NEAR REAL-TIME EXERCISE MACHINE POWER STATISTICS REPORTING

A Thesis presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Electrical Engineering

By
Brendan Cullen Asche
March 2010

COMMITTEE MEMBERSHIP

TITLE:	NEAR REAL-TIME EXERCISE MACHINE POWER STATISTICS REPORTING
AUTHOR:	Brendan Cullen Asche
DATE SUBMITTED:	March 2010
COMMITTEE CHAIR:	Dr. David Braun, Professor, Electrical Engineering
COMMITTEE MEMBER:	Dr. John Oliver, Assistant Professor, Electrical Engineering
COMMITTEE MEMBER:	Dr. Tina Smilkstein, Assistant Professor, Electrical Engineering

ABSTRACT

NEAR REAL-TIME EXERCISE MACHINE POWER STATISTICS REPORTING

Brendan Cullen Asche

A system composed of display, database, and power measurement modules can report exercise machine power statistics in near real-time. The modular components make this system compatible with multiple exercise machine models. The display module presents the statistics in a user-friendly way to a potentially large audience. A database module provides an efficient way of organizing and accessing stored statistics. Multiple types of power measurement modules gather power and energy generation measurements from exercise machines and transmit those measurements to the database over the computer network.

Keywords: energy harvesting from exercise machines

ACKNOWLEDGEMENTS

Above all I thank my parents, Doug Asche and Maggie Cullen, for supporting me throughout my college career both financially and emotionally. I also thank my entire family for their love and encouragement.

I also thank my thesis advisor, Dr. David Braun, for helping me choose my thesis topic and for all of the time he spent talking with me about the project and reading my document.

Finally, I thank my two thesis committee members, Dr. John Oliver and Dr. Tina Smilkstein, for being on my thesis committee and offering their feedback on my thesis document.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1 : INTRODUCTION

1.1 Introduction

This chapter introduces the context and motivation of this thesis project, presents an overview of the state-of-the-art, and introduces the thesis statement. Next, it introduces this thesis document by mapping out what subsequent chapters cover. Then, it describes the overall system's requirements and the modules that address them. To conclude, this chapter summarizes the overall system.

1.2 Context

Exercise machines dissipate most of the power generated by their human users as mechanical energy or heat. Self-powered exercise machines like the Precor EFX546i use a generator to convert the mechanical energy into electricity to power the user interface. But, the user interface does not require very much power, so a resistor dissipates the remaining energy as heat. To tap this source of wasted energy, Cal Poly students designed an energy harvesting system that redirects excess energy to the electricity grid, providing a small but environmentally friendly energy source [1]. Exercise bikes also provide an opportunity to harvest energy, so Cal Poly students designed a generator system [2] and a DC-DC converter [58] that retrofit one. These projects are the essence of Energy Harvesting From Exercise Machines (EHFEM).

While the energy harvesting system itself is critical, it is also important to know how much energy the system harvests. This thesis document presents a system that measures, stores, and displays the amount of energy harvested from exercise machines.

1.3 State-of-the-art

EHFEM is not a new idea. The following subsections talk about the EHFEM products currently available, the data reporting features those products offer, and non EHFEM-specific energy measurement and reporting systems currently available.

1.3.1 Existing EHFEM Products and Implementations

At the time of this document's writing, the market for EHFEM products is very small. Only three companies offer EHFEM products: The Green Microgym, The Green Revolution, and ReRev.

The Green Microgym, located in Portland, OR, says: "Much of our gym is already retrofitted to send electricity back to the building…" [3]. According to another page on their website [4], they harvest energy from ellipticals and stationary bikes.

The Green Revolution, Inc. is another company that offers EHFEM products. Their technology applies to ellipticals, cross-trainers, stepping machines, stationary bikes, and recumbent bikes [5].

ReRev, the third company that offers an EHFEM product, advertises that it can retrofit exercise machines to generate electricity compatible with the utility grid [6]. A number of universities and fitness clubs across the U.S. use ReRev's product. Their system consists of retrofitted exercise machines, one or more inverters, and a display computer according to an Oregon State University blog [7].

1.3.2 Data Reporting Features of EHFEM Products

Since this thesis document defends a system that displays the power and energy output of exercise equipment, the existing EHFEM products' display

capabilities are of particular interest. This section discusses those capabilities in as much technical detail as possible, though some of the companies do not reveal much about their technologies.

A wattage readout on each exercise machine's user interface appears to be the only information displayed to users at the Green Microgym [3]. If there is any data collection and reporting portion of their technology, they did not reveal it.

The Green Revolution, however, says they will soon provide their customers with energy output statistics via their website with the help of Fat Spaniels Technologies, a company that offers internet-based monitoring and reporting for renewable energy resources [5].

ReRev's product seems to vary for each customer. ReRev's website does not mention any specific power and energy output display systems, but news articles reporting on the implementations of ReRev's product indicate display options such as an LCD screen on the inverter box showing watts, a display computer [7], or a script running on the bottom of TV screens displaying the total energy produced so far [8]. Figure 1.1 shows a picture extracted from a video [9] that shows the ReRev inverter's LCD display.

Figure 1.1. A ReRev inverter's LCD display [9].

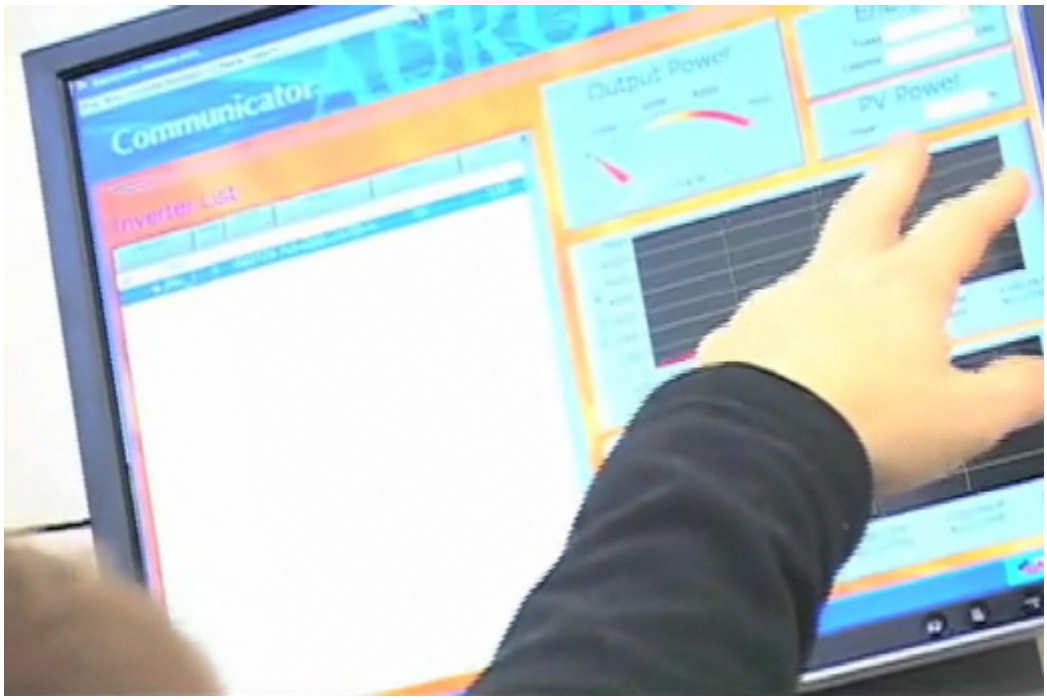


Figure 1.2. ReRev's display program [9].

Figure 1.2 shows a picture from the same video with ReRev's display program running on Windows XP. Visible at the screen's top is the logo "AURORA Communicator." Further investigation revealed that this is a monitor tool for Aurora

inverters [10]. This monitor tool shows the statistics transmitted by inverters using customizable graphs and charts. In terms of graphs and charts, the "AURORA Communicator"'s display capabilities depend on which software version ReRev uses [11]. It is also uncertain what connection medium ReRev uses to transmit measurements from Aurora inverters to the computer running the monitor tool. Since the administrator must connect the inverters' data outputs to the computer running the software and configure the inverters within the software, the software most likely runs on a computer in the gym. This limits the potential audience of viewers to gym users.

1.3.3 Non EHFEM Measurement and Reporting Systems

While there are very few EHFEM-specific products available, the power generation industry often uses systems that measure and report power and energy generation statistics. This section presents a couple of products in order to zoom in on the specific challenges faced by the system presented in this thesis.

PG&E is currently rolling out their SmartMeter system to customers [12]. This system replaces traditional residential energy meters with SmartMeters that wirelessly transmit hourly electricity and daily gas usage readings to one of PG&E's network access points [13]. Each customer can log into their online account via PG&E's website and view their energy consumption over the past day, week, or month. Although SmartMeters apply only to houses, their wireless capability and PG&E's web-based presentation of data are particularly interesting ideas since they could work well in the case of exercise machines; wireless data transmission could be applied to an exercise machine's power and energy measurement system, and a web-

based presentation of data reaches an audience much larger than the gym-going crowd.

Enphase sells inverter systems for residential solar power systems. One of Cal Poly's energy harvester designs [1] uses Enphase's Microinverter product. Part of Enphase's product is an appliance that collects diagnostic information, power measurements, and energy measurements transmitted via power cables by Enphase Microinverters. The appliance connects to a computer network and hosts a web page that allows computers to view the information it collects. If the computer network provides internet access, the appliance also uploads the collected information to Enphase's website, where customers can log in and view graphs and figures of their system's power and energy production [14]. This system potentially offers solutions to three challenges faced by the system presented in thesis document: how to measure the power and energy output from exercise machines, how to transmit those measurements, and how to display those measurements.

1.4 Motivation

Cal Poly's Recreation Center expansion project provides an opportunity to implement EHFEM with a statistics reporting system. The EHFEM products reviewed above suffer from significant limitations; The Green Microgym seems to lack power and energy production tracking all together, ReRev's product lacks the ability to display power and energy production to a large audience, and The Green Revolution's product is not ready yet. The system presented in this thesis provides a near real-time exercise machine power statistics reporting system accessible to anyone via a web page. It plays a critical role in accomplishing the Recreation

Center's goal—showing Cal Poly's use of cutting edge technology and renewable resources.

1.5 Thesis Statement

This thesis defends the design, development, and testing of a near real-time exercise machine power statistics reporting system composed of display, database, and power measurement modules. Chapter 2 describes two different power measurement module designs. Chapter 3 describes the database server configuration, the database schema, and the automated script that maintains part of the schema. Chapter 4 describes the display module's design and describes an implementation that followed a different design. Finally, chapter 5 concludes the thesis with a brief summary of the requirements this system fulfills, a discussion of other approaches for the system, and some suggestions for future work.

1.6 Requirements

The system presented in this document must satisfy three major requirements. For the first requirement, the system must measure the power and energy generated by exercise machines and transmit those measurements. For the second requirement, the system must store the power and energy measurements in a manner that makes it easy to search for measurements based on criteria. For the final requirement, the system must display power and energy measurements in an interactive and widely accessible way.

1.7 Modules

The system breaks down into three modules, each implementing one of the three major requirements. The power measurement module (PMM) implements the

first requirement, the database module implements the second requirement, and the display module implements the third requirement. In fulfilling its requirements, each module's design faces its own set of problems.

To measure the power and energy generated by exercise machines and then transmit those measurements, the PMM design focuses mainly on the device required to obtain measurements, the data transmission medium, and the software that delivers measurements to the database module.

To store measurements in an easy-to-search manner, the database module design addresses two main problems: the selection of the database's platform in terms of hardware and software, and the design of tables and table relationships in the database.

The display module's requirements present multiple problems. One problem is the execution of database queries to retrieve measurements. Another problem is the design of graphs, charts, animations, colors, and pictures that display measurements. The last problem is the selection of an output mechanism for the display that makes it widely accessible.

1.8 Overall System

Figure 1.3 shows a diagram of the system as a whole. The "Interactive Web Page" presents users with configurable graphs and charts. Decade, an EE department server, hosts a web page that contains the display module's software. This display software then runs in the user's browser window, queries the MySQL server running on Decade for statistics, and draws graphics using those statistics. Decade receives energy harvesting statistics from two types of PMMs: one to support Enphase

Microinverters, and another to support stand-alone network devices. The Enphase PMM type connects to the Cal Poly computer network via a standard Ethernet cable while the stand-alone network devices connect to a wireless access point.

The display module consists of the "Interactive Web Page" and the web server running on Decade. The database module consists of a schema implemented in MySQL Server. The PMM encompasses the Stand-Alone devices on the left and the Energy Management Unit (along with its inverters) on the right.

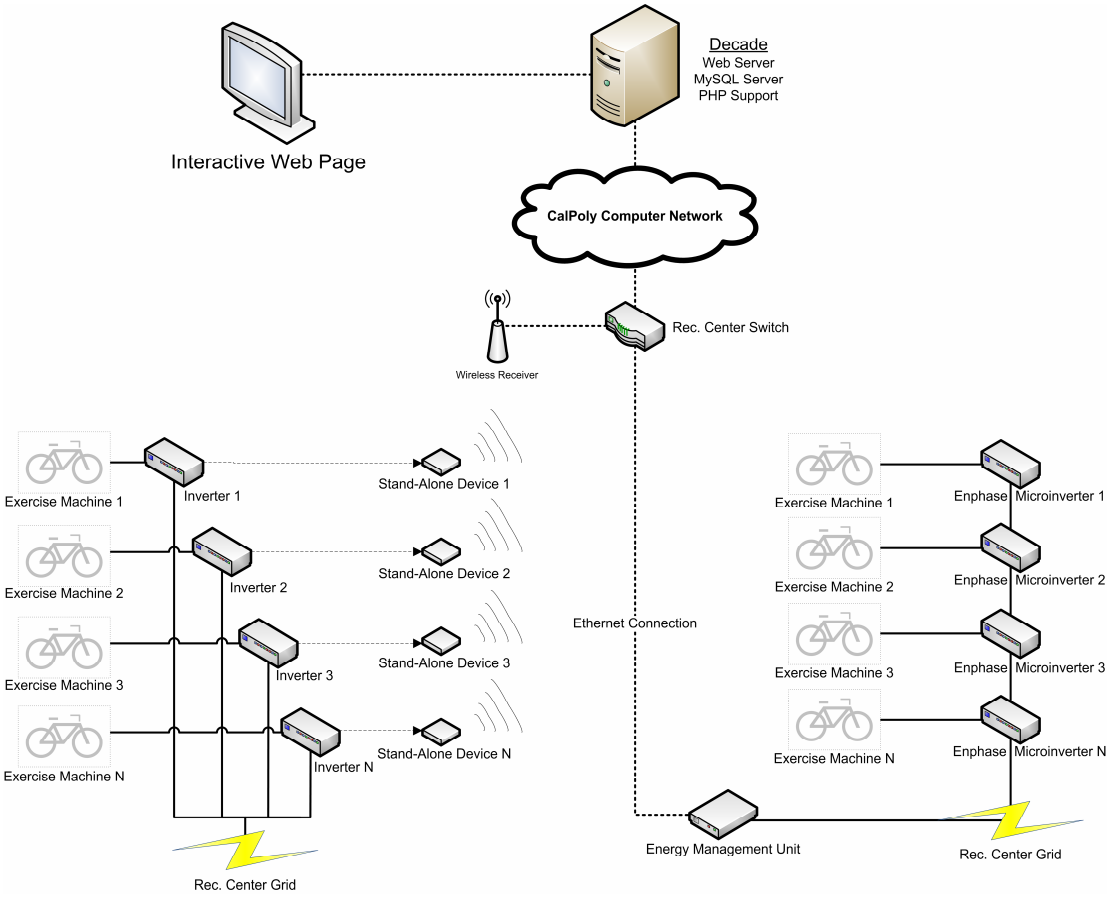


Figure 1.3. Diagram of the system as a whole.

CHAPTER 2 : POWER MEASUREMENT MODULE

2.1 Introduction

Chapter 2 begins by stating the power measurement module (PMM)'s overall goals and constraints. Next, it describes each PMM option considered and how they fulfill the overall goals and constraints. Then, it justifies the PMM options chosen for implementation. Finally, this chapter defends the design, development, and testing of the PMM options chosen for implementation, in terms of each design's power measurement system, data transfer method, and server-end software.

2.2 Goals and Constraints

Figure 2.1 shows the PMM block diagram. The PMM's first goal is to measure the power and energy produced by the inverter. For its second goal, server-end software retrieves or receives the power and energy production statistics and remembers which power measurement device they came from. The final goal requires that the server-end software connects to the database server and adds to it the power and energy measurements along with which power measurement device they came from.

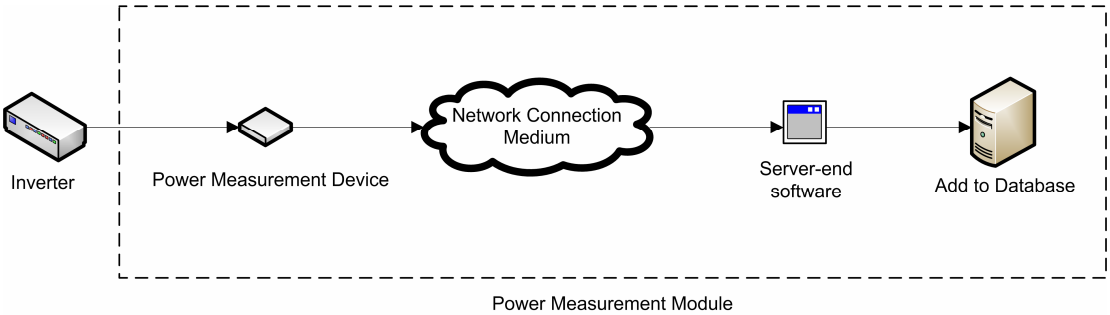


Figure 2.1. Power measurement module block diagram.

The constraints considered for each PMM are: monetary cost, design cost, network infrastructure cost, inverter compatibility, and exercise machine

compatibility. Monetary cost represents an estimation of the cost to purchase the equipment for the power measurement device. Design cost rates how complex the option's design is and how difficult or time consuming it is to implement. The network infrastructure cost accounts for any additional network cabling, Ethernet jacks, and Ethernet switches that an option requires. Inverter compatibility reflects the option's ability to work readily with any inverter type. Similarly, exercise machine compatibility represents any limitations on the exercise machine models an option works with.

2.3 Options Considered

Three options provide the functionality required by the PMM: a Communications Specification for Fitness Equipment (CSAFE) [15] program, an embedded system, and the Enphase Energy Management Unit (EMU). Table 2.1 shows the ratings in each constraint for the three options and their total scores.

		Options					
		CSAFE Program		Embedded System		Enphase EMU	
Criteria	**Weight**	Rating	Score	Rating	Score	Rating	Score
Monetary Cost	2	5	10	2	4	3	6
Design Cost	3	1	3	2	6	5	15
Network Infrastructure Cost	1	1	1	5	5	5	5
Inverter Compatibility	2	5	10	5	10	1	2
Exercise Machine Compatibility	3	1	3	5	15	5	15
Total (higher is better)			**27**		**40**		**43**

Table 2.1. Decision matrix for power measurement module options.

The design cost and exercise machine compatibility criteria are the most important since students should be able to implement the design in time for the new Recreation Center, and the implementation should work on as many exercise machine models as possible. Monetary cost and inverter compatibility are slightly less

important since it is likely that Cal Poly will provide funding, and only the Enphase EMU option has limited inverter compatibility. Network infrastructure cost is the least important criterion since the worst case scenario requires one Ethernet cable per exercise machine and possibly some additional network interface controllers, which wouldn't be a significant hurdle.

