OULU UNIVERSITY OF
APPLIED SCIENCES

Shi Xin

WINDMILL MONITORING USING PHP

WINDMILL MONITORING USING PHP

Shi Xin
Bachelor’s thesis
Spring 2013
Information Technology
Oulu University of Applied Sciences

PREFACE

The following Bachelor's thesis was done at Oulu University of Applied Sciences School of Engineering's Raahe Campus.

The topic came from Vesa Pehkonen, the owner of this windmill farm. The research was made about the saving and displaying the windmill status for the windmill monitoring software. I would like to express my deepest gratitude to my supervisor, Mr Lauri Pirttiaho, for his support and guidance throughout the research. He was extremely patient and gave me the necessary guidance.

I would like to thank Mrs Kaija Posio, for the language checking of my thesis. She has always been very kind and helpful. Another person that should not go unmentioned is Mr Santtu Pennanen as another thesis worker of this research and the person who was supporting me throughout the research process.

Special thanks to Vesa Pehkonen, for giving this challenging topic to work with. Last but not least, I would like to express my gratitude towards my family, for their forever support.

Raahe, May 2013
Shi Xin

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Shi Xin
Opinnäytetyön nimi:Windmill monitoring using PHP
Työn ohjaaja: Lauri Pirttiaho
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 57

Tämä opinnäytetyö liittyy pieneen tuulivoimapuistoon, joka tuottaa sähköenergiaa yksityiskäyttöön kuten tavallista omakotitaloa varten. Omistaja halusi tietää, kuinka paljon energiaa hänen tuulimyllynsä tuottavat ja kuinka paljon ylimääräistä hän voisi myydä. Opinnäytetyön tavoitteena oli suunnitella tuulimyllyjen seurantaohjelmisto niiden omistajalle. Tuulimyllyjä seurataan PHP sivuston avulla, joka näyttää laitteiden nykyisen ja historiallisen tilan.

Lähtötiedot luetaan Aurora inverttereiltä ja tallennetaan MySQL-tietokantaan, josta ne näytetään PHP verkkosivuilla. Hakemalla tietokannasta sivusto voi näyttää digitaalisessa muodossa sähkön tuotannon sekä tekstinä että kaavioina.

Tämän projektin tuloksena tämän tuulivoimapuiston omistajalla on seurantaohjelmisto, joka näyttää tuulimyllyjen tilan PHP sivustolla, joka voi näyttää sekä nykyiset ja historialliset tiedot tekstinä ja kaavioina. Laitteiston teknisten puutteiden vuoksi osaa luettavista tiedoista, kuten tuulen nopeutta ja talon energiankulutusta, ei voida vielä näyttää. Näiltä osin ohjelmisto valmistuu myöhemmin tai sen toteuttaa joku toinen.

Asiasanat:
Linux, Apache2, PHP5, MySQL5, HTML5, Tuulimylly Näyttö

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author: Shi Xin
Title of Bachelor's thesis: Windmill monitoring using PHP
Supervisor: Lauri Pirttiaho
Term and year of completion: Spring 2013 Number of pages: 57

This Bachelor's thesis was commissioned by a small windmill farm that generates electrical energy for personal use such as for a typical family house. The owner wanted to know how much energy is produced by his windmills and how much extra he could sell out. The aim of this thesis was to design a windmill monitoring software for the owner of the house. Windmills are monitored by a PHP website, which displays the current and historical statuses of the equipment.

Input information is read through an Aurora inverter and saved into a MySQL database, which is meant for displaying the result on a PHP website. By extracting contents from this database, the website can display the digital form of generation status in both text and chart format.

As a result of this project, the owner of this farm will have a windmill monitoring software, displays information about the windmills status on a PHP website, which can show the current and historical data in text and chart format. Due to the technical obstacles of the hardware, some of the equipment readings such as wind speed and house energy consumption are not able yet. Thus the software implementation of this part will be completed later or it will be taken over by someone else.

Keywords:
Linux, Apache2, PHP5, MySQL5, HTML5, Windmill monitor

TABLE OF CONTENTS

1 INTRODUCTION

This project was made for a small windmill farm located in Pyhäjoki. It aimed to provide a software application to monitor the windmill status. This windmill farm has three windmills. It produces electricity for the owner's own use and the extra electricity can be sold out. The owner was interested in knowing the productivity of the plant and the working condition of the equipment.

Another thesis worker Santtu Pennanen was dealing with the hardware part, which collects the measurement from the windmill equipment, and converts it to a digital signal. My responsibility was to save that data to a database and display it on a website using Apache2, HTML, PHP, JPGraph and MySQL.

The completed application allows the user to monitor the status of the windmills through a PHP programmed website, which can show the current and historical data in a text and chart format.

2 HARDWARE AND SOFTWARE ENVIRONMENT

This project is connected to the development of the windmill farm equipment. There are 3 windmills on this farm. The electricity is generated by the wind and used for the consumption of a typical house. The monitoring software provides a functionality to observe those windmills.

This software runs on a Raspberry Pi Linux server. The inverters of the windmills are connected to the monitoring server through a USB/RS-485 converter. It reads the status data from the inverters and saves it into a MySQL database. This database contains all data the owner is interested in, such as the grid voltage, the grid current and the energy produced. The website is the core part in this project. It acts as a displaying platform. It has only a few web pages, and can be used by a non-professional user very easily. Figure 1 shows the windmill hardware diagram.

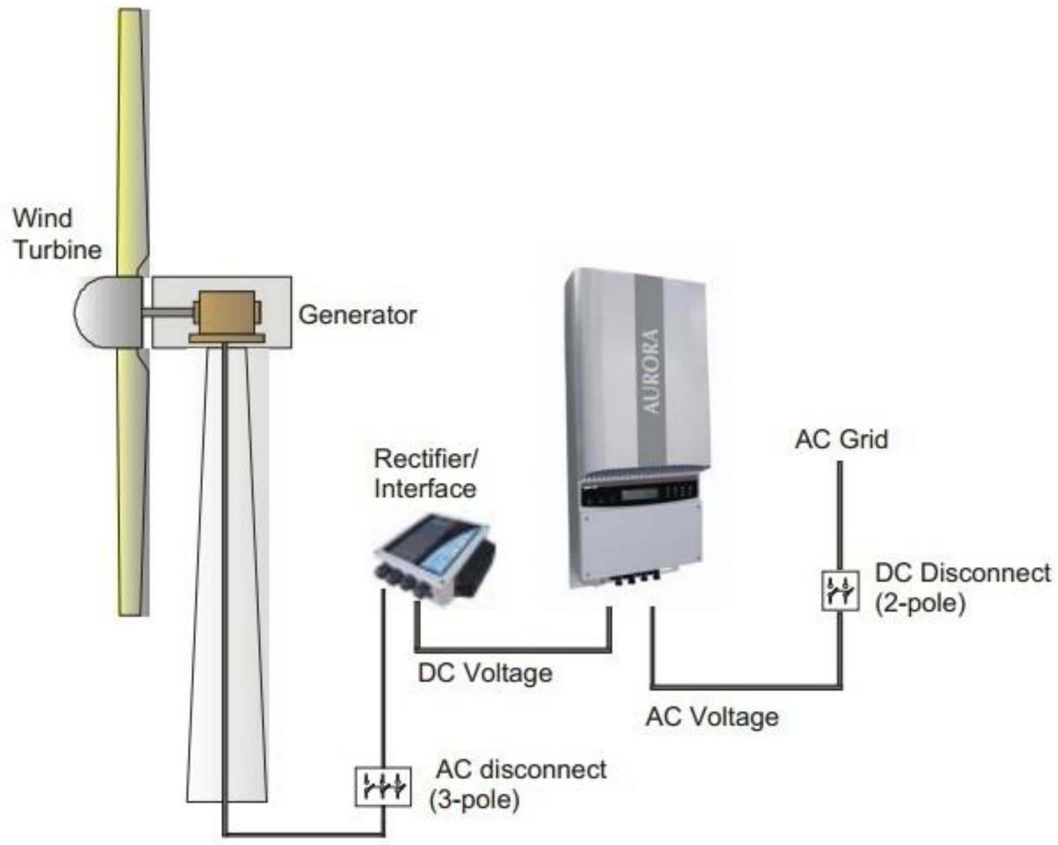


FIGURE 1. Wind Energy hardware Diagram.(PVI-5000-OUTD-W, PVI-6000-OUTD-W, Power-one Inc, data of retrieval 2.5.2013)

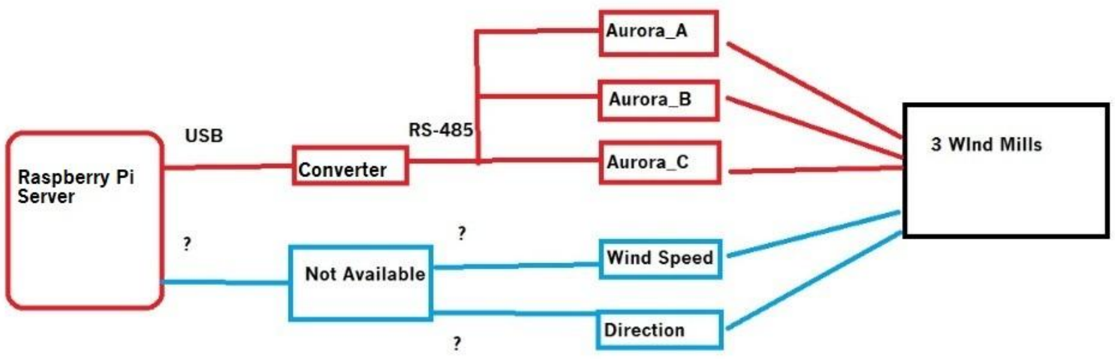


FIGURE 2. Hardware Equipment Diagram.

Figure 2 describes the structure of the hardware equipment.

2.1 Hardware Specification

The rated power of each generator of the windmill farm is 5kW. The generators are mounted to 17 meters high towers. The estimated annual production capacity is 5000-7500 kWh. The manufacturer of the generators and blades is Ningjin Huaya Industry Co., Ltd (Shandong, China). The towers and the controls are the owner's own design and production.

The conventional rudder control used in small windmills has been replaced with a turning motor and a programmable logic device like in a large scale (>500 kW) windmill. With this configuration, also some advanced control functions become available.

2.2 Linux

2.2.1 What is Linux

”Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution”.
(Linux, Wikipedia, date of retrieval 30.4.2013)

2.2.2 Why Linux

Linux is stable, secure and it is less demanding on the hardware than other alternatives like Windows. In addition, it is free and open source code. The monitoring website server like this demands the platform to be especially stable and safe. It does not need a gorgeous graphical user interface. Therefore the Linux platform becomes the most suitable choice for this project.

2.3 Apache2

The Apache HTTP Server is an open source web server software. It is one of the most popular Web server software because of its cross-platform and security. It can run on almost all widely used computer platforms.

2.4 HTML and PHP

"HyperText Markup Language (HTML) is the main markup language for displaying other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like<html>, within the web page content. (Wikipedia, 2013. HTML, date of retrieval 29.4.2013)

"PHP is an open source server-side scripting language designed for Web development to produce dynamic Web pages. It is one of the first developed server-side scripting languages to be embedded into an HTML source document rather than calling an external file to process data". (Wikipedia, 2013. PHP, date of retrieval 30.4.2013)

In short, HTML is used to describe what to be displayed and how they are displayed in a browser window. It is not a programming language and works on a client computer. HTML is able to display text, image, video, audio, table and hyperlink, but it is only in the form of static. The contents are not able to be updated dynamically. PHP is a server side scripting language, it is embedded into HTML. Since PHP is executed on the server side, it can connect to a database and output the dynamic data to HTML. In other words, HTML is the appearance and structure of a website, and PHP is the processor of this website.

