Nelson Mandela
Metropolitan
University

for tomorrow

The Design, Management and Testing of a Solar Vehicle's Energy Strategy

By

Johan Jordaan

211285978

A dissertation submitted in full satisfaction of the requirements of the degree of Master of Engineering in Mechatronics in the Faculty of Engineering, Built Environment and Information Technology of the Nelson Mandela Metropolitan University.

Promoter: Prof. Theo van Niekerk

11 April 2016

Acknowledgements

The author would like to acknowledge the following people and companies for their guidance, support and expert opinions during the course of this research:

- Prof T. van Niekerk (NMMU)
- Prof E. van Dyk (NMMU)
- Dr F. Vorster (NMMU)
- Mr R. Schultz (NMMU)
- Ms J. Crozier (NMMU)
- Volkswagen Group of South Africa
- J&J Electronics/Microcare

Many thanks to my family for their love and support:

- Mr L.H. Jordaan (Father)
- Ms D. Jordaan (Mother)
- Miss D. Jordaan (Sister)
- Ms C. Jordaan (Wife)

Abstract

In recent years the interest in implementing solar energy on vehicles (electrical and hybrid) has grown significantly [1]. There are currently limitations in this sector, such as the low energy density (efficiency of conversion) of this source, but it is still a renewable resource and as such, there is a growing interest [1].

A “smart” energy strategy implemented on a solar/electrical vehicle, in order to increase its energy harvesting volume, could enhance the growth of this sector.

A tracking algorithm for a solar vehicle’s MPPT (Maximum Power Point Tracker) can be designed to source solar energy very effectively and to increase the speed of finding (tracking) this optimal sourcing point (solar panel voltage and current).

Even though there are many different MPPT algorithms, it was decided that most of them were designed for stationary MPPT applications and the dynamics of implementing a MPPT on a vehicle create some unique scenarios. These include:

- Shadow flicker. This is rhythmic, rapid moving shadows across a solar panel, such as shadows from a line of trees.
- Rapid changes in solar panel orientation due to the road surface/relief.
- Rapid changes in panel temperature due to the location of the vehicle.

The aim of the research can be divided into three outcomes:

1 Creating a “Smart” energy strategy/control,
2 Implement the new control system on a solar vehicle’s MPPT, and
3 Harvesting maximum energy from solar panels using the new energy strategy.

The term “smart” is used to indicate the ability of the MPPT algorithm to be updated and improved based on previous results.

A MPPT and scaled solar vehicle is designed and manufactured in order to test the MPPT algorithm. The purpose of using a self-developed experimental setup is to have more control over the system variables as well as having the maximum freedom in setting up the system parameters.

This research is completed with a detailed report about the integration and testing of this energy strategy, as well as conclusions and future recommendations.

Table of Contents

List of Figures

Nomenclature

List of Abbreviations

MPPT – Maximum Power Point Tracker

MPP – Maximum Power Point

AI – Artificial Intelligence

P&O – Perturb and Observe

PV – Photovoltaic

NMMU – Nelson Mandela Metropolitan University

DC-DC – Direct Current to Direct Current (Voltage conversion)

PLC – Programmable Logic Controller

PVP – Photovoltaic Panel

OD – Outside Diameter

ID – Inside Diameter

RMS – Route Mean Square

PWM – Pulse Width Modulation

MOSFET – Metal-oxide-semiconductor Field-effect Transistor

NMOS – N-channel Metal-oxide-semiconductor

PMOS – P-channel Metal-oxide-semiconductor

ADC – Analogue-to-digital Converter

I/O – Input/output

LDR – Light Dependant Resistor

IC – Integrated Circuit

ANFIS – Adaptive-network-based Fuzzy Inference System

IDE – Integrated Development Environment

CSV – Comma Separated Variable

lux – luminous flux

List of Symbols

P_{PVout} - Power delivered by the PV panel

V_{PVout} - Output voltage produced by the PV panel

I_{PVout} - Output current drawn from the PV panel

$A_{cells} - Total\ surface\ area\ of\ PV\ cells\ on\ a\ car's\ roof$

$i_{ave} - Average\ irradiance\ from\ the\ sun$

$e_{cells} - Efficiency\ of\ PV\ cells$

$P_{solar} - Solar\ power\ converted\ to\ electricity$

V_{MPPmax} – Maximum MPP output voltage of the solar panel (Input to the system)

$V_{MPPnominal}$ – Nominal MPP output voltage of the solar panel (Input to the system)

μ_{TV} – Temperature coefficient of the solar panel voltage

$T_{MINactual}$ – Actual PV cell temperature

$T_{nominal}$ – Nominal PV cell temperature

T_s - Switching transistor

L_p - Power inductor

C_{out} - Output capacitor

C_{in} - Input capacitor

D_r - Rectifying diode

δ_{min} - Minimum duty cycle of the step-down DC-DC converter transistor switch

V_{outMin} - Minimum output voltage of the converter (V)

$V_{mpptMax}$ - Maximum output voltage of the solar panels (V)

I_{outMax} - Maximum output current of the step-down converter circuit (A)

P_{max} - Maximum power transferred by the converter (W)

I_{ppMax} - Maximum peak-to-peak current ripple of the inductor L_p (A)

L_p - Chosen inductance of the inductor (H)

F_{swMin} - Minimum switching frequency of the transistor T_s (Hz)

I_{pk} - The peak current that inductor L_p must accommodate (A)

L_p - The inductance of L_p as selected for trial and error method of comparing the minimum saturation current (H)

$\Delta V_{outMaxPP}$ – Maximum output ripple voltage of the step-down converter (V)

C_{out} – Output capacitance of C_{out} as selected to keep the output ripple voltage within the specification of the design process (conditional on the performance required from the converter) (F)

R - Resistance

T_{pv} – PV cell temperature

G_{solar} - Incident solar irradiation on panels

d_{MPPT} - Duty cycle of the MPPT

E_d – Output error of control system

Chapter 1 - Introduction

Solar technology has gained importance over the last few years as an electricity source [2]. Worldwide photovoltaic peak power capacity has increased by almost 70% between 2011 and 2012 [2]. The reasons are the improvement in technology, which reduces cost and increases efficiency, as well as the increase in fossil fuel electricity costs [3].

In recent years the interest in implementing solar energy on vehicles (electrical and hybrid) has grown significantly [1]. Even though there are currently serious limitations in this sector, such as low energy density (efficiency of conversion) of this source, it is still a renewable resource and as such, there is a growing interest [1].

In light of the above interests in implementing solar energy systems, specifically in the automotive sector, it is with a view to the future that it was decided to conduct research in this sector. A “smart” energy strategy implemented on a vehicle, in order to increase its energy harvesting volume, could enhance the growth of this sector.

1.1 Aim and Objectives

The main aim of this research is: To create a “Smart” energy strategy/control system, implemented on a solar vehicle’s maximum power point tracker (MPPT), in order to harvest the maximum energy from its solar panels.

This aim can be divided into three research objectives:

4 “Smart” energy strategy/control,
5 Implement on solar vehicle’s MPPT, and
6 Harvest maximum energy from solar panels.

An Artificial Intelligence (AI) MPPT control algorithm was chosen to meet the first objective of creating a “smart” energy strategy for a solar vehicle. Another important aspect of this energy strategy is to have a learning algorithm to update the parameters of the control algorithm.

It was decided to implement the control algorithm on a MPPT of a scaled model of a solar vehicle. For the purpose of having more control over the design parameters, as well as factoring out possible unwanted system variables, the MPPT and solar vehicle are designed and manufactured as part of the research. The implementation of the MPPT on a solar vehicle in turn necessitates the ability of the control algorithm to adapt to rapid changes in solar irradiation and temperatures. This is due to the presence of shadow flicker and rapid changes in solar panel orientation, along with varying temperatures due to the vehicle's location.

Harvesting the maximum power from the solar panels in a "smart" way requires that the MPPT control algorithm should be able to anticipate the maximum power point (MPP) or the optimal operation point of the panels. This anticipation is dependent on the ability of the MPPT to use external variables, such as solar irradiation and panel temperature, to calculate a result.
Figure 1-1 illustrates a summary of the research objectives.

