Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
In memory of my two grandfathers
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Summary

This report has two aims.

First aim is to evaluate the effect of installing a home energy monitoring system on the electrical consumption of inhabitants of a Smart City. Numerous literature urges for the need of more informative energy billing; historic, extra information or comparative feedback. This thesis conducts a further evaluation in scope of further validating these findings.

Second aim is to compare deployment and performance of Open vs commercial energy monitoring systems. Hence Open Energy Monitor which is as of the date of today the most feature-rich Open option, was chosen and some modifications necessary for the research scope added.

Testing of both aims was conducted as part of the European CIVIS project in Stockholm. 10 houses in Stockholm’s Fårdala district were used as a test base. Due to logistical and issues related to Open Energy Monitor, Smappee was used as replacement unit.

Data obtained with Smappee over the month of July 2015 is analysed in order to validate the assumption that energy monitoring systems have a positive effect on consumption. Further test results can be obtained from the findings of CIVIS.

As for the second aim, there were indeed a few complications using Open Energy Monitor. The houses tracked in Stockholm are supplied by a 3-phase network and Open Energy Monitor as such was incapable of offering correct measurements (1-phase works with no issues). The necessary fix as well as few other modifications to Open Energy Monitor, including a multi-user environment are also presented in this report.

From all the experiences throughout this project, the resulting modified Open Energy Monitor system is more suitable for deployment in user’s homes.
Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT.

Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus células a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Contents

Summary v

1 Introduction 1
  1.1 Energy Monitoring ............................... 1
    1.1.1 Why Energy Monitoring in a Smart City ...... 1
    1.1.2 Existing Communication Mechanisms (of energy information to consumer) ....... 3
    1.1.3 Existing Monitoring Systems .................. 3
      1.1.3.1 Commercial vs Open ...................... 4
    1.1.4 Motivation Behind Using Open System .......... 6
      1.1.4.1 Projects Running on Open Systems ....... 6
  1.2 Open Energy Monitor ............................ 6
    1.2.1 Hardware: emonTX & emonBase ............... 7
      1.2.1.1 Hardware Setup .......................... 7
      1.2.1.2 Arduino Software ....................... 8
      1.2.1.3 emonTX Arduino Sketches ................. 9
      1.2.1.4 NanodeRF Arduino Sketches ............... 11
      1.2.1.5 OEM Raspberry Pi Code .................. 11
    1.2.2 Software: emonCMS .......................... 12
      1.2.2.1 Content Structure ....................... 12
      1.2.2.2 MVC Architecture ....................... 13
      1.2.2.3 Modules .............................. 14
  1.3 CIVIS Project ................................. 16

2 Theory of Measurements 19
  2.1 General ....................................... 19
  2.2 AC Power ................................... 19
    2.2.1 Active, Reactive, Real & Apparent Power .... 20
### Implementation of Measurements

#### 3.1 Lab: 1-phase
- 3.1.1 Hardware Setup
  - 3.1.1.1 Calibration
- 3.1.2 Code Setup
  - 3.1.2.1 Arduino Sketches
  - 3.1.2.2 NanodeRF Fix
  - 3.1.2.3 Raspberry Pi
  - 3.1.2.4 emonCMS

#### 3.2 CIVIS: 3-phase
- 3.2.1 Setup
  - 3.2.1.1 Hardware Setup
  - 3.2.1.2 Code Setup
- 3.2.2 Defective Measurements
- 3.2.3 3-phase Fix
  - 3.2.3.1 Test Setup
  - 3.2.3.2 Logic

### System

#### 4.1 Motivation

#### 4.2 MySQLFiWa
- 4.2.1 Motivation
- 4.2.2 Functionality
- 4.2.3 Code Structure
- 4.2.4 Setup

#### 4.3 MySQLFiSum
4.3.1 Setup ......................................................... 47
4.4 Inter-user averaging & min-max ........................................ 47
4.4.1 Setup ..................................................... 48
4.5 NanodeRF Memory Backup ........................................... 48
4.5.1 Functionality .................................................. 49
4.5.1.1 Overview .............................................. 49
4.5.1.2 Arduino Sketch ......................................... 49
4.5.1.3 emonCMS Modifications .................................. 50
4.6 Other emonCMS Functional Enhancements ......................... 53
4.6.1 Deletion of Input/Feed ......................................... 53
4.6.2 MySQL (Basic Feed Model) Logging ............................. 53
4.6.3 Logging Bug .................................................. 53

5 Pilot Study - Entrepreneurship Chapter .......................... 55
5.1 Motivation ....................................................... 55
5.2 Setup ......................................................... 55
5.2.1 Smappee Replacement ........................................ 56
5.3 Tests .......................................................... 57
5.4 Smappee ......................................................... 57
5.4.1 MySmappee Interface ......................................... 57
5.4.2 Smappee API ................................................ 58
5.4.2.1 Authentication .......................................... 58
5.4.2.2 Data Retrieval ........................................... 60

6 Data Analysis ......................................................... 63
6.1 Circumstance ..................................................... 63
6.2 Raw Results ...................................................... 63
6.3 Analysis ........................................................ 64

7 Other CITCEA Projects Running on Open Energy Monitor ...... 69
7.1 Dish Stirling ...................................................... 69
7.2 Bachelor Project .................................................. 69

8 Budget .............................................................. 71

9 Environmental and Social Impact .................................. 73
10 Conclusion

10.1 Behavioral Impact of Energy Monitoring ........................................ 75
10.2 Evaluation of Open Energy Monitor ............................................. 75
   10.2.1 Drawbacks .......................................................................... 76
   10.2.2 Improvement Possibility ..................................................... 76

Acknowledgements ............................................................................ 77

A  emonTX Shield V2-NanodeRF 3-phase with Backup ......................... 79
B  MySQLFiWa ............................................................................... 93
C  MySQLFiSum ............................................................................ 123
D  Inter User Averaging ..................................................................... 129

Bibliography .................................................................................... 135
List of Figures

1.1 Energy Consumption by Sector in Europe (2009) .............................. 1
1.2 Average Household Energy Savings by Feedback Type .................. 3
1.3 Smart Home Hype Cycle (2012) ..................................................... 4
1.5 Interaction of Different Open Energy Monitor Components ............. 7
1.6 Different emonTX & emonBase Modules of Open Energy Monitor ..... 8
1.7 Raspberry Pi Components Functionality Overview ....................... 12
1.8 emonCMS File Structure ............................................................... 12
1.9 MVC Framework Functional Overview ....................................... 13
1.10 emonCMS Incoming JSON URL Structure .................................. 14
1.11 emonCMS Input processing ......................................................... 15

2.1 AC Waveform ................................................................. 19
2.2 AC Power Cycle .............................................................. 20
2.3 AC Power as a Complex Vector ............................................. 21
2.4 3-phase Wye Configuration ............................................... 22
2.5 3-phase Delta Configuration ................................................ 23

3.1 CITCEA Fridge Tracking Setup ............................................. 26
3.2 Current Sensor Direction Flow ........................................... 28
3.3 Defective Current Sensor .................................................. 28
3.4 Encapsulated DS18B20 Temperature Sensor ............................. 32
3.5 CIVIS Hardware Setup ..................................................... 32
3.6 CITCEA 3-phase Setup .................................................... 34
3.7 Schneider SEPAM Display .................................................. 35

4.1 MySQLFiWa Functionality .................................................. 40
4.2 MySQLFiWa Average Function Functionality ............................ 42
5.1 CIVIS Hardware Setup ............................................. 56
5.2 CIVIS Fuse Box with Installed Current Sensors ............... 56
5.3 MySmappee Interface - W ....................................... 58
5.4 MySmappee Interface - Wh ..................................... 59
5.5 MySmappee Interface - Events .................................. 60
6.1 ‘Total’ and ‘Always On’ Superimposed Consumption of User 4 . . . . . . . 64
6.2 ‘Total’ and ‘Always On’ Superimposed Consumption of User 5 . . . . . . . 64
6.3 ‘Total’ and ‘Always On’ Superimposed Consumption of User 8 . . . . . . . 65
6.4 ‘Total’ and ‘Always On’ Superimposed Consumption of User 9 . . . . . . . 66
6.5 ‘Total’ Consumption Variation of Considered Users .............. 67
6.6 ‘Always On’ Consumption Variation of Considered Users ............ 67
List of Tables

1.1 Summary of Likely Savings Achieved From Different Interventions . . . . . . 2
1.2 Quick Comparison of Hardware Setup for emonTX and emonBase Modules . 8
1.3 RFM Node Allocation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
1.4 Quick Comparison of RFM12 & RFM69 Modules . . . . . . . . . . . . . . . . 11
3.1 Open Energy Monitor Setups . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
3.2 Lab 1-phase Calibration Values . . . . . . . . . . . . . . . . . . . . . . . . . . 26
3.3 Burden Resistance Values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.4 3-phase Defective Measurement Data Sample . . . . . . . . . . . . . . . . . 33
3.5 3-phase Fix Measurement Sample - Open Energy Monitor . . . . . . . . . . 37
3.6 3-phase Fix Measurement Sample - SEPAM . . . . . . . . . . . . . . . . . . 37
3.7 3-phase Original Measurement Sample - Open Energy Monitor . . . . . . . 37
6.1 Weekly Average Wh Values for Testing Period . . . . . . . . . . . . . . . . . 63
8.1 Budget Breakdown of Project . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Chapter 1

Introduction

1.1 Energy Monitoring

1.1.1 Why Energy Monitoring in a Smart City

Increasing CO\textsubscript{2} levels and energy prices has made the topic of energy efficiency gain significant importance in today’s society. With abnormal meteorological incidents on the rise, the unwanted effects of Global Warming have started to be understood by society. Hence the interest of energy efficiency is not solely financial, but instead a challenge unequivocally benefiting the well-being of today’s society.

Looking further into the topic of urban energy efficiency, it can be noted that 40% of energy consumption in Europe is accountable by buildings. The majority of the remaining accounts for transport and industry. [1] A full breakdown is shown in figure 1.1.

![Figure 1.1: Energy consumption by sector in Europe (2009)](image)

Hence within the context of Smart Cities, increasing the energy efficiency of buildings is essential. A starting step is by installing an energy monitoring system. This due to the fact that a major change to the building’s infrastructure is not required and instead an improved overview of the building’s energy situation can be easily obtained. As Sir William Thomson, Lord Kelvin once said: “If you cannot measure it, you cannot improve it”.

Christophe Meyers
The outcome of an efficient energy monitoring system is twofold. Firstly those areas addressable by a simple behavior change are identified. Though this may sound trivial, it must be put forward that “up to 20% of energy savings can be achieved through different measures targeting consumer behaviour”. [2] A more detailed breakdown is shown in table 1.1.

<table>
<thead>
<tr>
<th>Intervention</th>
<th>Range of energy savings [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>5 - 15</td>
</tr>
<tr>
<td>Direct feedback (including smart meters)</td>
<td>5 - 15</td>
</tr>
<tr>
<td>Indirect feedback (e.g enhanced billing)</td>
<td>2 - 10</td>
</tr>
<tr>
<td>Feedback and target setting</td>
<td>5 - 15</td>
</tr>
<tr>
<td>Energy audits</td>
<td>5 - 20</td>
</tr>
<tr>
<td>Community-based initiatives</td>
<td>5 - 20</td>
</tr>
<tr>
<td>Combination interventions (of more than one)</td>
<td>5 - 20</td>
</tr>
</tbody>
</table>

At a later stage, the additional changes which cannot be fixed by a behavioral change can be efficiently addressed. That way the given allocated energy saving budget can be efficiently used. As an example for this next step, one can quote UPC’s Energy Program Sirena, where over €1 million is saved on an annual basis. [3] It must be highlighted that this represents a significant 18% of the on average yearly €5.5 million energy bill. By using an efficient energy monitoring system, quick identification of energy loss in the university’s infrastructure was made possible. However along the fact that there was a financial incentive, Didac Ferrer, head of the sustainability management office at the university emphasises that “The trick is to focus on people first, then technology. It is important to engage people from the beginning of a project and to keep them involved throughout”. Hence coming back to the previous paragraph, is crucial that the end user is educated in addition to the technology installed.

Numerous Smart City projects have shown that at the center of the challenge is the end-user or ‘smartcitizen’, which can be defined as follows: “Smartcitizen is a connected, aware and educated inhabitant of tomorrow’s so called Smart Cities”. This despite numerous technically efficient solutions available on the market. In today’s society numerous daily life examples show that there are two categories of city inhabitants: those who master the technology and can relate to what a Smart City is and those who unfortunately lack both the background and time to be among the prior category.

Another point to consider is that while on one hand understanding the general challenges of a Smart City, the smartcitizen needs to feel comfortable when it comes to the controversial topic of data-privacy in a Smart City. Open-Data released by the city’s administration, the possibility of being tracked with numerous smartphone apps requesting the user’s location as well the installation of smartmeters in one’s own private accommodation. One way of solving this point depends on communication. What and how to communicate to the smartcitizen, for them to feel better involved in a Smart City.

Closing the problem scope defined above, allows to introduce the topic of efficient home energy monitoring in a Smart City. Efficient implies the most effective mechanism to communicate to the smartcitizen their home energy situation. Installing such a system is trivial, but educating the end-user to make use of that data is an additional challenge.
1.1.2 Existing Communication Mechanisms (of energy information to consumer)

Over the past years, studies trying to answer this challenge of efficient communication have been carried out. Communication mechanism methods include working with the energy bills, displays and energy monitoring dashboard layout, as well as expressing the electric consumption in other equivalents. While numerous experiments validating the topic of efficient communication have been conducted since the 1970s, an extensive list will not be presented in this report. Instead the most relevant findings for the scope of this research will be briefly presented and if interested lists can be found in [2], [4] and [5].

Having mentioned different communication methods, a comparison overview with the associated energy savings is shown in figure 1.2. Clearly the advantage of real-time detailed feedback, i.e down to the appliance level is shown.

![Average Household Electricity Savings (4-12%) by Feedback Type](image)

**Figure 1.2:** Average household energy savings by feedback type [4]

A different review study conducted by the University of Oxford’s Environmental Change Institute [5], showed energy bills “to be inadequate for decision making, lacking the detail which would make sense of the bill”. Also having compared several studies, an interesting outcome was that: “Informative billing initiatives in Norway showed how customers appreciated improved accuracy and extra information (historic and comparative feedback, a guide to which end-uses were the highest consuming), began to read their bills more frequently and with more understanding, and began to alter their behaviour”. [6]

The above mentioned outcome portrays that information alone is senseless and as was brought forward in the previous subsection, educating the smartcitizen end-user is key for obtaining the desired results when installing a home energy monitoring system.

1.1.3 Existing Monitoring Systems

Home energy monitoring systems can be traced back to 2008, when the first start-ups entered this new market. [7] Looking more specifically at a Smart Home Hype Cycle from 2012 (fig.
it can be deduced that home energy monitoring is currently in the phase of *Trough of Disillusionment*.

![2014 SMART HOME HYPE CYCLE](image)

In addition utilities are increasingly teaming up with home energy monitoring system providers. Independent or utility sponsored, the market is forecast to grow, as illustrated in figure 1.4.

1.1.3.1 Commercial vs Open

The market situation introduced, as for an increasing number of modern products there are both commercial and open solutions for home energy monitoring.

Some European companies in the home energy monitoring sector include:

- C3 Energy
- Eliq
- Enerbyte
- Onzo
- Smappee
- Watty
While this report has no interest in promoting a particular commercial solution, some functionality overview will be given for Smappee, as it was used in the CIVIS project (section 1.3). Smappee is out of the box ready appliance recognition capable home energy monitoring system, integrated with a smartphone app for remote monitoring and control. Installation does not require any technical knowledge, as even in the case of a 3-phase installation, differentiating current sensors and input ports is irrelevant. Setup is instant via the smartphone app.

On the Open side, the list is composed of.

- EnerJar
- Mazzini
- Open Energy Monitor
- PowerBox
- Simple power monitor
- Tweet-a-watt

Most promising solution, in terms of features and user-friendliness as of today is Open Energy Monitor. This Open Source project that begun in 2009, is based on the Arduino and Raspberry Pi as hardware and uses PHP on the server software for processing and viewing the measured energy data. Appliance recognition is not possible, mainly due to hardware limitations of the Arduino base.\textsuperscript{1} A more detailed overview is given in the next section 1.2.

The market of home energy monitoring systems illustrated, it can be seen that unfortunately Open Energy Monitor is a lacking some functionality in comparison to commercial solutions, as of today. This can be explained by the fact that Open Energy Monitor has a more

\textsuperscript{1}New \texttt{emonPi} option based on RaspberryPi, could support computations for appliance recognition

---

Figure 1.4: Home energy monitoring market forecast (2013-2017) [7]
specific focus on *DoItYourself* enthusiasts. That said some enhancements have been added throughout this thesis (chapters 3 & 4) and a more detailed feature comparison will be covered in the upcoming section 1.2.

### 1.1.4 Motivation Behind Using Open System

It was emphasised in subsection 1.1.1, that educating the *smartcitizen* is crucial in order to achieve the desired energy savings when installing a home energy monitoring system. The best way of achieving such a result is if all included parties have equal privileges to the given project, enabling progress through collaboration based on transparency. Hence even though there are a multitude of more feature-rich commercial options, an Open home energy monitoring system was chosen during this thesis. Also by working with an Open system, exact modifications to fulfill the requirements of the given research project is possible. That was indeed the case during this thesis, as covered later in chapters 3 & 4. Another point was that by adding Open Energy Monitor to the CIVIS project, a further comparison between Open and commercial options was possible.

#### 1.1.4.1 Projects Running on Open Systems

Before going further, examples of a few projects both private and research running on Open Energy Monitor will be analysed for inspiration.

On the private side, a well implemented home energy monitoring system based on a modification of Open Energy Monitor, is *Boredom Projects*. Main interesting point is that the user built a customised Arduino shield in order to be able to connect 10 current sensors, instead of the standard maximum of 4 as offered by Open Energy Monitor. Some server side software modifications, more due to preference were also performed. [9]

On the research side, a current ongoing project is the *Euronet 50/50 Max* project in the southern part of Navarra region of Spain. [10] Although implementation is still in planning phase, it is interesting to note that the philosophy behind is similar to that of this thesis, being that sustainability in all aspects is pursued by using an Open system. The project aims to measure different energy flows including electricity, water, gas and diesel (heating) of schools and share it with the municipality. Cost and CO₂ savings will also be expressed.

### 1.2 Open Energy Monitor

Open Energy Monitor (OEM) as introduced in subsubsection 1.1.3.1, is as of today the most feature-rich and user-friendly Open energy monitoring system. Also as emphasised in subsection 1.1.1, educating the *smartcitizen* and pursuing sustainability in all aspects, was the motivation behind using Open Energy Monitor in this thesis.

Open Energy Monitor is an energy monitoring system, covering all aspects from measuring to processing and displaying of the collected data. An overview situation of how the different components interact is shown in figure 1.5.

Based on figure 1.5, an introduction to the main component modules can be given. The three main are *emonTX*, *emonBase* and *emonCMS*. *emonTX* and *emonBase* perform measurements, while *emonCMS* is the server software. The upcoming two subsections, will cover these respective modules in more detail.
Figure 1.5: Interaction of different Open Energy Monitor components [11]

emonGLCD is basically an LCD display add-on module. Looking at the project’s webpage ([11]), one can see that there are also emonTH emonPi modules. The emonTH is basically a temperature and humidity measuring node, while the emonPi is a new all-in-one inclusive solution (measuring and connecting to remote or running local server). These modules were not used for research and hence will not be further mentioned.

1.2.1 Hardware: emonTX & emonBase

The hardware modules can be split into two categories. The emonTX modules performing the measurements, includes emonTx V3 and emonTx Shield V2. Power, voltage, temperature and humidity measurements are possible with these emonTX modules. On the other hand hardware that connects to an external or acts as a local server, includes the NanodeRF and Raspberry Pi. All four modules are shown in figure 1.6. Hardware setup is covered in next subsubsection 1.2.1.1.

Another difference is the way in which the different modules are programmed. Both emonTX as well as the NanodeRF are Arduino compatible. This as the emonTx Shield V2 is basically an Arduino Shield, while designs of the emonTx V3 and NanodeRF are based on that of the Arduino. As for the Raspberry Pi based emonBase, programming is performed via SSH commands.

1.2.1.1 Hardware Setup

Before looking at explaining the code that runs on the emonTX and emonBase hardware modules, physical setup differences will be compared.

A summary of hardware specifications as well as programming connectivity of what was introduced in subsection 1.2.1 above, is presented in table 1.2. Key takeaway is that the emonTX V3 can be powered through the voltage sensor, only requiring one AC socket. However if combining the emonTx Shield V2 on the NanodeRF, two AC sockets are still required, as powering of the NanodeRF is independent from the added shield.
Table 1.2: Quick comparison of hardware setup for *emonTX* and *emonBase* modules [11]

<table>
<thead>
<tr>
<th>Module</th>
<th>Communication</th>
<th>Power Supply</th>
<th>Measurement Inputs</th>
<th>Programming Interface</th>
<th>Programming Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>emonTx Shield V2</em></td>
<td>• standard Arduino pins&lt;br&gt;• RFM12B/RFM69CW</td>
<td>• via connected Arduino&lt;br&gt;• USB (Mini-B) to UART*</td>
<td>• 4 current sensors&lt;br&gt;1 voltage sensor&lt;br&gt;multiple temperature sensors**</td>
<td>• USH-&gt;UART Arduino</td>
<td></td>
</tr>
<tr>
<td><em>emonTx V3</em></td>
<td>• RFM12B/RFM69CW</td>
<td>• 9V AC-AC adapter sensor&lt;br&gt;• Mini-B USB&lt;br&gt;3 x AA Batteries&lt;br&gt;• USB (Mini-B) to UART*</td>
<td>• 4 current sensors&lt;br&gt;1 voltage sensor&lt;br&gt;multiple temperature sensors**</td>
<td>• USH-&gt;UART Arduino</td>
<td></td>
</tr>
<tr>
<td>NanodeRF</td>
<td>• RFM12B/RFM69CW</td>
<td>• Micro-B USB</td>
<td>• 4 current sensors&lt;br&gt;1 voltage sensor&lt;br&gt;multiple temperature sensors**</td>
<td>• USH-&gt;5V FTDI Arduino</td>
<td></td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>• RFM12B/RFM69CW</td>
<td>• Ethernet&lt;br&gt;(Arduino WiFi Shield)</td>
<td>• Micro-B USB</td>
<td>• MicroSD&lt;br&gt;SSH&lt;br&gt;Linux</td>
<td></td>
</tr>
</tbody>
</table>

*only short periods while programming<br>**identification of multiple DS18B20 thermometer via unique serial number

1.2.1.2 Arduino Software

All available pre-written sketches (referred to as firmware) are available from the project’s Github account [12] and details about them is given in upcoming subsections 1.2.1.3 & 1.2.1.4.