2.6 What is MySQL

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

(Refsnes Data, 2013. MySQL Introduction, date of retrieval 2.5.2013)

MySQL is currently the most popular database management system with open source. It was developed, published and supported by Oracle. MySQL is often combined with PHP. PHP scripts can access the MySQL database and use Structured Query Language to store and retrieve data.

2.7 Aurora Inverter Software

The measurement data provided by the Aurora inverter is made available through the RS-485 bus by a custom protocol, that is not publicly documented. However, there is available a piece of the Linux software written by Curtis J. Blank (Curtis J. Blank, 2013. Curtronics.com, date of retrieval 2.5.2013), that can be used to read that data, such as the grid voltage, the grid current, etc.

3 DEFINITION

This section will describe the equipment of the software. This software has two parts, data collecting and displaying. The windmill status is read and saved twice a minute. This data will be used to display a website by both a text and a chart format.

3.1 The collected data of windmills

1. The software should be able to read the status data of windmills from the inverters, such as a grid voltage, a grid current, a grid power, an inverter temperature, and energy produced.

2. It should be able to store the status date to a MySQL database and save the readings.

3. It should also read wind speed, wind direction and windmill rotation from some other communication devices. This part of the equipment was left out because those devices were not available yet.

3.2 Create a database for historical viewing

The owner wants to know how much energy his windmill farm produces. He wants the working status displays of the windmills to be updated every 30 seconds. One database table is enough for this purpose. It calculates the amount of energy generated by the last 30 seconds and the time when it is inserted. The temperature is also stored to fit the other thesis worker's need.

3.3 Website displays the status

The website is accessed through a wireless connection and thus an authentication might be needed. A login page is provided although it was not requested by the owner.

The status of the windmills is divided into a current status and a history status. Displays Grid voltage, grid current and grid power from the last 30 seconds are shown in a table format. The history of the energy production is displayed as a chart.

3.4 Performance requirements

The software on the user side should be simple and easy to use. The user can manage the monitoring process with a single web page that shows a couple of charts and tables. For history readings, there is a lookup page. That can be used to fetch the data from the database according to the specific dates given by the user.

4 IMPLEMENTATION

The software architecture is shown in Figure 3 below.

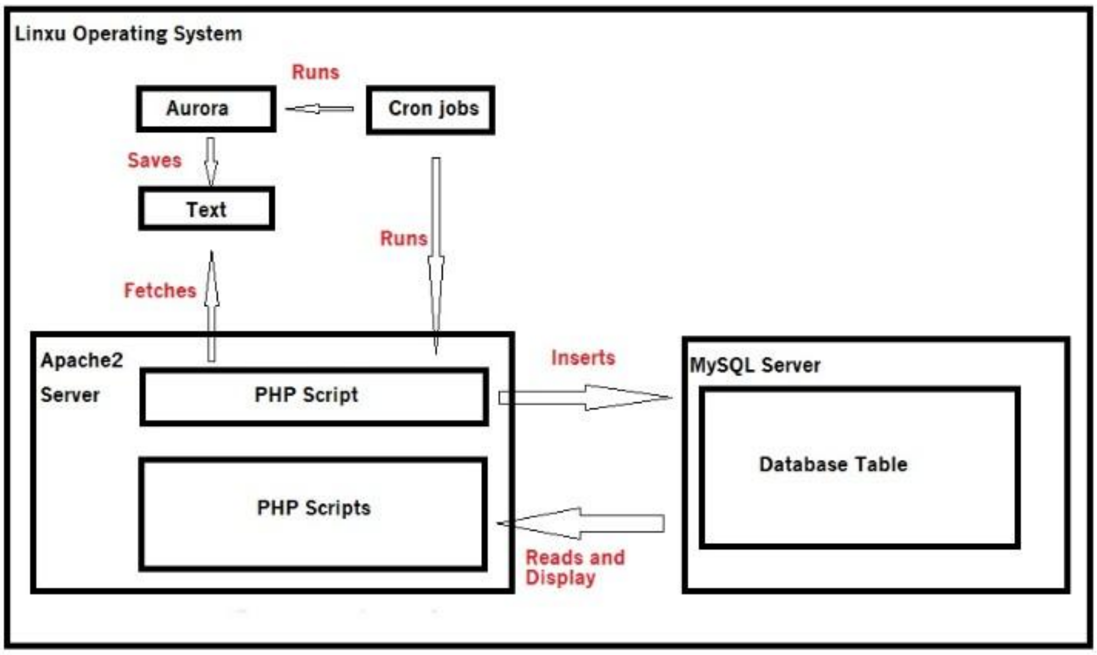


FIGURE 3. Software System Diagram.

4.1 Data Collecting

“Cron is the time-based job scheduler in Unix-like computer operating systems. Cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates.” (Wikipedia, 2013. Cron, date of retrieval 30.4.2013)

The description about how to use the Cron jobs can be found in ‘scrounge.org’.
(Wayne Larmon, 2013. Using cron to run programs on a schedule, date of retrieval 30.4.2013)

For instance, if I want to say “Good morning” at 8:00 am every day, the command will be like this

```
0 8 * * * echo “Good morning.” >> /tmp/test. txt
```

The result is not displayed on the screen. Instead, we will find the result in the ‘test.txt’ under ‘tmp’ directory, and it is also saved into the mailbox as an email under the root directory. In the following, I will explain how we can disable the email.

This software uses multiple Cron jobs: The first one is to read the data from windmills using the Aurora program into a text file. The second one is to read the file created by the previous one and store that data into the database using a PHP program. This program will read and fetch data from that text file, building a connection with the MySQL Windmill database, and it will insert values into a table named status.

There are four different ways to chain multiple commands in one shell.

1. Write each jobs on individual lines;
   All jobs will be executed.
2. Adding “;” between each job;
   All jobs will be executed.
3. Adding “&&” between each job;
   If jobs being executed till facing an invalid job fail, other jobs behind this job will not be executed.
4. Adding “||” between each job;
   Cron script will keep trying to find a valid job and finish it. If all of them have failed, Cron will try them one more time.

The Aurora inverter does not provide measurement data if there is no wind. Therefore, the first part of the Cron job, reading the data, may fail. If it does, there is no data that can be stored into the database and therefore the second part should not be run. This is inactived by chaining the two parts with the ’&&’ operator.

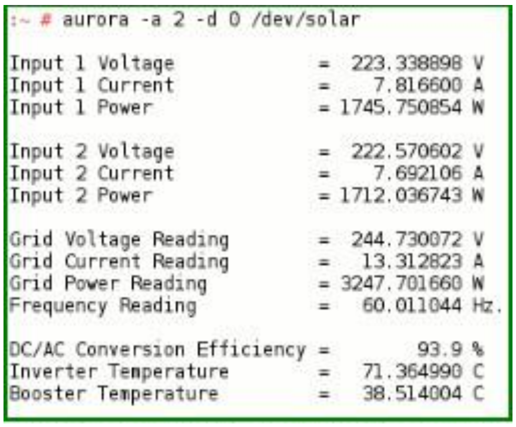
In a Linux system, when a Cron job is executed, the system will post an email to the current user. It contains either the result or the error message. This will take a lot of space as my Cron jobs work every 30 seconds. There are 3 ways to solve this problem -

1. /dev/null 2>&1; (redirect the output and error into a black hole)
2. Using –O- in front of the Cron job; (outputs the contents rather than saves them)
3. Enter MAILTO="" ; on the top of the file (disables the email for all Cron jobs)

In order to read the voltage, the current, the power, the conversion efficiency and the inverter temperature, we will use a command -

"aurora -a 2 -d 0 /dev/solar;"

which will print the output shown in Figure 4.



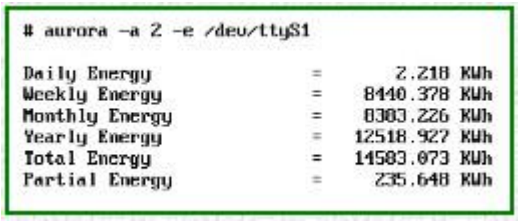
```
:~ # aurora -a 2 -d 0 /dev/solar

Input 1 Voltage              =  223.338898 V
Input 1 Current              =    7.816600 A
Input 1 Power                = 1745.750854 W

Input 2 Voltage              =  222.570602 V
Input 2 Current              =    7.692106 A
Input 2 Power                = 1712.036743 W

Grid Voltage Reading         =  244.730072 V
Grid Current Reading         =   13.312823 A
Grid Power Reading           = 3247.701660 W
Frequency Reading            =   60.011044 Hz.

DC/AC Conversion Efficiency =       93.9 %
Inverter Temperature         =   71.364990 C
Booster Temperature          =   38.514004 C
```

FIGURE 4. Aurora Command 1.

For the reading of the energy generated I will use –

"aurora -a 2 -e 0 /dev/ttyS1".

Which will print the output shown in Figure 5.



```
# aurora -a 2 -e /dev/ttyS1

Daily Energy              =       2.218 KWh
Weekly Energy             =    8440.378 KWh
Monthly Energy            =    8383.226 KWh
Yearly Energy             =   12518.927 KWh
Total Energy              =   14583.073 KWh
Partial Energy            =     235.648 KWh
```

FIGURE 5. Aurora Command 2.

These jobs can be managed in one task and separately by “&&”. A command for one single windmill will be

```
* * * * * aurora -a 2 -d 0 /dev/solar > /var/www/windmill/output_1. txt >/dev/null 2>&1&&
          aurora -a 2 -e 0 /dev/ttyS1 >> /var/www/windmill/output_1. txt >/dev/null 2>&1&&
          wget http: //localhost/windmill/fetch_aurora.php>/dev/null 2>&1;
```

“>>” sign means to redirect the output and append the data into a text file. “ > ” sign will redirect the output into a text file and overwrite the previous contents. Thus the text file only contains the current data. This will save the memory and makes it easier to get the data from the text file.
As we have 3 windmills running at the same time , these commands have to be written 3 times with different addresses, aurora –a 2 for inverter 1, aurora –a 3 for inverter 2 and aurora –a 4 for inverter 3.

The real requirement was not that simple, the owner requested the data to be collected every 30 seconds, but the minimum time scale of Cron job is 1 minute. Thus, I had to use an external shell script to do that.

The reason I decided to use Cron instead of using just one Bash script with a loop and sleep 30 seconds is that sleep is not accurate, the sampling thing would drift.

Bash has a ’sleep’ command, which can delay a process for a while. Therefore, in order to get the data collection executed twice a minute, there are two parts in the shell script, one collects the data once a minute and another after a 30 second delay.

The collection is done by ’a. sh’. It is executed once a minute.

```
#! /bin/bash
aurora -a 2 -d 0 /dev/solar > /var/www/windmill/output_1. txt >/dev/null 2>&1&&
aurora -a 2 -e 0 /dev/ttyS1 >> /var/www/windmill/output_1. txt >/dev/null 2>&1&&
wget http: //localhost/windmill/fetch_aurora.php>/dev/null 2>&1;
Sleep 30
```

```
aurora -a 2 -d 0 /dev/solar > /var/www/windmill/output_1. txt >/dev/null 2>&1&&
aurora -a 2 -e 0 /dev/ttyS1 >> /var/www/windmill/output_1. txt >/dev/null 2>&1&&
wget http: //localhost/windmill/fetch_aurora.php>/dev/null 2>&1;
```

The Cron job then runs this a. sh.

```
* * * * * bash /var/www/login/a. sh
```

4.2 Create Database and Table

The database is set up by a single PHP webpage. This page requires the MySQL username and password to login into the MySQL database. The appearance of this webpage is shown in Figure 6, which will create a new database named ‘windmill’ and a table named ‘status’. This script is shown in Appendix 1.