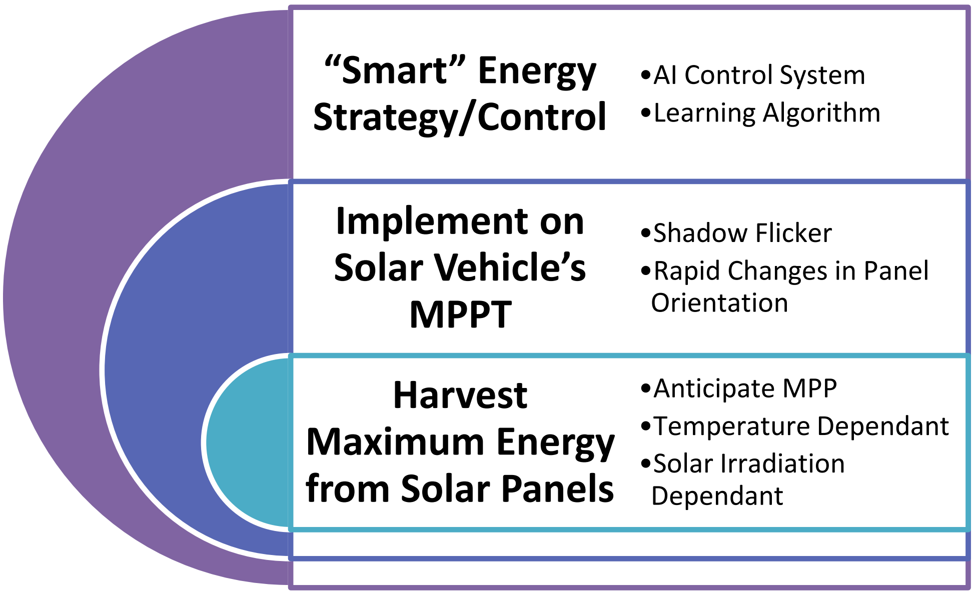

"Smart" Energy Strategy/Control
•AI Control System
•Learning Algorithm
Implement on Solar Vehicle's MPPT
•Shadow Flicker
•Rapid Changes in Panel Orientation
Harvest Maximum Energy from Solar Panels
•Anticipate MPP
•Temperature Dependant
•Solar Irradiation Dependant

Figure 1-1: Summary of research objectives

1.2 Hypothesis

A tracking algorithm for a solar vehicle's MPPT can be designed to source solar energy very effectively and to increase the speed of finding (tracking) this optimal sourcing point (solar panel voltage and current).

1.3 Delimitations

This project is limited to solar vehicles. This does however include electric vehicles with photovoltaic (PV) panels. The distinction is based on the fact that solar vehicles are more depended on the energy harvested from their PV panels, as opposed to electric vehicles that only use solar energy for low volume energy top-up.

The objective of the research is not to create a production ready MPPT, but rather to produce a MPPT control algorithm proof of concept. This algorithm is developed using smart technology or a basic form of artificial intelligence. The result of the research is to highlight the positive and negative aspects of the proposed algorithm, based on experimental results.

The MPPT is tested as part of a very simplistic electrical vehicle system. The purpose is to create a load for the MPPT as well as movement to change the orientation of the PV panels and to create shadow flicker.

The solar vehicle model is a scaled down simplistic version of a full sized vehicle. This still falls within the scope of the research, while greatly simplifying the project.

Only two solar panel units, connected in parallel, are used to investigate the impact of partial shadowing of the panels. Normally a vehicle should have more PV parallel/serial units and even more than one MPPT, however the same principle applies and two parallel PV units fall within the scope of this project.

The range of temperatures of the PV panels during testing is limited to a range of 20°C - 60°C.

The solar irradiation is not measured by the MPPT, but rather the light intensity, measured in lux, limited to a range of 12,000lx – 120,000lx. This was decided to simplify the measurement sensor needed. Conversion from solar irradiation to lux is not straight forward, because the energy per wave length is to be determined. However, there is a good estimation whereby the lux value is multiplied by 0.0079 in order to get the solar irradiation value, based on 50% photosynthetic active radiation of total solar radiation (Effectively half of the solar radiation is in the wavelength range of 400nm – 700nm) [4].

The vehicle was only tested in cold to hot dry atmospheric conditions.

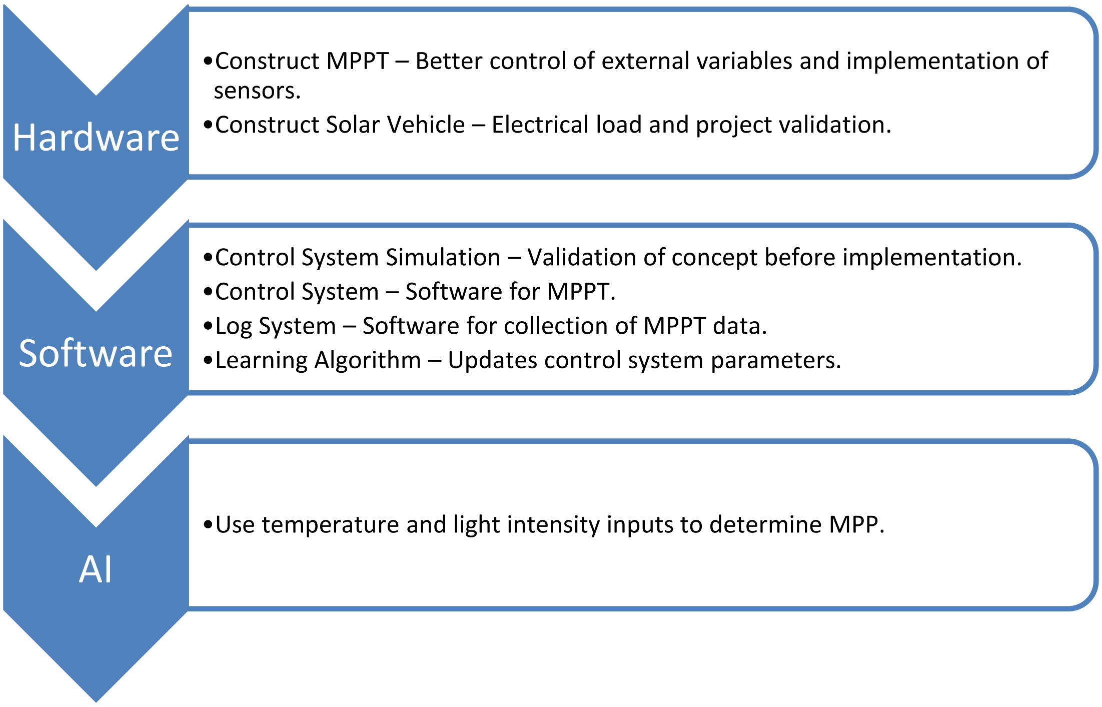
The vehicle was only tested on one altitude, latitude and longitude.

1.4 Significance of Research

The significance of the research is divided into three fields for the energy strategy requirements:

- Hardware development
- Software development
- Development of artificial intelligence

This breakdown of the energy strategy requirements is detailed in Figure 1-2.



Hardware

- Construct MPPT – Better control of external variables and implementation of sensors.
- Construct Solar Vehicle – Electrical load and project validation.

Software

- Control System Simulation – Validation of concept before implementation.
- Control System – Software for MPPT.
- Log System – Software for collection of MPPT data.
- Learning Algorithm – Updates control system parameters.

AI

- Use temperature and light intensity inputs to determine MPP.

Figure 1-2: Energy Strategy Requirements

The above energy strategy requirements add significance to MPPT development in general, with the following points highlighted:

- Implementation of artificial intelligence on a MPPT,
- The design and functioning of a MPPT, and
- The development of MPPT software.

The main field of impact is the use of MPPT modules in vehicles. This contribution adds insight to some of the technologies necessary to realize the growth of commercialized solar vehicles.

1.5 Organization of Dissertation

Chapter 2 elaborates more on the motivation behind using solar technology as an electricity source, and also highlights significance of this research. The focus of this research is on the application of solar energy in vehicles specifically; however similar strategies apply to stationary solar installations.

This is followed by Chapters 3 to 5 in which the research done to obtain the objective specified is documented. This includes the MPPT design and manufacturing, production of a solar vehicle system as well as the development and implementation of an energy strategy in the form of a smart MPPT control algorithm.

Chapter 6 details the integration and testing of the MPPT and control algorithm, along with the accompanying results.

This dissertation ends with Chapter 7, which contains all the conclusions following the research.

This research project was conducted by Mr J. Jordaan towards his MEng Mechatronics degree. The project was promoted and executed under the guidance of Prof T. van Niekerk of the Nelson Mandela Metropolitan University (NMMU).

1.6 Conclusion: Chapter 1

The main aim of this research is: To create a “Smart” energy strategy/control system, implemented on a solar vehicle’s maximum power point tracker (MPPT), in order to harvest the maximum energy from its solar panels.

The objectives to meet the project aim were set and a hypothesis formulated.

The delimitation and significance of this research was highlighted and an environment was set to test the energy strategy sufficiently in a cost effective way.

Lastly the organization of the dissertation was discussed.

Chapter 2 - Solar Energy, MPPT Application and Algorithms

2.1 Solar Energy

This section describes the motivation behind using solar energy as an electricity source, as well as using it as the main theme of this research project. It relates the importance of solar technology to using solar technology in vehicles, as well as why further research in this field is beneficial to this sector.

The section concludes with some basic information with regards to the composition of a solar vehicle.

Separate from solar technology research, there is also a definite interest in the development of Neuro-Fuzzy control systems for this project.

2.1.1 Importance of Solar Technology

Solar technology has gained importance over the last few years as an electricity source [2]. World-wide photovoltaic peak power capacity has increased by almost 70% between 2011 and 2012 [2]. The reasons are the improvement in technology, which reduces cost and increases efficiency, as well as the increase in fossil fuel electricity costs [3]. As seen in Figure 2-1, since 2007 the power capacity of photovoltaic (PV) installations has grown from almost an insignificant 10GW to about 80GW in 2011 [5]. This is on par with biomass installations and at a faster growth rate.

