Not exactly the Internet of Things for Outdoor Lighting

Winter Term Progress Report

Beta Stage

Oregon State University CS Senior Capstone Group 22

Malcolm Diller, Sean Rettig, Evan Steele

Client: Victor Hsu

Abstract

The goal of this project is to create a home lighting automation system from open source software and inexpensive commodity hardware that allows lights to be controlled automatically and remotely while also being easy to set up and use.

The system consists of a wireless network of tiny "client" computers that each control up to 4 sets of lights and are controlled by a central "server" computer, which automatically sends out commands to the clients when it's time to turn on or off. The central node runs a control program that can be easily accessed via a touch screen, a web browser, or a mobile device, where the user can locally or remotely control each light individually. The control interface allows users to easily set "rules" for what their lights do and when, depending on the time of day, the sun/moon position, and potentially even triggers such as weather conditions or calendar dates.

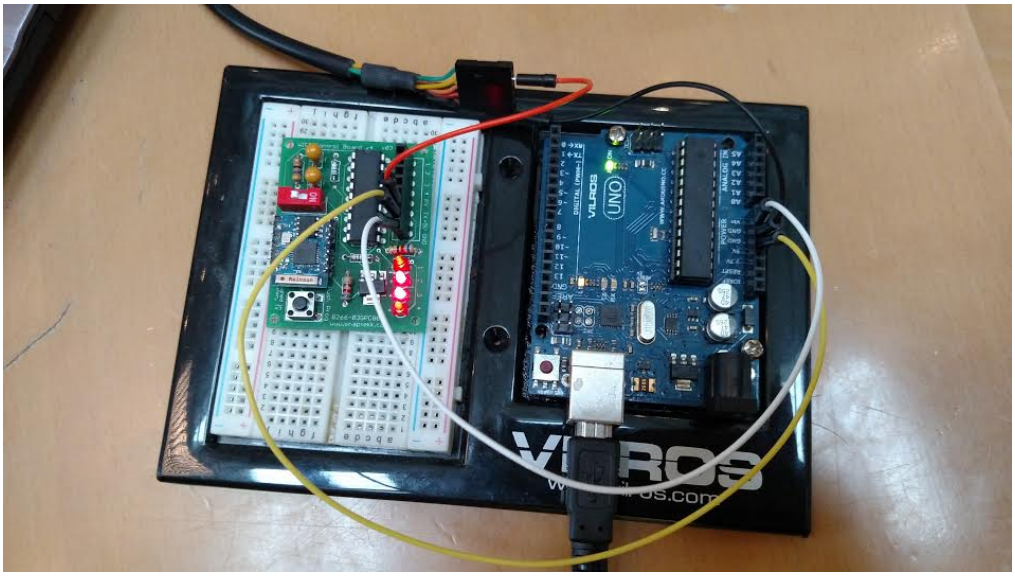
Throughout the last two terms, our team has produced initial designs, documentation, and an alpha-level deliverable for the project and now a beta version, which contains all the core functionality for our final release design next term. With regular communication through email and IRC, our group kept coordinated throughout the project terms, publishing progress reports to the Sharepoint site and code samples to our Github page. We will discuss our progress so far and describe how the prototype and documentation unfolded over the course of the first two terms of the capstone project. This should provide an assessment on where our groups stands in preparedness to move forward with increased functionality in preparation for the final submission and expo presentation.

