Enterprise Smart Outlet - iOS Application

By Riley McGovern

CSC 492: Senior Project

Dr. John Bellardo

June 10, 2015

Abstract

The Enterprise Smart Outlet is a system made up of a smart outlet, a Node.js server, and a companion iOS application. It builds on the already existing market of smart power outlets and aims to bring that type of functionality fully scaled up to the enterprise level. The smart outlet connects to the Node.js server and sends its power readings. The iOS application connects to the Node.js server to receive all power readings from all the connected smart outlets and can also toggle the power on and off for each outlet.

Contents

List of Figures

1 Introduction

When it comes to the world of smart outlets, there are many different types of consumer solutions for people to buy. When it comes to the enterprise scene, however, there are very few, if any, available solutions. The smart outlets that are created for enterprise use do not have the ability to either turn off remotely or display the power readings on a smart phone. The Enterprise Smart Outlet aims to solve these problems.

1.1 Project Goals

The goal of the project was to make a smart outlet that could be used in an enterprise setting and could be easily used both by a building's utility workers as well as anyone who walks into the building. The idea would be to use the Enterprise Smart Outlet in places such as the Cal Poly Library. When a student walks into the building, they could launch the ESO iOS app on their phone. When they launch the application, they will be given a list of all of the outlets currently in the building. At a glance, the user could easily determine which outlets were in use, available, or inactive. When looking at each smart outlet, they could view the most recent power readings from that outlet. The building's utility worker could also use the switch feature on the outlet when the building is ready to close. Say, when the library is ready to close at 2 AM, all of the outlets could remotely be switched off and it would both save the school power as well as get the students out in a timely matter.

1.2 Why iOS

Apple's iOS platform is one of the biggest mobile platforms today. Most people who own a smart phone either own an Android smart phone or Apple's iPhone. Apple also has the best platform for delivering applications to end users. I personally own a few iOS devices and have consistently preferred them to some of Android's offerings. Being able to access millions of users that you know already use iOS devices made choosing the platform an easy choice.

1.3 Technology

1.3.1 Swift

The entirety of the iOS application was written in Swift using Xcode 6. Swift is Apple's newest programming language that is intended to take over the existing Objective-C used for current and past iOS applications. Swift has new language features that Objective-C does not offer, is much more readable, and is easier to pick up as a new developer. [2]

1.3.2 Xcode

Xcode 6 is the integrated development environment that was used to create the iOS application. Xcode was created by Apple. On top of the useful auto-complete and syntax coloring, this IDE also helped create the storyboards with auto layout constraints to fit any size iOS device with ease. [3]

1.3.3 Node.js

Node.js is a JavaScript library that was built upon Google Chrome's JavaScript runtime. This was used to build both the HTTP side of the server for the iOS applications as well as the FTP server to connect and send and receive information from each of the outlets. [6]

1.3.4 MySQL

MySQL was used to store all of the outlets' data and their corresponding IP addresses on the server for easy look-up when they wanted to be turned off. [7]

1.3.5 FTP

File transfer protocol (FTP) was used to communicate between the outlet and the server. When an outlet was being either activated or deactivated, the server would send either a '0' or '1' to the server. When an outlet wanted to add itself or update it's current status/power values, it would send a specified string with it's MAC address and power readings so the database could properly update the correct entry. [4]

1.3.6 JSON

JSON, also short for JavaScript Object Notation, was used to get the information of all the outlets onto the iOS device. The iOS device made a GET request to receive the JSON string of all of the outlets and their data. [1]

1.3.7 Realm

Realm was used in order to save the outlets information, even if the user had left the local WiFi network. Every outlet has a "last contacted" field so the user knows whether or not that outlet is up-to-date or not. [8]

1.3.8 Android Companion App

An Android companion application was made by Aaron Koeppel in parallel with this iOS application, also titled Enterprise Smart Outlet.