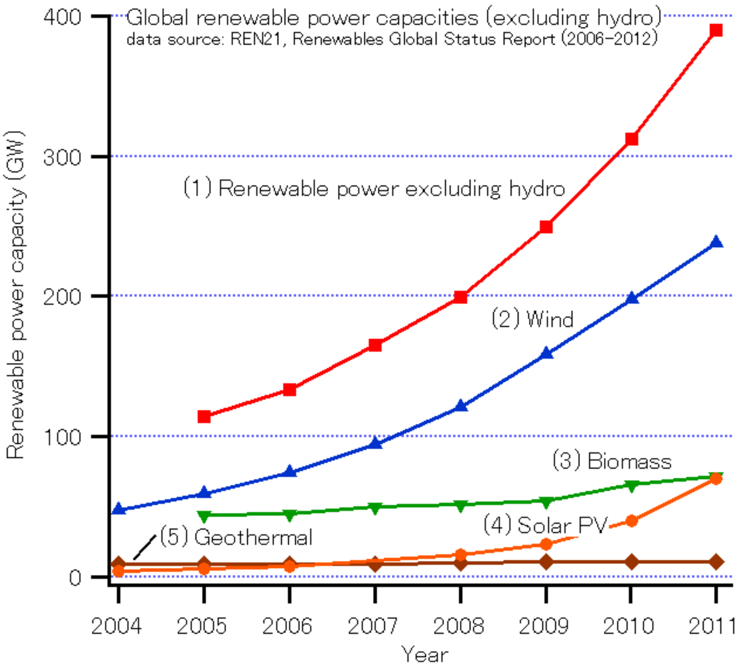


Figure 2-1: Global renewable power capacities [5]

The growth in solar technology for the use in electricity generation, especially the use of photovoltaic cells (panels), has opened up possibilities for the use of this technology in other sectors because of the increase in effectiveness and decrease in costs. One such sector is the use of solar energy in vehicles. The use of solar energy for vehicles starts from simple applications such as keeping a car cool while the vehicle is switched off (auxiliary power) up to being used in conjunction with fossil fuels in hybrid vehicles, full electric and even fully solar powered vehicles [6].

2.1.2 Significance of Solar Vehicles

The advantages of solar vehicles are highly dependent on the area of application. For instance, solar vehicles have the advantage of having a free energy source at its disposal, however this energy source is only available for limited times each day due to time of day, weather conditions, etc. When sunlight is not available, the vehicle needs to be charged from a normal power outlet. This limits the advantages of solar vehicles mostly to consumers who do not have to travel long distances with such a vehicle, but have short distances to travel each day, depending more on the capacity of the vehicle's battery bank.

With regards to the above, some of the advantages of solar vehicles are:

- The use of a freely available energy source
- Very low noise production (Electric motor, compared to an internal combustion engine)
- Integration with other technologies to form hybrid vehicles to overcome some of a purely solar vehicle's limitations
- Increasingly economical as fossil fuel prices rise and solar technology advances
- Can be charged at home with a normal power outlet

Currently solar vehicles are not readily sold for everyday transportation, but are rather used for demonstration and research purposes [7]. Limitations pertaining to this situation are [7]:

- Low storage energy density to weight in batteries
- Low energy density available for sourcing, limited by the surface area of the car facing the sun with PV's
- Some design considerations, such as the weight of PV panels and its durability

From the above it is clear that research needs to be done in optimizing the use of the energy available in such a vehicle in order to extend its usefulness. As technology in this field advances, it will be possible to source more energy per surface area unit of the car as well as storing more electrical energy at a reduced weight for when there is no sunlight available.

As the above mentioned progressions are made, solar vehicles will become more popular, especially in metropolitan areas where people travel short to medium distances each day. This technology does however has a significant role in full electric vehicles that, although they require charging at power outlets, can use every bit of free solar energy available to extend driving distance and reduce electrical outlet usage (reducing indirect pollution and running costs).

2.1.3 Energy Management in Solar Vehicles

Energy management in any vehicle can be broken up into four main parts:

- Sourcing energy as effectively as possible
- Sourcing energy as efficiently as possible
- Using the energy available as effectively as possible
- Designing the vehicle to minimize other energy losses (Efficient use of energy)

This research project will focus on the first point.

With solar vehicles the energy is sourced from PV panels, which are controlled by maximum power point tracker (MPPT) modules. The harvesting of as much energy as possible at a given moment is crucial, since sunlight is not readily available within the 24 hours of each day. The excess energy needs to be stored for later use when the irradiance from the sun is insufficient for proper operation.

The use of solar energy in vehicles is most advantageous when as much as possible energy from the sun is used and as little as possible from the power grids. This would mean that extreme caution must be used while driving in order to travel at the instance speed that would be best suited for the driver to reach his destination, using the least amount of energy. A proper driving strategy with feedback to the driver is needed to control the use of the energy as effectively as possible.

The energy management of such a vehicle is therefore a very complex one since it depends on a wide range of sometimes quick changing variables:

- Distance to drivers destination left
- Relief of the route taken
- Preference of driver needs, for example fasted speed or best energy retention
- Sun irradiance available
- Wind conditions

Furthermore, the optimization of every part of the vehicle where energy is lost increases the amount of energy available to the user.

All of the above variables need to be taken into account and handled properly in order to have an effective energy management strategy and control.

2.1.4 Composition of a Solar Vehicle

Only a basic overview of solar vehicles and their main electrical components are described below. Figure 2-2 illustrates the main electrical components in a solar/electrical vehicle as well as their interactions.

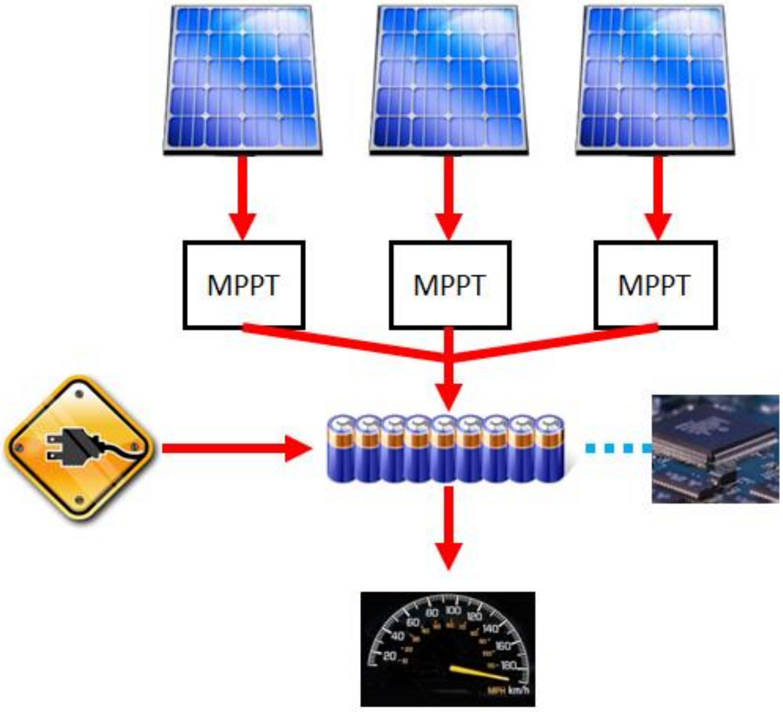


Figure 2-2: Energy flow schematic of a solar/electric vehicle

The main concept of a solar vehicle is very basic: Solar energy gets converted to electrical energy through solar panels and the electrical energy gets converted to mechanical energy through an electric motor. Various control devices are used to manage the energy transfers as well as methods to store energy for later use. The main electrical components of a solar vehicle, as depicted in Figure 2-2, are described below.

Photovoltaic Panels

A photovoltaic (PV) panel is an array of PV cells. Figure 2-3 is an example of solar panels, mounted on a building's roof in this instance [7]. As can be seen in the image, the panels are constructed from photovoltaic cells. These cells can be connected in either series, parallel or both, depending on the nominal voltage and current output required from the panels.

Figure 2-3: Photovoltaic panels [7]

Solar panels are normally very ridged and the fragile PV cells are well protected, although there also exists semi flexible panels [8].

Maximum Power Point Trackers

A PV panel acts as a constant current source under low load conditions [9]. This is due to the fact that the output current capabilities mostly rely on the amount of sunlight incident on the panel [9]. Under higher load conditions, the increase of the load increases the output voltage of the panel [9]. The amount of energy incident on the PV panel is limited, therefore the current would decrease as the load and panel output voltage increases. An example of the resultant power curve of such a panel is illustrated in Figure 2-4 [10].

