DESIGN DOCUMENT
for

Not Exactly the Internet of Things for Outdoor Lighting

Conceptual Release: Subject to change

Malcolm Diller, Sean Rettig, and Evan Steele
Client: Victor Hsu
OSU CS Senior Capstone Group 22

PiLight

1 Introduction 1
2 Design View 1
3 Design Viewpoints 3
4 User Experience 3
5 Testing 4
6 Timeline 5

1 Introduction

In the twenty-first century, your home should be smart and responsive. An automated home should be able to control appliances such as lights through single-board computer units that have Internet connectivity. Many commercial options exist, such as those from Nest Labs, but such devices are prohibitively expensive for most consumers, especially for simple tasks such as lighting automation. This project aims to provide an low-cost alternative for simple home automation, for devices like lights and garage door openers. Using a low-cost embedded computer such as the Raspberry Pi and cheap wireless communication modules such as the ESP8266, this project will provide a cheap, easy-to-install home automation system.

The project is called "Not Exactly the Internet-of-Things" because it's not designed as a typical IoT product. The network is entirely local, not worrying about an external connection to the Wide Area Network. Some additional components such as consulting an external API could be considered, but that's an addon to the finished product. As it is made as a home application, it is readily accessible through smartphones and web interfaces. With a clean user interface that enables both manual control and automation profiles, the application will require a minimal learning curve and be simple to set up.

2 Design View

The aim of this project is to create a home lighting automation system that is inexpensive, wireless, and easy to use. The user of this product should the following features:

1. Lights and Groups
   (a) The lights can be controlled from the built-in touchscreen on the control unit, or a mobile device such as a phone, tablet, or laptop
   (b) Lights can be put into "groups" so that they can be controlled all at once
   (c) Groups can also be nested into other groups to create hierarchies of lights, in order to control many lights at once
   (d) The user will be able to nickname lights and light groups
   (e) The intensity of each light can be adjusted using a slider
2. Rules / Types of Control
   (a) Lights can be toggled on and off manually, but they can also be set to be activated on specific schedules using "rules"
   (b) Rules can be set for lights or groups of lights
   (c) There are 6 main types of schedules that rules can trigger on

   i. Time of day (e.g. turn on at 8pm and turn off at 6am)
   ii. Day of week (e.g. turn on during Wednesdays)
   iii. Day of month (e.g. turn on every 1st of the month)
   iv. Day of year (e.g. turn on every January 1st)
   v. Specific dates (e.g. turn on from Dec. 24th at 2am to Dec 27th at 8pm)
   vi. Sunrise / sunset (e.g "turn on when the sun sets, turn off when the sun rises")

(d) Multiple rules can be applied to each light or group of lights by using AND or OR logic

(e) Lights can be toggled based on the toggle state of its parent group

(f) Rules can be enabled or disabled temporarily

(g) Lights or groups can be set to gradually dim/brighten over a set period of time instead of toggling

3. Stretch Goals

(a) The system can control devices other than just lights:
   i. Garage door openers
   ii. Music players
   iii. Holiday decorations
   iv. Sprinkler systems

(b) The web interface is accessible over the Internet (Instead of only on the local network)
   i. The interface is accessible for only the homeowner or authorized individuals
   ii. The interface hosted on a remote server if the user does not want to deal with port forwarding their home network.

(c) Additional types of triggers for rules
   i. Weather conditions (e.g. lights can turn on automatically if it starts raining)
   ii. Schedule in a Google calendar
   iii. Moon position or other celestial data (e.g. lights could turn off during a full moon or solar eclipse for better viewing)
   iv. Motion sensors (e.g. lights turn on when someone walks up to the front door)
   v. Light sensors (e.g. lights turn off at a certain brightness level)
   vi. Moisture sensors (e.g. lights turn on when it's wet outside)
   vii. Sound sensors (e.g. lights turn on after a loud noise, or strobe to the beat of music that is playing)

3 Design Viewpoints

We will be using a wide variety of open source technologies and inexpensive hardware to provide a low-cost home automation system. We will be using a varied set of tools to accomplish our goal, including:

1. The Yocto Project
   (a) The Yocto Project is a kernel building system designed for embedded systems. It easily incorporates additional features and custom packages directly from Git repositories. It makes it easy to build a customized image with only exactly what we want and no unnecessary packages [1]. It also allows us to customize elements of the kernel such as startup scripts in init.d, kernel modules, and network scripts [2]. The kernel will be compiled for the Raspberry Pi and copied to an SD card using Yocto's filesystem script.
2. Raspberry Pi 2
   (a) The Raspberry Pi 2 is an ARMv7-processor-based microcomputer designed for educational and embedded projects [3]. We will use it as a low-cost server solution for the lighting automation system. It can serve as the web server running NGINX and transmit TCP commands to the ESP8266 wifi modules [4]. The Pi will be outfitted with a 320x280 TFTLCD display [5] so that the interface for control can be accessed directly. The Pi will also act as a wireless router for the plug units to connect to, and run a DHCP server to distribute IP addresses to the plug units.
3. ESP8266
   (a) The ESP8266 is a self-contained SoC (System on a Chip) that has a built-in Wi-Fi radio [6]. Our module is outfitted with custom firmware, written in Lua and flashed to the device using the Arduino IDE, that allows the module to automatically connect to the Raspberry Pi (control unit) that acts as a WAP (Wireless Access Point) [7]. The board we have is designed to connect to an electronic relay using jumper cables. Using simple TCP commands, the board will flip the relay pins on and off [8].

