2013

Skini: A device ecosystem for integrating household appliance events into smartphone and tablet apps

Saro Iskanian
Grand Valley State University

Skini: A device ecosystem for integrating household appliance events into smartphone and tablet apps

By
Saro Iskanian
December 2013

Skini: A device ecosystem for integrating household appliance events into smartphone and tablet apps

By
Saro Iskanian

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
December, 2013

Dr. Jonathan Engelsma **Date**

Table of Contents

Abstract

Research in ubiquitous computing has been underway for almost two decades, yet the smart home is still far from commercial realization. This project highlights a cloud-based ecosystem that is an attempt to close the gap between the standardization efforts of smart home and the interfaces of household appliances. The web-based platform consists of an interface that allows household appliances to report about their operational events and change of states. It also provides a framework for third party web and mobile applications to consume those events to make higher level analysis and act upon them. The platform gives the user total freedom to program how these notifications are delivered to their mobile devices or third party applications. Finally, to overcome the integration readiness of today's legacy appliances, generic or specialized sensor-packs called "Digital Skin"(s) are used to add the layer of sensibility and connectivity required.

Introduction

Today, connectivity in home appliances is regarded as a high-end novelty in home devices. For consumers, smart homes, with its home networking and automation systems is becoming a focal point. As early as in 1980s the Las Vegas Consumer Electronic Show was already demonstrating intelligent home network devices. This view will become outdated as we move to a future where connectivity is pervasive and embedded in virtually all-household devices. Many analysts believe that the smart home of the future is likely to contain 15 to 30 connected devices and sensors, all linked via a home area network and connected to service providers' back-end systems and the Internet. The combined revenue from the smart metering, home automation and home energy management (HEM) segments is forecast to be worth more than $44bn in 2016. Mobile connectivity will be a crucial ingredient in bringing together the different parts of the smart home puzzle [1][2]

Today mobile devices have become part of our daily activities, a tool to communicate, socialize and control. Connectivity and new capabilities bring new expectations with it. Consumers and developers have been looking into new ways on the path to the realization of ubiquitous computing.

To address the market demand in home automation, every vendor started to work on a standard that is proprietary and introduced new smart devices that can only communicate with products of the same vendor. This disconnect between different vendors yielded into some kind of chaos and increased the gap between ubicomp and the realization of smart homes.

Another major issue is the dominance of legacy devices in the market. The lack of regulations and standards is becoming a hurdle in the face of vendors who are willing to adopt new technologies in their streamline products.

Therefore, my motivation to pursue this issue, as a project was to address the challenge faced by appliance vendors in integrating connectivity into their home-appliances.

Skini, is a cloud-based web application that provides an end point for appliances to report their operational conditions or states. It exposes a programmable interface for third party applications to consume events of interest for high-level analysis and turn them into further actions. It allows users to control their preferences for each appliance. Legacy appliances connect to the platform by the means of generic programmable "digital skins". Administrators can create configurations for each appliance type, vendor and model that instructs the "digital skin" how to collect environment data and which ones to communicate to the cloud.

In order to make the platform available to everyone, it utilizes RESTful API and adheres to the notion of “internet of things” perceiving entities as resources with unique resource identifiers.

Background and Related Work

In a partnership between Arris/Motorola and Grand Valley State University two early prototypes of the sensor pack were constructed Fall 2012. The web interface was built to accommodate basic provisioning functionality and as a data collection and interpretation endpoint.

Later in 2013 a team of senior engineering students with the supervision of Dr. Robert Bossemeyer introduced a prototype of their design proposal, which is the de facto prototype at the time of this project. The team has taken a new approach in collecting the data and reporting the events to the portal. They came to the conclusion that there are several states in which an appliance exists; each is characterized by a set of readings from each sensors. Some of these states are transitional and doesn't have to be reported back, some others are states that we are interested to know when they happen. The web portal was evolved to accommodate this new feature introduced in the new prototype, allowing devices to download configurations from the cloud and report back the operational conditions or events, instead of just raw collected data.