CONTENTS

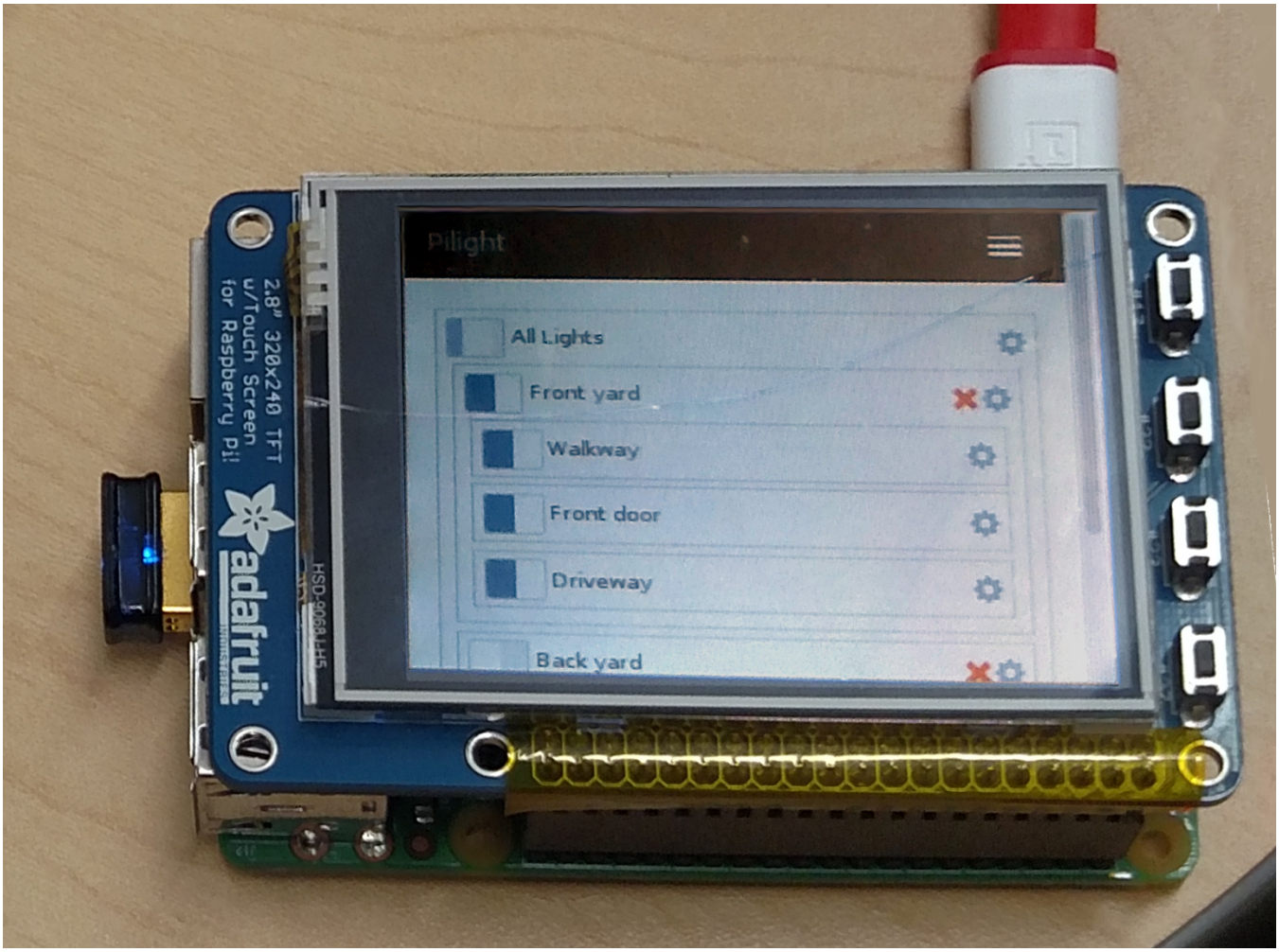
I. Status

With the alpha release, most of our functionality was already beta-level, but a few core components were missing. We caught and patched some bugs from the alpha release and implemented the few remaining features. Testing revealed some bugs, which we classified as high, medium, or low priority on Github.