Depending on the application, the necessary libraries need to be first downloaded. Detailed instructions on setting up the necessary Arduino environment can be found at [13].

Overview list of the libraries is as follows:

- DallasTemperature
- EmonLib
- EtherCard
- GLCD_ST7565
1.2.1.3 *emonTX* Arduino Sketches

The *emonTX* module firmware sketches offer different functionality.

The oldest *emonTX* module being the *emonTx Shield V2*, only power measurement functionality is offered by default. Possibilities include only current or additionally voltage. Also using 3 current sensors and 1 voltage 3-phase circuit measurements are possible. Additionally direct communication with a *NanodeRF emonBase* is possible, if not using wireless RFM12B/RFM69CW protocol to a remote *emonBase*.

On the other hand as *emonTx V3* is more recent, measurement of additional energy flows is possible. In addition to 1-phase (only current or additionally voltage) and 3-phase power, firmware sketches also can measure heat pump, temperature, humidity and mechanical ventilation heat recovery unit. Additionally direct Serial cable connection to *Raspberry Pi* instead of via wireless RFM12B/RFM69CW protocol is possible.

### EmonLib

*EmonLib* is the library used by all *emonTX* module firmware sketches, for 1-phase power measurements. As mentioned above either only current or additionally voltage readings are taken as input, offering two different computation algorithms. Brief overview of *EmonLib*’s structure will be given, while covering the two separate algorithm cases.

An Arduino library is basically composed of a header file (.h extension) and source file (.cpp extension). The header file is where the program structure is defined, whereas the source file includes the program functionality.

*EmonLib*’s header file apart from defining variables, sets the resolution of analog-to-digital conversion, by checking microcontroller type. This as the Arduino Due is the unique Arduino running on more powerful 32-bit microcontroller, instead of standard 8-bit as for other Arduino boards. Default is 10 bits, while 12 bits will be set in case of Arduino Due.

*EmonLib*’s source file begins by setting the relevant voltage and current parameters, including defined calibration, as well as offset value based on fixed analog-to-digital conversion resolution. Reading the source file carefully one will notice two sets of functions for voltage and current, notably *voltage* and *voltageTX* for example of voltage. The *voltage* function is called by sketches of both *emonTx Shield V2* and *emonTx V3*. *voltageTX* is called by sketches of older designs.

Follow *calcVI* and *calcIrms* functions. *calcVI* requires both current and voltage sensors to be connected, whereas *calcIrms* relies on a predefined constant voltage input value.

### calcVI

At the function start, initial voltage reading is taken multiple times, until a value close to zero is acquired.

*calcVI* takes two input parameters; ‘crossings’ and ‘timeout’. Both are loop limitation parameters, in that a fixed number of zero-crossings of the sine wave or a duration limit terminate the following while loop based sample acquisition. While respecting the loop
limit parameters, filtered voltage and current readings are taken. Filtered implies DC offset adjusted analog to digital converted sample. Square of voltage and current values are taken for RMS, while phase shifted voltage is computed for instantaneous power computation. A continuous sum of these three variables is updated.

Once the while loop is executed, after applying calibration, the computed sums can be divided by effective number of samples, outputting Vrms, Irms and real power. Apparent power is computed from Vrms and Irms. Power factor can also then be obtained.

calcIrms

calcIrms is a simplified algorithm. Firstly initial current reading does not need to be close to zero, as number of sine wave zero-crossings is not considered. Instead a predefined number of filtered current samples is taken and the overall sum of the squared value is computed in a for loop.

Once the for loop executed, after applying calibration, the computed sum can be divided by effective number of samples, outputting Irms. Then in the calling Sketch, by multiplying by a predefined fixed voltage value power is obtained.

RFM12B/RFM69CW

With the idea that the emonTX will not necessarily be at the same location as the emonBase, wireless communication in between is implemented by RFM12B/RFM69CW. This is also useful is measurements at multiple locations are performed, allowing all data to be collected by one emonBase. This is performed by using different nodeID parameters. Different RFM12B/RFM69CW nodes on a given network, as recommended by Open Energy Monitor is shown in table 1.3. Also if several RFM12B/RFM69CW networks of different users were to coexist, they can be differentiated by a Network Group parameter.

Table 1.3: RFM node allocation [11]

<table>
<thead>
<tr>
<th>node ID</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Special allocation in JeeLib RFM12 driver - reserved for OOK use</td>
</tr>
<tr>
<td>1-4</td>
<td>Control nodes</td>
</tr>
<tr>
<td>5-10</td>
<td>Energy monitoring nodes</td>
</tr>
<tr>
<td>11-14</td>
<td>Un-assigned</td>
</tr>
<tr>
<td>15-16</td>
<td>Base Station &amp; logging nodes</td>
</tr>
<tr>
<td>17-30</td>
<td>Environmental sensing nodes (temperature humidity etc.)</td>
</tr>
<tr>
<td>31</td>
<td>Special allocation in JeeLib RFM12 driver</td>
</tr>
<tr>
<td></td>
<td>Node31 can communicate with nodes on any network group</td>
</tr>
</tbody>
</table>

RFM12B/RFM69CW is a low power wireless protocol, similar to the popular Zigbee standard. Main difference is the lower cost per unit. RFM69CW is the newer version of RFM12B, offering encryption in addition to performance improvements. This is a critical step forward in an application such as home energy monitoring where data privacy is of concern. A more detailed comparison between RFM12B and RFM69CW is shown in table 1.4.

If sending data to an emonBase via RFM12B/RFM69CW, the sending interval has to be specified. Default is 10 seconds, any other time interval is possible. However choosing an interval below 10 seconds is not recommended due to stability hardware issues.
Table 1.4: Quick comparison of RFM12 & RFM69 modules [11]

<table>
<thead>
<tr>
<th></th>
<th>RFM12B</th>
<th>RFM69(C)W</th>
<th>RFM69HW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output power (max)</td>
<td>0 dBm</td>
<td>13 dBm</td>
<td>20 dBm</td>
</tr>
<tr>
<td>TX consumption</td>
<td>23 mA</td>
<td>45 mA</td>
<td>130 mA</td>
</tr>
<tr>
<td>RX consumption</td>
<td>12 mA</td>
<td>16 mA</td>
<td></td>
</tr>
<tr>
<td>Sleep current</td>
<td>0.3 µA</td>
<td>0.1 µA</td>
<td></td>
</tr>
<tr>
<td>FIFO</td>
<td>2 byte RX/TX</td>
<td>66 byte</td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td>none*</td>
<td>128 bit</td>
<td></td>
</tr>
<tr>
<td>Packet handling</td>
<td>none*</td>
<td>preamble, sync, address, CRC</td>
<td></td>
</tr>
<tr>
<td>Open air range**</td>
<td>150 m</td>
<td>250 m</td>
<td>400 m***</td>
</tr>
<tr>
<td>Indoor performance</td>
<td>very good</td>
<td>excellent</td>
<td></td>
</tr>
<tr>
<td>Freq bands</td>
<td>433, 868, 915 MHz, unlicensed ISM bands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample price</td>
<td>$5-10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Achievable in firmware for RFM12B
** Basic testing performed with 915MHz modules in open air with monopole wire antenna 1/4 wavelength
*** Probably 500m+ achievable with non basic antenna, more testing needed

1.2.1.4 NanodeRF Arduino Sketches

The NanodeRF firmware sketches offer functionality of acting as a single or multi-node emon-Base through RFM12B/RFM69CW. Hence interaction with emonTXs and emonGLCD is supported.

Setting correct RFM12B/RFM69CW parameters, including frequency (433Mhz (worldwide), 868Mhz (Europe), 915Mhz (USA & Australia only)), as covered in the previous subsubsection 1.2.1.3 is important.

When forwarding data to an emonCMS server, it is important to note that the NanodeRF can only connect to one server.

1.2.1.5 OEM Raspberry Pi Code

Functionality of the Raspberry Pi is to forward data received via RFM12Pi/RFM69Pi to the emonCMS server. Raspberry Pi pre-configured SD card comes with emonCMS and emonhub preloaded. The functionality structure block of these packages is shown below in figure 1.7.

emonHub has the simple functionality of decoding the received data via RFM12Pi/RFM69Pi and sending it to the emonCMS server. As for NanodeRF, RFM12Pi/RFM69Pi parameters need to be set, in this case via SSH. Note that since the launch of the new emonPi, integrated MQTT functionality and configuration via emonCMS instead of SSH is the new norm.

Raspberry Pi can also be run locally from the Raspberry Pi for simple applications (pre-loaded SD includes emonCMS). Setup and functionality will be covered in upcoming subsection 1.2.2. Most important to note is that local emonCMS install can be turned on/off.
1.2.2 Software: *emonCMS*

*emonCMS* is the software package deployed on a local or external server environment, with purpose of storing and graphically displaying the incoming data from the *emonBase* modules. It is based on the PHP and Javascript programming languages and requires access to a MySQL database. Optional although recommended is Redis in-memory database (required for MySQLFiWa and MySQLSum (chapter 4)) and Apache Log4php.

Package content, software framework and most important modules will be covered. All content is applicable to *emonCMS* version 8.5.

### 1.2.2.1 Content Structure

Main files at the top-level directory are ‘core.php’, ‘index.php’, ‘process_settings.php’, ‘route.php’ and ‘settings.php’. ‘docs’ directory contains some documentation files. ‘Lib’ as the name suggests contains the main CSS and Javascript code library files. ‘scripts’ contains some code for MQTT and input queue processing. Finally and most importantly is the ‘Modules’ folder, containing all the numerous functional addons where mainly modifications to these files is necessary should one want to change functionality of *emonCMS*. The described structure is shown below in figure 1.8.

![emonCMS file structure](image)
Looking more specifically at the ‘Modules’ directory, one can see that each sub-directory matches a certain *emonCMS* functional addon. Each of these relevant sub-directories is most of the time composed of four essential files. ‘⟨name⟩_controller’, ‘⟨name⟩_menu’, ‘⟨name⟩_model’ and ‘⟨name⟩_schema’, where ⟨name⟩ is the Module in question. ‘controller’ decodes requests that are specific to the Module, ‘menu’ adds menu structure the top-bar *emonCMS* menu, ‘model’ is where full Module specific functions are coded and finally ‘schema’ the database structure.

### 1.2.2.2 MVC Architecture

*emonCMS* follows the Model-View-Controller (MVC) framework. Basic functionality on how incoming server requests are handled by such a framework is shown in figure 1.9. Basiclly the returned page view structure and content functionality are separated, centrally controlled by the *controller*. This is an effective methodology where page content dynamically changes, such as for an energy monitoring system web interface.

![Figure 1.9: MVC framework functional overview. Source: SitePoint](image)

The so called requests can either be performed within *emonCMS* by clicking on the relevant command interface or via a JSON URL (as a matter of fact, clicking indirectly creates JSON URLs as well). All data incoming data from the *emonBase* modules is in the form of a JSON URLs. The JSON URL structure is illustrated in figure 1.10.

Structure of example JSON URL from figure 1.10, can be summarised as follows:

```plaintext
.../⟨controller⟩/⟨action⟩.⟨format⟩?⟨extra_parameters⟩
```

After decoding is performed by ‘route.php’, the information is passed onto the relevant Module controller defined by the ⟨controller⟩ parameter. This follows the outlined MVC framework logic. The ⟨action⟩ parameter defines what Module function to be executed.
1.2.2.3 Modules

As highlighted in the previous section, Modules are the building blocks of *emonCMS* as the other top-level folders and files mainly consist of libraries and MVC architecture components.

Pre-installed Modules include:

- **admin**: admin control functionality
- **dashboard**: dashboard component
- **feed**: storage & processing of incoming data
- **input**: incoming data control
- **log**: logging (requires Apache Log4php)
- **time**: time zone offset adjustment
- **user**: log-in/API key management
- **vis**: visualisation (graph components of dashboard)

Among other available add-on Modules, interesting to note are:

- **app**: pre-configured specific dashboards for electric consumption, solar PV and heat-pump production
- **notify**: send notification emails on inactive feeds

Additionally taking advantage of integrated MQTT functionality for more advanced command based interaction with *emonCMS* is interesting.

The most relevant Modules used during this project will be now presented.

Feed Module

The Feed Module has a vital role in *emonCMS*. When incoming data is processed by the Input Module, the ‘processList’ of the ‘input’ MySQL table is checked, to know which Feeds are linked to a specific input. This process is illustrated in figure 1.11.

The data of the ‘processList’ column consists of ‘FeedDataType:FeedId’. FeedDataTypes are as follows:
Figure 1.11: *emonCMS* Input processing

- 0 for Undefined
- 1 for Real Time
- 2 for Daily
- 3 for Histogram

The ‘engine’ column in the ‘feeds’ table defines which Feed Engine to be used for storing the feed data. There are several Engines available as can be seen in the ‘engine’ sub-directory of the ‘feed’ Module folder.

However the main ones to note are:

- MySQLTimeSeries
- PHPFiNa
- PHPFiwa
- PHPTimeSeries

All the PHP based Engines store data in `.dat` data files, not in a database. This as the Open Energy Monitor authors argue that reading from `.dat` files is faster than from MySQL, as well as the fact that if running *emonCMS* from a Raspberry Pi the SD card’s lifespan is extended. Logic is that by using a `.dat` file structure and relying on a fixed interval as for PHPFiNa and PHPFiWa, position of data in file determines the timestamp.

Default option of new *emonCMS* install is to use these PHP based Engines.

Feed data can also be exported in CSV format, should it be used in another application.

**Vis Module**

This Vis Module includes a number of graph to choose from, when displaying the Feed data:

- Bargraph
- Compare (two different Feeds via x-y axis)

---

2 This Engine was forked from previous *emonCMS* version 8.4 and is used for this thesis (chapter 4)
• *Edit* (modify stored data via the graph)

• *Histogram*

• *Multigraph* (multiple plots on same graph)

• *Order* (re-order Feed data by value level)

• *Rawdata*

• *Realtime*

• *Smoothie*

• *Stacked* (stacked bargraph)

• *Threshold*

Many of the Vis Modules are not fully functional. Hence the ones emphasised on during this thesis were Realtime, Rawdata and Multigraph.

**Dashboard Module**

Here the Vis Modules experimented with can be permanently kept for display, as if the Vis page is quit, the visualisation needs to be re-created. Additionally numerous text, containers and widgets can be added to the dashboard page.

This is effectively what is at the center of any energy monitoring system.

### 1.3 CIVIS Project

CIVIS is an EU research initiative, aiming to analyse the social dimension of a Smart City when users have access to their energy consumption data. This mainly through an app based mechanism, as well as community collaborative approach. The project has just begun in March and is scheduled to last 2 years.

As for any major research project, CIVIS is divided into several work package groups. The nine work packages are as follows:

• WP1 - Overall design of the smart energy cities

• WP2 - Energy and ICT System Analysis of Pilot Sites

• WP3 - Enabling SMART social participation

• WP4 - Integration and deployment of the system level ICT systems

• WP5 - Community Participation: Social, Regulatory and Institutional Dimensions

• WP6 - Definition of business models for an emerging social decentralized energy

• WP7 - Testing the energy performance of proposed Smart Grid / ICT installations

• WP8 - Dissemination, awareness and exploitation

• WP9 - Project management
WP4 and WP7, including the participation of KTH are the most relevant to this thesis. Further information about the CIVIS project can be found at [14].

In context of Smart Cities of what has been mentioned about the importance of the smart-citizen, the CIVIS project is a good research initiative aiming to further investigate the role of the end user in a Smart City. Given this context, the research of this Master thesis has been integrated into the CIVIS project in Stockholm.

Building upon the work described in the prior section, this thesis will focus on interaction between the end user and *emonCMS* dashboard layout.
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Chapter 2

Theory of Measurements

2.1 General

In order to understand the implementation of measurements covered in the next chapter 3, a quick overview of 1-phase/3-phase AC power theory will be covered in this chapter. Especially beneficial for the case of 3-phase, as modifications to the 3-phase Arduino firmware sketch were performed (section 3.2).

2.2 AC Power

Alternating Current (AC) implies that current entering a given load, will be both positive and negative over a given period. This is illustrated in figure 2.1, for one period of a 50Hz cycle wave.

![Figure 2.1: AC waveform](image)

The voltage waveform will be of the same shape, different in amplitude. The theory of AC power begins, when the current and voltage waveforms are not overlapping each other,
or in more specific terminology out of phase. Instantaneous power defined as the product of instantaneous current and voltage, the power will have negative in addition to positive portions, as shown in figure 2.2.

![Figure 2.2: AC power cycle](image)

### 2.2.1 Active, Reactive, Real & Apparent Power

That said, the concepts of **Active** and **Reactive** power can be introduced. Active power $P [W]$ refers to the positive part of averaged instantaneous power over a full period, whereas reactive power $Q [var]$ refers to the negative component that effectively returns to the source.

Purely resistive loads only consume active power, whereas reactive power is present for inductive and capacitive loads. Unfortunately most household loads are not purely resistive.

Both active and reactive power present in the household circuit, after all only active power is of use to the end user. Instead of referring to active and reactive power, real and apparent power terminology is more frequent.

The average instantaneous power can be seen as a complex vector (figure 2.3). **Apparent** power $S [VA]$ refers to the magnitude of the complex power vector. Relation with active and reactive power can also be seen in the figure.

**Real** power being the same concept as active power, can be defined as the positive of average instantaneous power over a cycle (equation (2.1)).

$$P = \text{Real}(\frac{1}{T} \sum_{t=0}^{T} V(t)I(t))$$  \hspace{1cm} (2.1)

### 2.2.2 Power Factor

The concepts of real and apparent power introduced, the notion of **Power Factor (P.F)** follows. As was emphasised, one would ideally maximise the real power, minimising the...
reactive power that effectively limits the transfer of real power.

That leads to the notion of power factor which is a simple ration between active power and apparent power (equation (2.2)).

$$P.F = \frac{P}{S} = \cos(\phi)$$ (2.2)

For a resistive load, power factor is evidently 1. For reactive loads, where current is leading the voltage waveform power factor is lagging and $\phi$ is positive. This is case for inductive loads. Opposite applies for a capacitive load.

### 2.2.3 Root Mean Square

Given that the instantaneous voltage and current waveforms are non-constant as is the case for basic DC power, an averaging concept can also be applied to AC waveforms. Concept is based on the statistical concept of quadratic mean.

*Root Mean Square (RMS)* serves this purpose (equation (2.3)).

$$R.M.S = \sqrt{\frac{1}{N} \sum_{n=0}^{N} x_n^2}$$ (2.3)

That said, RMS values of voltage and current can also be used to compute apparent power (equation (2.4)).

$$S = V_{RMS}I_{RMS}$$ (2.4)

### 2.3 1-phase Power

For 1-phase the basic notions defined in the previous section (2.2) can be simply applied to the circuit.
For voltage and current, RMS values through equation (2.3) is of interest. Followed by apparent power via equation (2.4). Then real power from equation (2.1) and finally power factor from equation (2.2).

Given that the *emonTX* hardware modules have 4 current inputs, in theory 4 different loads powered by the same circuit can be simultaneously tracked. The logic of *emonLib* (subsubsection 1.2.1.3) is simply applied to each current input respectively. Then values of Vrms, Irms of each input, as well as real and apparent power can be sent to the *emonCMS* server.

### 2.4 3-phase Power

3-phase systems are widely used, where transmission of larger amounts of instantaneous power is necessary. In addition to the power transferred (3 times power per phase if balanced system), savings of 25% on the required copper for distribution is possible in comparison to the same power being transferred by a 1 phase system. Also as part of the CIVIS project (chapter 5), the houses measured were all 3-phase powered.

All of the introduced concepts in section (2.2) are true for each phase, where voltage is understood to be the voltage between the phase and neutral. However 3 phase systems can be both Delta or Wye, for each case some of the additional differences will be covered.

#### 2.4.1 Wye System

In a 3-phase Wye system (figure 2.4), line-line currents are same as phase (line-neutral) currents. However phase voltages differ from line-line voltages and are given by equation (2.5).

![Y configuration](https://via.placeholder.com/150)

Figure 2.4: 3-phase Wye configuration. *Source: Wikipedia*

\[ V_{\text{line-line}} = \sqrt{3} V_{\text{phase}} \]  \hspace{1cm} (2.5)

Is important to note that line-line voltages lead by 30°phase shift the phase voltages. More specifically equation (2.6) shows the case for each set of line-line voltages, if referring to figure 2.4.
\[ V_{12} = V_1 - V_2 = \sqrt{3} V_{\text{phase}} \angle 30 \]
\[ V_{23} = V_2 - V_3 = \sqrt{3} V_{\text{phase}} \angle -90 \]
\[ V_{31} = V_3 - V_1 = \sqrt{3} V_{\text{phase}} \angle 150 \]  
(2.6)

2.4.2 Delta System

In a 3-phase Delta system (figure 2.5), line-line voltages are same as phase (line-neutral) voltages. However phase currents differ from line-line currents and are given by equation (2.7).

![Delta configuration](image)

Figure 2.5: 3-phase Delta configuration. Source: Wikipedia

\[ I_{\text{phase}} = \frac{I_{\text{line}}}{\sqrt{3}} \]  
(2.7)

It is important to note that line-line currents lag by 30° phase shift the phase currents. More specifically equation (2.8) shows the case for each set of line-line voltages, if referring to figure 2.5.

\[ \begin{cases} 
I_1 = I_{12} - I_{31} = \sqrt{3} I_p \angle -30 \\
I_2 = I_{23} - I_{12} = \sqrt{3} I_p \angle -150 \\
I_3 = I_{31} - I_{23} = \sqrt{3} I_p \angle 90 
\end{cases} \]  
(2.8)

2.4.3 Balanced System

The conversion from line-line variables to phase variables defined for both Wye (subsection 2.4.1) and Delta (subsection 2.4.2), the apparent power per phase can be computed (equation 2.9).

\[ S_{\text{phase}} = V_{\text{phase}} I_{\text{phase}} = V_{\text{line-line}} I_{\text{phase}} \]  
(2.9)

Adding the apparent power over the 3 phases, leads to equation 2.10 for a balanced system.

\[ S_{3\text{-phase}} = 3V_{\text{phase}} I_{\text{phase}}^* \text{ (\(*\) is complex conjugate, as sine wave is complex)} \]  
(2.10)
2.4.4 Balanced vs Unbalanced System

An unbalanced 3-phase system is defined as one where the power on each of the respective phases is unequally divided. When performing energy monitoring, this is unfortunately the case. In that case equation 2.10 unfortunately cannot be applied and instead equation 2.9 applied to each of the phases, the sum is taken.

2.4.5 Application to Open Energy Monitor

When using the emonTX hardware modules, there is only one voltage sensor per module. That said if performing 3-phase measurements with solely one emonTX module, the monitored circuit has to be more or less balanced. This as phase shifted values of the measured voltage on the first phase are applied to the two other phases in the Arduino sketch (section 3.2).

