POLITECNICO DI TORINO

Master degree course in ICT for Smart Societies

Master Degree Thesis

Desing and development of a novel smart-meter for improved Smart Grid management

Supervisors
prof. Andrea Acquaviva
prof. Edoardo Patti
prof. Enrico Pons

Candidate
Matteo Orlando
231414

Academic year 2017-2018

Acknowledgements

First of all I would like to thank my supervisors prof. Andrea Acquaviva, Edoardo Patti and Enrico Pons that followed me in the development of this thesis. I would like to thanks also the prof. Abouzar Etsebari for his help in the real-time simulation and testing.
A huge thanks to my parents and my family who sustained me during this years at the Politecnico and to my grandmother that fed me.
Thanks to Gianluca, Andrea e Sara, to the Pisolos, to the Disappointment group and to all the friends that supported me.

Contents

Chapter 1

Introduction

The electricity distribution system has faced many recent developments that revived interest in research and development that aims to bring benefits for both the enterprises and the customers. The first step is trying to understand what are the challenges that the grid will have to face and how the actual situation of the electric networks can be improved using new tools or already known technologies that belongs to different field but than can enhance the performance. The standard grid are shifting to a new generation of networks called *Smart Grid* that offers new feature to both energy suppliers and customers, the objective of this evolution is to meet the needs that are raising nowadays like the increasing spread of sustainable energy that reduce the utilization of fossil fuel lowering the environmental impact of the energy production and reducing the costs[1]. The introduction of technologies that comes from the Information and Communication Technologies (ICT) world are the main tools to proceed in this process. The main difference between traditional grid and

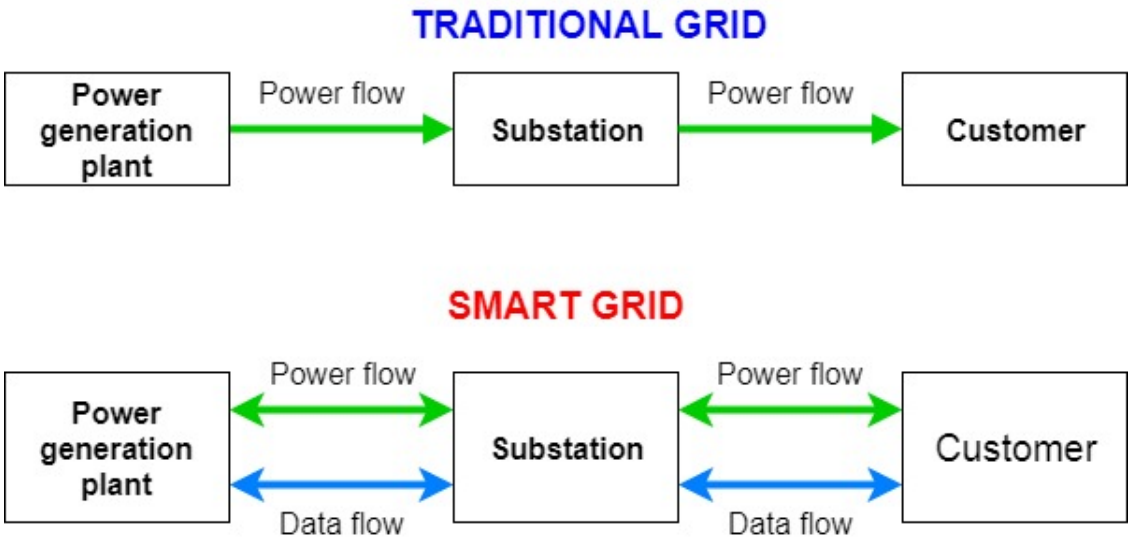

Figure 1.1. The two different scheme of traditional and smart grid

smart grid is the change in how the different actors of the networks are treated:as depicted in the figure 1.1 nowadays the the electric grid has a hierarchical architecture where the energy flows from the generation plant to the final user crossing different substations, in the new concept of grid instead all the actors are considered as equal element that have different flows of energy (that means that each element can introduce or extract power from the network) and information (each element can transmit or receive information from the others). This new vision is needed to integrate in the network the new devices that will be deployed and the upcoming Distributed Generation plant. The smart grids will use communications architecture and Advanced Metering Infrastructure (AMI) to acquire data and enhance the reliability and the forecasting capability.
The first step to better manage the grid is to have a better knowledge of what is happening inside the network, like the values of the physical quantities like voltages and currents, for this reason a lot of work is being done to develop the so-called *Smart Meter*, these devices are meter with added function such as communication interfaces, data-processing capabilities and more, hence the name *Smart*. We may divide this devices in 2 families according to their target user: customers or energy suppliers. Let's see their characteristics:

- *Customers*
  Most of this devices are installed by the provider of electricity and allows bidirectional communication between user and supplier bringing a lot of advantages, for example they allow more transparency in the billing systems and the deployment of Demand Response plans that can be extremely efficient by remotely controlling the appliances. On the market it's possible to find also plug&play devices for the private user that can increase the customer awareness about how he's using the energy in his property

- *DSOs*
  They're used by the DSOs (Distribution System Operators) to know at every moment the operating state of the grid. They are mainly used to collect data about the value of voltage, current and power of a point of the grid but they can also used to directly act on other smart devices deployed on the grid, for example to change the topology in case of outage or fault.

Despite the differences each smart-meter has as main objective to enable bidirectional communication to be coherent with the idea of smart grid and deploy new feature like self-healing or outage detection. The bidirectional communication can also increase the participation of the customers. To deploy and use efficiently any Smart-metering system it's needed an architecture with some fundamental entities.

- *Concentrator*
  His purpose is to collect and group the data coming from different smart-meter to send them to a Data Management System. It is not mandatory to have it but it is really useful if the number of meter is high as it reduce possible congestion on the communication channels.

- *Communication systems*
  It's used to create a safe network that allows the meter to communicate information of different kind (raw data, processed-data etc.) to a Control center or to the customers.

- *Management system*
  The Managements system has 2 main objective: it is responsible to receive the data coming from the meters or concentrator and to process it to perform various function(i.e. load forecasting, Demand Response etc.) this group of function are embedded in what it's called Data Management Systems (DMS), the *Management system* is also in charge of supervising the network status and keeping track of which smart devices are working on the grid.

Other important features that make the grid "smart" are the different algorithms that process the data coming from the meters to perform or to optimize the performance of the grid, here there is a brief list of the most common ones:

- *State Estimation*
  States estimation algorithms are used to calculate the operating conditions of a power system at a given point in time (i.e. the set of complex phasor voltages). Knowing the operational state of the grid is helpful to monitoring that the constraints under which the network has to work are respected and to analyze possible contingencies.

- *Fault Location*
  Fault location algorithms are used to find with the most possible precision the location of a fault to speed up the restoring operation from the land crew, they can be used also for the realization of self healing networks, a really attractive features of smart grids. The data gathered from various meters installed throughout the power system could be utilized to locate faults in both transmission and distribution systems.

- *Load analysis, modelling and forecasting*
  Models of the energy usage can be created analysing the past data coming from the meters, using this model with the real time data the total energy usage can be estimated and forecasted. This estimation can be used by the DSO to increase the energy savings, to avoid possible contingencies. One of the most common use for this model is the creation of Demand Response policies that aims to change the behaviour of the customer in the usage of the electricity to better match the supply and demand of energy at any time.

- *Distributed Energy Resources Management (DER)*
  It's expected that in future there will be lot of small energy generators, usually from renewable sources, spread inside the grid, this brings a lot of issues to resolve like the impredictability of this kind of generation that may create instability to voltages and current. For this reason some tools are needed to face this upcoming scenario and manage in the correct way the DER and the possible energy storage systems coming with it [2].

1.1 Motivation

One of the main reason to develop new software infrastructures and smart-meters is to improve the user experience ensuring a high-quality and reliable services in every situation. This means both to minimize the number of outages and in case this happens restore the

system as fast as possible but also to increase the users awareness about how they use the energy. This new tools can also help the energy suppliers to avoid critical situations for example if a DSO has an updated knowledge of the status of the grid and notice something that may cause a fault it can preventively take some action to avoid it. One of the most interesting features that we aspire to deploy is the possibility of have a self-healing network that in case of fault automatically avoid the outage by re-configuring the network to ensure the energy supply until the fault is repaired and then restore the normal condition. To reach this objective the smart-meters must recognize the location of the fault, isolate it by remotely opening and closing the correct switches and then re-establish the original connection of the grid, this set of operation is called Fault Detections, Isolation and Restoration (FDIR). To satisfy all these needs an appropriate software infrastructure has to be designed capable of manage the network controlling and acting on the different element of the grid, able to supply an almost real-time overview of the status of the network. Moreover a new smart-meters have must be design and deployed inside the grid, this devices has to be integrated in the software architecture providing bidirectional communication to collect data and transmit information to other services or command to devices. One of the main advantages of the introduction of a distributed infrastructure that use smart-meters is that these devices could perform more advanced function other than just collecting measurements allowing the possibility to decentralize the processing of the data processing and therefore reducing the utilization of the communication network, avoiding possible congestion.

1.2 Challenges

To realize this shift to the new concept of smart grid there are many challenges to face both in the design and in the deployment. The first obstacle that we may encounter is the economic front, huge investments are needed to realize and maintain this infrastructure, high-precision device with elevate frequency of data acquisition are needed and the cost of these devices can be quite high as we will show in the section 2. On an other side the bidirectional communication enabled by the new generation of devices should be reliable and safe, his presence opens new problem regarding privacy and security: external malicious attack by hackers can try to create fake information or cause issues to the network by modifying the correct functioning of the grid. The data coming from the house hold could also reveal the habits of the customer and some user may also be reluctant to share the information for privacy issues. Another problem that may affect the communication network is that when a lot of this devices are installed the amount of information exchanged by the different entities of the network can be huge and difficult to manage properly, by distributing some process on the devices this issue may be partially resolved but should not be underestimate. Among the diverse obstacle to face the spreading of Renewable Energy Resources (RER) and DER is one of the most complex, as said before renewable energy sources are extremely unpredictable and it is difficult to properly forecast their production, moreover if an end user has his own DER the situation becomes even more complicated because the unused power could be sold and injected on the grid [3].

