Cape Peninsula University of Technology

FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT

DEPARTMENT OF ELECTRICAL, ELECTRONICS AND COMPUTER ENGINEERING

HARDWARE SIMULATOR FOR MICROGRID SYSTEMS

A thesis submitted in part-fulfillment of the degree: Master of Engineering in Energy.

NAME	**: Mr E.P MULUH**
STUDENT ID	**: 211050490**
PROGRAM OF STUDY	**: MEng (Energy)**
SUPERVISOR	**: Prof Mohamed T. E. Kahn**

Abstract

The research thesis; Hardware simulator for Microgrid systems, takes a brief look at the general definition of a Microgrid system and the constituent of a Microgrid system. Special attention is paid to the solar PV as a stand-alone Microgrid system. The characteristics of solar panels are studied.

An interconnected software and an electronics hardware system that mimics a solar PV Microgrid system is then designed, simulated and built to operate within the solar PV voltage and current rages of 48V / 5A respectively.

The software component is a LabVIEW graphical user interface (GUI) that allows one to enter the data sheet parameters of a specific solar panel as well as any assumed environmental conditions such as temperature and insolation. The block code then continuously calculates and displays the power output info expected of the solar panel.

The built electronics hardware is a switch mode programmable buck converter (DC –DC power supply). The buck converter is powered by a multiple-output adjustable AC to DC power supply.

The electronics hardware’s current output limit is controlled from and by the GUI through a national instrument data acquisition device; the NI USB 6009. The hardware’s power output across a load is displayed using an LCD, Watt’s up power meter and also on the GUI via the data acquisition device still.

.

Acknowledgement

I am grateful to the almighty for his infinite blessing upon my life, must especially concerning my education at the Cape Peninsula University of Technology. These blessings were, and are still being manifested via the following ways:

- A family who gave me their moral and financial support, and are always willing to do more.
- CPUT post graduate bursary and research funding, who granted me a University bursary and Research Fund which made my research frictionless.
- The Department of Electrical, Electronics and Computer Engineering (DEECE) who equipped the laboratories and gave me 24 /7 accessibility.
- A supervisor, who was at all times available, listened kindly to my complaints and gave me the best advices.

Table of Content

List of Figures

List of figures

List of equations

Glossary

Abbreviation / Acronym	**Meaning**
AC	Alternating current
ADC	analogue to digital conversion
CPU	Central Processing Unit
DC	Direct Current
DER	Distributed Energy Resources
DG	Distributed Generation
GUI	Graphic User Interface
IC	Integrated Circuit
LCD	Liquid Crystal Display
MPP	Maximum Power Point
NI-daq	National instrument data acquisition
OS	Operating System
PV	Photovoltaic
PWM	Pulse width modulation
SAPV	Stand -alone photovoltaic
USB	Universal Serial Bus

1. Introduction

1.1. Background

A Microgrid is a minor power grid which is constituted of renewable power sources or distributed generation (DG) units, power conditioning & control interface circuit, energy storage, and loads [1]. These components that a Microgrid is made up of are referred to as the hardware of the Microgrid system. There are different types of Microgrid system, few of which are solar PV’s (photovoltaic), wind turbine etc. The primary energy resource for the solar PV Microgrid system is the sun. The irradiation varies from sunrise to sunset, summer to winter and as well as by latitude. The electrical output from the solar panel is as a characteristic of the particular panel, angle of inclination and irradiation. The primary energy resource for the wind turbine Microgrid system is the wind. The hardware simulator will determine the possible output power based on the Microgrid characteristics and environmental factors at its location to present a graphical result.

A Microgrid is important in electricity supply in that based on its ability to operate grid-connected as well as Islanded modes it can thus be separated from the utility during a utility disturbance, with little or no interruption to the loads powered directly by the Microgrid. In grid-connected mode, it reliefs the utility of overloading during peak load hours thereby avoiding utility grid failure. This renders energy service reliability and a decrease in the impact of load shedding [2]. A Microgrid leads to a reduction in fusil fuel (coal) usage for electricity generation thereby reducing greenhouse gas emissions which is environmentally beneficial. A Microgrid mitigates the cost of energy to its user by providing some or all of its electricity requirement. Smart switching of a Microgrid between grid-connected and Islanded modes allows for energy supply efficiency [3].

1.2. Problem statement

The problem is that projects’ implementations, especially Microgrid systems require intensive planning. Planning will be financial and time costly if a physical prototype has to be implemented for testing, considering all possible factors before the actual project is adopted or improved. This inspires the need for a low cost solar energy hardware simulator in the renewable energy research laboratory whereby virtual prototypes could be tested without necessarily setting up the actual system in an appropriate scenario such as experimenting a specific solar panel or wind turbine configuration, connected to a load over a period of time and at different locations. A Hardware-simulator is therefore required to be stationed in the laboratory allowing a researcher to record similar or same results while simulating a solar panel inside a laboratory and even during night time, thereby rendering much economic and time flexibility in future project planning.

1.3. Aim and objectives

The aim of this project is to design a Microgrid hardware simulator that would be able to simulate the maximum current output and thus the maximum power point of different types of solar panels' Microgrid configuration based on the number of series and/or parallel panels and the datasheet parameters entered. The simulator would then yield results of the configuration numerically, graphically and control a programmable switch mode direct-current (DC) power supply to supply a load following the characteristics (software results) of the simulated PV panel.

The objectives will be to:

1) Comprehend the factors and patterns which affect the behaviour of solar panel Microgrids
2) Design a solar PV graphical user interface based on a solar PV mathematical model.
3) Design and built a programmable switch mode DC power supply directly linked to the computer to mimic the software simulator's results.
4) Set up a desk-top hardware simulator work station in the renewable energy research laboratory.
5) Test and verify communication between the computer bases software and the programmable switch mode power supply as well as the efficiency of the switch mode power supply.

Methodologically, the first objective would be reached by downloading and reading articles and books from academic sites (IEEE explore, Publish or perish, google scholar, ScienceDirect) related to solar PV technology and their uses as in islanded PV Microgrid systems. The second would be reached by acquaintance with and use graphical design and modelling software such as Matlab and LabView. The third objective would be reached by downloading and ready articles on switch mode power converters and then design & simulate a chosen type using a power electronics software such as Powersim (PSIM). The optimised design would be implemented using power electronics components, a Multimeter and digital Oscilloscope. The fourth objective will be achieved by using a USB data acquisition device to connect the switch mode power supply to a laboratory computer CPU containing the graphical user interface. Finally, the set up will be tested and verified by entering the datasheet specifications of different solar panel and applying an adjustable load (Potentiometer or resistor bank) to the switch mode power supply. The load would be varied stepwise and the power flow is measured and recorded.

1.4. Thesis structure

Chapter-one: Here the background of the research topic was summarised and the problem statement was clearly stated. The aim and objectives of the project were listed.

Chapter two: It contains the literature review of the research topic. The PV system as a Microgrid, the operation/characteristics of solar panels, the disadvantages and disadvantages of stand-alone solar PV's as a Microgrid were discussed. The design of simulators and the intelligence behind Microgrid systems simulators, such as HOMER Energy were also studied.

Chapter three: This talks about the methodology adopted in the design of the hardware simulator. The simulator's data input, output methods, as well as the structural designs were mentioned. The simulation of the electronic structural system was done in this chapter.

Chapter four: Systems involving the intake, processing and outputting of data exhibit a logical manner by which it does these. The different control strategies of flow of data were discussed and a most convenient one for the design was adopted.

Chapter five: The steps as to how to efficiently use the desk-top simulator were out lined. These steps make up the users' manual of the simulator.

Chapter six & seven: Chapter six discusses other possible and attempted methodologies during this research thesis. It also talks about why some were discontinued. Chapter seven concludes the research. Last but not list, the appendix contains the source codes for the microprocessor devises used, build of material, and images of the project during and after completion.

2. Literature review

2.1. Microgrids

What is a Microgrid?

The universal definition of a Microgrid is yet indefinite on how small or how large one may be with respect to energy use or geographic area. [4] However, there are some unique characteristics of the definition [5]:

- Set of interconnected loads and distributed energy resources
- Can work in isolation or grid-connected if desired
- Can be linked and unlinked to the grid if desired
- Acts as a single controllable entity to the grid

For example, according to [6] a Microgrid is defined as a set of interconnected loads and distributed energy resources (DER) with definitely stated electrical boundaries that acts as a single controllable entity with respect to the grid. Also, [7] defines Microgrid as "a controllable system consisting of distributed sources (typically renewable energy sources), loads, and energy storage systems that together can work either when tied to the grid or singled out from the grid".

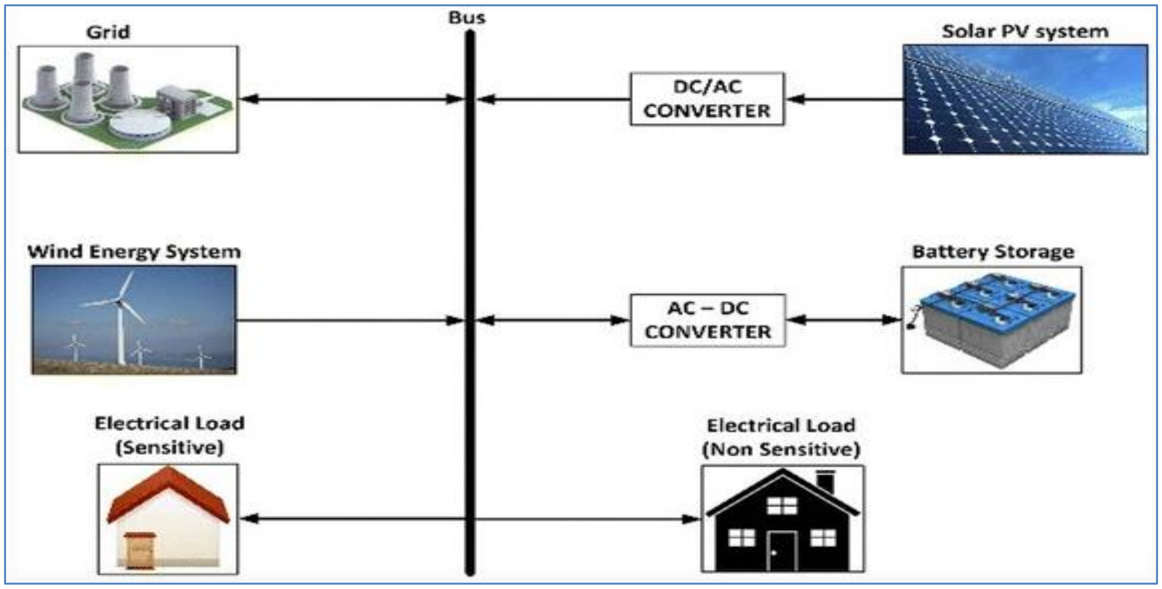


Figure 1: A typical Microgrid structure [8].

2.1.1. Solar Photovoltaic systems

A photovoltaic system is a connection of components generally such as PV panels, power converters and storage components designed to deliver usable electric power for a variety of purposes [9]. Photovoltaic systems can be generally grouped into two basic categories: stand-alone PV (SAPV) or off-grid and grid connected (on-grid) systems [6]. The SAPV systems are used in remote areas and for space applications, where the PV system (panels with batteries) only is able to assure the load demand. Other types of stand-alone systems are based on a hybrid methodology; figure (**Error! Reference source not found.**) above, which increases the efficacy and reliability of the installed system [10].

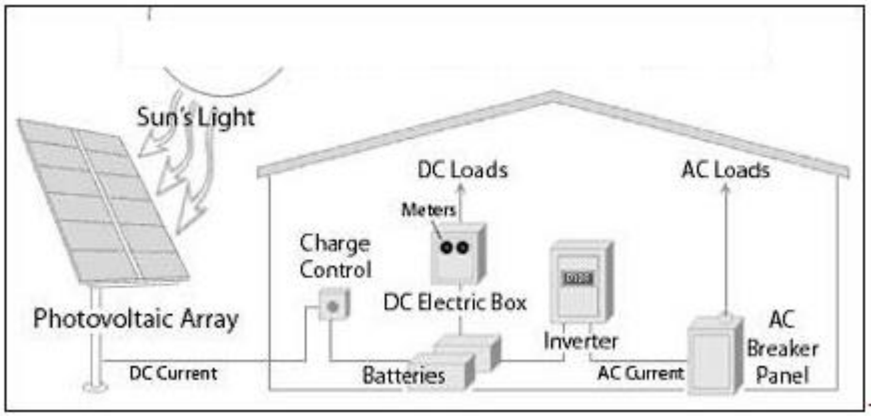


Figure 2: Stand - Alone PV systems [11]

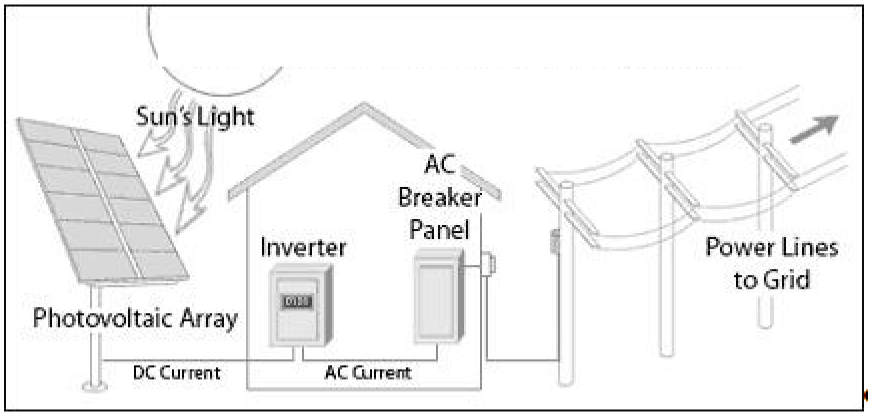


Figure 3: Grid – Tied PV System [9]

2.2. Solar panels

A solar cell, or photovoltaic cell, is an electrical device that converts the energy of light directly into electricity by an effect called photovoltaic, which is a physical and chemical occurrence. Photovoltaic cells make up a solar panel or module [10]. Solar photovoltaic (PV) panels are of different types based on the solar cell technology used by different manufacturers. There are basically two types of solar cell technology: first generation cells made up of crystalline silicon (monocrystalline, polycrystalline or multicrystalline) and second generation cells; thinfilm. These types of solar PV panels further differ not just in their temperature coefficients, spectral reactions, voltages and current values but also differ in their reactions to environmental factors such as radiation, temperature and wind speed [12].

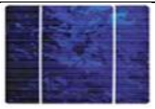
Module Technology	**Efficiency Range under standard test conditions (%)**	**Highest Reported Laboratory Efficiency (%)**
Monocrystalline silicon	16 - 19	22.9
Multicrystalline silicon 	15 - 17	18.9
Thin-film CdTe CIGS	14 - 15	17.5
Advanced silicon design (heterojunction, interdigitated back contact etc)	18 - 21	Not available
Concentrator	27 - 33	38.9

Table 1: Different type of solar cell technology and efficiency range [13]

Thin-film technology

Amorphous silicon (α-Si), Copper indium gallium selenide (CIGS) and Cadmium telluride (CdTe) are the major three thin-film solar cell technologies and there the most broadly marketed thin film solar cells. CdTe and CIGS are relatively new technologies to the α-Si. The α-Si solar cell is used largely in end-user electronics such as clocks, watches, calculators. Amorphous silicon requires less silicone compared to CdTe and

CIGS and it is also less toxic. Cadmium has proven to be detrimental to both the maker and the end-user thereby reducing a little bit its applications for commerce. The light-take-up coefficient of the thin film material is too less than that of its crystalline counterpart and they still lag behind in efficiency and reliability. [14]

Crystalline solar cell technology

Higher efficiency in the solar panels has always been desired. The conventional crystalline silicon cells are widely used due to a low production cost but it has been difficult to reach efficiency greater than 20%. The PERC (Passivated Emitter and Rear Cell) is becoming the mainstream of the next generation of cells and so the development and mass production of the PERC is rapidly in progress. The cell efficiencies of mono-crystalline PERC and multi-crystalline PERC are respectively 22.61% and 21.63%. The efficiency of SHJ (Silicon Hetero Junction) cells and IBC (Interdigitated Back Contact) cells which are highly efficient crystalline silicon cells has also been improved to a maximum efficiency of over 25%. The SHJ-IBC cell merges the merits of both SHJ and IBC and therefore has an efficiency of up to 26.33%. [15]

2.2.1. Operation of a PV cell

A PV cell operates by a phenomenon called photovoltaic (photoelectric) effect. Light is constituted of massless particles named photons. These photons possess energy which rely on the frequency and wavelength of the light and it is quantified by equation (1); Albert Einstein law as follows [16]:

$$E = hv. \tag{1}$$

Where:

E is the photon energy

h is planks constant = 6.626×10^{-34}Js

v is the photon frequency (speed)

The energy needed to release an outermost electron of an atom of an element; the incident surface is called the work function of the element, denoted Φ. This work function varies with different materials. The equation (equation (2)) describing the process is [16]:

$$hv = \Phi + E_{kin}. \tag{2}$$

Where:

hv is photon energy; E

Φ is Work out

E_{kin} is motion energy of ejected electron

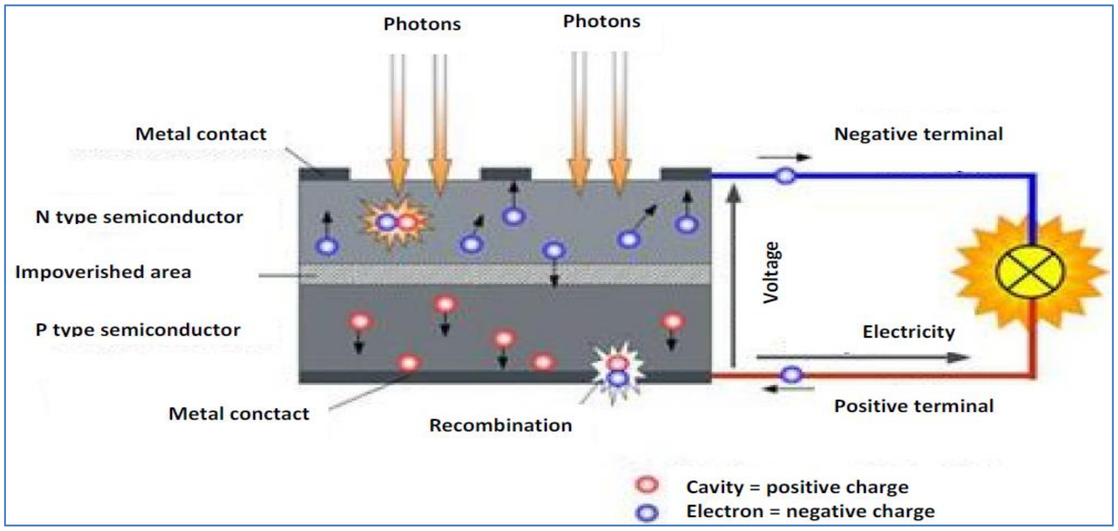


Figure 4: Functioning of a PV cell [6]

2.2.2. Solar PV array model

The Electric model

The electric replica of a PV cell can be considered base on the static characteristics using many intrinsic parameters referred to the data sheets provided by the maker including the detected irradiation, or the dynamic characteristics specifically when a power electronic converter is linked to the PV panel [17]. A PV cell is made up of a p-n junction, created in a thin coat of semiconductor and thus behaves as a diode [18]. This brings about two models known as the one-diode and the double diode models [19].

Static model

The analogous electrical circuit of a one-diode PV cell model is made up of a current source that represents the photocurrent, I_{ph} and one diode representing the diffusion occurrence and that allows the flow of a current I_d. A leakage current I_{sh} caused by the distributed manufacturing defects inside the PV cell flows through a shunt (parallel) resistor, R_{sh}. The PV cell current, I flows through a series resistor R_s resulting in power lost which represents the reduction in the PV cell efficiency by thermal power dissipated through the hole junction substrates [19] [20].

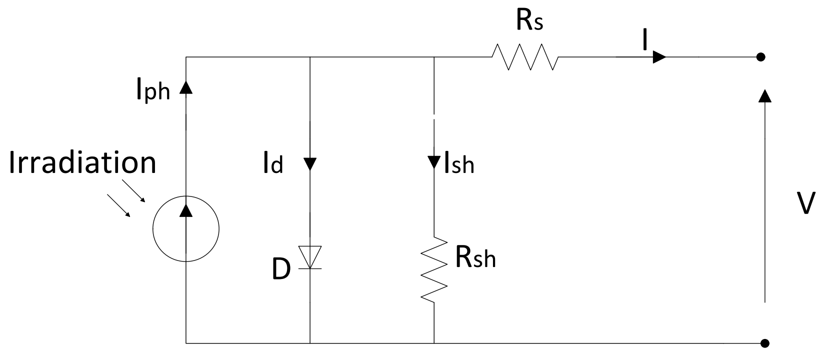


Figure 5: Equivalent electrical circuit of a single diode modelled solar cell

In the presence of insolation, the PV cell current, I can be given by the following equation:

$$I = I_{ph} - I_d - I_{sh} \tag{3}$$

The diode current I_d is given by the following equation:

$$I_d = I_{os}\left\{e^{\frac{V+IR_S}{N_s V_t}} - 1\right\} \tag{4}$$

The shunt current, I_{sh} that flows through the shunt resistor is related to the cell's output voltage by the following equation:

$$I_{sh} = \frac{V + IR_s}{R_{sh}} \tag{5}$$

The junction thermal (terminal) voltage, V_t is provided according to the following equation:

$$V_t = \frac{\alpha T K}{q} \tag{6}$$

The photocurrent, I_{ph} depends on the irradiation, G and temperature, T as follow:

$$I_{ph} = \frac{G}{G_n}[I_{pvn} + K_i(T - T_n)] \text{ ; } I_{pvn} = I_{sc} \tag{7}$$

Substituting the diode current I_d, the shunt current I_{sh}, and the junction terminal voltage V_t, the PV cell output current is express as follows:

$$I = N_p I_{ph} + N_p I_{os}\left\{e^{\frac{q(V-IR_S)}{\alpha T K N_S}} - 1\right\} - N_p \frac{V + IR_s}{R_{sh}} \tag{8}$$

Where;

- N_p is how many parallel cells there are.
- N_s is how many cells there are in series
- I_{os} is the dark or reverse saturation current of the diffusion phenomenon
- α is the diode quality (ideality) factor (1< α >2)
- T is the temperature of the operating cell (Degree Celsius)
- K is Boltzmann constant (1.38 x 10^{-23} J/K)
- q is electronic charge (1.6 x 10^{-19} C)

The unknown parameters such as α, I_{sh}, R_s, I_{os}, and I_{ph} are gotten from the datasheet of the manufacturer.

The PV cell reverse saturation current, I_{os} is expressed as follows:

$$I_{os} = I_{on}(\frac{T}{T_n})^3 EXP\left[\frac{qE_g}{\alpha K}\left(\frac{1}{T_n} - \frac{1}{T}\right)\right] \tag{9}$$

Where:

- I_{on} is the nominal (STC) diode reverse bias saturation current
- T_n is nominal (reference) temperature
- E_g is the band gap for silicon (1.22 eV) at 25 degree Celsius

The nominal diode reverse bias saturation current is derived as follows:

$$I_{on} = \frac{I_{scn}}{e^{(\frac{qV_{ocn}}{\alpha KTn_s})} - 1} \quad (10)$$

The double diode (D_1 & D_2) model further takes into accounts the recombination phenomenon unlike the single diode model, thereby providing improved accuracy for the I-V curve than the one-diode model. Two lumped resistances represent the shunt resistance R_{sh} and the series resistance R_s by which the leakage current I_{sh} flows and the reduction in cell efficiency occurs respectively. The double diode equivalent circuit diagram is as shown in figure 6 bellow.

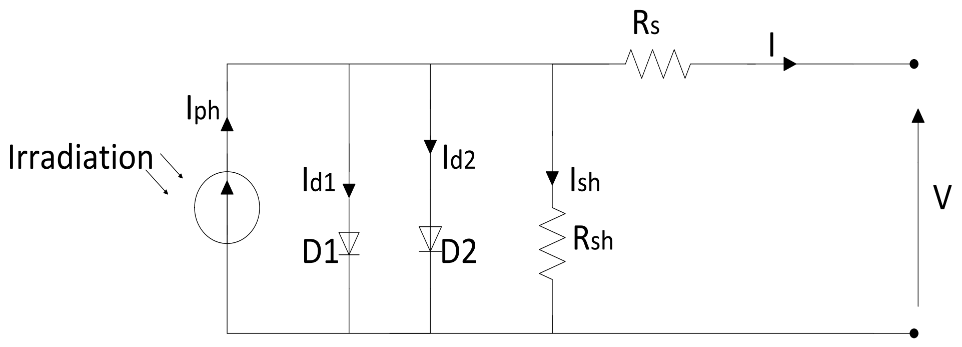


Figure 6: Equivalent electrical circuit of a double diode modelled solar cell [22]

The output current equation is shown in equation (   ) below, where I_{d2} is the current via D_2 due to recombination phenomenon:

$$I = I_{ph} - I_{d1} - I_{d2} - I_{sh} \quad (11)$$

The detailed output current equation for the double diode modelled solar panel is as shown below in equation (   ).

$$I = N_p I_{ph} + N_p I_{os1}\left\{e^{\frac{q(V-IR_S)}{\alpha TKN_S}} - 1\right\} - N_p I_{os2}\left\{e^{\frac{q(V-IR_S)}{\alpha TKN_S}} - 1\right\} - N_p \frac{V + IR_s}{R_{sh}} \tag{12}$$

The series resistance and the shunt resistance are calculated as shown in equations (13) and (14) below, respectively.

$$R_s = \frac{\alpha KTN_p}{qI_m} \ln\left[e^{\left(\frac{qV_{oc}}{\alpha KTN_s}\right)} - \frac{I_m}{I_{sc}}\left\{e^{\left(\frac{qV_{oc}}{\alpha KTN_s}\right)} - 1\right\}\right] - \frac{N_p V_m}{N_s I_m} \tag{13}$$

$$R_{sh} = \frac{\left[\left[N_s I_{sc} R_s - N_p V_{oc}\right]\left[e^{\left(\frac{q(N_s V_m + N_s I_m R_s}{\alpha KTN_s N_p}\right)}\right] q + \alpha KTN_s N_p \left[e^{\left(\frac{qV_{oc}}{\alpha KTN_s}\right)} - 1\right]\right]}{\frac{\left[{N_s}^2 \alpha KTN_p \left\{EXP\left(\frac{qV_{oc}}{\alpha KTN_s}\right) - 1\right\}\right]}{N_p V_m - N_s I_m R_s} - \left[EXP\left(\frac{q(N_s V_m + N_s I_m R_s)}{\alpha KTN_s N_p}\right)\right] qN_s I_{sc}} \tag{14}$$

Dynamic models

The photovoltaic panel is conventionally demonstrated by the Shockley diode in whereby the output current is described by a diode equation as in equation (8) above. Meanwhile, in practice a power electronic converter that pulls a high-frequency undulated current from the solar panel is used for tracking the maximum power point. The rippled current leads to dynamic characteristics dissimilar from the static characteristics derived by the conventional approach [17]. The equivalent electrical circuit model of a solar panel shown in figure (7) below is based on the dynamic conditions.

