SMART BUILDING ENERGY MANAGEMENT SYSTEM

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI‘I AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

April 2011

By

Chen Xia

Thesis Committee:

Xiangrong Zhou
Galen Sasaki
David Garmire

Keywords: wireless sensor network, smart grid, building energy management system
Abstract

Hawaii is dangerously dependent on imported oil to fuel its economy. To reduce this risk, the state government enacted the *Hawaii Clean Energy Initiative* which mandates a switch to 70% clean energy by 2030. About 30% of this conversion must come from energy efficiency improvement and 40% from renewable sources [1]. This thesis is dedicated to using the wireless sensor network technology to monitor each electric appliance that is connected to a building power grid. Building managers or occupants can easily access detailed power consumption data to detect energy waste and therefore adjust their energy usage and improve building energy efficiency. In this project, we contributed the design methods of system integration, energy sensor node hardware and firmware integration, data collection tool chain, database construction and web-based graphic user interface (GUI) for power profile visualization.

Keywords: wireless sensor network, smart grid, building energy management system.
# Table of Contents

Chapter 1 Introduction ................................................................................................................................. 1  
Chapter 2 Background Knowledge and Related Work .................................................................................. 3  
  2.1 Building Energy Reduction Approaches ............................................................................................. 3  
  2.2 Smart Grid .......................................................................................................................................... 4  
Chapter 3 System Architecture ..................................................................................................................... 7  
Chapter 4 Energy Sensor Hardware .............................................................................................................. 8  
  4.1 Power Supply ...................................................................................................................................... 9  
  4.2 AC Power Metering .......................................................................................................................... 10  
  4.3 Data Processing and Wireless Communication Unit ........................................................................... 13  
Chapter 5 Energy Sensor Firmware ............................................................................................................ 15  
  5.1 Embedded Operating System ............................................................................................................ 15  
  5.2 Firmware Architecture...................................................................................................................... 16  
  5.3 Task Scheduling ................................................................................................................................ 17  
  5.4 Component Design ............................................................................................................................ 19  
  5.5 NesC Coding and Implementation .................................................................................................... 22  
Chapter 6 Data Collection and Visualization ............................................................................................... 29  
  6.1 Data Collection and Gateway ........................................................................................................... 29  
  6.2 Daemon Software .............................................................................................................................. 30  
  6.3 Database ........................................................................................................................................... 32  
  6.4 Web Server ........................................................................................................................................ 34  
Chapter 7 Testing ........................................................................................................................................ 36  
  7.1 Real-time Active Power Test in Kitchen Scenario ............................................................................ 36  
  7.2 Real-time Active Power Test in Living Room Scenario .................................................................... 41  
  7.3 Daily Energy Usage Test over a Typical Kitchen ............................................................................. 46  
  7.4 Daily Energy Usage Pattern of a Typical Living Room .................................................................... 47  
  7.5 Daily Energy Usage Pattern of Holmes Hall 391 ............................................................................ 48  
  7.6 Time Domain Energy Consumption Aggregation ........................................................................... 50  
  7.7 Spatial Domain Power Consumption Aggregation .......................................................................... 51  
  7.8 Power Consumption Prediction ........................................................................................................ 51
List of Tables

Table 4.1. ADE7753 register ............................................................................................................ 13
Table 6.1. Packet prototype for energy and temperature messages ........................................... 31
Table 7.1. Electric utility and temperature correlation comparison ........................................... 54
Table 7.2. Power signature comparison ......................................................................................... 56
List of Figures

Figure 2.1. Building electric energy system ................................................................. 3
Figure 2.2. Smart Grid architecture ............................................................................. 5
Figure 3.1. Micro-grid energy management system architecture ................................. 7
Figure 4.1. Energy sensor hardware architecture...................................................... 8
Figure 4.2. BP5034D5 AC/DC power supply circuit ................................................ 9
Figure 4.3. Hall Effect sensor schematic circuits ..................................................... 11
Figure 4.4. Hall Effect sensor IC PCB layout design ............................................... 11
Figure 4.5. ADE7753 energy metering IC application circuit .................................. 12
Figure 4.6. Epic Core integration circuit ................................................................. 14
Figure 5.1. Firmware architecture ........................................................................... 17
Figure 5.2. Task scheduling model .......................................................................... 18
Figure 5.3. Firmware component abstraction ......................................................... 19
Figure 5.4. Wireless communication firmware integration ...................................... 21
Figure 5.5. Energy sampling firmware integration .................................................. 22
Figure 5.6. Component configuration ...................................................................... 23
Figure 5.7. Energy sensor firmware time execution model ....................................... 27
Figure 6.1. Daemon program .................................................................................. 31
Figure 6.2. Database for energy data storage .......................................................... 32
Figure 6.3. Power and temperature table in database .............................................. 33
Figure 6.4. Energy monitoring user interface .......................................................... 35
Figure 7.1. Power profile of a coffee machine .......................................................... 37
Figure 7.2. Power profile of a microwave oven ....................................................... 38
Figure 7.3. Power profile of a rice cooker ............................................................... 39
Figure 7.4. Power profile of a bread toaster ............................................................. 40
Figure 7.5. Power profile of a fan .......................................................................... 42
Figure 7.6. Power profile of a lamp ....................................................................... 43
Figure 7.7. Power profile of Sony Play Station 3 ..................................................... 44
Figure 7.8. Power consumption of Emerson 32” LCD TV ...................................... 45
Figure 7.9. Daily energy usage pattern of a typical kitchen .......................................................... 47
Figure 7.10. Daily energy usage pattern of a typical living room .................................................. 48
Figure 7.11. Daily energy usage pattern of Holmes Hall 391 on Dec 28 2010 ............................ 49
Figure 7.12. Power aggregation profile of Node 6 at Holmes Hall 391 from Jan 17 to Jan 21..... 50
Figure 7.13. Power aggregation profile of Holmes Hall 391 electric utilities at 2010 Dec 28..... 51
Figure 7.14. Power consumption prediction on 2010 Dec 15..................................................... 52
Figure 7.15. Temperature and fan correlation ............................................................................... 53
Figure 7.16. Temperature and desktop computer correlation....................................................... 54
Figure 7.17. Temperature and microwave oven correlation........................................................... 54
Chapter 1 Introduction

Driven by population growth, bigger houses, bigger TVs, more air conditioners and more computers, the growth in electricity power consumption has tripled in the past two decades [1]. The energy crisis in Hawaii is even more severe. Currently, Hawaii residents pay the highest cost per kilowatt-hour (cents/kWh) for electricity in the country. The Hawaii government enacted the Hawaii Clean Energy Initiative which mandates a switch to 70% clean energy by 2030. About 30% of this conversion must come from energy efficiency improvement and 40% from renewable sources [2].

To achieve this energy reduction goal, the first step is to find out which kind of energy consumption contributes the most. According to U.S Department of Energy Electricity Consumption report, 72% of the total U.S. electricity consumption occurs in residential and commercial buildings, and 30% of energy consumed in buildings is wasted [3]. If the U.S. building electric grid were just 5% more efficient, the energy savings would equate to permanently eliminating the fuel and greenhouse gas emissions from 53 million cars [4].