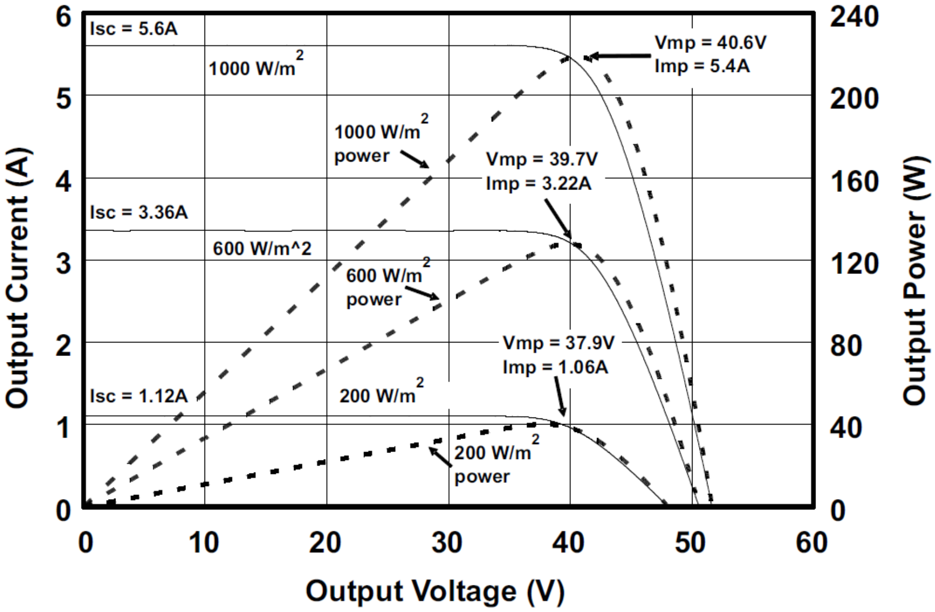


Figure 2-4: Example of a PV panel power curve [10]

From Figure 2-4 it can be observed that there is a definite maximum power point, varying depending on external conditions.

By manipulating the load of the PV panel it becomes possible to regulate the output voltage of the panel, affecting the output current of the panel and consequently controlling the operating point of the panel. Thus in order to operate the panel at its maximum power point, it is required to be able to control the load of the panel, without negatively affecting the output characteristics of the PV system. This is achieved by using a DC-DC converter as a controllable load. The converter is controlled via any hardware that can implement the controlling algorithm. The execution of the algorithm leads to the tracking of the maximum power point, hence the name MPPT (Maximum Power Point Tracker).

This research is focused on the algorithm needed to track the MPP (Maximum Power Point).

Motor and Drive

The electrical characteristics of the motor and drive systems used in electrical/solar vehicles are not of much importance for the purpose of this research. There is a wide variety of motors and drives that can be used, depending on the power available and the application/design of the vehicle. The basic principles are that an electric motor is needed as a primary source of propulsion for the vehicle, controlled by a drive so that the vehicle speed can be controlled.

In conjunction with the above, a battery pack is used to stabilize the output voltage of the MPPT (as observed during experimentation) and to store electricity that can be used when external conditions are not favourable to extract energy from the PV panels. The battery pack can also be charged from the electricity grid if required.

Control Electronics

Any control electronics can be used to manage the MPPT as long as it can implement an algorithm on the DC-DC converter. This can range from industrial PLC’s to homemade microprocessors. Depending on the scale of the system, the electronics might be required to manage the safety of the system, broadcast information about the system through a network or even controlling the system remotely.

2.2 Function and Application of a MPPT

Figure 2-5 and Figure 2-6 are examples of a compilation of PV panel power curves for a specific cell temperature and a variance in solar irradiance [11]. Figure 2-7 and Figure 2-8 are similar graphs, but at constant solar irradiance and a variance in cell temperature [11].

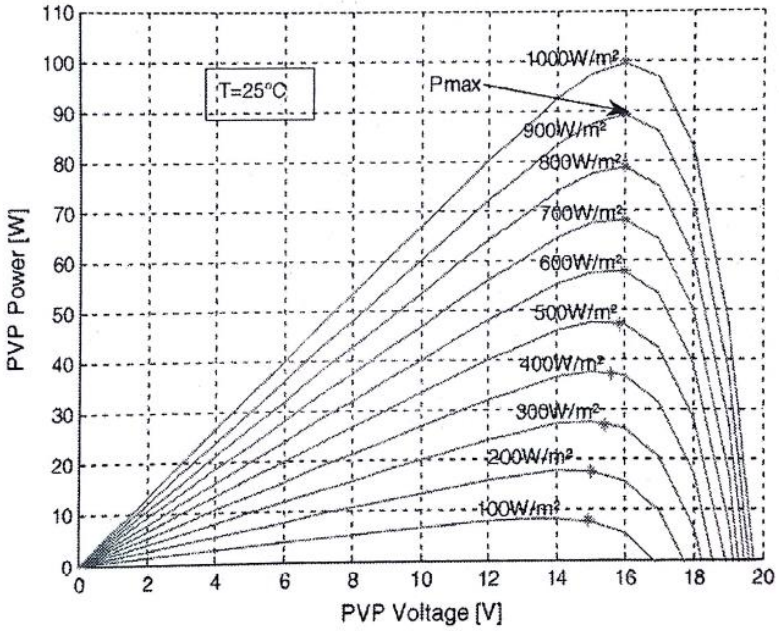


Figure 2-5: Simulated power/voltage curve characteristics of a PV panel (PVP) model at constant temperature (T) of 25°C [11]

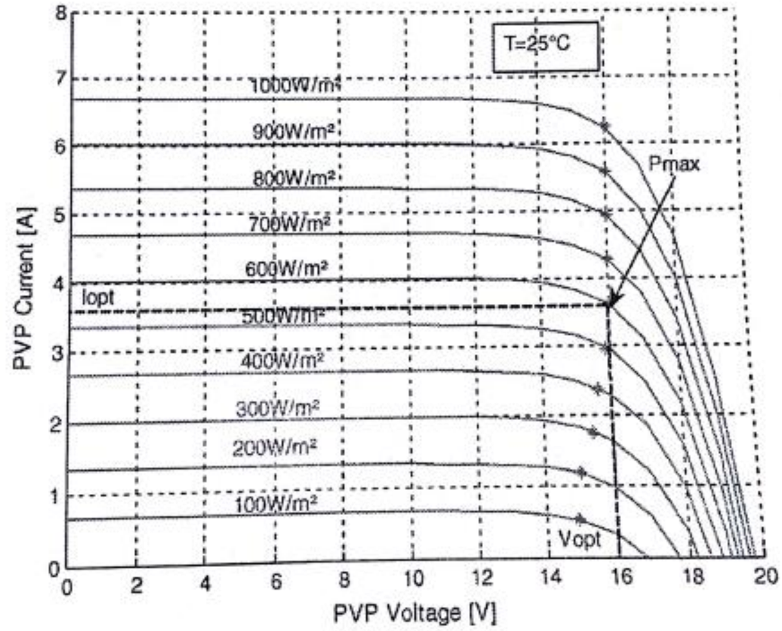


Figure 2-6: Simulated current/voltage curve characteristics of a PV panel (PVP) model at constant temperature (T) of 25°C [11]

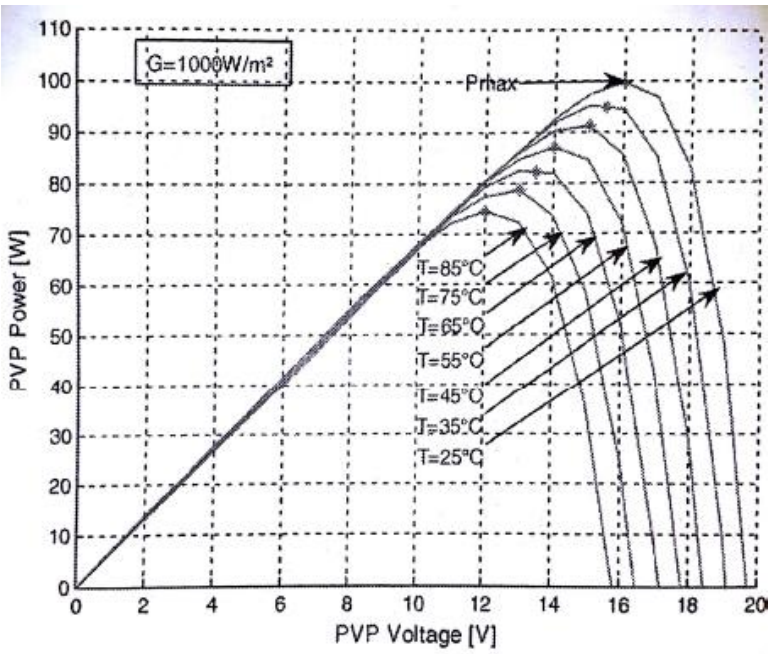


Figure 2-7: Simulated power/voltage curve characteristics of a PV panel (PVP) model at constant solar irradiation (G) of 1000W/m2 [11]