2 Related Projects

2.1 Belkin WeMo

The Belkin WeMo is a smart power outlet that is aimed at home use. The power can be monitored as well as the ability to turn off the outlet if necessary. The outlet is controlled via a smart phone and can turn the outlet on and off based on a predefined schedule by the user. [5] You can see an example of the Belkin WeMo in Figure 1.

Figure 1: Belkin WeMo

3 User Guide

3.1 Launching the App

To launch the application, select the icon with the 'O' logo and 'ESO' app name on the home screen as seen in Figure 2.

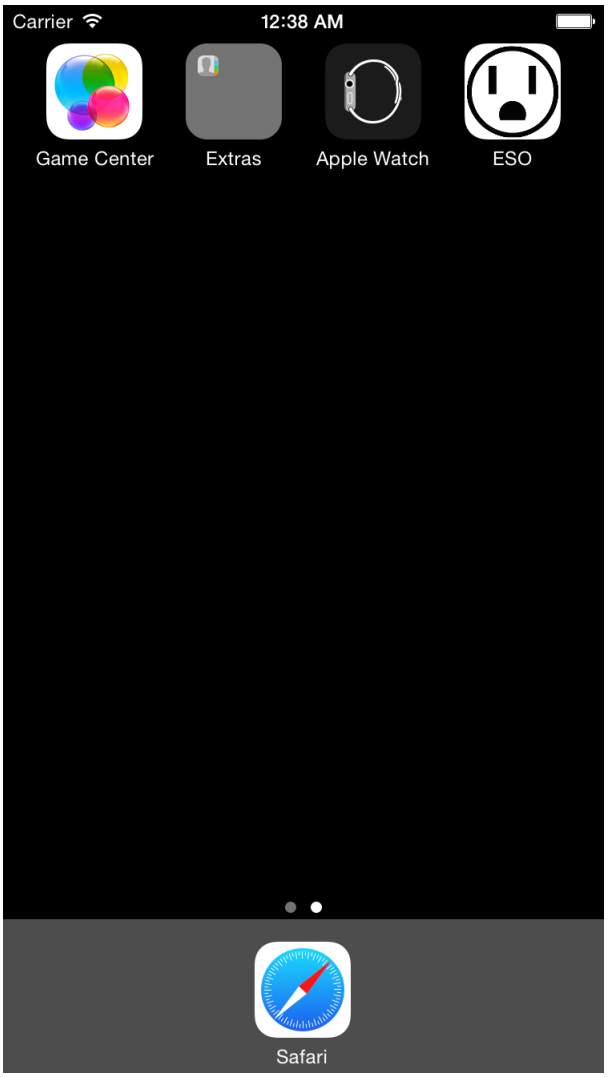

Figure 2: The application on the iOS home screen.

3.2 Entering IP Address of Server

When the application first launches, you will get a screen seen in Figure 3.

