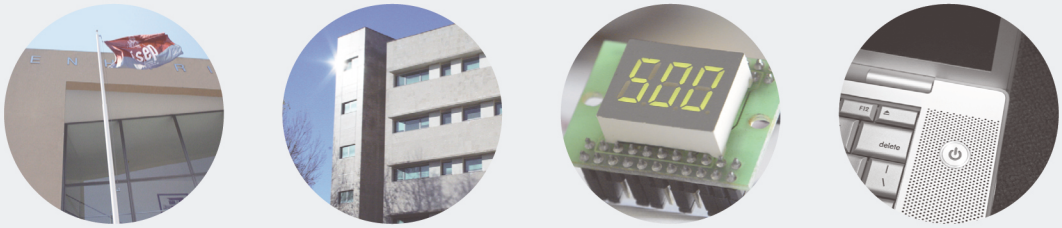
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA ELECTROTÉCNICA - SISTEMAS ELÉCTRICOS DE ENERGIA



Using Matlab and Modbus TCP/IP connection for real-time monitoring of a demonstration microgrid

LOUIS CASTELEYN

Junho de 2016

POLITÉCNICO
DO PORTO

Master degree in Electrical Engineering

Academic Year 2015–2016

Using Matlab and Modbus TCP/IP connection for real-time monitoring of a demonstration microgrid

Louis Casteleyn

Supervisor: Prof. Dr. Z. Vale

Co-Supervisor: P. Faria

Abstract

In cooperation with the research facility GECAD (Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (`http://www.gecad.isep.ipp.pt/GECAD/Pages/Presentation/Home.aspx`)) located next to ISEP (Instituto Superior de Engenharia do Porto), a project was made to retrieve electrical data from the research lab at the facility. This lab is able to simulate small electrical grids (also called micro-grids).

Since electrical energy monitoring has become more important due to environmental and economical causes in the last decades, this has become an interesting subject for studies. Not only quantity, but nowadays also quality (power quality) becomes more important for consumers as well as producers. A good monitoring system is needed in order reach these goals. However, in this paper, only a little part of monitoring will be discussed: the actual retrieving of the data. Complete monitoring holds more content as explained further in the paper.

The main goal is to establish a MODBUS TCP/IP connection between Matlab and 9 Janitza Power Analyzers (UMG96RM-EL) located in the main distribution board of the lab. The retrieved data can be visualised in Matlab or can be exported to Excel for further analysis or manipulation. Also a real-time solution will be provided in the end. In this way the user is able to analyze the different power flows trough the buses connected with the loads making it possible to measure exact energy consumption and losses when simulating a costum-made grid.

2 different approaches were used in Matlab and subjected to speedtests under the same circumstances. Next to speed, another important factor is simplicity and flexibility for future adjustments by others that are not the authors of the code. For this reason regular comments where implemented to facilitate comprehensive reading.

This project was made as an alternative version to a similar project that uses Simulink to get the same results as stated above.

In the last chapter, a case-study is done to verify the correct working of the program and to visualize the results of the simulation.

As a student in electrical engineering, programming is not a big part of the curriculum. Since the project mainly consists of coding, a few weeks where needed in the beginning to understand the Matlab environment. Once this was done, the actual code was written in the following months. The final step was the case study. The results of this were achieved in the final week of the project. The total amount of time spent on the project is 4 months.

Keywords: Matlab; Janitza Power Analyzers; MODBUS TCP/IP; Powerflow simulation; Energy Monitoring; Real-Time Simulation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Matlab

Programs to catch information from energy meters already exist and often come as a package with the actual hardware. For the Janitza power meters for example this is GridVis ®. This is provided for free on the site of Janitza (`http://www.janitza.com/gridvis-en.html`).

However the possibilities of this software are limited and only work with the designated brand of power analyzers. Therefore a different and more flexible or general approach is needed. A solution that can handle different power analyzers from more brands without needing to constantly install other software that basically do the same thing. Matlab provides a more robust environment that can result in a general code to retrieve data from power analyzers. The only condition is that the device has to support MODBUS TCP/IP.

Matlab easily lets the user export the measured data to Excel to visualize or further manipulate the data as wanted. In this way the data is also open to computers where Matlab is not installed. This is important because Matlab requires a valid license.

However, some analyzing requires real-time monitoring. Matlab is able to plot the retrieved data while the program is still executing with a reasonable step size.

The monitoring of the meters can also be expanded to more than 9 meters or more than one distribution board. Future expansion can easily be acquired and this is an important asset in the project.

1.1.2 Energy monitoring

Electricity has become one of the most interesting methods for energy transportation and energy consumption. The usage of it has become a daily routine and is taken for granted. A life without electricity has become unimaginable (lights, heating, electrical appliances). Access to it is a critical asset in the fight against poverty [1], this implies that facilitating this access is an important topic. Good constructed grids and cooperation between different grids can ensure this.

However, the production of electricity still leaves his mark on the environment e.g. nuclear production or production by coal. As stated in [1] the role of coal still plays a big part (41 %) in the world-wide production.

Environmental concerns have become the topic of the latest decades. Even though a lot of technical improvements have been made in reducing wastage and pollution while producing the huge amount of energy that is consumed daily, the most simple solution is to use it in an economical way. For this reason it is important to know what the biggest consumers are and how to use them in the right way. Of course, this is only possible if the correct parameters are measured and interpreted in a good way. A good understanding of energy monitoring is needed.

Nowadays energy monitoring systems are used widely in all kinds of industry to observe the energy consumption. The quantity as well as the quality needs to be monitored (change of voltage, energy consumption, power factor, frequency, current parameters) [21]. Today it is also used in domestic situations. As a result of fast growing developments in electronic and software technologies, this has become possible. People have become aware that monitoring electricity consumption/production can bring economical benefits. Not only by saving on the electrical bill, but interesting projects can be elected to get grants.

With the change of electrical distribution networks the term of smart-grids has become reality[21]. This brings up the task to monitor this in it's own 'smart' way [23]. However in this work only a small asset of total monitoring is treated: the actual retrieving of the data. A more extensive research would contain for example a comparison with the expected consumption for the particular grid where the monitoring is done and eventually an explanation for differences between the actual and expected data. Even more elaborate, guidance on how to reach a particular energy consumption profile could be given.

All this together explains why this topic has become a trend and why it is important to not only industrial users, but also for the small consumer (or prosumer nowadays[1]). If research is funded enough and done in the right way, the future will bring us a lot less waisting of energy and a chance for everybody to manage his own consumer profile, adapted to his/her economical situation.

A rising way of monitoring electrical waves on an electricity grid is **Phasor Measurement Unit**. It allows real-time measurement of multiple remote measurement points on the grid. The result is called a **synchrophasor**[11]. This method is very precise and has accurate time references. The phasors can be transmitted to a local or remote receiver at rates up to 60 samples per second[12]. However this system requires a rather expensive communication network and is a high investment, but it is promising for future applications.

1.2 Background

1.2.1 Matlab

Matlab is a full-featured technical computing environment used by engineers all over the world. The name is derived from the longer 'Matrix Laboratory' (referring to the use of matrices). Built-in graphics make it easy to visualize and gain insights from data. It invites experimentation, exploration and discovery[5]. It allows you to analyze large groups of data and enables the user to deploy algorithms and applications within web, enterprise and production systems. It also allows interfacing with programs written in other languages, including C, C++, Java, Fortran and Python[6].

Matlab uses its own scripting language to create scripts and functions. These functions, created by the user, can be called and executed trough the command window.

In order to have access to the Matlab platform, a valid license key is required. For this particular project, a student license was used. It provides all the toolboxes needed to execute the program.

[1] Prosumer is a combination of a producer and a consumer

It was coded in the build R2015b. Correct functionality can not be guaranteed if used with older builds.

1.2.2 Janitza Power Analyzer

General

The model of the Janitza Power Analyser used is UMG 96 RM-EL. It is manufactured by 'Janitza electronics GmBH'. The UMG 96RM-EL is provided for the measurement of electrical parameters such as voltage, current, power, energy, harmonics etc. for building installations to distributors, circuit breakers and busbar trunking systems. It is suitable for installation in permanent, weatherproof switchboards [16]. The measurement results can be displayed, read and processed over an Ethernet interface. More about the configuration and the connection schemes can be found in[16].

Specifications

The voltage measurement inputs are designed for measuring in low voltage grids in which nominal voltages up to 300V phase can occur. The device measures uninterrupted and calculates all root mean squares over a 10/12-period interval (200ms). It measures true root mean square (TRMS) of the voltages and currents applied to the measuring inputs[16]. The current transformers used are 50A/5A. The voltage transformers used are 400V/400V.

