Wireless-Enabled Smart-Lights Hub Prototype

DESIGN DOCUMENT

Senior Design December 2019 Team 16

Dr. Ravikumar Gelli

Dr. Manimaran Govindarasu

Alexander Beaver

Ryan Bush

Aaron Ramsey

Logan Zasada

http://sddec19-16.sd.ece.iastate.edu/

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

Our faculty adviser/client is Dr. Manimaran Govindarasu and will be a mentor for the project by helping us manage our time and progress, as well as give guidance for the direction of the project and any problems we face.

Manimaran's post Doctorate, our main client, is Dr. Ravikumar Gelli, who will help make large decisions in regards to the direction of the project, as well as give feedback on ways we can improve the project as it is created.

1.2 Problem and Project Statement

Presently, the PowerCyber lab uses a number of relay boxes that are hardwired to light bulbs to display the status of power components in a simulation. This system is used as an outreach tool for the general public and a training tool for industry partners. The main problems with this system are that it can't be connected to enough lights to represent larger simulations, and it's not portable enough to bring to most outreach locations. To address these problems, our team is developing a system that will take inputs from the relay boxes and wirelessly control a portable display of wireless lightbulbs.

The light bulb display system can be made as large as it needs to be by adding or removing light bulbs from it. This approach wouldn't work with the existing system because the light bulbs have to be hardwired to the relay boxes and there isn't enough space near the boxes to neatly arrange the bulbs. Because the system will utilize wireless lightbulbs, it will be portable enough to take to any outreach location. By implementing this system with will increase the PowerCyber labs' ability to perform outreach and train industry partners.

1.3 Operational Environment

The light bulb display system will only be operated indoors. The temperature of the system is expected to be around room temperature and not fluctuate. Some dust may accumulate on the system if it remains in one location for an extended period of time. The relay boxes are stationary, so the part of the system connected to these boxes will likely accumulate some dust and experience slightly higher temperatures. Overall, the expected operational environment is stable and doesn't pose any significant design challenges.

1.4 Intended Users and Uses

For the proposed system there will be two groups of users: primary and secondary. The primary user group are the members of the PowerCyber lab. The primary users will be the ones that configure, maintain, and move the system. System configuration will consist of mapping relay box outputs to specific light units. System configuration will also consist of routine setup tasks such as placing light units on to the magnetic mounting board. To maintain the system, the primary users will need to make sure the light units are charged before the system is deployed and verify the mapping between the relay outputs and the light units. To move the system, the primary users will install the magnetic mounting board at whatever location the system is to be used.

The secondary users of the system will be industry partners and the general public. The primary interaction of the secondary users will be observing the system and presentations from the PowerCyber lab to learn about cyber-security in a power systems context. Industry partners will also learn how cyber-security principles and intelligent grid management can be used to improve the quality of their operation. Because the secondary users will only be observing the system it's important that the system will have pleasing aesthetics to make sure they're satisfied.

The following case diagram simply illustrates how each user is intended to use the system.

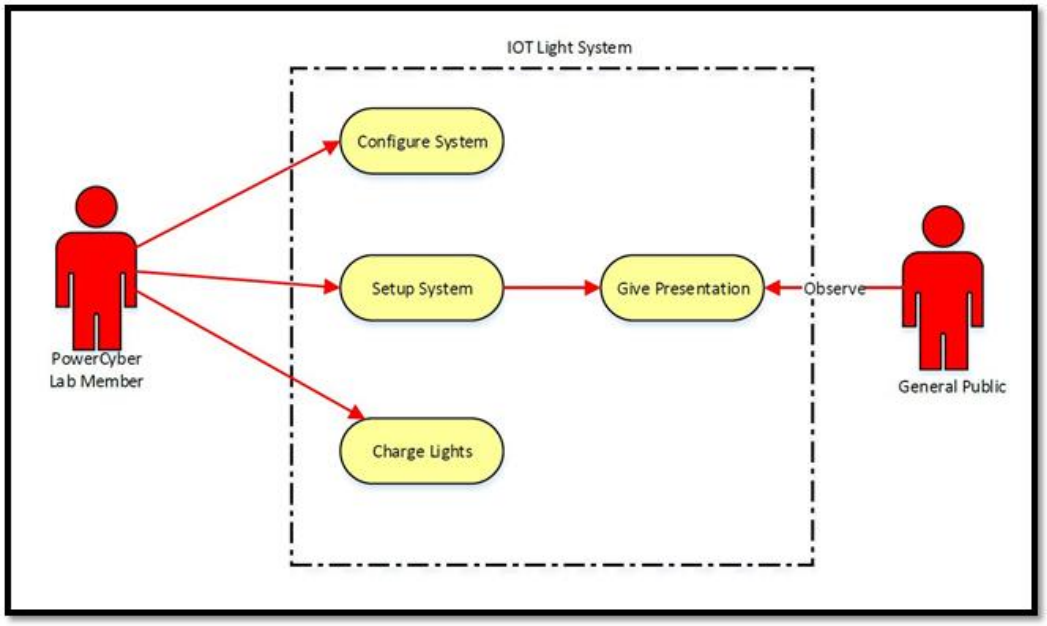

Figure 1: Case Diagram of Light System

1.5 Assumptions and Limitations

1.5.1 Assumptions

- Wi-Fi connectivity will always be present whenever the system is in use
- Upon completion of the project, maintenance and care of system will transfer to Dr. Manimaran Govindarasu
- Project will be funded entirely by Dr. Manimaran Govindarasu within reasonable costs
- PowerCyber Lab Members will be the only ones handling the system

1.5.2 Limitations

- System must take input from relay modules directly
- People outside of those authorized will be unable to initiate any wireless communication between any part of the system
- Lights must be battery powered and last for a minimum of 8 hours

1.6 Expected End Product and Deliverables

The end product for this project will be multiple remotely-controlled wireless-enabled smart-lights that have a rechargeable lithium-ion battery will be charged from a docking station and have a battery life of approximately 10-12 hours. The lights will receive data wirelessly from the transmitter junction box (XBee Gateway with Ethernet capabilities) and determine which units (light modules) need to be illuminated accordingly. To communicate this data the XBee Gateway and XBee modules will be using Zigbee (as opposed to WIFI or Bluetooth). Since the already used system for determining the status of the data is just on computer software and only has a few LEDs to hold this status, it can often be difficult to tell what each light represents according to the data of the substation. Therefore, the purpose of the new system is to be organized similarly to the software's visual diagram visual but to give the system a much more aesthetically pleasing visual which is easy to understand. The lights will be magnetic and able to be attached to a magnetic board with a projected geographical representation according to the region of interest. Since this system will be portable, anyone who may have a need or use for it would theoretically be able to use the half of the system with the XBee gateway and modules as long as there is an ethernet connection and a place to plug in the recharging station.

