IoT Smart Light

DESIGN DOCUMENT

Sr. Design Team 21-20
Client: Govindarasu Manimaran
Adviser: Gelli Ravikumar

Team Members and Roles:
Hamza Mostafa Sayed Mahmoud - Hardware
William Gavins - Hardware
Xinlei Yu- Software
Nathan Orts - Software

Team Email: sddec21-20@iastate.edu
Team Website: https://sddec21-20.sd.ece.iastate.edu

Third edition: April 25 / Version 3.0

Executive Summary

Development Standards & Practices Used

This system will use the Zigbee wireless communication protocol for communication between the smart lights and the Zigbee controller.

Summary of Requirements

- Lights must be able to operate continuously for 8 hours.
- Lights must be able to change to at least 4 different colors.
- Lights must operate wirelessly while being controlled from a host computer connected to the PowerCyberLab simulation.
- Latency between a relay status change and light status update must be less than one second.
- Light enclosures must be entirely self-contained.
- Software interface between the user and the light system must be easy to use and understand.

Applicable Courses from Iowa State University Curriculum

- CPRE 281 & 288 & 430 & 450 & 489
- COM S 309 & 311
- EE 201 & 230

New Skills/Knowledge acquired that was not taught in courses

- PCB design is by far the most important skill we are to acquire outside of class.
- Embedded IoT communication protocol (i.e Zigbee) usage is also another skill we are to learn

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

We would like to express our sincere gratitude to Dr. Gelli Ravikumar as our advisor and Dr. Govindarasu Manimaran as our client and secondary advisor. They both provided invaluable guidance, feedback, and resources to ensure the successful completion of this project.

1.2 Problem and Project Statement

- Problem Statement

Currently, when running power grid simulations in the PowerCyberLab in Coover Hall, the status of relays within the simulated power grid is shown by twelve lights mounted on the hardware running the simulations. In the past, when doing outreach demonstrations or presentations, there would be a livestream of the lights via a video camera in the lab. Though this worked, it is hard for viewers to visualize the simulated power grid and the status of the relays when the status lights are fixed to the hardware, rather than being arrayed over a map of the simulated power grid.

- Solution Approach

Our goal is to provide a way to more easily demonstrate the power grid simulations in outreach presentations. We would like to use wireless, magnetic lights in an internet of things environment such that the simulated power grid can be projected onto a surface and our lights can be placed where the relays are located in the grid, providing a better visual aid for the demonstrations.

The deliverables for this project are a wireless, magnetically mountable LED light, a Zigbee coordinator that controls the lights, and software that configures the lights to fit the current simulation and transmits the relay statuses to the lights to update the demonstration in real time.

The Light display system needs to have enough nodes to fit the simulations possible within the PowerCyber lab, as such this network scheme for the IOT lights needs to be highly scalable. Each node will be able to be configured to represent individual relays in the PowerCyber Lab

1.3 Operational Environment

Our wireless light system will be used in indoor presentation venues, such as classrooms, lecture halls, or electronics labs. Our system must have access to the internet so that it can communicate with a server on campus that serves as the intermediary between the light system and the power grid simulation in Coover hall.

1.4 Requirements

Previous team's requirements	Our team's requirements
Functional Requirements ➢ Accept input from 100 relays. ➢ Output to 100 IOT lights. ➢ Capability to map relay inputs to light outputs via UI. ➢ IOT lights should be magnetically mountable. ➢ IOT lights should be able to operate continuously for 8 hours. ➢ IOT lights should come in at least 4 different colors. ➢ IOT lights must be operated wirelessly. ➢ System should be capable of charging at least 8 lights at once.	Functional Requirements: ➢ Host PC communicates with PowerCyber Lab Simulation. ➢ Hardware Coordinator communicates with the host PC. ➢ IOT Light devices communicate with the Hardware Coordinator. ➢ Light devices are configured on the host PC to represent relays or nodes in a simulation. ➢ Light device shows the state of a relay or node in a simulation using an RGB LED. ➢ Light devices are battery powered, with a battery life of eight hours or more. ➢ Light devices must be operable while charging. ➢ Light devices must be operated on a scalable network topology. ➢ Light devices are able to send integer data to the host PC to trigger an attack on their respective relay.
Non-Functional Requirements ➢ System should be easy to move through standard door frames. ➢ Magnetic mounting board should not obscure any image projected onto it. ➢ Light status should update in less than a second when the corresponding relay is updated. ➢ Relay interface components should be easy to connect. ➢ IOT lights should be similar in size to existing solution(s).	Non Functional Requirements ➢ Light enclosure is aesthetically pleasing and relatively small. ➢ Light enclosure must be entirely self-contained. ➢ All system components together must be portable enough to fit through doors. ➢ Access to white board or magnetic board as a mounting surface ➢ Mounting surface should not obscure any image projected upon it. ➢ Light devices update to reflect changes in the simulation in less than one second . ➢ Light devices must be magnetically mountable. ➢ Relay interface components must be easy to connect. ➢ Software interface between the user and the light system must be easy to use and understand.

(Table 1)

1.5 Intended Users and Uses

This system is intended for use by members of the PowerCyber lab at Iowa State University to visually represent the status of relays in power grid simulations, though the system could be easily adapted to visually represent of the status of nodes in any graph-type system, such as city streets, wireless networks, or supply chains.

1.6 Assumptions and Limitations

Assumptions:

- Users can read and understand english.
- Users have their own computer(s) that can run our software.
- Users are in an environment with an internet connection.
- If outside of the Iowa State University campus, users will have access to the Iowa State University VPN.
- Users have access to electricity when needing to charge the wireless lights.
- The system will be used indoors and will not be exposed to the elements.
- The system will be kept dry at all times.
- The colors for the following statuses will be: blue for functional, yellow for <intermediate>, and red for non-functional or non-responsive.

Limitations:

- System is capable of two way communication.
- System must be chargeable and usable at the same time
- System in its entirety must fit through a standard door.

1.7 Expected End Product and Deliverables

The primary deliverables for this project are the wireless LED lights, the software that runs on the user's computer which is used to configure the system, and the Zigbee coordinator that acts as the intermediary between the software and the wireless lights. We will also include a way to charge multiple lights at the same time, in the form of a charger hub.