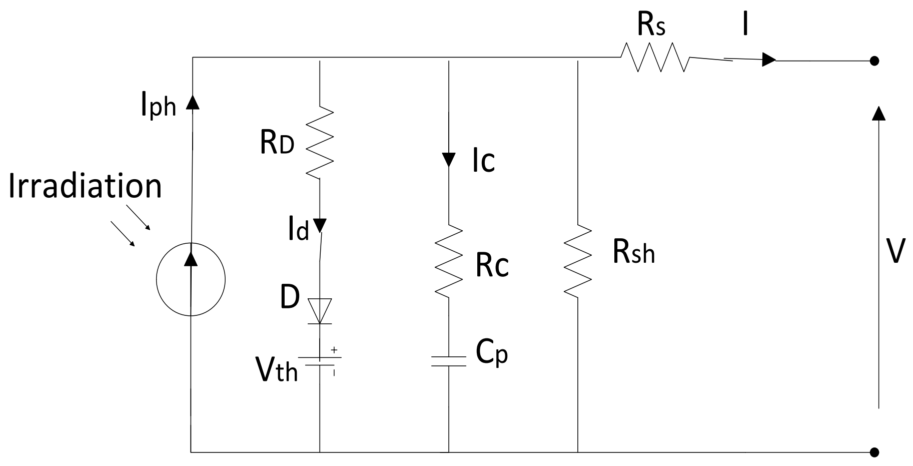


Figure 7: Dynamic equivalent electric circuit of a solar panel [17]

The ‘dynamic’ equivalent electrical circuit of a solar panel differs from the ‘static’ single-diode equivalent electrical circuit, in that the resistive losses in the p-n junctions of the PV cells are represented in a lumped resistance R_d. A voltage source matching threshold voltage, V_{th} which is roughly similar to the voltage at the maximum power point is included. A parasitic capacitor, C_p, in series with a resistance, R_s, is introduced to the static circuit model to account for the dynamic behaviours of the solar panel as figure (7) above. The current flow into the parasitic capacitor is denoted I_c [17].

2.2.3. Performance of solar photovoltaic panels

Figure (8) below shows the Current-Voltage and Power–voltage curves of a photovoltaic cell or module or array demonstrating the maximum power point (MPP). The maximum power point is utilized to quantify the efficiency of the device as provided by the standard equation represented in equation (15) below [13].

$$Efficiency, \eta = \frac{P_{out}}{P_{in}} = \frac{V_{mpp} I_{mpp}}{E_{stc} * A} \qquad (15)$$

Where:

- P_{out} is electrical output power
- P_{in} is radiation power (sun)
- V_{mpp} is effective value of output voltage
- I_{mpp} is effective value of the electricity output

➢ E_{stc} is specific radiation power (1000W/m2 @ Standard test Conditions)

➢ A is area of solar PV panel

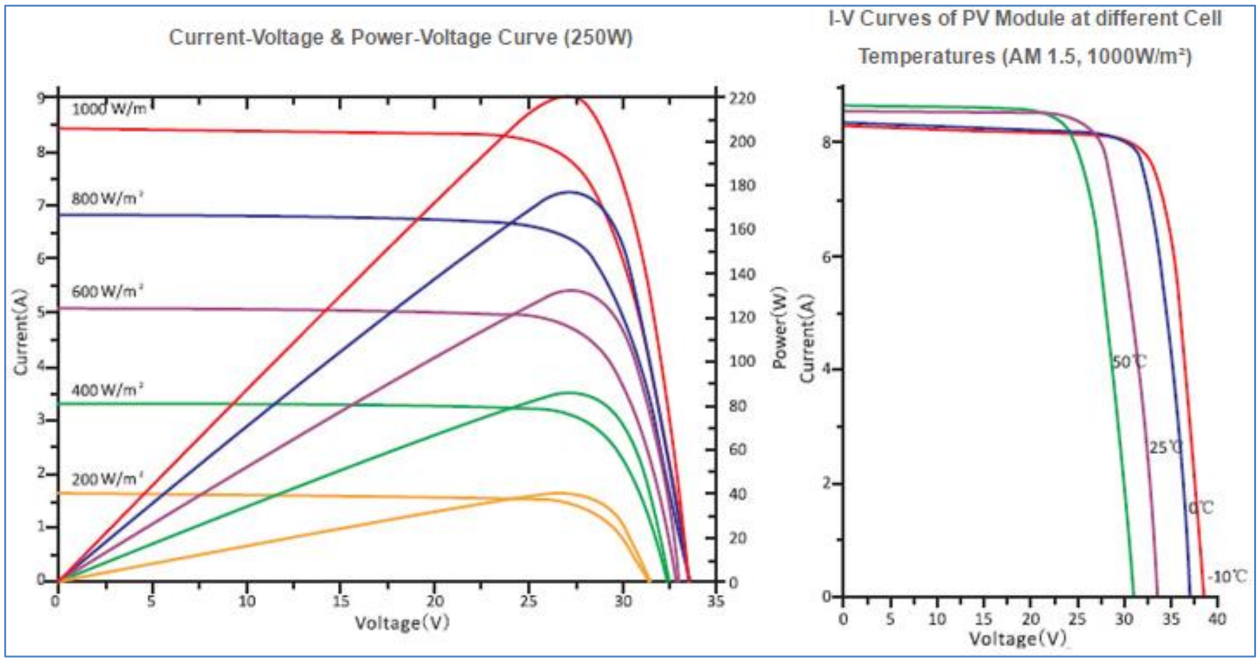


Figure 8: I-V and P-V characteristic curve for a PV cell, module or array variation [23]

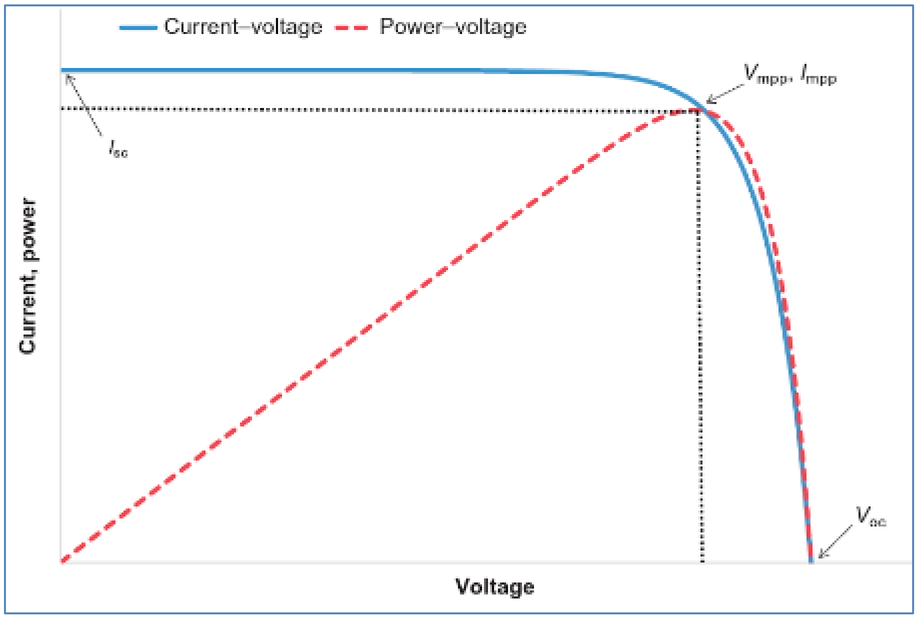


Figure 9: Maximum power point for a PV cell, module or array [13]

2.2.4. Advantages of stand-alone PV systems

The outstanding advantages of a PV system are [24]

- Long life cycle of up to about 20 – 25 years.
- Null operation cost because it is a free resource.
- More reliable results and low variability of the system.
- The cost of up keeping of the system is low.
- No air and sound pollution.
- Energy conservation

2.2.5. Disadvantages of off-grid PV systems

Solar photovoltaic systems have some disadvantages as follows: [25]

- The most burden of affording a PV system is it high up-front cast which most of low income families cannot afford up-front.
- Small stand-alone PV systems often require batteries for power storage and may also need backup electric generators.
- Solar PV's systems are not economical for thermal loads such as heating, cooking and ironing. Often require efficient or direct current (DC) appliances.

2.3. Charge controllers

Charge controllers or charge regulators or battery regulators are used to charge batteries with solar power. Generally, charge controllers actually allow the flow of power only from the panel to the battery / load. Charge controllers also protect the battery from overcharging and "deep" discharge which can damage the battery. There are three types of charge controllers; the series, shunt, PWM (pulse width modulation) and the MPPT (maximum power point tracking). [26] [27]

2.3.1. Series and Shunt charge controllers

The series controller is commonly used in small PV systems, though it can also be used for large systems. It has a type of control element which is connected in series with the solar PV array and the battery. The shunt charge controller controls the charging of a battery by shot-circuiting the array internal to the controller. [28]

2.3.2. PWM Charge controller

The PWM charge controller acts as a switch connecting the solar array to the battery and load. It will pull down the nominal voltage of the solar panel connected to its input, to near that of the battery plus the voltage losses in the cabling and controller (referred to as V_{PWM}). It therefore requires that the solar panel's nominal voltage be equal to that of the battery. This way, the solar panel is not used at its maximum power point, quite often. Unlike the series and shunt charge controller, the PWM charge controller does control the charging rates (reduce the battery current as the battery gets full) of the battery [29].

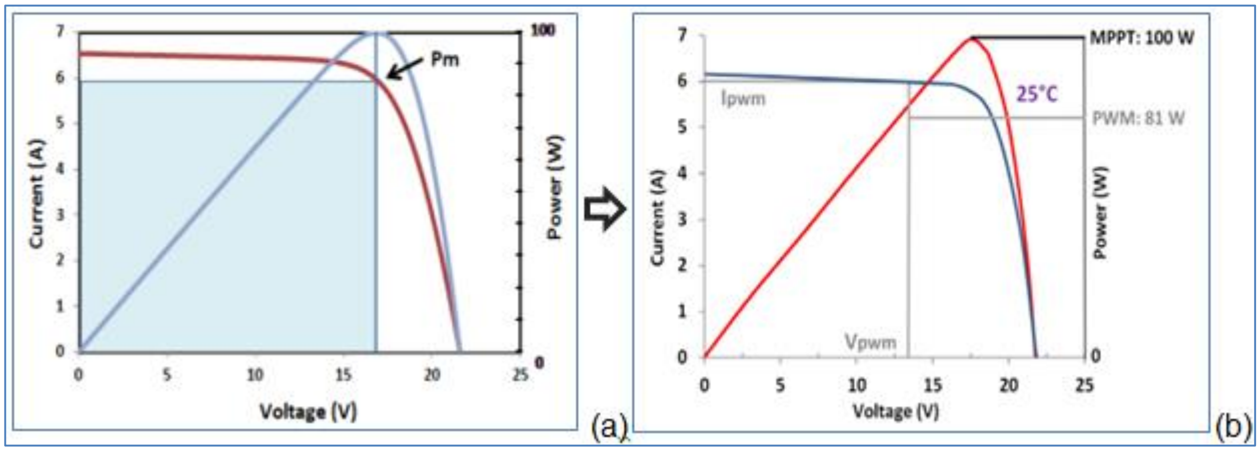


Figure 10: (a) Solar Panel maximum power output, (b) Charge controller's harvested power from solar PV [29]

$P_{max} = V_{max} \times I_{max}$, that is 18V x 5.6A = 100W

The power harvested from the solar panel by the PWM charge controller will be:

$P_{in} = V_{PWM} \times I_{PWM}$, that is 13.5V x 6A = 81W which is 19% less. V_{PWM} is equal to the discharged battery voltage plus the cabling and charge controller losses.

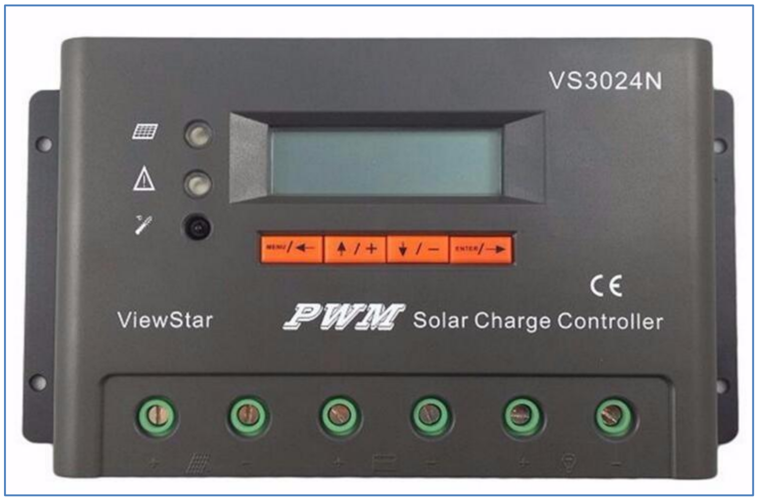


Figure 11: 30A 12V/24V PWM charge controller

2.3.3. MPPT Charge controller

The maximum power point charge controller operates by transforming DC power at a higher voltage with lower current to DC power at a lower voltage with higher current. Therefore the amount of power at the input equals the amount of power at the output. It also controls the battery charging rate by reducing the battery charging current as it approaches full state. [29].

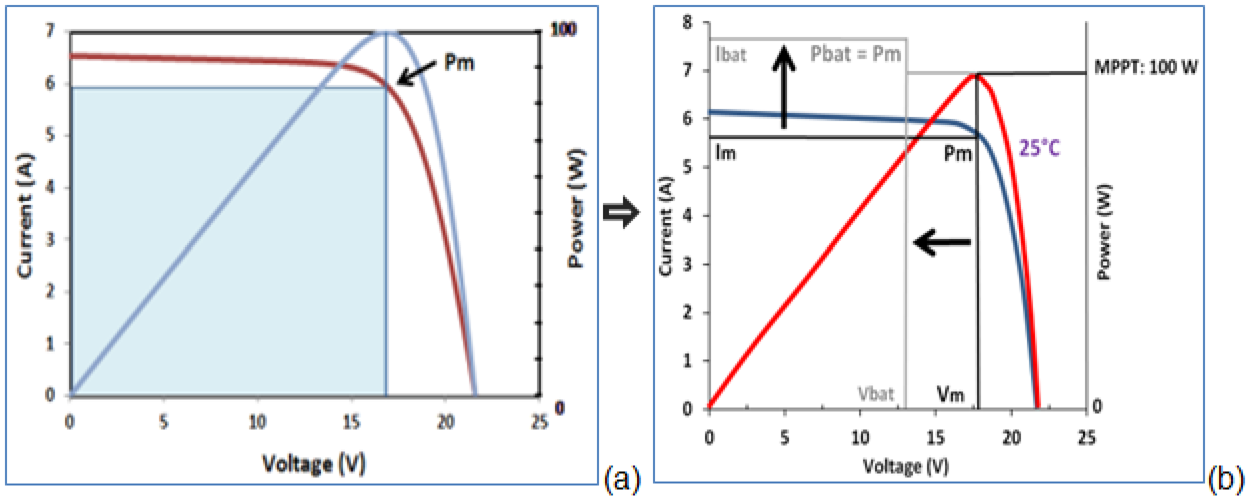


Figure 12: Graphical representation of the MPPT charge controller DC to DC transformation

Just like the PWM charge controller, the MPPT charge controller will pull down the solar panel's nominal voltage to near that of the charging battery (V_{bat} = 13V) but will pull up the battery's charging current, I_{bat} to the solar panel's wattage (100W) divided by V_{bat}.

I_{bat} = 100W / 13V = 7.7A

P_{max} = V_m x I_m = 18V x 5.6A = 100W

P_{bat} = V_{bat} x I_{bat} = 13V x 7.7A = 100W

Therefore the solar panel's power at maximum power point is same as the charge controller output power to the charging battery at all times.

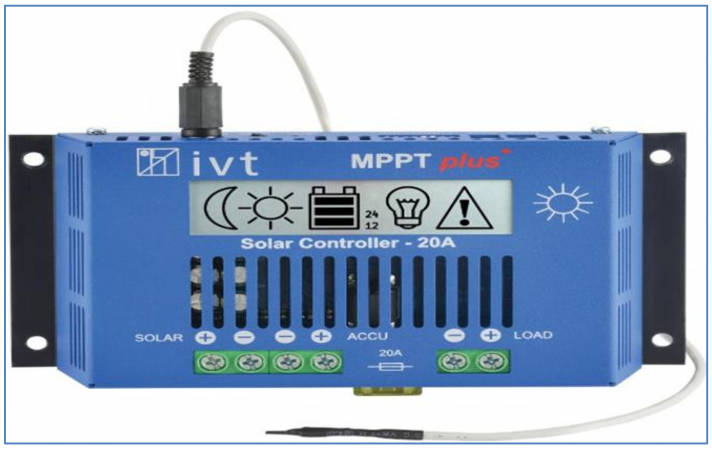


Figure 13: 12V/24V-20A MPPT solar charge controller series [30].

The MPPT- plus solar charge controller series exhibits the shunt, series and MPPT modes of operation. This charge controller series is available in 10A, 20A, and 30A amperages. Its most important characteristics are [30]:

- Suitable for lead acid, lead AGM, lead gel, and LIFePo.
- Solar input maximum voltage of 70V.
- Automatic battery voltage recognition 12/24V.
- Maximum efficiency of up to 97%

2.3.4. Advantages and disadvantages of both PWM and MPPT charge controller

The MPPT charge controller is more sophisticated than the PWM charge controller. They both have advantages and disadvantages over one another as follows:

	PWM	MPPT
Advantages	25% to 50% cheaper than the MPPT charge controller	Highest efficiency, especially in cold climates
	Has fewer Electronic components and less thermal stress thus longer lifespan	Can be used with 60 cell panel
Disadvantages	Requires care and design experience in sizing the PV array and battery bank	2 to 3 type more expensive than a comparable PWM charge controller

	Cannot be used efficiently with 60 cell panels	Has more electronic components and thermal stress, thus shorter lifespan is expected

Table 2: Pros and Cons of PWM & MPPT charge controllers [26]

2.3.5. Charge controllers 3 – 4 stages

Charge controllers charge batteries in three to four stages; bulk charging, absorption charging, float and/or recondition charging [31].

Bulk stage: During bulk charging, the charge controller supplies the maximum constant charge current to the battery. Here, the flat battery voltage increases with time until it reaches the set charging voltage.

Absorption stage: When the set charging voltage is reached, the battery continues to charge at this regulated voltage for a while until the charging current tapers (drops) to 6.25% of the maximum charging current of the battery. Here the battery is fully charged.

Storage (Float) stage: This charging stage conserves the fully charged battery at 100% without allowing the battery to overcharge and be damaged.

Recondition stage: If the battery is allowed connected to the charge controller, the charge controller switches back to bulk stage for about 85 minutes in order to revive the battery, after about 12 hours of being fully charged.

2.4. Energy storage

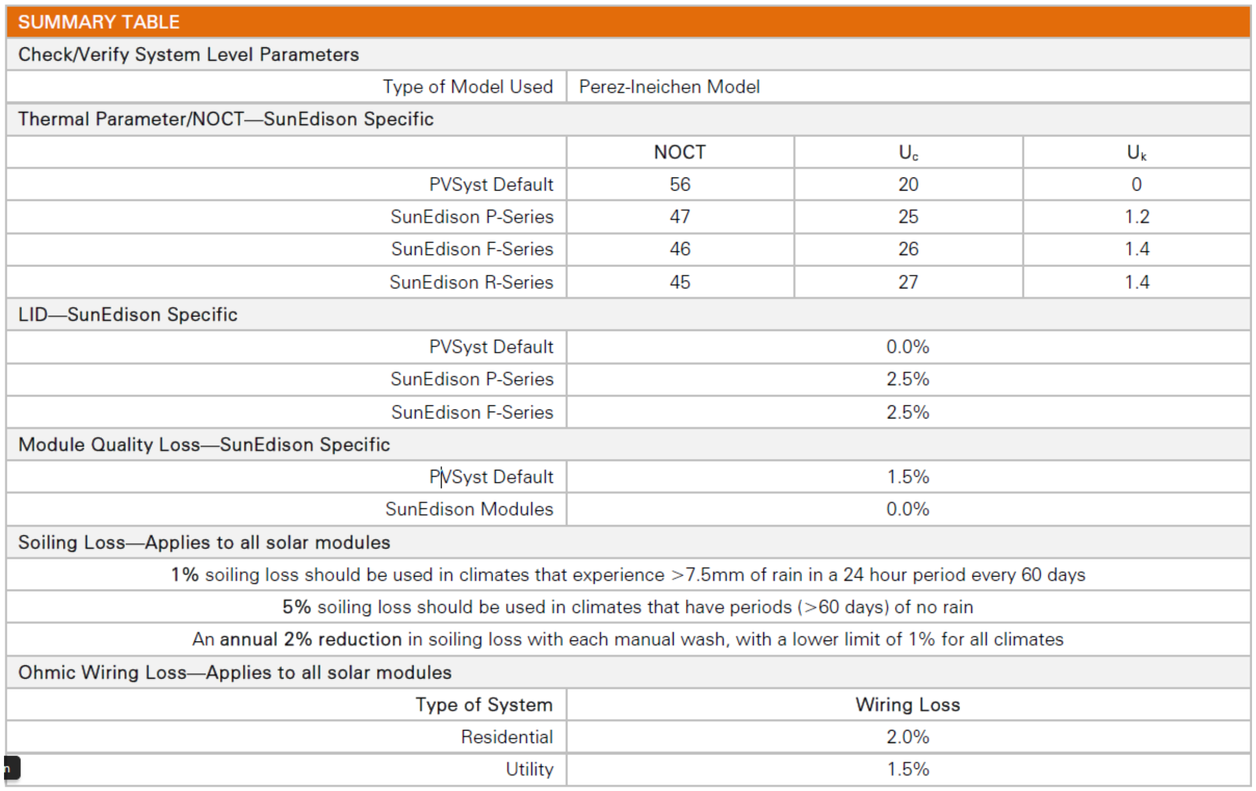
The need for energy storage is very critical in renewable energy technologies since some of the resources such as wind and solar generate electricity in irregular daily patterns depending on the availability of the resource. Storage therefore allows the use of electricity instantaneously or stored for later use, thereby reducing wasted energy. Lead acid batteries are frequently used for the storage of the energy generated from renewable energy sources most especially as stand-alone systems for domestic purposes. The most important advantage of energy storage is to provide a smoother and more stable electrical supply to the domestic house. Furthermore, electrical storage, benefits demand management, current quality and supply, and, improves the reliability of electrical supply. On the other hand, energy storage has some few disadvantages in that they can be costly and their lifetime can be significantly limited which can be directly associated with their reversibility [32].

2.5. Simulators

A device or system that imitates well defined conditions or the behaviours of a true process or machine for the essence of research or operator training is termed a simulator [33]. This device or system that simulates is commonly computer based and therefore has a desktop application referred to as a software (simulation) package. Standard software off the shelf simulation packages are PVSyst, PSIM, Homer Pro, Matlab-Simulink and so on. These software packages are programs that contain fundamental information about the systems it is designed to simulate. The fundamental information (default input parameters) is termed input data to the simulator and can be modified and added.

2.5.1. PVSyst

PVSyst has an input file with the extension “.PAN” that gives specific module parameters as simulation inputs. Simulation input files can be generated from measurements of production modules under the different conditions of temperature and irradiance. The PAN files of SunEdison’s PV modules are generated by independent and accredited test laboratories while some manufacturers generate their own PAN files. These laboratories measure randomly chosen production modules, then create PAN files by regression analysis techniques [34].



SUMMARY TABLE			
Check/Verify System Level Parameters			
Type of Model Used	Perez-Ineichen Model		
Thermal Parameter/NOCT—SunEdison Specific			
	NOCT	U_c	U_k
PVSyst Default	56	20	0
SunEdison P-Series	47	25	1.2
SunEdison F-Series	46	26	1.4
SunEdison R-Series	45	27	1.4
LID—SunEdison Specific			
PVSyst Default	0.0%		
SunEdison P-Series	2.5%		
SunEdison F-Series	2.5%		
Module Quality Loss—SunEdison Specific			
PVSyst Default	1.5%		
SunEdison Modules	0.0%		
Soiling Loss—Applies to all solar modules			
1% soiling loss should be used in climates that experience >7.5mm of rain in a 24 hour period every 60 days			
5% soiling loss should be used in climates that have periods (>60 days) of no rain			
An **annual 2% reduction** in soiling loss with each manual wash, with a lower limit of 1% for all climates			
Ohmic Wiring Loss—Applies to all solar modules			
Type of System	Wiring Loss		
Residential	2.0%		
Utility	1.5%		

Table 3: Typical simulation input parameters [34]

2.5.2. Homer Pro

HOMER is a simulation package and also an electrical system design that makes easy the job of evaluating design options for both stand-alone and on-grid power systems for distributed generation (DG) applications. The users are allowed to evaluate the technical feasibility and economics of a large number of technology options and to account for the energy resource availability, uncertainty in technological costs, and other variables, by means of the software's optimization and sensitivity analysis algorithms [35]. The accuracy of the economic analysis is specifically based on the price information of the system hardware components entered into Homer. The information about project location and selected PV panel are already contained in the simulator as default input data and are used for energy resource feasibility of the project. The figure below shows a sample project that can be simulated.

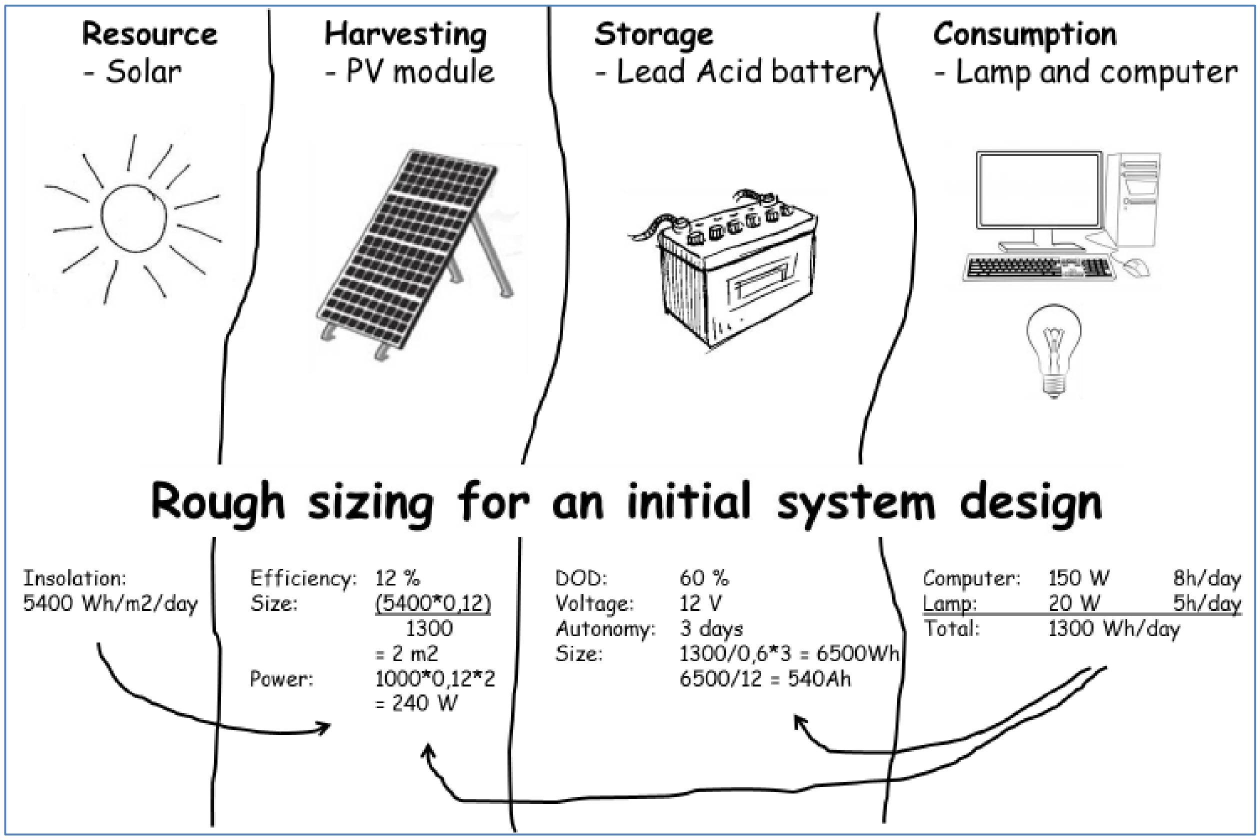


Figure 14: Typical microgrid Power system hardware components [16]

2.6. Photovoltaic emulator control strategy

The control strategy of the simulator is about how to predict the operating point of the PV module which is being simulated. It therefore does this by locating the voltage and the current on the I-V characteristic curve depending on the resistance of the load at the output of the simulator. This is though different from the MPPT algorithm which is aimed at locating the highest point of the power-voltage (P-V) behaviour curve of the PV module [36].