The Janitza Power Analyser is programmable in two ways:

- Directly on the device using two buttons
- Via the programming software Gridvis® as mentioned before.

For this particular project the first option was used. This way the extra download of the software is not required.

Connection

The connection is made trough Ethernet connecting the device to a switch and the particular switch to the computer of the user as seen in Figure 1.1

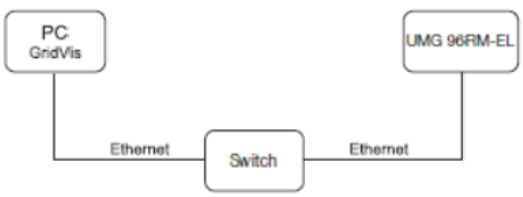


Figure 1.1: Connecting a UMG 96RM-EL to a PC via Ethernet

Chapter 2

MODBUS TCP/IP

2.1 MODBUS Protocol

A short introduction to Modbus is described underneath. More important technical information will be provided for the use of Modbus with a TCP/IP connection (Modbus TCP/IP)[1] in the next section.

Modbus is an application-layer messaging protocol, postioned at level 7 of the OSI model[2]. Modbus has its benefits in flexibility and easy implementation. It can be used for communication with microcontrollers, computers, PLC's and other devices[8][7]. It can be done on serial line as on Ethernet TCP/IP networks.

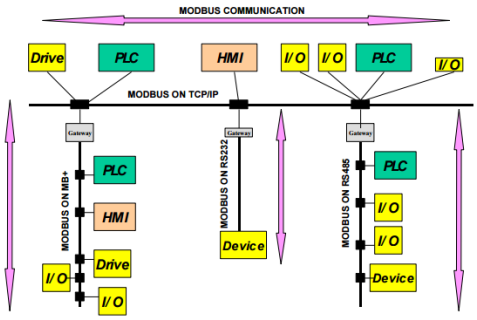


Figure 2.1: Example of MODBUS network architecture

Modbus communication is based on messages and not on the type of physical communication. This means that it can be used on different physical communication systems, connecting old and new as illustrated in Figure 2.1. This also results in a flexible network where an upgrade in hardware doesn't need an entire upgrade of software [13].

Important to know is that Modbus is an open protocol, so it's free to use for anyone. This is also one of the reasons why it has become a standard communication protocol in the industry.

Another communication protocol could be serial communication. The problem with this protocol is the fact that it requires a hard wired, point-to-point cable connection with a limited distance

[1] Transmission Control Protocol

[2] Open System Interconnection model: a conceptual model that characterizes and stadardizes the communication functions of a telecommunication without regard to their underlying internal structure and technology. It provides client/server communication between devices connected on different types of buses or networks [20]

[4]. Every instrument has to be connected with separate cable. However nowadays a lot of improvements can be made for serial communication trough software solutions.

2.2 MODBUS TCP/IP Protocol

A brief explanation of the Modbus messaging system over TCP/IP will be described below. More information can be found in [19] and [7].

2.2.1 Client/Server model

The MODBUS messaging system is based on the Client/Server model between devices connected on an Ethernet TCP/IP network. This model contains four types of messages as seen in Figure 2.2:

- MODBUS Request: message sent on the network by the Client to initiate a transaction.
- MODBUS Confirmation: the request message received on the Server side.
- MODBUS Indication: the response message sent by the server.
- MODBUS Response: the Response message received on the Client side.

Figure 2.2: MODBUS Client/Server model

2.2.2 Protocol Description

PDU

The general MODBUS frame consists of the Protocol Data Unit (PDU) and can be extended to the application data unit (ADU). This is illustrated in Figure 2.3.

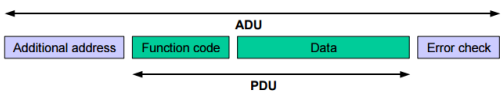


Figure 2.3: General MODBUS frame

This ADU is constructed by the client and defines the initiation of the MODBUS transaction. The function code field of the ADU is coded in one byte and has a range of 1 to 255 decimal. This function code, sent from the Client to the Server, defines the action that has to be performed by the server. As seen in Figure 2.3 the data field of the message contains additional information that can be used by the server to take action (e.g. register addresses, the quantity of items to be handled, the count of the actual data bytes of the field).

If no errors are found, the response from the server will hold the requested data. In the other case, the field contains an exception code that the server application can use to determine the

next action that has to be taken. When the server responds to the client it will use the function code field to indicate whether there is an error or not. In the first case, it will hold an exception response. In the other case it will simply echo the original function code[3]. The model can be found in Figure 2.4.

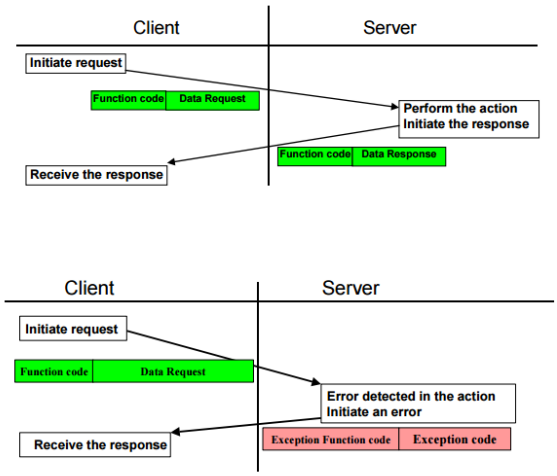


Figure 2.4: Client/Server model with and without error

The MODBUS protocol contains 3 different PDU's:

1. Request PDU {function_code, request_data} with function_code [1 byte] and request_data [n bytes]
2. Response PDU {function_code, response_data} with function_code [1 byte] and response_data [n bytes]
3. Exception Response PDU {exception-function_code, exception_code} with exception-function_code [1 byte] and exception_code [1 byte]

Data Encoding

Of further importance is the fact that MODBUS uses a 'big-Endian' representation for adresses and data items. Meaning that when a message larger than a byte is transmitted, the most significant byte will be sent first.

ADU

As mentioned before, the ADU (Application Data Unit) holds the PDU (Protocol Data Unit) as well as some additional fields. If this is carried on TCP/IP network it this means it will have an MODBUS Application Protocol header or MBAP (Figure 2.5).

This MBAP header contains 4 fields:

1. Transaction Identifier
2. Protocol Identifier

[3]Contains the original function code with its most significant bit set to a logic 1

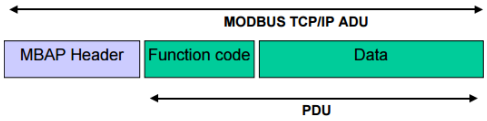


Figure 2.5: MODBUS TCP/IP frame

3. Length
4. Unit Identifier

The **Transaction Identifier** contains 2 bytes and serves as the identification of the MODBUS Request/Response transaction. It is initialized by the client and recopied by the server from the received request.

The **Protocol Identifier** contains 2 bytes and is by convention 0 for the MODBUS protocol. It serves for intra-system multiplexing[4]. It is initialized by the client and recopied by the server from the received request.

The **Length** contains 2 bytes and indicates the number of bytes yet to come. It is initialized by the client for the request and by the server for the response.

The **Unit Identifier** contains 1 byte and is used for the identification of a remote slave connected on a serial line or on other buses. It is initialized by the client and recopied by the server from the received request. Because the Server is addressed on MODBUS TCP/IP with the IP address, the unit identifier is useless. The value 0xFF (255) can be used.

After the MBAP header, the actual MODBUS request follows. It contains the function code, the starting register adress[5] and the quantity of registers. The function code will specify what action the Server has to take and can be found in [19]. For this paper function code 3 will be used[6].

Further, the registered port **502** is used for all MODBUS TCP/IP communication.

TCP/IP Connection Management

A MODBUS TCP/IP communication can be established in two different ways.

1. By the User Application Module. In this case, an application-programming interface has to be provided that can manage the entire connection. It offers more flexibility but a good understanding of the protocol is needed. This will be the case using Matlab.
2. Automatically by the TCP connection management module. In this case the connection is hidden from the user application.

Further there exists a set of implementation rules as defined in the MODBUS TCP/IP Implementation Guide[19].

1. Without explicit user requirement, it is recommended to implement the automatic TCP connection management.
2. It is recommended to keep the TCP connection opened with a remote device and not to open and close it for each MODBUS/TCP transaction.

[4]Combining different analog or digital signals to one signal
[5]This can be for example '800' to retrieve the frequency measured by one of the Janitza analyzers
[6]Function code 3: read multiple holding registers, readable, 16 bits, essentially measurements and statuses.

3. It is recommended for a MODBUS Client to open a minimum of TCP connections with a remote MODBUS server (with the same IP adress). One connection per application could be a good choice.