2 Early Project Plan

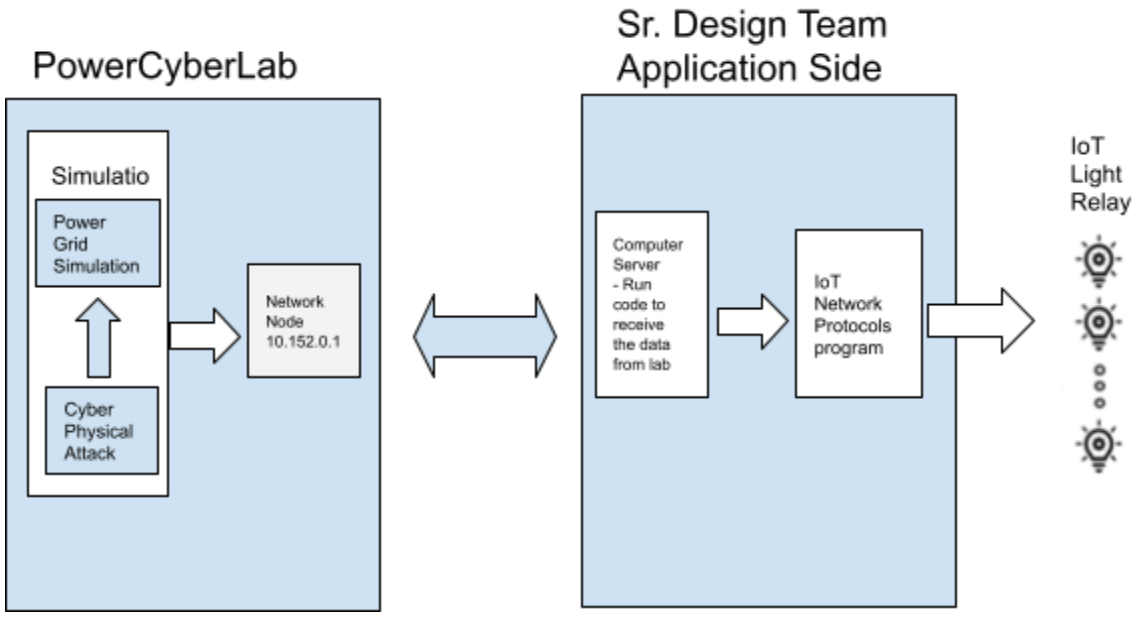
2.1 Task Decomposition

The technical parts will be divided into two parts: hardware and software. Hardware components are PCB board design, power charger system, and circuits design. The software components are GUI design, IoT network protocols, and software control units. Both hardware and software are subjective to update if there is a more efficient or useful case to come up.

The plan and design phase is divided into three parts. Because our project is adopted from the senior design team 19-16, our first part of the senior design is to understand the previous design and implementation. Once we comprehend the previous team's accomplishments, our second task is to

iterate our own design upon the previous or even rebuild it more efficiently, securely, and better in performance. Lastly, after we finalize our prototype implementation plan for the design, we will need approval from both the client and advisor. This will require the team to revise our design and ensure it meets our expected standards.

Each phase will have deeper tasks intertwined within themselves.



(Figure 1: System diagram)

2.2 Risks And Risk Management/Mitigation

In general, we have decided there are three levels of risks we may have in the way of completing the project. Critical level risk means a red flag, all members and the advisor/client should be notified at the first moment. An immediate meeting between the team and the advisor need be scheduled as soon as possible to discuss mitigation strategy. Moderate level risk means a warning sign, all group members should schedule a meeting to discuss the issue and give a mitigation plan for the risk. Depending on how the relativity of the issue and flexibility of the advisor, the advisor may not need to join the meeting. Small fix means some level of attention is needed. This isn't an urgent issue or something off the track. Team members can help each other on this level of risk, able to resolve the risk within the team, and no need to raise the advisor's attention. Below are some examples of each level of risk.

- Critical: Project/Mission fail due to design issue, Milestone overdue, final deliverable overdue, financial resources shortage, personnel out of duty or injury, and a new pandemic.
- Moderate: Weekly plan off track, significant technical issue.
- Small fix: Excused absence of weekly meeting etc.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones

To guarantee that our team's project satisfies all requirements, we keep track of our progress against the schedule, keep the advisor and client updated on progress, and revise and reflect our work. Since the spring 2021 is focused on the design and plan of the project.

1. Project Research
   - 1.1. Review the previous team's documents. (Design Docs and report, Presentation, and virtual tour of PowerCyber Lab by the advisor)
   - 1.2. Research alternative technologies. (Network Protocols: LoRa, ZWave, etc)
   - 1.3. Brainstorm possible changes and/or upgrades to the previous team's work.
2. Develop a project timeline
   - 2.1. Add due dates for deliverable documents for the first semester.
   - 2.2. Add tentative dates for development during the second semester.
   - 2.3. Build Gantt chart for scheduling and project management.
3. Design hardware and software model
   - 3.1. Design hardware, such as the PCB of the light devices.
     - *3.1.1. Simulate power grid attacks on IoT light devices.*
     - *3.1.2. Better Battery life or/and faster charging.*
   - 3.2. Design an application with a GUI to configure the IoT light devices
     - *3.2.1. Create an easy-to-navigate interface for User.*
     - *3.2.2. Interact the IoT light via GUI and user side applications.*
     - *3.2.3. Show the visibility of system status.*
   - 3.3. Design embedded software for IoT light devices.
     - *3.3.1. Better scalability and fault tolerance.*
4. Develop software and hardware components
   - 4.1. Develop the physical IoT light devices.
   - 4.2. Develop the application to configure light devices.
   - 4.3. Develop embedded software for the light devices.
5. Testing
   - 5.1. Test hardware devices
     - 5.1.1. *Xbee coordinators and secondary nodes are functional. All IoT smart lights have correct output from PowerCyber Lab's simulation and two-way communication is set up successfully. Charging circuits and updated batteries can extend the work life of the IoT smart lights.*
   - 5.2. Test software
     - 5.2.1. *GUI deployed successfully and can interrupt with the IoT smart lights. User side application can handle communication and files correctly.*
   - 5.3. Test system interoperability
     - 5.3.1. *Test the whole system to ensure all functionalities are running smoothly.*

In order for this project to be successful, we are aiming for a hardware proof of concept prototype by the end of the semester. This will be done using the components we intend to use in the final project, but performed on breadboards such that we can work out hardware and software issues before fabricating a pcb. Because of the lead time involved with PCB printing and prototyping,

aiming for a functional prototype before the end of the semester gives the team more time to identify potential problems in the development process.

Metrics

Due to the limitation of time and the COVID-19, the team wasn't able to play with the existing software and hardware by the time of this first version of the design document. Numeric specification of metrics and evaluation criteria may be added in the next version of this document. The metrics are measured in four main aspects: power, GUI, and PCB board. Most metrics are integrated with the milestone above. Here is the list of metrics with quantity below.

Power:

1. Easier charging, meaning that the end user wouldn't have to do as much to charge the devices
2. Battery life, meaning that with maximum power draw for the devices would still fall within the required 8 hours of usage.