2.6.1. Type of control strategy

There are many types of control strategies such as the direct referencing method, hybrid-mode controlled method, perturb and observe method, resistance comparison method, analogue based control method etc.

The direct referencing method was used in this research. It is a common strategy since it does not require additional algorithm (such as the look-up table method) to locate the operating point of the PV module which is being simulated.

The direct referencing method is subdivided into two types according on the closed-loop system and the PV model used for the PV simulator. These are the current-mode controlled system with the PV module's voltage as the input parameter, and the voltage mode controlled system with the PV module's current as the input parameter and shown in figure 30a & 30b below respectively.

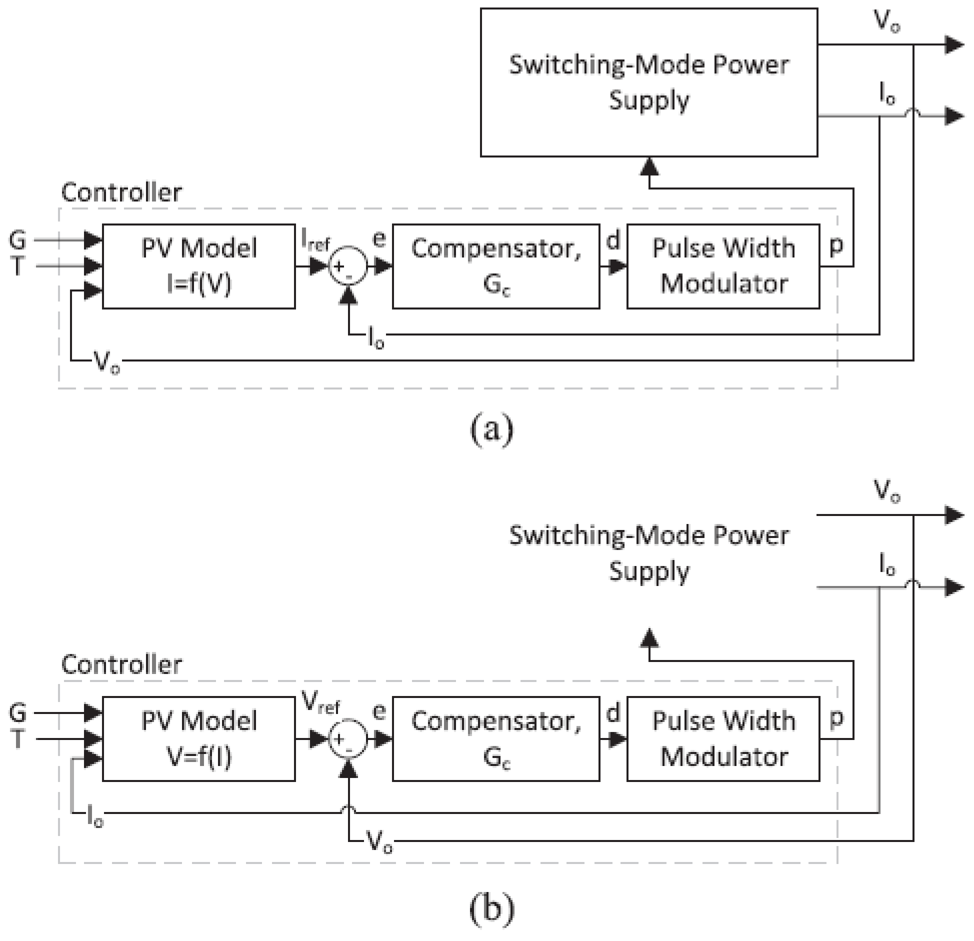


Figure 15: Direct referencing method's control strategies: Current-mode (a), voltage-mode (b) [36]

2.7. Power supplies/amplifier

In broad terms, a power supply is anything that delivers any form of power such as an internal combustion engine or a hydraulic pump. In this research, power supplies will be limited to electrical power supplies that are frequently used for test and measurement, maintenance, and product development activities. In this regards, there are three major types of electrical power supplies: the AC to DC power supply which can be seen in battery chargers, bench-top power supplies, or then DC to AC power supplies as in inverters and generators, the AC to AC power supplies as in isolation transformers, frequency changers, and the DC to DC power supplies as in converters. The theory of operation of these electrical power supplies is either by linear regulation or switching mode [37]. The difference between these two modes of operations is as follows

Type	**advantage**	**Disadvantage**
Switching	Higher efficiencies of up to 95%. Lighter weight and take little space. More affordable for higher power.	Noisier and including EMC disturbances and impulse noise. Much slower transient response.
Linear	Good line and load regulation. Fast transient response. Low noise and EMC disturbances. Can produce very low current output.	Low efficiency of about 30 to 40%. Much weight due to Transformer. Large heat sinks. Larger for higher power.

Table 4: Pros and Cons of Power supply operation mode. [38]

3. Methodology (Simulator design procedure)

3.1. Graphical user interface (GUI)

A simulator requires data to be entered so it can then process the data and yield results. A LabVIEW bloke code was designed to receive a solar PV system data via its graphical users' interface and then process it using the solar PV mathematical model. Different solar cell materials have different diode quality (ideality) factors, α as well as different band gap voltages, E_g. in electron volts (eV). These are as shown on the table below.

PV Cell Material Types	Diode Ideality Factor	Bandgap Voltage (eV)
Si-mono	1.2	1.11
Si-poly	1.3	1.14
a-Si-H	1.8	1.65
a-Si-tandem	3.3	2.9
a-Si-tripple	5	4.6
Cadmium telluride (CdTe)	1.5	1.49
Copper Indium Sellenium (CIS)	1.5	1.48
Gallium arsenide (AsGa)	1.3	1.43

Table 5: PV material Band gap voltages and diode quality factors [39] [40]

3.2. AC to DC Power Supply

An AC to DC power supply was designed and built to be used in powering a programmable switch mode DC-DC buck converter and its peripheral circuit network.

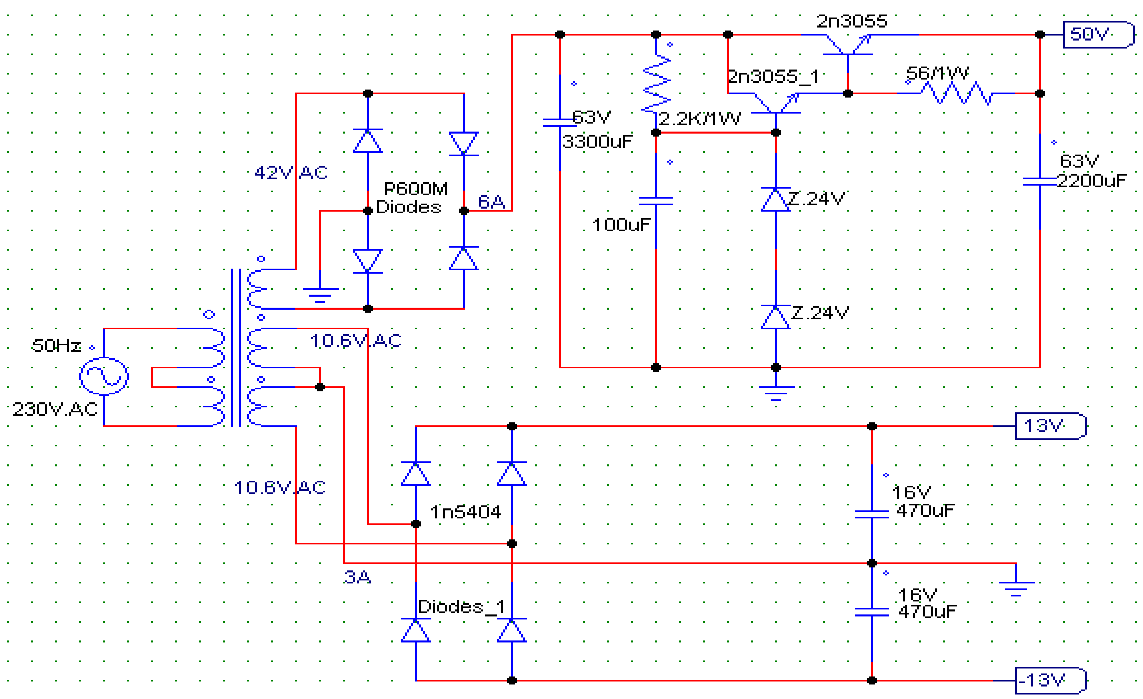


Figure 16: AC to DC power supply

3.3. DC – DC Buck Converter

A programmable switch mode power supply was designed and built. The control was implemented such that the user of the simulator can adjust the output voltage by a push of a button but the output current limit was controlled via the GUI by the set irradiation value.

3.3.1. Power stage

The power stage was designed as shown in figure 16 below:

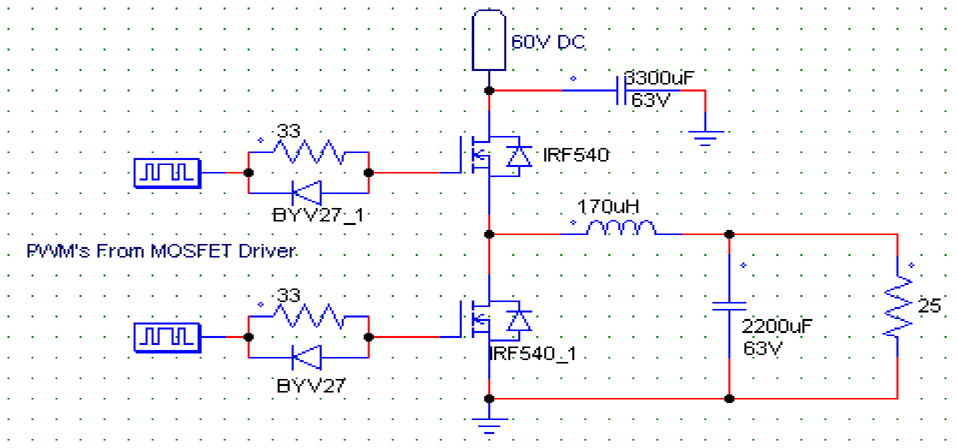


Figure 17: Synchronous buck converter topology design

Power MOSTFET's (IRF540) were used whereby one served as a power switch and the other as a free-wheeling diode for the inductor current. The MOSFET's were driven using a MOSFET driver IC (IR2113).

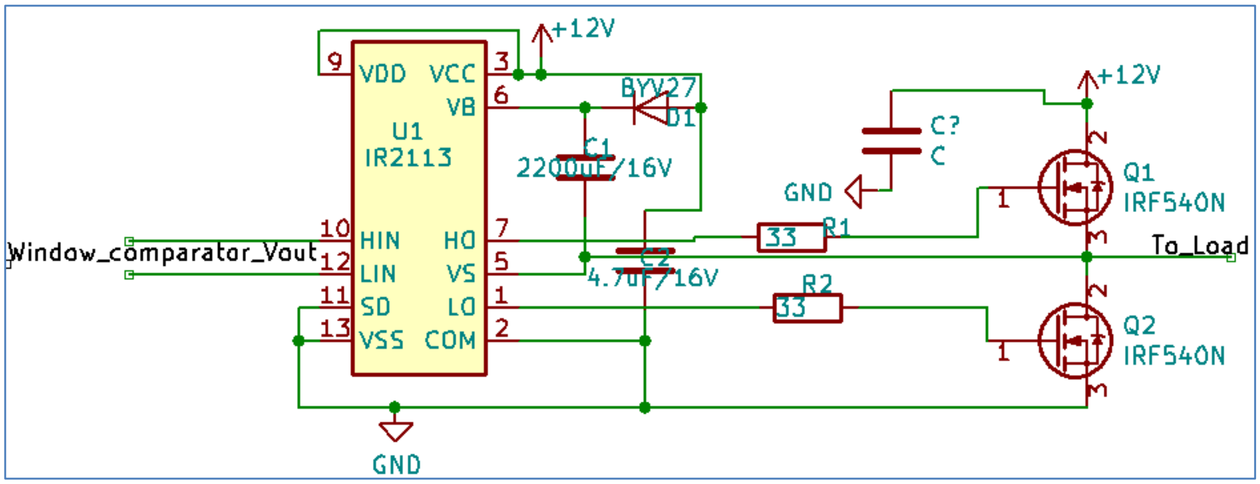


Figure 18: MOSFET driver typical connection diagram

3.3.1.1. *Component Calculations*

The design specifications were as follows:

V_in = 60V

V_out = 5V – 50V

Frequency = 50KHz

I_out = 0.25A to 5A

Change in V_out = 10mv

Duty cycle (D_min) = 5/60 = 8.3%

Duty cycle (D_max) = 50/60 = 83%

% V_out ripple = 10%

Sizing the critical inductance (Lc) value

$$Lc = \frac{(1 - Dmax).Rmax}{2f} \tag{16}$$

Lc = ((1- 0.83)100)/(2*50KHz)

= 170µH

L = 1.05*Lc

= 178.3µH

Minimum output capacitance (C_out)

$$Co\_min = \frac{(1 - Dmin).Vo}{8.L.f^2.\Delta Vo} \tag{17}$$

Co_min = ((1 - 0.083)*50) / (8*178.5µ*50K*50K*0.01)

= 1284.3µF

A 2200µF @ 60V was used.

Minimum input capacitance (C_in)

$$Cin\_\mathrm{min} = \frac{(1 - Dmax).Dmax.Io\_max}{f.\Delta Vin} \tag{18}$$

Cin_min = ((1 - 0.83)*0.83*5) / (50K*0.01)

= 1411µF

A 2200µF @ 60V was used.

3.3.1.2. Power stage simulation

The buck converter simulation was done using PSIM software based on these calculated values above.

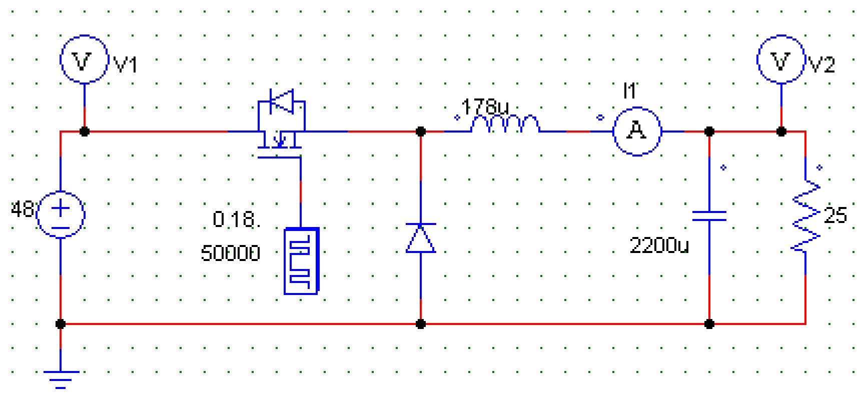


Figure 19: DC to DC buck converter simulation diagram

The simulation was performed at a low duty cycle, mid duty cycle and high duty cycle, and the output voltage and current signals analysed.

Low duty cycle

The following input (V1) and output (V2) voltages, and output current (I1) signals were recorded at a low duty cycle of about 5%.

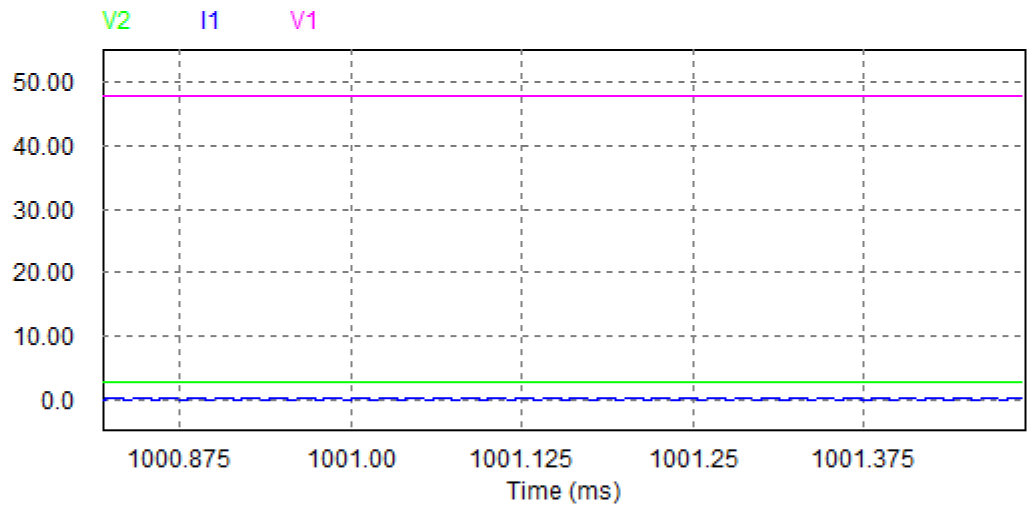


Figure 20: Buck converter simulation results at 5% duty cycle

The ripples of the output voltage (V2) and output current (I1) were analysed one at a time as shown in figures 18 & 19 below.

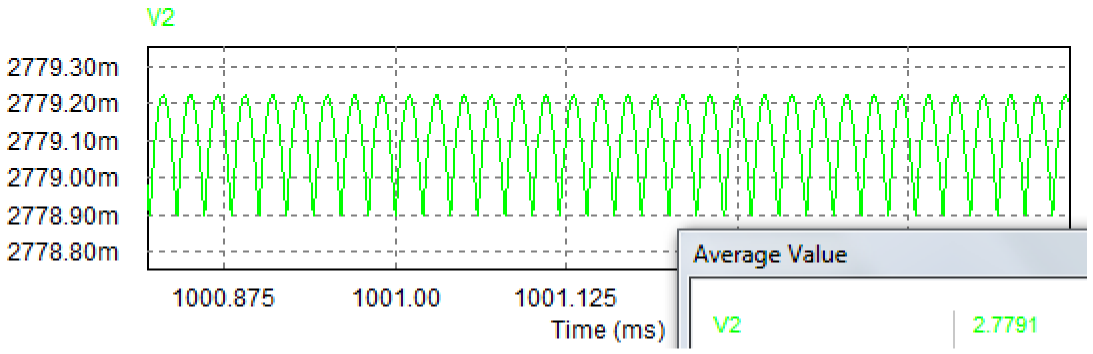


Figure 21: Buck converter output voltage ripple at 5% duty cycle

An average output voltage of 2.771V with a ripple of about 0.3mV was noted as shown above.

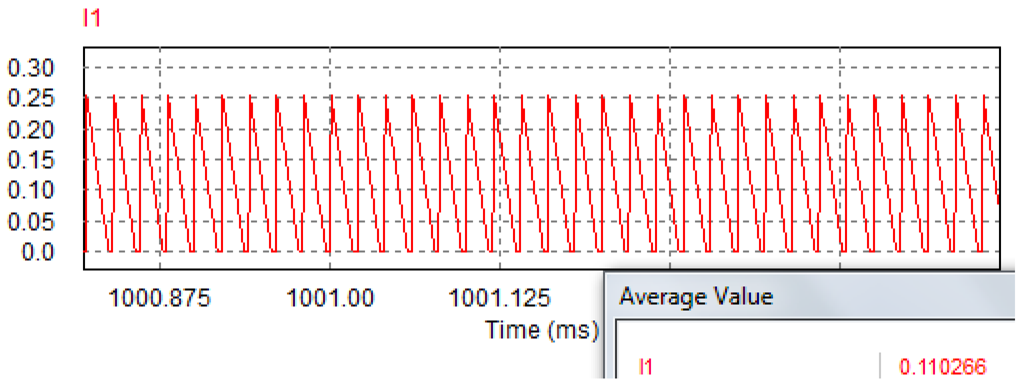


Figure 22: Buck converter output current ripple at 5% duty cycle

An average output current of 0.110266A with a ripple of about 0.25A was noted as shown above.

Mid duty cycle

The following input (V1) and output (V2) voltages, and output current (I1) signals were recorded at a mid-duty-cycle of about 50%.

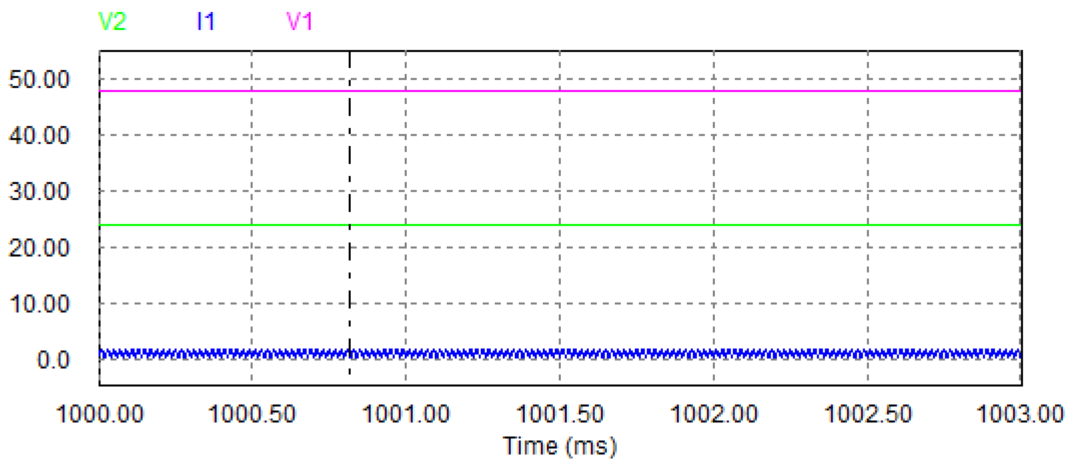


Figure 23: Buck converter simulation results at 50% duty cycle

The ripples of the output voltage (V2) and output current (I1) were analysed one at a time as shown in figures 23 & 24 below

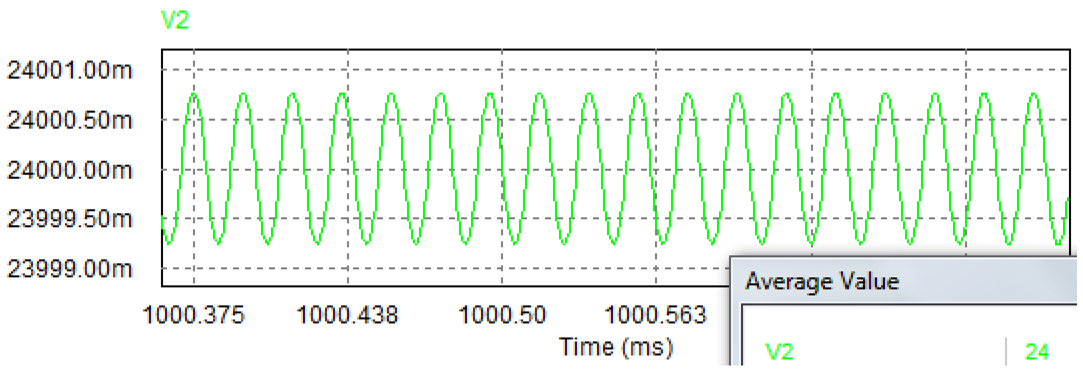


Figure 24: Buck converter output voltage ripple at 50% duty cycle

An average output voltage of 24V with a ripple of about 1.5mV was noted as shown above.

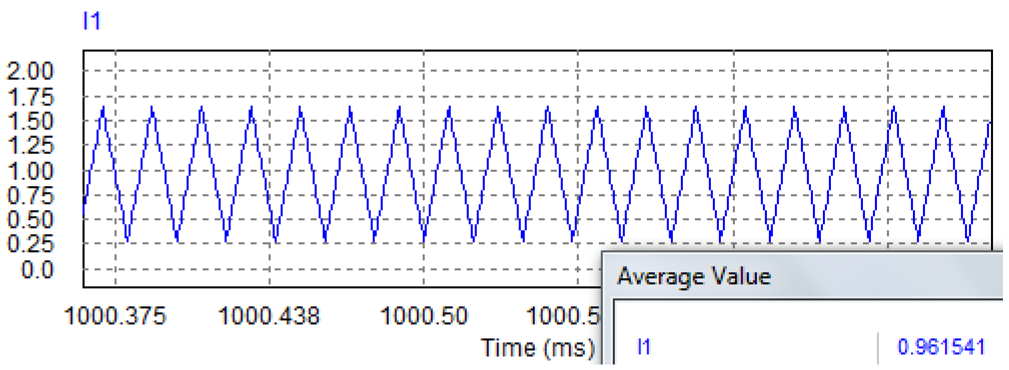


Figure 25: Buck converter output current ripple at 50% duty cycle

An average output current of 0.961541A with a ripple of about 1.5A was noted as shown above.

High duty cycle

The following input (V1) and output (V2) voltages, and output current (I1) signals were recorded at a mid-duty-cycle of about 95%.

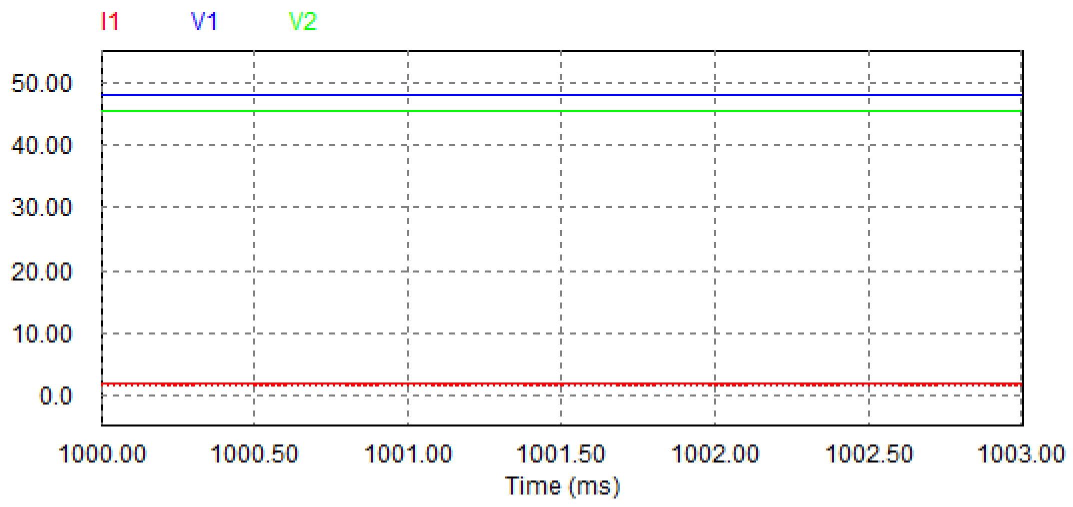


Figure 26: Buck converter simulation results at 95% duty cycle

The ripples of the output voltage (V2) and output current (I1) were analysed one at a time as shown in figures 26 & 27 below.

