Design of a Prototype for Inverter Monitoring with SunSpec Modbus Protocol

Ulysse Boudier

(Mandalay Yoma, PV mini-grid in Magway region)
Title: Design of a Prototype for Inverter Monitoring with SunSpec Modbus Protocol

TRITA-ITM-EX 2018:613

Ulysse Boudier

<table>
<thead>
<tr>
<th>Approved Date</th>
<th>Examiner</th>
<th>Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Björn Laumert</td>
<td>Rafael Guedez</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commissioner</th>
<th>Contact person</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract

In a context of growth of the company in which the project was conducted, and to provide a simple and cost efficient solution to PV system monitoring, the development of prototype using Modbus Sunspec is studied in this thesis. The project is developed using an engineering approach, which defines a clear scope, a schedule and detailed objectives for the prototype. After analysing the available options and assessing the needs for an own product, a complete design of a solution is developed. Divided in two blocks, it uses several technologies. On the hardware side, a Raspberry Pi is coded in python and is using an Ethernet cable to communicate with the inverters with the protocol Modbus TCP/IP. The hardware is interfaced with Emoncms, the software and an open source website specifically designed to monitor energy uses. On site tests could not happen partly due to a slowness of the PV market in Myanmar, however other tests with artificial data show good operations and stability of the prototype. Designed to be capable of interfacing with several brands of inverters and gather the monitoring in one only website, the prototype meets most of the objectives. To use it, it will need further improvements and validation of reliability, another approach could be to enhance the automation and control capabilities for smart grid designs.

Sammanfattning

Table of Contents

1 Introduction .................................................................................................................. 6
  1.1 Background and state of the art ................................................................................ 6
  1.1.1 Introduction to photovoltaics: potential and growth ............................................... 6
  1.1.2 The situation in Myanmar .................................................................................... 7
  1.1.3 Technological introduction to PV systems ............................................................. 10
  1.1.4 Why is it important to monitor PV plants? ............................................................. 11
  1.1.5 Practical forms of monitoring ............................................................................... 12
  1.1.6 Mandalay Yoma and the needs of for a new solution ............................................. 12

1.2 Scope .......................................................................................................................... 15

2 Objectives ...................................................................................................................... 16

3 Methodology ................................................................................................................ 17

4 Schedule ......................................................................................................................... 17

5 Design, implementation and tests of the prototype ......................................................... 19
  5.1 Inverter to hardware communication ....................................................................... 19
  5.1.1 Introduction to the Modbus protocol ................................................................... 19
  5.1.2 SunSpec Alliance and the inverter models ............................................................ 20
  5.1.3 Inverter analysis: Modbus TCP/IP selection ......................................................... 22
  5.2 Hardware selection .................................................................................................. 23
  5.2.1 Requirements .................................................................................................... 23
  5.2.2 Arduino ............................................................................................................... 24
  5.2.3 Raspberry Pi ........................................................................................................ 24
  5.2.4 Overall comparison and selection ...................................................................... 25
  5.3 Server selection ........................................................................................................ 26
  5.3.1 Introduction to web servers ................................................................................. 26
  5.3.2 Requirements ..................................................................................................... 27
  5.3.3 The OpenEnergyMonitor project and Emoncms ............................................... 28
  5.4 Coding of the hardware and interfacing with the software ........................................ 29
  5.4.1 Coding requirements and Python ....................................................................... 29
  5.4.2 General approach ............................................................................................... 29
  5.4.3 Code explanations: general idea, parts and additional implementations ............... 30
  5.5 Internal implementation and tests .......................................................................... 38
  5.5.1 Setting of the hardware ...................................................................................... 38
  5.5.2 Setting of a local web server .............................................................................. 38
  5.5.3 Setting of the code ............................................................................................. 38
  5.5.4 Internal tests ....................................................................................................... 39

6 Discussion ...................................................................................................................... 41
  6.1 Prototype summary ................................................................................................. 41
| 6.1.1 | Overview and total cost | 41 |
| 6.1.2 | Implementation | 42 |
| 6.1.3 | Capabilities | 42 |
| 6.2 | Review of the prototype | 44 |
| 6.3 | Product comparison | 45 |
| 6.4 | Review of the implementation | 48 |
| 6.5 | Value addition and future improvements | 49 |
| 7 | Conclusion | 50 |
| 8 | References | 51 |
| 9 | Appendices | 54 |
1 Introduction

Through the ages, information has always been an extremely valuable resource. Now that our world is evolving into numerisation, companies are looking for data scientist to analyse all this new type of information on the internet. But what makes information so important? It is through information that humans are able to analyse situations and react accordingly. Knowing as much as you reasonably can, is key to a successful investment. Hence, a complete analysis on the site, on the estimated electricity production, on the project costs, using all the available data is necessary to make the right decisions before installing a PV system. Then simulation and reality can differ, getting data about the actual operation of the system is as much a necessity given the large initial investment that was conceded.

This projects aims to facilitate the monitoring of PV systems by using automated processes and by monitoring the central component in the installations: the inverter(s). Used in all conventional installations, it is the key transformation point from a DC electricity to the transportable and usable AC electricity. Capable of communication, it is already a ‘smart device’ making it the essential and easiest component to monitor in order to get a view of the complete system. Standardisation in components allows a shift from proprietary languages only to the open protocol Modbus. This results in a standardised way of saving data and communicating between devices, offering the possibility to access the data in a simple and automated way.

This project is thought for Mandalay Yoma, a start up in Myanmar entering a completely new market for solar PV. In parallel, to working as a technical and sales engineer to develop C&I solar projects, I worked to develop a monitoring solution that would be used by the company in its future projects. As the company expects to quickly grow, this was an interesting area to develop a solution that could be used in all future projects.

Starting with a short recall of PV history and growth, the report will then quickly move towards Asia, and Myanmar to analyse the opportunities in this country for solar installation and present Mandalay Yoma, the company in which this work took place. We will then study the monitoring technologies and analyse the available market options, from inverter manufacturers or from a third party, to understand why developing an own solution would be a more interesting choice. This thesis will focus on the design and the implementation of an own solution to monitor inverters using the open protocol Modbus. The scope will set the limits while the objectives a precise the target. Moreover, a methodology and a schedule were established to develop this prototype using an engineering approach.

The first part will explain how the prototype was designed, by studying briefly the various technologies available. Comparisons between the different possibilities with regards to the scope and the objectives will help the decisions. The justification of the choices will follow the structure of the prototype, divided in two main parts: the materials, also referred as the hardware and the code of the application, referred as the software. After this design phase, an implementation phase will consist of the the set up of the prototype followed by tests to assess the performance and capabilities.

Finally, a discussion on the design will be conducted to analyse the results to the tests and more generally the performance of the prototype in comparison to the objectives and to other available products. The discussion will also cover the implementation and the engineering process used, to identify good practices or barriers to this project.

1.1 Background and state of the art

1.1.1 Introduction to photovoltaics: potential and growth

Solar energy is a resource whose energetic input through sun rays is largely sufficient to supply the world’s energy needs. It was calculated that the incident energy on the surface of the earth per year is more than 7500 times the annual global primary energy demand. (Goswami & Besarati, 2013) Solar energy present other advantages such as providing a better energy security because it is widely available over the world and as an alternative to fossil fuels in order to reduce greenhouse gas emissions leading to climate change. Although challenges remains regarding the intermittency of the production, the costs and the scaling, solar energy is considered as the potential leading energy source for the future. (MIT Study on the Future of Solar Energy, 2015)
Starting as a niche application for in the space industry, solar photovoltaics (PV) is now the largest net addition in electricity capacity with 164 GW added in 2016. (IEA, 2017). This exponential growth was made possible by first investments by Japan and European countries in the early 2000s and by the recent boost of investment by China since 2010. As a result, the average price per watt (Figure 1) fell drastically following a learning curve (Figure 2) and making PV a technology competitive with conventional sources in parts of the word. (REN21, 2016)

![Figure 1 - Drop in price for a module according to technologies and areas of the world](image1)

![Figure 2 - Solar PV module cost learning curve for crystalline silicon and thin-film](image2)

Projections are on the bright side and forecast growth for the photovoltaic sector in the next years. If this growth is still mainly due to support for photovoltaics and driven by government mandates and subsidies, it is very realistic to assume that the number of installation will rise substantially with grid parity in focus: the total capacity installed in the world could increase between 1721 and 2500 GW by 2030 according to the scenarios.

<table>
<thead>
<tr>
<th>Year</th>
<th>2016</th>
<th>2030</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Capacity Installed (GW)</td>
<td>303</td>
<td>1721</td>
</tr>
</tbody>
</table>

Table 1 - Growth in total capacity of PV installed in the world according to scenarios of the IEA and IRENA (IEA, 2016) (IEA, 2014) (IRENA, 2016)

However, these figures cover a large variety of situations: in South East Asia (ASEAN*), the target is to increase by 13% the share of renewables in the primary energy supply while at the same time the demand will grow by 49%. Meeting the demand is not the only challenge in this area, 65 million people don’t have access to electricity and many countries are still considered poor, this is the case for Myanmar (IRENA, 2018).

1.1.2 The situation in Myanmar

Myanmar has developed for its electricity the following (2017): hydro 3221 MW (61%), gas 1919 MW (36%) and coal 120 MW (3%) (MEE, 2017). The electricity generation for the year 2015/2016 is showed by Figure 3:

(*) members of the ASEAN: Brunei Darussalam, Cambodia, Indonesia, the Lao People’s Democratic Republic (Lao PDR), Malaysia, Myanmar, the Philippines, Singapore, Thailand and Viet Nam.)
Despite a good access to natural resources, the hydraulic potential is still largely untapped and over 90% of the gas extracted is exported, on the other hand the government is struggling to meet the growing demand, enhance the weak grid network and provide electricity to all the inhabitants. In Yangon, the economic capital, frequent power cuts have forced the users to rely on backup diesel generators. (Eurocham Myanmar, 2018)

Myanmar also presents one of the lowest electrification rate in Asia, with 43% of the population of over 53 million inhabitants lacking access to electricity in 2016 according to an estimation by the World Bank (World Bank). To tackle this issue, the National Electrification Plan (NEP) has been set in line with the Sustainable Development Goal #7 which is to “Ensure access to affordable, reliable, sustainable and modern energy for all” by 2030 with the support of the World Bank. The project, which should connect 7.2 million households with a total cost estimated at 6 billion dollars, involves: grid extension (most of the connections), pre-electrification with off-grid systems in rural areas “unlikely to receive it in the next 10 years or more”, permanent off-grid solutions in remote areas, see Figure 1 (DRD, 2017).

Renewables are the promoted technologies for the off-grid systems and solar is definitely the most prevalent option: whether as simple kit with the Solar Home Systems (SHS), or as mini-grid coupled with storage and/or diesel generators. Indeed, theses are often the cheapest solution available, easily scalable and with increased yield due to being in areas with large potential. A study lead by Solargis, shows high Global Horizontal Irradiation (GHI) in most of the country except the Northern mountainous locations, and particularly good values in central mainland which correlates with the population density, Figure 4.

Despite this, and the fact that the country has opened up for foreign investment there is only a few PV projects at early stages of development for large scale power production. High corruption rate and the lack of clear and transparent policies are two factors slowing down the development.
The company in which this project was realised, Mandalay Yoma, has developed solar energy as a way to provide reliable, affordable and clean electricity to people of Myanmar. Formed in 2014 but launching operations in 2016, it first targeted the mini grid sector as part of the NEP and successfully built 2 solar power off-grid systems in Magway region, providing 24/7 electricity to 3 villages with over 500 households and over 2000 beneficiaries. The current systems are 110kWp and 120kWp PV capacity, currently the largest mini-grids in Myanmar, and include batteries to store the excess power during daytime to provide supply during night. Mandalay Yoma is also building the complete distribution network in these villages. To ensure precise monitoring and analysis of electricity consumption, each household has a smart meter.

Mandalay Yoma goal is to provide electricity to over 500 villages and 30,000 households in the next 5 years, be a market leader in this space for Myanmar and enable the country to leap frog from central grid to distributed generation as the whole world moves in that direction. This is still a small portion of the huge Myanmar off-grid potential.

The 2nd area of development is the Commercial and Industrial (C&I) market. Mandalay Yoma offers ground and rooftop solar PV systems optimized for captive consumption of factories, warehouses, hotels and any other type of businesses. Coupled with electrical grid and diesel generators (used by most businesses as back-up during grid outages) solar PV rooftop systems enable reliable and cheap electricity ensuring reduction of electricity expenses including fuel expenditure and resultant pollution for these businesses. Like the off-grid market, Mandalay Yoma aims to develop an early entrant strategy and win a completely new market, with only few solar rooftops in operation: 117 and 92.6 kWp built in 2017 by Sunlabob, a competitor and a 59 kWp in 2017 by Myanmar Eco Solutions. (Mandalay Yoma) (Sunlabob, 2017) (Myanmar Eco Solutions)
Patterning with the Indian conglomerate Tata (more specifically the branch Tata Solar), Mandalay Yoma has yet to catch the opportunities and conclude significant growth to exit the start-up stage. The company is composed of 11 people set according to this organisational structure:

At Mandalay Yoma, I was responsible for the engineering of the C&I sector. Projects were handed over from the commercial team and the first task was to analyse the site through documents from the client, if necessary, obtain additional information and optionally to plan a site visit. From the data analysed, I was designing a system with a software, HelioScope, realising a production report for the offer and a cost estimation which was then passed to finance to produce a refined quotation. The last side of my activities was to handle the procurement and supplier relations for most of the off-grid and C&I components.

Although still undeveloped, C&I projects are profitable in Myanmar as grid parity (under some assumptions) is reached. The main factors for this are the good GHI available in Myanmar, high inflation and discount rates in favour of self consumption, the relatively high cost of electricity for industries (the price rise along with the number of units consumed and is increased by the cost of fuel for diesel generator backup) and the sharp reduction in cost of PV systems over the last years. As a result, most projects show internal rate of returns with 2 digits and a return over investment after approximately 10 years.