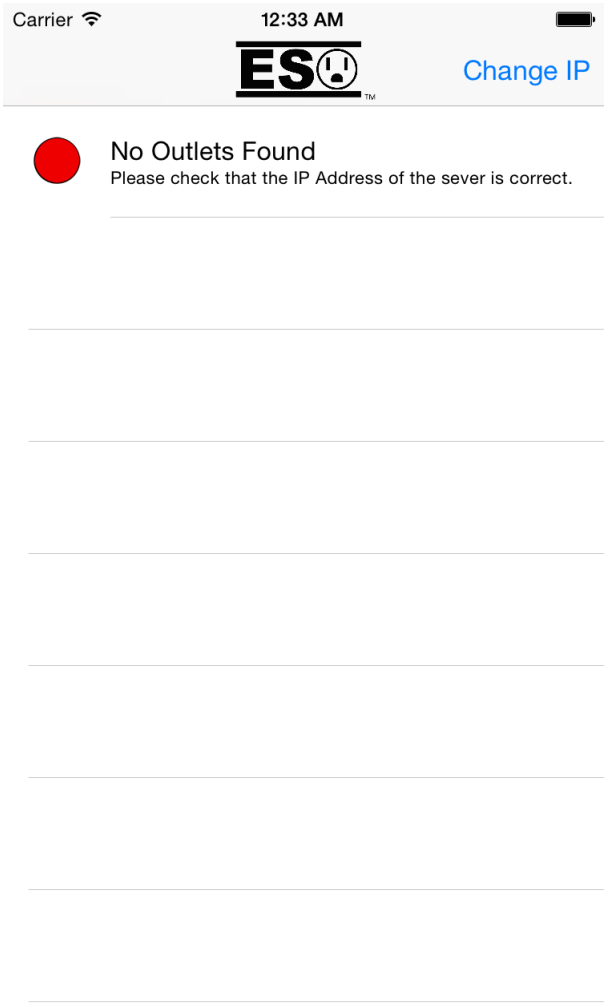

Figure 3: The main page with no outlets.

No outlets will currently be showing as shown in Figure 3, so you must enter the IP address of the server that is connected to the outlets, as shown in Figure 4. Currently there is no support for a DNS server to automatically connect to the server.

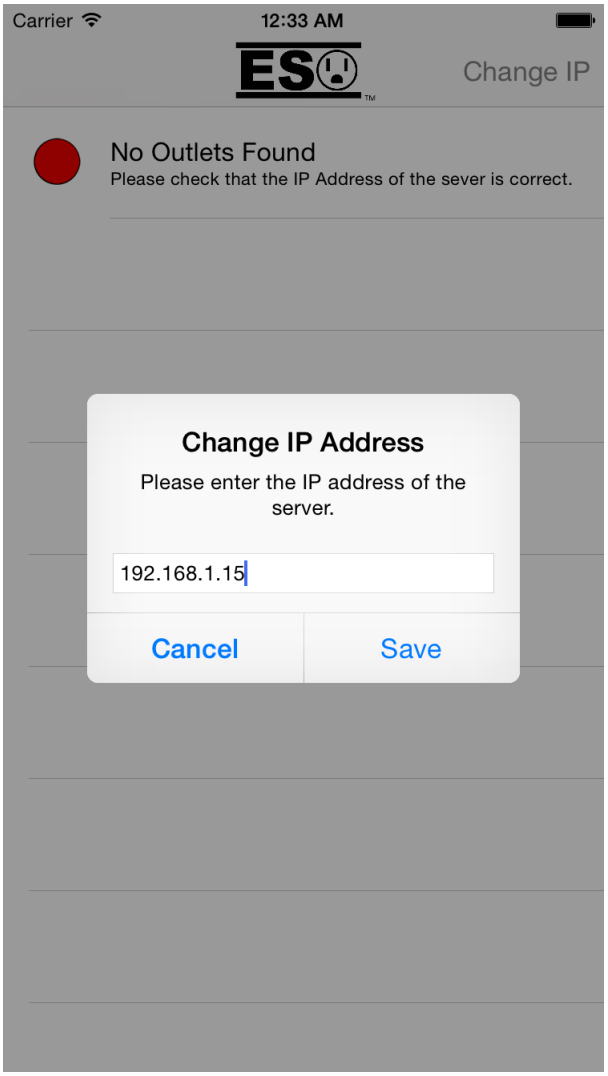

Figure 4: Entering an IP Address of the server.

3.3 Knowing the Different Symbols

After the application has successfully connected to the server, and assuming outlets have connected to the server, you will see something similar to Figure 5. Each outlet will have either a red, yellow, or green circle next to it's name along with it's group.

- red = The outlet is currently inactive.
- yellow = The outlet is currently in use.
- green = The outlet is currently free.

