Nosferatu

Capstone 1, Summer 2 2015

By: Connor Igo, Joe Gorse, Dan Cardin, Gunnar Boehm, Chris Holtsnider

Advisor: John Kimani





Table of Contents

Abstract 3

Introduction 3

Problem Formulation / Impact 4

Hardware Design 5

Software Design 7

Tasks 8

Conclusion 9

Appendices 10

References 11

# Abstract

Nosferatu is an energy efficient, convenient new way to control your home lighting. Connecting to your current home network Nosferatu will be able to automatically control the lights in your house. Bringing Wi-Fi and motion sensors to your light switches allows you to set schedules and rules for managing your home lighting, saving money, energy and time.

# Introduction

Nosferatu will be a direct replacement for your current lighting setup. In order to automate a given light switch in your home, that switch will be replaced with a Nosferatu. Each of these new “switches” will look similar to a normal light switch plate, except instead of a simply a switch sticking out, there will a few things. The most prominent will be a small dome shape which is the motion sensor. Then there will also be a button, which replaces the manual input that the original switch controlled as well as serving as a manual override to any automated commands.

Each switch, also has a Wi-Fi chip inside of it, allowing it to network to the other switches through a centralized hub. This hub will be responsible for communicating with all the switches in your network, as well as hosting a web service which is used to do any configuration of the system. Through any internet browser, any switch can be connected to, then controlled through a number of inputs.

Any given switch can be manually turned on and off, through the web service, which simply acts as a toggle. Next, each switch can be configured to turn on/off on a schedule, independent of any other switches. Finally, more complex rules can be configured to turn on/off a switch. Rules include input such as motion sensor data, time of day (such as sunrise and sunset), the status of other lights, and more.

# Problem Formulation / Impact

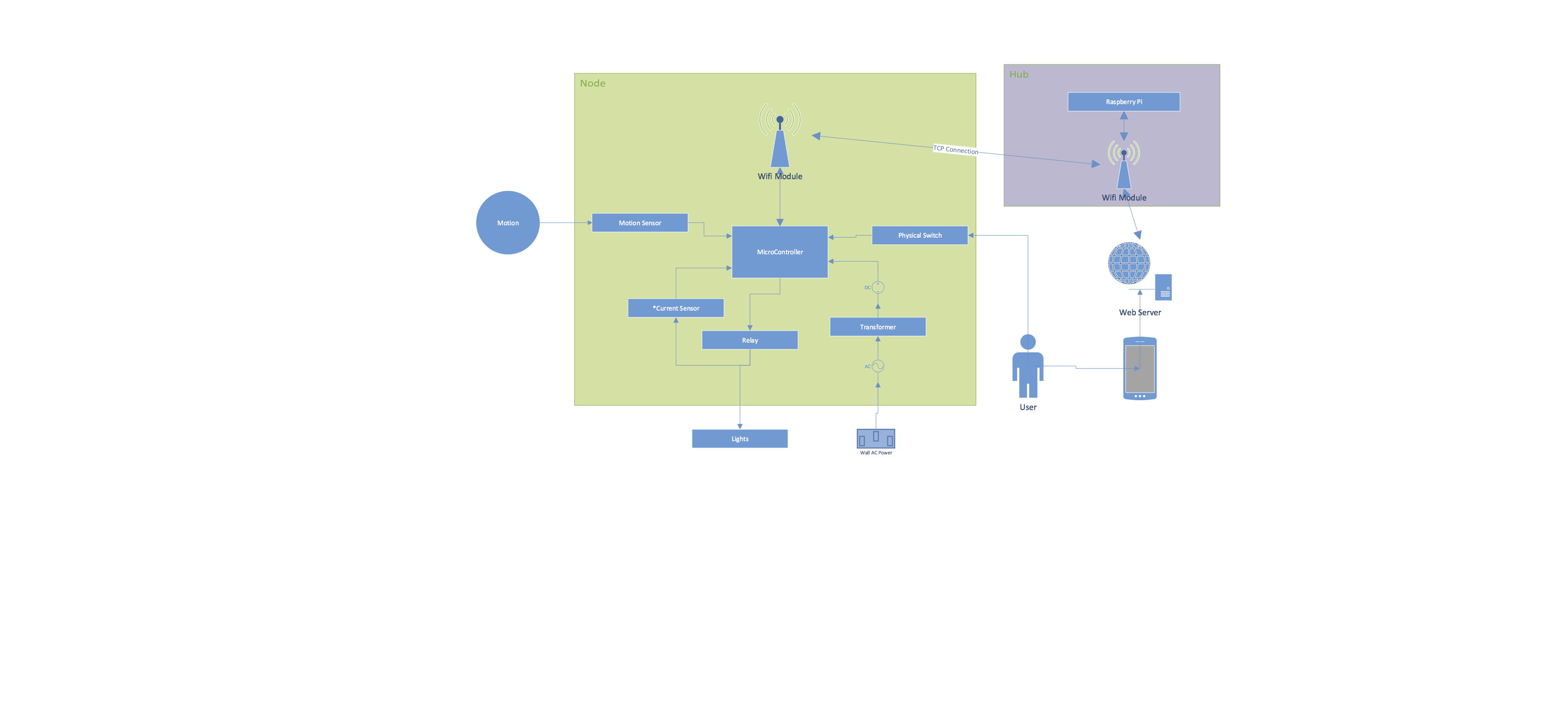
Currently there are very few existing consumer available solutions for automated lighting. Really, Wi-Fi enabled lightbulbs and the Belkin WeMo are the only real competition in the market. Even out of the market, i.e. patents, where there isn’t necessarily a product yet, there are only a few, and they tend to not directly impact the space that Nosferatu fills.