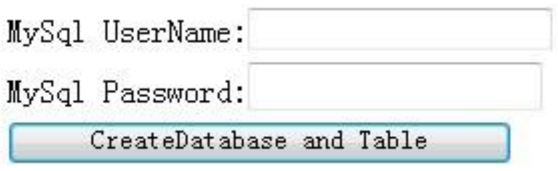
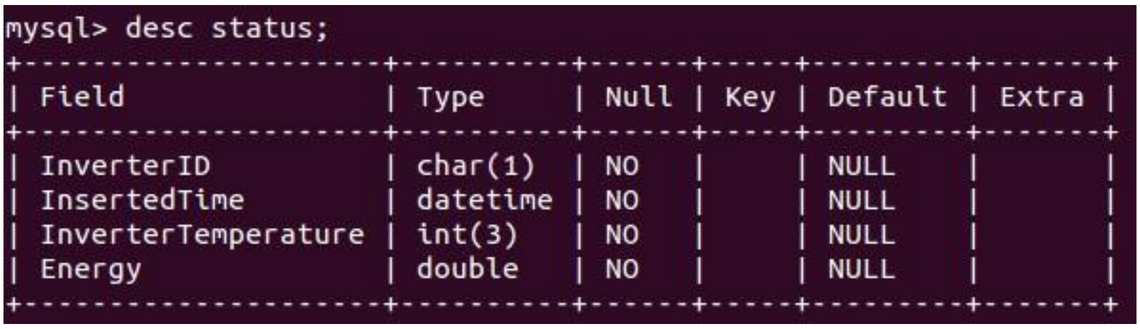


FIGURE 6. Database Creation Webpage

The Figure 7 below shows the table structure.

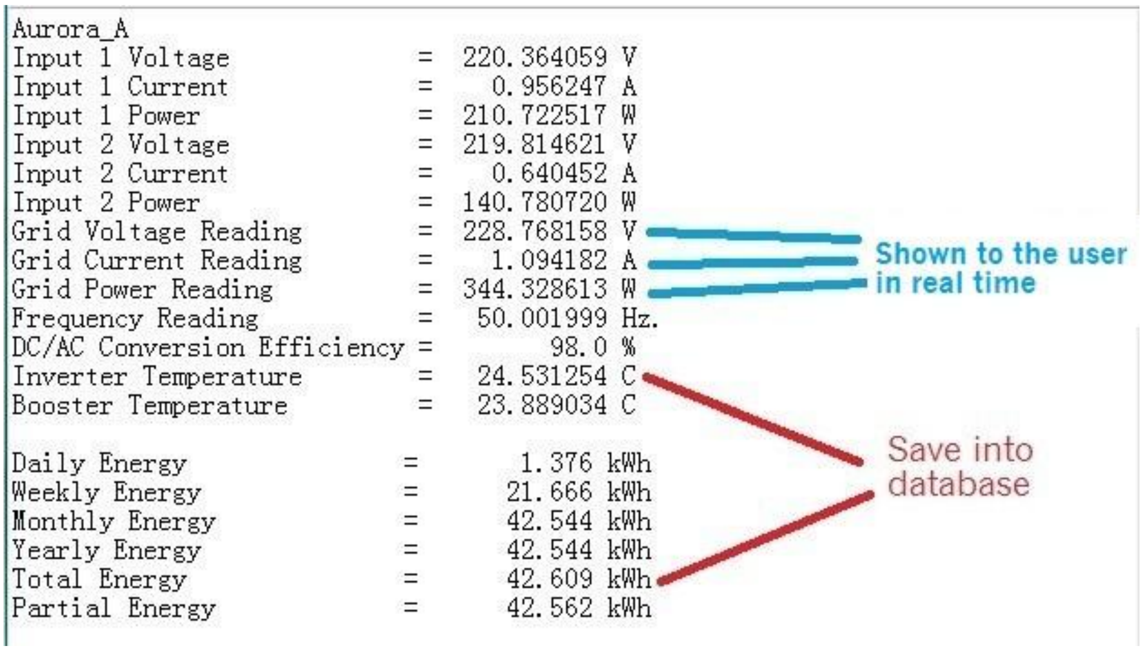


```
mysql> desc status;
+---------------------+----------+------+-----+---------+-------+
| Field               | Type     | Null | Key | Default | Extra |
+---------------------+----------+------+-----+---------+-------+
| InverterID          | char(1)  | NO   |     | NULL    |       |
| InsertedTime        | datetime | NO   |     | NULL    |       |
| InverterTemperature | int(3)   | NO   |     | NULL    |       |
| Energy              | double   | NO   |     | NULL    |       |
+---------------------+----------+------+-----+---------+-------+
```

FIGURE 7. Database table Structure.

4.3 Implement a Program for fetching data and save into MySQL database

The status data of the windmills are read from the Aurora inverters to a temporary text file as shown in Figure 8. The needed information is fetched from that key word, by a script called ’fetchgetcharacter.php’. It will look for the keywords from the text file and it takes the data from those lines.



```
Aurora_A
Input 1 Voltage              =  220.364059 V
Input 1 Current              =    0.956247 A
Input 1 Power                =  210.722517 W
Input 2 Voltage              =  219.814621 V
Input 2 Current              =    0.640452 A
Input 2 Power                =  140.780720 W
Grid Voltage Reading         =  228.768158 V
Grid Current Reading         =    1.094182 A
Grid Power Reading           =  344.328613 W
Frequency Reading            =   50.001999 Hz.
DC/AC Conversion Efficiency  =        98.0 %
Inverter Temperature         =   24.531254 C
Booster Temperature          =   23.889034 C

Daily Energy               =      1.376 kWh
Weekly Energy              =     21.666 kWh
Monthly Energy             =     42.544 kWh
Yearly Energy              =     42.544 kWh
Total Energy               =     42.609 kWh
Partial Energy             =     42.562 kWh
```

FIGURE 8. Aurora example of the temporary text file.

Next, this script opens a connection with the ’windmill’ database, and stores the fetched values into the table ’status’. This data contains the inverter temperature and total energy of the windmills from the last 30 seconds. The rest of the data such as ’Grid Voltage’, ’Grid Current’, ’Grid Power’ will be read directly from this text file when the monitoring website is opened. This script is shown in Appendix 2. The activity diagram of the fetching script is shown in Figure 9.

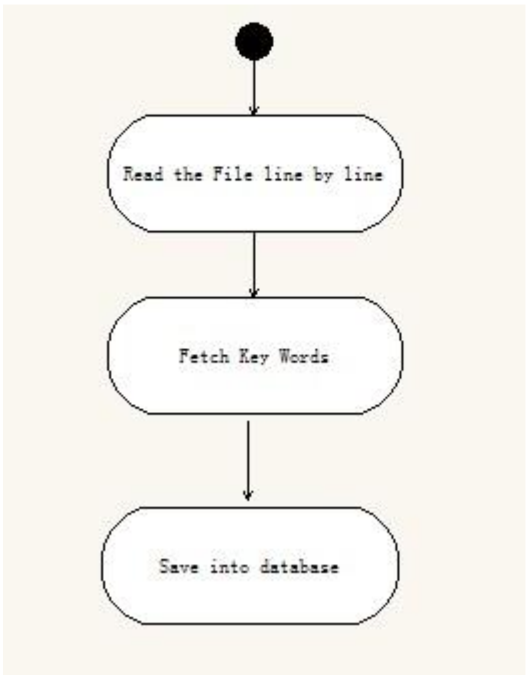


FIGURE 9. Activity Diagram of the fetching script.

4.4 PHP Website

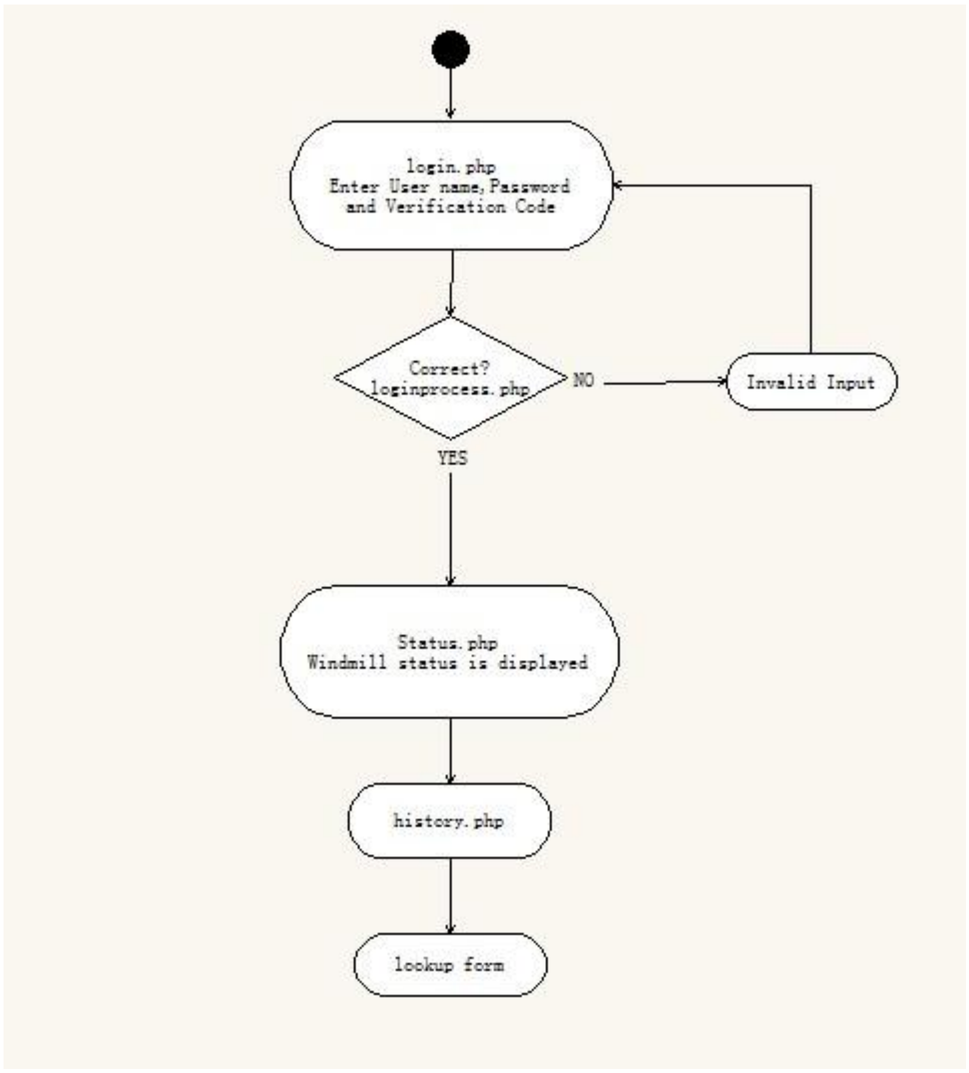


FIGURE 10. Activity diagram of the website.

Figure 10 above describes how this website works.

4.4.1 Login page

As it was mentioned, the website is accessed through a wireless connection. The user may not want anyone else to see this monitoring web page. The authentication process is shown in Figure 11.

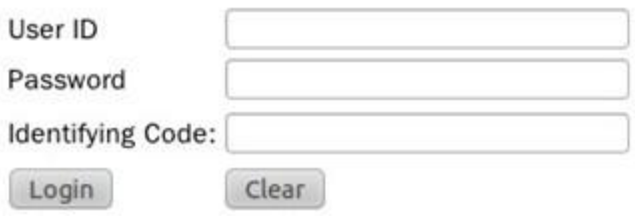


FIGURE 11. Login identification.