Improving building energy efficiency and integrating distributed renewable energy (DRE) sources are typical solutions to achieve building energy reduction. One of the most effective methods to improve building energy efficiency is to implement a real-time power consumption monitoring system. With the aid of energy monitoring system, building facility managers or building occupants can access detailed power consumption data. They can then detect energy waste, and therefore adjust their energy usage to maximize energy utilization, such as turning off the electric appliance immediately after use. According to the Google power meter project estimation, accessing real-time energy usage data could help users reduce energy expenditures by 12-15% [5].

In this thesis, we built a building real-time power consumption monitoring system. We also contributed to design methods of system integration, energy sensor node hardware and firmware integration, data collection tool chain, database construction and graphic web user interface for power profile visualization.

Chapter 2 discusses the background and related work. Chapter 3 illustrates the system level architecture. Chapter 4 looks into the details of the energy sensor node hardware. Chapter 5
illustrates the firmware development. Chapter 6 discusses the development of data collection tool chain, data storage and visualization. Chapter 7 evaluates some preliminary testing results. Chapter 8 concludes the thesis and discusses some future work.
Chapter 2 Background Knowledge and Related Work

2.1 Building Energy Reduction Approaches

Industrial and academic projects are being investigated and deployed for building energy reduction. In building energy system, the amount of energy supplied to a building (E\text{supply}) equals to the amount of energy consumed (E\text{consume}). In addition, if a building has its local green energy power plant that generates E\text{local} amount of energy, then we have an equation like this: 
E\text{supply} = E\text{consume} - E\text{local}. In order to reduce building energy consumption, we can decrease E\text{consume}, e.g., improving building energy efficiency and reducing the number of electrical loads, and increase E\text{local}, e.g., to using more renewable energy.

![Building electric energy system](image)

Figure 2.1. Building electric energy system

Figure 2.1 illustrates a typical building electric system diagram. As mentioned above, the first solution to achieve building energy reduction is to use more renewable energy. In Figure 2.1, DRE represents Distributed Renewable Energy power generation, e.g., power generation from solar panels and small-scale wind turbines. Power output from these energy sources is unlike traditional power plant, which is unstable and depends on environmental changes, e.g.,
energy generation from solar panel has higher energy output on sunny days than cloudy days. One solution to adapt environmental variance is to build a system that assumes worst-case power generation scenario for all time operation to satisfy building ordinary power consumption. However, this method could result in waste from significant system over-design. To solve this problem, we can implement a smart system which is able to make dynamic decisions on different renewable energy power source selection and adapt to environment changes.

Another solution to achieve building energy reduction is to implement the energy efficient infrastructure architecture and install various power efficient electric appliances. Similarly, fixed energy efficient implementation won’t adapt to environmental changes and human activities. For example, there is lot of energy wasted on an unoccupied room’s power supply. To solve this problem, we can adaptively turn on and off the lighting, air conditioning, and individual electric appliances in our building according to human activities from time to time to eliminate energy waste. This requires us to collect, analyze and predict both system level and individual level energy consumptions in real-time.

In a nutshell, renewable energy power generation and building energy consumption are largely affected by a lot of variations in time and space. We have to seek a way to let building energy systems adapt to environment changes and human activities in real-time. This can be done by detailed level power consumption monitoring and real-time on/off control.

2.2 Smart Grid

Currently, our nation’s electric power infrastructure - known as the grid - has supported the growth of modern civilization for more than a century; however the slow electric grid revolution is rapidly running up against many limitations and issues, in terms of energy efficiency, power transmission capability, and security. The electric grid’s growth in size, scale and complexity makes integration of renewable energy and improvement of energy efficiency more difficult to achieve. All these factors catalyze a revolution in changing our current electric grid. By combining the modern information-era’s technology, we can think of the smart grid as the internet brought to our electric system. By the most up to date definition from U.S. Department of Energy, smart grid is the way that makes the transformation of the electric industry from a
centralized, producer-controlled network to one that is less centralized, more informative, consumer-interactive and adaptive [6].

As mentioned in the Section 2.1, the main cause of electric power wastage energy is the variation from environment or human activities. To solve this problem, the electric grid has to adapt to these changes. Figure 2.2 further illustrates the system architecture of a smart grid. A smart grid adds an information layer in the form of a communication network to the electric grid so that the power user can monitor, analyze and control the entire electric grid corresponding to every change in the system.

Public facilities and communities are prime grounds for energy usage reduction programs because the users of the facilities are often not responsible for paying the monthly energy bill. Such facilities are in government buildings, military housing, college dormitories, and public housing. Real-time power usage monitoring could identify users that have wasteful behavior. The ultimate bill payers in these communities have a clear economic incentive for investment in energy-reducing technologies.

Figure 2.2. Smart grid architecture
In the upcoming decade, the move to a smart grid will change the industry’s entire business model. The estimation changes may apply to stakeholders, involving and affecting utilities, regulators, energy service providers, technology and automation vendors and consumers of electric power [7]. With “smart” features integrated into the traditional grid, power generation and consumption information can be more visible. Based on this information, an electric company could render a dynamic electric price for users which is inversely proportional to the renewable energy power generation output, thereby providing customers with more incentives to use green energy. Meanwhile, the electricity consumer can also become an electricity provider, e.g., sell unused local generated renewable energy back to some energy providing company through the smart grid. Moreover, a smart grid can spur new business models. For example, a possible model could be energy storage businesses that make profits from selling energy when the electric price is high and store energy when the price is low. Under this business model, stock market strategies could be used in an electric network.

In a nutshell, a smart grid can make an electric system more visible and controllable that provides countless benefit to electric companies and users in the future.
Chapter 3 System Architecture

In this project, we deployed our own system architecture of building energy management system that is shown in Figure 3.1. For each electric socket outlet, the power measurement is performed by an energy sensor node (shown as a blue dot) that is attached between the electric power load and the electrical socket outlet. The energy sensor node reads power consumption values of the power load appliance and then broadcasts the values through a wireless communication network. A neighboring energy sensor node relays the value to the next nearest node and then to the gateway. All wireless messages are collected in gateway node. Wireless mesh network routing enables long distance communication. When gateway receives the wireless packet message, it extracts useful information and then forwards to database. The web server is connected to the database system and hosts a power profile graphical web user interface. System users can therefore be able to view detailed power consumption profiles of all electric units in the building by smart phone or web browser. The long-term energy profiles can be used by facility managers to make energy efficient policies, such as mandating building occupants to adjust their behavior to eliminate energy waste.

![Figure 3.1. Micro-grid energy management system architecture](image-url)
Chapter 4 Energy Sensor Hardware

Our energy sensor node hardware integrates three major components. The first component is power supply which converts AC power to DC power. The second component is an energy metering IC which measures the electric power consumption. The third component is Epic Core [15] which is used for central processing radio communication. The major hardware configuration is shown in Figure 4.1.

