Ambient Home Server: Location Aware Home Automation

Cameron Calka
Grand Valley State University

Ambient Home Server: Location Aware Home Automation

By
Cameron Calka

April, 2014

Ambient Home Server: Location Aware Home Automation

By
Cameron Calka

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

April, 2014

____

Jonathan Engelsma **April, 2014**

Table of Contents

Abstract

Currently in the realm of home automation, there are two primary approaches. One approach creates independent devices which can be controlled by a smart phone or send notifications when its task is complete, such as door locks and washing machines. The other approach has devices with sensors on them which attempts to determine if people are around before performing an action, such as the Nest thermostat and the WeMo motion sensors. The goal of this project is to merge both of these approaches together by using a smart phone to create a location aware device which can inform the home of where people are located. This project requires a Java server and MariaDB to handle interactions between various sensors and controllers, sensors built with off the shelf parts to monitor the home and an Android application to inform the server of its location, as well as view and control sensors inside of the home.

Introduction

Home automation and ubiquitous computing has been the dream of many computer scientists and makers. With the advancement of the internet of things, it has become easier to communicate with various devices inside of the home. Mobile phones have also become our personal communication devices that we carry with us the majority of the time. The presence of these phones gives us new options on how to communicate with devices in our homes, control devices as well and give our homes data about us. By letting our homes become more aware of our activities and preferences, the home itself can make better decisions about how to behave.

The development of home automation so far has not been in a unified manner. Unlike the internet, which has W3C to manage the development of new technology and create standards, Home Automation has no central group to manage the interaction between different groups. This means that each company uses its own mix of proprietary and universal standards to accomplish their device's goal. In order to manage this problem, one of the first goals is to create a centralized server to handle the interaction between different devices.

Background and Related Work

Currently in the area of Home Automation, there are many proprietary systems that have been developed. Companies such as Belkin, ZigBee, X10, Nest and GE have developed their own systems for home automation and how it should work. One of the greatest issues with each of these systems is that they don't communicate with each other in any fashion, requiring a user to only invest in a single system or setup multiple controls for what they want to do.

One project in particular, Nest, attempts to accomplish what this project does. The Nest Thermostat has a series of activity sensors to determine if users are inside of the house and a wifi connection to check the weather outside. The sensors of the Nest Thermostat are limited though, especially in a multi-zone building where activity may not be taking place in the proximity of the Thermostat itself. Currently there is no way to add more sensors from other sources or to share the Thermostat's sensors with other devices.

In addition to the physical controls that most of these systems have, many of them also have phone applications as well. Those applications though are only for the purposes of viewing the status of objects in the house (power switches on/off, current temperature) and changing the state of those objects. Few, if any, of them use any sort of location information to determine rules on how objects in the home should be working. Since the smart phone has become a popular device that many adults have, it also becomes the perfect device for tracking the location of a user and for controlling home automation.

Program Requirements

The first objective will be to create custom sensors from off the shelf parts. This is mostly due to the variety of tasks that the sensors will need to do, as well as having room for future expansion. The basic requirements of the sensors are to be able to monitor temperature, humidity and proximity. At the same time, the sensors need to communicate over wireless. The sensor must also be able to act as a iBeacon in order to help determine a user's position inside of a building.

Since this project will be using custom built hardware, as well as hardware constructed by vendors, the system will require a central server in order to handle communication between all of the various nodes. This server should also store sensor data and preferences in a database. The sensor data is stored so that the algorithms can be run upon it at a later time to improve the actions of the server, as well as to audit actions taken in the process of managing the home. Communication should be done via REST, as it is not a language specific protocol and allows for the use of HTTP for communication.

After sensors are created and sending data to the central server, an Android application will be developed. The Android application has multiple purposes. Like most home automation applications, it is important for the user to be able to view current sensor data as well as the ability to manually control nodes inside of the home. In additional feature will send the user notifications when the server detects something happening that the user should be aware of. Most importantly to this project though, the application will gather data from the phone and send it to the main server.