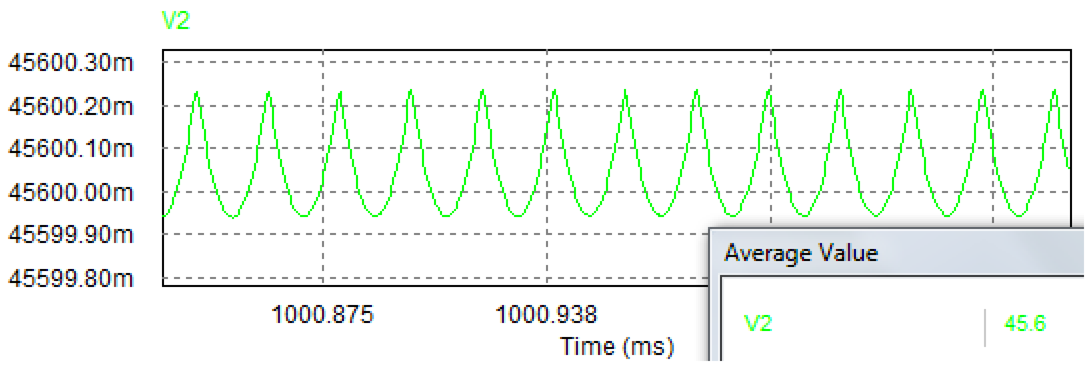


Figure 27: Buck converter output voltage ripple at 95% duty cycle

An average output voltage of 45.6V with a ripple of about 0.3mV was noted as shown above.

The output voltage ripples at low and high duty cycles are the same but the ripple signals appear relatively inverted. This is because at high duty cycles (longer ON-time) the output capacitor is allowed to charge more (for longer time) and to discharges less unlike at low duty cycle (shorter ON-time). At the mid duty cycle, the ripple signal is equally shaped at the troughs and crest because of the ON & OFF time of the MOSFET gate signal. Thus, equal charge and discharge time of the output capacitor.

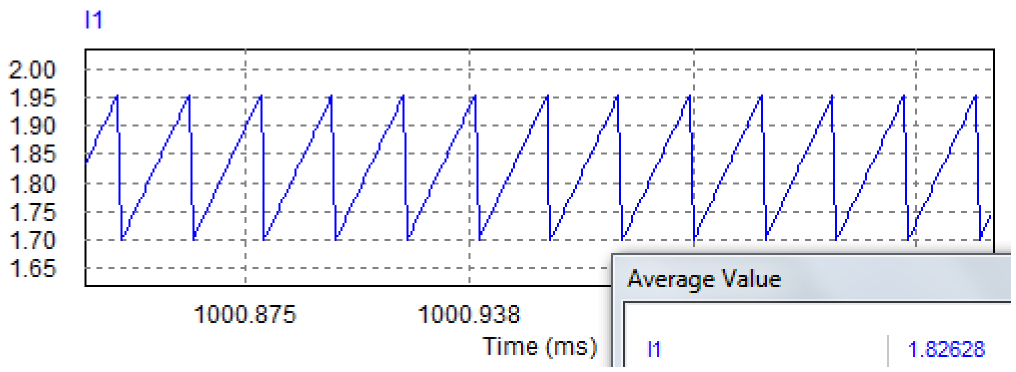


Figure 28: Buck converter output current ripple at 95% duty cycle

An average output current of 1.826A with a ripple of about 0.25A was noted as shown above.

The output current ripples at low and high duty cycles are the same but the ripple signals are sloped more on opposite side to one another. This is because at high duty cycles (longer ON-time) the inductor is allowed to charge for a longer time and to discharge for a shorter time unlike at low duty cycle (shorter ON-time). At the mid duty cycle, the ripple signal is equally sloped on both sides because of the equal ON & OFF time of the MOSFET gate signal. Thus, equal charge and discharge time of the inductor.

3.3.2. PWM signal conditioning circuit

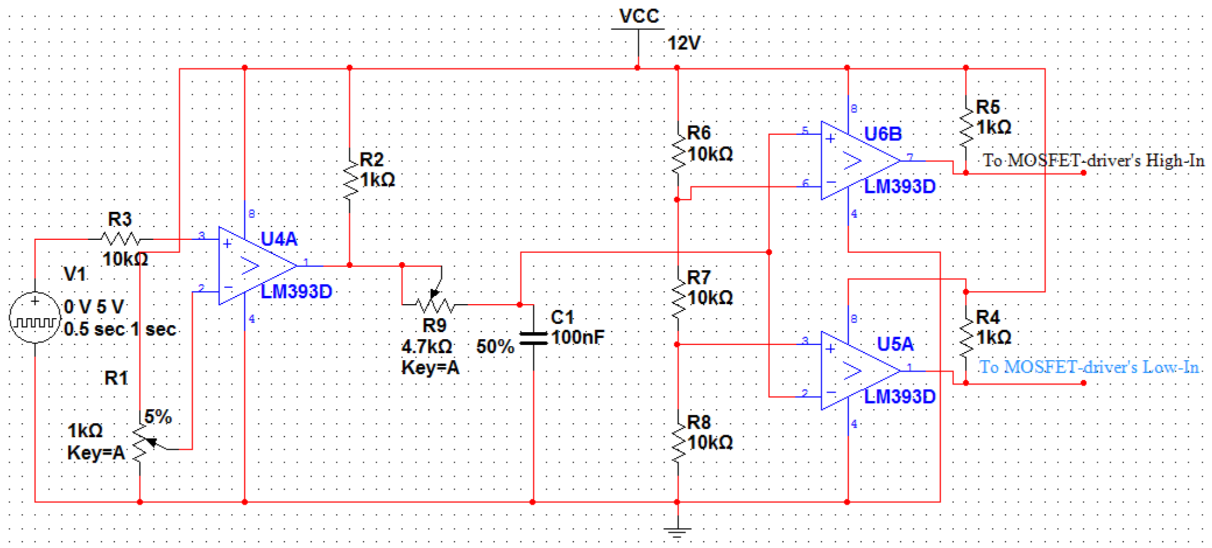


Figure 29: PWM signal conditioning circuit

The PWM signal conditioning circuit as shown above is made up of a comparator circuit, a first order low pass filter, and a window comparator.

The comparator takes as input a 5Vp-p PWM signal from a microcontroller as shown in figure 28 below with practical reference (digital-oscilloscope screen shot) in appendix E, figure 49.

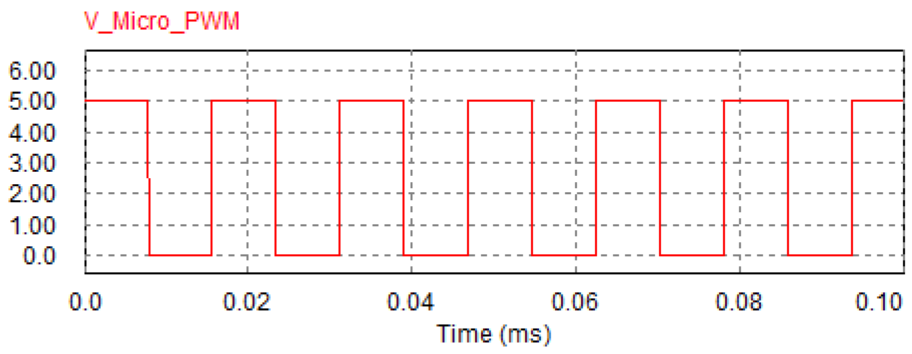


Figure 30: Microcontroller PWM (0-5V)

The comparator then yields a 12Vp-p PWM signal at its output as shown in figure 29 below with practical reference (digital screen shot) in appendix E, figure 50.

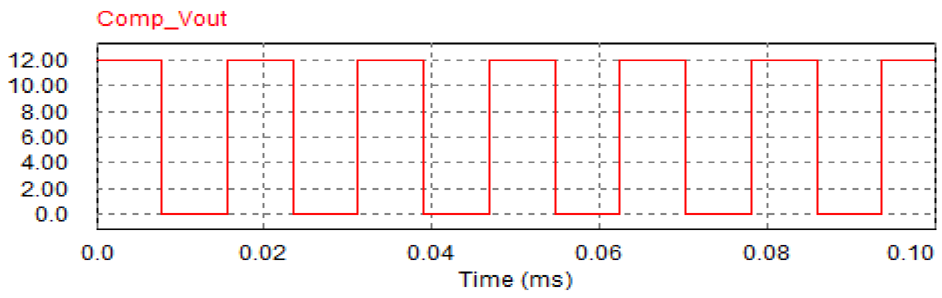


Figure 31: comparator's 0-12Vp-p output PWM signal

The 0-12V PWM is the passed through a first order low pass filter to prolong the rise and fall time of the 0-12V PWM. This delay in rise and fall of the PWM is important for the death time of the power-stage switching MOSFET's in that the MOSFET's are allowed to recover from an ON-state to an OFF-state.

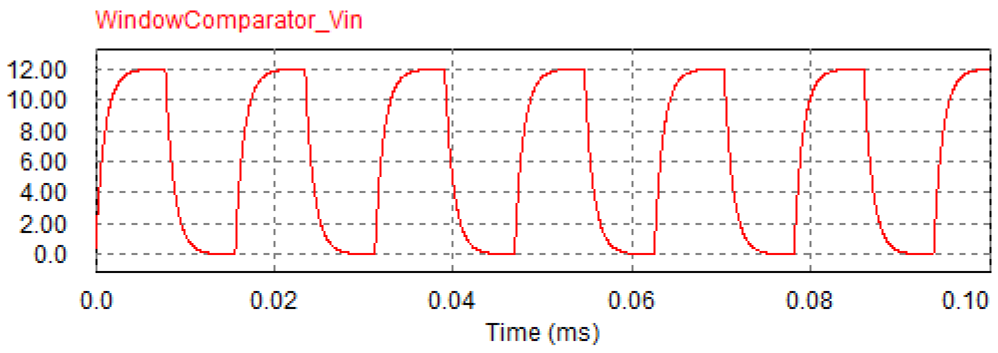


Figure 32: Low Pass filter output PWM signal

The window comparator takes as input, the low pass filter PWM signal, to yield a PWM signal at its two outputs each. Due to the above death time, the two signals appear off together but never ON together. The practical reference (digital screen shot) is shown in appendix E, figure 51.

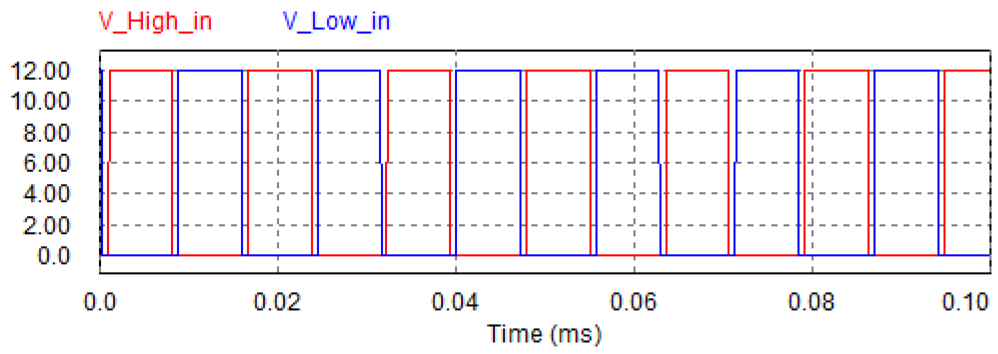


Figure 33: Window comparator output PWM signals

These two PWM signals at the comparators' outputs are almost 180^0 out of phase such that they both are allowed to be OFF at the same time (the death time) but never ON at the same time. These two PWM signals go into the High_in and Low_in inputs of the IR2113 MOSFET driver integrated circuit (IC). The practical reference (digital screen shot) is shown in appendix E, figure 52.

4. Hardware simulator’s control strategy

Voltage-mode control system was the type of direct referencing control strategy used in this research. The output current and the output voltage were read by the control microcontrollers (arduino uno & NI daq-USB 6009) and then used the information to drive the PWM signal in order to make the proper adjustments of the output voltage, thereby regulating the load current and voltage to the operating point.

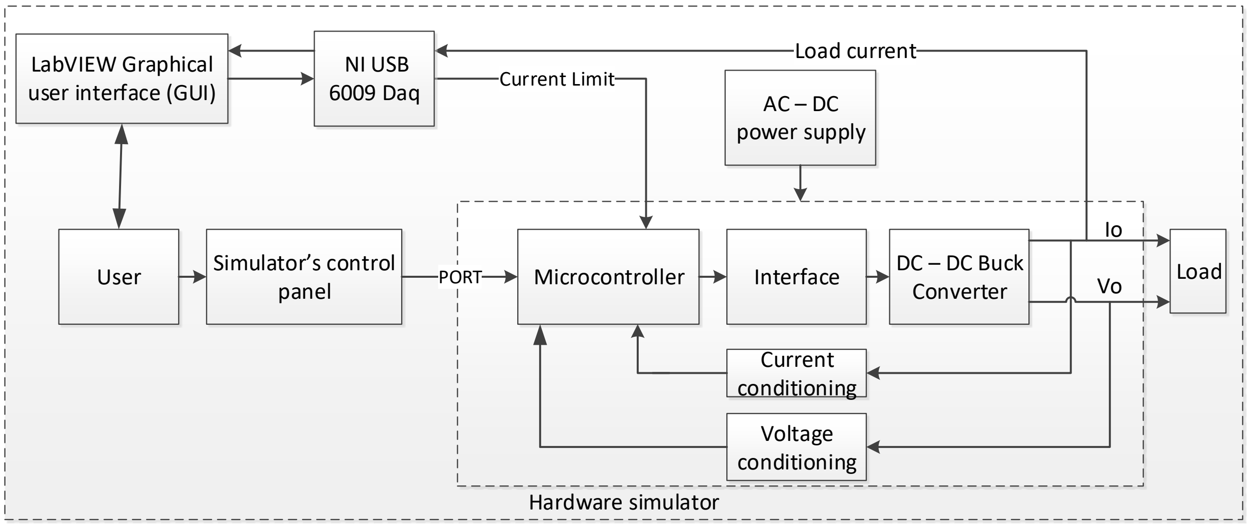


Figure 34: Hardware simulator block diagram

Once more, the simulator controller requires both the output voltage value as well as the output current value, in order to carry out the appropriate control.

4.1. Output voltage feedback

The expected maximum output voltage of the buck converter was 48V. The maximum input allowed at the microcontroller’s analogue input was 5V (analogue reference: AREF). The 48V had to be scaled down within the microcontroller’s analogue reference rage 0V to 5V. An LM741 operational amplifier was used to designed and build an attenuating amplifier, with a gain of 5V divided by 48V; 0.1042 as follows:

$$A_V . A_V = \left(-\frac{V_{out}}{V_{in}}\right)(-1); \quad A_V = -\frac{R_f}{R_{in}} \qquad (19)$$

From the calculated gain, the input and feedback resistors were calculated.

$0.1042 = -R_f / R_{in}$

Let $R_{in} = 10K\Omega$

Therefore, $R_f = 10K * 0.1042$

$= 1.04K\Omega$

As shown in figure 34 below, the input voltage to the attenuating amplifier was going to be followed by a buffer because it actually appears across the load at the output of the buck converter. This was not the case because the attenuating amplifier is itself a buffer. This is so in order that R1 does not in any way upset the electrical dynamics at the buck converter output when a load is connected. This design takes the advantage that a buffer has very large input resistance and a very low output resistance. The attenuating amplifier (IC1) is an inverting amplifier (IC1, R1 & R2). Inverting amplifiers exhibits a gain of less than unity unlike a non-inverting amplifier. Finally, the negative voltage at the output of the inverting amplifier is again re-inverted to a positive voltage by a unity gain inverting amplifier (IC2, R3 & R4)

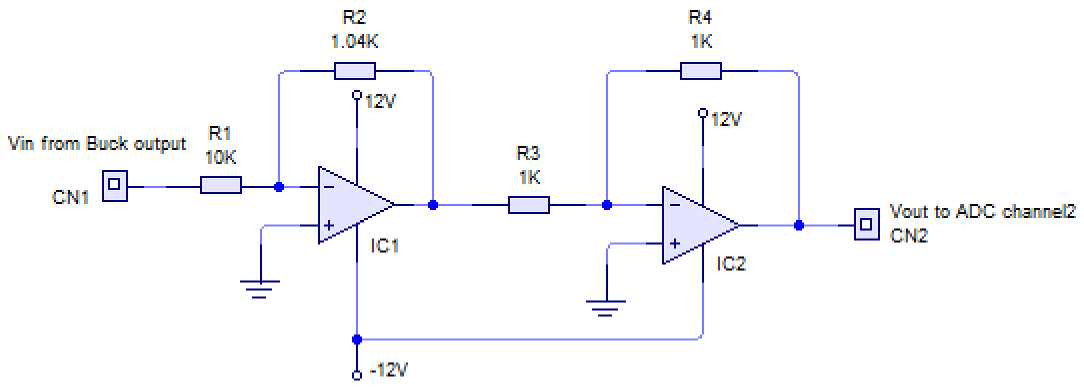


Figure 35: Output voltage feedback conditioning circuit.

The output voltage (Vout) was then ‘read’ (sampled) by the microcontroller’s analogue to digital conversion (ADC) circuits. The microcontroller’s ADC converter has a 10 bits resolution and the converted value is stored in a 16 bits register but only the eight most significant bits will be ‘read’ from the ADCH; 8 bits register.

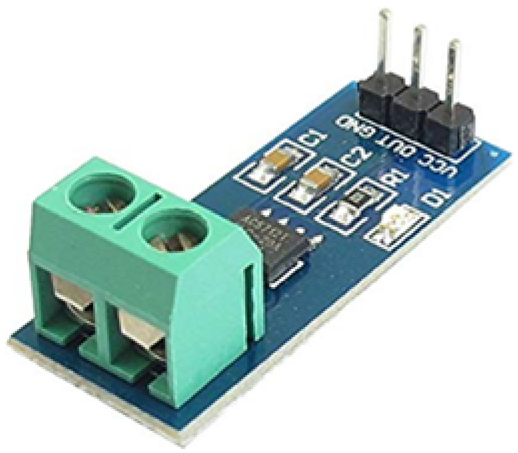
$$Converted\ Voltage = ADCH * \frac{Aref}{2^{8bits}} \quad (20)$$

The microcontroller then recalculates the actual buck converter output voltage (V_{in} to amplifier) by using the attenuating amplifier gain and the converted voltage value.

$$Vin = \frac{Converted\ voltage}{0.1042}$$

4.2. Output current feedback

The output current value was measure using an ACS712T ELC-20A current sensor. The sensor converts the current flowing throw it to voltage at its output terminal. The voltage was then 'read' (sampled) by the microcontroller's analogue to digital conversion (ADC) circuits.



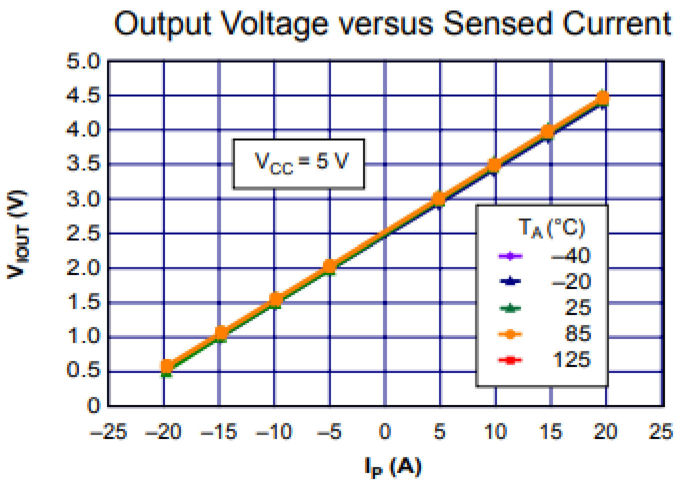


Figure 36: ACS712T Current sensor and V_out versus Sensed current curve [42].

The resolution of the current sensor is 100mV/A as stated in the datasheet. That is 100mV/1000mA. Better still, that is 0.1mV/mA.

Given that the microcontroller's resolution (sampling step size) is 19.5mV, the current sensor resolution is therefore not good for the micro controller's ADC step size or vice versa. From the current sensor's output voltage versus sensed current graph, the range of the voltage change from 0A @ 2.5V to 5A@3.0V is just 0.5V. In order to improve on this voltage range and therefore the current sensor sensitivity, a differential amplifier was designed and built with a gain of 10 such that the 0.5V range becomes 5V and equal to the Aref. This implies the 0.1mV/mA becomes 1mV/mA.

As shown on figure 36 below, the input of the buffer (IC2), at the differential amplifier's (IC1) negative input is fixed at 2.5V such that it cancels out the initial voltage of the current sensor; 2.5V @ 0A. The current sensor's output will be connected to the input of the other buffer (IC3) at the differential amplifier's (IC1) positive input. The output of the differential amplifier will therefore be at 0V when no current flows through the current sensor and be positive when there is current flow throw the sensor.

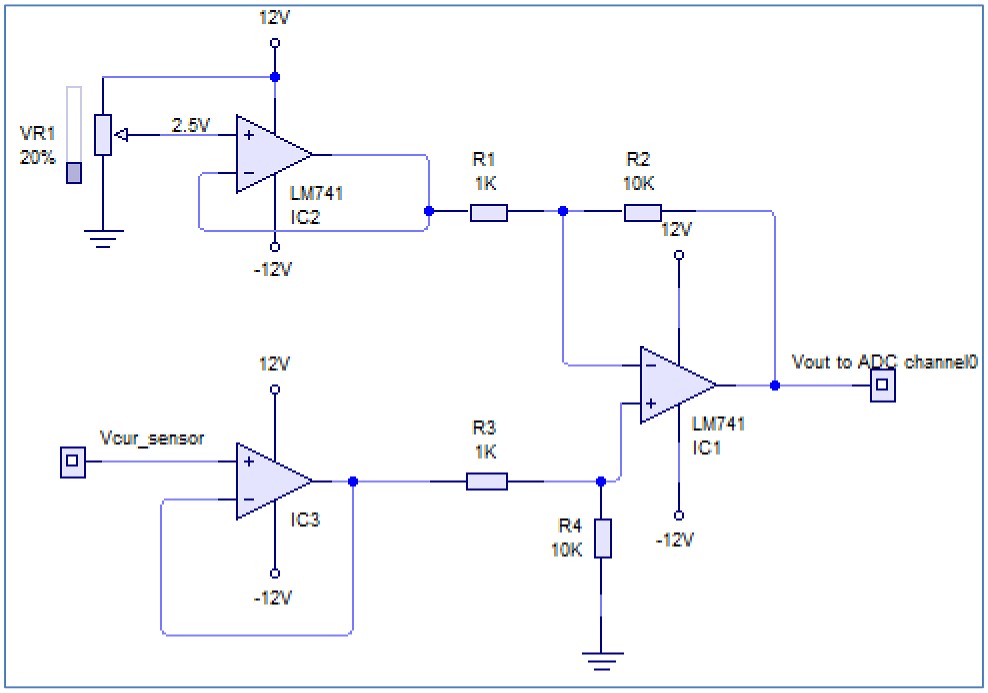


Figure 37: Differential amplifier circuit

$$Vout = \left(\frac{R_4}{R_3 + R_4}\right) * \left(\frac{R_1 + R_2}{R_1}\right) * V_2 - \left(\frac{R_2}{R_1} * V_1\right) \qquad (21)$$

$$if\ R_1 = R_3\ and\ R_2 = R_4, then\ V_{out} = \frac{R_2}{R_1}(V_2 - V_1)$$

$$therefore, V_{out} = \frac{10K}{1K}(V_{cur-sensor} - 2.5)$$

The microcontroller then reads the output voltage of the differential amplifier, and the ADC converted voltage (V_{ADC_Con}) is given by:

$$V_{ADC\_Con} = ADCH * 0.01953\text{V}$$

The current sensor voltage (V_{cur_sensor}) is given as follows:

$$Av = \frac{V_{out}}{V_{in}}\ that\ is, \qquad 10 = \frac{V_{ADC\_Con}}{V_{cur\_sensor}}$$

$$therefore, \qquad V_{cur\_sensor} = \frac{V_{ADC\_Con}}{10}$$

The current (I_{sensor}) through the sensor is calculated using the gradient equation of a straight (linear) line.

$$Gradient = \frac{\Delta y}{\Delta x} \tag{22}$$

$$\frac{\Delta y}{\Delta x} = \frac{2.5V - 3V}{0A - 5A} = 0.1$$

Therefore, at the output of the differential amplifier, the equivalent current value is calculated as follows:

$$0.1 = \frac{V_{cur\_sensor}}{0A - Isensor}.$$

$Note: V_{cur\_sensor}\ is\ the\ change\ from\ 2.5V\ due\ to\ current\ flow, thus\ the\ diff. Amplifier$

$$This\ implies, Isensor = \frac{V_{cur\_sensor}}{0.1}$$

4.3. Control flow

The microcontroller used in this project was the Atmega328P integrated circuit (IC) and the national instrument data acquisition device; USB 6009. The Arduino Uno board, based on this microcontroller IC was used to program it. Atmel Studio 6.1 integrated development environment (IDE), was used to write, develop, compile and download the embedded c++ source code into the IC.

Alternative microcontrollers were available and advance in processing speed and more PORTs such as the Arduino Mega, Arduino Due and ARM cortex 4 etc. The major disadvantage with these advance microcontrollers is that the IC is permanently attached to their respective boards. The ARM cortex 4 is especially least robust, and also has a lesser user friendly application IDE; kiel-microvision4 for source code development.

Figure 38: Arduino Uno ATmega328PU programing board.

