Home Power MAP

Final Project Report

Alain Bustamante

Kevin Herrera

Yasamin Nouriboshehri

Steven Prsha

Joe Waisman

Sponsored by:

San Diego Gas & Electric

Submitted to: John Kennedy and Lal Tummala

Department of Electrical and Computer Engineering

San Diego State University

May 2013

Table of Contents

Abstract

The future of power use inside the house looks bleak for San Diego home owners. By the summer of 2013, SDG&E predicts to have the highest tier rates for electricity use in San Diego County. How does one deal with such a power crisis? Home Power MAP has strived to help SDG&E and its customers by providing a web GUI and an iPad application to help save money and reduce power consumption. Ultimately, the current system allows access to statistics about the consumer's electricity use as well as the ability to monitor and control of their home's electrical appliances.

Introduction

Within one day, power consumption is inconsistent and unpredictable, which may cause problems for SDG&E in the long run. Mainly, the sharp increase in power usage makes the current system more susceptible to power outages. With the rising availability of electronics and, especially, electric cars, San Diego County faces power crisis during peak times. While SDG&E has provided a way to monitor home electricity usage online to its customers, the system is not complete; the current system only keeps tabs on the total power consumption for every hour. In addition, this data is provided with latency. Home Power MAP has developed a software system in hopes of educating the consumers as well as being able to predict electricity usage patterns to remedy this issue. The system is designed to poll electricity usage of different devices in a household such as the smart outlets, HVAC, and solar panel system; three different teams focused on the custom design of aforementioned hardware. Power MAP software is responsible for being the common point for these systems in order to collect data, send messages, and receive reports as well as manual overrides.

Home Power MAP's Solution

The system has three user interfaces: a web-based application, an iPad application, and a utilities application for SDG&E. All of these interfaces will, in essence, have access to the customer's usage data. However, each interface is designed for a different purpose. The web-based application is made for the user to have the most control over—to be able to see detailed data and customize settings for their home power usage. The iPad application is made for ease of use and portability, but it is designed more or less the same way as the web application. Both of these interfaces are intended for customers. The corporate interface is a more technical interface meant for the SDG&E utilities use. While this utility gets the same data as the user, it gives the user the ability to analyze and predict future power consumption patterns as well. Home Power MAP has achieved this by making efficient use of the Python scripting language to act as a bridge between the software and hardware teams.

Python Bridge

The heart of Home Power MAP is the Python Bridge. It consists of scripts that allow our software to communicate with the various hardware nodes. Our python code will query data from the various nodes, as well as send commands to have devices do a specific function such as turn off, on, raise the temp, etc. Our Python code can receive information from the hardware as well. The various nodes can send an acknowledgment back informing our system that a node has completed a task such as turning on, as well as sending a report with specific information regarding a query asking for information. Once information has been received, the data runs through a parser to extract the specific strings which can be saved in the database.

In the early weeks of the semester, a common set of variables was assigned in order to allow communication between the software team and the hardware teams. This

information was stored in a COMM document, which was distributed to all other teams. Once the teams agreed on the document, we began to build a Python Parser. While Python is an efficient language for scripting, it poses challenges when multiple processes are required. In the early stages of building the Python Parser, we ran into a couple issues: The first issue was that Python is space-sensitive, meaning that if a block of code is not indented with the exact amount of space, then the code would not execute properly; this became very frustrating when multiple spaces were incorrect. Second, Python did not like nested if-statements. If there was more than one if-statement in the code, it would read the first statement and if the conditions were not met, Python would skip all the other if-statements and jump to the else-statement. We mitigated these issues by searching the Internet regarding our issue which led to the use of if-else statements. As more information was obtained about the limits of the hardware teams microcontrollers was known, the COMM document was revised and so was the parser itself. After about a month, the kinks were worked out. Our parser was finished which contained fifteen different files, each file with a very specific task.

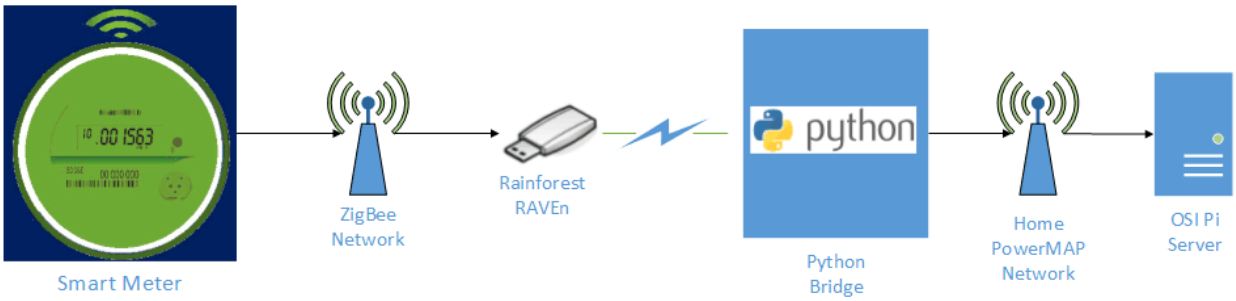
Our next task was uploading the data received from hardware to the various databases. Although our parser worked well when testing each file individually, the issue of integration became self-evident; on the other hand, the parser was not as efficient as we would have liked. Home Power MAP eliminated the original parser because it yielded too much overhead. We redesigned a parser using C#, which allowed us to create executable files to run a cohesive parser continuously. This new parser could receive specific data and upload the data to the databases without any issues nor without any extra overhead.

Python and Communication

The Python Bridge is used in two major applications of our design: a) the smart meter real-time data streaming, and b) sending commands and receiving data on a long-term basis. Each utilizes a different facet of Python to essentially send a variety of data from the hardware level to the user interfaces. This data includes but is not limited to: electricity usage, price of usage, status of hardware, inside temperature, outside temperature, etc.

The team had to design an optimal path for the information to stream data from the smart meter to the OSI Pi server. The following diagram shows this general flow of data:

Figure 1. Flow of Data from Smart Meter to OSI Pi Server



But before implementing this design, our team needed to understand how the smart meter works in the first place. The smart meter is mounted on top of a warmer box to draw a variable load. The load can be changed on the warmer box itself with a knob. The smart

meter, then, displays the instantaneous energy being used in kilowatt hours, and this data point is a function of the load. The instantaneous energy usage is then beaconed approximately every five seconds via the meter's ZigBee radio protocol. In order to capture these transmissions, we have paired a Rainforest RAVEn ZigBee dongle with the smart meter. This dongle can display the meter's usage, but this information is not very readable or user friendly. Finally, Python is used as a bridge: It extracts this data from the RAVEn's serial port, creates a connection with the OSI Pi server, and ensures that the data is sent to the server. The data is then parsed before being inserted into a table on the database. Later sections of this report will go into more detail of what happens to streaming data after it has reached the server and how it will eventually be displayed on the client interfaces.

The Python Bridge also serves as the mediator between the custom hardware and the server. It has several different modes, which operate very similarly to each other: polling, sending and receiving data, and manual overrides. All information flow from Python to the nodes and back are formatted strings which have been standardized, and each team has agreed to comply and conform its code to it.