The recharging station should just have the ability to plug into a wall and then have each magnetic light unit plugged into it to be recharged to a battery power high enough to last about 10 hours.

By the end of this semester, we plan to be able to take the data from the relays to at least one light and have the light turn on. This is a good benchmark because once this is achieved, we will just need to scale up the process for many more lights, which will take time to order and set up correctly. If we accomplish this by the end of this semester, we will have ample time to order parts over the summer and get started right away by next fall.

2. Specifications and Analysis

2.1 Proposed Design

The final design will be composed of 4 modules: The Transmitter Junction Box (TJB), individual light modules, a charging station for the lights, and a software user interface.

Functionally, the TJB will give power to each port of the relay boxes it is connected to. For an individual port, if the contact in that relay is closed, the power will be returned to a corresponding port in the TJB, demonstrated in figure 2 below. This will tell the TJB that the corresponding port is closed, updating the TJB's memory map of ports. This memory map has 2 inputs, the direct ports that correspond directly to a light, as well as the configuration software, discussed later, that allows a user to override the port map and program their own function for the lights, in the event of a demonstration.

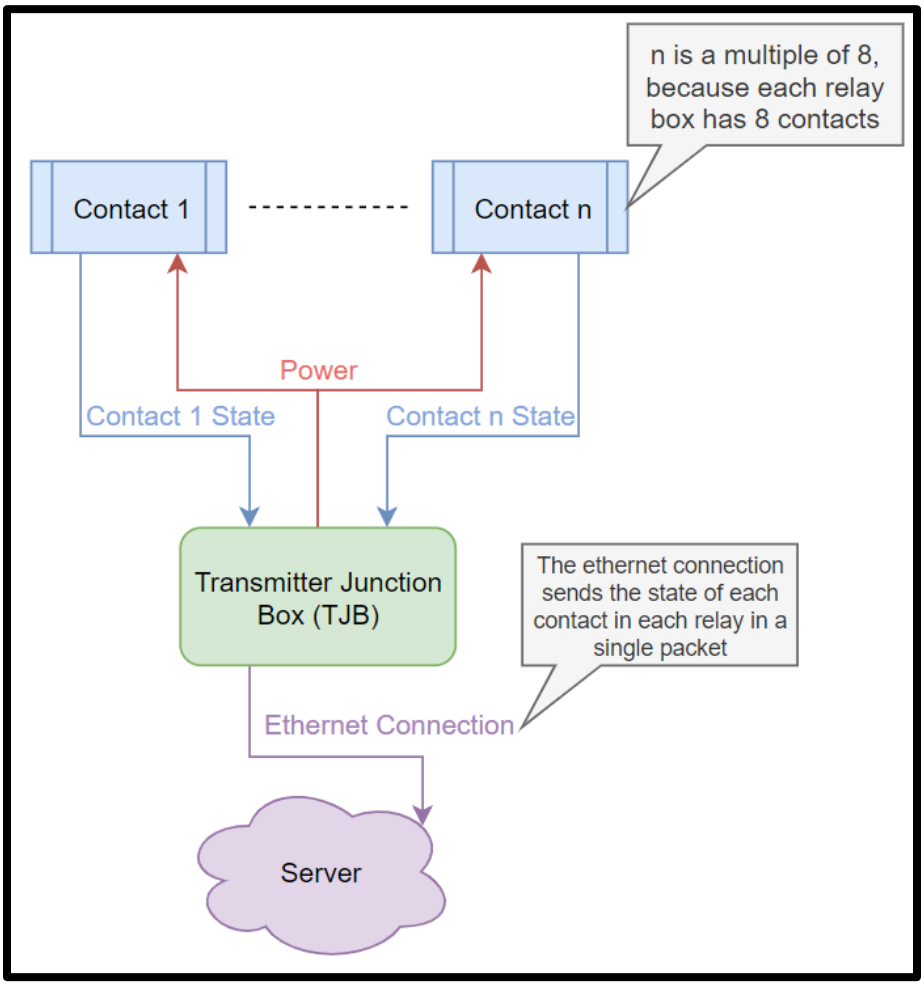

FIGURE 2: BLOCK DIAGRAM OF TJB

The TJB's output will be a connection to the internet where it updates a known page with a map to lights and their requested state, on or off. There will be a local computer which then reads this information and tells each individual light whether to turn on or off, demonstrated in figure 3. The lights will contain a battery, the light itself, some sort of controlling system, and a wireless communication module. The exterior of the light will have easy to access terminals for quick connection to the charging station.