1.3 Proposed solution

After this considerations, the analysis of the current state of the research and the solution offered on the market nowadays, this thesis propose a software infrastructure that is able to manage a large amount of meter exploiting two well known communication paradigm, it supplies a periodically updated list of the meters deployed in a grid and allows to communicate to them their settings and the function these must perform, according to the needs. This infrastructure is also able to collect information coming from the different meter and forwards it to the services interested. A prototype of smart-meter has also been realized to work with this infrastructure, it is a low-cost, open source, self-configurable, auto-updating and Internet connected device. It has a set of on-board for grid management that can be run selectively and supports self-healing systems. Tests have been made to test the performances of both the software and the smart-meter.

Chapter 2

Related Work

Literature solutions are focused on how to exploit information coming from smart meter to enhance smart grid's performance with particular emphasis on management and on optimizing customers' experience. In this view, the main objectives of future smart meters can be classified in three main categories:

- performing demand response events
- acting as interface between the grid and external agent (DER, electric vehicle, etc.)
- perform data processing and run algorithms for grid management (state estimation, fault location, etc.)

The common characteristics between the different proposal is the usage of Internet of Things (IoT), the IoT approach consists in supply internet connection to a device to enable it to receive and send command and information, this paradigm allow to deploy devices with improved or new function respect to the old ones. For example nowadays most of the end-user meter used for billing purposes uses Power Line Communication (PLC) to transmit the power measurements to the DSO, using instead an Internet connection the meter could provide additional information both to customers and DSOs to deploy new function like Demand Response. In the next sections some of the solutions proposed in literature and in the market will be analysed and compare to the one proposed in this thesis.

2.1 Literature's meters

Minchala-Avila et al.[4] proposed a meter designed specifically to perform Demand Response, in order to achieve this a meter needs to: (i) collect data about customers power consumption and generation (nowadays DER are not so common but in a general case they've to be taken in account) (ii) enable bidirectional communication between user and DSO. To satisfy this constraints the meter proposed is made with a Raspberry board, that works as a processing unit, and an integrated circuit coupled with a voltage divider (to measure voltage) and an hall effect current transducer (to convert the current in a tension and measure it) collects the measures and sends these to the Raspberry through

the SPI interface. The Ethernet port provided by the Raspberry makes possible all the needed communications. The test were made by connection this mono-phase meter to an household prototype in conjunction with a simulation of other meters with the respective household consumption.
There are 2 major difference between this meter and the one designed in this thesis:

- Demand Response purpose vs. general purpose
  Only two case study are evaluated in this thesis but the meter doesn't set particularly big limitations about the algorithms that can be deployed

- Mono-Phase vs. 3-phase
  A 3-phase meter enables functions that are precluded a single-phase meter like MV-distribution monitoring

On the other side the test are executed in a similar way by connecting the meter to something that emulates the real-world case (house-prototype vs. grid simulator).
Qiao et al.[5] developed a meter to interface the electric vehicles to the grid to manage charging and discharching (electric vehicle could be uses as DER). Also in this case the two main operation needed by the meter are: (i) data collection (ii) bidirectional communication. To collect the 3-phase measurements an integrated circuit, called ADE7758, is used, the data are sent through SPI to the Central Processing Unit (CPU), a SAMSUNG's S3C2440 that is a single board computer with characteristics similar to the Raspberry Pi, General Packet Radio Service (GPRS) is used to communicate with the DMS.
Also in this case the proposed meter can perform just one function however the possibility to acquire 3-phase measurements but the usage of the GPRS may reduce the possibilities on what can be implemented because that puts some limitations in the throughput.
Angioni et al.[6] designed a meter able to collect 3-phase measurements that satisfies IEEE c37.118.1-2011 standard for accuracy. Their objective was to realize a meter that can work as a Phasor Measurement Unit (PMU), this kind of device are really important for monitoring application and can provide data for different applications. The hardware used to make the meter are a MCC-USB 201 as interface for collecting the data and a Raspberry Pi board as CPU. The result of the paper is in effect a meter that could be used in a real network as it satisfies the requested standards however it lacks of some feature that are instead present in the one proposed in this thesis like: IoT Infrastructure, on-board algorithm, self configuration.

2.2 Commercial meters

The EPM 7000 provides continuous metering of three-phase systems with waveform and data logging. It performs energy measurements and supports optional function, relay, status, and analog output communication modules [7]. This meter can be used for a wide range of high accuracy applications including disturbance recording and power quality studies. This meters as an higher sample rate compared to the other analyzed before (25.6kHz) and has an high accuracy (0.1% accuracy for Voltage and Current). However there are some drawbacks:

- the Ethernet module is optional

- it's quite expensive
- additional software are limited, not open-source and must be bought from the manufacturer

The Devolo G3-PLC Modem 500k is the meter developed for medium-voltage network and is ideally suited for smart grid applications but it's flexible as it can collect also single-phase measurements [8]. The communication occurs through Ethernet over PLC (Power Line Communication) and once the meter is installed the PLC network independently build itself, the network configuration can be modified with a normal web browser. This self-configuration capability of the communication network is an interesting feature, however using PLC, due to his limited range, force the deployment of PAN (Personal Area Network) Coordinator that manages the meter. The absence of any additional function other than data collection is also a weak point if compared with the meter proposed in this thesis.
The MTR 3000 P2P is a 3-phase meter that support the P2P Gateway Module: a modem-equipped device that can be used to connect to NES System Software and connect the meter over a cellular network [9]. There are a lot of different function can be installed on the meter (phase loss, total harmonic distortion, etc.) but nothing can be added to the stock feature. As said before for the vehicle-to-grid meter employing the cellular network limit the throughput, moreover the necessity to use a property solution for the network (NES System Software) adds further limitations. However the possibility for the meters to communicate among them is an interesting feature that's also present in the architecture defined in this thesis.

2.3 Scientific novelties

As said before the aim of the thesis was to design and deploy a software infrastructure and a meter that could satisfy the main needs of the future smart grid, to succeed in this objective the meter should have the possibilities to perform different function according to the position it has on the grid and the software architecture should be able to manage this devices despite their different purposes. To achieve this result two mechanism has been used: *self-configuration* and *auto-update*. Through this two mechanism the meter is always aware of his position in the electric network and knows what function has to perform, on the other hand the DSO knows which meters are active, their functions and can modify their configuration remotely. Therefore thanks to this mechanisms a variety of different function can be installed on the meter and but just the desired ones are performed according to the needs or the preferences. That is a big advantages respect to the standard approach used in the design of a meter as this devices are usually able to perform just the function they were designed for.
In the 2.1 there is a synthesized comparison of the characteristics of the meters presented above and the one designed in this work.

Meter	3-phase	IoT enabled	Sampling rate	Multi-functions	SMI integration	Self-configurable	Low-cost
Our Solution	✓	✓	6.4 kHz	✓	✓	✓	✓
Minchala-Avila et al.	x	✓	0.017 Hz	x	✓	x	✓
L. Qiao et al.	x	✓	n.a.	x	✓	x	✓
Angioni et al.	✓	✓	10 kHz	x	x	x	✓
EMP 7000	✓	✓	25.6 kHz	✓	x	x	x
G3-PLC	✓	✓	n.a.	x	✓	x	x
MTR 3000	✓	✓	n.a.	x	x	x	✓

Table 2.1. Comparison with literature solutions

Chapter 3

Smart-meter to enable novel services in smart grid

None of the device that has been analysed in the previous chapter possesses all the features that we consider as a must-have for a meter,hence we designed another solution. In this chapter we will describe the meter we proposed, the technologies that are used, the hardware and the function that are implemented on the device and the infrastructure in which it works.

3.1 Enabling technologies

3.1.1 Request/response

The Represantational State Transfer(REST) is an architecture to transfer data through the Hypertext Transfer Protocol (HTTP) in a distributed environment like internet. It has been described and developed at the beginning of this century [10] and has become a staple for the communication in the Web. REST is used to transfer data from the source to the user, this data are sent in a format that depends on their nature and on the capabilities of the end user (computational power, application etc.). Any data in REST is called *resources* and the REST user act on this resources through *representations* that are, in other words, a photograph of a state of the selected resource. Depending on the *control data* used to obtain a representation it may contains the actual state of the resource, the state in which the resource should be or other info. The main actors in a REST architecture are called *user agent* and *origin server*, the first component is essentially the client that needs the data that are contained in the second component that stores it. All the interaction inside a REST architecture are *stateless*, that means that each of them is independent from the ones performed before and from the following ones, the main benefit of this constrains is that a server does not need to store additional information for each request coming from each client. However there is an element called *cache* that allows the client, when specified, to store the data contained in the server's response to be used later for similar requests. Another strength is that REST application does not need a particular programming language, any language that allows to make HTTP-based request can be used

to implement a REST application. If an application respects all the constraints imposed by the REST paradigm it is called RESTful.
HTTP request is the most common method for a client to interact with a REST resource, HTTP defines some basic keyword that specifies how the client wants to interact with the server, these keywords are called *verbs* [11], four of them are used in a RESTful architecture:

- *GET*, it is used to obtain the representation of the actual state of the resource when this method is called, this verbs is the safest one as it does not change the state of the resource.
- *POST*, it is used to ask the server to store a new resource with the data contained in the request
- *PUT*, it is used to update the state of a resource on the client according to the data contained in the request,if the resource does not exist it is created
- *DELETE*, as the name suggest is used when a resource has to be deleted from the server

3.1.2 Publish/subscribe

One of the downsides of a REST architecture is that the server is able to send information to the client only if it performs a request, in case the state of a resource changes the client will not receive the update until it will request again the state of that resource. For a lot of application this restriction is not an issue but in some cases the user needs to be informed as soon as possible about a state change, in this kind of situation a REST architecture is not the best choice while the Publish/subscribe (P/S) paradigm can satisfy this requirement [12]. In the figure 3.1 is shown an example of a P/S system, with the possible interaction between the the main entities of this paradigm that are 3: :

