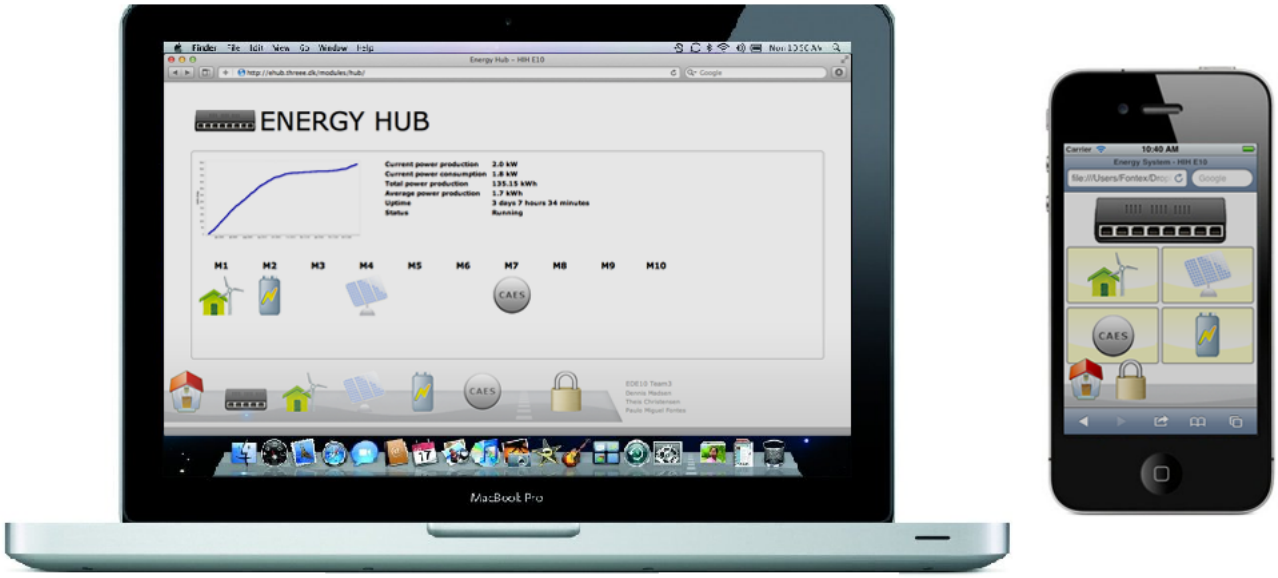
WEB2 - Web Technologies

Project - Energy Hub

Counselor: Martin Olsen



BY E10 - Team3:

Theis Christensen (10691)

Dennis Madsen (90248)

Paulo Fontes (10484)

Table of Content

1 Introduction

This report provides an understanding and documentation of a dynamic web interface application part of an energy hub system. An overview off the energy hub project is given in this introduction, for an easy understanding of the functionalities on the web interface. The report is divided into sections starting with an analysis of the system to be, fowling by the design and implementation for a scaled down prototype version of the energy hub project.

For the development and evaluative procedures, a LAMP(Linux Apache MySQL PHP) web server is set up, this process is explained in the implementation section of this report. The web interface can be accessed on the university network at the address http://10.1.18.223/.

The web interface application includes a great amount of different scripting languages such as server side PHP and client side JavaScript, along with markup languages HTML and XML. The code for each will not be explained extensively, instead the most important functionalities will be filtered and explained in great detail.

MySQL is a requirement in this web interface application, being the most used relational database in a open-source project environment. The integration between MySQL and PHP is easy and well documented making the development process faster. The development of the database is made more interactive and intuitive using phpMyAdmin tool, this tool is installed and explained in the web server implementation sections, and can be accessed at: http://10.1.18.223/phpMyAdmin/ , using the user name **eval** and password **ede10eval** with view only permissions.

An extra web application is developed in PHP only for evaluative propose, it allows the user to view the code of all the files on the web server file structure. By this method the last version of the code is always updated for visualization. The application is called SeeIt and can be accessed at the address: http://10.1.18.223/SeeIt/ . This web application is not explained or documented in this report due to its simplicity and off the topic methods.