Wi-Fi enabled light bulbs like the LumenBulb (3) have been the most common to date. Their main drawback, however being that any time a bulb burns out, the whole light needs to be replaced. Given that each light costs many times more than a normal lightbulb, this can become expensive very quickly. This also become unnecessarily expensive when a single switch turns on more than a single light bulb, or actually impossible if the bulbs being controlled are more esoteric and not in a size where Wi-Fi light bulbs are sold. Even ignoring all of these issues, they tend to be sold singularly, so any attempt at controlling the lights as a whole system, rather than individually essentially requires a lot of manual setup.

For in wall solutions, the Belkin WeMo is the only real similar product on the market. However, the WeMo is also a large step backwards from Nosferatu. On the hardware side there is not much different, other than the WeMo’s lack of a motion sensor. Belkin ***does*** sell a motion sensor that is compatible the WeMo, however it is sold separately (and when paired with the WeMo, costs nearly twice as much as Nosferatu’s prototypes), and does not fit into the wall unit itself, and needs to be placed on something nearby. On the software side, while they do have the ability to control the unit on a schedule, there is much less the WeMo can do with rules; basically limiting it to time based automation. Finally, by default the WeMo is unable to be controlled through anything but their proprietary app. Third parties have found ways around this, and created self-hosted solutions, however Nosferatu is built with this, in mind so that out of the box, it can be interacted with through any device that can access the internet.

As far as patents go, there seemed to be only two patents that might have any overlap with Nosferatu’s functionality. The first (1), essentially outlines a method of controlling lights by manually inputting a pattern that the light should follow, and afterward it being capable of playing that pattern back. While a user might be able to replicate that functionality with Nosferatu, the methods that we use are neither through manual input like that, nor are they controlled through recording patterns. The second patent (2) details a much more generic method for an expandable home automation system, which is controlled through voice or screen based input terminals installed directly into the house, and controls various different types of devices. Again, while Nosferatu does control lights, and is not specifically limited to controlling lights alone, it does not require the use of input terminals in the house, nor does it use alternate methods of input such as voice control.