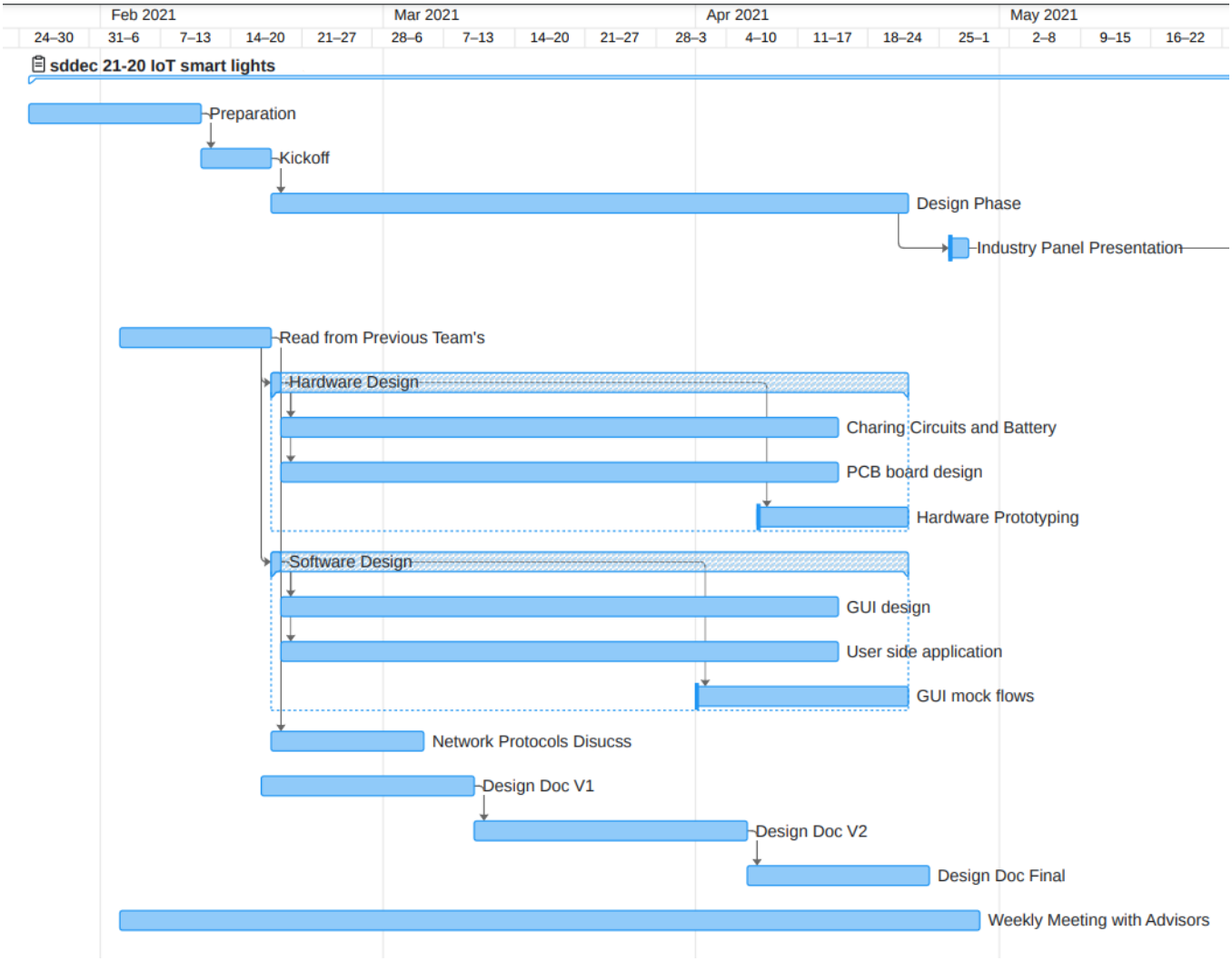
GUI

1. UI/UX easy to use and make significant updates.
2. More interactive and better visibility.

PCB Board

1. Efficiency
2. Newer and more compacted design

2.4 Project Timeline/Schedule.

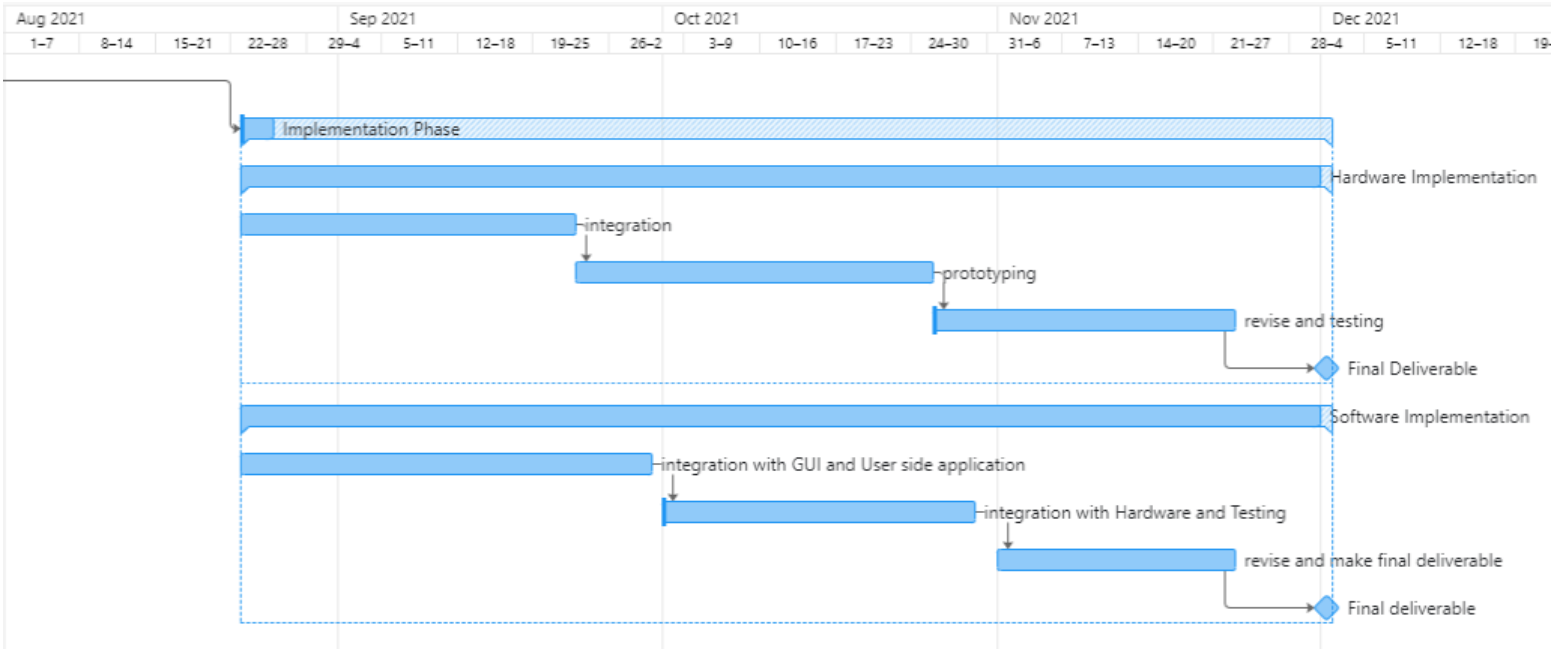


(Figure 2: Gantt chart timeline of Spring semester/ design phase)