If the monitored system is significantly or more unbalanced, three separate emonTX modules, one-per-phase have to be used.

For the CIVIS deployment, the houses were assumed to be more or less balanced, hence only one emonTX per house was used.
Chapter 3

Implementation of Measurements

3.1 Lab: 1-phase

When initially working with the Open Energy Monitor system, power consumption of CITCEA’s EUETIB Aula Schneider’s fridge was tracked. At the same time, an evaluation of the two systems (table 3.1) on offer by the Open Source project was performed. The two setups offer same functionality, apart that Setup 2 is based on better hardware and hence more precise. This of course comes with an additional purchasing cost.

<table>
<thead>
<tr>
<th>Setup</th>
<th>emonTX</th>
<th>emonBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup 1</td>
<td>emonTX Shield V2</td>
<td>Nanode RF</td>
</tr>
<tr>
<td>Setup 2</td>
<td>emonTX V3</td>
<td>Raspberry Pi</td>
</tr>
</tbody>
</table>

3.1.1 Hardware Setup

Photos of the two options are shown in figure 3.1. The idea last semester being to build foundation work for this thesis, power consumption of solely one simple common household device was tracked in parallel by the two setup options.
Current measurements were performed by a non-invasive clip-on current transformer sensor, as can be seen in figure 3.1. Voltage readings were taken from a 230/9 V AC-AC adapter. Logically requirement is that the tracked device and AC-AC adapter are in phase with each other, or in more practical terms powered from the same circuit branch of the building’s electricity installation. Voltage sensor is not compulsory (table 1.2), but offers more precise readings.

### 3.1.1.1 Calibration

First step was to adjust the calibration constants of the *emonTX* Arduino sketches (sub-subsection 1.2.1.3). The difference between pre-defined and final values is shown in table 3.2.

<table>
<thead>
<tr>
<th></th>
<th>Pre-defined</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Voltage</td>
<td>Current</td>
</tr>
<tr>
<td>Shield V2</td>
<td>260.0</td>
<td>60.606</td>
</tr>
<tr>
<td>V3 (CT 1-3)</td>
<td>260.0</td>
<td>90.9</td>
</tr>
<tr>
<td>V3 (CT 4)</td>
<td>260.0</td>
<td>16.67</td>
</tr>
</tbody>
</table>

Theoretical calibration constants are based on the hardware design of the *emonTX Shield V2* and *emonTX V3*. Current calibration value is based on the burden resistance value and obtained from equation (3.1), using burden resistance values of table 3.3. As for the voltage calibration constant value, it is based on datasheet values of the 230/9 V AC-AC adapter.
Current Calibration = \frac{\text{CT Ratio}}{\text{Burden resistance}} = \frac{100[A]/0.05[A]}{\text{Burden resistance}} \tag{3.1}

Table 3.3: Burden resistance values

<table>
<thead>
<tr>
<th>Burden Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shield V2</td>
</tr>
<tr>
<td>Shield V3 (CT 1-3)</td>
</tr>
<tr>
<td>Shield V3 (CT 4)</td>
</tr>
</tbody>
</table>

Procedure requires a resistive load in order to have a power factor of 1, as well a separate multimeter or oscilloscope for reference measurements.

During the semester project, following equipment was used for calibration:

- 141Ω (3 X 47Ω in series) resistor bank
- PicoScope 4424

For adjusting voltage and current calibration, equation (3.2) was applied. For phase calibration, the power factor reading was tracked. If below 1 the calibration constant value is increased, or decreased if value is larger than 1.

New Calibration Constant = \frac{\text{Multimeter Variable Value}}{\text{emonTX Variable Value}} \tag{3.2}

Some points to pay attention to during the process, include:

- final calibration constants not too different from initial
- negative real power values and single wire only
- phase shift constant within range [0.0,2.0]

Negative real power readings imply that the current sensor is connected the wrong way around. Is important that the printed side of the sensor points in the direction of power flow (figure 3.2).

Non-invasive clip-on current transformer sensor operate through induction. Hence if two wires each with opposite direction flows of equal current are used with the sensor, the effect will be readings of zero as the two wires cancel each other out. Hence is crucial that only a single current carrying wire is measured.

If the phase shift calibration constant is not in the range [0.0,2.0], the possibility of a defective current sensor has to be taken into account. This was actually the case for one of the sensors, due to the fact that the sensor cores were not aligning properly (figure 3.3).
3.1.2 Code Setup

3.1.2.1 Arduino Sketches

The Arduino Sketches used were as follows:

- ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_Voltage’ for emonTX Shield V2 Calibration
- ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_Voltage_NanodeRF’ for emonTX Shield V2-NanodeRF setup
- ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxV3/RFM12B/emonTxV3_RFM12B_DiscreteSampling’ for emonTX V3

Adjusting the calibration constants for emonTX Shield V2 requires changing following section at beginning of code:

```c
if (CT1) ct1.current(1, 60.606);
.if (CT1) ct1.voltage(0, 268.97, 1.7);
```

---

Adjusting the calibration constants for *emonTX V3* requires changing following section at beginning of code:

```c
const float Ical1 = 90.9;
const float Vcal = 276.9;
const float phase_shift = 1.7;
```

Apart from calibration constants adjusted, RFM12B/RFM69CW (table 1.3) are usually adjusted. However the default values were Arduino Sketch kept for communication between *emonTX V3* and *Raspberry Pi*.

For the *NanodeRF* case, *emonCMS* server parameters also need to be set. Changes are set in the following section of the code:

```c
// 1) Set this to the domain name of your hosted emoncms - leave blank if posting to IP address
char website[] PROGMEM = "emoncms.org";

// 2) If your emoncms install is in a subdirectory add details here i.e "/emoncms"
char basedir[] = "";

// 3) Set to your account write apikey
char apikey[] = "YOUR APIKEY";
```

As covered in subsection 1.2.2, a ‘write API key’ is associated with each user account.

### 3.1.2.2 NanodeRF Fix

When using the Arduino sketch ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_Voltage_NanodeRF’, there were some issues establishing a data connection to the *emonCMS* server.

The fix was as follows:

- add ‘const’ to the line ‘const char website[] PROGMEM = “emoncms.org”;’ (line 51)
- changed line 78 to ‘ether.printIp("Netmask: ", ether.mymask);’
- if posting to a static IP server, the following should be added after line 51

```c
boolean use_hisip = false;
static byte hisip[] = {}; // set IP address here
```

Optionally if additional parameters need to sent to the *emonCMS* server, the JSON string built over lines 126-135 can be modified with addition of those extra parameters.

### 3.1.2.3 Raspberry Pi

Two parameters need to be checked in the case of the *Raspberry Pi*.

Firstly RFM12B/RFM69CW parameters need to be matched to those of the *emonTXs*, in this case *emonTX V3*. Procedure to do so is via SSH and is as follows:
• enable write mode of the SD card:
  ```
sudo rpi-rw
  ```

• access emonHub configuration file:
  ```
sudo nano /boot/emonhub.conf
  ```

• update parameters (lines 37-38, 52-54):
  ```
url: http://oem.smartcitizen.se/emoncms
apikey: 0b5af973054d8f85849e83e109cf951
RFM12B frequency (group = 210, frequency = 868 & baseid = 15)
  ```

• disable write mode of the SD card:
  ```
sudo rpi-ro
  ```

Secondly if posting to a remote server, the local emonCMS needs to be disabled:
  ```
sudo nano /usr/bin/locemoncms--disable
  ```

Else if posting to the local Raspberry Pi emonCMS:

• enable SD card write mode:
  ```
sudo rpi-rw
  ```

• set url parameter on line 37 of emonhub.conf to:
  ```
'http://localhost/emoncms'
  ```

### 3.1.2.4 emonCMS

emonCMS was installed on an Ubuntu 14.04 LTS Server.

User accounts created and the write API keys set on both the Nanode Shield V2 Arduino sketch and Raspberry Pi code, data was able to be processed on the emonCMS server.

Emphasis is placed on low-write by the Open Energy Monitor authors, as emonCMS can be run from a Raspberry Pi with SD card. Consequently default option is to store data in `.dat` files as introduced in subsubsection 1.2.2.3. While this is efficient for a pure emonCMS application, when one would need to have access to the raw data whether for data management purposes or sharing with other applications (without needing to rely on the export component of the `.dat` based Feed Engines), a database based storage mechanism is favourable.

While an especially coded more efficient MySQL storage (subsection 4.2) mechanism was added to emonCMS during the thesis, during the semester project a few configuration modifications were made to emonCMS so that the project originally supplied MySQLTimeSeries Feed Engine module could be integrated.

The fix was as follows:

---

2 low-write implies optimising the number of write cycles when storing data

3 Modifications below refer to emonCMS version 8.4 from autumn 2014. Version 8.5 used for the thesis has had quite some restructuring.
• adding the MySQL option to relevant lists in ‘Modules/input/process_model.php’.
  More precisely adding option ‘ENGINE::MYSQL’ to the lists spanning over lines 50-94.
• enabling MySQL under ‘Modules/input/Views/input_view.php’.
  More precisely adding ‘⟨option value=0⟩MySQL TimeSeries(MYSQL)⟨/option⟩’ to the list after line 78. The reason for ‘value=0’ has to do with the matching the array structure under lines 60-70 of ‘Modules/feed/feed_model.php’.

After that fix and getting familiar with the entire Open Energy Monitor framework, preparation work for the CIVIS deployment could begin.

3.2 CIVIS: 3-phase

Open Energy Monitor is supposedly also capable of tracking 3-phase power, as the emonTX hardware allows up to four current sensor inputs. According to the project’s website, as long as the monitored load is reasonably well balanced, measurements with one emonTX unit is possible.

That said 3-phase measurements were performed during deployment of the CIVIS project in Stockholm (chapter 5). Although the obtained results made no sense at start, a fix was performed to the supplied Arduino code (subsection 3.2.3).

3.2.1 Setup

The setup on the emonTX Arduino side will be covered here. A more detailed explanation of the rest of the system as well as details of the pilot study is given in chapters 4 and 5 respectively.

3.2.1.1 Hardware Setup

Due to cost preference, the emonTX Shield V2 combined with NanodeRF hardware option was chosen. In addition to electricity monitoring, temperature was also monitored by the setup.

Temperature measurements were performed with an encapsulated DS18B20 temperature sensor (figure 3.4). The supplied jack plug was not used, the cables directly mounted into the DS18B20 terminal block on the emonTX Shield V2.

It is important to note that the emonTX Shield V2 does not come pre-assembled. Hence soldering of all components was first performed.

Full setup overview is shown in figure 3.5. As the monitoring system was installed in private houses, it was necessary to keep the setup enclosed in a box.

One crucial point to emphasize is that when ordering the emonTX Shield V2, it is recommended to select the ‘Extra Long “stackable” Headers’ even if effectively no additional Arduino Shield will be stacked on top. Else the NanodeRF FTDI port has to be removed, to make room for the voltage sensor input of the emonTX Shield V2. Note that the setup is programmed via the UART port of the emonTX Shield V2, so removal of the NanodeRF FTDI port has no overall effect.

Calibration was performed with the following equipment:
3.2.1.2 Code Setup

The Arduino sketches used were as follows:

- modified version of ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_Voltage_NanodeRF’ (see subsection 3.1.2.2)
• ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_3Phase_Voltage’
• ‘Sketchbook/libraries/Arduino-Temperature-Control-Library/Single’

Basically a new Arduino sketch was made where the 3-phase functionality as well as temperature sensing code was merged with the original 1-phase-emonTX Shield V2-NanodeRF integrated sketch from last semester.

When merging temperature functionality, a few considerations/modifications were taken into account:

- Data bus is plugged into port 4 for emonTX Shield V2 case
- Resolution set to 12 bits for accuracy instead of 9
- Automatic assignment of sensor address, so can function with any replacement sensor

Logic of this functionality can be found in Appendix A. Note that this Sketch includes other functionality described in upcoming subsection 3.2.3, as well as section 4.5.

### 3.2.2 Defective Measurements

During installation phase, it was noted that the power measurement data obtained was faulty. A data sample is shown in table 3.4.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1433517305</td>
<td>9759.94</td>
</tr>
<tr>
<td>1433517315</td>
<td>-12595.5</td>
</tr>
<tr>
<td>1433517325</td>
<td>-53884.3</td>
</tr>
<tr>
<td>1433517335</td>
<td>9128.77</td>
</tr>
<tr>
<td>1433517345</td>
<td>-12567</td>
</tr>
<tr>
<td>1433517355</td>
<td>24017.4</td>
</tr>
<tr>
<td>1433517365</td>
<td>-71618</td>
</tr>
<tr>
<td>1433517375</td>
<td>-0.53</td>
</tr>
<tr>
<td>1433517385</td>
<td>0</td>
</tr>
<tr>
<td>1433517395</td>
<td>-0.59</td>
</tr>
</tbody>
</table>

A thorough check of all integration modifications was performed, as well as order of connected current sensors switched (voltage sensor needs to be on same phase as current sensor 1). Unfortunately a return to Barcelona was on schedule and 3-phase consumption data using Open Energy Monitor was not possible (note that data using Smappee as a replacement is presented in chapter 6). Performing some research of the forum topics on Open Energy Monitor’s website showed that other people are facing the same issue. [15]

### 3.2.3 3-phase Fix

Upon returning to Barcelona, the logic of the basic ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxShield/Shield_CT123_3Phase_Voltage’ Arduino sketch was re-thought.
Logic of the official sketch is to store samples of the measured voltage for phase 1 and then to access these values again at a later stage when applying 120° phase shifts to the other two phases (subsection 2.4.5). Confirming what was discussed by other users [15], indeed the way in which the stored voltage values array is accessed for phases 2 and 3 was the cause of the problem. Following the forum topic’s, the newer hardware ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxV3/RFM/emonTxV3.4/emonTxV3.4_3Phase_Voltage’ Arduino sketch was first consulted. However the issue still persisted and a modified logic was thought of.

3.2.3.1 Test Setup

The 3-phase setup in CITCEA’s EUETIB Aula Schneider was used. Figure 3.6 shows an overview of the load (rows of light bulbs) and control, as well as how current sensors were connected to the different phases of the setup supply.

![Figure 3.6: CITCEA 3-phase setup](image)

Left: Current sensors connection to supply, Right: Load & control

In order to have accurate measurements, calibration of sensors was performed. Procedure followed same procedure as described in subsubsection 3.2.1.1. (Note that after finishing the alternative logic sketch, calibration was re-performed). Following equipment was used:

- Fluke 179 True RMS Multimeter
- Chauvin Arnoux PAC 12 AC-DC current clamp
- DL 1520 Yokogawa 8bits 200Ms/s 150 MHz Digital Oscilloscope
- 212Ω resistive load

CITCEA’s 3-phase setup load is monitored by Schneider Sepam digital protection relays. More specifically 80 Series at the supply point of the entire setup and 40 Series for the circuit branch used by the load. The readings from these units (figure 3.7) was used to compare measurements returned by Open Energy Monitor setup.

For reference of measurements from Schneider Sepam digital protection relays at beginning, as well as voltage from *emonTX Shield V2*, following equipment was used in parallel:

- Tektronix TDS 1002B oscilloscope
• LEM PR30 current probe

It must be noted that the current values displayed by Schneider SEPAM are multiplied by a factor 10. Also the power factor is negative, when it should be positive.

3.2.3.2 Logic

Logic of ‘calcVI3Ph’ from ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxV3/RFM/emonTxV3.4/emonTxV3.4_3Phase_Voltage’ Arduino sketch was analysed, as emonTX V3.4 as not only it was referenced in the related forum topic, but is the newest hardware version and hence the newest emonTX Firmware code.

Functional overview of this official sketch will now be explained. It is important to note that unlike the 1-phase sketches introduced in subsubsection 1.2.1.3, emonLib functionality is integrated into the 3-phase sketch. Line numbers refer to ‘Sketchbook/firmware/OpenEnergyMonitor/emonTxFirmware/emonTxV3/RFM/emonTxV3.4/emonTxV3.4_3Phase_Voltage’. Logic is as follows:

• Calibration constants are declared lines 161 through 184. Different values apply for emonTX V3 and emonTX Shield V2. Also if a fourth current sensor is used (for solar), this can be specified on line 141. PHASE2 and PHASE3 fine-tuning parameters are set lines on 147-149. As in emonLib, analog-to-digital resolution can be specified to 12 bits if using an Arduino Due.

• The Phase3 fine tuning parameter defined on line 149, sets the size of voltage storage array storedV[]. ‘#define BUFFERSIZE (PHASE3+2)’ on line 192 sets the array size on line 395 ‘double storedV[BUFFERSIZE]’ within the ‘calcVI3Ph’ function. The array size is a little oversized to be on the safe side.

• Logic within ‘calcVI3Ph’ function is then
  – Make sure the initial voltage reading has a value close enough to zero. In worst case ‘calcVI3Ph’ will exit, if the defined function input timeout parameter elapsed during this initial process.
  – First filtered voltage value is stored at index=0 of storedV[] array. This as numberOfSamples was initialised to BUFFERSIZE on line 391, where index of array is defined as ‘numberOfSamples%BUFFERSIZE’.
– Every time the initial voltage reading $startV$ is crossed, implies a new cycle. Upon which voltage, current and power sums are reset. As well the power averaging count parameter $numberOfPowerSamples$. Additionally the $crossCount$ is of course incremented.

– Offset and filtering is then applied to current input values. Related sums (voltage and current) are incremented by these new values.

– Before updating power sums, voltage needs to be phase shifted. For phases 2 and 3, delayed samples by an amount of $PHASE2$ and $PHASE3$ are taken. This value is applied a modulo of $BUFFERSIZE$, the effective array size, for instance \((numberOfSamples-PHASE2)\%BUFFERSIZE\). It is believed that this operation is the core of wrong voltage values on phases 2 and 3. As a reminder $BUFFERSIZE$ was defined as \(PHASE3+2\), so taking \('storedV[(numberOfSamples-PHASE3)\%BUFFERSIZE]'\) for the voltage value of phase 3 is questionable. There seems to be a flaw in the way delayed voltage samples are obtained for phases 2 and 3.

– Calculations of final values after loop, follows same logic as that of $emonLib$ for 1-phase measurements.

Alternative logic was to consider if values delayed by 1/3 and 2/3 of the voltage array size, were effectively being accessed. Rethinking, modifications to ‘calcV13Ph’ function are as follows:

• Size of $storedV[]$ is set to size 30, just to be on safe side as usually 24-26 voltage samples per cycle are stored.

• under section B ‘index="numberOfSamples\%samplesPerCycle"’ is always used as index of $storedV[]$ for storing new values

• Then under section C once ‘crossCount=0’ is reached, last stored value is moved to index=0 and removed from current index. This as applying the new value to of ‘samplesPerCycle’ to ‘numberOfSamples\%samplesPerCycle’, will cause the subsequent value to be stored at index=1, as well as the fact that index=‘numberOfSamples\%samplesPerCycle’ will not be used subsequently.

• Under section F, ‘V3\_current’, ‘V3\_prev’, ‘V2\_current’ and ‘V2\_prev’ for phase shifted V2 and V3 are computed. Logic is simply to go back 1/3 or 2/3 of a cycle in the store$V[]$ array. Fine tuning an extra 1 or 2 array indexes with ‘PHASE2’ and ‘PHASE3’ is possible. Finally in the case negative array indexes are obtained, ‘samplesPerCycle’ is added (simple logic).

Also as the $emonTX Shield V2$ does not have a constant supply voltage of 3.3V as is the the case for $emonTX V3$, \('int SupplyVoltage = 3300'\) was replaced by \('int SupplyVoltage = readVcc()'\) where $readVCC$ function’s code is shown from line 554 onward of Appendix A.

Also calibration of sensors was re-performed with this modified Arduino sketch, to be sure about the new functionality. In the case of this sketch, given that the voltage for phases 2 and 3 is truly shifted by 1/3 and 2/3 of a cycle respectively, applying the same calibration logic (sensors on same phase when obtaining all calibration constant values) as used for the original sketch cannot be applied for obtaining phase calibration constants of phases 2 and 3. Instead the same phase calibration value from phase 1 was used for phases 2 and 3.

The logic of having used same phase calibration for phases 2 and 3 is justified by the fact:
• Sensors are of same model. Hence it is safe to assume the same calibration value, as in worst case scenario, value will defer by maximum $\pm 0.2$ on a $[0.0,2.0]$ scale.

• Throughout all experimentation with Open Energy Monitor, this was effectively the case, that phase calibration values were close to identical.

Performing measurements with modified ‘calcVI3Ph’ functionality and re-calibrated sensors gave results as shown in table 3.5. Parallel measurements as shown in figure 3.7 are displayed in table 3.6. Values match well, although if separate Power Factor readings by the SEPAM units were possible further discussion about the Power Factor values returned by Open Energy Monitor would be possible. However these values make much more sense than the initial data returned, as presented in table 3.7.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Current</th>
<th>Real Power</th>
<th>Apparent Power</th>
<th>Power Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.27</td>
<td>45.64</td>
<td>61.59</td>
<td>0.74</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>38.43</td>
<td>57.60</td>
<td>0.67</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>40.11</td>
<td>57.35</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Table 3.6: 3-phase fix measurement sample - SEPAM

<table>
<thead>
<tr>
<th>Phase</th>
<th>Current</th>
<th>Overall Power Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.28</td>
<td>0.29</td>
</tr>
<tr>
<td>2</td>
<td>0.29</td>
<td>0.27</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 3.7: 3-phase original measurement sample - Open Energy Monitor

<table>
<thead>
<tr>
<th>Phase</th>
<th>Current</th>
<th>Real Power</th>
<th>Apparent Power</th>
<th>Power Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>33.18</td>
<td>54.76</td>
<td>0.61</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>122.29</td>
<td>77.70</td>
<td>1.57</td>
</tr>
<tr>
<td>3</td>
<td>0.33</td>
<td>-150.75</td>
<td>76.99</td>
<td>-1.96</td>
</tr>
</tbody>
</table>

New code is shown in Appendix A, more precisely ‘calcVI3Ph’ function from line 312 onward. Rest of code also includes temperature measurement, as well as backup functionality as covered in section 4.5.
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Chapter 4

System

4.1 Motivation

emonCMS is a nice visual tool with numerous graph options to graphically display the incoming feed data. Hence it serves as an ideal tool for energy monitoring setups, as well as a robust base for more advanced applications.

Which kind of data to display on the dashboard being at the center of this study, some work on pre-display data processing functionality was necessary. This due to the fact that emonCMS while ideal for a single home user environment, is not suitable as such in a multi-user setting.

The following six sub-sections describe the addons that were developed, as well some functionality enhancements to emonCMS, for use in the social study of this thesis. All modifications were implemented on emonCMS version 8.5.