The web interface layout was built with the help of different interaction design sessions, used in Interaction Design 1 and Web Technologies 1 courses. With the help of this courses, the web page layout was built in HTML and the user experience was improved from the feedback of the interaction design sessions. The web site graphical layout analysis and implementation is not explained or documented in this report, it can be seen on the appendix on the CD-ROM given.

1.1 Energy Hub Overview

An energy hub is a device able to route the energy to the devices connect to it in the most efficient and effective way, the energy hub is part of an green energy system, the devices connected to the it are: photovoltaic panels, wind turbines, batteries, and a dynamic number os different modules that can me available in future.

The user have to be able to analyse stored measurements, the values are retrieved by the modules to the energy hub and this is responsible to store the data so it can be available to the end user (web interface). A scaled down prototype is made for the energy hub, this is made to give and overview over the system to be in is main functionality: the communication between modules and energy hub, communication between the energy hub and the web server and the ability of switch the power between the connected modules.

2 Analysis

In this section an extensive analysis of the web interface is explained, using visual aids for the easy understanding of the system functionalities.

2.1 Database

For a well constructed database, it has to be consistent, flexible and efficient so no data is lost or repeated when data is saved. The analysis of the system information given is fundamental for the overview and development of the database structure.

The following use cases are retrieved from the pre-project of the energy hub project. (Project 3 found in appendix):

Use Cases

- Jan is looking at the web interface for the energy hub. From where he can see the status for each modules connected to it in a graphically way.
- A new energy module is connected to the hub, Jan opens the web interface for the energy hub, he logs in the administrator section of the system to start, stop or see a more detailed overview of each module.
- Jan arrives at the university in the morning and an email was send to him reporting a failure in the green energy system, he login to the administrator web interface, and he can see what the problem might be, and if it's possible to solve it directly on the interface.

From this use cases given by the customer the relevant information for the database design is filtered:

- The status of each module have to be shown.
- A detailed overview of each module such as current production (Voltage, Current, Power, etc.), efficiency of the module, etc.
- Errors have to be reported to the system administrator / maintainer.

In a deeper analysis of the information system for the power energy hub, some technical data is needed to be stored such as:

- Each module have is unique id, this is similar to the MAC address, this way when a module is connected is possible to check if the module have been connected before, so the data can be stored for the same module instead of instantiate a new module which can generate ambiguous data.
- Three types of modules are defined: input, output and bidirectional. This are seen from the power hub point of view where inputs are producers (solar panels, wind turbine, ...), output are consumers (Inverter, 30V output socket) and bidirectional (C.A.E.S, batteries, ...).

Data sets From the system information analysis a basic tabular structure is designed for a better understanding and low level overview of the database, this is called data set structure.

TYPE(ID_TYPE, NAME);

ID_TYPE	Auto increment integer, a new id number is generated when a different type is needed to the system.
NAME	Describe the type name for example: Input, output, bidirectional, etc.

STATUS(ID_STATUS, NAME);

ID_STATUS	Auto increment integer, a new id number is generated when a different status is needed to the system.
NAME	Describe the status name for example: Running, stopped, warning, etc.

MODULES(ID_MODULE, ID_TYPE, NAME, HUB_PORT);

ID_MODULE	Module unique ID, this id as primary key ensures that no module with is repeated in the database.
ID_TYPE	This is a foreigner key for the table type, this way if some other type of module is needed it can be dynamical add.
NAME	The name of the module for example, Solar Panel, wind turbine, battery.
HUB_PORT	Actual connected port for this module

As a requirement, the database have to store the measurement from different sensors. This is stored in a the table MEASUREMENTS.

MEASUREMENTS(ID_MEASURE, ID_SENSOR, DATE_TIME, HUB_PORT, VALUE)

ID_MEASURE	Measurement id, auto increment field.
ID_SENSOR	This is a foreigner key for the table sensor, this associated the value retrieved to the correspondent sensor.
DATE_TIME	Date and time of the measurement is saved so it can be plotted or in case of a lower efficiency a detailed history can analysed.
HUB_PORT	Actual connected port for this module
VALUE	Measurement retrieved by the energy hub.