2.3.1 CSAFE Program

The CSAFE design uses the exercise machine's embedded system as a platform to run custom-designed software. CSAFE is a protocol capable of transmitting, among other things, exercise machine power generation statistics to a desktop computer. The desktop computer must run a program that interfaces with the exercise machine's CSAFE port over standard Ethernet cables. This approach costs no additional money since the Precor EFX546i, the exercise machine that Cal Poly Electrical Engineering (EE) students are developing an energy harvester for, has built-in support for CSAFE. The lack of a development platform to reprogram the Precor EFX546i and the lack of a computer program to interface with the exercise machine results in a high design cost. Since each exercise machine needs to connect to the computer running the CSAFE interface program, this option requires either peer-to-peer capabilities [16] to connect the exercise machines' Ethernet cables in a chain, with the last exercise machine connecting to the computer, or it requires each exercise machine to connect to a different Ethernet port on the computer. In either case, the design requires more Ethernet cables and possibly more network interface controllers to provide enough ports. The CSAFE option does not depend on a specific inverter, but it is greatly limited in its exercise machine compatibility—only the

Precor EFX546i supports CSAFE. Cal Poly students are developing a harvesting system for stationary bikes [2], [58], which are incompatible with this option because they do not support CSAFE.

2.3.2 Embedded System

The embedded system option utilizes a programmable chip to interface with an energy meter and a network interface controller. This option is monetarily costly compared to the rest since an embedded system attaches to every exercise machine. While it is possible to design the embedded system from scratch (adding to the design cost), a number of products exist that could provide programmable chips with integrated wireless network controllers. These products range in price from about $150 to $200 per unit [17]-[21]. With one of these products serving as the platform, an energy meter is the only component to add to the hardware design. The rest of the design cost resides in the programmable chip's firmware design. Since the platform supports 802.11b/g wireless local area network connectivity, it takes advantage of the wireless access points available in the Recreation Center, requiring no additional network infrastructure. There are no foreseeable limitations on which inverters and exercise machines this option works with.

2.3.3 Enphase EMU

The last option considered is the Enphase EMU. Although Power-One [48], Fronius [49], and Schneider Electric [50] also offer inverters with computer connectivity and data reporting capabilities similar to Enphase's product, this document considers only Enphase's product because Cal Poly EE students [1] selected it for their energy harvester's design. This option's monetary cost is slightly

less than the embedded system option's because a single EMU, for a price of about $350 through online vendors, gathers the power and energy output statistics of multiple Enphase Microinverters simultaneously, meaning multiple exercise machines simultaneously. Its design cost is extremely small since the EMU itself hosts a web page containing power and energy statistics—gathering statistics is simply a matter of plugging the EMU's Ethernet port into the Cal Poly network and periodically extracting the desired information from the web page. The option requires no additional network infrastructure; inverters use their output power lines and a proprietary protocol to transmit their data to the EMU, which plugs into a wall socket on the same circuit. The Enphase EMU option scores low in inverter compatibility because it only works with Enphase Microinverters. This option could work with any exercise machine.

2.4 Options Selected

2.4.1 Enphase EMU System Design

The Enphase EMU option is one of two options this thesis defends. Although it costs $350 per EMU and only works with Enphase Microinverters, this option is the best choice since it scores so highly in the other constraints.

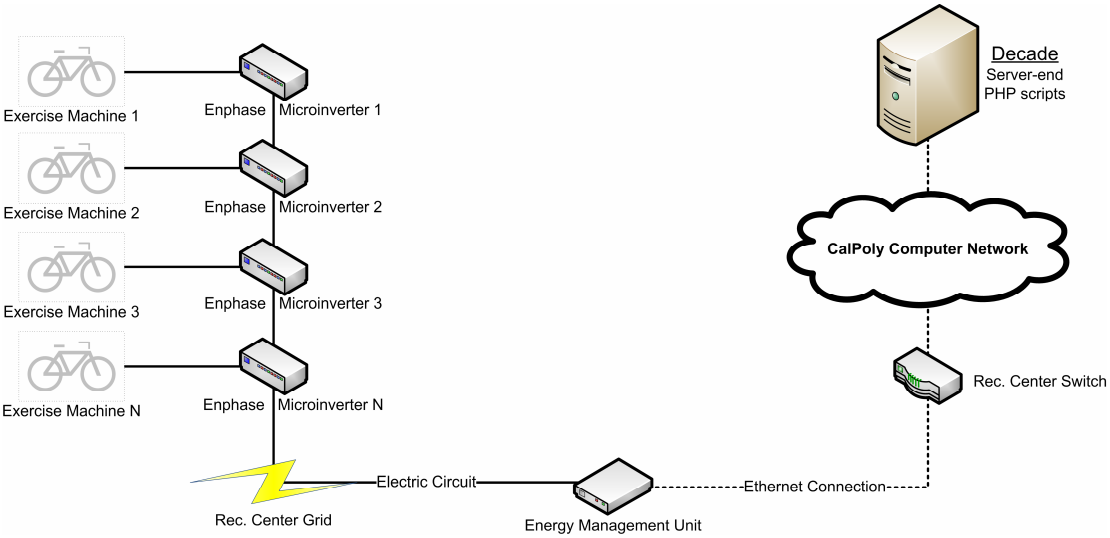


Figure 2.2. Enphase power measurement module configuration.

Figure 2.2 shows the Enphase PMM configuration. Each inverter, which must be an Enphase Microinverter in this design, has three connectors: one inverts the DC power from a single exercise machine, another outputs power, and a third connector accepts the power output from another inverter. When installed, the first inverter's output connects to a wall outlet, the second inverter's output connects to the first inverter, the third inverter's output connects to the second inverter, and so on to form a chain. The next step is to add an Enphase EMU to the configuration.

The Enphase EMU plugs into a wall outlet on the same circuit and searches for Enphase Microinverters over a preset time. After the search period ends, the EMU will only listen to data broadcasted from the inverters it found during the search period; a new search must be manually initiated to add more inverters. The EMU then collects via power lines the statistics broadcasted by each inverter it found during the search. The EMU's Ethernet port plugs into the Cal Poly network through a Recreation Center Ethernet switch, which allows the server, Decade, to access the web site hosted at the EMU's Internet Protocol (IP) address.

The final part of the Enphase PMM configuration is the PHP script that executes on Decade. The script visits a list of EMUs and extracts each EMU's power and energy statistics. It then connects to the database server on Decade and adds the new statistics to the proper table in the schema.

2.4.2 Enphase EMU System Development

Since the Enphase EMU already provides power and energy measurement capabilities and network connectivity, development focused on the script that retrieves those measurements from the web site hosted on the EMU. This data retrieval script accomplishes two tasks: it retrieves power and energy information from the EMU, and it adds the new information to the database. The development strategy for the data retrieval script was iteration—each iteration refines the previous iteration's implementation or adds new features to solve new problems.

2.4.2.1 First Development Cycle

During the first development cycle, the design outlined two scripts, one for each task. A Visual Basic Script would have retrieved statistics from the EMU's web site and sent them to a PHP script, which would have added them to the database. After investigating the capabilities of Visual Basic Script and PHP, it turned out that PHP provides superior functionality for downloading a web site compared to Visual Basic Script. In addition, moving information between Visual Basic Script and PHP added a significant amount of complexity. From then on, development focused on a single PHP script that accomplished both tasks.

2.4.2.2 Second Development Cycle

The code written during the second development cycle performs the two tasks in two different sections of a single script, provided in Appendix A. Figure 2.3 shows a flow chart of the script.

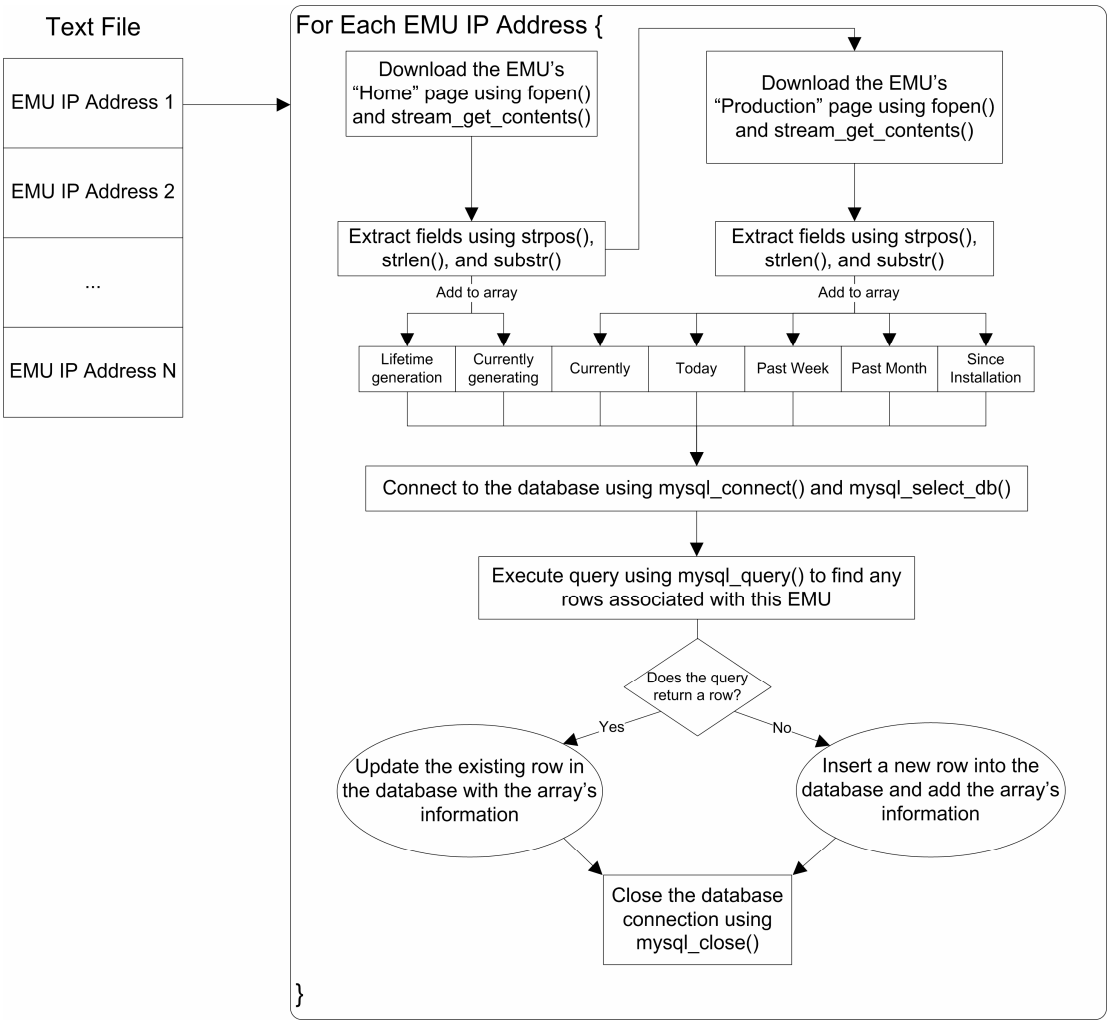


Figure 2.3. Retrieval script flow chart.

The first section, tasked with retrieving power and energy information from the EMU, begins by using the PHP functions "fopen()" and "stream_get_contents()" to connect to the EMU's "Home" web page and download its text. Next, the script uses the "strpos()," "strlen()," and "substr()" functions to extract the EMU's serial number and store it in a variable. The script uses the same functions again to extract

the values associated with the "Lifetime generation" and "Currently generating" fields that the EMU's "Home" web page displays in chart format. The script stores these values in an array. The script then repeats the entire process, starting at the "fopen()" function, for the EMU's "Production" web page to extract values from the "Currently," "Today," "Past Week," "Past Month," and "Since Installation" fields. The script's first section ends when it appends these values to the array.

The script's next section adds the extracted information to the database. It begins with hard-coded variables representing the MySQL server's host name, database login information, and schema name. The script then executes the "mysql_connect()" and "mysql_select_db()" functions to connect and log in to the MySQL database running on Decade. Next, the script executes a hard-coded query that returns any rows of data associated with the EMU serial number extracted earlier. If this initial query returns a row, then the script executes a query to update the existing row associated with this EMU serial number. The update uses the values stored in the previous section's array. If the initial query does not return any rows, the script executes a query that inserts a new row into the database containing the EMU serial number and the information stored in the array. The script uses the "mysql_query()" function to execute queries. As a final step for this section, the script closes the connection to the database using the "mysql_close()" function.

2.4.2.3 Third Development Cycle

The script's third and final development cycle added the following features to address problems realized during development: expandability, improved error

handling, more descriptive status messages, and the ability to detect the power and energy statistics' measurement units.

For expandability, the script reads a text file named "emu_ip_addresses.txt," which contains one IP address per line. The script uses the "file()" function to open "emu_ip_addresses.txt" and adds each IP address into an array. Next, the script uses a "foreach" loop to repeat its two sections for each IP address. With this change in place, adding additional EMUs to retrieve statistics from equates to adding additional EMU IP addresses to "emu_ip_addresses.txt."

To better tolerate file stream errors that it might encounter, the script checks the "file()" and "fopen()" functions' return values to verify that their file streams successfully opened. If one of these operations fails, the script prints an error message explaining which stream failed. The script also prints out a message indicating the EMU address it intends to connect to and another message when it begins reading from the stream. If the script cannot open a file stream to an EMU, it indicates a connectivity problem between Decade and the EMU.

The third development iteration's final improvement addresses the case where the EMU's web site reports statistics in units other than W or kWh. The script reads the unit of each power and energy statistic, extracts the order of magnitude by searching for "k," "K," "M," "m," or an empty string (which indicates units in W or Wh), and multiplies statistics by their order of magnitude before adding them to the database. This ensures all values in the database represent Watts or kilo-Watt-hours.

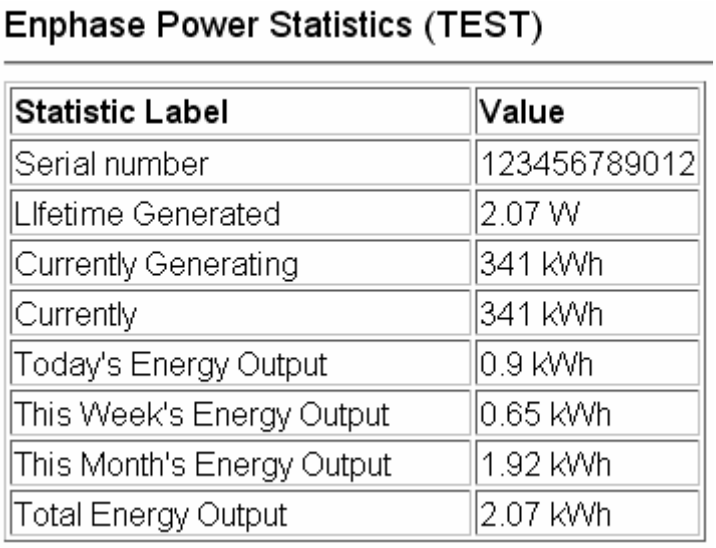
2.4.3 Enphase EMU System Testing

The Enphase EMU testing strategy employs two environments: a test environment, and an implementation environment. The test environment for the Enphase EMU system consists of a Dell E6400 laptop running the Windows Server 2003 operating system. The laptop runs a web server using the Internet Information Services (IIS) Manager component of Windows Server 2003. IIS serves copies of the EMU's "Home" and "Production" web pages, which contain a few minor changes in their HTML code to make the links work and the logos load. This configuration simulates an EMU web site to test the script's data retrieval section on.

The implementation environment for the Enphase EMU option consists of Decade, which also runs Windows Server 2003. Duplicating the test environment on Decade begins with the creation of a new folder, named "ehfem," located in C:\Inetpub\wwwroot. Next, the duplication process uses IIS to create a new web page at decade.ee.calpoly.edu/ehfem/, sets the new web page to use the test web page (copied from the test environment) as the default file, and removes the references to the default web pages. The implementation environment uses a database schema identical to the test environment's. The duplication process's final step verifies a properly functioning PHP installation by creating a website containing just "<?php phpinfo(); ?>", which displays the PHP configuration.

To test if the script's second section successfully adds new information to the database, a test web page displays the database's information in a chart. The test web page, shown in figure 2.4 and provided in Appendix B, contains a PHP script that logs into the database, executes a hard-coded query to retrieve the data, and stores the data in variables accessible to a HTML-based chart. The hard-coded database login

information contained in the PHP poses no security risk since clients accessing the web page cannot see the PHP script source; the web server compiles and executes the PHP code and removes the script source from the web page before serving it.



Enphase Power Statistics (TEST)

Statistic Label	Value
Serial number	123456789012
Lifetime Generated	2.07 W
Currently Generating	341 kWh
Currently	341 kWh
Today's Energy Output	0.9 kWh
This Week's Energy Output	0.65 kWh
This Month's Energy Output	1.92 kWh
Total Energy Output	2.07 kWh

Figure 2.4. Enphase EMU test web page.

The implementation environment tests if the script works with a real EMU plugged into the Cal Poly network. Initial tests revealed a small issue where the script printed “CTYPE” instead of numeric measurement values, and failed to add information to the database. Explicitly casting the measurements to *float* variables fixed this issue. With this fix in place, both sections of the script functioned correctly, and the script successfully retrieved statistics from the EMU and added them to the database. Monitoring the database through MySQL Query Browser revealed that the script correctly added a new row to the proper table in the database schema or updated an existing row if the serial number already existed in the table.

2.4.4 Embedded System Design

The embedded system option is the second of the two options chosen from table 2.1. It is the best option to accommodate any inverter, in the case that the energy

harvesting system does not use Enphase products. Although the option has a high monetary and design cost, its scores in network infrastructure cost, inverter compatibility, and exercise machine compatibility make it a much better option than the CSAFE Program option.

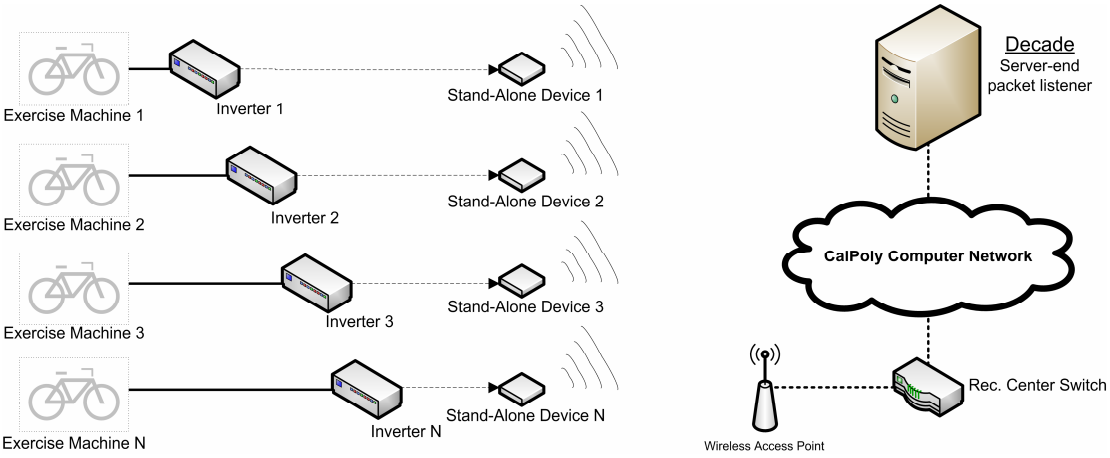


Figure 2.5. Embedded system power measurement module configuration.