The Gantt chart above shows the timeline we plan to follow for this project. This chart identifies all the designing, building and testing/revising that need to be done for the project. The chart demonstrates how the system planning will happen as we plan in this first version of the design document.

This is because the timeline is designed with teamwork in mind. The tasks are separated, so that one or two people can complete the tasks reasonably within a given time frame, dedicating 5 or more hours a week. This seems to be the most effective way to break down tasks, because it would be unfair if a task or series of tasks requires more hours of investment per week than other tasks.

The following Gantt chart shows the Fall schedule of our senior project. Implementation is divided into two parts: hardware and software. After each part is implemented as expected, we will integrate both sides and start to test them. Final deliverable will be sent to our client by the end of the fall semester.



(Figure 3: Gantt chart timeline of the Fall semester/implementation phase)

2.5 Project Tracking Procedures

To track our team's progress, one helpful tool we utilize is the Trello board. We have our all team members and the advisor added to the Trello board. In the Trello board, we set up sections such as development modules, To Do list, Done list, meeting notes, resources, and archives. Each section keeps track of one aspect of the project. Once we move to the implementation phase, we will add a specific task section to track the progress. Overall, by using the Trello board, it minimizes miscommunication around the project with simple tools to organize tasks and track progress.

Github will be set up and configured to store the software coding. The Github issues and milestone is also a good tool to track our software development progress.

Another tool we utilize is discord. Discord is mainly for our day-to-day communication tool. We can easily set up text and voice channels. This can supplement the Trello board because the Trello board doesn't support real time communication. Any notification and risk happened, team members can set out a text in the discord channel and other team members can be notified immediately. This helps all group members stay tuned.

2.6 Personnel Effort Requirements

Group Member (first name)	Technical Role	Efforts in term of hours per week
Hamza	Hardware Developer 1. PCB board design 2. Charging circuits	• 2 hours team meeting, twice a week.(The frequency and length of meetings may increase when project load increases. • 1 hour meeting with our client and advisor • 5 hours of effort in design, plan, and develop the IoT smart light project. (The load may be lighter in the first week or heavier when we face some technical issue.)
William	Hardware Developer 1. Hardware Prototyping 2. Charging circuitry prototyping 3. Embedded software	
Xinlei	Software Developer 1. Web-based GUI 2. Python application on user side	
Nathan	Software Developer 1. Web-based GUI 2. Python application on user side	

(Table 2: Group member roles and expected efforts)

2.7 Other Resource Requirements

As far as we have decided in our project, there isn't any other resource requirement needed other than purchase additional hardware parts or software license. Although we may add items once we start implementing the project, most parts can be collected from the previous team and existing resources of PowerCyberLab.

2.8 Financial Requirements

From the project proposal, there is a financial budget for this project. If we find we need any more resources to implement the project, we will propose it to our advisor and client and revise this section. The department/board will decide whether we have any financial resources.

Item	Quantity	Price per Item	Total cost
XBee s2c prog. module	101	$29.00	$2,929.00
50x WS2812B LED	2	$25.00	$50.00
Li-Polymer 803860 Battery	100	$12.50	$1250.00
Miscellaneous costs	-	-	$200

Total projected cost: $4,429.00

3 Design

3.1 Previous Work And Literature

Our project is an enhancement on previous team work (https://sddec19-16.sd.ece.iastate.edu). Our purpose is making the design more usable by adding more features and enhancing communication protocol and battery time. The previous team was able to set up a one way communication using Zigbee protocol and Xbee chips. They designed the PCB and the housing of the Smart light and was successfully mountable on a white board using magnets.

The current product's battery can only last for 8 hours. Another large drawback for the hardware is it doesn't fit well inside the enclosure, which gets in the way of when a user needs to charge the device. The device has a physical switch that needs to be toggled to switch from charge and discharge modes, making usability quite difficult. The device also only has one RGB light for presenting data, with very little variance in color selection. One final issue is the lack of user input. Users are unable to interact with the active simulation represented with the lights, which calls for two way communication back to the simulation. This is not to understate the fact that the previous team successfully created a device which could functionally represent the state of the simulation, rather to outline some potential targets our team is looking for improvement on the project.

(Figure 4: Previous team's light module)

3.2 Design Thinking

Design is currently using the Zigbee communication protocol for simulating processes running in different locations. In our design thinking, group members suggested using different protocols, i.e LoRa and Z-Wave. Another proposed idea was using bluetooth instead of IoT. We found ZigBee however to be the best protocol to be used as it consumes the least amount of energy and allows a wider range of devices to be connected to and communicate with.

	Owner	Frequency (MHz)	Range	Power requirement	Security	Compatibility
Zigbee	Zigbee Alliance	868 - 868.6 (Europe) 902 - 928 (US)	10–100 meters line-of-sight	Low-Power, Potential Batteryless	Low, basic encryption	Compatible across Zigbee devices. DotDot OS.
Lo-RaWan	LoRa Alliance	169, 433, 868 (Europe) 915 (US)	Up to 6.2 miles or 10 km.	Low-Power	Basic 64-128 bit encryption	Depends on OEM
LTE-M	GSMA - Cellular Carriers	LTE Bands: 450-2350 (uplink)	Global	Band dependant	NSA AES-256	Application dependant
IEEE 802.11af (White-Fi)	Open - IEEE Certified	470 - 710 (Digital Dividend)	Short, up to 100m	Low	WPA	Application dependant
IEEE 802.11ah (HaLow)	Open - IEEE Certified	850 (Europe) 900 (US) 700 (China)	Up to 13 miles or 20 km.	Medium	WPA	Application dependant