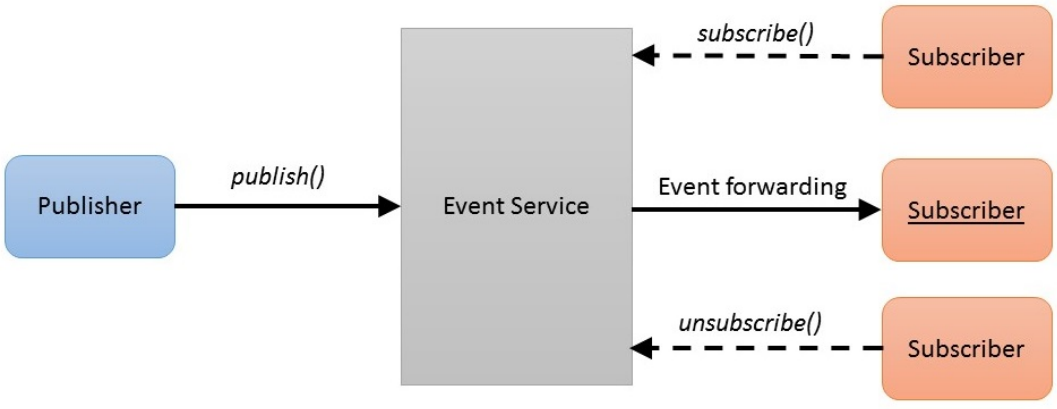

Figure 3.1. Example of a Publish/Subscribe system

- the *Publisher* which is the actor that generates an event by using a method called *publish()*
- the *Subscriber*, an entity that calls a *subscribe()* method to express to kind of events it is interested in receiving information about or the method *unsubscribe()* to stop receiving updates
- the *Event Service* is the element that acts as mediator between publishers and subscribers, it stores the subscription information, receive the event generated by the publishers and forwards it to the subscribers interested

This paradigm allows a separation between publisher and subscriber that do not need to know each other, a publisher does notusually know how many subscribers has, neither their position, in the same way when a subscriber receives the information about an event it does not know which publisher triggered and where it is located (the data carried in the notification may contains this information but the notification itself does not provide it). Another characteristic is that publishers and subscribers does not need to take part in the interaction at the same time, a publisher can trigger events even if all his subscriber are not connected and the subscriber can subscribe even if the publisher is offline. A big advantage of this paradigm of communication is that *publish()* and *subscribe()* can be executed while doing some other process, the publication and the notification are asynchronous and do not block the normal work flow of the users. A common protocol that exploits the Publish/Subscribe paradigm is Message Queue Telemetry Transport (MQTT), in MQTT each user is a client that must connect to a broker(i.e. Event service), then it can subscribe to events it is interested in (called *topic*) or publish to trigger events [13]. It allows 3 differents possibility regarding Quality of Service (QoS) to ensure different levels of reliability according to the needs of a sytem. MQTT is a lightweight protocol and is a common choice for communication in networks with simple devices as it minimizes power and network usage, the characteristics of P/S indeed are extremely suited to systems that needs scalability like sensor networks [14]

3.2 Proposed Meter

As said in the chapter 1 the aim of this project was do develop a smart-meter that capable to perform a variety of algorithms, the ones that have been deployed on the meters are three: Outage Detection, Fault Location ans State Estimation that will be described in the chapter 4. In order to be able to run this algorithm the meter should have particular requirements regarding the sampling rate and the computational power. Regarding the data acquisition part the initial idea was to use an Arduino Leonardo board, that is a micro-controller board with an on-board analog-to-digital converter (ADC). This board has been tested to collect measurements for 6 analog input, the standard sampling rate of this board is 9600 Hz that becomes 1600 Hz for each of the 6 channels, that means 32 samples for each waveform of the inputs but the target we aimed for was to have at least 128 samples in order to have an amount of data adequate to perform the largest number of function possible. To try to reach this objective we tried to modify the settings of the Arduino board to increase the sampling rate but this could be increased only to the detriment of the resolution that drastically reduced the quality of the measurements.

For this reason this initial solution with the Arduino Leonardo was abandoned to look for another device that satisfies the requirements. After comparing some other devices the

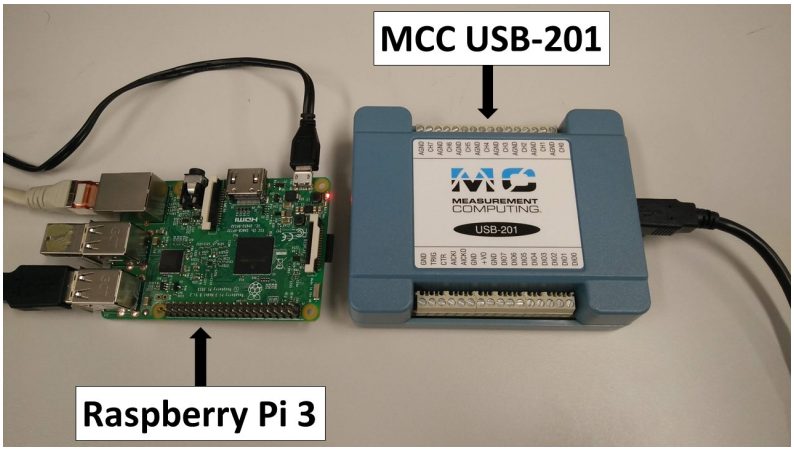

Figure 3.2. The hardware that compose the proposed meter

meter developed in this thesis (3.2) was composed by an analog to digital interface named MCC USB-201 [15], connected to a Raspberry Pi 3 model B [16] through the USB port. The MCC USB-201 has up to 8 analog inputs, 6 of them are used to collect the 3 voltages and 3 current, the range of the input channel is 0-5 V, therefore in a real world scenario some transducers and bias voltage would be needed to scale the real voltage in this range and move the 0 values of the input sinusoidal waves to 2.5 V. The maximum sampling rate of this device for a single channel is 100 kHz, for a real case scenario a sample rate of 6.4 kHz would be an optimal solution. The data acquisition however does not happen simultaneously for all the channel but through a multiplexer that reads one channel at the time and stores the value in a buffer that contains 768 values as shown in 3.3. This

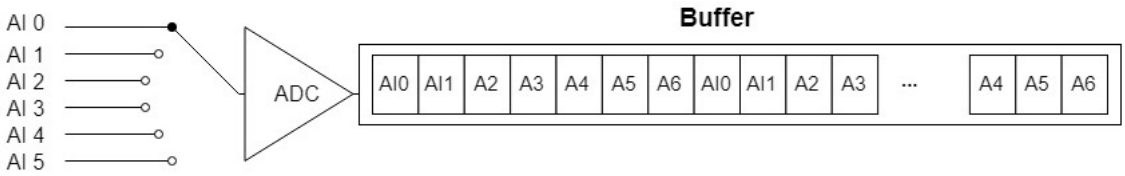

Figure 3.3. Scheme of the data acquisition process

array with the sampled measurements is then sent to the Raspberry Pi through the USB port. The Raspberry Pi (RPi) is the computational core of the meter, it manages the data processing, the Internet connection and the routines for self-configuration. The RPi receives the data packet containing the samples of each cycles of 20 ms of each inputs, this data are processed by doing a Discrete Fourier Transform (DFT) to obtain magnitude and phase of each input (the phases are given taking the first input as reference) and for possible further elaboration a window of one second of measurements is stored. The code

dedicated to the measurements acquisition is written in C language while the scripts for data processing, self-configuration and update are written in Python. RPi can also store and run the algorithms for grid management, in this project three algorithm are deployed:

- *Outage Detection*, it inspects the processed measurements to identify possible shift from the correct values expected
- *Fault Location*, it is triggered if the from the previous one finds any issue and it tries to find the position of the fault in grid branch it is supervising
- *State Estimation*, it estimates the working state of a portion of the grid using the data collected from other meters

These are just few of many possible function that can be implemented directly on this meter to ease the management of the grid for the DSO and improve the customer experience.

3.3 Distributed software infrastructure

The objective was to deploy a software infrastructure that is able to manage a large amount of meter exploiting REST and MQTT, it should supply a periodically updated list of the meters deployed in a grid and allows to communicate to them their settings and the function these must perform, according to the needs. This infrastructure is also able to collect information coming from the different meter and forwards it to the services interested. As shown in figure 3.4, the main entities in the proposed software infrastructure are:

- two *Message Brokers*
- one *Device Catalog* (DeC)
- the *Services* for smart grid
- the various 3-Phase Smart-meters

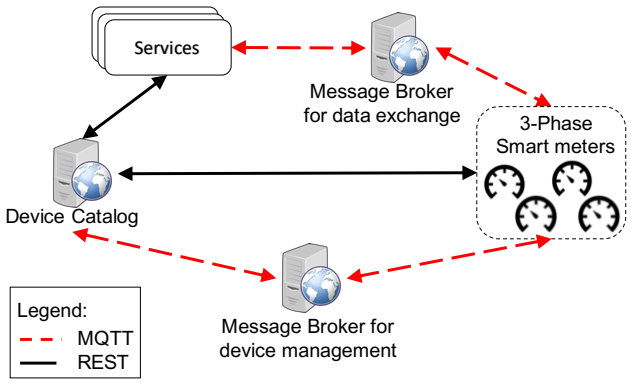

Figure 3.4. Model of the proposed software infrastructure

The *Device Catalog* (DeC) is the core software to supervise and manage the infrastructure. It keeps a list (i.e. the catalog) of the meter that are active on the grid, giving information about their status and their settings (i.e. MQTT topics, function enabled on the meters, etc.). In order to maintain this list updated a function is deployed that periodically checks whether each meters is alive or not with a procedure that it is described below. DeC provides to each meter the configuration settings with the structure shown in figure 3.5. This file contains the following information:

- the *Broker list* section contain the IP of the two MQTT brokers and for each of them the list of topic to which the meter has to subscribe to and publish to.
- the *Meter ID* uniquely identifies the device according to his position in the grid
- the *Dec IP* is the IP address of the catalog, it is used by the meter to perform the HTTP request to retrieve the configuration. The time interval at which the updates has to be done is also defined in the settings
- the *Algorithm list* defines with a value that can be *True* or *False* whether or not an algorithm has to run on the meter
- the *Last update* is the time at which the meter has sent the last HTTP request to update his settings. That parameter is essential in the procedure that keeps the catalog updated.
- the *Matrices* part contains all the information about the structure of the topology, the possible DER present in the grid