4. Several MODBUS transactions can be activated simultaneously on the same TCP connection.

5. In case of bi-directional communication between two remote MODBUS entities (each of them is client and server), it is necessary to open separate connections for the client data flow and for the server data flow.

6. A TCP frame must transport only one MODBUS ADU. It is advised against sending multiple MODBUS requests or response on the same TCP PDU.

Communication trough MODBUS TCP/IP follows 3 steps:

The first step is to **establish the connection**. To exchange data with a remote server, the messaging system must open e new client connection with a remote port 502. The local port[7] must be higher than 1024 and different for each client connection.

The second step contains the **actual data exchange**. As illustrated in figure 2.4, a request has to be send. The IP adress is hereby used to find the remote device in the network. It is important that de connection has to be kept open untill communication is complete.

In the last step, the **connection will be closed**.

The build of a MODBUS Request

Again three steps are involved in building a MODBUS Request. A flowchart is shown in Figure 2.6.

1. The **Instantiation** of the transaction that enables the Client to memorize the required information for the message.

2. The **Encoding** of the request (PDU + MPAB header as explained in figure 2.5). When all fields are filled, the ADU is built connecting both the the PDU with the MBAP header.

3. The **Sending** of the request to the TCP management module which has the responsibility to find the right socket[8] towards the remote Server. In addition to this, the IP address of the remote Server has to be transmitted too.

[7] Port of the client
[8] Socket is the communication-endpoint of a network

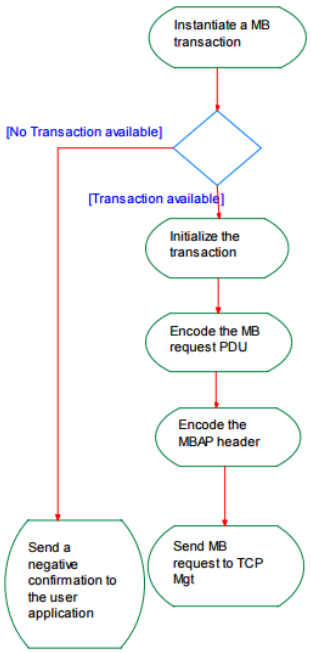


Figure 2.6: Client/Server connection

To illustrate the encoding, an example (Figure 2.6) is given as mentioned in [19].

Table 2.1: Example of a MODBUS request with MBAP Header

	Description	Size	Example
MBAP Header	Transaction Identifier Hi	1	0x15
	Transaction Identifier Lo	1	0x01
	Protocol Identifier	2	0x0000
	Length	2	0x0006
	Unit Identifier	1	0xFF
MODBUS request	Function Code	1	0x03
	Starting Address	2	0x0004
	Quantitiy of Registers	2	0x0001

The Transaction Identifier, located in the MBAP, will be used to associate the response with the original request that was sent. If this Transaction Identifier refers to a pending transaction, the response must be parsed by verifying the MPAB Header and the MODBUS PDU response. Next follows the verification of the Protocol Identifier that must be 0x0000 and the length of the MODBUS response. The Unit Identifier is not needed if the Server that gave the response is connected directly to the TCP/IP network. On the other hand, if it is connected trough a bridge, router or gateway the value 0xFF is mandatory.

Regarding the PDU, the function code must be verified and the response format must be analyzed according to the MODBUS Application Protocol. If the function code is the same as the one used for the request and the response format is correct, the response is given as a **positive confirmation**. It gives confirmation that the request has arrived at the Server and has been responded by the Server.

If the function code is a MODBUS exception code (as mentioned in 2.2.2). The MODBUS excep-

tion response wil be given as a **positive confirmation** as well. This exception code means the request has arrived at the Server but it failed to perform the requested action.

The function code can also be different from the one used to send the request or if the format is not correct. An error will be signaled using a **negative confirmation**.

Chapter 3

Algorithm for data retrieval

In this paper, 2 different main algorithms were coded to identify the faster one.

1. The first attempt exists of just 1 m-file[1]. Everything is coded in 1 file. This program will be named *RetrieveData1*.

2. The second attempt uses multiple m-files in combination to retrieve the same result. This program will be named *RetrieveData2*.

Both attempts were subjected to speed tests to identify the faster working program.

3.1 Configuring the TCP/IP connection between the analyzer and Matlab

3.1.1 Configuration of the Janitza Power Analyzer

As discussed before, a TCP/IP connection uses the IP address to find devices connected to the network. For this reason it is important to know the exact address of the 9 meters installed in the main distribution board of the lab. To identify the address of each meter the two buttons on the front of the analyzers can be used.

[1] m-files are files used by Matlab to store functions or scripts

Figure 3.1: Front of the Janitza Power Analyser

Pressing both buttons at the same time will result in the program menu. Cycling trough the different parameters is done by using the right button. Changing the parameters can be done by using the left button. In this way the IP address can be identified or changed as wished. It is important that every analyzer has his own unique address. More about the programming mode can be found in[16].

3.1.2 Configuration of Matlab for RetrieveData1

NOTE: The appendix is not published in the digital version. Please contact GECAD for the full appendix.

To establish the TCP/IP connection, the code in Appendix A is used, based on [22]. The main information will be in the code as comments. More commentary will be given in this section.

- Declare the function that connects Matlab to the Janitza Power Analyzer via a TCP/IP connection. The user can choose the amount of registers that have to be read. Because of this, the amount of input and output arguments is variable (`varargout` and `varargin`). A complete list of the register addresses can be found in [15, pag.18-19]. An example for this function could be `RetrieveData1(800,802,804)` in the Matlab command window retrieving the frequency, the zero voltage sequence and the negative voltage sequence.
- Start timer.
- Create 9 variables that are used as the IP addresses of the analyzers.
- Actual code to open the connection specifying the IP address and port (502).
- Create a tcpip object for every IP address.
- Define the input buffer size (512 by default). This means 512 bytes can be read at a time which is enough for this application.
- Try to open the connection and display success or failure.
- Define the parameters of the message as illustrated in section 2.2.2. This creates the MBAP Header.
  - The Transaction Identifier is chosen 1.
  - The Protocol Identifier is zero for MODBUS.
  - The length of the remaining part, a total of 6 bytes[2] has yet to come.
  - The Function Code that will be used is 3 [7, pag. 11]: read multiple holding registers.
- Make 2 8-bit words into a 16-bit word.
- Initialize variable a1
- Initialize 2 cell arrays with number of rows = `nargin` (number of input arguments) and 1 column. For example, in the case with the three address registers as mentioned above this would create an array with three rows, 1 column.
- While function to do this for all input arguments.
- Create the actual MODBUS Request. For Function Code 3 this will contain the adress of the first register to read (16 bit) and the number of registers to read (16 bit).

  Create variable `Add` that will define the starting address and convert it to an unsigned 16-bit integer. In the first loop, this will be the first element in the array. In the example this will be `800`. Next create variable `Val` that will define the number of registers that have to be read and convert it to an unsigned 16-bit integer. In this case 2 is the right value since the registers defined in [7] go as pairs.
- Concatenate the different parameters in one array that will hold the message.

[2] UnitID (1), FunCod (1), Add(2), Val(2)

- Use the Matlab function `fwrite` to send the message to the tcpip object in a 16-bit integer format.
- Use the Matlab function `fread` to read the amount of bytes available in the response message of the tcpip object and store in the variable `res`. The response frame will consist of 13 bytes from which the last 4 bytes will contain the actual measured data.
- Create a word of the last 4 bytes of the variable `res` and store it in the variable `result`. This is done by the Matlab function `bitshift`. Take byte 10 from `res` and shift it to the left with 8*3 = 24 bits or 3 bytes. A word with 4 bytes has been created with the particular byte as most significant byte. Do the same with byte 11, 12 and 13 (with different number of bitshift) to create a word consisting of the four particular bytes.
- The response answer is in the single-precision floating-point format. This part of the code converts the response to a decimal format understandable for the user. This part is courtesy of Roger Stafford and published on the MathWorks forum[2].
- Variable y1 is the response in decimal format. Put this in the first element of the output array.
- Put the input argument (address register) of this loop in cell array A in the element of the number of the loop (e.g. in the first loop this will be in the first element of the cell array).
- Do the same with the output in cell array B.
- Horizontally concatenate cell arrays A and B into cell array C and display C[3] as seen in Figure 3.2.

```
C =

    [800]    [49.9748]
    [802]    [ 0.6515]
    [804]    [ 0.1297]
```

Figure 3.2: Output C

- Close the connection with the tcpip object with the Matlab function `fclose`.
- Stop the timer. This time gives the total execution time for all data to be retrieved.
- End loop of line 23.
- End function `RetrieveData1`.

