Vietnamese-German University

COPYRIGHT WARNING

FRANKFURT UNIVERSITY OF APPLIED SCIENCES
Faculty 2: Computer Science and Engineering

VIETNAMESE-GERMAN UNIVERSITY (VGU)
Electrical Engineering and Information Technology (EEIT)

An Off-grid Photovoltaic Generation System with Energy Information Communication Framework

by
Dinh Huu Phuc
Matriculation No. 1104950

Supervisor
Dr. Bui Minh Duong

Second Supervior
M.Sc. Tran Quang Nhu

A report submitted to the EEIT Study Programme
in partial fulfillment of the requirements
for the degree of Bachelor of Engineering (B.Eng.)

Ho Chi Minh City, Vietnam
26th June, 2018


VGU
Vietnamese - German University

Dinh Huu Phuc
1104950
606/54, 13th National Road, Thu Duc District
TPHCM, Vietnam
26th July, 2018

Electrical Engineering and Information Technology
Vietnamese German University
Nam Ki Khoi Nghia, Hoa Phu District
Binh Duong New City, Vietnam

Dear Dr. Bui Minh Duong

Enclosed is my Bachelor's report titled “An Off-grid Photovoltaic Generation System with Energy Information Communication Framework”.

I appreciate the time you are taking to review this report and hope that it meets your approval. If you have any questions, feel free to contact me either by telephone 01258 121314 or e-mail: huuphucdinh@gmail.com.

Sincerely yours,

Dinh Huu Phuc

Statement of Authorship

I certify that the attached report is my original work. No other person's work has been used without due acknowledgment. Except where I have clearly stated that I have used some of this material elsewhere, it has not been presented by me for examination in any other course or subject at this or any other institution.

I understand that the work submitted may be reproduced or communicated for the purpose of detecting plagiarism. I understand that should this declaration be false, I am liable to be penalized under the Vietnamese-German University regulations.

Dinh Huu Phuc _____

Date 26th July, 2018

Abstract

Photovoltaic (PV) technology nowadays becomes more popular due to its versatility, durability and flexibility. Specifically, the PV technology is mainly focused on efficiency improvement of solar panels. Therefore, many off-grid PV generation systems have been built for households at the community size. The aim of this project is to develop a smart energy management system for the community-sized off-grid PV generation system by building an energy information communication framework. The energy information communication framework (EICF) can dispatch the efficient amount of PV power to the customers by applying an Internet of Thing (IoT) technology. In this project, all the data which collected from solar panels are analyzed and transmitted by .XML documents extracted from LabVIEW program. The information will be update in real-time, process in JAVA language and exchange to an aggregator via online network using TCP/IP method.

Key Words: Photovoltaic, Renewable energy resources, solar power, TCP/IP communication, XML documents, LabVIEW, JAVA language

Acknowledgments

I would also like to express my deepest appreciation to Dr. Bui Minh Duong and Mr. Tran Quang Nhu for his tremendous support and guidance for this project.

Abbreviations

PV Photovoltaic

RES Renewable energy source

MPPT Maximum power point tracking

EMS Energy management system

VTN Virtual Terminal Node

VEN Virtual End Node

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1. Motivation

Global energy crisis and climate change threats are among major concerns confronted by the present civilization. Some of the major problems are greenhouse gases emission and the limited source of fossil fuels. Renewable energy sources (RES) such as solar, wind, and tidal are considered to overcome these concerns. Among these RES, solar energy is considered one of the potential sources to solve the crisis as it is available in abundance, pollution-free, and inexhaustible.

The PV technology is mainly focused on efficiency improvement of solar panels. Therefore, it attracts the attention of different customers (e.g. industrial customers, residential customers) to build their own off-grid PV generation systems. Furthermore, it can be possible to aggregate the individual off-grid PV generation system into a community-sized PV generation system to meet more load requirements for the customers.

The aim of this project is to develop a smart energy management system for the community-sized off-grid PV generation system by building an energy information communication framework. The energy information communication framework (EICF) can dispatch the efficient amount of PV power to the customers by applying an Internet of Thing (IoT) technology.

Economic efficiency is a major concern for any PV system; therefore; planning and forecasting are important for energy purchase. The PV measurement data are statistically analyzed and linked to get demand forecasts. The result of data analysis in this project can be expanded for a greater-scaled PV system.

1.2. Aim

The aim of this project is to develop an off-grid photovoltaic energy management system. The aim consists of 5 main parts:

1. Overview of PV generation system: this part gives detail knowledges about 4 main components of a solar system: PV panels, MPPT solar charge controllers, DC/AC inverters, an energy storage and AC loads.

2. Data Processing: the data from MPPT solar charge controller is processed by JAVA programming language.

3. Graphical User Interface: LabVIEW is used as a working environment to visualize aspects of the project such as graph, table, and hardware configuration.

4. Communication: the data under .xml format is transmitted between server and client using TCP/IP method in LabVIEW.

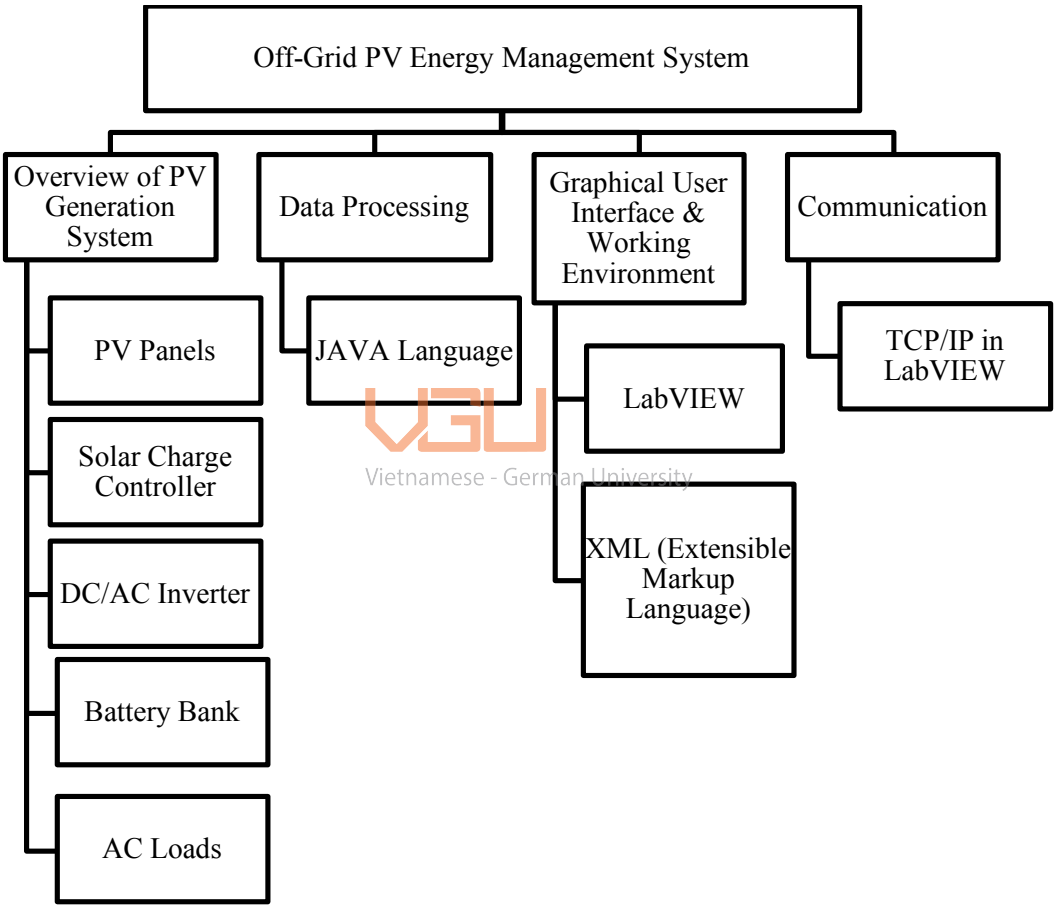


Figure 1.1 Main contents in the project

1.3. Objectives

The objectives of this project are:

- Extracting the data from MPPT solar charge controller such as voltage, current, power & temperature in .csv format.
- Using JAVA programming language to automatically calculate the collected values with a sample size of every 5 minutes.
- Using LabVIEW to create a GUI (Graphical User Interface) to observe real-time values, flatten and unflatten .xml data.
- Studying the server/client web application (TCP/IP) to communicate between Virtual Terminal Node (VTN) and Virtual End Node (VEN) in LabVIEW program.

1.4. Gantt Chart

The chart below implements the process of accomplishing this project. It starts from 1st February to 15th August. Each essential element such as build the small off-grid PV system, data analysis, and data collection is broken down into smaller ones.

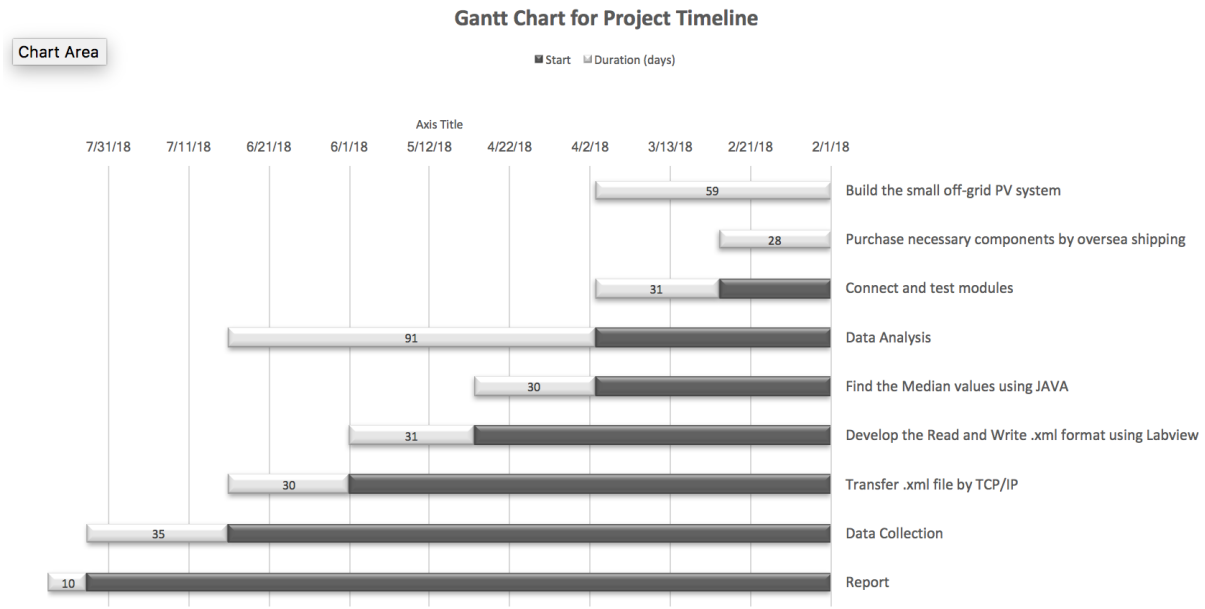


Figure 1.2 Gantt Chart for project timeline

1.5. Structure of the thesis

This report includes six chapters. Each chapter starts with a brief introduction, and follows by a details discussion.

- Chapter 1 is the introduction of the project. This chapter contains motivations, aim, objectives, project timeline (Gantt chart), and the Thesis outline.
- Chapter 2 mentions the theoretical background on PV generation system: the design of a small-scaled PV generation system, solar panels, solar charge controller, battery banks, DC/AC inverter and AC loads
- Chapter 3 introduces the design of a small-sized off-grid PV generation system: how to design and choose components for the system. This chapter also introduces a case study for data analysis purpose.
- Chapter 4 discusses the development of an energy information communication framework (EICF).
- Chapter 5 shows steps, methods and results of the project.
- Chapter 6 reviews all achievements, suggestions and limitations of the project.

2. Theoretical background on PV Generation System

2.1. Overview of PV Generation System

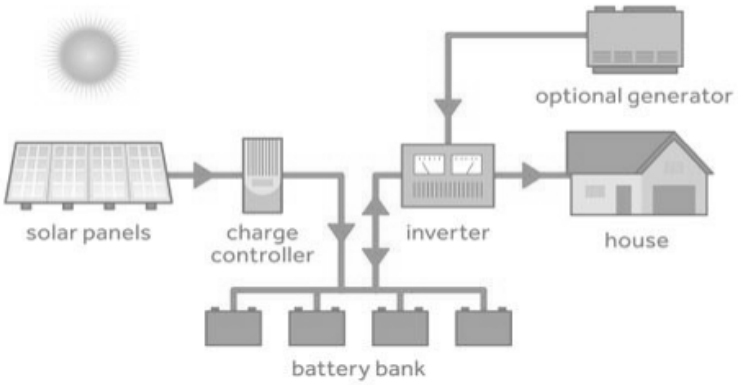


Figure 2.1 A PV generation system

The PV generation system includes 5 main parts, namely: solar panels, a solar charge controller (MPPT), a DC/AC inverter, a battery bank, and AC loads.

- Solar panels convert light energy directly to DC power. (refer to Section 2.2)
- Solar charge controller which uses Maximum Power Point Tracking method (MPPT) utilizes maximum power going from solar panels to battery bank. (refer to Section 2.3)
- Battery bank is used to store energy, for a few hours or a few days. (refer to Section 2.4)
- DC/AC inverter is used to convert DC input voltage into symmetric AC output voltage of desired frequency and magnitude. (refer to Section 2.5)

2.2. Solar Panels

2.2.1. Photovoltaics

The terms “photovoltaics” refers to the conversion of light energy directly into electricity. PV devices are manufactured into modules, each producing an identified voltage and a direct current.

PV modules are assembled as a group of solar cells. They are mounted in arrays and wired in series or parallels connection as shown in Figure 2.1. In a series connection, the current stays the same and the voltage adds up. In opposite way, the voltage remains a constant and the current is added up.

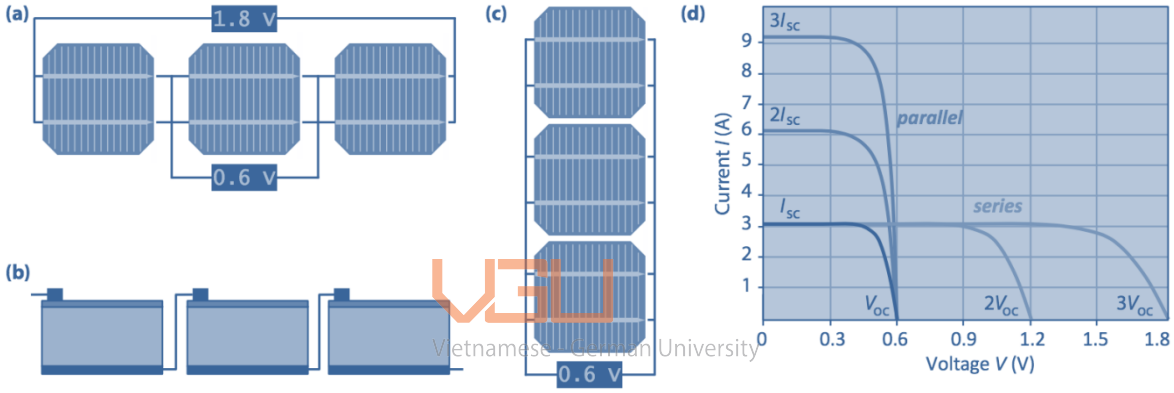


Figure 2.2 An example of series/parallel connection (a) A series connection of 3 solar cells (b) Series connection for cells with s classical front metal grid. (c) A parallel connection of 3 solar cells (d) I-V curves of solar cells connected in series and parallel [6]

2.2.2. Open-Circuit Voltage and Operating Voltage of a PV Cell

Voltage of a PV Cell depends on the band-gap energy of the material more than anything else. That means how much energy we have to put in to separate an electron (-) from a hole (+). Band-gap energy is the energy to level one electron by 1 Volt potential. Its unit is electron Volts (eV).

We are now considering some important characteristics by this formula:

$$V_{oc} = \frac{\varepsilon_{gap}}{q} + \left(\frac{2kT}{q}\right) \ln\left[\frac{\Delta n_{light}}{(N_C N_V)^{1/2}}\right] \qquad \text{(Equation 2.1)}$$

Where v_{oc} is the open circuit voltage, ε_{gap} is the band gap, Δn_{light} is the increase in the concentration of electrons induced by light ($1/cm^3$), N_C and N_V are the densities of conduction band states and valence band states, respectively ($1/cm^3$).

- As temperature increases, voltage drops linearly: The value of the 2nd term is always negative because the argument of the ln function is always less than 1. In practice, we do not use this equation to calculate voltage related to temperature, but using a curve of test results for each PV module.

- As light intensity increases, voltage rises logarithmically: As the intensity of the light increases, more electron hole pairs are created, and Δn_{light} increases, thus increasing voltage in the equation.

2.2.3. Effect of insolation and temperature on the i-v curve

As incident solar radiation, insolation, I_C (W/m^2) rises up, each extra photon creates an extra electron-hole pair and since electric current refers to the number of electrons and holes travelling around a circuit, the current increases linearly.

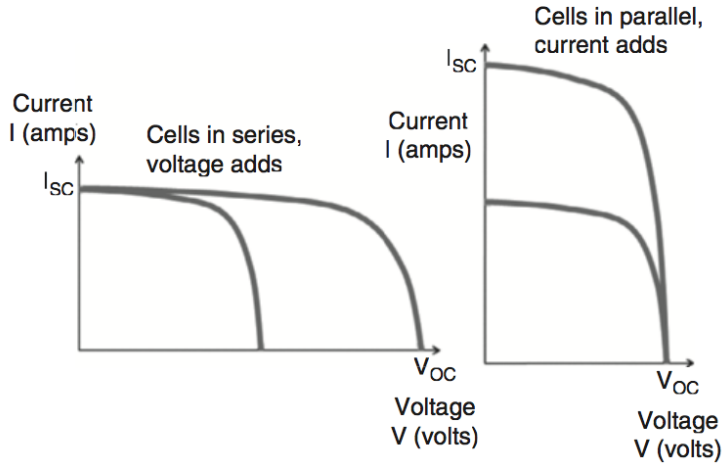


Figure 2.3 The voltage adds when PV cells connected in series, the current adds when PV cells connected in parallel [5]

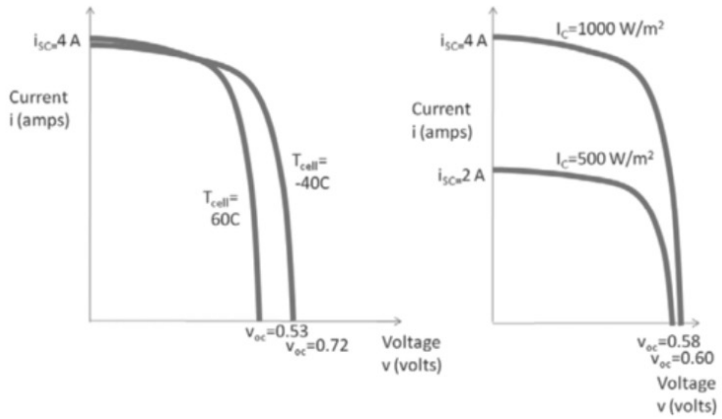


Figure 2.4 The current fluctuates linearly with isolation, the voltage varies logarithmically with temperature [5]

2.2.4. The relationship between volts, amps, ohms, watts, watt-hours and watt-peak

Voltage (Volts) is equal to current multiplied by resistance. (Ohm's Law):

$$\text{Volts} = \text{Current} * \text{Resistance} \quad (V = I * R) \quad \text{(Equation 2.2)}$$

Power (Watt) is equal to volt times current:

$$\text{Power} = \text{Volts} * \text{Current} \quad (P = V * I) \quad \text{(Equation 2.3)}$$

It is also equal to the square of the current multiplied by the resistance:

$$\text{Power} = \text{Current}^2 * \text{Resistance} \quad (P = R * I^2) \quad \text{(Equation 2.4)}$$

Energy is a measurement of power over a period of time. It demonstrates how much power is used/generated, by a device, typically over a period of an hour. In electrical systems, it is measured in watt-hours (Wh) and kilowatt-hours (kWh). For instance, a device usage is 10 watts of power, has an energy demand of 10Wh per hour. A solar panel that can generate 90 watts of power per hour, has an energy creation potential of 90Wh per hour.