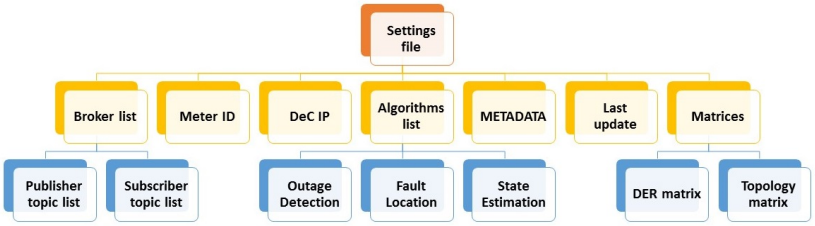

Figure 3.5. Structure on the setting file

Two brokers are used for the MQTT communication to separate the messages devoted to the grid management from the messages coming from the meters containing information about data processing or notifications. One of the most innovative features of the proposed meter are the self-configuration and the self-update mechanisms, these functions allow the meter to be always aware of the condition of the networks while the DSOs can always know which meter are operative on the grid. Before the deployment on the grid two information are stored inside the memory of the meter: its ID and the IP address of the DeC, as shown in the diagram 3.6 when it is switched on it automatically sends an HTTP GET request to that IP address containing his ID, when the catalog receives this request it reads the file correspondent to that ID and sends back to the meter its settings. After that the

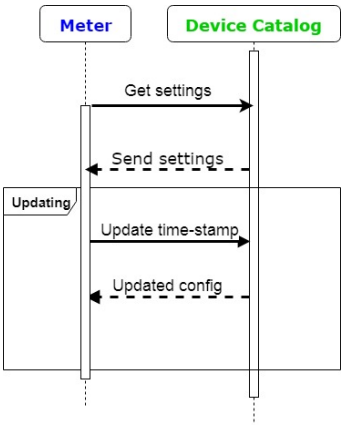

Figure 3.6. Scheme of the communications between a meter and the DeC

meter periodically perform a PUT request to the DeC updating the time-stamp contained in the *last update* parameter, the time interval at which this update has to be performed can be modified according to the necessities, for example if a lot of meters are working a larger interval avoids possible congestion of the network. On the other hand the DeC perform another loop in which checks the time of *last update* of each meters, if the difference between this time-stamp ad the actual time is bigger than a given threshold the meter is removed from the list of active meter. This threshold can also be customized according to the situations. One of the important characteristics is that by receiving this periodic update the meters always knows his position inside the grid and even if it is moved in another point of it would know his new position as soon as it is plugged and switched on.s

Chapter 4

Algorithms

Many algorithms [17] [18] [19] [20] were designed and developed to improve the performance of the grid by using the data coming from the meters installed in the network, some of them are focused on improve the customer experience while others aim to simplify the management operation for the DSOs. One of the main objective of the algorithm that belongs to the second group is to use communication technologies and the measurements devices to manage efficiently situation of fault and increase the reliability of the grid. In case enough Intelligent Electronic Devices (IED) are deployed on the network (remote switches, meter etc.) also an automatic self-healing of the network could be possible, allowing the network to restore the functioning of the grid by isolating the fault. Among the possible different algorithms chose to deploy on the meter the ones already mentioned before: Outage Detection, Fault Location and State estimation.

4.1 Outage Detection

An outage happens when a portion of the network is affected by a loss of electricity caused by a fault in a particular point of the grid, a major percentage of the outages are caused by faults in the distribution network. Outages can be dangerous for the network itself and also for critical application, therefore it's important to detect them as soon as possible and define the area affected by them and proceed to isolate it and restore the normal service. Therefore Outage Detection process has as objective to recognize the occurrence of an outage in case of a permanent fault of the electric distribution system and notify it to the DSO. Nowadays, in the conventional electricity grid, the network operators carry out this process by analysing the telephone calls made by the customers that communicate that their electric service is interrupted. This method has some drawbacks: during night time the number of calls may be really low and increase the difficulty of the detection, fake reports are hard to recognize and the whole process is time consuming and therefore inefficient as the outage time has a direct influence on the system reliability, on the quality of the energy and on the satisfaction of the customer. This process nowadays can take from few minutes to even hours if depending on the location of the fault and on the time of the day while with a proper smart-meter less than a minute is enough, indeed when the values of voltages and current for different point of the grid are available we can check almost in

real time if a fault has occurred. For this project we decide to develop a meter that could perform this process of outage detection by running the algorithm shown in figure 4.1, it works as follow:

1. the measurement of one cycle of each input channel (3 voltages and 3 currents) are utilized as input, the DFT is performed on this data to find the magnitude of the main frequency component of the waveform on the input and their phase shift
2. the algorithm checks if this calculate values respect the expected one with a certain margin of error given by some predefined thresholds
3. if the constraints are respected the algorithms stops and it will run for the next cycle of data otherwise before stopping it will launch a Fault Location algorithm that we will describe in the section 4.2

Before the fault is actually notified and the fault location algorithm is performed at least 25 "faulty" cycles must occur, this design choice has been made to avoid that the algorithm notifies a fault that may for example be caused by sporadic error in the measurements or by some data losses. In the chapter 5 we will analyse the performance to check if it can improve the management of the grid.

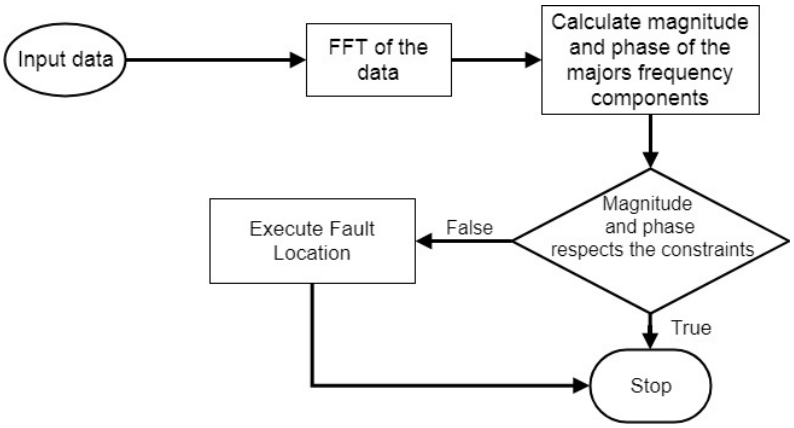

Figure 4.1. Flowchart of the Outage Detection algorithm

4.2 Fault Location

In case of an outage the next step after the definition of the interested area is to find the location of the fault, nowadays this operation is made by a repair crew that has to control with special tools the power distribution lines of all the area interested. The time needed to perform this operation may take from few minutes to hours depending on the dimension of the interested area and the location of the fault, also the weather condition can influence this time. In a smart grid the measure coming from the meters or even the meters itself can be use to speed up this process by using accurate fault location methods, they can reduce the area to inspect and accelerate the restoration to the normal conditions. With

this objectives in mind a lot of research about fault location has been done to realize new algorithm or improving existing ones using different techniques to process the data (neural network, impedance based methods, etc.). For this thesis the algorithm proposed in [20] has been chosen, it combines an impedance based method with sparse voltage measurements to combine the strength of this two approaches. The impedance base method has good results regarding the position of the fault along the line but it gives multiple solution, meanwhile the voltage sag method gives a single solution but it needs accurate and high frequency measurements that may not be available in a traditional distribution network and this solution consists in just the nearest node to the fault location.
Let's now describe how these two algorithms (impedance method and voltage sag) would work separately and how they were combine to increase the overall performances. The impedance-based fault location algorithm uses the frequency, voltage and current measured at the primary substation to find the fault location. It starts from the first section of the line and estimates where the fault occurred by solving a set of equations. In this step, if the calculated distance is bigger than the actual length of the considered section, the result is discarded and the same process is repeated for the next sections updating the voltage and current values. This process is repeated for each section to find results coherent with the network dimensions. The sag voltage method considers every node, one for each cycle of the algorithm, as the possible fault location, it calculates the change of the voltages at all nodes. At each iteration a load that emulates the fault is considered attached to the node inspected a the possible location of the fault. After this step, comparing the measured and the calculated voltages, it calculates the index $\eta_i = \frac{1}{\Delta V^m - \Delta_i^c}$, the node i with the highest η_i is taken as solution as it has the lowest difference between measured and calculated voltages, so has the highest probability to be the location of the fault. The

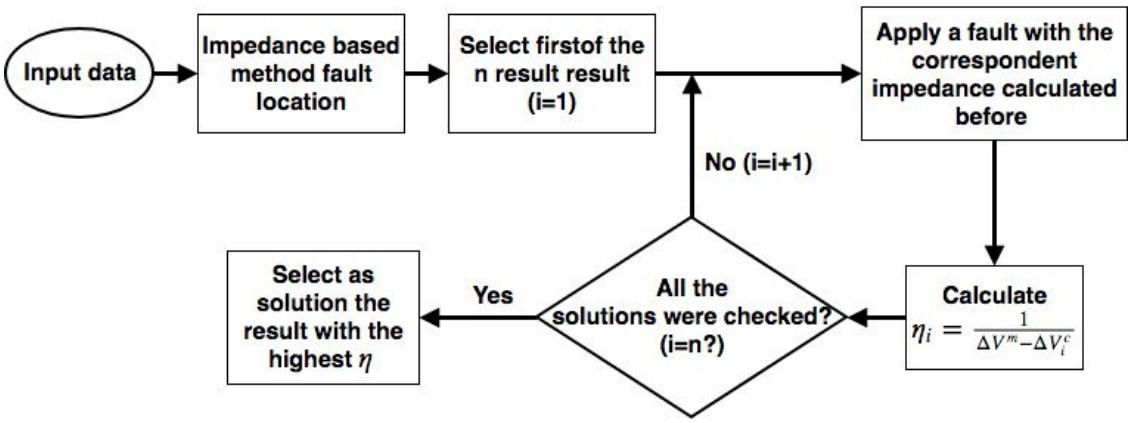

Figure 4.2. Structure of the Fault Location algorithm implemented on the meter

algorithm for fault location deployed on the Raspberry Pi use a combination of this two methods described above, it is shown in figure 4.2 and proceeds as follows:

1. the algorithm is triggered by the Outage Detection that in case of permanent outage calls the Fault Location algorithm giving as input the one seconds of measurements, 0.5 seconds before the fault and 0.5 after.