Figure 2.5 shows the embedded system module configuration. Each stand-alone device contains a power meter chip, 802.11b/g wireless network adapter, and an embedded processor. The device uses the exercise machine's output as its power source. The power meter chip reads the inverter's power and energy output. The embedded processor runs a program that continuously reads power and energy measurements from the power meter, encapsulates the measurements and a device-unique identification number into packets, and sends the packets to the server, Decade, via a wireless access point in the Recreation Center.

2.4.5 Embedded System Development

This section presents the development at a high level, with the embedded system's hardware and software implementations left as future work for Cal Poly students. The development process's high level presentation covers the problems that

an embedded system implementation must solve, namely: an IP transport layer protocol, a packet payload format, a packet sending program, and a packet listener program.

2.4.5.1 IP Transport Layer Protocol

The embedded system uses the Transmission Control Protocol (TCP) transport layer protocol. The TCP significantly reduces the design cost associated with the packet listener program when compared to the User Datagram Protocol (UDP), but requires slightly more communication overhead [22]. The TCP's increased overhead is due to its connection-oriented design, which requires the transmission of more packets to establish and maintain a connection between the client and the host, and the TCP's larger header [23] versus the UDP's [24]. Although this technically results in higher power consumption, the embedded system transmits measurements at a slow rate to minimize power consumption, making this issue much less significant than the packet listener program's design cost. Unlike the UDP, the TCP prevents problems like dropped packets, duplicated packets, and packets arriving out of order [22]. In addition, the TCP's connection-oriented design guarantees that each exercise session opens a new TCP connection to the packet listener program. The packet listener program can then create a new thread for each connection, which provides a couple of advantages.

First, it implicitly distinguishes measurements coming from any number of embedded systems by their TCP connection. This means that each thread only processes measurements from one particular exercise session, and the thread exits when its corresponding exercise session ends. It also means that each thread opens its

own connection to the database and only updates one row in the database, a row that only this thread's exercise session will modify.

The second advantage of using threads is expandability—when there are enough exercise sessions transmitting measurements to outpace Decade's ability to process them, Decade's receive buffer will overflow, resulting in dropped packets and potentially inaccurate energy measurements in the database. Given that the packet listener program is multithreaded, the UDP's connection-less design requires the packet listener program to explicitly decide when an exercise session begins and ends, explicitly direct incoming measurements to the proper thread, and check for duplicated and out of order packets. These additional requirements add unnecessary design cost to the packet listener program, making TCP a better choice.

2.4.5.2 Packet Payload Format

Although both the packet sending program and packet listener program use the TCP, they must follow a consistent format within the TCP's payload so the packet listener program correctly parses the information in each packet's payload. Table 2.2 shows the format of each TCP packet's payload.

Bits	0-15	16-31
0	MAC ID	MAC ID
32	MAC ID	Instantaneous Power
64	Instantaneous Power	Total Energy So Far
96	Total Energy So Far	

Table 2.2. Embedded System packet payload format.

The first 48 bits contain the device-unique identifier. Since every network adapter has a unique Media Access Control (MAC) address, it makes an ideal device-unique identifier. It is also possible to give each embedded system a name, but this

adds work to the implementation since each embedded system requires its own unique hard-coded name. The following 32 bits contain a floating point number that represents the exercise machine's instantaneous power output. The final 32 bits contain another floating point number that represents the total energy measured so far in the exercise session.

2.4.5.3 Packet Sending Program

The packet sending program runs on the stand-alone device's embedded processor. This program reads power and energy measurements, encapsulates them into packets, and sends those packets to the packet listener program, which runs on Decade. The program sends one packet per second, although students developing this module may choose a different rate. Four functions accomplish these tasks: the measurement reader, connection initializer, packet payload maker, and packet sender.

The measurement reader function takes no arguments and returns a measurement structure containing floating point values that represent the instantaneous power output and the total energy generated so far. This function implements an interface with the energy meter to support communication over an input/output connector, allowing the function to read the energy meter's power and energy readings.

The connection initializer requires the IP address of Decade as an argument. For its first task, it uses a hard-coded string value representing Mustang Wireless' Service Set Identifier (SSID) to connect to Mustang Wireless. For its last task, it uses Decade's IP address to set up the TCP type socket needed to send TCP packets over the network. On a side note, Cal Poly's Information Technology Service Desk must

know each embedded system's MAC address in order for Mustang Wireless access points to provide connectivity.

The packet payload maker takes two arguments. The first argument is a measurement structure containing the instantaneous power output and the total energy generated so far this session. The second argument is a pointer to a packet payload structure. This function copies the MAC address of the embedded system's wireless network adapter, the instantaneous power output, and the total energy generated so far into the packet payload structure in the format outlined in table 2.2.

The packet sender function uses two arguments: a pointer to a packet payload structure and a destination port number. It sends the packet payload using the socket set up by the connection initializer to the specified destination port number on Decade.

2.4.5.4 Packet Listener Program

The packet listener program, which runs on Decade, consists of a connection acceptor thread and any number of exercise session threads.

The connection acceptor thread waits for TCP connection requests on a specific port. When an exercise session begins and the packet sending program requests a TCP connection on that specific port, the connection acceptor thread accepts the connection, creates an exercise session thread, hands the connection off to the exercise session thread, and waits for the next TCP connection request.

Each exercise session thread processes the packets transmitted by one particular embedded system over the course of one exercise session. The exercise session thread begins by connecting to the database, creating a new row in the proper

table, and filling in the appropriate row attribute with the exercise session's start time. Next, it enters a loop that waits for new packets. For each new packet, it extracts the measurements from the packet payload, determines if the packet's instantaneous power measurement is the new peak output power, and updates the previously created row in the database with the new total energy, peak output power, and time at which the packet arrived. At the end of an exercise session, the embedded system loses power, and the TCP connection times out. At this point, the exercise session thread closes its connection to the database, closes the TCP connection, and exits.

2.4.6 Embedded System Testing

The embedded system testing strategy focuses on four incremental tests to verify the packet sending and packet listener programs' functionality. The first test verifies network connectivity, the second verifies TCP connectivity, the third verifies that the embedded system can transfer an exercise session's measurements, and the fourth tests the packet listener's ability to handle multiple exercise sessions in parallel and in sequence.

For the first test, the developer runs just the connection initializer on the embedded system. The developer then pings the embedded device from Decade; if the embedded system responds to the ping, then it successfully connected to Mustang Wireless. In the second test, the developer runs the connection acceptor on Decade and the connection initializer on the embedded device. If the connection initializer opens a TCP connection to the connection acceptor, the second test is successful. In the third test, the developer adds in all of the packet sending and packet listener programs' functions. If the packet sending program opens a TCP connection with the

packet listener program, transmits the measurements of an exercise session to the packet listener program, and the packet listener program adds those measurements to the database, then the third test is successful. In the fourth and final test, the developer uses multiple embedded devices to conduct multiple exercise sessions at the same time and in succession to test whether the packet listener program can handle multiple exercise sessions. The next chapter details the database to which the packet listener program adds measurements.

CHAPTER 3 : DATABASE MODULE

3.1 Introduction

Chapter 3 first covers the database module's overall goals and constraints. After that, it describes and justifies the host computer's specifications, the database server program, and the database administration program. Next, it explains and justifies the database schema design and development. Finally, this chapter describes and justifies the strategy used to test the database module's functionality and the results.

3.2 Goals and Constraints

Figure 3.1 shows the database module block diagram. The database module's first goal is to store incoming measurements from the PMM in the database's "ehfem" schema. Its second goal is to send information to the display module when the display module queries it. The display module uses the information to draw its graphics.

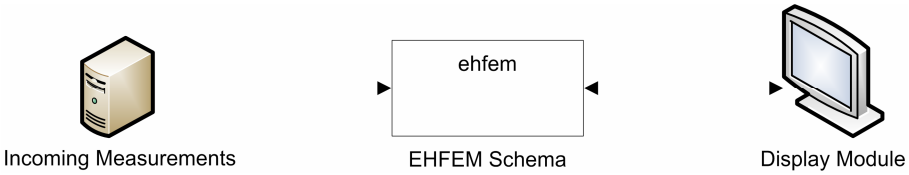


Figure 3.1. Database module block diagram.

The database module's constraints are: the physical server platform's monetary cost, the compatibility between the physical server platform and the database server and administration programs, the schema design and implementation costs associated with the database server program, the database server program's monetary cost, the database administration programs already in use on Decade, and the limits on which information attributes each PMM type can provide to the schema.

3.3 Host Computer Specifications

The EE department provided the EHFEM project with Decade to host any applications the EHFEM project required, such as the database server and database administration programs. Decade runs the Microsoft Windows Server 2003 operating system on a 2.00GHz Intel Pentium 4 processor with 2GB of RAM. Cal Poly computer hardware availability dictates these specifications due to the monetary cost constraint's high importance. The compatibility between the host computer and the database software is not a significant constraint since there are database servers available for any host computer platform within reason.

3.4 Database Server

The database module uses MySQL Server by Sun Microsystems, Inc. for its database server. MySQL provides a good option when considering schema design cost, schema implementation cost, and monetary cost. Its high popularity [25] means widespread availability of tutorials and other resources that greatly reduce the design and implementation costs. In terms of monetary cost, Sun Microsystems, Inc. offers MySQL for free under a General Public License [26]. Since Decade already has MySQL installed, configured, and supporting the existing Efficient Deployment of Advanced Public Transportation Systems (EDAPTS) Research Project [29], the EHFEM project must utilize MySQL as well. Installing a different database server might conflict with MySQL or cause downtime, posing an unnecessary risk to the EDAPTS Research Project.

3.5 Database Administration

A database administration program with a graphical user interface provides a more efficient way of creating and maintaining a database schema than a command line interface. Several such programs can provide this functionally for the database module. phpMyAdmin [57] is a popular and free administration program that provides a web interface. Another option is MySQL Administrator [27], available directly from MySQL. The final option, also directly from MySQL, is MySQL Workbench [28], which is a new product that replaces MySQL Administrator.

The "database administration programs already in use on Decade" constraint points to one database administration program. Since the existing MySQL database uses MySQL Administrator, and installing a new database administration program puts the EDAPTS Research Project at risk, MySQL Administrator is the only option.

3.6 Database Schema

This section presents the database schema, named "ehfem." "ehfem" stores measurements and the information associated with them in a set of tables, and these tables implement a design that simplifies the queries required to locate a desired piece of information. The subsections below explain and justify the database schema's design, development, and testing.

3.6.1 Database Schema Design

MySQL Server and the "ehfem" schema it hosts use a database model based on the relational model. For the "ehfem" schema's purposes, this means that a table in "ehfem" uses a foreign key to identify one or more of its columns that refer to one or more columns in another table. Figure 3.2 shows the "ehfem" schema. The "ehfem"

schema contains four tables: parent, stand alone device sessions, energy management units, and emu history. PK indicates the table attribute serving as the table's primary key and FK1 indicates the attribute serving as the foreign key. Appendix C provides the SQL queries that create the "ehfem" schema. The following subsections cover each table in detail.

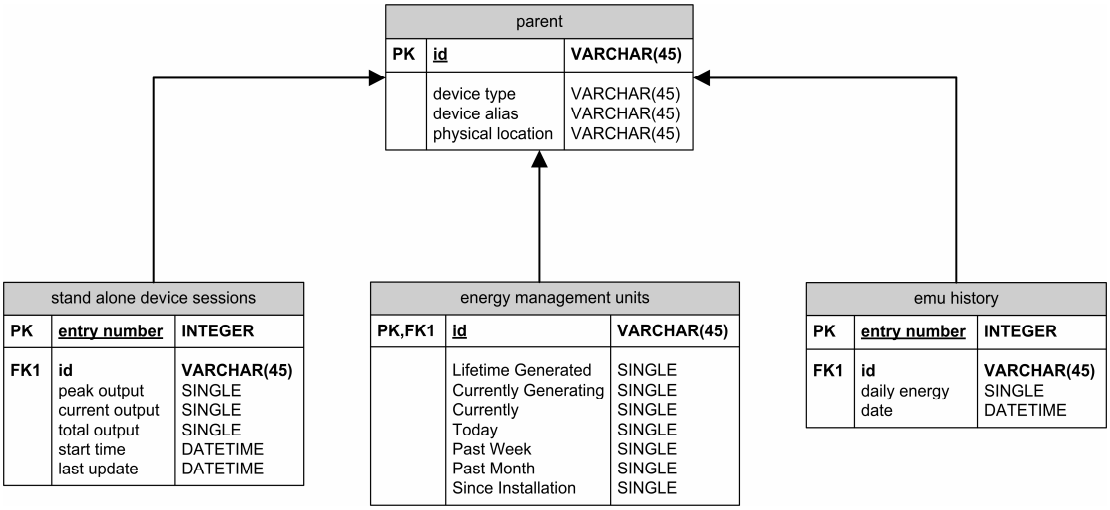


Figure 3.2. "ehfem" schema.

3.6.1.1 The "ehfem" schema's parent table

The parent table provides a non-redundant list of all PMMs attached to the system. Each entry in the table has an "id," "device type," "device alias," and "physical location" attribute. These attributes allow clients accessing the database, namely the display module, to select various PMM groupings to get additional information on. The parent table as a whole simplifies the display module's queries by preventing the need to search multiple tables to find devices of a certain "id," "device alias," or "physical location." This benefit justifies the "parent" table. Table 3.1 shows the parent table filled with simulated information.

id	device type	device alias	physical location
030906000932	Enphase EMU	Group 1	Front room
11:22:33:44:55:66	Stand-alone network device	Bike 1	Back room
123456789012	Enphase EMU	Group 2	Front room
99:88:77:66:55:44	Stand-alone network device	Bike 2	Back room

Table 3.1. The parent table filled with simulated information.

The "id" attribute provides the primary key for this table, which means that it is the attribute that differentiates one parent table entry from another. The fact that each PMM present in the system has a unique "id" but may share the same "device type," "device alias," and "physical location" as other PMMs justifies choosing "id" as the primary key. For the Enphase EMU system design, "id" contains the EMU's serial number. An EMU gathers measurements from multiple exercise machines, so an EMU's "id" represents multiple exercise machines. For the embedded system design, "id" contains the embedded system's MAC address. Each embedded system gathers measurements from a single exercise machine, so the "id" for this PMM type represents a specific exercise machine. A variable length character string with a maximum length of 45 characters stores the "id" attribute.

The "device type" attribute describes what PMM type the device with a certain "id" number uses. This system uses the Enphase EMU and embedded system types of PMM, so the "device type" attribute's possible values are "Enphase EMU" and "Stand-alone network device." Like the "id" attribute, a variable length character string with a maximum length of 45 characters stores the "device type" attribute.

The "device alias" attribute provides a more user friendly way of referring to specific power measurement devices. Potential values might be "Bike 1" or "Group 1," but there is no naming convention since the placement of exercise machines within the new Recreation Center may play an important role in how to label power

measurement devices. A variable length character string with a maximum length of 45 characters stores this attribute.

The final attribute is “physical location.” This attribute describes the PMM’s physical location in the Recreation Center. Like the “device alias” attribute, this attribute also depends on the placement of exercise machines within the new Recreation Center, and a variable length character string with a maximum length of 45 characters stores it.

3.6.1.2 The “ehfem” schema’s stand alone device sessions table

The “stand alone device sessions” table stores exercise sessions transmitted by the embedded system PMM type, where an exercise session represents a gym user’s workout on a specific exercise machine. Table 3.2 shows the “stand alone device sessions” table populated with simulated information. The table’s columns are: “entry number,” “id,” “peak output,” “current output,” “total output,” “start time,” and “last update.”

entry number	id	peak output	current output	total output	start time	last update
1	99:88:77:66:55:44	4	0	8.7234e-005	2009-11-16 17:00:21	2009-11-16 17:14:22
2	11:22:33:44:55:66	12	0	0.000784236	2009-11-16 18:32:43	2009-11-16 18:37:23
3	11:22:33:44:55:66	32	0	0.0024324	2009-11-16 15:09:14	2009-11-16 15:17:34
4	99:88:77:66:55:44	5	0	0.00590835	2009-11-16 07:00:21	2009-11-16 07:14:22
5	99:88:77:66:55:44	5	0	0.00132905	2009-11-16 08:15:23	2009-11-16 08:50:22

Table 3.2. “stand alone device sessions” table filled with simulated exercise sessions.

One of the database module’s constraints, the limits on what information fields the PMMs can provide to the schema, justifies the “stand alone device sessions” table—since the embedded system PMM type requires a different set of attributes to store its measurements than the Enphase EMU PMM type, “ehfem” should contain a table to accommodate this unique representation of measurements.

The “stand alone device sessions” table’s primary key is “entry number.” This attribute is an automatically incrementing integer, so the first exercise session’s “entry number” is 1, the second exercise session’s “entry number” is 2, and so on, which guarantees that each entry’s “entry number” is unique. This attribute must serve as the primary key because the other attributes are not guaranteed to be unique. The MySQL Query Browser, a part of MySQL Administrator, requires that each table contain a primary key. Otherwise, it does not allow administrators to manually alter the tables’ contents, which is an important tool for testing.

The “stand alone device sessions” table’s “id” attribute is a foreign key that links this table’s “id” attribute with the parent table’s “id” attribute. This ensures that only devices with “id” values listed in the parent table can add entries to the “stand alone device sessions” table. Although each entry in the parent table must have a unique “id” number since “id” is parent’s primary key, it is possible for the same “id” number to appear in multiple “stand alone device sessions” entries since “id” is the foreign key of “stand alone device sessions,” not a primary key. Since “id” is a foreign key, it must use the same storage variable as the parent table’s “id,” namely a variable length string with a maximum if 45 characters.

Three single precision floating point variables store the “peak output,” “current output,” and “total output” attributes. “peak output” stores the maximum Wattage reported during the exercise session. “current output” stores the newest Wattage reading reported by the exercise session. “total output” contains the total energy, in Watt-hours, reported by the exercise session. During the course of an

exercise session, the PMM's packet listener program receives new measurements and uses them to continually update these three values.

The final attributes, "start time" and "last update," exist as *datetime* variables. They provide enough information to the display module for it to find exercise sessions by date and time, and calculate statistics like exercise session length. When an exercise session begins, the PMM's packet listener program adds a new entry to the "stand alone device sessions" table and fills in the "start time" and "last update" attributes with the current time. During the course of an exercise session, the packet listener program continually updates the "last update" value with the time at which it received the measurement.

3.6.1.3 The "ehfem" schema's energy management units table

The "energy management units" table stores the information provided by each Enphase EMU PMM type. The same constraint that justifies the "stand alone device sessions" table applies to this table too—the Enphase EMU PMM type provides a different set of statistics than the embedded system PMM type, and this warrants the addition of a table specifically designed to accommodate it. The "energy management units" table's attributes are: "id," "Lifetime Generated," "Currently Generating," "Currently," "Today," "Past Week," "Past month," and "Since Installation." Table 3.3 shows the table with simulated rows.

id	Lifetime Generated	Currently Generating	Currently	Today	Past Week	Past Month	Since Installation
030906000932	3.03	123	123	0.12	0.74	2.78	3.03
123456789012	2.07	341	341	0.9	0.65	1.92	2.07

Table 3.3. "energy management units" table with simulated rows.