Polling data is the main function of the Python Bridge. It sits on a port at a designated IP address and sends queries every 30 seconds; these queries are driven by a C# executable code which will be covered in detail in later sections. After the queries are sent, Python then listens for an acknowledgement from the nodes on a different port. This acknowledgement string is then forwarded to the SQL server. Then the actual data concerning the query is sent back to Python, and that is also forwarded to the server. This loop continues infinitely until stopped manually; this helps the server generate enough data points in near real-time so that the client can be informed almost instantaneously, for example, when or if a specific appliance in the house is using too much electricity. It should be noted that while the user sees the results of polling, he or she does not directly have control over the polling process.

Python Bridge also sends and receives manual commands and data. These commands actually originate from the user interface: A couple of examples include toggling the on/off switch of a smart outlet or changing the temperature of the HVAC system from either the web or iPad applications. Python is responsible for sending the right commands to the right nodes according to their IP address. Much like the polling process, when commands are sent to each node from Python, each node kickbacks an acknowledgement. Again, these acknowledgements are sent back to the server. After the nodes have processed these commands, they send updated information to Python, which then gets sent to the database after being parsed properly.

Last but not least, the Python Bridge receives manual overrides from the devices themselves. If the user chooses to control the device directly from the hardware instead of the software interface. Python sits on a port that is designated for manual override commands and whenever those commands are performed (i.e. a push button on the smart outlets is pushed or the temperature dial on the HVAC system is turned), Python receives a string and sends the data to the SQL server to be parsed. It should be noted that manual override commands override any polling command from the server.

SQL and OSI Pi Databases

During the initial stage of the SDG&E home power monitoring and control project, we knew that some form of database was needed to store and access data. At the time that we started to lay the foundation of what is now known as Home Power MAP, the project manager was very familiar with MSSQL (Microsoft Structured Query Language). Knowing how to setup a server and having computers connect to it, it was a clear choice to use. The team was also aware that an OSI PI server, explained later on in this document, was a database that SDG&E wanted to incorporate. Due to licensing issues, the server could not be accessed until closer to the end of the semester. For this reason, part of the project was designed with OSI PI in mind while most of it used SQL. Having a project with the ability to connect to both servers actually provides a more robust system. There are advantages for each server and having access to both allows us to choose which one to use when it is most convenient. For instance, since SQL stores data in tables, it is easy to correlate more types of data, while OSI performs calculations much quicker in a larger scale. How and what data is accessed on either server will change depending on future revisions of this project.

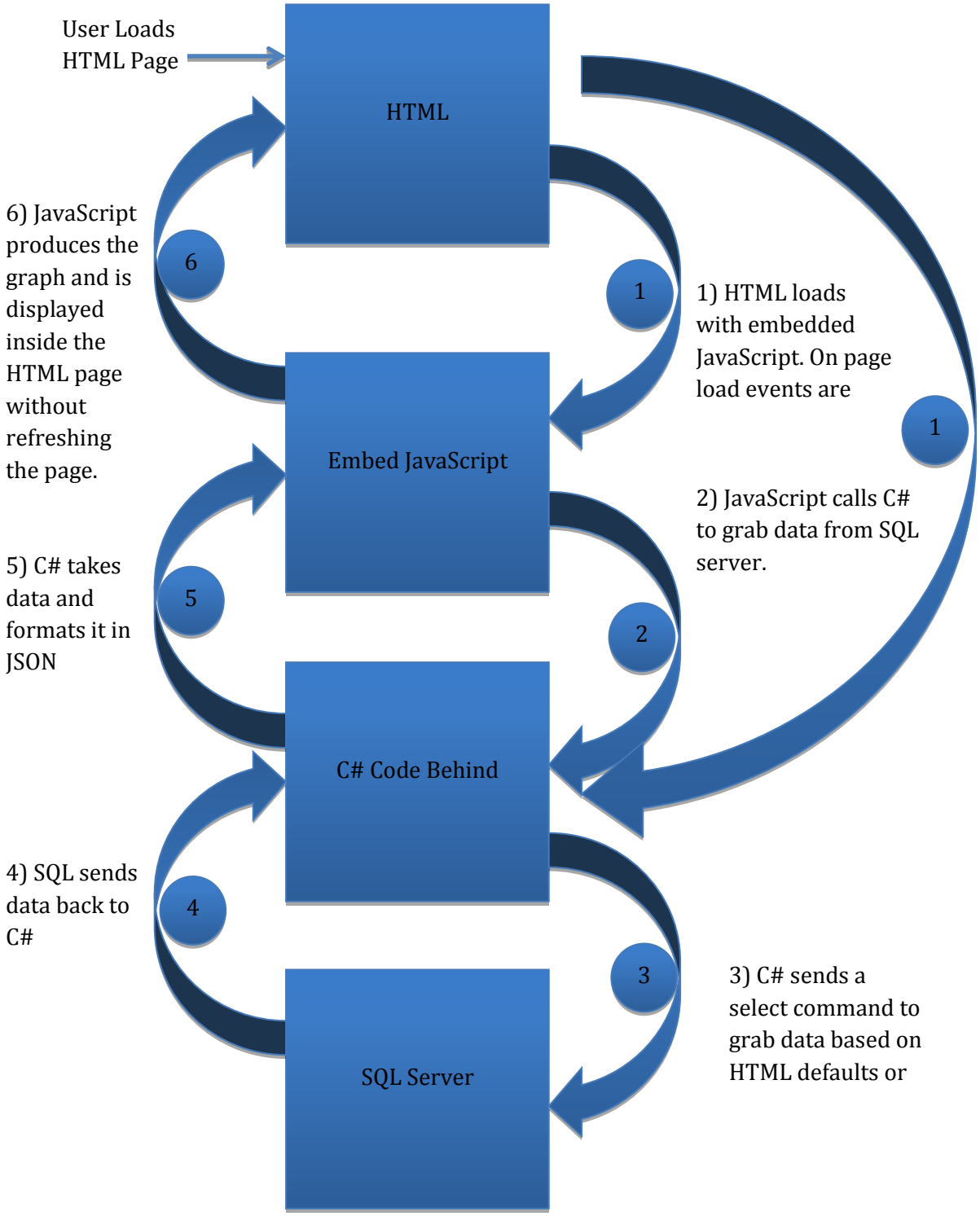
SQL serves as the main database to the Home Power MAP project. It stores data coming from all the nodes—the HVAC system, smart meter, smart outlets, and solar panels—in table formats that can easily be read via C# code. This data is then manipulated to produce graphs and cost analysis such that a user can easily view and understand the data. The exact data point stored varies from device to device, but they all share certain crucial information such as power consumption (kWh) and timestamps.

Since OSI PI is the server of choice for SDG&E, it would make sense that a project sponsored by them would utilize OSI PI in one form of another. Integrating this project with this database allows students who would otherwise be unaware of the power of OSI PI to learn and gain some sense of how it can be utilized for home automation and networking. OSI PI is quick, and it can actually be a better solution to the SQL server depending on its implantation. Obviously both databases have their distinct uses, but in larger scale environments, OSI PI is the obvious choice.

OSI PI is the secondary database used to store real-time data coming from the smart meter. The server is ran on a Lenovo ThinkPad provided by SDG&E. Given more time, OSI PI could have played a bigger role in this project. Because access to the server was delayed due to licensing issues beyond the group's control, we were only able to integrate OSI PI to receive and insert data incoming from the smart meter. In future additions to this project, OSI PI can serve a larger role as the main, or only, database server. For our project, data from the smart meter is inserted every five seconds and can be retrieved the moment it is saved. Graphing modules pull this real-time information and display it on our web interface and iPad application.