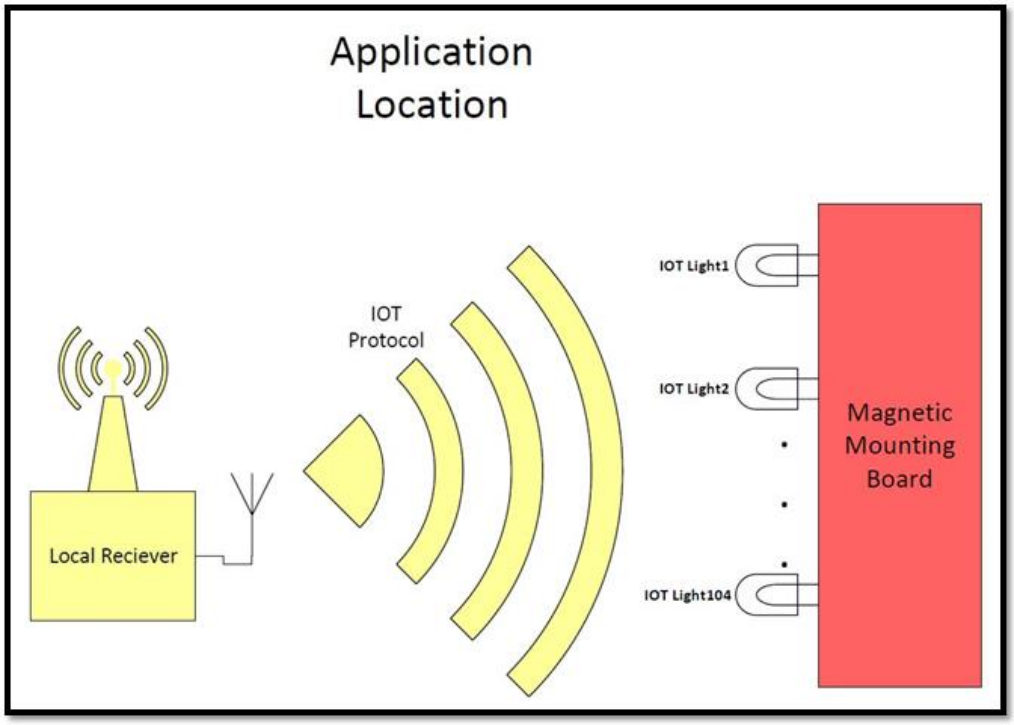

FIGURE 3: COMMUNICATION BETWEEN LOCAL RECEIVER AND INDIVIDUAL LIGHTS

Charging stations will come in blocks of 8 stations tied together, giving the ability to charge lights in batches of 8, as described in figure 4 below. The stations are assumed to be plugged in semi-permanently and distribute power to individual light modules.

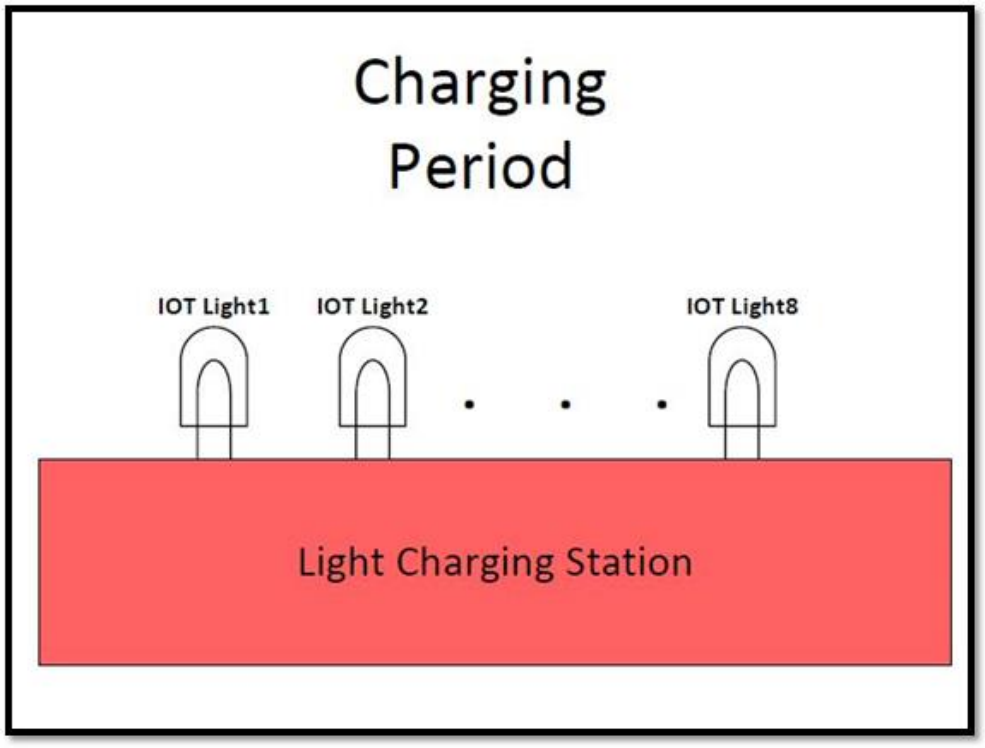

FIGURE 4: DIAGRAM OF CHARGING STATION

The software system, at a minimum, will allow functionality of a user individually programming lights so they can set patterns for a light display. We plan to have a web-based interface that would mean users travel to a certain site, give some login credentials (for security) and allow them to program the lights graphically. They would have a light, know its ID, and then in the interface, choose that light by its ID and tell it what to do.

2.2 DESIGN ANALYSIS

Thus far, most of our efforts have been focused towards planning, communication of different roles, learning about how to interface the different parts of the system, and writing the associated documentation and making the diagrams.

We have ordered many different parts to use for this project in order to determine which plan of action will be the most productive. We ordered a few different types of lights. The issue with our design was that we originally planned to buy a DC powered light that has the Zigbee capabilities, but we haven't been able to find one with DC capabilities, only AC powered lights. At this point one of our biggest difficulties is figuring out how to hack into

one of the AC lights to use its Zigbee capabilities but make it DC powered. At this point, we plan to continue down this path as it does seem to be our biggest chance of success.

The design of the system consists of three main sub-systems. The first is where the relays are transferring the data to the Arduino with ethernet capabilities. This part of the project is pretty straight forward and members in the group have prior experience with it so it is definitely one of our strengths.