A total of 9 cell arrays (1 for each meter) will be displayed in the final output in the form illustrated in figure 3.2. Each array will have as much rows as input arguments (address registers) and 2 columns (address and value).

[3]";" suppresses the output of the line, in this case the output is requested

3.1.3 Configuration of Matlab for RetrieveData2

As mentioned in the beginning of Chapter 3 this version uses a combination of more than 1 m-file, 10 to be precise. These are the following:

1. GetIP1
2. GetIP2
3. GetIP3
4. GetIP4
5. GetIP5
6. GetIP6
7. GetIP7
8. GetIP8
9. GetIP9
10. RetrieveData2

The first 9 m-files are functions[4] and will do the same as RetrieveData1.m but only with one analyzer at a time. An array with the requested registers will be created. The code can be found in Appendix B

- Declaration of the function `GetIP1` with input argument `(in)` that is declared in line 3 of RetrieveData1.m and output array `A`.
- Declaring a fixed length of the variable `A` (Preallocating[5]).
- Use the Matlab function `numel` to determine the amount of elements in the input array.
- For loop to use all the elements in the input array.
- Store the values of the response in the correct place of the array `A`.

The last m-file is a script that will order the data in arrays alike RetrieveData1.m. Again the example with 3 input address registers will be used (800,802,804). The code can be found in Appendix C.

- Create array with input address registers.
- Transfer array to variable A0.
- Run all 9 functions (9 analyzers) with as input the array stored in the variable `in` and store in variables.
- Horizontally concatenate the two arrays. This will give the following output as in Figure 3.3:

[4] notice the declaration in the beginning with `function`
[5] Good coding implies the use of preallocation

```
A =

  800.0000  802.0000  804.0000   49.9800    0.2164    0.0696
```

Figure 3.3: Output A

- Reshape array to become an output alike RetrieveData1.m (Figure 3.4):

```
AS =

  800.0000   49.9800
  802.0000    0.2164
  804.0000    0.0696
```

Figure 3.4: Output AS

3.1.4 Speedtests

The speedtests performed here are an attempt to determine the fastest program. **It is an indication but certainly not conclusive**. This because the timing is not always accurate because of 2 parameters:

1. The speed of Matlab fluctuates due to background processes on the PC of the user. For this reason, as many backbground processes as possible have to be shut down before beginning tests.
2. The time also fluctuates due to differences in traffic on the network. Best results are obtained when a low amount of traffic is active (e.g. when the user is the only one connected to the network).

Because of this, a number of speedtests will be done. The results are displayed in table 3.1. The measured times are only used for comparison and are not absolute (will fluctuate with different networks). The tests will be done for a total of 25 registers (registers that have been found useful for the paper).

The commands for both programs in the command window will be:

- `RetrieveData1(800,802,804,806,808,810,812,814,816,818,820,822,824,926,928,930,932,934,936,868,870,872,884,886,888)`
- `RetrieveData2`

Table 3.1: Speedtest

RetrieveData1	RetrieveData2
1.119435 s	1.407517 s
1.208404 s	1.323759 s
1.107682 s	1.261952 s
1.225675 s	1.326995 s
1.092705 s	1.711060 s

It can be concluded that overall RetrieveData1.m is the faster program, even with the extra loop[6]. Possible explanations are that the use of several m-files takes more time. It is not the aim of this paper to figure out how Matlab handles different code and how it affects the timing.

However, if the data has to be exported to Excel. RetrieveData2.m is easier to implement. It holds more code, but is easier to understand and the readability is better. Also, the speed is not remarkable worse. The next section will describe the expansion of RetrieveData.2 and implement Excel as output software.

For real-time monitoring, RetrieveData1.m will be used since this is the fastest and will be able to have a slightly smaller step size.

[6]Loops are in general a time consuming process

Chapter 4

Expansion of RetrieveData2 for export to Excel

As mentioned before, the code in this chapter will be used to export data to Excel. It can be used for database purposes.

4.1 Configuration of the grid

The grid in the lab can be configured in 4 ways as illustrated in figure 4.1.

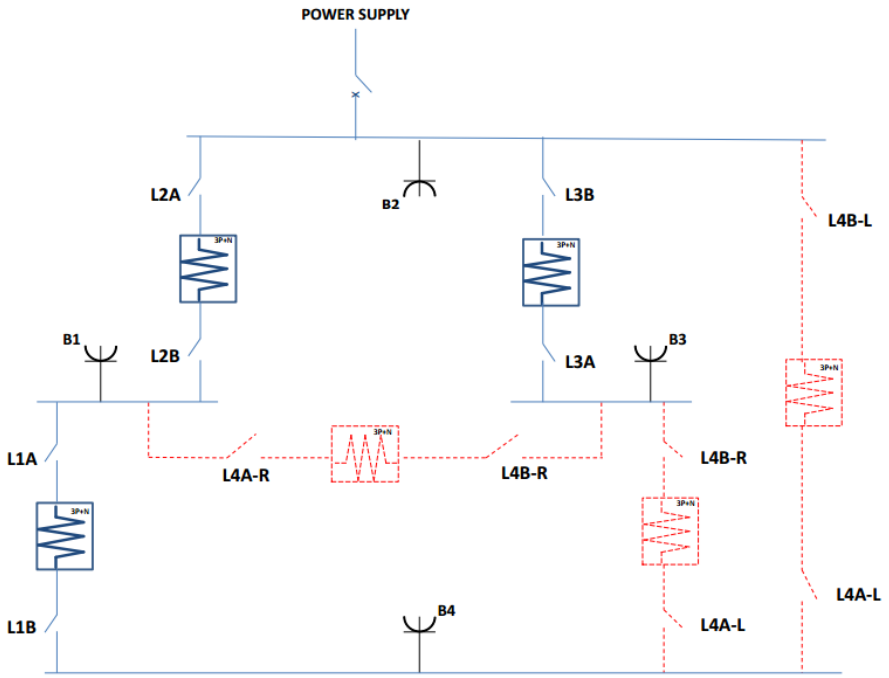


Figure 4.1: Configuration of the grid

The resistor elements in the scheme simulate the lines. There are 4 output buses where loads can

be connected to (B1, B2, B3 and B4). L1A, L1B, L2A, L2B, L3A and L3B are power analyzers. L4A_1, L4A_2, L4B_1 and L4B_2 are power analyzers as well but their connection can be adjusted in a way illustrated by the dotted lines in Figure 4.1. A total of 4 situations are possible in this way.

- L4 not connected to the grid, this program will be called RetrieveData1a.m
- L4 connected between B1 and B3, this program will be called RetrieveData1b.m
- L4 connected between B3 and B4, this program will be called RetrieveData1c.m
- L4 connected between B2 and B4, this program will be called RetrieveData2d.m

4.2 Retrieving the data

The code used in the rest of the chapter is RetrieveData2b.m in Appendix D.1. RetrieveData2a.m, RetrieveData2c.m and RetrieveData2d.m are also added in the Appendices.

```
1 %Scheme with L4 connected between B1 and B3
2 in = [800 802 804 806 808 810 812 814 816 818 820 822 824 926 928 930 932 934 936 868 870 872 884 886 888];
```

- Declare the address registers that are wanted, these are chosen as the most useful.
- The data is retrieved 3 times to check the variety in time (tic and toc) and exclude outliers. The order of the GetIP functions are optimized for voltage drop calculations. This means that the analyzers in the beginning and at the end of the lines will be addressed after each other in pairs.

4.3 Creating tables

This code makes tables with the address register and the 3 values that have been retrieved for this address.

```
66 TA = array2table(AS,'VariableNames',{'Register_Adress' 'Value1' 'Value2' 'Value3'});
```

- The Matlab function `array2table` is used for this. The argument `VariableNames` is used to display the titles of the variables after the comma. These titles are `Register_Adress`, `Value1`, `Value2` and `Value3`.

4.4 Voltage Drops

Next the voltage drops between the beginning and the end of the lines are calculated.

```
77 %%Voltage drops
```

- Take the rows 5 - 7 (addresses for the voltages of the lines: L1, L2, L3) and the columns 2 - 4 (the 3 values) of the array BS and store them in the array VB.

- Do the same with CS.
- Calculate the voltage drop between VB and VC.
- Make a table with the information.

This is repeated for the other analyzers.

4.5 Complex currents

```
100 %%Complex currents
```

In this section the complex currents that flow trough the buses will be calculated.

- Define an array with the real parts of the complex current. Take row 14 - 16 (addresses for the real parts of the currents of the lines: L1, L2, L3) from column 2 - 4 from array AS.
- Do the same with the imaginary part.
- Define an array existing of complex numbers using the real and the imaginary parts as stated before.

This is repeated for all analyzers.