The Responsible Business Fund (RBF), is a 3-years initiative part of the Danish Development Assistance to Myanmar, which aims to “increase the competitiveness and responsible behaviour of Myanmar enterprises” by offering grants to small and medium enterprises (SMEs) in 7 areas of projects. Solar energy falls into one these areas, offering a possibility to finance 65% of a project up to 59,000$ and enabling for small scale projects (typically up to 70-80 kWp) to benefit from a less than 5 years return of investment. (RBF, 2018)

The PV monitoring project for this thesis was realised in parallel of these activities, and was scaled with the type of project ambitioned by Mandalay Yoma: small and medium PV systems for the C&I sector sold as a CAPEX model or PPAs. Hence, the product developed should be easy to integrate to these projects.

### 1.1.3 Technological introduction to PV systems

Solar photovoltaics, currently the main technology used in solar energy, consist of cells typically made of silicon (Si), layered and doped to enable the circulation of a flux of electron when the cell is hit by photons (ie. sunrays). This transformation from solar energy to a direct current (DC) current is called the photovoltaic effect. A typical grid-tie system consists of the following:

- **PV Arrays(s)** (DC design)
- **Inverter (DC to AC)** (Power conditioner)
- **Distribution AC-AC:**
  - Electric Grid
  - (AC loads)

Figure 7 - Schematic of a grid-tie system (Guedez, 2016/2017)
Solar cells are ultimately grouped into panels (or modules) to give them a protection and for commercial purposes. Disposed in strings and connected in parallel they compose the **PV arrays**. Converting the DC electricity into alternative current (AC) electricity, **inverters** have a central role, and even more nowadays as they can also optimise the performance, protect the system, monitor and log data and events. Finally, the **AC distribution** system connects the system to the electric grid and, if there are, loads. The rest of the components are aimed for production (cables, mountings, …), protection (breakers, fuses, …) and other services (monitoring, performance, …). (Guedez, 2016/2017)

Other components can be found according to the type (on grid, off grid) and the scale of the PV system. Figure 8 shows a grid-tie hybrid PV/ diesel generator, the most common type of system found among the industry in Myanmar due to the poor reliability of the grid.

![Components of a PV installation for a factory](Mandalay Yoma C&I Brochure, 2018)

Monitoring systems are developed with the use of information technologies, they allow to collect and analyse certain number of parameters in the system. Their use was first mostly restricted to large scale power plant due to the complexity and additional cost of such installation. As PV systems became a mainstream technology, a diversity of options has developed making monitoring essential in most cases.

### 1.1.4 Why is it important to monitor PV plants?

The high investments used for solar installation drive this need for monitoring options in order to maintain the high availability of the systems. The second benefit is to be able to assess the performance of the plant. The more detailed is the monitoring, the more understanding it can give on the issues and external factors affecting the system. Research purposes, to give a better knowledge of the technology, feedbacks on how the plant was modelled, designed or built and its operation, improvement the current practices, are many other reasons for monitoring precisely a plant. The key parameters monitored are usually PV array power, AC, grid power and PV array currents, for a more detailed performance analysis it is necessary to include at least environmental data like module temperature, ambient temperature and solar irradiance. (Anwari, Dom, & Rashid, 2011)

Functions of the monitoring systems can be divided in 3 areas:
1. As a part of the operation management: monitoring tools are supposed to provide, present and pre-analyse data for operational optimisation of the plant. Monitoring is then often included in wider systems with control possibilities: Supervisory Control and Data Acquisition (SCADA) or for instance Energy Management System for hybrid applications. These systems are able to perform actions to ensure reliability and continuity of the supply. For grid-tie application, another application could be in the domain of smart-grids: every large power plant connected to the grid must comply the local grid codes where it is installed but as to date there is no active support to the grid. New system could control the power quality injected in the network. (Moreno-Garcia, et al., 2016)

2. As alarms: which is stressing any element that is outside its normal predetermined performance area. More detailed techniques include the analytical monitoring which is the automated acquisition of data in order detect in a timely manner any flaw in the system and address it before it can induce a production loss.

3. As reporting: the generation of report regarding the operations and/or the performance of the plant. Reports can stand as legal tools for the calculation of liquefied damaged and to validate the fulfilment of contracts (for instance with the EPC contractor). They can also serve for communication as promotional or educational purposes for users about the solar system. Most residential installations now include a form monitoring, as such installations do not come with frequent maintenance checks and because it is often included in a wider ‘Home Management’ system enabling to users to understand and act on the energy uses in their home.

1.1.5 Practical forms of monitoring

As described previously monitoring can respond to several needs, as the result the forms of monitoring will show various complexity according to the needs, usually getting more complex as the size of the project (and the potential losses in case of a component failure) increase.

The simplest form is the sole monitoring of the PV inverter without remote communications which is therefore done locally, by reading the display on the screen. Nowadays almost all parameters will be available and can be read on the inverter’s screen with values such as voltages, current, and power on the DC and AC side. With the advance of electronics, these functionalities will be available for most inverters at a residential or commercial scale (string inverters).

Additional information will require the installation of dedicated sensors (typically a small weather station measuring temperature, wind, pressure and solar insulation) increasing the complexity. Gathering all the measured data can require to set up a specific system acting as the main interface with the use of one or several communication protocols and inverter manufacturer often sell such devices. A local network can also be set to transfer information over short distances (Ethernet, Bluetooth) however, mobile networks and internet being easily accessible and for a low cost almost everywhere, monitoring forms with remote communication are now the norm. As most inverters have means of communication such as Wi-Fi, Ethernet plugs or RS485, their fabricants will offer online forms of monitoring as long as the device is connected to the web.

Micro inverters or DC to DC optimisers are alternatives to string and central inverters which can give access to a more detailed monitoring: at a module level but require one device per module (or pair of module) leading to an increase in costs. The question of the level of precision desired is also central for large scale monitoring systems: the granularity set in the measurement (module, string or inverter) will have a large impact on the cost of the monitoring system and choices need to be based on techno-economic analyses. For performance analysis the IEA recommends at least a string monitoring in junction boxes adding that for electricity yield measurements, energy meters or true-rms power meters should be used. Inverter-integrated measurements are declared “usually not sufficiently precise”. (IEA PVPS, 2014)

1.1.6 Mandalay Yoma and the needs of for a new solution

The usual size for a project at Mandalay Yoma is in the range of hundreds of kWp, whether it is for mini-grids in rural areas or for industrial systems connected to the grid, and with 3-phases string inverters. Project will typically have one or a few more inverters which all offer (in-built) monitoring options. Specific cases are lease to own or power purchase agreement models of sells which require a particularly reliable and possibly detailed level of monitoring for legal purposes. On other cases, the two purchasable options are: the inverter manufacturing companies with their solutions and 3rd party monitoring companies. SCADA
(Supervisory Control And Data Acquisition) systems are too complex and too expensive for this projects which only require alarms and reporting functions with no performance analysis.

**a) Inverter manufacturer systems**

Remote monitoring is offered by the suppliers for inverters of their brand once the it is connected to internet. This will also include a software to view and manage the plant. The low costs (for inverter only monitoring there is usually no additional cost) and the ease of installation and integration are advantages of such solution. Another advantage is the possibility to add other devices in the product range offered, such as weather stations or data loggers (Figure 9) to connect sensors and interface monitoring tools with larger SCADA systems or online applications.

![Figure 9 - Commercial offer with a data logger (VSN700-05) centralising the monitoring system (ABB)](image)

On the other hand, these solutions come with evident constraints. Manufacturer have historically and still use proprietary languages inside the inverters: for instance, Aurora for ABB, YASDI for SMA and Conext for Schneider. There is a lock-in effect which force users to buy only tools from the inverter manufacturer and making it very complex to directly interface them with a 3rd party monitoring system. With ABB this can be done by buying advanced data logger versions or a specific communication converter.

The second constraint is on the data availability and software interface ergonomy. Monitoring is not the core business of inverter manufacturers and the quality of services proposed can vary. The accessible data by this monitoring is often not complete in comparison to what the inverter can actually measure: for instance, the monitoring system installed in Mandalay Yoma’s two rural mini-grid projects can only transfer one parameter from the inverter which is the input power from the DC arrays. Such tools are also constrained in terms of interface design and in the customisation of data analysis or alarm reporting.
Finally, each manufacturer has a different website, software and proprietary language, in case of various brands of inverter among one project or over several projects there will be no possibility to have a standardised monitoring on a unique webpage. Although in the recent years, the Modbus protocol, non proprietary and used in many industrial application, is increasingly installed in inverters through the SunSpec project, opening a way for 3rd party monitoring.

b) 3rd party monitoring

A large range of companies sell monitoring services for PV plants allowing more choices for the users. They often promote the flexibility of their solutions with devices able to connect to a wide selection of inverter or sensors, and software offering customizable dashboard, charts, alarm, reports and options such as mobile interface or weather overlays. Their business model is based on a one-time hardware sale (data logger) and a subscription for the online monitoring (software), making it a rather expensive solution considering the lifetime of PV plants.

A more detailed comparison over costs and functionalities of monitoring options will be made later in this report but a quick market survey showed solutions in the range of [5000 – 10000] € for monitoring systems from 3rd party vendors. This includes annual fees to use the monitoring software over 20 years and is considering solutions designed for commercial to small utility scale projects (100 kWp to MW range).

As 3rd party monitoring providers are usually smaller companies of the solar industry, it is also harder to assess their reliability. What will happen if the company goes bankrupt? What are the services that remain? For projects with a long lifetime such as solar PV, these questions have more an increased importance and can be drawbacks to choose 3rd party solutions.

c) Summary and own solution

Another concern is on the ownership of the data which can apply with both external monitoring solutions. How to ensure that the data is not used by the vendors for commercial purposes as it can evidently have a value? An example use could be to estimate PV production with data from actual power plant rather than on weather simulations.

Table 2 summarise advantages and drawbacks on both solutions:
<table>
<thead>
<tr>
<th></th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inverter Vendors</strong></td>
<td>Simplicity of installation</td>
<td>Lock-in effect</td>
</tr>
<tr>
<td></td>
<td>Ease of integration with other products from the same vendor</td>
<td>Low customisation available</td>
</tr>
<tr>
<td></td>
<td>Relatively low costs</td>
<td>Only works with their inverters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restriction on the data collected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data ownership</td>
</tr>
<tr>
<td><strong>3rd Party</strong></td>
<td>More choice</td>
<td>Expensive</td>
</tr>
<tr>
<td></td>
<td>Wider services and customisation</td>
<td>Reliability as a company for the next 25 years?</td>
</tr>
<tr>
<td></td>
<td>Possibility to gather data from several different inverters</td>
<td>Data ownership</td>
</tr>
</tbody>
</table>

Table 2 - Advantages and drawbacks for the monitoring solutions offered by the inverter vendors and 3rd part company

Mandalay Yoma has the ambition to develop numerous projects, which can be with various brands of inverter, and therefore would benefit from a central monitoring with a unique web interface or software. The best solution would be the use of a 3rd party company but this would come at a higher cost. The last option would be to develop Mandalay Yoma’s own solution: this is possible thanks to the development of similar projects with the use of the internet of things and the SunSpec project.

Several individuals have developed and programmed their own solution to monitor their (residential) installation. They use tools such as Raspberry Pi and Arduino as gateways connected to the inverters and/or other sensors. Advanced projects are: the DIY Solar Powered Home which can monitor the PV output, load and battery state of charge, the Raspberry Pi Solar Logger by Brian Dorey monitoring charge controller and many other sensors in a ‘smart home’. (Hubbert) (Dorey, 2012)

However, they do not use the recent advancement of the SunSpec project. The SunSpec Alliance is an organisation whose goal is to establish an open, easy-to-adopt solar data exchange architecture and standards that enable the free flow of data between existing software products, this is done using the Modbus protocol. Hence, the product developed will be inspired by the previous homemade projects available on the internet and use the recent standardisation developments of the Modbus communication protocol in solar inverters.

### 1.2 Scope

The aim of this thesis is to develop a prototype to monitor PV systems: small to medium scale, with 1 to 10 inverters, lower than 1 MWp and at least 10 kWp. The level of detail in the monitoring will be first restricted to the inverters only for simplicity and cost reasons but it should provide all the information measured by the inverters. The functionalities will be limited to monitoring and analysis of the data, no control over the inverter will be set up but this can be an area of future development. The solution designed is the complete solution including the hardware, a data logger connected to inverters and able to send data online and the software, an online server receiving the data, presenting displays and analysis. To avoid unnecessary work that has already be done, this project should use open source resources available online and focus only on gathering the different parts and interfacing them together to achieve a functional prototype.

Robustness and security of the product, whether physical as it is supposed to be used onsite with inverters or from a computer science point of view to prevent errors and hacks, is not the primary concern although it should be part of the final discussion. The prototype’s purpose is to test functionalities and the reliability of the system, further ameliorations would be to have a reliable enough solution that can be used onsite and the addition of functionalities.

The questions that this thesis will answer are the following. What are the technologies used for this product? How was it developed? What were the difficulties in the development of this prototype? How does it compare with the existing solutions? Moreover, inverters tend to have more and more maximum power point trackers, MPPTs, (the recent SMA Core I has 5), in particular in the commercial sector, to
accommodate complex roofs and optimise the production. This means that that voltages levels, currents and power could be read for 5 different zones, allowing a “block” level monitoring. Can a more detailed monitoring be developed using only the inverter data? Can this improve the fault prevention? What is the final value and scope of application of the prototype? What are the areas for improvement?

These interrogations are combined in one main question:

How to monitor a medium scale PV system using the standardised SunSpec Modbus protocol to communicate to the inverters on-site and transfer data to an online server in an efficient, simple, and economical way using an engineering approach?