Related work

Most of the vendors who are well known for their innovative stand have had some kind of approach to the creation of smart home enabled products. Some of these projects are worth mentioning:

1. **GE Brillion ™:** [The] technology allows you to interact with your appliances using your iPhone or Android smart phone. Currently, the app only works with certain GE wall ovens (models PT9050, PT9550, PK7000, PK7500), but more connected home appliances are on the way in 2014. [3] As we can see the project is in its initial phases, doesn't connect every product in GE and doesn't promote any openness.
2. **Smart ™:** Is a project developed by whirlpool that lets you monitor, manage and maintain appliances designed to work with the system. This project is more mature, since it connects to a broader variety of products than the first project.
3. **Creston:** Is a leading provider of control and automation systems for homes, offices and other venue types. Their product line goes beyond the scope of this project in terms of capabilities and concept. But in the big picture their products are tightly coupled with their systems and there is no explicit declaration about the standards that they use for communication
4. **Revolv:** Is a multi-radio device that connects to the home network through Wi-Fi and observes surrounding smart home and entertainment systems. A cloud platform allows the usage of iPhone to control those appliances or devices that Revolv understand and speak to. In one word, it is an imitator or repeater that centralizes control into one mobile interface. It also gives some level of programmability to the user by making possible to set certain actions to be triggered by predefined events or at certain times.

Program Requirements

The main purpose of Skini web portal is to provide the users with capability to digitally skin their legacy home appliances by adding a layer of sensibility and connectivity, and bringing related information to the hands of the user through a responsive web platform or mobile applications. It also gives them the freedom to control how they get notified about related events through a user-friendly web interface.

The main features of the cloud platform has include the following:

- Provisioning of Devices
- Provisioning of Appliance Types, Vendors, Skin profiles and States
- Provisioning of Users
- Provisioning of Skins
- Providing an API for 3rd party application and digital skins
- Integration with Apple Push Notification Service

The Devices provisioning section is intended for administrators of the portal. Each device is assigned a Universal Unique Identifier (UUID) that is then stored inside the device and used for communications with the portal. Because the ID is automatically generated and needs to stay unique it is not possible to edit it manually.

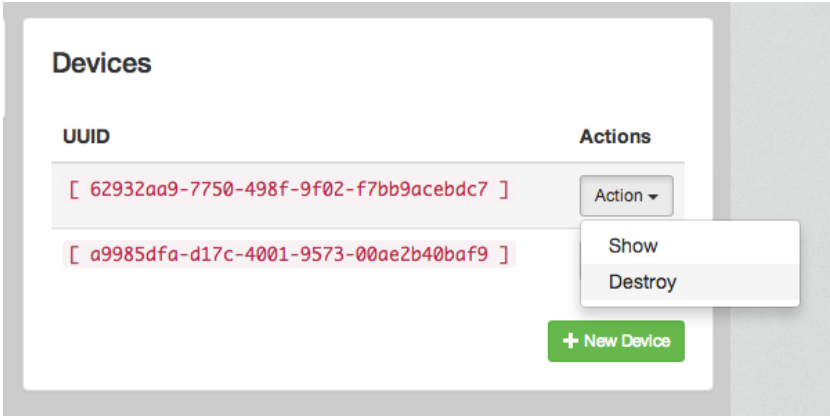

Figure 1: Device provisioning

The next section is where the device states are defined and their relative entities are provisioned.

Appliance types are the generic types that identify one from another, i.e. Garage door, Washing machine, TV, Coffee maker, Dryer, etc.