2. the impedance method is used to find the different possible fault location (in a general case n results)
3. the first results of the n found in the previous step is selected (i=1)
4. a fault is applied according to the i^{th} result with the correspondent fault resistance
5. the voltage sag and then the index η_i are calculated
6. if all the result of the impedance method were evaluated (i.e. i=n) the i^{th}result with the highest η_i is considered as solution otherwise the next result in processed (i=i+1) returning to the step 4

4.3 State Estimation

Nowadays the increased spreading of DER is offering opportunities to reduce the costs and the environmental impact of energy generation but it also creates a new scenario for the DSOs where the grids are shifting from a unidirectional flow (from supplier to customer) to a bidirectional flow (power can be produced and injected at any points inside the network). The deployment of a new generation of devices and technologies is needed to face this challenges and increase the efficiency and the reliability of the future smart grids [18]. With the developing of smart grid SE can be really helpful to increase the DSO's awareness about the operating conditions of the network. One of the tools that can offer a lot of benefit for the network management is the Stase Estimation (SE), it is function that uses similar data coming from multiple sources (i.e. the meters in the network) to increase the accuracy of this data and it even allows to obtain an estimation of the physical quantities in portion of the grid that are not supervised by measurement devices [21]. In a general approach the parameters we want to calculate with a state estimation are:

- voltage magnitude and phase at each node
- transformers turn ratio and phase
- active and reactive power at each node

To obtain this parameters we used real or pseudo-measurements like:

- voltages and current magnitudes and phase shifts
- active and reactive power

The objective of the state estimation is to minimize the difference $z - h(x)$, where z represents the measurements while *h(x)* is a function that link the measurements to the state variable. A lot of different approaches can be used to perform SE, but most of the algorithm make use of the Weighted Least Squares (WLS) as they can provide also the error correspondent to the given estimated state. Knowing these uncertainties can help the DSO in the grid management and in the prevention of critical situations. The processing capabilities needed for perform a SE for a complete grid are not negligible so, in order to avoid excessive computational load, a layered structure poposed in [18] has been selected, this

architecture for SE uses three different levels, each layer uses a different algorithm to perform an estimation about the portion of network and communicates its results to another layer through an MQTT broker as shown in figure 4.3 . These three layers correspond to

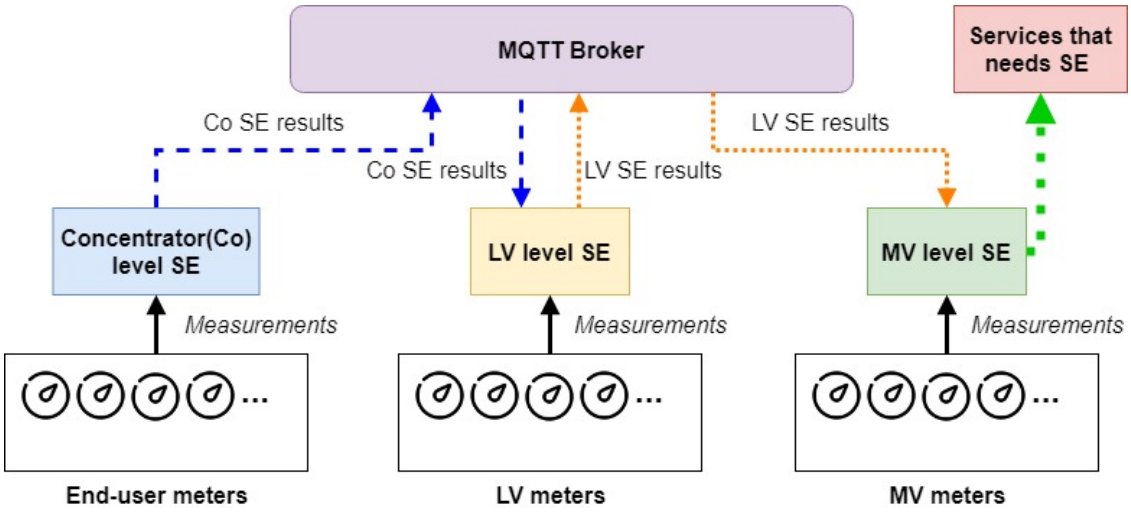

Figure 4.3. Scheme of the interaction that occurs while performing state estimation

the different voltage levels present in the grid, the structure is shown in figure and :

- *Concentrator level*
  The data collected at this levels comes directly form the smart meters deployed in the a portion associated to a particular feeder (a feeder is a node of the grid that provides the energy to a group of customers). Voltage magnitude and active and reactive power measurements are then provided as input to the concentrator SE algorithm. The results will be an estimation of the state of the branch below that feeder with the corresponding uncertainties, these are published through MQTT to be received by the upper layer.

- *Low Voltage* (LV) level
  The LV area is the portion of grid below an MV/LV substation. At this level the results from the concentrator level SE are received through MQTT, when these are available the LV SE algorithm starts using this data to perform the LV SE, additional measurements coming from LV meters can also be used . Once the results for this level are obtained the routine proceeds like before by forwarding this data to the upper layer through MQTT.

- *Medium Voltage* (MV)level
  The MV area is the portion of grid below an HV/MV substation and that's the highest level in this layered SE. When the results from the LV SE are received the last algorithm is launched to obtain the overall SE, also in this case if some MV meters are deployed on the grid their measurements can be used to improve the estimation results.

This approach is completely consistent with the design of the meter as each meter contains the algorithms for SE at each level and they can remotely switched on and off according to the position of the meter inside the grid.

Chapter 5

Case studies and results

5.1 Testing environment set-up

Real Time Simulator (RTS) simulates a different of possible real-world scenarios. Before deploying on the grid any meters or control equipment we should test them in the real world, however this test may be harmful for the grid or needs uncommon or rare situation that may be hard or unsafe to reproduce. For example if we need to test a device that performs some action during a fault we should wait an unknown amount of time for an outage or we've to generate one, clearly both of this choice are not practical. Therefore when using an Hardware In the Loop (HIL) simulation we substitute the grid with the RTS that through its analog output ports let us measure the physical quantities like we would to in the real world and that allows to perform any possible test in a safe environment [22] To test the proposed solutions we set up an HIL environment shown in the figure 5.1. The IoT infrastructure was deployed on two laptops to act as MQTT brokers and as Device Catalog, the RPi has been connected to the laptop in a Local Area Network (LAN) with an Ethernet. Then the MQTT broker and the software for the DeC where launched on the laptop, Eclipse Mosquitto [13] was used as broker as it is lightweight and open source, meanwhile on the RPi the metering software was started. As explained in the section 3.3 when the application on the meter starts it automatically reads his settings that contains its ID and the IP address of the DeC, using this information the device can obtain its configuration from the DeC. As it is shown in the screenshot in figure 5.2 after this initial self-configuration the meter will periodically update the information about which algorithms has to run , this self-update mechanism has also the function to notify to the DeC that the meters is alive, indeed the time at which the last update was requested is stored by the DeC, this will be used to check if a meter is alive through the process described in the section 3.3. As shown in figure 5.3 the DSO or any authorized external service can access to the list of the meters that are alive through the Internet. The RTS employed for the tests is an OP-5600 from OPAL-RT®, it uses a software called RT-LAB®that emulates the electric network, the model used from this application is designed by using Matlab Simulink®. Simulink is a software that allows to model a large variety of systems, the electric grid in this case, by utilizing *blocks* to represent and act as particular part of the system and then connect them together to simulates the whole system. The grid model deployed for the test is the same used in [22] and it is a section of Turin's MV network

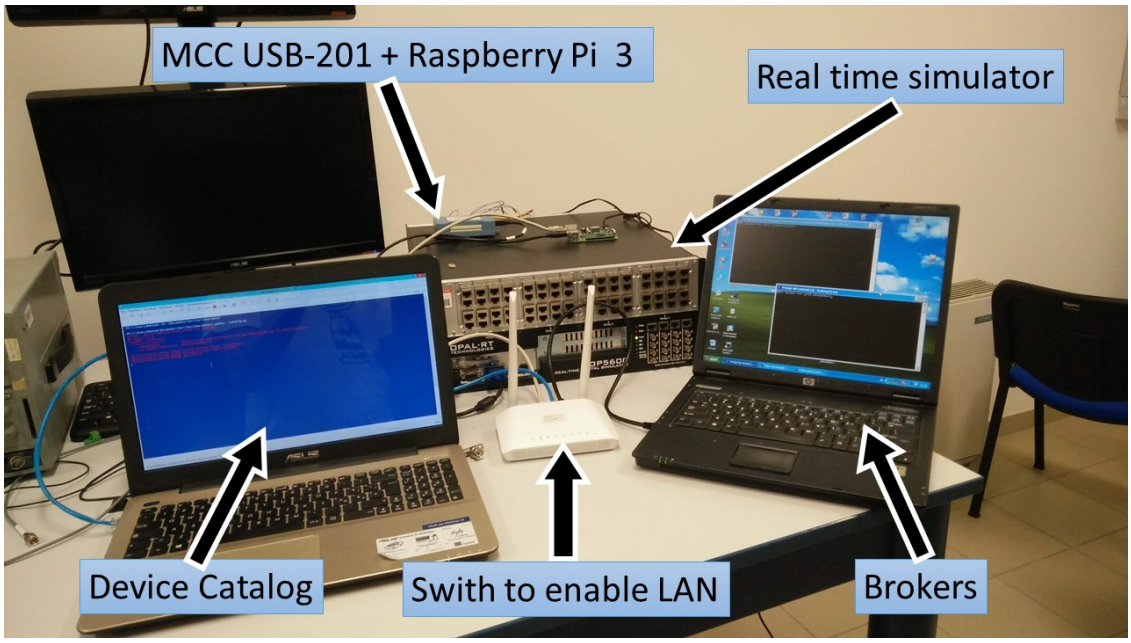

Figure 5.1. Set up of the test environment for outage detection and fault location

```
starting
Started thread for catalog update
<Response [200]>
{'Algorithms': {'Fault_location': 'True', 'State_estimation': 'True', 'Fault_detection': 'True'}, 'last_update': '1530284333.89'}
1530284333.89

subscried to the topic topology_update
Subscribed to all the topics
```