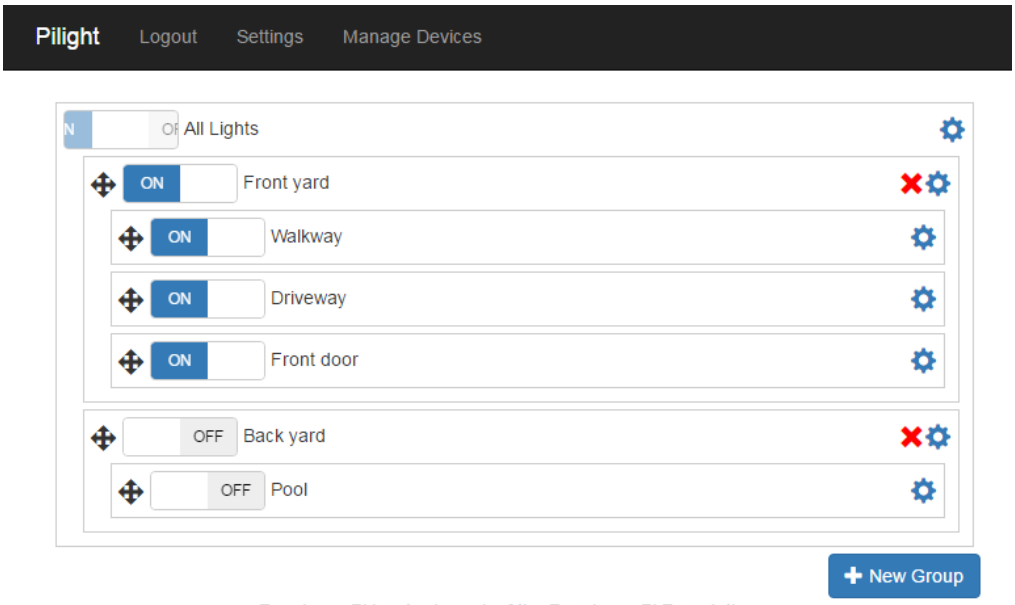
- The ESP firmware has been written and can automatically connect to the Pi wirelessly.



- The Pi is loaded with our custom build of Yocto Linux and can act as a wireless router for the ESPs to connect to.
- The Pi is set up to run our Flask web site, and it can be connected to by wireless devices and viewed in a browser.



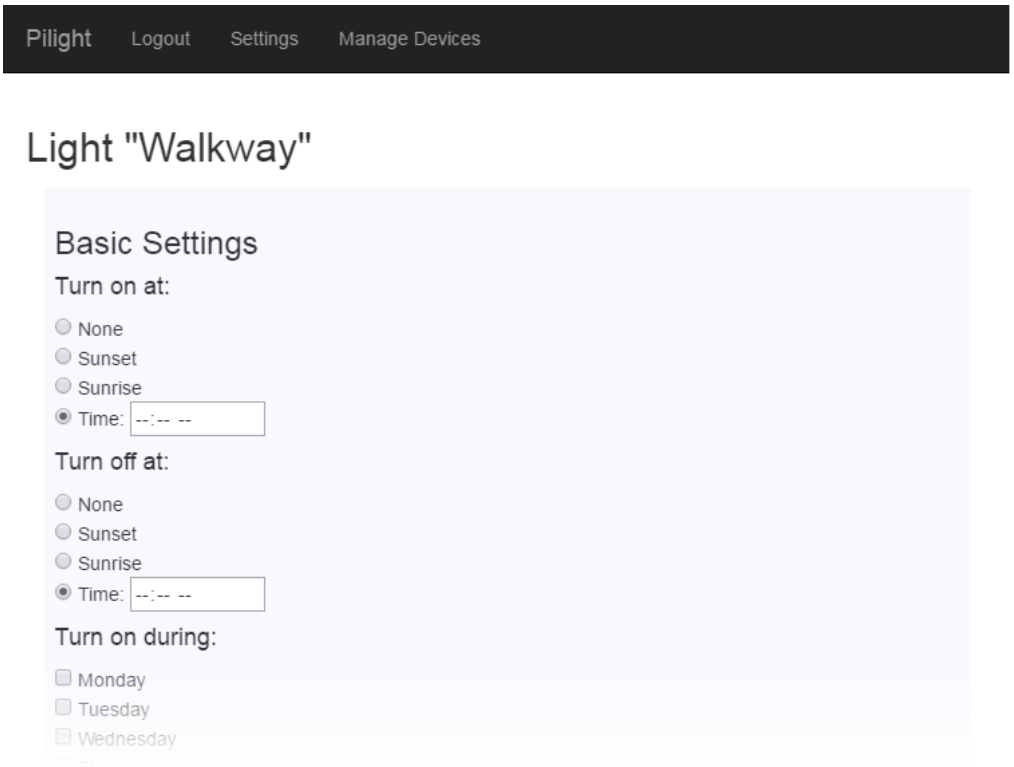
- The Flask web site has a SQL database that stores information about the lights, groups, devices, rules, and users.
- The web interface has a main page that lists all available lights, where lights can be renamed and sorted into groups.