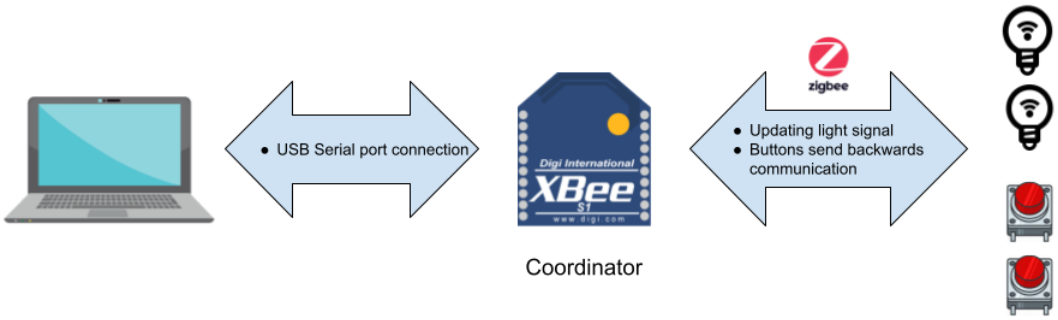
(Table 3: Wireless communication protocol comparisons)

Our team also came up with the idea of using smart batteries that have an inside charging circuit. Another approach was designing our own designing circuit and using another battery with more capacity. The team decided to go with ther second option; one of the limitations of our project is size, and therefore we will be using a small battery and reorganize our components on the PCB in order to charge our battery. Especially in the context where smart lights are transmitting information rather than just receiving information power usage can be a lot higher than in the previous cases, as such power consumption must be considered more than in the previous context.

3.3 Proposed Design

When approaching this project, we divided our requirements into two categories, the core requirements of which we're aiming for, and the non-functional requirements of which we must satisfy. Each of these requirements was used to help arrive towards our prototype platform conclusion.

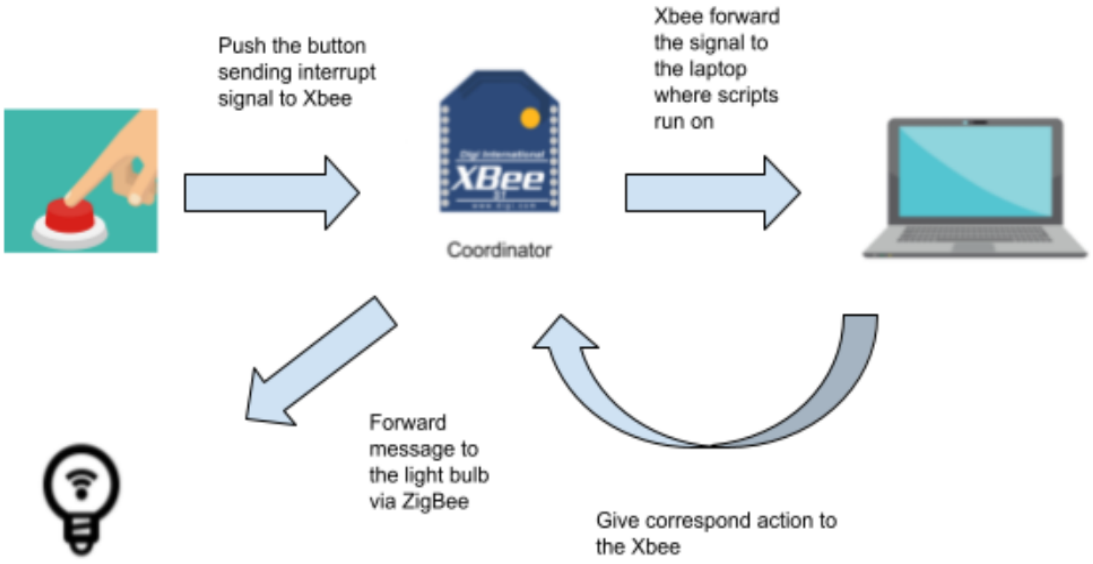
From these requirements we decided to move forward with Xbee 2 as the platform from which we develop our smart lights. The Xbee 2 is capable of several types of network topologies which allow for network scalability and flexibility for the number of devices during a PowerCyber simulation. The Xbee 2 also features an implementation of Zigbee, which can allow low power wireless communication. For now, the smart lights are broken up into singular nodes, and a coordinator connected to a host PC is used to handle the inputs and outputs of the smart lights and interface with the simulation.



(Figure 5: System diagram of IoT devices and host machine)

Coordinator

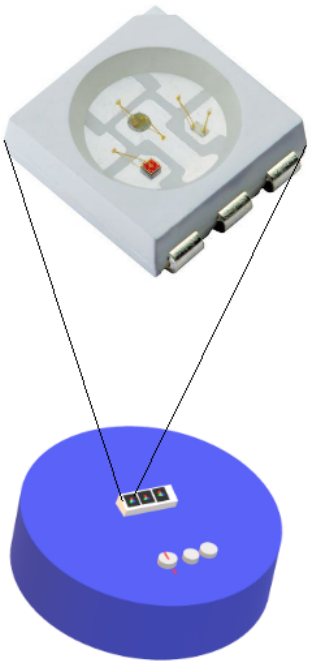
The Coordinator is responsible for receiving and sending information to the host simulation consisting of status updates from the smart lights that would be triggered by an end user, as well as the current status of the relay that any given smart light may represent during the simulation. This Coordinator communicates with the host PC software via usb serial, and communicates with the xbee 2 based nodes via the Zigbee wireless protocol.



(Figure 6: Coordinator)

Smart Lights

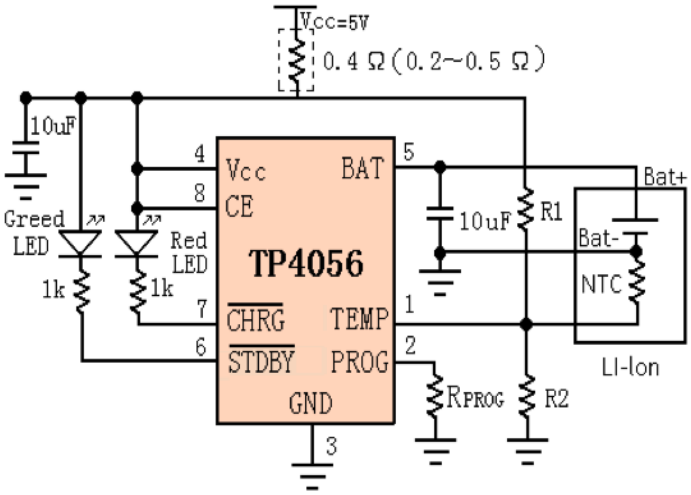
The smart lights work very similar to the coordinator, and are also based on the XBee 2 platform. Each node, powered by a Digi Xbee 2, has access to micropython, a C based python implementation that can run embedded code on the hardware. One key advantage from this is micropython access to the neopixel library. Because of the need to represent the states of the relays via RGB lights, we can take advantage of the neopixel library with ws2812b RGB lights (figure 7). Each light is individually addressable and serialized, meaning that chaining rgb lights this way is much easier than the previous implementation. In addition to this, we're also aiming to improve the battery charging circuitry. In its current state the charging circuitry offers charging only after the toggling of an onboard switch, meaning that charging requires hardware teardown for PCB access, and gates the user from using the node during this time. Using a sophisticated hardware design that allows for concurrent usage is ideal, and several transistors as well as a TP4056 IC are able to be implemented to satisfy this requirement. Addition of hardware buttons and a dial are also needed for the end users to simulate a "hack" on the given relay, this requires a dial to select what type of attack is to be simulated, as well as a button to actually trigger the update to the back-end server.