Figure 5.2. View of the console of the meter while performing self-configuration and self-update

as shown in figure 5.4. It is composed by three substation each of them powered by a transformer that brings the voltage from 22 kV to 220 V, from this substation a five lines originates that supply 49 branches. Once the model is load on the RTS the simulation is run and there are just 2 ways to interact with it: the RT-LAB software can launch, pause or reset the simulation while through a Simulink window it is possible to trigger some events by interacting with an user interface. This graphic interface is created by Simulink when the simulation is started, it has to be designed during the modeling of the grid and that it the only tools that can be used to interact with the element of the model during the simulation. The one used for this project is shown in figure 5.4, it shows the scheme of the grid and has some switches that if activated they trigger a fault in a specific point of the network, between the substations *204813* and *203975* at 177 meters from the first of these. By using this setup we tested the both the outage detection and the fault location algorithms as we could trigger the fault and check how the meter would react. The software infrastructure was emulated by 2 laptop: one running the Device catalog and the other running the two brokers for the MQTT communications. The overall setup is shown

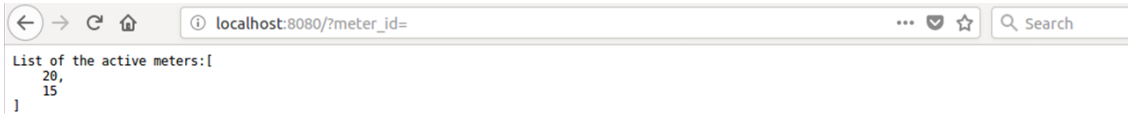

Figure 5.3. A browser window showing the list of meters alive

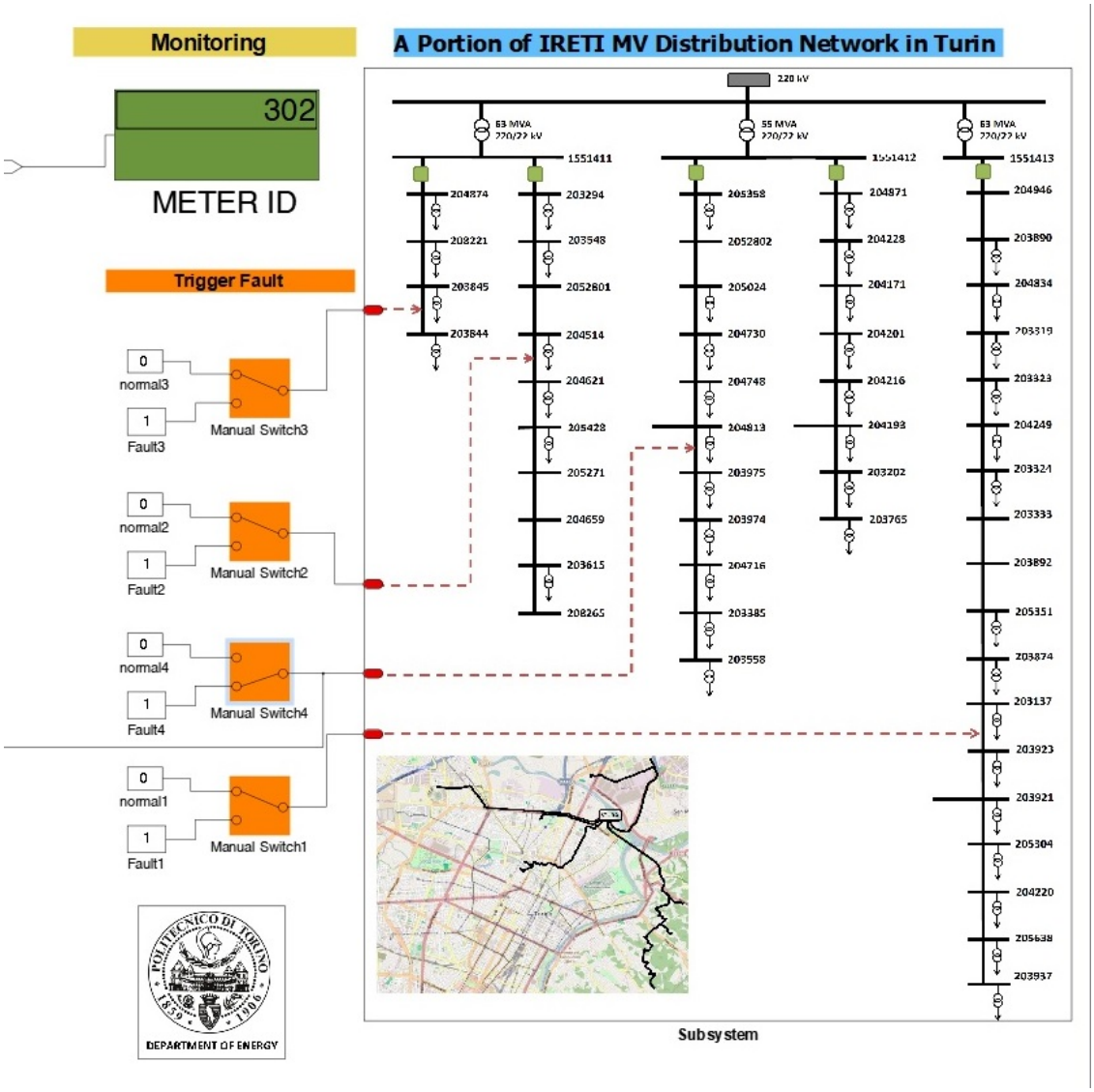

Figure 5.4. Graphic interface to interact with the grid during the simulation

in figure 5.1: the data collector part of the meter was connected to the analog output of the RTS while RPi, DeC and the two brokers were connected in LAN through a general Ethernet switch.

For the testing of the state estimation algorithms we had to proceed in a different way, as said in the section 4.3 of the chapter 4 to perform the SE we need the data collected from multiple meters and this approach was not feasible as we had just one device, therefore we used a model of a distribution grid made by 13 bus: one primary substation, 9 that feeds groups households, 2 bus that supply energy to industry and the last bus represent a generation plant. This differentiation between *residential* and *industrial* has been done also to generated different daily profiles of consumption to make the behavior of this virtual grid as realistic as possible. One day of activity has been simulated with a time step of one minute, at each step the correct value of the voltages and the current along the grid was calculated, a random noise with a normal distribution was added to this data to emulate the measuring error that could affect a real meter, these "noisy" measurements where then used to perform the SE. By adding this noise we could also test accuracy of the algorithm by confronting the result with the expected values. To test the performance the meter has been substituted to one on the building concentrator of the grid used in the simulation and its results were compared with the real values.

5.2 Outage detection results

To test the performance of the outage detection algorithms two main indicators has been analyzed: the time to recognize the fault and the difference between the number of fault occurred and the number of fault notified. While the simulation of the grid was running on the RTS a fault was triggered by acting on the switch no.4 on the Simulink window shown in the figure 5.4, the workstation stored the time-stamps at which the faults occurred while the meter stored the time at which the faults are notified according to the rules described in the section.
All the fault generated through the Simulink interface were correctly recognized by the meter and there was no case of false alarm (the meter notifies a fault that did not happened), that proofs the efficacy of the design strategy that has been used that declare an only if at least 0.5 s of abnormal measurements are collected as explained in the section 4.1. As it's shown the figure 5.5 the performance of the algorithm are quite constant, the median time needed by the meter to recognize a fault is 2.42 s, with a maximum of 2.49 s and a minimum 2.36 s. Even by taking in account also the delay due to the network, that in most case should be negligible, this result is a huge improvement respect to the time needed nowadays by a DSO to be aware of an outage situation. Indeed as explained in the section 4.1 in the conventional grid the process of delimitation of the area of outage is done through the phone calls received by the customer affected, the amount of time needed is extremely variable ad can take from minutes to even hours. In a scenario instead where a lot of device like the one proposed here are deployed on the grid all the meters that register the outage would notify it and since the DSO knows the position of each of them an accurate map of the outage zone could be realized in few seconds. This function alone could already speed up the traditional process of fault location as the field crew would have a precise indication of the area to patrol.

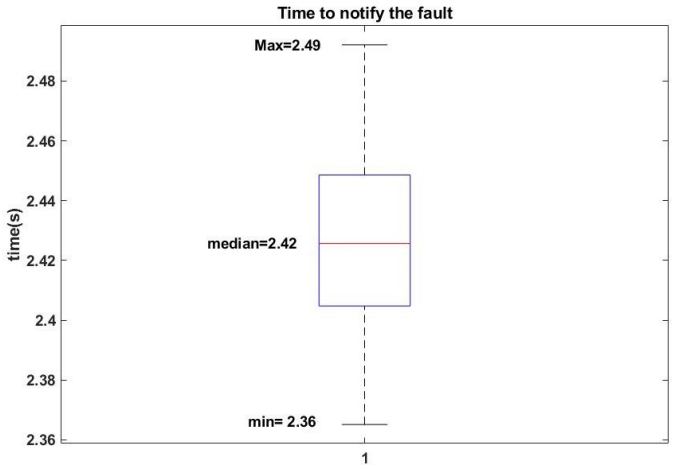

Figure 5.5. Boxplot of the outage recognition time

5.3 Fault location results

As we mentioned before the input of the algorithm is a 1 second window of measurements like the one showed in the figure 5.6, where the instant at which the fault occurred is in the middle of the measurements, it is possible to notice almost the exact instant at which the fault happens as we can see the voltage of the first phase that quickly reduce its amplitude while the sinusoid of the correspondent current rapidly increase. This data are processed as described in the section 4.2. Two parameter were analyzed to evaluate the performance of