HTML Web Interface

There are three main tabs that were developed for the web interface. Since this project is envisioned to take place over multiple semesters, more pages will be added over time. Even though these three pages display the information requested, there are several references to other classes created to help retrieve and manipulate the data from the databases. The following is a block diagram of the data flow in the web interface. General concepts are seen and can be applied to any current or future pages of Home Power MAP.



"At a Glance" serves as a quick and easy way to view the most up-to-date information which does not require user input. The page is broken down into multiple sections where each is updated on page load with the most recent data. The first area is the line graph that updates data from 12AM up to the current hour. The next section displays the day, the power usage (correlated with the line graph), and the cost associated with that power usage. The third section deals with the tiered cost and the current power consumption (in kWh) from the smart meter. Two graphs are displayed: The left one shows the current tier cost within a twenty-four-hour clock; the needle points at the current hour where the background color reflects the cost. On the right, there is a gauge that updates every half a second. Whenever data is inserted into the PI server, it will be reflected on this graph. The fourth section is the month-to-date information. It takes all the power usage from the smart meter, adds a total, and shows the cost. Lastly, there is an empty temperature section and a "Did you know?" portion to reflect little bits of information that may be unknown to most clients.

Figure 2a. "At A Glance" Top Section of Web Interface

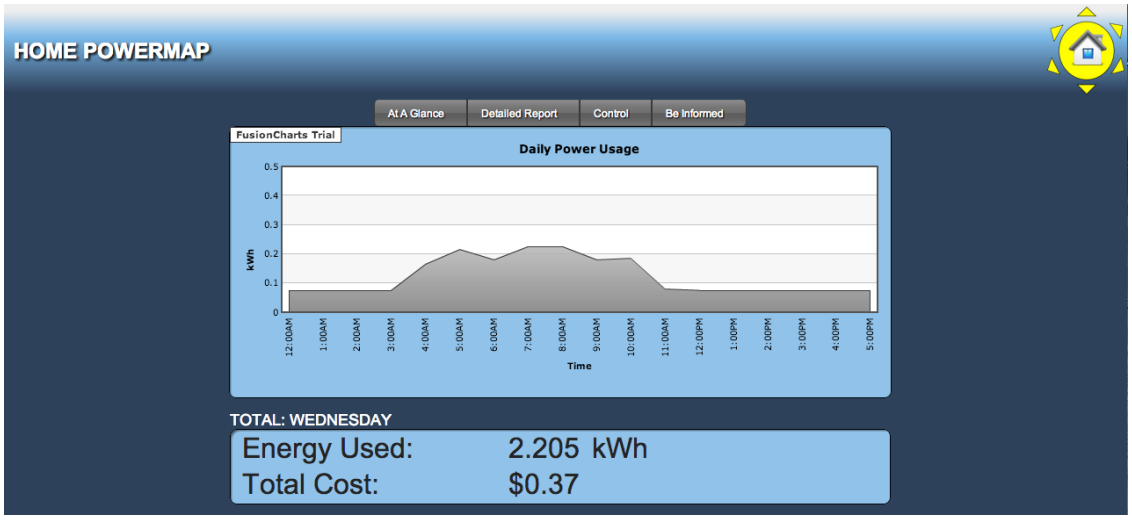
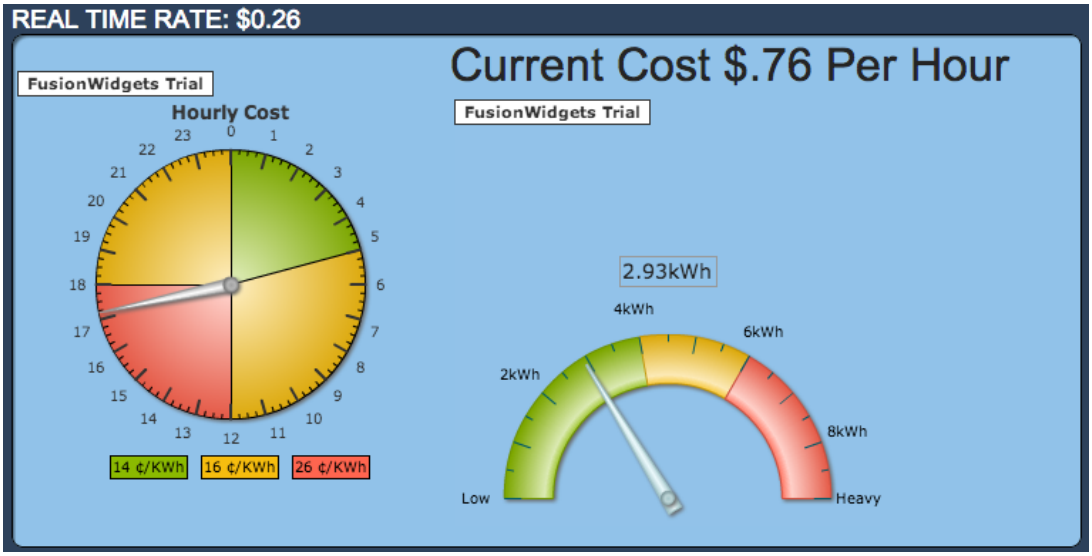
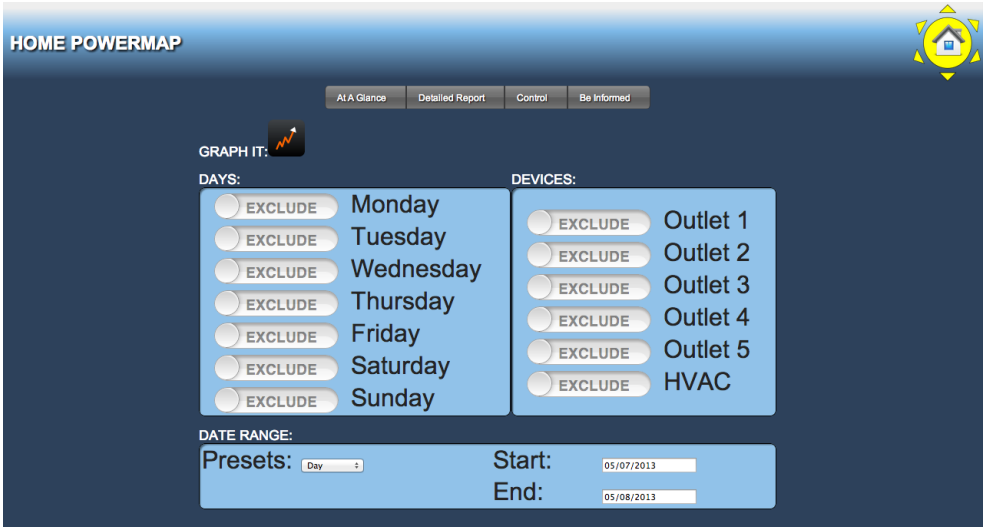