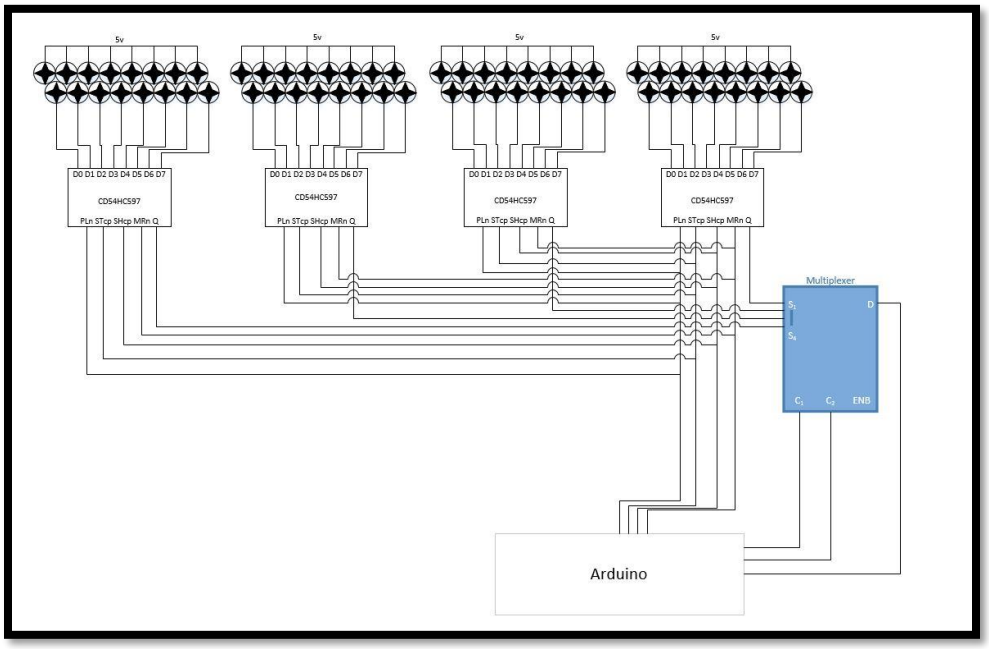

FIGURE 5: RELAY SUB-SYSTEM

Next the data is sent and received to the server from both the microcontroller (arduino) and the Zigbee hub (XBee Gateway). The server is moving along quite well, so this would be considered one of our strengths at the moment too.

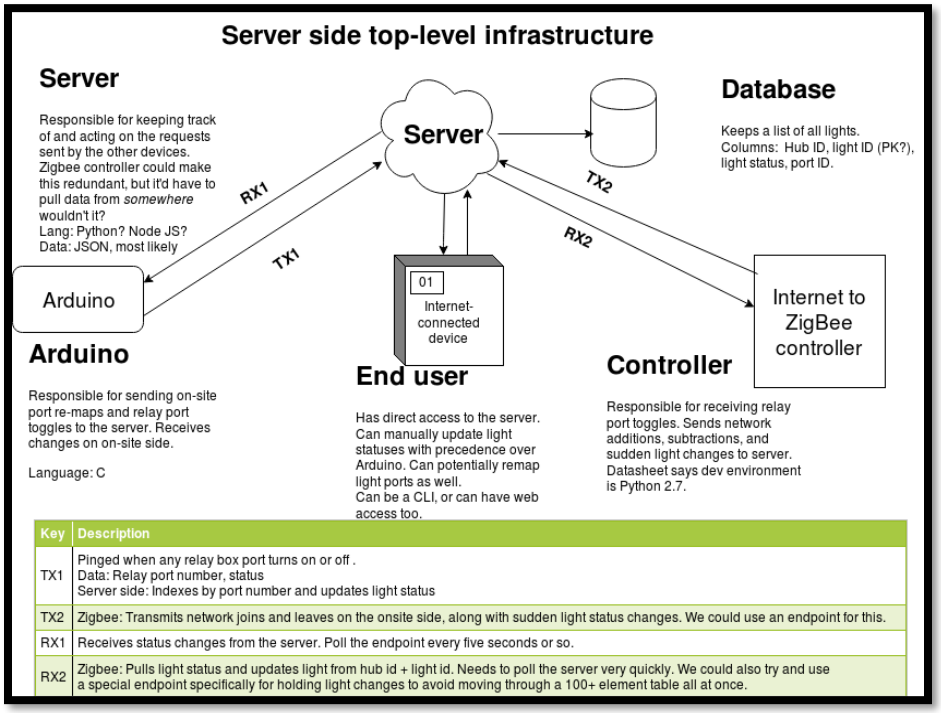

FIGURE 6: SERVER SUB-SYSTEM INTERFACE

Finally, the Zigbee hub will send this correct status of the light to the modules themselves using Zigbee. This is where our team is the weakest since no one had ever heard of Zigbee before, much less worked on it. That is the reason two of the group members are focused on this portion of the project.