The login page contains a form that requests account, password and verification code. The user can login into this windmill monitoring web from 'login.php', by providing the user name, the password and a verification code. The script is shown in Appendix 4. The verification code is used in order to prevent someone trying to guess the username and the password with the Trojans. The verification code will be displayed as an image, generated by a PHP script named 'identificationcode.php'. (Shunping Han, 2013. From the entry to the master, date of retrieval 30.5.2013) This script will generate 4 random hexadecimal letters from 0 to 15, as shown in Figure 12.

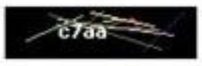


FIGURE 12. Verification Code.

For security reasons, these 4 digits should not be machine readable. Therefore, the font size is set randomly, and also need some random graphics to confuse the character recognizer, like drawing 10 lines from different directions with random colours. With the help of these, the identifying code will be difficult for a computer program to recognize but will still be readable by human beings. The script of this code is shown in Appendix 5.

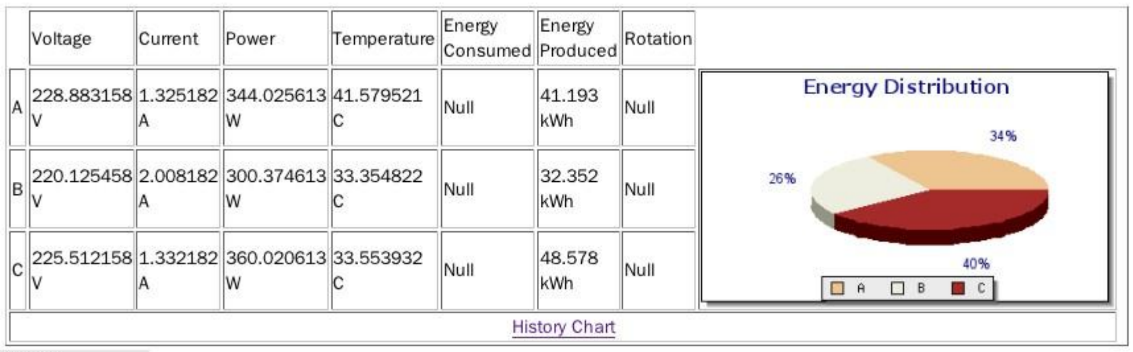
After all required information is filled in and the 'enter' button is clicked, the form is submitted to the 'loginprocess.php', which will check if the provided information is valid or not. From each failure, it will post a notification to the user telling which of the provided information was invalid. The script is shown in Appendix 6.

The value of the verification code is not a constant value. Still the one entered by the user needs to be compared with the generated one. This means the generated one has to be saved somewhere. To solve this problem, a 'session' is needed. (The PHP Group, 2013. Session Handling, date of retrieval 30.5.2013) The verification code is stored in a session variable and can be used from there to validate the entered code.

4.4.2 Daily Monitoring page

If all of the entered information is valid, 'login.php' directs the browser to 'status.php', which will display the data as texts and charts.

Texts are displayed in a table as shown in Figure13. The Charts are created using JpGraph (See Appendix 7).



	Voltage	Current	Power	Temperature	Energy Consumed	Energy Produced	Rotation
A	228.883158 V	1.325182 A	344.025613 W	41.579521 C	Null	41.193 kWh	Null
B	220.125458 V	2.008182 A	300.374613 W	33.354822 C	Null	32.352 kWh	Null
C	225.512158 V	1.332182 A	360.020613 W	33.553932 C	Null	48.578 kWh	Null

History Chart

FIGURE 13– Current status.

4.4.2.1 What is JpGraph

This Web page consists mostly of charts. Those charts are plotted using the JpGraph library. "JpGraph is an Object-Oriented Graph creating library for PHP >= 5.1. It is completely written in PHP and can be used in any PHP scripts (both CGI/APXS/CLI versions of PHP are supported)". (Asial Corporation, 2013, date of retrieval 30.4.2013)

JpGraph provides many templates that can be modified and used to create images. Using JpGraph charts shows the status of the windmill at a glance.

4.4.2.2 Features of JpGraph

JpGraph supports several plot types, spider-plots, pie-charts, line-plots, box plots, etc. It also supports flexible scales such as integer, linear, text scales and any combination of them on x/y axis. (Asial Corporation, 2013. JpGraph, date of retrieval 30.4.2013)
There are two kinds of charts used in this project. One is a line chart used to display the power of a single windmill during one single day. The other is a pie chart, used in showing the distribution of power generated by each windmill. From this chart the user can see when the windmills are not producing appropriate equally, which may indicate that there are problems with some of them, as shown in Figure 14. The graphic drawing PHP codes are in the Appendix 8.

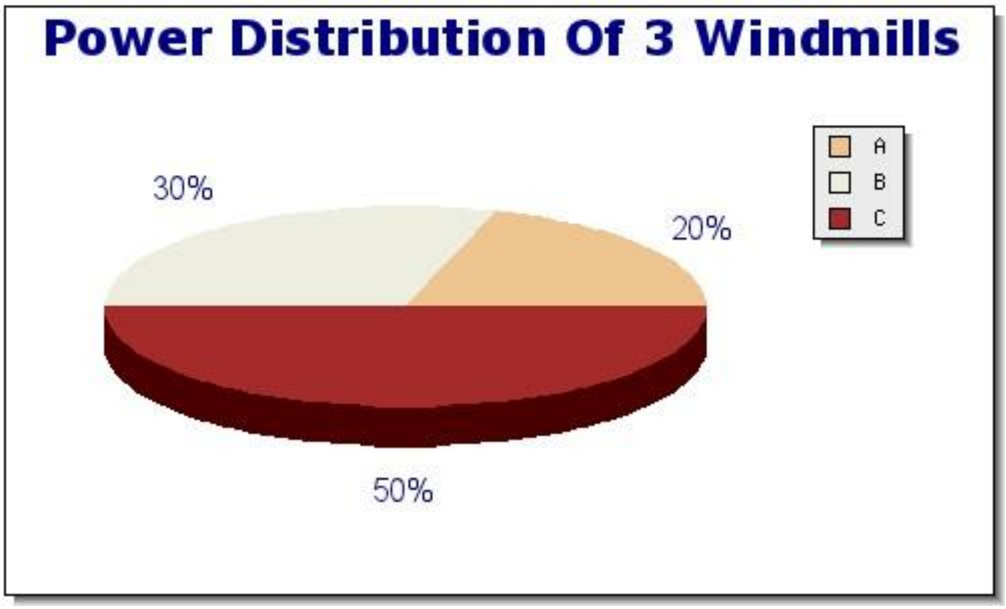


FIGURE 14. A pre-chart showing the power distribution of the windmills.

The most important and interesting part of this daily page is the line chart, which shows the average power as a function of time. The average power is calculated by dividing the energy produced during an interval by its length. For instance, if the interval is 10 minutes, the average power of these 10 minutes can be calculated by –

Average Power = (Energy [10 minutes after] – Energy [10 months before]) /10 * 60 seconds.

Figure 15 is a pre-chart according to the calculation above. The graphic drawing PHP codes are in the Appendix 9.

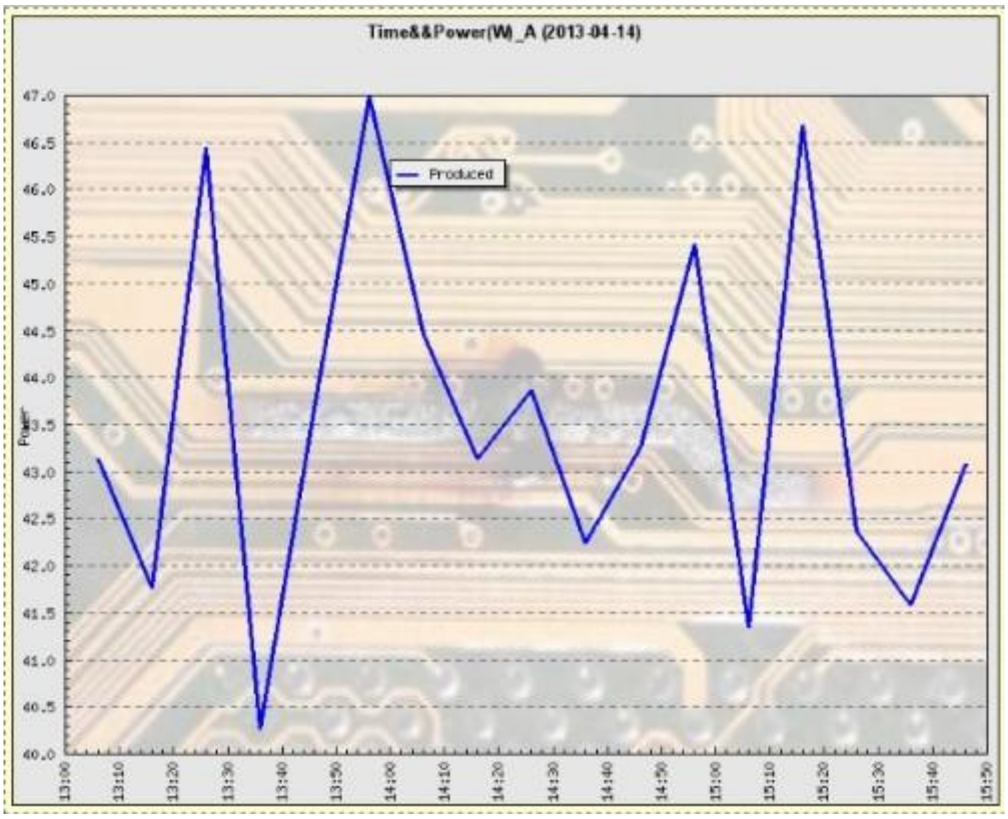
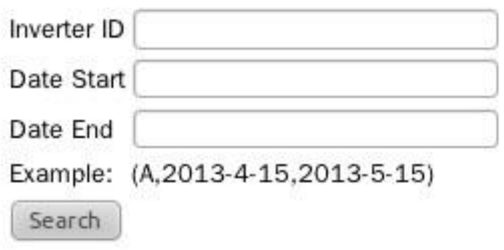


FIGURE 15. A line chart showing the average power as a function of time

4.4.5 History page

This page contains only one large line chart, which is used to display the monthly status of the 3 windmills.

The history page shows the status data of the current month. For further information, the user may use the lookup form, as shown in Figure 16. The user can enter the inverter ID, the start date and the end date into the form and press the ’Search’ button. The form is submitted to a PHP script which will show a new chart according to the provided information. The script of this code is shown in Appendix 10.


Inverter ID

Date Start

Date End

Example: (A,2013-4-15,2013-5-15)

FIGURE 16. Lookup function

5 TESTING

5.1 Test Plan

For testing I have simulated the windmill environment on my own laptop, which is not connected to the real windmill equipment. I have received one sample text file from Santtu Pennanen. This sample file contains all necessary information like voltage, power, etc. Thus, I could generate some similar testing data.

The line chart will show the power as the function of time. The power is not available in the text file. Instead, it is calculated by dividing the energy produced by the time.

“Daily Energy = 1.376kWh.”

The reading of daily produced energy is 1.376kWh from the sample text file.

Energy produced during 30 seconds = {[(Daily Energy/24hours) /60minutes]} /2= 0.000475kWh

Assume the energy produced sample is between 0.0004kWh and 0.0006kWh every 30 seconds. The data inserted into database will be a random number between 0.0004 and 0.0006 plus the previous value.

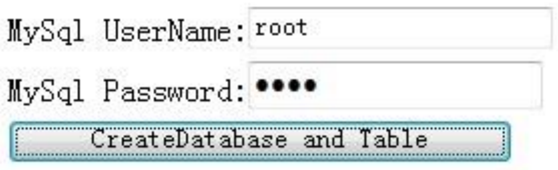
To calculate the average power, the interval is set to be 10 minutes, which has 20 samples. The average power will be the energy difference between these 10 minutes divided by 10 minutes.