Logs are a simplified method of keeping track about what is happening in the system, this is one of the most fundamental functionalities of the system.

LOGS(ID_LOG, ID_MODULE, DATE_TIME, ID_STATUS, ID_USER, ID_ERROR);

ID_LOG	Auto increment integer, a new id number is generated every time a measurement is add.
ID_MODULE	This is a foreigner key for the table modules, this allow the system to know from which module correspond the measurement .
DATE_TIME	Date and time of the log is saved, in case of malfunction it will help with a time line.
ID_STATUS	The new status of the module that was changed.
ID_USER	Foreigner key for the table users, this will allow the system to know which user made the change.
ID_ERROR	Foreigner key for the table error, if any error occurs it will be stored.

For error handling situations a table ERRORS is created this way the administrator can have more control and act in case some mall function of the system.

ERRORS(ID_ERROR, NAME);

ID_ERROR	Auto increment integer, a new id number is generated when a different unit is needed to the system.
NAME	Predefined error description.

Measurements are add to the database constantly, which in a non-stop system database size could be a problem, to avoid this situation a MERGES data set is created. This will store an average of values between a time span for each sensor, allowing the customer either to erase the values from the log or export them out from the database.

MERGERS(ID_MERGE, DATE_FROM, DATE_TO, ID_SENSOR, VALUE);

ID_MERGE	Auto increment integer, a new id number is generated every time an wrap is needed.
ID_SENSOR	This is a foreigner key for the table sensors, this allow the system to know from which sensor correspond the values .
DATE_FROM	Date and time of the time span start.
DATE_TO	Date and time of the time span end.
VALUE	Average measurements value.

SENSOR(ID_SENSOR, ID_MODULE, ID_UNITS);

ID_SENSOR	Auto increment integer, a new id number is generated when a different sensor is needed to the system.
ID_MODULE	This is a foreigner key for the table sensors, this allow the system to know from which module correspond the sensor, being a primary key with the id_sensor, this way each sensor correspond to only one module .
ID_UNITS	This is a foreigner key for the table units, this allow the system to know which units the sensor is measuring.

UNITS(ID_UNIT, NAME);

ID_UNIT	Auto increment integer, a new id number is generated when a different unit is needed to the system.
NAME	Describe the unit name for example: A, V, deg ,m/s,etc. (Ampere, Volt, Degrees, Velocity)

To increase security, an USERS table is added associated with different PRIVILEGES for different users. A user root is created leaving the system able to manage multiple users with different privileges if needed.

USERS(ID_USER, NAME, PASS, EMAIL, ID_PRIV);

ID_USER	Auto increment integer, a new id number is generated when a different privilege is needed to the system.
NAME	Username credential.
PASS	User password (Not encrypted in this version).
EMAIL	Email to which warnings are send.
ID_PRIV	Foreigner key for the table privileges.

PRIVILEGES(ID_PRIV, NAME);

ID_UNIT	Auto increment integer, a new id number is generated when a different privilege is needed to the system.
NAME	Describe the user privileges.

For the communication process between the web interface and the energy hub, the ip address of the device need to be always accessible this way a table DEVICES is created in which the last known IP address of the

device is stored.

DEVICES(ID_DEVICE, IP);

ID_DEVICE	Auto increment integer, a new id number is generated when a different ip added.
IP	IP address of the device.

At this point, in an abstract way, the database can initialise a new module or identify if the module where connected before, change the status for each module and add new measurements for each sensor.

Data Model Data model is a high-level overview of the database structure. At this stage data sets are translated to a logic structure with the relationship between tables.