![Figure 4.1. Energy sensor hardware architecture](image)

As shown in Figure 4.1, the power supply module is attached to an AC plug, which converts the AC power to DC power. The power metering IC obtains an input voltage signal directly from the load circuit, and also obtains an input current signal from a current-to-voltage transducer. From the input voltage and current signal, the IC calculates the power value and stores the result into its register. The power metering IC output is connected to the Epic Core IC, which performs digital data processing and wireless communication. To reduce the amount of work in RF hardware design, Epic Core is used in this project which integrated MSP430 microcontroller and radio chip CC2420 from TI.
4.1 Power Supply

There are three components that need a DC power supply - Epic Core needs 3.3V DC, and the energy metering IC and the Hall Effect sensor IC need 5V DC. Therefore, our power supply unit needs to convert 110V AC input to 5V and 3.3V DC output.

There are usually three circuit design methods for AC/DC conversion - direct half wave rectification, step-down transformer followed by a bridge rectification, and AC/DC conversion IC. There are tradeoffs between these methods. The direct half wave rectification circuit is simple and inexpensive, but the power utilization is quite low because only half of the AC waveform is used effectively. The step-down transformer followed by a bridge rectification circuit has the advantage of smooth output voltage and great power utilization efficiency, but the transformer has higher cost and larger PCB space occupancy. In this project, we adopted the AC/DC conversion IC circuit design method. It has the disadvantage of higher cost, but the advantage of less PCB space occupancy and smooth output voltage. Moreover, most of the commercial ICs have been well tested for potential safety issues in extreme cases, such as lighting high voltage electric shock protection. Our final circuit used the BP5034D5 AC/DC IC which is produced by ROHM [16]. Figure 4.2 shows the schematic drawing of BP5034D5 AC/DC conversion IC application circuit. The output of BP5034D5 AC/DC conversion IC is 5V DC, a voltage regulator MCP1700 is then used to generate 3.3V DC.

![Figure 4.2. BP5034D5 AC/DC power supply circuit](image-url)
4.2 AC Power Metering

The most typical approach to obtain energy consumption is to calculate the product of the digitalized current and voltage values and accumulate over time. To digitalize the voltage signal, an ADC (analogue to digital converter) is typically used. The voltage signal can be directly connected to an ADC; however, the current signal is not measurable by ADC and it must be converted to a voltage signal first. The easiest way to perform the current to voltage conversion is to insert a small resistor in serial to the circuit. The small amount of voltage that drops across the resistor is directly proportional to the current magnitude. The proportional factor can be then easily scaled by software calibration later on. This circuit design method is simple and inexpensive, but it is not ideal for large power consumption application, the resistor which used to measure the current will experience the over-heating problem since the resistance is also proportional to the temperature. On the other hand, for low power consumption applications, this method is a reasonable choice.

The Hall Effect sensor is another way to perform current measurement. This sensor is a transducer that varies its output voltage in response to the changes in magnetic field, and the flux density of the magnetic field is directly proportional to the magnitude of the current flowing through the magnetic field. Therefore, the Hall Effect sensor can be used to measure the current without interrupting the circuit [17]. In this thesis, we used the Hall Effect sensor IC for current to voltage conversion. Figure 4.3 shows the schematic drawing of the Hall Effect sensor application circuit which is referred to IC data sheet [18]. In Figure 4.3, the A-OUT pin and CO-COM pin are the voltage output. Figure 4.4 shows our PCB layout of the Hall Effect sensor application circuit. In Figure 4.4, the label “CSA” is the footprint of the Hall Effect sensor, the current is carried by a thick metal wire goes beneath the surface mounting IC.
Power engineers may measure real power, reactive power and etc for different reasons. In this thesis, we used ADE7753 IC to perform energy measurement, which provides the readings of active power, reactive power, and apparent power.

The ADE77531 IC features for high accuracy over large variations in environmental conditions and time. The ADE7753 IC incorporates two second-order 16-bit resolution ADCs, a digital integrator (on CH1) to perform active, reactive, and apparent energy measurements. The ADE7753 IC provides a serial interface to read and write data. The ADE7753 also provides various calibration features, i.e., channel offset calibration, phase calibration and gain calibration, to ensure measurement accuracy [20].
Figure 4.5 shows the schematic drawing of the ADE7753 energy metering IC [20] application circuit. In Figure 4.5, the analog inputs for Channel 1, V1P is connected to current transducer – the Hall Effect sensor. The maximum input voltage is 0.5 V. The analog input for Channel 2 is connected with the voltage divider which shares the similar attributes with channel 1. The ADE7753 IC outputs energy reading with a signed two’s complement 24-bit data-word at maximum speed of 27.9kSPS. The logic pin Data Input for the Serial Interface (DIN) and the pin Data Output for the Serial Interface (DOUT) are connected with the microcontroller.

![Figure 4.5. ADE7753 energy metering IC application circuit](image)

There are some important ADE7753 registers that are used to obtain the energy readings listed in Table 4.1[16]. The Active Energy Register (AENERGY) stores the value of active energy reading in a 24-bit, read-only register. The RAENERGY register is similar to AENERGY register, but the RAENERGY register is cleared after each read operation. Therefore, if we set the reading frequency to be 1 Hz, the RAENERGY register eventually stores the instantaneous active power consumption result rather than the energy value. Apparent Energy Register (VAENERGY) stores apparent energy reading. The RVAENERGY register is similar to the VAENERGY register but stores the instantaneous apparent power reading. The APOS register stores the offset value. WGAIN stores the power gain value.
Table 4.1. ADE7753 register

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>R/W</th>
<th>No. Bits</th>
<th>Default</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>AENERGY</td>
<td>R</td>
<td>24</td>
<td>0x0</td>
<td>Signed</td>
</tr>
<tr>
<td>0x03</td>
<td>RAENERGY</td>
<td>R</td>
<td>24</td>
<td>0x0</td>
<td>Signed</td>
</tr>
<tr>
<td>0x05</td>
<td>VAENERGY</td>
<td>R</td>
<td>24</td>
<td>0x0</td>
<td>Signed</td>
</tr>
<tr>
<td>0x06</td>
<td>RVAENERGY</td>
<td>R</td>
<td>24</td>
<td>0x0</td>
<td>Signed</td>
</tr>
<tr>
<td>0x11</td>
<td>APOS</td>
<td>R/W</td>
<td>16</td>
<td>0x0</td>
<td>Signed</td>
</tr>
<tr>
<td>0x12</td>
<td>WGAIN</td>
<td>R/W</td>
<td>12</td>
<td>0x0</td>
<td>Signed</td>
</tr>
</tbody>
</table>

4.3 Data Processing and Wireless Communication Unit

In this thesis, Epic Core is used to perform as the central processing unit and the wireless communication unit. The University of Berkeley’s Epic Core integrates a TI MSP430 ultra low power microcontroller with a radio chip CC2420 [22]. Figure 4.6 shows our Epic Core integration circuit which is based on Epic Core Integration Design Guide [24]. The IC integration is pretty straightforward based on datasheet. J2 is the programming port that is connected with USB serial port for firmware installation. GND1, GND2, GND3 are digital grounds and AGND is analog ground. DVDD is connected to the 3.3 V DC. RVDD and FVDD are connected to the 5V DC. SPI IO DIN and DOUT are connected with the ADE7753 energy metering IC. RFOUT, RFGND2, RFGND1 are the connection ports for antenna. Epic Core provides an U. FL port for external antenna connection, but considering the bulky outlook, we used internal antenna instead. In this project, we implemented a PCB printed antenna, called PIFA (patch antenna and inverted F antenna) [25]. Our PIFA antenna PCB layout drawing is followed by the Irene sensor mote [26] circuit design.
Figure 4.6. Epic Core integration circuit
Chapter 5 Energy Sensor Firmware

We developed a firmware to drive the operation of energy sensor node hardware. To achieve firmware multitasking, we used a small scale open-source operating system called TinyOS. We wrote an application program that integrates different open-source components running for the tasks of energy data acquisition, data processing and active message communication.