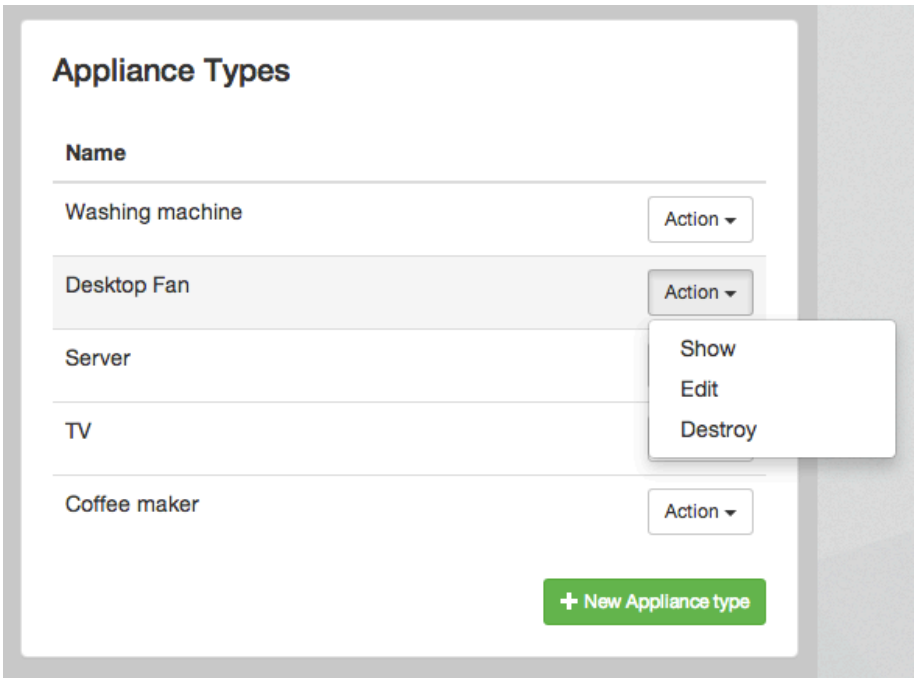

Figure 2: Appliance types provisioning

Vendors are the manufacturers of the appliances that digital skins will work with.

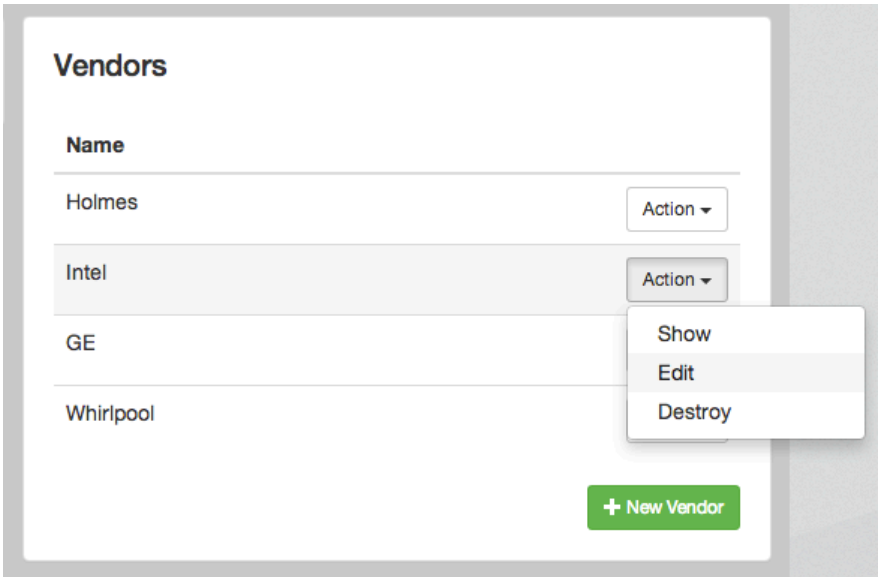

Figure 3: Vendors provisioning

Profiles are the sets that combine an appliance type, a vendor and an identifying model number of a specific appliance. It is important to differentiate between one appliance and the other, even within the same type, because not all same type appliances have exactly the same characteristics. This will also make it easier for the user to search profiles by vendor or by type.