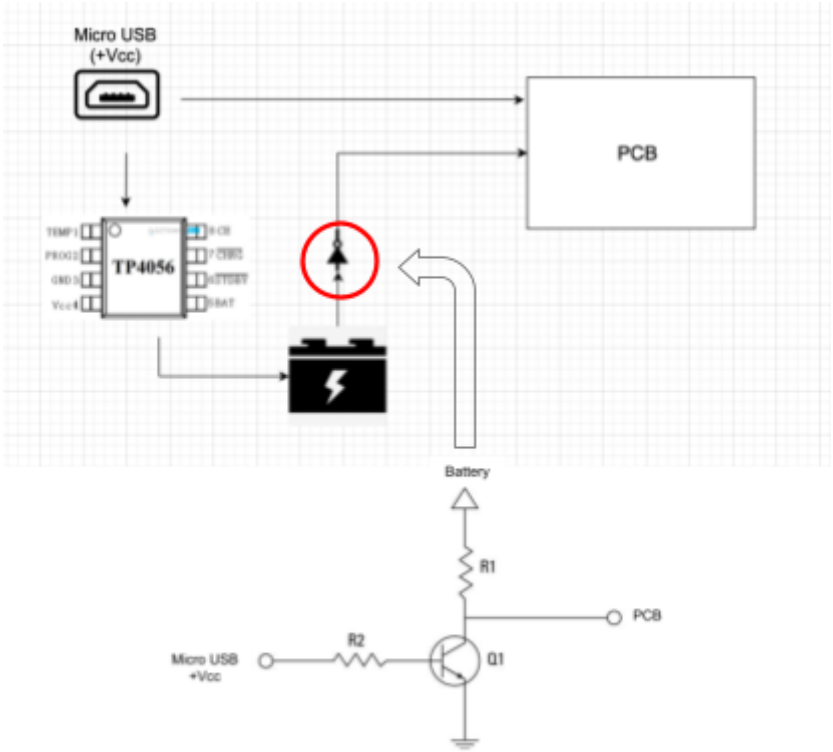
(Figure 7: Ws2812b RGB Light module)

Charging circuit

We will be designing our charging circuit using the TP4056 IC (figure 8). The IC will help both charge the battery and indicate if battery charge is full. We will also implement an automatic alternating design as shown in figure 8 using a not gate in order to alternate the power source between battery and USB while the battery is charging.



(Figure 8: circuit diagram)

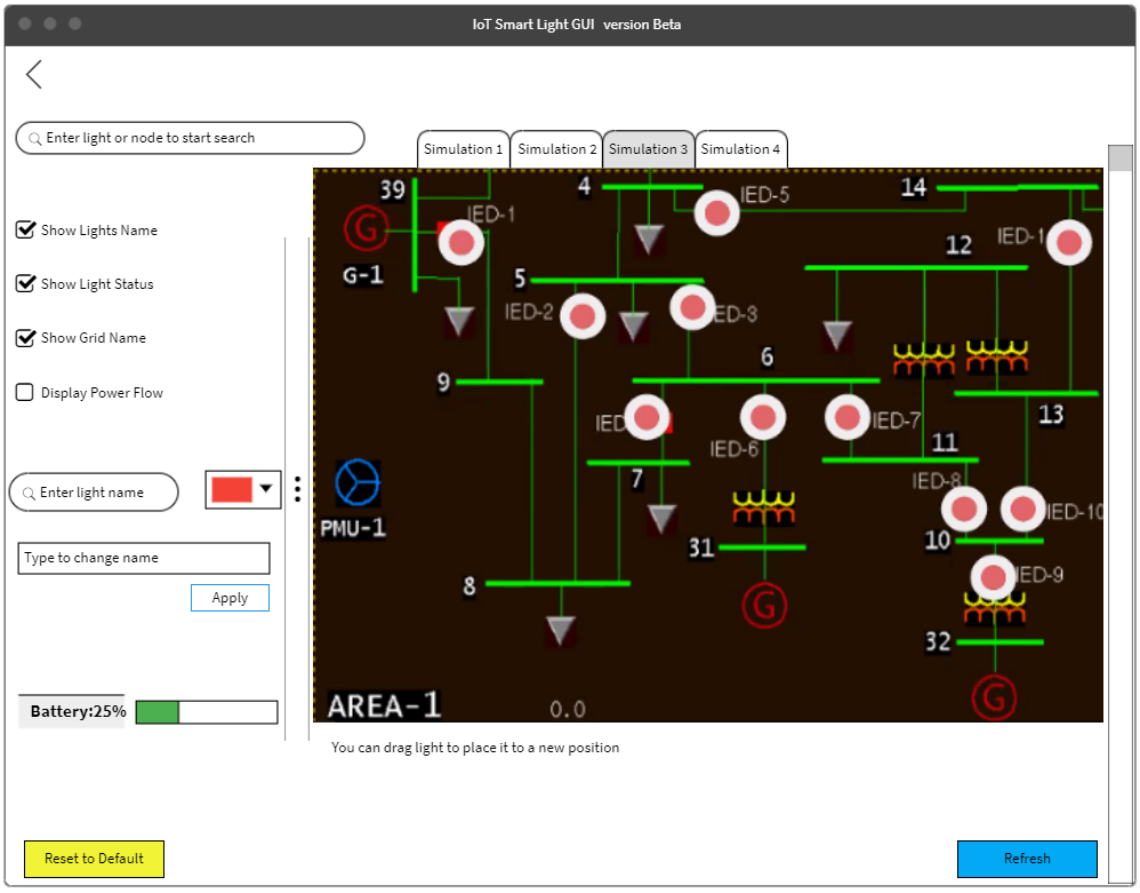


(Figure 9: charging circuit diagram)

Wireless Charging

In relation to the initial charging circuit we also discussed the addition of wireless charging, which would simplify the charging using both an inductive charging station and the addition of an inductive charging circuit on the smart lights. This would work on top of the previous circuit. The largest issue here would be the voltage requirements needed by the microcontroller and the rest of the devices. As it stands the battery requires 4.2 V to charge, which puts any 3.3V Inductive charging set out of bounds. The next solution would be using a 5V circuit with a power step down module. This addition was added later on in the design phase, which can pose some issues but research will be done prior to the implementation to determine if this is feasible. This would require some hardware added to the base of each smart light as well as a basic table fitted with the correct inductive charging circuit and a power supply.