Figure 2b. "At A Glance" Bottom Section of Web Interface



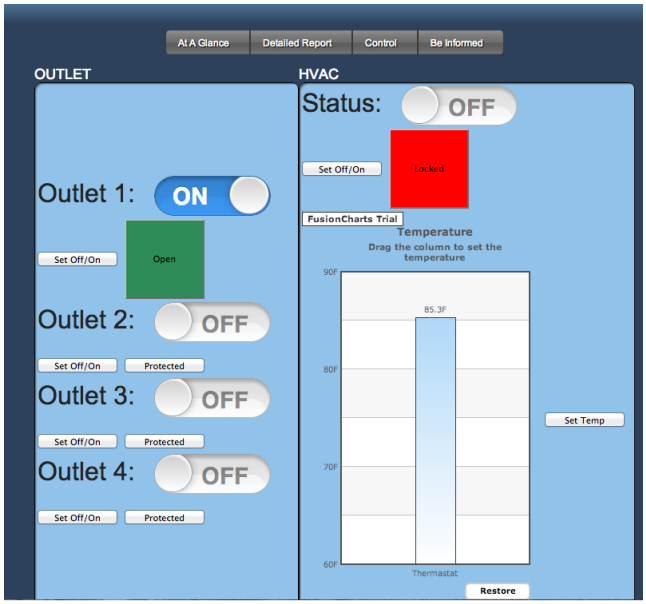
The “Detailed Report” tab will provide more control as to what data is viewed. The user has the option to select any or all days, a select date range, and any particular nodes. With these three inputs they can graph the power usage (in kWh) with each node adding an additional line to the graph. The cost is also displayed once the graph is populated.

Figure 3. “Detailed Report” Section of Web Interface



Along with displaying the power consumption and the cost associated with it, we needed to control the nodes; this is where the “Control Center” tab is used. We have a long list of commands that can be sent back and forth on a document called COMM. The “Control Center” will send a command to Python which will give a command to the nodes. For example, the command can be turning the node off or on, or in the particular case of HVAC, setting a certain temperature. A sliding graph is used to set the temperate and with the “Set” button, the web page will trigger the command to set the HVAC temperature to the specified value.

Figure 4. “Control Center” Section of Web Interface



Python Support Application

The Python Bridge does a wonderful job of handling communication traffic for the complete system. Due to its nature of being a scripting language, it does not handle processing the same way that conventional programming languages, like C#, do. It is for this reason that multiple C# executables were created to alleviate the processing requirements and be triggered similar to how a function is called.

SQLParse is the main executable used by Python. Python keeps track of the previous sent message and its latest received message throughout its process of data traffic. The moment any data is received, it will send the TX and RX messages as arguments to this executable. The data will then be parsed and filtered through case statements to understand what to do with this data. Depending on the TX/RX combination, it will either "Update" or "Insert" data into the database. In some cases, neither is performed but instead, just a print-line is executed.

SmartUpdate takes information from the smart meter and inserts the data into the OSI PI database. As data is streamed from the smart meter through the Rainforest RAVEn dongle, this executable is called to parse out the data. Since the information from the smart meter is sent in quick waves of five second, every single line is parsed. The data we care about is found between '<Demand>' and '</Demand>'. As an example, it will look like the following string: <Demand>0x10567</Demand>. Each line is searched for this string, and if found, it will parse out the data in hexadecimal, convert it to float, and insert it into the OSI Pi database. Most times the executable is called, it will not perform the update. It will be called an average of eight times without any insert and only insert once every five seconds due to the smart meter only providing data in that interval.

QueryNodes serves as a timer. At a certain interval, it will send a query message to the Python Bridge, which will then send the query to the respective nodes according to their IP addresses. In its current implementation, nodes will not response with report information until a query is made. This is the primary reason the QueryNodes executable is used. It will send query messages for both the outlet and the HVAC teams at some interval. At the time of writing this document, the outlet nodes are set to one second while the HVAC node is queries every thirty seconds. Note that this setup may change at any point. This executable does not take any inputs; it just serves are a timer to send query requests. At the top you will find all the variables that are hard-coded. The output message is in the form "message#IP," where # is the delimiter for the Python Bridge.

iPad App

Home Power MAP on iPad strives to offer SDG&E's customers with the necessary tools to better understand their power consumption behavior and take money saving actions. The principal goal was to design an interface that empowers the user to make well-informed decisions based on aesthetically pleasing real-time data representations from the convenience of their iPad. The iPad interface offers four principal views: The Dashboard view provides the user with the necessary real-time power monitoring tools while the HVAC, Smart Outlets, and Solar Panels control views offer both monitoring and system specific control capabilities. All controls and monitoring visuals were developed in

accordance with Apple's iOS Human Interface Guidelines. This was done to maintain an iOS native look and feel which users expect on their iOS Platform devices.

Figure 5. Dashboard View of iPad App

When launched, the iPad app welcomes the user with the Dashboard view. This view contains three simple yet helpful widgets. The first is a yearlong billing history chart. This allows the user to see whether or not they have made any improvements in lowering their utility bill throughout the year. The second is real-time data from the smart meter, which provides the user with instantaneous feedback on the current power usage being read from their home's smart meter. Roughly, a five-second delay exists between updates due to the Itron Smart Meter's internal sampling rate. By default, the iPad queries the Home Power MAP SQL Server every one and a half seconds to update the iPad's smart meter widget reading. Since SDG&E plans to implement Time of Use (ToU) tiered pricing in the future, a third widget was added to help orient the customer on this new pricing model. A twenty-four-hour clock divided into four distinct colored areas, each corresponding to a different pricing range, helps illustrate to the user the concept of ToU pricing. It lets the customer see how much they are paying per kWh at any time of the day. The intent is that by understanding that electricity costs different amounts of money at different times of the day, the customer will make more conscious decisions as to which appliances to use throughout the day knowing that electricity is more expensive during the evening and most affordable during off-peak hours.

The HVAC view offers controls that allow the user to operate their HVAC unit in one of three modes: AC, Eco and Heater modes. To visually indicate the current mode of

operation, a mode-dependent icon is displayed on the left-hand side. Temperature and ventilation settings are configured through the "Temperature Inspector" button. As shown in the figure below, clicking on the "Temperature Inspector" triggers a popover view with temperature and fan speed configuration controls. As a precaution, the user is prompted to verify any new settings before changes are made. Controls to open and close windows as well as turning the main unit on or off are also provided.

Figure 6. HVAC View of iPad App

Home Power MAP on iOS also provides a basic level of integration with the EMACS Smart Outlet system. Users are able to turn on and off individual outlets, either to eliminate phantom power usage or prior to leaving for a vacation. The figure below shows the EMACS Smart Outlet controls interface.

Figure 7. EMACS Smart Outlet Control Interface

Users interested in monitoring their solar panel's health and performance can do so from the Home Power MAP Solar Panels controller view. Apart from listing each solar pane's on/off state, power generation and health status, the Solar Panels controller view also allows the user to tap on any one of the blue disclosure buttons to show further details. An example is shown in the figure below. By tapping on any of the blue disclosure buttons, the user can find out about solar panel installation's Solmetric SunEye solar access measurements, energy storage, current, and total power generation.

Figure 8. Solar Panels Controller View

iPad – Server Interface

The ipad_real.aspx file provides the dynamic content needed by the iPad application. The application uses C# to write data to an otherwise empty console. This is optimal since the iPad application relies on raw, ASCII data. Any HTML or XML wrappers will cause it to fail.

Communication is initiated by the iPad with its data needs specified in URL escape codes. The URL escape codes for ipad_real.aspx are listed in Table 1.