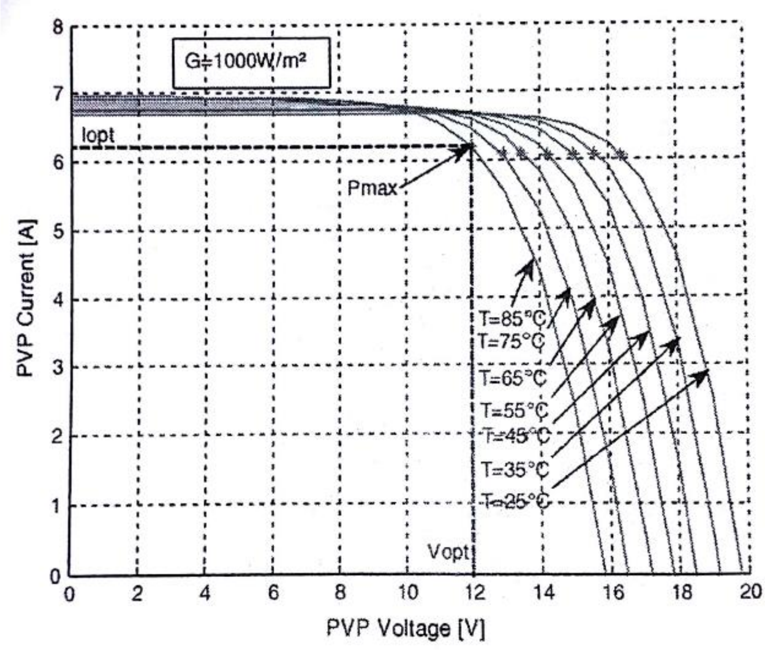


Figure 2-8: Simulated current/voltage curve characteristics of a PV panel (PVP) model at constant solar irradiation (G) of 1000W/m2 [11]

These figures are good illustrations of the factors impacting the MPP location as well as locating the MPP, since:

$$P_{PVout} = V_{PVout}.I_{PVout} \quad (2.1)$$

Where:

P_{PVout}: Power delivered by the PV panel

V_{PVout}: Output voltage produced by the PV panel
I_{PVout}: Output current drawn from the PV panel

It follows that these non-linear characteristics of PV panels require intricate methods to determine the MPP for a specific panel as well as tracking it accurately through different ambient conditions. It is also clear from the above figures that the power output performance of PV panels is very much dependant on its operating conditions and that the determination of its MPP is crucial to the performance of any PV panel system.

2.3 MPPT Algorithms Analysed

Numerous methods to find and track maximum power points have lately been designed and implemented for different solar power applications. These methods range from simplistic to intricate control systems, each with their own advantages and disadvantages. Below are examples of different algorithms/methods.

Perturb and Observe (P&O) [12]

This method is popular because it is easy to implement in MPPT systems [12]. As the name suggest, the PV panel voltage is perturbed (increased and decreased) and the resulting output power is monitored (observed) in order to determine the maximum power point.

Various problems could however occur when using this method [12]:

- It is not able to track the MPP when the solar irradiation is low.
- Rapid changing solar radiation makes it very difficult to track the MPP
- Tracking may get isolated around a false MPP due to a local maximum point on the solar panel power curve.
- When the optimum MPP is found, this method still requires the tracking point to oscillate around this optimum point in search of changing conditions (Shift in MPP).

Different variations exist to increase the efficiency of this method, but due to the slow response to condition changes and its inefficiency due to optimum point oscillation, this method is not considered for this specific research application [12]. Due to its popularity and ease of implementation, this basic method is used as a benchmark to compare the proposed control algorithm with.

Incremental Conductance Method [12]

This method is a modified version of the normal perturb and observe method. It reduces the oscillation involved during MPP tracking through the calculation of the tracking direction. It can also determine whether the MPP has indeed been reached, reducing oscillation further [12].

Some disadvantages of this method include [12]:

- Computationally intensive
- MPP tracking speed depends on the step size used for the perturbation

Artificial Intelligence Based Methods [12]

Artificial intelligence (AI) is a collective name for computers/software with the ability to display intelligent behaviour [13].

The type of intelligence that is of importance for this research is based on logic. The idea is to have a computer based product that can reason according to set of rules. The outcome of the reasoning is an updated operation duty cycle for the MPPT.

These sets of rules are based on certain parameters and by updating these parameters according to historic results (errors) it is possible to teach the system to produce better results

Different types of artificial intelligence methods exist. Two popular methods include:

- Fuzzy logic
- Neural networks

Neural networks though have distinct disadvantages compared to fuzzy logic with regards to:

- Intensive data acquisition
- Large memory space requirements

Fuzzy logic does not require any complex mathematical formulas for non-linear control. The behaviour of such a control system depends on the:

- Shape of the membership functions
- Input and output scale factors
- Size of the rule base

The advantages of the fuzzy logic real time fast response and simplistic calculations makes this method ideal to be used in control systems to counter performance degrading factors such as shadow flicker in moving PV panels.

There is however no formal method to determine the above control system factors and scaling values. Experience is used to modify these parameters in order to increase efficiency of the control system. Due to the many external factors that requires continuous fine tuning of control parameters, it was decided to use a neural network to determine the fuzzy logic parameters to create a fuzzy control system that can ‘teach’ itself based on performance results.

Simulation data can be used to set-up the fuzzy logic parameters [11]. This simplifies the control system design significantly.

2.4 Conclusion: Chapter 2

This chapter gave an overview of solar energy and the specific implementation in vehicles. The importance of a MPPT was highlighted and the need for such control in solar energy systems is evident.

Consequently some methods of controlling the operation point of a solar panel were summarized to create an awareness of the current methods commonly used, as well as their advantages and disadvantages.

Chapter 3 - Scale Model of Solar Vehicle for Research

Figure 3-1 is an overview for an experimental electrical and electronic system to conduct tests on a MPPT.

- The main electrical components are indicated in red and are discussed in detail in this chapter, except for the MPPT.
- The orange block indicates a laptop that is used to collect data from the MPPT, as well as offline re-calculation of control system parameters.
- The green blocks indicate the electronic controller and sensors for the MPPT. Chapter 4 details the electrical and electronic design and manufacturing of the MPPT.
- The blue blocks illustrate the software programs developed for the MPPT.

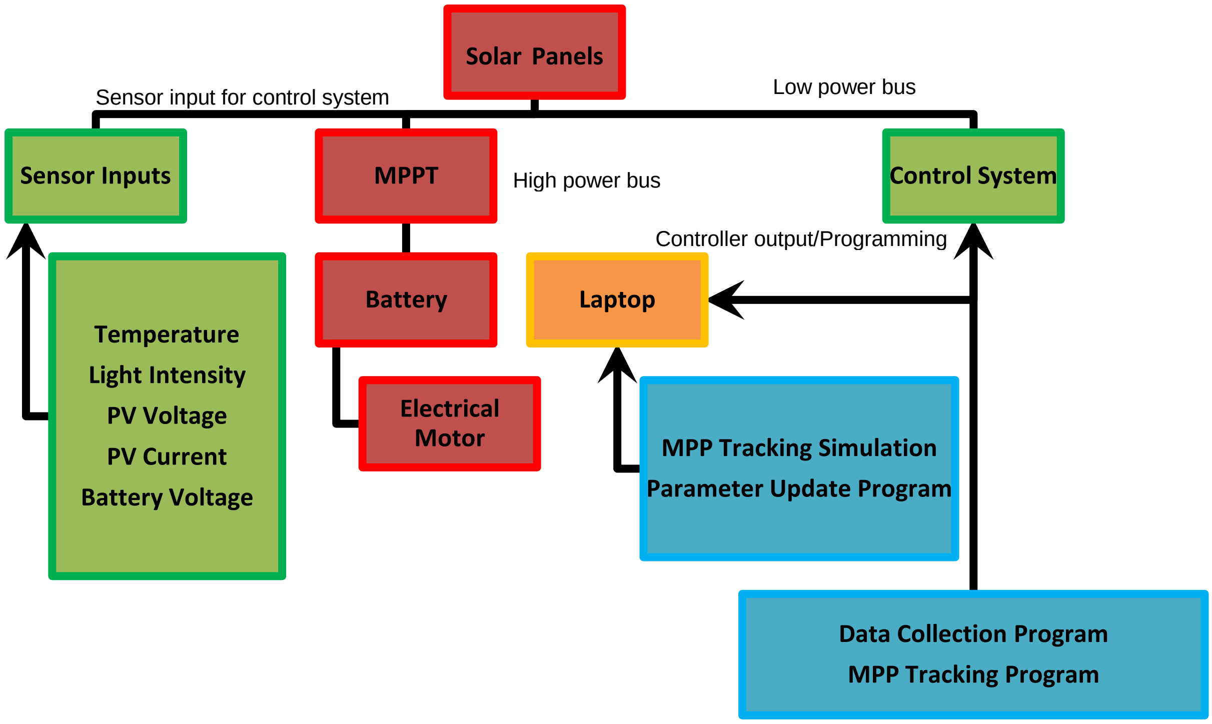


Figure 3-1: Experimental electrical and electronic system overview

The above components all form part of the solar vehicle prototype needed for this research. Mechanical components were chosen to support the experimental electrical system, but no special features were required to create significant impact on the test results.