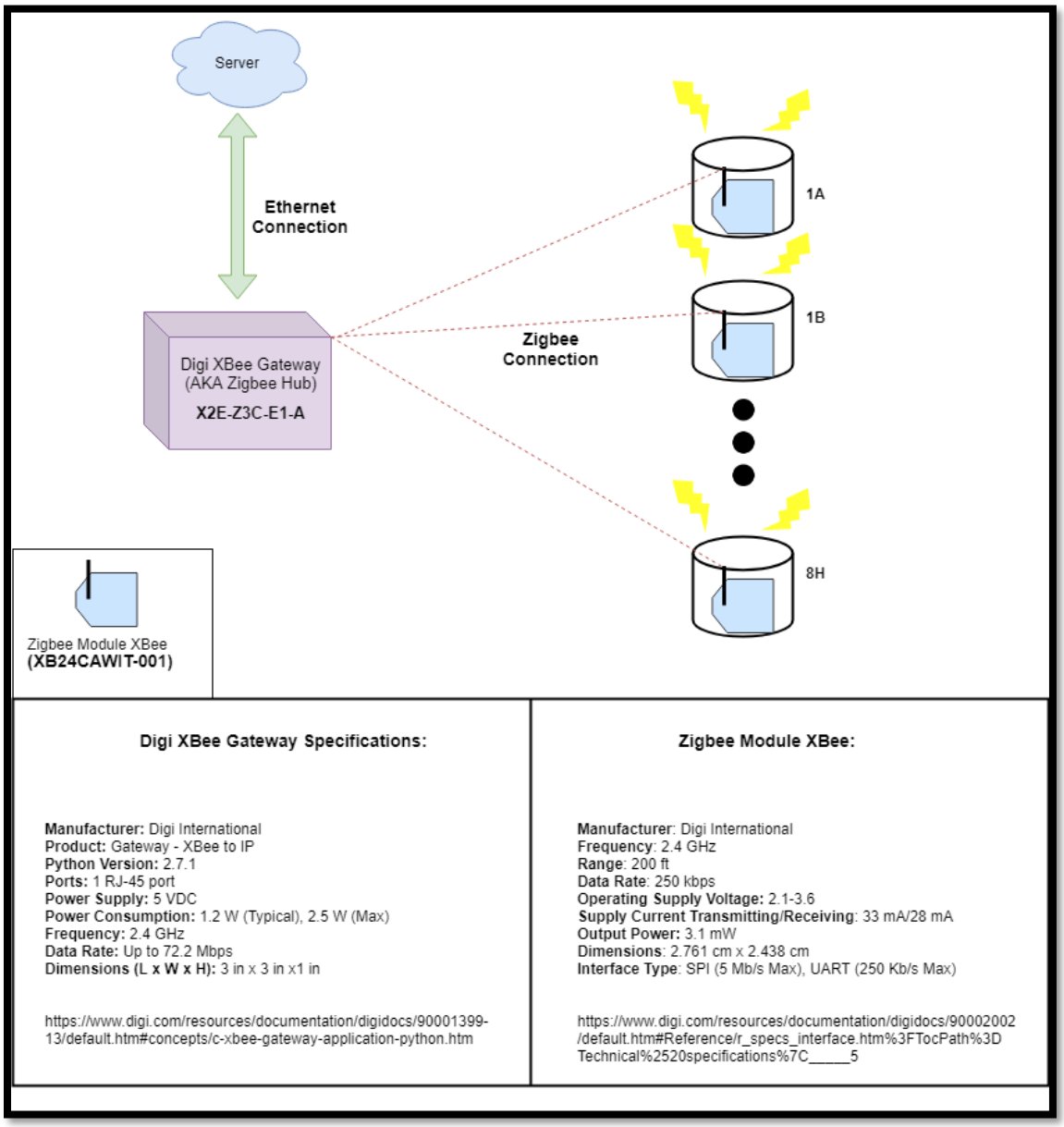

FIGURE 7: ZIGBEE SUB-SYSTEM INTERFACE

3 Testing and Implementation

3.1 Interface Specifications

For testing the project, there are a number of main interfaces that are used for communication. The two most important are the interfaces between the relay and the server and between the server and the server and the lights.

3.1.1 Relay-to-Server Interface

The interface between the relay and the server is largely expected to be made out of HTTP requests as per section 2. Whenever any relay box changes, an HTTP POST request is targeted at the server's IP address. The port the request is serviced by should be port 80. Furthermore, a custom endpoint could be created to specifically service light updates. If this was to be done, the data transferred could be in the JSON format.

On the server side, the endpoint webpage would receive the request and update the database. The updated database would be reflected on the user's screen.

3.1.2 Server-to-Light Interface

The server-to-light interface is split into two parts: The server-to-hub interface and the hub-to-light interface. For the server-to-hub interface, the server would signal the on-site hub that there is some change in the relay box. If the transceiver monitors a web page, the server would pass on the earlier relevant changes to that webpage. Otherwise, the transceiver itself should be notified.

The hub-to-light interface would be largely dictated by the IEEE 802.15.4 standard, which governs low data rate communication between devices. No matter how the hub picks up server updates, the transceiver would communicate the changes to the lights using the IEEE standard.

3.2 Hardware and Software

As the project is developed, we will need to test it at various levels of integration. E.g. we will need to test the Transmitter Junction box by itself, and then the transmitter junction box with the configuration software, and then the IOT light units and so on until we're testing the fully assembled system. To do this testing we will need to utilize various hardware and software tools.

3.2.1 Hardware to be used

3.2.1.1 Breadboard

To prototype all the circuits that we will be using we first need to order ICs and other components and test them individually. To do this, we will be using breadboards to create test circuits and using lab equipment to determine circuit behavior.

3.2.1.2 IOT Light

To fully grasp how the Zigbee protocol works we're first going to work with an off-the-shelf Zigbee light and Zigbee radio. This testing will be useful as it's very likely that the IOT light unit in the final product will be a hacked version of a Zigbee light. An off-the-shelf solution that meets our requirements hasn't been found.

3.2.1.3 Lamp Fixture

The Zigbee lights we've ordered are designed to operate on 120VAC from light bulb sockets. We will be using a lamp fixture to power the lights while testing Zigbee communication options.

3.2.1.4 Arduino

We plan to use microcontrollers for some sub-systems in the final product. Rather than testing with a specific microcontroller, we will do our testing with Arduino's as the development environment is much friendlier and will allow us to prototype quicker.

3.2.2 Software to be used

3.2.2.1 Putty

Because this is an IOT project we will need to emulate IOT inputs and outputs. Using putty will allow us to quickly verify IOT TX is correct and simulate outputs for verifying IOT RX.

3.2.2.2 Python

We will be using python to write and test protocols for interacting with the light status database. Python is a very high-level language which will allow us to rapidly prototype and test our software schemes.

3.2.2.3 MySQL

To properly test database reads and writes, we need to visually see the database. MySQL will be used to access a database in order to read or write entries to it. Furthermore, it comes with a python API to interface with the database.

3.3 Functional Testing

As each aspect of the project is completed, we will need to test our project thoroughly to ensure smooth transitions and a satisfactory final delivery. When each individual module is complete, there will be unit tests. When two adjacent modules are complete, there will be integration testing between the two. When the entire project is complete, we will conduct system testing to ensure that everything works as expected together, then conduct non-functional testing as outlined in the next section.

3.3.1 UNIT TESTING

Unit testing will be broken up into each individual module: the relay-to-server module, the server module, and the server-to-light module.

3.3.1.1 RELAY-TO-SERVER

The function of this module is to take inputs from each relay and upload the results to the server. The tests we will conduct before integrating with server are two-fold: Testing that this module can communicate with multiple relay boxes, and testing that this module can turn the data received from the relay boxes into a format ready for communication with the server.

To test that the module can communicate with multiple relay boxes, we can plug the module into a computer to observe its data output, and set up a data stream to show the state of each individual relay. We can then plug it into multiple relay boxes and change states on each individual relay and observe that it is updating in our module.

Once we know that the module can communicate with relay boxes, we can now check that the module packs the data correctly into packets that the server will be able to read. Without the server, we can't test that the server receives them, but we can have it print out what we plan to send to the server, and make sure that the data is in the agreed upon format and each relay is represented in the way it's supposed to.

3.3.1.2 SERVER

The server has 3 interfaces, and to unit test the server, we would need to test that each interface works and that the internal "wiring" is correct. By internal wiring, we mean that

if we tell the server to update a certain element in the database, it updates the correct element, as well as updating each interface that's relevant to the data.