The source code flow diagram is as shown below:

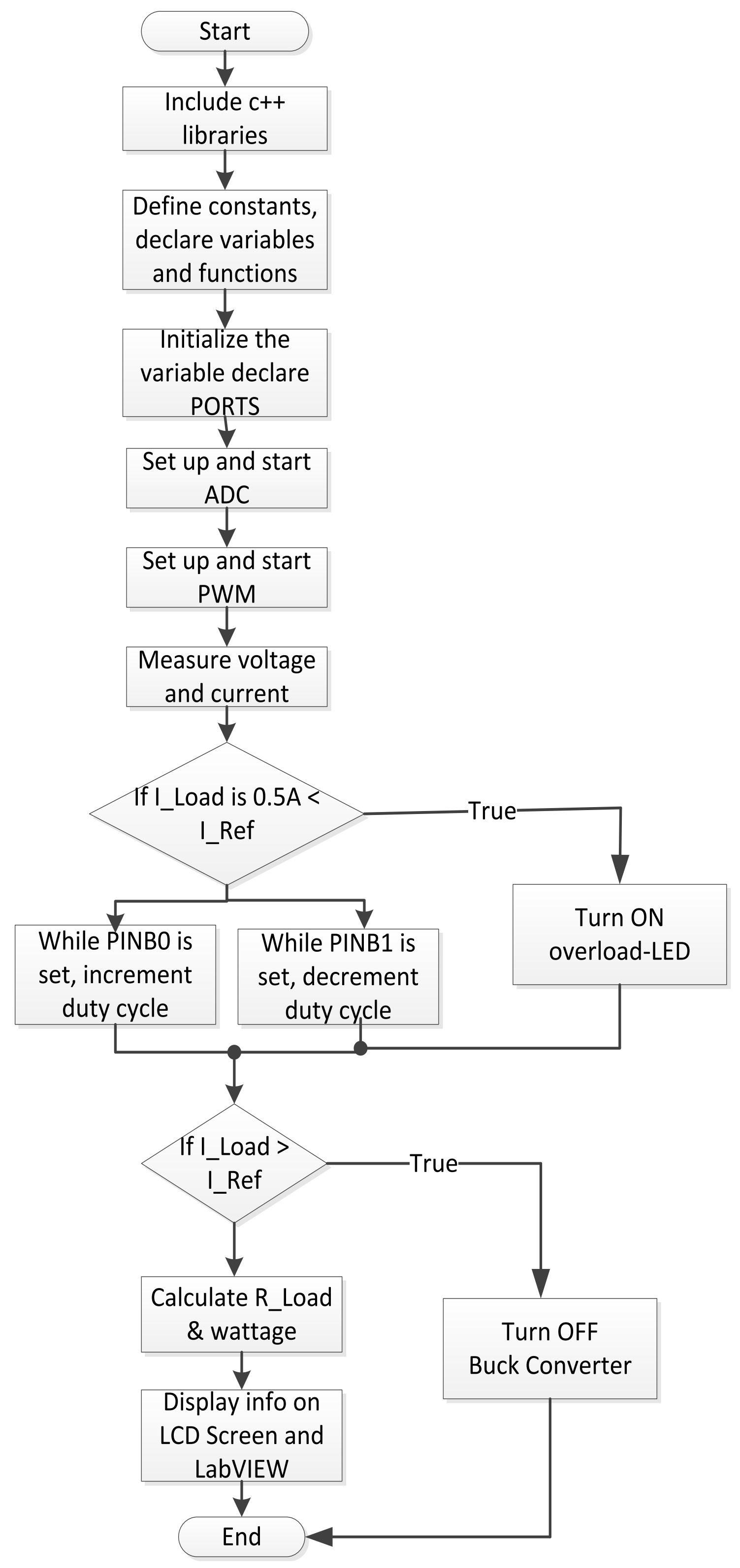

Start
Include c++ libraries
Define constants, declare variables and functions
Initialize the variable declare PORTS
Set up and start ADC
Set up and start PWM
Measure voltage and current
If I_Load is 0.5A < I_Ref
True
Turn ON overload-LED
While PINB0 is set, increment duty cycle
While PINB1 is set, decrement duty cycle
If I_Load > I_Ref
True
Turn OFF Buck Converter
Calculate R_Load & wattage
Display info on LCD Screen and LabVIEW
End

Figure 39: Microcontroller source code flow diagram

4.4. Calibration

The calibration was done using a Watt’s UP DC power meter. It was connected in series with the load. The output display circuit of the Hardware simulator was then tuned such that the displayed data matched that displayed by the power meter. This was done over a range of display values to obtain an optimal adjustment of the Hardware simulator display circuit.

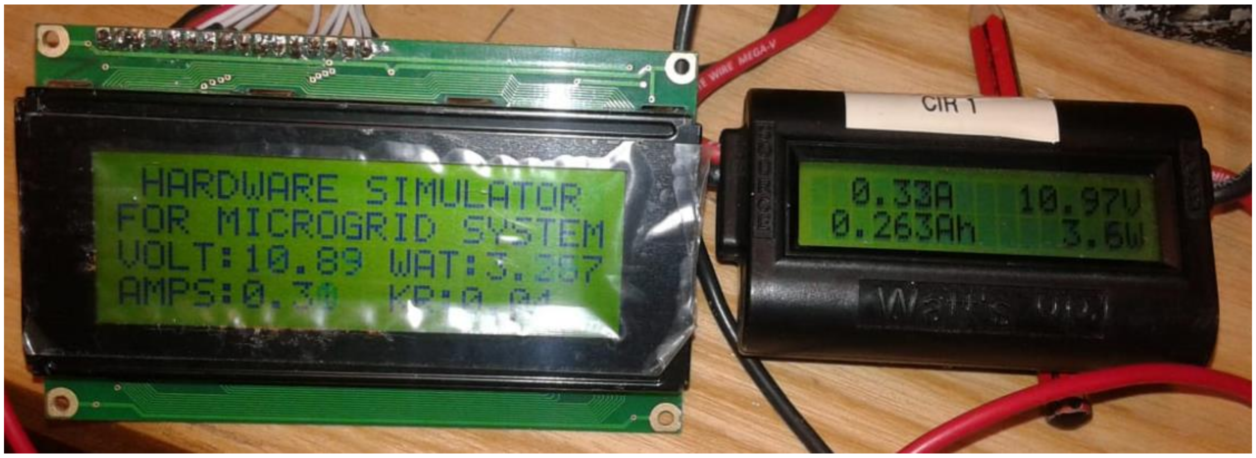


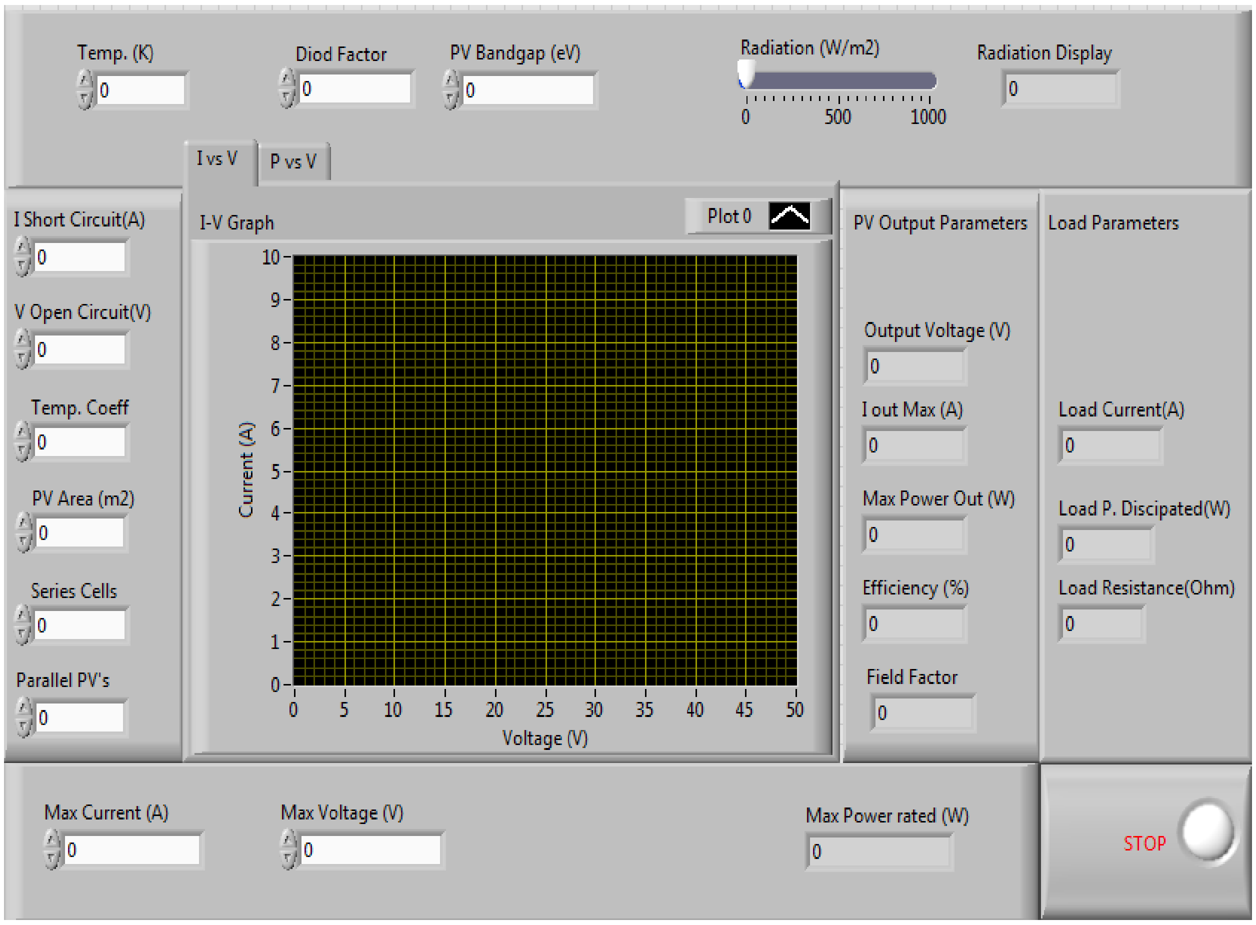
Figure 40: LCD Display calibration

5. Simulator operation

5.1. Operation Procedure

The bench-top simulator's operation starts by double clicking the GUI-file icon placed on the desktop of the monitor.

A blank GUI opens up as shown in the figure 40 below.



Temp. (K) 0

Diod Factor 0

PV Bandgap (eV) 0

Radiation Display 0

I Short Circuit(A) 0

V Open Circuit(V) 0

Temp. Coeff 0

PV Area (m2) 0

Series Cells 0

Parallel PV's 0

PV Output Parameters

Output Voltage (V) 0

I out Max (A) 0

Max Power Out (W) 0

Efficiency (%) 0

Field Factor 0

Load Parameters

Load Current(A) 0

Load P. Discipated(W) 0

Load Resistance(Ohm) 0

Max Current (A) 0

Max Voltage (V) 0

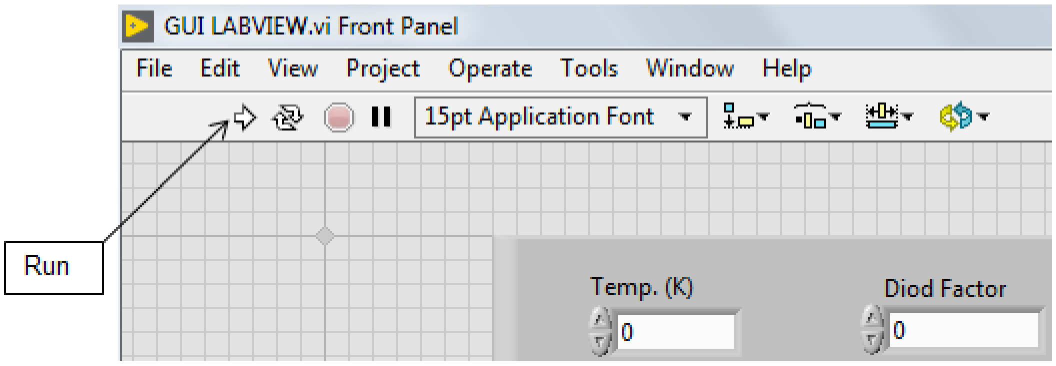
Max Power rated (W) 0

Figure 41: Blank LabVIEW GUI

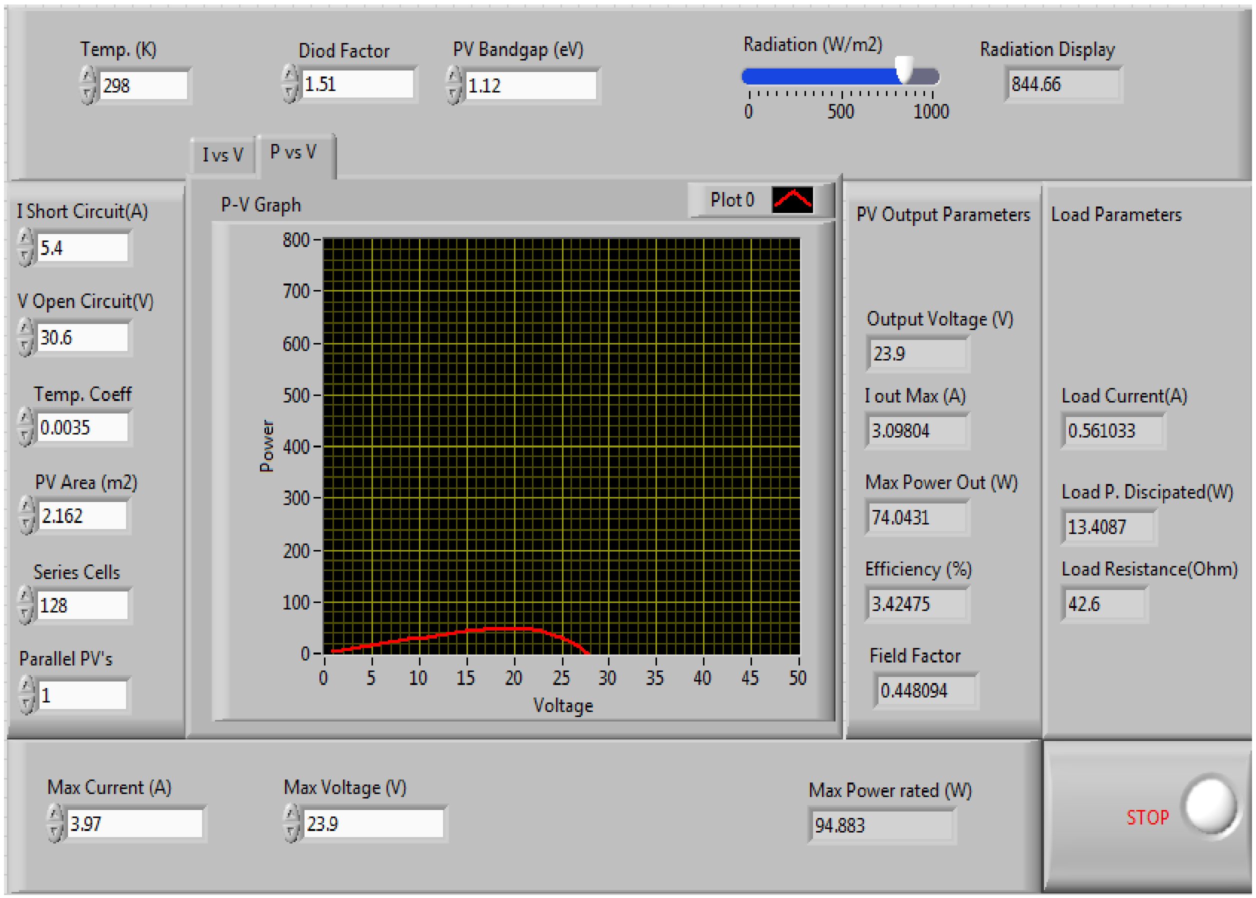
Information related to the solar panel to be simulated is entered on the white spaces, on the left side of the GIU, from the solar panel’s data sheet. The solar irradiating is adjusted by a click & hold, and then drag on the horizontal pointer slide.

The hardware of the simulator (buck converter) is pre-set to an output voltage equal to the simulated panel’s maximum voltage, by a push of the red button (increment PWM).

The thick blank arrow on the top right side of the GUI tool bar is clicked to get the simulator running.


GUI LABVIEW.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Run
Temp. (K)
0
Diod Factor
0

The performance of the simulated solar panel at the pre-set irradiation as well as the load characteristics are displayed both on the right side of the GUI (grey spaces) and, power meter (Watt’s up) and LCD display shown in figure 39 above.



Temp. (K) 298

Diod Factor 1.51

PV Bandgap (eV) 1.12

Radiation (W/m2)

Radiation Display 844.66

I Short Circuit(A) 5.4

V Open Circuit(V) 30.6

Temp. Coeff 0.0035

PV Area (m2) 2.162

Series Cells 128

Parallel PV's 1

Output Voltage (V) 23.9

I out Max (A) 3.09804

Max Power Out (W) 74.0431

Efficiency (%) 3.42475

Field Factor 0.448094

Load Current(A) 0.561033

Load P. Discipated(W) 13.4087

Load Resistance(Ohm) 42.6

Max Current (A) 3.97

Max Voltage (V) 23.9

Max Power rated (W) 94.883

Figure 42: Gui display with Power vs voltage curve

Click on the P-V Graph tabs to view either the power vs voltage or current vs voltage behaviour curves.

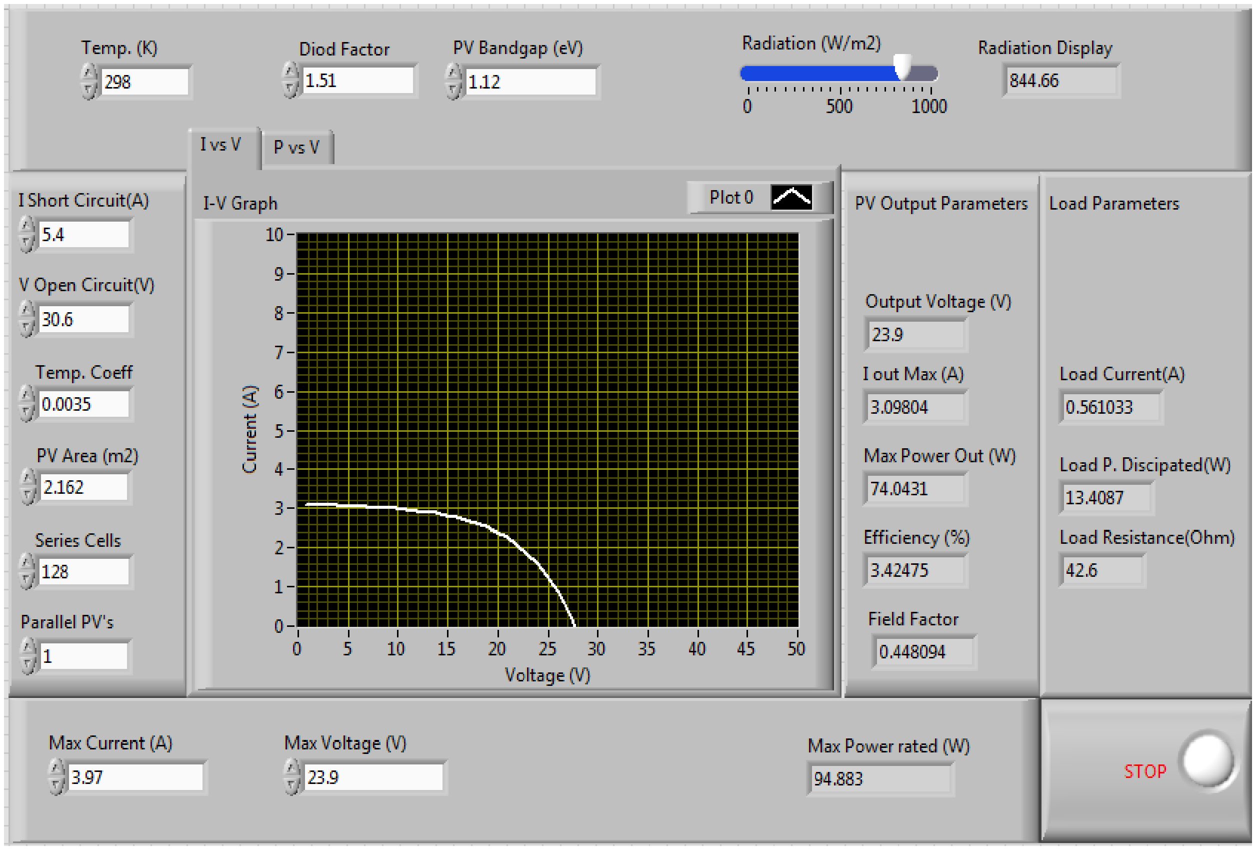

Temp. (K) 298
Diod Factor 1.51
PV Bandgap (eV) 1.12
Radiation (W/m2)
Radiation Display 844.66
I Short Circuit(A) 5.4
V Open Circuit(V) 30.6
Temp. Coeff 0.0035
PV Area (m2) 2.162
Series Cells 128
Parallel PV's 1
PV Output Parameters
Output Voltage (V) 23.9
I out Max (A) 3.09804
Max Power Out (W) 74.0431
Efficiency (%) 3.42475
Field Factor 0.448094
Load Parameters
Load Current(A) 0.561033
Load P. Discipated(W) 13.4087
Load Resistance(Ohm) 42.6
Max Current (A) 3.97
Max Voltage (V) 23.9
Max Power rated (W) 94.883

Figure 43: GUI didplay with Current vs Voltage curve

The irradiation can be adjusted to vary the hardware current limit (I-out Max). The load (potentiometer) can also be varied while view the variation of the load current on both the GUI and the power meter. Click on the stop button at the right bottom corner of the GUI to stop the GUI from running continuously.

The LED on the stop button at the right bottom corner of the GUI turns red when the load current is 0.5A less than the current limit. The simulator shots down the output voltage to 2% PWM duty when the load current exceeds the current limit. The black button (Decrease output voltage) or the red button (increase output voltage) can be pushed to reset the buck converter.

5.2. Performance Testing

Once more, multiple data sheets of different solar panels from different manufacturers were simulated and the exact results were reproduced on the GUI. Examples of solar panels are Solarex MXS-60 and MSX-64, E20/435 series solar panels. The current

/voltage characteristic of the E20/435 solar panel with dependence on irradiance and module temperature is shown in figure 45 below.

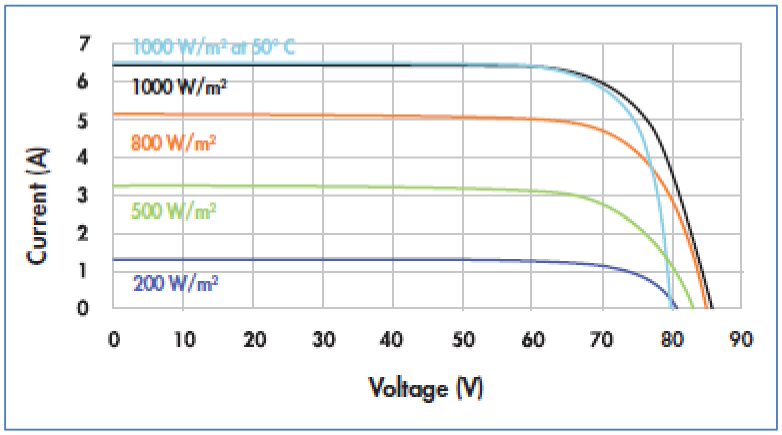


Figure 44: Solarex Solar Panel I-V curves (from data sheet) [43]

The simulator's hardware (buck converter) was loaded using resistor banks to draw high current. Since a DC buck converter tends to have low efficiency at lower voltages, it was powered at 15V_{DC}, and driven with 50KHz square signal @ 80% duty cycle and then loaded incrementally while monitoring the input and output power as shown on the table below. This process was then repeated by powering the converter at 32V_{DC} and driven by a 50KHz square signal @ 75% duty cycle.

Supply	Input current (A)	Power input (W)	Output		Power Output (W)	Efficiency (%)
			Voltage (V)	Current (A)		
15V_{DC} 50KHz Square wave @ 80% duty cycle	0.23	3,45	11.63	0.22	2,5586	74%
	0.28	4,2	11.5	0.27	3,105	74%
	0.37	5,55	11.57	0.39	4,5123	81%
	0.5	7,5	11.5	0.6	6,9	92%
	0.8	12	11.3	0.97	10,961	91%
	1.1	15	11.2	1.4	15,68	95%
	1.5	22,5	10.7	1.86	19,902	88%
	2.1	31,5	10.4	2.6	27,04	86%
	2.5	37,5	10.3	2.8	28,84	77%

	3.2	48	10	3.6	36	75%
Average Power Efficiency						**83%**
32V_{DC} Square wave @ 75% duty cycle 50KHz	0.5	16	23.9	0.6	14,34	90%
	1	32	23.58	1.26	29,7108	93%
	1.5	48	22.8	1.7	38,76	81%
	2	64	23.0	2.2	50,6	79%
	2.5	80	23.5	2.3	54,05	68%
	3	96	23.5	3.1	72,85	76%
Average Power Efficiency						**81%**

Table 6: Hardware Experimental data

The above data shows that the hardware component of the designed simulator (buck converter) is generally efficient up to 75% and above. The efficiency tends to be greater with high resistive loads where low current output is drawn and with high duty cycle signal where the output voltage is closer to input voltage. At less resistive loads, more current is drawn and so more of the power is lost in the form of heat energy dissipated in the switching power transistors hence the inefficiency. At lower duty cycle where the output voltage is lesser to the input voltage, the inductor charge time is less and thus less output current gain, hence the inefficiency.

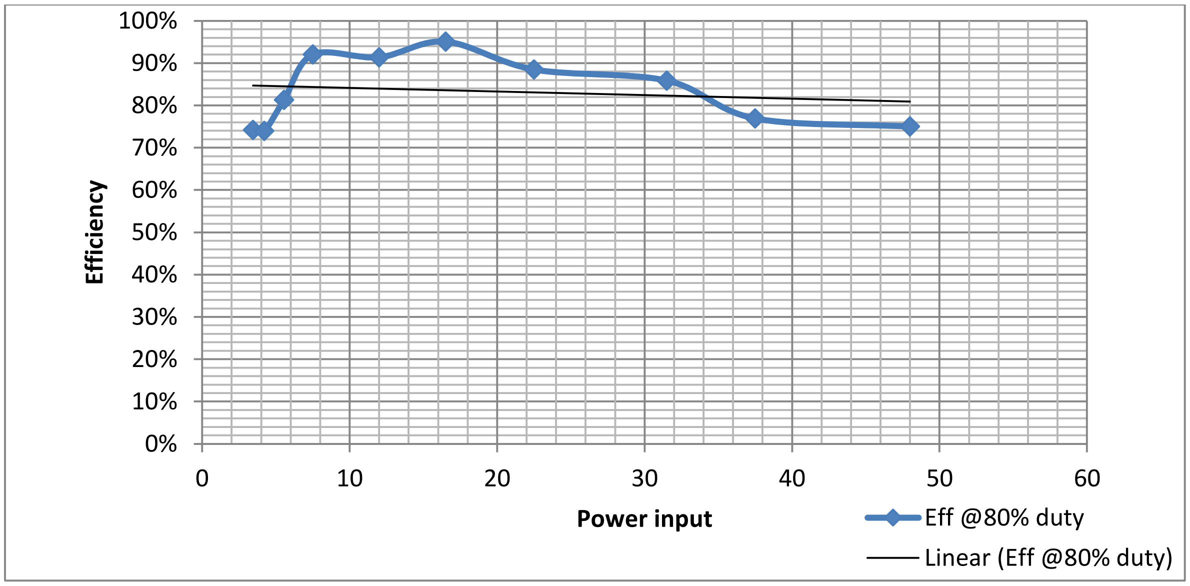


Figure 45: Efficiency @ 80% duty cycle with variable load.