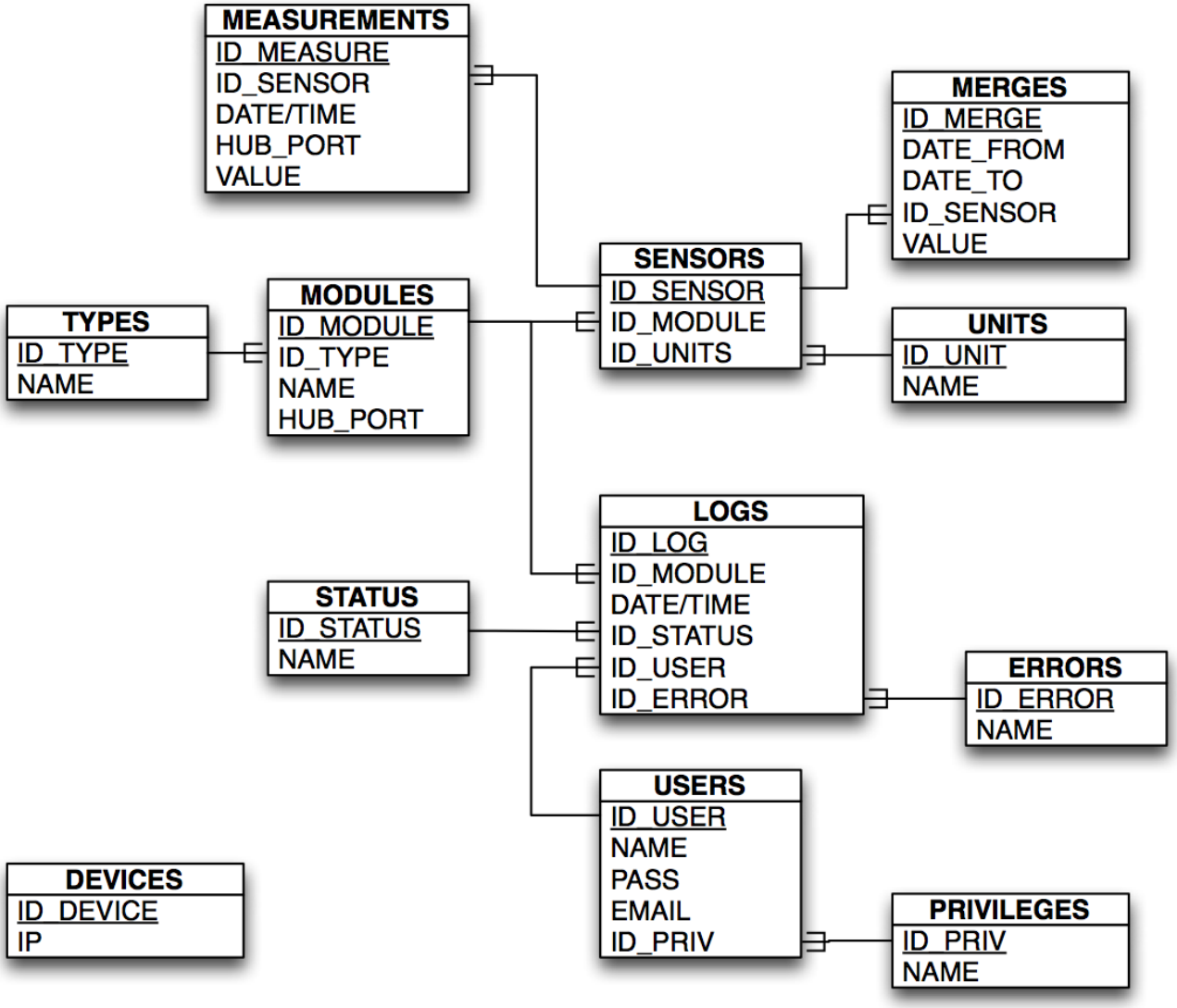


Figure 2.1: Database relational overview (Data Model)

2.2 File Structure

The file structure defines how files are grouped on a system, in a web server architecture the files are stored in the root directory defined by the web server (Apache) configuration. This section describes the analysis of a file structure that allows the scalability of the system.

Overview New modules are added are removed from the energy hub as needed. Some modules have different functionalities that have to be taken in consideration when defining the file structure.

In the root directory of the web server the main page is the entrance to the web interface navigation, from here the user can see the modules connected to the energy hub, and some values about the system such as the efficiency of the green energy system. From the main page the user is able to navigate to the different modules page.

To keep the system dynamic, and flexible the modules page are included inside a predefined layout using PHP. This will keep the graphic layout and navigation system in each page consistent and further improvements for the user experience can be made.