Unlike the "stand alone device sessions" table, the "energy management units" table's "id" attribute serves as a primary key and a foreign key. The Enphase

EMU PMM design cannot track individual exercise sessions, so each entry in the table tracks the current state of a single, unique EMU, making the "id" field a valid primary key. The Enphase EMU PMM's script uses the "id" field to locate and continually update the attributes of each EMU's entry in the "energy management units" table.

The "Lifetime Generated" and "Since Installation" attributes store single precision floating point values representing the total energy, in Watt-hours, generated by the EMU. Although they appear redundant, they provide a way to cross-check the information gleaned from the EMU because the Enphase EMU PMM's script extracts these values from two different places on the EMU's web site.

Single precision floating point values also store the "Currently Generating" and "Currently" attributes. These attributes represent the EMU's current power output in Watts. Like the previous two attributes, the EMU provides these values in two places on its web site.

The remaining three attributes, also stored in single precision floating point values, are: "Today," "Past Week," and "Past Month." These attributes store the energy generated, in Watt-hours, over their respective time periods. The EMU itself generates these values; the Enphase EMU PMM's script simply retrieves them.

3.6.1.4 The "ehfem" schema's emu history table

The "emu history" table increases the flexibility of the information retrieved from Enphase EMUs. The need to further address the "limits on which information attributes each PMM type can provide" constraint's effect on the "energy management units" table justifies this table's design. The display module should not

be limited to the four time periods (today, past week, past month, and lifetime) offered by the Enphase EMU. To increase the time periods available to the display module, the "emu history" table stores daily snapshots of each Enphase EMU device. A snapshot script running every 24 hours adds entries to the "emu history" table. A copy of this script appears in Appendix D. The "emu history" table's attributes are: "entry number," "id," "daily energy," and "date." Table 3.4 shows simulated "emu history" entries.

entry number	id	daily energy	date
1	030906000932	.2114	2009-11-09
2	030906000932	.368932	2009-11-10
3	123456789012	.129234	2009-11-22
4	030906000932	.283466	2009-11-22
5	123456789012	.0923438	2009-11-22

Table 3.4. Simulated "emu history" table entries.

Entries in this table may contain the same values for "id," "daily energy," and "date," so an additional attribute, "entry number," serves as a primary key much like the "stand alone device sessions" table. Although this table and the "stand alone device sessions" table use the name "entry number" as their primary key attribute, the attributes are not functionally related; they just happen to have the same name. This table's "entry number" is an automatically incrementing integer value which guarantees that it is unique, unlike the other attributes which could have the same value across multiple entries.

The "id" attribute, a foreign key referring to the "parent" table, stores the Enphase EMU's "id" in a variable length string with a maximum length of 45 characters. There may be many entries in the "emu history" table with the same "id" since the snapshot script copies each EMU's information on a daily basis.

A single precision floating point value stores the “daily energy” attribute. During a snapshot, the snapshot script copies the “Today” attribute from the appropriate entry of “energy management units” into the new “emu history” entry’s “daily energy” attribute. In effect, this stores the energy production, in Watt-hours, of an Enphase EMU device over a 24 hour period.

The final attribute, “date,” resides in a *datetime* variable. The snapshot script fills this value with the date and time when the snapshot took place. This allows the display module to retrieve the energy production of Enphase EMU modules over a specific date range.

3.6.2 Database Schema Development

The database schema’s development strategy focuses on design iterations. The schema’s early versions contain a single table designed to aid in proving the concept of the system as a whole. Each design iteration yields a new version of the schema with better functionality, ultimately leading to the schema’s final version, the one described in section 3.6.1.

The schema’s first version, named “enphase_data,” targets just the Enphase EMU PMM type. At this point in the project, it is unknown what information the embedded system PMM type provides, while it is known exactly what information the Enphase EMU PMM type provides. The first version’s goal is to provide a database schema for a proof-of-concept demonstration, so pursuing a schema to accommodate the Enphase EMU PMM type offers the shortest path to a successful demonstration. The “enphase_data” schema contains a single table named “energy

management units" that is identical to the table covered in section 3.6.1.3 except for the "id" field, which is named "serial number" in this first schema version.

The schema's second version accommodates the requirements of Decade. To make it easier for other Decade users to identify the EHFEM project's schema, the schema's name must change to "ehfem." Additionally, the database engine changes to InnoDB, the one used by the EDAPTS Research Project [29]. Making these changes requires the entire database, which contains only the "energy management units table" at this point, to be recreated by hand. After making these changes in the test environment, the schema is ready to roll out to Decade, where it is created by hand using the MySQL Administrator on Decade. Exporting the test environment's schema could introduce problems to Decade and negate the advantage of keeping the test environment isolated from Decade.

The schema's third version adds support for the embedded system PMM type. This design iteration adds the "parent" and "stand alone device sessions" tables to the "ehfem" schema. It also changes the name of the "energy management units" table's "serial number" attribute to "id" and assigns "id" as the foreign key referring to the "parent" table's "id" column. These changes take place in the test environment first and then roll out by hand to Decade.

The schema's fourth and final version addresses the need to track the daily energy generation of each device in the "energy management units" table. This design iteration adds the "emu history" table to the schema and creates the snapshot script to support the "emu history" table. Like the previous iterations, these developments roll out to Decade by hand.

3.6.3 Database Schema Testing

The testing strategy compliments the development strategy; each database schema's design iteration culminates with a two stage test process that verifies the schema's functionality and accuracy. Figure 3.3 shows the database schema test process.

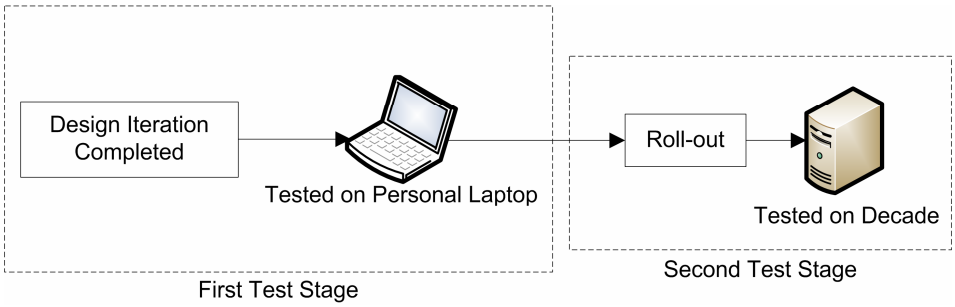


Figure 3.3. Database schema test process.

The test process's first stage takes place on the same test environment used for Enphase EMU PMM testing—a Dell E6400 laptop running the Windows Server 2003 operating system, MySQL Server, phpMyAdmin, MySQL Administrator, and MySQL Query Browser. The schema's development takes place on the laptop; once the design iteration reaches its completion, the laptop tests the schema's functionality and accuracy using a simulated PMM. Functionality refers to the PMM's ability to successfully alter tables and the display module's ability to successfully access tables. Accuracy refers to how well the attributes' data types represent information. If this first stage's tests yield positive results, then the test process's second stage begins.

The test process's second stage consists of rolling out the schema to Decade where a real PMM tests the schema's functionality and accuracy. This two stage test strategy isolates potentially critical problems to the test environment and tests the schema on two different computers, which exposes any portability issues that affect

functionality or accuracy. The safety and portability verification offered by this testing strategy justifies its use in this project.

Tests of the schema's first version, "enphase_data," took place before the EE department provided the EHFEM project with Decade, so testing resided completely within the test environment. This version successfully stored measurements inputted by a simulated Enphase EMU PMM, and a simulated display page successfully retrieved the schema's information. These tests verified the schema's functionality and accuracy, and demonstrated a proof-of-concept.

Tests of the schema's second version, now called "ehfem," focused on verifying that the Enphase EMU PMM type worked with the new schema name and the new database engine. The test environment did not reveal any problems, and after rolling out to Decade, the second testing stage completed without any problems.

Since the schema's third version added new tables to support the embedded system type PMM, this version's testing focused on making sure the automatically incrementing attributes and foreign key constraints functioned properly. Testing also created fake entries in the "stand alone device sessions" table, verifying that the *datetime* data type accurately represented the information expected from the embedded system PMM type.

Testing of the schema's final version, which added support for tracking the daily energy generation of each device in the "energy management units" table, focused on the "emu history" table and the snapshot script. Since the "emu history" table and snapshot script depend on one another, tests targeted both of them at the same time. The snapshot script successfully extracted the appropriate attributes from

each entry in the “energy management units” table and created a new entry in “emu history” containing those attributes. This verified both tables’ accuracy and the script’s functionality. Database queries verified the tables’ functionality. These queries simulated the display module, which the next chapter covers.

CHAPTER 4 : DISPLAY MODULE

4.1 Introduction

Chapter 4 opens by describing the display module's overall goals and constraints. Then, it describes each display module option considered and shows how well they fulfill the overall goals and constraints. Next, it justifies the display module option chosen for implementation in terms of design, development, and testing. The final section describes a different display module implementation developed by a senior project student.

4.2 Goals and Constraints

Figure 4.1 shows the display module block diagram. The display module's first goal, accomplished by the display software, is the execution of database queries to retrieve database information. The display software uses the query results to accomplish the second goal—the generation of its output capabilities like graphs and charts. The display module's third and final goal is to present the output options on a physical display device such as a television or computer monitor.

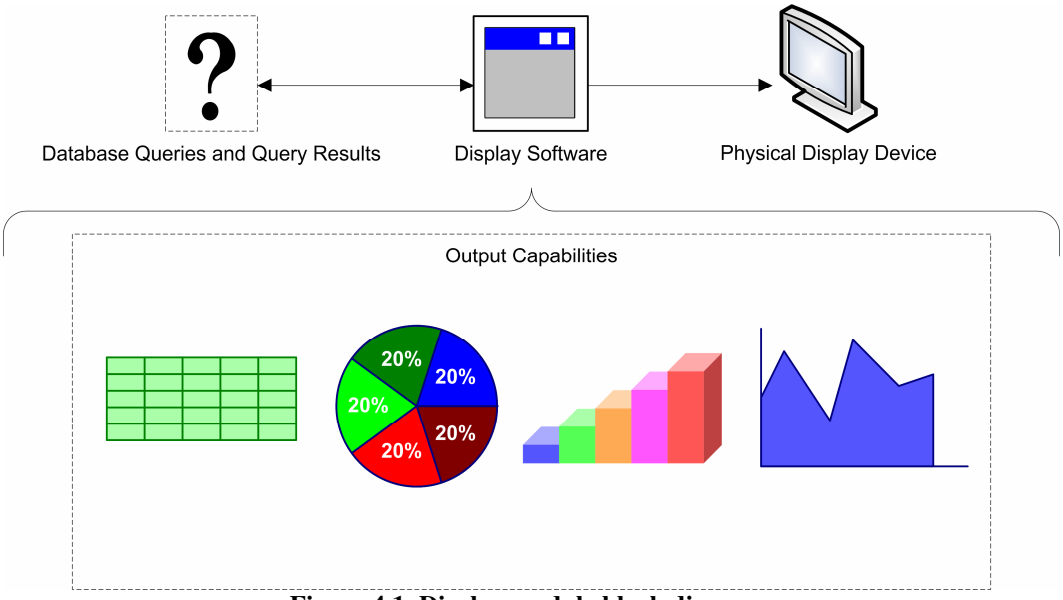


Figure 4.1. Display module block diagram.

The display module's constraints are: the display software's database interfacing capabilities, the display software's output capabilities, the audience the display software can serve, the design cost, the monetary cost, and inverter compatibility. The database interfacing capabilities constraint ensures that the display software can connect to the database and retrieve the information it needs. The output capabilities constraint refers to the graphical representations the display software generates, in terms of graphs, charts, colors, animations, and how interactive the graphics are. The next constraint, the audience the display software can serve, denotes the audience of people that could access the display software—a TV in a gym would only serve gym-goers while anybody can access a website. Design cost represents the amount of work required to realize a display module option. Monetary cost accounts for how much money the display module software costs. The final constraint, inverter compatibility, accounts for any limits on which inverter types the display module option works with.

4.3 Options Considered

Four display software options potentially provide the functionality required by the display module: an Adobe Flash-based program, a Java Swing-based program, an exercise machine user interface-based program, and Enphase's Enlighten web page. Table 4.1 shows the options' ratings for each constraint along with total scores. Database interfacing, audience, and inverter compatibility are the most important criteria since they represent how well an option fulfills the display module's goals. The output capabilities criterion is slightly less important since the previous three criteria govern the information available for output as well as the output device

available to reach the audience. Design cost is less important than the previous criteria because an option with better capabilities is more desirable than a less capable but easier option. Finally, monetary cost is the least important criterion because none of the options are too costly to pursue Cal Poly funding for.

		Options							
		Adobe Flash-based display program		Java based display program		Exercise machine user interface-based program		Enphase Enlighten web page	
Criteria	**Weight**	Rating	Score	Rating	Score	Rating	Score	Rating	Score
Database Interfacing	5	5	25	5	25	1	5	1	5
Output Capabilities	4	5	20	5	20	1	4	3	12
Audience	5	5	25	5	25	2	10	3	15
Design cost	3	5	15	2	6	1	3	5	15
Monetary cost	2	3	6	5	10	5	10	5	10
Inverter compatibility	5	5	25	5	25	5	25	1	5
Total (higher is better)			**116**		**111**		**57**		**62**

Table 4.1. Decision matrix for the display module options.

4.3.1 Adobe Flash-based Display Program

An Adobe Flash-based display program uses Adobe's Flash technology, the same technology used for many websites and banner advertisements, to deliver statistics using an interactive user interface. A display program utilizing Adobe's product takes advantage of existing tools that aid in the creation of graphics. One tool, Adobe Flash CS4 Professional [30], provides a platform for developing any type of Flash content. A much more cost effective tool is FusionCharts, a stand-alone program that aids in the creation of "animated and interactive Flash charts for web and desktop applications" [31].

FusionCharts offers built-in support for MySQL Database interfacing [31], making it ideal for the display module of this project and achieving a high score for the database interfacing constraint. In terms of output capabilities, FusionCharts covers “45 types of 2D/3D charts including line, area, bar, column, pie, doughnut (donut), combination, scatter, bubble, scroll charts etc.” [31]. This abundance of graphics justifies the option’s high output capabilities constraint score. The option also scores highly for the audience constraint because Decade could host a web page containing FusionCharts graphics, maximizing the audience size that can access the display module’s output capabilities. This option’s design cost constraint score is also high since FusionCharts does not require the developer to know Flash (or any programming languages for that matter, barring PHP to interface with the database), and provides plenty of templates. This means that the tool handles the graph and chart rendering rather than the developer. The only potential disadvantage of this design is the monetary cost constraint—if the free version of FusionCharts [32] lacks the options to fulfill all constraints, the full version costs $99 or more. Inverter compatibility is not a problem since this option gets information only from the database.

4.3.2 Java based Display Program

A Java based display program uses the Java language coupled with the Swing toolkit and other packages to create graphics from scratch. The Java Foundation Classes Swing toolkit [33] helps the developer create applications that use a Graphical User Interface (GUI) [34]. Additional packages like JFreeChart [35],

JGoodies [36], and SwingX [37] offer even more help to the developer, especially in the realm of graphs and charts.

This display module option scores highly in the database interfacing constraint thanks to an application programming interface called Java Database Connectivity (JDBC) [38]. JDBC provides access to relational databases like MySQL [39], making it possible to retrieve information from the database module and display it in a GUI.

Swing and the additional packages address the output capabilities constraint as successfully as the Adobe Flash-based design. They provide comparable flexibility, and the graphs and charts they help generate are visually appealing like the Adobe Flash-based option's.

Like the Adobe Flash-based option, a web page can encapsulate a Java based display program. Since it can reach the same audience as the Adobe Flash-based option, this design also completely satisfies the audience constraint.

Unlike the Adobe Flash-based option, this option scores poorly for the design cost constraint. While the Adobe Flash-based option takes advantage of development tools that handle graphics rendering, this option requires the developer to write code in Java to render each graphic. Since graphics rendering is in the developer's hands for this option, the Java based display program requires a much more technical and elaborate design.

The Java language is free to use, so this design has no monetary cost, yielding a high score for that constraint. This design also works with any inverter type since it uses the database's information.

4.3.3 Exercise Machine User Interface-based Display Program

An exercise machine user interface-based display program takes advantage of the exercise machines' built-in user interface to display output capabilities. This design adds more functionality directly to the software driving the exercise machine's user interface or takes advantage of protocols like CSAFE [16] to add more functionality without modifying the exercise machine's software.

This design fails for one of the same reasons the PMM's CSAFE Program of section 2.3.1 fails—the lack of a development platform to reprogram the Precor EFX546i to add functionality to its software. In addition, the EFX546i's manual [40] reveals no ways to add functionality to the display using the CSAFE interface.

Since there is no way to add functionality to the exercise machine user interface, this design is not feasible. Assuming it were feasible, the design would score poorly in the database interfacing and design cost constraints because getting information to the exercise machine software requires an elaborate software program. This program would have to translate CSAFE messages to database queries and back again, and implement its own CSAFE protocol to pass the CSAFE messages. The design would also score poorly for the audience constraint since only exercise machine users could use it. This design's monetary cost would not be significant since Cal Poly students would design the software component. The design would work with any inverter type.

4.3.4 Enphase Enlighten Web Page

The Enphase Enlighten Web Page option uses a feature included with the Enphase EMU PMM type. The EMU itself automatically uploads energy production

information to Enphase's Enlighten website [41], where customers can log in and view their system's energy harvesting information in many ways [42].

This option scores poorly in database interfacing since it completely negates the need for a database module, instead retrieving information directly from the Enphase EMU. The Enlighten website's output capabilities include graphical representations of daily, weekly, monthly, and lifetime totals [42]. This option doesn't score as highly as the Adobe Flash-based and Java based options since Enphase completely controls its output capabilities—there is no flexibility to design custom graphics. The Enlighten website, although widely accessible, limits the audience somewhat since users need to use a customer login to gain access, resulting in a slightly lower score for the audience constraint than the Adobe Flash-based and Java Swing-based options. The design scores strongly for the design cost constraint since there is no design work; Enphase sells the design with their EMUs. This also means no additional monetary cost, since the EMU is part of the Enphase EMU PMM. This design scores very poorly for the inverter compatibility constraint since it only works with Enphase Microinverters. This means the design only works for Enphase EMU type PMMs.

4.4 Option Selected

The Adobe Flash-based option best meets the constraints according to table 4.1. Sections 4.4.1 through 4.4.3 cover the Adobe Flash-based module's design, development, and testing.

4.4.1 Adobe Flash-based Design

The Adobe Flash-based design focuses on three goals: the selection of graphics, the selection of which user-configurable options those graphics support, and the layout of graphics on a web page.

Each web page graphic should present database information in a clear, colorful, and animated way. FusionCharts provides templates for a number of graphs and charts. Figure 4.2 shows Enphase EMU statistics using a bar graph based on a template. This bar graph utilizes clear axis labels, different colors for each bar, and employs animation to make the bars start at the bottom axis and extend upwards until they reach their value on the vertical axis. Bar graphs could show a specific PMM's energy production or compare the energy production of different exercise machine groups, different exercise machine locations, different PMM types, or different time intervals. Appendix F supplies the source code for this bar graph.