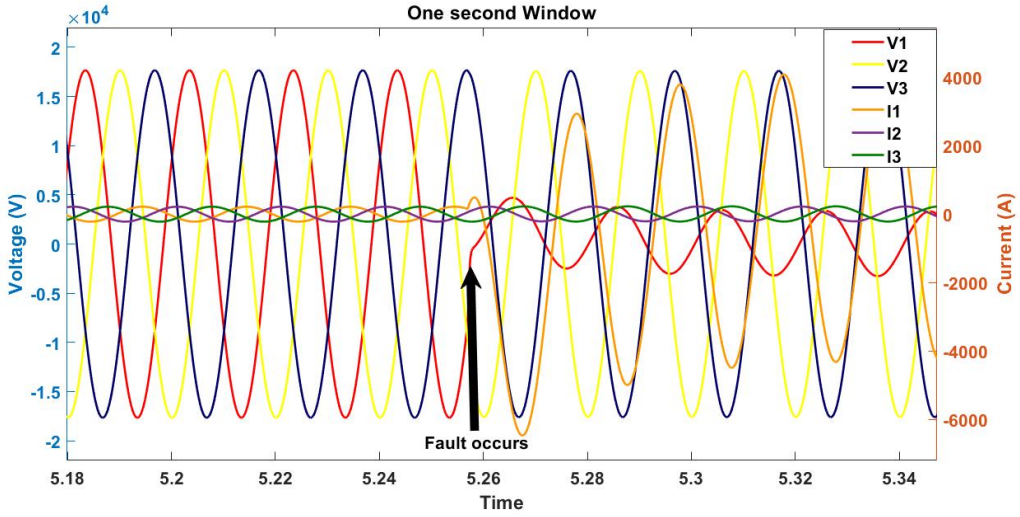

Figure 5.6. Input voltages and currents measurements for the algorithm

this algorithm: the computational time and the accuracy in the result of the meter. As the

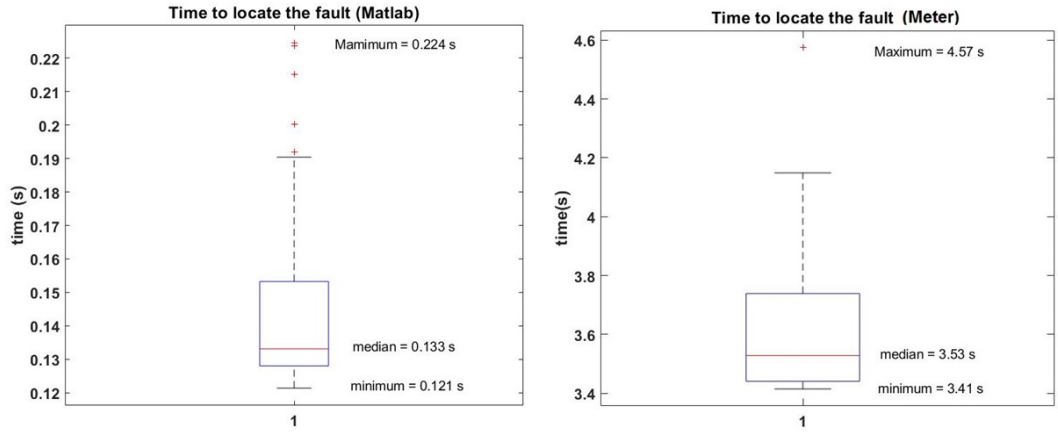

Figure 5.7. Boxplot showing the time performance of the meter for fault location compared to matlab

boxplot in the figure 5.7 shows the meter takes less than 5 seconds to locate the fault since this happens, with a median of 3.53 s and a maximum time of 4.57 s, on the same picture we can see that the performance of Matlab that is essentially 20 times faster, with a maximum time of execution of 224 ms. This difference among the time performances was expected as the RPi that is an hardware with limited , anyway this results opens opportunities for future works to trying to utilize other hardware that maybe supports Matlab, more powerful CPU or implement the algorithms in different coding languages. On the other hand on the figure 5.8 are displayed distances given as result from the algorithms gives as output the couple of substations where the fault occurred indicating the distance in meter from the first substation. The real location of the fault triggered in the model is around 177 meters from the substation with the ID 204183, considering the the median value of the results is 172 m with a maximum of 226 and a minimum of 118 m this results can be considered quite good. On this front the results of the meter are really similar to the Matlab ones, that is a really important result that despite executing the algorithm locally on a limited hardware the result would have the same accuracy if the algorithm was executed by collecting the raw data from the device, process that would imply to utilize the communication network that instead is left unused in the first case. As explained in the section 4.2 nowadays the location of the fault is done by defining the area affected from the outage according to the notifications of the customers affected by it, therefore the time to delineate this area and its dimension depends of different factors, like the time of the day or the dimensions of the faulty line itself. The figure 5.9 shows the errors made by the algorithm during the test, the overall performance is really good indeed the ability to find

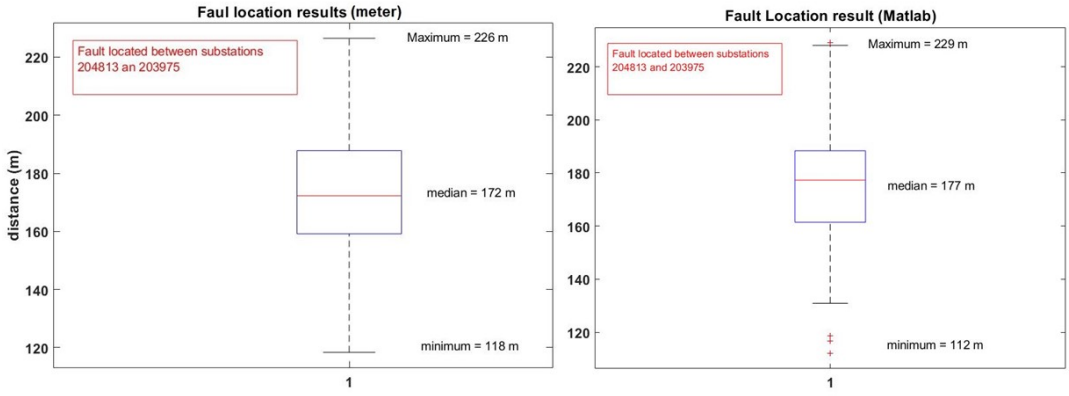

Figure 5.8. Boxplot showing the resulting distance from the first substation(204183) found by the meter and by matlab

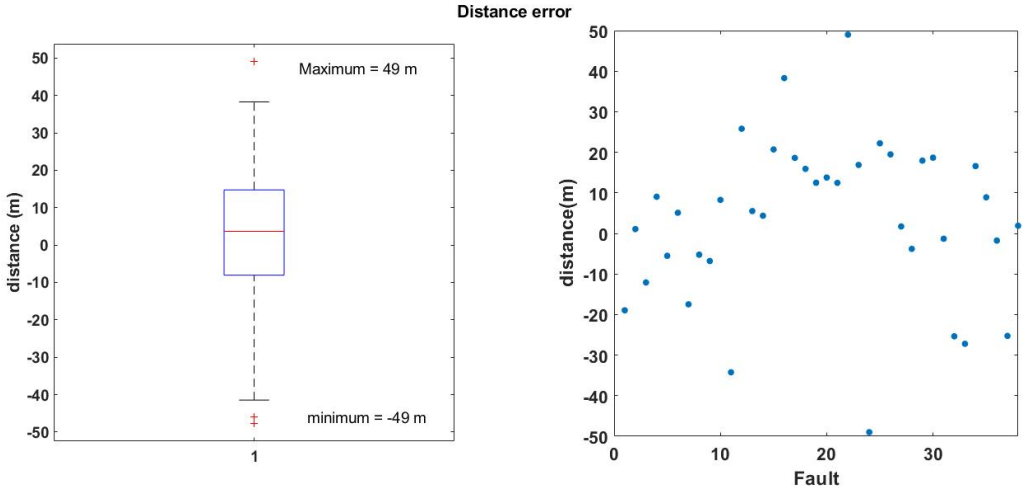

Figure 5.9. Error of the results of the Fault Location algorithm of the meter

the location of a fault with an error around ± 50 meters in less than 5 seconds represent a big improvement of the process compared to the standard process used nowadays that takes at least 10 minutes in the best cases [23]. Even with this error in the distance in the worst case the field crew has to check a portion of the line of few hundreds of meters while

with nowadays the area to inspect can of some kilometers. Moreover by speeding up this process of fault location and consequently the restoration of the line reduce the impact of the fault in the overall performance of the network, reduce the costs of the operations and increase the satisfaction of the customers.

5.4 State estimation results

The figure 5.10 shows a plot of the magnitude of the voltages estimated by the meter for the state of one of the point monitored by a virtual meters that provides the virtual measurements. The estimation is done every minute in one day. The figure 5.10 contains

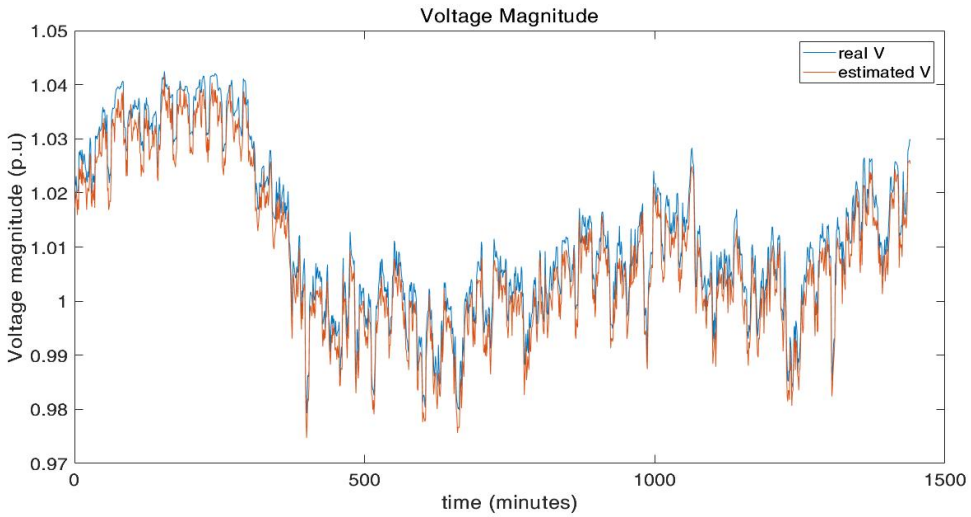

Figure 5.10. Comparison between the estimated voltage and real voltage

also the plot of the true values for the voltages and as we can see the difference between estimation is really small, indeed the maximum error of the results of the building concentrator SE turns out to be less than 0.84%. The results of the power estimation shown in the figure 5.11 have a similar accuracy with a maximum error of 0.58% One of the possible issues that could happen by using a CPU with a relatively low computation power as the RPi was that the time needed to perform the SE would be greater that one minute forcing to reduce the frequency at which the measurements from the virtual meters were collected and lowering the accuracy of the estimations. However as shown in the figure 5.12 the simulation demonstrated that there was not such problem indeed the computation time was always lower than 1 minute with a mean of 1.96 s and a maximum of 8.99 s. This results demonstrates the strength of the state estimation algorithm and shows how even by using low cost devices with limited capabilities it is possible to increase the reliability of the electric network and facilitate the grid management for the DSO, giving an updated and accurate overview of the operating conditions of the electric network.