Table 1. iPad URL Escape Codes

Option	Description
?query=chartMaster &startime= __&endtime= __	Returns historical data for the time periods specified
?query=getRate	Returns present hourly cost per kWh
?query=getHourlyCost ?query=getMonthlyCost	Returns present hourly/monthly cost based upon real time energy consumption.
?query=TimeD	Returns present system time
?query=gethvacstatus	Returns "on" or "off" mode status for the HVAC
?query=gethvacprotected	Returns "protected" or "unprotected"
?query=gethvactemp	Returns HVAC temperature
?query=sethvacon ?query=sethvacoff	Sets HVAC status to on/off and returns HVAC status for verification
?query=sethvactemp&temp=89	Sets HVAC temperature and returns temperature status for verification
?query=sethvaclock ?query=sethvacunlock	Enables or disables protection

Corporate Utility Application

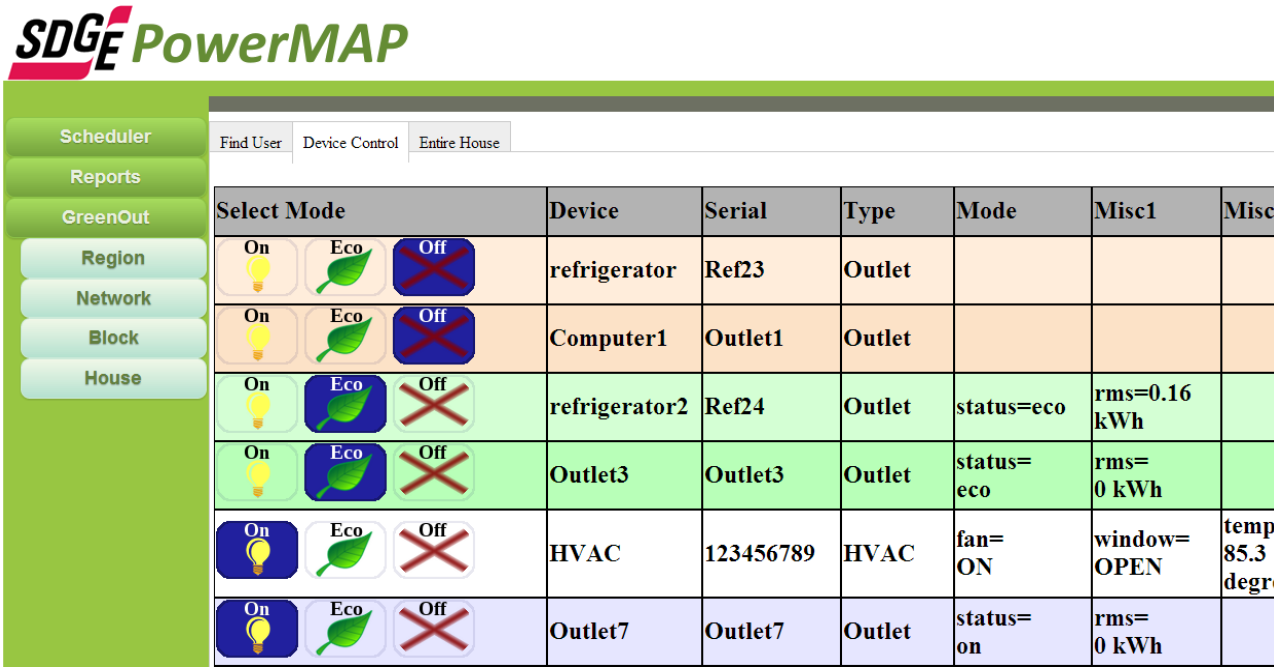
The Corporate Utility Application, "SDGE PowerMAP" allows SDG&E employees to remotely gather information and control home user smart devices. The application design focuses primarily on three categories; these three categories follow and will be detailed in this section of the report:

- Graphical User Interface
- Security and Privacy
- Functionality

Graphical User Interface

SDGE PowerMAP is designed for employees who will see the interface daily and are trained to operate the product. Thus, the interface presents more information in a condensed and readable format to maximize efficiency.

Figure 9. SDGE PowerMAP Interface



Shading

Each row is highlighted based upon the settings for that specific device. Even rows are shaded roughly ten percent darker to enhance readability. The settings for each device are also depicted by button-shading.

Context Menus

The sub-menus ("Region", "Network", "Block," and "House" in Figure 9) are active when the parent selection is active. So clicking Reports or Scheduler will generate different sets of sub-menus. This simplifies the interface.

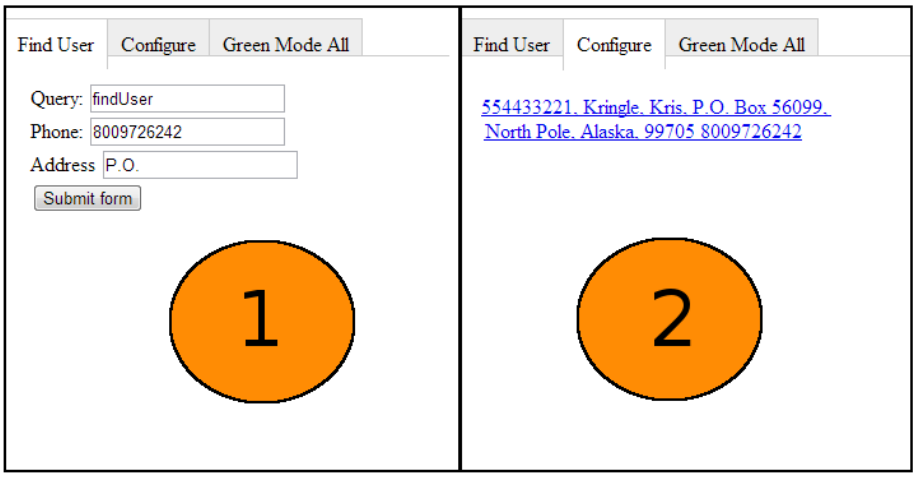
Context Tabs

The Tabs ("Find User," "Device Control" and "Entire House" in Figure 9) provide access to information and processes based upon need and security criteria. For example, in the picture above, "Device Control" only becomes available after user authentication.

Security and Privacy

The application empowers consumers but should not allow unwanted control or violate their privacy. Therefore, authentication is built into the application. In the example below, the user must call SDG&E and provide both their address and full phone number before the SDG&E employee can gain access to their record and the device list shown on the previous page.

Figure 10. User Authentication Process



Functionality

The corporate interface gathers dynamic data from the server and presents context-sensitive options to the user. For example, the user record in step two of Figure 10, *"User Authentication Process"*, is dynamically generated and returned by the server. Javascript enables the display of context-based functionality in the "Configure" tab; in this case, the SDG&E employee may click the link to view the users list of devices.

The corporate interface may be loaded with several options that produce different content. These options are listed in the Table 2.

Table 2. Corporate Interface Options

Option	Description
/corporate.aspx	Loads the main interface corpUI.html
/corporate.aspx?query=findUser	Locates the user by address and phone number, and returns a links for matching users. The links load a corresponding device list.
/corporate.aspx? query=getDeviceTable&userid=xyz	Locates the list of devices where xyz is an actual user id number.