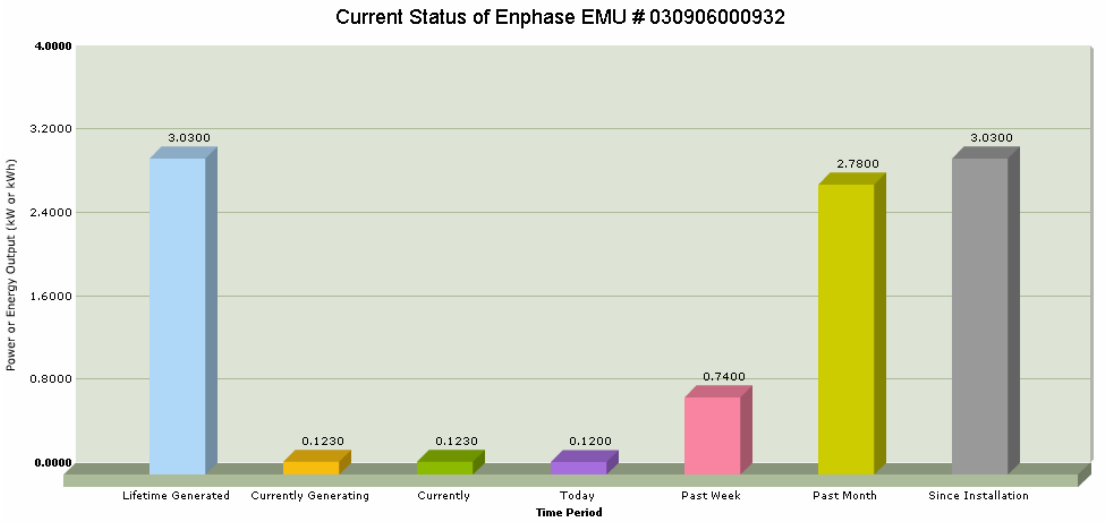


Figure 4.2. A bar graph showing Enphase EMU statistics.

FusionCharts also has line chart templates. Figure 4.3 shows a line chart of embedded system measurements based on a template. This example highlights the line chart's ability to show energy production over a time interval. This line chart also

uses animation; the lines first appear as horizontal bars, and then bend to where each data point is. This graph's source code appears in Appendix H.

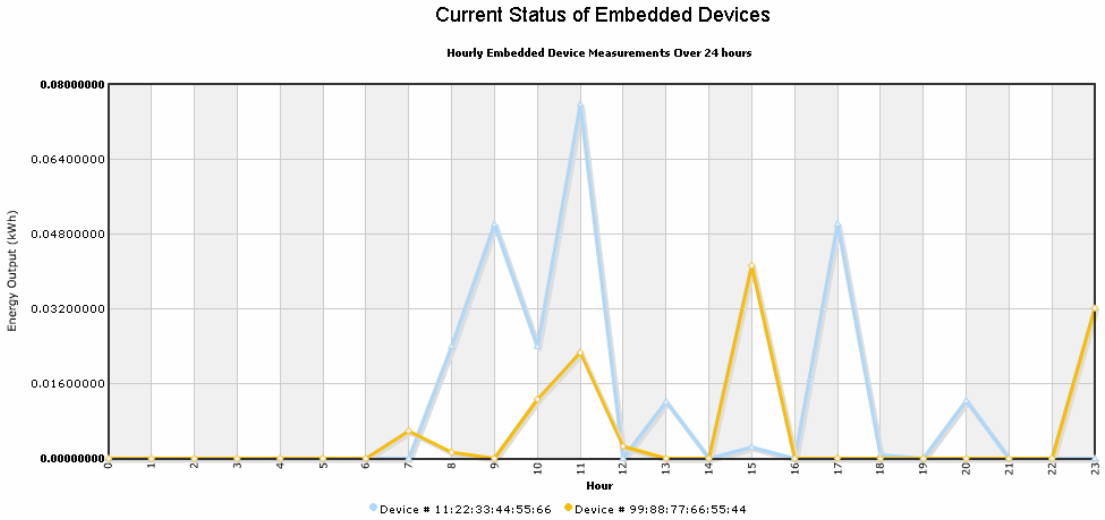


Figure 4.3. A line chart of embedded system measurements.

Users might also want to see what percentage of total energy production each PMM type contributes—a doughnut chart like Figure 4.4 accomplishes this. In this example, the Stand-alone devices only contributed 0.37 kWh while the Enphase devices contributed 5.10 kWh. For its animation, the doughnut chart draws the doughnut in a counter-clockwise fashion. Appendix E provides the doughnut chart's source code.

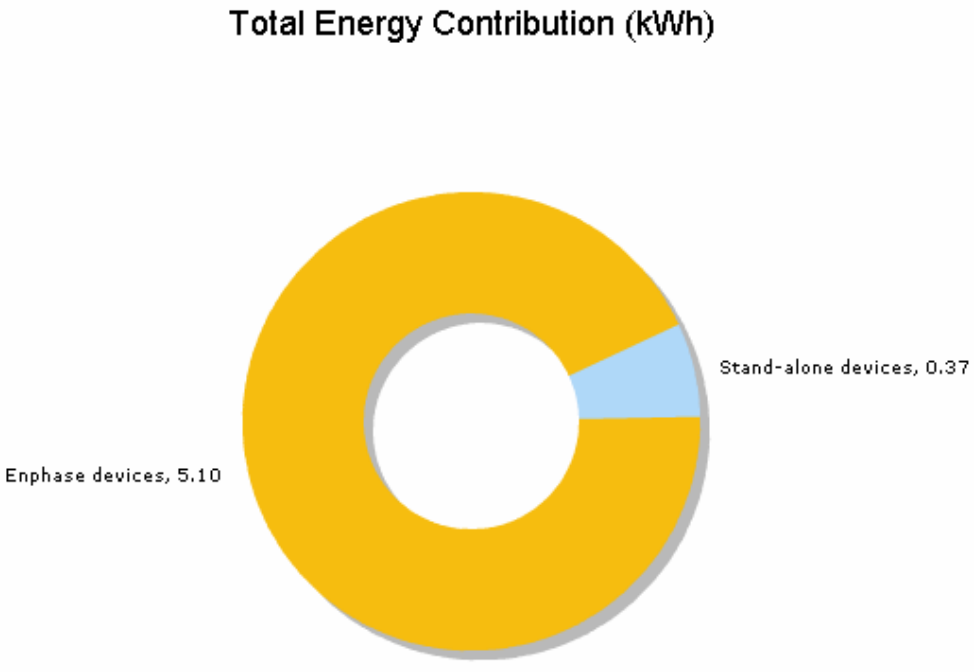


Figure 4.4. A doughnut chart showing total PMM energy contribution.

Although static graphs and charts may satisfy many users, they do not fully exploit the database module's capabilities. The database module stores statistics like date, time, device type, device alias, device location, and more; a collection of static graphs and charts can only supply a few comparisons of the total comparisons possible. To address this limitation, the web page contains radio buttons, drop-boxes, and text input boxes, providing the user with a way to configure what information the graphs show. Figures 4.5, 4.6, and 4.7 show what radio buttons, drop-boxes, and text input boxes look like. Appendix J provides the source code for them.

Figure 4.5. Three radio buttons.

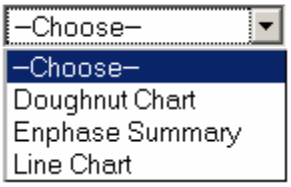


Figure 4.6. A drop-box.

Figure 4.7. A text input form.

The web page's layout determines the text's location on the web page and where each graphic, radio button, drop-box, and text input box is. The text includes a title and a brief description of the web site's purpose, and resides at the web page's top. The layout groups together radio buttons, drop-boxes, and text input boxes with the graphics they alter, which functions as a visual cue to users that helps them recognize which graphic a radio button, drop-box, or text input box changes. Another important layout feature is the screen width required by the website. Tracking internet browser viewport sizes to figure out the web site layout's optimal width is out of this project's scope. Instead, the layout assumes a width of 1000 pixels based on the Cal Poly Web Authoring Resource Center (WARC)'s recommendation [43]. The final issue the layout must address is Cal Poly's IT Responsible Use Policy. If this webpage becomes an official Cal Poly web page, it must follow the WARC's style requirements [44] and the header contents guidelines [45]. The design does not require the web page to be an official Cal Poly web page, so it instead follows the policies of unofficial web pages [46].

4.4.2 Adobe Flash-based Development

The Adobe Flash-based display module's development strategy breaks up the development into cycles. The initial development cycles focus on the layout. Subsequent development cycles incrementally implement more advanced features, all the while refining the layout. The cycle completes when all features are implemented.

The first development cycle establishes a layout using placeholder graphs and charts—that is, the graphs and charts will not yet show information from the

database. This cycle's result is a web site with title text and specific locations for each graphic type and its corresponding drop-boxes, radio buttons, and text input forms.

The second development cycle replaces the placeholder charts. To do this, the developer exploits FusionCharts' database interfacing features to generate multiple graphs using static database queries. These graphs replace the placeholder graphs on the webpage. At this point, the drop-boxes, radio buttons, and text input forms have no functionality since the graphs always show the same set of information.

The third and final development cycle adds support for dynamic database queries to the web page. The web page's display software formulates database queries based on what options or buttons the user selects from the drop-boxes and radio buttons, and what information they type into the text input forms. The display software then builds charts and graphs to represent those queries' results. This development cycle's product is a finalized web page that allows users to configure what each graph shows.

4.4.3 Adobe Flash-based Testing

The Adobe Flash-based display module's testing strategy compliments the development strategy. Tests at the end of each development cycle verify that its product accomplishes its goal. Like the database module's testing strategy, testing occurs first in a test environment isolated from Decade. If the tests yield positive results, the development cycle's product migrates to Decade where it is accessible to the World Wide Web.

Testing for the first development cycle's web page focuses on the layout. Tests should show that this cycle's final web page correctly renders the layout design in the test environment and on Decade.

The second development cycle's tests should show graphics correctly rendering the information provided by static database queries. This demonstrates that the web page's display software can successfully query the database, and that the graphics can successfully utilize the results provided by those queries. Tests that take place in the test environment access a test database filled with fake entries to simulate what a database module with interesting energy production activity might contain. This setup exposes any problems the display software might have while querying for data and reading that data into graphics. Decade may or may not have a database module storing energy production activity, so tests on it might only verify database connectivity. If Decade's database module does contain energy storage activity, it tests and verifies the web page's functionality in its final environment.

The final development cycle's tests should show that the drop-boxes, radio buttons, and text input forms correctly manipulate the graphics. By showing this, it verifies that the display software formulates the correct database queries given any combination of drop-box selection, radio button selection, or text input. It also verifies that the display software redraws the graphics to show the information associated with the drop-box selection, radio button selection, or text input. Tests performed in the test environment can again take advantage of simulated energy production activity to expose any issues arising from the additional database queries

or the graphics redrawing. Decade's usefulness for testing this development cycle is comparable to the second design cycle's.

4.5 A Different Display Module Implementation

For his senior project, Alex Chernetz developed a display module implementation [47]. Alex chose to pursue the Java based design; although this document defends the Adobe Flash-based display module, Alex's work offers a foundation that future senior project students could use to create a fully functional display module. This section describes the capabilities of Alex's display module and the problems that future students need to work on.

Alex wrote his program using the Java language [47]. Instead of creating graphics from scratch, he used the JFreeChart [35], JGoodies [36], and SwingX [37] packages to help him generate bar graph, line chart, and "chemical and energy offsets" displays. Additionally, Alex implemented buttons to navigate between each display. Figures 4.8 through 4.11 show each display.

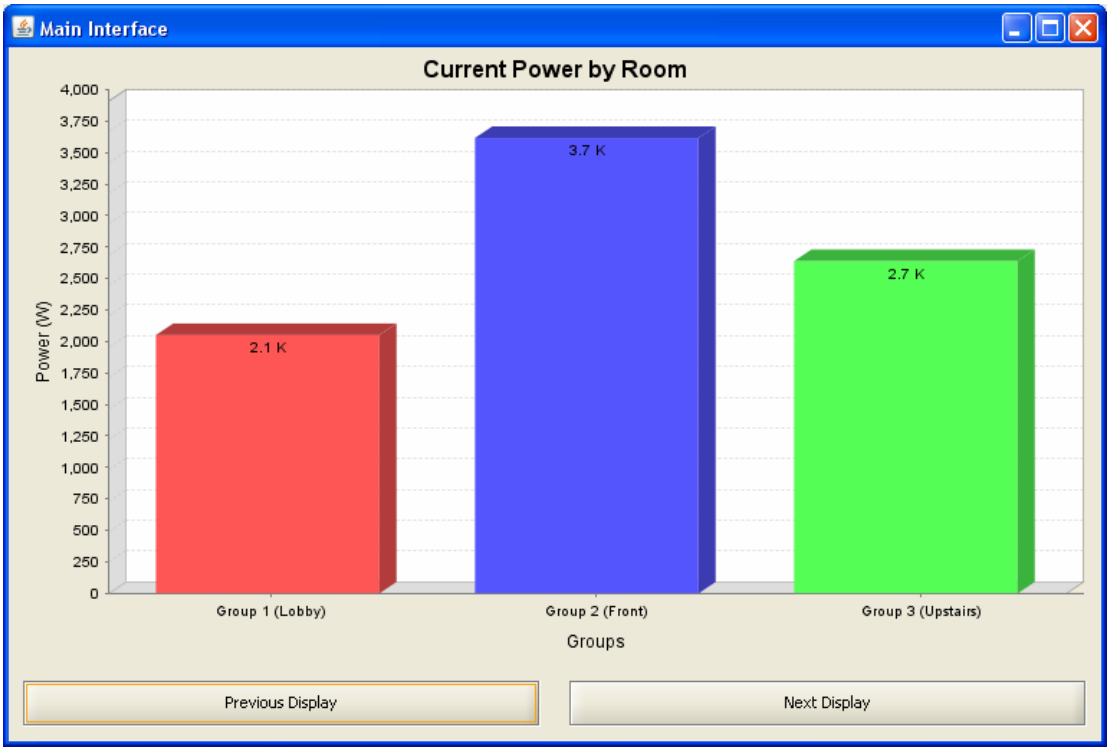


Figure 4.8. Alex's bar graph display.

Alex's bar graph animates the bars by extending them up from the "Groups" axis shown in figure 4.8, much like the bar graph in figure 4.2. Alex's bar graph's overall presentation is comparable to the FusionCharts bar graph; they both employ a three-dimensional bar effect, colors, and animation.

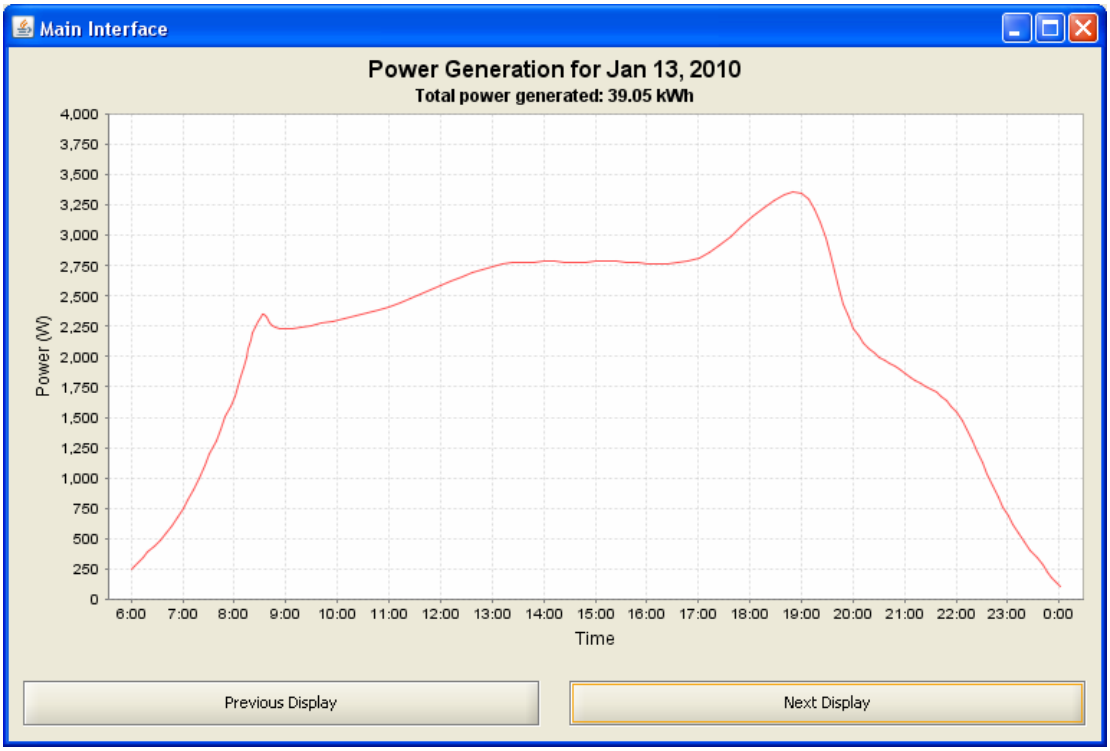


Figure 4.9. Alex's line chart display.

Alex's line chart display, shown in figure 4.9, looks similar to the FusionCharts chart in figure 4.3. The major differences are the FusionCharts chart's use of shadows below the lines and Alex's animation, which draws the line along its data points from left to right.

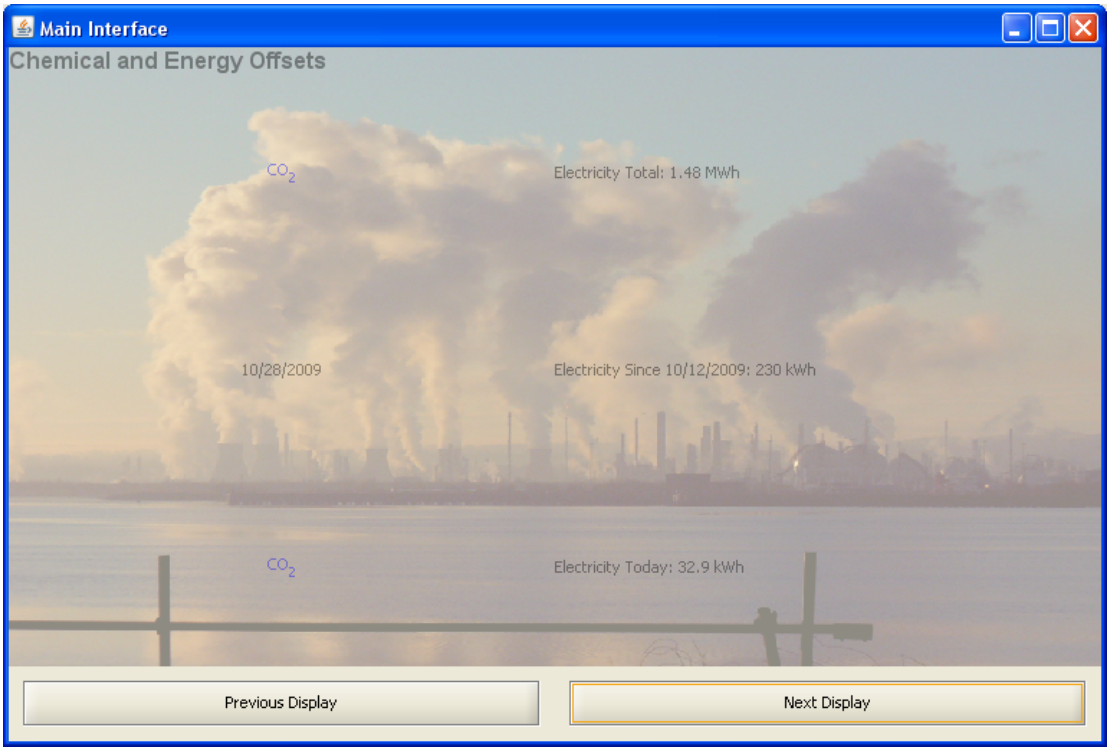


Figure 4.10. Alex's chemical and energy offsets display with the first background image.