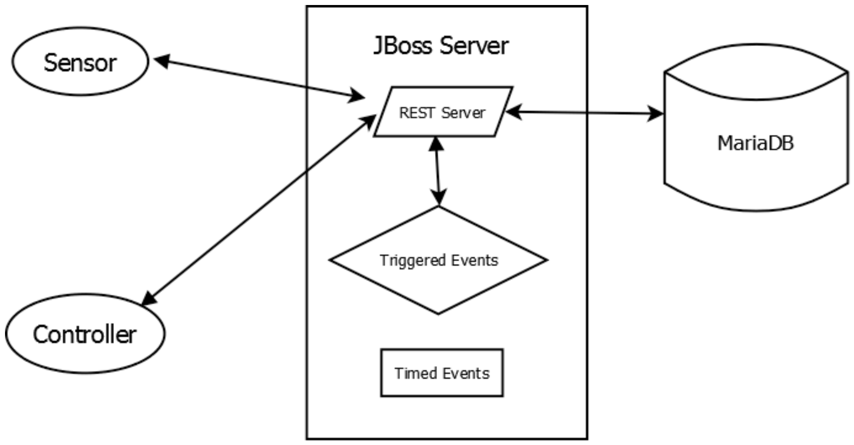
Implementation

The first step of this project was to create a number of sensors to place inside of my own home. I chose to create the sensors devices from raspberry pi devices with a number of standalone sensor components such as a DHT-22 temperature sensor and a motion sensor. The raspberry pi was chosen because it has additional computing power over a device such as arduino, though it is not as efficient with energy. Each of these devices was to be plugged into a wall though, so power requirements were not as big of an issue.

The raspberry pi also has the benefits of two USB ports on the device. This allows for the easy addition of a wireless communication module as well as a Bluetooth 4.0 LTE module. The wifi module is needed for communication, while the Bluetooth module is needed for iBeacon support. For each of the sensor nodes, python was selected to be the programming language used. Python was selected because it allowed access to low level hardware, such as the GPIO pins on the device. It also had easy to implement libraries for HTTP connections, which would be needed to send data back to the main server.

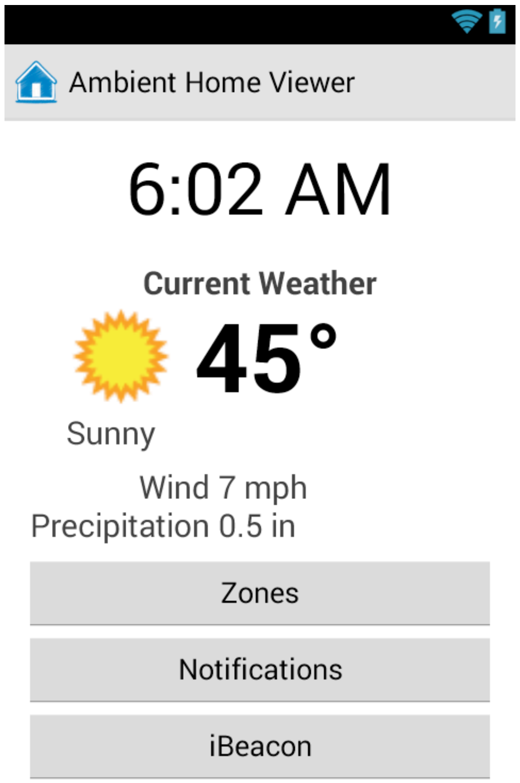
I initially expected this to be the most challenging part of the project. I have never learned about electrical engineering, so my first step was to dive into a number of tutorials about how to use the raspberry pi and similar devices to interact with attached modules. The best resource I found for this was Adafruit's tutorials for the Raspberry Pi. A number of their tutorials gave me the stepping stones I needed for this project, as they showed how to hook up a temperature/humidity sensor and gather readings from it. They also explained how the various GPIO connectors worked and how they should be used. I was then able to take this data and create my own implementation. I still ran into a number of problems with my devices, from simple things such as having a resistor put in backwards on the protoboard to frustrating problems such as certain connections just not working due to the placement on the protoboard. Rewiring the project on a different spot on the board caused it to spontaneously work correctly.

Once the sensors were working, the next step was to create a central server to coordinate the movement of data. The main goal of the central server is to facilitate communication between devices, regardless of which company the device was produced by. A secondary goal is to store data gathered by the sensors in order to later on run statistics or develop a data model on how the house should behave. Since my profession is that of a Java developer, I decided to use a java infrastructure for the backend. JBoss was selected as the server due to my prior work experience with it. MariaDB was selected for the database layer. MariaDB is fully functional and free database that has a number of enhancements over the popular MySQL database. As much as possible, I wanted to eliminate problems with new technology in the infrastructure layer. Due to my prior experience with JAX-RS, jdbc, EJBs and the Quartz scheduler; I felt that using these familiar industry standard technologies was better than developing new technology from the ground up.



Even though I was using technology similar to what I used at work on a daily basis, I still spent a signification amount of time debugging a number of problems. JBoss does not have an official MariaDB module, so a lot of time was spent figuring out how database connections worked in JBoss and how to hook it up to the database layer. I also found a number of JBoss tutorials subpar for teaching someone how to create a certain type of project. They do have a number of in-depth tutorials to show a developer how the REST framework is used and how to send data across EJB layers… but I found that many aspects of the documentation were missing if the developer wanted to create a similar project from the ground up and not from using one of their code-packed tutorials as a launching point. Sometimes something as renaming the project in Eclipse would cause the REST service to no longer be recognized. In retrospect, I would recommend spending more time investigating other light-weight frameworks for this process. JBoss is mainly used as an enterprise application server and as such requires a higher level of knowledge to work correctly.