Figure 3-2 is a very basic diagrammatic concept of a solar vehicle.

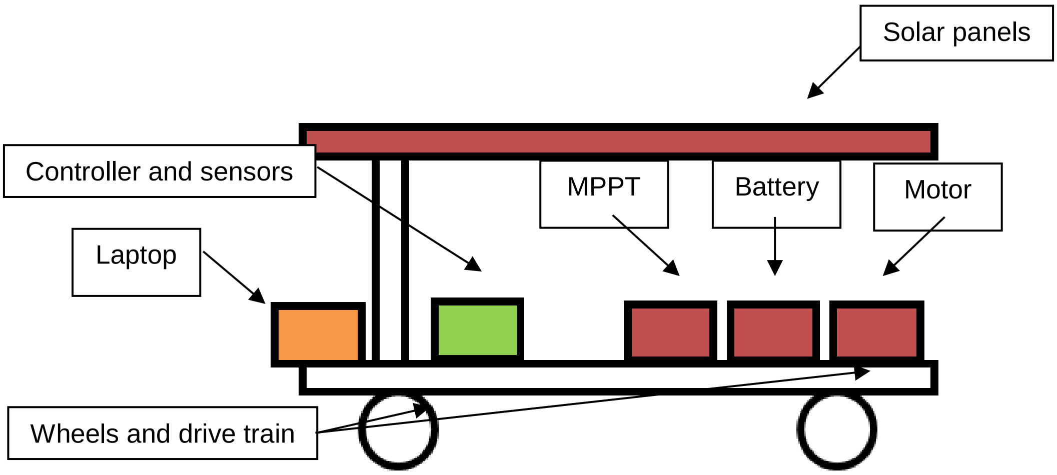


Figure 3-2: Basic solar vehicle schematic

The testing and manufacturing of a full scale solar vehicle is very expensive and time consuming. Furthermore, the design of such a vehicle would be better if research is done first on a model of a smaller scale. The advantages of using a small scale model can be summarized as follows:

- Less expensive for trial and error testing
- Fewer resources required in test phase
- Less complexity of the overall system for better focus on specific research areas
- Quick test results because of fast manufacturing

The advantages and importance of such a small scale model is of such a nature that it will form the main part of the equipment used for this dissertation.

Even though a small scale model is used, it still acts as a good representation of an actual solar vehicle and is chosen as such to convey meaningful results. After the research on the small scale model is complete, experiments will be conducted to prove its validity. The verification process is located in chapter 6. This chapter details the requirements, design and manufacturing of such a small scale solar vehicle model.

3.1 Requirements of Solar Vehicle Model

In order to yield meaningful results, the small scale solar vehicle has to adhere to the following requirements:

- The vehicle must consist of all the main components that would be used in a full scale model, as related to solar energy harvesting, solar energy storage and the controlled used of this energy. These components are listed below:
  1. Solar Panels
  2. MPPT
  3. Battery Pack

4. Micro Controller Unit
5. Electric Motor
6. Mechanical Drive Train and Wheels
7. Vehicle Mechanical Structure (Body)

- The chosen scale should be small enough to justify the advantages listed in this chapter, but large enough to still yield meaningful result for this research. A scale model of approximately 1:20 is chosen as a first approach.
- The solar vehicle must be inexpensive and easy to manufacture to accommodate rework.
- Because the research is done by an individual, the solar vehicle must be easy to transport and to gain test results from.
- Even though basic vehicle functionalities such as steering are not required for this research, it is required that the vehicle should have some form of speed control for safety.
- A communication interface with a standard notebook is necessary for easy retrieval of data.

With these requirements of the research in mind, the vehicle hardware specifications were formed. Some of these specifications were chosen as a first approach and can be modified if needed, for a wider range and scope of results.

Specification 1: Use solar panels with a combined power rating between 20W and 40W.

Standard commercial electric vehicles do not have large surface areas available for PV cells. Most electric and hybrid vehicles can only accommodate a limited capacity of PV cells to partially charge the battery pack [14]. An example is Ford's *C-Max Solar Energi*, which utilizes 1.5m^2 of solar cells on its roof [15]. They do, however, claim 50% more energy harvesting capabilities from conventional PV cells [15]. These specifications are used as a guideline to determine the specification of the solar vehicle model. Figure 3-3 is an image of Ford's C-Max [15].

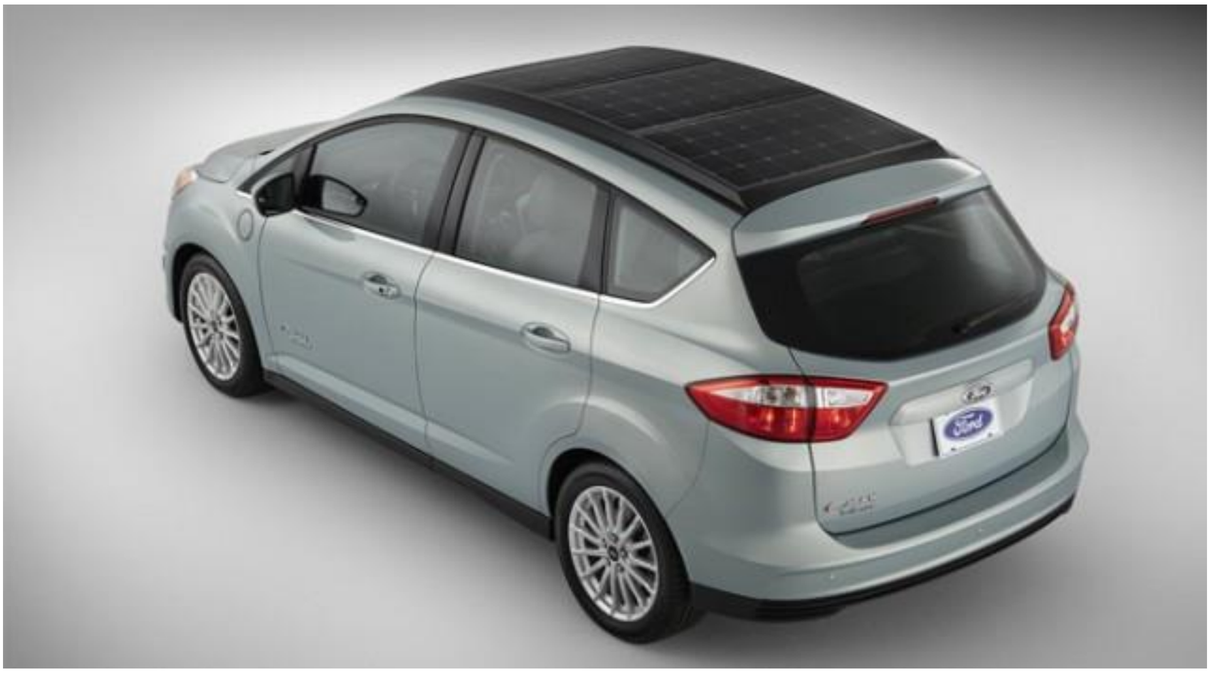


Figure 3-3: Ford C-Max [15]

If conventional solar cells can have a maximum efficiency between 20% and 40% (Such as multijunction photovoltaic cells which produce very good efficiencies [16]), then 50% higher efficiency from conventional PV cells is assumed to be 50% higher than silicon based PV cells, thus 40%.

This does however excludes other methods of concentrating more sunlight on the solar panels. The average irradiance on a sunny day is 1000W/m^2. The effects of irradiation concentration are not taken into account as this can be utilized on the same small scale system, without changing the dynamics of the system more than just producing better solar energy harvesting. It also greatly simplifies and reduces the cost of the research. The power produced by these PV cells on the roof of such a vehicle can be estimated as follows:

$A_{cells} - Total\ surface\ area\ of\ PV\ cells\ on\ a\ car's\ roof$

$i_{ave} - Average\ irradiance\ from\ the\ sun$

$e_{cells} - Efficiency\ of\ PV\ cells$

$P_{solar} - Solar\ power\ converted\ to\ electricity$

$$P_{solar} = i_{ave} \times A_{cells} \times\ e_{cells} \qquad (3.1)$$

$$P_{solar} = 1000 \times 1.5 \times\ 0.4$$

$$P_{solar} = 600W$$

If using an estimate of one twentieth to form the small scale solar vehicle, this produces a baseline PV power capacity of 30W. The accuracy of this specification is not too crucial to the outcome of this research, due to the high variability in the use of PV cells on electric and hybrid vehicles. It is estimated that for this dissertation, a range of 20W to 40W should be acceptable. This is because, as in the case of the C-Max, the PV cells are used as a smaller auxiliary power source for the batteries pack, compared to an electrical outlet point [15].

Specification 2: Use an electric motor for the propulsion of the vehicle with power rating between 20 and 150W.
In the case of the C-Max, the power rating of the electric propulsion motor is 1.4kW [17]. If scaled by twenty, this produces a baseline model electric motor power rating of 70W.