4.2 MySQLFiWa

4.2.1 Motivation

emonCMS (up to version 8.4) had a MySQL based data storage mechanism, where very simply each incoming data value is stored along with its timestamp in a database. However when viewing data over a large timeframe, fetching every single datapoint in the range can be slow. This especially in a multi-user environment, where several parallel data-fetching requests from different users occur.

emonCMS addressed this issue with its PHP based .dat file storage Feed Engines (subsection 1.2.2.3). Nonetheless in a multi-user environment, where inter-user averages are to be calculated, as well as data administration helpful, a database storage mechanism is preferred. The idea of inter-user averaging being that of computing averages based on all points within a certain timeframe, selection of the points has to be efficient. Also the same point position and number of points in the .dat file cannot be simply assumed, as each user has incoming data at different timings and intervals.

With this in mind and building upon the logic from PHPFiWa, a MySQLFiWa Feed Engine was coded.
4.2.2 Functionality

Keeping speed as key factor (in addition to the use of a database), this new Feed Engine relies on a Redis in-memory database for both metadata and pre-averaging temporary data. Metadata consists of data such as intervals of the different tables and number of points, which is accessed on a regular basis by the Feed Engines. In order to reduce fetch requests to the database, points necessary for averaging are temporarily stored in Redis before writing the computed average to the relevant interval MySQL table. This is illustrated in figure 4.1.

![Diagram](image)

**Figure 4.1: MySQLFiWa functionality**

As can be understood from the description, Redis plays a crucial role in the MySQLFiWa Feed Engine. Consequently a mechanism to recover both metadata and temporary averaging data was incorporated into the Feed Engine’s code. This was done by adding a few additional columns to the Feeds table (emonCMS functional MySQL table listing all feeds, subsubsection 1.2.2.3), while retrieving number of points from MySQLFiWa’s data tables. That way the relevant number of points could be stored back to Redis for averaging purposes. This is useful if a ‘flushall’ command were to be run in Redis or worst case scenario the server were to crash.

The above mentioned a mechanism against faulty time lapses is necessary. Even though data is posted on average every 10 seconds, posting may effectively occur after 9 or 11 seconds. Despite normalising elapsed time to 10 seconds, every once in a while effective time difference will approach 0 or 20 seconds. First scenario incoming datapoint is ignored. Second scenario was dealt with by duplicating the incoming point. As the MySQLFiWa Feed Engine relies on a fixed interval, these adjustments were crucial. If a lapse of more than 20 seconds were to occur, 0 value data is inserted into the database. While this may be imprecise, the fixed interval spacing requirement is respected. Wrong datapoints can always have their value updated and user averages will automatically be recalculated.

4.2.3 Code Structure

MySQLFiWa being a emonCMS Feed Engine, the code structure had to fully respect the sample Feed Engine template. Hence structure is composed of the following functions:

- `__construct`
Additionally two new functions were added:

- `average_compute`
- `redis_startup`

The logic of each function will be briefly explained. Full code is shown in Appendix B.

**_construct**

When a new `MySQLFiWa` feed class is created, a related new `MySQLFiWa` object is created. In PHP programming, this is achieved by calling the `__construct` constructor method. Hence the `mysqli` (link to MySQL database) and Redis instances are called. Also for error tracking purposes, the functionality of `emonLogger` is linked.

**create**

The purpose of this function is to setup the necessary MySQL tables and Redis keys. As explained in subsection 4.2.2, MySQL is for permanent data storage, while Redis stores metadata as well as temporary values for averaging computations.

Number of MySQL tables dependent on the incoming data interval; the number of averaging tables created in addition to the base table is dependent whether the averaging interval in question is larger than the incoming interval (i.e if the base incoming interval is 1 minute, only averaging tables of an interval larger than 1 minute will be created). Possible averaging table intervals are: 1 minute, 10 minute, 1 hour and 24 hours.

Redis being used, metadata is added to the existing ‘feed:id’ keys\(^1\). This is done through the `create_meta` function.

If something were to go wrong during the process, `delete` function is called, to clear all data related to the faulty feed.

**average_compute**

\(^1\)Redis key is an equivalent terminology to table
This function both computes and re-computes (if values updated) average values. As explained in subsection 4.2.1, having pre-computed average values is crucial when data is fetched for displaying on graphs. Also when exporting data, a larger interval other than that of the incoming data may be desired.

Firstly as will be the case in all functions, metadata is fetched through \textit{getmeta} function, as a guarantee that the feed actually exists. Then case of new and updating average computations is differentiated through the ‘$\text{update}$’ parameter.

Calling the \textit{lastvalue} function, timestamp of last posted value is fetched. This value along with the input timestamp variable of the \textit{average\_compute} request, is used later on to determine whether the request makes sense to proceed with. For instance if $\text{update}=\text{true}$ but the input timestamp variable is larger than the last timestamp value, clearly the request should not be considered.

Firstly an array of table suffixes (both for MySQL and Redis storage), dependent on number of tables from metadata is built. Then if update of recent timestamp (timestamp belongs to same or newer range of directly higher order interval) is to be performed, higher order (not base) Redis keys are deleted as they will be overwritten. Now \textit{average\_compute}’s main function can be performed. The case of posting vs updating is differentiated, as well as timestamp validity test performed at function start.

Basically there are two cases to consider:

- posting or update of recent timestamp
- updating of older timestamp

First case includes operations on Redis, whereas the second does not. Summary functional diagram is shown in figure 4.2.

![Summary functional diagram](image.png)

\textbf{Figure 4.2: }\textit{MySQLFiWa} average function functionality

In first case, flow of operations is as follows. Determine number of points required in averaging calculation by taking ratio between table above (where computed value will be stored) and table required for computation. This determines whether to proceed with computation or not. If insufficient points are fetched, function will be aborted without error message. Simply means that the computation in question was too early to perform. If valid, computed average is stored both in the MySQL and Redis higher interval table. The temporary Redis key is then deleted, as that data is no longer required (new incoming data will be written in a new Redis key). Note that when posting data, the timestamp is adjusted, in order to respect fixed interval spacing. This loop is re-performed for higher intervals, until number
of stored points is insufficient. Note that when computing daily averages values are not stored in Redis, as there is no higher interval averaging to use that for. Finally metadata is updated, in case of posting, not updating of recent timestamp.

In second case, solely MySQL values are updated in all averaging tables. Hence the selection of lower and upper time limit values. There is a difference between the first and higher order cases, as in the higher order cases the last higher value is used for the lower limit value. Each time data within the range is selected, average value computed and result posted in the higher table. This until the timestamp belongs to a more recent higher order range, in which case operations will switch to that of the first case described above, as data needs to be stored in Redis as well.

In both cases, data prior (not including specific timestamp) is taken for computations. For example if a minutely average were to be computed, data corresponding to timestamps of 0, 10, 20, 30, 40 and 50 seconds will be taken (not 60 seconds).

**post**

The purpose of this function is to post new incoming data to the MySQL database.

After metadata validation, it is important to determine not only if the incoming timestamp variable is bigger than the last posted value, but additionally how many values to post. This has to do with the fact that incoming data will not be incoming at a perfect regular interval of the configured value. Datapoint is ignored if effective time lapse is smaller than incoming interval. Additionally if there was a pause in data posting, whether due to a fault on the *emonCMS* server or on the *emonTX/emonBase* hardware side, it is important to account for given that *MySQLFiWa* relies on a fixed interval and consequently averaging computations will otherwise be corrupted (subsection 4.2.2).

Hence incoming dataset will either be ignored, posted or duplicated to the database. In case of pause in data posting, additional points to be posted will be done so with a data value of zero. This would make sense if there were a power failure on the hardware side. In other cases this is evidently wrong, but the zero values in the database can of course be updated through the update function.

Each time a dataset is added to the MySQL database, the same value is added to Redis. However values in Redis simply consist of the datavalue along with an incrementing numeric index, as there is no need to take up more memory by adding the UNIX timestamp. Metadata with number of points is of course updated after each posting through update function.

If initial posting, start time in both MySQL and Redis is effectively updated from zero to relevant timestamp.

For all subsequent posting, the 'average_compute' function is called. This as after new values are posted, computation of new average values is necessary.

In both cases, last value details (time and value) are updated in Redis.

**update**

Purpose of this function is to update existing data in the database.

After adjusting timestamp to respect fixed interval, data in MySQL is updated. If recent Redis data requires to be updated as well.

After updating data, the 'average_compute' function is called. This as after values are updated, re-computation of average values is necessary.
get_data
Purpose of this function is to fetch data from the database, called when populating graphs on the dashboard.

Dependent on the time interval of the request, table to fetch data from is determined. More precisely the corresponding table with an interval directly smaller than or equal to request interval will be chosen.

Depending if request interval is directly equal to or bigger than that of table in question, will determine how data is read.

Basically there are two cases to consider:

- data is read as such
- averaging of number of points within request interval, using largest timestamp from range

The above two scenarios are looped over the selection range defined by the incoming request (range is multiple of interval parameter).

lastvalue
Simply fetches last value data from Redis. Includes both UNIX timestamp, as well as point value.

export
Here data is converted to downloadable CSV. Similar to csv_export, apart that time interval of export will match that of the table used, no averaging to exactly satisfy the requesting interval. Again lower limit interval table is used for process.

delete
Deletes all MySQL/Redis tables as well as Redis metadata.

Also called if something were to go wring during the create function process.

In order to keep a clean structure of emonCMS’s functional MySQL tables, the additional column structure used by the MySQLFiWa Feed Engine is removed if the last MySQLFiWa type were to be deleted.

get_feed_size
Upon request from system, total size of MySQL tables for considered Feed is computed. This through MySQL table information of the associated Feed.

get_meta
Used throughout all core functions, fetches metadata from Redis. Includes start time, number of tables, number of points and interval arrays for those tables. Serves as a guarantee that feed exists, before executing core of other MySQLFiWa functions.

create_meta
Creates Redis metadata, for use throughout existence of feed. Initial data is in parallel stored in MySQL (only Redis is updated throughout MySQLFiWa operation).
More specifically, ‘start\_time’, ‘ntables’, ‘interval’ and ‘npoints’ will be added. ‘start\_time’
is used as a reference when data is being exported or new data added. ‘ntables’ an integer
is used in the averaging process, to determine how many times to loop the process over.
‘interval’ and ‘npoints’ are a list of the respective interval in seconds as well as the number
of rows in the respective tables. ‘interval’ and ‘npoints’ are converted back to array when
used in the MySQLFiWa code. ‘start\_time’ is stored as a datetime and converted to a UNIX
timestamp. ‘ntables’ is converted to an integer.

**update\_meta**

Used to update number of points in Redis metadata. Only Redis, not MySQL. Also note
that when updating other metadata, such as start time this function is not used.

**csv\_export**

Here data is converted to downloadable CSV. Same logic as get\_data, that time interval of
export will perfectly match that of requesting, even if averaging has to be applied.

**redis\_startup**

Purpose is to reconstruct Redis metadata, temporary data tables, as well as last point data,
should a Redis ‘flushall’ command have been executed, or a server crash occurred.

Start time, number of tables and interval is available as such from MySQL. Then number
of points is obtained by effectively counting rows in relevant data tables.

Then applying modulus calculations between number of points and interval, number of
points to add to Redis data tables is computed.

Finally last point data is stored in Redis, from MySQL data base table.

### 4.2.4 Setup

The MySQLFiWa Feed Module file cannot be integrated as such in the emonCMS environ-
ment, as is the case for any other intended addon. Unfortunately the addition of the two
new ‘average\_compute’ and ‘textitredis\_startup’ functions, requires some additional changes
to emonCMS functional files.

As a reminder, changes were implemented on emonCMS version 8.5. They are as follows:

- Add MySQLFiWa option to general Feed Engine array list in file ‘Lib/enum.php’.
  More specifically, under:

  ```php
  class Engine {
    const MYSQL = 0;
    const TIMESTORE = 1;       // Deprecated
    const PHPTIMESERIES = 2;
    const GRAPHITE = 3;        // Not included in core
    const PHPTIMESTORE = 4;    // Deprecated
    const PHPFINA = 5;
    const PHPFIWA = 6;
  }
  
  add ‘const MYSQLFIWA = 7;’ to end of array
  ```

- Add link to MySQLFiWa engine from Feed Model. More specifically, under ‘Mod-
  ules/feed/feed\_model.php’, add following two lines:
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

- after line 38, add:
  ```php
  require Modules/feed/engine/MySQLFiWa.php;
  ```

- after line 49, add:
  ```php
  $this->engine[Engine::MYSQLFIWA] = new MySQLFiWa(
    $mysqli, $redis);
  ```

- Add interface dropdown menu option
  - under 'Modules/input/process_model.php'. Line 57 onwards, add:
    ```php
    Engine::MYSQLFIWA
    ```
    to each relevant $list[] of array.
    For example:
    ```php
    $list[1] = array("Log to feed"), ProcessArg::FEEDID,
    $log_to_feed", 1, DataType::REALTIME,"Main", array(
    $Engine::PHPFIWA, $Engine::PHPFINA, $Engine:::
    PHPTIMESERIES, $Engine::MYSQLFIWA));
    ```

- under 'Modules/input/Views/input_view.php'. After line 78, add:
  ```html
  <option value=7><?php echo ('MySQLFiWa'); ?></option>
  ```

- under 'Modules/feed/Views/feedlist_view.php'. Line 133, add $MYSQLFiWa to 'engine' array. Result will be:
  ```json
  'engine':{
    'title':"<php echo ('Engine')>; "type
    $"fixedselect", 'options':['MYSQL', 'TIMESTORE',
    $PHPTIMESERIES', 'GRAPHITE', 'PHPTIMESTORE', 'PHPFINA
    $', 'PHPFIWA', 'MYSQLFiWa']}
  ```

- Add interval selection:
  - under 'Modules/input/Views/processlist.js'. Line 95, add: 'engine==7' to if condition.
  - under 'Modules/feed/feed_model.php'. After line 98, add:
    ```php
    if ($engine!="7")
    ```

- Remove Redis last_value-update_time functionality: '$this-)set_timevalue($feedid, $value, $updatetime)';. When there is a faulty time lapse (either less or more than the fixed interval), the point is skipped or duplicated in $MYSQLFiWa. Hence the Redis last_value-update_time is implemented there instead. Under 'Modules/feed/feed_model.php':
  - line 401 (insert_data function): preceed with 'if ($engine!="7")' (other engines still need this line)
  - line 454 (update_data function): preceed with 'if ($engine!="7")' (other engines still need this line)

- Integrate with Redis rebuilding (after crash or Redis Flushall). Under 'Modules/feed/feed_model.php':
  - after line 602 (load_to_redis function) and after line 625 (load_feed_to_redis function), add:
Additionally if hosting multiple applications requiring access to Redis on the same server, specification of which Redis database `emonCMS` should use, is possible. To do so under `index.php`, simply follow line 36 (`$connected = $redis->connect("127.0.0.1");`) by `$redis->select("1");`

### 4.3 MySQLFiSum

`MySQLFiSum` is based on `MySQLFiWa`, but modified for accumulative measurements such as energy during a certain time interval. This as having energy averages over a given timeframe does not make sense. `emonCMS` has an ‘Wh Accumulator’ Feed option, however the accumulation is continuous over time not for a given time interval, again not portraying clear information to the end user.

Consequently ‘average compute’ was modified to a new function ‘sum compute’, in order to compute sums and not averages. Main structure of function is identical, just the difference in output. For reference, functionality of new function is illustrated in Appendix C.

#### 4.3.1 Setup

Setup is similar to that of `MySQLFiWa` (subsection 4.2.4). Points are reapplied just replacing `MySQLFiWa` by `MySQLFiSum` where necessary.

### 4.4 Inter-user averaging & min-max

In order to display averages of other users to each specific user, average values computation functionality was also added to `emonCMS`. Code relies on `Event` and `Procedure` functionality of the MySQL environment and is shown in Appendix D.

A MySQL `Procedure` serves same functionality as a function. An `Event` is a stored process in the `Scheduler` and will be executed every minute. This as the first averaging layer is 1 minute, followed by 10 minutes, 1 hour and daily, identical to that of `MySQLFiWa` and `MySQLFiSum`.

Functionality of the `Procedure` is to loop through the averaging intervals, only computing averages if the relevant time lapse occurred. Then it will create a list of all the associated users and loop through the list, computing different averages for each of those users. Different in the sense that data of all other, but that specific user will be taken into account.

Note though that unlike `MySQLFiWa` and `MySQLSum`, inter-user averages are not updated, if the Feed Update functionality is taken advantage of. This as MySQL and `emonCMS` PHP are separate server instances and doing so would require significant changes to the `emonCMS` structure.
4.4.1 Setup

Setup requires access to the MySQL configuration or root privilege access, if event_scheduler is not active. Once the event_scheduler permanently enabled (if server crashes, the setting should not have to be re-enabled each time), rest can be done through the database user of emonCMS.

Enable event_scheduler

In order to enable event_scheduler, either:

- from MySQL terminal (logged in as user with root privileges), type the command:
  
  SET GLOBAL event_scheduler = ON;

- edit MySQL configuration file. In case of Ubuntu, file is located at ‘etc/mysql/my.conf’. Add ‘event_scheduler=ON’ under the ‘mysqld’ section of the file, to enable the MySQL system variable.

Then restart MySQL server. In case of Ubuntu, the command is as follows:

/etc/init.d/mysql restart

Setup Averaging

Easiest method of adding the newly created Procedure and Event, is by importing the .sql file contents directly via the MySQL terminal.

Once logged in as the emonCMS database user, type the following commands:

- ‘USE emoncms’; (replace emoncms by the database used by emonCMS install).
- ‘source averaging.sql’ (replace averaging by name of .sql file)

In order to confirm that the file contents were properly imported, the following two commands can be run:

- SHOW PROCEDURE STATUS;
- SHOW EVENTS;

4.5 NanodeRF Memory Backup

CIVIS partners in Trento also decided to use Open Energy Monitor for some of their installations. Consequently some exchange of ideas about how to use Open Energy Monitor occurred. Their setup relies on a Raspberry Pi as emonBase, offering a more robust hardware base in comparison with using an Arduino, if adding customisations.

Trento partners relied on the Raspberry Pi, as it served as base for other monitoring sensors as well. The full modifications undertaken will not be covered. However what is of interest is that they incorporated a mechanism to deal with Internet connection cuts. Taking advantage of the Raspberry Pi’s Redis memory, unsent data is marked with a flag, and temporarily stored in Redis with periodic writes to disk. Also the Raspberry Pi does not have its
own oscillator (durable clock), leading to imprecision in case of power cut and Internet connectivity. For this an integer describing the status of the clock, is sent with the outgoing requests (Raspberry Pi clock is synced with that of server).

That said in order to maintain the quality of CIVIS in Stockholm, an attempt to add similar functionality to the NanodeRF base was tried. As covered in section 4.2, irregular time intervals is already dealt with on the server side. However backup of data needs to be addressed. The logic is presented in upcoming subsection, although it must be emphasised that only limited values can be stored, due to memory limitations of the NanodeRF.

## 4.5.1 Functionality

### 4.5.1.1 Overview

An Arduino has three memory types as follows:

- Flash memory for Sketch (program)
- SRAM (static random access memory) for Sketch variables
- EEPROM that can be used to store long-term information

Flash and EEPROM is non-volatile, which is why when an Arduino is rebooted the loaded Sketch is still stored. On the other hand SRAM with faster write speeds, is volatile and hence all data is lost in case of power cut. This functionality is optimised for general performance of an Arduino.

Functionality of the proposed Arduino Sketch relies on remaining SRAM (other variables are also stored), as well as EEPROM memory of the NanodeRF, in parallel with the Feed update function on emonCMS server. The two components will be described separately.

### 4.5.1.2 Arduino Sketch

Full code is shown in Appendix A. All code prior to `calcVI3Ph` is of interest for this section, as that function includes a 3-phase fix that was added upon return to Barcelona (subsection 3.2.3).

As unsent data is stored in SRAM, first step was to have available as much memory as possible for this functionality. For that the `pgmspace.h` library was loaded, in order to move all constant variables from SRAM to Flash memory. This is the case for all variables including `const PROGMEM` in their declaration. These variables include hardware pin numbers, calibration constants and `emonCMS` website URL, as well as node id.

However given the limited SRAM available space of 2048 bytes, only five values can be stored, before having to move the data to EEPROM. Overall result is that 97% of SRAM is used up. For writing to EEPROM, `EEPROMex` library is loaded.

When posting data to `emonCMS`, the server response is analysed with the ‘http_response_parse’ function. The `emonCMS` server will either return “ok” or an error message. ‘http_response_parse’ reads the response and in the case of an error message will back-up the unsent data. If SRAM array has reached its size limit, data will be moved to EEPROM before the SRAM array can be overwritten.

When the Arduino is booted, EEPROM memory will be checked for stored values. If so an attempt to post these via the ‘url_update’ function is made. Similar concept in the
Sketch’s main ‘loop’ function, but in addition SRAM memory is checked after EEPROM. Upon successful re-posting, the array (SRAM and EEPROM) contents are erased. However in case of another un-successful re-posting, default behaviour of ‘http_response_parse’ is to append the SRAM/EEPROM arrays, creating duplicate entries. To avoid this if the array index has changed, ‘url_update’ will erase the duplicate array addition.

Rest of ‘loop’ function performs measurements from sensors and posts results via ‘url_post’ function.

4.5.1.3  emonCMS Modifications

emonCMS MVC framework based functionality was introduced in subsection 1.2.2. Based on that as postings from the NanodeRF are based on the Input module, it was necessary to first modify the ‘Modules/input/input_controller.php’ file.