2 Objectives

The prototype should have the listed under functionalities:

- On the hardware part
  - Interface and read the data measured by inverters
  - Ability to interface with various brands of inverters
  - Ability to interface with several inverters
  - Store locally the data
  - Transmit the data online to a central server

- On the server part:
  - Receive the data from the hardware part
  - Verify the authenticity of the hardware
  - Parse and store the data
  - Provide a display of the data
  - Allow the analysis and processing of the data (alarms, …)

And follow general requirements, to be:

- Efficient
  - Attention to the expected lifetime of the prototype
  - Attention to the robustness and security of the programmed device and server
  - Ease of integration (ability to interface with selected brands of inverter)
  - Reliable, according to a report by the IEA PVPS with guidelines on monitoring systems availability of the monitored data should be at least 99% and an availability of less than 95% show poor quality of the monitoring system (IEA PVPS, 2014)

- Simple
  - Attention to the installation time which should be minimal and to the possibility of remote configuration
  - Modularity of the functionalities, with the possibility of improvements and additions of new functionalities later
  - Independence (a low reliability on external services that could be interrupted in the next 25 years)

- Economical
  - In comparison to the available products on the market
  - Attention to the resources used (budget, a time allowance, schedule)
3 Methodology

Figure 11 - Methodology process for the thesis

The general approach was first to get a general understanding of the potential solution for the prototype and then later slowly getting more into the details. As each choice in the design is constraining the future options available to solve future issues, it was important to have an iterative process during the first two steps. After reviewing existing technologies and solutions, a more precise idea of the design can be formed. By making choices and going more in detail in the design of the prototype, new issues appear leading to a new phase of documentation which can change the design. This also allowed to keep this process dynamic, with documentation phases only when it was necessary to understand more in detail the technologies involved.

Furthermore, designs of the hardware and the server are interdependent: a decision in the hardware can limit possibilities in the way the data is sent to the web server, adding another cyclic process in the general workflow.

The implementation phase is much more practical however this does not mean that the design will not change, as set up and tests that are operated can lead to minor changes. Most of the work takes place during the set-up: in addition to initializing the tools used, writing the programs necessary for the operations of the prototype was the most significant part.

After the implementation, an analysis will be conducted answering the following questions:

- Does it fit the needs?
- What are the value, strengths, weaknesses of the prototype?
- How does it compare with the market?
- How can it be improved?

4 Schedule

A schedule was set with a GANTT chart, in order to show the tasks and represent their durations. This chart also ensures the methodology is followed and the hierarchies in the task (dependencies, delays, priorities) are taken into account. Finally, it is a way to monitor the advancement of the project.
Figure 12 - Gantt chart for the development of the monitoring prototype
5 Design, implementation and tests of the prototype

5.1 Inverter to hardware communication

The prototype must be able to connect with the inverters, the easiest and most promising way is to use the standardisation developments made by the SunSpec Alliance with the Modbus protocol.

5.1.1 Introduction to the Modbus protocol

Modbus is widely used by the industry, typically to transmit signals from instrumentation and control devices back to a main controller or data gathering system. As an open protocol it can be used without paying royalties. The data is sent by bits (a zero or a one) at a defined rhythm: the baud rate.

It can be used with various physical connections, buses or networks which will define the layer: “Modbus over X” with X being an application layer. Figure 13 shows examples of the main layers available, for inverter communication the main layers are: Modbus over TCP/IP, Modbus over RTU & ASCII (Master / Slave type), Modbus over UDP (variation of Modbus TCP).

![Figure 13 - Stacks for the application layer messaging: Modbus (Modbus-IDA, 2004)](image)

Independently of the layer, a simple Protocol Data Unit (PDU) is used specifying how the data is transferred. According to the use of the layer, additional fields are used, this is Application Data Unit (ADU). Modbus is using request/reply methods to communicate and the actions are specified by function codes.

![Figure 14 - General Modbus frame for data transmission (Modbus-IDA, 2004)](image)

A request initiated by a client toward a server will contain information about the server address, the format of this address is layer dependent but the basic rule is to ensure that each device has a unique address. The function code is used to specify to the server the action to perform, it is coded in 1 byte ranging from 1 to 255 decimal. Data fields are additional information for the server to use according to the function code received. The data field can be empty, and in this case it means that the server does not need any additional
information, the function code alone is specifying the action. For instance, to request data from a device, the function code will ask to read some registers and the data field will specify which.

In case of correct operation, the server will initiate a response containing the same function code and in the data field the data response, for example the data requested. In case of an error, the data field contains an exception code that can be used by the client. Finally, additional error checks can also exist according to the ADU.

The Modbus protocol sets how the data is identified: the data is stored in registers (16 bits) and each of them is addressed by a unique number ranging from 0 to 65536 (maximum number of integers with 2 bytes). The organisation of this data is referred as the data model and is completely vendor specific. Modbus maps are document listing where the data is stored (address of the register for instance, 40194), the name of the data (voltage at phase 1, in volts), and how it is stored (as a 16 bits-integer, in a read only register).

The use of the Modbus protocol allows 3rd party device to connect with the inverters as long as a “register map” is provided to understand how the data is organised. However, the lack of standardisation is a real issue: for programming, a different code has to be written for each inverter model as all can have a specific mapping and it is necessary to rely on documents provided by the manufacturer which are not always updated at the same rhythm as the inverters are released.

5.1.2 SunSpec Alliance and the inverter models

The SunSpec Alliance, is developing information standards with Modbus, targeting the whole chain of photovoltaics (inverters, charge controllers, meters, etc.) with the long term goal to increase value and performance and to drive innovation.

A standardised structure was set for all devices based on two or more information models: a Common model and one or more Standard model(s) or Vendor model(s). Each model being defined by an identifier (ID) and a length (L), allowing to easily skip models with unknown IDs.

a) Common model

The Common model serves as an identifier to signify that the data is complying the SunSpec standards and to identify the device from a manufacturing point of view. The data is organised according to the following table:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Address Offset</th>
<th>Block Offset</th>
<th>Size (in registers)</th>
<th>Name</th>
<th>Label</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>0</td>
<td>1</td>
<td>ID</td>
<td>Common</td>
<td>1</td>
<td>uint16</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>Mn</td>
<td>Manufacturer</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>18</td>
<td>16</td>
<td>16</td>
<td>Md</td>
<td>Model</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>34</td>
<td>32</td>
<td>8</td>
<td>Opt</td>
<td>Options</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>42</td>
<td>40</td>
<td>8</td>
<td>Vr</td>
<td>Version</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>50</td>
<td>48</td>
<td>16</td>
<td>SN</td>
<td>Serial Number</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>66</td>
<td>64</td>
<td>1</td>
<td>DA</td>
<td>Device Address</td>
<td>uint16</td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>67</td>
<td>65</td>
<td>1</td>
<td>Pad</td>
<td>pad</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 - Common model representation (SunSpec Alliance)

A value of 1 for the ID will identify this model as the Common model, for this model the length is 66 registers (this does not include the ID and L). The data starts at the address offset 2, after the first two registers, with the Manufacturer name (Mn), followed by the Manufacturer model (Md) and 3 values to identify models options, firmware versions, serial number. The Device Address (DA) is protocol specific, for Modbus it will consist of the Modbus address; finally, Pad is a reserved field used to round up the model to an even number of registers.
b) Standard models or Vendor models

One or more Standard or Vendor models follow the common model; both are structured in the same way but Vendor models do not need a review by SunSpec allowing more flexibility to the vendors. By creating these models, they can extend existing models and create space for specific values. The canonical structure of both models is explained by Table 4:

<table>
<thead>
<tr>
<th>Component</th>
<th>Size (16-bit Registers)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model ID</td>
<td>1</td>
<td>Assigned value</td>
</tr>
<tr>
<td>Model Length</td>
<td>1</td>
<td>Length of data portion (fixed length block + total repeated block length)</td>
</tr>
<tr>
<td>Fixed Length Data Block</td>
<td>Data block length</td>
<td>May be zero length</td>
</tr>
<tr>
<td>Repeating Block</td>
<td>Number of instances * instance length</td>
<td>May be zero length</td>
</tr>
</tbody>
</table>

Table 4 - Standard and Vendor model canonical structure (Sunspec Alliance)

Fixed blocks are made for data points appearing only once in the model while repeating blocks allow the model to include several times the same type of point. However, there must be a limited number of repeating blocks allowing the calculation of one Length (L) for the model.

SunSpec also specify several rules for the data points:

- A point must be among the accepted the data types (see Annex: List of data formats SunSpec, p54)
- For each point it must be stipulated if this is a Read only (R) or a Read/Write (RW) register
- For each point it must be stipulated if this is a mandatory point or if the vendor can return the value “Not implemented” for this data points
- When relevant, the value must be within the good unit which is specified in the model
- Floating values can be represented with 2 registers (32 bits): one for the “integer” digits and one for the “fractional” digits
- Alternatively, floating values can be represented by integers (one register) with signed scale factors applied (one more register)

c) Inverter models

Available on SunSpec’s website, an Excel document lists all the standard models and the data points associated. For this project, the relevant models are in the “100 series: Inverter”:

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>Inverter (Three Phase)</td>
<td>Include this model for three phase inverter monitoring</td>
</tr>
<tr>
<td>113</td>
<td>Inverter (Three Phase) FLOAT</td>
<td>Include this model for three phase inverter monitoring using float values</td>
</tr>
<tr>
<td>160</td>
<td>Multiple MPPT Inverter Extension Model</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 - Standard models for devices listed as inverters (Sunspec Alliance)

This models confirm that basic values used by the inverter such as input voltage and current are stored and can be accessed. Similarly, in the model 160 used for multiples MPPT, voltage and current can be read for each MPPT which could allow a “zone monitoring”. Nevertheless, this data (\(V_{dc}\) and \(I_{dc}\)) is marked as optional and could be not implemented by manufacturers meaning this has to be tested on-site case by case or to be verified, if available, with the register map of specific inverters.

SunSpec standard models reduce greatly the complexity to access the desired data in the inverter, as there is only one register map, up to date and which will be accurate for any SunSpec compliant model of inverters. As of now, several manufacturers have models which are SunSpec compliant, among which are some of the
largest: ABB, Fronius, Kaco, Schneider and etc. A complete list can be found on SunSpec website or by analysing the inverter datasheets.

The way the data is organised is also perfectly fitted for reading applications such as data logger because the structure allows to correctly parse it. A reading device can be programmed to ignore models it does not understand thanks to the ID and Length parameters and will be able to continue its execution with models it can understand.

### 5.1.3 Inverter analysis: Modbus TCP/IP selection

The selection of inverters includes the following: the current suppliers, potential suppliers and the main inverter manufacturers of the market however it is far from exhaustive. For each brand, one inverter in the range of 20-60 kWp was selected (see Annex: List of inverters for the comparison, p54). By reading the datasheets associated with each model, a comparison on the physical connectivity and the Modbus capacities shows that Modbus TCP over Ethernet is the most widely implemented. This confirms the standardisation efforts by SunSpec pushing Modbus (TCP/IP) as the main communication protocol for solar applications.

#### Table 6 - Comparison of the physical connectivity in the inverter selection

<table>
<thead>
<tr>
<th></th>
<th>ABB</th>
<th>Fronius</th>
<th>Huawei</th>
<th>Kaco</th>
<th>Schneider</th>
<th>SMA</th>
<th>SolarEdge</th>
<th>Total “Yes”</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS485</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Optional</td>
<td>Yes*</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes**</td>
<td>Yes*</td>
</tr>
<tr>
<td>WLAN</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>USB</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

#### Table 7 - Comparison of the Modbus implementation in the inverter selection

<table>
<thead>
<tr>
<th></th>
<th>ABB</th>
<th>Fronius</th>
<th>Huawei</th>
<th>Kaco</th>
<th>Schneider</th>
<th>SMA</th>
<th>SolarEdge</th>
<th>Total “Yes”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus RTU</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Modbus TCP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>6</td>
</tr>
<tr>
<td>Modbus UDP</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes*</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Modbus ASCII</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>0</td>
</tr>
</tbody>
</table>

TCP/IP, which stands for Transmission Control Protocol and Internet Protocol, are stacked together over Modbus allowing more flexibility and control for the transmission of the data. The advantage of using a layered protocol stack, is to be able to divide functions and that any layer can be changed without affecting others. Figure 15 explains the function of each layer:

![Figure 15 - Modbus TCP/IP over Ethernet layer diagram](image-url)
5.2 Hardware selection

Internet of Things (IoT) refers to projects with objects which are connected over internet or networks and as consequence can communicate with each other or to humans though an interface. IoT projects have a hardware part, the physical objects and a software part, the programs implemented to run the project.

On the hardware side, to simplify the prototyping, standard and easily available product such as sensors, circuit boards, and microcontrollers are developed and used. In this section, we will analyse the requirement of the prototype and select accordingly the components.

5.2.1 Requirements

The data logger is the interface with: inverters through Modbus TCP/IP over Ethernet and the server through an internet connection so it needs to be able to communicate with these two objects.

Additional requirements are the following:

- Simplicity, the computational power of the device does not need to be high as it will perform simple task of interfacing, collecting and sending the data but some simple calculation can be needed.
- Connectivity, the wider the capabilities the easier it will be to develop this prototype, to connect other devices (sensors) or make changes: Ethernet, WIFI, analogue entries, SD/USB ports
- Programming capabilities: with recognised programming languages. Because this project aims to use standardised protocols and to inspire itself from similar projects, the use of famous languages offers the possibility to use libraries (collection of pre-programmed functions), API (interfaces) or existing blocks of code.
- Ability to store data, local storing is necessary if the connection to the server fails, this can be done with external storage (USB or SD cards for instance).
- Extendable capabilities such as:
  - The ability to host a web server: a web server is basically a program able to handle external request over internet, as such it would be possible to set the server for monitoring a PV installation device on the same hardware as the data logger.
  - Ability to modify the hardware for adding functionalities (camera plug, sensors, etc.)
- Remote accessibility, the data logger is intended for onsite implementation, having such an access allows an easier set up of the device and the possibility to fix issues remotely in case of problems.
- Low electrical consumption, for energy efficiency purposes.

The two most widely used device (called boards) for prototyping are Arduino and Raspberry Pi. These two boards have also proved to be able to connect with inverters and act as data loggers: a few project on internet explain how this was done. (Dorey, 2012) (Heinz Pieren, 2014) We will compare capabilities and select the board that will fit the best the requirements of this project.
5.2.2 Arduino

Arduino is a microcontroller produced by the company Arduino. It is also the name of the software used to program the board. Both are open-source, allowing anyone to use them, create ameliorations or compatible accessories: a large community has used Arduino over time, enriching its capacities. The software is available on most operating systems (Windows, Mac OS, Linux) and one the main advantage is that it can easily support configurations and background communications between software and hardware because it was conceived to be used with Arduino specific hardware.