4.6 Power

In this section the power flow through the buses will be calculated.

```
139 %%Power (P&Q&S)
```

- Define an array with the active power P.
- Define an array with the apparent power (magnitude) S.
- Calculate the reactive power Q. $S = \sqrt{P^2 + Q^2}$ or $Q = \sqrt{S^2 - P^2}$.

4.7 Bus calculations

In this section the current and power flows trough the 4 buses will be calculated. This part differs depending on the configuration of the grid. The calculations are made based on **Figure 4.1**.

```
187 %%Bus calculations
```

- Apply Kirchoff's law at bus 1 and store in a variable. Take the magnitude.
- Store the complex value in a variable, this will be used later to plot the vectors.

- Analyze the power flow of bus 1 (active and reactive) and store in a variable. With the apparent power, the Matlab function `abs` is used to calculate the magnitude of the complex number.
- Convert the information into a table.

The remaining lines will calculate the other buses in a similar way.

4.8 Losses, Time and Vectors

In this section, 3 things will be done:

- The losses due to the currents trough the lines will be calculated.
- The time for every retrieval (see tic toc) will be calculated.
- The vectors of the complex currents will be plotted.

```
246 %%Losses
```

- Subtract all the bus power flows from the feeder flow to obtain the total losses and convert the result into a table.
- Calculate the power losses over the lines.
- Collect the times and convert the results into a table.
- The use of `subplot(rows,columns,number of graph)` will plot the 4 graphs into one picture. The Matlab function `compass` is used to plot complex vectors. The argument will define what complex numbers to graph. Also an argument with the color can be added.

4.9 Excel

This part of the code will write the tables that have been made in the previous lines to a file in Excel. The Excel file will be stored in the working directory.

```
280 %%Write to Excel
```

- Define the filename of the Excel file.
- The Matlab function `writetable` is used to write the tables to Excel. `TA` is the name of the table that has to be written. `filename` is the name of the file that will be used, in this case it is `Measured_Values2b.xlsx`. `'Sheet',1` will define the sheet of the file that has to be used. The number `'A2'` will define in what cell the table has to begin.
- To write individual words to a cell the Matlab function `xlswrite` has to be used. The use of `filename` is the same. What follows is the text that has be written in the cell. `'Blad1'` is the name of the Excel sheet, in this case it is the Dutch translation of 'Sheet1'. The last code has the same usage as in the previous lines.

- One extra argument is used to include the names of the rows of the table: `'WriteRowNames', true`.

The remaining lines are coded in a similar way.

Chapter 5

Expansion of RetrieveData1.m for real-time monitoring

RetrieveData1 will be used to monitor the data of the analyzers in real-time. 2 programs can be used:

- RealTimeMeter.m will allow the user to track the data of 1 meter by plotting the L-N voltage, the L-L voltage, the current, the active power and the power factor as registered by the analyzer.
- RealTimeBuses.m will display the current and the active power of the buses and the losses over the lines connecting those buses. However, this implies that data of all the meters have to be retrieved resulting in a reduction of execution speed. The step size will have a minimum value that has to be respected. As for the configuration, the scheme as in RetrieveData2a.m is used.

5.1 RealTimeMeter.m

This code can be found in Appendix E.

- The user will have to define a simulation `time`, a `step_time` and a series of addresses as fixed values. It is important for the correct working of the code that the fixed addresses are `[808,810,812,814,816,818,926,928,930,932,934,936,868,870,872,820,822,824]`. These are the values the will be plotted. The input argument `varargin` provides the possibility to send more addresses. These values will not be plotted but can be given in the output. A possible correct command in the command window could be for example: `RealTimeMeter(30,1,[808,810,812,814,816,818,926,928,930,932,934,936,868,870,872,820,822,824],880,890)`.
- Close all windows that could have been opened in a previous run.
- Initialize a variable that will count the step of the measurement.
- Use of the Matlab function `num2cell` to convert the array 'addresses' into a cell array. This has to be done because `varargin` is by standard a cell array.
- Horizontal concatenation of all the addresses used by the user as input.
- Use of the Matlab function `numel` to count the number of elements in the cell array.

- The time divided by the step time will define the number of data points. This has to be an integer because the input for an array (R on line 19) has to be an integer as well. A good example could be time: 20 s and step time: 1 s or time: 30 s and step time: 0.5 s. A bad example would be time: 10 s and step time: 3 s.
- Use of the Matlab function `zeros` to initiate a 2-D array with a number of rows equal to the number of input addresses and a number of columns equal to the amount of data points (r). This is the use of preallocation.
- Start of the timer.
- Initialize the time variable `t1`.
- While loop to keep executing the code as long as the input time of the simulation is not exceeded.
- Store time in variable `t2`. This will be used to define the time that will have to be paused at the end of the loop to respect the step time.
- Use the Matlab function `cell2mat` to convert a normal array to a cell array.
- B contains all of the response values. The value of B changes with every count of `temp`. The columns of R will be stored with this array of response values until all columns are filled, this is when the simulation time is over.
- Store the time in variable `t1`. This time will be used as the value on the time axis of the plot to indicate the time of the measurement.
- Open a new figure window where the plot will be in displayed. Only do this the first time, that is if the variable `temp` equals 1.
- Use the Matlab function `subplot` to plot the first graph of a window with a total of 5 graphs (2 rows, 3 columns).
- Use the Matlab function `drawnow` to update figures.
- Use the Matlab function `hold on` to prevent the deletion of the previous plots when the next loop starts.
- Scatter plot the value stored in the first, second and third row of the column with the number equal to the value of temp. The x-coordinate is the time `t1` (line 109). The values 1,2 and 3 are where the L-N voltages of the lines are stored. The scatters will be blue, green and yellow as indicated by 'b', 'g' and 'y'.
- Add the title of the graph.
- Label the axises.
- Insert a legend. Note that this will not be executed correct in build R2015b because of a known bug. A patch that can be downloaded on `http://www.mathworks.com/support/bugreports/1283854` is needed in order to work around this.

  The text of the legend is L1, L2 and L3. The location is defined as `bestoutside` (outside the axis) and with a vertical orientation.
- The current can't be plotted right away. The real and imaginary parts have to be combined to their complex form.
- An extra point will be plotted in the color black with the total current.
- Store the end time in variable `t3`.
- t3 - t2 is the time that the loop takes to execute. Depending on the network traffic, this is in general smaller than 0.5s. In order for the code to respect the step size, the loop has

to be faster then the step. For that reason, 0.5 s or more is recommended. In this way the pause time will be positive and the step will be respected.

- Use the Matlab function `hold off` to cancel the `hold on` that was used before.

5.2 RealTimeBuses.m

The first lines of code are similar to the ones used in RealTimeMeter.m. The actual code can be found in Appendix F.

- Initialize a 3-D array, same as in section 5.1, but with one extra dimension representing the 9 meters.
- Calculation of the power losses over the lines.
- As this program retrieves all the meters, it takes a longer time to execute a loop. For this reason a higher step size is recommended e.g. 1 s.

Chapter 6

Case Study

To verify the correct operation of the program, a case study was done at the end of the project. 3 loads were being connected to 3 buses (B1, B2 and B4). A full micro-grid simulation with the 4 loads connected was not possible due to works on the distribution board. The loads were the following:

- A 4kVA three phase variable load to B1
- A 1kW resistive load with ventilation to B2
- A 0.37 kW induction motor to B4

Since not all meters work, not all bus calculations are correct. The images below are just an example of the visualization. For this reason only the voltage drop and the losses will be shown over distribution line 1 (L1A - L2A). The values of the other lines can not be calculated without the other meters working properly.

6.1 RetrieveData2a.m

The program RetrieveData2a.m was tested while all 3 loads were connected.