To test each interface, we'd follow a similar procedure to testing the internal wiring. We would change data in one interface, and then ensure that the data we entered updates correctly in the database. With the output interface, we'd make sure that what is in the table is what is actually getting sent out.

3.3.1.3 SERVER-TO-LIGHT

To individually test the server-to-light module, we can plug in switches to a breadboard, then connected to an Arduino, to pretend to be connected to the server. Each switch will act as an entry in the table which corresponds to an individual relay. Flipping a switch should turn on one of the lights. If this works, we can integrate the server and the server-to-light module and begin integration testing.

3.3.2 INTEGRATION TESTING

There are two interfaces between modules and these are what need to be tested when integrating. The two interfaces are the server interacting with the other two modules. The relay-to-server module just needs to automatically update the table, and this can be tested by plugging the relay-to-server module into the internet, toggling individual relays, and verifying that they update the server.

The server-to-light module will be more fun to test because we get to play with lights. Server-to-light testing can be done by going into the server on a personal computer, manually changing the "light state" column of the table, and watching that lights automatically turn on.

3.3.3 SYSTEM TESTING

System testing will basically be a combination of the two integration tests at the same time, and we should be able to toggle an individual relay, and ensure that the corresponding light turns off.

3.3.4 ACCEPTANCE TESTING

To see if our client and adviser are satisfied, we will basically perform a system test in their presence, teach them how to use each aspect of the system, and take their feedback. If the user interface of any module is too tricky or ugly, we will take that feedback and if we have time, fix it.

3.4 Non-Functional Testing

The final non-functional testing of the system will happen when the subsystems have been fully developed and integrated. At this point, any compatibility issues would have been resolved, so we need to verify the system is secure, usable, and performs to our expectations.

To verify the systems security, we will let it operate undisturbed for our specified maximum operation time. If the system doesn't have any random changes during this period, we will know that it has the proper security level.

To test the usability of the system, we will have each of the team members use the system in all use cases. After the testing by the team members has been done, we will have the project advisor setup the system, and test the system for their specified use cases. After this testing is done, we know qualitatively how usable the system is.

To test the systems performance, we will do the usability test for our system but compare the systems performance to all the specifications outlined in the project plan. This will let us know if our system is performing at the agreed upon levels.

3.5 Process

Figures 8 & 9 below illustrate how we plan on testing our proposed solution. To test our relay to server communications we first tested the circuits that communicate from the relay to the transmitter junction box. Then we tested the communication scheme for the transmitter junction box to the server. To test the server, we used scripts to emulate the expected input from the transmitter junction box and verify that the light statuses in the server changed as expected. To test the server to light communication scheme, we manually manipulated the light statuses in the server and verified that the server to light communication module changed the physical light statuses. Finally, we put everything together and used switches to emulate the relays. By toggling the switches, we can verify the functional performance of the system. Unfortunately, it takes a while for parts to get here so we are still working on testing our proposed implementation method for each subsystem. As this testing progresses and we validate/invalidate various methods it's possible that our testing process will also change.