Graphical User Interface



(Figure 10: Smart Light GUI diagram

Our GUI will allow the user to configure the IoT light devices individually and connect them to the simulation. The user will be able to place the light devices onto the node or object that they wish the light to represent. The user can also toggle certain UI elements, such as the light names, to clean up the view of the simulation from the application side. This GUI is not intended to be visible to the audience of the presentation.

3.4 Technology Considerations

The proposed design will be able to simulate a running process in a different location using multiple smart lights. The design still lacks the ability to run for longer than 8 hours, however with adding the charging and using at the same time feature, we will be able to overcome this issue. Some trade-off has to be made as the PCB is very limited in size as per the client's requirements, and therefore limited amount of features can be added. Such an issue will be overcome by getting more creative with different ways to present data using only an LCD screen or multiple RGB lights. The design will have the ability to perform two way communication which will allow us to add more features.

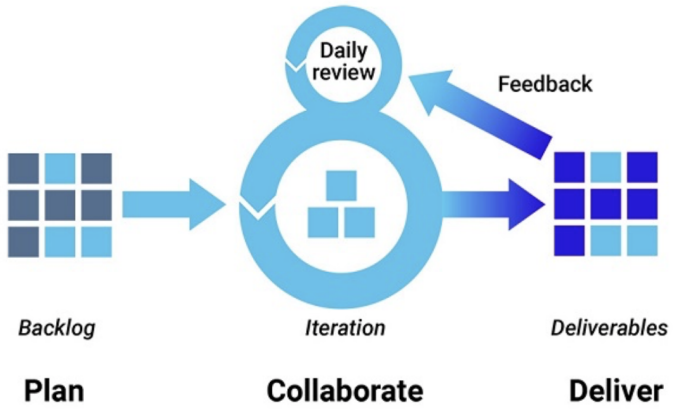
3.5 Design Analysis

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.

When looking at the previous work we decided to try to iterate on the design, as we agreed that using a lot of the same core hardware would yield better results with some slight changes to the design. As a team we agreed that some improvements could be made, especially with the previously stated issues regarding the lack of two way communication between the IOT smart lights and the coordinator. This requires additional buttons and a dial. Another aspect is more configurable rgb lights for the xbee modules, through the usage of ws2812bs, which are a bright module that is capable of 255^3 colors, a significant improvement from the static nature of the last version.

3.6 Development Process

Our team is following the Agile development process. We are using this process as our project needs to always have more features added to it, and therefore we will be planning which features we will be working on, work and collaborate on that feature till delivering it, and then back to considering new features.



(Figure 11: Agile development process)

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

The final design will be composed of 3 modules: Zigbee Coordinator, individual light modules, and a software user interface.

Functionally, the Zigbee Coordinator will assign each smart light to the relay it is representing. The end user will be able to assign each node to a specific relay on the Power Cyber simulation and interact with it during its lifecycle. Buttons on the Smart lights allow tactile user interaction during the simulation and live feedback.

4 Testing

Testing is an extremely important component of most projects, whether it involves a circuit, a process, or software.

The project will be divided into phases, such that each phase will have time to debug and catch any large time consuming errors that could hold the team back. Because of the nature of IOT there may be hardware design revisions. Getting some level of debugged hardware prototype functioning is the first step for testing on all levels.

Each phase of testing will be depending on what modules of software are complete, as well as what revision of hardware is currently functioning. This iterative process will allow changes to be made to hardware to prevent lockups in the development process.

4.1 Unit Testing

Tests are broken down into units to emphasize modular testing, this way the team can debug stages one level at a time to prevent confusion. This can allow software to be broken up for easy testing

Each module will be broken down and tested individually before combining into more complicated sets. Between each unit test there will be interface tests to ensure that two quantities are communicating correctly.

After a PCB Prototype has been fabricated, it is important to ensure that the printed and soldered prototypes are matching the circuit diagram completely, this will require testing each connection and component to the parts specifications and verify the design to the original specification.

4.1.1 Smart Light Modules

Each revision of the module will need the following verified:

The module can Integer data to the host regarding its updated state based on a button press. This will send a packet to the server via zigbee. Testing is needed to ensure that ranges are clear and packet loss is minimized.

The module can receive data from the host based on its changing state from the backend simulation.

The module can correctly cycle through all the colors from each previous color. For example when the module is currently red, it can then change the color to any other color.

In the early phases this will be done on a breadboard to make sure that hardware is both functional as well as modular for rapid design changes. Testing the functionality will need to be

repeated for each iteration of the PCB. At this point an artificial system will need to be in place to simulate the communication between the IOT smart light and the back end server.

Early stages of testing will involve this module as well as the Coordinator Module to test wireless communication and scalability.

4.1.2 Coordinator Module

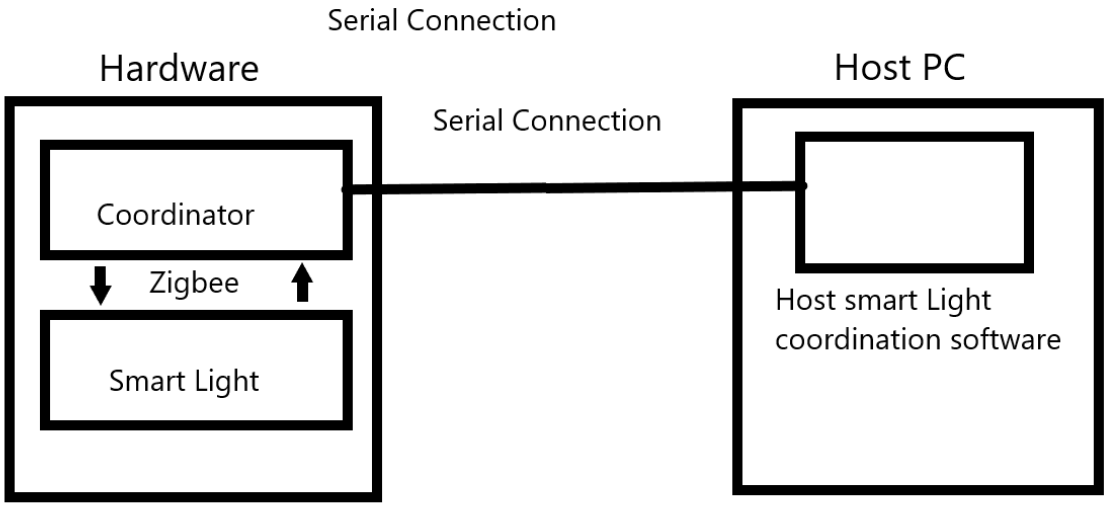
The broadcaster module designates the iot smart light modules to reflect the simulation that takes place within the PowerCyber Lab. In order to prove that the module works when a signal is received, a simulated signal will be sent to show that the backend updates based on the input. Ensuring that the module can trigger changes in the backend is the foundation of the two way communication needed for a successful deliverable. Simulated functions used to make it "appear" that a signal is received or sent will be used to mock test this functionality.