Response Values

192.168.2.245			
Register_Adress	Value1	Value2	Value3
800	49,98336	49,98278	49,98257
802	9,529582	8,410503	9,536617
804	1,538668	1,159498	1,453292
806	239,8911	239,3107	239,8918
808	243,6449	243,7583	244,6787
810	247,3253	244,7077	246,3866
812	229,1308	229,8603	229,0406
814	418,1567	416,5489	418,0724
816	414,9383	413,9664	414,7288
818	413,6801	413,2353	413,9767
820	0,818528	0,815798	0,820657
822	0,897704	0,893402	0,896413
824	0,787805	0,79131	0,79059
926	2,233116	2,237041	2,2337
928	2,500355	2,49389	2,502001
930	0,291946	0,289046	0,302207
932	1,580093	1,586886	1,580647
934	1,872646	1,805067	1,852806
936	2,710116	2,706884	2,70567
868	548,2307	556,0759	552,1436
870	695,9455	668,3113	689,9892
872	494,5363	497,4461	495,4691
884	685,5945	695,3151	688,8781
886	789,8026	765,3833	784,1429
888	641,0821	640,4999	640,5579

192.168.2.246			
Register_Adress	Value1	Value2	Value3
800	49,9831	49,98284	49,9834
802	8,598277	8,249031	8,493142
804	1,591722	1,209503	1,633781
806	239,5815	238,7919	239,604
808	229,5777	229,7265	229,6809
810	245,9065	241,5178	245,986
812	243,6475	245,5226	243,5276
814	414,2668	412,3346	414,3274
816	417,7341	415,7482	417,8332
818	413,1569	412,9657	413,1142
820	0,19853	0,178818	0,197609
822	0,236807	0,237568	0,237187
824	0,123358	0,14201	0,123204
926	0,158644	0,155506	0,17664
928	0,83491	0,821383	0,84897
930	0,696602	0,687149	0,691196
932	0,819885	0,827102	0,821854
934	0,271309	0,264046	0,25469
936	0,548659	0,562794	0,568474
868	38,03824	36,84039	38,1308
870	51,0713	51,10625	51,6652
872	26,5893	28,20736	26,7943
884	191,7424	193,1581	193,0908
886	215,9102	215,1885	218,0568
888	214,6419	218,4086	217,9969

192.168.2.250			
Register_Adress	Value1	Value2	Value3
800	49,98313	49,98273	49,98332
802	7,726542	8,63953	8,717597
804	1,567512	1,52194	1,606416
806	239,093	239,2417	239,3846
808	230,0224	229,2014	229,2237
810	244,9939	245,0455	245,6162
812	242,6182	243,87	243,7066
814	413,6755	413,5526	413,8909
816	416,7772	417,0614	417,4233
818	412,1644	412,7761	412,8184
820	0,192419	0,186341	0,188705
822	0,231301	0,236433	0,236944
824	0,124586	0,123955	0,121367
926	0,153758	0,153945	0,173994
928	0,832885	0,837517	0,846583
930	0,695953	0,701143	0,689486
932	0,819919	0,828441	0,823144
934	0,274091	0,270806	0,251206
936	0,545487	0,556146	0,570931
868	36,89315	35,9573	36,36887
870	49,63909	50,94389	51,34699
872	26,66727	26,99109	26,41258
884	191,9135	193,1547	192,873
886	214,8464	215,7258	216,9283
888	214,5979	218,3058	218,2224

Figure 6.1: The response values of 3 of the 9 meters

800	Measured frequency
802	Voltage, zero sequence
804	Voltage, negative sequence
806	Voltage, positive sequence
808	Voltage U1 L1-N
810	Voltage U2 L2-N
812	Voltage U3 L3-N
814	Voltage U1 L1-L2
816	Voltage U2 L2-L3
818	Voltage U3 L3-L1
820	Fund. Power factor, CosPhi;ULN,IL1
822	Fund. Power factor, CosPhi;ULN,IL2
824	Fund. Power factor, CosPhi;ULN,IL3
926	Current, real part I L1
928	Current, real part I L2
930	Current, real part I L3
932	Current, Imaginary part I L1
934	Current, Imaginary part I L2
936	Current, Imaginary part I L3
868	Real power P1 L1-N
870	Real power P2 L2-N
872	Real power P3 L3-N
884	Apparent power S1 L1-N
886	Apparent power S2 L2-N
888	Apparent power S3 L3-N

Figure 6.2: The address registers

Figure 6.1 shows the adresses that have been found useful and the three values. The time spent to retrieve each value can be found in Figure 6.6. Figure 6.2 shows what the addresses stand for.

Voltage Drops

Voltage Drops

Row	Value1	Value2	Value3
L1-N	-0,44464	0,525146	0,457245
L2-N	0,912613	-3,52777	0,369781
L3-N	1,029297	1,652573	-0,17899

Figure 6.3: Voltage drops over transportation line 1

Bus Parameters

Bus1

I [A]

Row	Value1	Value2	Value3
L1	1,035515	1,028939	1,017616
L2	1,250908	1,258535	1,25543
L3	1,153514	1,163153	1,154972

P [W]

Row	Value1	Value2	Value3
L1	154,2446	154,6465	153,7059
L2	148,0445	149,4508	149,068
L3	179,5408	177,7672	177,6361

Q [VAR]

Row	Value1	Value2	Value3
L1	204,874	202,5189	204,7293
L2	210,1042	210,0761	205,4168
L3	196,244	190,2642	195,2911

S [VA]

Row	Value1	Value2	Value3
L1	256,4464	254,8126	256,007
L2	257,0232	257,813	253,8057
L3	265,9823	260,3875	263,9947

Bus2

I [A]

Row	Value1	Value2	Value3
L1	2,735598	2,742728	2,736395
L2	3,123873	3,078596	3,113342
L3	2,725796	2,722273	2,722495

P [W]

Row	Value1	Value2	Value3
L1	548,2307	556,0759	552,1436
L2	695,9455	668,3113	689,9892
L3	494,5363	497,4461	495,4691

Q [VAR]

Row	Value1	Value2	Value3
L1	411,6831	417,4239	411,935
L2	373,4273	373,057	372,5521
L3	407,9461	403,4694	405,9863

S [VA]

Row	Value1	Value2	Value3
L1	685,5945	695,3151	688,8781
L2	789,8026	765,3833	784,1429
L3	641,0821	640,4999	640,5579

Bus3

I [A]

Row	Value1	Value2	Value3
L1	0	0	0
L2	0	0	0
L3	0	0	0

P [W]

Row	Value1	Value2	Value3
L1	0	0	0
L2	0	0	0
L3	0	0	0

Q [VAR]

Row	Value1	Value2	Value3
L1	0	0	0
L2	0	0	0
L3	0	0	0

S [VA]

Row	Value1	Value2	Value3
L1	0	0	0
L2	0	0	0
L3	0	0	0

Bus4

I [A]

Row	Value1	Value2	Value3
L1	0,834211	0,842623	0,841332
L2	0,876826	0,88021	0,883067
L3	0,884254	0,894931	0,895183

P [W]

Row	Value1	Value2	Value3
L1	36,89315	35,9573	36,36887
L2	49,63909	50,94389	51,34699
L3	26,66727	26,99109	26,41258

Q [VAR]

Row	Value1	Value2	Value3
L1	188,334	189,7783	189,4131
L2	209,0333	209,6243	210,7638
L3	212,9345	216,6308	216,6181

S [VA]

Row	Value1	Value2	Value3
L1	191,9135	193,1547	192,873
L2	214,8464	215,7258	216,9283
L3	214,5979	218,3058	218,2224

Figure 6.4: I, P, Q and S of the 4 buses

Current Vectors

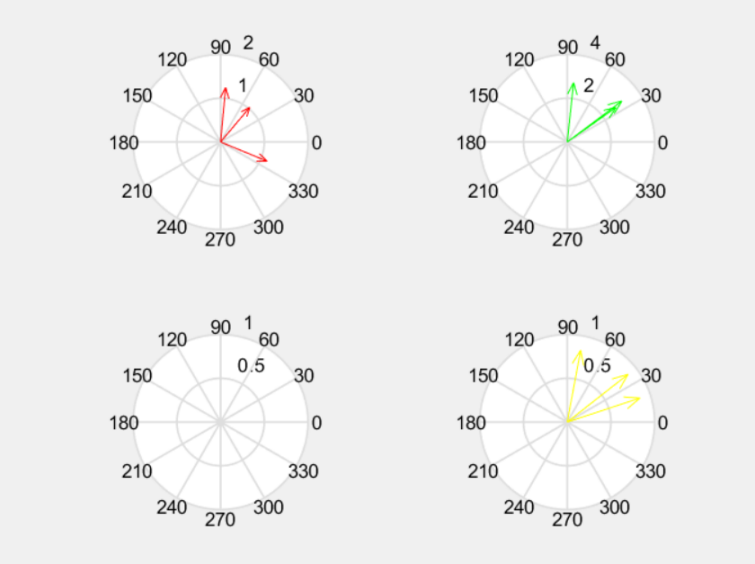


Figure 6.5: Current Vectors of B1, B2 and B4

As seen in Figure 6.5 bus 3 has no current because there is no load connected. The exact length of the vectors can be found in Figure 6.4. For bus 1 this will be around 1 A, for bus 2 this will be around 2 A. Bus 4 will be below 1 A. In Figure 6.5.

Time

Row	Value1	Value2	Value3
Time [s]	1,431355	1,357132	1,337463

Figure 6.6: Time needed for the three communications

Line Losses

Line 1 Power Loss [W]

Row	Value1	Value2	Value3
L1	1,145088	0,883091	1,761929
L2	1,432209	0,162361	0,318211
L3	-0,07797	1,216267	0,381721

Figure 6.7: Losses over transportation line 1

As seen in figure 6.3 and figure 6.7 some of the values seem to be negative. This is simply because the current is not great enough to cause a significant voltage drop that is measurable by the meters. The accuracy of the analyzers can be found in [16, pag.72-73].