# Hardware Design



The key design aspect we took into consideration is the connectivity between our endpoints and server. There were three main considerations when choosing between connection methods; Bluetooth, Wi-Fi, and PLC (Power Line Communication). The advantages of Bluetooth, were that it lends itself well to being used as a very low power mesh network, with endpoints interconnecting between each other starting from a centralized hub. This could easily double or triple the effective range of the project, compared to Wi-Fi, however the use of a mesh network would require Bluetooth LE, a relatively new development. This would have required more expensive, and less compact components which would not have been ideal for the scope of the project. PLC, using the X10 protocol for communication would have been an interesting route. This modulates the frequency of the AC power directly in the walls of your house to communicate between nodes. The could have been an ideal solution, however PLC tends to be unreliable and susceptible to error, in addition to being much more expensive compared to Wi-Fi, or even the Bluetooth LE chips, due to needing to be directly connected to the AC current. Finally, Wi-Fi is a relatively simple setup, where each node must connect directly to a centralized hub. This can lead to a relatively small effective radius, and also can lead to network congestion. However, due to its cost Wi-Fi remains a strong contender. We ultimately decided to go with Wi-Fi because of the cost and the potential to implement a Wi-Fi mesh network which would negate any issues with range. Also due to the small amounts of data that will be being transferred over a single network, we found the network congestion issue to be a non-issue.

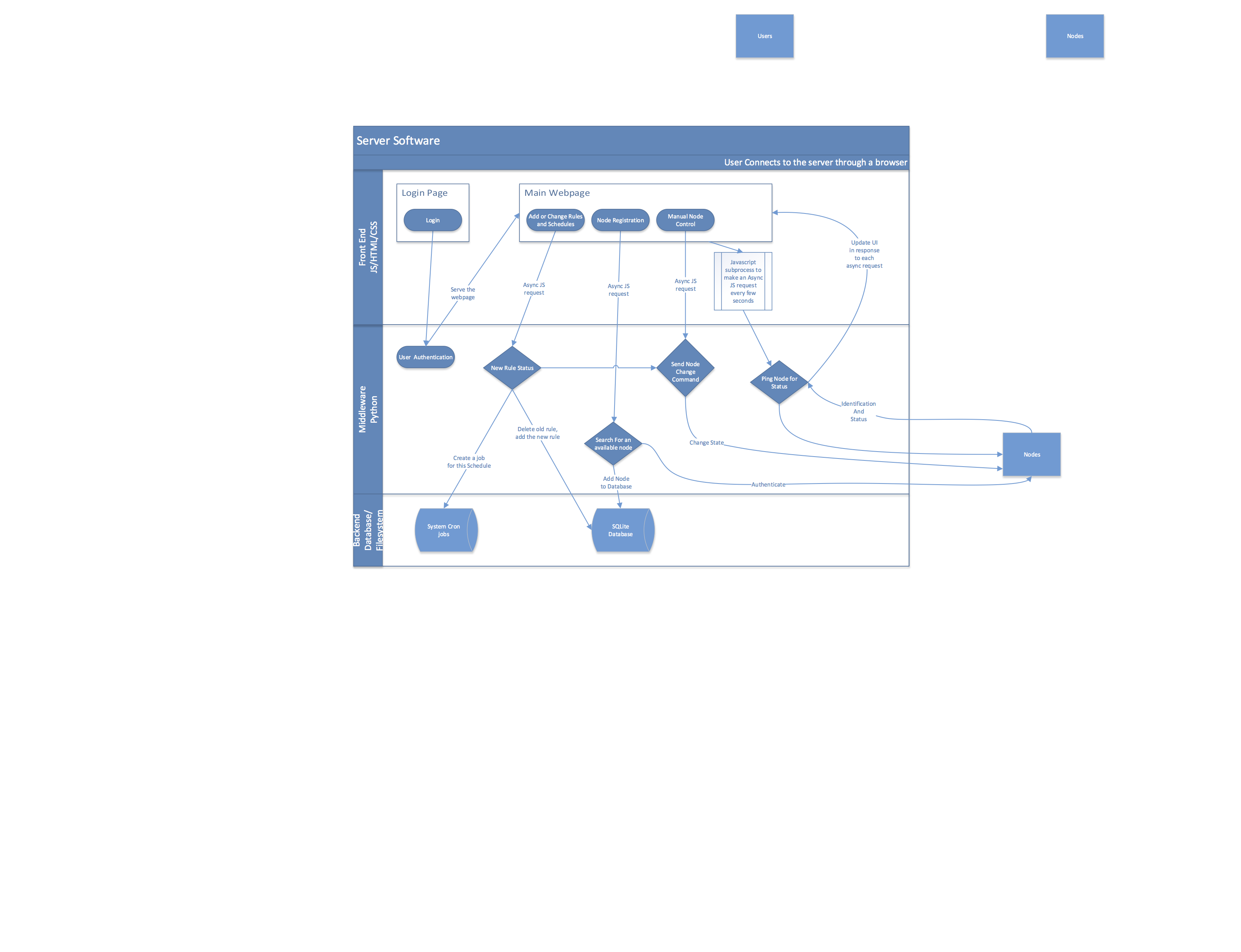
The next biggest issue was deciding how to power each endpoint. The easiest solution would be to use batteries to power the boards. However, requiring the user to replace batteries every so often was deemed unacceptable, given the premise behind the project being home **automation**. So rather than batteries, we decided to power the endpoints directly through the wall’s AC current. In putting a transformer in to our design, we limited the ability of our product to be used in older houses which might not have all the required wiring to allow for our design, and gave our unit a slightly higher power requirement. However, the WeMo (the only other solution on the market) came to the same conclusion that we did, the convenience of a permanent solution that works as long as your house has power, is ultimately the best choice.