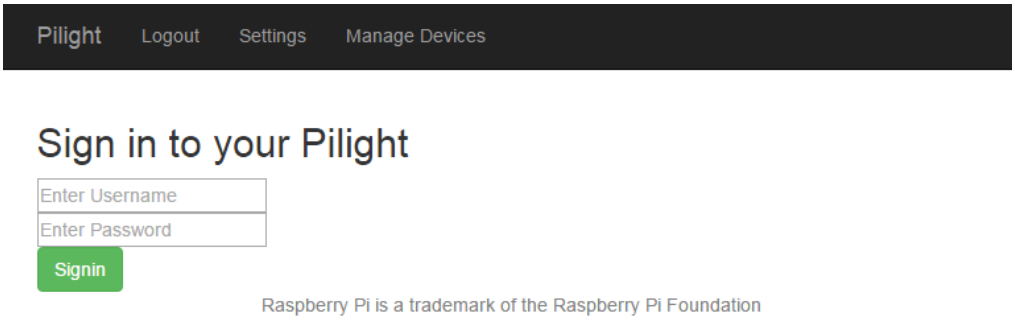
- Groups in the main page can also be created, renamed, nested into each other, and deleted, though deletion has not

been fully implemented yet.

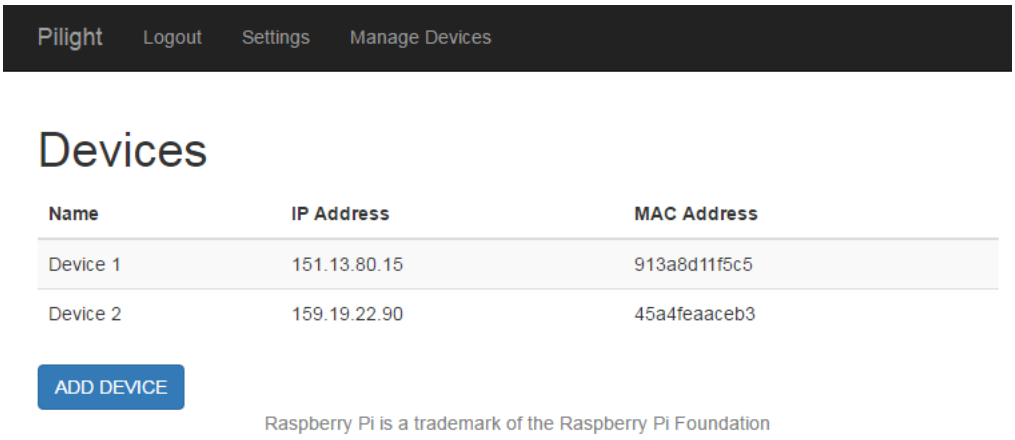
- The web interface has an advanced page for each light/group where rules for when to automatically toggle on/off can be created.
- All types of rules (including time of day, day of week, day of month, day of year, specific dates, sunrise/sunset times, and even parent interaction and offsets/staggering) are available to be created.