P_v = (P_last – P_first) /10 minutes.

Because the test environment cannot communicate with the inverter, the Aurora program, which reads the data, cannot be used in the testing. Instead, I implemented a PHP script that extracts the last inserted data from the database and adds the energy producing assumption and then inserts the new data into the database.

```
$assumption = sprintf (“%. 5f”, rand (60,100) /100000);
$sql = "select energy from mytest where id ='A'  order by mydate desc limit 0,1;";
$res = mysql_query ($sql, $conn) or die (mysql_error ());
$result = mysql_fetch_array ($res);
$energy += $result ['energy'];
```

Now we can begin the testing by creating a new database and the table using the PHP script mentioned in the implementation chapter, as shown in Figure 17.


MySql UserName: root
MySql Password: ●●●●
CreateDatabase and Table

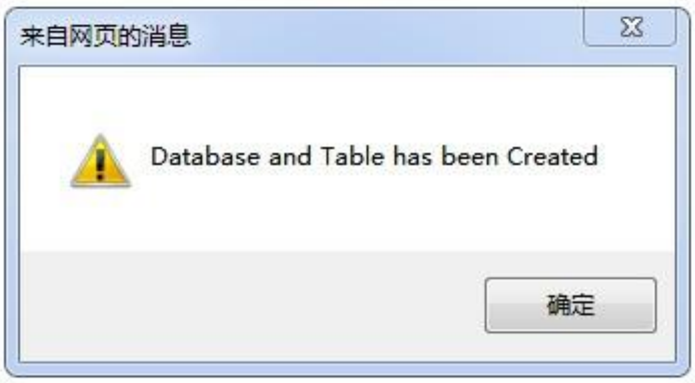
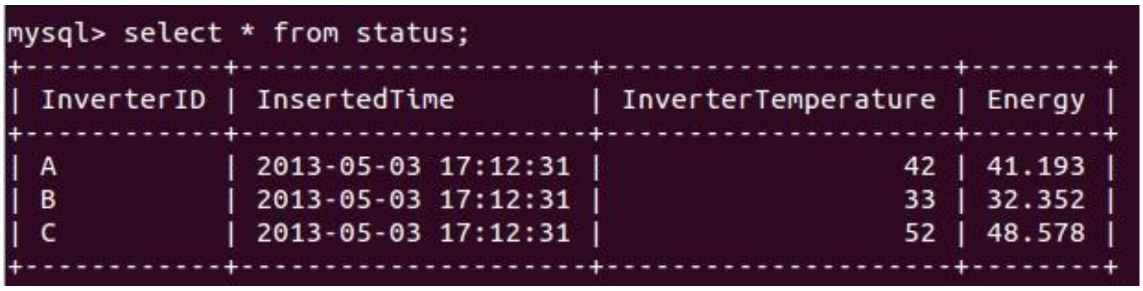

来自网页的消息
Database and Table has been Created
确定

FIGURE 17. Database Creation test.

The next step is to create one single Cron job to run this PHP script twice a minute. After a few moments, we will get some samples to fill in the charts.

```
* * * * * wget http://localhost/login/30seconds.php.
```

Figure 18 shows few sample data in the table.

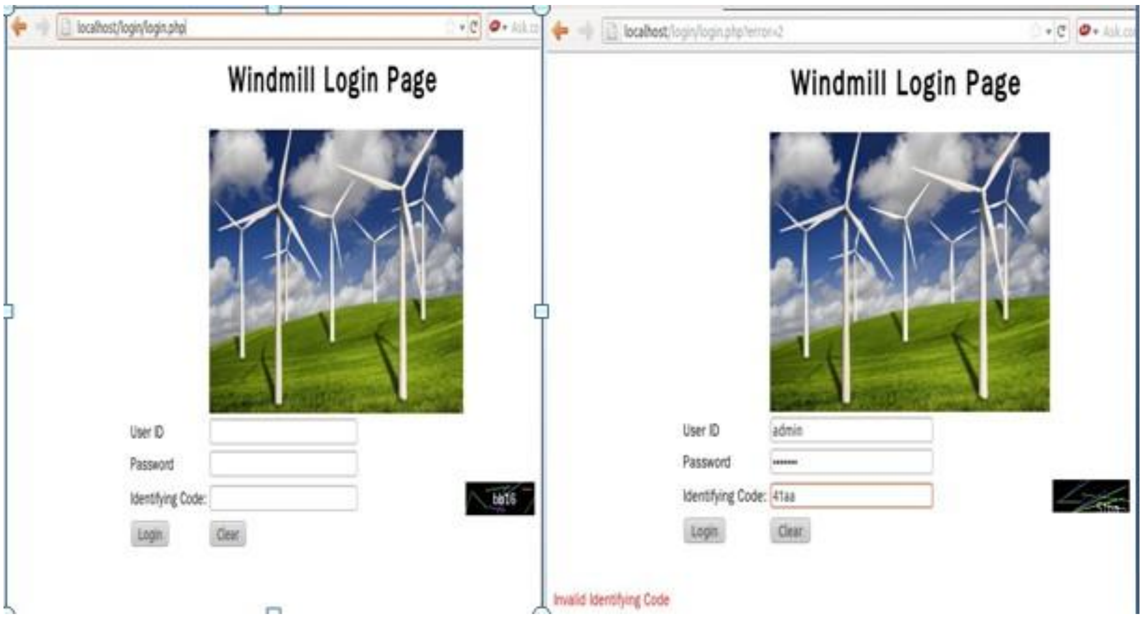


```
mysql> select * from status;
+------------+---------------------+---------------------+--------+
| InverterID | InsertedTime        | InverterTemperature | Energy |
+------------+---------------------+---------------------+--------+
| A          | 2013-05-03 17:12:31 |                  42 | 41.193 |
| B          | 2013-05-03 17:12:31 |                  33 | 32.352 |
| C          | 2013-05-03 17:12:31 |                  52 | 48.578 |
+------------+---------------------+---------------------+--------+
```

FIGURE 18. Data Samples.

5.2Testing of Login Page

Files are in the directory /var/www/login/. To test this website, we start by entering the address of the saves into the web browser, which is ‘localhost/var/www/login/login.php’.



localhost/login/login.php

Windmill Login Page

User ID:

Password:

Identifying Code:

localhost/login/login.php?error=2

Windmill Login Page

User ID: admin

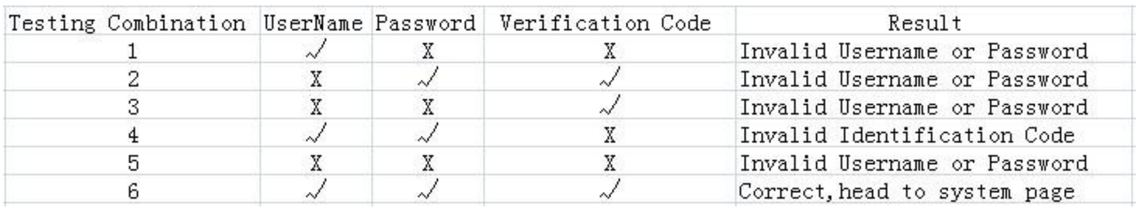
Password: ••••••

Identifying Code: 41aa

Invalid Identifying Code

FIGURE 19. Login Testing.

The legal Username and Password are both ’admin’, as it shows in the picture above. Providing the correct Username and Password with an incorrect verification code still gives an error. The combination of the error notification is shown in Figure 20.

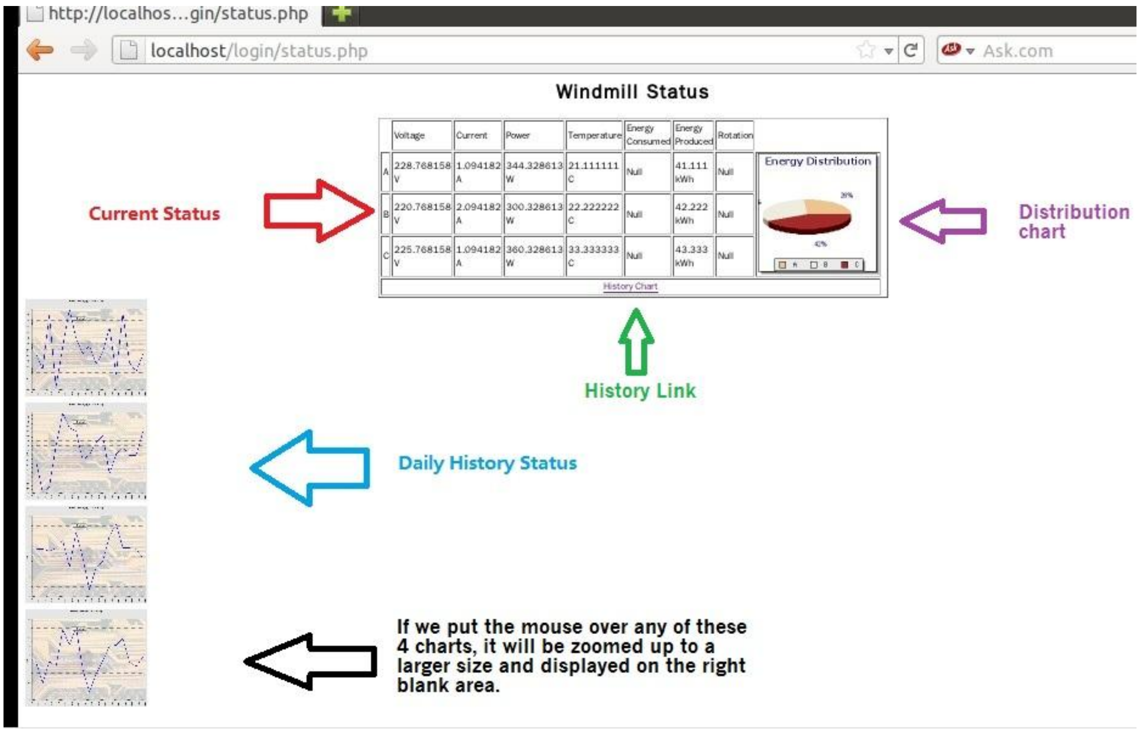


Testing Combination	UserName	Password	Verification Code	Result
1	√	X	X	Invalid Username or Password
2	X	√	√	Invalid Username or Password
3	X	X	√	Invalid Username or Password
4	√	√	X	Invalid Identification Code
5	X	X	X	Invalid Username or Password
6	√	√	√	Correct, head to system page

FIGURE 20. Notification combinations.

5.3Testing of Monitoring Page

After providing the valid login information, the browser loads the system page which shows the current status table, one distribution chart, and 4 line charts, as shown in Figure 21.



http://localhos...gin/status.php

localhost/login/status.php

Windmill Status

	Voltage	Current	Power	Temperature	Energy Consumed	Energy Produced	Rotation
A	228.768158 V	1.094182 A	344.328613 W	21.111111 C	Null	41.111 kWh	Null
B	220.768158 V	2.094182 A	300.328613 W	22.222222 C	Null	42.222 kWh	Null
C	225.768158 V	1.094182 A	360.328613 W	33.333333 C	Null	43.333 kWh	Null

History Chart

If we put the mouse over any of these 4 charts, it will be zoomed up to a larger size and displayed on the right blank area.

FIGURE 21. Daily page Testing.

By moving the mouse onto any of these 4 charts, a larger line chart should be shown as in the picture below.