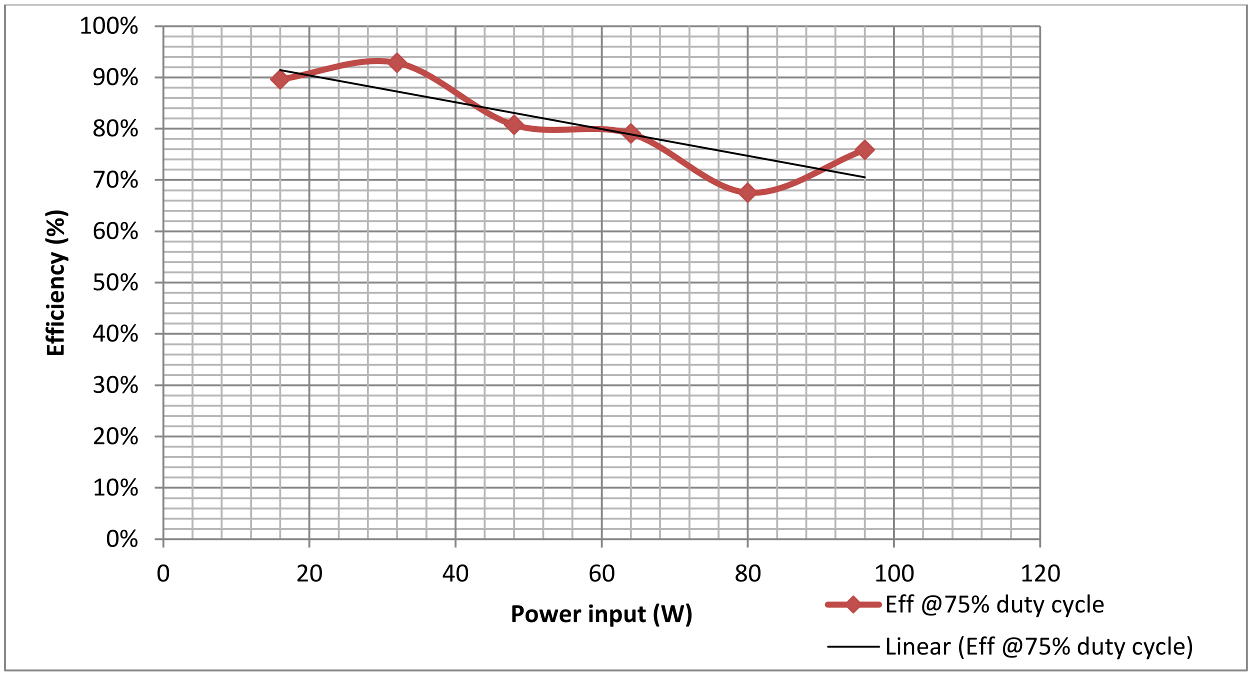
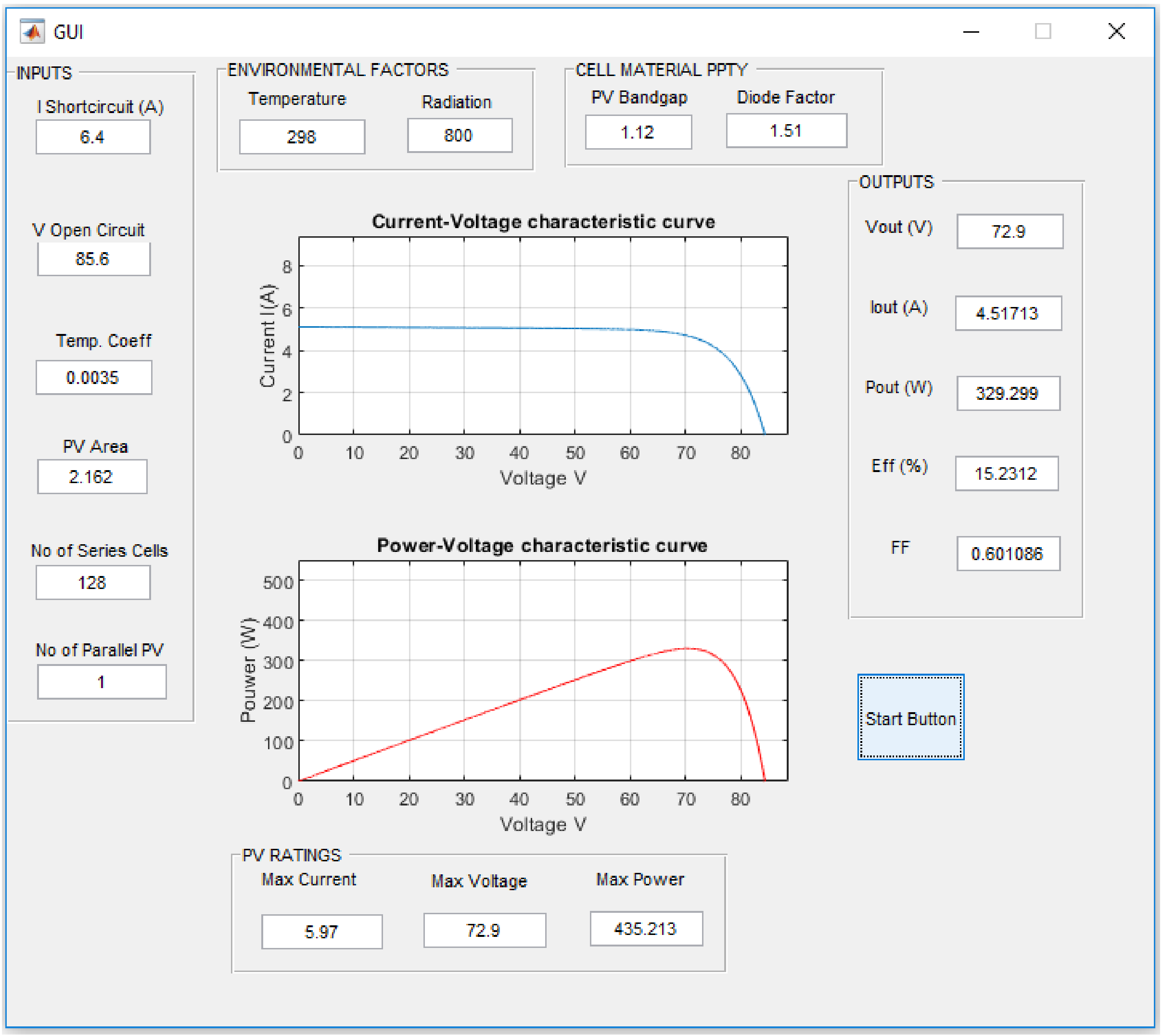


Figure 46: Efficiency @ 75% duty cycle with variable load

The difference between the inclination of the above graphs show that the converter is more efficient at higher duty cycles than at lower duty cycles. From the table, this information is show by the difference between the efficiency average values.

6. Attempted design approaches

The graphical users interface of the simulator was intended to be designed, operated and controlled from a matlab GUI as shown in figure 44 below, by exploring either the Matlab Support Package for Arduino, or NI-daq and using a serial monitor (Tera Term) by serial communication. These three alternative design approaches had significant issues.



GUI

INPUTS
I Shortcircuit (A): 6.4
V Open Circuit: 85.6
Temp. Coeff: 0.0035
PV Area: 2.162
No of Series Cells: 128
No of Parallel PV: 1

ENVIRONMENTAL FACTORS
Temperature: 298
Radiation: 800

CELL MATERIAL PPTY
PV Bandgap: 1.12
Diode Factor: 1.51

OUTPUTS
Vout (V): 72.9
Iout (A): 4.51713
Pout (W): 329.299
Eff (%): 15.2312
FF: 0.601086

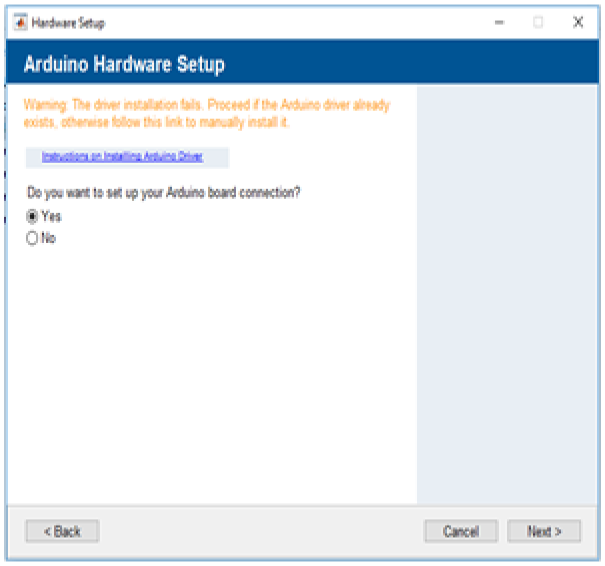
PV RATINGS
Max Current: 5.97
Max Voltage: 72.9
Max Power: 435.213

Current-Voltage characteristic curve
Current I(A)
Voltage V
Power-Voltage characteristic curve
Pouwer (W)
Voltage V
Start Button

Figure 47: simulator's Matlab GUI

6.1. Matlab Support Package for Arduino

MATLAB Support Package for Arduino Hardware makes possible a communication between Matlab (Simulink, command window, GUI) and an Arduino board, serially. This support package is functional for Matlab version R2014a and beyond. It did require registration on the Mathworks website where access to download and install the support package for the specific Arduino family (UNO, DUE, Mega, NANO) board was allowed. The registration on the Mathworks website was done and the installation done.


Hardware Setup

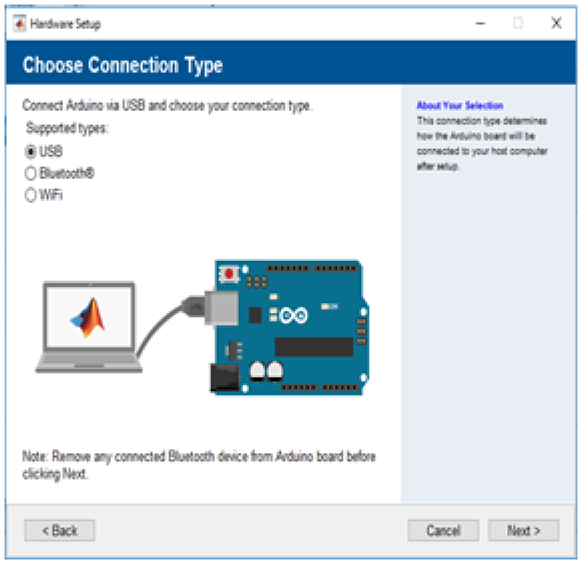
Arduino Hardware Setup

Warning: The driver installation fails. Proceed if the Arduino driver already exists, otherwise follow this link to manually install it.

Instructions on Installing Arduino Driver

Do you want to set up your Arduino board connection?

[✓ Yes] No


Hardware Setup

Choose Connection Type

Connect Arduino via USB and choose your connection type.

Supported types:

[✓ USB] Bluetooth® WiFi

Note: Remove any connected Bluetooth device from Arduino board before clicking Next.

About Your Selection

This connection type determines how the Arduino board will be connected to your host computer after setup.

Figure 48: Matlab / Arduino Hardware Setup

This setup was tested with the following matlab script:

```
global a
a = arduino('com3','uno'); % connecting to target(arduino)


DutyCycle = Vm/24;
while 1
if DutyCycle <= 0.900
    writePWMDutyCycle(a,'D3',DutyCycle);
    writeDigitalPin(a,'D9',1);
    V_sensor = readVoltage(a,'A3');
    Load_current = (V_sensor - 2.5)/0.1
    %.........maintain Iout (output current) limit
    %if Load_current > (Iout+0.5)
        while Load_current > (Iout+0.5)% +0.5 is for tolerance

            DutyCycle = Dutycycle - 0.05
            writePWMDutyCycle(a,'D3',DutyCycle);
        end
    %else
%     if
%         while Load_current > (Iout+0.5)% +0.5 is for tolerance
%
%             DutyCycle = Dutycycle - 0.05
%             writePWMDutyCycle(a,'D3',DutyCycle);
%          end
%      end
else
    writeDigitalPin(a,'D12',1);
```

It did run the arduino UNO board but had some major challenges in that the PWM hard a very low frequency of about 400 Hz. Converter operate best at higher frequencies of about 50KHz and beyond. Besides, the converter for this research was prior designed to operate at 62 KHz. The set of instruction available for use in the matlab script allows no control over the microcontroller registers and subsequently little control over the microcontroller's PWM frequency.

Interfacing the matlab GUI with an NI-data acquisition device was promising. The downloading of the NI-data acquisition support package for matlab, from mathworks website and then installing it in order to experience this technique and for matlab skill development was reserved for future research.

6.2. Serial monitor / serial communication

When the matlab support packages presented their shortcomings, the use of a keyboard via a serial monitor, to operate the simulator was adopted. The microcontroller does manage the ADC and PWM activities simultaneously. It was realized that when the serial PORT is activated by running the serial monitor (Tera Term), the ADC conversions and PWM tasks are interrupted. This implies the serial monitor as a third task cannot be manages by the microcontroller processor since the time require of the serial PORT in other to receive from the key board is not controlled by the processor but the operator. Writing of data to the serial monitor from the processor did not pose any issues with the PWM nor ADC tasks since the time required of the serial PORT, in this case is dependent on the processor's speed. This approach of operation was abandoned.

7. Conclusion and future research

A device or system that imitates well defined conditions or the behaviours of a true process or machine for the essence of research or operator training is termed simulator [33]. This device or system that simulates is commonly computer based and therefore has a desktop application referred to as a software (simulation) package. Standard software off the shelf simulation packages are PVSyst, PSIM, Homer Pro, Matlab-Simulink and so on. These software packages are programs that contain fundamental information about the systems it is designed to simulate. The fundamental information (default input parameters) is termed input data to the simulator and can be modified and added.

The aim of this project was to design a Microgrid hardware simulator that would be able to simulate different types of solar panels and wind turbines with different configuration systems. The simulator would have a photovoltaic (PV) and a wind turbine system configuration via a graphical user's interface. The simulator would then yield results of the configuration graphically. Overall, this objective was achieved in this thesis project.

The development of a hardware simulator for Microgrid systems was successfully accomplished by implementing both hardware and software designs. The hardware part of the simulator was the same as a desk top computer made up of a mouse and keyboard connected to the central processing unit (CPU) via a universal serial bus (USB) port, a monitor connected to the CPU via an high-definition multimedia interface (HDMI) or video graphic array (VGA), and an analogue output of the CPU into a power amplifier that powered up the load.

The software component consisted of the attached source code with a graphic user interface (GUI) designed, for a desk top application. Several computer architectures (Desk top PC, embedded systems, parallel systems, distributed systems etc) need different operation systems (Windows, server 2008, UNIX, Linux, Mac-OS, Novell Netware, and BSD). An operating system suitable for both desktop PC (personal computer) and desktop-application development software (Java, visual studio) was installed first. Then the desktop-application development software was installed and used to develop the source code and GUI for the simulator.

A graphical user interface based on a solar PV mathematical model was designed and a programmable switch mode DC power supply was then built to mimic the software calculation results. The software (GUI) and the hardware (buck converter) were linked via a data acquisition device, setting up a desktop hardware simulator work station as shown in appendix E, figure 55. The deliverables of this research were therefore achieved. The three attempted alternative approaches (matlab / arduino matlab-support package, matlab / National Instrument DAQ matlab-support package, and matlab /

Tera-term serial monitor) as mentioned in chapter 6 above were also expanded upon but some aspects can be undertaken as future research to refine it.

Appendices

Appendix A: Build of material

Bill of material				
Components	**Type/Discription**	**Values**	**Quantities**	**Ratings**
Resistors	4 band	10KΩ	3	0.25W
		1.5KΩ	4	0.25W
		1.2KΩ	1	0.25W
		1KΩ	6	0.25W
		100Ω	2	0.25W
		22Ω	2	0.25W
Pots	Horzintal surface mount	22KΩ	2	0.25W
		4.4KΩ	1	0.25W
		2.2KΩ	5	0.25W
		1KΩ	2	0.25W
		220Ω	1	0.25W
		100Ω	1	0.25W
Capacitors	Electrolytic	3300µf	1	63v
		2200µf	2	16V & 63V
		470µf	2	25V
		150µf	2	35V
		56µf	2	35V
	Ceramic	18pf	4	/
Intergrated circuits (IC's)	7805 regulator	/	2	Io=1A, Vo=5V
	74LS47	/	2	/
	LM393	/	2	/
	UA741	/	5	/
	IR2113	/	1	/
	Atmega328PU	/	2	/
10mm Width IC sucket	8 Pins	/	7	/
	14Pins	/	1	/
	16Pins	/	2	/
	28Pins	/	2	/
Diodes	14.2mm 7-segment Common Anode (CA)		2	30ma-70mA
	5mm LED	/	1	/
	General purpose	/	4	6.0A-If/1000V-RRM
	Zener	/	2	24V
	Fast Recovery(BV27Y)	/	3	/

Heat shrinks	1mm Diameter	/	/	/
	2mm Diameter	/	/	/
	5mm Diameter	/	/	/
	10mm Diameter	/	/	/
Heat sinks	Aluminium heatsink, 29x37.5mm x11.5 for TO66 transistor	/	2	11K/wat
	Heat sink LIP 19488 NAT 100mm	/	1	1 degree/wat
	solid aluminium heat sink for TO-220	/	2	/
Insulation washers	TO-220 Plastic washers set	/	4	/
	TO-3 Power transistor insulation set.	/	2	/
Connectors	Female Single row pin header strip	/	1	/
	Male Single row pin header strip	/	1	/
	5.08mm 2 pin terminal screw	/	16	/
	5.08mm 3 pin terminal screw	/	4	/
	4mm Red Banana plug socket	/	1	5A
	4mm black banana plug socket	/	1	5A
Transistors	IRF540	/	2	Vds=100V, Ids=20A@100degree, Ids=28A@25degree
	2N3055	/	2	Ic=15A, VCE= 60V, 115W
Inductor	Ring Iron core	170µH	1	10A
Current sensor	ACS712T ELC-20A	/	1	20A
Switches	NC Push button	/	2	/
	NO Push button	/	1	/
	Toggle DPDT, 2 position	/	2	10A
Oscillator	Quartz crystal	16M Hertz	2	/
LCD	20 columns by 4 rows	/	1	/
Traformer	Center tapped 13V/-13V, and 48V/0V	/	1	290VA

Fuse	Glass Cartridge Fuse, 5 x 20mm	/	1	5A, 250VAC
Fuse holder	20mm fuse holder	/	1	10A250VAC
DC Fan	D80SH-12		1	12V, 0.18A

Appendix B: LabVIEW block code

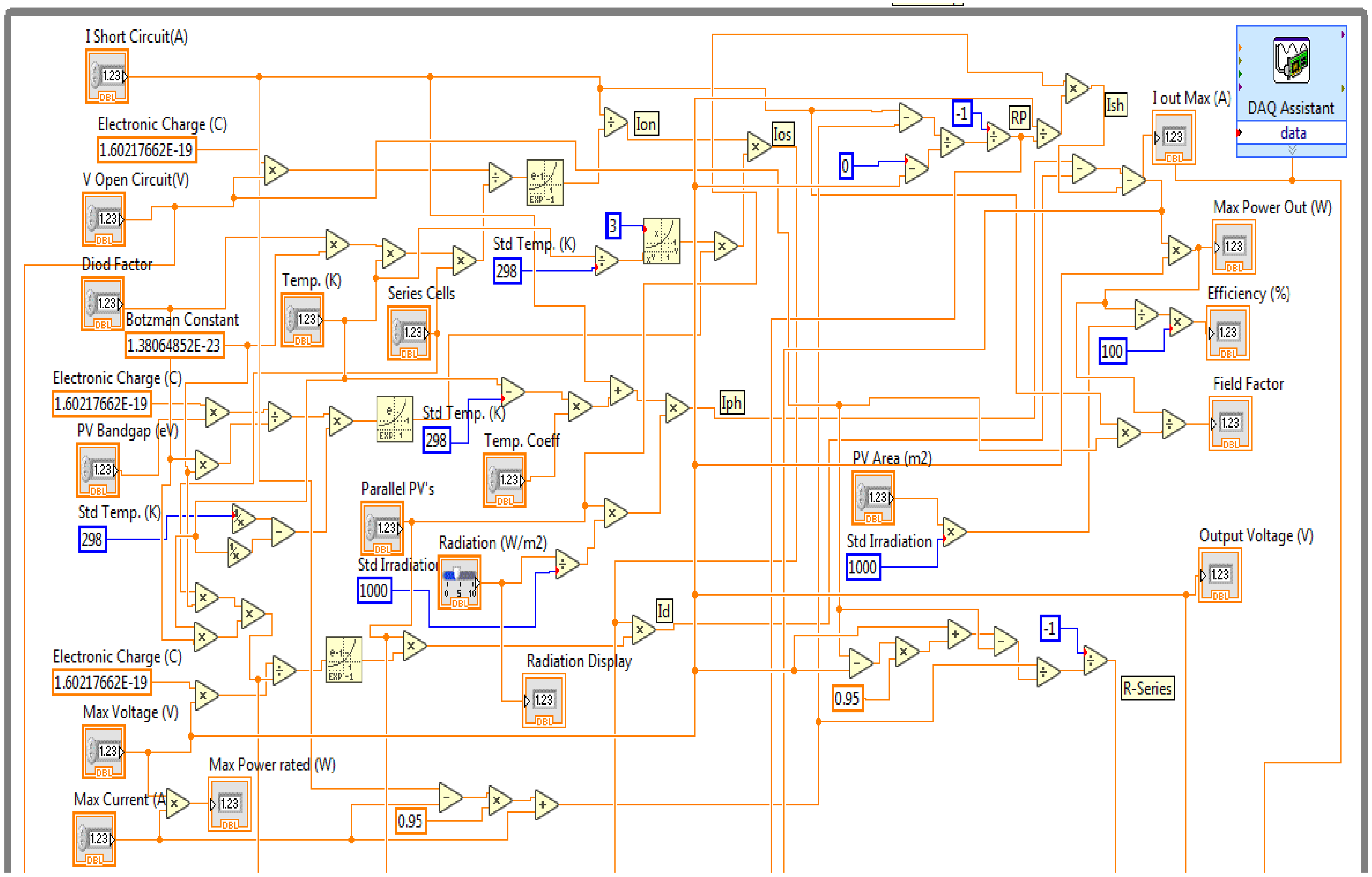


Figure 49: GUI sub VI

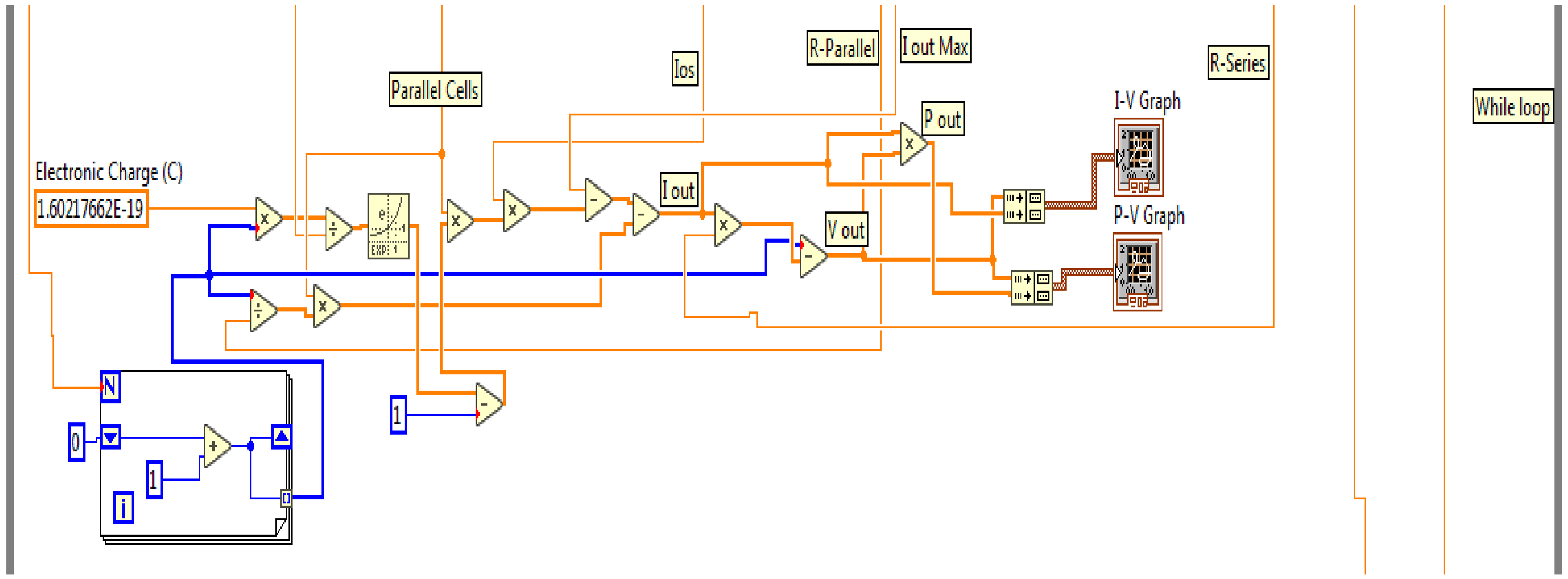


Figure 50: I-V & P-V graph ploting sub-VI

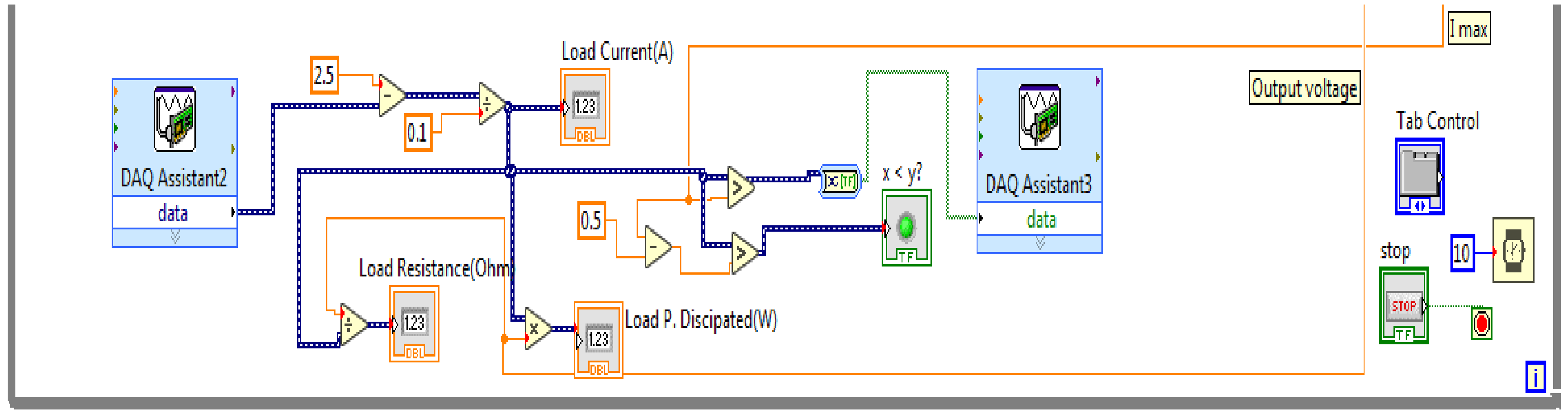


Figure 51: Load current control sub-VI

Appendix C: Arduino Uno c-code

PWM control C-code

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000L
#include <util/delay.h>
#include <stdlib.h> //itoa / atoi function header file.
#include <stdio.h> //itoa / atoi function header file.
#include <math.h>

#define USART_BAUDRATE 9600 //Declare baudrate
#define BAUD_PRESCALE (((F_CPU/(USART_BAUDRATE*16UL)))-1)//Declaring baud_prescale
//( 'BAUD_PRESCALE' is the variable which the 16bits baud prescale is assigned.