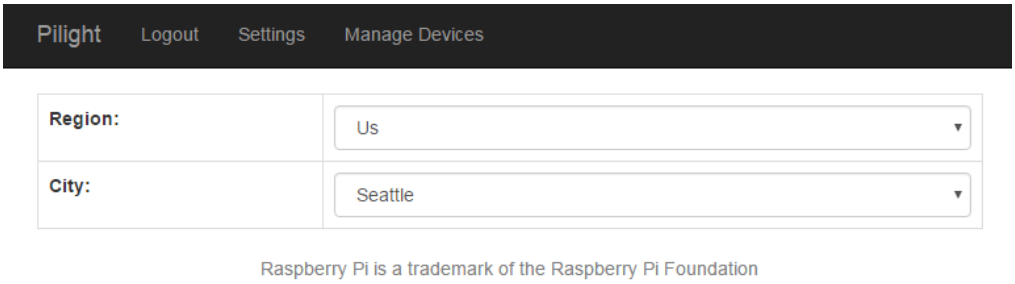
- The web interface has a login screen.



- The web interface has a device pairing screen.



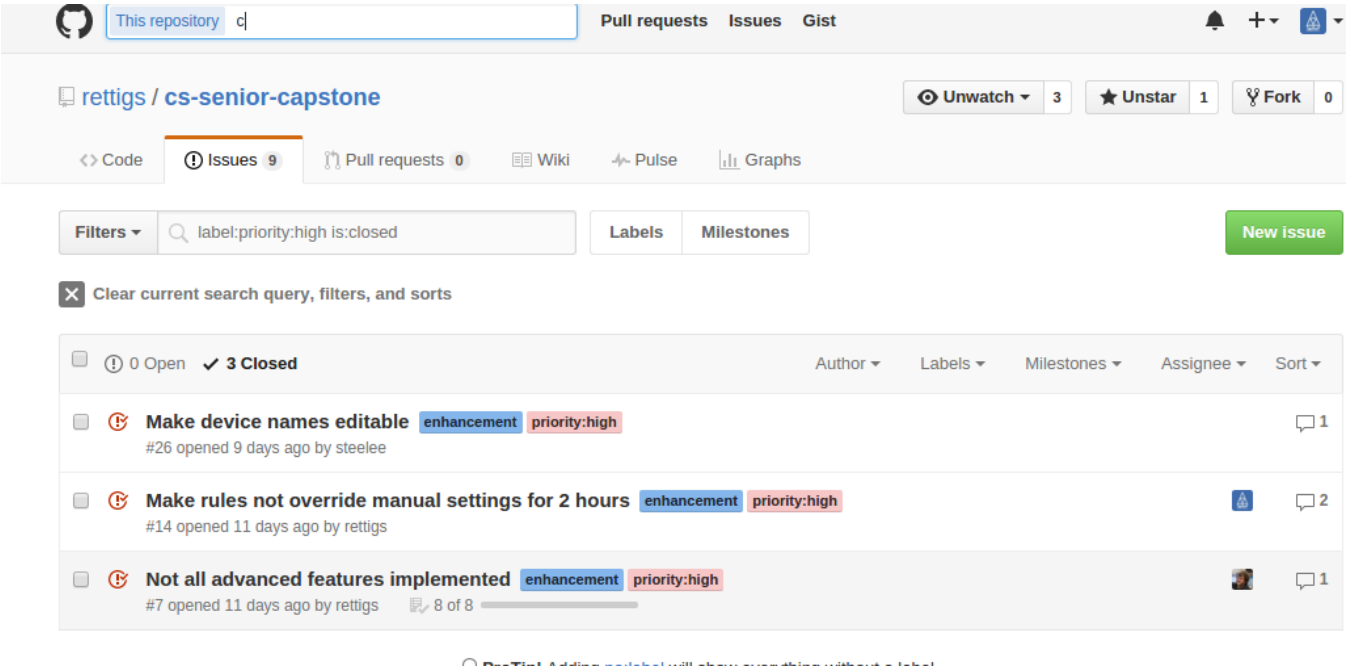
- There is now a Settings page, where global settings for the application can be set.



- The web interface manual control will override the automated rules until the next rule change.