5.1 Embedded Operating System

A wireless energy sensor network is constructed with distributed sensor nodes, which collects energy information and communicates among each other through radio channels. The tasks like data acquisition, data processing and radio communication must be scheduled and executed properly. Thus, an operating system is needed. There are four main requirements which must be considered for embedded operating system selection. First of all, our energy sensor node is confined with the hardware resources, e.g., limited 10k RAM, 48k size program flash memory and 8MHz CPU. The selected operating system has to be small with low requirements of CPU and RAM, and small footprint of firmware size on flash memory. Secondly, the energy sensor node has to be able to fulfill the energy sensor’s functionality needs, e.g., periodic data sampling, radio transmission, and routing data. Many of these light tasks require real-time CPU resource allocation, so that the operating system must have effective concurrency management. Thirdly, there must be compatible hardware drivers available. Finally, the operating system should be energy efficient, because we don’t want the energy sensor itself to waste too much power.

Traditional operating system implements multithreaded architecture, which creates a thread for each task. Each thread has a memory stack which results in a lot of RAM waste. Moreover, the frequent data sampling and radio communication tasks can lead to intensive thread context switch which requires hundreds of processor execution cycles. On a small and low power microprocessor, like our TI MSP430, frequent context switching is unacceptable which may lead to memory overflow and data loss.

To avoid this problem on embedded hardware, the event based architecture allows us to have only one memory stack and single execution context, which ensures sampling values to be
processed in real time and to avoid data loss. To perform parallel tasks, the selected operating system should provide some kind of multithreaded features, e.g., task preemption. TinyOS is such an event based architecture operating system which is especially designed for wireless sensor networks [28]. To improve memory utilization and run time efficiency, TinyOS uses static memory allocation method which has no heaps or any other dynamic memory structures, all memory requirements is determined at compile time.

5.2 Firmware Architecture

The energy sensor firmware is featured with concurrency intensive applications which use the event based programming architecture to master concurrency. Figure 5.1 shows our firmware architecture. Energy sensor node hardware contains four major independent modules, micro-processor, energy sensor, radio chip and LEDs. Each of the hardware modules has at least one corresponding software component. Plus the application layer and task scheduler, the firmware is constructed.

There are three layers of the components with different colors shown in Figure 5.1. The components in different layer are connected vertically. To call a component in a lower level, a “command” is used; to call a component in an upper layer, an “event” is used.

The bottom layer contains hardware components which accept “commands” to operate actual hardware chips it also sends “event” signals to indicate hardware interruption. The middle layer contains hardware synthetic components which simulate the behavior of advanced hardware. The hardware synthetic component can be a chain of components that connected vertically by “command” and “event” handlers. For example, the “Active Message” component communicates with its underlying “Packet” component, the “Packet” component communicates with its underlying component “ByteReceive” component and so on until the connection reaches the actual hardware radio chip’s abstraction component which reads/writes IC registers. The application layer is the highest level component which is responsible for integrating and controlling all other lower level components. Our firmware has a simple task scheduler which controls the task execution order that posted by commands and events handler inside each individual component.
5.3 Task Scheduling

Some software components can run asynchronously since they encapsulate hardware module functions. For example, the radio and the energy sensing components can run in parallel without interfering with each other. In Figure 5.2, the parallel asynchronous components’ execution timelines are shown in blue arrows. The components have execution code. In TinyOS, these codes are inside Command/Event Handlers that shown in the orange rectangle boxes. When two or more Command/Event Handlers execute at the same time, some of the handlers may be blocked due to limited CPU resources. So, in our code design, we make the length of the code in Command/Event Handlers to be as short as possible. For example, Command/Event Handlers only do critical tasks, such as state transmission. Complex and non critical computations are placed in task queue which can be executed when CPU resource is free.

The tasks are posted by Command/Event Handlers. The task posting operation is a one step uninterruptable atomic action. TinyOS has the capability to buffer 7 tasks in the task queue so that task posting operation will fail if the task queue is full.
There are only two types of execution priority in our firmware. The Command/Event Handlers have higher priority than Tasks in the task queue. There is no preemption between each of the tasks. A standard FIFO policy is implemented for Task execution sequence scheduling. The tasks in the task queue are expected to be run-to-completion. Tasks execution could be preempted by hardware interrupts, or Command/Event handlers and the tasks in the task queue will not be executed until the current pending Command/Event Handlers are completed. Therefore the Command/Event Handlers can be returned immediately which makes our system to be very responsive. To avoid the race condition that generated by two Command/Event Handlers in asynchronous code, we usually place the conflicting code into the tasks queue. Most of the race conditions can be eliminated in compilation period.

Figure 5.2. Task scheduling model
5.4 Component Design

A typical component in our firmware, as shown in Figure 5.3, has five coding parts. The first coding part is Command Handler which handles function calls from upper level component; the second coding part is Event Handler which handles the interruptions that generated from lower level component; the third coding part is Interface which encapsulates the function calls of Command/Event Handlers; the fourth coding part is Data Frame which stores the global variables and the last coding part is task which has the lowest execution priority.

![Firmware component abstraction](image.png)
Figure 5.4 illustrates the component integration model of wireless communication module. The RFM is the radio IC CC2420’s hardware abstraction component which provides the “command” interface for upper level components. The RFM component controls the I/O pins of the radio transceiver and translates hardware interruptions to “event” signal, e.g., bit level transmission status. The Radio Byte component sends and receives byte level data, fires event when a byte transmission completes. The Packet component sits on top of the Radio Byte component, encapsulates the packet communication function. The Active Message component provides the final encapsulation for each packet transmission, the “send_message(data)” and “receive_message(data)” components interact with application layer and Ad Hot Routing component for data routing. Our design centralized the control information on high level components.
Figure 5.4. Wireless communication firmware integration

Figure 5.5 illustrates the components integration model for energy sampling module. The application component communicates with the Energy Sensor component by sending Meter_start(interval) command to instruct energy meter sampling, receives a Sample_done(value) event to get sampling result. The Energy Sensor component interacts with the ADE7753’s hardware abstraction component to set up energy metering IC and retrieve energy readings. The Energy Sensor component also interacts with the TimerM component to control sampling frequency. The ADE7753 and TimerM components are lowest level hardware abstraction components which directly control the hardware.
5.5 NesC Coding and Implementation