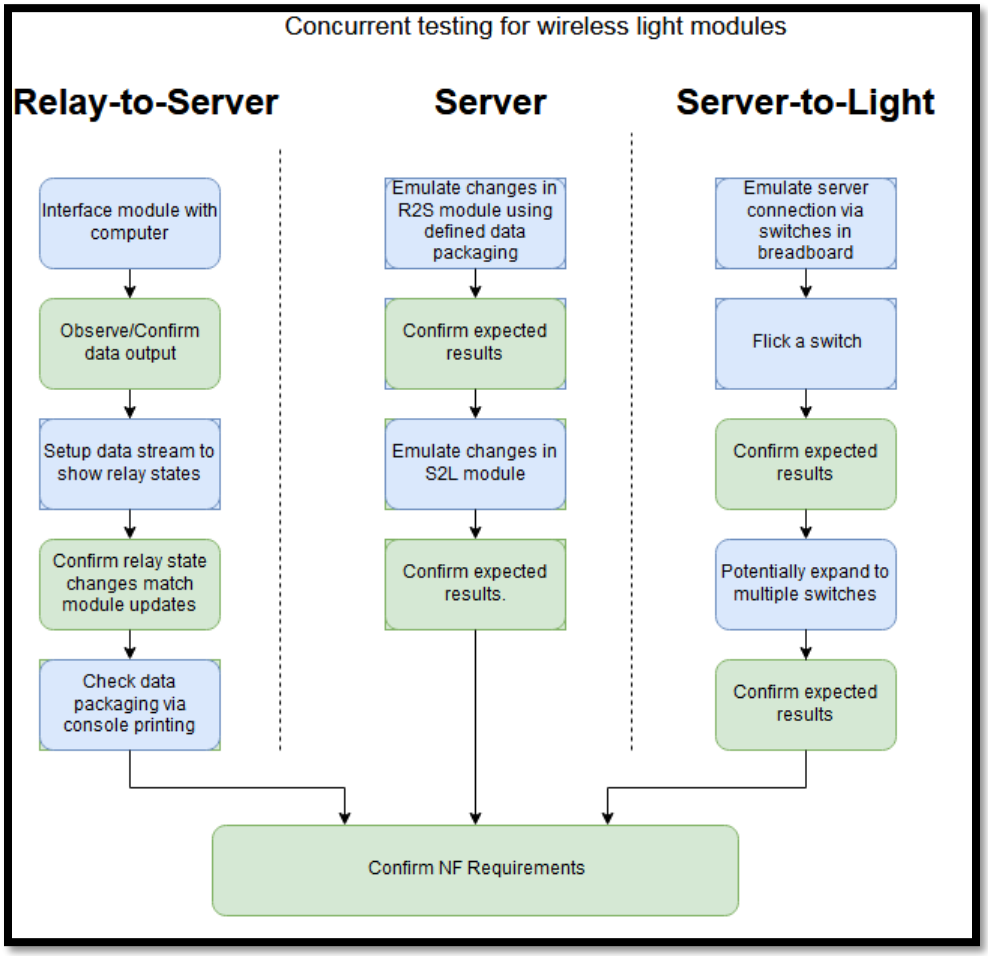

FIGURE 8: PRE-INTEGRATION TESTING FOR WIRELESS LIGHT MODULES

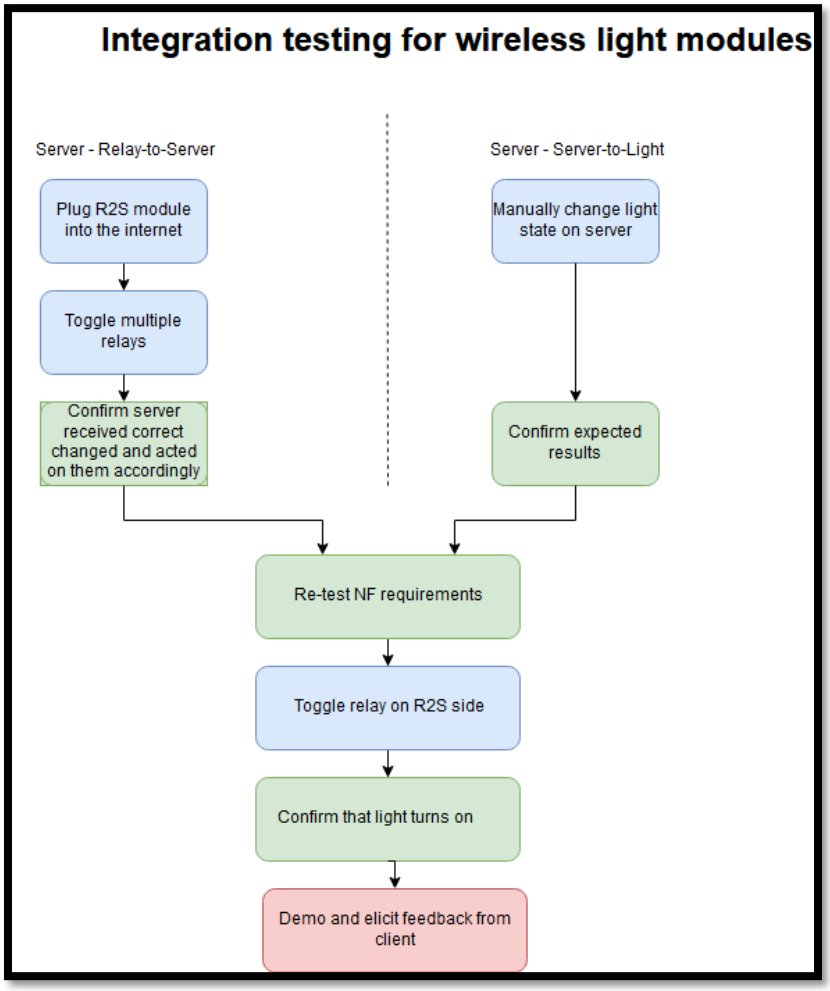

FIGURE 9: INTEGRATION AND POST-INTEGRATION TESTING FOR WIRELESS LIGHT MODULES

3.6 RESULTS

3.6.1 Transmitter Junction Box Results

So far, we have tested the IC that will serialize the parallel data stream from the relays and feed a single serial data line into the microcontroller. The oscilloscope output in figure 7 shows how by feeding the IC an input clock (yellow) we can read the relay status by checking the IC output (green). The test circuit was setup such that the first relay was off and the rest alternated off and on. The next step is using a multiplexing/demultiplexing

scheme to drive multiple ICs at the same time. Then we will prototype the interface from the transmitter junction box to the server

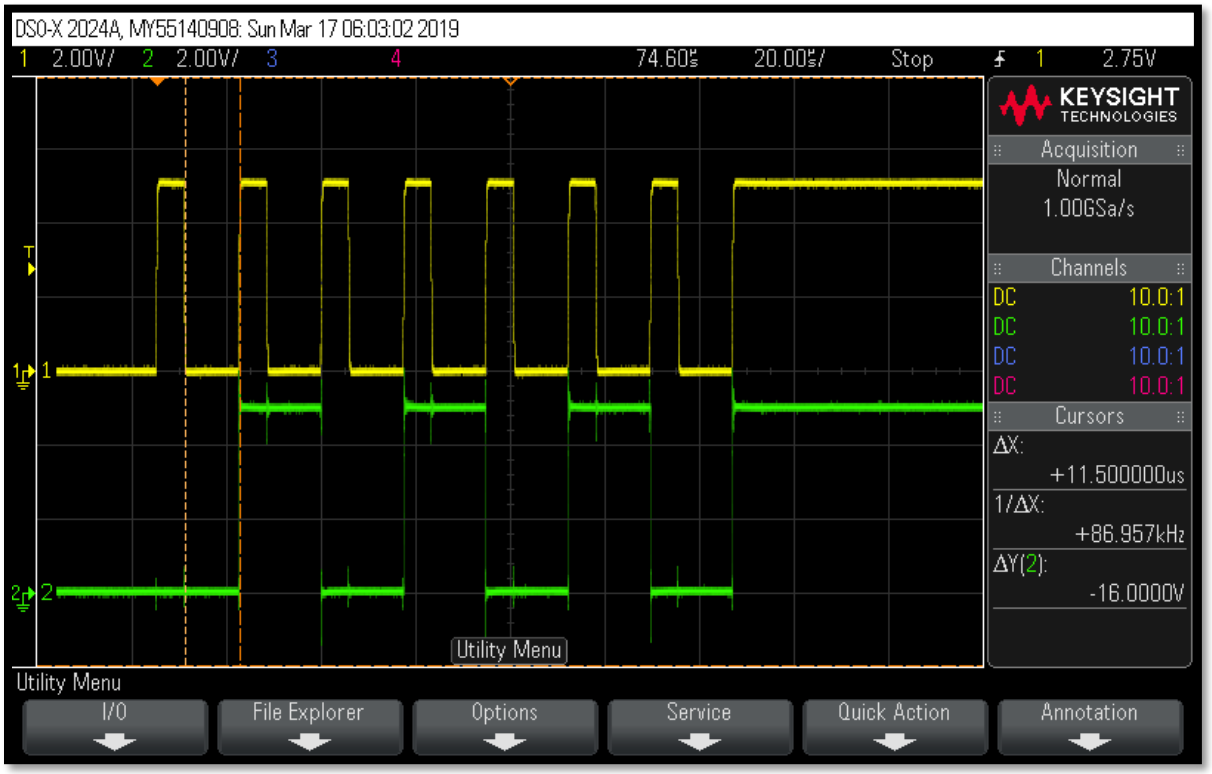

FIGURE 10: OSCILLOSCOPE WAVES OF IC CLOCK (YELLOW) AND IC OUTPUT (GREEN)

3.6.2 Server Test Results

Using python, we have been able to successfully modify an SQL data base containing light statuses through a locally hosted server. An example output can be seen in figure 8. The next step is writing a script to emulate the IOT commands that will come from the transmitter junction box and using the simulated inputs to change the light status in the database.