II. Work in Progress

Much of our work after the alpha release had to do with the bugs we encountered adding the missing features into the project. The override system provided some bugs, such as issues with when a rule would have been moved back after manually changing it. Additionally, we had some issues with implementing the advanced options and adding new changes to the database. Our full list of high priority issues was on our Github issues page. All these issues were successfully resolved, leaving us with only low-priority issues.



A. *Device Communication*

One of the current post-Alpha goals is to complete the communication mechanism between the Raspberry Pi and the ESP8266 modules. We are working on finalizing our logical evaluation subsystem and determining what do to in the case of manual user override. The databases and communication parameters have been completed; we know what the function that will flip the lights themselves and we have finalized the function that sends the command. The same function is used for both query execution and manual control, and has been vigorously tested with all the relevant devices. Database control has also been implemented so that the lights use the persistent data.

The devices are already prepared in a database to draw from, using the MAC address as the unique key to locate the IP address from the table. We don't anticipate the ISC DHCP server flipping addresses around, but by using the physical hardware addresses we are eliminating any possibility of sending the code to the wrong device. The Python ISC DHCP library provides us with a mechanism for determining if a device's IP has changed, and we are writing those changes to the devices table.

B. *User Interface/Website*

Much of the functionality that we need for Beta was completed at the Alpha stage. The database has been created and the main page is currently able to persist user-submitted data, such as the names of groups and lights. The advanced page now triggers server communication utilizing newly created user-submitted data, using the same core functionality as the front page.

The user interface saw some slight design changes, such as changing the CSS to make it more friendly for mobile devices by shrinking some elements on the page. Additionally, the web application has been set to auto-deploy on the Pi with sufficient error handling when powered on, and it sets the display on the TFT LCD to properly show the application.

C. Scheduling System

The scheduling system is what ties together the website with the hardware communication; it periodically evaluates all rules for all lights/groups and initiates the sending of signals to the ESP8266 modules based on which lights are to be turned on/off. The scheduling system is also in charge of parsing the group/light rules that are stored in JSON format into boolean logic expressions that can be evaluated, including replacing keywords such as ”sunset” and ”sunrise” with actual times based on the user’s location and the time of the year. A complex JSON-driven data collection and query analysis system has been a major focus since the Alpha. After the remaining elements were added to the query and written to the database, testing proved that the queries were being executed correctly.

III. Problems

A. Wireless

We had an issue early on with how we planned to connect to the ESP8266 modules using our Raspberry Pi. Our proof-of-concept had the client connecting to the wireless module itself to make changes, but we wanted remote management to work from the Pi and have it be reachable from the Internet. We began looking into ways to perform local routing so that the device could play with the end user’s home network, but ran into more snags trying to configure multiple wireless network devices on the same Pi. We contacted our client, who suggested that we not worry about external connectivity for the project. His expectations were that users would directly connect to the Pi to make configuration changes. The project title was ”not *exactly* the Internet-of-Things”, after all.

To fix the issue, we reconfigured the Raspberry Pi so we could connect wirelessly. We needed to first recompile *hostapd* to run with a version of our USB wireless driver that could broadcast an SSID. After recompiling, we confirmed that we could connect using a wireless device, such as a smartphone. The other task was to change the firmware on the ESP8266 so that it behaves like a client instead of a host. In addition to flashing the new firmware, a startup file was added so that it automatically connects to the nearest Raspberry Pi with ”Pilight” in the SSID. We will likely change this later to better incorporate new devices.

B. Pulse Width Modulation

We had the idea that a device manageable via the Raspberry Pi would be able to control the brightness of the lights using PWM. Unfortunately the 10A gates on our relay just can’t flip fast enough for any desired pulses. Our group contacted our client to see if it was worth the trouble of acquiring a relay unit that could do PWM, but his response was that he was only concerned with turning the devices on and off. We also considered a software-defined PWM solution, but were again affected by the speed of the relay itself not being sufficient.