Our energy sensor firmware is written in nesC (network embedded system C) which is an extension to traditional C language. nesC application encapsulates code in different components. As shown in Figure 5.6, there are two parts defined in a component - the first part is configuration code which defines components’ connections; the second part is implementation code which defines the real functions. The component design method in nesC is similar to object oriented programming. Each instance of a component has its own state and functionality.
The configuration file declares the components’ mapping methods. The configuration file lists all the components that are connected to the current component and specifies their functional wirings. The nesC compiler compiles a nesC application when given this mapping information. In this project, meterC. nc is our configuration code for MeterApp, where MeterApp is our application level component that is responsible for mapping the components in system level. The configuration for my application program is shown as following:
configuration MeterAppC {}

implementation {
    components MainC, MeterApp, MeterC, LedsC;
    components new AMSenderC as AMSenderC;
    components new AMReceiverC as AMReceiverC;
    components ActiveMessageC as ActiveMessageC;

    MeterApp. Boot -> MainC. Boot;
    MeterApp. AMSend -> AMSenderC;
    MeterApp. Packet -> AMSenderC;
    MeterApp. AMReceive -> AMReceiverC;
    MeterApp. Leds -> LedsC;
    MeterApp. MeterControl -> MeterC. SplitControl;
    MeterApp. AMControl -> ActiveMessageC;
    MeterApp. Meter -> MeterC. Meter;
}

In the first line of the above code, the keyword “configuration” indicates the current file is a configuration file. Inside the curly braces, the code specifies function connection interfaces, which mirrored the heading part of implementation module (explained in Chapter 5.5) for function encapsulation. The MeterAppC configuration sits on the highest layer so that it doesn’t have any “provide” interface for other modules.

The lines which started with “Components” specify which set of components are wired with the current configuration file. In this case, components Main, MeterApp, MeterC, LedsC, AMSenderC, AMReceiver and ActiveMessageC are wired with our application program. The remaining of the MeterAppC configuration file consists of function interfaces and wiring information. The MainC. Boot function interface is part of Boot code which is the generic component to start up MSP430 microcontroller; MeterApp. Boot is wired with MainC. Boot. ACMeterApp. Boot implements the exactly function as MainC. Boot. Similarly, the last seven lines wiring interfaces indicates that meterApp component uses interfaces from MeterApp,
MeterC, LedsC, AMSenderC, AMReceiver and ActiveMessageC components. The firmware has many layers of configurations. Each component defines its own providing and using interfaces.

Implementation is the actual execution code part. Our firmware is heavily based on asynchronous code and concurrency operation. Thus implementing “split phase” programming technique is significant important to avoid asynchronous code’s blocking scenario. In this thesis, our code takes the advantage of such concept to achieve components parallel execution. For example, to acquire an energy reading from energy metering IC ADE7753, microcontroller must quit the current executing task and issue a command to write some values to the ADE7753’s registers to configure hardware IC. After the hardware configuration is done, microcontroller’s hardware resource is released immediately, and CPU will continue the previous preempted task. When sampling is completed, the ADE7753 IC would then issue an interrupt signal to the microcontroller telling microcontroller the sampling action is done so that microcontroller can start reading out the ADE7753’s registers and get the energy value. Rather than waiting for the ADE7753 to finish its task, microcontroller keeps busy doing some useful tasks all the time. There is no idle period involved. On the other hand, if we let the microcontroller to do nothing but wait until ADE7753 finishes its tasks, the CPU utilization would be very different and totally depend on the ADE7753’s hardware sampling speed.

Software’s split-phase idea mimics the hardware parallel strategy. Wireless energy sensor node is quite limited with hardware resources, e.g., RAM and CPU capability, and some real time operations are periodically implemented so that the split-phase interface wiring method is a better option as it enables parallel operation in software and reduces the responding and waiting period. In this thesis, our energy sensor includes many pieces of functional hardware components, to achieve components parallelism and real-time processing, especially for hardware split components, such as energy data sampling component and wireless communication component so that split-phase interface connection strategy is very necessary. Spit-phase interface connection specifies a set of commands, which are functions to be implemented by the interface's provider, and a set of events, which are functions to be implemented by the interface's user.

Our application layer program is a split-phase component so that Command/Event Handlers are appeared in pair. For example, the following piece of code calls the active message component in split-phase.
Main{
    call AMControl. start();
    event void AMControl. startDone(error_t err) {
        if (err == SUCCESS) {
            call MeterControl. start();
        }
        else {
            call AMControl. start();
        }
    }
}

The function “call AMControl. start();” invokes the active message component in parallel with the main function. Invocation is a one step operation, CPU resources is then released to assign other tasks so that it won’t waste time on waiting active message component to complete its task. When active message component completes its task (setup wireless chip registers), it issues an “event” (software interrupt) “AMControl. startdone” to the main program. The main program then responds to the execution result, if the active message component started successfully, the main program enters the next state, invokes “call MeterControl.start” to sample energy usage; or else, if the active message component start fails, the main function re-invokes the active message component again by “call AMControl.start();”.

The rest of our energy sensor firmware code is also built upon the concept of split-phase; Figure 5.7 shows the energy sensor firmware time execution model in our application program. The red boxes indicate critical tasks which have the highest execution priority. Since critical tasks have to be executed in real-time so that the length of the code must be very short, or else, asynchronous code will cause some unnecessary blocking problem. The blue boxes indicates the task queue. Functions in task queue have lower execution priority, and they are executed when CPU resources is released from critical tasks. We placed complicated computation tasks and non-critical function code in task queue whose execution time is not very important. Tasks in the task queue have no preemption between each other so that they are executed in FIFO sequential order. The white boxes are components wired with other open source components, and they have their own time execution models.
In Figure 5.7, the generic module “boot” starts upon reset, and it which boots up the system. When microprocessor is powered on, the microprocessor’s abstract software component would generate an event signal “boot.done” and received by the application program. The “event” handler in application program captures the “boot.done” signal and then enters main function. “RoutingControl.start” is the first critical task that invoked by the application program in split-phase. “RoutingControl.start” task starts the Ad hoc routing daemon software. “AMControl.start” invokes the active message component. When the invoked task is completed, the CPU hardware resource is released and assigned to do tasks in the task queue. Then the Active Message component is executed in parallel with the main function, which sets up the wireless chip hardware registers. When the Active Message component completes its task, it
issues an event “AMControl.startdone” to the main function. The main function receives the event, and then preempts the task queue execution to respond the critical task.