![Arduino Uno board](image16.png)

The main use of Arduino is to act as the control of small devices such as lamps or sensors. The hardware is available with several versions with different sizes, characteristics or specificities making each model a better of less fit for a specific field of application. The Arduino is definitely hardware oriented and one the advantage is the ability to easily extend the hardware capabilities by using shields, to add for instance a touchscreen, radio transmitters, Wi-Fi connectivity, etc. Most are “plug and play” making their integration easy.

5.2.3 Raspberry Pi

Raspberry Pi is a miniaturised fully functional computer, which comes with Linux and computer components: memory ram, a processor and a graphic card. As computers, it has the ability to run multiple programs at the same time. To use the Pi as a normal computer it is necessary to connect a power block, a screen, a keyboard and a mouse however, it is not made to execute applications requiring significant processing power like video games.

With regards to the IoT, it includes two rows of integrated input / output pins, making it easy to connect electronic components directly to the board. Like Arduino, it is possible to add shields to enhance functionalities but this is usually more complex. There is also a good community backing Raspberry Pi with tutorials, forums and explained projects to get help.

The boards, sold by the Raspberry Pi Foundation, evolved a few times and diversified versions were made available: from the lowest single cost board (RPi Zero) to more power and connectivity (RPi Model 3 B+).

![Raspberry Pi Zero (left) and Model B+ (right)](image17.png)
5.2.4 Overall comparison and selection

Table 8 compares both boards in relation to the requirements listed above:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Arduino</th>
<th>Raspberry Pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>Easy to get started, few programming skills are required. Can run one program at a time. Can run on batteries no shutdown issue</td>
<td>More complex Can run multiple programs at the same time. Requires (like a computer) a shutdown process and is vulnerable to power cuts</td>
</tr>
<tr>
<td>Connectivity</td>
<td>Doable internet access: with Ethernet or WIFI shields Analogue to Digital ports</td>
<td>Relatively easy to very easy to access internet: built-in Wi-Fi and Ethernet RJ45 ports (works like a computer for the set-up) Only digital ports</td>
</tr>
<tr>
<td>Programming capacities</td>
<td>Only Arduino or C/C++</td>
<td>Almost any language Linux software available</td>
</tr>
<tr>
<td>Data Storage</td>
<td>Limited to 32 kB (Uno) Connecting a USB or SD storage is possible but requires some work</td>
<td>Built in Micro-SD port</td>
</tr>
<tr>
<td>Extension capabilities</td>
<td>Can host a web server (more complex) Large choice of shields to add functionalities</td>
<td>Can easily host a web server</td>
</tr>
<tr>
<td>Remote access</td>
<td>Yes, with the use of an Ethernet shield</td>
<td>Yes, via SSH and via an application</td>
</tr>
<tr>
<td>Low electrical consumption</td>
<td>Yes : [0.3 – 0.6] W</td>
<td>Higher but still low: [1 – 2] W</td>
</tr>
<tr>
<td>Costs</td>
<td>10 – 20 $</td>
<td>5 $ (Zero) – 40 $ (model 3 B+)</td>
</tr>
</tbody>
</table>

Table 8 - Comparative table between Arduino and Raspberry Pi

The Arduino is cheaper but this does not take into account the need to add shields for the WIFI or Ethernet. The data logger will be used on site but the access to a reliable electricity source should no be a difficulty (either from the grid or from the mini-grid itself). Furthermore, the Raspberry Pi has an overall better connectivity with the possibility to have WIFI, a SD storage without the need to attach and install these components; it is also easier to control remotely.

The Arduino is less complex than the Raspberry Pi and will be easier to set up, but it seems more suitable for machine to machine communication and control, and electronics which is not exactly needed for the data logger. Indeed, the interaction with the inverters (or sensors) will be a simple request of data and the amount of inverters or sensors interfaced is limited. On the contrary for limited hardware interaction but a more complex software side or if there is a need to connect to internet, the Raspberry Pi is usually easier to use.

For these reasons, a Raspberry Pi 3 Model B+ will be used for this prototype. Only the latest models: Raspberry Pi 3 (B and B+) and Zero (W and WH) have WIFI in built however only the Model 3 has an Ethernet port. This model is also more user friendly: it will be easier to install it and operate tests.
5.3 Server selection

On the server side, the prototype needs a web server able to receive the data, store it and display it with a graphical user interface (GUI). This involves a several technologies such as communication protocols, programming languages and a database which will be explained in this section.

5.3.1 Introduction to web servers

A web server is the combination of: a physical machine named the host, which is connected to the internet and the web server software, run by the host, which is the programme based on the HTTP (Hypertext Transfer Protocol) responsible for providing the web content to the user.

The user, called a client, can be an internet browser (Chrome, IE, Safari, etc.) or any other machine using the HTTP. Clients and servers communicate using request and responses with the HTTP and similarly to Modbus these communications are usually encapsulated with other protocols, usually TCP / IP. On the client side, the response from the request is programmed in frontend languages, usually HTML, Javascript or CSS and translated into a more graphical interface by the browsers. (1&1, 2017) (Openclassrooms, 2018)

![Layered structure of the web: client frontend languages, HTTP, transport and networking protocols](Mozilla, 2018)

On the server side, also known as backend, the web server software operates the server. The most used web server software is Apache, developed as an open-source, meaning it is completely free for personal and commercial use. Moreover, it is considered reliable and is frequently updated. It is a good choice for the project because, as one of the most used web server, several explanations, kits and courses explaining how to set up Apache exist on the Internet. (Apache) (1&1, 2017)

The second part of the backend, is the server language, operating with the server software, it is the programming language specifying how the web server should work. A few server side languages and example of applications are: PHP (Facebook), Python (Google), Java (Twitter), Ruby (Airbnb), C# (Microsoft). There is no language server considered better than the other, each has its specificities, and all of the listed are widely used. For this project, PHP was chosen for these reasons: it has a large community on the internet which can help in case of issues, it is an easy language to begin with and is often used in tutorials for web server programming, finally, it is a good combination with MySQL. (Openclassrooms, 2018)

MySQL is a database management system using the Structure Query Language (SQL). It is used for storing data in tables. Websites can be classified in 2 types:

- Static websites, which are programmed only using client languages such as HTML, CSS and Javascript (frontend). They can not update their content automatically and interact with the user, this is the reason why only a few websites are developed this way and mainly to act as the front display.
- Dynamic websites: they are more complex and called dynamic because their contents interact with the user. Most websites are now dynamic.
The addition of a database in the backend, which is referred as having a database driven website, is the most common way to develop dynamic websites. The content of the website resides in the database and can be pulled to create web pages, simple examples of functionalities made possible by databases are: the management of users and profiles, management of messages in forums, actuality feeds and etc. Figure 19 shows the flow of requests and responses for a database driven website:

![Database Driven Website Diagram](image)

*Figure 19 - Operation of PHP and database driven website (Yank, 2009)*

The selection of MySQL, is based on the fact that on the contrary to the most deployed database management system, Oracle, it is free to use. It is widely used and included in many tutorials with PHP. Indeed, the combination PHP and MySQL is one of the most commonly used for free web server programming. (Openclassrooms, 2018)

### 5.3.2 Requirements

The web server for the prototype must be able to receive data from the hardware and display it for the user. Ideally, it can offer analysis tools such as graphical display, calculation possibilities with the data or handle alarms. However, programming from scratch a complete website is long, requires a lot of knowledge and experience with the tools previously described. Two solutions are possible:

- To create very simple website acting as a crossing point for the data. It would receive the data as a file (for instance, text or CSV) and provide the user with the ability to download any file from each project. For security reason, this page would only be accessible with a password set beforehand. The analysis would be conducted later, with computer data analysis tools like Excel or Matlab.
- To use one of the already available templates of websites and interface it with the hardware. The website would be completely functional and ideally created with the idea to monitor solar energy production. It needs to be open-source or available at a reasonable cost.

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Own website</strong></td>
<td>More control and possibility to develop it exactly as wanted</td>
<td>More complex to develop</td>
</tr>
<tr>
<td></td>
<td>Easier interfacing with hardware</td>
<td>Less functionalities</td>
</tr>
<tr>
<td></td>
<td>Free</td>
<td>Time consuming</td>
</tr>
<tr>
<td><strong>Template website</strong></td>
<td>More functionalities</td>
<td>Could be difficult to interface</td>
</tr>
<tr>
<td></td>
<td>No coding required</td>
<td>Less control and customisation possible: if a modification is required, understanding how can be very difficult</td>
</tr>
</tbody>
</table>

*Table 9 - Comparative table between building our own website and using a pre-coded template*
It was decided to use a template website rather than creating from zero our own website because the OpenEnergyMonitor project with Emoncms provides most of the requirements for this project and was developed with solutions targeting the disadvantages previously listed.

5.3.3 The OpenEnergyMonitor project and Emoncms

OpenEnergyMonitor is a project, as stated by the name, providing open-source energy monitoring tools to facilitate the implementation and understanding of energy systems, notably sustainable energy. OpenEnergyMonitor is supported by a large community and constantly in development, it has been improved over the years making it a popular tool for the monitoring of energy: the founders reported more than 3500 active users for the online web application Emoncms in April 2018. The products proposed by this project are:

- Pre-assembled hardware units (EmonPi, Emonbase, EmonTx and EmonTH). Based on Arduino or Raspberry Pi boards, the main product is the EmonPi which can monitor 2 single phase AC circuits, set up a local storage and a local web server with Emoncms. EmonTx and EmonTH are hardware sensors nodes including devices such as AC current transformers (CT), temperature and pressure sensors; both can communicate remotely with the EmonPi. Finally, the Emonbase is conceived as an alternative for the EmonPi: it is a web gateway, acting as a data logger only for the EmonTx and EmonTH.
- Emoncms, an open source web application to process and visualise data. It is available for local installation on the EmonPi (or any Raspberry Pi) to set up a local monitoring but also online (Emoncms.org) for remote monitoring. A mobile app is also available for Android and IOS.
- Other services such as documentation regarding the energy, tutorials for the set-up of their product and a forum for the community to exchange.

The EmonPi and other associated products, have the same function as the hardware part of the developed prototype, however for several reasons they do not fit the needs. Indeed, it is designed for residential applications only: CT have maximum inputs of 100A and 24kW, and are not made to monitor a 3-phase system. Moreover, it does not have the functionality to interface with the inverters using Modbus protocol. (OpenEnergyMonitor)

Emoncms

The website is based on Apache web software, PHP language and MySQL database, in order to make it function on your own, these are required. It is able to view inputs (a piece of data) which are pushed to the web site and record them as feeds: time-series in a database used to store all the data. Several options are possible to display the data: there is a graphical interface, cumulative kWh energy graphs, csv export possibility, histograms, dashboards and email alerts. Emoncms offers many options and can be quite complex, however thanks to the documentation and the forum it is relatively easy to use it and more importantly to interface it with a data logger.

The data can be pushed to the web server using various methods but a precise format and an API key are needed. API which stands for Application Program Interface, is an interface whose role is to simplify programming tasks. In this case, it is used as the interface to send the data to Emoncms. The API key allows to link the data sent to a profile among the profile list on Emoncms website. Each account created receive a unique key which can be used as a parameter when posting the data to authenticate its source.

Emoncms was chosen as the web server application for this prototype because it saves the need to create a website which would be too much time consuming and because interfacing it with the hardware is relatively easy. Its field of application is exactly what is required and it has many useful functionalities which can be set as desired allowing flexibility for the website. Finally, it is open-source so it can be used freely as a local server or to create our own server, however the online version (emoncms.org) recently changed its business model implementing a pay-pay-use model to finance the running costs of the web servers (hosts).

As the hardware part (on site, physically connected to the inverter) and the user browsing the Emoncms web application will be in 2 separate networks, an online set up is required. The first solution will be to publish to Emoncms.org and pay-per-data usage while the second is to set up our own host with the Emoncms web application. The second option involves buying a host or renting shared hosting (with size and performance in mind: storage, processor, etc.), setting up security, an IP address, a domain name before
it can be accessed online. As both are costly, this will only be done when the local implementation tests are concluded.

Using Emoncms for commercial projects is possible as it comes under the GNU Affero General Public License also allowing to make changes and distribute the original or modified work (OpenEnergyMonitor). A more detailed interpretation of what can be done with this license is shown in Figure 20:

![Summary of the Affero General Public License possibilities](Tldrlegal, 2014)

### 5.4 Coding of the hardware and interfacing with the software

#### 5.4.1 Coding requirements and Python

The Raspberry Pi, in order to act as a data logger, needs to execute a program previously coded and set for execution. When plugged to power, it must start and automatically run a script which would operate as long as the Raspberry Pi is powered. The program must initially load the configuration settings, then fetch the data from the inverter and send it to the server. Other aspects of the code include enhancement of the stability and security, handling of errors and logging of useful information.

This program can be written in any Raspberry Pi supported language, and here, Python was used. The main advantages of Python are that it is easy to learn and used by many programmers. It is among the already installed languages on most Raspberry Pi kits and is probably the reference for Raspberry Pi projects. Finally, it has many libraries (set of programmed functions or tools) in several areas. The most interesting one for this project is the pySunSpec library which provides access to Modbus TCP devices and easy device scripting.

#### 5.4.2 General approach

To develop the program, the approach was to start by following for a few days an online initiation course about Python in order to get basic syntax and understanding of the language. Python was then installed on a computer with IDLE, the integrated development environment included with Python. Creating the code on the computer allows an easier development and the ability to test it before transferring it on the Raspberry Pi. The version of Python installed on the computer is the same as the one available on the Pi to prevent compatibility issues.

The data logger, requires a form of data storage for several reasons: to avoid the loss of data if the connection with the server is not working (internet issue, web server not online, etc.), to act as backup and to send averaged values over a short period of time rather than sending instantaneous values. As the program should always be running, one option could be to save the data as arrays or other python objects on the RAM, however this would slow down the execution with a large amount of data and everything would be lost if the Raspberry Pi accidently shuts down.