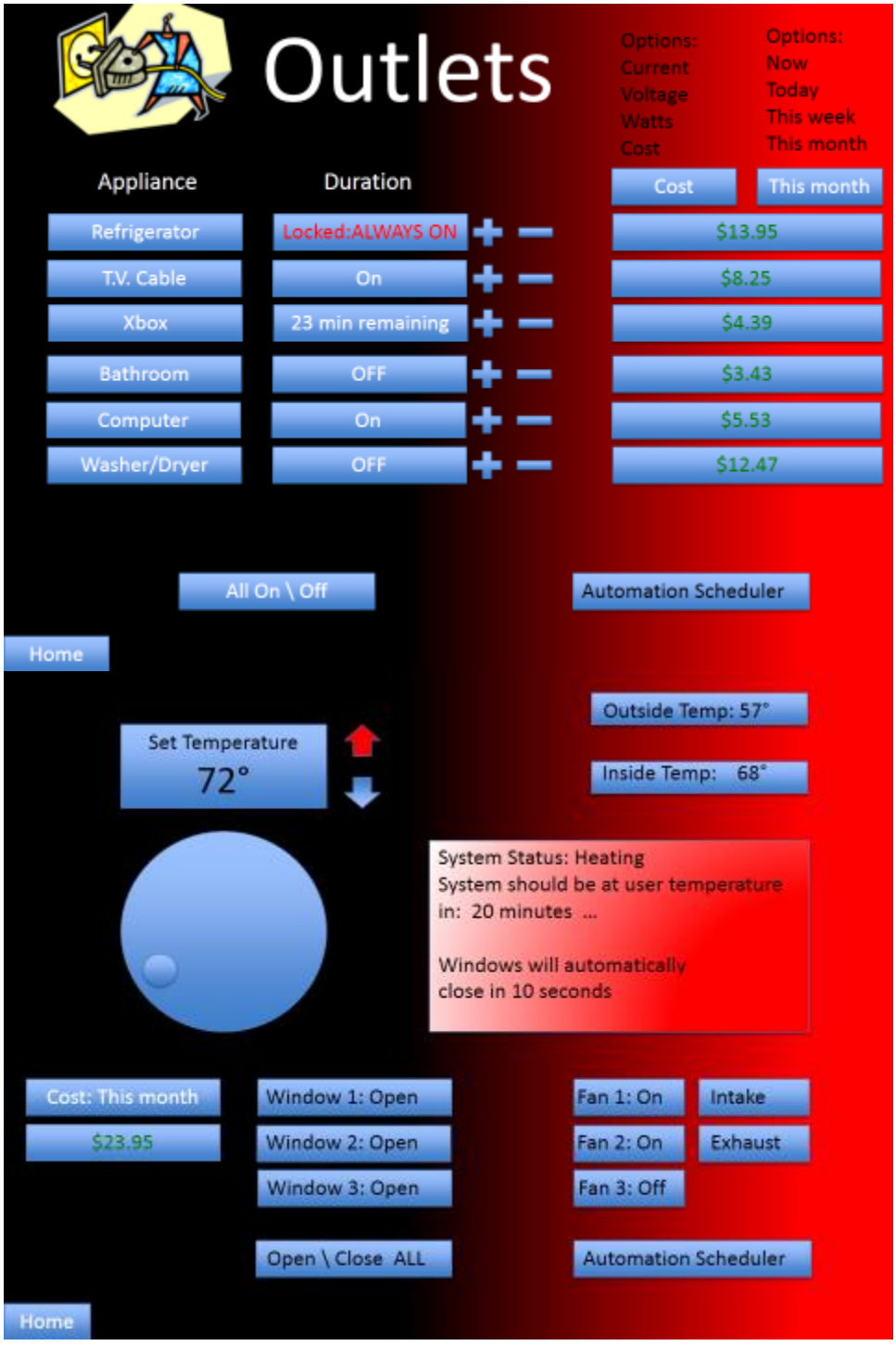
Conclusion and Recommendations

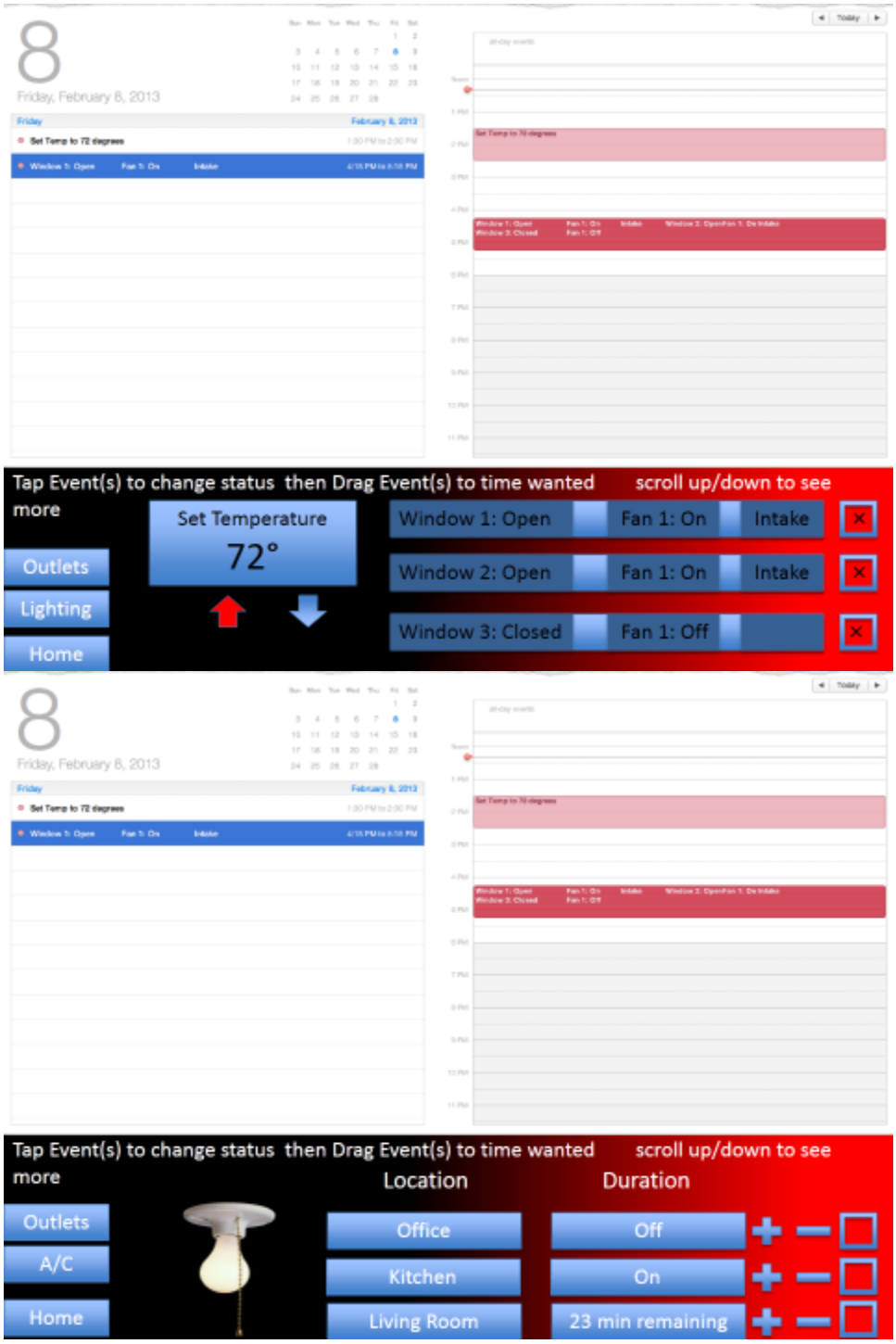
Starting from scratch, it is impressive what Home Power MAP has accomplished in just a few short months. We set up a network using two databases, which send and receive data from the software to the hardware nodes. Additionally, we have created three independent user interfaces which can send commands to various nodes as well as display various data collected by the servers. One of our biggest breakthroughs occurred when we were able to establish communication from the smart meter to the OSI PI server. This was accomplished almost a month before the deadline of our project, which allowed us to begin incorporating information from the SQL and OSI PI databases to our interfaces in real-time. With this new data we could build and test our different user interfaces' ability to receive data at a high rate. Following that, we established communication with the HVAC group. Lastly, we created a survey on SurveyMonkey regarding our different interfaces. Our plan was to have individuals take the survey after viewing our interface on design day. The survey was intended to give us ideas about additional features that the public would be interested in for our interfaces' future development. Regrettably, SurveyMonkey will only allow one survey per device, so we were only able to get a couple people to take the survey.