“Metercontrol.start” Command handler is then invoked to execute meter control component which initializes energy sensor chip. When meter control component completes its task, it invokes the energy sampling component periodically in every 10 second. The energy sampling component runs in parallel with the main function. When the sampling task completes, the application program posts a “data processing” task and a “send” task to the task queue, where the tasks are queued in sequential order. When CPU resource is released from the critical tasks, the tasks in the task queue are executed in FIFO order.
Chapter 6 Data Collection and Visualization

We also developed a tool chain for data collection, data storage and power profile visualization. We integrated open source Collection Tree routing daemon software. It runs in the background of each energy sensor firmware for active message routing. We integrated Telosb Mote as our Gateway, which is developed by Crossbow Technology, to collect wireless messages from each energy sensor node. We wrote a Daemon program to bridge data from serial port to database entries. We also constructed a database that stores the sampled value. Finally, we constructed a computer server and wrote web-server program to host a graphical user interface for real-time power profile visualization.

6.1 Data Collection and Gateway

Our wireless energy sensor network is built in tree topology. Each energy sensor hardware node in the wireless network is like a tree node in the tree. The Gateway node which used for data collection is like the root of the tree. Based on tree algorithm, each tree node has a unique path to the root. By following this path to forward data, all wireless packets that contain the power consumption value can be forwarded to Gateway. We used the open source component Collection Tree routing daemon running in the background of the firmware to forward the active messages.

Another function of the Gateway is to build a bridge between wireless network and computer server. The Gateway receives a wireless packet and then forwards to serial port which is connected to a computer server. In this thesis, Telosb mote hardware (developed by Crossbow Technology) was chosen for our Gateway. Telosb has a serial port interface, and a USB Virtual COM port. Virtual COM port (VCP) driver is supposed to be installed on the host computer server so that USB device will appear as an additional COM port.

The Gateway firmware can be found in the TinyOS open source tree. It is a basic TinyOS utility tool for sensor data collection. The firmware simply receives a wireless packet and forwards the data to serial port so that computer server can communicate with the wireless sensor networks. We can use terminal monitoring program, e.g., minicom, to directly view the raw active message data packets. To sniff packets coming from the serial port and forward data to
Unix standard input, a TinyOS tool “net. tinyos. tools. Listen” written in Java is integrated in this project, which simply streams out the binary contents of a packet coming from serial port to the Unix standard input.

### 6.2 Daemon Software

Daemon software parses the active message packets and then dumps useful data to the corresponding database tables. The Raw active message data can’t be dumped to database directly, because the data is arrived in binary format. The raw data has to be interpreted from binary format to decimal format and actuated with gain offset values. Then, data are dumped into database entries. Our Daemon program runs in the background of the computer server. In this thesis, our Daemon program is written in PERL programming language.

Our Daemon program is invoked by “$java net. tinyos. tools. Listen –comm serial@/dev/ttyUSB0:115200 | Daemon” command line. The command line feeds the standard output from serial port to Daemon input. Figure 6.1 shows the execution flow chart of our Daemon program. The Daemon program listens to the computer server serial port for data collection. Then, the Daemon program multiplexes the received temperature and power packets into two processes. For each process, the Daemon program firstly parses the raw data, e.g., trimming off the packet header, taking out the node ID, making a time stamp, and converting raw binary digits to decimal number. After that, raw data is actuated by applying the predefined offset and gain. Then, the Daemon program establishes the connection with predefined MySQL database and dump useful information into database tables. The data from different sensors are stored in separated database tables. In our code, Database entry insertion is done by standard SQL scripts.
The prototypes of the active message packets are shown in Table 6.1. There are two types of packets: Power and Temperature packets.

Table 6.1. Packet prototype for energy and temperature messages

<table>
<thead>
<tr>
<th>Packet</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>00 FF FF 00 01 06 00 08 00 00 00 00 00 05</td>
</tr>
<tr>
<td>Temperature</td>
<td>00 FF FF 00 04 04 00 06 00 04 0B 4B</td>
</tr>
</tbody>
</table>

The power packet has the size of 14; in contrast, the temperature packet has the size of 12. The Daemon program differentiates power packets and temperature packets by comparing the
length of the packet string. However, wireless packets may arrive incompletely occasionally. For example, if data loss happened during transmission, the size of the power packet may be chopped down to 12 which may cause the ambiguity of packet type identification. Our solution is to set the 7th byte of the packet array to be a packet identification byte. The 7th byte is “0x08” in the power packet but “0x06” in the temperature packet.

6.3 Database

We built a database to store the history data. In this thesis, we selected MySQL as our software storage engine because it is open source software. As shown in Figure 6.2, we constructed the “greenHall” database to store energy data and temperature data. There are two tables built in the database, one is “power” table, which stores the power consumption data from each energy sensors. Another is a temperature table, which stores the temperature data from each temperature sensor.

```
mysql> show databases;
+---------------------+
| Database            |
+---------------------+
| information_schema  |
| employees           |
| greenHall           |
| temperature         |
+---------------------+
4 rows in set (0.00 sec)
```

```
mysql> show tables;
+---------------------+
| Tables in greenHall |
+---------------------+
| power               |
| temperature         |
+---------------------+
2 rows in set (0.00 sec)
```

Figure 6.2. Database for energy data storage
For each “power” table and “temperature” table, there are three columns (fields), nodeID, timestamp, and data are constructed. Figure 6.2 shows the description of our database table fields.

- “nodeID” field is an unique identification number for each of the wireless sensor nodes, this number could be paired with the information of sensor node’s physical location, for e.g., sensor node 1 refers to office 101’s computer server power consumption, so that the nodeID can be used to indicate where the data come from during power waste analysis. The valid data format of this field must be an integer number, from 0 to 200.
- “timestamp” field records a wireless packet message’s arriving time. The valid data format of this field is datetime, YYYY-MM-DD hh:mm:ss, e.g., 2010-04-08 13:06:32
- “power” field stores power consumption values of a energy sensor node. The valid data format of this field must be an integer number. The default unit of this field is Watt.
- “temperature” field stores temperature values of a temperature sensor node. The valid data format of this field must be an integer number. The default unit of this field is Fahrenheit.

![Screen capture of database fields]

Figure 6.3. Power and temperature table in database
6.4 Web Server

Web-server module provides the visual access to current and historical power and temperature profiles. These long-term profiles can be used by building managers or occupants to detect energy waste and thus make smart policies to save energy.

In this thesis, we used Apache Server to host a web-GUI interface for power profile visualization. Apache is invoked when the computer server boots up, running in the background as a daemon program to provide http service.

Our server program is written in combination of HTML code and PHP code. PHP code is embedded in HTML code context. HTML code defines graphic user interface and PHP code performs the function of data processing, e.g., establishing database connection, searching and retrieving data from MySQL tables, etc.

The execution of our server program takes three steps. Firstly, the server program prompts user to input request information, e.g., which sensor node to monitor. Secondly, according to user selection, the server program searches and retrieves necessary data from database tables. Finally, the server program invokes Google Chart API to generate a chart that shows sensor monitoring results and displays the picture on webpage. The Google Chart API invocation is implemented by URL string request. We used line chart to show energy usage and temperature changes.

Figure 6.4 shows the GUI user interface of our web server. User can choose which node to monitor, at Node ID selection bar. Node ID can be used to associate with its physical monitoring location. Below the Node ID, user can type in a monitoring date. Then, the user can hit the button “Submit Query”, and a chart which carries the temperature or power information will be displayed. The line chart compares power consumption and temperature variance over time. The left vertical axis represents power consumption value in Watts; the right vertical axis represents temperature value in Fahrenheit. The horizontal axis represents the time line of a day.