However, because solar energy generation is so changeable, based on weather conditions, temperature, the time of day and so on, a new unit is now shown specifically for solar systems: a watt-peak rating (Wp). A watt-peak rating shows how much power can be generated by a solar panel at its peak rating. It points out the fact that the amount of energy a solar panel can generate.

2.3. Solar Charge Controller

A solar charge controller has to utilize the power going from solar panels to batteries. Battery life will be reduced significantly if overcharging it, and at worst it can be unusable.

The most basic charge controller is able to monitor the output voltage and open the circuit to stop charging if needed. The older controllers used a mechanical relay to open/close the circuit, to start/stop power going to the batteries.

More modern charge controllers are Pulse Width Modulation (PWM) and Maximum Power Point Tracking (MPPT), which mostly used nowadays due to its capabilities.

2.3.1. Pulse Width Modulation (PWM)

When the batteries get closer to its fully charge, PWM controller slowly reduces the amount of power applied to these batteries by using complex algorithms. It is built on a time-tested technology, which has been used for years in solar system. The controllers are sized up to 60 Amps, which are available in many sizes for a wide range of applications.

2.3.2. Maximum Power Point Tracking (MPPT)

MPPT controllers are the most recent and the best type of solar charge controller. It basically converts excess Voltage into Ampere that has advantages in different areas. By converting voltage into ampere, the charge voltage steadily stay at its requires level. The time needed for the battery to fully charge will be reduced.

Another area that is enhanced by an MPPT charge controller is the power loss. Higher voltage going from the PV panels to the charge controller results less power loss in the wire than lower voltage.

Finally, MPPT controllers prevent reverse-current flow. It is able to detect whether the energy is coming from the PV panels and disconnect the circuit from the battery.

We have an essential chart that illustrates the current-voltage characteristics as shown in Figure. Open circuit voltage v_{OC} and short circuit current $i_{sc} = J_{sc} * A$ are the maximum voltage and

maximum current of the i-v curve. When the resistance imposed, the current drops down to the point where the current is 0, the voltage is maximum and the resistor is infinite.

In practical, we attempt to operate the PV cell that has maximum power output as $P_{max} = v_{mp} * i_{mp}$.The operating point can be changed by adjusting the load resistance values. In Figure 2.2 we can see that operating point is not ideal because it does not maximize the power resistance. In this case, we use a DC to DC converter to adjust the operating voltage of the PV panels and deliver a different voltage to the load. It is called a maximum power point tracker (MPPT).

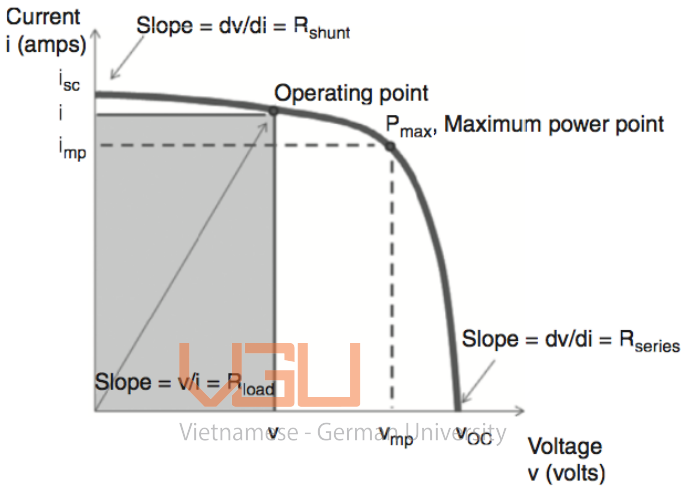


Figure 2.5 The i-v curve for a photovoltaic device showing open-circuit voltage, short-circuit current, maximum power point, and point corresponding to the actual operating voltage. [5]

Here are some pros and cons of PWM and MPPT solar charge controller:

PWM Solar Controllers	MPPT Solar Controllers
PROS	
– Inexpensive, usually selling for less than $350 – Sizes up to 60 Amps – Durable, most with passive heat sink style cooling	– Offer a potential increase in charging efficiency up to 30% – Offer the potential ability to have an array with higher input voltage than the battery bank

– These controllers are available in many sizes for a variety of applications	– Sizes up to 80 Amps – MPPT controller warranties are typically longer than PWM units – Offer great flexibility for system growth – MPPT is the only way to regulate grid connect modules for battery charging
CONS	
– The solar input nominal voltage must match the battery bank nominal voltage – There is no single controller sized over 60 amps DC as of yet – Many smaller PWM controller units are not UL listed – Many smaller PWM controller units come without fittings for conduit – PWM controllers have limited capacity for system growth – Can't be used on higher voltage grid connect modules	– MPPT controllers are more expensive, sometimes costing twice as much as a PWM controller – MPPT units are generally larger in physical size – Sizing an appropriate solar array can be challenging without MPPT controller manufacturer guides

Table 1 Pros and cons of PWM and MPPT controller [2]

2.4. Battery Bank

A solar system requires a battery bank to store solar energy absorbed from the sun and use it when needed. Deep cycle batteries such as Nickel-Cadmium Lead Acid (Seal, Flooded) or Lithium-Ion are main kinds of batteries used in PV system.

Storage of a few hours enable peak-shifting, demand response, and storage of the intermittent renewable energy resource.

Features	Lithium-Ion	Lead-Acid	Nickel-Cadmium
Weight	Light.	30% more than Lithium-Ion.	20%-35% more than Lithium-Ion.
Efficiency	Nearly 100%	Less than 80%.	70%-90%
Cycle Life	Rechargeable Lithium-ion batteries cycle 5000 times or more	400-500 cycles	Longer life cycle than Lead-Acid.
Voltage	Lithium-Ion batteries can maintain their voltage throughout the entire discharge cycle. It calls nominal area.	When discharge voltage of lead- acid drops consistently.	Nickel- Cadmium has the ability to maintain a steady voltage until it is almost completely depleted.
Energy Density	High	Low	Moderate
Memory effect	No memory effect	No memory effect	Noticeable memory. Can be eliminated by periodic conditioning
Cost	High	Low	Moderate
Environ-ment Impact	Lithium-ion batteries are a much cleaner technology and are safer for the environment	Lead-acid batteries require more raw material to achieve the same energy as a result making a much larger impact on the environment during the mining process.	Contain toxic metals and are generally being phased out. (Which present serious landfill and incinerated problems.)
Important note	Can permanently damage if stored at	The inefficiency of Lead-Acid battery leads	Requires fully discharge before recharge.

	too low discharge level. Lithium-Ion requires protection system which can limit voltage and current to avoid provoking and being over charged and discharged too far.	to a loss of 15 amps while charging and rapid discharging drop voltage quickly as well as reduces the batteries capacity.	

Table 2 Types of battery bank

2.5. DC/AC Inverters

2.5.1. Introduction

DC/AC inverters are used to convert a DC input voltage into symmetric ac output voltage of desired frequency and magnitude. The output voltage of ideal inverters should be sinusoidal. These inverters are commonly applied in variable voltages/frequencies AC supplies in adjustable speed drive, and constant regulated power supplies.

2.5.2. One-leg switch mode inverter

We will discuss about the Pulse Width Modulated (PWM) of one-leg switch mode inverter as shown below.

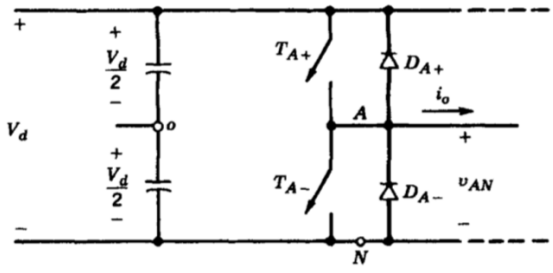


Figure 2.6 One-leg switch mode inverter [3]

In order to produce a sinusoidal output Voltage waveform, a sinusoidal control signal at the desired frequency is compared with a triangular waveform, amplitude $\widehat{V_{tri}}$

The amplitude modulation ratio m_a is defined as

$$m_a = \frac{\widehat{V_{control}}}{\widehat{V_{tri}}}$$

where $\widehat{V_{control}}$ is the peak amplitude of the control signal, $\widehat{V_{tri}}$ is kept constant.

The frequency modulation ratio m_f is defined as

$$m_f = \frac{f_s}{f_1}$$

where f_s is the switching (or carrier) frequency, f_l is the desired fundamental (or modulating) frequency.

When T_{A+} is on, $v_{Ao} = \frac{1}{2}V_d$. When T_{A+} is off, $v_{Ao} = -\frac{1}{2}V_d$

The peak amplitude of the fundamental frequency component

$$\widehat{(V_{Ao})_1} = m_a \cdot \frac{1}{2}V_d = \frac{\widehat{V_{control}}}{\widehat{V_{tri}}} \cdot \frac{V_d}{2}$$

The harmonics in the inverter output voltage waveform appear as sidebands, centered around the harmonic m_f, $2m_f$, $3m_f$ and so on. The frequencies at which voltage harmonics occur can be shown as

$$f_h = (jm_f \pm k)f_1$$

The harmonic order *h* corresponds to the *k*th sideband:

$$h = j(m_f) \pm k$$

The harmonic m_f should be an odd integer in order to result odd symmetry as well as half-wave symmetry.

The rms values of the fundamental frequency voltage is $V_{rms} = \frac{V_{peak}}{\sqrt{2}}$

$$(V_{Ao})_h = \frac{\widehat{(V_{Ao})_h}}{\sqrt{2}} = \frac{1}{\sqrt{2}} m_a \cdot \frac{V_d}{2} = \frac{1}{\sqrt{2}} \frac{V_d}{2} \cdot \frac{\widehat{(V_{Ao})_h}}{V_d/2}$$

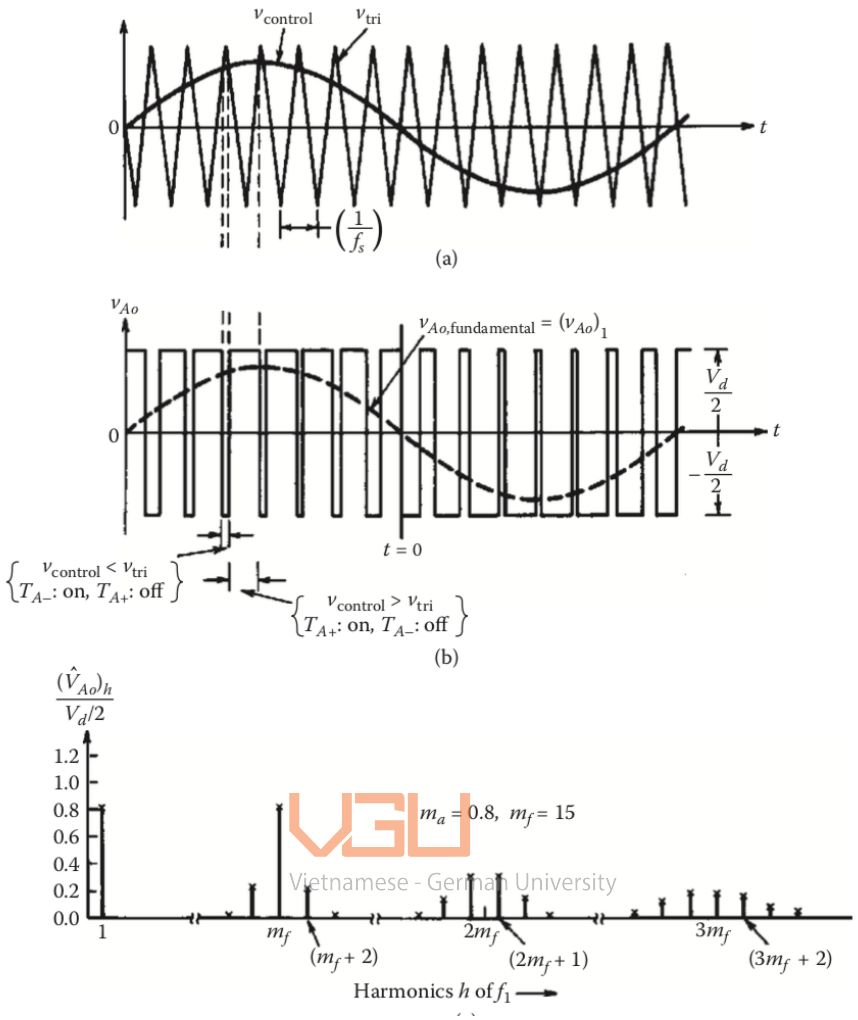


Figure 2.7 The representation of PWM on one-leg inverter, discrete signal graph of harmonics h [3]

2.5.3. Three-phase inverters

Three-phase inverters are frequently used to supply three-phase balanced loads (star-form or triangular form). We are now considering a square-wave mode inverter. For sufficiently large values of m_a, PWM degenerates into square-wave operation, and each switch is on for 180^0.

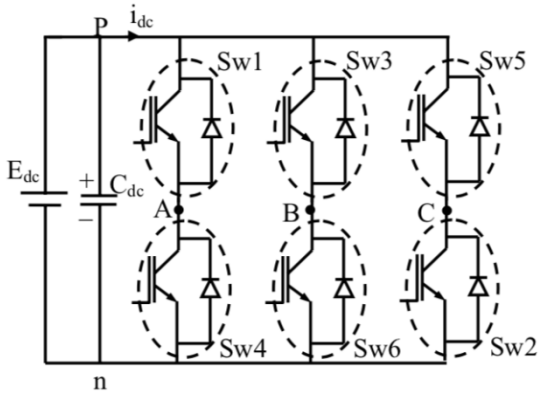


Figure 2.8 Three phase dc to ac inverter [7]

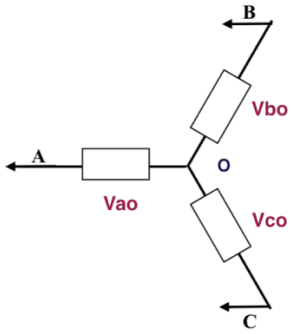


Figure 2.9 Star-connected 3-phase load [7]

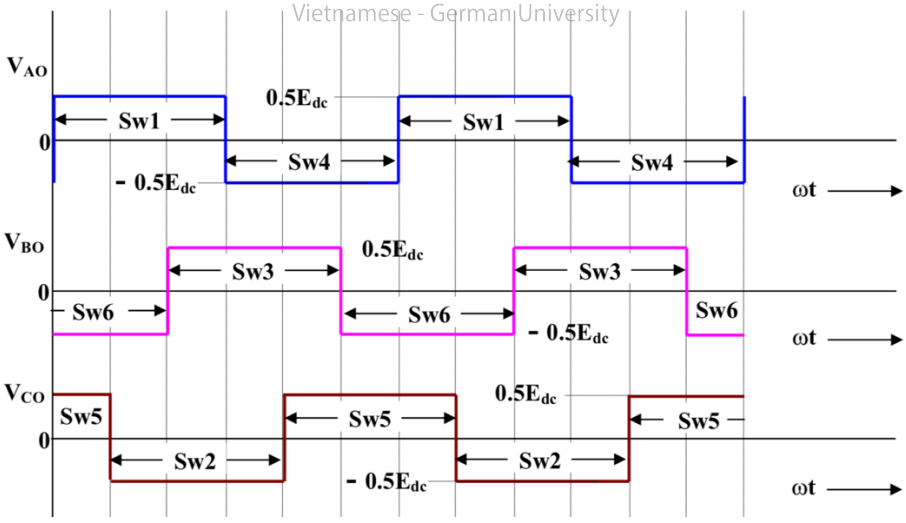


Figure 2.10 The operation of 180^0 degree firing of 3-phase inverter [7]

As illustrated from *figure 2.16*, Sw1 will start from 0 rad to π rad with the amplitude of $0.5E_{dc}$. After that, Sw4 turned on from π rad to 2π rad with the amplitude of $-0.5E_{dc}$.

Sw3 will be enabled simultaneously from $\frac{2\pi}{3}$ rad to $\frac{5\pi}{3}$ rad.

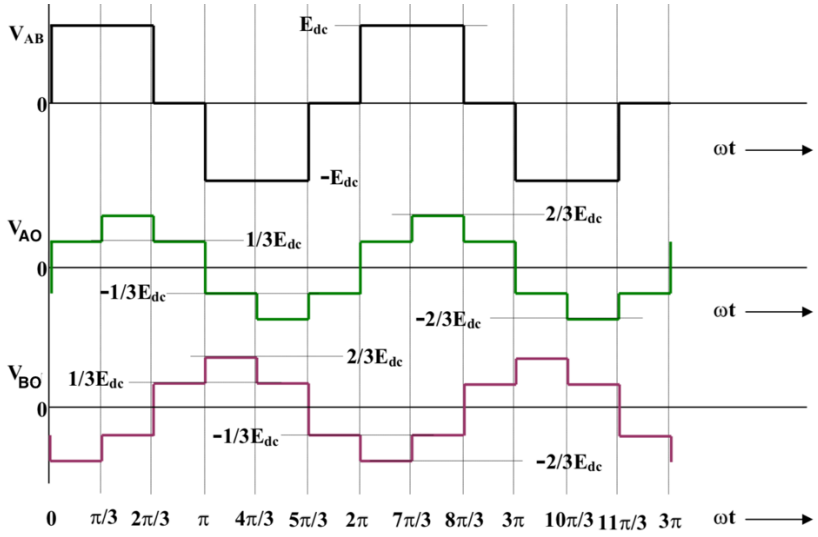


Figure 2.11 The voltage magnitude of 3-phase loads [7]

The magnitude of voltage V_{AO} from the start is $1/3E_{dc}$. After lasting for $\frac{\pi}{3}$ rad, it is increasing to the maximum value $2/3E_{dc}$ from $\frac{\pi}{3}$ rad to $\frac{2\pi}{3}$ rad, then dropping to $1/3\ E_{dc}$. As a result of enabling time of Sw4, the magnitude of V_{AO} is $-1/3E_{dc}$ after π rad and reach its peak of $-2/3E_{dc}$.