The proposed file structure can be seen bellow:

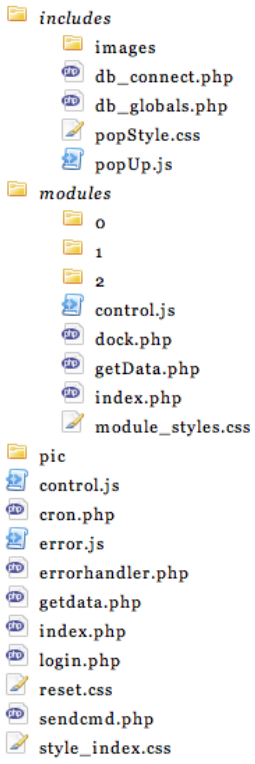


Figure 2.2: Proposed File structure

A short description for each script listed above can be seen in this list:

- includes\images\- Contains images related with the system functionalities.
- includes\db_connect.php - Handle the connection to the MySQL database.
- includes\db_globals.php - Includes all the global variables with the credentials for the database.

- includes\popStyle.css - Popup login style definition.
- includes\popUp.js - Client side script to handle the user login.
- modules\[0-2] - Modules pages, the numbers are associated with the modules ids, being 0 the hub, 1 the battery and 2 the photovoltaic panels.
- modules\control.js - Client side script, makes background requests to a PHP script, this script is part of the user experience optimization.
- modules\dock.php - PHP script contains the navigation system for the web interface.
- modules\getdata.php - PHP script called in background by the client side script running recursively control.js, this script retrieves the system status and data form database in real time, is part of the user experience optimization.
- modules\index.php - This page includes each module page structure.
- modules\module_styles.css - Graphical layout definition.
- control.js - Client side script, makes background requests to a PHP script and handle the connections of new modules, this script is part of the user experience optimization.
- cron.php - Cron jobs are running by the server in a predefined time, in this case the cron.php at the web server root will point to the cron jobs inside each module folder.
- error.js - Client side script, make background requests to a PHP script part of the error handling system.
- errorhandler.php - PHP script called in background by the client side script running recursively, is part of the error handling system.
- getdata.php - PHP script called in background by the client side script running recursively control.js, this script retrieves the modules connected to the system in real time. Is part of the user experience optimization.
- index.php - First page of the web interface.
- login.php - This PHP script is called in background by the pop up login client side script.
- reset.css - Layout definition of the web interface.
- sendcmd.php - Webservice to send commands to the desired module/enegy hub.
- style_index.css - Layout definition of the web interface.

Due to the size of such system and the few time for the development, the project was scaled down so a high fidelity prototype where the system to be can be implemented and the main functionalities of the system tested.

2.3 Communication

The web interface is the connection between the user and the system, in this interface the user have to be able to see the data retrieved by the modules and at the same time be able to send commands to the energy hub/modules to do predefined tasks.

To retrieve the measurements from the modules, an application running at the energy hub translate the data retrieved from the module through PLC to a URL request at the web server. In the server side a script is implemented to translate this request to a SQL query that saves the measurement to the database. It is important to define which data the server side script will expect from the energy hub request. Since each module can have more than one sensor is important to retrieve the sensor id, the measured value and the energy hub port that this module is connected to.

The communication between the energy hub and the web server isn't exclusive for adding new measurements to the database, this communication is also used to update the database regarding the status of the modules in case of new connection, disconnection, a malfunction or stopped for security reasons. The data is saved in the table logs and shown to the user.