C. Behavior of On/Off Switches

Before starting to implement the rules, we had not put much thought into what the behavior should be for the On/Off switches of the lights and groups. After we began to implement more of the project, we realized that there are a few ways that the behavior could be implemented:

1) Toggling the switch would change the state of the light until the user chose to resume automated control.

2) Toggling the switch would change the state of the light for a preset amount of time before resuming automated control.
3) Toggling the switch would change the state of the light until the next time that the rules determined the light should change, upon which automated control would resume.

The are benefits to each implementation and it was a difficult decision to figure out which path we should take. The first option makes sense because the user is never overridden by the system. The second option is useful because the user most likely is turning on the light for a temporary purpose, and probably will not want to have to go into the advanced settings all the time. We decided to go with the third option, as it provided the benefits of the second option, but more cleanly eases back into the pattern of automated control. However, this solution does require extra state tracking on the server side to know when a light is being toggled by the re-evaluation of a rule, as the original design was to have the rule evaluation happen independently of the current state of the light.

IV. Code

A. Acquire New Devices

This code, upon user request, scans currently connected devices for a new ESP8266 device that does not already exist in the devices database.

```
// Get all DHCP leases currently given out
leases = IscDhcpLeases('/var/lib/dhcp/dhcpd.leases')
leases = leases.get()

// No DHCP leases at all, the server is not operating
if not leases:
    return "NODHCP"

additions = 0
// For each lease, first see if it is an ESP8266
for iteration,lease in enumerate(leases):
    if(ESP8266_check(lease.ip)):
    // We have an ESP8266 mode!
        database_check = model.Device.query.filter_by(mac=lease.ethernet).first()
        if database_check is None:
        // We have an ESP8266 AND it is not in the database!
            new_device = model.Device(mac=lease.ethernet,ipaddr=lease.ip,name="ESP8266@"+
                lease.ethernet)
            model.db.session.add(new_device)
            model.db.session.commit()
            additions = additions + 1
    return str(additions)
```

B. Editing Lights and Groups

We decided to use the X-editable library for executing inline edits to the names of groups and lights. This involved creating a span element to represent the name with information about the light or group embedded in it. We then included

a few lines of JavaScript to specify where to send the POST request and set up the editable span element. When submitted, a POST request would be sent to the Flask server. Below is the Flask function that handles the POST request and changes the name of a light or group in the database.

```
@app.route('/change_name', methods=['POST'])
def change_name():
    name = request.form['value']
    pk = request.form['pk']
    isLight = request.form['name'] == "light"
    if isLight:
        light = model.Light.query.filter_by(id=pk).first()
        light.name = name
    else:
        group = model.Group.query.filter_by(id=pk).first()
        group.name = name
    model.db.session.commit()
    return "Success!"
```

V. User Studies

A. *User Study A*

After exploring the interface for a time, the user was asked to perform a series of tasks to replicate common use cases for the system, including logging in, toggling lights/groups, renaming lights/groups, reorganizing lights/groups, creating additional groups, connecting additional client modules to the server, and creating advanced rules for lights/groups.

- The user had no problems with the login system, but they disagreed with the text on one of the buttons.
  - The SignIn text on the button should be changed to Sign In.
- There was a slight issue with the Add Devices button, as the user was confused by the loading animation, and had a suggestion for how to fix it.
  - There should be text to indicate what is going on during the loading animation for adding devices.
- The user had no problems changing the name of a light.
- The user was able to correctly identify that the group/light system was a hierarchy where lights were owned by groups and groups were owned by parent groups.
- The user was able to toggle the light on and off, however they thought that the switches were not very intuitive, and they had a few suggestions for improvement.
  - The switches should be green/red, not blue/gray.
  - The switches are too slow, and should be twice the speed.
  - The switches should show "On" and "Off" text.
  - The groups should have two on and off buttons instead of switches, and should not have a state. The intermediate state of the switch does not make sense.
- The user was able to correctly identify that the red "x" was the intended button for deleting a group.