Finally, the magnitude of V_{AB} is defined as $V_{AB} = V_{AO} - V_{BO}$

2.6. AC Loads

2.6.1. Single-phase AC load

In Vietnam and many other countries, single-phase electricity is used as typical power supplies for residential houses. It is commonly a three-wire 120/240 VAC configuration service as shown in picture. $V^i_{NB} = V^i_{NR} = 120$ volts, $V^i_{RB} = 240$ V

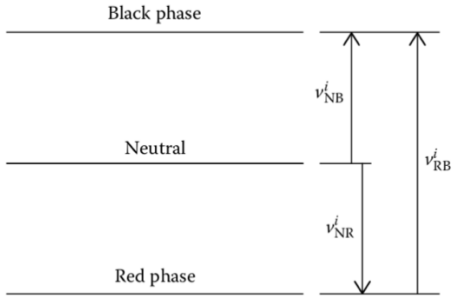


Figure 2.12 Single-phase AC load

The power consumption in a single-phase AC circuit:

$$S = |V||I| \cos(\theta_V - \theta_I)$$

where $\cos(\theta_V - \theta_I)$ is the power factor, $\theta_V - \theta_I$ is the angle of impedance.

Complex power in single-phase AC circuit:

$$S = |V||I| (\cos\theta + j \sin\theta) = P + jQ = \frac{|V|^2}{Z}$$

P is the active power (real), Q is the reactive power, Z is the total impedance.

As the steady state, v and I can be presented by phasor forms as V and I. The design of steady state operation as:

$$V = V_{rms}\angle\theta_V = \frac{120}{\sqrt{2}}\angle\theta_V$$

$$I = I_{rms}\angle\theta_I \text{ with } \theta_I = \omega t + \theta_{I,0}$$

2.6.2. Three-phase AC load

A three-wire three-phase circuit is usually more economical than an equivalent two-wire single-phase circuit at the same line to ground voltage because it uses less conductor material to transmit a given amount of electrical power.

Three sinusoidal voltages have the same amplitude, frequency, but differing by 120 degree

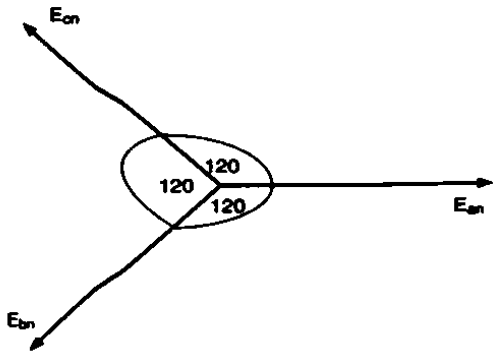


Figure 2.13 Three-phase circuit illustrated in vector field [8]

$$E_{cn} = E\angle 0$$

$$E_{bn} = E\angle -120 = E\angle 240$$

$$E_{cn} = E\angle 120 = E\angle -240$$

$$E_{\mathrm{a}} + E_b + E_c = 0$$

2.6.2.1. 3-phase balanced Y to Y system (Y as load)

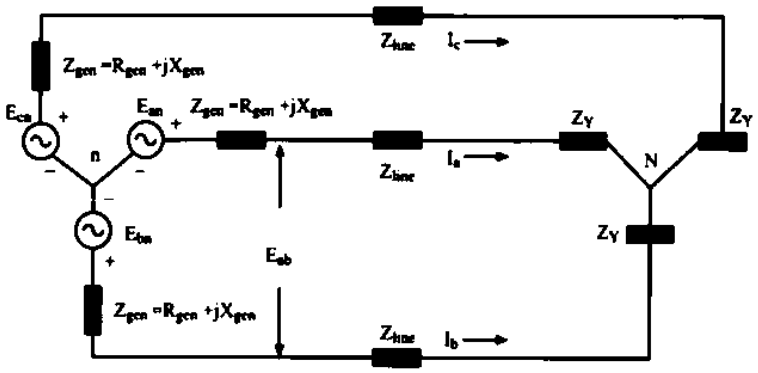


Figure 2.14 Three-phase Y to Y load [8]

Line to line voltage:

$$E_{ab} = E_{an} - E_{bn}$$

$$E_{ab} = E_{ac} = E_{bc} = E\sqrt{3}$$

and $E_{ab} + E_{bc} + E_{ca} = 0$

Line to line current:

$I_a = \frac{E_{an}}{Z_Y}$; $I_b = \frac{E_{bn}}{Z_Y}$; $I_c = \frac{E_{cn}}{Z_Y}$

and $I_{ab} + I_{bc} + I_{ca} = 0$

E_{an}, E_{bn}, E_{cn} is line to neutral voltage (phase voltage)

I_a, I_b, I_c is line current (phase current)

2.6.2.2. 3-phase balanced Y-∇ system (∇ as load)

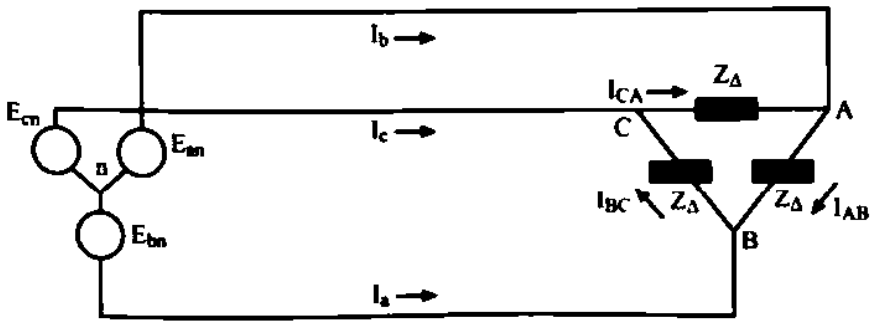


Figure 2.15 Three-phase Y to ∇ load [8]

Phase current:

$$I_{AB} = \frac{E_{AB}}{Z_\nabla} = \sqrt{3}\frac{E}{Z_\nabla};\ I_{AC} = \frac{E_{AC}}{Z_\nabla} = \sqrt{3}\frac{E}{Z_\nabla};\ I_{BC} = \frac{E_{BC}}{Z_\nabla} = \sqrt{3}\frac{E}{Z_\nabla}$$

Voltage:

$$E_{AB} = E_{AC} = E_{BC} = E\sqrt{3}$$

I_{AB}, I_{AC}, I_{BC} is phase current (or ∇ current).

3. Design of a small-sized off-grid PV generation system

3.1. Introduction

This chapter presents a design of 300Wp system with its detailed specifications & main function. Commercial products with brands & model numbers are also listed in the following tables. The flows of this design are:

- Three of 100W PV panels connected in series to collect solar irradiation and convert it to DC power.
- The MPPT charge controller operates as a system brain to control the charging/discharging process for battery from PV power source. The controller is also a sufficient protector against harsh conditions as overload, and battery over-charge/over-discharge. Besides, it has positive/negative terminal for small DC load devices such as LED, motors, and so on.
- Batteries discharge DC power to the inverter, where 12V DC power is converted to 220V AC power, 50 Hz frequency. The charge controller controls this process to ensure over-discharge is not occurred.
- The DC/AC inverter distributes AC power for household usage.

A case study of a real community-sized PV system in Taiwan is mentioned. Its data will be used to analyze further in this project.

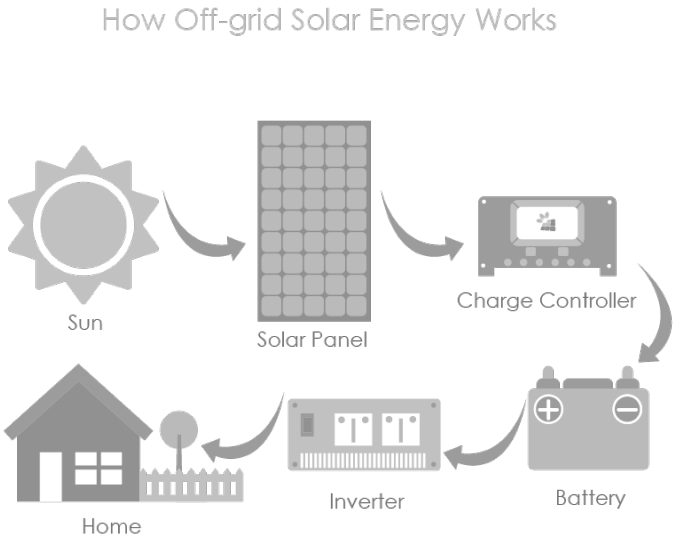


Figure 3.1 An example of a household PV system

3.2. Design of 300Wp PV system

3.2.1. Solar Panel

The system consists of 3 monocrystalline solar panels 100W connected in series. The voltage & current at maximum power is V = 56.34 V, and $I = 5.32A$.

When there is no connection between 2 terminal of PV panels, the voltage is $V_{oc} = 22.64 * 3 = 67.92V$. The current which PV panels can withstand is not greater than $I_{sc} = 5.7A$.

Model Number	SG36M-100W
Rated Maximum Power (P_{max})	100Wp
Open-Circuit Voltage (V_{oc})	22.64V
Short-Circuit Current (I_{sc})	5.7A
Voltage at P_{max} (V_{mp})	18.78V
Current at P_{max} (I_{mp})	5.32A

Table 3 Specifications of PV panels

3.2.2. MPPT Solar Charge Controller

PV panels connect directly to a solar charge controller using MPPT technology (section 2.3.2 for theoretical background). Main functions of this charge controller are to track out the maximum DC power of PV array at any time, and to charge the battery when needed.

The charge controller measures battery temperatures using Temp sensors and stop the charge/discharge state when overheat occurred.

The PV open-circuit voltage V_{oc} must be less than 100V, and it is sufficient since PV panels $V_{oc(PV)}$ was only 67.92V.

The controller can fully sustain hard conditions such as:

- PV over current: it will limit charge power in rated charge power. An over-sized PV will not operate at maximum power point.
- PV & battery short circuit: when short-circuit occurs, the controller will stop charging.
- PV & battery reverse polarity: when wires connected to wrong polarity, no damage has been done to the controller.
- Battery overheating: if the battery temperature exceeds 65^{o}C, the controller stops working and reconnects below 55^{o}C.
- Load overload: if the load current exceeds the maximum load current rating 1.05 times, the controller will disconnect the load.

The charge controller also has negative & positive terminal for small voltage DC loads.

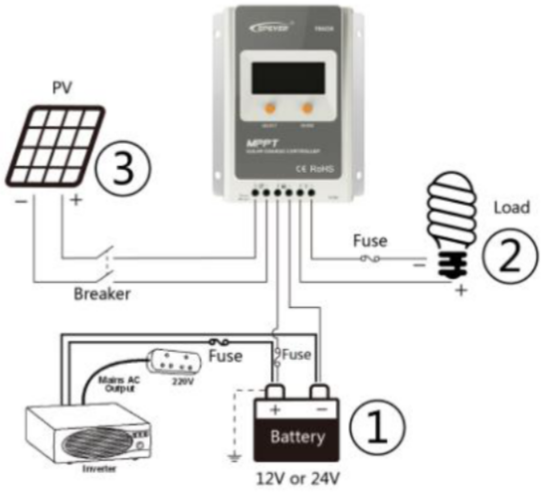


Figure 3.2 A MPPT solar charge controller

Model	Tracer3210A
Rated Charge Power	390W/12V
Rated Charge Current	30A
Maximum PV Array Power	1170W/12V
Maximum PV Open-Circuit Voltage	100V
Protection	Battery Overcharge, Battery Overheating, Load Overload, Load Short-Circuit

Table 4 Specification of the MPPT solar charge controller

3.2.3. Battery

An energy storage is critical for an off-grid PV system. An average daily sunlight time in Binh Duong city is about 4 hours so power has to be stored for shaded sun time. There is a well-known formula to calculate the energy storage needed for a PV system:

$$Battery\ Capacity(Ah) = \frac{Sum\ of\ power\ consumption\ (Wh)\ daily}{Battery\ efficiency * DOD * battery\ voltage}$$

(formula 3.1)

whereas

- DOD stands for Depth of Discharge; it is used to describe how deeply the battery is discharged. If a battery is 100% charged, it means DOD is 0%. If the battery has discharged 30% of its capacity, the DOD is 30%. Here we choose 60% for longer lifespan.

- Battery efficiency: The range is from 70% to 100%, depends on batteries type. Here we use 85% as an average value.

- Sum of power consumption daily: the sum of total power consumption in a day, which is 300Wh. The table below used as reference, which illustrates power consumption of household appliances daily:

Index	Device	Quality	Power (W/h)	Usage time (hours/day)	Total power consumption (Wh/day)
1	Compact light	5	26	2	260
2	DVD	1	40	3	120
3	Laptop	1	60	5	300
4	Television	1	80	1	80
In total (daily)					760W

Table 5 An example of household appliances power usage

The battery capacity needed per day is calculated:

$$Battery\ Capacity = \frac{760\ Wh}{85\% * 60\% * 12V} = 124\ Ah$$

This means if there is no power coming from PV panels, the battery with 124 Ah capacity is able to supply 760W AC loads for a day (assumed DC/AC inverter efficiency is 100%).

200Ah AGM (Absorbed Glass Mat) sealed battery is chosen for this case due to economic benefit and long life (12 years).

Nominal Voltage		12V
Nominal Capacity (10 hours rate)		200Ah
Charge (Constant Voltage) 25^0C (77^0F)	Float	Voltage 13.6V – 13.8V
	Cycle	Voltage 14.4V – 14.9V
Self-Discharge	After 3 months	90%
	After 6 months	80%
	After 12 months	62%
Terminal Type		Cooper
Life Span		12 years

Table 6 Specifications of a battery

3.2.4. DC/AC Inverter

DC/AC inverter is used to convert DC voltage which comes from battery to 220V AC voltage, 50Hz. The total rated power needed for the system is 300W, so the maximum power of inverter is equal or greater than $300 * 100\% = 300W$. A 350W inverter is chosen to fulfil this requirement.

The inverter input voltage must sustain the battery discharge voltage, which is around 12V. The converted voltage can vary in range 220VAC +/- 5% (or 209 VAC – 231 VAC).

If a load consumes more power to turn on like microwaves & water heater, there is a peak power of 1000W in 5 seconds for these appliances. The power then consumed normally at 350W on average.

The output waveform is a sine-wave with very low harmonic distortion and clean power like utility supplied electricity. Inductive loads such as microwaves and motors can run faster, cooler and less noise.

The inverter which has its own protection against overload or anti-islanding, immediately stop feeding power where no load is used.

Brand	Vuphong
Power	500VA/350W
Input Voltage	12VDC
Output Voltage	220VAC +/- 5%
Peak Power (5s)	1000W
Frequency	50Hz
Form	Sine-wave
Protection	Overload, anti-islanding

Table 7 Specifications of an inverter

Figure 3.3 An example of 350W DC/AC inverter

3.3. Case study for the project

3.3.1. Introduction

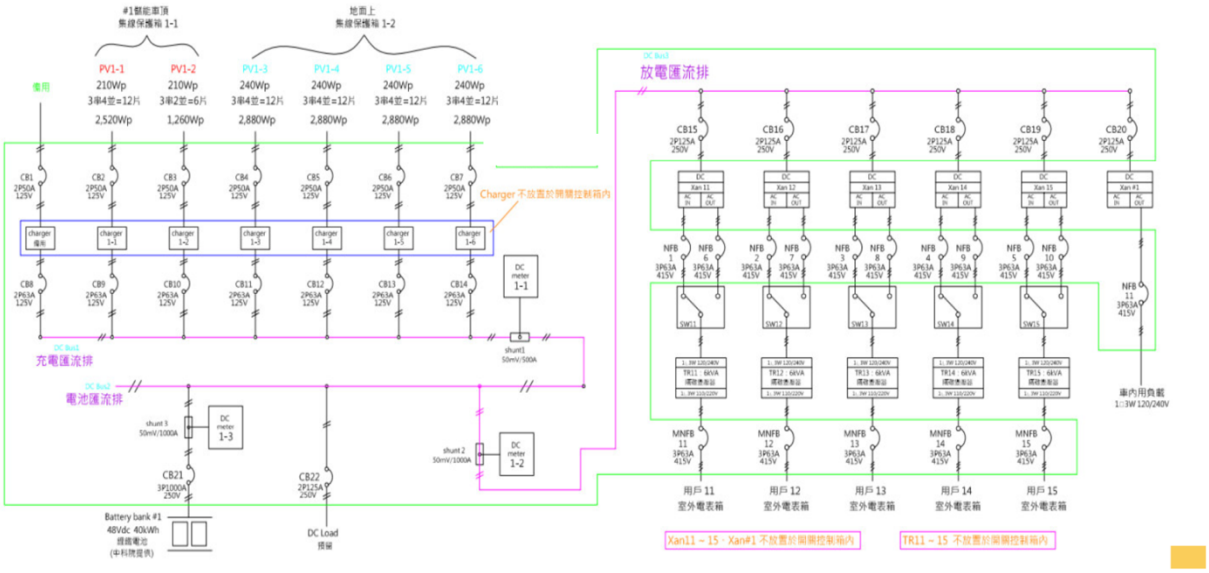


Figure 3.4 A single-line diagram of the battery station-2 in the LVDC Dongkeng microgrid

In this project, the data is taken from the 1380W Dongkeng microgrid in Taiwan. Dongkeng microgrid project is a real community-sized DC microgrid structure using PV generation systems and battery energy storage devices (BESS). The system is chosen because of similarities with the design of 300W off-grid PV system such as DC structure, MPPT solar charge controller, AC/DC inverter, battery, loads and so on. The Dongkeng microgrid system consists of:

- Two 210Wp and four 240Wp PV Panels.
- Six MPPT solar charge controllers.
- A battery bank of 48VDC and 40kWh.
- Six DC Loads.
- AC Loads.
- 2 DC meters. A meter is for measure voltage, power and current of DC loads. Another meter is used to measure charge/discharge voltage, power, current and temperature of the battery bank.

3.3.2. Specifications of the Case Study module components

Four tables below show the specific specifications of the module components: PV panels (210W & 240W), MPPT solar charge controller, and the battery bank. The data format is discussed in Chapter 4.2.1 (Data Sources).

Maximum power (P_{max})	210W
Maximum power voltage (V_{mpp})	26.6 V
Maximum power current (I_{mpp})	7.9 A
Open circuit voltage (V_{oc})	33.2 V
Short circuit current (I_{sc})	8.58 A

Table 8 Specifications of 210W PV panels

Maximum power (P_{max})	240W
Maximum power voltage (V_{mpp})	29.93 V
Maximum power current (I_{mpp})	8.02 A
Open circuit voltage (V_{oc})	37.43 V
Short circuit current (I_{sc})	8.55 A

Table 9 Specifications of 240W PV panels

Nominal System Voltage	48 VDC
Maximum PV Open Circuit Voltage	150 VDC
Maximum Battery Current	45 Amps
Nominal Maximum Solar Input	2400 Watts
Battery Operating Voltage Range	5-72 volts DC
Protections	Overload, short circuit, high voltage.