This project was not only challenging but very complex as well. In retrospect, a strong knowledge of various coding languages would be recommended as a fundamental step to continue this project. Some members of our group spent weeks learning various languages needed to develop our code and user interfaces. Another recommendation is to work closely with other team members. It is amazing how you can look over your code so many times trying to figure out where the mistake is but can never find it. By getting another perspective as to where your team member believes the problem is, your get the review of a fresh set of eyes and start to think a little differently. On another note, our team had the advantage of not needing to wait for hardware to start coding. Overall this was a fun, challenging, and stimulating project for everyone who worked on it. The team at Home Power MAP would like to thank the professors at San Diego State University for giving us the fundamental tools which allowed us to complete such a complex project. We would also like to thank SDG&E for sponsoring and creating the proposal for this project.

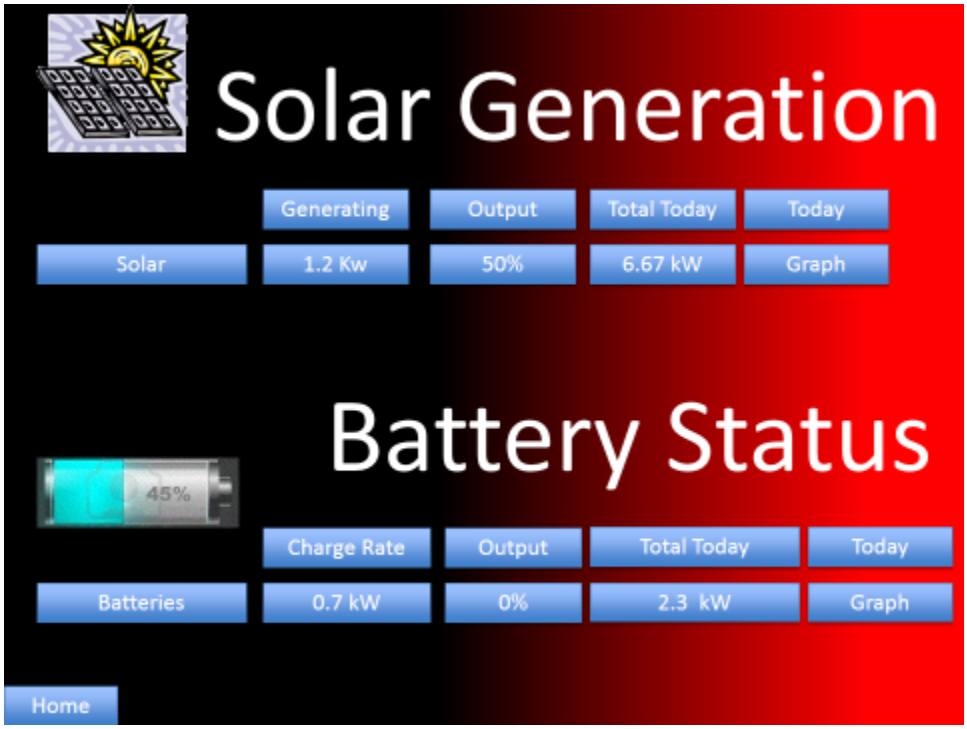
Appendices

Appendix A – Initial Web GUI Design

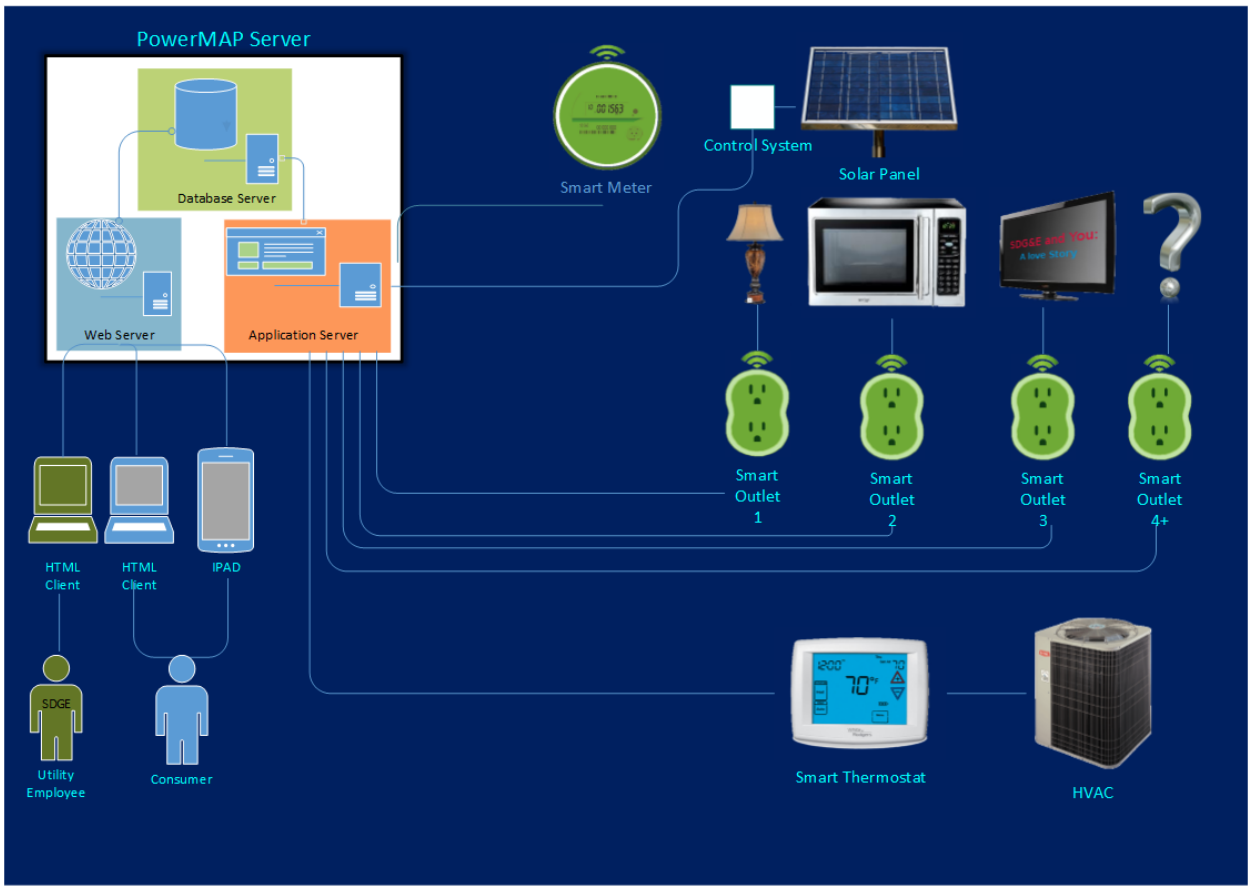

Outlets
Options:
Current
Voltage
Watts
Cost
Options:
Now
Today
This week
This month
Appliance
Duration
Cost
This month
Refrigerator
Locked:ALWAYS ON
$13.95
T.V. Cable
On
$8.25
Xbox
23 min remaining
$4.39
Bathroom
OFF
$3.43
Computer
On
$5.53
Washer/Dryer
OFF
$12.47
All On \ Off
Automation Scheduler
Home
Outside Temp: 57°
Set Temperature
72°
Inside Temp: 68°
System Status: Heating
System should be at user temperature
in: 20 minutes ...
Windows will automatically
close in 10 seconds
Cost: This month
$23.95
Window 1: Open
Window 2: Open
Window 3: Open
Fan 1: On
Intake
Fan 2: On
Exhaust
Fan 3: Off
Open \ Close ALL
Automation Scheduler
Home

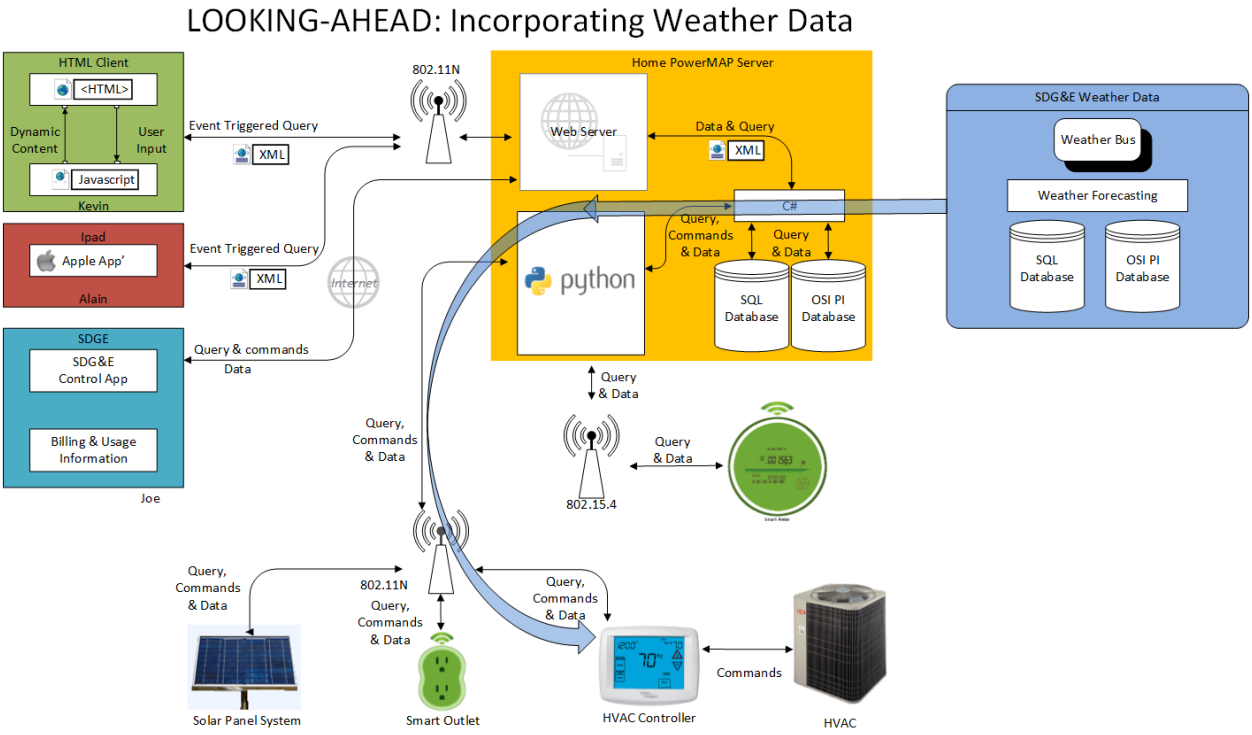

8
Friday, February 8, 2013
Tap Event(s) to change status then Drag Event(s) to time wanted