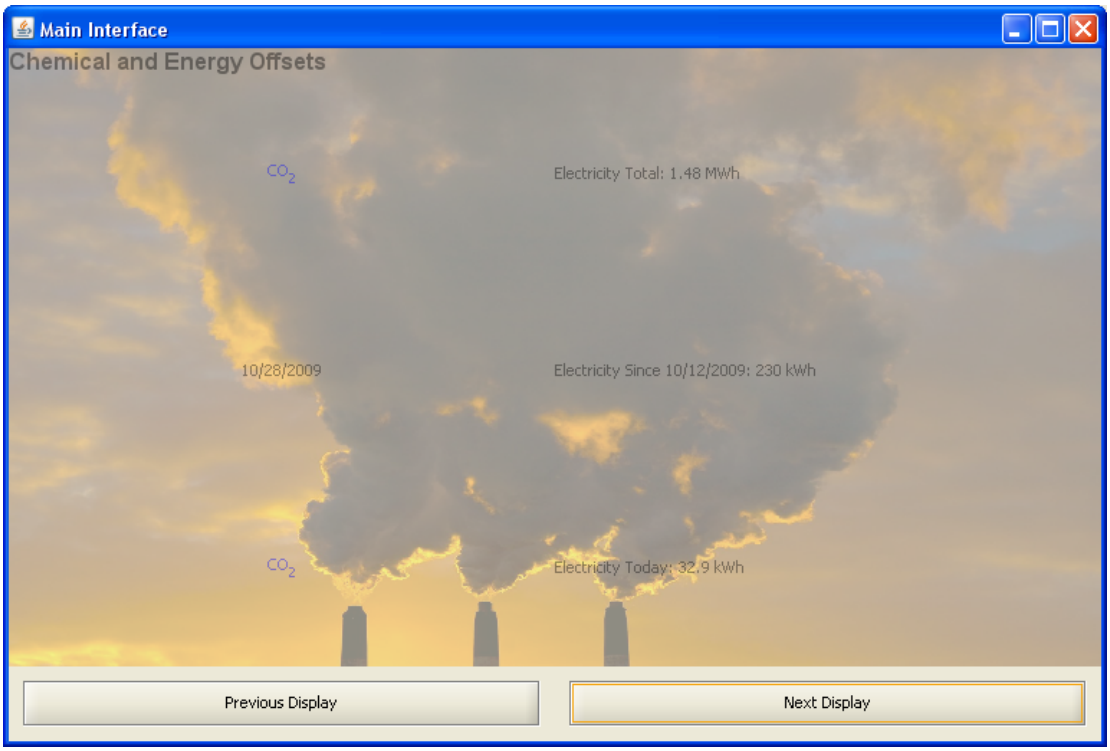


Figure 4.11. Alex's chemical and energy offsets display with the second background image.

Alex called his last display "Chemical and Energy Offsets." Figures 4.10 and 4.11 show the two background images his program chooses from. This display is a

work in progress; given more time, Alex planned to add more graphics to show how much CO_2 or other chemicals were offset by the exercise machines' energy production. Figure 4.12 illustrates a display module concept drawing supplied to him, which inspired his "Chemical and Energy Offsets" display.

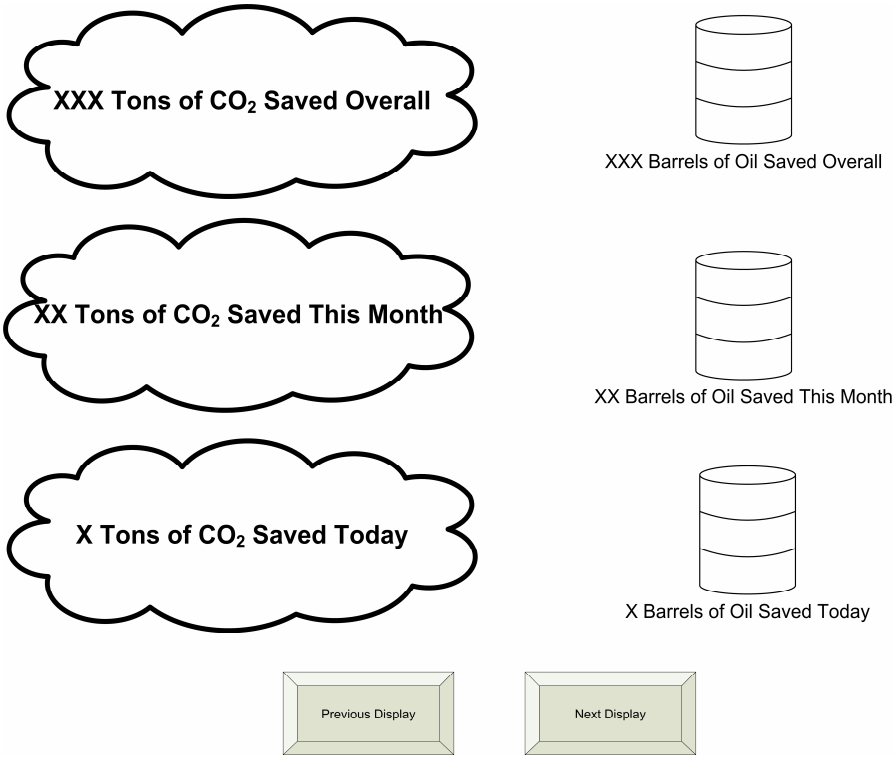


Figure 4.12. The concept drawing that inspired Alex's "Chemical and Energy Offsets" display.

Students continuing Alex's work have four major problems to solve. The first problem requires them to add drop-boxes, radio buttons, or text input forms to the displays to allow users to configure what the graphs and charts show. To address the second problem, students must add database connectivity and implement queries to retrieve database information based on the drop-boxes, radio buttons, and text input forms. For the third problem, students must complete Alex's unfinished display and remove some bugs. The fourth and final problem requires students to convert the display program to an applet, allowing Decade to serve it as a web page.

CHAPTER 5 : CONCLUSION

5.1 Conclusion

This thesis defends a near real-time exercise machine power statistics reporting system's design, development, and testing. This system represents a solution to a novel problem that no state-of-the-art systems address.

The system presented in this document satisfies the three major requirements presented in the introduction. This document presented two PMM types, each providing a design, development, and testing approach that satisfies the requirements of measuring an exercise machine's power and energy production and transmitting those measurements. This document also proposes a database module's design, development, and testing process that meets the second major requirement: the storage of measurements in an easily searchable manner. Finally, the display module section offers a design, development, and testing approach that meets the third major requirement: the capability to display measurements in an interactive and widely accessible way.

Although this document defends specific approaches for the system's modules, other approaches exist. In the PMM's case, Power-One [48], Fronius [49], and Schneider Electric [50] offer products comparable to Enphase; they might also fulfill the PMM's requirements. For the database module, MySQL is one of many database servers that satisfy the database module's requirements. Other database servers include Microsoft SQL Server [51], Oracle Database [52], PostgreSQL Database Server [53], and DB2 Express-C [54]. There are also many possible database schema designs that could fulfill the schema's requirements; this document offered only one schema. Finally, alternatives exist for the display module as well.

Many products and computing languages can produce the graphics described in this document, such as XML/SWF Charts [55], Open Flash Chart [56], and the PHP, Javascript, and C++ languages. A comprehensive analysis of the graphics produced by various products and computing languages was out of this project's scope. Another concern is that the display module's wide accessibility requirement limited the display delivery method to a web site. If this requirement were relaxed, the display module might include approaches that specifically target the new Recreation Center's television displays.

While the system presented in this paper meets its requirements, additional work could implement system modules. If the energy harvesting hardware uses non-Enphase inverters, then future students could implement the embedded system PMM option. The display module also offers work to future students, whether they choose to continue Alex's work or pursue the Adobe Flash-based option.

Furthermore, additional research and planning could address the system's installation challenges, upkeep, failure recovery strategy, maintenance costs, and mean time between failures. Students or faculty installing the system should work closely with the Recreation Center to decide how to install PMM equipment on exercise machines, how to maintain them, and how to recover from failures. The system uses off-the-shelf devices in its modules, so research on expected maintenance costs and mean time between failures should begin by searching the devices' documentation or contacting their manufacturers.

Additional work could also add desirable features to the overall system. One desirable feature might use CSAFE to run an interactive software program on each

individual exercise machine's user interface. Such a program could utilize the system's capabilities to allow an exercise machine user to browse energy harvesting statistics during their workout session. The program could also implement connectivity with other exercise machines to allow users to compete with individual users or groups of users to see who can generate the most energy. If the system were implemented on multiple college campuses, it would be possible to extend the programs capabilities to support inter-school energy generation contests.

Yet more work might yield a system that tracks all of Cal Poly's renewable energy production; or better yet, a system that tracks every renewable resource in the world, allowing users to make any comparisons.

REFERENCES

[1] M. Lum and J. Yuen. (2009, Dec.) "Energy Harvesting from Elliptical Machines: DC Converter Troubleshooting." [Online]. Available: http://digitalcommons.calpoly.edu/eesp/12/ [Jan. 26, 2010].

[2] J. Rounsevell, C. Shubert, M. Snitowsky, and A. Wong. (2009) "Harvesting Human Exercise Power at the Cal Poly Rec Center: Exercise Bike Power Generator II." [Online]. Available: http://digitalcommons.calpoly.edu/mesp/13/ [Jan. 26, 2010].

[3] The Green Microgym. "Grid Tied Equipment." Internet: http://thegreenmicrogym.com/grid-tied-equipment/, [Feb. 3, 2010].

[4] The Green Microgym. "Our Green Advantage." Internet: http://thegreenmicrogym.com/our-green-advantage/, Jan. 1, 2010 [Feb. 3, 2010].

[5] The Green Revolution, Inc. "About the Technology > FAQs." Internet: http://www.egreenrevolution.com/faqs.aspx?setting=1, [Feb. 3, 2010].

[6] ReRev. "Welcome." Internet: http://rerev.com/default.html, [Feb. 3, 2010].

[7] B. Trelstad. (2009, Apr. 1). "Update on OSU human power installation." *Ecologue*. [Online]. Para. 3. Available: http://oregonstate.edu/sustainability/blog/2009/04/update-on-osus-human-power-installation/ [Jan. 26, 2010].

[8] D. Santschi. (2009, Sep. 11). "Workouts on 20 machines feed the Cal State San Bernardino rec center's power grid." *The Press-Enterprise*. [Online]. p.5. Available: http://rerev.com/downloads/Articles/Workouts_On_20.pdf [Jan. 26, 2010].

[9] L. Fong. (2009, July 29). "A New Power Work Out." *Oregon Daily Emerald*. [Online]. Available: http://www.dailyemerald.com/2.2354/a-new-power-work-out-1.189714 [Jan. 26, 2010].

[10] N. Lorenzo. (2009, Oct. 10). "Aurora Communicator Monitor tool for Aurora Inverters User Manual." [Online]. Available: http://www.power-one.it/digilab/Files/Aurora_Communicator_User_Manual_v0200.pdf [Jan. 26, 2010].

[11] Power-One. "Power-One Renewable Energy Downloads." Internet: http://www.power-one.com/renewable-energy/downloads.php, [Feb. 3, 2010].

[12] Pacific Gas and Electric Company. "SmartMeter™ Benefits." Internet: http://www.pge.com/myhome/customerservice/meter/smartmeter/benefits/, [Feb. 3, 2010].

[13] Pacific Gas and Electric Company. "SmartMeter™ System—How It Works." Internet: http://www.pge.com/myhome/customerservice/meter/smartmeter/howitworks/, [Feb. 3, 2010].

[14] Enphase Energy. "Envoy (EMU)." Internet: http://www.enphaseenergy.com/products/products/envoy.cfm, [Feb. 3, 2010].

[15] CSAFE Steering Committee. "Welcome to CSAFE." Internet: http://www.fitlinxx.com/csafe/, Aug. 4, 2004 [Feb. 3, 2010].

[16] CSAFE Steering Committee. "CSAFE Implementation Notes." Internet: http://www.fitlinxx.com/csafe/Notes.htm, Aug. 4, 2004 [Feb. 3, 2010].

[17] Roalan Ltd. "Embedded CPU Wireless Module." Internet: http://www.roalan.com/embedded_cpu_wifi_module.htm, [Feb. 3, 2010].

[18] Neteon Technologies, Inc. "SW1001T." Internet: http://www.neteon.net/Product/336-91-529/SW1001T, [Feb. 3, 2010].

[19] Lantronix, Inc. "MatchPort® b/g." Internet: http://www.lantronix.com/device-networking/embedded-device-servers/matchport.html, [Feb. 3, 2010].

[20] Silex Technology America, Inc. "SX-560." Internet: http://www.silexamerica.com/products/wireless_modules/sx-560.html, [Feb. 3, 2010].

[21] Quatech, Inc. "Airborne Embedded Wireless Device Server Modules." Internet: http://www.quatech.com/catalog/airborne_wirelessdeviceserver_modules_emb.php, [Feb. 3, 2010].

[22] Lay Networks. "Comparative analysis - TCP - UDP." Internet: http://www.laynetworks.com/Comparative%20analysis_TCP%20Vs%20UDP.htm, [Feb. 3, 2010].

[23] J. Postel. (1981, September). "Functional Specification." *Transmission Control Protocol*. [Online]. Available: http://tools.ietf.org/html/rfc793#section-3.1 [Jan. 26, 2010].

[24] J. Postel. "User Datagram Protocol." Internet: http://tools.ietf.org/html/rfc768, Aug. 28, 1981 [Jan. 26, 2010].

[25] Oracle Corporation. "Why MySQL?." Internet: http://www.mysql.com/why-mysql/, [Feb. 3, 2010].

[26] Oracle Corporation. "MySQL Downloads." Internet: http://dev.mysql.com/downloads/, [Feb. 3, 2010].

[27] Oracle Corporation. “MySQL GUI Tools Bundle: Archived Downloads.” Internet: http://dev.mysql.com/downloads/gui-tools/5.0.html, [Feb. 3, 2010].

[28] Oracle Corporation. “Download MySQL Workbench.” Internet: http://dev.mysql.com/downloads/workbench/5.2.html, [Feb. 3, 2010].

[29] EDAPTS Smart Transit System Project. “The EDAPTS Project.” Internet: http://itrans.calpoly.edu/EDAPTS/, [Feb. 3, 2010].

[30] Adobe Systems Incorporated. “Adobe Flash CS4 Professional.” Internet: http://www.adobe.com/products/flash/, [Feb. 3, 2010].

[31] InfoSoft Global (P) Ltd. “FusionCharts v3.” Internet: http://www.fusioncharts.com/, [Feb. 3, 2010].

[32] InfoSoft Global (P) Ltd. “FusionCharts Free.” Internet: http://www.fusioncharts.com/free/, [Feb. 3, 2010].

[33] Sun Microsystems, Inc. “The Java Foundation Classes.” Internet: http://java.sun.com/products/jlf/ed2/book/HIG.Classes.html, [Feb. 3, 2010].

[34] Oracle Corporation. “Trail: Creating a GUI With JFC/Swing.” Internet: http://java.sun.com/docs/books/tutorial/uiswing/, [Feb. 3, 2010].

[35] Object Refinery Limited. “Welcome To JFreeChart.” Internet: http://www.jfree.org/jfreechart/, Apr. 20, 2009 [Feb. 3, 2010].

[36] JGoodies. “Java User Interface Design.” Internet: http://www.jgoodies.com/, [Feb. 3, 2010].

[37] Oracle Corporation. “swingx Project home.” Internet: https://swingx.dev.java.net/, [Feb. 3, 2010].

[38] Oracle Corporation. “The Java Database Connectivity (JDBC).” Internet: http://java.sun.com/javase/technologies/database/index.jsp, [Feb. 3, 2010].

[39] Oracle Corporation. “JDBC.” Internet: http://java.sun.com/javase/6/docs/technotes/guides/jdbc/, [Feb. 3, 2010].

[40] Precor Incorporated. (2006, Nov. 16). *Product Owner’s Manual*. [Online]. Available: http://www.precor.com/pdf/oms/49098-110%20EFX546i%20Lit%20Kit_0608_EN.pdf [Jan. 26, 2010].

[41] Enphase Energy. “Products.” Internet: http://www.enphaseenergy.com/products/products/index.cfm, [Feb. 3, 2010].

[42] Enphase Energy. “Enlighten.” Internet: http://www.enphaseenergy.com/products/products/enlighten.cfm, [Feb. 3, 2010].

[43] Cal Poly Web Coordination Team. “Designing Pages For The Web.” Internet: http://www.warc.calpoly.edu/building/page_design/fluid_vs_fixed_layout.html, Mar. 20, 2009 [Jan. 26, 2010].

[44] Cal Poly Web Coordination Team. “Style Requirements.” Internet: http://www.warc.calpoly.edu/stylerequirements/index.html, Oct. 31, 2008 [Jan. 26, 2010].

[45] Cal Poly Web Coordination Team. “University Identity and Communication Elements Required for Official Cal Poly Web Pages.” Internet: http://www.warc.calpoly.edu/universityid/elements.html, Oct. 31, 2008 [Jan. 26, 2010].

[46] P. Zingg. “Information Technology Resources Responsible Use Policy.” Internet: http://www.its.calpoly.edu/Policies/RUP-INT/#e12, Apr. 11, 2003 [Jan. 26, 2010].

[47] A. Chernetz. (2009, Dec.). “Exercise Power Grid Display and Web Interface.” [Online]. Available: http://digitalcommons.calpoly.edu/cpesp/2/ [Jan. 26, 2010].

[48] Power-One. “Overview.” Internet: http://www.power-one.com/renewable-energy/, [Feb. 3, 2010].

[49] Fronius International GmbH. “Fronius DATCOM.” Internet: http://www.fronius.com/cps/rde/xchg/SID-509A7D42-3BA1B75C/fronius_usa/hs.xsl/2714_1458.htm, [Feb. 3, 2010].

[50] Schneider Electric. “Xantrex.” Internet: http://www.schneider-electric.com/sites/corporate/en/products-services/renewable-energies/products-offer/range-presentation.page?c_filepath=/templatedata/Offer_Presentation/3_Range_Datasheet/data/en/shared/renewable_energies/xantrex.xml, [Feb. 3, 2010].

[51] Microsoft. “SQL Server 2008.” Internet: http://www.microsoft.com/sqlserver/2008/en/us/, [Feb. 3, 2010].

[52] Oracle Corporation. “Oracle Database 11g Release 2.” Internet: http://www.oracle.com/technology/products/database/oracle11g/index.html, [Feb. 3, 2010].

[53] PostgreSQL Global Development Group. “Home.” Internet: http://www.postgresql.org/, [Feb. 3, 2010].