Table 10 Specifications of the MPPT solar charge controller

Nominal Capacity	64.0 ± 2Ah
Nominal Voltage	51.2V
Nominal Power	3.2KWh
Rated Charge Voltage	57.6V
Maximum Charge Voltage	58.4V
Rated Charge Current	18 A
Rated Discharge Current	18 A
Maximum Charge Current	60 A
Maximum Discharge Current	120 A

Table 11 Specifications of the battery bank

4. Development of an Energy Information Communication Framework for PV Generation System

4.1. Introduction

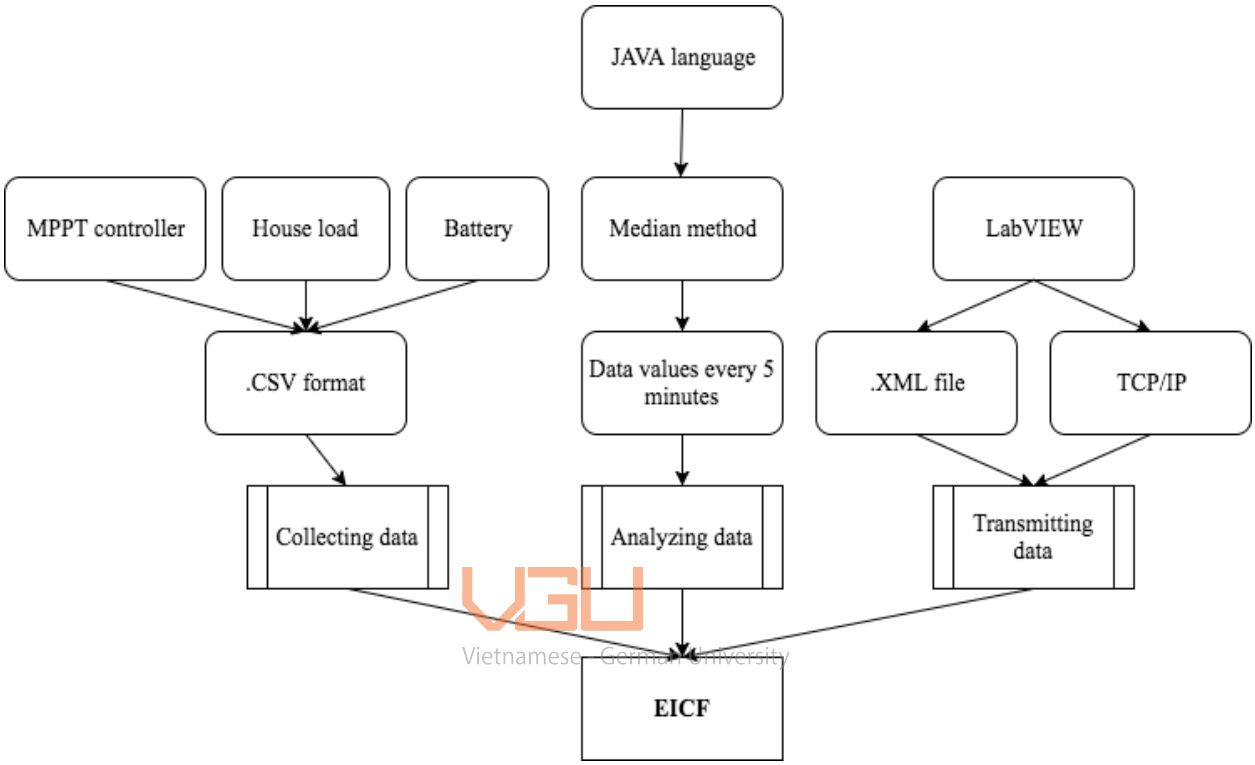


Figure 4.1 A module illustrates steps to develop EICF

In this chapter, the development of EICF includes three steps:

- Collecting data: the data which is coming from solar charge controller & battery such as PV voltage/current, and battery charge/discharge power are collected and saved under .csv format.

- Analyzing data: using JAVA program to synchronize times in tables of recorded data. Using median method to remove outliers of the data & extracting a median value every 5 minutes.

- Transmitting data in LabVIEW: the analyzed data is transmitted under .xml documents extracted from LabVIEW program. .XML files are exchanged to an aggregator via online network using a TCP/IP method.

This chapter introduces briefly all of three steps and their important components. Basic knowledge is also shown for better understanding.

4.2. Collecting data

4.2.1. Data sources

This project has 3 main source of data from MPPT charge controller, battery and house load.

The MPPT charge controller has a log-data module, which records continuously every 3 minutes and 20 seconds. The log-data module extracts .xls file, which contains information about date, PV voltage/current/power, battery voltage/current/power, battery temperature, total PV power (kWh) and daily consumption power (Wh).

A	B	C	D	E	F	G	H	I	J	K
		TSMPPT1_V	TSMPPT1_I	TSMPPT1_P	BAT_V	BAT_I	BAT_P	TEMP	Total kWH	Daily WH
11/14/14	上午 05:03:25	0.18	0	0	51.49	0	0	19	3238	5230
11/14/14	上午 05:06:45	0.29	0	0	51.49	0	0	19	3238	5230
11/14/14	上午 05:10:05	0.15	0	0	51.48	0	0	19	3238	5230
11/14/14	上午 05:13:25	0.23	0	0	51.48	0	0	19	3238	5230
11/14/14	上午 05:16:45	0.23	0	0	51.48	0	0	19	3238	5230
11/14/14	上午 05:20:05	0.2	0	0	51.48	0	0	19	3238	5230

Figure 4.2 A .xls file of the log-data module

Note that TSMPPT_V, TSMPPT1_I, TSMPPT1_P is the voltage, current and power of MPPT charge controller 1, respectively. BAT_V, BAT_I and BAT_P is the voltage, current and power of the battery. TEMP is the temperature of the battery, and Daily Wh is the daily consumption power.

The battery meter extracts .xls file, which contains information about date, battery module power(W), and charging/Discharging power of the battery(kW). The meter records every 10 seconds.

The negative sign shows the discharging state, and the positive sign shows the charging state.

A	B	C	D	E
Day	**Time**	**Battery Module Power(W)** 正值為充電，負值為放電	**Charging/Discharging Power of Battery (kW)**	**Charging/Discharging Power of Battery (kW)**
11/14/14	上午 01:00:05	-146	-0.146	-0.14590748
11/14/14	上午 01:00:15	-145	-0.145	-0.14486842
11/14/14	上午 01:00:25	-143	-0.143	-0.1427974
11/14/14	上午 01:00:35	-149	-0.149	-0.14900448
11/14/14	上午 01:00:45	-142	-0.142	-0.14176396
11/14/14	上午 01:00:55	-147	-0.147	-0.14694338

Figure 4.3 A .xls file of the battery meter

Lastly, the house load meter extracts data about date, user's main power, load power, and PV power supplied. The meter records every 2 seconds.

A	B	C	D	E	F	G	H	I	J	K
Day	**Time**	**Users 27-1 Main power**	**User 27-1 Load power**	**User 27-1 Green power supply**	**User 26 main power**	**User 26 Load power**	**User 26 Green power supply**	**User 26-3 mains power**	**User 26-3 Load power**	**User 26-3Green power supply**
11/14/14	0:00:02	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05
11/14/14	0:00:04	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05
11/14/14	0:00:06	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05
11/14/14	0:00:08	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05
11/14/14	0:00:10	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05
11/14/14	0:00:12	0.55	-0.46	0.09	0.01	-0.02	-0.01	0.21	-0.17	0.05

Figure 4.4 A .xls file of the house load meter

4.2.2. CSV format

Comma-separated values (CSV) file is a delimited text file that uses a comma to separates value. It is often used to exchange data between applications.

For example, a spreadsheet contains the following data:

Date	Time	Charger's Output Current	Inverter Output Current	Battery Current
11/14/2014	12:03:00	0.0	15.8	-15.8
11/14/2014	12:06:00	0.0	16.0	-16.0

Table 12 An example of data

The above data could be represented in a CSV-formatted file as follows:

Date, Time, Charger's Output Current, Inverter Output Current, Battery Current
11/14/2014, 12:03:00, 0.0, 15.8, -15.8
11/14/2014, 12:06:00, 0.0, 16.0, -16.0

Table 13 The table represented in .csv format

CSV formats are best used to represent set or sequences of records in which each record has an identical list of fields. In this project, all of the records data are in .csv formats. Therefore, the data can be used with any spreadsheet program, such as Microsoft Excel, Open Office Calc, or Google Spreadsheets.

4.3. Analyzing data

4.3.1. JAVA Programming Language

4.3.1.1 Introduction

JAVA is a computer software developed by Sun Microsystems, and later acquired by Oracle company. It is a well-known programming language which has a good execution engine, and a variety range of libraries. The list below shows some of its packages, such as Java.util.*, Java.util.regex.*, Java.io*, and Java.lang.Math. JAVA package used in my application are listed in Table 14.

Java.util.*;
Java.util.regex.*;
Java.io.*;
Java.lang.Math;

Table 14 JAVA packages

There are two principal products of the JAVA family: JAVA SE Runtime Environment (JRE) and JAVA Development Kit (JDK).

The JRE provides libraries, the JAVA Virtual Machines, and other components to write in the JAVA programming language. There are two main program types for workstations: applets and applications. The applet is defined that the applet is started by a Web browser loading an HML

file. However, since applets are downloaded over the web, in consideration of the download lag to receive the files from the web server and then start them up in the browser. Application, on the other hand, is installed on the client, get started from the command line and suffer from n download limitations.

The JDK by far has been the most widely used software development. When starting with JAVA, many developers begin with Sun's development kit. JDK is free with latest version. The SDK is a set of a tool for creating JAVA programs. Standard JDK tools and utilities are listed in Table 10.

Basic Tools (javac, java, javadoc, apt, appletviewer, jar, jdb, javah, javap, extcheck)
Security Tools (keytool, jarsigner, policytool, kinit, klist, ktab)
Internationalization Tools (native2ascii)
Remote Method Invocation (RMI) Tools (rmic, rmiregistry, rmid, serialver)
Java IDL and RMI-IIOP Tools (tnameserv, idlj, orbd, servertool)
Java Deployment Tools (pack200, unpack200)
Java Plug-in Tools (htmlconverter)
Java Web Start Tools (javaws)
Java Troubleshooting, Profiling, Monitoring and Management Tools (JConsole, Java VisualVM)
Java Web Services Tools (schemagen, wsgen, wsimport, xjc)

Table 15 Standard JDK stools

Java is available for many operation system (OS) such as Microsoft Windows, macOS, as well as Linux. The advantage is that JAVA codes can be built and operated in any programs in different operating system.

4.3.1.2. Using JAVA in the project

In this project, JAVA is used to:

- Get date of recorded data such as day, and specific time (hour, minute, second).
- Code the median method and apply it for a day of recorded value.

- Extract a .csv file which include selected values in every 5 minutes.

The process will be discussed in chapter 5 in details.

4.3.2. Median Method

4.3.2.1. Introduction

The median is the middle number of the data set when in order from least to greatest. The focusing point is to find a value that occurred the most often. The median value can be found by 3 steps:

- Put all the numbers in numerical order.
- If there is an odd number of results, the median is the middle number.
- If there is an even number of results, the median will be the mean of the two central numbers.

The position of the median is $\{(n+1) \div 2\}^{th}\ value$, where n is the number of values in a set of data.

For instance, a set of voltage information in 1 minutes is A = {5.1, 4.8, 4.3, 3.5, 3.1} (Volts). First, the values are put in ascending order: {3.1, 3.5, 4.3, 4.8, 5.1}. The middle value is figured out by using the formula:

$$Median = \{(n+1) \div 2\}^{th}\ value = (5+1) \div 2 = 3$$

The third value in the data set will be the median. Since 4.3 is the third value, 4.3 (Volts) would be the median value.

In case of even number of values, assumed a set of DC current value B = {1, 2, 4, 5, 3, 2} (Amp). The values in ascending order: {1, 2, 2, 3, 4, 5}. The median value is

$$Median = \{(n+1) \div 2\}^{th}\ value = (6+1) \div 2 = 3.5$$

The median is the 3.5^{th} value in the data set meaning that it lies between the third and fourth values. The formula below is used to get the average value:

$$Average = (value\ below\ median + value\ above\ median) \div 2$$

$$= (third\ value + fourth\ value) \div 2 = (2 + 3) \div 2 = 2.5$$

The value 2.5 falls directly between the third and fourth values in this data set, so 2.5 (Amps) would be the median value.

4.3.2.2. Using median method in this project

First, assume all the recorded data is normally distributed. The goal of using median method is to find a value that occurred the most in a set of recorded data. Due to a large amount of data needs to be analyzed, a median value is selected every 5 minutes.

The median method has a considerable advantage over other method such as mean value. The median method is used to get rid of extremely high or extremely low values, which called outliers. These abnormal values may be due to variability in the measurement or recording process, and they cause serious problems in statistical analyses.

4.3.2.3. What is an outlier and how to detect

The table below illustrates an outlier of a set of data. Value 26th (BAT-P column) is extremely high in compare with nearby values. An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. In this project, the Box Plot graphical technique is used for identify outliers.

The box plot uses median and the lower and upper quartiles (defined as the 25th and 75th percentiles). If the lower quartile is Q_1 and the upper quartile is Q_3, then the difference $(Q_3 - Q_1)$ is called the interquartile range of IQ.

The following quantities (called fences) are used to identifying extreme values in the tails of the distribution:

- Lower inner fence: $Q_1 - 1.5 * IQ$
- Upper inner fence: $Q_3 + 1.5 * IQ$

- Lower outer fence: $Q_1 - 3 * IQ$
- Upper outer fence: $Q_3 + 3 * IQ$

A point beyond an inner fence is considered a mild outlier, a point beyond an outer fence is considered an extreme outlier.

Let's use BAT-P column, figure 4.5 as an example set of data. The data set of N = 39 ordered is A = {0, 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0, 4, 5, 17, 21, 32, 54, 66, 78, 97, 110, 125, 141, 159, 172, 250}. The computations are as follows:

$$Lower\ quartile = Q_1 = 0.25(N+1)^{th}\ ordered\ point = 10^{th}\ ordered\ point = 0$$

$$Upper\ quartile = Q_3 = 0.75(N+1)^{th}\ ordered\ point = 30^{th}\ ordered\ point = 54$$

$$Interquartile\ range = IQ = Upper\ quartile - Lower\ quartertile = 54 - 0 = 54$$

$$Lower\ inner\ fence = Q_1 - 1.5 * IQ = 0 - 1.5 * 54 = -81$$

$$Upper\ inner\ fence = Q_3 + 1.5 * IQ = 54 + 1.5 * 54 = 135$$

$$Lower\ outer\ fence = Q_1 - 3 * IQ = 0 - 3 * 54 = -162$$

$$Upper\ outer\ fence = Q_3 + 3 * IQ = 54 + 3 * 54 = 216$$

From the example of the fence points and the data, the value 250 exceeds the upper inner fence and stands out as an extreme outlier.

1			TSMPPT1_V	TSMPPT1_I	TSMPPT1_P	BAT_V	BAT_I	BAT_P
2	11/14/14	上午 05:03:25	0.18	0	0	51.49	0	0
3	11/14/14	上午 05:06:45	0.29	0	0	51.49	0	0
4	11/14/14	上午 05:10:05	0.15	0	0	51.48	0	0
5	11/14/14	上午 05:13:25	0.23	0	0	51.48	0	0
6	11/14/14	上午 05:16:45	0.23	0	0	51.48	0	0
7	11/14/14	上午 05:20:05	0.2	0	0	51.48	0	0
8	11/14/14	上午 05:23:25	0.18	0	0	51.47	0	0
9	11/14/14	上午 05:26:45	0.17	0	0	51.47	0	0
10	11/14/14	上午 05:30:05	0.16	0	0	51.47	0	0
11	11/14/14	上午 05:33:25	0.25	0	0	51.47	0	0
12	11/14/14	上午 05:36:45	0.25	0	0	51.46	0	0
13	11/14/14	上午 05:40:05	0.21	0	0	51.46	0	0
14	11/14/14	上午 05:43:25	0.33	0	0	51.45	0	0
15	11/14/14	上午 05:46:45	0.42	0	0	51.45	0	0
16	11/14/14	上午 05:50:05	0.56	0	0	51.45	0	0
17	11/14/14	上午 05:53:25	1.04	0	0	51.44	0	0
18	11/14/14	上午 05:56:45	2.22	0	0	51.44	0	0
19	11/14/14	上午 06:00:05	5.23	0	0	51.44	0	0
20	11/14/14	上午 06:03:25	13.8	0	0	51.43	0	0
21	11/14/14	上午 06:06:45	23.55	0	0	51.43	0	0
22	11/14/14	上午 06:10:05	34.36	0	0	51.43	0	0
23	11/14/14	上午 06:13:29	44.34	0	0	51.42	0	0
24	11/14/14	上午 06:16:49	51.69	0	0	51.42	0	0
25	11/14/14	上午 06:20:09	51.73	0.02	1	51.42	0.1	5
26	11/14/14	上午 06:23:29	51.75	0.054	3	51.42	0.136	250
27	11/14/14	上午 06:26:49	51.75	0.129	7	51.43	0.251	0
28	11/14/14	上午 06:30:09	51.99	0.076	4	51.43	0.068	4
29	11/14/14	上午 06:33:29	52.25	0.137	7	51.44	0.327	17
30	11/14/14	上午 06:36:49	52.98	0.21	11	51.46	0.417	21
31	11/14/14	上午 06:40:09	54	0.325	18	51.46	0.62	32
32	11/14/14	上午 06:43:29	64.63	0.554	36	51.5	1.047	54
33	11/14/14	上午 06:46:49	67.04	0.808	54	51.53	1.289	66
34	11/14/14	上午 06:50:09	68.38	1.023	70	51.57	1.509	78
35	11/14/14	上午 06:53:29	67.94	1.228	83	51.61	1.88	97
36	11/14/14	上午 06:56:49	71.09	1.375	98	51.66	2.131	110
37	11/14/14	上午 07:00:09	71.86	1.558	112	51.72	2.41	125
38	11/14/14	上午 07:03:29	73.49	1.763	130	51.78	2.72	141
39	11/14/14	上午 07:06:49	73.51	2.019	148	51.86	3.071	159
40	11/14/14	上午 07:10:09	74.59	2.17	162	51.93	3.311	172

Figure 4.5 An example of an outlier. Data from the MPPT table in Case Study (chapter 3)

4.4. Transmitting data

4.4.1. Design of the XML schema

4.4.1.1. Introduction

XML stands for Extensible Markup Language. It is a text-based markup language. Here are some of the reasons why this project uses XML format:

- XML is extensile: XML allows to create self-descriptive tags, or language, that suits the application.
- XML carries the data, does not present it: XML allows to store the data irrespective of how it will be presented.

XML is a public standard: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

Using XML to exchange information offers many benefits:

- XML can be used to exchange the information between organizations and systems, or to offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.

And more importantly, any type of data can be expressed virtually as an XML document.

4.4.1.2. A Simple XML Document Structure

```
<?xml version="1.0"endcoding =”UT-8”?>        //XML Declaration
<XMLelements>                                 //Root
  <Date>7/4/2018 8:18 AM</Date>               //Element “Date”
  <PVHour>13</PVHour>
  <PVPowerkW>14</PVPowerkW>
  <note>Peak power</note>
</XMLelements>

<XMLattribute Date="7/4/2018">                 //Attribute “Date”
  <PVHour>5</PVHours>
  <PVPowerkW>6</PVPowerkW>
  <note>low PV power</note>
</XMLattribute>
```

Table 16 An example of XML document

XML documents must have a root element that is the parent of all other elements. Document elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose.

- XML element: each XML element needs to be closed either with start or with end elements as shown <element>…..</element>
- XML attribute: an attribute specifies a single property for the element, using a name/value pair.

4.4.2. LabVIEW

4.4.2.1. Introduction

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments. It can be used in multiple operating system (OS) such as Microsoft Windows, macOS, and Linux.