A better solution is to physically write in the memory (hard storage) the data with the use of csv files (or text files). This is simple to implement thanks to a library available for python and will be very useful for
tests as the csv files can also be opened with Excel. This was chosen rather than a database because of the simplicity it offers.

The first phase of coding started by the development of the basic functionalities (inverter connection, storage and pushing of the data) using a simple syntax and secondly with improvements by adding more functionalities, robustness and security.

5.4.3 Code explanations: general idea, parts and additional implementations

5.4.3.1 Global structure

The code of this project is divided in two files:

- Main: this is the general code executing the infinite loop in order to get the data, store it and send it.
- Settings: this is where the key parameters for the projects are inserted. The idea is to modify only this file when implementing the product in a new location.

When running the script of Main, the settings are called in the script using the module “import”.

Main

First, the program is initialised: checks are made to verify that the settings are correct, a log is created, variables are initialised and updated using existing files to continue the previous execution without losing data. This initialisation is important to ensure no errors will happen later and in order to have a smooth restart after a shutdown.

Then, the program will enter a loop, infinite as it has no condition to stop, for the connection to the inverter, acquisition of the data, storage of the read items in a csv file and sending the data to the defined web server. At the end of the loop a pause is made using a timer, before it starts again, this prevents an overload of the database and is coherent as all measured parameters have variation time in the range of several minutes. To further reduce the internet data usage and pressure on the storage, the possibility of calculating an average value for the last X minutes, then store and send only this value to the server was added. Indeed, a general recommendation by the IEA PVPS, is to collect data “every second or faster” and average the values over 5 to 15 minutes for final storage (IEA PVPS, 2014). This creates two paths for the execution of the main loop and is handled using indicators and counters. Functionalities and separate parts of the main code will be described later in chapter.

Settings

The file Settings was created to facilitate and shorten a new implementation. It can also serve to disable or modify some functionalities. The main parameters declared in this file and imported by the main are:

- Identification values: project name, project number
- Parameters to adapt to inverters type and read the data*
- Local directory: where to save the data
- Parameters for the functionalities: number of values before an average is made, if the connection is lost set up of an incremental increase of wait time before trying to send again, etc.

* This part was not fully developed because the prototype could not be tested with an inverter (see later Read Data function and part 5.5.4 Internal tests)

Figure 27 summarise the the process for the code: Main and Settings as well as the loops used in the main script:
5.4.3.2 Inverter connection

Due to a lack of accessibility to inverters, the inverter connection could not be tested leading the function Read_data to be unfinished, however the intended general idea, parts of the code and the implications on the settings parameters will be explained in this section. The actual implementation for test purposes is explained in part 5.5.4 Internal tests.

The PySunSpec library allow to easily connect to a device via TCP (or RTU) and create an object, a SunSpecClientDevice, composed with classes based on the models founds in the device. Each points of the class are points within the model, with the scale factor, if existing, already incorporated in the value.

The documentation from the library explains how to create a SunSpecClientDevice and use it:

```python
1. d = client.SunSpecClientDevice(device_type = client.TCP, slave_id = 1 , ipaddr = '169.254.255.255', ipport = 502)
2. print(d.models)
   >>> ['common', 'inverter', 'nameplate', 'settings', 'status', 'controls', 'volt_var']
3. print(d.common.points)
   >>> ['Mn', 'Md', 'Opt', 'Vr', 'SN', 'DA']
4. print(d.common)
   >>> common(1):
       Mn: SunSpecTest
       Md: TestInverter-1
       Opt: opt_a_b_c
       Vr: 1.2.3
       SN: sn-123456789
       DA: 1
5. print(d.common.Mn)
   >>> SunSpecTest
6. d.close()
```
Using the simplicity of the object generated by the library, the actions of the code for implementing a new device and operate the Read_data function are:

**Set up, first time only**
- Create a device with the correct parameters (client.TCP, Modbus id, IP address and IP port)
- Print the models to see which are implemented (command n°2)
- Print the points or the complete models desired (command n°3 and 4)
- In the Settings, create an array with the names of the models and points that should be read
- Close the device (command n°6)

**Normal operation, in the Read_data function**
- Create a device calling the parameters in the file Settings
- Using a loop, access all the points defined in the file Settings (command n°5) and save them in an array or a dictionary
- Close the device (command n°6)
- Save the array in a file ("To_average") using the function Store_data and exit the function

### 5.4.3.3 Data storage

The function Store_data stores data as a csv file when inputted an ordered array of data and a file name as a string. It uses the Settings to know where to store it and to get the correct data name in the array. The data names are used to include a header in the csv file.

As this function is called twice in the main loop, to save the data which is later averaged in Read_data and later to save the averaged data, a test is initially made to check which is the case for each call. To avoid the pre-averaged data to be saved twice, if the inputted data is of the type None, which is what is returned by the function Average_data when the number of points to average is not reached, the function Store_data will exit. In the other case, it executes normally.

To save data as a csv file, two cases can occur:
- the file does not yet exist, in this case it is created, the header line is inserted and finally the data is added
- the file already exists, in this case the data is just added at the end of this file. The order of the array inputted in the function Store_data must be consistent over time or the data will be mixed up.

- **Figure 22 - Flowchart of the function Store_data**
The python module csv is imported, allowing to open, read and write easily csv files. The syntax is the following (file does not exist):

```python
with open (path, 'a') as myFile:  #Complete path (including file name)
    writer = csv.writer(myFile)
    writer.writerow(Settings.header)  #Write the header from the Settings
    writer.writerow(my_data)  #Write the array of data
```

The parameter ‘a’ signifies that the file is opened for writing purposes and that the cursor should be positioned at the end of the file: when creating a new one it does not matter, a ‘w’ for write and truncate would also work however if the files exists, this is necessary to avoid the suppression of all the previous data.

### 5.4.3.4 Average of the data

The data is averaged to prevent an overload of the database and because the parameters which are measured in a PV system do not have large variation over a few seconds but rather several minutes. The logic for this function is described by Figure 23:

![Flowchart of the function Average_data](image)

To check if the number of points to average has been reached, the file ‘To_average’ is opened and the number of lines in it is counted, if it inferior to the desired number of points (declared in the Settings) the function is exited. If the function is exited without returning the averaged array, it will return a None value which is used by Store_data and Send_data to know if they should execute or exit as well.

If there are enough data points in the file ‘To_average.csv’, after the data is read, stored in an array and averaged, the file is deleted using a simple command line from the imported module ‘os’:

```python
os.remove(Settings.directory + " /To_average.csv")
```

### 5.4.3.5 Push and post of the data to the web server

#### Data formats to send the data

The Emoncms website details the API input processes, helping to send the data with the correct parameters and in the correct format. Two ways are possible to send the data:

- With an input/post: to send a single update.
- With an input/bulk: to send historic and/or multiple data in a single update.
The first method is the most appropriate to send the live averaged data, among this method several methods and formats exist but the recommended one is using a Json format:

<table>
<thead>
<tr>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>curl --data &quot;node=1&amp;data={power1:100,power2:200,power3:300}&amp;apikey=MyApiKey&quot; &quot;<a href="http://localhost/emoncms/input/post">http://localhost/emoncms/input/post</a>&quot;</td>
</tr>
</tbody>
</table>

Table 10 - Recommended Json format and input/post method to send data to Emoncms (OpenEnergyMonitor)

The function `Send_data` receives as input an array of data, converts it into the correct format and send it, the code is the following:

```python
for i in range(n_data):  #Create a dictionary (keys & values)
    data_dict[Settings.header[i]] = my_data[i]

data_json = json.dumps(data_dict) #convert the dictionary in JSON

input_data = {'apikey':Settings.API_KEY,  #add the api key
              'data':data_json,
              'node':Settings.project_number}

r = requests.post(url = Settings.API_ENDPOINT+"/emoncms/input/post", data = input_data) #Send the post request
request_data = r.status_code 
request_answer = r.text 
print("The answer URL is:%s"%request_answer)
```

Looping over the data of the array and using the value names in the Settings, a dictionary is created, which is then converted into Json using the module ‘json’. Another dictionary named `input_data`, gathers all the parameters of the data: the API key, the data in Json format and the node before the request is sent to the web server. The request, using the python module ‘requests’, is composed of the url address (from the settings and terminated by “/emoncms/input/post”) and the data. This module also handles the responses from the post request: status code and text answer.

**Send_data**

The responses from the requests are essential to verify if the data was sent correctly, if it is not the case the data should be saved to be sent later, for this the function `Store_data` is used with the filename “Temporary_unsent”. As in the function `Store_data`, an initial check is made in the function `Send_data` allowing to know if there are data points to send or if it should exit as the data has not been averaged yet.

![Flowchart of the function Send_data](image)

---
**Send_unsent_data**

The function Send_unsent_data, is called every time in the loop of the main program, to send the data that could not be sent due to errors in the function Send_data. This prevents from loosing data if the internet connection is lost. For this function the data is sent using the input/bulk method:

<table>
<thead>
<tr>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>curl --data &quot;data=[[10,16,1137],[8,17,1437,3164]]&amp;time=1528217076&amp;apikey=MyApiKey&quot; &quot;<a href="http://localhost/emoncms/input/bulk">http://localhost/emoncms/input/bulk</a>&quot;</td>
</tr>
</tbody>
</table>

Where:
- the 1<sup>st</sup> number is the offset
- the 2<sup>nd</sup> number is the node id
- the rest of the numbers are data points (here node 16 has one value 1137)
- time allows to send the send the offset in absolute time format (Epoch: number of seconds since 01/01/1970)

**Table 11 - Recommended Json format and input/post method to send data to Emoncms (OpenEnergyMonitor)**

The module ‘requests’ is used again with the data passed as a string:

```python
input_data = {'apikey':Settings.API_KEY,  #add the api key
              'data':str(array_data),  #Convert the data as a string
              'time':offset_time,  #Epoch time when the data was 1st read
              'node':Settings.project_number}

r = requests.post(url =Settings.API_ENDPOINT + '/emoncms/input/bulk', data = input_data)
```

The logic of the code for this function is slightly more complex than the others: the function loops to read line by line the file 'Temporary_unsent.csv', to get the value offset_time and the data and send the request. If at some point the data can not be send, it is saved in a 2<sup>nd</sup> temporary csv file, and the functions stops trying to send the data to save the remaining lines in this file. At the end, this 2<sup>nd</sup> temporary file replaces the 'Temporary_unsent.csv' if it exists. If it does not, this means that all the data have been sent. Figure 25, explains this with a flowchart:
5.4.3.6 Configuration checks and error handling

The code described in this section aims to increase the robustness of the program, which means prevents errors, early terminations or unexpected actions. A code is described as robust if it can handle wrong or absurd inputs.

The creation of a log is essential to monitor the program and understand what is happening during the execution. Recording events and errors in the execution, also serves as the base to understand the weakness of the code and to improve it. The log is initiated using the python module ‘logging’ in the main program, before entering the infinite loop, and it is set to record events with the following format: date, name of the program, type of event, custom message. Here is an example if the function Store_data has an issue:

```
2018-06-05 18:54:35,684 - __main__ - ERROR - Error the data could not be sent
```

In order to keep the program running even if an error occurs, it was encapsulated with the use of Exceptions. Exceptions allow to intercept errors and handle them in a desired way, in the code they are used with the log to record the errors, perform actions before exiting the function and continue the execution of the main loop. The format is the following:
try:
    # normal code
except Exception:  # if an error occurs
    logger.error("Error message for this exception", exc_info=True)
    # code to execute if an exception happens (ie save the data, exit the function, etc.)

Here, all types of exception are intercepted but this can also work with specific types of exception (division by zero, error in the name, etc.). However, exception should be used carefully as critical errors that can damage the correct execution of the program should force the program to exit.

The variables declared in the settings are the point of interaction between the user and the program, they are essential for the operation of the code and if incorrectly inputted the program can not work. To prevent further error and be able to assess this error, a function Check_Settings perform tests on the variables, checking if every variable is declared and if the data types are consistent with what the program is expecting. This function is called in the main program before entering the loop.

### 5.4.3.7 Time functions and frequency of the actions

The python module ‘time’ is used for several actions:

**Raspberry Pi Uptime**

When executing the main program, the first action performed is to save the initial time. It is later user used to calculate the uptime for the execution and send it to web server along with the data. This allows to know for how long has the data logger been working correctly. To calculate the uptime, the current time is subtracted to the initial starting time.

**Monthly data savings**

The module is also useful to name correctly and create monthly csv files with the data. The date is called in the main loop and decomposed in order to save the final data (averaged) under the following name: ProjectName_Month_Year, where the project name is called from the Settings. Using monthly savings for the data allows to divide the data in several files, increasing the flexibility to manually access some data and reducing the weight of the individual files.

**Sample time in the loop**

In the infinite loop, at the end of the actions, the possibility to pause for a desired time is achieved by setting a timer. The measuring time is declared in the file settings and should be in the range of a few seconds to achieve the best sample precision. (IEA PVPS, 2014)

**Waiting time before an average / Number of points to average**

The waiting time before an average can also be set in the Settings. The number of points before an average is then calculated by dividing the the waiting time by the sample time for a loop. The function Average_data compares this number to the number of lines in the file 'To_average' to know when to average the data.

**Incremental waiting time for Unsent_data**

The function Unsent_data was using a significant amount of time and power from the processor as every time it was called (every loop), the whole file ‘Temporary_unsent.csv’ is read, sending a request to the web server line by line. This was especially the case if the internet connection was disabled for several hours, where a very large amount of data is handled.

The first solution (explained in 5.4.3.5 Push and post of the data to the web server) was to stop trying to send the data if an error occurs in the middle of the process: if the first line can not be sent, the program will still run line by line but in a simplified version without posting requests, only to save the rest of the data.

The second solution is to use an incremental waiting time for this function with the use of a global variable acting as an indicator: if the state of the indicator is False, the function Unsent_data does not execute. In the Settings, 3 different time levels can be declared (ie 1, 3 and 5 minutes) to specify how much time the function should wait before sending again (1 minute after the 1st unsuccessful try, 3 after the 2nd and 5 for all the remaining ones). If the internet is reconnected, which can be detected by a success in sending the
data by the function Send_data or Send_unsend_data, the indicator changes to True and the function will run normally.