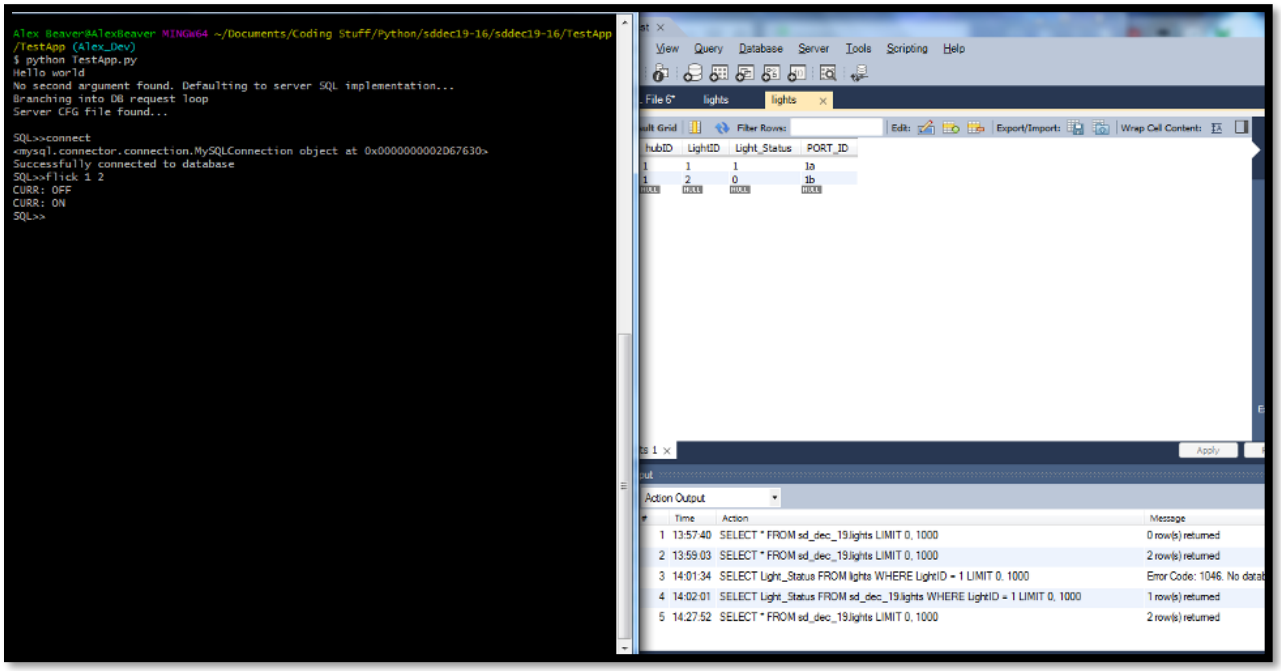

FIGURE 11: CONSOLE PROGRAM OUTPUT (LEFT) AND RESULTING DATABASE AFTER CHANGES (RIGHT)

3.6.3 Light Unit Testing

Unfortunately, all of the parts that we need to test the light units haven't arrived in Ames yet. At the time this report was written, we have no test results to report for the light units.

3.6.4 Implementation Challenges and Issues

So far, the most challenging thing has proven to be getting all the materials in a timely manner. Because of lead and shipping times our testing schedule has been seriously delayed from what we originally planned. We've also faced challenges in extracting design requirements and have continually updated our design to verify if it meets our client's expectations. As testing continues, IOT integration may provide a significant technical challenge as none of the team members have any IOT experience.

4 Closing Material

4.1 Conclusion

Up to the point of writing, we've begun prototyping each module. The server currently contains the table and can be updated by our personal computers. The relay-to-server module has a detailed schematic and is breadboarded. The server-to-light module is a little further behind because we are struggling to find purchasable lights that have the functionality we require.

By the end of this semester, we want to have a working prototype of each individual module so at the beginning of next semester we can begin integration and testing. To achieve this goal, we have a subject matter expert working on each of the first two modules (server and relay-to-server), and the other two team members are constantly researching and working on the server-to-light module, and slowly becoming subject matter experts.

Our current design is still a valid solution because it makes the system portable and increases the number of lights the system can handle. After discussing all wireless communication methods, we decided to go with Zigbee between the server and lights because of its superior performance in low data rate applications and its ability to handle many devices. We chose to use the schematic we're using for the relay-to-server box because it is modular and easily scalable, and low cost.

4.2 References

https://www.digi.com/pdf/ds_xbeegateway.pdf

Digi, "Digi XBee Gateway Family datasheet." Digi, 2017.

https://www.digi.com/resources/documentation/digidocs/PDFs/90001398-88.pdf

Digi, "Quick Start Guide XBee Zigbee Cloud Kit." Digi, 2016.

http://ftp1.digi.com/support/documentation/90001503.htm

Digi, "XBee ZigBee Cloud Kit Getting Started Guide," *XBee ZigBee Cloud Kit Getting Started Guide*, 31-Jul-2018. [Online]. Available: https://www.digi.com/resources/documentation/Digidocs/90001503/Default.htm. [Accessed: 25-Mar-2019].

http://ftp1.digi.com/support/documentation/90001399-13.htm

Digi, "XBee Gateway User Guide," *XBee Gateway User Guide*, 09-Oct-2017. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/90001399-13/default.htm. [Accessed: 25-Mar-2019].

http://ftp1.digi.com/support/documentation/guide_to_transmit_data_to_Device_Cloud.pdf

Digi, "A Beginner's guide to send data to Device Cloud from a ZigBee Network." Digi, 12-Sep-2014.

http://ftp1.digi.com/support/documentation/xbee_gateway_xbeefw_update.pdf

Digi, "Upgrade the XBee module firmware on an XBee Gateway." Digi, 01-Dec-2015.