A new function to handle update actions, based on the post actions handling function was created. Code is follows:

```php
1 // update for unsuccessfully sent data from emonBase
2 if ($route->action == 'update') {
3  $valid = true; $error = "";
4
5  $nodeid = preg_replace('/[\^\w\s-./]' , '' , get('node'));
6  $error = " old".$max_node_id_limit;
7  if (!isset($max_node_id_limit)) {
8    $max_node_id_limit = 32;
9  }
10  $error .= " new".$max_node_id_limit;
11  if (!$input->check_node_id_valid($nodeid)) {
12    $valid = false;
13    $error = "NodeID must be a positive integer between 0 and 
14             "$max_node_id_limit." , nodeid given was out of range 
15             " ;
16  }
17  if (!$valid) {
18    return array('content'="$error");
19  }
20
21  global $timestamp_fault;
22  $time = $timestamp_fault;
23
24  $datain = false;
25  // code below processes input regardless of json or csv type
26  if (isset($_GET['json'])) $datain = get('json');
27  if (isset($_GET['csv'])) $datain = get('csv');
28  if (isset($_GET['data'])) $datain = get('data');
29  if (isset($_POST['data'])) $datain = post('data');
30  if ($datain!="") {
31    $json = preg_replace('/[\^\w\s-./]' , '' , $datain);
32    $datapairs = explode(' , ' , $json);
```
$csvi = 0;
for ($i=0; $i<count($datapairs); $i++) {
    $keyvalue = explode(':', $datapairs[$i]);
    if (isset($keyvalue[1])) {
        $valid = false; $error = "Format error, json key missing or invalid character";
    }
    if (!is_numeric($keyvalue[1])) {
        $valid = false; $error = "Format error, json value is not numeric";
    }
    $data[$keyvalue[0]] = (float)$keyvalue[1];
} else {
    if (!is_numeric($keyvalue[0])) {
        $valid = false; $error = "Format error: csv value is not numeric";
    }
    $data[$csvi+1] = (float)$keyvalue[0];
    $csvi ++;
}
$userid = $session['userid'];
$dbinputs = $input->get_inputs($userid);
$tmp = array();
foreach ($data as $name => $value) {
    $inputid = $dbinputs[$nodeid][$name]['id'];
    if ($dbinputs[$nodeid][$name]['processList']) {
        $tmp[] = array('value'=>>$value, 'processList'=>>$dbinputs[$nodeid][$name]['processList']);
    }
}
foreach ($tmp as $i) {
    $process->update_data($time, $i['value'], $i['processList']);
} else {
    $valid = false; $error = "Request contains no data via csv, json or data tag";
}
if ($valid) {
    $result = 'ok';
    global $timestamp_fault;
    $timestamp_fault = $timestamp_fault + 10;
} else {
    $result = "Error: $error\n";
}
Note the $timestamp\_fault$ variable used needs to be declared at the beginning of the Modules/input/input_controller.php file. This as the global variable is also updated in the post actions handling function, as a reference of the last successful post time. More precisely the modified end of the function will become as follows:

```php
if ($valid) {
    $result = 'ok';
    global $timestamp\_fault;
    $timestamp\_fault = $time;
} else {
    $result = "Error: $error\n";
}
```

The next step was to create the corresponding 'update_data' function under 'Modules/input/process_model.php', as called by the post actions handling function on line 67 via 'process->update_data'.

```php
public function update_data($time, $value, $processList)
{
    $this->log->info("input() received time=$time, value=$value")
    $process\_list = $this->get\_process\_list();
    $pairs = explode(",", $processList);
    foreach ($pairs as $pair) {
        $inputprocess = explode(":", $pair);  // Divide into
        $processid = (int) $inputprocess[0];  // Process id
        $arg = 0;
        if (isset($inputprocess[1]))
            $arg = $inputprocess[1];  // Can be value or feed id
        $value = $this->update\_log\_to\_feed($arg, $time, $value);  //
        $this->feed\_update\_data($id, $time, $value);
    }
}
```

The 'update_log_to_feed' function called on line 15, was also added under 'Modules/input/-process_model.php', as follows:

```php
public function update_log_to_feed($id, $time, $value)
{
    $this->feed->update_data($id, $time, $time, $value);
    return $value;
}
```

An attempt to include the timestamp storage and processing on the NanodeRF or send bulk update requests instead of multiple single requests was attempted. However given both the limited memory as well as restricted C/C++ functions on the NanodeRF, this option was not possible. Hence the complicated procedure of having to also add 3 new functions to emonCMS's Input module.
4.6 Other emonCMS Functional Enhancements

The above four sections describe significant functionality enhancements made to Open Energy Monitor. However in view of offering the best possible experience to the end user, some other smaller functional fixes were made to the emonCMS interface.

As a reminder, changes were implemented on emonCMS version 8.5.

4.6.1 Deletion of Input/Feed

When a Feed is deleted, its associated Redis keys are not entirely deleted. In order to have a clean delete process, ‘Modules/feed/feed\_model.php’ was modified by adding the following after line 508:

```php
// delete feed and last value keys
$this-&gt;redis-&gt;del("feed:feed\_id");
$this-&gt;redis-&gt;del("feed:last\_value:feed\_id");
```

Also after an Input/Feed is deleted, it is practical if the MySQL auto_increment were reset, to have a consistent list of associated ids. Hence changes were as follows:

- after line 503 of ‘Modules/feed/feed\_model.php’ add
  ```php
  $this-&gt;mysqli-&gt;query("ALTER TABLE feeds AUTO_INCREMENT = 1;"); // reset auto_increment counter
  ```

- after line 336 of file ‘Modules/input/input\_model.php’ add
  ```php
  $this-&gt;mysqli-&gt;query("ALTER TABLE input AUTO_INCREMENT = 1;"); // reset auto_increment counter
  ```

4.6.2 MySQL (Basic Feed Model) Logging

The forked MySQLTimeSeries Feed Engine forked from previous emonCMS version 8.4 did not have integration with emonLogger. Also the CSV export feature was not properly working.

The changes are based on functionality of MySQLFiWa and hence code details will be omitted.

4.6.3 Logging Bug

With the release of emonCMS version 8.5, the emonLogger suddenly stopped working. Issue was due to the addition of a new file causing conflicts. Through a Github Pull Request submission, this issue was promptly fixed.
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT.

Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterno y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Chapter 5

Pilot Study - Entrepreneurship Chapter

5.1 Motivation

Testing and familiarisation with the Open Energy Monitor system, was performed over the autumn semester. However a system can only be validated, once evaluated by an end user. Hence the interest of deploying Open Energy Monitor in a research project focused on feedback from individuals.

Given the possibility of collaborating with the European CIVIS project, the monitoring system was deployed in 10 houses of the Fårdala district of Stockholm.

Additionally by using an Open system in parallel to the commercial Smappee option deployed in other buildings of the project in Stockholm, another motivation was to test the difference in performance between commercial and Open systems in a given project. This in context of promoting an Open system in relation to the progress enabled through collaboration and transparency (subsection 1.1.4).

5.2 Setup

Due to cost preference, the system was composed of an emonTX Shield V2 combined with NanodeRF. In addition to electricity monitoring, temperature was also monitored. Overview of the physical setup is shown in again in figure 5.1, subsection 3.2.1 containing further technical hardware and software details.

Space in the fuse box limited (figure 5.2), an electrician was required to install the non-invasive clip-on current transformer sensors. Then the hardware setup box was attached to the fuse box cover with double sided tape. Power extension cables and multi-outlets were required for connecting both the voltage sensor as well as NanodeRF USB power supply (table 1.2). As the NanodeRF is not WiFi compatible, an Ethernet cable had to be attached to the wall/ceiling from the user’s nearest router point.

Once the physical setup in place, the Arduino sketch of Appendix A was loaded onto the NanodeRF by specifying the user-specific ‘write API key’. Same server environment as for 1-phase testing; Ubuntu 14.04 LTS Server was used for hosting emonCMS. Power and
temperature data was sent to the server from each user’s home. Energy values can be derived on the server.

5.2.1 Smappee Replacement

However due to a code bug for 3-phase measurements (subsection 3.2.2), installation of all physical setups was not finalised. A thorough check of all integration modifications was performed, as well as order of connected current sensors switched (voltage sensor needs to be on same phase as current sensor 1). However a rescheduled flight to Barcelona was
on schedule, this after other delays related to delayed and incomplete arrival of the Open Energy Monitor equipment.

Fortunately the Open Energy Monitor equipment replaced by Smappee alternative, consumption data was monitored over the month of July. Note that an electrician was required again, to replace the Open Energy Monitor current sensors by Smappee sensors. Open Energy Monitor equipment is to be returned, after the Open Energy Monitor shop agreed to re-imburse, upon return of equipment.

That said the 3-phase code bug has been addressed, upon return to Barcelona (subsection 3.2.3).

5.3 Tests

Following the faulty code issue after physical setup (subsection 5.2.1) as well as delay in schedule, the initially planned social case study could not longer be carried out as planned. This mainly due to the fact that the allocated time for the tests of this thesis by CIVIS in Stockholm expired. Some data of less wide scope is still presented in upcoming chapter 6.

Nonetheless, the original social study outline will be presented. Idea was to perform three different tests each lasting a week, after one week of baseline measurements (user does not have access to dashboard).

Different dashboard layouts:

- realtime
- realtime & history
- realtime combined on average of other users

`emonCMS` offers the Realtime, Rawdata and Multigraph visualisation options, for the different dashboard layouts suggested above. Setup of the interface can be easily done after setting up a mockup layout in the admin user’s account. By modifying ‘content’ and ‘userid’ columns of the ‘dashboard’ table under the `emonCMS` MySQL database, the mockup view with correct data can easily be added for the other users. Note that in the case of Multigraph, the ‘multigraph’ database table first needs to be modified for each user as well (user’s rawdata and average data feeds need to be specified).

With the Smappee data from July, solely the estimated effect of installing an energy monitoring system could be evaluated. Estimated as due to the holiday season, access to past consumption bills from all the concerned users was not possible.

5.4 Smappee

5.4.1 MySmappee Interface

Smappee has a dashboard option for each user, solely displaying the user specific power consumption data down to 5 minute intervals. Energy Wh has a time resolution of 1 day. Device recognition on the other hand is performed through a separate high (kHz) sampling rates mechanism. Instant recognition response time, less than 5 seconds.
This collected and analysed data can be viewed on an online [16] as well as smartphone app interface. Solely the MySmappee online interface will be covered, as the smartphone app although used by the tested users, is not necessary for the purpose of this report as well as the fact that similar data as the MySmappee interface is displayed.

Upon logging into the MySmappee interface, both graphs and tables of power and Wh consumption data can be viewed. Additionally a table of Events (consumption breakdown per recognised device) is accessible. As consumption is lowered, the user receives messages from the Smappee system, congratulating them for the effort.

Sample screenshots of the MySmappee interface are shown figures 5.3 - 5.5. The user’s account data used in the examples has been used with their approval.

5.4.2 Smappee API

For other sorts of applications (such as inter-user averaging) or data backup (data storage on Smappee servers is not permanent), data can be exported through an API. A quick overview of the API will be given, in order to clarify how data presented in chapter 6 was obtained. All examples use PHP as the scripting language. Further API usage details can be found at [17].

5.4.2.1 Authentication

All data retrieval instances through the API require including an Access Token in the HTTP Header of the request. In order to make such a request, in addition to the user account’s username and password, a Client ID and Client Secret are required for the authentication process (obtained upon request from Smappee Support).

Example PHP code requesting an Access Token is:
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

$\texttt{ch} = \texttt{curl\_init()}$

$\texttt{param} = \textquoteleft ?\texttt{grant\_type=password\&client\_id=***\&client\_secret=***\&username=***\&password=***\'}}$

$\texttt{url} = \textquoteleft \texttt{https://app1pub.smappee.net/dev/v1/oauth2/token}' . $\texttt{param}$

$\texttt{curl\_setopt(\$ch, CURLOPT\_URL,} $\texttt{url})$

$\texttt{curl\_setopt(\$ch, CURLOPT\_RETURNTRANSFER, true)}$

$\texttt{smappee\_api\_response} = \texttt{json\_decode(curl\_exec(\$ch), true)}$

$\texttt{curl\_close(\$ch)}$

Server response obtained in return is:

\begin{verbatim}
Array ( 
    [access_token] => 789ad916-9963-3a61-bb50-d5dd355735b6 
    [refresh_token] => a24d13cb-427a-3433-ba4f-5bbdbeffe7b6 
    [expires.in] => 86400 )
\end{verbatim}

The \textit{Access Token} can then be used for all subsequent data retrieval requests for a period of 24 hours (86400 seconds corresponds to 24 hours). If more requests are made beyond that time period, either the \textit{Refresh Token} is made use of or a new authentication request can be made, in both cases to obtain another \textit{Access Token}.

Christophe Meyers
5.4.2.2 Data Retrieval

Using the Access Token in the data request’s HTTP Header, kW Consumption data as well as Events can be obtained. Consumption data includes intervals of 5 minutes, 1 hour, 1 day, 1 month and 1 year.

In order to obtain Consumption and Events data, a Service Location ID corresponding to an identification of the place regardless if multiple locations are being tracked under same Smappee account is needed. Linked to a Service Location ID are all the recognised devices.

For example making a Service Location ID request as (code relies on previous authentication request response):

```bash
$ch = curl_init();
$header = array('Authorization: Bearer ' .
    $smappee_api_response[access_token]);
curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
$url = 'https://applpub.smappee.net/dev/v1/serviceLocation/';
$url = $smappee_ServiceLocations_response[serviceLocations][0][
    serviceLocationId].'/info';
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$smappee_ServiceLocation_Info_response = json_decode(curl_exec(
    $ch), true);
curl_close($ch);
```

Server response is ('Actuators' refers to remotely controllable switches, not covered in this section):
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología "alwaysOn" a total W values over the given period. Hence a conversion to Wh is

Comparing with data from the MySmappee interface, the ‘consumption’ corresponds to total Wh and ‘alwaysOn’ to total W values over the given period. Hence a conversion to Wh is
necessary before comparing. ‘consumption’ corresponds to total and ‘alwaysOn’ to constant consumption sub-component.
Chapter 6

Data Analysis

6.1 Circumstance

Smappee replacement data obtained for the month of July, a 4 week data sample was obtained through the associated API (section 5.4.2). Two users (6 and 10) from the sample of ten considered changed their password from the initial account settings during setup and were not available, hence their data could be made use of unfortunately.

Looking at daily consumption or Watt values without the intended social case study tests (section 5.3) carried out, is meaningless. This especially without access to past electricity consumption bills, in order to evaluate the effect of installing an energy monitoring system. Events data (precise history of which device was turned on/off) was also not used for similar reasons.

Given the circumstance, daily Wh values for the 4 week period from July 6th to August 2nd were solely used. Both the ‘total’ and ‘always on’ of that period were exported and daily averages for each of the four weeks were calculated.

6.2 Raw Results

The weekly average daily average Wh values (‘total’ and ‘always on’) with weekly variations for the four week period from July 6th to August 2nd are shown in table 6.1.

<table>
<thead>
<tr>
<th>User</th>
<th>Always On</th>
<th>Total</th>
<th>Always On</th>
<th>Total</th>
<th>Always On</th>
<th>Total</th>
<th>Always On</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1313.71</td>
<td>7234.2</td>
<td>393.71</td>
<td>4 -50.3</td>
<td>246.49</td>
<td>5583.9</td>
<td>27.4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>246.429</td>
<td>4583.9</td>
<td>60</td>
<td>27.4</td>
<td>268.857</td>
<td>3545.9</td>
<td>-22.6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3920.57</td>
<td>60</td>
<td>24137.44</td>
<td>33.6</td>
<td>3970.14</td>
<td>21060.1</td>
<td>-12.7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>790</td>
<td>5203.94</td>
<td>-59.4</td>
<td>3.168</td>
<td>2500.29</td>
<td>7195.14</td>
<td>-38.2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1599.71</td>
<td>8368.3</td>
<td>-9.8</td>
<td>5.8</td>
<td>1764</td>
<td>8883.76</td>
<td>-22.6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1579.29</td>
<td>7532.59</td>
<td>-1.27</td>
<td>105</td>
<td>1721</td>
<td>5836.21</td>
<td>-41.2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2285.86</td>
<td>5283.94</td>
<td>3.16</td>
<td></td>
<td>3126</td>
<td>14275.7</td>
<td>22.5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5326</td>
<td>14644.5</td>
<td>13.7</td>
<td></td>
<td>5326</td>
<td>14275.7</td>
<td>22.5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>14275.7</td>
<td>14644.5</td>
<td>22.5</td>
<td></td>
<td>5326</td>
<td>14275.7</td>
<td>22.5</td>
<td></td>
</tr>
<tr>
<td>10*</td>
<td>14275.7</td>
<td>14644.5</td>
<td>22.5</td>
<td></td>
<td>5326</td>
<td>14275.7</td>
<td>22.5</td>
<td></td>
</tr>
</tbody>
</table>

* data of these users was inaccessible, as they changed initial default Smappee account password
The results presented as such, cannot lead to a direct conclusion whether installing a home energy monitoring system, has a direct positive effect on the user’s energy consumption. Also some users went on holiday during the evaluated period, as can be interpreted from unusual large variations in electricity consumption.

6.3 Analysis

From the results of table 6.1, it can nonetheless be noted which users had a stable energy usage over the considered four weeks. More precisely users 4, 5, 8, 9 have consumption values with no major variation.

User 4 had a steady decrease of ‘always on’ sub-consumption, despite a significant increase of the ‘total’ consumption. The user’s ‘total’ and ‘always on’ superimposed consumption variation over the four week period is shown in figure 6.1. The significant increase of ‘total’ consumption is clearly visible for week 2.

User 5 shows an ideal example. Overall decrease of 39.8% in ‘total’ consumption and a 10% decrease of ‘always on’ sub-consumption. The user’s ‘total’ and ‘always on’ superimposed consumption variation over the four week period is shown in figure 6.2.

User 8 despite an increase of ‘total’ consumption by 9.4%, still managed to lower their ‘always on’ sub-consumption by 10%. The user’s ‘total’ and ‘always on’ superimposed consumption variation over the four week period is shown in figure 6.3.
variation over the four week period is shown in figure 6.3. As for user 4, variation of ‘total’ consumption has erratic sections.

User 9 unfortunately increased both their ‘total’ consumption and ‘always on’ sub-consumption by 14.4% and 2% respectively. The user’s ‘total’ and ‘always on’ superimposed consumption variation over the four week period is shown in figure 6.4. The increase of both consumption types is clearly visible for week 4.
Figure 6.4: ‘Total’ and ‘always On’ superimposed consumption of user 9

The consumption variation of these four considered users is graphically presented in figures 6.5 and 6.6. Despite the negative details presented here above, overall effect of installing an energy monitoring system over a 1-month period seems more or less positive.
Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.
Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Chapter 7

Other CITCEA Projects
Running on Open Energy Monitor

7.1 Dish Stirling

As part of the European project DIDSOLIT-PB [18], a parabolic dish solar collector of around 4m diameter referred to as Dish Stirling was installed on ETSEIB’s roof. CITCEA in charge of monitoring the unit’s energy production planned to use Open Energy Monitor as part of the energy monitoring system.

On 12th March as part of the Dish Stirling Training, an introductory presentation about the benefits of using Open Energy Monitor with such a setup was given.

Dish Stirling’s electric output is approximately 1 kW AC 1-phase. Hence the 3-phase code bug is not of an issue, as well as that 1-phase in a single user environment measurements has been successfully demonstrated (chapter 3).

7.2 Bachelor Project

Two Bachelor students Mathieu Degand and Florian Leniere used the emonTX Shield V2 combined with NanodeRF to monitor CITCEA’s EUETIB Aula Schneider’s fridge temperature variation in parallel to energy consumption, as part of their semester project. They did not use a DS18B20 sensor, but instead their own setup based on an LM335Z sensor.

emonCMS being a complex system to understand, in parallel to this thesis, guidance on how to integrate their setup and obtain the desired visualisations in emonCMS was given throughout the semester.
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consecuencia se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo aislado de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Chapter 8

Budget

Budget for the project was calculated based on

- total amount of work hours dedicated
- purchase of equipment
- travel expenses

Detailed budget breakdown for this project is shown in table 8.1 and amounts to a total of €36,901.68.
Table 8.1: Budget breakdown of project

<table>
<thead>
<tr>
<th>Type of Work</th>
<th>Profession</th>
<th>Dedicated Time [hours]</th>
<th>Cost [€/h]</th>
<th>Total Cost €</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>Engineer</td>
<td>520</td>
<td>50</td>
<td>26,000</td>
</tr>
<tr>
<td>Equipment Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>Technician</td>
<td>100</td>
<td>30</td>
<td>3000</td>
</tr>
<tr>
<td>- Soldering/Calibration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Other Purchases</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Displacement/Installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Organisation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Project Support</td>
<td>Consultant</td>
<td>15</td>
<td>50</td>
<td>750</td>
</tr>
<tr>
<td>(Dish Stirling &amp; Bachelor Project)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative/Planning</td>
<td>Administrative</td>
<td>10</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>(Barcelona)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIVIS Partner Work</td>
<td>Consultant</td>
<td>8</td>
<td>50</td>
<td>400</td>
</tr>
<tr>
<td>(Stockholm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Energy Monitor Forum</td>
<td>Administrative</td>
<td>3</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Administrative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report Writing</td>
<td>Administrative</td>
<td>130</td>
<td>10</td>
<td>1,300</td>
</tr>
<tr>
<td>Installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Open Energy Monitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Purchase of Equipment</td>
<td></td>
<td>1657.14*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Return of Defective Equipment</td>
<td></td>
<td>0**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Electrician</td>
<td></td>
<td>261.48***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Extra Equipment</td>
<td></td>
<td>295.51***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Ethernet Cables, Boxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Extension Multi-Outlet Cables,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cable Tunnels, Tape</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Smappee</td>
<td></td>
<td>1,987.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Purchase of Equipment</td>
<td></td>
<td>1726.46***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Electrician</td>
<td></td>
<td>261.48***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stockholm Travel</td>
<td></td>
<td>1,119.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Flight</td>
<td></td>
<td>321.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Accommodation***</td>
<td></td>
<td>654.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Transport***</td>
<td></td>
<td>143.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Total Amount</td>
<td></td>
<td>36,901.68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 1 Euro = 0.73 GBP, rate of 27 August 2015
** To be returned for full amount hopefully
*** 1 Euro = 9.56 SEK, rate of 27 August 2015
Chapter 9

Environmental and Social Impact

The aim of this project being to reduce home energy consumption, target was clearly to have both positive environmental and social impacts. However the manufacturing and shipping environmental effects of the used equipment, as well as that due to traveling have to be considered in the entire scope. As well as the associated effects of ICT systems of energy monitoring systems.

First half of project consisted of programming the necessary modifications to Open Energy Monitor system (chapter 4) as well as administrative/planning tasks necessary for equipment purchase and travel funding for stay in Stockholm. Hence environmental effect was negligible and direct social impact nonexistent.

After arriving in Stockholm, the ordered Open Energy Monitor equipment arrived. This obviously had a negative environmental impact during manufacturing and shipping. Despite being an Open Source project, the Open Energy Monitor equipment has similar costs to other commercial alternatives. Hence it can be assumed that adequate manufacturing conditions are associated with the project. Social impact is hence positive. As an Open Source project was used, social impact is further maximised. Only environmental effect due to shipping needs to be considered. This effect can unfortunately not be excluded when working with an energy monitoring system.

An electrician hired during the installation process, is another positive social impact. Also the author of this thesis solely relied on public transport when carrying equipment to the houses, as well as purchasing other necessary equipment. This is definitely a positive environmental impact.

Unfortunately Smappee replacement units needed to be ordered. By doing so there was a positive social impact through the effect on the economy. As for the Open Energy Monitor purchase, there was a negative environmental impact during shipping, but inevitable. However by the users reducing their home energy consumption, this negative effect is compensated in the long run. Same argument goes for the associated impacts of managing webservers used by the energy monitoring system.