For the scope of this research, it was decided that a larger motor should still yield good results, because a perfect scaling of the weight, aerodynamics and drive train system of the car will not be used. This results in a large performance difference between the full scale vehicle and the model and the most important aspect remains the fact that the PV cells might not have the capacity to power the vehicle by itself. Consequently a small scale motor power rating between 20W and 150W can be chosen.

A 30W motor was selected to keep the load low enough so that the battery is not drained to quickly, but large enough to ensure that there is a more or less constant current pulled during tests.

Specification 3: Use a lead-acid battery pack capacity between 40Wh and 80Wh.
The battery capacity specification was chosen such as to allow continuous testing of the system between two and three hours (if no significant charging is possible). The battery pack capacity and size are not chosen according to scale as the influence this has on the system is mainly on the physical dynamics of the vehicle, which falls outside the scope of this research.

Furthermore, if the weight and size of the battery are not important to the outcome of this research, then any type of battery technology can be used, because the focus is more on the fact that the system does have the ability to store enough energy to power the vehicle’s motor for a meaningful amount of time. A lead-acid battery of good enough capacity is sufficient and the technology is rather inexpensive, compared to others such as lithium-ion [18].

The above specifications and requirements are used in Section 3.2 to design the solar vehicle model. Because the emphasis is on the testing of the control system of the MPPT and the energy management of the vehicle, the parameters above should be adequate for the scope of this research.

3.2 Design of Solar Vehicle Model

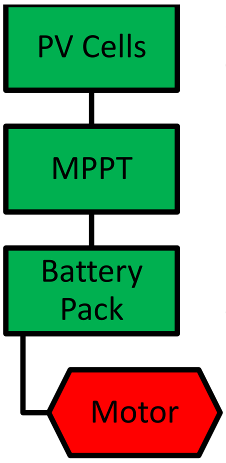
The design of the solar vehicle model is broken down into three sections, namely:

1. Electrical design: The energy harvesting, -storage, power electronics and propulsion system of the vehicle.

2. Electronic design: Control system of the MPPT, communication interface and energy management system.
3. Structural design: Mechanical structure of the vehicle, including the drive train.

3.2.1 Solar Vehicle Electrical Design

Figure 3-4 is a basic flow diagram of the solar vehicle electrical system. The items in the flow diagram are the four main electrical components of the vehicle.



Converts sunlight energy into electrical energy.

Electrically isolates the battery pack from the PV panels.

Battery pack stabilizes voltage for motor and stores energy for later use.

Converts electrical energy into mechanical energy to move vehicle.

Figure 3-4: Basic flow diagram of solar vehicle electrical system

The function of each main electrical component and its contribution to the research is listed below:

- ***Photovoltaic Cells:*** The PV cells harvest energy from the sun to be used by the vehicle. The aim will be to control the solar panel so that it can harvest energy at its maximum power point (i.e. producing maximum power).
- ***MPPT:*** The maximum power point tracker is used to control the energy harvesting of the solar panel (PV cells). It also functions as a voltage regulator and dc-dc converter. The aim will be to implement the dissertation MPPT control system on this unit.
- ***Battery Pack:*** The battery pack is used to store electrical energy in the vehicle. The energy management system controls the best use of this energy source.
- ***Motor:*** The electrical motor is the propulsion device of the vehicle. This is necessary for consumption control of the vehicle, as well as dynamic testing of the vehicle.

The whole electrical power circuit is designed to function on direct current only and throughout this chapter if a voltage or current is stated, then it is implied to be direct current unless stated otherwise.

Photovoltaic Cells

Polycrystalline silicon based PV cells were chosen for this research because they are not so expensive and they are easy to acquire [19].

Figure 3-5 is an image of two polycrystalline silicone solar panels. The two panels in the image were made available for this research by NMMU and can be integrated into the system without modifications. Because these panels do not require purchasing, it reduces the cost of the research even more.

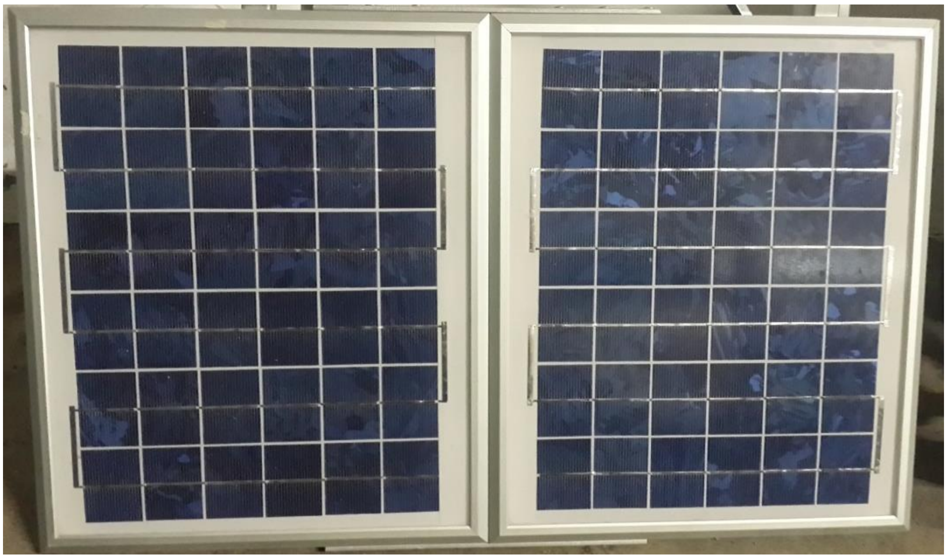


Figure 3-5: Two polycrystalline solar panels used in this research

The specifications of these panels are summarized in Table 3-1: Solar panel specifications as provided by Suntech [20]. These specifications were provided under the following test conditions:

- Atmospheric pressure: 1.5atm
- Irradiation: 1000W/m^2
- Cell temperature: 25°C

SUNTECH STP010-12/Kb	Single Panel	Parallel Module
Rated Maximum Power (Pmax)	10W	20W
Output Tolerance	±10%	±10%
Current at Pmax (Imp)	0.57A	1.14A
Voltage at Pmax (Vmp)	17.4V	17.4V
Short-Circuit Current (Isc)	0.65A	1.3A
Open-Circuit Voltage (Voc)	21.6V	21.6V
Nominal Operating Cell Temperature	45°C ± 2°C	45°C ± 2°C
Weight	1.5kg	3kg
Dimension	310x368x18 mm	620x368x18 mm
Maximum System Voltage	715V	715V
Maximum Series Fuse Rating	5A	5A
Cell Technology	Multi-Si	Multi-Si

Table 3-1: Solar panel specifications as provided by Suntech [20]

The combined nominal power ratings for the two panels are 20W. This falls within the requirements and specifications of this project.

It was decided to connect the solar panels in parallel in order to keep the nominal operating voltage of the panels at a relatively low 17.4V. The MPP voltage is expected however to decrease since the temperature of the cells will rise to more than 25°C. Even with this effect included, within the scope of this research, the MPP voltage should not drop below the 12V of the battery and motor system.

A 12V propulsion system was chosen because of available parts as discussed later in this chapter and a much wider voltage range could have been used without any loss of integrity to the research.

The combined weight (3kg) and size (620x368x18 mm) of the panels makes them easy to handle by one person during testing.

Maximum Power Point Tracker (MPPT)

The MPPT is designed and manufactured by Mr J. Jordaan and is detailed comprehensively in Chapter 4.

Battery pack

A single 12V, 4.5Ah lead-acid battery was chosen because of easy procurement and its low price. The capacity of the battery is enough in order for the system to run for a few hours of continuous testing. With the vehicle's motor drawing a nominal current of 1.3A, the battery can supply the motor for about 3hrs.

Figure 3-6 is an image of the acquired battery manufactured by Discover®. The most significant specifications of the battery, for the scope of this research, are listed in Table 3-2.

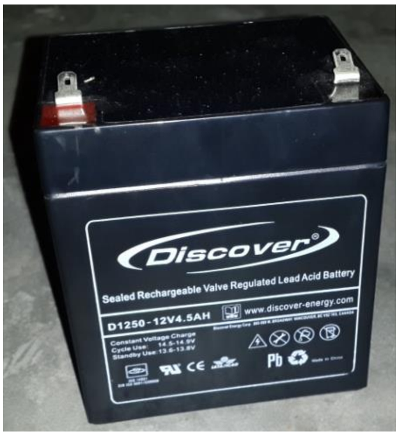


Figure 3-6: Selected lead-acid battery for main energy storage of solar vehicle

D1250 Lead Acid	
Nominal Capacity	4.5Ah

Nominal Voltage	12V
Maximum Voltage	14.5-14.9V

Table 3-2: Primary specifications of selected lead-acid battery

The charging specifications and curve is detailed in a later chapter as part of the control circuit of the MPPT.

Motor

The electric motor is the propulsion source of the solar vehicle. In order to make the experiments done on the vehicle easier and to limit the space needed for testing, no great speeds are required. As a first approach the focus will be more on having a vehicle capable of performing speed control.