5.5 Internal implementation and tests

5.5.1 Setting of the hardware

After reception (January 2018) of the Raspberry Pi kit ordered (December 2017), it was configured. This was made simple as the Raspberry Pi 3 Model B+ has 4 USB plugs allowing to connect a mouse, a keyboard and a screen simultaneously. The kit also included a SD card which was pre-installed with the operating system Linux Raspbian. Because the computer accessories, will not always be available (on site), because the USB ports could be used for other purposes (sensors, USB to RJ45 Ethernet for the inverter connection) or because the Raspberry Pi could be located remotely, ways to access and control it from distance were installed.

- SSH, standing for Secure Shell, allows to access the command line of a remote Raspberry Pi on the same network using the local IP address and the password. The functionality needs to be enabled in the Raspberry Pi settings, so this can not be used for the first installation.
- VNC, standing for Virtual Network Computing, is a graphical desktop sharing system allowing the control of one computer (Server, for instance the Pi) from another computer (Viewer). Both, direct on the same network (using the IP address and password) and cloud (over internet, an account is required) are possible with RealVNC, a VNC provider already installed with Raspbian OS.

SSH was enabled but VNC was preferred for future use because it is more intuitive to use: controlling the Pi with only command lines requires a specific syntax and knowledge about commands. On the contrary, using VNC makes it similar to using your own computer, moreover it has the ability to easily transfer files between the computer and the Pi.

Figure 27 - Remote access with SSH or VNC, on a local network (left) and online (right) (RealVNC)

5.5.2 Setting of a local web server

The Emoncms web server, fully available to download (it is a set of files coded in PHP, HTML and CSS) as it is open source, was installed locally in order to test the server part plus the interfacing with the hardware. In Chapter 5.3, the requirement to install Emoncms were explained: to run the website Apache, PHP and MySQL must be installed.

MAMP meaning Macintosh Apache MySQL PHP, is a software facilitating the installation and centralising with only one interface these technologies. Versions for Windows (WAMP) and Linus (LAMP) also exists and all of this can also be installed individually but it is more complex. Tutorials are easily available online explaining how to set up MAMP. For Emoncms, a guide through the OpenEnergyMonitor web page is also available describing the set up.

Additional modules (graph and dashboard) available for download were installed by placing the modules’ folder in the .../Emoncms/Modules directory. The graph module provides advanced graphing functionalities (cumulative graphs for energy, export to csv, histograms, etc.) and the dashboard module allows to create, view and publish dashboards.

5.5.3 Setting of the code

Most of the code was written from zero, and tested on the computer with the local Emoncms. This was particularly useful for the troubleshooting of data types errors when posting to Emoncms. Developing from the computer also allow to simultaneously browse the internet for resources or help about specific syntax of python.
Transferring the programs to Raspberry Pi is done using the VNC server – VNC viewer connection and only a few settings need to be changed to account for the change of environment, Mac OS to Linux (directories).

5.5.4 Internal tests

Internal tests are mandatory on the complete prototype, hardware and software, to assess how it reacts, to see if the data is correctly collected and passed to the web server and to correct potential errors. The term internal refer to the fact that the web server is hosted on the local network and to the impossibility to conduct the tests on-site.

First, virtual data was manually created to test the whole chain (except reading the data from the inverter) providing the ability to debug errors in the code, such as format errors that occurred when sending the requests to the web server.

However, in order test the hardware connection, a device using Modbus TCP and with SunSpec implemented, is required. When creating the timeline and objectives for the development of the prototype, 8 different projects were approved by the RBF grant with estimated date of completion scheduled between March and May and a delivery of the components as soon as February. The idea was to test these projects for on-site tests, as 5 of them are in the same city as our office. Large delays and cancellation of the several projects lead to the impossibility to test the prototype with an inverter. On-site testing with the existing projects was neither possible as both projects are in remote locations and do not have inverter with Modbus TCP and SunSpec.

Finally, data from one inverter of the inverter used in the existing projects was stored on a USB drive during a maintenance visit, to be tested by the prototype. The USB was connected to the inverter and by navigating in the menus displayed on the screen, the energy log can be exported. Only the energy and the event logs can be retrieved using this method although the Modbus register maps for this inverter shows that values such as voltages, currents or reactive power could be accessed connecting with Modbus RTU over RS-485. The prototype could not be modified in time to connect using this version of Modbus and it would also present few interest as the inverter is not SunSpec compatible.

A specific script was written to automate the reading of the data files (one for each day between 01/01/2018 and the day the data was placed in the USB drive, 21/03/2018). The data was saved as “nrg” files, standing for Nero Burning ROM, but it could be opened by a simple text software and contained the data in comma separated value format. The python program Inverter_data.py, gathers the data from every file and save it in one unique csv file. The function Main program was slightly modified to read the data from this file and, without averaging the data, as it is already one point for every 5 minutes, send it to the local web server.

The log contains four values, two concerning the energy production: peak power and the sum of the energy harvested. However, some parts of the logs are corrupted as it can be seen in Figure 28:
The test protocols with both virtual data and inverter data are described under:

### Test with manually created values (virtual data):

<table>
<thead>
<tr>
<th>Sample rate: 5s</th>
<th>Execution for 40 min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time: 2 min</td>
<td>One internet interruption for 20 min</td>
</tr>
<tr>
<td>4 threads of data</td>
<td>Execution for 1 hour</td>
</tr>
<tr>
<td>Live data sending</td>
<td>Interruption for 1 hour</td>
</tr>
<tr>
<td>Duration: 4 hours</td>
<td>Execution for the remaining time (1 hour)</td>
</tr>
<tr>
<td>Emoncms feed sampling time: 2 min (how often the data is saved by Emoncms)</td>
<td></td>
</tr>
</tbody>
</table>

Some data points appeared to be not correctly send during the execution of this first test. The main identified cause of this is that the frequency at which one value is send (which is set as the average time) is actually higher than the saving frequency of the feed in Emoncms, which control how often the input is saved in the database. If the sampling time is set to five seconds, the duration of one loop is actually a little more than five seconds as it is the five second of timer wait plus the time it took to execute all the actions in the loop (approximately 0,1 second). The solution is to measure the execution time of one loop and subtract it to the sampling time, the result is the time which needs to be used by the timer. This solution was used and the test was made a second time.

Internet interruptions are provoked manually by disconnecting the server to test the resilience to disturbances. The results show that one data point is missing each time that the server is re-connected.

### Test with the inverter data:

<table>
<thead>
<tr>
<th>Sample rate: 5s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time: disabled (the data contain one point for each 5 minutes)</td>
</tr>
<tr>
<td>4 threads of data: Power, Energy, Operating Time, and Timestamp</td>
</tr>
<tr>
<td>Historic data using the bulk method</td>
</tr>
<tr>
<td>Duration: from 01/01/2018 to 21/03/2018</td>
</tr>
<tr>
<td>Emoncms feed sampling time: 5 s (how often the data is saved by Emoncms)</td>
</tr>
</tbody>
</table>

With the inverter data, no interruption is scheduled because this test aims at verifying the resistance of the prototype with corrupted data.
Results of the test are shown in Table 13:

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>Passed data</th>
<th>Missing data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual data</td>
<td>No interruption&lt;br&gt;Sporadic values missing.&lt;br&gt;One data point is missing at each reconnection of the web server.</td>
<td>120</td>
<td>4 (3,2%)</td>
</tr>
<tr>
<td>Virtual data (improved code)</td>
<td>No interruption&lt;br&gt;One data point is missing at each reconnection of the web server. The rest of the data is correctly sent.</td>
<td>122</td>
<td>2 (1,6%)</td>
</tr>
<tr>
<td>Inverter (22 223 points)</td>
<td>No interruption&lt;br&gt;The data is correctly sent but several data points are missing. Mainly due to corrupted data (no the correct input format)</td>
<td>22 100</td>
<td>123 (0,5%)</td>
</tr>
</tbody>
</table>

Table 12 - Results from three tests

6 Discussion

6.1 Prototype summary

6.1.1 Overview and total cost

The prototype was designed using several technologies and is divided in two parts (hardware and software), the complete solution is presented by Figure 29:
The list of components and their associated costs for the prototype is given in Table 13:

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost (£)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi Model 3 B+ kit including:</td>
<td>52.50</td>
</tr>
<tr>
<td>- 1 Raspberry Pi Model 3 B+</td>
<td></td>
</tr>
<tr>
<td>- 1 SD card 16 GB</td>
<td></td>
</tr>
<tr>
<td>- 1 protection case</td>
<td></td>
</tr>
<tr>
<td>- 1 power supply plug</td>
<td></td>
</tr>
<tr>
<td>1 USB to Ethernet port</td>
<td>19.99</td>
</tr>
<tr>
<td>Total</td>
<td>72.49</td>
</tr>
</tbody>
</table>

Table 13 - List of components and costs for the prototype

The cost to put the web server online was excluded, it would be necessary to rent or buy a host and a domain name. A domain name which is not among famous or desired names can cost about 10€ per year while a shared host sized for a web site with low traffic is in the range of [50 – 70] € per year. (WP Site Care, 2018) (SiteGround)

6.1.2 Implementation

The prototype design was completed but its implementation was not finalised. For the hardware part, it was impossible to test the feasibility of the inverter connection with Modbus TCP and with the pySunSpec library. The test protocol was neither developed because it was impossible to determine what would be the test conditions without visibility on an access to an inverter. However, as the pySunSpec library provides a comprehensive documentation on how to use it and because it was created specifically to connect with SunSpec devices, it should work normally. The main uncertainty is the ability to set correctly a local network with the prototype and the other Modbus devices, dealing with the concepts of IP address, port number and master/slaves behaviours. The rest of the implementation of the hardware is finalised, with the prototype able to automatically store the data and send it through HTTP post requests.

On the server side, the implementation was partially finalised: it is functional on a local network but was not tested online. The reasons for this are that the hardware tests are not finished, because putting the webserver online is not free and because it would require to analyse more in detail potential security issues.

A few tests of the prototype were made with manually created data and with data collected from an inverter on-site: both the program, on the hardware, and the webserver are able to operate for several days without crashing.

6.1.3 Capabilities

The prototype is able to correctly store the data locally on the Raspberry Pi as a csv and it can be accessed manually with Excel. It is also able to send this data to the web server through HTTP post requests. The Emoncms website captures the data which is inputted by any device on the ‘Inputs’ page. In order to save the data in the web server database ‘feeds’ can be created.

Feeds are processes that store the input data in the database, several types can exist: to store the data on a fixed interval or for status values for instance, with irregular intervals of input. Simple and linear operations such as additions of two inputs, multiplying by a fixed factor can be applied and allow to process data before saving it. Post processing is also possible using virtual feeds which have the same abilities. Operations are limited to the moment of time when the feed is saved (ie you can not subtract a value by the value the same feed previously had), except one option which can add values over time to convert power to energy.
The graph module (see Figure 31 for an example) is intuitive to use and various functionalities are accessible. It allows to display any data between two dates of your choice, with two scales that can be used on the same graph and the data can be outputted as csv if desired. It is also possible to save the graph and adjust various options (colours, number of points, etc.). Dashboard allows to quickly view data after processing with a customised page. The tool for the creation of a dashboard is simple to understand and uses pre-existing symbols. Simple dashboard to very complex ones (Figure 32) can be created to present an overview of the system monitored and share them to external user (for a display on a screen at a reception for instance).

Figure 31 - Screenshot from the graph module, here with 4 of the manually created inputs (9 am to 6 pm)

Figure 32 - Example of a dashboard on Emoncms (S J Smith - OpenEnergyMonitor)
6.2 Review of the prototype

The prototype was implemented with all the functionalities described in section 2 Objectives, and is working without errors. The hardware has all the basic functionalities working except the interfacing with the inverters which could not be tested. This is an important issue since it is one of the main functionality and represented the main value addition of this prototype. This issue was also a barrier for the implementation of more advanced functionalities. It was impossible to prove that a more detailed monitoring can be achieved using inverters with several MPPTs to view power generation per zones of the installation. The interest of such a method is still to be tested and discussed. With the choice to use of Emoncms web server which provides powerful analysis tools and flexibility in the display of the data, the objectives were exceeded for the server part: the website has much more functionalities and reliability than if it was developed on my own, it also has a good graphical interface and documentation to facilitate the set up.

Tests, following the protocols described in 5.5.4, with the virtual data and with the inverter data show that sporadic values are not stored in the feeds resulting in a gap with a missing value. With the manually created values, the first version of the code had around 3% of the value disappearing. The correction in the timer reduced this phenomenon to only 1.6% of the values not transmitted, with one value being lost every time the server re-connects. The data which can not be send immediately when the internet connection is off, is correctly send once it re-connects. The test with the inverter data shows much more data points incorrectly unsent, 123 in total, this is coherent as many data points were completely unreadable even by humans and as expected sections which were completely corrupted were not transmitted but the program did not interrupt. Furthermore, the percentage of missing data points (0.5%) is lower as much more points are sent (one every five minutes for almost three months). The results essentially show that the code has some reliability and is able to resist disturbances such as internet disconnections or data corruption.

I was not able to solve the error at the internet reconnection resulting in the loss of one data point. The program in the Raspberry Pi indicates that this point is correctly sent and the server responds that it has accepted the input however it still does not appear in the graph. This means that the data is not saved in the feed correctly, meaning the error is probably due to the way inputs are saved as feed. As I have not programmed the website myself, finding and solving this particular error is difficult.

The lifetime of a monitoring device should be at least 20 years, however, for a prototype, one to three years is acceptable. The issue would certainly come from the Raspberry Pi which is the most sensitive part, being deployed on site. It was bought with a protection case to prevent damages and undesired infiltrations (bugs). There are many examples of Raspberry Pi used 24h/7days without troubles nevertheless the main issue is the instability of the power supply: a user reported 1% yearly losses on installations with ‘bad’ power supply. (Stack Exchange Forum, 2015). A potential solution is to install an uninterruptible power supply (UPS) to be able to shutdown normally the prototype if the power is lost. This would not be an issue for the monitoring of the system as commercial grid tie inverters will only work if they can synchronise to a power source.