LabVIEW is designed to interface with any kind of measurement hardware, specifically to extract information from set of acquired data of the measurement devices. LabVIEW also provides tools for user interface design, data visualization, report generation, and data management. Flexibility of LabVIEW environment combined with modular hardware solutions resulted in a customized, user-defined and scalable data acquisition.

LabVIEW develops a program's subroutines so called virtual instrument (VI). Each VI has 3 main parts: Front Panel, Block Diagram and Connector Panel.

- The Front Panel is built using controls and indicators, which allow user to supply data and display the result based on the input.
- The Block Diagram (or back panel) contains graphical source code. Each front panel has a terminal in the block diagram.
- Programs is built by dragging and dropping the icons, then wires them to make a complete block.

4.4.2.2. Using LabVIEW in the project

In this project, LabVIEW is used as a main design platform. There are three important modules to be designed: extract .xml file, TCP/IP server and TCP/IP client.

Extract .xml file module:

- After data is analyzed in JAVA, the information is written in .csv format. LabVIEW imports this .csv file and illustrates them by a table in Front Panel.
- By using the table, a graph is used to display each column value over time. Specific values are picked by Graph's cursors. A module is developed to read these cursor values.
- The cursor values are flattened to .xml format, which carries needed data.

TCP/IP server module: the main purpose is to send .xml file via interconnect network devices.

TCP/IP client module: the module receives .xml file, unflattens it for normal user to read.

4.4.3. TCP/IP

4.4.3.1. Introduction

Transmission Control Protocol/Internet Protocol (TCP/IP), is a suit of communication protocol used to interconnect network devices on the internet. [9]

TCP/IP indicates how information is exchanged over the internet by providing end-to-end communications as the way it broke into packets, transmitted, routed and received at the destination. It requires central management to be able to recover automatically from failure of any device on the network. Some of major advantages of TCP/IP are:

The compatibility of all operating systems, and all types of computer hardware & network.

By using TCP/IP with http protocol, we are able to send/receive the .XML file between VTN (Virtual Terminal Node) and VEN (Virtual End Node). VTN is known as the server/aggregator side, which sends schedule, electricity price/load object, and its target to VEN. VEN is the customer as residents, who update and report their electric usage.

4.4.3.2. Using TCP/IP in the project

TCP/IP communication provides a simple user interface that conceals the complexities of ensuring reliable network communications.

In this project, a server and a client will be developed in 2 different LabVIEW modules. The server module sends a .xml file through TCP/IP. The .xml file contains information about schedule, day and time, and power values. The client module receives and unflatten that .xml file for client to observe those analyzed values.

5. Results, Analysis and Discussion

5.1. Collecting data

As mentioned in section 4.2, data is collected from 3 different sources: MPPT charge controller, battery meter and house load meter. The source files are in .xls format, so Microsoft Excel program is used to convert these .xls file to .csv file as shown below:

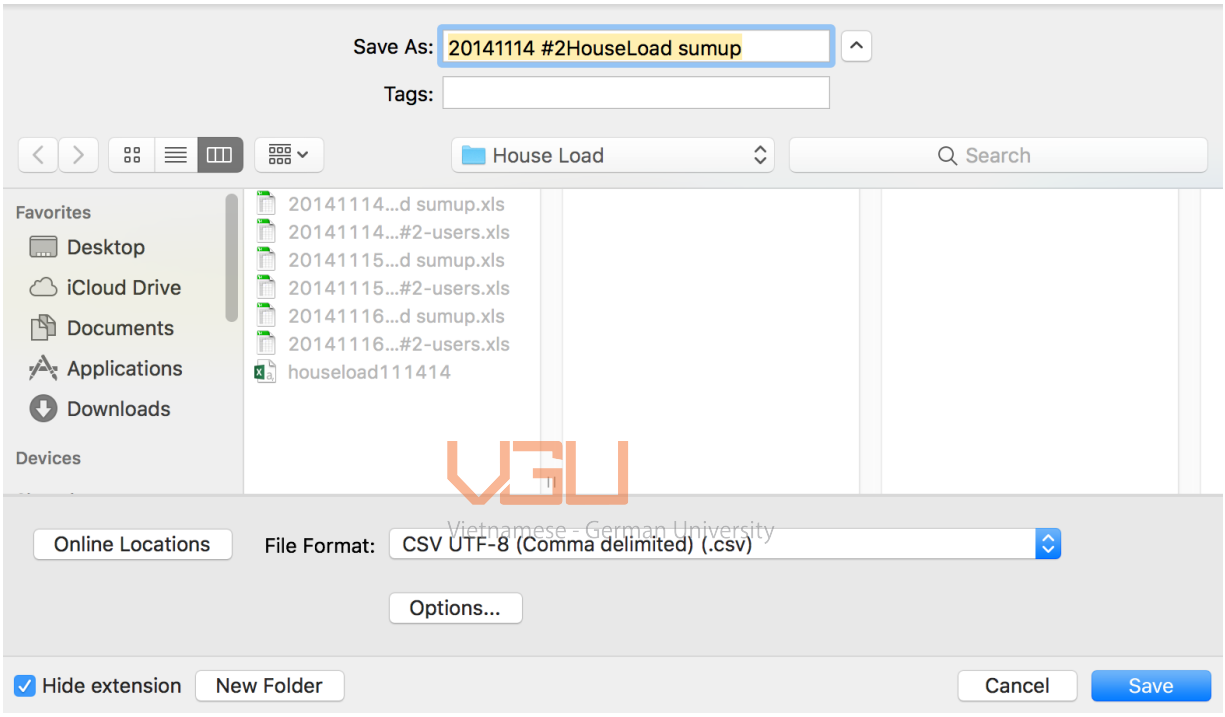


Figure 5.1 Save .xls file in .csv format

At File Format option, CSV UTF-8 (Comma Delimited) (.csv) is chosen. Note that this is the Microsoft Excel (ME) version for MacOS, so it might be different than ME versions in others operating system.

After converting all the files, there are 3 .csv files in a folder: MPPT.csv, House Load.csv and Battery.csv.

5.2. Analyzing data

In this project, IntelliJ IDEA CE program is used to develop JAVA. The program runs in MacOS system. The basic knowledge and brief introductions of data are mentioned in Chapter 4.2. The full source code is in the Appendix part.

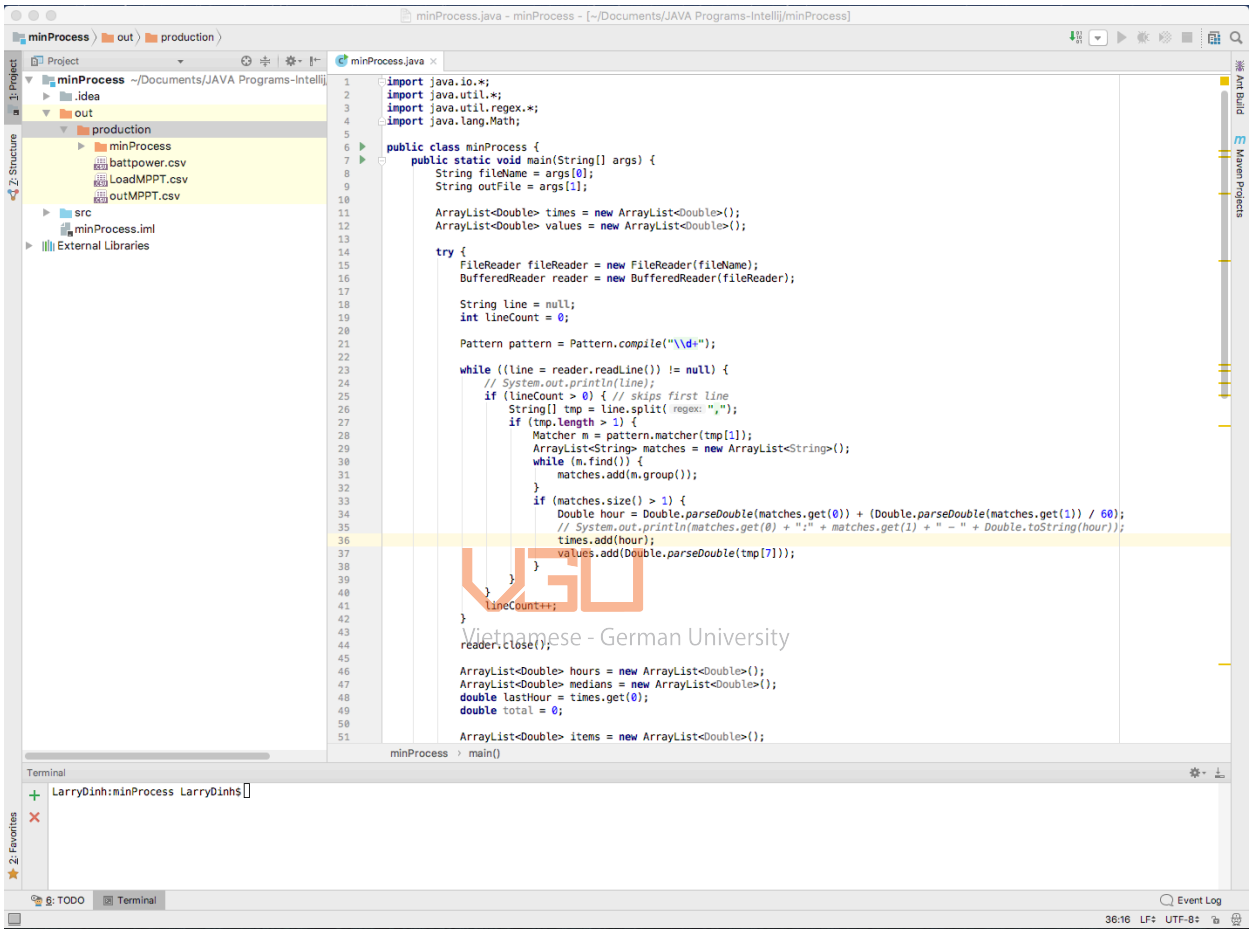


Figure 5.2 the user interface of IntelliJ IDEA program

The folder contains three .csv files will be in IntelliJ working space directory.

Here are the steps to extracted a calculated value:

- At the Terminal window, enter the line

java minProcess MPPT.csv outMPPT.csv

Table 17 Extract .xml file

	A	B	C	D	E	F	G	H
1			TSMPPT1_V	TSMPPT1_I	TSMPPT1_P	BAT_V	BAT_I	BAT_P
2	11/14/14	上午 05:03::	0.18	0	0	51.49	0	0
3	11/14/14	上午 05:06:4	0.29	0	0	51.49	0	0
4	11/14/14	上午 05:10:(	0.15	0	0	51.48	0	0
5	11/14/14	上午 05:13::	0.23	0	0	51.48	0	0
6	11/14/14	上午 05:16:4	0.23	0	0	51.48	0	0
7	11/14/14	上午 05:20:(	0.2	0	0	51.48	0	0
8	11/14/14	上午 05:23::	0.18	0	0	51.47	0	0
9	11/14/14	上午 05:26:4	0.17	0	0	51.47	0	0
10	11/14/14	上午 05:30:(	0.16	0	0	51.47	0	0
11	11/14/14	上午 05:33::	0.25	0	0	51.47	0	0
12	11/14/14	上午 05:36:4	0.25	0	0	51.46	0	0
13	11/14/14	上午 05:40:(	0.21	0	0	51.46	0	0
14	11/14/14	上午 05:43::	0.33	0	0	51.45	0	0
15	11/14/14	上午 05:46:4	0.42	0	0	51.45	0	0
16	11/14/14	上午 05:50:(	0.56	0	0	51.45	0	0
17	11/14/14	上午 05:53::	1.04	0	0	51.44	0	0
18	11/14/14	上午 05:56:4	2.22	0	0	51.44	0	0
19	11/14/14	上午 06:00:(	5.23	0	0	51.44	0	0
20	11/14/14	上午 06:03::	13.8	0	0	51.43	0	0

Figure 5.3 the .csv file before calculated in JAVA

- The output file is in the same directory as the input file.

- An 'outMPPT.csv' file is extracted from the program as shown in Figure 5.4 below. JAVA code calculates the entire TSMPPT_I column, and outputs Time and Median values of that column.

- The Time column shows fractional numbers after calculated, with a selected data sized of 5 minutes. For example, from 5 to 6 hours, the table column shows A = {5.05; 5,1667; 5.2667; 5.3833; 5.5; 5.6, 5,717, 5.833, 5.933}. This set of data can be understand in hour value: B = {5h 3m (5 hours 3 minutes), 5h 10m, 5h 16m, 5h 23m, 5h 30m, 5h 36m, 5h 50m, 5h 55m}.

- The Median column illustrates the representative values corresponding to Time column values. In this Figure 5.4, the median column is the data of TSMPPT_I.

	A	B	C	D	E	F	G	H
1	Time	Median						
2	5.05	0						
3	5.16666667	0						
4	5.26666667	0						
5	5.38333333	0						
6	5.5	0						
7	5.6	0						
8	5.71666667	0						
9	5.83333333	0						
10	5.93333333	0						
11	6.05	0						
12	6.16666667	0						
13	6.26666667	0.01						
14	6.38333333	0.0915						
15	6.5	0.1065						
16	6.6	0.2675						
17	6.71666667	0.681						
18	6.83333333	1.1255						
19	6.93333333	1.4665						
20	7.05	1.891						

Figure 5.4 A file extracted from IntelliJ program

The code processes only 1 column of .csv data at a time. After finished to process one column of data, the code has to be changed the column position, and the program has to be rebuilt.

Each selected median value is shown corresponding to the time. The final .csv file is sorted manually as shown below:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Time	Load 1	Load 2	Load 3	Load 4	Load 5	Sum of Load (kW)	MPPT2	MPPT3	MPPT4	MPPT5	MPPT6	MPPT7	Sum of MPPT(W)	Sum of MPPT(kW)	Px(Pmpp t-Pload)	batt2	batt3	batt4	batt5	batt6	batt7	Sum of Batt(kW)
2	0	0.55	0.01	0.19	0.21	0.51	1.47	0	0	0	0	0	0	0	0	-1.47	0	0	0	0	0	0	0
3	0.183	0.55	0.01	0.16	0.21	0.515	1.445	0	0	0	0	0	0	0	0	-1.445	0	0	0	0	0	0	0
4	0.367	0.51	0.01	0.16	0.22	0.48	1.38	0	0	0	0	0	0	0	0	-1.38	0	0	0	0	0	0	0
5	0.55	0.53	0.01	0.16	0.25	0.59	1.54	0	0	0	0	0	0	0	0	-1.54	0	0	0	0	0	0	0
6	0.733	0.49	0.01	0.16	0.25	0.61	1.52	0	0	0	0	0	0	0	0	-1.52	0	0	0	0	0	0	0
7	0.917	0.49	0.01	0.16	0.25	0.62	1.53	0	0	0	0	0	0	0	0	-1.53	0	0	0	0	0	0	0
8	1.1	0.47	0.01	0.17	0.28	0.58	1.51	0	0	0	0	0	0	0	0	-1.51	0	0	0	0	0	0	0
9	1.283	0.47	0.01	0.17	0.29	0.59	1.53	0	0	0	0	0	0	0	0	-1.53	0	0	0	0	0	0	0
10	1.467	0.46	0.02	0.17	0.29	0.545	1.485	0	0	0	0	0	0	0	0	-1.485	0	0	0	0	0	0	0
11	1.65	0.46	0.02	0.16	0.28	0.41	1.33	0	0	0	0	0	0	0	0	-1.33	0	0	0	0	0	0	0
12	1.833	0.21	0.02	0.16	0.28	0.46	1.13	0	0	0	0	0	0	0	0	-1.13	0	0	0	0	0	0	0
13	2.017	0.22	0.02	0.16	0.27	0.45	1.12	0	0	0	0	0	0	0	0	-1.12	0	0	0	0	0	0	0

Figure 5.5 the file .csv file after sorted manually

5.3. Transmitting data

This chapter is the most important part of this project. In this chapter, LabVIEW programming techniques which used to developing the user interface is considered.

Each block diagram will be discussed in details about main functions and connection between VIs. Tables contains definition about each VI usage are also shown for greater understanding.

There are three main VIs: Extract .xml file module, TCP/IP client module and TCP/IP server module. Let's discuss the Extract .xml file module.

5.3.1. Extract .xml file Module

Two pictures below are the Front Panel and the Block Diagram of the module. The Front Panel contains:

- The path of the final .csv file and its illustrated table.
- The XY Graph with PV/load power continuous graph.
- Cursors to pick significant points in the graph such as maximum load power, and maximum PV power.
- Indicators to show cursors values.
- The XML file path out and the XML Test Summary indicator.

Whereas, the Block Diagram contains codes to connect and control the Front Panel objects.