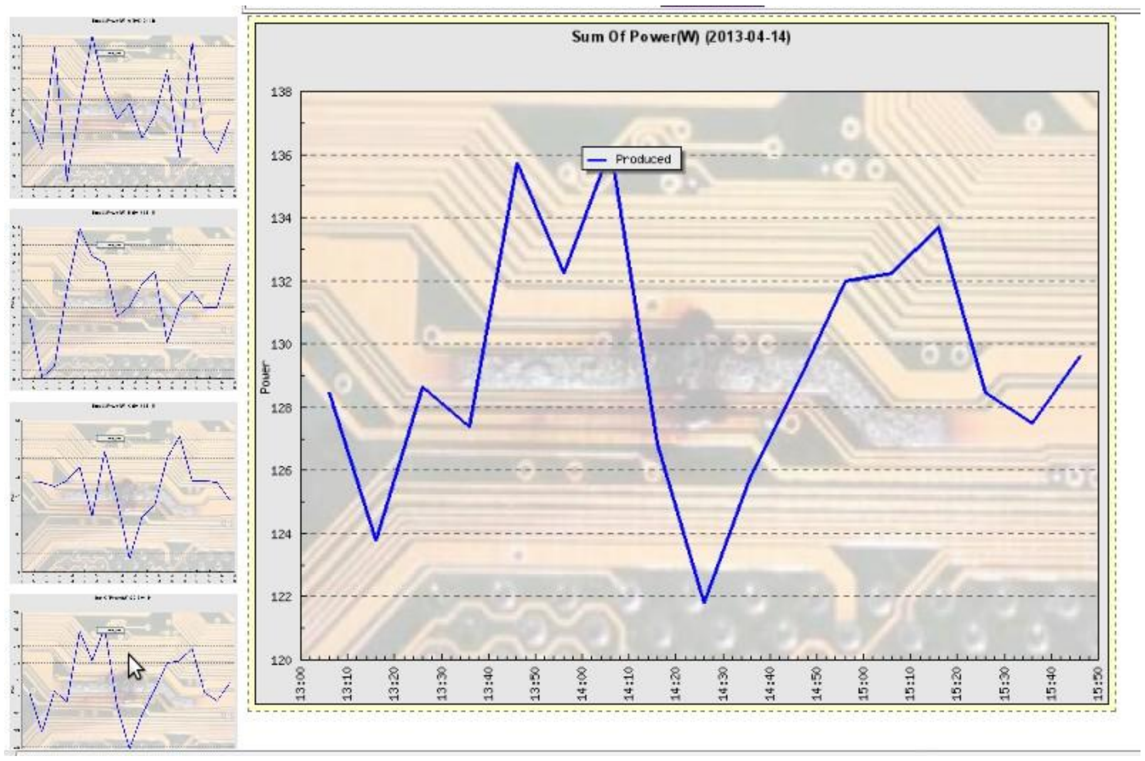


FIGURE 22. Zoom up line Chart.

The history part is under the history hyperlink. It contains one history chart and a search form. This history chart displays the status during the current month. If the user wants to see some specific dates, he can provide the ID of the windmill and the dates he is interested in. The history is shown in Figure 23.

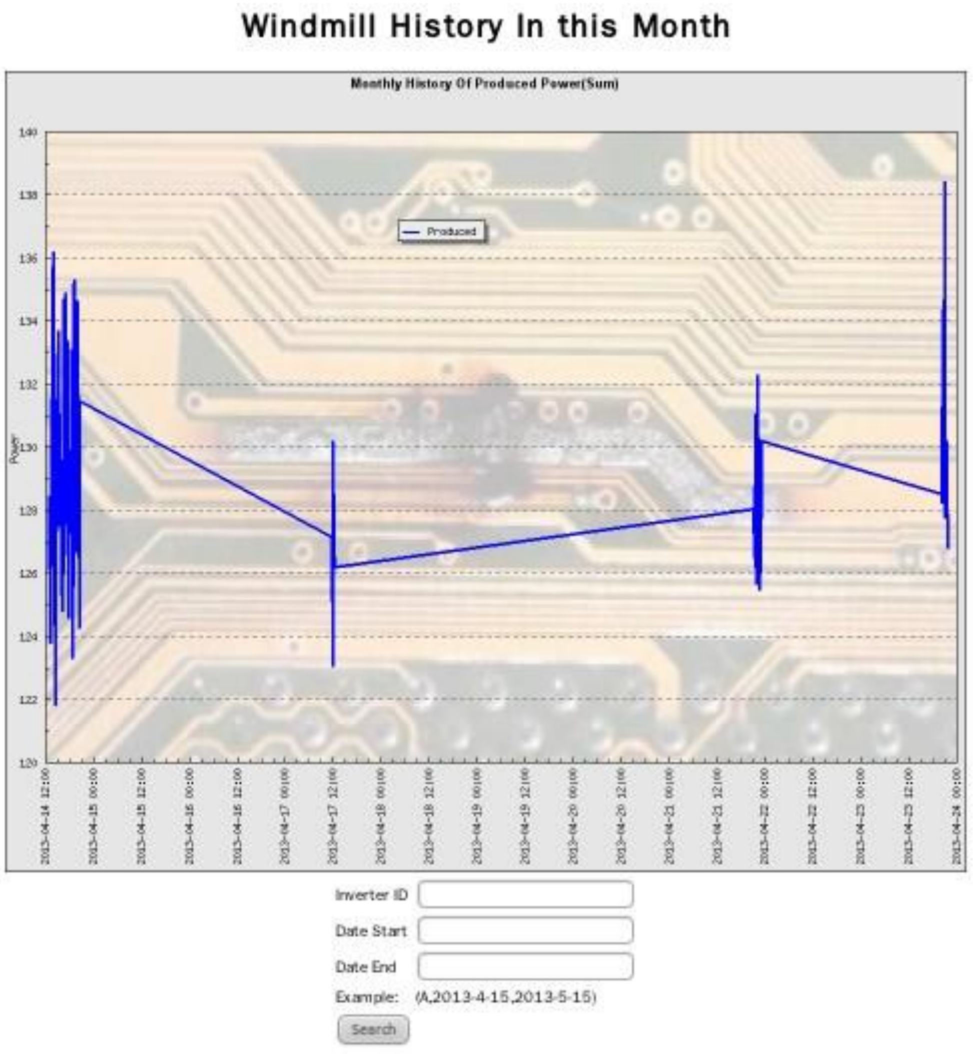


FIGURE 23. History Page.

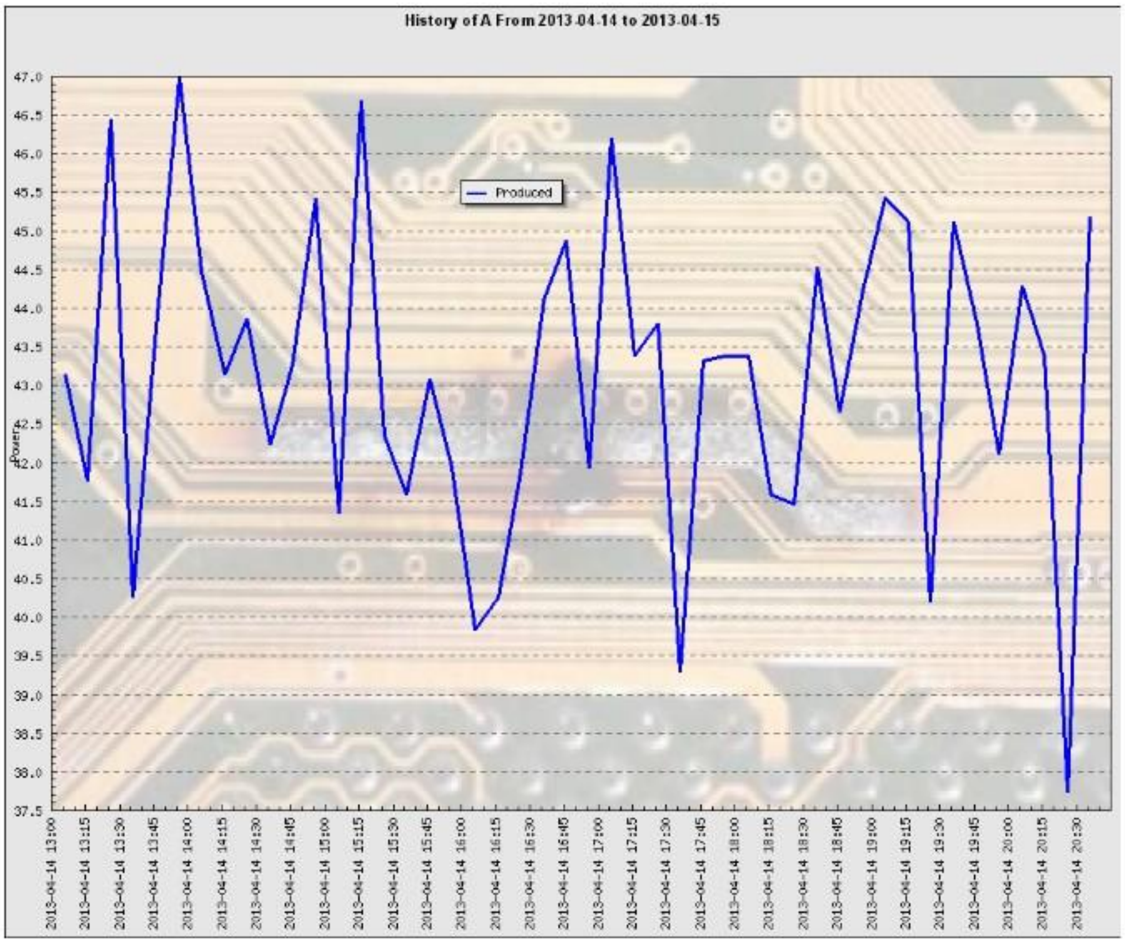


FIGURE 24. Example of a Detailed history search.

Figure 24 above is one example result of the search form.

6 POSSIBILITIES OF FURTHER DEVELOPMENT

Due to some technical obstacles, some readings such as the wind speed, wind direction and the power consumption of the house are not available yet. The software of this part will be implemented as soon as Santtu Pennanen has resolved these problems. After this the whole software is entirely completed.

7 CONCLUSION

Before this thesis work I had a very little experience with PHP and MySQL programming. In this project I had to develop a website that uses PHP and MySQL for almost all functionality. Therefore, to accomplish this thesis I needed a better understanding of PHP and MySQL and some software design skills. Even though this topic was challenging, the result was exciting and the knowledge I have learned from this project will be of a great help in my future studying career.

Wind power is an alternative to fossil fuels. It is clean, plentiful and renewable, and has zero emissions in the production of electric energy. In recent years, the using of wind power has become increasingly popular. However, the windmill monitoring software is not that popular yet, because a normal family using wind power may have a problem about how to monitor the process of the power generation. My thesis will help people to monitor their windmill status using a very simple website, which does not demand the person to have any specific skill.

PHP, as a scripting language designed for web developing, can easily be embedded into an HTML source document. With the help of PHP, the webpage becomes able to access dynamic objects such as database. This makes the using of a simple website to monitor the windmill status possible.

This project might have been done better if I had had my PHP skills at the level I currently have. During this work my design and programming skills improved and this thesis helped me to learn a lot about PHP and MySQL languages, and gain a better understanding of project planning. I believe this will be useful in the future.