After performing the 3-phase fix (subsection 3.2.3), solution was shared on Open Energy Monitor forum. Also the other fixes from chapter 4 were published on Github, as well as shared on the project’s forum. This in view of increasing this research project’s social impact.
Flying back and forth to Stockholm, obviously had a negative effect on the environment. However collaborating in the European CIVIS was a unique opportunity and this environmental impact was the price to pay. If it were not for the research scope as offered by CIVIS, much less modifications would have been performed to the Oen Energy Monitor system.

Nonetheless the negative impacts due to traveling and shipping of the equipment are outweighed by the other positive environmental and social impacts presented. Hence the environmental and social impact of this project can be concluded as being positive.
Chapter 10

Conclusion

10.1 Behavioral Impact of Energy Monitoring

From the analysis of results carried out in section 6.3, it unfortunately cannot be directly stated that home energy monitoring systems reduce the electricity consumption of its inhabiting smartcitizen.

Though there were results for both sides of the argument, overall the effect of installing a home energy monitoring system seems positive. Both users 4 and 5 decreased their ‘total’ consumption, while user 8 decreased their ‘always on’ sub-consumption.

As already stated not only the considered time period was short, but additionally the considered results had no baseline testing (e.g. past energy bills). This was of course taken into account in the initial social case study testing, however as outlined due to issues encountered while deploying Open Energy Monitor in Färdala Stockholm, was not possible.

It can be questioned whether the drastic energy savings of users 4 and 5 will persist over time, as the presented result is most probably due to an initial positive reaction to that of being aware of one’s energy consumption. The longer term testings of the CIVIS project will determine this, as well as if the promised 12% reduction as promoted by some home energy monitoring suppliers is valid. Installing an home energy monitoring system seems to overall have a positive impact.

Most important after all, are the user actions encouraged through an effective energy consumption communication via the monitoring system. Some of the simple user actions encouraged through CIVIS can be found at [20].

10.2 Evaluation of Open Energy Monitor

Another aspect of this thesis was to evaluate the performance between Open and commercial energy monitoring systems. Although Open Energy Monitor fulfilled performance requirements for 1-phase measurements under a laboratory environment, deployment in CIVIS was far below standards. 3-phase erroneous algorithm being one factor, ease of setup and user-friendliness was another major drawback.
10.2.1 Drawbacks

An all-in-one WiFi compatible solution is a critical requirement for acceptance by the common non-technical user base of the population. This even if installation is performed for them, so that Ethernet cabling is effectively taken care off. There was one user who could not tolerate the idea of having an Ethernet cable going around the corners of their living room and refused to have the Open Energy Monitor setup installed. One could argue to use the *emonTX V3* with *Raspberry Pi* through an RFM12B/RFM69CW wireless link. However there is a significant cost difference and the majority of commercial options are a cheaper alternative.

Also with *emonTX Shield V2* combined with *NanodeRF* setup, two power supply sockets are required; one for the voltage sensor of the shield and another for the *NanodeRF* base. Reducing the power supply requirement to one would be of practical convenience.

Casing inclusion (*emonTX V3* does have a case, although not very aesthetic), auto-detection of current sensor phases (differentiation of current sensors and input ports is irrelevant) and better integration with other technology such as IFTTT, are some further development suggestions that would be beneficial to Open Energy Monitor.

Nonetheless Open Energy Monitor is as of today, the most feature rich Open option for energy monitoring. Another point of improvement would be to lower overall cost, related to additional purchase of Ethernet cable and box. Cost breakdown of Open (Open Energy Monitor) vs commercial (Smappee), can be seen in table 8.1.

10.2.2 Improvement Possibility

The 3-phase erroneous algorithm addressed, the other drawbacks described (subsection 10.2.1) still need to be addressed. Still the resulting modified Open Energy Monitor system is one step closer for deployment in user’s homes. Taking an entrepreneurial approach to tackling the issue, contact was established with another like-minded Open Energy Monitor forum user and the possibility of a spin-off project addressing the gap with commercial alternatives is being discussed. It is important to emphasise that this user is working as an energy monitoring consultant and hence has both quite some extended experience and contacts in the sector. As a starting point, a blog offering a detailed comparison of commercial energy monitoring systems has been opened. [19]
Acknowledgements

I would like to thank Andreas Sumper for having made the collaboration between UPC and KTH possible. As well as his and Pau Lloret’s understanding and patience, when deploying the Open Energy Monitor system in Stockholm.

At KTH I would like to thank Björn Palm and Omar Shafqat, for having allowed me to collaborate in the CIVIS project. A special recognition to Omar who guided me during the complicated deployment of the Open Energy Monitor system. And Hossein Shahrokni for initial contact setup.

Additionally thank you to KIC InnoEnergy and Mar Martinez; without their financial support, this project would not have been possible. Also to KTH for purchasing the return flight.

Numerous programming concepts and tools needed to be mastered throughout this project, I would like to thank the people who gave those time-saving debugging tips when necessary.

Lastly I am grateful to the support offered by my friends during my time both in Barcelona and Stockholm, as well as my family who kept me on track during this thesis.

I surely forgot numerous names above, the support offered during my thesis being extensive.
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consecuencia se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Appendix A

**emonTX Shield V2-NanodeRF**

3-phase with Backup

```c
/*
 * NanodeRF-emonTX Shield V2: 3-phase-Temperature with Backup
 * 
 * Auto-detection of DS18B20 sensor (no need to specify sensor address)
 * Sends Power & Temperature data to emonCMS server
 * Unsuccessful send requests have data backed-up in SRAM, with periodic writes to EEPROM
 * (SRAM is volatile, EEPROM is non-volatile)
 * Stored data is then sent via update post requests
 * 
 * See attached readme.md for instructions on how to integrate with emonCMS
 * 
 * Based on Open Energy Monitor emonTxFirmware
 * Part of the openenergymonitor.org project
 * Licence: GNU GPL V3
 * 
 * Author: Christophe Meyers
 */

// Comment below when not debugging
#ifndef DEBUGGING

// Store constant variables in Program Flash memory
#include <avr/pgmspace.h>

// Post interval
const PROGMEM uint16_t rate[] = {10000}; // milliseconds

// Electricity sensing setup
```

Javier Saiz Anadón

Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo eléctrico para garantizar la alimentación de una parte de la carga considerada prioritaria.
// Detect CTs in use
bool CT1inUse = true;
bool CT2inUse = true;
bool CT3inUse = true;

// Set Voltage and current input pins
const PROGMEM uint16_t inPinV[] = {0};
const PROGMEM uint16_t inPinI1[] = {1};
const PROGMEM uint16_t inPinI2[] = {2};
const PROGMEM uint16_t inPinI3[] = {3};

// Calibration Coefficients (need to be set in order to obtain accurate results)
const PROGMEM float Vcal[] = {260.0}; // Calibration constant for voltage input
const PROGMEM float Ical1[] = {60.606}; // Calibration constant for current transformer 1
const PROGMEM float Ical2[] = {60.606}; // Calibration constant for current transformer 2
const PROGMEM float Ical3[] = {60.606}; // Calibration constant for current transformer 3
const PROGMEM float Phasecal1[] = {1.7}; // Calibration constant for phase shift L1
const PROGMEM float Phasecal2[] = {1.7}; // Calibration constant for phase shift L2
const PROGMEM float Phasecal3[] = {1.7}; // Calibration constant for phase shift L3
const PROGMEM uint16_t PHASE2[] = {0}; // Number of samples delay for L2 [fine tune by −1 or 1]
const PROGMEM uint16_t PHASE3[] = {0}; // Number of samples delay for L3 [fine tune by −1 or 1]

// Variable Declaration
double realPower1, apparentPower1, powerFactor1, Irms1, realPower2,
    apparentPower2, powerFactor2, Irms2, realPower3,
    apparentPower3, powerFactor3, Irms3, realPower, Vrms;
const PROGMEM uint16_t node_id[] = {0};

// Arduino Library (see emonlib.h)
// To enable 12-bit ADC resolution on Arduino Due,
// include the following line in main sketch inside setup()
// function:
// Otherwise will default to 10 bits, as in regular Arduino–
// based boards
#if defined(__arm_)
    #define ADC_BITS 12
#else
    #define ADC_BITS 10
#endif
#define ADC_COUNTS (1<<ADC_BITS)

#if defined(ARDUINO) && ARDUINO >= 100
    #include "Arduino.h"
#else
    #include "WProgram.h"
69  #endif
70  
71 // Ethernet & Server
72 #include <EtherCard.h>
73 // Ethernet Interface MAC Address (must be unique on the LAN)
74 static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
75 // Account Write API Key
76 char apikey[] = "";
77 const char website PROGMEM = "emoncms - leave blank if posting to IP address"
78 byte Ethernet::buffer[500]; // Small tcp/ip buffer
79 boolean use hisip = false;
80  
81 // DS18B20 sensor
82 #include <OneWire.h>
83 #include <DallasTemperature.h>
84 #define ONE_WIRE_BUS 4 // Data wire is plugged into port 2 on the Arduino
85 #define TEMPERATURE_PRECISION 12
86 OneWire oneWire(ONE_WIRE_BUS); // Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/
87 DallasTemperature sensors(&oneWire);
88 DeviceAddress address,T1;
89 double temperature;
90  
91 // Store unsent values in SRAM if unsent (2048 bytes available)
92 // ARRAY SIZE ONLY NEEDS TO BE CHANGED FOR SRAM
93 typedef struct {double W; double celsius;} SRAM_struct;
94 SRAM_struct SRAM_array[5];
95 int SRAM_index = 0;
96 int SRAM_index_old = 0;
97 // Move unsent values to EEPROM, when SRAM array full (1024 bytes available)
98 #include <EEPROMex.h>
99 typedef struct {double W; double celsius;} EEPROM_struct;
100 EEPROM_struct EEPROM_array[sizeof(SRAM_array)];
101 int EEPROM_index = 0;
102 int EEPROM_index_old = 0;
103  
104 // On-board emonTx LED
105 const PROGMEM uint16_t LEDpin[] = {9};
106  
107 // Loopping variable in [void loop ()] function
108 unsigned long timer = 0;
109  
110 //------------------------------------------------------------------------------
111 // The PacketBuffer class is used to generate the json string that is send via ethernet - JeeLabs
112 //------------------------------------------------------------------------------
class PacketBuffer : public Print {
public:
    PacketBuffer () : fill (0) {}
    const char* buffer () { return buf ; }
    byte length () { return fill ; }
    void reset ()
    {
        memset(buf,NULL,sizeof(buf));
        fill = 0;
    }
    virtual size_t write (uint8_t ch) 
    {
        if (fill < sizeof buf) buf[fill++] = ch;
    }
    byte fill;
    char buf[150];
private:
};
PacketBuffer str;

// Called for each packet of returned data from the call to
// browseUrl

static void http_response_parse (byte status, word off, word len)
{
    Ethernet::buffer[off+300] = 0;
    Serial.print("HTTP Response: "); Serial.println((const char*)
        Ethernet::buffer + off);
    char *ptr = strstr((const char*)Ethernet::buffer + off,"ok");

    if(ptr== NULL)
    {
        // Store unsuccessful data in SRAM array
        // If SRAM_index=5 first move array content to EEPROM
        if(SRAM_index==sizeof(SRAM_array))
        {
            EEPROM.writeBlock(EEPROM_index, SRAM_array);
            EEPROM_index += sizeof(SRAM_array);
        }
        //Reset SRAM_index
        SRAM_index = 0;
        }
    SRAM_array[SRAM_index].W = realPower;
    SRAM_array[SRAM_index].celsius = temperature;
    SRAM_index += 1;
    }

// Post measured values

static void url_post (float url_realPower, double url_temperature)
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

```c
{
  str.reset();
  str.print("/emoncms/input/post.json?");
  str.print(" apikey="); str.print(apikey);
  str.print("&node="); str.print(pgm_read_word_near(node_id+0));
  str.print("&json={realPower:"); str.print(url_realPower); str.print("}");
  str.print(" temperature:"); str.print(url_temperature);
  str.print("}");
  str.print("\0"); // End of json string
  ether.browseUrl(PSTR(""), str.buf, website, http_response_parse);
  #ifdef DEBUGGING
  Serial.print(" Data sent: "); Serial.println(str.buf);
  #endif
}

// Post backed-up values

static void url_update (float url_realPower, double url_temperature)
{
  str.reset();
  str.print("/emoncms/input/update.json?");
  str.print(" apikey="); str.print(apikey);
  str.print("&node="); str.print(pgm_read_word_near(node_id+0));
  str.print("&json={realPower:"); str.print(url_realPower); str.print("}");
  str.print(" temperature:"); str.print(url_temperature);
  str.print("}");
  str.print("\0"); // End of json string
  SRAM_index_old = SRAM_index;
  EEPROM_index_old = EEPROM_index;
  ether.browseUrl(PSTR(""), str.buf, website, http_response_parse);
  #ifdef DEBUGGING
  Serial.print(" Data sent: "); Serial.println(str.buf);
  #endif
  if (SRAM_index>SRAM_index)
  {
    SRAM_index = SRAM_index_old; //reset
  }
  if (EEPROM_index>EEPROM_index_old) {
    EEPROM.readBlock(EEPROM_index_old, SRAM_array);
    EEPROM.writeBlock(EEPROM_index, 0);
    EEPROM_index = EEPROM_index_old;
  }
```
void setup()
{
  // Initialise Ethernet library
  ether.begin(sizeof(Ethernet::buffer), mymac);
  ether.dhcpSetup();
  ether.printIp("My IP: ", ether.myip);
  ether.printIp("Netmask: ", ether.netmask);
  ether.printIp("GW IP: ", ether.gwip);
  ether.printIp("DNS IP: ", ether.dnsip);
  ether.dnsLookup(website);
  ether.printIp("SRV: ", ether.hisip);

  ether.packetLoop(ether.packetReceive());

  // If unsent data (only EEPROM, as SRAM is empty)
  if (EEPROM_index>0) {
    for (int i=0; i<EEPROM_index; i+=sizeof(SRAM_array)) {
      EEPROM.readBlock(i, EEPROM_array);
      EEPROM.write(i, 0); // Erase contents
      for (int j=0; j<sizeof(EEPROM_array); j++)
        { url_update(EEPROM_array[j].W, EEPROM_array[j].celsius); }
      EEPROM_index = 0; // Reset
    }

  // Temperature setup
  sensors.begin();
  sensors.getAddress(address_T1, 0);

  // Do some measurements to allow the software filter to settle —
  // don't use the result
  if (DEBUGGING)
    Serial.println("Initial 5 filter settling measurements
  occuring. Wait a while ...");
}

Building Block Apartments Energy Monitoring Interactive System With Social Case Study

```c
#endif
for (int j=0; j<5; j++)
{
    calcVI3Ph(11,2000);
}

// Setup indicator LED
pinMode(pgm_read_word_near(LEDpin+0), OUTPUT);
digitalWrite(pgm_read_word_near(LEDpin+0), HIGH);

void loop()
{
ether.packetLoop(ether.packetReceive());
if ((millis()−timer) >= pgm_read_word_near(rate+0)) //
    // Rollover handled by Arduino
    
    timer = millis();

    // If unsent data
    // Get EEPROM data & clear EEPROM
    if (EEPROM_index>0) {
        for (int i=0; i<EEPROM_index; i+=sizeof(SRAM_array))
            {
                EEPROM.readBlock(i, EEPROM_array);
                EEPROM.write(i, 0);
                for (int j=0; j<sizeof(EEPROM_array); j++)
                    {
                        url_update (EEPROM_array[j].W, EEPROM_array[j].celsius);
                    }
                EEPROM_index = 0; // Reset
            }
    }

    // Get SRAM data & clear SRAM
    if (SRAM_index>0) {
        for (int j=0; j<sizeof(SRAM_array); j++)
            {
                url_update (SRAM_array[j].W, SRAM_array[j].celsius);
            }
        SRAM_index = 0;
    }

    // Calculate Voltages & Currents
    calcVI3Ph(11,2000);
    realPower = realPower1 + realPower2 + realPower3;
    // Measure temperature
    sensors.requestTemperatures(); // Send the command to get
    // temperatures
    temperature = sensors.getTempC(address_T1);
    // Send to server
```

Christophe Meyers

Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
url_post (realPower, temperature);
}
}

//*************************************************************

// Implementation of emonLib for 3-phase measurements

void calcVI3Ph(int cycles, int timeout)
{
  // Variable declaration for filters, phase shift, voltages,
  // currents & powers

  static int lastSampleV, sampleV;  // 'sample' holds the
                                      // raw analog read value, 'lastSample' holds the last sample
  static int lastSampleI1, sampleI1;
  static int lastSampleI2, sampleI2;
  static int lastSampleI3, sampleI3;

  static double lastFilteredV, filteredV;  // 'Filtered' is the
                                           // raw analog value minus the DC offset
  static double lastFilteredI1, filteredI1;
  static double lastFilteredI2, filteredI2;
  static double lastFilteredI3, filteredI3;

  static double offsetV = ADC_COUNTS>>1;  //Low-pass filter
                                          // output = start at half-rail, or zero if CT was not
                                          // detected.
  static double offsetI1 = ADC_COUNTS>>1;
  static double offsetI2 = ADC_COUNTS>>1;
  static double offsetI3 = ADC_COUNTS>>1;

  double phaseShiftedV1;  // Holds the
t                         // calibrated delayed & phase shifted voltage.
  double phaseShiftedV2;
  double phaseShiftedV3;

  double sumV, sumI1, sumI2, sumI3;
  double sumP1, sumP2, sumP3;  // sq = squared, sum = Sum
                               // inst = instantaneous

  int startV;  // Instantaneous voltage
              // at start of sample window.
  int SupplyVoltage = readVcc();
int crossCount = -2; // Used to measure number of times threshold is crossed.
int samplesPerCycle = 0; // Used for storedV[]
int numberOfSamples = 0; // This is now incremented of voltages needs to be stored before use
int numberOfPowerSamples = 0; // Needed because 1 cycle
boolean lastVCross, checkVCross; // Used to measure number of times threshold is crossed.
double storedV[30]; // Array to store >240 degrees of voltage samples. Over-dimension slightly with 30, as array size is not dynamic
int V2_current;
int V2_prev;
int V3_current;
int V3_prev;

// 1) Waits for the waveform to be close to 'zero' (500 adc) part in sin curve.

boolean st=false; // An indicator to exit the while loop
unsigned long start = millis(); // millis()-start makes sure it doesn't get stuck in the loop if there is an error.
while(st==false) // Wait for first zero crossing...
{
    startV = analogRead(pgm_read_word_near(inPinV+0));
    if ((startV < 550) && (startV > 440)) st=true; // Check voltage is close to zero
    if ((millis()-start)>timeout) st = true; // Exit if overtime
}

// 2) Main measurement loop

start = millis();
while ((crossCount < cycles * 2) && ((millis()-start)<timeout))
{
    lastSampleV=sampleV; // Used for digital high pass filter - offset removal
    lastSampleI1=sampleI1;
    lastSampleI2=sampleI2;
}
```c
lastSampleI3 = sampleI3;
lastFilteredV = filteredV;
lastFilteredI1 = filteredI1;
lastFilteredI2 = filteredI2;
lastFilteredI3 = filteredI3;

// A) Read in raw voltage and current samples
sampleV = analogRead(pgm_read_word_near(inPinV+0));     // Read in raw voltage signal
sampleI1 = analogRead(pgm_read_word_near(inPinI1+0));   // Read in raw current signal
sampleI2 = analogRead(pgm_read_word_near(inPinI2+0));   // Read in raw current signal
sampleI3 = analogRead(pgm_read_word_near(inPinI3+0));   // Read in raw current signal

// B) Apply digital high pass filters to remove 2.5V DC offset (to centre wave on 0).
offsetV = offsetV + ((sampleV-offsetV)/1024);
filteredV = sampleV - offsetV;
offsetI1 = offsetI1 + ((sampleI1-offsetI1)/1024);
filteredI1 = sampleI1 - offsetI1;
offsetI2 = offsetI2 + ((sampleI2-offsetI2)/1024);
filteredI2 = sampleI2 - offsetI2;
offsetI3 = offsetI3 + ((sampleI3-offsetI3)/1024);
filteredI3 = sampleI3 - offsetI3;

storedV[(numberOfSamples%samplesPerCycle)] = filteredV;     // Store this voltage sample in circular buffer

// C) Find the number of times the voltage has crossed the initial voltage
//    - every 2 crosses we will have sampled 1 wavelength
//    - so this method allows us to sample an integer number of half wavelengths which increases accuracy
//    Build storedV array

lastVCross = checkVCross;
checkVCross = (sampleV > startV) ? true : false;
if (numberOfSamples==0)
```
lastVCross = checkVCross;  // Initial set for condition
              ↭ below
if (lastVCross != checkVCross)
    crossCount++;
    ↭
    if (crossCount == 0)  // Started recording at -2
            ↭ crossings so that one complete cycle has been stored
            ↭ before accumulating.
    {
        samplesPerCycle = numberOfSamples;  // StoredV[] size
            ↭ now
        storedV[0] = filteredV;
        storedV[samplesPerCycle] = 0;
        sumV = 0;
        sumI1 = 0;
        sumI2 = 0;
        sumI3 = 0;
        sumP1 = 0;
        sumP2 = 0;
        sumP3 = 0;
        numberOfPowerSamples = 0;
    }
}

#if define DEBUGGING
    Serial.print(" Fixed samplesPerCycle : "); Serial.println(
            ↭ samplesPerCycle);
#endif

    // D) Root-mean-square method voltage
    // sum += square voltage values
    sumV += filteredV * filteredV;

    // E) Root-mean-square method current
    // sum += square current values
    sumI1 += filteredI1 * filteredI1;
    sumI2 += filteredI2 * filteredI2;
    sumI3 += filteredI3 * filteredI3;

    // F) Phase calibration – for Phase 1: shifts V1 to correct
    // transformer errors

Christophe Meyers
// for phases 2 & 3 delays V1 by 120 degrees & 240 degrees
// respectively
// and shifts for fine adjustment and to correct
// transformer errors.

if (crossCount >= 0) {
    // Compute indexes
    V3_current = (numberOfSamples%samplesPerCycle) - round((samplesPerCycle /3)) - pgm_read_word_near(PHASE3+0);
    V3_prev = (numberOfSamples%samplesPerCycle) - round((samplesPerCycle /3)) - 1 - pgm_read_word_near(PHASE3+0);
    V2_current = (numberOfSamples%samplesPerCycle) - round(2*(samplesPerCycle /3)) - pgm_read_word_near(PHASE2+0);
    V2_prev = (numberOfSamples%samplesPerCycle) - round(2*(samplesPerCycle /3)) - 1 - pgm_read_word_near(PHASE2+0);
    if (V2_current < 0) {
        V2_current += samplesPerCycle;
    }
    if (V2_prev < 0) {
        V2_prev += samplesPerCycle;
    }
    if (V3_current < 0) {
        V3_current += samplesPerCycle;
    }
    if (V3_prev < 0) {
        V3_prev += samplesPerCycle;
    }
    phaseShiftedV1 = lastFilteredV + pgm_read_float_near(Phasecalc1+0) * (filteredV - lastFilteredV);
    phaseShiftedV2 = storedV[V2_prev] + pgm_read_float_near(Phasecalc2+0) * (storedV[V2_current] - storedV[V2_prev] - 1);
    phaseShiftedV3 = storedV[V3_prev] + pgm_read_float_near(Phasecalc3+0) * (storedV[V3_current] - storedV[V3_prev] - 1);
}

// G) Instantaneous power calc

sumP1 += phaseShiftedV1 * filteredI1;  // Sum +=
sumP2 += phaseShiftedV2 * filteredI2;
sumP3 += phaseShiftedV3 * filtered13;

numberOfPowerSamples++; // Count number of times looped for Power averages (reminder: different from counter below)

numberOfSamples++; // Count number of times looped.