4 User Experience

The final deliverable of a commercial version of this product would likely include the following:

- A central control unit consisting of the Raspberry Pi, its power cable, the touchscreen module, a case, and an SD card with the server software preloaded onto it.

- One or more plug units which would likely resemble a power strip, with each plug being individually controllable by the system.

To set up the system, the user simply plugs both the control unit and the plug unit(s) into wall power. The plug units will preconfigured to connect to the control unit. The user can then plug devices into the plug units, such as lamps, and the display on the control unit will update itself automatically with a list of lights. The lights can have either numbered (e.g. "Zone 1, Light 1") or randomized but memorable names by default (e.g. "Blue Koala") which can be later edited through the web interface. To access the web interface, the user connects to the wifi network provided by the control unit using the credentials included with the manual, or perhaps printed on the control unit itself. They then open a web browser and connect to a specific URL, provided to the user with the wifi password. From the web interface, all features are accessble, such as renaming lights, creating light groups, and applying rules for when to toggle lights.

5 Testing

To test the functionality and correctness of our system, we will test the various components both individually and when integrated with other components:

- We plan to test the wireless communication capabilities of the system by attempting to send packets between the control unit and plug units at varying distances and in varying environments, including open and walled areas. Ideally, the range should be great enough to fully cover the average American house/apartment. Once this is complete, it will likely not need to be changed or extended much, reducing the need for regression testing.
- We plan to test the actual light toggling (as performed by the plug units) manually. Once this is complete, it will likely not need to be changed or extended much, reducing the need for regression testing.
- We plan to test the functionality of the touchscreen interface through manual human interaction testing. Once this is complete, it will likely not need to be changed or extended much, reducing the need for regression testing.
- We plan to test the functionality of the web interface through a combination of manual human interaction testing, automated API-driven testing, and automated GUI testing (using a platform such as Selenium). The plug units can send back acknowledgement packets to confirm what actions they took in response to the tests, allowing nearly complete integration tests to be performed automatically when changes are made to the control program and web interface. These automated tests will help prevent regressions as new features are developed and save time in ensuring the correctness of the system.

6 Timeline

Device control	**Due Date**
Server can boot with GUI and touchscreen support	Thu 11/19/15
Server starts control program on power on	Mon 11/23/15
Clients are loaded with client control program	Thu 11/12/15
Clients discover connected relays/lights	Thu 11/19/15
Clients can toggle lights individually	Tue 11/24/15
Lights can be toggled over time	Fri 11/27/15
Server acts as wireless access point for clients	Mon 11/30/15
Clients can be paired with servers	Fri 12/4/15
Clients tell server which lights are available	Tue 12/8/15
Server can send light instructions to clients	Thu 12/10/15
Clients can receive light instructions from server	Mon 12/14/15
Clients can perform light instructions from server	Wed 12/16/15
User Interface	
Server serves web site with control interface for user	Mon 11/30/15
Accessible via the systems built-in touchscreen	Mon 12/7/15
Accessible to other devices like phones	Thu 12/3/15
Displays list of connected clients	Mon 12/7/15
Can nickname clients	Tue 12/8/15
Displays list of connected lights	Wed 12/9/15
Can nickname lights	Thu 12/10/15
Buttons to toggle lights individually	Thu 12/10/15
Sliders to change light intensity	Fri 12/11/15
Can put lights into groups	Mon 12/14/15
Displays list of groups	Tue 12/15/15
Can nickname groups	Wed 12/16/15
Groups can be nested	Wed 12/23/15
Can toggle lights/groups based on rules	Fri 1/1/16
Rules	
Lights can be toggled manually, overriding any rules	Tue 12/15/15
Rules can be temporarily disabled	Wed 1/6/16
Lights can be toggled based on time of day	Tue 1/12/16
Lights can be toggled based on day of week	Thu 1/14/16
Lights can be toggled based on day of month	Thu 1/14/16
Lights can be toggled based on day of year	Thu 1/14/16
Lights can be toggled based on specific dates	Thu 1/14/16
Lights can be toggled based on sunrise/sunset times	Fri 1/22/16
Multiple rules per light/group	Fri 1/15/16
Lights can be toggled based on parent group	Tue 1/5/16
Rules can toggle early/late by random time	Tue 1/5/16
Lights can toggle early/late randomly within groups	Wed 1/6/16
Lights can gradually toggle over time	Thu 1/7/16

References

[1] "Yocto Project yocto build server," https://www.electronickits.com/4-channel-auto-roll-rf-relay-board-assembled/.

[2] "Flask flask python microframework," http://flask.pocoo.org/.

[3] "Raspberry Pi embedded devices," https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[4] "NGINX nginx web server," https://www.nginx.com/resources/wiki/.

[5] "TFTLCD pi display," http://www.adafruit.com/products/1774.

[6] "ESP8266 wifi module," https://www.tindie.com/products/hbouzas/general-purpose-wifi-control-board-x4/.

[7] "Lua lua scripting language," http://www.lua.org/.

[8] "Relay 10a relay board," https://www.electronickits.com/4-channel-auto-roll-rf-relay-board-assembled/.