scroll up/down to see more
Set Temperature
72°
Outlets
Lighting
Home
Window 1: Open
Fan 1: On
Intake
Window 2: Open
Fan 1: On
Intake
Window 3: Closed
Fan 1: Off
8
Friday, February 8, 2013
Tap Event(s) to change status then Drag Event(s) to time wanted
scroll up/down to see more
Location
Duration
Outlets
A/C
Home
Office
Kitchen
Living Room
Off
On
23 min remaining


8
Friday, February 8, 2013
Tap Event(s) to change status then Drag Event(s) to time wanted scroll up/down to see more
Appliance
Duration
A/C
Lighting
Home
Medical Equipment
T.V. cable
Xbox
Locked:ALWAYS ON
On
23 min remaining
This Month
Last Month
This Year
KWh
48 kWh
125 kWh
325 kWh
Tier 1 Cost
19 kWh / $17.28
$27.28
$50.28
Tier 2 Cost
17 kWh / $19.48
$29.48
$59.48
Ties 3 Cost
12 kWh / $12.37
$22.37
$52.37
Overall Cost
48 kWh / $49.13
$79.13
$162.13
Graph
Graph
Graph
Appliance
Current
Voltage
P. F.
Watts
Cost
Time: Now
Washer/Dryer
10.3 A
219.6 V
0.95
2261.8
$0.30/h
Graph
Did you know: A 100 W light bulb uses 100 units of power(W) / per hour or .1 KWh
Home
Solar Generation


Solar Generation
Generating
Output
Total Today
Today
Solar
1.2 Kw
50%
6.67 kW
Graph
Battery Status
45%
Charge Rate
Output
Total Today
Today
Batteries
0.7 kW
0%
2.3 kW
Graph
Home

Appendix B – Block Diagrams





Appendix C – IP Addressing Scheme

The IP addressing scheme is designed statically for a private home area network which monitors and gives user control of smart outlets, HVAC, and solar panels. All devices will talk with the software aspect of the system using these designated addresses. All routers, servers, and data bases will also use this table to communicate with each other.

DEVICES	IP ADDRESS
Smart Outlets	192.168.144.1 – 192.168.144.30
HVAC	192.168.144.40
Solar Panel	192.168.144.50
Router	192.168.144.200
Python Bridge	192.168.144.110
SQL Server	192.168.144.120
OSI Pi Server	192.168.144.130
HTML Client Devices	192.168.144.140
iPad App Client Device	192.168.144.150
SDG&E Utilities Devices	192.168.144.160