float V_C_measured, Load_resistance, V_outConverter;
float step_size, Current, Duty_cycle, V_out_C_Sensor;
float I_Max_Ref;
int n;
int main(void)
{
	unsigned char a = 0x01;
	unsigned char b = 0x02;
	unsigned char c = 0x04;


	int V_source = 23; // DC-DC convert input voltage

	DDRB = 0X78;//declare PB0-2 and PB3-6 as I/O respectively
	DDRD = 0x0F;//declare PD0-3 as output
	n = 12; //initializing duty cycle to 5% default.
	I_Max_Ref = 4; //initializing current limit(I_Max) to 500mV default
	PORTB =I_Max_Ref;
	PORTB = PORTB<<3;

	/*------------------SET UP ADC CONVERTION------------------*/

	ADMUX = 0x00;                           //ADC analog input channel (pin //0) select.
	ADMUX  |= (1<<REFS0) |(1<<ADLAR);  //Reference voltage select (5V internal) &
//Data-out Left adjustment.
	ADCSRA |= (1<<ADEN) |(1<<ADATE);   //ADC enable(turn on ADC) & //Set free running
mode respectively.
	ADCSRA |= (1<<ADPS2) |(1<<ADPS1) |(1<<ADPS0);   //ADC clock select (prescale =
//128)
	ADCSRA |= (1<<ADIE); //Enable interupt
	ADCSRA |= (1 << ADSC);              //START ADC  conversion


	/*---------------SET UP PWM--------------------------------------*/

	DDRD |= (1 << PORTD6);// PD6 is now an output
	OCR0A = n;// set(initialize) PWM for 5% (12/255)duty cycle
	TCCR0A |= (1 << COM0A1);// set none-inverting mode
	TCCR0A |= (1 << WGM01) | (1 << WGM00);//set fast PWM Mode
	TCCR0B |= (1 << CS00);// set prescaler to one(crystal freq divide by 1) and starts
PWM
```

```
	while(1)
	{
		sei(); //turn on internal interrupt

		/*....................PWM ADJUST.....................*/

		while((PINB & 0x01) == a)//permanently closed to gnd push to open button
		{
			n = n+1;
			OCR0A = n;
			_delay_ms(100);
			if (n > 242){n = 12;} //set duty cycle 5% to 95%
		}

		while((PINB & 0x02) == b)//permanently closed to gnd push to open button
		{
			n = n-1;
			OCR0A = n;
			_delay_ms(100);
			if (n < 12){n = 242;}//set duty cycle 5%(OCR0A=12) to 95%(OCR0A=242)
		}

		PORTB = I_Max_Ref;
		PORTB = PORTB<<3;
		PORTD = 10 * fmod(I_Max_Ref,1.0);


		if((PINB & 0x04) == c)//NI Daq sets 5V on PINB2 for shot circuit Protn
		{
			Duty_cycle = 2 / V_source;//duty cycle = 8% if over load
			OCR0A = (Duty_cycle * 256)-1;//generate duty cycle
		}

	}
}

ISR(ADC_vect)
{
	switch (ADMUX)
	{
		case 0x60:
		step_size = 19.53125;//0.01953125 = 19.53125mV to consider more DP's
		V_C_measured = (ADCH  * step_size)/1000; //V at output of amplifier

		V_out_C_Sensor = V_C_measured / 10; //Equivalent V at current sensor output
		Current = V_out_C_Sensor / 0.1; //current drawn equivalent to sensor V_out
		ADMUX |= 0x62;//switch to analog channel 2
		break;
		//----------------
		case 0x62:
		step_size = 19.53125;
		I_Max_Ref = (ADCH * step_size)/1000; // calculating input voltage in V(In
volts and
		ADMUX &= 0x60; //switch to analog channel 0
		break;
	}
}
```

LCD display C-code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //Microcontroller digital pins to be used by the LCD.

const int analog_input_pin_Zero = 0;  //Declaring the analog input pin (A0)

const int analog_input_pin_Two = 2;

int returned_analog_value_0, returned_analog_value_2;  //Variable to store the analog
values

float converted_analog_voltage_0, converted_analog_voltage_2,
Actual_conveter_voltage, current;

float Power, Load_resistance;

//float current;

/*----------------------------------------------------------------------*/

void setup()

{

  lcd.begin(20, 4);          //set up the LCD's numbers of columns and rows.

  //analogReference(INTERNAL); //Selects the internal 1.1V reference voltage.

  //pinMode(buzzer, OUTPUT);  //declaring the buzzer pin(Digit 13) as output.

 /*----------------------------------------------------------------------*/

 lcd.setCursor(0,0);

  lcd.print("STUDENT: E.P MULUH");

  lcd.setCursor(0,1);

  lcd.print("SUPERVISOR:PROF KAHN");

  delay(5000);

  lcd.clear();

}

/*----------------------------------------------------------------------*/
```

```
void loop()
{
  lcd.setCursor(1, 0);          //locate cursor at second column,first row.
  lcd.print("HARDWARE SIMULATOR");
  lcd.setCursor(0, 1);          //locate cursor at second column,first row.
  lcd.print("FOR MICROGRID SYSTEM");

  /*-----------------------Current drawn----------*/

  returned_analog_value_0 = analogRead(analog_input_pin_Zero);
  //Calculating the converted analog voltage(mV).
  converted_analog_voltage_0 = returned_analog_value_0 * (5.0 / 255);
  current = (converted_analog_voltage_0 / 2);
  //current = (2.6 - converted_analog_voltage_0) / 0.1;

  //current = (converted_analog_voltage_0 / 10) / 0.1 //gain=10 & grad =0.1)
        //= converted_analog_voltage_1
  lcd.setCursor(0,3);//setting cursor at (column 0, row 3)
  lcd.print("AMPS:");
  lcd.setCursor(5,3);//setting cursor at (column 5, row 3)
  lcd.print(current);
  //lcd.setCursor(11,3);
  //lcd.print("A");

 /*----------------Voltage calculation-----------------------------*/
  returned_analog_value_2 = analogRead(analog_input_pin_Two);

  converted_analog_voltage_2 = returned_analog_value_2  * (5.0 / 1023 );

  //Calculate actual converter output Voltage

  Actual_conveter_voltage = converted_analog_voltage_2 / 0.105;

  lcd.setCursor(0,2);

  lcd.print("VOLT:");

  lcd.setCursor(5,2);//setting cursor at (column 6, row 1)

  lcd.print(Actual_conveter_voltage);

  //lcd.setCursor(11,2);

  //lcd.print("V");


 /*--------------------Power Calculation----------------------------*/


 Power =  Actual_conveter_voltage * current;//P = IV

 lcd.setCursor(11,2);//setting cursor at (column 6, row 1)

 lcd.print("WAT:");

 lcd.setCursor(15,2);

 lcd.print(Power);


 /*-------------Load Resistance calculation--------------*/


 Load_resistance =  Actual_conveter_voltage / current;//V=IR

 lcd.setCursor(11,3);//setting cursor at (column 6, row 1)

 lcd.print("OHM:");
```

```
 lcd.setCursor(15,3);
 lcd.print(Load_resistance);
}
```

Appendix D: simulator's Matlab Graphical-user-interface script

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI

% Last Modified by GUIDE v2.5 06-Dec-2018 23:30:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI (see VARARGIN)

% Choose default command line output for GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function PVbandGap_Callback(hObject, eventdata, handles)
% hObject    handle to PVbandGap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of PVbandGap as text
%        str2double(get(hObject,'String')) returns contents of PVbandGap as a
double


% --- Executes during object creation, after setting all properties.
function PVbandGap_CreateFcn(hObject, eventdata, handles)
% hObject    handle to PVbandGap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function IdealityFactor_Callback(hObject, eventdata, handles)
% hObject    handle to IdealityFactor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of IdealityFactor as text
%        str2double(get(hObject,'String')) returns contents of IdealityFactor
as a double


% --- Executes during object creation, after setting all properties.
function IdealityFactor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to IdealityFactor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function AtmosphereTemp_Callback(hObject, eventdata, handles)
% hObject    handle to AtmosphereTemp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of AtmosphereTemp as text
%        str2double(get(hObject,'String')) returns contents of AtmosphereTemp
as a double


% --- Executes during object creation, after setting all properties.
function AtmosphereTemp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AtmosphereTemp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Insolation_Callback(hObject, eventdata, handles)
% hObject    handle to Insolation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Insolation as text
%        str2double(get(hObject,'String')) returns contents of Insolation as
a double
```

```
% --- Executes during object creation, after setting all properties.
function Insolation_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Insolation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MaxCurrent_Callback(hObject, eventdata, handles)
% hObject    handle to MaxCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxCurrent as text
%        str2double(get(hObject,'String')) returns contents of MaxCurrent as
a double


% --- Executes during object creation, after setting all properties.
function MaxCurrent_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MaxVoltage_Callback(hObject, eventdata, handles)
% hObject    handle to MaxVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxVoltage as text
%        str2double(get(hObject,'String')) returns contents of MaxVoltage as
a double


% --- Executes during object creation, after setting all properties.
function MaxVoltage_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to MaxVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MaxPowerRated_Callback(hObject, eventdata, handles)
% hObject    handle to MaxPowerRated (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxPowerRated as text
%        str2double(get(hObject,'String')) returns contents of MaxPowerRated
as a double


% --- Executes during object creation, after setting all properties.
function MaxPowerRated_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxPowerRated (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ShortCircuitCurrent_Callback(hObject, eventdata, handles)
% hObject    handle to ShortCircuitCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ShortCircuitCurrent as
text
%        str2double(get(hObject,'String')) returns contents of
ShortCircuitCurrent as a double


% --- Executes during object creation, after setting all properties.
function ShortCircuitCurrent_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ShortCircuitCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function OpenedCircuitVoltage_Callback(hObject, eventdata, handles)
% hObject    handle to OpenedCircuitVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of OpenedCircuitVoltage as
text
%        str2double(get(hObject,'String')) returns contents of
OpenedCircuitVoltage as a double


% --- Executes during object creation, after setting all properties.
function OpenedCircuitVoltage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to OpenedCircuitVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function TemperatureCoef_Callback(hObject, eventdata, handles)
% hObject    handle to TemperatureCoef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TemperatureCoef as text
%        str2double(get(hObject,'String')) returns contents of
TemperatureCoef as a double


% --- Executes during object creation, after setting all properties.
function TemperatureCoef_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TemperatureCoef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function SolarPVarea_Callback(hObject, eventdata, handles)
% hObject    handle to SolarPVarea (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SolarPVarea as text
%        str2double(get(hObject,'String')) returns contents of SolarPVarea as
a double


% --- Executes during object creation, after setting all properties.
function SolarPVarea_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SolarPVarea (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NumOfSeriesCells_Callback(hObject, eventdata, handles)
% hObject    handle to NumOfSeriesCells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NumOfSeriesCells as text
%        str2double(get(hObject,'String')) returns contents of
NumOfSeriesCells as a double


% --- Executes during object creation, after setting all properties.
function NumOfSeriesCells_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NumOfSeriesCells (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function NumOfParallelPV_Callback(hObject, eventdata, handles)
% hObject    handle to NumOfParallelPV (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NumOfParallelPV as text
%        str2double(get(hObject,'String')) returns contents of
NumOfParallelPV as a double


% --- Executes during object creation, after setting all properties.
function NumOfParallelPV_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NumOfParallelPV (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function OutPutVoltage_Callback(hObject, eventdata, handles)
% hObject    handle to OutPutVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of OutPutVoltage as text
%        str2double(get(hObject,'String')) returns contents of OutPutVoltage
as a double


% --- Executes during object creation, after setting all properties.
function OutPutVoltage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to OutPutVoltage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function OutputCurrent_Callback(hObject, eventdata, handles)
% hObject    handle to OutputCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of OutputCurrent as text
%        str2double(get(hObject,'String')) returns contents of OutputCurrent
as a double


% --- Executes during object creation, after setting all properties.
function OutputCurrent_CreateFcn(hObject, eventdata, handles)
% hObject    handle to OutputCurrent (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function OutputPower_Callback(hObject, eventdata, handles)
% hObject    handle to OutputPower (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of OutputPower as text
%        str2double(get(hObject,'String')) returns contents of OutputPower as
a double


% --- Executes during object creation, after setting all properties.
function OutputPower_CreateFcn(hObject, eventdata, handles)
% hObject    handle to OutputPower (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Efficiency_Callback(hObject, eventdata, handles)
% hObject    handle to Efficiency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of Efficiency as text
%        str2double(get(hObject,'String')) returns contents of Efficiency as
a double


% --- Executes during object creation, after setting all properties.
function Efficiency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Efficiency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FillFactor_Callback(hObject, eventdata, handles)
% hObject    handle to FillFactor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FillFactor as text
%        str2double(get(hObject,'String')) returns contents of FillFactor as
a double


% --- Executes during object creation, after setting all properties.
function FillFactor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FillFactor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 %.............CONSTANT VARIABLES..........%
q = 1.60217662*10^(-19); % Electronic Charge
K = 1.38064852*10^(-23); %Botzman constant
Go = 1000.0;% Standard irradiation
Tn = 298.0; % Refernce temperature
```

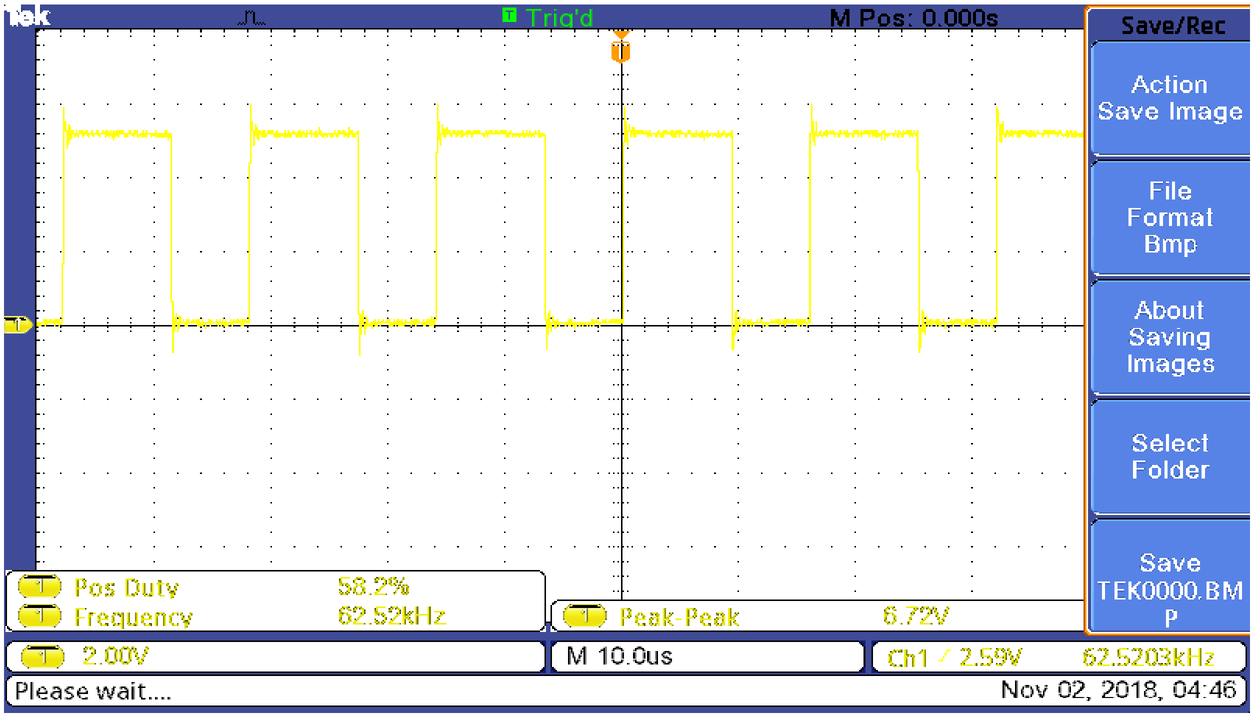
```
%..............INPUT VARIABLES..............%
Np = str2double(get(handles.NumOfParallelPV,'String'));
Ns = str2double(get(handles.NumOfSeriesCells,'String'));
DF = str2double(get(handles.IdealityFactor,'String')); % Diode ideality
factor
G = str2double(get(handles.Insolation,'String'));
T = str2double(get(handles.AtmosphereTemp,'String'));
Ki = str2double(get(handles.TemperatureCoef,'String')); % current temperature
coef
Voc = str2double(get(handles.OpenedCircuitVoltage,'String'));
Vd = 0.0:0.01:Voc;
Vm = str2double(get(handles.MaxVoltage,'String')); %MMP Voltage (Rated
Voltage)
Isc = str2double(get(handles.ShortCircuitCurrent,'String'));
Im = str2double(get(handles.MaxCurrent,'String')); %MMP current (I-rated)
Eg = str2double(get(handles.PVbandGap,'String')); %Bandgap
A =str2double(get(handles.SolarPVarea,'String'));

%% .............CALCULATIONS................%

Ion = Isc / (exp((q*Voc)/(DF*K*T*Ns))-1);
Ios = Ion*(T/Tn)^3*exp(((q*Eg)/(DF*K))*((1/Tn)-(1/T)));
Iph = Np*(G/Go)*(Isc + Ki*(T-Tn));

%% .............Shunt and Series Resistance Calcualtion.............
%Gsc = (Isc - Im)/ (0 - Vm) % Gradient at Shot-circuit condition (Isc point)
Gsc = (Isc - (Im+((Isc-Im)*0.75)))/ (0 - Vm); %accuracy
Rp = (-1)/Gsc;
%Goc = (Im - 0)/(Vm - Voc)% Gradient at opened-circuit condition (Voc point)
Goc = ((Im+((Isc-Im)*0.75)) - 0)/((Vm+(Voc-Vm)*0.75) - Voc); %accuracy
Rs = (-1)/Goc;

%% ................Iout at Max Power (or Vmax)....................
Id = Np*Ios*(exp((q*Vm)/(DF*Ns*K*T))-1);
Ish = Np*(Vm / Rp);
Iout_max = Iph - Id - Ish; % Max output current
Pout_max = Iout_max*(Vm);
Eff = (Pout_max / (1000*A))*100;
FF = Pout_max / (Voc*Isc);
Pmax_rated = Im*Vm;
%% ...............Writing results to output fieldd......

set(handles.OutputCurrent,'String',Iout_max);
set(handles.OutputPower,'String',Pout_max);
set(handles.Efficiency,'String',Eff);
set(handles.FillFactor,'String',FF);
set(handles.MaxPowerRated,'String',Pmax_rated);
set(handles.OutPutVoltage,'String',Vm);

%..........Vout & Iout variation over the panel voltgae rage (0 - Voc) for
%graphical purpose.....................
Id = Np*Ios*(exp((q*Vd)/(DF*Ns*K*T))-1);%Diode current (Vd = V+IRs)
Ish = Np*(Vd / Rp);
Iout = Iph - Id - Ish; % output current
```

```
Vout = Vd - (Iout*Rs); % output voltage
Pout = Iout.*Vout;

%% ............Ploting.............

axes(handles.axes1)
plot(Vout, Iout)
grid, xlabel('Voltage V'), ylabel('Current I(A)')
xlim([0 ((Voc)+3)]);%x-axis scale limit
ylim([0 ((Np*Isc)+3)]);%y-axis scale limit
title('Current-Voltage characteristic curve')

axes(handles.axes2)
plot(Vout, Pout, 'r');
grid, xlabel('Voltage V'), ylabel('Pouwer (W)')
xlim([0 ((Voc)+3)]);% x-axis scale limit
ylim([0 ((Np*Isc*Voc)+3)]);%y-axis scale limit
title('Power-Voltage characteristic curve')
```

Appendix E: Project Images

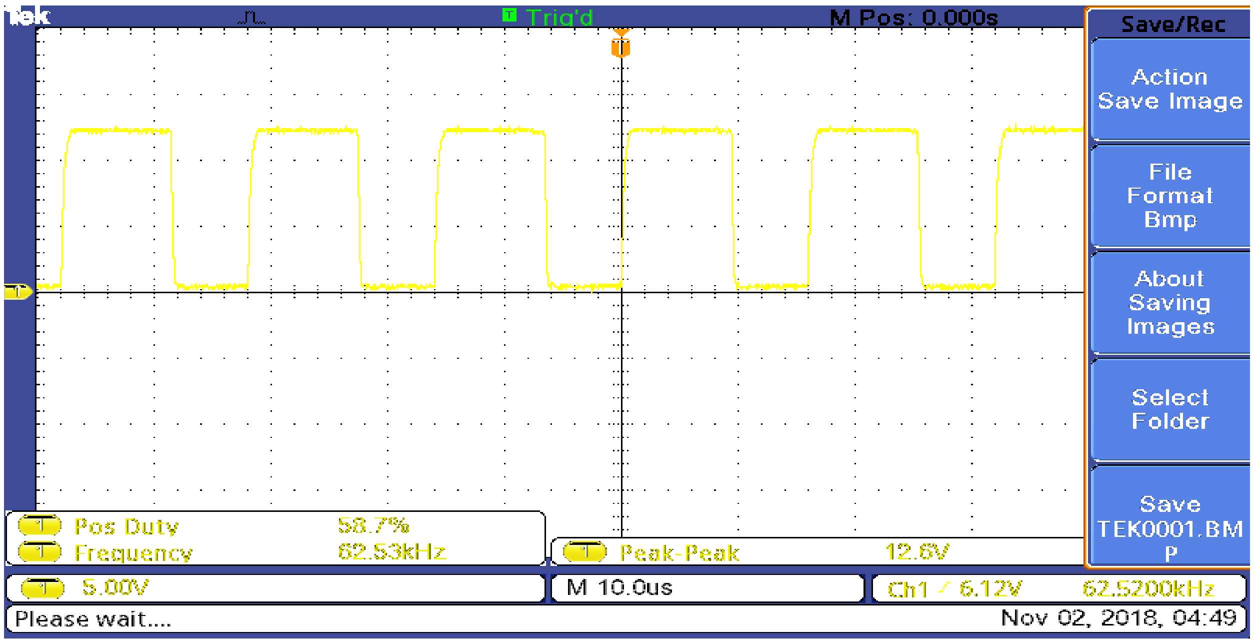
Refer to chapter 3.3.2 (PWM signal conditioning)

Oscilloscope screen shots



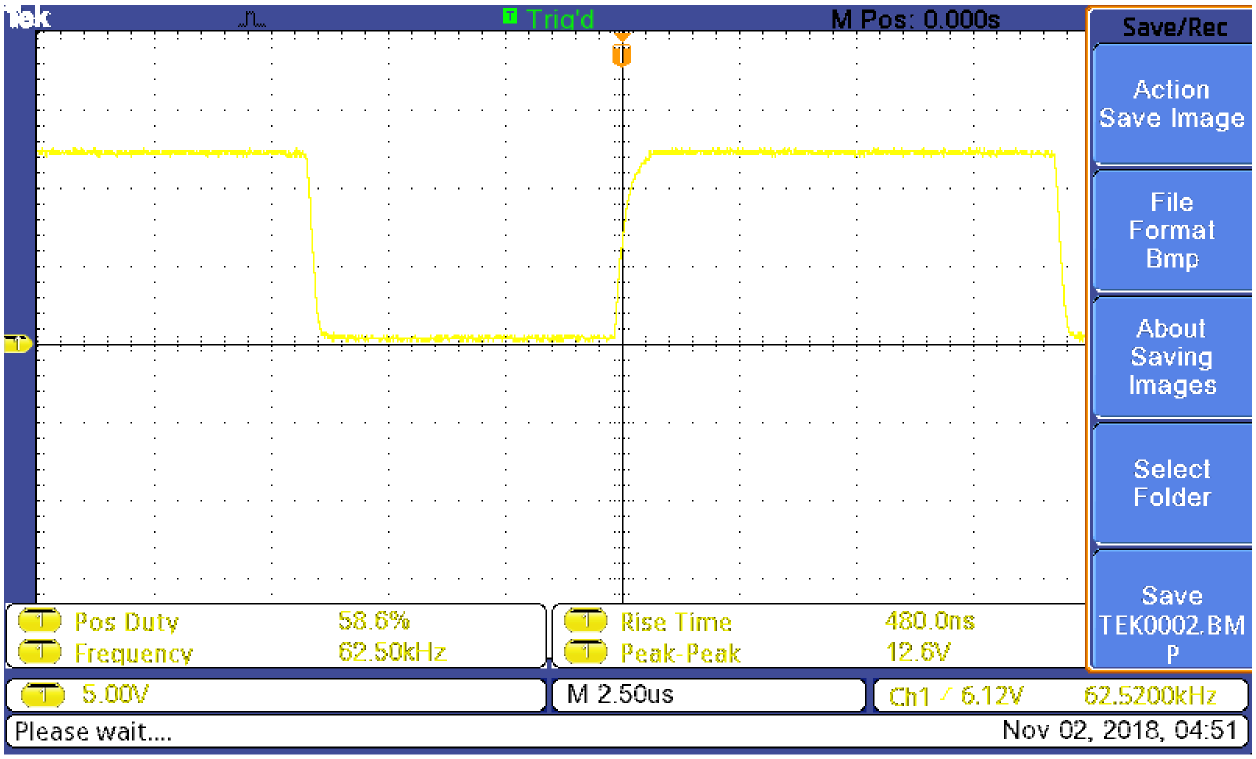
```
Pos Duty     58.2%
Frequency    62.52kHz
Peak-Peak    6.72V
2.00V        M 10.0us        Ch1 ⁄ 2.59V   62.5203kHz
Please wait....              Nov 02, 2018, 04:46
```

Figure 52: PWM signal from Arduino Uno (5V PWM)



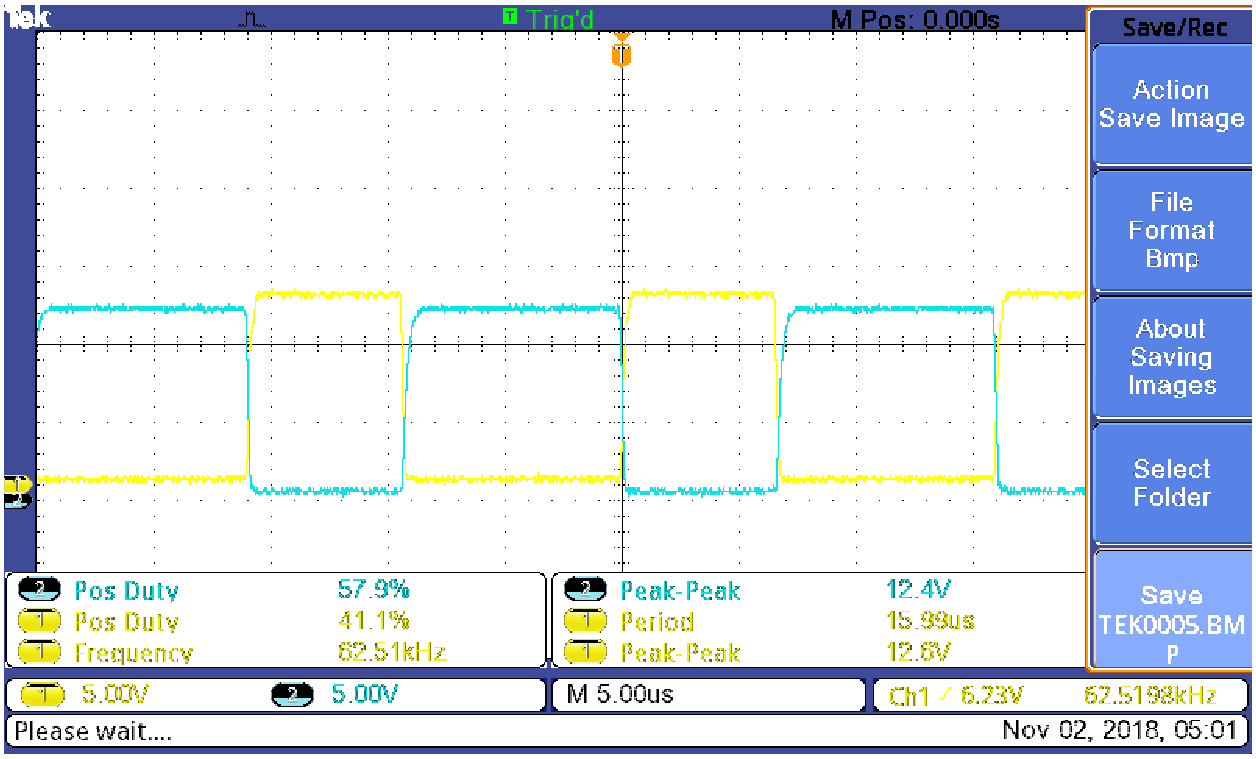
```
Pos Duty     58.7%
Frequency    62.53kHz
Peak-Peak    12.6V
5.00V        M 10.0us        Ch1 ⁄ 6.12V   62.5200kHz
Please wait....              Nov 02, 2018, 04:49
```

Figure 53: Stage-one comparator output (12V PWM)



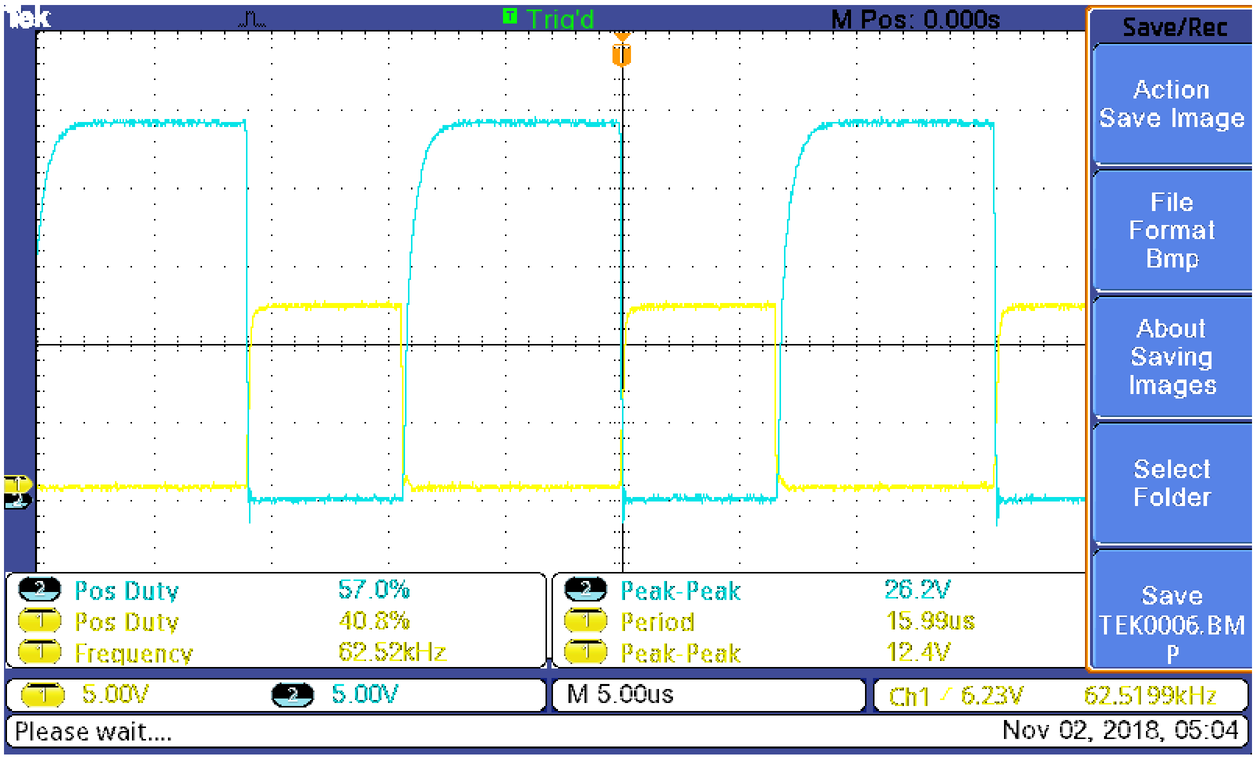
```
Pos Duty     58.6%        Rise Time    480.0ns
Frequency    62.50kHz     Peak-Peak    12.6V
5.00V                     M 2.50us     Ch1 ⁄ 6.12V   62.5200kHz
                                       Nov 02, 2018, 04:51
```

Figure 54: RC circuit output signal (12V PWM with longer rise time)