For the security aspects of the hardware, the connection with the web server is authenticated by the API key, which ensure only devices sending the data with the correct secret API key will be accepted. On the server side, the security concerns will not be discussed as it was not published online, yet this would be an absolute necessity before doing it.

The python program was developed with the aim to be robust, indeed as shown previously, it is able to handle loss of internet connection or corrupted data as historic input. Other robust features are the initial checks of the settings or the use of python modules and functions (csv, for instance). However, my knowledge in python remain basic and several things should be improved to increase the robustness with:

- The handling of wrong data type inputs. If a string is used as data (a text or a date instead of a value for one data point) the data will not be accepted by the web server but there will be no major errors in the program as long as no average is made. To prevent this, there should be a verification of the type of the input before parsing it.
- Corrupted data blocking the ‘Temporary_unsent.csv’ file. If some data can not be send, it will be transferred to this temporary storage file, and the way the program works is that every time it will try to send this data. There is no intelligence that will supress this line because it can
never be accepted by the web server. The solution is similar to the previous problem: prevent
the errors by analysing the data before using it.

The prototype is modular and leave room for the addition of new functionalities: the Raspberry Pi has the
ability to interface with many devices and and 3 usb ports are unused with the current solution. The python
code can easily be improved or modified as this is an open source language. Blocks of work could even be
further modified: for instance, there is no specific need for a Raspberry Pi, only for a hardware able to
execute python scripts, connect to the internet through wifi or GSM and have an Ethernet port.
Additionally, every communication protocol used is open source ensuring no dependency on external
services.

And finally, the lack of test for inverter interfacing implies that some of the initial objectives can not be fully
measured:
- The complete reliability of the product. It can only be estimated: on all the tested
  functionalities the product has shown a reliable performance and no major error has been left
  unresolved. However, the inverter connection was not tested, and complication could occur to
  set a functional local network with IP addresses, port numbers and Modbus ID.
- The ease of integration and installation time. The design was thought to make it simple and
  flexible with the possibility to access the Raspberry Pi remotely, the declaration of the specific
  parameters for this project in the file Settings and finally the use of SunSpec standardised
  Modbus protocol allowing to connect to any inverter using this standard protocol. Yet, for the
  same reasons as previously listed complications could occur for the inverter connection on the
  local network.

6.3 Product comparison

The prototype was compared with available products on the market: from the inverter manufacturer, from
3rd part monitoring companies (two solutions) and with the EmonPi from the OpenEnergyMonitor project.
The comparison will be on presented functionalities, connectivity and prices, assuming that the reliability of
each solution is similar as they were not tested. A deep analysis of the software functionalities was neither
conducted as this would need to test each of them as well.

Solution A is from the inverter manufacturer and is the sole monitoring of the inverter through internet. It
was explained more in detail in section 1.1.6 a).

Device B is a complete acquisition and control device for solar installations. The hardware comes with a
protection enclosure (IP66, dust and water proof): it can monitor up to 12 devices among inverters,
performance instruments and meters, has 30 MB of local data storage for communication outage and a large
connectivity (2 RS-485, 1 Ethernet switch and instruments power ports). The software comes with the
following functionalities: custom charting engine, reporting engine, mobile friendly options, weather
overlays, dashboards, inverter heat maps (identification of risks) and remote on/off control. It is made to
monitor commercial to utility scale projects and it was the proposed device by the sale team for project of
a size of 100 kWp.

Device C, seems more intuitive to use because it is provided with a touch display screen plus automatic
installation and setup. It has similar functionalities with B but limited to a smaller size (100 kWp) which
means less connectivity and a software offering similar functionalities as of device B.

The EmonPi, (which was already presented in section 5.3.3) is only designed for residential projects but it
was included in the comparison as it provides a similar simplicity as the prototype. It can also connect with
sensors, has enough internal storage to cope with internet outages and a protection enclosure (metallic).
Limitations are that it does not communicate with inverters but only measures currents and voltages. The
sensors have ratings that are not fitted for project over 20kWp. The web server is the online version of
Emoncms (Emoncms.org), which was largely presented before. Buying an EmonPi gives 20% of its price
as credit for the use of the server.

The prototype functionalities are described in details previously but it can be summarised as a device
focusing, at the moment, only on inverter monitoring through the use of communication protocols. It can
theoretically access to data from a large number of inverters simultaneously, even if they are from different
brands and send this data over internet. The software used to visualise the data is the same as with the EmonPi, Emoncms.

**Review**

It is difficult to compare a prototype with professional devices, that are finalised products and which have much more reliability and functionalities. The inverter manufacturer solution (A) is limited as it will only work with inverters from the same brand and it is not possible to gather different installations under the same software if different brands are used. Solution C seems the most flexible as it can monitor a large number of inverters and can be used with various sensors. It is also less expensive than device B, which although it seems a little more professional seems to be more fitted for large projects.

The prototype developed is similar to the EmonPi, as it has the same software plus the same aspect of a simple and less professional data logger, but which would be thought only to connect to inverters as it lacks the capabilities at the moment to interface with sensors. However, it has the possibility to monitor installations regardless of their size as the only limitation concerns the number of devices connected in a network. Compared to the products B & C, it is definitely economical.
Table 14: Comparative table of the 3rd party monitoring solutions

<table>
<thead>
<tr>
<th>Device</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>EmonPi</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project range</td>
<td>Commercial to utility</td>
<td>Mid-commercial to utility</td>
<td>Medium size Max 100kWp</td>
<td>Residential to small commercial</td>
<td>Residential to medium commercial</td>
</tr>
<tr>
<td>Description</td>
<td>Monitoring of the inverter only but additional equipment can be purchased to monitor / communicate with other components</td>
<td>Device B offers professional monitoring solutions, able to connect to a large number of devices and promote a strong reliability of the system</td>
<td>Device C offers a professional solution with a focus on the simplicity of installation and use</td>
<td>Easy to implement and small hardware, functions can not really be extended</td>
<td>Developed to monitor only the inverters of the system.</td>
</tr>
<tr>
<td>Ability to connect to inverters</td>
<td>Yes, only from the same brand</td>
<td>Yes, 2 central inverters and approx. 100 string inverters</td>
<td>Up to 100 inverters</td>
<td>No</td>
<td>Yes, but they need to be SunSpec compliant</td>
</tr>
<tr>
<td>Ability to connect to monitoring devices</td>
<td>Additional equipment required</td>
<td>Weather station, AC meters, temperature, pyrometers and wind speed sensors via RS485 connection. A and B have more extendibility and in-built connectivity</td>
<td>Limited, temperature, voltage, current and pulse (utility meters) sensors</td>
<td>Not implemented</td>
<td></td>
</tr>
<tr>
<td>Further functionalities</td>
<td>n.a</td>
<td>Remote on/off control of inverters</td>
<td>Feed in power management reductions (charged)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software functionalities</td>
<td>Chart, alarms and export to csv but more limited</td>
<td>Flexible options such as charts, alarms, export to csv, dashboards, mobile portability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software differentiation</td>
<td>n.a</td>
<td>Performance analysis and reports. Designed medium to large scale solar plants</td>
<td>Designed for the monitoring of energy values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price of the hardware (sensor excluded)</td>
<td>n.a</td>
<td>4 380 £  Plus 600 £ for installation and engineering fees</td>
<td>696 £</td>
<td>164 £ (185 €)</td>
<td>72 £ (excluding the time spent to develop the prototype)</td>
</tr>
<tr>
<td>Price of the software (per year)</td>
<td>0 £</td>
<td>170 £ (192 £)</td>
<td>220 £</td>
<td>1 £ per value fed 32 £ of free usage for buying the hardware</td>
<td>Approx. 70€</td>
</tr>
</tbody>
</table>
6.4 Review of the implementation

The schedule established with the GANTT was to one exception, reasonably followed. An overall delay of one to two weeks appeared during the design phase due to a longer time than expected to understand the various technology involved which were for most completely new to me. The implementation phase would also have taken a little more time than expected, because of difficulties rising from my low knowledge of use with a Raspberry Pi and its control via a command window, but thanks to the discovery of Emoncms the server part was in advance, balancing with the implementation of the hardware.

As mentioned earlier, on site tests could not be achieved due to time constraints but mostly because of a lack of resources. There were no installations ready or simply inverters in stock that I could use to test this prototype. At the time when the schedule for this project was established, eight projects were in the pipeline with completion dates expected between February and May. This would have left me one to two months to test the prototype, as five of these projects were in Yangon and its area, the main city where our office was also located. As all of them were accepted at the second and final phase of application to receive a financial grant by the RBF (between 40 and 65% of the total cost is paid), I expected all of them to be executed. Finally, only two were signed and the construction dates were delayed by several months. Another project, without a grant, signed in February, did not allow as well to test the prototype as the construction was scheduled to start in May. As I left the country, at the end of March, no projects had been started, and the two existing mini-grids were using inverters without Sunspec with Modbus TCP/IP implemented.

Barrier to the installation of PV systems

Although many conditions are favourable to the installation of PV systems for self-consumption, there are only a few installations of PV systems in the commercial and industrial sectors in Myanmar. In May 2018, only three projects had been developed by competitors. It was difficult to sell projects even with the grant opportunities that lead to estimated returns over investment as low as 4-5 years. This slow market for solar systems used in self consumption can be attributed to several potential barriers.

The first one being the novelty of this technology, Myanmar has been for a very long time a closed market for intentional companies and the introduction of PV is very recent. The barriers to enter the market from external companies are also high in a country which has only been opening to foreign investments at the same time as engaging the path toward democracy since 2012. Many directors of businesses were eager to see projects already finished in the country but Mandalay Yoma is lacking reference in this field. Apart from the two mini-grid villages installed in the summer 2017 it has not developed any other projects and it seems that the strategic partnership with Tata solar, which has built the largest solar rooftop in the world, was not sufficient to inspire confidence (Tata Power Solar).

Another barrier in Myanmar is the lack of financing solutions, this is particularly problematic as solar PV projects require high initial investments. Although 65% percent of the total cost, up to approximately 60 000$ could be paid as a grant, access to financing solution is a struggle for small and medium enterprises, with asking customer and relatives being the most common solution (Myanmar Times, 2016). The Myanmar economy is growing fast, with a 6,8% growth of the GDP in 2017 (Asian Development Bank, 2018). As a consequence, I also discovered that many business owners preferred to invest their money to expend their assets rather than investing in a system reducing most of the time only a fraction of their electricity expenditures. The price of this electricity is slightly high for commercial and industrial users, around 0,10 to 0,12 $/kWh, not including the need of fuel for a back-up system. The government subsides heavily the residential electricity (sold at around 0,03 $/kWh) resulting in losses close to $500 millions over the last fiscal year (Myanmar Times, 2018). The government has to increase the electricity prices to achieve balance and the last one was in 2014. A similar increase (+30%) was expected to happen in late 2017 but it was finally postponed to an unknown date. The last barrier is legal: it is forbidden in Myanmar to inject active power in the grid. Every system was then dimensioned only for self-consumption.

Of the eight projects, one was cancelled due to international sanctions after tensions in the country with the Rohingyas crisis. The rest were mostly cancelled as the owner did not want to invest in such systems, another argument was that the investments were to happen at the end of the fiscal year (one year runs from April to March). Finally, most clients did not see the final application for the grant as a commitment which can arise from a lack of communication between the various parties (RBF, Mandalay Yoma and the clients).
I believe that a more precise approach with project management tools would have help solving some of the issues listed. Indeed, a risk mitigation plan from the beginning with fall back options would have been necessary, another way should have been to include more control such as mid-term evaluation and correction for the project. The lack of on-site tests had some beneficiary consequences, it has lead me to focus more on the internal tests and on some key characteristics: the reliability and security of the system.

On the other hand, it also blocked progress on other parts: the online tests (publishing our own version of Emoncms or using the worldwide available online version), the analysis of data from sites and a study of the possibilities to have a ‘zone monitoring’ using the various MPPTs were not conducted.

### 6.5 Value addition and future improvements

The monitoring product is not finished but it shows that it is relatively simple to use various existing blocks and open technologies to monitor on your own an inverter. This is made possible because solar energy is becoming a mainstream technology with more and more standardisation available. It can be re-used by any actor wishing to monitor its PV plant, such as developers but also clients that could wish more control on their sites. The prototype is not site or country specific as the Modbus standard are becoming increasingly implemented in the solar industry. The prototype, although not tested on site, meets most of the objectives, is relatively simple and economical to install in any commercial system.

I believe this product still needs numerous improvements before being used, for this I see two axis of improvements. First, with the long term goal to be a professional solution, which means to have enough confidence to sell it to clients or use it internally without errors. For the hardware, it would first require a phase of on-site testing followed by error corrections. On the server part, there is a need to go online whether with Emoncms.org and paying a cost per feed used, which would be the best way to test the connection at the beginning or by publishing our own version of Emoncms on internet, but I would recommend this to happen later, once the product is finished. Further improvements would also be necessary such as the addition of a night mode to use the device only when it is necessary, or the capability to connect with pre-defined sensors. The hardware could also be changed to reduce the cost by moving to custom made industrial board. Finally, to make it user friendly, a documentation and interface would need to be created.

The other area of improvement is to develop additional uses in the automation field. This would mean to focus on the interesting possibilities offered by SunSpec Modbus. Beside the fact that the data logger could be automated to recognise the inverter, the models of data which are implemented and automatically collect all or some of the data, it could also change parameters in the inverter using Modbus, allowing to control it in a smart way. This concept is called the smart-inverter which is a part of the smart-grids and interacts with the other components of the smart-grids. It could provide services such as voltage control, active and reactive power control. It is also very interesting in the perspective of hybridisation with batteries or genset for instance with a curtailment of the power harvested if the spinning reserve of the generator is too low. This prototype although developed with the idea in mind to monitor inverters, could also add any other SunSpec compliant device and control it in a ‘smart’ way. Focusing in this area of development would lead to the development of an energy management system (EMS) but would require a large amount of work.
7 Conclusion

As Sunspec Modbus is increasingly implemented among solar devices, this project has investigated its usage to monitor inverters in a PV system. This standardised protocol is of particular interest because it allows to connect to any inverter regardless of the brand. Secondly, because the inverter is the central piece of PV installation, monitoring it via the internet of objects can provide a simple and cost efficient solution.

