$15.00
All	-	-	\$487.04

Figure 18.1, Project budget

In the end, the actual budget (neglecting certain printing costs) was far less. Below (Fig. 18.2) is the actual budget. It includes the date of purchase as well as whether or not the purchase will be reimbursed according to Boeing/Leidos policy.

Item & Qty	Date	Total Cost	Reimbursement Happening?	Purchase r
(1x) USB A->B	2/13/2016	\$5.31	y	E
(3x) 2N7000	2/20/2016	\$3.83	y	E
(1x) 13 Pin Pigtail Connector	2/20/2016	\$4.26	y	E
(1x) (2/pkg) 2.1mm Coax plug	2/13/2016	\$4.27	y	E
(1x) (10/pkg) Bootloaded Atmega328	3/13/2016	\$32.76	y	E
(1x)(~10/pkg) PCBs	3/15/2016	\$33.00	y	E
(1x)(50/pkg) 22pF Capacitors	3/13/2016	\$4.39	y	R
(2x)(2/pkg) Bootloaded Atmega328	3/13/2016	\$10.99	y	R
(1x) LANMU UNO R3	3/8/2016	\$9.99	n	R
(1x) Bluetooth Module	3/4/2016	\$8.99	n	R
(1x) WiFi Bee	3/4/2016	\$19.95	n	R
(1x) FTDI to USB TTL Serial Adaptor	2/26/2016	\$5.69	y	R
(1x)(10/pkg) Crystal	2/23/2016	\$5.85	y	R
(1x) Addicore ESP 8266	2/4/2016	\$6.95	y	R
(1x) DIYmall ESP 8266	2/4/2016	\$7.99	y	R
(1x) 830pt Breadboard + Wires	12/21/2015	\$9.42	n	R
(1x) 400pt Breadboard + Wires	12/18/2015	\$8.90	n	R
(1x)(5/pkg) IR LEDs	12/12/2015	\$5.79	y	R
(1x)(50/pkg) RGB LEDs	12/12/2015	\$5.13	y	R
(2x) IEIK UNO R3	12/12/2015	\$10.29	y	R
(1x) Addicore ESP 8266	12/12/2015	\$6.95	y	R
(1x)(10/pkg) PIR sensors	12/12/2015	\$15.99	y	R
Various parts	4/1/2016	\$30.25	y	E
Various equipment	4/4/2016	\$11.29	y	E

Figure 18.2, Itemized Expenses

The figure below (18.3) is a tabulation of the full cost of the project, as compared to our expected values, and is also calculated with respect to each individual.

Current Total Costs Incurred	Current Budget Remaining	Current Overshoot
\$268.23	\$322.76	\$37.82
Person	Purchases	
Evan	\$124.97	
Matt	\$0.00	
Ronnie	\$143.26	
Nick	\$0.00	

Figure 18.3, Totals and individual expenses

19. Future Applications

The Dibs LightLinks technology has many applications currently. From lighting to crowd control to security applications, anyone can find a use for these devices. Despite the versatility it offers, however, the Dibs LightLinks System has many further uses, for both technologies currently available and technologies the future may provide.

An up and coming network focused technology is Li-Fi, or Light Fidelity. Li-Fi works by using LED based technology to alter the brightness of the light rapidly in order to send signals similarly to Wifi. Li-Fi has many advantages over Li-Fi, including speed of information transfer, however it also has disadvantages. Li-Fi requires direct line of sight for information transfer. This may prove problematic in many different circumstances, however the Dibs LightLinks System may alleviate the issue by providing floor lighting that can transfer Li-Fi signals wherever the user wishes.

As a versatile system that can be placed anywhere on the ground, the Dibs LightLinks System can have applications dealing with solar energy. Both indoors and outdoors, it's highly likely the Links will receive light from external sources, so implementing solar panels into them seems to be a logical step towards energy efficiency. As solar technology improves, it's possible that solar panels will be able to provide enough power to each individual link such that a battery link or external source of power is unnecessary. Alternatively, the battery link could be specialized to implement solar panels more effectively than other links, granting them the capability to power the whole system off solar power alone.

The security the system currently provides is meager in comparison to specialized automated home security systems. Rather than being a focused security system, the Dibs LightLinks System offers some security in addition to its primary function. In the future, however, this could change. With advances in home automation technology and sensor technology, the Dibs LightLinks System could change to be an easily movable security system.

The Dibs LightLinks System is designed especially for ground usage, both indoors and out. This may change. As new needs arise and technologies improve, so might the Dibs. The device may be placed on walls in the future, either through adhesives or through more permanent implants. In a more distant future, the Links may even hover slightly above the ground, leading to even more potential applications and usages.