This Module will be connected to a host PC, and will be sent information regarding the intended status of the Smart Light Modules over serial communication, and will communicate updates through the same interface back to the host PC.

4.2 Interface Testing

For each level of software development, testing protocol connections will ensure that communication between the broadcaster and the endpoint avoids any regressions. This can be done by effectively flooding the interface with packets to measure when the signal finally breaks down.

Testing will be done broken down into the following interfaces with each interface being predefined and understood by both ends of development.



(Figure 12: embedded hardware connectivity)

4.3 Overall System Testing

When other testing has finished, and it has been determined that the hardware, interfaces, and the software is functional, actual tests with the powerCyberLab will be needed to validate the overall functionality of the system.

This will be done with a full setup of the entire system, including the software and hardware all working together to reflect the specifications of the project. Each relay will be connected

4.4 Acceptance Testing

The best way to ensure that the end product is to the specification of both our client and advisor, both a functioning deliverable and demonstration are in order according to the requirements. This would be both to clear up usability and to teach the client how to use the system. This will also ensure that non-functional requirements are clearly completed to the clients satisfaction.

4.5 Results

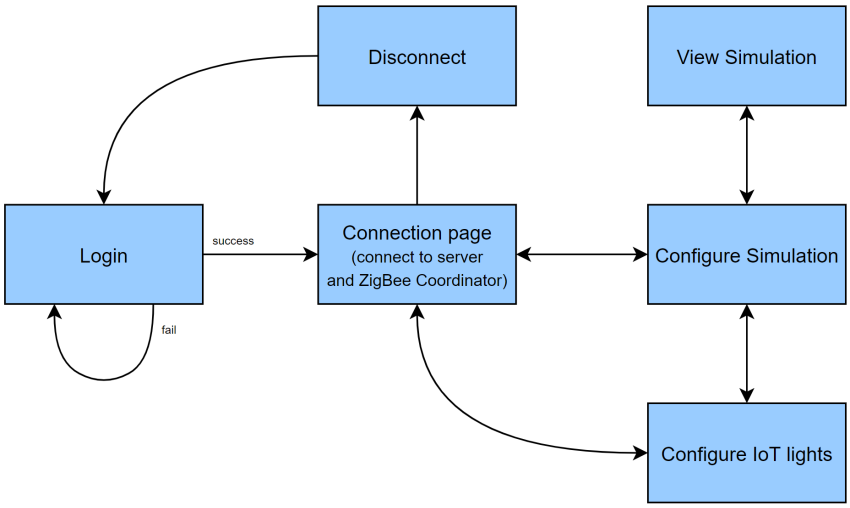
As we are still in the planning and design phase of the project we have not yet constructed any hardware components, nor built any software. We will have more results to show in the second semester of senior design.

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

Currently our team is prototyping basic breadboard circuits with mixed success. We're currently working on a prototype of the Iot smart light on breadboards but running into some hardware issues when it comes to actually writing to the devices.

As it stands the actual hardware that we have is the xbee models that aren't fitted with microcontrollers, for the time being the prototype is working with an arduino as the controller while we wait for hardware to arrive. This makes the configuration a bit more complicated because it requires more modules to implement but also works as a proof of concept.



(Figure 10: Software flow diagram. Each blue rectangle represents a page in the python application.)

6 Closing Material

6.1 Conclusion

In conclusion, our goal this semester is to create an Internet of Things (IoT) solution for the Power Cyber lab to solve the problem of inadequate presentation capabilities. We would like to use wireless, mountable IoT light devices. communicating using the ZigBee protocol, to represent the status of relays within the power grid simulations being run in the Power Cyber lab. The IoT lights will be configured and controlled by a python program running on a host computer.

We have chosen this solution over other possible solutions because of several factors. The first is that using mountable, wireless devices provides us with greater flexibility over wired devices. The wires would also get in the way of any presentations that the devices would be used for, thus reducing their effectiveness. The second is that our solution must be scalable, up to 100 devices, and managing that many cables would take up a lot of space and lead to unnecessary complexity. The wireless communication protocol we have chosen, ZigBee, is advertised as a low-power protocol, thus battery life can be extended. Lastly, we plan on implementing two-way communication between the IoT light devices and the power grid simulations so that users can interact with said simulations during a presentation, and demonstrate certain effects if desired.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

[1] Augustin, Aloÿs, et al. "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things." Sensors, vol. 16, no. 9, 9 Sept. 2016, doi:10.3390/s16091466.

[2]Digi, Setup Devices, 2018, [Online]. Available: https://www.digi.com/resources/documentation/digidocs/90001526/Default.htm#containers/cont_setup_devices.htm%3FTocPath%3DSet%2520up%2520%2520your%2520XBee%2520devices%7C_____0

[3]"LoRa Specification." *LoRa Alliance Resource*, Nov. 2020, lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf.

[4]"What Is Z-Wave Long Range and How Does It Differ from Z-Wave? - Z-Wave Alliance." *Z*, 17 Dec. 2020, z-wavealliance.org/what-is-z-wave-long-range-and-how-does-it-differ-from-z-wave/.

[5] XBEE® AND XBEE-PRO® ZIGBEE, Digi, 2016, [Online]. Available: https://cdn-shop.adafruit.com/product-files/967/p967b+datasheet.pdf

[6] XBee®/XBee-PRO S2C Zigbee®, Digi, January 2020, [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf

[7]"Zigbee." *Zigbee Alliance*, 9 Dec. 2019, zigbeealliance.org/solution/zigbee/

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.

For high resolution Gantt charts.pdf file, please visit
https://drive.google.com/file/d/1k6aQkD4UcFIDgFBK1CtHrsNlCSI92nlV/view?usp=sharing