- There was a slight problem when the user attempted to move a light to a different group. The user identified the move icon as the correct tool for moving rows, but had difficulty clicking on it, as the spaces in between the arrows do not register a click. Once they had grabbed it and were moving it, they were then slightly confused because there was no preview of the destination, and instead the light was attached to the mouse. They did drop it and move the light correctly, but they noted that the task was slightly less intuitive than the previous ones.
  - When dragging and dropping lights or groups, there should be a preview of the destination while the user is holding it.
- The user had no problems adding a new group, but they disagreed with how the new group was set.
  - When adding a new group, the switch should be set to Off.
- The user was able to navigate to the advanced page for a light by clicking on the gear.
- When asked to set the light to turn on at sunset and turn off at sunrise, the user was able to find the correct radio buttons. The user then noted that they were not sure whether they needed to do anything else to save what they had done.
  - There should be a ”Save Changes” or ”Submit” button
- The user was able to correctly set the light to turn on at a specific time of the day.
- The user was unable to set multiple rules, as it was not obvious to them how the rule system worked, they made a few suggestions for how the system could be improved.
  - The New Rule and rules controls should be at the bottom of the page, and that the wording should be changed to something other than ”Rule.”.
  - Advanced query box should be moved to bottom of page or hidden in something like an expanding box, as most users will not want to see or understand it.
  - There should be a preview of the rules for a group or light when hovering over the name or the link leading to the advanced page.
  - The wording ”Rule” should be changed to something that represents the idea more clearly.
- The user was able to identify the purpose of the menu bar but thought that it was not very noticeable.
  - Menu bar should have greater contrast so it is more noticeable.

As we conducted this user study near the end of Alpha stage, we have yet to decide whether to implement the suggested changes and have yet to make any changes as a result of the usability tests. We will go over these in the future and make judgments about what changes should be made then.

B. User Study B

- User had no issues with the login page.
- User attempted to rename a light, but clicked away when done editing rather than hitting enter, causing the rename to not be saved.
  - Suggestion: Clicking away should also save a rename.
- User noticed that it was not obvious which features were available on the front page.

  - User suggested that a small blurb or tutorial should pop up on the first use that tells the user about renaming, moving, creating, deleting, and editing lights/groups.
- User was not sure whether lights were on or off.
  - User suggested that the switches should use actual ”on/off” text and maybe more obvious colors than blue and gray. Possibly gray out the entire light’s box if it’s off.
- User was not sure how to create a new group.
  - User suggested that the ”create group” button be made more obvious.
- When asked to add a device to the system, the user was stumped because they did not notice the ”manage devices” tab.
  - User suggested that the area for adding devices be displayed more prominently.
- On the advanced page, the user felt that there were too many elements in general.
  - The user suggested that only the most basic and commonly used elements of the rule builder should be shown by default. The more complex settings and the advanced query should only show up if the user clicks a ”show more options” button or the like.
- On the advanced page, the user found it confusing that they had to click the ”create rule” to finalize the rule and that there could be multiple rules for one light.
  - This could possibly be alleviated by only allowing one rule for each light. This would help simplify the UI, and most users probably won’t want multiple rules for each light anyway. Advanced users would still have the option to manually edit the advanced query in order to achieve the effect of having multiple rules.
- The user found it difficult to edit the settings for groups of lights due to the clumsiness of having to navigate back to the front page in order to view the other lights.
  - The user suggested that something like ”next light” and ”previous light” buttons or the like be added to the advanced page in order to improve ease of navigation.

C. User Study C

- User did not understand the multi-state light switch.
  - Suggestion: Add a tooltip (maybe only for the first time?) or add a hover element to the button indicating the current state.
- User did not understand what the group symbols meant.
  - Suggestion: This one could just be ”read the documentation” that we will include in the final project. It will probably be a PDF attachment or a Github wiki page.
- User did not know if a custom query was running
  - Suggestion: We’ll add a note next to a light if it is currently controlled by a query. We can have a hover element to display the specific query.