The subsequent chapter has examples of charts which displays power consumption over time. No example is given to show the chart for temperature over time. However the system is capable to monitor temperature.
Figure 6.4. Energy monitoring user interface
Chapter 7 Testing

Chapter 7 explores some preliminary testing results of our wireless energy sensor network. We collected and analyzed some energy consumption data in different scenarios.

7.1 Real-time Active Power Test in Kitchen Scenario

The first test explores energy consumption power profiles of kitchen electric appliances. The time scope of this test is set to 240 seconds. Figure 7.1 to Figure 7.4 reviewed power profiles of the four different types of electric appliances in the author’s kitchen, which is used to represent a typical kitchen. The real-time active power consumption in watts is shown in the terminal, and the GUI chart is shown on the web page. This test proved the basic functionality of our energy sensors.
Figure 7.1. Power profile of a coffee machine
Figure 7.2. Power profile of a microwave oven
Figure 7.3. Power profile of a rice cooker
Wireless Sensor Networks

University of Hawaii at Manoa (Electrical Engineer)

Please specify the node and period for monitoring

Node ID: 1

Submit Query

Node: 1

Figure 7.4. Power profile of a bread toaster
Figure 7.1 shows the power consumption profile of a coffee machine which consumes around 1015 watts during the operation. Figure 7.2 shows the power consumption profile of a micro-wave oven, the power consumption alternates between two values, high power consumption is around 1500 watts and low power consumption is around 58 watts. Figure 7.3 shows the power consumption profile of a rice cooker; in “off” mode the rice cooker consumes 0 watt; in “cook” mode, the rice cooker consumes around 600 watts; and in “keep warm” mode, the rice cooker consumes around 79 watts. Figure 7.4 shows the power consumption profile of a bread toaster which consumes around 1306 watts during its operation.

7.2 Real-time Active Power Test in Living Room Scenario

Figure 7.5 to Figure 7.8 display the power consumption profile of four different living room electric appliances. The testing time scope is set to 240 second.
Figure 7.5. Power profile of a fan
Wireless Sensor Networks

University of Hawaii at Manoa (Electrical Engineer)

Please specify the node and period for monitoring

Node ID: 1

Submit Query

Node: 1

Sensor Monitor Result

Figure 7.6. Power profile of a lamp
Wireless Sensor Networks
University of Hawaii at Manoa (Electrical Engineer)

Please specify the node and period for monitoring

Node ID: 1 2

Submit Query

Node: 1

Sensor Monitor Result

![Graph showing the power profile of Sony Play Station 3](image)

Figure 7.7. Power profile of Sony Play Station 3
Figure 7.8 Power consumption of Emerson 32” LCD TV
Figure 7.5 shows the power consumption profile of a fan which consumes around 96 watts during operation. Figure 7.6 shows the power consumption profile of a typical lamp which uses a CFL light bulb and only consumes 33 watts. By providing the same amount of illumination as 100 watts ordinary light bulb, the CFL light bulb saves 67% electric energy. Figure 7.7 shows the power consumption profile of a Sony Play Station 3 game console, which consumes around 110 watts during the gaming time and consumes 0 watt in stand-by mode. Figure 7.8 shows the power consumption profile of an Emerson 32” LCD TV, which consumes around 0.47 watt in stand-by mode and 127 watts in operation mode. Even though the TV stand-by mode consumes a little power, users of such TV could save some energy if they can unplug their TV every night.

7.3 Daily Energy Usage Test over a Typical Kitchen

Figure 7.9 shows the daily power profile of a typical kitchen. Four different kitchen electric appliances were tested for 24 hours in December 26, 2010. Figure 7.11 shows that the rice cooker was operated at around 6pm, and the hourly average power consumption was 400 watts. The toaster machine was operated only once at 7am, the hourly average power consumption was 100 watts. The Coffee machine was operated twice, around 7am to 8am and 5pm, and the hourly average power consumption was 100 watts. The Microwave oven was operated twice; the hourly average power consumption was quite different because of the different operation time period. From these daily power profiles, we can find out the kitchen electric appliances were not operated very long for each use, but the power consumptions for each operation was very high. Therefore, forget to turn off the kitchen electric appliances will cause a lot of energy waste.
7.4 Daily Energy Usage Pattern of a Typical Living Room

Figure 7.10 shows the daily power consumption profile in a typical living room. Four groups of testing data were collected from living room electric appliances - a typical standing fan, a 32” LCD TV, a SONY Play Station 3 game console and a lamp with CFL energy saving light bulb. As shown in Figure 7.12, the living room electric appliances were mostly operated during the evening, and the hourly average power consumption was around 100 watts.
7.5 Daily Energy Usage Pattern of Holmes Hall 391

Figure 7.11 shows the daily power consumption profile at Holmes Hall 391 on Dec 28 2010. Four student computers, one printer and one laptop were tested. The Computer 1 consumed 10 watts during the night, and consumed around 150 watts during the daytime. The Computer 2 consumed 10 watts all the time. The Computer 1 and the Computer 2 had never been completely turned off while not use, which results in 10 watts power waste. The Computer 3 consumed 140 watts constantly all the time. The Computer 4 had a similar power usage pattern with the Computer 1, but it consumed 70 watts more during operation. This was due to the Computer 1 was connected with a LCD monitor, but the Computer 4 was connected with an older CRT monitor. The laptop consumed less than 60 watts during the daytime and 0 watt during the nighttime. The printer was operated twice during the daytime, and consumed 8 watts during the rest of the time.
Figure 7.11. Daily energy usage pattern of Holmes Hall 391 on Dec 28 2010
7.6 Time Domain Energy Consumption Aggregation

We can also apply different levels of power consumption data aggregation to assist building energy waste analysis. Figure 7.12 shows an example of the time domain power consumption aggregation.

In this test, we selected an energy sensor which monitors the power consumption of a student desktop computer, the hourly average power consumption values were dumped to the computer server’s database. By adding hourly average power consumption data, we can get a daily energy consumption profile. Figure 7.12 shows the time domain power aggregation profile of the energy sensor Node 4 from 2011 Jan 17 to 2011 Jan 21. In this test, Monday, Tuesday, Wednesday consumed more energy than Thursday and Friday. On Saturday little energy was consumed, and on Sunday no energy was consumed.

Figure 7.12. Power aggregation profile of Node 6 at Holmes Hall 391 from Jan 17 to Jan 21
7.7 Spatial Domain Power Consumption Aggregation

The spatial aggregated power profile of any group of electric utilities can be obtained from adding up individual power consumption data. Figure 7.13 shows the power aggregation profile of six individual electric utilities at Holmes Hall room 391. The total maximum power consumption is 615 watts and the total minimum power consumption is 167 watts. Spatial power consumption aggregation can be applied to different areas in the building to analyze waste in system level energy use. For example, we can compare the power consumption profiles of different computer labs, floors and departments to identify which part of the building wastes more energy.