As a device that can be easily picked up and moved to a more convenient location, it only makes sense that people make robots to do that for them. This is a technology that is available now, but still in a more primitive stage. As the technology improves, however, it won't be long before the Dibs LightLinks System is outfitted with small legs of other movement-focused device in order to automate the relocation aspect.

The Dibs LightLinks System is currently still primitive. Its algorithms and designs focused towards the simple security aspects and basic modes that have been implemented. Future engineers working on the device may expand drastically, writing more advanced algorithms for other implementations. These algorithms may permit the system to be able to distinguish between different individuals, and light up differently depending on how each individual requests.

Voice automation has been improving drastically in the past years. With modern smartphones being able to distinguish human words to respond to various commands, so too might the Dibs LightLinks system. If voice commands are implemented, it could also open up a fusion of technologies with household automation.

Overall, the future looks bright for the Dibs LightLinks system. From technologies people already have access to, to technologies that are on the verge of commercial use, the system offers versatility and promise. With so many up and coming technologies that can improve on what we've designed, this technology can evolve with the future.

20. Conclusion

Throughout all of history, humanity has sought light. From the discovery of fire to the creation of the light bulb, some of humanity's greatest inventions have dealt with light generation. This is because humans need light to see. People sleep at night because they cannot see, but sometimes they need to traverse the darkness. Sometimes, people need a guiding light.

The Dibs LightLinks System offers a safe and power efficient source of light. The Dibs LightLinks System is composed of multiple Links of four different flavors and one extender called the Connector. The different flavors of Links are Door, Interface, Battery, and Standard. The system is designed to be set up by connecting numerous Links on two different Chains. Each Chain must comprise of at least one Interface Link and one Battery Link, but will almost certainly contain several Standard Links.

When the Interface Link receives commands over a Wifi network, it will inform the other Links' microcontrollers of the command and perform whatever action needs to be taken. When connected to other Links, the Battery Link will provide power to the system whilst retaining the functionality of a Standard Link. The Standard Links simply receive commands via their microcontroller, act on these commands, and relay them further down the line. The Door Links are unnecessary for the system to function and are designed to provide similar functionality to the Standard Link whilst fitting under a door. Lastly, the Connector is simply a male port and a female port with an extendable wire in between them.

The Dibs LightLinks System doesn't just provide light in one mode and color of light, but several modes of operation and a vast spectrum of colors. The modes of operation come in two forms, active and passive. Active modes include Standard Pathway Mode, Power Saving Pathway Mode, Party Mode, and Crowd Mode. Passive modes include Daytime Security Mode, Nighttime Security Mode, and Sleep Mode. Some modes may activate security lights, which are simply the LEDs of a unit flashing a bright red.

Active Modes are modes where non-security lights are active. Standard Pathway Mode and Power Saving Pathway Mode lights up the system a single color, where the Power Saving mode is timed and uses different sensors. Party Mode lights up different colors

at different predetermined intervals. Crowd Mode lights up a single color in a pattern of alternating levels of brightness.

Passive modes are modes where all lights, barring security lights, remain inactive. Daytime Security Mode passively detects motion and messages the user via in-app notification. Nighttime Security Mode passively detects motion and lights up security lights on the Links where the motion was detected. Sleep Mode functions as the system's off mode, where no lights or sensors are active.

Overall, the Dibs LightLinks System is a versatile and customizable systems that has uses in every household and workspace. It can provide security, safety, and entertainment; or simply offer ambient lighting. The user can use the system to fit whatever their needs may be, and in the future, the system can offer so much more. As technology improves, so can the Dibs LightLinks system, as it adapts with the future.

21. Appendix

Citations:

1. Executive Summary

[1] <http://www.crimeinamerica.net/2010/09/30/28-percent-of-burglaries-involve-people-at-home-good-doors-windows-prevent-violent-crime/>

2. Project Description

-

3. Research

[1] https://upload.wikimedia.org/wikipedia/commons/f/fb/Virgin_America_A320_cabin.jpg

[2]

https://upload.wikimedia.org/wikipedia/commons/1/18/Path_near_floating_restaurant_with_moon,_Infosys_Mysore.JPG

[3] https://upload.wikimedia.org/wikipedia/commons/2/2e/Motion_detector.jpg

[4]

https://upload.wikimedia.org/wikipedia/commons/thumb/2/2c/LED_holiday_lights.jpg/1280px-LED_holiday_lights.jpg