// 3) Post loop calculations

// Calculation of the root of the mean of the voltage and current squared (rms)
// Calibration coefficients applied.

double V_Ratio = pgm_read_float_near(Vcal+0) *((SupplyVoltage /1000.0) / 1023.0);

Vrms = V_Ratio * sqrt(sumV / numberOfPowerSamples);

double I_Ratio1 = pgm_read_float_near(Ical1+0) *((SupplyVoltage /1000.0) / 1023.0);

Irms1 = I_Ratio1 * sqrt(sumI1 / numberOfPowerSamples);

double I_Ratio2 = pgm_read_float_near(Ical2+0) *((SupplyVoltage /1000.0) / 1023.0);

Irms2 = I_Ratio2 * sqrt(sumI2 / numberOfPowerSamples);

double I_Ratio3 = pgm_read_float_near(Ical3+0) *((SupplyVoltage /1000.0) / 1023.0);

Irms3 = I_Ratio3 * sqrt(sumI3 / numberOfPowerSamples);

// Calculation power values

realPower1 = V_Ratio * I_Ratio1 * sumP1 / numberOfPowerSamples;

apparentPower1 = Vrms * Irms1;

powerFactor1 = realPower1 / apparentPower1;

realPower2 = V_Ratio * I_Ratio2 * sumP2 / numberOfPowerSamples;

apparentPower2 = Vrms * Irms2;

powerFactor2 = realPower2 / apparentPower2;

realPower3 = V_Ratio * I_Ratio3 * sumP3 / numberOfPowerSamples;

apparentPower3 = Vrms * Irms3;

powerFactor3 = realPower3 / apparentPower3;

// Reset accumulators

sumV = 0;

sumI1 = 0;

sumI2 = 0;

sumI3 = 0;

sumP1 = 0;

sumP2 = 0;
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

```c
sumP3 = 0;

#elif DEBUGGING
  Serial.println();
  Serial.print("Total Samples: "); Serial.println(numberOfSamples);
  Serial.print("Power Samples: "); Serial.println(numberOfPowerSamples);
  Serial.print("Time: "); Serial.println(millis() - start);
  Serial.print("Crossings: "); Serial.println(crossCount);
  Serial.print("storedV:");
  for (int j=0; j<30; j++)
    Serial.print(storedV[j]); Serial.print(" ");
  Serial.println();
#endif

exampleModalLabel
  //***************************************************************
  // Read Vcc, as not fixed at 3.3V as for emonTX V3
  //***************************************************************

  long readVcc()
  {
    long result;
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    delay(2);
    ADCSRA |= _BV(ADSC);
    while (bit_is_set(ADCSRA,ADSC));
    result = ADCL;
    result |= ADCH<<8;
    result = 1126400L / result;
    return result;
  }
```

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus células a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrogero para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Appendix B

MySQLFiWa

```php
<?php

/// **REQUIRES REDIS !!!**

/* See attached readme.md for instructions on how to integrate
   with Emoncms*/

This MySQL based engine stores timeseries type data in an InnoDB
   (more robust) table
and updates linked MYISAM (faster reads) averaging tables (same
   logic as PHPFiWa)

Average intervals: 1 minute, 10 minutes, 1 hour & 24 hours
   (if feed range is larger than certain proposed average interval,
   interval is ignored)

Temporary average data (collecting required number for average
   computation) is stored in Redis as well, to avoid
   unnecessary MySQL Select queries

Some new columns for meta data will be added to the 'feeds'
   table of the MySQL database (as Redis is not permanent)

Add-on to Open Energy Monitor (http://openenergymonitor.org)
   Emoncms code
Written by Christophe Meyers
Released under the GNU Affero General Public License.
*/

// no direct access
defined('EMONCMS_EXEC') or die('Restricted access');

class MySQLFiWa
{
    private $mysqli;
    private $redis;
```
private $log;

/**
 * Constructor
 * @param api $mysqli Instance of mysqli
 * @param api $redis Instance of redis
 * Logging via EmonLogger
 */
public function __construct($mysqli,$redis)
{
    $this->mysqli = $mysqli;
    $this->redis = $redis;
    $this->log = new EmonLogger(__FILE__);
}

/**
 * Automatic interval averaging tables creation
 * @param integer $feedid id of feed table to be created
 * @param array $options Interval parameter
 */
public function create($feedid,$options)
{
    //parse inputs
    $feedid = (int) $feedid;
    $interval = (int) $options['interval '];
    $feedname = "feed_" . trim($feedid);

    //check to ensure we dont overwrite an existing feed (meta data already exists)
    if (!$meta = $this->get_meta($feedid)) {
        //log & create initial meta data
        $this->log->info("MySQLFiWa: Feed $feedid, Create started");
        $meta = new stdClass();
        $meta->id = $feedid;
        $meta->start_time = 0;

        //interval determines how many averaging tables to create
        if ($interval<5) $interval = 5; //5 seconds is minimum required interval
    } else if ($interval==5 || $interval==10 || $interval==15 ||
              $interval==20 || $interval==30) {
        $result[] = $this->mysqli->query("CREATE TABLE
e$feedname (time INT UNSIGNED, data float, INDEX ("time ") ENGINE=InnoDB;");
        $result[] = $this->mysqli->query("CREATE TABLE
e$feedname"."_1mins (time INT UNSIGNED, data

Javier Saiz Anadón

Christophe Meyers
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$\text{result}[] = \text{mysqli} \rightarrow \text{query}("\text{CREATE TABLE} \\
\rightarrow \text{feedname} (\text{time INT UNSIGNED}, \text{data float}, \\
\rightarrow \text{INDEX (‘time’) ENGINE=InnoDB;}");
$meta->interval = "$interval,86400";
$meta->npoints = "0,0";

//possible errors during table creation -> delete data
$i = 0;
while ($i<sizeof($result)) {
    if ($result[$i]==false) {
        $j = $i+1;
        $this->log->warn("MySQLFiwa: Feed $feedid, Create Failed, MySQL table $j of ",$meta->ntables."
            " creation unsuccessful");
        $this->delete($feedid);
        return false;
    }
    $i++;
}
unset($result);

//metadata (adds data to Redis & MySQL tables)
if (!$meta_create = $this->create_meta($feedid,$meta)) {
    $this->log->warn("MySQLFiwa: Feed $feedid, Create Failed, see warning from Create_Meta");
    return false;
} else {
    $this->log->info("MySQLFiwa: Feed $feedid, Create Successful");
    return true; //only if all successful
}

//Meta already exist
} else {
    $this->log->warn("MySQLFiwa: Feed $feedid, Create Failed, metadata already exists");
    return false;
}

/**
 * Computes values for average tables, based on input changes
 * Queries performed on Redis temporary stored values, results
 * written to MySQL
 * @param integer $feedid id of the feed to compute average on
 * @param integer $timestamp UNIX timestamp of the data point
 * [seconds]
 * @param float $value Value of the data point
 */
private function average_compute($feedid,$timestamp,$value,
    $update)
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología en sus tendencias y aplicaciones. Para garantizar la alimentación de una parte de la carga considerada prioritaria, se ha diseñado el grupo electrógeno para adaptar la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red, se ha basado en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación del modelo de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado del acumulador.

Christophe Meyers

Case Study
Building Block Apartments Energy Monitoring Interactive System With Social
Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de almacenamiento e interconexión. La microrred se ha modelado con el objetivo de garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

↑ $meta->interval[1]) + 1) * ($meta->interval[1])
↑ + ($meta->start_time); // scale to fixed
↑ interval spacing

$timestamp = $avg_time;

} else {
  $avg_time = $timestamp;
}

if ($update === false) {
  $result = $this->mysqli->query("INSERT INTO feed_meta (timestamp) VALUES ('$avg_time', '')");
}

if ($result === false) {
  $this->log->warn("MySQLFiwa: Feed $feedid -
↑ timestamp=$timestamp avg_table =".$tab_append[
↑ $i+1].", Average $action failed, posting
↑ computed data to MySQL unsuccessful");
}

return false;

} // update meta only if update=false

if ($update === false) {
  // update num_points in metadata
  $npoints[$i+1] += 1;
  $result = $this->update_meta($feedid, $meta);

  if ($result === false) {
    $this->log->warn("MySQLFiwa: Feed $feedid -
↑ timestamp=$timestamp avg_table =".$tab_append[
↑ $i+1].", Average $action failed, updating
↑ metadata num_points unsuccessful");
  }

  } // ii) non-sufficient points

  } elseif ($stored_points != 0) {
    unset($tab_append);
    return true; // try again when next post/update
  } // iii) no points fetched

} else {
  $this->log->warn("MySQLFiwa: Feed $feedid - timestamp=
↑ $timestamp, Average $action failed, no data
↑ available");

} // update of older timestamp

} elseif ($update === true && ($last_timestamp - $meta
↑ ->start_time)/$meta->interval[($i+1)] > floor {
  $timestamp = $meta->start_time)/$meta->interval[($i+1) |

// get data from mysql
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la necesidad de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de una de sus celdas. En el modo isla de la microrred se ha priorizado utilizar en la menor medida posible la red. En el modo de conexión al distribuidor se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas con conexiones en paralelo. Se ha diseñado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

---

Case Study
Building Block Apartments Energy Monitoring Interactive System With Social Media Insights

---

public function post($feedid,$timestamp,$value)
{
   //parse inputs
   $feedid = (int) $feedid;
   $timestamp = (int) $timestamp;
   $value = (float) $value;

   $feedname = "feed_".trim($feedid);

   //if meta data does not exist then exit
   if (!$meta = $this->get_meta($feedid)) {
     $this->log->info("MySQLFiwa: Feed $feedid - timestamp= " . $timestamp . " value="$value" , Post failed , fetching metadata unsuccessful");
     return false;
   }

   //1) Initial point -> first update meta start time (both Redis & MySQL)
   if ($meta->start_time==0) {
     $this->log->info("MySQLFiwa: Feed $feedid - timestamp= "$timestamp" value="$value" , Post--initial started");
     $start_time = date("Y-n-j H:i:s" , $timestamp);
     $result = $this->redis->hSet("feed:$feedid" , 'start_time' . $start_time);
     if ($result===false) {
       $this->log->warn("MySQLFiwa: Feed $feedid - timestamp= "$timestamp" value="$value" , Post--initial fail , Redis start_time set unsuccessful");
       return false;
     }
   }

   $result = $this->mysql->query("UPDATE feeds SET 'start_time'='$start_time' WHERE 'id'='$feedid';");
   if ($result===false) {
     $this->log->warn("MySQLFiwa: Feed $feedid - timestamp= "$timestamp" value="$value" , Post--initial fail , MySQL start_time set unsuccessful");
     return false;
   }

   //add first datapoint

---

Christophe Meyers
En este proyecto se empieza analizando la definición de microrred, su estado actual y la conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de conversión de energía solar. En el modo de conexión a la red, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza a modelar y simular una microrred fotovoltaica con el fin de garantizar la alimentación de una parte de la carga prioritaria a través de la generación fotovoltaica. El diseño de la microrred se ha realizado para placas de células policristalinas con la finalidad de optimizar la utilización de la energía solar. Las placas solares se conectan a un convertidor de CA a CC, que se utiliza tanto para las placas PV como para el acumulador o batería.

El control de la microrred se basa en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red y aislado, la fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas con la finalidad de optimizar la utilización de la energía solar. Las placas solares se conectan a un convertidor de CA a CC, que se utiliza tanto para las placas PV como para el acumulador o batería.

Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de la duración de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de la duración de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado.

Después de esto, se busca realizar una microrred en sus estados de tendencia en su diseño. Para lograr esto, se busca realizar una microrred en sus estados de tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de tendencia en su diseño.

Christophe Meyers
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de una microrred basada en generación fotovoltaica. El diseño de la misma se ha realizado para placas de células policrystalinas con conexiones en paralelo. De esta forma, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red y aislado la microrred se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la red se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Christophe Meyers
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

array = timestamp

return false;

// check to see if timestamp of post value does not already exist
$list_array = $this->lastvalue($feedid);
$list_timestamp = strtotime($list_array[0]);
if ($timestamp>$list_timestamp) {
    $this->log->warn("MySQLFiwa: Feed $feedid - timestamp=
    $timestamp value=$value, posting of datapoints to be
    made via post function");
    return false;
}

// if exists -> proceed
$this->log->info("MySQLFiwa: Feed $feedid - timestamp=
    $timestamp value=$value, Update started");

// scale to fixed interval spacing
$timestamp = (int) floor((strtotime($meta->start_time))/(($meta->interval[0]) * ($meta->interval[0]) + $meta->start_time);
$result = $this->mysql->query("UPDATE feedname SET data=
    $value WHERE 'time' = '$timestamp';");
if ($result===false) {
    $this->log->warn("MySQLFiwa: Feed $feedid - timestamp=
    $timestamp value=$value, Update failed, posting to
    MySQL table unsuccessful");
    return false;
}

// Redis (only if recent) -> fetch from from MySQL then post
$start_time = floor((strtotime($meta->start_time))/(($meta->interval[1]) * ($meta->interval[1]) + $meta->start_time); // lower end of range
$result_sql = $this->mysql->query("SELECT 'data' FROM feed_
    ":$feedid." WHERE 'time' >= '$start_time';");
if ($result_sql->num_rows) <= ((strtotime($meta->start_time))/(($meta->interval[0])[$meta]))
    while ($row = $result_sql->fetch_array()) {
        $values[] = $row['data'];
    }
    $redis_table_name = "$feed_data:".$feedid.":".$feedid;
    $this->redis->del($redis_table_name);
    for ($i=0; $i<sizeof($values); $i++) {
        $value = $values[$i];
        $result = $this->redis->hSet("$redis_table_name", $i, $value);
485 }  
486 unset($values);  
487 $i = 0;  
488 // possible errors  
489 while ($i<sizeof($result)) {  
490 if ($result[$i]===false) {  
491 $this->log->warn("MySQLFiWa: Feed $feedid – timestamp=  
492 $timestamp value=$value, Update failed, posting  
493 to Redis table unsuccessful");  
494 return false;  
495 }  
496 $i++;  
497 }  
498 unset($result);  
499 }  
500 // posting successful – compute averages  
501 $update = true;  
502 $result = $this->average_compute($feedid,$timestamp,$value,  
503 $update);  
504 if ($result===false) { // error computing averages  
505 $this->log->warn("MySQLFiWa: Feed $feedid – timestamp=  
506 $timestamp value=$value, Update failed, see  
507 warning from Average-updating");  
508 return false;  
509 }  
510 $this->log->info("MySQLFiWa: Feed $feedid – timestamp=  
511 $timestamp value=$value, Update successful");  
512 return true;  
513 }  
514  
515 /**  
516 * Return the data for the given timerange  
517 *  
518 * @param integer $feedid id of the feed to fetch from  
519 * @param integer $start UNIX timestamp IN MS of start of data  
520 * @param integer $end UNIX timestamp IN MS of end of data  
521 * @param integer $outinterval Interval size of range in  
522 * seconds  
523 */  
524 public function get_data($feedid,$start,$end,$outinterval)  
525 {  
526 // parse data (now all is seconds, not ms)  
527 $feedid = intval($feedid);  
528 $start = floatval($start/1000);  
529 $end = floatval($end/1000);  
530 $outinterval = intval($outinterval);  
531 // if meta data does not exist then exit  
532
528 if (!$meta = $this->get_meta($feedid)) { 
529   $this->log->warn("MySQLFiwa: Feed $feedid – range=(
530   ↔ $start , $end , $outinterval) , Get_Data failed ,
531   ↔ fetching metadata unsuccessful");
532   return false;
533 }
534 if ($outinterval<1) $outinterval = 1;
535 $dp = ceil(($end - $start) / $outinterval); // datapoints
536 $end = $start + ($dp * $outinterval); // correct to respect
537   ↔ range increments
538 if ($dp<1) return false; // minm datapoints less than 1
539 $this->log->info("MySQLFiwa: Feed $feedid – range=(
540   ↔ $start , $end , $outinterval) , Get_Data started");
541 $feedname = "feed_" . trim($feedid);
542 $data = array();
543 $range = $end - $start;
544 // depending on datatype (user_range, minute, 10 min,
545   ↔ hourly, daily)
546 if ($outinterval >=86400) {
547   $feedname_table=$feedname."_daily";
548   } else if ($outinterval >=3600) {
549     $feedname_table=$feedname."_hourly";
550   } else if ($outinterval >=600) {
551     $feedname_table=$feedname."_10mins";
552   } else if ($outinterval >=60) {
553     $feedname_table=$feedname."_1mins";
554   } else {
555     $feedname_table=$feedname;
556   }
557 // return data array with spacing of $outinterval (using
558   ↔ prepared statements) [via a loop computing averages
559   ↔ over $interval increments]
560 $data = [];
561 if ($outinterval ==86400 || $outinterval ==3600 ||
562 $outinterval ==600 || $outinterval ==60 ||
563 $outinterval ==$meta->interval[0]) {
564   $stmt = $this->mysqli->prepare("SELECT ‘time’, ‘data’
565     ↔ FROM $feedname_table WHERE ‘time’=?");
566   $stmt->bind_param("i", $end);
567   $stmt->bind_result($dataTime, $dataValue);
568 } else {
569   $stmt = $this->mysqli->prepare("SELECT MAX(‘time’), AVG(‘
570     ↔ data’) FROM $feedname_table WHERE ‘time’<=? AND ‘time
571     ↔ ‘<=? ORDER BY ‘time’ ASC;");
572   $stmt->bind_param("ii", $start, $end);
573   $stmt->bind_result($dataTime, $dataValue);
574 }
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrogeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
** Return export data for specified range
* (no averaging, just lower range match among available
  averaging tables)

* @param integer $feedid id of the feed
* @param integer $start UNIX timestamp of start in seconds
* @param integer $interval Interval in seconds
*
*/

public function export($feedid, $start, $interval)
{
    // parse inputs
    $feedid = intval($feedid);
    $start = intval($start);
    $interval = intval($interval);

    // if meta data does not exist then exit
    if (!$meta = $this->get_meta($feedid)) {
        $this->log->warn("MySQLFiwa: Feed $feedid - start=
            $start interval=$interval, Export failed,
            fetching metadata unsuccessful");
        return false;
    }

    // determine which table to export from based on $interval
    // if not exact match, closest lower limit will be taken
    if ($interval<$meta->interval[1]) {
        $table_interval = $meta->interval[0];
    } elseif ($meta->ntables==2) {
        $table_interval = $meta->interval[1];
    } elseif ($meta->ntables==3 && $interval<$meta->interval
            [2]) {
        $table_interval = $meta->interval[1];
    } elseif ($meta->ntables==3) {
        $table_interval = $meta->interval[2];
    } elseif ($meta->ntables==4 && $interval<$meta->interval
            [3]) {
        $table_interval = $meta->interval[2];
    } elseif ($meta->ntables==4) {
        $table_interval = $meta->interval[3];
    } elseif ($meta->ntables==5 && $interval<$meta->interval
            [4]) {
        $table_interval = $meta->interval[3];
    } else {
        $table_interval = $meta->interval[4];
    }

    // determine which table from MySQL table to export from
    if ($table_interval==60) {

Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de carga. La conexión de ambos componentes se hace mediante el teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Para el modelado de un grupo electrógeno se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con la potencia activa y reactiva.
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

```php
704 }
705
706 $moredata_available = false;
707 while($row = $result->fetch_array()) {
708     // write block as csv to output stream
709     if (!isset($row['data2'])) {
710         fputcsv($fh, array($row['time'], $row['data']))
711     } else {
712         fputcsv($fh, array($row['time'], $row['data'], $row['data2']));
713     }
714 }
715
716 // set new start time so that we read the next block along
717 $start = $row['time'] + $table_interval;
718 $moredata_available = true;
719 }
720 // 2) Sleep for a bit
721 usleep($sleep);
722 }
723 fclose($fh);
724 exit;
725 }
726
727 /**
728 * Delete Entire Feed (average tables as well)
729 * (Removal of individual points will cause averaging complications)
730 *
731 * @param integer $feedid id of the feed
732 */
733
734 public function delete($feedid)
735 {
736     // parse inputs
737     $feedid = (int) $feedid;
738
739     // if meta data does not exist then exit
740     if (!$meta = $this->get_meta($feedid)) {
741         $this->log->warn("MySQLFiwa: Feed $feedid, Delete failed, fetching metadata unsuccessful");
742         return false;
743     }
744
745     // remove Redis data (feed:$feedid metadata in feed_model)
746     if ($this->redis) {
747         $result = $this->redis->del("feed_data:$feedid"); // all subkeys
748     } else {
749     }
```

Christophe Meyers

Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus células a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consecuencia se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrogeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conectado a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de una pila de células policristalinas. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación efectuada para su diseño. El diseño de la misma se ha realizado para placas de células monocristalinas, con el objetivo de mejorar la eficiencia fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas, con el objetivo de mejorar la eficiencia fotovoltaica. El diseño de la misma se ha realizado para placas de células monocristalinas, con el objetivo de mejorar la eficiencia fotovoltaica.