After struggling through building the central server, the next part was to start development of the Android application. Android was selected to be the mobile operating system because I had already taken a class at GVSU that dealt with mobile development on Android. From that I had learned the ease of developing the applications UI and using REST calls to transfer data back and forth between the phone and server.



Ambient Home Viewer

6:02 AM

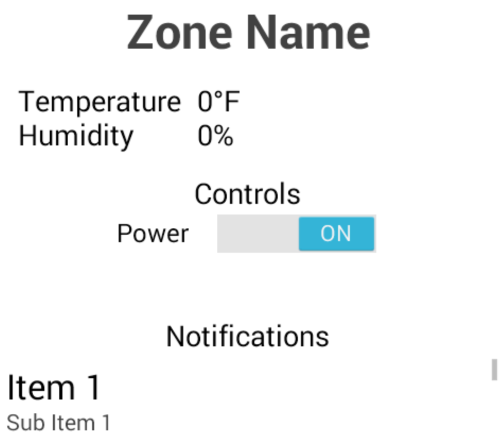
Current Weather

45°

Sunny

Wind 7 mph

Precipitation 0.5 in



Zone Name

Temperature 0°F

Humidity 0%

Controls

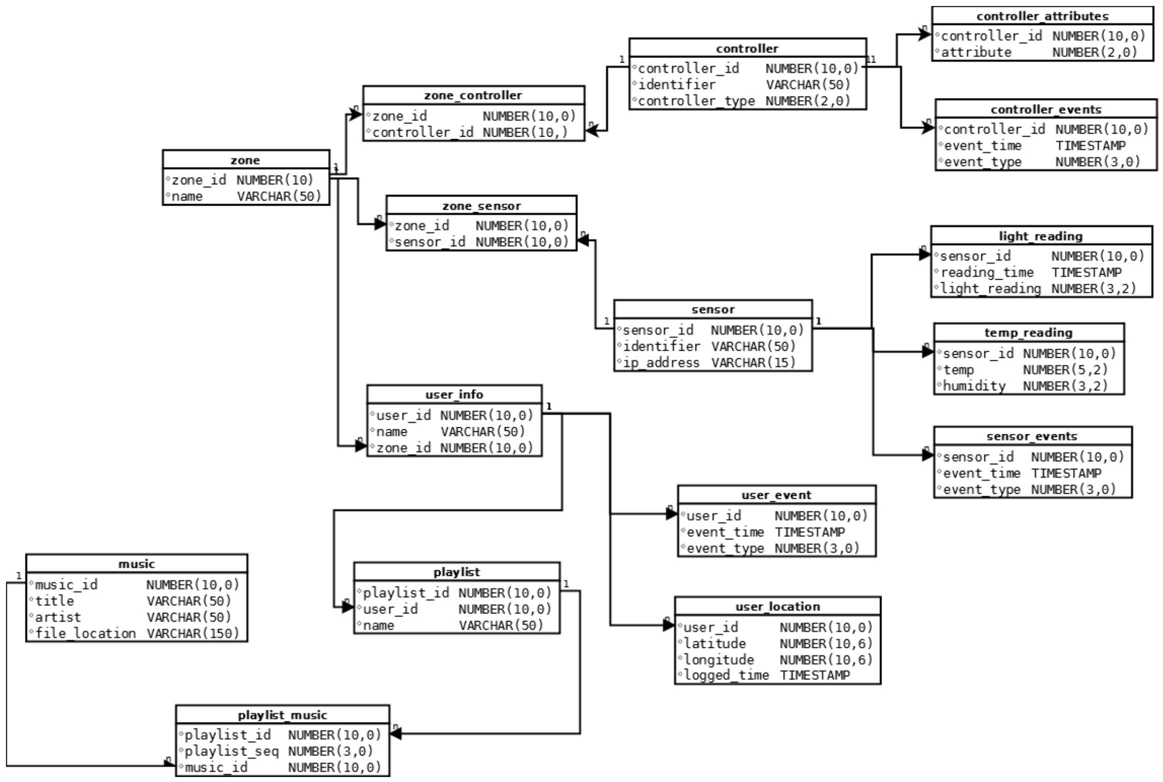
Power ON

Notifications

Item 1

Sub Item 1

At first the Android application was only developed in order to display data from the home and be able to control certain aspects of the home. A home would be broken up into a set of zones, usually one zone per room. Sensors and controllers would then be assigned to each zone. By bringing up the appropriate zone in the application, a user could see current data such as temperature and humidity in the room, as well as turn off power sources.