LIST OF REFERENCES

Asial Corporation, 2013. JpGraph. Date of retrieval 30.04.2013

http://jpgraph.net/

Curtis J. Blank, 2013. Curtronics.com. Date of retrieval 02.05.2013

http://www.curtronics.com/Solar/AuroraData.html

Larmon Wayne, 2013. Using cron to run programs on a schedule. Date of retrieval 30.04.2013

http://www.scrounge.org/linux/cron.html

Power-one Inc, 2013. PVI-5000-OUTD-W, PVI-6000-OUTD-W. Date of retrieval 02.05.2013

http://www.power-one.com/sites/power-one.com/files/documents/renewable-energy/tech-manual/pvi-6000_outd-us-w-install_manual.pdf

Refsnes Data, 2013. MySQL Introduction. Date of retrieval 02.05.2013

http://www.w3schools.com/php/php_mysql_intro.asp

Shunping Han, 2013. From the entry to the master. Date of retrieval 03.05.2013

http://222.186.16.104/file/MDAwMDAwMDGSc3KPLCTxyKDXfywE7mYMImlH9YmuLXxxcyPWW92LDA../e6ce264d089bcaa2d8c077adc46384c4161e87f/%E4%BC%A0%E6%99%BA%E6%92%AD%E5%AE%A2_%E9%9F%A9%E9%A1%BA%E5%B9%B3_php%E4%BB%8E%E5%85%A5%E9%97%A8%E5%88%B0%E7%B2%BE%E9%80%9A%20%E8%A7%86%E9%A2%91%E6%95%99%E7%A8%8B%20%E7%AC%AC133%E8%AE%B2%20php%E7%BB%98%E5%9B%BE%E6%8A%80%E6%9C%AF%E2%91%A7-%E7%BB%98%E5%88%B6%E9%AA%8C%E8%AF%81%E7%A0%81.zip?key=AAABQFGEIi3ASWKy&p=&a=40429458-c1a79231-48049-0/010128&c=lr:5c15e6a7a34826829600a9654cc04bf4/200k&mode=download

The PHP Group, 2013. Session Handling. Date of retrieval 03.05.2013

http://www.php.net/manual/zh/book.session.php

Wikipedia, 2013. Cron. Date of retrieval 30.04.2013

http://en.wikipedia.org/wiki/Cron

Wikipedia, 2013. Linux. Date of retrieval 30.04.2013

http://en.wikipedia.org/wiki/Linux

Wikipedia, 2013. HTML. Date of retrieval 29.04.2013

http://en.wikipedia.org/wiki/HTML

Wikipedia, 2013. PHP. Date of retrieval 30.04.2013

http://en.wikipedia.org/wiki/PHP

APPENDICES

Appendix 1: The codes of NewDatabaseAndTable.php

```
<html>
<head>
<?php
if (! empty ( $_POST ['newdatabase'] )) {
		$mysql_hostname = "127.0.0.1";
		$mysql_user = addslashes ( $_POST ['username'] );
		$mysql_password = addslashes ( $_POST ['password'] );
		$mysql_database = "windmill";
		$bd = mysql_connect ( $mysql_hostname, $mysql_user, $mysql_password ) or
die ( "Opps some thing went wrong" );
		mysql_query ( "set names utf8" );
		$sql = "create database windmill";
		$res = mysql_query ( $sql );
		mysql_select_db ( $mysql_database, $bd ) or die ( "Opps some thing went
wrong" );

		$sql = "create table status (id char(1),mydate datetime,invertertemperature
double(4,1),energy double)";
		$res = mysql_query ( $sql );
		if (! $res) {
			echo "<script>alert('Creation Failed');</script>";
		}
		echo "<script>alert('Database and Table has been Created');</script>";
}
?>
<form action="" method="post">
		<table align='center'>
			<tr>
				<td><label>MySql UserName:</label><input
type="text" name="username" /><br /></td>
			</tr>
			<tr>
				<td><label>MySql Password:</label><input
type="password"
					name="password" /><br /></td>
			</tr>
			<td><input type="submit" name="newdatabase"
				value="CreateDatabase and Table" /></td>
			</tr>
		</table>
</form>
</head>
</html>
```

Appendix 2: The codes of FetchGetCharacter.php

```
<?php
function myfetch($name) {
		$handle = fopen ( $name, "r" );
		$myarray = array ();
		if ($handle) {
			while ( ! feof ( $handle ) ) {
				$buffer = fgets ( $handle, 1024 );
				$sign = substr ( $buffer, 0, 3 );
				$pos = strpos ( $buffer, '=' );

				if ($sign === 'Inv') {
					$myarray [0] = substr ( $buffer, $pos
+ 1, - 2 );
				} else if ($sign === 'Tot') {
					$myarray [1] = substr ( $buffer, $pos
+ 1, - 4 );
				}
			}
			fclose ( $handle );
			return $myarray;
		}
}
$array_A = myfetch ( "real_aurora_A.txt" );
$array_B = myfetch ( "real_aurora_B.txt" );
$array_C = myfetch ( "real_aurora_C.txt" );

$mysqli = new MySQLi ( "localhost", "root", "8862281", "windmill" );

if ($mysqli->connect_error) {
		die ( $mysqli->connect_error );
}

$sql = "insert into mytest (id,mydate,invertertemperature,energy)
values('A',now(),\"$array_A[0]\",\" $array_A[1]\");";
$sql .= "insert into mytest (id,mydate,invertertemperature,energy)
values('B',now(),\"$array_B[0]\",\" $array_B[1]\");";
$sql .= "insert into mytest (id,mydate,invertertemperature,energy)
values('C',now(),\"$array_C[0]\",\" $array_C[1]\");";

$b = $mysqli->multi_query ( $sql );
if (! $b) {
		echo "fail";
}
$mysqli->close ();
```

```
?>
```

Appendix 3: The codes of Lock.php

```
<?php
session_start();
$user_check=$_SESSION['login_user'];
$login_session=$user_check;
if(!isset($login_session))
{
header("Location: login.php");
}
?>
```

Appendix 4: The codes of Login.php

```
<?php
$identifycode = "";
for($i = 0; $i < 4; $i ++) {
                    $identifycode .= dechex ( rand ( 1, 15 ) );
}
session_start ();
$_SESSION ['MyIdentyfyCode'] = $identifycode;
$image1 = imagecreatetruecolor ( 110, 30 );

$white = imagecolorallocate ( $image1, 255, 255, 255 );
for($i = 0; $i < 10; $i ++) {
                    imageline ( $image1, rand ( 0, 110 ), rand ( 0, 30 ), rand ( 0, 110 ), rand ( 0, 30 ),
imagecolorallocate ( $image1, rand ( 0, 255 ), rand ( 0, 255 ), rand ( 0, 255 ) ) );
}
imagestring ( $image1, rand ( 2, 5 ), rand ( 0, 80 ), rand ( 0, 20 ), $identifycode, $white );
header ( "content-type:image/png" );
imagepng ( $image1 );
imagedestroy ( $image1 );
?>
```

Appendix 5: The codes of Verificationcode.php

```
<html>
<head>
<meta http-equiv = "content-type" content = "text/html;charset = utf-8"/>
</head>
<!--<center><img src="windmill.jpg"/></center> -->

<h1 align = center>Windmill Login Page</h1>
<body>
<form action = "loginprocess.php" method = "post">
<table align = "center" >
<tr><td></td><td><img src="windmill.jpg" width = "400px"/></td><td></td></tr>
<tr><td>User ID&nbsp;</td><td><input type = "text" name = "id"/></td></tr>
<tr><td>Password</td><td><input type = "password" name = "password"/></td></tr>
<tr>
<td>Identifying Code:</td><td><input type= "text" name = "identifycode"/><td><img src = 
"identify.php"/></td>
</tr>
<tr>
<td><input type = "submit"value = "Login"/></td>
<td><input type = "reset" value = "Clear"/></td>
</tr>
</table>
</form>
<?php
 if (!empty($_GET['error'])){
                    $error = $_GET['error'];
                    if($error == 1){
                                        echo"<br/><font color = 'red' size = '3'> Invalid User ID or 
Password</font>";
                    }else if ($error == 2){
                                        echo"<br/><font color = 'red' size = '3'> Invalid Identifying 
Code</font>";
    }
}
?>
</body>
</html>
```

Appendix 6: The codes of Loginprocess.php

```
<?php
$username = "admin";
$password = "admin";
$thisid = $_POST ['id'];
$thispassword = $_POST ['password'];
$myIdentifyCode = $_POST ['identifycode'];
session_start ();
if ($thisid == $username && $thispassword == $password) {
                if ($myIdentifyCode == $_SESSION ["MyIdentyfyCode"]) {
                                session_register ( "thisid" );
                                $_SESSION ['login_user'] = $thisid;
                                header ( "Location:status.php" );
                                exit ();
                } else {
                                header ( "Location:login.php?error=2" );
                                exit ();
                }
} else {
                header ( "Location:login.php?error=1" );
                exit ();
}
?>
```

Appendix 7: The codes of Status.php

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset = utf-8" />
<meta http-equiv="refresh" content="60">
<!--<style>#pic{max-width:600px; max-height:600px}</style> -->
<style type="text/css">
swap:hover {
                background-color: #FFFF44;
                z-index: 50;
}

.swap span,.thumbnail span {
                background-color: #FFFFCC;
                border: 1px dashed gray;
                color: black;
                left: -1000px;
                padding: 5px;
                position: absolute;
}

.swap:hover span {
                left: 220;
                position: absolute;
                top: 360px;
}
</style>
</head>
<h1 align=center>Windmill Status</h1>
<body>
                <table align="center" border="2px" width="33%">
<?php
include ('lock.php');
function myfetch($name) {
                $handle = fopen ( $name, "r" );
                $myarray = array ();
                if ($handle) {
                                while ( ! feof ( $handle ) ) {
                                                $buffer = fgets ( $handle, 1024 );
                                                // $num = substr($buffer, 6,1);
                                                $sign = substr ( $buffer, 0, 3 );
                                                $pos = strpos ( $buffer, '=' );
                                                // get char of unit symble
                                                $unit = substr ( $buffer, - 2, 1 );

                                                if ($sign === 'Gri') {
```

```
                                        if ($unit === 'V') {
                                                $myarray [0] = substr
( $buffer, $pos + 1 );
                                        } else if ($unit === 'A') {
                                                $myarray [1] = substr
( $buffer, $pos + 1 );
                                        } else if ($unit === 'W') {
                                                $myarray [2] = substr
( $buffer, $pos + 1 );
                                        }
                                } else if ($sign === 'Inv') {
                                        $myarray [3] = substr ( $buffer, $pos
+ 1, - 2 );
                                } else if ($sign === 'Tot') {
                                        $myarray [4] = substr ( $buffer, $pos
+ 1, - 4 );
                                }
                        }
                }
                fclose ( $handle );
                return $myarray;
}
$array_A = myfetch ( "real_aurora_A.txt" );
$array_B = myfetch ( "real_aurora_B.txt" );
$array_C = myfetch ( "real_aurora_C.txt" );
$energy_A = $array_A [4];
$energy_B = $array_B [4];
$energy_C = $array_C [4];
session_register ( "energy_A" );
$_SESSION ['energy_A'] = $energy_A;
session_register ( "energy_B" );
$_SESSION ['energy_B'] = $energy_B;
session_register ( "energy_C" );
$_SESSION ['energy_C'] = $energy_C;
echo "
<tr>
<td></td>
<td>Voltage</td>
<td>Current</td>
<td>Power</td>
<td>Temperature</td>
<td>Energy Consumed</td>
<td>Energy Produced</td>
<td>Rotation</td>
<td></td>
</tr>
<tr>
<td>A</td><td>$array_A[0]</td><td>$array_A[1]</td><td>$array_A[2]</td><td>$array_A[3]C</td
```

```
><td>Null</td><td>$array_A[4]kWh</td><td>Null</td>
<td rowspan=\"3\"><img src = \"distribution.php\"></td>
</tr>
<tr>
<td>B</td><td>$array_B[0]</td><td>$array_B[1]</td><td>$array_B[2]</td><td>$array_B[3]C</td
><td>Null</td><td>$array_B[4]kWh</td><td>Null</td><td></td>
</tr>
<tr>
<td>C</td><td>$array_C[0]</td><td>$array_C[1]</td><td>$array_C[2]</td><td>$array_C[3]C</td
><td>Null</td><td>$array_C[4]kWh</td><td>Null</td><td></td>
</tr>
<tr><td align = \"center\" colspan = \"9\"><a href = \"history.php\">History Chart</a></td></tr>";
?>
                    </table>
                    <table align="center">