We looked into a few different avenues for where and how to host our server. Our initial discussion was whether to host the server locally, on a small local hosted server in the user’s home, or to host remotely using a cloud service such as Amazon Web Services or Heroku. We decided that hosting the server locally would give us more flexibility in software design, by providing a readymade solution to the initial setup of the wall endpoints. Also starting with local hosting, and switching to a remote cloud hosted server later is a much easier task than the other way around. We then had to choose between both a Raspberry Pi and a Beagle Bone Black for the local server hardware. Comparing price, processing power, I/O connectivity, community development support, and our group members’ familiarity with each, we chose to use a second generation Raspberry Pi to host our web server and act as our hub.

Safety is the biggest concern we have as a general design aspect, due to adding a device that goes in to a wall and will be running off AC power, in close proximity to DC powered circuits. This had an impact in our choice of relay for the actual on/off control of the lights. The choice of relay was between an SPDT relay and a Solid State relay. SPDT relays use much less power, but use a mechanical, physically moving arm to turn the circuit on and off, leading to a much diminished lifespan compared to SSRs which have no moving parts, but use more power. The deciding factor, however, was that SPDT relays tend to fail “open”, which means if it does fail the circuit will tend to be broken and avoid an electrical fire, unlike a SSRs which often fail closed (because of the lack of moving parts) which would be more dangerous.

Finally, our power conversion method was the final real hardware decision and safety concern, given that we need to convert 120V 20A input into 5V 1A output. We decided to go with an existing consumer transformer that we will take apart and use for our needs because most sources tended to agree this was relatively safe. The only other real choice would have been to attempt to design our own transformer, or otherwise design a transformer-less system to convert high voltage AC input to DC. Neither of these were deemed to be particularly safe, so we went with the premade transformer. The largest ongoing consideration in this regard is heat. Given that we do not yet know how hot our components might get, there might be issues with the rated working temperatures for our components. Though each endpoint is expected to only be “active” for a very short amount of time, really the heat output is an unknown. If it does turn out to be a problem, we can put a small vent in the faceplate of our switch to disperse heat outside of the wall.

# Software Design



There are two main software blocks in our system; the firmware running on the endpoint, and the server running on our hub. The server will have three software layers; frontend (HTML, JavaScript), middleware (Python), and backend (database, Linux system calls). The node firmware will be written in C.

While the software for the nodes is almost entirely direct hardware interaction, the software for the server is mostly http requests. The general flow of data will start with a user logging into the web service. After being authenticated, they will see the main page of the web application.