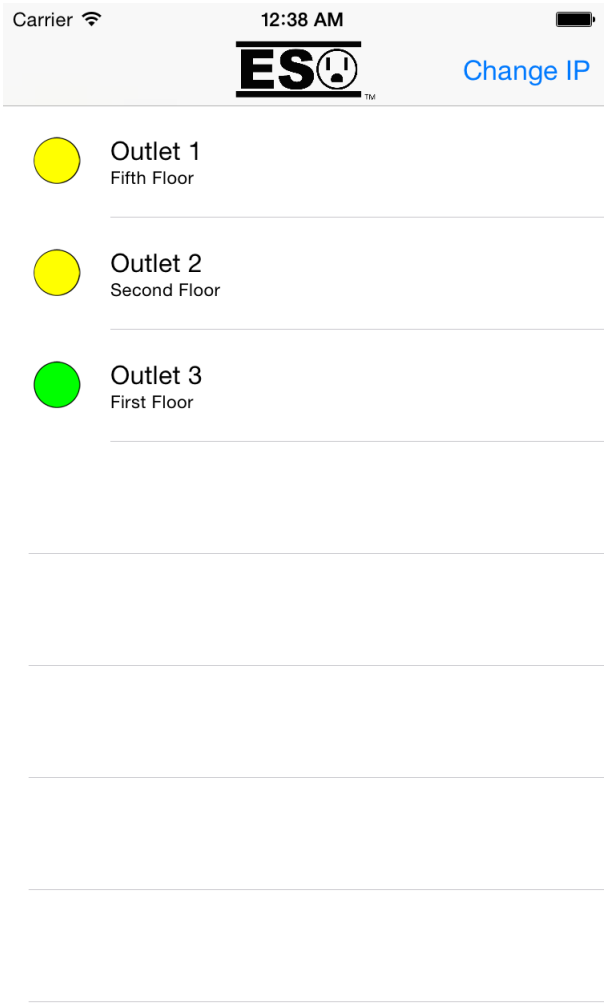

Figure 5: The main page with some outlets.

3.4 Individual Outlet Screen

When tapping on an outlet on the main list of outlets, you will be brought to a more detailed view of the outlet, as seen in Figure 6. From here, you can view the outlet's voltage, current, power, description, group, and the last time it was contacted. You can also view it's current status and activate/deactivate based on it's current status. The button to activate/deactivate is on the bottom.

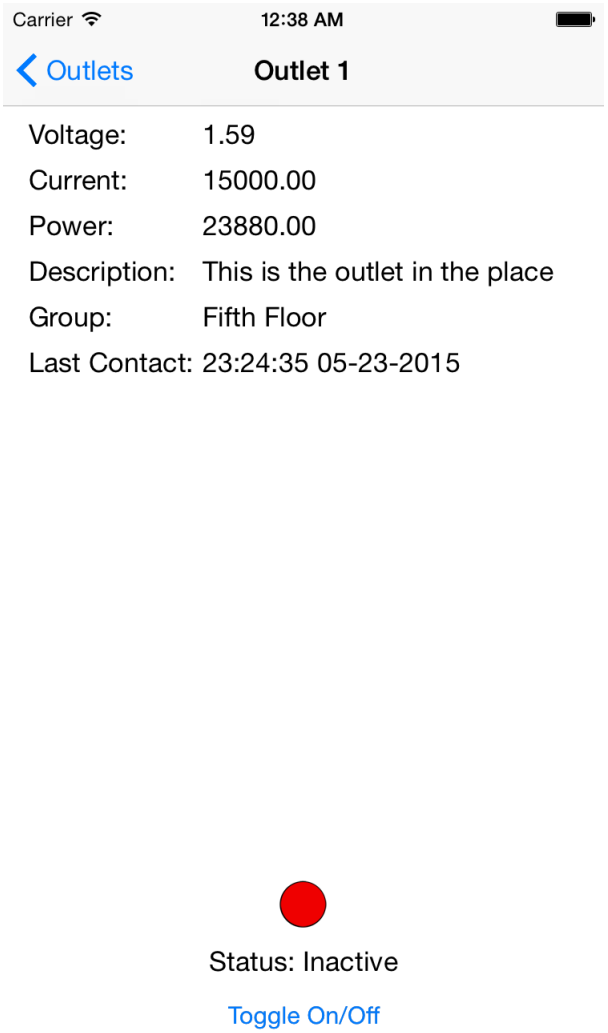

Figure 6: The detail screen of an outlet.