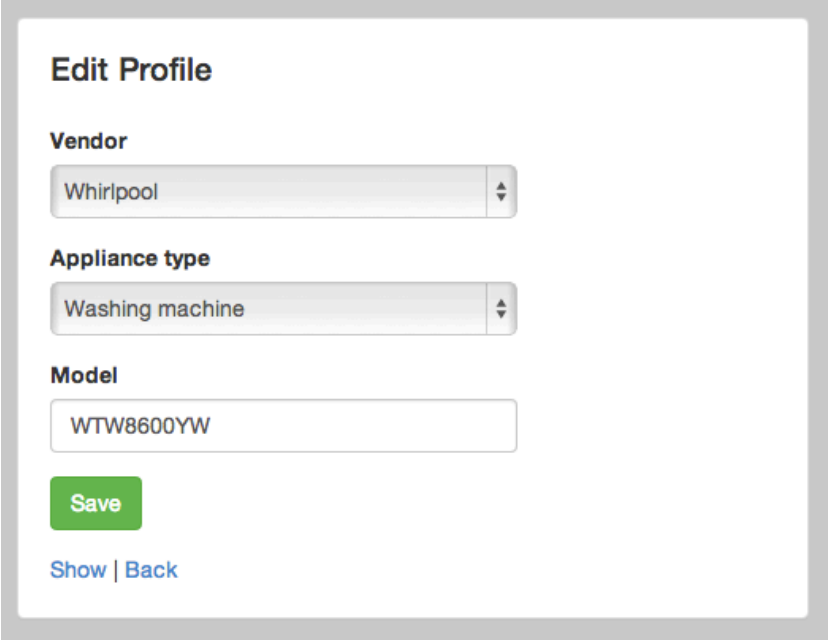

Figure 4: Profile provisioning. The admin can select from pre-existing vendors and appliance types

States are the different operational conditions that an appliance may be present in. They belong to a profile, which is the collecting set for a specific appliance type manufactured by a specific vendor and has an identifying model number. Each state has a message field, which is a human readable message that is used to communicate the current state to the user. The portal groups states by their profile to make it easier to manage.

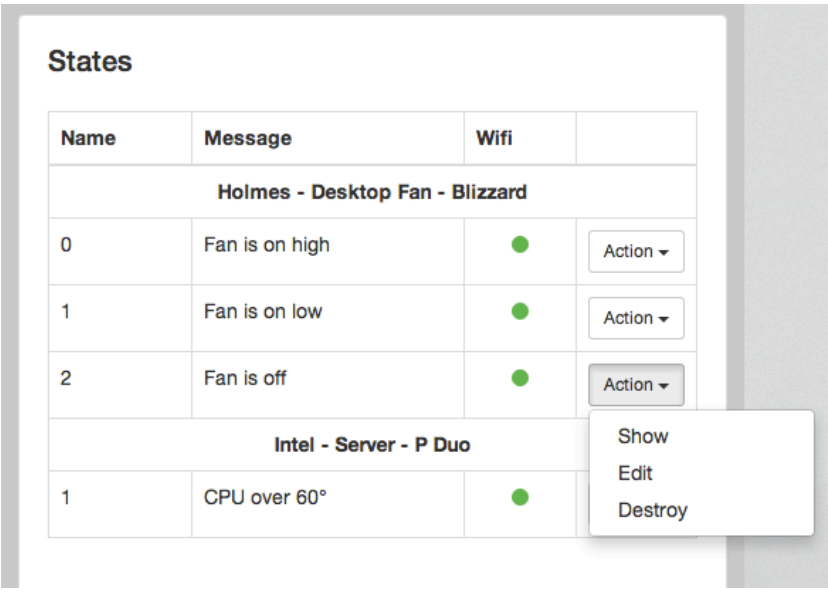

Figure 5: States provisioning

Each state contains information that dictates the digital skin how to operate, i.e. which sensors to use, how often to sample, what are the expected values and if the state should be reported to the cloud.