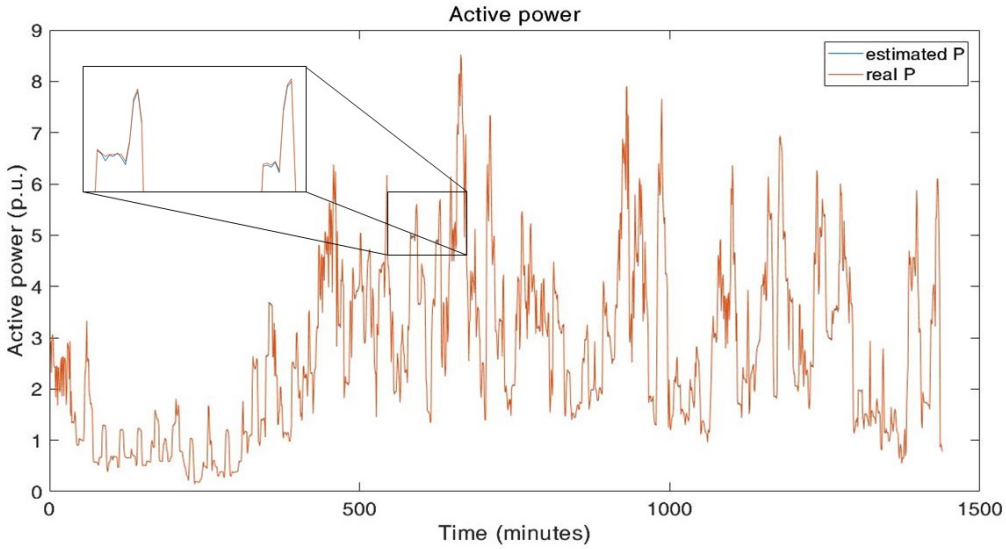

Figure 5.11. Comparison between the real active power and the estimated active power

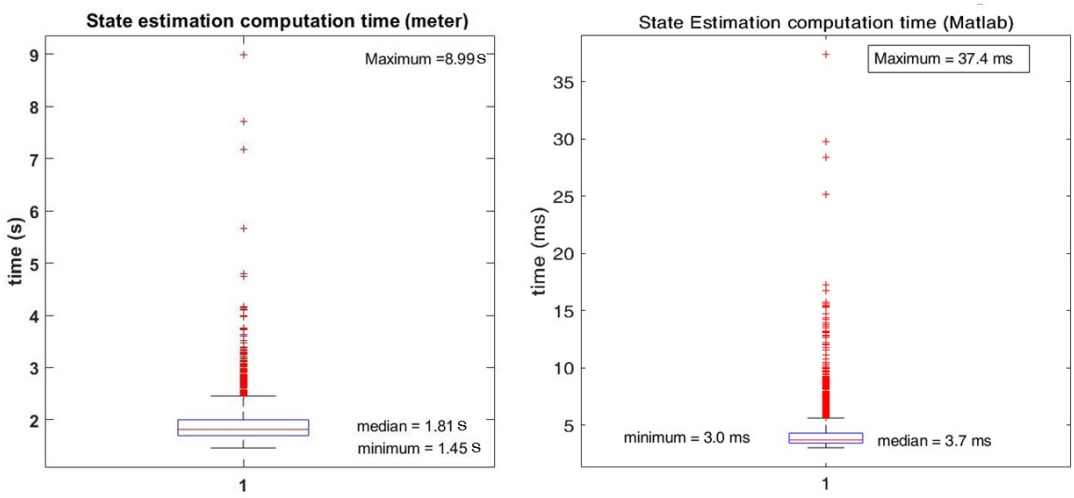

Figure 5.12. Boxplot of the computation time for the building level SE

Chapter 6

Conclusions and future works

The objective of this work was to design and develop a smart-meter and software infrastructure that could satisfy the necessities that are arising with the evolution of the smart grid. The meter has been realized with two open source and low cost devices, it's self-configurable and self-updated, it also implements 3 function that aim to increase the reliability and facilitate the management of an electric network. The software infrastructure is able to manage the meters that could be deployed in a smart grid, supplying the interested user with a constantly updated view of the operating state of the meters, it is also ready for the deployment of IoT actuators. The performance of the algorithms on the meter where tested by performing an HIL simulation with an RTS and the results showed that such smart-meter can drastically improve the reliability of the network ensuring fast response in case of outages and fault and they can increase the DSO's awareness of the status of the grid. The total time for for outage detection and fault location needed by the meter is less than 10 seconds in the worst case while the process performed nowadays takes 10 minutes in the best case speeding up the operation of FDIR operations. There are multiple way s to proceed in this project the first one can think of is to develop and deploy new algorithms that can automate as much as possible the FDIR operation or algorithms for Demand Response and DER management. It would be an interesting to create a meter that uses another CPU that supports Matlab to compare the performance Python on hardware with similar capabilities.

Appendix A

Acronyms

AMI *Advanced Metering Infrastructure*
DeC *Device Catalog*
DER *Distributed Energy Resources*
DFT *Discrete Fourier Transform*
DMS *Data Management System*
DSO *Distribution System Operator*
FDIR *Fault Detection, Isolation and Restoration*
HIL *Hardware in the Loop*
HTTP *Hypertext Transfer Protocol*
HV *High Voltage*
IED *Intelligent Electronic Devices*
IoT *Internet of Things*
LAN *Local Area Network*
LV *Low Voltage*
MQTT *Message Queue Telemetry Transport*
MV *Medium Voltage*
PLC *Power Line Communication*
PMU *Phasor Measurement Unit*
P/S *Publish/Subscribe*
RER *Renewable Energy Resources*
REST *Representational State Transfer*
RPi *Raspberry Pi*
RTS *Real-time Simulator*
SE *State Estimation*
WLS *Weighted Least Squares*

Bibliography

[1] Edoardo Patti et al. “Distributed software infrastructure for general purpose services in smart grid”. In: *IEEE Transactions on Smart Grid* 7.2 (2016), pp. 1156–1163.

[2] Pekka Koponen et al. “Definition of smart metering and applications and identification of benefits”. In: *Deliverable D3 of the European Smart Metering Alliance ESMA (available at www. esma-home. eu, members area)* 42 (2008).

[3] Abouzar Estebsari et al. “Emerging smart meters in electrical distribution systems: Opportunities and challenges”. In: (2018).

[4] Luis I Minchala-Avila et al. “Design and implementation of a smart meter with demand response capabilities”. In: *Energy Procedia* 103 (2016), pp. 195–200.

[5] Libiao Qiao, Xiaojun Liu, and Baochen Jiang. “Design and implementation of the smart meter in vehicle-to-grid”. In: *Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on*. IEEE. 2011, pp. 618–621.

[6] Andrea Angioni et al. “A Low Cost PMU to Monitor Distribution Grids”. In: *Applied Measurements for Power Systems (AMPS), 2017 IEEE International Workshop on*. IEEE. 2017, pp. 1–6.

[7] *EPM 7000 Power Quality Meter*. URL: `http://www.gegridsolutions.com/multilin/catalog/epm7000.htm`.

[8] devolo AG. *devolo G3-PLC Modem 500k Powerline adapter*. 2018. URL: `https://www.devolo.com/en/SmartGrid/Products/devolo-G3-PLC-Modem-500k`.

[9] *Enabling the Most Efficient, Reliable and Secure Delivery of Energy for Utilities*. URL: `https://www.networkedenergy.com/`.

[10] Roy T Fielding and Richard N Taylor. “Principled design of the modern Web architecture”. In: *ACM Transactions on Internet Technology (TOIT)* 2.2 (2002), pp. 115–150.

[11] Roy Fielding et al. *Hypertext transfer protocol–HTTP/1.1*. Tech. rep. 1999.

[12] Patrick Th Eugster et al. “The many faces of publish/subscribe”. In: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[13] *MQTT man page*. 2018. URL: `https://mosquitto.org/man/mqtt-7.html`.

[14] *MQTT*. URL: `http://mqtt.org/`.

[15] Biggee, Jason, and J Reid. *USB-201*. 2018. URL: https://www.mccdaq.com/usb-data-acquisition/USB-201.aspx.

[16] *Raspberry Pi 3 Model B*. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[17] S. Joseph and E. A. Jasmin. "Stream computing framework for outage detection in smart grid". In: *2015 International Conference on Power, Instrumentation, Control and Computing (PICC)*. 2015, pp. 1–5. DOI: 10.1109/PICC.2015.7455744.

[18] Marco Pau et al. "A cloud-based smart metering infrastructure for distribution grid services and automation". In: *Sustainable Energy, Grids and Networks* (2017).

[19] James Momoh. *Smart grid: fundamentals of design and analysis*. Vol. 63. John Wiley & Sons, 2012.

[20] Abouzar Estebsari et al. "An improved fault location method for distribution networks exploiting emerging LV smart meters". In: *Environmental, Energy, and Structural Monitoring Systems (EESMS), 2016 IEEE Workshop on*. IEEE. 2016, pp. 1–6.

[21] Alcir Monticelli. *State estimation in electric power systems: a generalized approach*. Springer Science & Business Media, 2012.

[22] Abouzar Estebsari et al. "An iot realization in an interdepartmental real time simulation lab for distribution system control and management studies". In: *Environment and Electrical Engineering (EEEIC), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1–6.

[23] Julio Romero Aguero. "Applying self-healing schemes to modern power distribution systems". In: *2012 IEEE Power and Energy Society General Meeting*. 2012.