3.5 Pulling Down to Refresh

To refresh the outlet data, simply pull down to refresh on the main screen.

3.6 Error Connecting to Server

If for some reason the application cannot find the server, an error message will pop up. Please check to make sure the IP address was entered correctly.

4 Technical Specifications

4.1 iOS Application

A custom Outlet object was created to store the MAC address string, outlet group string, voltage double, current double, last contact string, and a description string. The Outlet object also extended a Realm object so that it could be stored in a database for offline viewing. The main screen of the application was created using a UITableView with custom UITableViewCell's to display the status, name, and group of each outlet. All of the outlets were stored in an array of Outlets. When the user enters a new IP address, the application forms a URL the contact the server and pulls the JSON data. When the user selects the individual cell, the main screen passes the Outlet object to the detail screen controller. On that screen, all of the Outlet data is displayed. If the user wants to activate/deactivate an outlet, the application forms a URL based on the IP address of the sever and the MAC address of the selected outlet. The status of the outlet will change based on it's current status.

4.2 Node.js Server

When an outlet first connects to the server, it sends a specified 'add' string, with its MAC address, name, group, and description via FTP to the server. To update its values on the server, the outlet sends a specified 'update' string via FTP with its MAC address, voltage, and current. The server is backed by a MySQL database and when the server receives either a 'add' or 'update' string, the server will pass along those values to the database accordingly. When the iOS application connects to the server, the server uses JSON.stringify() to take the information from the MySQL database and translates that into JSON data. This happens every time the iOS application requests new outlet data.

5 Future Improvements

While almost all of the base code to monitor and turn on/off the outlets was finished, security was never a thought in the first run of the project. If we were to continue working on the project, security would definitely be on of the first things to improve upon. Because all of the electrical engineers who worked on the hardware side of the project are graduating, they decided not to continue their endeavors with this project, so this will likely be the end of the line for the Enterprise Smart Outlet.

6 Conclusion

As someone who really enjoyed the iOS ecosystem in the first place, the next obvious step as a Computer Scientist was to delve into the world of iOS development. Learning Apple's new language, Swift, and iOS programming as a whole from Dr. Bellardo was honestly one of my most fun experiences here at Cal Poly and I hope to start a career in iOS development. I enjoyed working with Aaron Koeppel as he was working on the Android application in tandem with this one as well as collaborating on the server. It was also very rewarding getting to work with electrical engineers and to get all of our technologies integrated together and working. I think an idea like this has great potential if it is well executed by a large, experienced team with enough time and resources to make this great.

References

[1] "Json," 2015. [Online]. Available: http://json.org

Used for the descriptions of JSON.

[2] Apple, "Swift," 2015. [Online]. Available: https://developer.apple.com/swift/

Used for the descriptions of Swift.

[3] ——, "Xcode," 2015. [Online]. Available: https://developer.apple.com/xcode/

Used for the descriptions of Xcode.

[4] V. Beal, "File transfer protocol," 2015. [Online]. Available: http://www.webopedia.com/TERM/F/ftp.html

Used for the descriptions of FTP.

[5] Belkin, "Wemo," 2015. [Online]. Available: http://www.belkin.com/us/F7C029-Belkin/p/P-F7C029;jsessionid=A8BCF5735BFB87F5AA563139FB39931B/

Used for the descriptions of the Belkin WeMo.

[6] I. Joyent, "Node.js," 2015. [Online]. Available: https://nodejs.org

Used for the descriptions of Node.js.

[7] MySQL, "Mysql," 2015. [Online]. Available: https://www.mysql.com

Used for the descriptions of MySQL.

[8] Realm, "Realm," 2015. [Online]. Available: https://realm.io

Used for the descriptions of Realm.