Through the report, several technologies have been analysed and compared in order to choose the best available option for this project and according to the given objectives. The results are a cost efficient prototype, with a complete monitoring solution. It uses a Raspberry Pi for prototyping and network capabilities, which is coded in python, the best language for this as it comes with a complete Modbus Sunspec library. The Raspberry Pi and the inverter communicates using an Ethernet connection over Modbus TCP/IP. These choices allow an ease of integration with open protocols, a simplicity of understanding and design with widely used technologies backed by strong communities (forum, tutorials). The code is developed from scratch with a focus on facilitating a future installation and to provide a reliability of the execution even in case of internet loss or if wrong data are passed. Emoncms, an open source website for energy monitoring is used as the monitoring software because of the gain of time and the multiples functionalities it provides. The communication between the hardware and the software is made using HTTP requests which are secured by an identification (API) key provided by the website allowing a strong reliability and security for the data.

Tests performed with the prototype demonstrate good operations and a relatively good reliability. Only a handful data points are lost during operations with the prototype able to handle wrong data input or internet interruptions without stopping. However, the product could not be tested on site as per the schedule because of delays in the company’s projects. The Myanmar market for self-consumption C&I projects offers a lot of opportunities but there are still important barriers preventing the deployment of numerous installations, mainly the lack of support from the government and the quick growth of Myanmar which favours other investments.

This projects brings the advancements, the standardisation and the shared knowledge available from the internet of objects, to show that a new form of monitoring is both possible and simple to implement, using only standardised and open-source communication protocols. It allows developers and owners to uses several different brands of inverter on one or more projects and monitor everything in central and unique website. Finally, it provides an affordable alternative to existing monitoring solutions that can be used on any Sunspec compliant installation over the world. The prototype still requires effort to turn it into a viable and fully reliable monitoring device, however a more interesting area of work is to improve its communication capabilities by making it able to control and operate components in a smart grid.
8 References


9 Appendices

List of inverters for the comparison

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB</td>
<td>ABB TRIO TM 50-400/60-480</td>
</tr>
<tr>
<td>Fronius</td>
<td>FRONIUS ECO 25.0-3-S / FRONIUS ECO 27.0-3-S</td>
</tr>
<tr>
<td>Huawei</td>
<td>SUN2000 8–28KTL</td>
</tr>
<tr>
<td>Kaco</td>
<td>Blueplanet 50.0 TL3 SM</td>
</tr>
<tr>
<td>Scheiner</td>
<td>CL-60E</td>
</tr>
<tr>
<td>SMA</td>
<td>Sunny Tripower CORE1</td>
</tr>
<tr>
<td>SolarEdge</td>
<td>SE66.6K-SE100K</td>
</tr>
</tbody>
</table>

List of data formats SunSpec

- Int: signed integer value
- Uint: unsigned integer value
- Pad: reserve field, used to round a Model to an even number of registers
- Acc: accumulated values, used for ever increasing values that may roll over
- Enum: enumerated value, used for status or state
- Bitfield: a collection of bits, multi-valued alarms or state
- String: a null terminated or fixed length value
- Ip: internet protocol formatted network address
- Sunssf: scale factor value

List of the standard models in the Inverter series

<table>
<thead>
<tr>
<th></th>
<th>Inverter (Single Phase)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Include this model for single phase inverter monitoring</td>
</tr>
<tr>
<td></td>
<td>Inverter (Split-Phase)</td>
</tr>
<tr>
<td>102</td>
<td>Include this model for split phase inverter monitoring</td>
</tr>
<tr>
<td></td>
<td>Inverter (Three Phase)</td>
</tr>
<tr>
<td>103</td>
<td>Include this model for three phase inverter monitoring</td>
</tr>
<tr>
<td></td>
<td>Inverter (Single Phase) FLOAT</td>
</tr>
<tr>
<td>111</td>
<td>Include this model for single phase inverter monitoring using float values</td>
</tr>
<tr>
<td></td>
<td>Inverter (Split Phase) FLOAT</td>
</tr>
<tr>
<td>112</td>
<td>Include this model for split phase inverter monitoring using float values</td>
</tr>
<tr>
<td></td>
<td>Inverter (Three Phase) FLOAT</td>
</tr>
<tr>
<td>113</td>
<td>Include this model for three phase inverter monitoring using float values</td>
</tr>
<tr>
<td></td>
<td>Nameplate</td>
</tr>
<tr>
<td>120</td>
<td>Inverter Controls Nameplate Ratings</td>
</tr>
<tr>
<td></td>
<td>Basic Settings</td>
</tr>
<tr>
<td>121</td>
<td>Inverter Controls Basic Settings</td>
</tr>
<tr>
<td></td>
<td>Measurements Status</td>
</tr>
<tr>
<td>122</td>
<td>Inverter Controls Extended Measurements and Status</td>
</tr>
<tr>
<td></td>
<td>Immediate Controls</td>
</tr>
<tr>
<td>123</td>
<td>Immediate Inverter Controls</td>
</tr>
</tbody>
</table>
Various models

Various controls, functionalities and curves of the inverter

144 HFRTX

HFRT extended curve

145 Extended Settings

Inverter controls extended settings

160 Multiple MPPT Inverter Extension Model

### List of values in the Inverter (Three Phase) model (103)

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Name</th>
<th>Label</th>
<th>Value</th>
<th>Type</th>
<th>Units</th>
<th>SF</th>
<th>Mandatory M/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>ID</td>
<td>Inverter (Three Phase)</td>
<td>103</td>
<td>uint16</td>
<td></td>
<td>M</td>
<td></td>
<td>Include this model for three phase inverter monitoring</td>
</tr>
<tr>
<td>Header</td>
<td>L</td>
<td></td>
<td>50</td>
<td>uint16</td>
<td></td>
<td>M</td>
<td></td>
<td>Model Length</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>A</td>
<td>Amps</td>
<td></td>
<td>uint16</td>
<td>A</td>
<td>A_SF</td>
<td>M</td>
<td>AC Current</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>APHA</td>
<td>Amps PhaseA</td>
<td></td>
<td>uint16</td>
<td>A</td>
<td>A_SF</td>
<td>M</td>
<td>Phase A Current</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>APHB</td>
<td>Amps PhaseB</td>
<td></td>
<td>uint16</td>
<td>A</td>
<td>A_SF</td>
<td>M</td>
<td>Phase B Current</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>APHC</td>
<td>Amps PhaseC</td>
<td></td>
<td>uint16</td>
<td>A</td>
<td>A_SF</td>
<td>M</td>
<td>Phase C Current</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>A_SF</td>
<td></td>
<td></td>
<td>sunssf</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PPVPHAB</td>
<td>Phase Voltage AB</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>O</td>
<td>Phase Voltage AB</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PPVPHBC</td>
<td>Phase Voltage BC</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>O</td>
<td>Phase Voltage BC</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PPVPHCA</td>
<td>Phase Voltage CA</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>O</td>
<td>Phase Voltage CA</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PVPHA</td>
<td>Phase Voltage AN</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>M</td>
<td>Phase Voltage AN</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PVPHB</td>
<td>Phase Voltage BN</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>M</td>
<td>Phase Voltage BN</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>PVPHC</td>
<td>Phase Voltage CN</td>
<td></td>
<td>uint16</td>
<td>V</td>
<td>V_SF</td>
<td>M</td>
<td>Phase Voltage CN</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>V_SF</td>
<td></td>
<td></td>
<td>sunssf</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>W</td>
<td>Watts</td>
<td></td>
<td>int16</td>
<td>W</td>
<td>W_S F</td>
<td>M</td>
<td>AC Power</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>W_SF</td>
<td></td>
<td></td>
<td>sunssf</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>Hz</td>
<td>Hz</td>
<td></td>
<td>uint16</td>
<td>Hz</td>
<td>Hz_S F</td>
<td>M</td>
<td>Line Frequency</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>Hz_SF</td>
<td></td>
<td></td>
<td>sunssf</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>VA</td>
<td>VA</td>
<td></td>
<td>int16</td>
<td>VA</td>
<td>VA_S F</td>
<td>O</td>
<td>AC Apparent Power</td>
</tr>
<tr>
<td>Fixed Block</td>
<td>VA_SF</td>
<td></td>
<td></td>
<td>sunssf</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Block</td>
<td>Variable</td>
<td>Description</td>
<td>Type</td>
<td>Scale</td>
<td>Offset</td>
<td>Units</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------------</td>
<td>------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>VAr SF</td>
<td>VAr_</td>
<td>AC Reactive Power</td>
<td>int16</td>
<td>var</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PF SF</td>
<td>PF_</td>
<td>AC Power Factor</td>
<td>int16</td>
<td>Pct</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WH SF</td>
<td>WH_</td>
<td>AC Energy</td>
<td>acc32</td>
<td>Wh</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCA_SF</td>
<td>DCA_</td>
<td>DC Current</td>
<td>uint16</td>
<td>A</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCV_SF</td>
<td>DCV_</td>
<td>DC Voltage</td>
<td>uint16</td>
<td>V</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCW_SF</td>
<td>DCW_</td>
<td>DC Power</td>
<td>int16</td>
<td>W</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TmpCab SF</td>
<td>Tmp_</td>
<td>Cabinet Temperature</td>
<td>int16</td>
<td>C</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TmpSnk SF</td>
<td>Tmp_</td>
<td>Heat Sink Temperature</td>
<td>int16</td>
<td>C</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TmpTrns SF</td>
<td>Tmp_</td>
<td>Transformer Temperature</td>
<td>int16</td>
<td>C</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TmpOt SF</td>
<td>Tmp_</td>
<td>Other Temperature</td>
<td>int16</td>
<td>C</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>St SF</td>
<td>St_</td>
<td>Operating State</td>
<td>enum16</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StVnd SF</td>
<td>StVnd_</td>
<td>Vendor Operating State</td>
<td>enum16</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evt1 SF</td>
<td>Evt_</td>
<td>Event1</td>
<td>bitfield32</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evt2 SF</td>
<td>Evt_</td>
<td>Event Bitfield 2</td>
<td>bitfield32</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EvtVnd1 SF</td>
<td>EvtVnd_</td>
<td>Vendor Event Bitfield 1</td>
<td>bitfield32</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EvtVnd2 SF</td>
<td>EvtVnd_</td>
<td>Vendor Event Bitfield 2</td>
<td>bitfield32</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EvtVnd3 SF</td>
<td>EvtVnd_</td>
<td>Vendor Event Bitfield 3</td>
<td>bitfield32</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EvtVnd4 SF</td>
<td>EvtVnd_</td>
<td>Vendor Event Bitfield 4</td>
<td>bitfield32</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Enum Values:**

- **OFF**: Device is not operating
- **SLEEPING**: Device is sleeping / auto-shutdown
<table>
<thead>
<tr>
<th>enum16</th>
<th>STARTING</th>
<th>Starting</th>
<th>3</th>
<th>Device is starting up</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum16</td>
<td>MPPT</td>
<td>MPPT</td>
<td>4</td>
<td>Device is auto tracking maximum power point</td>
</tr>
<tr>
<td>enum16</td>
<td>THROTTLE</td>
<td>Throttled</td>
<td>5</td>
<td>Device is operating at reduced power output</td>
</tr>
<tr>
<td>enum16</td>
<td>SHUTTING_DOWN</td>
<td>Shutting down</td>
<td>6</td>
<td>Device is shutting down</td>
</tr>
<tr>
<td>enum16</td>
<td>FAULT</td>
<td>Fault</td>
<td>7</td>
<td>One or more faults exist</td>
</tr>
<tr>
<td>enum16</td>
<td>STANDBY</td>
<td>Standby</td>
<td>8</td>
<td>Device is in standby mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bitfield3</th>
<th>GROUND_FAULT</th>
<th>Ground fault</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitfield3</td>
<td>DC_OVER_VOLT</td>
<td>DC over voltage</td>
<td>1</td>
</tr>
<tr>
<td>bitfield3</td>
<td>AC_DISCONNECT</td>
<td>AC disconnect open</td>
<td>2</td>
</tr>
<tr>
<td>bitfield3</td>
<td>DC_DISCONNECT</td>
<td>DC disconnect open</td>
<td>3</td>
</tr>
<tr>
<td>bitfield3</td>
<td>GRID_DISCONNECT</td>
<td>Grid shutdown</td>
<td>4</td>
</tr>
<tr>
<td>bitfield3</td>
<td>CABINET_OPEN</td>
<td>Cabinet open</td>
<td>5</td>
</tr>
<tr>
<td>bitfield3</td>
<td>MANUAL_SHUTDOWN</td>
<td>Manual shutdown</td>
<td>6</td>
</tr>
<tr>
<td>bitfield3</td>
<td>OVER_TEMP</td>
<td>Over temperature</td>
<td>7</td>
</tr>
<tr>
<td>bitfield3</td>
<td>OVER_FREQUENCY</td>
<td>Frequency above limit</td>
<td>8</td>
</tr>
<tr>
<td>bitfield3</td>
<td>UNDER_FREQUENCY</td>
<td>Frequency under limit</td>
<td>9</td>
</tr>
<tr>
<td>bitfield3</td>
<td>AC_OVER_VOLT</td>
<td>AC Voltage above limit</td>
<td>10</td>
</tr>
<tr>
<td>bitfield3</td>
<td>AC_UNDER_VOLT</td>
<td>AC Voltage under limit</td>
<td>11</td>
</tr>
<tr>
<td>bitfield3</td>
<td>BLOWN_STRING_FUSE</td>
<td>Blown String fuse on input</td>
<td>12</td>
</tr>
<tr>
<td>bitfield3</td>
<td>UNDER_TEMP</td>
<td>Under temperature</td>
<td>13</td>
</tr>
<tr>
<td>bitfield3</td>
<td>MEMORYLOSS</td>
<td>Generic Memory or Communication error (internal)</td>
<td>14</td>
</tr>
<tr>
<td>bitfield3</td>
<td>HW_TEST_FAILURE</td>
<td>Hardware test failure</td>
<td>15</td>
</tr>
</tbody>
</table>