[5] http://www.gradusworld.com/files/parentalpha/aisle_floortrims-1000x497px.jpg

[6] https://upload.wikimedia.org/wikipedia/commons/f/f1/RGB_LED.jpg

[7] https://upload.wikimedia.org/wikipedia/commons/a/a5/PIR_Motion_Detector.jpg

[8] https://upload.wikimedia.org/wikipedia/commons/f/f8/Ultrasonic_sensor.jpg

[9] https://upload.wikimedia.org/wikipedia/commons/3/34/IR_led.jpg

[10] https://c2.staticflickr.com/4/3380/4564772171_4d91e677e5.jpg

[11] <https://upload.wikimedia.org/wikipedia/commons/1/10/Robinetterie-PVC.JPG>

[12] <https://upload.wikimedia.org/wikipedia/commons/1/18/PIC18F8720.jpg>

[13] <http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>

[14] https://upload.wikimedia.org/wikipedia/commons/a/ac/Nexus_5_Front_View.png

4. Realistic Design Constraints

-

5. Communication Standards

[1] <https://learn.sparkfun.com/tutorials/i2c>

6. Microcontrollers

[1] <https://upload.wikimedia.org/wikipedia/commons/0/0c/ATMEGA328P-PU.jpg>

[2] https://c1.staticflickr.com/1/494/19681470919_9a9bcd5692_z.jpg

[3] <http://ecx.images-amazon.com/images/I/61b8oCGYJdL. SY355 .jpg>

7. LED System

-

8. Human Presence Sensor

-

9. Door State Sensor

-

10. Obstacle Presence Sensor

-

11. Power

[1] <http://gadgetmakersblog.com/power-consumption-arduinios-atmega328-microcontroller/>

12. Feature Set

-

13. Physical Design

-

14. Application Software

[1] <https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1>

[2] <http://blog.appannie.com/app-annie-index-market-q1-2015/>

[3] <http://opensignal.com/reports/2015/08/android-fragmentation/>

[4] <http://press.blackberry.com/press/2013/blackberry-10-re-designed-re-engineered-and-re-invented.html>

[5] <http://crackberry.com/blackberry-z10>

[6] <http://us.blackberry.com/smartphones.html>

[7] https://developer.blackberry.com/native/documentation/getting_started/ide/explore_ide.html

[8] <http://us.blackberry.com/smartphones/priv-by-blackberry/features.html>

[9] <http://www.omgubuntu.co.uk/2013/02/ubuntu-touch-developer-preview-released-available-for-nexus-devices>

[10]

[https://wiki.ubuntu.com/SaucySalamander/ReleaseNotes#Ubuntu for Phones](https://wiki.ubuntu.com/SaucySalamander/ReleaseNotes#Ubuntu_for_Phones)

[11] <http://gadgets.ndtv.com/mobiles/news/canonical-launches-first-ubuntu-phone-with-spains-bq-658711>

[12]

<http://www.omgubuntu.co.uk/2015/06/new-5-inch-ubuntu-phone-goes-on-sale>

[13] <http://www.phonesreview.co.uk/2015/02/26/meizu-mx4-variant-will-be-second-ubuntu-phone/>

[14]

<http://developer.android.com/training/improving-layouts/optimizing-layout.html>

15. Server Software

[1] <https://aws.amazon.com/>

[2] <http://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

16. Arduino/Atmega328 Software

[1] https://github.com/itead/ITEADLIB_Arduino_WeeESP8266

17. Project Testing

-

18. Administrative Content

-

19. Future Applications

-

20. Conclusion

-

21.1. Permissions

This section consist of the different forms of proof that this document has or is in the process of gaining the permission to use these licensed materials.

21.1.1. PuTTY

PuTTY is copyright 1997-2015 Simon Tatham.

Portions copyright Robert de Bath, Joris van Rantwijk, Delian Delchev, Andreas Schultz, Jeroen Massar, Wez Furlong, Nicolas Barry, Justin Bradford, Ben Harris, Malcolm Smith, Ahmad Khalifa, Markus Kuhn, Colin Watson, Christopher Staite, and CORE SDI S.A.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.



THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

21.1.2. Amazon


Ronauldus Woods, University of Central Florida 



Ronauldus Woods

To:  copyright@amazon.com; 



Reply all | 

Wed 12/9/2015 11:30 PM

My name is Ronauldus Woods and I am a Senior Computer Engineering student at the University of Central Florida. In my group's senior design project we're using the Amazon Web Services in order to develop a web application for our design to interact with. I've really enjoyed my experience with AWS thus far and I was wondering is there any chance that I could have permission to use screenshots that I take throughout my process of making the web application as well as write about the steps I take in making the web application in our documentation of our design? Thank you so much for your time and have a wonderful day.