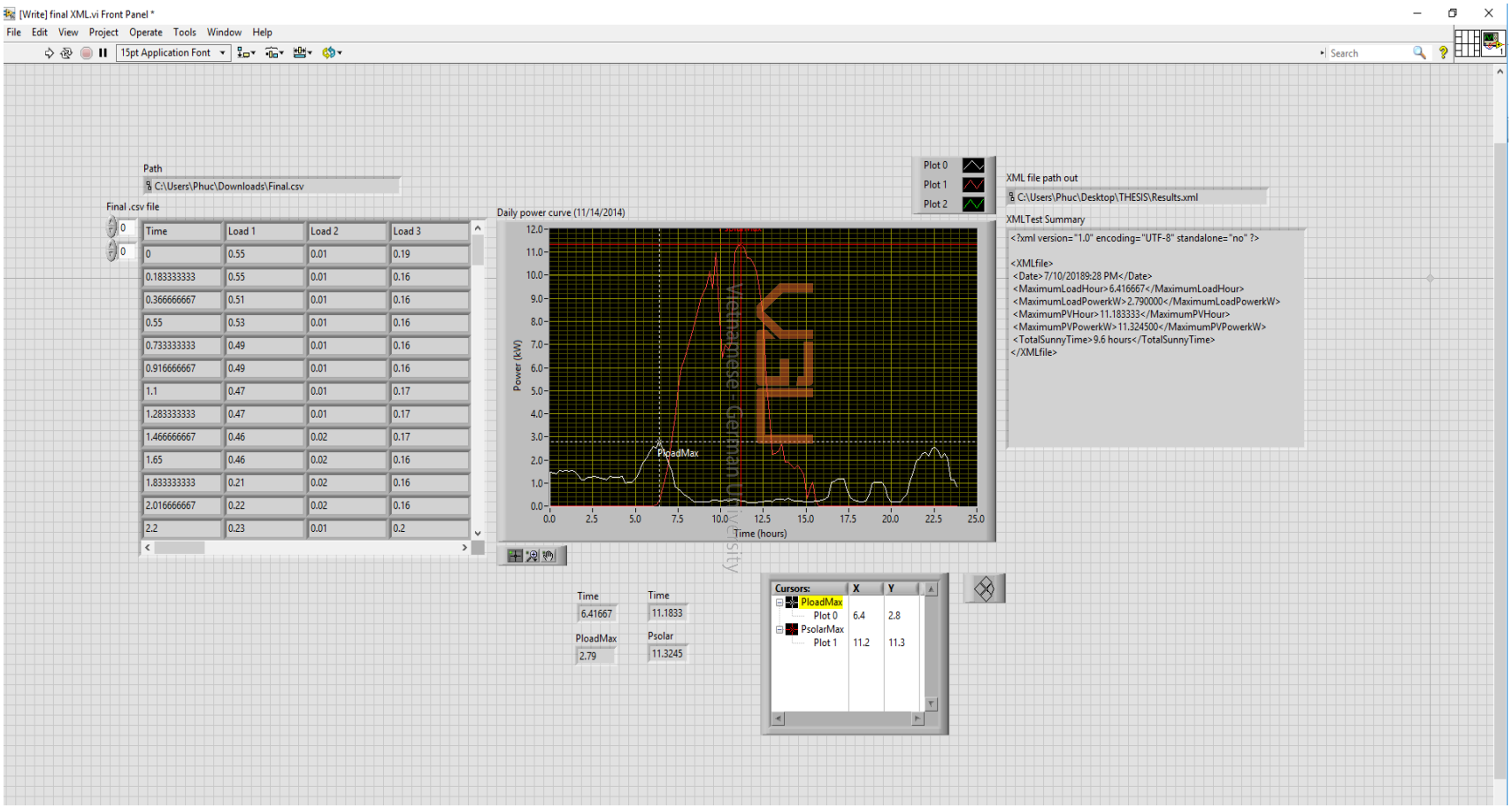


Figure: The final user interface (Front Panel) of the Extract .xml module

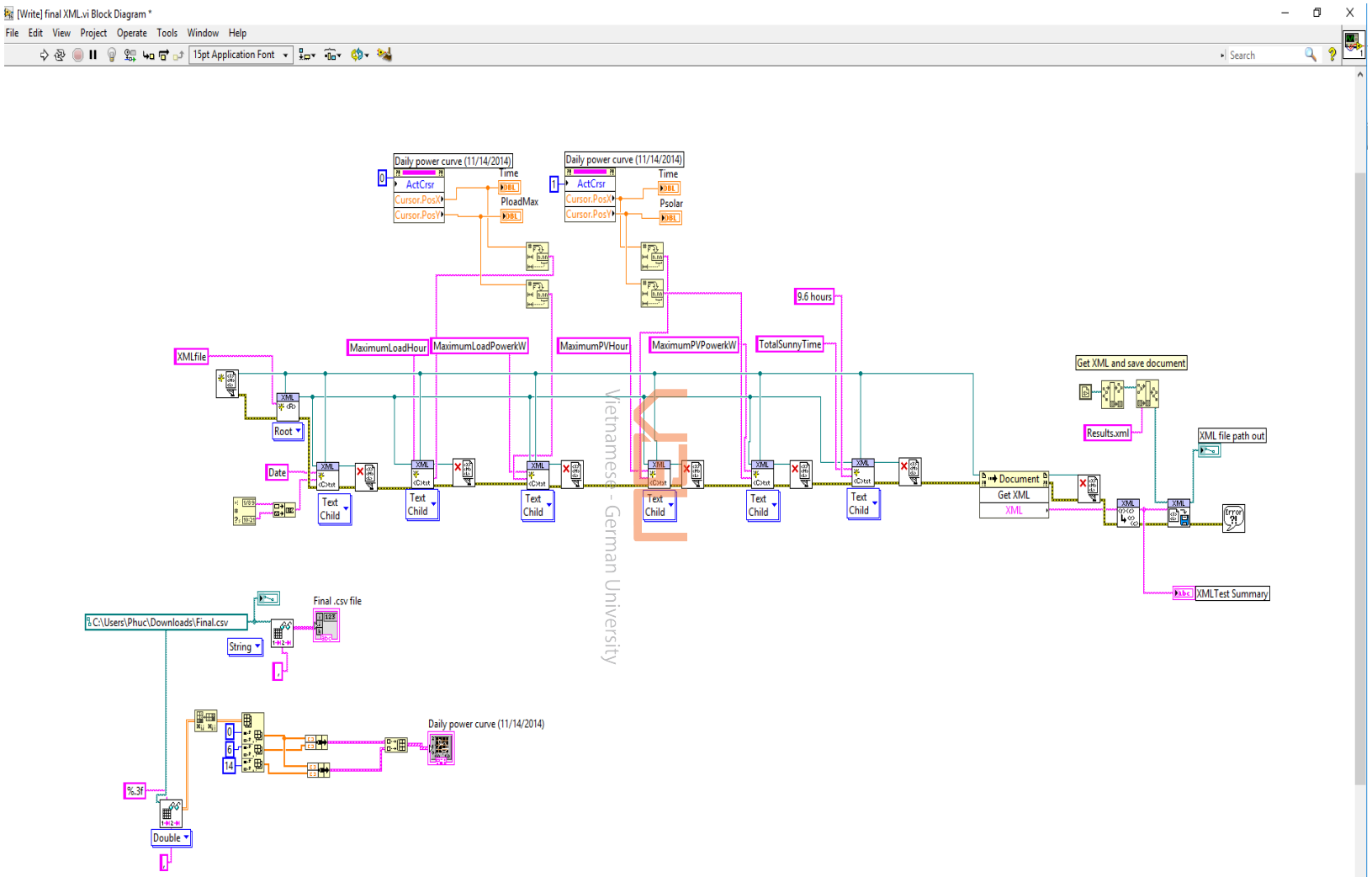


Figure: The programming code (Block Diagram) of Extract .xml module

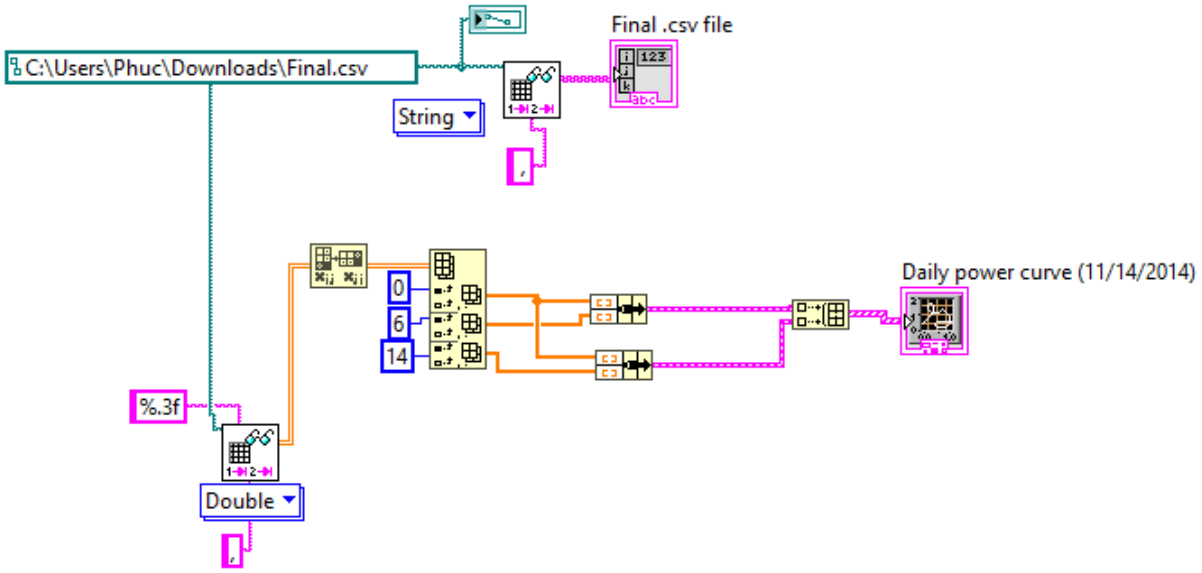


Figure 5.6 The block diagram of inputting .csv file and drawing XY-graphs

First of all, this block diagram input the Final.csv file by the Read Delimited Spreadsheet.vi and convert it to a 2D array. Transpose 2D Array block is used to rearrange each column of 2D array into 1D array rows. These 1D array rows with indexing can be used to draw XY-graphs.

Block Name	Main Function
Read Delimited Spreadsheet.vi format (%.3f) file path (dialog if empty) number of rows (all:-1) error in (no error) transpose? (F) delimiter (\t) 1 2 3 4 5 6	Read a specified number of lines or rows from a numeric text file and convert the data to a 2D, double-precision array of numbers, strings, integers.
Transpose 2D Array 2D array → transposed array	Rearranges the elements of 2D array such that 2D array[i,j] becomes transposed array[j,i]
Index Array n-dimension array index 0 index n-1 → element or subarray	Returns the elements or subarray of n-dimension array at index.
Bundle cluster element 0 element 1 ... element n-1 → output cluster	Assembles a cluster from individual elements.

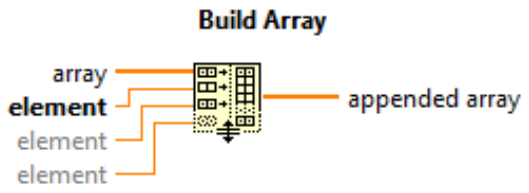
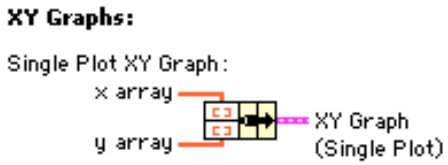
	Concatenates multiple arrays or appends elements to an n-dimensional array.
	Plot an XY-graph with x and y 1D array as inputs.

Table 18 Blocks used in the draw XY-graphs diagram

In this example, the total power of PV system and the total power of loads are drew over 24 hours. The result is shown in the Front Panel below:

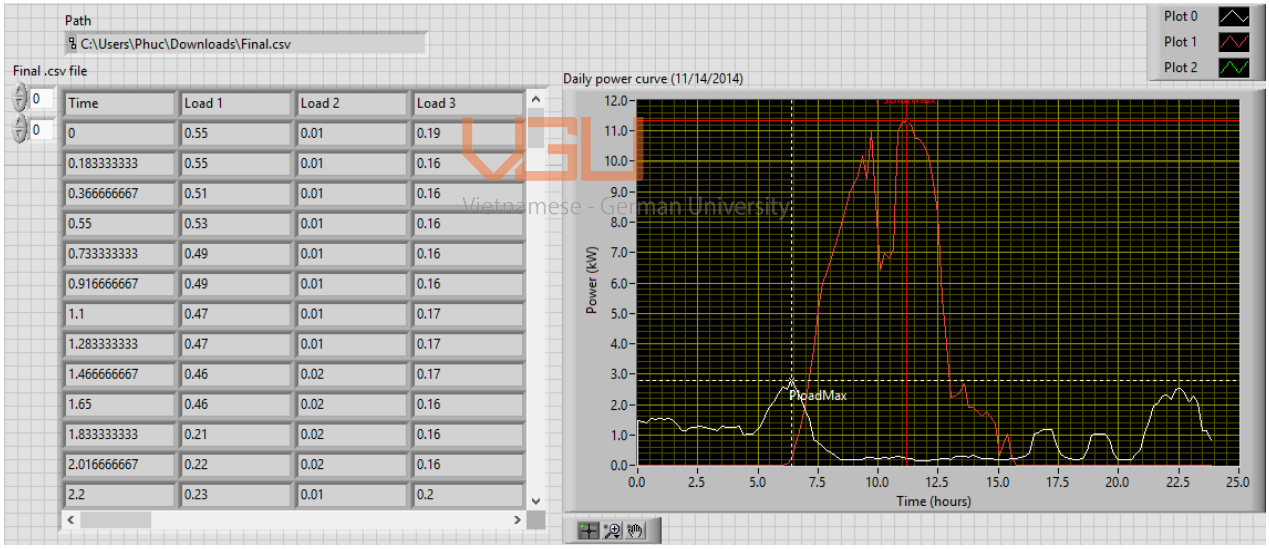


Figure 5.7 The Front Panel of the Extract .xml file module

Next, the cursor module is programmed to extract values from the graph. The maximum PV power generated and the maximum load power is concerned in this case.

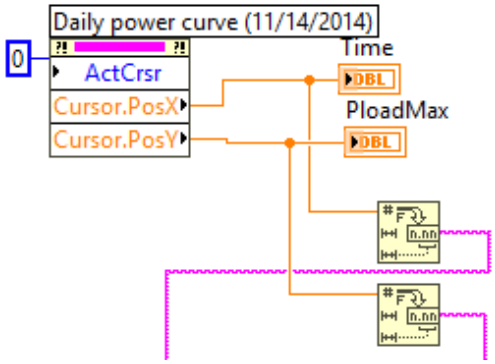


Figure 5.8 This block diagram is used to extract cursors values

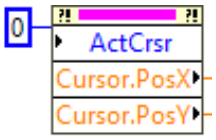
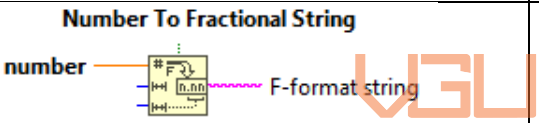
Block Name	Main Function
	Property Active Cursor. Use this property to get and set the active cursor and set properties and methods on that cursor.
	Convers number to an F-format (fractional notation), floating-point string.

Table 19 Blocks used in the extract cursors values module

In the Cursors table, X is the time value, and Y is the power value. These values is displayed on the left side of Figure 5.9.

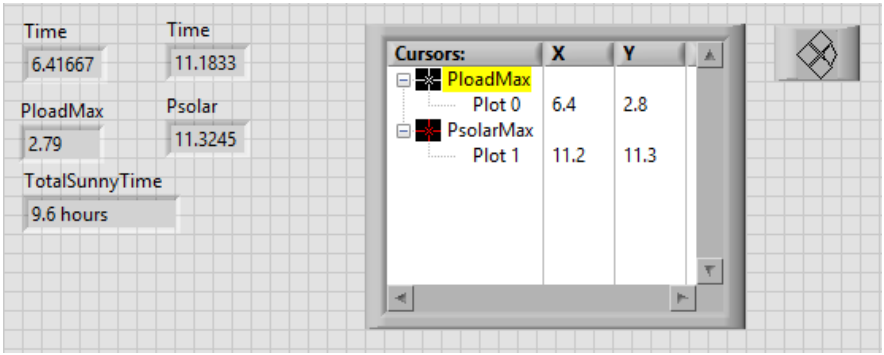


Figure 5.9 this Front Panel illustrates cursors and real values

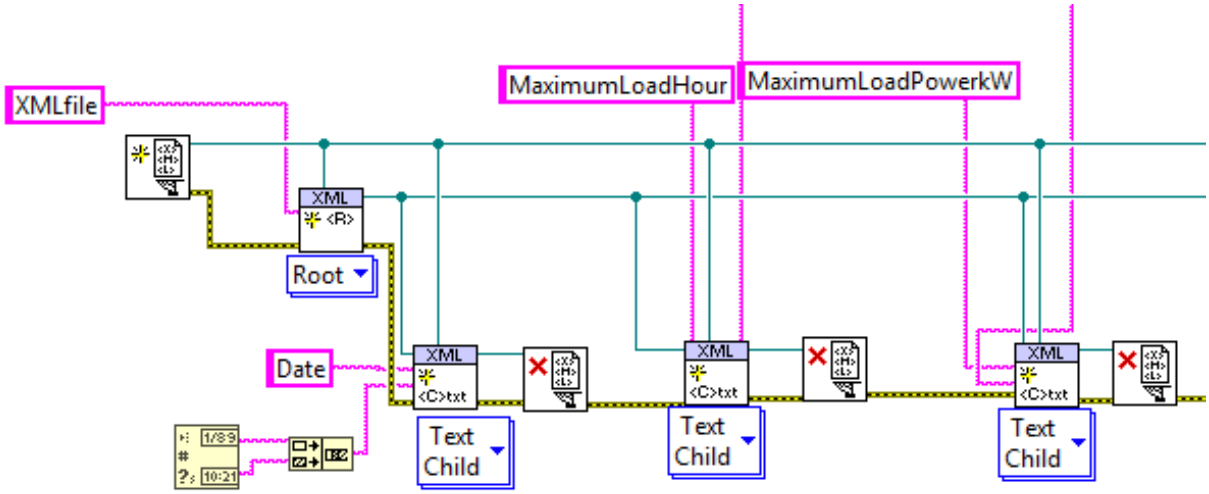


Figure 5.10 Flattens .XML block diagram

This block diagram flattens data to a .xml file. When converting LabVIEW data to XML format, data such as values, names, and type of data can be easily identified by tags.

The following block diagram builds a "pretty print" string contains text child data of the file's created date, maximum load hour/power, maximum PV hour/power, and the total sunny time in a day. The programming flow starts from left to right.

A DOM document.vi must be defined for any new XML parser session. DOM document out node connects to root, child and text child. Root tags are then created with their Child or Text Child. Multiple root tags may exist in the document. Close.vi is required after each Create Tag elements.

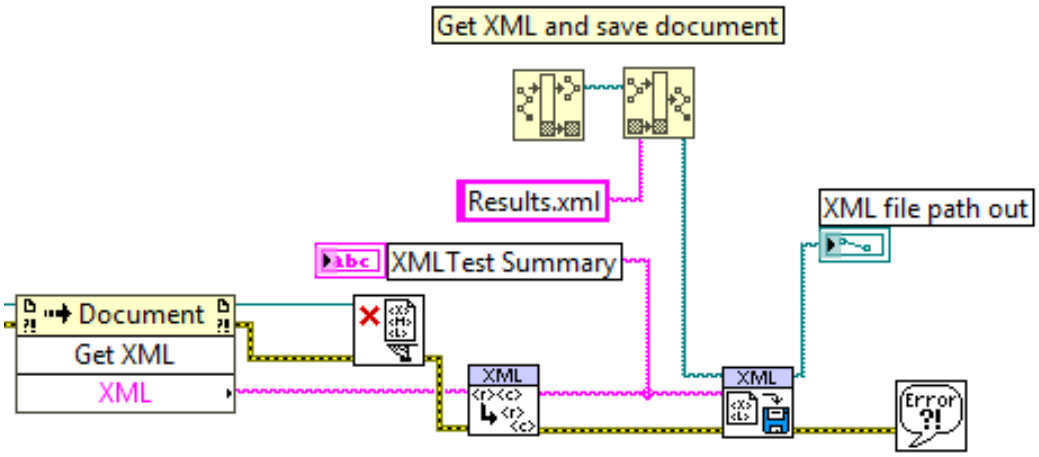


Figure 5.11 This block diagram extracts the final XML file and output a path

This block diagram combines all data tags to a single .xml format file. Pretty Print XML.vi is used to separated tags and data in a human-readable format. The "pretty print" XML document is saved, and a file path is created.