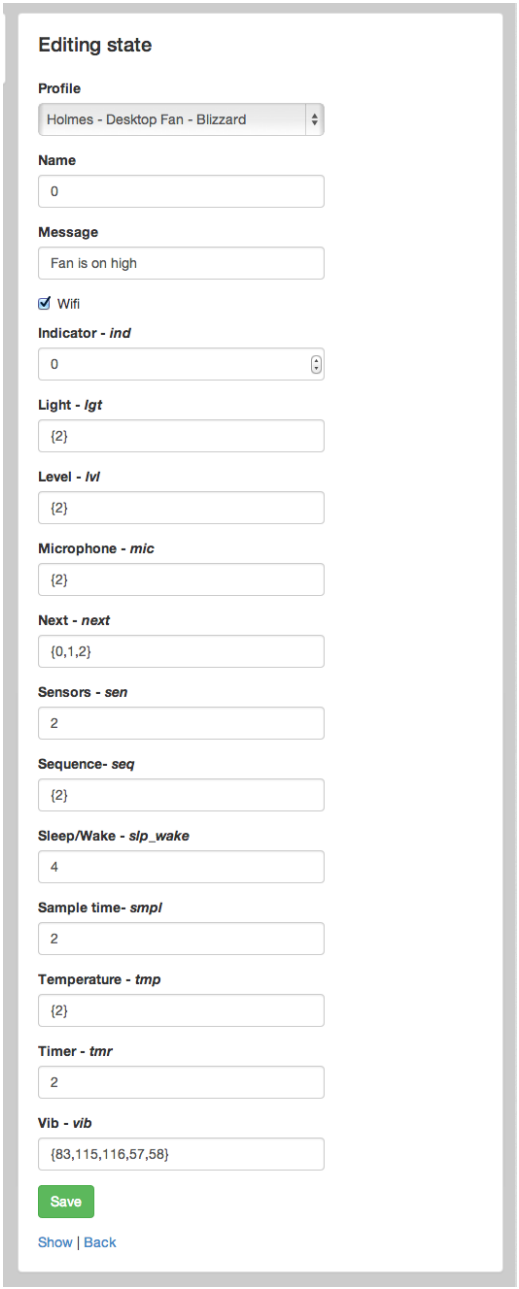

Figure 6: A state provisioning

Users create **Skins**. A user enters the UUID of their device; select one of the available profiles that match the current appliance to which the skin is attached. In case the device is already configured, the next time it communicates to the server the response code will indicate of an outdated profile configuration, and that the device will request the new set of states.

An Event is the state change that takes place on a digital skin and is reported back to the cloud. It has a timestamp for the time that it appeared.

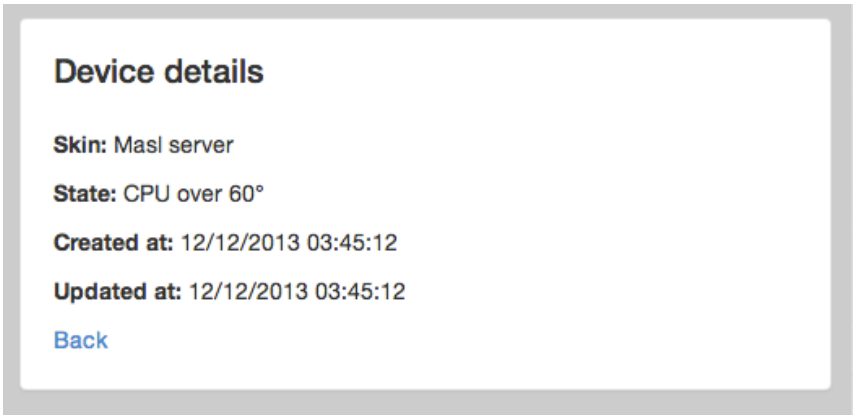

Figure 7: Event details

RESTful API

The API is intended to provide an interface for skins to communication to the server, as well as for 3rd party application to communication and fetch notifications addressed to them.

The portal employs OAuth 2.0 standard and acts as a service provider. OAuth 2.0 is the protocol used to authenticate and authorize requests for all API accesses.

Because skins were not integrated with OAuth 2.0 at the time of their creation, their interaction endpoints are temporarily left open.

Apple Push Notification System integration

The cloud portal also integrates with the state of art notification system that Apple developed. It allows sending specific messages to applications installed on iOS devices. Apple's notification system provides each application on a particular device a unique ID that is used to send messages. The iOS integration SDK makes use of this system, and provides interfaces to integrate the functionality in any iOS app that would like to integrate with the Skini portal. More information about APNS can be found on Apple inc.'s website [5]

Implementation

At a high level of architectural view (illustrated in Figure 8) the web portal provides two main communication end points. Administrators and users use computer or mobile web browsers to enter the portal and perform provisioning and management tasks. Sensor skins, mobile apps, desktop widgets and 3rd party authorized applications use the RESTful api to communication with the cloud. The cloud communicates with notification service providers, such as SMS, Email, Push notifications, etc., using their proprietary protocols based on specification provided by them.