While researching additional ideas to make the phone more useful, I stumbled upon iBeacon. iBeacon is an indoor positioning system that uses low power Bluetooth. By determining the signal strength of the Bluetooth signals, the device is able to determine how far away from a “beacon” it is. It recently has been deployed to the GVSU library and a number of super markets. Upon discovering this technology, I decided to showcase it inside of my project as it was the missing link between how home automation currently works today and how it should work in the future.

In the Android application, a background process will update the central server with the user’s information using both Bluetooth and GPS. When the central server receives this information, it will make a determination on which room the user is currently in and what it should be doing. This will be based upon a pre-defined set of rules stored on the server. Such rules would be “how long after a user leaves a room should the lights stay on” or “proximity sensors should trigger notifications if the user is not in the house”. This is done by registering each Bluetooth signal inside of the house on the central server. The phone will then do a quick scan and send the list of adapters it finds back to the server. A basic version of the server’s algorithm would assign nearest zone based upon the closest distance to a particular sensor, but an advanced version would attempt to triangulate the sensor signals from all of the available sensors. This would further increase the utility of iBeacon as you wouldn’t need one in each zone.

Results, Evaluation, and Reflection

The overall project was a success. The goal was to create a location aware home automation server, and that was accomplished. I think the major faults are not in what was accomplished, but what more could have been accomplished. Due to supply problems of certain hardware devices, building the sensors was pushed back and other devices didn't become available until late April. If I was to do this again, I would procure all of the hardware needed before this project began, instead of anxiously waiting to see if it would become available.

I also feel that setting up my own infrastructure to host the central server was a mistake. A similar server could have been setup using Google App Engine or a similar service. One of the reasons I made the choice to self-host a server was to remove a layer of fault in controlling the home. The house shouldn't stop working if Google servers suddenly go down, as the central server is located inside of the house itself. Even if external internet were to be disabled, the local network inside of the house would still function properly. While all of these are benefits that I think need to be argued for, the flip side of the coin is that I spent weeks getting the infrastructure to work myself. More than once I had to revert a nights worth of work because an erroneously changed setting caused the server to no longer communicate properly. This was mostly due to my own inexperience with being a sysadmin and having to create the infrastructure from the ground up.

Conclusions and Future Work

This project was one of the main reasons why I pursued a Masters at GVSU. I wanted to take knowledge of many different areas (databases, machine learning, distributed computing) and be able to apply them all to a single project. The goals for my future work will be to further use each of those areas to make this project better. One constant goal will be the integrate more types of sensors and controllers into the central server. These would range from Belkin power devices to Rocki music controllers, allowing the end user to use whatever devices they already own to setup a home automation system.

One of the goals of this project is to allow most people to use this, a user-friendly GUI will have to be created. This will allow the user to configure rules and settings on the server, add devices to the network and create the zones for their home. This would also allow them to create users for the home and set the user interaction rules, which governs how the home should react to a user entering and leaving a zone. Should music follow the user through the house? Are certain areas quiet zones that music should not automatically play in (a bedroom for example)? All of these rules are currently hardcoded in the server and need to be dynamic.

Other than the phone or other mobile device, other ways to interface with the end user need to be explored. The one that I think would be most beneficial is voice activation. If the user can speak to the room, even to follow basic commands such as "lights off", it will lower the burden of using a home automation server significantly. While many people constantly carry their phones with them at all times, they are frequently in our pockets or we are in the middle of doing something else. Telling my home to set a timer for 15 minutes while I prepare food eliminates the need for me to stop what I'm doing and allows the home to work for me.

I think the realm of home automation is very exciting and will become even more so in the upcoming years. People have personal computing devices with them at all times, allowing them to integrate their lives with technology. I think integrating their home with the same level of technology will spark ideas of all sorts. Already devices like the Xbox One are attempting to use voice activation to drastically improve the home theater experience, while other devices attempt to increase energy efficiency by turning off electrical devices when not in use. I think technology has a place in the home to make the lives of everyone easier and we are only at the cusp of discovering its potential.

Bibliography

[1] "MariaDB," [Online]. Available: https://mariadb.org/.

[2] "JBoss," [Online]. Available: www.jboss.org.

[3] "Adafruit - Raspberry Pi Tutorials," [Online]. Available: https://learn.adafruit.com/category/raspberry-pi.

[4] "iBeacon Library," Radius Networks, [Online]. Available: http://developer.radiusnetworks.com/ibeacon/android/.

[5] "iBeacon," [Online]. Available: http://en.wikipedia.org/wiki/IBeacon.

[6] "Raspberry Pi," [Online]. Available: http://www.raspberrypi.org/.

[7] "Android Developers," [Online]. Available: http://developer.android.com/index.html.

[8] "Welcome to Python.org," [Online]. Available: https://www.python.org/.

[9] "WeMo Home Automation - Belkin," Belkin, [Online]. Available: http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/.

[10] "Nest: Home," [Online]. Available: https://nest.com/.