```
2 Pos Duty     57.9%        2 Peak-Peak    12.4V
1 Pos Duty     41.1%        1 Period       15.99us
1 Frequency    62.51kHz     1 Peak-Peak    12.6V
1 5.00V   2 5.00V           M 5.00us       Ch1 ⁄ 6.23V   62.5198kHz
                                           Nov 02, 2018, 05:01
```

Figure 55: Window comparator output (Two 12V-PWM 180degree out of phase)



2 Pos Duty	57.0%	2 Peak-Peak	26.2V
1 Pos Duty	40.8%	1 Period	15.99us
1 Frequency	62.52kHz	1 Peak-Peak	12.4V

```
1 5.00V   2 5.00V   M 5.00us   Ch1 ∕ 6.23V   62.5199kHz
Please wait....                      Nov 02, 2018, 05:04
```

Figure 56: MOSFET's gates PWM signal

Hardware images

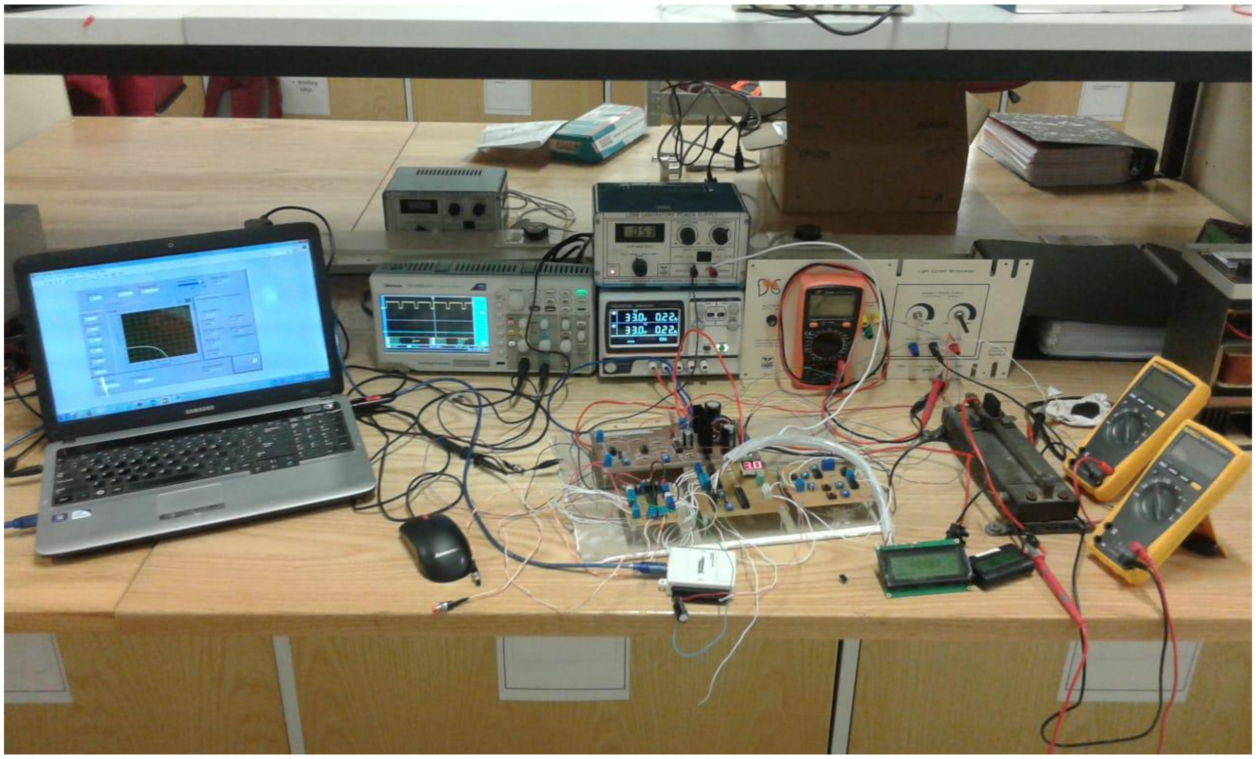


Figure 57: Completed project

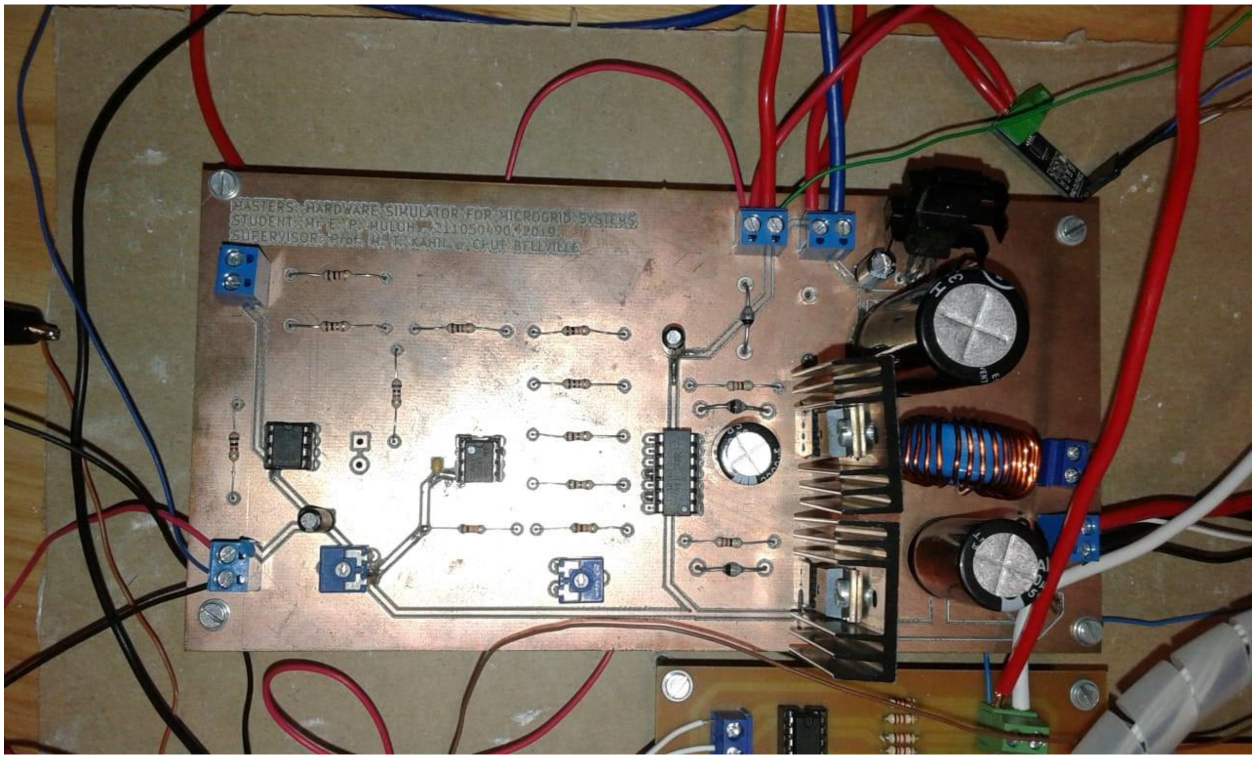

MASTERS: HARDWARE SIMULATOR FOR MICROGRID SYSTEMS
STUDENT: Mr E. P. MULUH, 211050490, 2019
SUPERVISOR: Prof. M. T. KAHN, CPUT BELLVILLE

Figure 58: Buck converter board

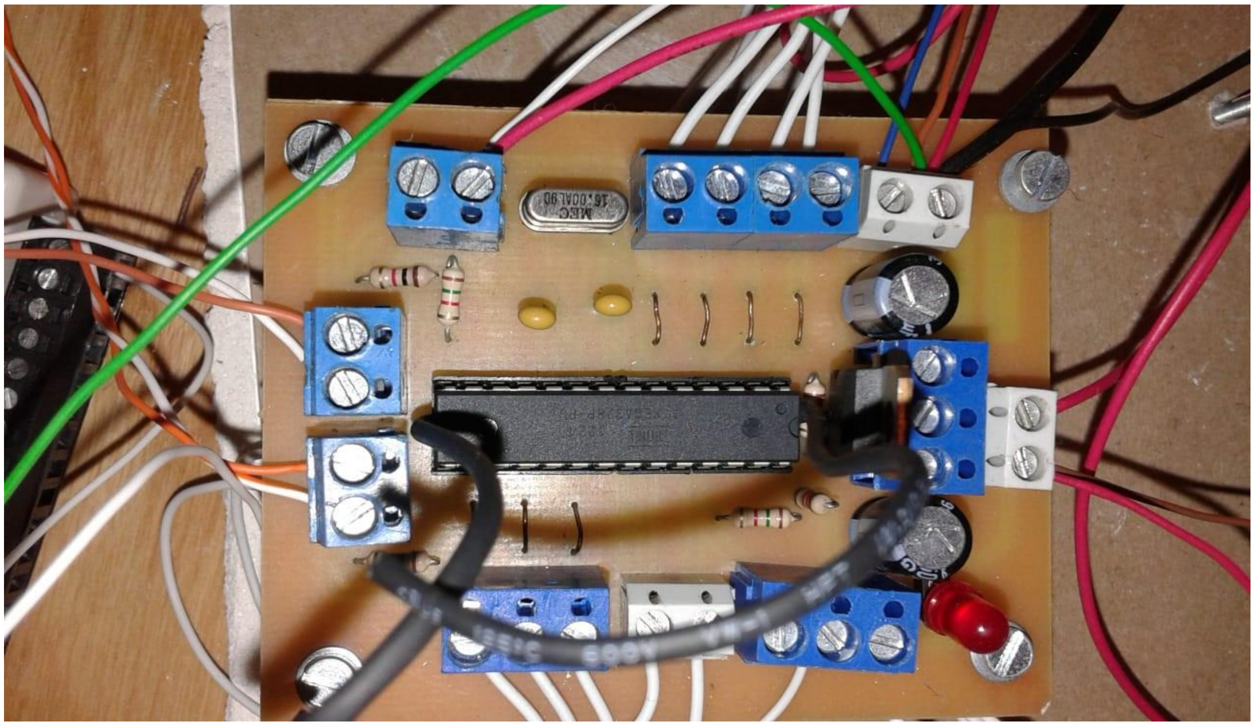


Figure 59: PWM control board

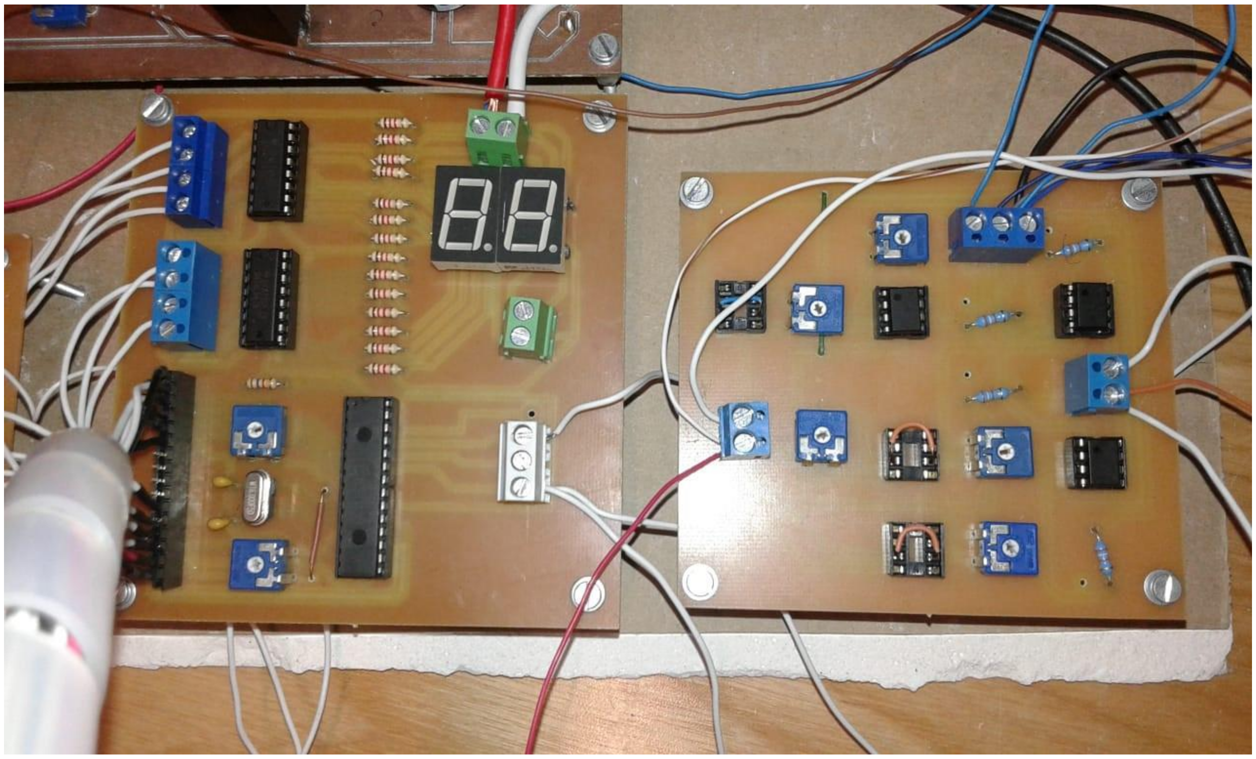


Figure 60: Display circuit boards

List of References

[1] M. Mohiti, M. Mazidi and A. A. Moghaddam, "Microgrid optimal energy and reserve scheduling considering frequency constraints," in *2019 International Conference on Smart Energy Systems and Technologies (SEST)*, 2019.

[2] B. S. Hartono, Budiyanto and R. Setiabudy, "Review of Microgrid Technology," in *2013 International Conference on QiR*, 2013.

[3] E. Wood, "microgridknowledge.com," MICROGRID KNOWLDGE, 04 November 2018. [Online]. Available: https://microgridknowledge.com/microgrid-benefits-eight/. [Accessed 24 spetember 2020].

[4] DNV KEMA Energy & Sustainability and Massachusetts clean energy center, "Microgrids – Benefits, Models, Barriers and Suggested Policy Initiatives for the Commonwealth of Massachusetts," KEMA, Inc., Burlington, February 2014.

[5] P. P. Thomas Bialek, "http://cseweb.ucsd.edu," 22 May 2013. [Online]. Available: http://cseweb.ucsd.edu/~trosing/lectures/cse291_microgrid.pdf. [Accessed 12 June 2017].

[6] D. Jardas, "www.irena-istra.hr," january 2012. [Online]. Available: http://www.irena-istra.hr/uploads/media/Photovoltaic_systems.pdf. [Accessed 12 June 2017].

[7] V. A. Subramony, S. Doolla and M. Chandorkar, "Microgrids in India," *IEEE Journals & Magazines,* vol. 5, no. 2, pp. 47 - 55, 2007.

[8] F. S. A. Iqbal, "Optimal configuration analysis for a campus microgrid—a case study," *Protection and Control of Modern Power Systems volume 2,* p. 3, 24 June 2017.

[9] K. Benmouiza, M. Tadj and A. Cheknane, "Classification of hourly solar radiation using fuzzy c-means algorithm for optimal stand-alone PV system sizing," *International Journal of Electrical Power & Energy Systems,* vol. 82, pp. 233-241, November 2016.

[10] A. M. Bagher, M. M. A. Vahid and M. Mohsen, "Types of Solar Cells and Application," *American Journal of Optics and Photonics,* vol. 3, pp. 94 -113, 2015.

[11] World of " Energy ", "https://shaikmohasin.wordpress.com," Shaik Mohasin, 13 October 2012. [Online]. Available:

https://shaikmohasin.wordpress.com/2012/10/13/explained-solar-pv-system-for-home/. [Accessed 12 June 2017].

[12] E. Elibol, Ö. T. Özmen, N. Tutkun and O. Köysal, "Outdoor performance analysis of different PV panel types," *Renewable and Sustainable Energy Reviews,* vol. 67, pp. Pages 651-661, January 2017.

[13] N. Pearsall, "Chapter 1: Introduction Photovoltaic system performance," in *The Performance of Photovoltaic (PV) systems*, Duxford, Woodhead publications, 2016, pp. 1 - 12.

[14] T. D. Lee and A. U. Ebong, "A review of thin film solar cell technologies and challenges," *Renewable and Sustainable Energy Reviews,* vol. 70, p. 1286–1297, 2017.

[15] K. Nakamura, "Current Status and Technology Trend of Crystalline Si Solar Cell," in *24th International Workshop on Active-Matrix Flatpanel Displays and Devices (AM-FPD)*, Kawasaki-shi, 2017.

[16] VIKRAM BOOKS, "Chapter12: Dual nature of radiation and mater," in *INTERMEDIATE II YEAR PHYSICS(English Medium) Question Bank*, Vikram Publishers Pvt Ltd, September 2015, pp. 350 - 358.

[17] T.-H. Wu, W.-C. Liu, C.-S. Moo, H.-L. Cheng and Yong-Nong, "An Electric Circuit Model of Photovoltaic Panel with Power Electronic Converter," in *IEEE Conferences*, 2016.

[18] Y. Chaibi, M. Salhi, A. El-jouni and A. Essadki, "A new method to extract the equivalent circuit parameters of a photovoltaic panel," *Solar Energy,* vol. 163, p. 376–386, 2018.

[19] F. Masmoudi, F. B. Salem and N. Derbel, "Single and Double Diode models for Conventional Mono-Crystalline Solar Cell with Extraction of Internal Parameters," in *IEEE Conferences*, 2016.

[20] A.-E.-H. M. Mohamed, "Efficient Approximation of Photovoltaic Model Using Dependent Thevenin Equivalent circuit based on exponential sums function," in *IEEE Conferences*, Qena, 2015.

[21] F. Rasool, M. Drieberg, N. Badruddin and B. S. M. Singh, "PV panel modeling with improved parameter extraction technique," *Solar Energy,* vol. 153, p. 519–530,

2017.

[22] V. Tamrakar, S. Gupta and Y. Sawle, "SINGLE-DIODE AND TWO-DIODE PV CELL MODELING USING MATLAB FOR STUDYING CHARACTERISTICS OF SOLAR CELL UNDER VARYING CONDITIONS," *Electrical & Computer Engineering: An International Journal (ECIJ),* vol. 4, pp. 1-11, June 2015.

[23] kavonesolar, "kavonesolar.com," kavonesolar, [Online]. Available: http://kavonesolar.com/polycrystalline.html. [Accessed 20 June 2017].

[24] S. C. CHUN, "DEVELOPMENT OF A HYBRID SOLAR WIND TURBINE FOR SUSTAINABLE ENERGY STORAGE," University of Hussein Onn Malaysia, Hussein Onn, JULY 2015.

[25] M. Hankins, Stand-alone Solar Electric Systems, Washington DC: The Earthscan Expert Handbook, 2010.

[26] Phocos North America, Inc., "www.phocos.com," 2015. [Online]. Available: http://www.phocos.com/wp-content/uploads/2015/12/Guide-Comparing-PWM-MPPT-Charge-Controllers.pdf. [Accessed 25 April 2018].

[27] M. Rokonuzzaman and M. Hossam-E-Haider, "Design and Implementation of Maximum Power Point Tracking Solar Charge Controller," in *2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, Dhaka, 2016.

[28] M. LokeshReddy, P. P. Kumar and S. A. M. Chandra, "Comparative study on charge controller techniques for solar PV system," *Energy Procedia,* vol. 117, pp. 1070-1077, June 2017.

[29] Victron Energy B.V, "www.victronenergy.com," 28 June 2014. [Online]. Available: https://www.victronenergy.com/upload/documents/White-paper-Which-solar-charge-controller-PWM-or-MPPT.pdf. [Accessed 23 April 2016].

[30] IVT_Solartechnik, "http://www.ivt-hirschau.de/IVT_Solartechnik_Prospekt2017-ENG-MAIL.pdf," March 2017. [Online]. Available: http://www.ivt-hirschau.de/IVT_Solartechnik_Prospekt2017-ENG-MAIL.pdf. [Accessed 03 May 2018].

[31] COTEK, "www.mantech.co.za," September 2015. [Online]. Available: http://www.mantech.co.za/Datasheets/Products/CX-SERIES_COTEK.pdf.

[Accessed 02 05 2018].

[32] N. Dodds, "Feasibility study of small-scale battery systems for domestic application," University of Strathclyde, UK, Strathclyde, 2015.

[33] Collins Dictionaries, Collins English Dictionary, HarperCollins Publishers, 2014.

[34] SunEdison, "Obtaining Accurate Energy Harvest Estimations From SunEdison Modules Using PVSyst v6.23 Solar Simulator," SunEdison Products Singapore Pte. Ltd, 2016.

[35] M. Little and J. Persson, "Homer training for renewable energy system modeling," Community Energy Malawi, 2016.

[36] Razman, C. Ayop, Wei and Tan, "A comprehensive review on photovoltaic emulator," *Renewable and Sustainable Energy Reviews,* vol. 80, p. 430–452, 2017.

[37] EE IIT, Kharagpur, "https://nptel.ac.in," [Online]. Available: https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Power%20Electronics/PDF/L-21(DP)(PE)%20((EE)NPTEL).pdf. [Accessed 05 06 2019].

[38] B&K Precision Power Supply Guide, Power Supply Guide, Yorba Linda: B&K Precision Corporation, 2009.

[39] E. Płaczek-Popko, "Top PV market solar cells 2016," *Opto-Electronics Review,* vol. 25, pp. 55 - 64, 2017.

[40] H. Bellia, R. Youcef and M. Fatima, "A detailed modeling of photovoltaic module using MATLAB," *NRIAG Journal of Astronomy and Geophysics,* vol. 3, no. 1, pp. 53-61, June 2014.

[41] Dr. and Taufik, "Practical design of buck converter," California Polytechnic State University, USA, California, 2018.

[42] Allegro MicroSystems; Inc., "www.sparkfun.com," [Online]. Available: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf. [Accessed 05 02 2019].

[43] SUNPOWER, "https://pdf.directindustry.com," [Online]. Available: https://pdf.directindustry.com/pdf/sunpower-corporation/e20-435-solar-panel/54792-420469.html. [Accessed 05 10 2020].

[44] learnabout-electronics, "http://www.learnabout-electronics.org," 2012. [Online]. Available: http://www.learnabout-electronics.org/Downloads/amplifiers-module-05.pdf. [Accessed 22 June 2017].

[45] N. Gupta, "http://www.sustainabilityoutlook.in," sustainabilityoutlook, 09 March 2012. [Online]. Available: http://www.sustainabilityoutlook.in/content/want-buy-solar-panel-%E2%80%93-key-things-be-considered. [Accessed 12 June 2017].