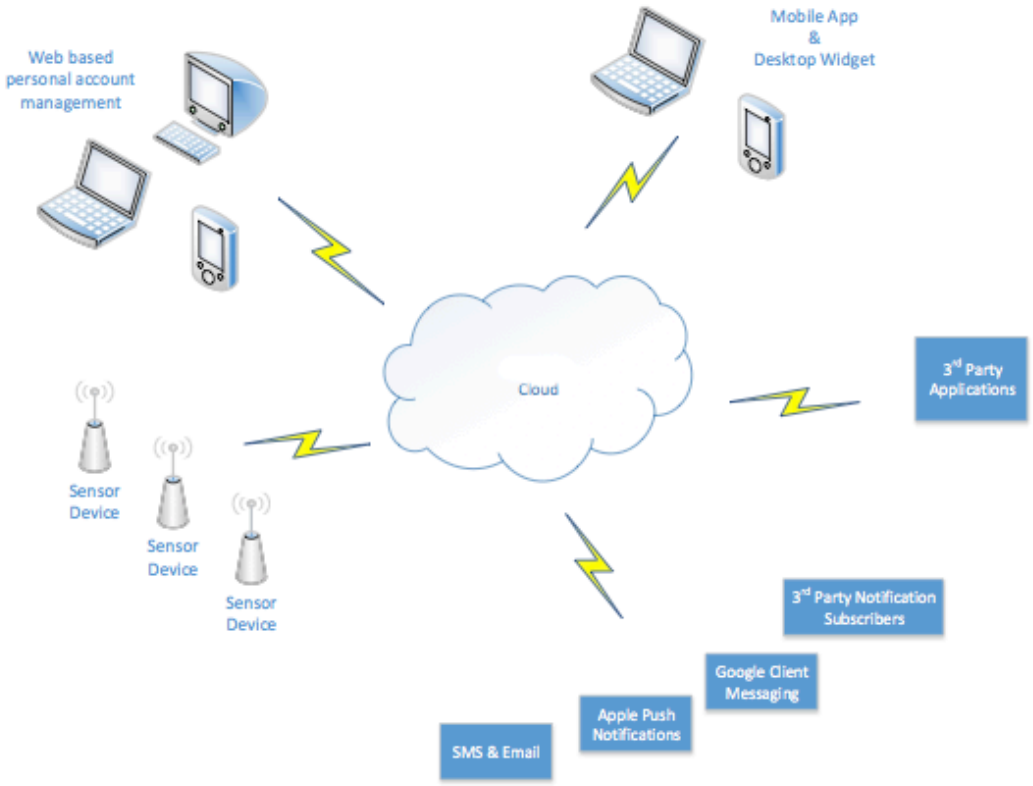

Figure 8: Architectural View

The web framework

The Skini portal is implemented in Ruby based on its famous Rails framework. Rails is built having agility in mind, and that is a very important factor in such projects where small increments will be made with the exploration of new opportunities. The MVC architecture, database migration capabilities, the large community and the availability of tutorial materials for a variety of proficiency levels were all considered during the selection of the framework.

The portal makes use of the following gems:

1. Doorkeeper: is a gem that makes it easy to introduce OAuth 2 provider functionality to rails application.[6] The gem is under constant development. At the time of the project it was based in the version 22 of the OAuth specification [7] and it still does not support all OAuth features.
2. Devise: Is a user management gem that provides user registration, session management capabilities [8]
3. APNS: Is used to interact with Apple Push Notification Service [9]

Database

The database that backs the web portal is Postgresql 9.2

The new capabilities that can be added to Postgresql databases using extensions made it very popular and deployed in large number of web applications.

The portal uses two main extensions of the database.

a. Arrays field: Allows storing of multiple values of the same generic type in one field.
b. Hstore: Allows the persistence of hashed objects into a single field and provides operators that make it easy to perform queries based on keys and values in a specific record. It is the schemeless approach of Postgresql.