[54] IBM. “DB2 Express-C.” Internet: http://www-01.ibm.com/software/data/db2/express/, [Feb. 3, 2010].

[55] maani.us. “New Version 5.” Internet: http://www.maani.us/xml_charts/, [Feb. 3, 2010].

[56] J. Glazebrook. “Open Flash Chart.” Internet: http://teethgrinder.co.uk/open-flash-chart/, [Jan. 26, 2010].

[57] phpMyAdmin devel team. “phpMyAdmin.” Internet: http://www.phpmyadmin.net/home_page/index.php, [Feb. 3, 2010].

[58] H. Ureh and C. Henry. (2009, Dec.) “DC-DC Converter for Harvesting Energy from an Exercise Bike” [Online]. Available: http://digitalcommons.calpoly.edu/eesp/20/ [Jan. 29, 2010].

APPENDIX A: Enphase Data Retrieval Script

```
<?php
/*
 * ehfem_retriever.php
 * Purpose: Connect to all emus in the text file and download their energy
 *          production statistics.
 * Author: Brendan Asche
 * Cal Poly
 * Fall 2009
 */

//Open the text file containing emu IP addresses (one per line)
$addresses = file("emu_ip_addresses.txt", FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);

if( $addresses != FALSE )
{
  foreach( $addresses as $address )
  {
    //Try connecting to this emu's home page
    echo "Attempting to connect to " . $address . "...";
    //Get the information from the home and production web interfaces of the Enphase EMU
    $handle = fopen("http://" . $address . "/home", "r");
    if( $handle == FALSE )
      echo "Couldn't connect to " . $address . "'s home page, aborting and moving to the next
entry.\n\n";
    else
    {
      //Read the web page text
      echo "done.\nReading " . $address . "...";
      $home_text = stream_get_contents($handle);
      fclose($handle);
      echo "done.\n";

      //Initialize variables
      $str_len = 0;
      $position_start = 0;
      $position_end = 0;
      $serial_number = 0;
      $fields = array(
          0 => "Lifetime generation",
          1 => "Currently generating",
          2 => "Currently",
          3 => "Today",
          4 => "Past Week",
          5 => "Past Month",
          6 => "Since Installation" );
      $output_array;
      $magnitude;

      //Parse the serial number
      $position_start = strpos($home_text, "EMU Serial Number: ", 0);
      $position_end = strpos($home_text, "<", $position_start);
      $str_len = $position_end - $position_start - strlen("EMU Serial Number: ");
```

```
$serial_number = substr($home_text,
     $position_start + strlen("EMU Serial Number: "), $str_len);
echo "id number: " . $serial_number . "\n";

//Parse each field of interest
for ($i = 0; $i <= 1; $i++ )
{
  //Parse the field value
  echo $fields[$i] . ": ";
  //Find the offset in the web page's text where the field's name
  //occurs. The HTML code looks like:
  //<TD>Lifetime generation</TD><TD>1.953 kWh</TD>

  //Find the field by its name e.g. Lifetime generation
  $position_start = strpos($home_text, $fields[$i], 0);
  //Shift to the <TD> after that (right before 1.953)
  $position_start = strpos($home_text, "<TD>", $position_start);
  //Find the first space occuring after position_start (the space
  //that happens right after the value (1.953 kWh)
  //                            Right here^
  $position_end = strpos($home_text, " ", $position_start);
  //Calculate the number of characters between the <TD> and the
  //space
  $str_len = $position_end - $position_start - strlen("<TD>");
  //Pull out the text between <TD> and the space (which contains the
  //value 1.953 in this case)
  $output_array[$i] = (float)
    substr($home_text,
         $position_start + strlen("<TD>"), $str_len);
  echo $output_array[$i];

  //Parse the value's magnitude using the same process
  $position_start = $position_end;
  $position_end = strpos($home_text, "W", $position_start);
  $str_len = $position_end - $position_start - 1;
  $magnitude = substr($home_text, $position_start + 1, $str_len);

  //kW or kWh case (want units in Watts or kilowatts)
  if( 0 == strcmp($magnitude, "K") || 0 == strcmp($magnitude, "k") )
  {
    echo " * 10^3";
    //Multiply by 1000 for wattage (using kWh for energy)
    if( $i == 1 )
      $output_array[$i] *= 1000;
  }
  //Megawatts or megawatt-hours
  else if( 0 == strcmp($magnitude, "M") )
  {
    echo " * 10^6";
    if( $i == 1 )
      $output_array[$i] *= 1000000;
    else
      $output_array[$i] *= 1000;
  }
  //Milliwatts
  else if( 0 == strcmp($magnitude, "m") )
```

```
        {
          echo " * 10^-3";
          if( $i == 1 )
            $output_array[$i] *= .001;
          else
            $output_array[$i] *= .000001;
        }
        //No magnitude (just watts or watt-hours)
        else if( 0 == strcmp($magnitude, "") )
        {
          echo " * 10^0";
          if( $i == 0 )
            $output_array[$i] *= .001;
        }
        //An unknown unit
        else
          echo " * ?";

        echo " W or Wh\n";
      }

      //Open the production web page
      $handle = fopen("http://" . $address . "/production", "r");
      if( $handle == FALSE )
        echo "Couldn't connect to " . $address . "'s production page, aborting and moving to the next
entry.\n\n";
      else
      {
        //Download the production page's text
        $production_text = stream_get_contents($handle);
        fclose($handle);

        for ($i = 2; $i <= 6; $i++ )
        {
          //Parse the fields the same way as the home page
          echo $fields[$i] . ": ";
          $position_start = strpos($production_text, $fields[$i], 0);
          $position_start = strpos($production_text, "<TD>", $position_start);
          $position_end = strpos($production_text, " ", $position_start);
          $str_len = $position_end - $position_start - strlen("<TD>");
          $output_array[$i] = (float)
            substr($production_text,
                $position_start + strlen("<TD>"), $str_len);
          echo $output_array[$i];

          //Parse the magnitude the same way
          $position_start = $position_end;
          $position_end = strpos($production_text, "W", $position_start);
          $str_len = $position_end - $position_start - 1;
          $magnitude = substr($production_text, $position_start + 1, $str_len);
          if( 0 == strcmp($magnitude, "K") || 0 == strcmp($magnitude, "k") )
          {
            echo " * 10^3";
            if( $i == 2 )
              $output_array[$i] *= 1000;
          }
```

```
  else if( 0 == strcmp($magnitude, "M") )
  {
    echo " * 10^6";
    if( $i > 2 )
      $output_array[$i] *= 1000;
    else
      $output_array[$i] *= 1000000;
  }
  else if( 0 == strcmp($magnitude, "m") )
  {
    echo " * 10^-3";
    if( $i > 2 )
      $output_array[$i] *= .000001;
    else
      $output_array[$i] *= .001;
  }
  else if( 0 == strcmp($magnitude, "") )
  {
    echo " * 10^0";
    if( $i > 2 )
      $output_array[$i] *= .001;
  }
  else
    echo " * ?";

  echo " W or Wh\n";
}

//Add the information to the database.

$host="REMOVED FOR SECURITY";
$username="REMOVED FOR SECURITY";
$password="REMOVED FOR SECURITY";
$database="ehfem";
$query;
$result;
$num_rows;

//Connect to the database
mysql_connect($host,$username,$password);
@mysql_select_db($database) or die ("Unable to select database");

//Try to find a row that has this emu's serial number
$query = "SELECT `id` FROM `energy management units` WHERE `id`=$serial_number";
$result = mysql_query($query);
$num_rows = mysql_numrows($result);
echo "Number of rows = " . $num_rows . "\n";

//Update the emu's serial number (if the emu already exists in the
//table)
if ( $num_rows == 1 )
{
  echo "id no. query result = " . mysql_result($result, 0, "id") . "\n";
  //Formulate the query
  $query = "UPDATE `ehfem`.`energy management units` SET
    `Lifetime Generated` = '$output_array[0]',
```

```
    `Currently Generating` = '$output_array[1]',
    `Currently` = '$output_array[2]',
    `Today` = '$output_array[3]',
    `Past Week` = '$output_array[4]',
    `Past Month` = '$output_array[5]',
    `Since Installation` = '$output_array[6]'
      WHERE `energy management units`.`id`=$serial_number;";

  //Execute the query
  $result = mysql_query($query);
  echo "Updated the database. Query result = " . $result . "\n\n";
}

//Otherwise create a new row for this emu and fill in its values
else
{
  //Make a query to add the new emu to the parent table
  $query = "INSERT INTO `ehfem`.`parent` (
    `id` ,
    `device type` ,
    `device alias` ,
    `physical location`
      )
      VALUES (
          '$serial_number', 'Enphase EMU', 'unknown', 'unknown'
          );";


  //Execute the query
  $result = mysql_query($query);

  echo "Added a new row into the parent table. Query result = " . $result . "\n\n";

  //Make a query to add a new row to the emus table
  $query = "INSERT INTO `ehfem`.`energy management units` (
    `id` ,
    `Lifetime generated` ,
    `Currently Generating` ,
    `Currently` ,
    `Today` ,
    `Past Week` ,
    `Past Month` ,
    `Since Installation`
      )
      VALUES (
          '$serial_number', '$output_array[0]', '$output_array[1]', '$output_array[2]',
          '$output_array[3]', '$output_array[4]', '$output_array[5]', '$output_array[6]'
          );";

  //Execute the query
  $result = mysql_query($query);

  echo "Added a new row into the emu table. Query result = " . $result . "\n\n";
}

mysql_close();
```

```
        }
      }
    }
}
else
   echo "Could not open emu_ip_addresses.txt.\n";

?>
```

APPENDIX B: Retrieval Script Test Web Page

```
<html>
<head>

<title>Power Output Test Page</title>

</head>

<?php
/*
 * chart.php
 * Purpose: Connects to the ehfem schema of Decade's MySQL database and
 *              displays the statistics of a single emu.
 * Author: Brendan Asche
 * Cal Poly
 * Summer 2009
 */

$username="REMOVED FOR SECURITY";
$password="REMOVED FOR SECURITY";
$database="ehfem";
$host="REMOVED FOR SECURITY";

mysql_connect($host,$username,$password);
@mysql_select_db($database) or die( "Unable to select database");

$query = "select * from `energy management units`";
$result = mysql_query($query);
$num = mysql_numrows($result);

mysql_close();

$i=0;
while ($i < $num) {

$serial_number = mysql_result($result,$i,"id");
$lifetime_generated = mysql_result($result,$i,"Lifetime generated");
$curr_power_output = mysql_result($result,$i,"Currently Generating");
$currently = mysql_result($result,$i,"Currently");
$todays_energy_output = mysql_result($result,$i,"Today");
$weeks_energy_output = mysql_result($result,$i,"Past Week");
$months_energy_output = mysql_result($result,$i,"Past Month");
$total_energy_output = mysql_result($result,$i,"Since Installation");

$i++;
}

?>

<body>
<meta http-equiv="refresh" content="5" >
<td class="TEXT" valign="top" width="505">
<font face="arial, helvetica, ventura" size="2"><font size="+1">Enphase Power Statistics
(TEST)</font>
<HR>
```

```
<table width="350" border="1" cellpadding="1" style="font-family: arial; font-size: 12pt">
<tr>
<td width="250"><b>Statistic Label</b></td>
<td width="100"><b>Value</b></td>
</tr>
<tr>
<td width="250">Serial number</td>
<td width="100"><?php echo $serial_number; ?></td>
</tr>
<tr>
<td width="250">LIfetime Generated</td>
<td width="100"><?php echo $lifetime_generated; ?> W</td>
</tr>
<tr>
<td width="250">Currently Generating</td>
<td width="100"><?php echo $curr_power_output; ?> kWh</td>
</tr>
<tr>
<td width="250">Currently</td>
<td width="100"><?php echo $currently; ?> kWh</td>
</tr>
<tr>
<td width="250">Today's Energy Output</td>
<td width="100"><?php echo $todays_energy_output; ?> kWh</td>
</tr>
<tr>
<td width="250">This Week's Energy Output</td>
<td width="100"><?php echo $weeks_energy_output; ?> kWh</td>
</tr>
<tr>
<td width="250">This Month's Energy Output</td>
<td width="100"><?php echo $months_energy_output; ?> kWh</td>
</tr>
<tr>
<td width="250">Total Energy Output</td>
<td width="100"><?php echo $total_energy_output; ?> kWh</td>
</tr>
</table>
<p> </p>
</td>
</body>
</html>
```

APPENDIX C: SQL Queries to Create the ehfem Schema

```
CREATE DATABASE IF NOT EXISTS ehfem;
USE ehfem;

--
-- Definition of table `emu history`
--

DROP TABLE IF EXISTS `emu history`;
CREATE TABLE `emu history` (
  `entry number` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id` varchar(45) NOT NULL,
  `daily energy` float NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`entry number`) USING BTREE,
  KEY `FK_emu history` (`id`),
  CONSTRAINT `FK_emu history` FOREIGN KEY (`id`) REFERENCES `parent` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

--
-- Definition of table `energy management units`
--

DROP TABLE IF EXISTS `energy management units`;
CREATE TABLE `energy management units` (
  `id` varchar(45) NOT NULL,
  `Lifetime Generated` float DEFAULT NULL,
  `Currently Generating` float DEFAULT NULL,
  `Currently` float DEFAULT NULL,
  `Today` float DEFAULT NULL,
  `Past Week` float DEFAULT NULL,
  `Past Month` float DEFAULT NULL,
  `Since Installation` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `FK_energy management units` FOREIGN KEY (`id`) REFERENCES `parent` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Definition of table `parent`
--

DROP TABLE IF EXISTS `parent`;
CREATE TABLE `parent` (
  `id` varchar(45) NOT NULL,
  `device type` varchar(45) NOT NULL,
  `device alias` varchar(45) NOT NULL,
  `physical location` varchar(45) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Definition of table `stand alone device sessions`
--
```

```
DROP TABLE IF EXISTS `stand alone device sessions`;
CREATE TABLE `stand alone device sessions` (
  `entry number` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id` varchar(45) NOT NULL,
  `peak output` float NOT NULL,
  `current output` float NOT NULL,
  `total output` float NOT NULL,
  `start time` datetime NOT NULL,
  `last update` datetime NOT NULL,
  PRIMARY KEY (`entry number`),
  KEY `FK_stand alone device sessions` (`id`),
  CONSTRAINT `FK_stand alone device sessions` FOREIGN KEY (`id`) REFERENCES `parent`
(`id`)
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=latin1;
```

APPENDIX D: EMU History Snapshot Script Source Code

```
<?php
/*
 * emu_history_updater.php
 * Purpose: to connect to the database and get the entries in the EMUs table.
 * Author: Brendan Asche
 * Cal Poly
 * Fall 2009
 */

$host="REMOVED FOR SECURITY";
$username="REMOVED FOR SECURITY";
$password="REMOVED FOR SECURITY";
$database="ehfem";
$query;
$emu_result;
$insert_result;
$row;
$num_rows;

//Connect to the database
mysql_connect($host,$username,$password);
@mysql_select_db($database) or die ("Unable to select database");

//Get the id and Today columns from the emus table
$query = "SELECT `id`,`Today` FROM `energy management units`;";
$emu_result = mysql_query($query);
$num_rows = mysql_numrows($emu_result);
echo "Number of rows to add = " . $num_rows . "\n";

for($i=0; $i < $num_rows; $i++)
{
  //Get one row of the query results
  $row = mysql_fetch_row($emu_result);

  //Make a new query to insert the id and today values, along with the
  //current date, into the emu history table.
  $query = "INSERT INTO `ehfem`.`emu history` (`id`,`daily energy`,`date`) VALUES
    ('$row[0]',
     '$row[1]',
     now() );";
  $insert_result = mysql_query($query);
  echo "Added an entry for EMU # " . $row[0] .
    " to emu history. Query result = " . $insert_result. "\n";
}

mysql_close();

?>
```

APPENDIX E: Doughnut Chart Source Code

```
<?php
/*
 * donut.php
 * Purpose: Generates a doughnut chart comparing stand-alone network devices
 *          to Enphase devices.
 * Author: Brendan Asche
 * Cal Poly
 * Winter 2010
 */

  include("includes/FusionCharts.php");
  include("includes/database.php");
  include("includes/fusion_charts_colors.php");
  include("includes/FusionCharts_Gen.php");

  //Create a new fusion chart object
  //Doughnut2D.swf is the file name...parameter doesn't want the .swf
  $chart = new FusionCharts("Doughnut2D","650","400");

  //Initialize two date strings
  $start_time = "2009-11-16 00:00:01";
  $end_time = "2009-11-17 00:00:01";

  //Set the desired database field to "total output"
  $output_type = "total output";
?>
<HTML>
<HEAD>
  <TITLE> EHFEM - Enphase vs Stand-alone </TITLE>
  <SCRIPT LANGUAGE="Javascript" SRC="javascripts/FusionCharts.js"></SCRIPT>
</HEAD>
<BODY>
<CENTER>
<td class="TEXT" valign="top" width="505">
<font face="arial, helvetica, ventura" size="2"><font size="+1">Total Energy Contribution
(kWh)</font>
<?php
  //Connect to the DB
  $link = connect_to_db();

  //Set the folder where the flash chart file resides
  $chart->setSWFPath("./charts/");

  //Set chart parameters (see FusionCharts documentation)
  $chart_params ="showNames=1;decimalPrecision=2";

  $chart->setChartParams($chart_params);

  //Execute a query to return the sum of all stand alone device energy
  //production
  $query = "SELECT sum(`total output`) FROM `stand alone device sessions` s;";
  $result = mysql_query($query) or die(mysql_error());
  $rows = mysql_fetch_array($result);
```

```
  $chart->addChartData($rows['sum(`total output`)'], "name=Stand-alone devices");

  //Execute a query to return the sum of all EMU energy production
  $query = "SELECT sum(`Lifetime Generated`) FROM `energy management units` e;";
  $result = mysql_query($query) or die(mysql_error());
  $rows = mysql_fetch_array($result);
  $chart->addChartData($rows['sum(`Lifetime Generated`)'], "name=Enphase devices");

  //Close the database link
  mysql_close($link);

  //Render the chart
  $chart->renderChart();
?>
</CENTER>
<BR>
<CENTER>
Jump to:
<form name="linkform">
<select name="linkselect"

onChange="window.location=document.linkform.linkselect.options[document.linkform.linkselect.sele
ctedIndex].value">
  <option selected value="donut.php">--Choose--</option>
  <option value="stand_alone_line.php">Line Chart</option>
  <option value="enphase_summary.php">Enphase Summary</option>
  <option value="test_area.php">Configurable Options Test Area</option>
</select>
</form>
</CENTER>
</BODY>
</HTML>
```