![Power aggregation profile of Holmes Hall 391 electric utilities at 2010 Dec 28](image)

Figure 7.13. Power aggregation profile of Holmes Hall 391 electric utilities at 2010 Dec 28

7.8 Power Consumption Prediction

The history power consumption data can be also used to predict future power consumption. In this test, we used the average power consumption data of every Wednesday in the past four weeks to predict the next coming Wednesday power consumption’s status. The energy sensor
Node 4 is selected in this test. Figure 7.14 compares the actual and predicted power consumption results. In Figure 7.14, the blue curve represents the power consumption prediction for 2010 December 15. The red curve represents the actual measured power consumption in 2010 December 15. There is a perfect match between two curves from 0 o’clock to 16 o’clock and from 20 o’clock to 24 o’clock, but the power consumption prediction is less accurate from 16 o’clock to 20 o’clock, this may due to the graduate research students’ office leaving time was in random.

![Figure 7.14. Power consumption prediction on 2010 Dec 15](image)

7.9 Power Consumption Correlation with Temperature

Analyzing the relationship between the power consumption and the environment changes is the first step to achieve smart building adaption. The most common correlation analysis is to find out the Pearson correlation coefficient between any two variable’s sample values. The Pearson correlation coefficient computation is very simple which is only sensitive to a linear relationship [29] [30] [31] analysis. Pearson correlation coefficient calculation formula is shown in Formula 7.1 [29]. The Pearson correlation coefficient’s calculation result is at the range between -1 to 1. If the result is close to -1 or 1 which indicates the two variables are closely
related. On the other hand, if the result is close to 0 which indicates there is less relationship between the two variables.

Our correlation test is applied to analyze the relationship between room temperature and living room electric utility power consumption. In this test, we trimmed off the nighttime sample values, and then computed Pearson Correlation coefficient [32] along with some by-product values that generated by the calculator tool, e.g., Scatter Plot, Pearson Product Moment Correlation, Covariance, Determination, and the Correlation T-Test values which is shown in Figure 7.15, Figure 7.16 and Figure 7.17.

\[
r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}
\]

Formula 7.14. Pearson correlation coefficient [29]

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Variable X</th>
<th>Variable Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>25</td>
<td>75.72222222</td>
</tr>
<tr>
<td>Biased Variance</td>
<td>1506.66666666667</td>
<td>7.20061728395062</td>
</tr>
<tr>
<td>Biased Standard Deviation</td>
<td>38.815804341359</td>
<td>2.6839659460740</td>
</tr>
<tr>
<td>Covariance</td>
<td>97.5294117647059</td>
<td></td>
</tr>
<tr>
<td>Correlation</td>
<td>0.884338569590808</td>
<td></td>
</tr>
<tr>
<td>Determination</td>
<td>0.782054705665917</td>
<td></td>
</tr>
<tr>
<td>T-Test</td>
<td>7.5771312448482</td>
<td></td>
</tr>
<tr>
<td>p-value (2 sided)</td>
<td>1.1365888333331e-06</td>
<td></td>
</tr>
<tr>
<td>p-value (1 sided)</td>
<td>5.5682944166690566-07</td>
<td></td>
</tr>
<tr>
<td>Degrees of Freedom</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Number of Observations</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.15. Temperature and fan correlation
Table 7.1. Electric utility and temperature correlation comparison

<table>
<thead>
<tr>
<th>Utility Environment Variable</th>
<th>Fan</th>
<th>Desktop Computer</th>
<th>Microwave Oven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>0.88</td>
<td>-0.30</td>
<td>-0.28</td>
</tr>
</tbody>
</table>

Table 7.1 concludes the relationship between the temperature and the three different electric utilities’ power consumption. The Pearson correlation coefficient between the fan and the temperature is 0.88 which is by far larger than the Pearson correlation coefficient value between
the temperature and desktop computer which is -0.30. The fan power consumption has stronger relationship with temperature than the desktop computer and microwave oven in the living room.

7.10 Power Consumption Signature

Different electric utility may have different power usage pattern, which can be referred as *power signature*. The power signatures can be used to identify which electric utility is being connected to electric grid based on its power consumption patterns. For example, microwave oven power consumption pattern (frequently alternates between two power consumption levels) is largely different from toaster machine power consumption pattern (constant power consumption). We invented Formula 7.2 to calculate a coefficient which can be used to help us analyze the relationship between different power signatures.

For any two selected electric utility, their power consumption data was collected in 240 second. Define $P_{1\text{peak}}$ as the maximum power consumption of an electric utility in the testing time scope, define $P_{1\text{min}}$ as the minimum power consumption and define $P_{1\text{average}}$ as the average power consumption. The similar definition is applied to $P_2$ for the second electric utility. In addition, we require that keep $P_1$ to be less than or equal to $P_2$ so that the final calculated Similarity value should be at the range between 0 and 1. A Similarity value close to 1 indicates the two power signatures are perfectly matched, while a value close to 0 indicates power signatures are not well matched.

Power signatures database can be created from collecting different electric utilities’ power consumption profiles. We can use this database to help us identify which electric utility is currently being connected to the electric grid. Energy inefficient electric utilities can thus be detected by building energy manager and be forced to replace.

$$\text{Similarity} = \frac{P_{1\text{peak}}}{P_{2\text{peak}}} \times \frac{P_{1\text{min}}}{P_{2\text{min}}} \times \frac{P_{1\text{average}}}{P_{2\text{average}}}$$

Formula 7.2. Power signature Similarity
Table 7.2. Power signature comparison

<table>
<thead>
<tr>
<th>@time2</th>
<th>Microwave Oven</th>
<th>Lamp</th>
<th>Rice cooker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electric Utility 1</td>
<td>0.86</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Electric Utility 2</td>
<td>0.00</td>
<td>0.96</td>
<td>0.00</td>
</tr>
<tr>
<td>Electric Utility 3</td>
<td>0.02</td>
<td>0.00</td>
<td>0.98</td>
</tr>
</tbody>
</table>

In this test, three unidentified power signatures were compared with the power signatures that we had already known. Then, the Similarity coefficient was calculated based on Formula 7.2. Table 7.2 shows the comparison results for different Similarity values. For example, the Similarity value between the power signature of electric utility 1 and a microwave is 0.86, between a lamp is 0.00, between the rice cooker is 0.00. Thus, electric utility 1 can be identified as a microwave oven. Similarly, electric utility 2 can be identified as a lamp and electric utility 3 can be identified as a rice cooker.
Chapter 8 Conclusion

In this thesis, we represented a method using wireless sensor network technology to monitor building energy power consumption in detailed levels. We tested the functionality of our system and illustrated some power profile results under different circumstances. By visualizing and understanding energy usage profiles, users can get more detailed information about how their electric bill was generated, where and when the energy is wasted.

In our future study, we will be working on reducing system complexity and cost. For example, design a more intuitive graphic user interface for easy energy profile accessing and replace the local computer server with cloud computing to cut system construction cost.
Bibliography