User interface

The user interface was created using Twitter's Bootstrap 3 framework. The framework provides useful interface components and a grid system that makes it easy to create responsive web applications. [10]

Hosting

Skini portal is hosted on Heroku which is a PaaS built on top of Amazon Web Services. Depending on the future usage requirements, this will allow horizontal and vertical scaling.

VCS

The source code of the application is controlled using Git. It is always best to check the final version of the source code on the repository. For credentials please contact the author.

Results, Evaluation, and Reflection

The process of the project has been a great learning experience. Being new to the Rails framework and not being able to predict what the learning curve would look like, I limited the requirements to the most important basic features when choosing from many interesting ideas.

Rails community has proven itself with its continuous growth, forums, Google groups and availability of tutorial material for different levels of proficiency. The maturity of ‘gems’ (feature libraries) speaks highly of the maturity of the framework and its community.

Rails is built having the convention over configuration and is highly dependent on naming conventions to integrate different pieces of MVC together, therefore there is a lot of magic taking place behind the scenes. This could be an inconvenience when building custom behaviors. Since it was not the case in this project, Rails has been the right choice. All functionality were used with their conventions, therefore the framework helped focus on work instead of re-inventing the wheel.

The portal requires field-testing, which could only be possible by moving the digital skins from development to production level. With the deployment of pilot skins, it will be possible to collect feedback from users and improve the integration between different services that the portal relies on.

Conclusions and Future Work

1. Currently, in data collection mode, a set of numbers is collected for further analysis to define new states on the cloud. A new feature that automates the process could be added that observes continuous readings from the digital skin and intelligently extract distinguishing states as well as their sequences from the data. This makes adding of new models and appliance types to the portal easier.
2. Only the skins can initiate communication with the cloud. Currently it is not possible to send down commands at any time. This was a design decision driven by the fact that Wi-Fi usage comes with high power consumption and skins should only communicate whenever there is a change in states that needs to be communicated. This helps preserve longer battery life, which was one of the mian goals of the project. With the unveiling of new low power Wi-Fi modules intended for home automation and user electronics [11] this limitation could be revisited. A messaging system will be implemented to send control commands increasing the capabilities of both the cloud platform and the compatible devices connected to it.
3. The mobile framework provides an interface to authenticate and register on the portal with the user credentials. Some extensions need to be added to the SDK to allow apps to view the feed of events, user skins and other useful information.
4. Improve the user experience: There is always the need to improve the user experience on the web portal for both administrators and users to make it easier and comprehensive.
5. The portal currently integrates with Apple Push Notification System. Other mobile operating systems should be considered, especially Google Cloud Messaging for Android.

Bibliography

1. GSMA; *Vision of Smart Home. The Role of Mobile in the Home of the Future*; http://www.gsma.com/connectedliving/wp-content/uploads/2012/03/vision20of20smart20home20report.pdf
2. IEC (June 2011)- *Standards for intelligent homes* http://www.iec.ch/etech/2011/etech_0611/wld-5.htm
3. GE Appliances - *GE Brillion™ Connected Home FAQs* http://www.geappliances.com/connected-home-smart-appliances/brillion-appliances-faqs.htm
4. Revolv: *SMART HOME AWESOMATION ON YOUR SMART PHONE.* http://revolv.com/
5. Apple: *Local and Push Notifications* https://developer.apple.com/notifications/
6. DoorKeeper https://github.com/applicake/doorkeeper
7. The OAuth 2.0 Authorization Protocol draft-ietf-oauth-v2-22 http://tools.ietf.org/html/draft-ietf-oauth-v2-22
8. Devise gem https://github.com/plataformatec/devise
9. APNS gem https://github.com/jpoz/APNS
10. Twitter – *Bootstrap* http://getbootstrap.com/
11. Qualcomm: *Press Release; Qualcomm Unveils Low-Power Wi-Fi Platform for Major Home Appliances and Consumer Electronics* http://www.qualcomm.com/media/releases/2013/09/06/qualcomm-unveils-low-power-wi-fi-platform-major-home-appliances-and

Appendices

Appendix A: DB structure of Skins constructing entities.