Javier Saiz Anadón

112 Building Block Apartments Energy Monitoring Interactive System With Social Case Study

Christophe Meyers
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

```php
function get_feed_size($feedid) {
    // if meta data does not exist then exit
    if (!$meta = $this->get_meta($feedid)) {
        $this->log->warn("MySQLFiwa: Feed $feedid, Get_Feed_Size failed, fetching metadata unsuccessful");
        return false;
    }
    $feedname = "feed_".$feedid;
    $result = $this->mysqli->query("SHOW TABLE STATUS LIKE '%$feedname%'");
    if ($result!==false) {
        $this->log->warn("MySQLFiwa: Feed $feedid, Get_Feed_Size failed, fetching MySQL data unsuccessful");
        return false;
    }
    $row = $result->fetch_array();
    $tablesize = $row['Data_length']+$row['Index_length'];
    return $tablesize;
}
```

```php
/**
 * Return metadata about feed
 * (Used by other functions, to ensure feed exists)
 * @param integer $feedid id of the feed
 */
public function get_meta($feedid) {
    if ($this->redis->hExists("feed:$feedid", 'start_time')===true) {
        $meta = new stdClass();
        $start_time_temp = $this->redis->hGet("feed:$feedid", 'start_time');
        if ($start_time_temp==='0') {
            
```
```
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza... electrógeno para garantizar la alimentación... carga considerada prioritaria.

Javier Saiz Anadón

Building Block Apartments Energy Monitoring Interactive System With Social Case Study

114

/**
  * Create metadata about new feed
  * Redis and feeds table in MySQL (data is kept in MySQL for...
  */

public function create_meta($feedid,$meta)
{
  /*Redis
    $result = $this->redis->hMSet("feed:$feedid", array('...
    if ($result===false) {
      $this->log->warn("MySQLFiwa: Feed $feedid, Create_Meta fail, Redis unsuccessful");
      $this->redis->del("feed:$feedid");
      return false;
    }

    //MySQL
    $col_exists = $this->mysqli->query("SHOW COLUMNS FROM feeds LIKE 'start_time';");
    if ($col_exists->num_rows==0) {
      $result[]= $this->mysqli->query("ALTER TABLE feeds ADD COLUMN 'start_time' DATETIME, ADD COLUMN 'ntables'");
  */
Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Despues de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas tendencia en su diseño. Despues de esto, se busca realizar una microrred en sus estados de

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red y aislado, se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Case Study

Building Block Apartments Energy Monitoring Interactive System With Social Media Functionality

Christophe Meyers
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de conversión fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas con conexiones y aisladas con la potencia activa y reactiva. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la mínima medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la mínima medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva.

Javier Saiz Anadón
Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la microrred, se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la red se ha priorizado utilizar el grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

$\text{interval} = 10\text{mins}$

$\text{interval} + 1)

$\text{interval} == 600)$

$\text{interval} == 3600)$

$\text{interval} == 86400)$

$\text{time limit}(120)$

$\text{sleep} = 80000$

$\text{header}('\text{Cache-Control: no-cache, no-store, must-revalidate}')$
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado del conversor. El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo isla de la red se ha priorizado utilizar en la menor medida posible la red. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red.

Javier Saiz Anadón

1008  // tell the browser to handle output as a csv file to be downloaded
1009  header('Content-Disposition: File Transfer');
1010  header('Content-type: text/csv');
1011  header('Content-Disposition: attachment; filename='
1012      '<?php echo $fileName ?>', '');
1013  header('Expires: 0');
1014  header('Pragma: no-cache');
1015  // write to output stream
1016  $fh = @fopen( 'php://output', 'w' );
1017  // determine number of points to select from table per averaging computation
1018  $num_points = $interval/$table_interval; // assumed to be integer value
1019  // load new feed blocks (SQL Select and average calculations) until there is no more data
1020  $stmt = $this->mysqli->prepare("SELECT * FROM $feedname WHERE `time` >= ? LIMIT $num_points;" );
1021  $stmt->bind_param("i", $sql_start);
1022  $stmt->bind_result($dateTime, $dataValue);
1023  $sql_start = $start;
1024  $moredata_available = true;
1025  while ($moredata_available) {
1026      // 1) Load a block
1027      $stmt->execute();
1028      $value = 0;
1029      while ($stmt->fetch()) {
1030          if ($dateTime>0 && $dataValue>0) {
1031              $time = $dateTime * 1000;
1032              $value += $dataValue;
1033          } else {
1034              $this->log->warn("MySQLFiwa: Feed $feedid -> range=($start, $end, $interval), CSV_Export failed, fetching MySQL data unsuccessful");
1035              return false;
1036          }
1037      }
1038  }
1039  $avg_value = $value/$num_points;
1040  unset($value);
1041  // write block as csv to output stream
1042  fputcsv($fh, array($time, $avg_value));
1043  // set new start time so that we read the next block along
1044  $sql_start = $sql_start + $interval;
1045  if ($sql_start<=$end) {
1046      $moredata_available = true;
1047  } else {
1048      $moredata_available = false;
1049  }
1050  // 2) Sleep for a bit

Building Block Apartments Energy Monitoring Interactive System With Social Case Study
while ({$i<$size($tab_append)}) {
    $table = "$feed_.$feedid.$tab_append[$i]";
    $result_mysql = $this->mysqli->query("SELECT 'start_time', 'ntables', 'interval' FROM feeds where 'id'='".$feedid."';");
    $row = $result_mysql->fetch_array();
    $result[] = $this->redis->hSet("feed:$feedid", 'start_time', $row['start_time']);
    $result[] = $this->redis->hSet("feed:$feedid", 'ntables', $row['ntables']);
    $result[] = $this->redis->hSet("feed:$feedid", 'interval', $row['interval']);
    $table = "$feed_.$feedid.$tab_append[$i]";
    $result_mysql = $this->mysqli->query("SELECT Count(*) AS $table from feed_data WHERE feedid = "$feedid" AND feedid = "$feedid" AND start_time BETWEEN CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE AND CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE");
    $result[] = $result_mysql->fetch_row();
}

// num_points
$i = 0;
$num_points = "";
while ($i<$size($tab_append)) {
    $table = "$feed_.$feedid.$tab_append[$i]";
    $result_mysql = $this->mysqli->query("SELECT Count(*) AS $table from feed_data WHERE feedid = "$feedid" AND feedid = "$feedid" AND start_time BETWEEN CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE AND CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE");
    $result[] = $result_mysql->fetch_row();
    $i++;
}

// set array of table name suffixes
$ntables = intval($row['ntables']);
if ($ntables==5) {
    $tab_append[] = ".1mins";
}
if ($ntables==4) {
    $tab_append[] = ".10mins";
}
if ($ntables==3) {
    $tab_append[] = ".hourly";
}
$tab_append[] = ".daily";

// num_points
$i = 0;
$num_points = "";
while ($i<$size($tab_append)) {
    $table = "$feed_.$feedid.$tab_append[$i]";
    $result_mysql = $this->mysqli->query("SELECT Count(*) AS $table from feed_data WHERE feedid = "$feedid" AND feedid = "$feedid" AND start_time BETWEEN CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE AND CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE");
    $result[] = $result_mysql->fetch_row();
    $i++;
}

// metadata to Redis
$result_mysql = $this->mysqli->query("SELECT 'start_time', 'ntables', 'interval' FROM feeds where 'id'='".$feedid."';");
$row = $result_mysql->fetch_array();
$result[] = $this->redis->hSet("feed:$feedid", 'start_time', $row['start_time']);
$result[] = $this->redis->hSet("feed:$feedid", 'ntables', $row['ntables']);
$result[] = $this->redis->hSet("feed:$feedid", 'interval', $row['interval']);

/ * Build up Redis metadata (after redis flushall or crash) * /

public function redis_startup($feedid)
{
    $feedid = (int) $feedid;

    // metadata to Redis
    $result_mysql = $this->mysqli->query("SELECT 'start_time', 'ntables', 'interval' FROM feeds where 'id'='".$feedid."';");
    $row = $result_mysql->fetch_array();
    $result[] = $this->redis->hSet("feed:$feedid", 'start_time', $row['start_time']);
    $result[] = $this->redis->hSet("feed:$feedid", 'ntables', $row['ntables']);
    $result[] = $this->redis->hSet("feed:$feedid", 'interval', $row['interval']);

    // set array of table name suffixes
    $ntables = intval($row['ntables']);
    if ($ntables==5) {
        $tab_append[] = ".1mins";
    }
    if ($ntables==4) {
        $tab_append[] = ".10mins";
    }
    if ($ntables==3) {
        $tab_append[] = ".hourly";
    }
    $tab_append[] = ".daily";

    // num_points
    $i = 0;
    $num_points = "";
    while ($i<$size($tab_append)) {
        $table = "$feed_.$feedid.$tab_append[$i]";
        $result_mysql = $this->mysqli->query("SELECT Count(*) AS $table from feed_data WHERE feedid = "$feedid" AND feedid = "$feedid" AND start_time BETWEEN CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE AND CURRENT_TIMESTAMP - INTERVAL "$table" MINUTE");
        $result[] = $result_mysql->fetch_row();
        $i++;
    }
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación del diodo doble. El algoritmo utilizado para su control es el del MPPT.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red y aislado se ha priorizado utilizar en la menor medida posible la red. En el modo isla la conexión de todos los componentes se hace mediante el bus de alterna. Se ha diseñado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
$i++ convictions.

> $tab_append [$i] = $this->redis->hGet($redis_table, $i);

1136  $j = 1;
1137  while ($j<=$size) {
1138      $k = $j - 1;
1139      $result [] = $this->redis->hGet($redis_table, $k, $sql_array [(size-$j)]); // returned array is reversed order
1140      $j++;
1141  }
1142  unset ($sql_array);
1143  // possible errors loading metadata
1144  $k = 0;
1145  while ($k<sizeof($result)) {
1146      if ($result [$k]===false) {
1147          $this->log->warn("MySQLFiwa: Feed $feedid, Redis_Startup unsuccessful, loading feed_data to Redis $tab_append [$i]"
1148      }
1149      $k++;
1150  }
1151  unset ($result);
1152  }
1153  $i++;
1154  }
1155  unset ($tab_append);
1156
1157  // update Redis lastvalue metadata
1158  $result = $this->mysqli->query("SELECT * FROM feed_. $feedid . ORDER BY 'time' DESC LIMIT 1;"));
1159  if ($result!==false) {
1160      $row = $result->fetch_array();
1161      $updatetime = date("Y-n-j H:i:s", $row ['time']);
1162      $data = $row ['data'];
1163      $this->redis->hSet("feed:lastvalue:$feedid", array('value' => $data, 'time' => $updatetime));
1164  }
1165  }
1166 }
1167 ?>

Christophe Meyers

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se ha hecho mediante el bus de alterno y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón

Christophe Meyers
Appendix C

MySQLFiSum

Only code for ‘sum_compute’ replacing ‘average_compute’ is shown here, as rest is identical to MySQLFiWa.

```c
/**
 * Computes values for average tables, based on input changes
 * (called by post/update)
 * Queries performed on Redis temporary stored values, results
 * written to MySQL
 *
 * @param integer $feedid id of the feed to compute average on
 * @param integer $timestamp UNIX timestamp of the data point
 * [seconds]
 * @param float $value Value of the data point
 */
private function sum_compute($feedid, $timestamp, $value, $update = false) {
    // if meta data does not exist then exit
    if (!$meta = $this->get_meta($feedid)) {
        $this->log->warn("MySQLFiSum: Feed $feedid - timestamp = $timestamp, Average-" . $action . " failed, fetching metadata unsuccessful");
        return false;
    }

    // get last_point (for $update=true)
    $last_array = $this->lastvalue($feedid);
    $last_timestamp = strtotime($last_array[0]);

    // set suffix for logging
    if ($update===true) {
        $action = 'updating ';
    } else {
        $action = 'posting ';
    }

    // set array of table name suffixes (last not useful)
```
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología en su diseño. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
$\sum = 0 ;$

while ($\sum <$stored_points) {
    $\sum += (\text{float}) \text{redis\_array}[0][\$j] ;$
    $\$j++ ;$
}

// discard temporary Redis table
\$this\rightarrow\text{redis}\rightarrow\text{del}(\$redis\_table\_name) ;
unset (\$redis\_array) ;

// store average value in Redis & MySQL higher table
// post to Redis (not last table)
if (\$i<(\$meta\rightarrow\text{ntables}-2)) {
    \$redis\_sum\_table\_name = "feed\_data:" . \$feedid . ":" .
        \$feedid . \$tab\_append [\(\$i+1\)] ;
    \$redis\_sum\_table\_index = (\text{int}) \$this\rightarrow\text{redis}\rightarrow\text{hLen} (\$redis\_sum\_table\_name) ;
    \$result = \$this\rightarrow\text{redis}\rightarrow\text{hSet} (\$redis\_sum\_table\_name ,
        \$redis\_sum\_table\_index , \$sum) ;
if (\$result==false) {
    \$this\rightarrow\log\rightarrow\text{warn} ("MySQLFiSum: Feed \$feedid -. 
        \$result=". \$i+1) . \$sum . \$action\_failed , posting
        \$computed\_data\_to\_Redis\_unsuccessful") ;
    return false ;
}
}

// compute/update MySQL sum
// scale to fixed interval spacing, $\sum\_time$ consists
// of data prior to then
if (\$tab\_append[\$i]==") {
    \$sum\_time = (\text{floor} ((\$timestamp-\$meta\rightarrow\text{start\_time}) /
        (\$meta\rightarrow\text{interval}[1])+1) * (\$meta\rightarrow\text{interval}[1])
        + (\$meta\rightarrow\text{start\_time}) ; // scale to fixed
        \$interval\_spacing
        \$timestamp = \$sum\_time ;
} else {
    \$sum\_time = \$timestamp ;
}

if (\$update==false) {
    \$result = \$this\rightarrow\text{mysqli}\rightarrow\text{query} ("INSERT INTO feed_. 
        \$result = \$this\rightarrow\text{mysqli}\rightarrow\text{query} ("UPDATE feed_. trim ( \
        \$result = \$this\rightarrow\text{mysqli}\rightarrow\text{query} ("MySQLFiSum: Feed \$feedid -. 
}

return false ;
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas suficientes, lo que es posible gracias a la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a la red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a la red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la frecuencia y amplitud de tensión con la potencia activa y reactiva.
return false;

}$sum = 0;
while ($row = $result->fetch_array()) {
    $sum += $row['data'];
}

if ($tab_append[$i] == "") {
    $result = $this->mysqli->query("UPDATE feed_".$feedid.$tab_append[$i+1]." SET 'data'='$sum'
WHERE 'time'='$lim_high';");
} else {
    $lim_high = $lim_high - $meta->interval[$i];
    $result = $this->mysqli->query("UPDATE feed_".$feedid.$tab_append[$i+1]." SET 'data'='$sum'
WHERE 'time'='$lim_high';");
}

if ($result == false) {
    $this->log->warn("MySQLFiSum: Feed $feedid - timestamp
    =$timestamp sum_table=".$tab_append[$i+1]." ,
    Average-$action failed, updating MySQL
    unsuccessful");
    return false;
} else {
    unset($tab_append);
    return true;
}
$i++;
}
En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Appendix D

Inter User Averaging

/*
   See attached readme.md for instructions on how to setup with MySQL Event Scheduler
*/

This SQL script is triggered every min_avg_interval by the MySQL engine
Computes inter-user average values
Average intervals: 1 minute, 10 minutes, 1 hour & 24 hours
(if min_avg_interval is larger than certain proposed average
  interval, interval is ignored)
All tables are MYISAM (faster reads) tables

%%%% Under Event: %%
%%%% - Change 'EVERY 1 MINUTE' to relevant min_avg_interval (1
  interval higher than feed_interval) %%
%%%% - Also set feed_name & feed_interval (feed dependant in 'Feeds' table)

------------------------------

Add-on to Open Energy Monitor (http://openenergymonitor.org)
  Emoncms code
Written by Christophe Meyers
Released under the GNU Affero General Public License.
*/
/
/-- Stored Procedure --
------------------------------
DELIMITER $$
CREATE
PROCEDURE
'proc_averaging' (IN feed_name VARCHAR(30), IN feed_interval INT, IN now INT)
BEGIN

DECLARE average FLOAT;
DECLARE avg_table_name VARCHAR(30);
DECLARE avg_table_name_feed VARCHAR(30);
DECLARE feed_now INT;
DECLARE feed_table_id INT;
DECLARE feed_table_name VARCHAR(30);
DECLARE interval_range INT;
DECLARE interval_range_suffix VARCHAR(10);
DECLARE time_old INT;
DECLARE total FLOAT;
DECLARE user_id INT;
DECLARE users_num INT;

-- create interval array for looping
CREATE TEMPORARY TABLE IF NOT EXISTS
  temp_interval_range_suffix (interval_range INT, suffix VARCHAR(10))
INSERT INTO temp_interval_range_suffix (interval_range, suffix)
  VALUES (60', '1mins'), (600', '10mins'), (3600', 'hourly'), (86400', 'daily'));

-- initialise @time_pastloop
IF (SELECT @time_pastloop IS NULL) THEN
  SET @time_pastloop = 0;
END IF;

-- loop in interval ascending order
WHILE (SELECT COUNT(*) FROM temp_interval_range_suffix) > 0) DO
  SET interval_range = (SELECT MIN(interval_range) FROM
                    temp_interval_range_suffix);
  only if interval is bigger than minimum defined & enough
  time elapsed
  VALID: BEGIN
  IF (feed_interval < interval_range) AND (now -
     @time_pastloop > interval_range) THEN
    SET interval_range_suffix = (SELECT suffix FROM
      temp_interval_range_suffix WHERE interval_range =
      interval_range);
    SET time_old = now - interval_range;
    -- for MySQL query
    statements
  END IF;
  -- loop through all of concerned feed type
  SET @SQL = CONCAT("CREATE TEMPORARY TABLE IF NOT EXISTS
    temp_feed_now AS (SELECT DISTINCT('id') FROM feeds
    WHERE 'name' LIKE '%',feed_name,'%');
  
Christophe Meyers
PREPARE stmt FROM @SQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
WHILE ( (SELECT COUNT(*) FROM temp_feed_now) > 0) DO
  SET feed_now = (SELECT MAX('id') FROM temp_feed_now);
  SET users_num = 0;
  SET total = 0;
  --- build list of other than feed_now
  SET @SQL = CONCAT("CREATE TEMPORARY TABLE IF NOT EXISTS
    temp_feed_list AS (SELECT DISTINCT('id') FROM
    feeds WHERE 'name' %",feed_name,"%%) AND 'id' NOT LIKE feed_now")
  PREPARE stmt FROM @SQL;
  EXECUTE stmt;
  DEALLOCATE PREPARE stmt;
  --- loop through list (other than feed_now) & compute average
  WHILE ( (SELECT COUNT(*) FROM temp_feed_list) > 0) DO
  SET feed_table_id = (SELECT MAX('id') FROM
    temp_feed_list);
  SET feed_table_name = CONCAT('feed_','feed_table_id','_'
    ',interval_range_suffix); --- build full feed table name
  SET @SQL = CONCAT("CREATE TEMPORARY TABLE IF NOT EXISTS
    temp_feed_values AS (SELECT data FROM ",feed_table_name," WHERE time<now AND time>="
    time_old")
  PREPARE stmt FROM @SQL;
  EXECUTE stmt;
  DROP TABLE temp_feed_list;
  END WHILE;
  DROP TABLE temp_feed_list;
  SET average = total/total_users_num;
  END IF;
END WHILE;
EXECUTE stmt;

Building Block Apartments Energy Monitoring Interactive System With Social Case Study

DEALLOCATE PREPARE stmt;
IF ( (SELECT COUNT(*) FROM temp_exists) = 0 ) THEN
  SET @SQL = CONCAT("CREATE TABLE ", avg_table_name, " ("
  "time INT UNSIGNED, data float, INDEX ('time')"
  "ENGINE=MYISAM")
)
PREPARE stmt FROM @SQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
SET @SQL = CONCAT("SET user_id = (SELECT 'userid' FROM feeds WHERE 'id'="
  "feed_table_id,')" );
PREPARE stmt FROM @SQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
SET avg_table_name_feed = CONCAT('avg', user_id);
SET @SQL = CONCAT("CREATE TEMPORARY TABLE IF NOT EXISTS temp_avg_exists AS (SELECT * FROM Feeds"
  "WHERE 'name' LIKE ", avg_table_name_feed,"')" );
IF ( (SELECT COUNT(*) FROM temp_avg_exists) = 0 ) THEN
  SET @SQL = CONCAT("INSERT INTO feeds (userid, name,"
  "datatype, public, engine) VALUES (" ,userid ,
  "avg_table_name_feed,"')" );
PREPARE stmt FROM @SQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
DROP TABLE temp_feed_now;
DELETE FROM temp_interval_range WHERE 'interval_range'="
  interval_range";
ELSE
  DROP TABLE temp_interval_range_suffix;
  SET @time_pastloop = now; -- for next call
  LEAVE VALID;
END IF;
END WHILE;
DROP TABLE temp_feed_now;
DELETE FROM temp_interval_range WHERE 'interval_range'="
  interval_range";
ELSE
  DROP TABLE temp_interval_range_suffix;
  SET @time_pastloop = now; -- for next call
  LEAVE VALID;
END IF;
END WHILE;
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

143        -- end of while loop
144        DROP TABLE temp_interval_range_suffix;
145        SET @time_pastloop = now; -- for next call
146
147        END $$
148
149        DELIMITER ;
150
151/>
152 */,-- CREATE NEW EVENT --
153 */
154 DELIMITER $$
155
156
157 CREATE
158 EVENT
159 IF NOT EXISTS
160 'event_averaging'
161 ON SCHEDULE EVERY 1 MINUTE
162 DO BEGIN
163
164        DECLARE feed_name VARCHAR(30);
165        DECLARE feed_interval INT;
166        DECLARE now INT;
167
168        -- Set feed_name & feed_interval --
169        SET feed_name = 'power'; -- keyword to specify => 'name'
170        SET feed_interval = 10; -- seconds!!
171
172        -- time now
173        SET now = (SELECT UNIX_TIMESTAMP());
174
175        -- call to stored procedure
176        CALL proc_averaging (feed_name, feed_interval, now);
177
178        END $$
179
180 DELIMITER ;
Modelado y simulación de una microrred basada en generación fotovoltaica

Resumen

En este proyecto se empieza analizando la definición de microrred, su estado actual y la tendencia en su diseño. Después de esto, se busca realizar una microrred en sus estados de conexión a red y aislado. La fuente de generación utilizada es la solar, mediante la tecnología fotovoltaica. El diseño de la misma se ha realizado para placas de células policristalinas mediante la teoría del diodo doble. El algoritmo utilizado para su control es el del MPPT. Como unidad almacenadora se ha diseñado una batería de Ni-Cd, mediante la simplificación de una de sus celdas a una fuente de tensión no lineal, función de la corriente y el estado de carga. La conexión de todos los componentes se hace mediante el bus de alterna y por consiguiente se ha diseñado un convertidor CA/CC modelado como fuente de tensión. Este mismo convertidor se ha utilizado tanto para las placas PV como para el acumulador o batería.

El control de la microrred se ha basado en la teoría del droop de tensión, la cual relaciona frecuencia y amplitud de tensión con la potencia activa y reactiva. En el modo de conexión a red se ha priorizado utilizar en la menor medida posible la red. En el modo isla de la microrred, se ha modelado un grupo electrógeno para garantizar la alimentación de una parte de la carga considerada prioritaria.

Javier Saiz Anadón
Building Block Apartments Energy Monitoring Interactive System With Social Case Study

Bibliography


[20] CIVIS project. Energy Advices. 2015. [online]. [https://docs.google.com/spreadsheets/d/1Dhb7-7fe4aAdXYJ6NKgjRJdoTvtx06cqXc2RqQDKVIY], 31 Aug. 2015. 75