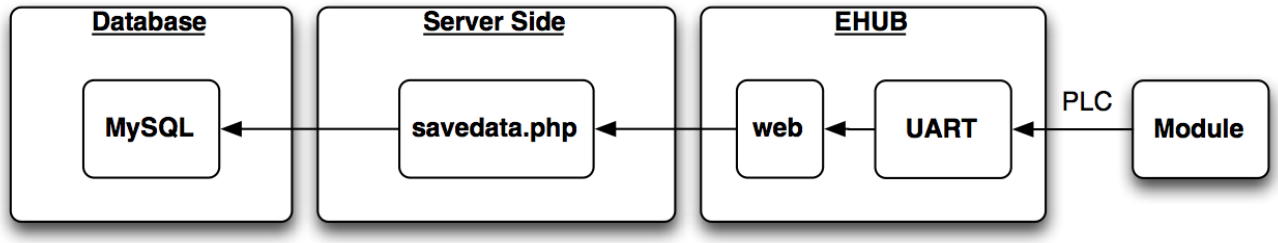


Figure 2.3: Save data to the database

As defined in the Project 3 (in appendix) requirements the system have to give the possibility for the administrator to control the modules and the energy hub. As such the web server have to stablish a connection to the system so commands can be send. A feedback of success or not is always given to the user since the command to be send can be of great importance for the well function of the system.

The flow of communication from the user until the final destination in this case the module or the energy hub.

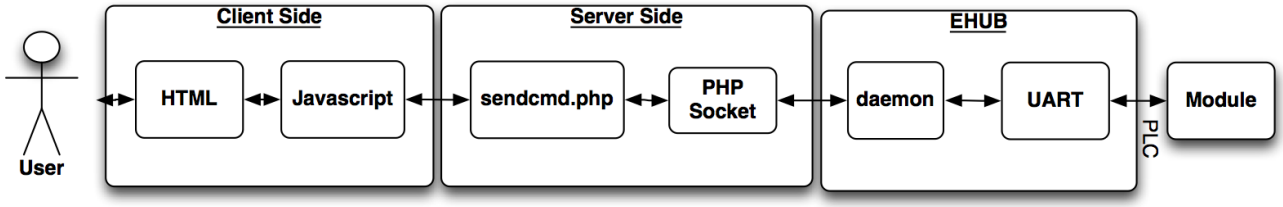


Figure 2.4: Send a command to a module or energy hub

A background application running at the energy hub, ensure that after a reboot, the IP address assigned by DHCP to the energy hub is saved in the database, this way commands can be send through the web interface. The system becomes more flexible in case of internal network changes.

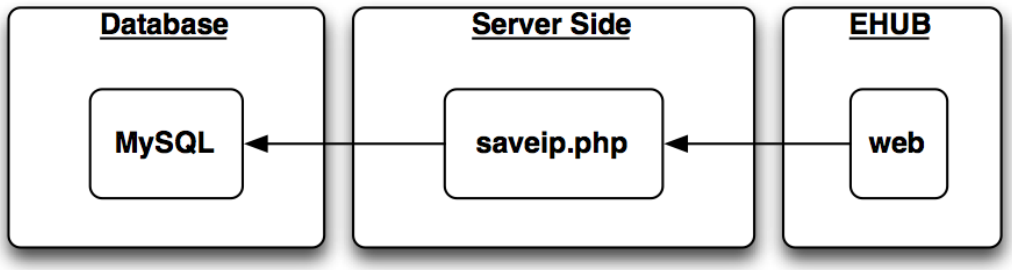


Figure 2.5: Save the IP address of the Embedded device

2.4 Error Handling

The web server have to be able to handle errors such the lost of communication with the energy hub or database connection problems. For a reliable system, a script running on the client side using JavaScript have to be

implemented, this way the web interface can recursively test the communication to the energy hub and database.

Is necessary to define the web interface response for both situations database and energy hub connection lost. If the connection is lost to the device the user is able to see the data stored in the database, so the system is not blocked, but a warning is shown to the user/administrator, do the correspondent debug can be done. In case the connection to the database is lost, the user interface is blocked with an warning, the energy hub will continue working, since it doesn't affected the correct work of the system to route the energy, in this situation all the data retrieved by the modules will be lost.

Since a script is running recursively at the client side, when the communications are re-establish to the energy hub or database the user will be able to use the system normally, this is a great improvement in the user experience and functionalities as an web application.

3 Implementation

This section explains the implementation process of the energy system user interface

3.1 Web server set up

3.2 Database

3.3 Communication

3.4 Error Handling

3.5 Login

4 Discussion

5 Conclusion

6 References

7 Appendixes