6.2 RealTimeMeter.m

RealTimeMeter.m addresses 1 analyzer. In this case, the IP of the power supply analyzer was used. The step size was taken 1 s, but because of bad connection status at the time of the test, this step could not be respected. In this case, the program will run it as fast as possible with the particular connection.

1 kW resistive load with ventilation

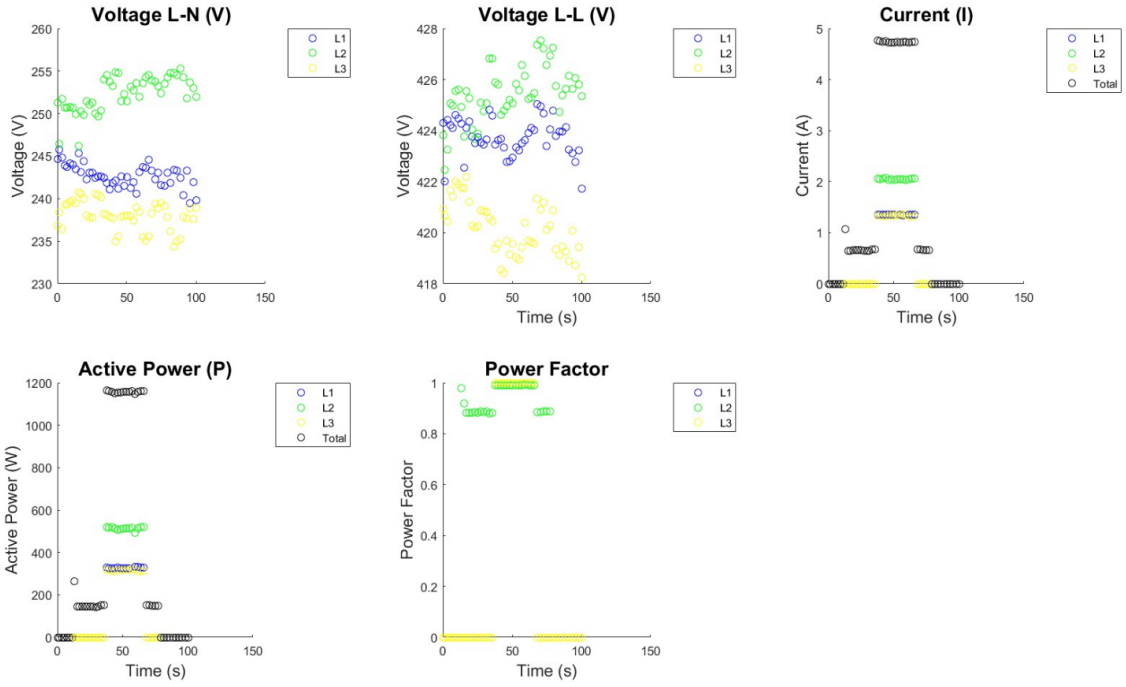


Figure 6.8: 1 kW resistive load with ventilation

As seen in Figure 6.8 the voltages of the lines are as expected. The total currents starts off at zero. This is because at the beginning of the measurement the load was not turned on yet. Around 10 s, the ventilation of the load was connected resulting in a current around 0.8 A with a peak current at the beginning of more then 1 A. Around 40 s the actual load is connected. This results in a current of almost 5 A. The ventilation must be connected to L2. This explains the higher current through that line.

A similar graph can be observed of the active power.

The power factor is around 0.9 with only the ventilation connected. When the actual load is connected the total load becomes a lot more resistive resulting in a power factor of almost 1.

4kVA capacitive load

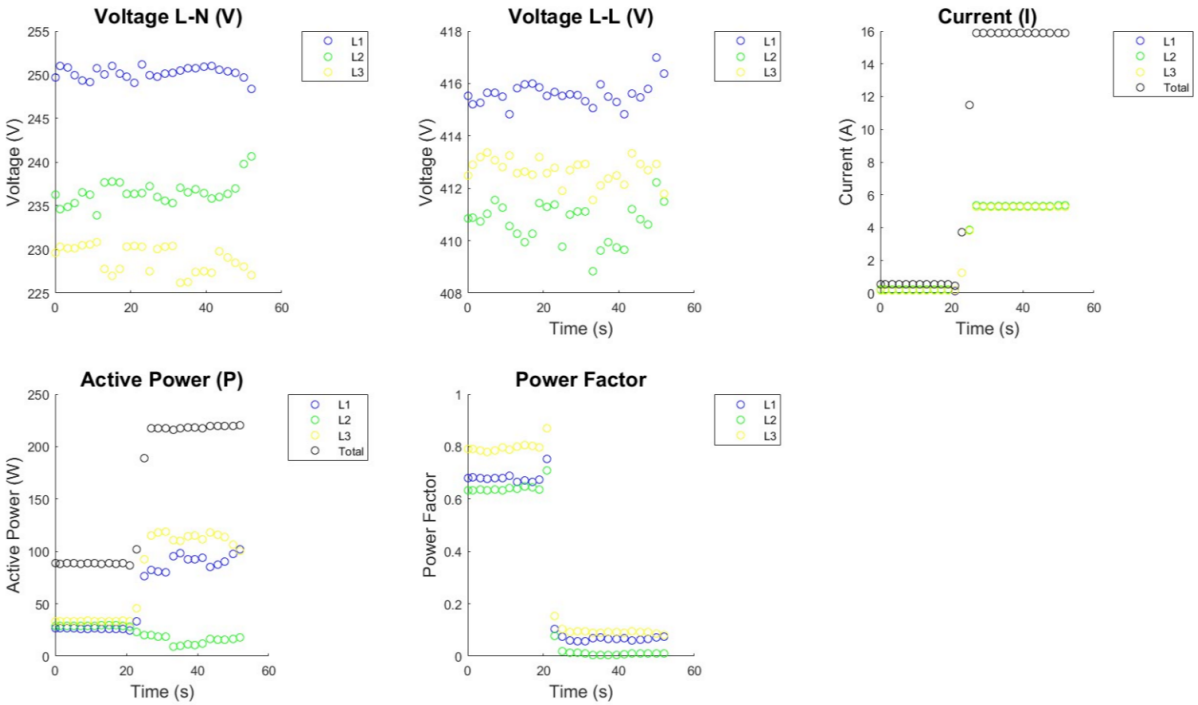


Figure 6.9: 4kVA capacitive load

.

Until 20 s, a small resistive load is connected. From that point and further, an extra capacitive load is connected, resulting in a rise of currents in the lines and a total current around 16 A.

The active power is the same for the three lines when only the resistive load is connected. When the extra capacitive load is connected active power of L1 and L3 go up. However the power for L2 does not rise. This can be explained because of the fact that as seen in the last plot, the power factor of L2 is very low, allowing almost no active power te be consumed.

In the last plot, a decrease of the power factor can be seen, as expected when connecting a capacitive load on to a resistive load.