The picture below shows the final .xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
- <XMLfile>
    <Date>7/11/201810:37 AM</Date>
    <MaximumLoadHour>6.416667</MaximumLoadHour>
    <MaximumLoadPowerkW>2.790000</MaximumLoadPowerkW>
    <MaximumPVHour>11.183333</MaximumPVHour>
    <MaximumPVPowerkW>11.324500</MaximumPVPowerkW>
    <TotalSunnyTime>9.6 hours</TotalSunnyTime>
  </XMLfile>
```

Figure 5.12 The final XML file

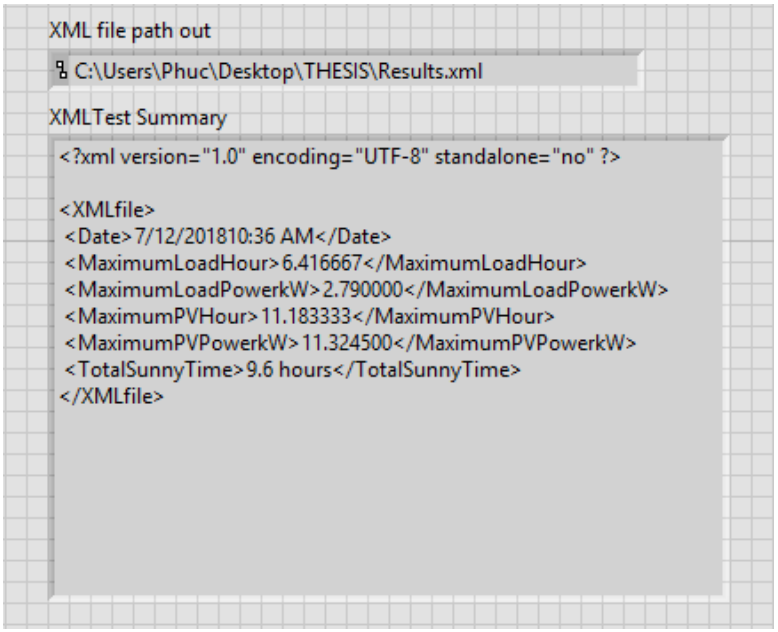


Figure 5.13 The final XML file shown in the Front Panel

Block Name	Main Function
NI_XML.lvlib:New.vi DOM document out error in (no error) error out	Returns an XML parser session in LabVIEW. Use this VI each time you create a new XML parser session.
Simple XML.lvlib:Create Tag - Root.vi DOM document reference Tag Name error in (no error) Attr Names Attr Values Element reference error out	Starts an XML tree by creating a root tag. Multiple root tags may exist in the document, and at least one root tag must be defined. This VI cannot create a child of another tag. Attributes are optional.
Get Date/Time String date format (0) time stamp want seconds? (F) date string time string	Converts a timestamp value or a numeric value to a date and time string in the time zone configured for the computer.
Concatenate Strings string 0 string 1 ... string n-1 concatenated string	Concatenates input strings and 1D arrays of strings into a single output string. For array inputs, this function concatenates each element of the array.

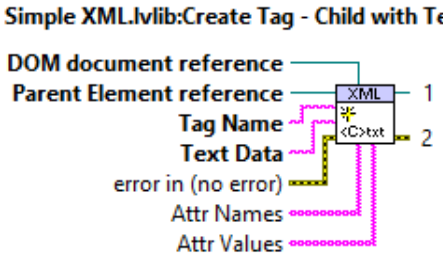
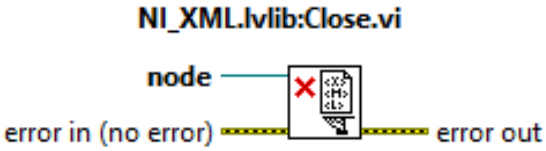
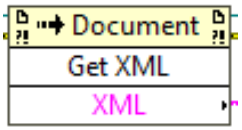
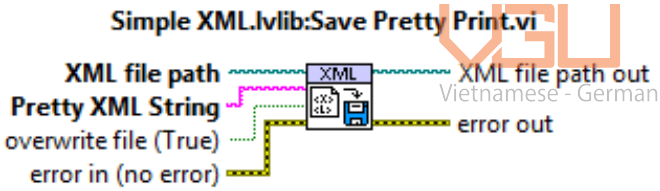
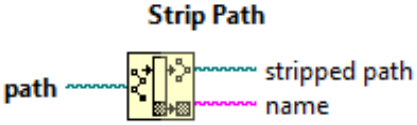
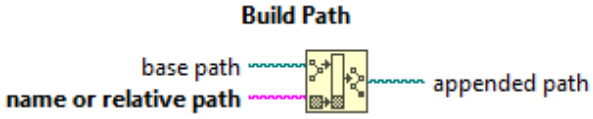
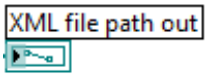
	Adds a branch to an XML tree. This tag contains text data.
	Close a reference for all XML parser classes.
	Returns a string that contains the XML for the node.
	Builds a “pretty print” string from the XML document string. Use this VI to separate tags and data in a human-readable format.
	Save an XML document that has been formatted using the “Convert to Pretty Print” VI.
	Saves an XML document that has been formatted using the “Convert to Pretty Print” VI.
	Create a new path by appending a name or a relative path to an existing path.
	Contains the path where the VI saved the file.

Table 20 Blocks used in Flatten xml file module

5.3.2. Transmitting data from TCP Server to TCP Client Module

This module uses the TCP/IP functions located on the Function→Communication→TCP communication in LabVIEW. The process involves opening the connection, reading and writing the information, and closing the connection.

With TCP/IP connections, a computer can function either as the client or the server. The following block diagram represents a Client application that initiates a connection to a remote server with TCP Open Connection. The server, or daemon, listens for remote connections and responds appropriately.

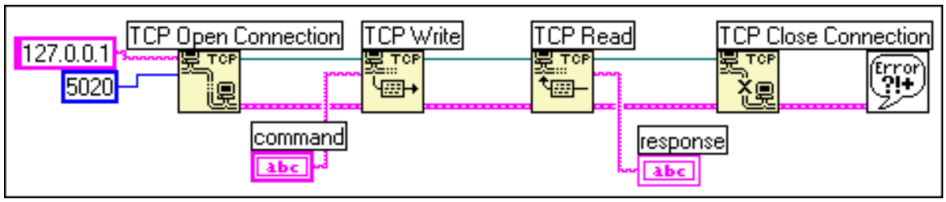


Figure 5.14 An example of a TCP Client function in LabVIEW

Because anyone can initiate a connection to a server, server should access control. The following block diagram shows how the server uses the remote address output value of the TCP Listen VI to determine whether a remote client has permission to access the server.

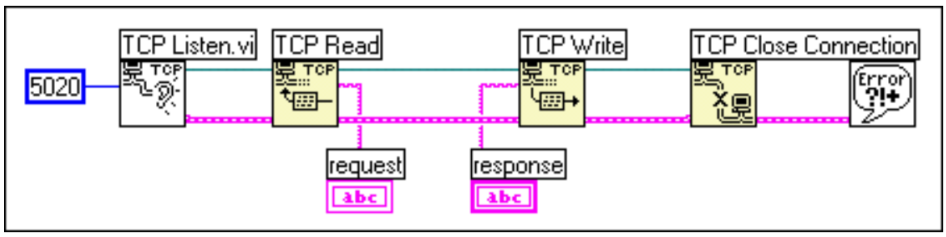


Figure 5.15 An example of a TCP Server function in LabVIEW

Here are steps to implement and execute code:

- First of all, open TCP-Server.vi and TCP-Client.vi, respectively.
- In the TCP-Server.vi, select the server IP, port and timeout values. Timeout is the amount of time the client should wait for the server to respond during the initialization stage. The

default values are localhost, 6340 and 30000ms, correspondingly. In TCP Server Loop, a .XML reader is designed and connected in order to transfer .xml file.

- In the TCP-Client.vi, select the address, port, bytes to read and timeout values. The default values are localhost, 6340 and 1000ms. In TCP Client Loop, an unflatten .xml module is developed to read .xml string. This module reads the .xml string and links to the Front Panel to show for users.

- After finishing transmit the data, choose Close from TCP-Client.vi and TCP-Server.vi to stop the module.

5.3.2.1. TCP Server Module

Let's discuss the design of TCP-Server.vi. The TCP-Server.vi contains 3 main parts: TCP Listen VI, TCP Server Loop and TCP Close Connection:

- The TCP Listen VI generates a connection reference whenever the client connects on the specified port. The client has 30 seconds to connect before the server times out.

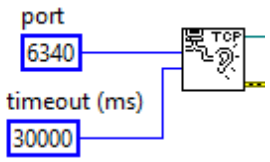


Figure 5.16 The port and timeout value of the TCP Listen VI

- TCP Server Loop has 2 important parts: TCP Write and TCP Read. The first TCP Write function specifies the amount of data being sent, and the second TCP Write functions sends the data. The TCP Read function checks to see if the client has written any data. If it has, then the client is telling the server to stop executing. Note that the timeout on TCP Read function is 0, as the client will not always be sending data to the server. Since TCP Read will return an error on a timeout, the error case structure is used to ignore the timeout error within the loop.

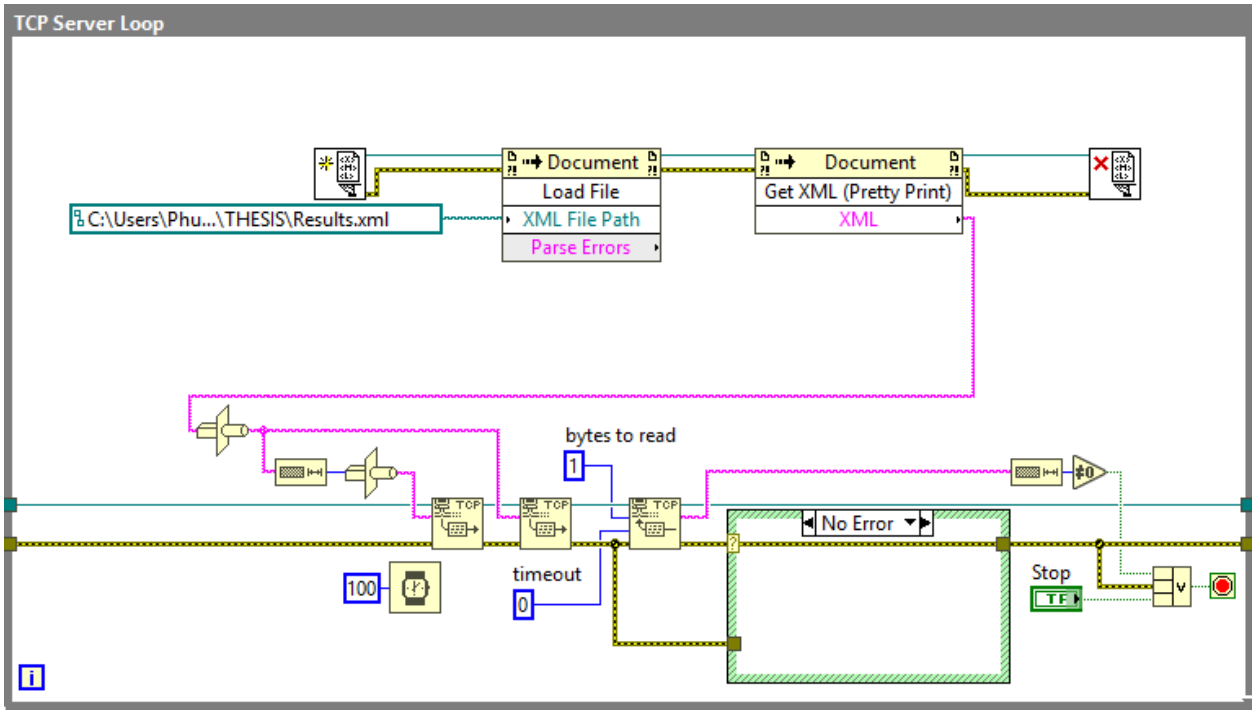


Figure 5.17 TCP Server Loop

- This block diagram below inputs the final .xml file and read this file with Get XML (Pretty Print).vi. String type of data is executed until full package is transmitted.

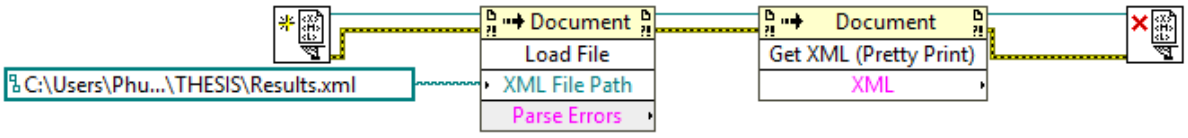


Figure 5.18 This block diagram inputs and reads XML file

- TCP Close Connection closes the connection when the user clicks Stop or an error occurs. Certain error codes can occur if the client VI closes the connection. In these cases, ignore the potential errors that can occur, and instead pop up a dialog indicating that the client closed the connection.

- Full displayed of TCP-Server.vi can be found at Appendix.

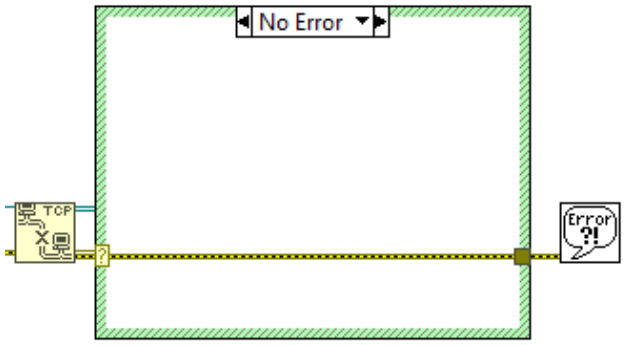


Figure 5.19 TCP Server Close Connection module

5.3.2.2. TCP Client Module

Let's discuss about the TCP-Client.vi. The TCP-Client.vi contains 3 main parts: TCP Open Connection, TCP Client Loop and TCP Close Connection.

- First, TCP connection is opened with the TCP Open Connection function. Note that the port must match the one specified in TCP-Server.vi.

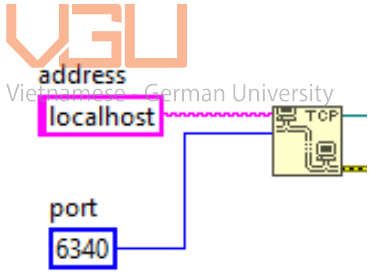


Figure 5.20 TCP Client address and port

- TCP Client Loop includes TCP Read and TCP Write function. There are two TCP Reads to read the data on the specified connection. The first TCP Read function acquires the size of the data, and the second TCP Read function reads the data itself if the size of the data (specified by the server) is greater than zero. TCP Write is used to send a single character to the server to indicate that the client has stopped.

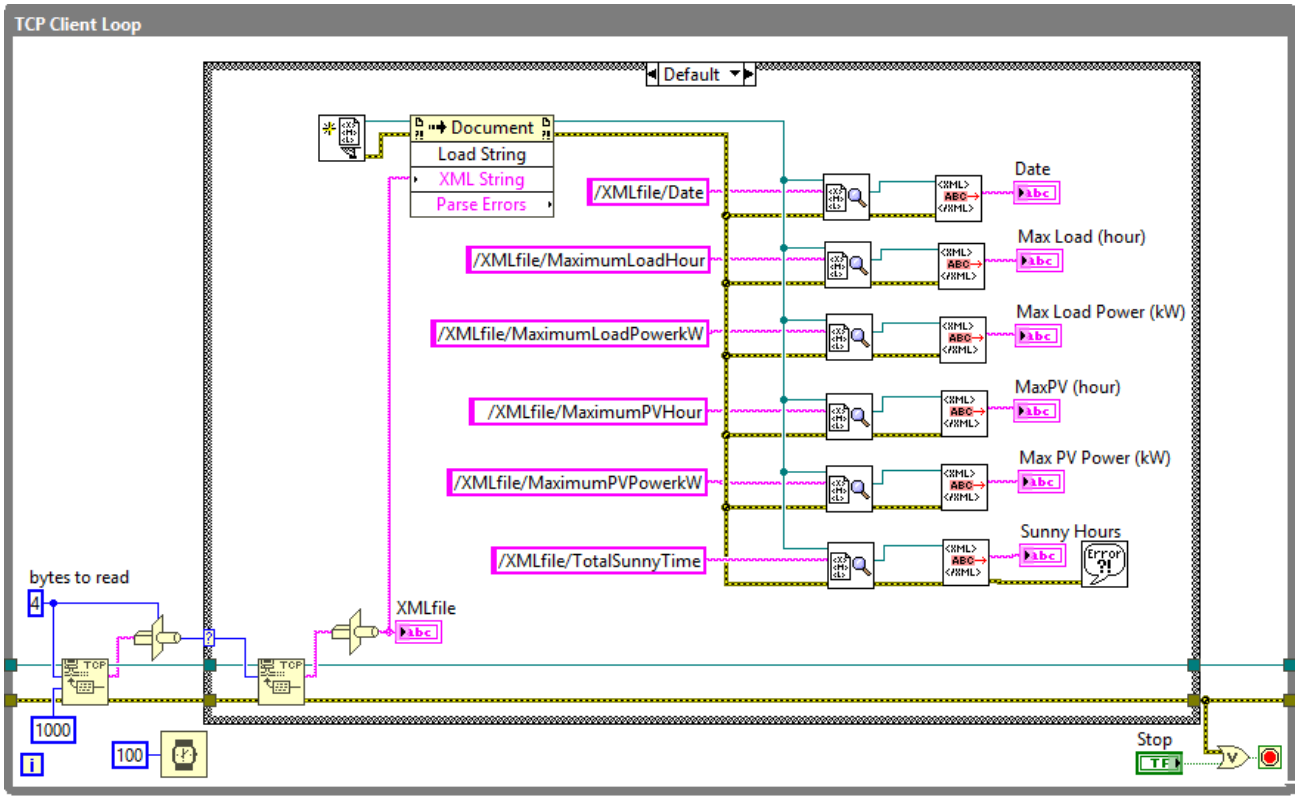


Figure 5.21 TCP Client Loop

- TCP Close Connection closes the connection when the user clicks Stop or an error occurs. Certain error codes can occur if the server VI closes the connection. In these cases, ignore the potential errors that can occur, and instead pop up a dialog indicating that the server closed the connection.

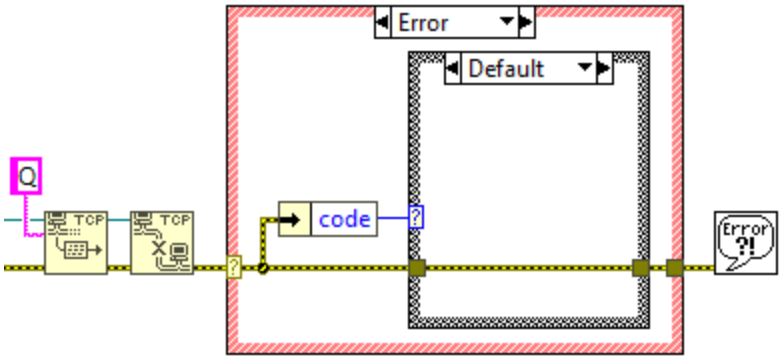


Figure 5.22 TCP Client Close Connection Module

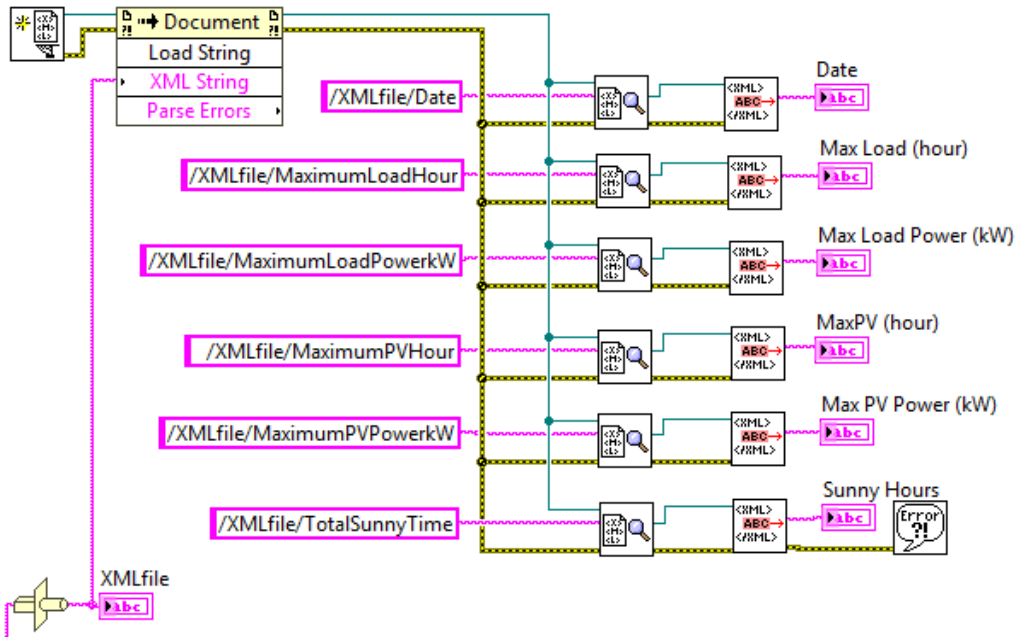


Figure 5.23 This block diagram loads XML strings

- The block diagram above loads XML strings, reads and shows data on the Front Panel. Nodes must match the XPath expression.

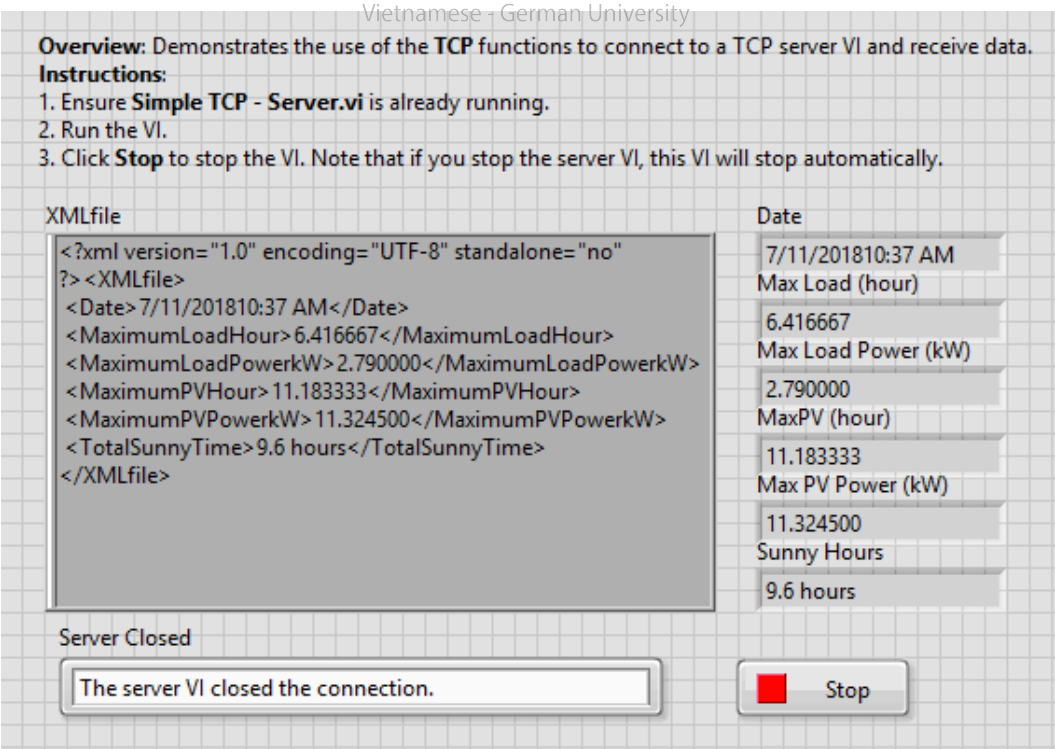


Figure 5.24 This front panel shows XML data for clients

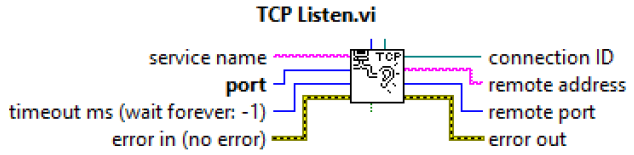
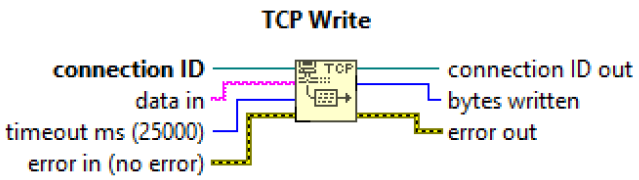
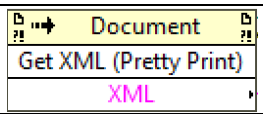
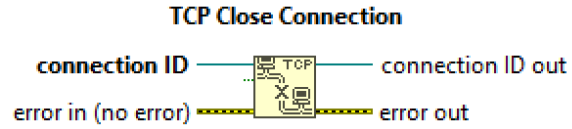
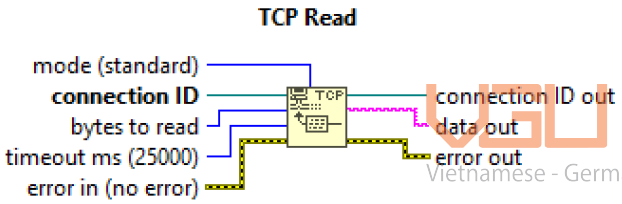
Block Name	Main Function
	Creates a listener and waits for an accepted TCP network connection at the specified port.
	Writes data to a TCP network connection.
	Get XML (Pretty Print) for a node to a string.
	Close a TCP network connection.
	Reads a number of bytes from a TCP network connection, returning the results in data out.

Table 21 Blocks used in TCP Server and TCP Client module

6. Conclusion

6.1. Achievements

The thesis has achieved main results as the following:

- The design of a small-sized off-grid PV generation system has been developed successfully. The system is operated under sunlight at VGU campus.
- The data is extracted from solar devices & meters. The information is analyzed by JAVA programming language to automatically calculate the collected values with a sample size of every 5 minutes.
- Flatten/unflatten the data to .xml format using LabVIEW program, and transmit the .xml file through TCP/IP protocol.

6.2. Suggestions

There are some subjects of the thesis that could be improved and optimized:

- The data has to be processed manually in Microsoft Excel in order to get a complete table of data. JAVA codes can be updated to reduce the amount of work.
- The .xml format file is not so scalable. It contains limited data. The LabVIEW system has to be developed in a more flexible way to overcome this problem.
- The TCP client.vi can only receive the .xml format. A new module should be designed to send files from TCP client to TCP server, such as .xls files, and .csv files.

6.3. Conclusion

This project has presented a systematic approach has been presented regarding sizing and designing a real-time off-grid PV generation system. A small-scaled PV system (approx. 300W) has been built for data analyzing and further studying on the EICF. The system includes 4 main parts: solar panels, a solar charge controller (MPPT), a DC/AC inverter, and a battery storage.

Guidelines for selection of components of the PV system with function of maximum power point tracking (MPPT) are provided.

JAVA program is used to analyzed the data getting from PV system meters & MPPT solar charge controller. The program gets date of recorded data such as day, and specific time (hour, minute, second). The median method is applied for a day of recorded value. JAVA extract a .csv file which include selected values in every 5 minutes.

LabVIEW program is used as a main design platform and user interface. There are three important modules to be designed: extract .xml file, TCP/IP server and TCP/IP client. Extract .xml file module flattens value to .xml format that carries needed data. TCP/IP server module sends .xml file via interconnect network devices. Last but not least, TCP/IP client module receives .xml file, and unflattens it for normal user to read.

Appendices

JAVA code