                    </table>
                    <table align="left">
                                        <tr>
                                                            <td><a class="swap"><img width="200px"
src="daily.php?id=A"> <span> <img

                    alt="Fail" src="daily.php?id=A">
                                                                                </span> </a></td>
                                        </tr>
                                        <tr>
                                                            <td><a class="swap"><img width="200px"
src="daily.php?id=B"> <span> <img

                    alt="Fail" src="daily.php?id=B">
                                                                                </span> </a></td>
                                        </tr>
                                        <tr>
                                                            <td><a class="swap"><img width="200px"
src="daily.php?id=C"> <span> <img

                    alt="Fail" src="daily.php?id=C">
                                                                                </span> </a></td>
                                                            </td>
                                        </tr>
                                        <tr>
                                                            <td><a class="swap"><img width="200px"
src="daily.php?id=SUM"> <span>
                                                                                                          <img
alt="Fail" src="daily.php?id=SUM">
                                                                                </span> </a></td>
                                        </tr>
                    </table>
```

```
</body>
</html>
```

Appendix 8: The codes of Distibution.php

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_pie.php');
require_once ('jpgraph/jpgraph_pie3d.php');

// Some data
session_start();
$energy_A=$_SESSION['energy_A'];
$energy_B=$_SESSION['energy_B'];
$energy_C=$_SESSION['energy_C'];
$data = array($energy_A,$energy_B,$energy_C);
// Create the Pie Graph.
$graph = new PieGraph(350,200);
$graph->SetShadow();
$graph->SetMargin(10,30,30,30);
// Set A title for the plot
$graph->title->Set("Energy Distribution");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor("darkblue");
$graph->legend->Pos(0.5,0.97,'center','bottom');
$graph->legend->SetColumns(3);
// Create pie plot
$p1 = new PiePlot3d($data);
$p1->SetTheme("sand");
$p1->SetCenter(0.55);
$p1->SetAngle(30);
$p1->value->SetFont(FF_ARIAL,FS_NORMAL,9);
$p1->SetLegends(array("A","B","C"));
$graph->Add($p1);
$graph->Stroke();

?>
```

Appendix 9: The codes of Daily.php

```
<?php
// content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_line.php');
require_once ('jpgraph/jpgraph_date.php');
// getting data from mysql

$date = date ( "Y-m-d", time () );
$month = date ( "m", time () );
$conn = mysql_connect ( "localhost", "root", "8862281" ) or die ( "connction failed" . mysql_error
() );
mysql_select_db ( "windmill", $conn ) or die ( mysql_error () );

$eydata = array ();
$exdata = array ();
$pydata = array ();
$pxdata = array ();
$factor = 10;
// ////////////////////////////////////////////////
$id = $_GET ['id'];
$format = 'H:i';

if ($id == 'A') {
                $mytitle = "Time&&Power(W)_A ($date)";
                $sql = "select * from mytest where date(mydate)=date(now()) and id='A'";
                $res = mysql_query ( $sql, $conn ) or die ( mysql_error () );
                $i = 0;
                while ( $row = mysql_fetch_array ( $res ) ) {
                                $eydata [$i] = $row ['energy'];
                                $exdata [$i] = strtotime ( $row ['mydate'] );
                                $i ++;
                }
                for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                                $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                                $pxdata [$j] = $exdata [$j * $factor + 9];
                }

                mysql_free_result ( $res );
                $width = 750;
                $height = 600;
} else if ($id == 'B') {
                $mytitle = "Time&&Power(W)_B ($date)";
                $sql = "select * from mytest where date(mydate)=date(now()) and id='B'";
                $res = mysql_query ( $sql, $conn ) or die ( mysql_error () );
                $i = 0;
                while ( $row = mysql_fetch_array ( $res ) ) {
                                $eydata [$i] = $row ['energy'];
                                $exdata [$i] = strtotime ( $row ['mydate'] );
                                $i ++;
                }
                for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                                $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                                $pxdata [$j] = $exdata [$j * $factor + 9];
                }

                mysql_free_result ( $res );
                $width = 750;
                $height = 600;
} else if ($id == 'C') {
                $mytitle = "Time&&Power(W)_C ($date)";
                $sql = "select * from mytest where date(mydate)=date(now()) and id='C'";
                $res = mysql_query ( $sql, $conn ) or die ( mysql_error () );
                $i = 0;
                while ( $row = mysql_fetch_array ( $res ) ) {
                                $eydata [$i] = $row ['energy'];
                                $exdata [$i] = strtotime ( $row ['mydate'] );
                                $i ++;
                }
                for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                                $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                                $pxdata [$j] = $exdata [$j * $factor + 9];
                }

                mysql_free_result ( $res );
                $width = 750;
                $height = 600;
}

else if ($id == "SUM") {
                $mytitle = "Sum Of Power(W) ($date)";
                // //////////////A
                $sql_A = "select * from mytest where date(mydate)=date(now()) and id='A'";
                $sql_B = "select * from mytest where date(mydate)=date(now()) and id='B'";
                $sql_C = "select * from mytest where date(mydate)=date(now()) and id='C'";

                $res_A = mysql_query ( $sql_A, $conn ) or die ( mysql_error () );
                $res_B = mysql_query ( $sql_B, $conn ) or die ( mysql_error () );
                $res_C = mysql_query ( $sql_C, $conn ) or die ( mysql_error () );

                $i = 0;
                while ( $row_A = mysql_fetch_array ( $res_A ) ) {
                        $eydata [$i] = $row_A ['energy'];
                        $exdata [$i] = strtotime ( $row_A ['mydate'] );
                        $i ++;
                }
                $i = 0;
                while ( $row_B = mysql_fetch_array ( $res_B ) ) {
                        $eydata [$i] += $row_B ['energy'];

                        $i ++;
                }
                $i = 0;
                while ( $row_C = mysql_fetch_array ( $res_C ) ) {
                        $eydata [$i] += $row_C ['energy'];
                        $i ++;
                }
                for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                        $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                        $pxdata [$j] = $exdata [$j * $factor + 9];
                }
                mysql_free_result ( $res_A );
                mysql_free_result ( $res_B );
                mysql_free_result ( $res_C );
                $width = 750;
                $height = 600;
} else if ($id == "MONTH") {
                $mytitle = "Monthly History Of Produced Power(Sum)";
                $sql_A = "select * from mytest where month(mydate)=month(now()) and id='A'";
                $sql_B = "select * from mytest where month(mydate)=month(now()) and id='B'";
                $sql_C = "select * from mytest where month(mydate)=month(now()) and id='C'";
                $res_A = mysql_query ( $sql_A, $conn ) or die ( mysql_error () );
                $res_B = mysql_query ( $sql_B, $conn ) or die ( mysql_error () );
                $res_C = mysql_query ( $sql_C, $conn ) or die ( mysql_error () );
                $i = 0;
                while ( $row_A = mysql_fetch_array ( $res_A ) ) {
                        $eydata [$i] = $row_A ['energy'];
                        $exdata [$i] = strtotime ( $row_A ['mydate'] );
                        $i ++;
                }
                $i = 0;
                while ( $row_B = mysql_fetch_array ( $res_B ) ) {
                        $eydata [$i] += $row_B ['energy'];
                        $i ++;
                }

                $i = 0;
                while ( $row_C = mysql_fetch_array ( $res_C ) ) {
                        $eydata [$i] += $row_C ['energy'];
```

```
                                $i ++;
                        }
                        for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                                $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                                $pxdata [$j] = $exdata [$j * $factor + 9];
                        }
                        mysql_free_result ( $res_A );
                        mysql_free_result ( $res_B );
                        mysql_free_result ( $res_C );
                        $format = 'Y-m-d H:i';
                        $width = 1680;
                        $height = 1050;
} else if ($id == "FETCH") {
                        $num = $_POST ['num'];
                        $start = $_POST ['start'];
                        $end = $_POST ['end'];

                        $mytitle = "History of $num From $start to $end";
                        $sql = "select * from mytest where id='$num' and date(mydate)>='$start' and
date(mydate)<='$end'";
                        $res = mysql_query ( $sql, $conn ) or die ( mysql_error () );
                        $i = 0;
                        while ( $row = mysql_fetch_array ( $res ) ) {
                                $eydata [$i] = $row ['energy'];
                                $exdata [$i] = strtotime ( $row ['mydate'] );
                                $i ++;
                        }
                        for($j = 0; $j < sprintf ( "%u", $i / $factor ); $j ++) {
                                $pydata [$j] = ($eydata [$j * $factor + 9] - $eydata [$j * $factor]) *
((1000 * 60) / $factor); // kwh divide 10min is = * 6
                                $pxdata [$j] = $exdata [$j * $factor + 9];
                        }
                        mysql_free_result ( $res );
                        $format = 'Y-m-d H:i';
                        $width = 1680;
                        $height = 1050;
}

mysql_close ( $conn );

// Create the graph. These two calls are always required
$graph = new Graph ( $width, $height );
$graph->SetScale ( "datlin" );
$graph->SetMargin ( 40, 40, 80, 50 );
// Setup the grid and plotarea box
$graph->ygrid->SetLineStyle ( 'dashed' );
$graph->ygrid->setColor ( 'darkgray' );
```

```
$graph->SetBox ( true );
// Steup graph titles
$graph->title->SetFont ( FF_ARIAL, FS_BOLD, 10 );
$graph->title->Set ( $mytitle );
$graph->subtitle->SetFont ( FF_COURIER, FS_BOLD, 9 );
// $graph->subtitle->Set('"Time&&Power(W)"');
$graph->subtitle->SetColor ( 'darkred' );
$graph->yaxis->title->Set ( "Power" );
// $graph->xaxis->title->Set("Time (O'clock)");
// $graph->xaxis->scale->SetTimeAlign(MINADJ_10);
$graph->SetMargin ( 40, 10, 60, 110 );
// Add background with 25% mix
$graph->SetBackgroundImage ( 'heat1.jpg', BGIMG_FILLPLOT );
$graph->SetBackgroundImageMix ( 25 );
$graph->xaxis->SetLabelAngle ( 90 );
$graph->xaxis->scale->SetDateFormat ( $format );
// $graph->xaxis->scale->SetTimeAlign( MONTHADJ_1 );
// Create the linear plot
$lineplot = new LinePlot ( $pydata, $pxdata );
$lineplot->SetColor ( "blue" );
$lineplot->SetWeight ( 3 );
$lineplot->SetLegend ( "Produced" );
$graph->legend->Pos ( 0.5, 0.2, 'top', 'center' );
// Add the plot to the graph
$graph->Add ( $lineplot );
// Display the graph
$graph->Stroke ();
?>
```

Appendix 10: The codes of History.php

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset = utf-8" />
</head>
<h1 align="center">Windmill History In this Month</h1>
<table align="center">
                <tr>
                                <td><img src="daily.php?id=MONTH"></td>
                </tr>
</table>
<form action="daily.php?id=FETCH" method="post">
                <table align="center">
                                <tr>
                                                <td>Inverter ID</td>
                                                <td><input type="text" name="num" /></td>
                                </tr>
                                <tr>
                                                <td>Date Start</td>
                                                <td><input type="text" name="start" /></td>
                                </tr>
                                <tr>
                                                <td>Date End</td>
                                                <td><input type="text" name="end" /></td>
                                </tr>
                                <tr>
                                                <td>Example:</td>
                                                <td>(A,2013-4-15,2013-5-15)</td>
                                </tr>
                                <tr>
                                                <td><input type="submit" value="Search" /></td>
                                </tr>
                </table>
</form>
<br />
<a href='status.php'>Return to previous page</a>
</html>
```