Resistive, capacitive and inductive load

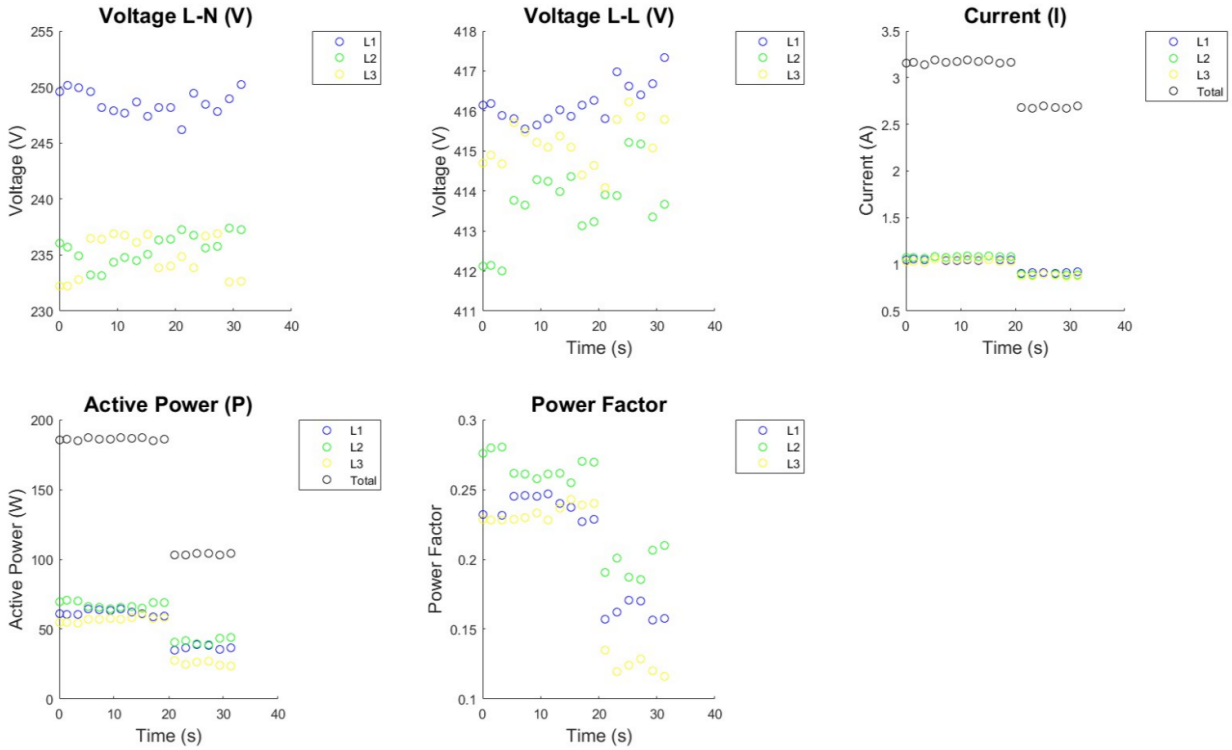


Figure 6.10: Resistive, capacitive and inductive load

.

At first, all 3 loads are connected. At 20 s, the resistive load is disconnected together with the capacitive load. This implies a decrease in total current. Same goes for the active power.

Because of the disconnection of those loads, the power factor falls back to smaller value as expected.

6.3 RealTimeBuses.m

For this simulation, all 3 loads were connected from the start. Loads were disconnected in the progress. A step size of 2 was chosen to give the give the loops enough time to retrieve the data.

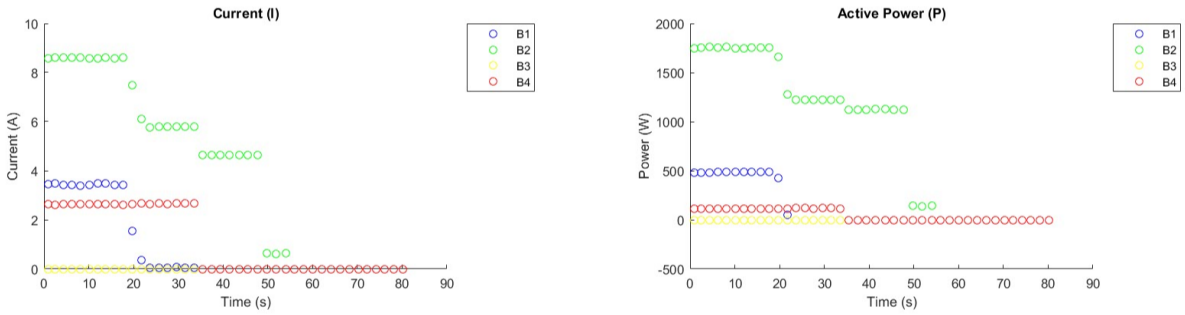


Figure 6.11: Simulation with all 3 loads connected to B1, B2 and B4

.

The power losses of the lines are plotted as well, but are not included in Figure 6.11 because of the same reason as mentioned at the beginning of Chapter 6.

As seen in Figure 6.11, bus 3 will remain zero because no load is connected with that bus.

At 20 s, the first load on bus 1 disconnected. Around 35 s, the load on bus 4 is disconnected. Finally, around 50 s the load on bus 2 was disconnected in 2 steps (ventilation and actual load).

Chapter 7

Conclusion

In this work, 2 programs have been written using 2 different approaches to retrieve the data of the analyzers with Matlab and conducted to speed tests. These 2 ways were used for different goals regarding database and real-time purposes.

In the end, the case study was implemented to verify the correct working of the code. However, with not all meters working, it is not possible to take full advantage of the program (see power losses). In the future, when the grid is configured with all meters working, this will be possible and a good monitoring of the full micro-grid can be done.

Overall the conclusion can be made that Matlab offers a good way to connect to power analyzers but that the speed really depends on the network. Also the connection is not always as stable as wanted. This causes the program to not reach the speed that is wished by the user. For database purposes, this is not a big problem, however for real-time monitoring it could be. If the connection speed is not fast enough, it causes problems related to the step size and sometimes peak values will not be seen in the plots. Of course optimizing the code will also give speed benefits, but this needs a better knowledge and more experience in programming. Simulink on the other hand offers blocks that are optimized for these kind of simulations, generally resulting in a faster and easier way to monitor in real-time. However, Matlab gives you more possibilities and is more flexible in the way it is displayed and so on. In the end, the program behaves as it should, but for real-time monitoring, adjustments will have to be made to reach optimal speed. Another way is to use the software that is especially made to do this: GridVis ®. The possibilities are limited, but overall it has been found a useful tool while making this project.

Further, MODBUS is the standard because its fast and easy for troubleshooting[**?**] but the network has to be as fast as the protocol otherwise it slows it down and an optimal working can not be guaranteed.

During the project, some difficulties arose. First of all, my knowledge in programming was not sufficient to begin with the coding right away. Some weeks had to be spend to generally understand the Matlab environment. Further, the Internet connection was not always stable and slow because of the traffic on the network. This resulted in a lot of time waiting for a good connection to verify the code. This has been a major problem the entire duration of the project and prevented me from optimizing the code as I would have wished. Finally, the micro-grid was not fully functioning when the tests wanted to be done. Some time was lost waiting until the case study could be executed. A more profound study could have been done if this was not the case.

Energy monitoring alike the one in this paper is only a step in what full monitoring has to offer. For industrial application it can become far more complicated and expensive in the end. These analyzers become faster, more accurate and can measure parameters for power quality. This is needed by companies to meet requirements made by energy suppliers.

In the end I would like to thank the GECAD research center with in particular Pedro Faria and Zita Vale for helping set up the project and supervising it. I would also like to thank Omid Abrishambaf for helping us with the practical problems and setting up the labratory. I hope The GECAD research center will be able to use it for future projects where energy monitoring is needed and/or will be able to expand it for further needs.

Bibliography

[1] Coal& electricity. http://www.worldcoal.org/coal/uses-coal/coal-electricity.

[2] Decoding floating point (ieee-754). http://nl.mathworks.com/matlabcentral/newsreader/view_thread/132795.

[3] Energiemonitoring. https://nl.wikipedia.org/wiki/Energiemonitoring.

[4] Interfacing a serial device to a tcp/ip network. http://www.taltech.com/datacollection/articles/interfacing_rs232_to_t

[5] The language of technical computing. http://nl.mathworks.com/help/matlab/index.html.

[6] Matlab. https://en.wikipedia.org/wiki/MATLAB.

[7] *MODBUS application protocol specification V1.1b3.*

[8] Modbus interface tutorial. http://www.lammertbies.nl/comm/info/nl_modbus.html.

[9] Modbus over tcp/ip. http://nl.mathworks.com/matlabcentral/answers/73725-modbus-over-tcp-ip.

[10] Osi model. https://en.wikipedia.org/wiki/OSI_model.

[11] Phasor measurement unit. https://en.wikipedia.org/wiki/Phasor_measurement_unit.

[12] Phasor measurement unit or synchrophasors. http://best.eng.buffalo.edu/Teaching/EE611/Phasor_Measurement_Un

[13] Simply modbus. http://www.simplymodbus.ca/FAQ.htmModbus.

[14] title.

[15] Janitza electronics GmbHn. *UMG 96 RM-EL, Modbus-address list and Formulary.*

[16] Janitza electronics GmbHn. *UMG 96 RM-EL, Operating insructions and technical data.*

[17] Clever Moler. The origins of matlab. http://nl.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html, 2004.

[18] Pedro Faria Zita Vale Omid Abrishambaf, Luis Gomes. Simulatoin and control of consumption and generation of hardware resources in microgrid real-time digital simulator.

[19] Modbus Organization. *Modbus messaging on TCP/IP implementation guide V1.0b.*

[20] Modbus Organization. Modbus protcol. http://modbus.org/specs.php.

[21] Ilhami Colak Askin Bektas Ramazan Bayindir, Erdal Irmak. Development of a real time energy monitoring platform.

[22] S.Meenatchisundaram. Interfacing of yokogawa controllers with matlab using modbus protocol for process control applications. 4.

[23] Kejun Qian Wei Jiang Chengke Zhou Xiaosheng Peng Xiang Dong, Xiaoping Jing. A smart grid needs smart monitoring.