```
import java.io.*;
import java.util.*;
import java.util.regex.*;
import java.lang.Math;

public class minProcess {
    public static void main(String[] args) {
        String fileName = args[0];
        String outFile = args[1];

        ArrayList<Double> times = new ArrayList<Double>();
        ArrayList<Double> values = new ArrayList<Double>();

        try {
            FileReader fileReader = new FileReader(fileName);
            BufferedReader reader = new BufferedReader(fileReader);

            String line = null;
            int lineCount = 0;

            Pattern pattern = Pattern.compile("\\d+");

            while ((line = reader.readLine()) != null) {
                // System.out.println(line);
                if (lineCount > 0) { // skips first line
                    String[] tmp = line.split(",");
                    if (tmp.length > 1) {
                        Matcher m = pattern.matcher(tmp[1]);
                        ArrayList<String> matches = new ArrayList<String>();
                        while (m.find()) {
                            matches.add(m.group());
                        }
                        if (matches.size() > 1) {
                            Double hour = Double.parseDouble(matches.get(0)) +
(Double.parseDouble(matches.get(1)) / 60);
                            // System.out.println(matches.get(0) + ":" +
matches.get(1) + " - " + Double.toString(hour));
                            times.add(hour);
                            values.add(Double.parseDouble(tmp[3]));
                        }
                    }
                }
                lineCount++;
            }

            reader.close();

            ArrayList<Double> hours = new ArrayList<Double>();
            ArrayList<Double> medians = new ArrayList<Double>();
            double lastHour = times.get(0);
            double total = 0;
```

```
            ArrayList<Double> items = new ArrayList<Double>();
            for (int i = 0; i < times.size(); i++) {
                if ((Math.abs(times.get(i) - lastHour) > 0.167) || i ==
times.size() - 1) {
                    hours.add(lastHour);

                    double median = 0;
                    Collections.sort(items);
                    if (items.size() > 0 && items.size() % 2 == 0) {
                        int idx = items.size() / 2;
                        median = (items.get(idx - 1) + items.get(idx)) / 2;
                    }
                    else {
                        median = items.get(Math.floorDiv(items.size(), 2));
                    }

                    medians.add(median);
                    items.clear();

                    // start new count
                    items.add(values.get(i));
                    total = values.get(i);
                    lastHour = times.get(i);
                }
                else {
                    items.add(values.get(i));
                    total += values.get(i);
                    // lastHour = times.get(i);
                }
            }

            PrintWriter writer = new PrintWriter(outFile, "UTF-8");
            writer.println(String.join(",", "Time", "Median"));
            for (int i = 0; i < hours.size(); i++) {
                writer.println(String.join(",",
                        Double.toString(hours.get(i)),
                        Double.toString(medians.get(i))
                ));
            }
            writer.close();
        }

        catch (FileNotFoundException ex) {
            ex.printStackTrace();
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }

        System.exit(0);
    }
}
```

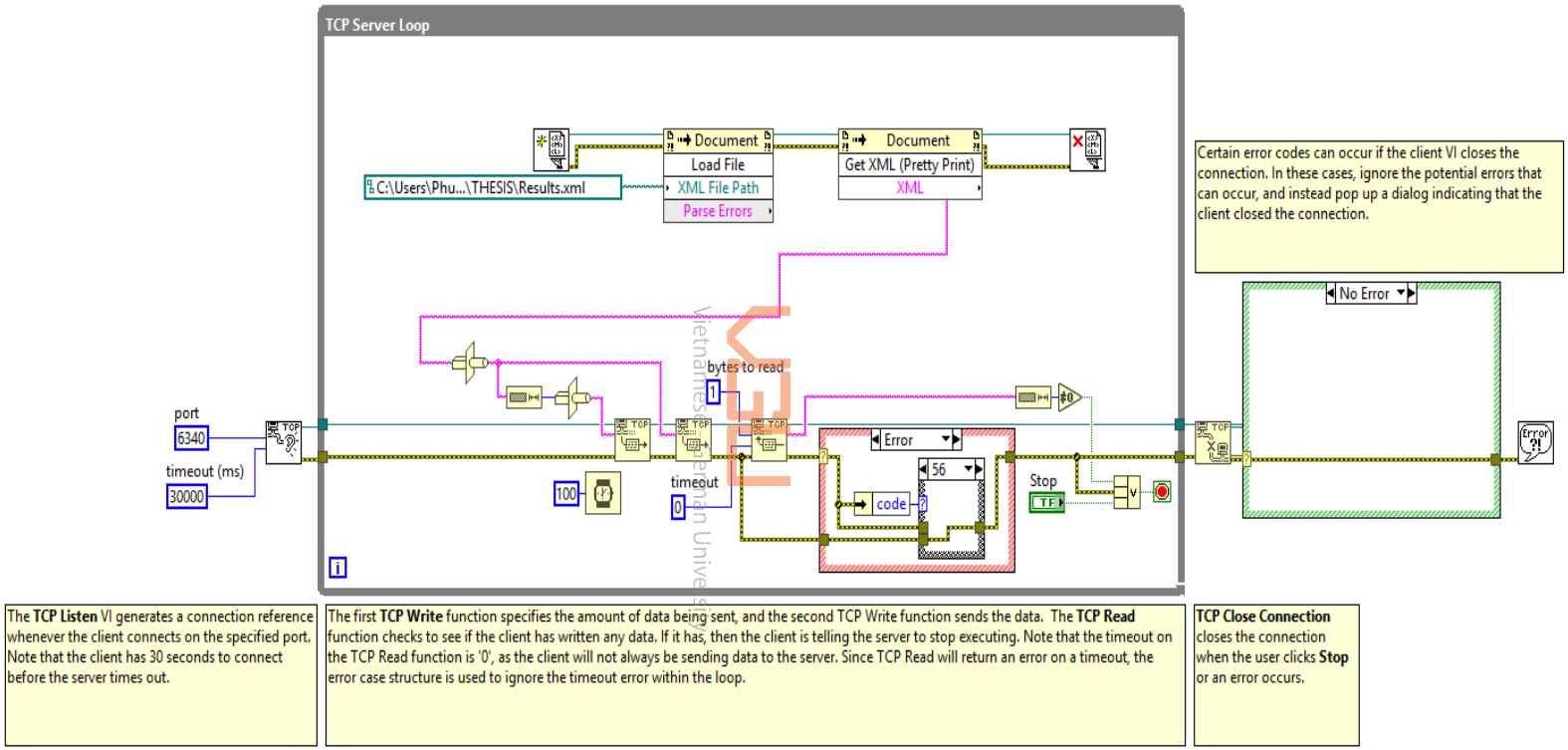


Figure: the programming code (Block Diagram) of TCP Server module

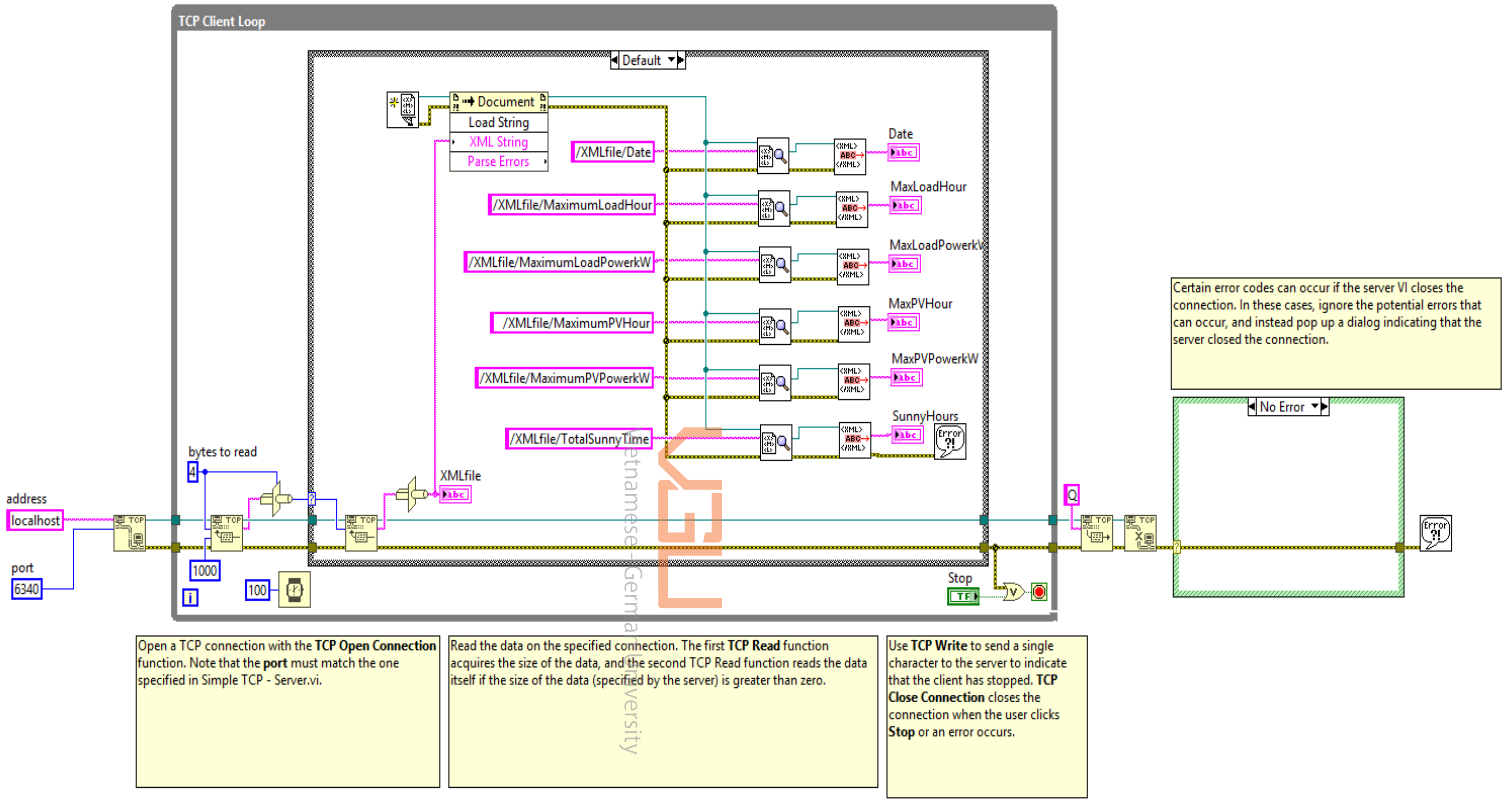


Figure: the programming code (Block Diagram) of TCP Client module

Reference

[1] Solanki S. Chetan. (2012) Solar Photovoltaics: Fundamentals, Technologies and Applications, New Delhi, PHI.

[2] Retrieved from http://www.enerdrive.com.au/mppt-vs-pwm-solar-controllers/

[3] Ned Mohan, Tore M. Undeland, William P. Robbins. (2002) Power electronics: Converters, Applications, and Design. John Wiley & Sons, Inc.

[4] Muhammad H. Rashid. (2006) Power Electronics Handbook: Devices, Circuits, and Applications. Elsevier.

[5] Andy Walker. (2013) Solar Energy: Technologies and the Project Delivery Process for Buildings. Wiley.

[6] Klaus Jäger, Olindo Isabella. (2014) Solar Energy: Fundamentals, Technologies, and System. Delft University of Technology.

[7] Kharagpur. Instructor. 3-phase Voltage Source Inverter with Square Wave Output. Version 2 EE IIT, Kharagpur. Retrieve from: http://www.nptel.ac.in/

[8] Hong, Ying-yi, PhD. Chapter 2, Power Grids part 2. Chung Yuan Christian University.

[9] Retrieved from: http://searchnetworking.techtarget.com/definition/TCP-IP

[10] Duong, Minh Bui, PhD. Study on XML, HTTP, TCP/IP, JAVA, LabVIEW for EICF (Energy Information Communication Framwork).