Upon connection, the frontend will kick off an asynchronous background task on a timer to send requests to the nodes for status updates. This means the user will get a close to real-time view of what all of the nodes are doing. From there, a user can do any number of actions, such as controlling the lights, rules and schedules. Any given action a user can do will send off an asynchronous JavaScript HTTP request which either will make modifications to the database (or Cron job for schedules), or request updated data from the server to update the UI elements on the webpage.

For example, if a user wants to add a new rule, they'll fill out the information on the webpage, and submit it, which will send the request to the middleware running Python, which will then add the new rule to the database. If that rule happens to immediately update the state of the node, a request will also be sent to the node to change its state. Finally this will kick off a response from the node, which will communicate back to the server with its updated status.

The firmware running on our nodes will have three core functional tasks: identifying itself to register with a hub, changing the state of the relay to turn its light on/off, and responding to ping requests with the current status of the relay (and therefore the light).

# Tasks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part List** | **Cost Estimate Per Part** | **Quantity** | **Cost Estimate** | **Source** |
| Arduino | $10.00 | 4 | $40.00 | https://www.sparkfun.com/products/11114 |
| Motion Sensor | $10.00 | 4 | $40.00 | https://www.sparkfun.com/products/13285 |
| Wifi Module | $7.00 | 4 | $28.00 | https://www.sparkfun.com/products/13678 |
| Relay | $8.00 | 4 | $32.00 | https://www.sparkfun.com/products/11042 |
| Transformer | $7.00 | 4 | $28.00 | http://www.amazon.com/Genuine-Authentic-  Apple-Adapter-Charger/dp/B00QSB3UXE/ref=  sr\_1\_cc\_1?s=aps&ie=UTF8&qid=1437849720  &sr=1-1catcorr&keywords=apple+wall+charger |
| Micro SD Card | $13.00 | 1 | $13.00 | http://www.amazon.com/SanDisk-Memory-Ad  apter-SDSDQUAN-032G-G4A-Version/dp/B00M  55C0NS/ref=pd\_bxgy\_147\_text\_y |
| Button | $0.50 | 4 | $2.00 | https://www.sparkfun.com/products/9190 |
| LED | $0.35 | 4 | $1.40 | https://www.sparkfun.com/products/9650 |
| Rasberry Pi | $42.00 | 1 | $42.00 | http://www.amazon.com/Raspberry-Pi-Model  -Project-Board/dp/B00T2U7R7I |
| Wifi Source Module | $20.00 | 1 | $20.00 | http://www.amazon.com/gp/product/B00JDV  RCI0/ref=as\_li\_qf\_sp\_asin\_il\_tl?tag=htpcbeg-20&ie=UTF8&camp=1789&creative=9325&cre  ativeASIN=B00JDVRCI0&linkCode=as2&linkId  =2PT3YPZ25MGWJJ3P |
| Wall Adapter | $3.00 | 1 | $3.00 | http://www.amazon.com/Raspberry-Pi-Power  -Supply-5v/dp/B00LSEOTYK |
| Programming Board | $15.00 | 1 | $15.00 | https://www.sparkfun.com/products/9716 |
| 3D Printing Materials | $20.00 | 1 | $20.00 | http://library.northeastern.edu/services/3d-pri  nting-studio |
| **Totals** |  |  | **$284.40** |  |

# Conclusion

Nosferatu will enable the average homeowner to automate their home lighting without worrying about replacing all of their light bulbs or a long, arduous setup. Ideally we would like to expand this idea to other home automation uses that could utilize the same setup, such as power outlets and home security. We would also like to expand our interface to include a mobile application to make interfacing even easier when away from a computer. With an open API, Nosferatu could also be used to hook into other personal and home applications, even other home automation products such as Nest. With more software on the node end we could remove the need for a central hub to connect the nodes and host the server, and instead just use the current home wireless setup. Given the expandability of Nosferatu, its usefulness should be able to outclass any existing competitors in home lighting and automation.

# References

Patents:

1. <https://www.google.com/patents/US6531836>

2. <https://www.google.com/patents/US5086385>

LumenBulb:

3. <http://www.lumenbulb.net/>

WeMo:

4. <http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/>