APPENDIX F: Enphase Detailed EMU Statistics Chart Source Code

```
<?php
/*
 * enphase_detailed_emu.php
 * Purpose: Generate a bar graph of a single EMUs statistics.
 * Author: Brendan Asche
 * Cal Poly
 * Winter 2010
 */

  include("includes/FusionCharts.php");
  include("includes/database.php");
  include("includes/fusion_charts_colors.php");
  include("includes/FusionCharts_Gen.php");

  //Get the serial number from the web address of this page
  $serial_number = $_GET['id'];

  //Create a new FusionCharts object using the Column3D.swf file (leave out
  //the .swf)
  $chart = new FusionCharts("Column3D","1000","450");
?>
<HTML>
<HEAD>
  <TITLE> EHFEM - Energy Management Unit Details </TITLE>
  <SCRIPT LANGUAGE="Javascript" SRC="javascripts/FusionCharts.js"></SCRIPT>
</HEAD>
<BODY>
<CENTER>
<td class="TEXT" valign="top" width="505">
<font face="arial, helvetica, ventura" size="2">
<font size="+1">Current Status of Enphase EMU # <?php echo $serial_number; ?></font>

<?php
  //Connect to the DB
  $link = connect_to_db();

  //Set the folder where the chart swf files are
  $chart->setSWFPath("./charts/");

  //Set the chart parameters (see FusionCharts documentation)
  $chart_params = "xAxisName=Time Period;yAxisName=" .
  "Power or Energy Output (kW or kWh);decimalPrecision=4;" .
  "formatNumberScale=0";

  $chart->setChartParams($chart_params);

  //Get all the information on each EMU
  $query = "SELECT * FROM `energy management units` WHERE `id`=$serial_number;";
  $result = mysql_query($query) or die(mysql_error());

  $i = 0;
  //Assign each field of the EMU's table to a bar on the graph
  if ($result) {
```

```
    while($rows = mysql_fetch_array($result))
    {
      foreach ($rows as $key => $value)
      {
        //$rows uses two keys for each value, so only use odd keys to get actual names
        //Also, skip the first 2 row indices since they concern the id number
        if( $i > 1 && $i % 2 > 0)
        {
          if( $key == "Currently Generating" ||
              $key == "Currently" )
            //Add a statistic to the chart (dividing the Watt statistics
            //by 1000 to make them kW).
            $chart->addChartData($value/1000,
              "name=" . $key . ";color=" . getFCColor());
          else
            $chart->addChartData($value,
              "name=" . $key . ";color=" . getFCColor());
        }
        $i++;
      }
    }
  }
  //Close the database connection
  mysql_close($link);

  //Draw the chart
  $chart->renderChart();
?>
</CENTER>
<BR>
<CENTER>
Jump to:
<form name="linkform">
<select name="linkselect"

onChange="window.location=document.linkform.linkselect.options[document.linkform.linkselect.sele
ctedIndex].value">
  <option selected value="enphase_detailed_emu.php">--Choose--</option>
  <option value="donut.php">Doughnut Chart</option>
  <option value="enphase_summary.php">Enphase Summary</option>
  <option value="stand_alone_line.php">Line Chart</option>
  <option value="test_area.php">Configurable Options Test Area</option>
</select>
</form>
</CENTER>
</BODY>
</HTML>
```

APPENDIX G: Enphase Summary Chart Source Code

```
<?php
/*
 * enphase_summary.php
 * Purpose: Draw a bar graph showing the total energy production of the energy
 *          management units.
 * Author: Brendan Asche
 * Cal Poly
 * Winter 2010
 */
  include("includes/database.php");
  include("includes/fusion_charts_colors.php");
  include("includes/FusionCharts.php");
  include("includes/FusionCharts_Gen.php");

  //Create a new FusionCharts object using the Column3D.swf chart file (leave
  //out the .swf)
  $chart = new FusionCharts("Column3D","650","450");
?>
<HTML>
<HEAD>
  <TITLE> EHFEM </TITLE>
  <SCRIPT LANGUAGE="Javascript" SRC="javascripts/FusionCharts.js"></SCRIPT>
</HEAD>
<BODY>
<CENTER>
<td class="TEXT" valign="top" width="505">
<font face="arial, helvetica, ventura" size="2"><font size="+1">Enphase Device Summary</font>
<?php
  //Connect to the DB
  $link = connect_to_db();

  //Set the chart .swf file's folder
  $chart->setSWFPath("./charts/");

  //Set the chart parameters (see the FusionCharts documentation).
  $chart_params = "xAxisName=Serial Number;yAxisName=" .
    "Total Energy Output (kWh);decimalPrecision=4;" .
    "formatNumberScale=0";

  $chart->setChartParams($chart_params);

  //Execute a query to return all EMU id numbers
  $query = "SELECT `id`, `Lifetime Generated` FROM `energy management units` e;";
  $result = mysql_query($query) or die(mysql_error());

  //If we got a query result
  if ($result)
  {
    //Loop through all the rows returned from the query
    while($rows = mysql_fetch_array($result))
    {
      //Add the lifetime generated value and its corresponding id no. to
      //the chart, get a new color for its bar, and encode the link into
```

```
        //the bar so users can click on the EMU's bar to see detailed info on
        //it.
        $chart->addChartData($rows['Lifetime Generated'],
          "name=" . $rows['id'] . ";color=" . getFCColor() . ";" .
          "link=" . urlencode("enphase_detailed_emu.php?id=" . $rows['id']));
      }
    }

    //Close the database connection.
    mysql_close($link);

    //Draw the chart.
    $chart->renderChart();
?>
</CENTER>
<BR>
<CENTER>
Jump to:
<form name="linkform">
<select name="linkselect"

onChange="window.location=document.linkform.linkselect.options[document.linkform.linkselect.sele
ctedIndex].value">
  <option selected value="enphase_summary.php">--Choose--</option>
  <option value="donut.php">Doughnut Chart</option>
  <option value="stand_alone_line.php">Line Chart</option>
  <option value="test_area.php">Configurable Options Test Area</option>
</select>
</form>
</CENTER>
</BODY>
</HTML>
```

APPENDIX H: Embedded System Line Chart Source Code

```
<?php
/*
 * stand_alone_line.php
 * Purpose: Draw a line graph of 24 hours worth of stand-alone network device
 *          exercise sessions.
 * Author: Brendan Asche
 * Cal Poly
 * Winter 2010
 */
  include("includes/FusionCharts.php");
  include("includes/database.php");
  include("includes/fusion_charts_colors.php");
  include("includes/FusionCharts_Gen.php");

  //Create a new FusionCharts object using the MSLine.swf file (leave out
  //the .swf)
  $chart = new FusionCharts("MSLine","1000","450");
  $start_time = "2009-11-16 00:00:01";
  $end_time = "2009-11-17 00:00:01";

  $output_type = "total output";
  $num_time_segs = 24;
?>
<HTML>
<HEAD>
  <TITLE> EHFEM - Embedded Device Measurements </TITLE>
  <SCRIPT LANGUAGE="Javascript" SRC="javascripts/FusionCharts.js"></SCRIPT>
</HEAD>
<BODY>
<CENTER>
<td class="TEXT" valign="top" width="505">
<font face="arial, helvetica, ventura" size="2">
<font size="+1">Current Status of Embedded Devices</font>
</CENTER>
<CENTER>
<?php
  //Connect to the DB
  $link = connect_to_db();

  //Set the folder that contains the .swf file
  $chart->setSWFPath("./charts/");

  //Set the chart parameters (see the FusionCharts documentation)
  $chart_params ="caption=Hourly Embedded Device Measurements Over 24 hours;" .
    "xAxisName=Hour;yAxisName=Energy Output (kWh);decimalPrecision=8;" .
    "formatNumberScale=0;hovercapbg=80FF80;hovercapborder=000000;" .
    "showgridbg=1;showvalues=0;animation=1;numdivlines=4;" .
    "numVdivlines=" . ($num_time_segs - 2) . ";showAlternateVGridColor=1;" .
    "canvasBgColor=FFFFFF;" .
    "lineThickness=3;rotateNames=1;shadowColor=DDDDDD";

  $chart->setChartParams($chart_params);
```

```
//Add hours 0 through 23 to the x-axis
for ($i = 0; $i < 24; $i++ )
{
  $chart->addCategory($i);
}

//Query for a list of stand-alone network device ids
$query = "SELECT p.`id` FROM parent p " .
  "WHERE p.`device type`='Stand-alone network device';";
$result = mysql_query($query) or die(mysql_error());

//Check if there's any results
if ($result)
{
  //If so, store the results in array format (into the rows variable)
  while($rows = mysql_fetch_array($result))
  {
    //Set the dataset's name to be the device's id
    $chart->addDataset("Device # " . $rows['id']);

    //Loop through the hours (0 to 23)
    for($i = 0; $i < 24; $i++ )
    {
      $start_time = "2009-11-16 " . $i . ":00:00";
      $end_time = "2009-11-16 " . $i . ":59:59";

      //Query for the sum of this device's energy production over each
      //hour
      $query_by_date = "SELECT SUM(`" . $output_type . "`) " .
        "FROM `stand alone device sessions` s " .
        "WHERE s.`id`='" . $rows['id'] . "' AND " .
        "s.`start time` between '" . $start_time . "' AND" .
        " '" . $end_time . "';";

      $by_date_results = mysql_query($query_by_date) or die(mysql_error());
      if( $by_date_results )
      {
        //If we get some results, store them in another array
        $rows_by_date = mysql_fetch_array($by_date_results);

        //If the query returned null, hard code a 0 for that data point
        if( $rows_by_date[0] <= 0 )
          $chart->addChartData(.0000);
        else
          //Otherwise, add a new data point containing the sum.
          $chart->addChartData($rows_by_date[0]);
      }
    }
  }
}

//Close the database connection
mysql_close($link);

//Draw the chart
$chart->renderChart();
```

```
?>
</CENTER>
<BR>
<CENTER>
Jump to:
<form name="linkform">
<select name="linkselect"

onChange="window.location=document.linkform.linkselect.options[document.linkform.linkselect.sele
ctedIndex].value">
  <option selected value="stand_alone_line.php">--Choose--</option>
  <option value="donut.php">Doughnut Chart</option>
  <option value="enphase_summary.php">Enphase Summary</option>
  <option value="test_area.php">Configurable Options Test Area</option>
</select>
</form>
</CENTER>
</BODY>
</HTML>
```

APPENDIX I: Database Connection Function Source Code

```
<?php
/*
 * database.php
 * Purpose: Connects to the ehfem schema of Decade's MySQL database.
 * Author: Brendan Asche (based on FusionCharts php example)
 * Cal Poly
 * Winter 2010
 */

//Function to connect to the DB
function connect_to_db() {
   //These four parameters hard code the login and schema for Decade
   $host = 'REMOVED FOR SECURITY';
   $username = 'REMOVED FOR SECURITY';
   $password = 'REMOVED FOR SECURITY';
   $database = 'ehfem';

   //Connect to the database
   $link = mysql_connect ($host, $username, $password);

   if (!$link) {
      //Print a message and quit if the connection attempt failed
      die('Could not connect: ' . mysql_error());
   }

   //Select the EHFEM database
   $db_selected = mysql_select_db($database);
   if (!$database) {
      //Print a message and quit if ehfem couldn't be selected
      die ('Can\'t use database : ' . mysql_error());
   }
   return $link;
}
?>
```

APPENDIX J: Configurable Options Test Web Page

```
<html>
<head>

<title>User-Configurable Options Test Page</title>

</head>

<body>
<td class="TEXT" valign="top" width="505">
<center>
<table width="400" border="1" cellpadding="1" style="font-family: arial; font-size: 12pt">
<tr>
  <th width="400"><center>
  <font face="arial, helvetica, ventura" size="2">
  <font size="+1">Configurable Options Test Area</font>
  </center></th>
</tr>
<tr>
  <td width="400"><br><center>
  <form>Enter time period:
  <input type="text" name="time_period" />
  </form>
  </center></td>
</tr>
<tr>
  <td width="400"><br>
  Select a device type<p>
  <form>
  <input type=radio name="device_type" value="emu" checked>Enphase EMU<br>
  <input type=radio name="device_type" value="stand_alone">Embedded System (stand alone
network device)<p>
  <center>
  <input type=submit value="Submit">
  </center>
  </form>
  </td>
</tr>
<tr>
  <td width="400"><br><center>
  Jump to:
  <form name="linkform">
  <select name="linkselect"

onChange="window.location=document.linkform.linkselect.options[document.linkform.linkselect.sele
ctedIndex].value">
  <option selected value="test_area.php">--Choose--</option>
  <option value="donut.php">Doughnut Chart</option>
  <option value="enphase_summary.php">Enphase Summary</option>
  <option value="stand_alone_line.php">Line Chart</option>
  </select>
  </form></center>
  </td>
</tr>
```

```
</table>
</center>
<p> </p>
</td>
</body>
</html>
```

APPENDIX K: Analysis of Senior Project Design

Brendan Asche
1-29-10

ABET Senior Project Analysis

Economic Considerations

The Energy Harvesting from Exercise Machines (EHFEM) data reporting system adds a maximum of $289.95 to each elliptical exercise machine's cost. This cost estimate includes an Enphase micro-inverter and extension cables required to reach power outlets. Each set of inverters requires an Enphase Energy Management Unit (EMU), adding another $335 to $350 per exercise machine group. Each stationary bike requires a $536 energy harvester designed by Cal Poly ME students, a $255 DC-DC converter designed by Cal Poly EE students, and a $129-$200 embedded system to measure power and energy. Cal Poly's EE department provided the EHFEM project with Decade, a Windows 2003 server, to support any web page or database needs. A 24-port switch to connect each EMU to the network, if needed, adds $65 to $100 to the total cost. If the new Recreation Center lacks wireless access points, the embedded measurement systems require one or more 802.11b/g switches at $50 each. A spool of network cables costs $100 to $200 depending on gauge and insulation. Table 1 shows each item's low and high cost estimates.

In the long term, the exercise machine-generated power could offset the system's implementation cost or even save money. Another aspect connects to marketing; with a successful system implementation, visitors may be impressed enough to contribute to the university economically by donating, sponsoring, or attracting new students. In addition, it could attract higher numbers of existing students and faculty to utilize the equipment, increasing power generation.

Item name	Estimated Cost Minimum Cost per unit	Maximum Cost per unit	Notes
Enphase micro-inverter	$165.95	$209.95	Depends on Watt rating
Enphase extension power cables	$65.00	$80.00	6ft and 12ft prices. 20ft for $90
Enphase Envoy Energy Management Unit	$335.00	$350.00	
EE team's DC-DC Inverter	$255.00	?	
ME team's Bike Energy Harvester	$536.00	?	
Embedded power measurement device	$129.00	$200.00	
Server	$0.00	$0.00	The EE department provided Decade
24 port ethernet switch	$65.00	$100.00	
802.11b/g switch	$0.00	$50.00	Any surplus EE/CPE/CSC switches?
Network cables	$100.00	$200.00	1000ft spools. Depends on gauge and insulation.

Table 1. Estimated costs.

Table 2 shows a table of possible funding sources. Although there are no guaranteed sources, the project could be pitched to these entities and secure funding if the entities are willing. Enphase contributed an EMU and two inverters to the EE team working on the elliptical.

Funding sources	Notes
Recreation Center Budget	Not a guarantee; depends on whether the product meets the Recreations Center's requirements.
TGIF: The Green Initiative Fund	Campaign for student fee supported grant-making funds for sustainability projects. http://digitalcommons.calpoly.edu/susconf/74/
Donations	Computers, routers/switches/network adapters may available from CPE/CSC/EE departments or donated.
Precor	Precor may be interested in donating.
Enphase	Enphase provided the EE team working on the elliptical with two Microinverters and an EMU for development. Enphase also offers academic discounts.
Department of Energy	They could provide grant money.
California Energy Commission	They could provide grant money.

Table 2. Funding options.

Environmental Considerations

The system offers users an environmentally friendly way to work out. The system could impact the environment in two additional ways besides its energy generation capability: heat and embedded energy. Ideally, the system will generate more electricity than it consumes.

Out of the box, the new exercise machines will dissipate the user's output power in the form of heat using a resistor, which would increase air conditioning usage. In other words, even though the new exercise machines may not need power from a wall outlet, they can indirectly consume energy by increasing air conditioning costs. If a generator and inverter replaced the resistor in the exercise machine, the user's output power would be converted to AC power with a much smaller portion dissipated as heat. This could reduce the exercise machine's heat output, resulting in savings on air-conditioning usage compared to an unmodified exercise machine.

The embedded energy contained in the inverters and data transfer hardware is another issue. Each component added to the system costs energy and resources to manufacture and transport.

Sustainability Considerations

The system could provide a positive example of sustainability if it can gather enough energy to offset the energy embedded in its components and the energy to maintain them. Since the energy it gathers would normally dissipate as heat, the system harnesses an untapped energy source. Since this system will dissipate less heat than an unmodified exercise machine, it will save on air-conditioning usage too.

The data collected and presented visually by the system could increase users' awareness of their impact on the environment and encourage them to take actions that reflect more concern for a sustainable way of life.

Manufacturability Considerations

Inverter Considerations

If Enphase's products are used, no additional hardware would need to be manufactured. Each micro-inverter would be installed in conjunction with the existing exercise machine hardware. The embedded power measurement option uses existing products also. The stationary bike EE/ME teams are responsible for their product's manufacturability.

Data Transfer Considerations

Enphase inverters would transfer data to an EMU over the power lines, and the web server will retrieve the power generation statistics from the EMU over the existing Cal Poly data network. A switch may be required to provide enough ports for the EMUs.

If the exercise machines employ embedded power measurement devices, an 802.11b/g wireless access point would receive their data packets and send them to the web server over the existing Cal Poly data network.

In both data transfer scenarios, all components would be off-the-shelf products that require no manufacturing at Cal Poly.

Expandability

Expandability, in terms of exercise machine units, depends on the network type used. Ethernet switches with more ports could be necessary, and wireless performance could degrade as more stand alone modules are added. Additional network hardware would consume power, reducing the system's net power generation.

Each additional exercise machine would require another inverter. Additional Enphase inverters would need to be purchased for elliptical machines and additional energy harvesters would need to be manufactured for stationary bikes.

Ethical Considerations

Ethical Approach

The ethical approach for the system is utilitarianism; the system should result in a net increase in happiness when all people are considered.

Ethical Issues

The information displayed on the web page does not include exercise session-specific information; the privacy of users' power outputs are respected. If implemented in the future, exercise machine user interface software that displays information from other exercise machines on its LED display should include functionality that prompts the other exercise machine's user for consent before displaying his or her data.

Health and Safety Considerations

Safety Considerations

The exercise machine's inverter could produce enough heat to be hazardous after use, and the electrical cables could be an electrocution hazard if mishandled or disconnected. Wiring routed on the floor could be a tripping hazard. The additional hardware added to each exercise machine should be installed in such a way that water spillage will not create an electrocution hazard. The Precor EFX 546i design has a number of health and safety issues that are covered in that product's documentation.

ADA Compliance

The final web page version should comply with ADA standards regarding text size when displayed on a computer monitor. The Cal Poly Web Authoring Resource Center's (WARC) requirements for official Cal Poly web pages also account for this.

WARC Web Accessibility Requirements

The final web page version must also comply with the WARC web accessibility requirements if it becomes an official Cal Poly web page, outlined here: http://www.warc.calpoly.edu/accessibility/index.html.

Social and Political Considerations

A successful system implementation could enhance Cal Poly's reputation as a cutting edge school. The funding for this system could depend on politics to some degree, although the indication is that if the system meets its marketing requirements, it could receive Recreation Center funding.

The system could also have a few social impacts on users. With first hand knowledge of their environmental impact, users may become more concerned about sustainability and take steps to help the environment. If users can view another exercise machine's power output, it may provide a means of social networking as users compete with each other to generate the most power.

The data network the system will implement introduces another political issue. The system's data network infrastructure will need to be coordinated with Cal Poly's IT department so it complies with any applicable policies.

The power generated by the system may be affected by policies or guidelines the utility company requires of products that put power back onto the grid.