A second-hand 30W golf trolley motor and speed control assembly were purchased for this project. The speed control is done by means of resistive loading. A pod controls the speed of the motor and is adjusted by hand. This is adequate for the scope of this research as the focus is on vehicles that are still manually operated by humans and the speed control is more for the ease of experimentation and safety. Figure 3-7 is an image of the acquired motor.

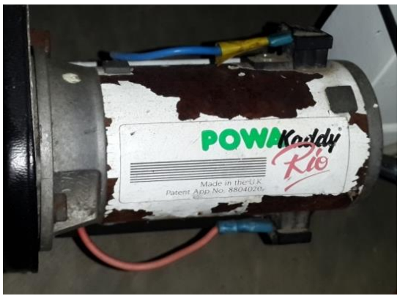


Figure 3-7: Electric motor selected for the solar vehicle model

The original motor and gearbox assembly were designed to transport significant weight. This is very helpful to the project budget, as less money and research time has to be spent on making the vehicle light in weight and aerodynamics, without impeding on the integrity of the experimental results. The torque is more than enough to drive the assembled vehicle.
The motor speed is fast enough to propel the car at a maximum speed of roughly 6km/h.

3.2.2 Solar Vehicle Electronic Design

The solar vehicle model is designed to contain one micro controller unit. This unit controls the following functions within the vehicle:

- Maximum power point tracking: Controls the current output of the solar panel in order to produce the maximum power supply as the current-voltage curve of the panel changes with the change in incident irradiance.

- Charging of the battery: Maintains the charging curve of the battery and prevents overcharging.
- Communication interface: Presents the solar vehicle with an USB interface to communicate with a standard notebook.

Because the solar vehicle's energy management is the main focus of this dissertation, time and electronic resources were spent mostly on the sensors and control unit to accommodate the above functions.

The design, programming and manufacturing of the control circuit board are detailed in Section 4.2.

3.2.3 Solar Vehicle Mechanical Design

The focus of this dissertation is mostly on the electrical dynamics of a solar vehicle. Consequently only a basic mechanical structure was needed for the test vehicle.

The mechanical components needed for this project are as follows:

- Body structure for the vehicle (frame)
- Mechanical drive train
- Wheels
- Brackets to mount components on the vehicle

An electrical golf trolley that was purchased to obtain the electrical motor needed for the project was found to have a good chassis and support frame, battery mounting plate, wheels and mechanical drive train. As a first approach, this was a very good way to simplify the manufacturing the vehicle, without any loss of integrity to the research.

Body structure of the solar vehicle

The body structure of the vehicle, or frame in this instance, must be:

- Strong enough to carry the weight of the solar panels, battery and motor mounted on top of it
- Designed for easy fitment of components inside
- Durable for repeated testing and travelling between test sites
- Able to accommodate the wheel set securely
- Light enough so that one person can easily transport it

The body structure of the golf trolley was found to be adequate for the vehicle model in all the aspects mentioned above. Part of the top structure was removed to make it more compact.

Mechanical drive train of the solar vehicle

The gearbox assembly and wheel axle of the golf trolley, as it is, can propel the vehicle at a maximum speed of about 6km/h. This is more than satisfactory for the scope of this research as it is just important to have significant speed variations, not specific speed variations.

The gearbox is also already enclosed for safety and installed, which reduces the complexity of the solar vehicle build. Figure 3-8 is an image of the golf trolley’s gearbox assembly and motor as it will be used for the solar vehicle.

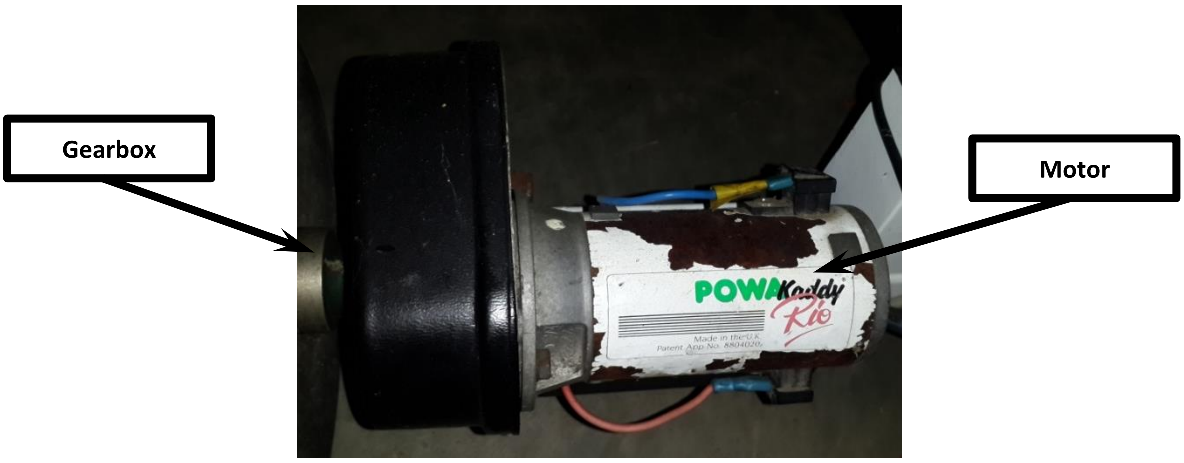


Figure 3-8: Gearbox assembly and motor of the electric golf trolley

Wheels of the solar vehicle

The wheels of the golf trolley are light and strongly built. They are more than adequate within the scope of this research. They also provide good traction for incline testing.

Brackets to attach components to the solar vehicle

Several components are required to be mounted securely on the vehicle body structure. These include:

- Solar panels
- MPPT
- Electronic control system

Many items such as the electrical motor, battery and drive train assembly are used on the chassis as mounted for the golf trolley.

Solar panels

The solar panels are required at the top of the vehicle for the best irradiance exposure. To protect the PV cells and glass front cover from damage, the panels must be fitted securely. No loose movement of the panels will also contribute to the integrity of the research.

It was decided to manufacture a support platform to the golf trolley chassis for the mounting of the solar panels. Aluminium is both strong and light enough to use for this purpose, especially if aluminium tubing is used for the brackets. Two pieces of aluminium tubing are modified, drilled and then riveted in place on the top part of the base structure of the car

MPPT

The MPPT is mounted between the electrical motor and battery of the vehicle on a galvanised steel base plate. The plate is riveted securely to the base structure of the vehicle.

Electronic control system

The electrical control system is mounted onto of the electrical module of the MPPT. Spacers are used to separate the prototype circuit boards from the power circuit.

3.3 Manufacturing of Solar Vehicle Model

Because of the simplistic approach to the design of the solar vehicle, the manufacturing was made very easy. Except for the brackets, only the MPPT and control electronics required manufacturing. This is detailed however in Chapter 4 and this section is about the assembly of the vehicle, including the manufactured MPPT and control electronics.

Solar panels

The solar panels are secure to the support bracket by means of aluminium plates and rivets through the frame of the panels. Figure 3-9 is an image of the panels after the mounting on the vehicle is completed.

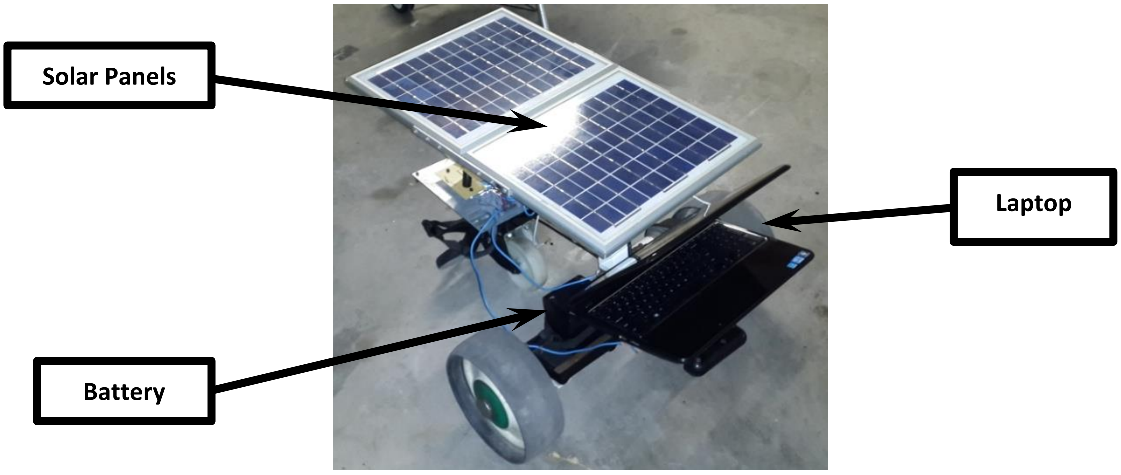


Figure 3-9: Solar panels mounted on the solar vehicle

MPPT

The MPPT circuit board is mounted riveted to the galvanised base plate on the vehicle. Figure 3-10 displays the mounted MPPT. The electronic control circuits can also be seen mounted on top of the power circuit.

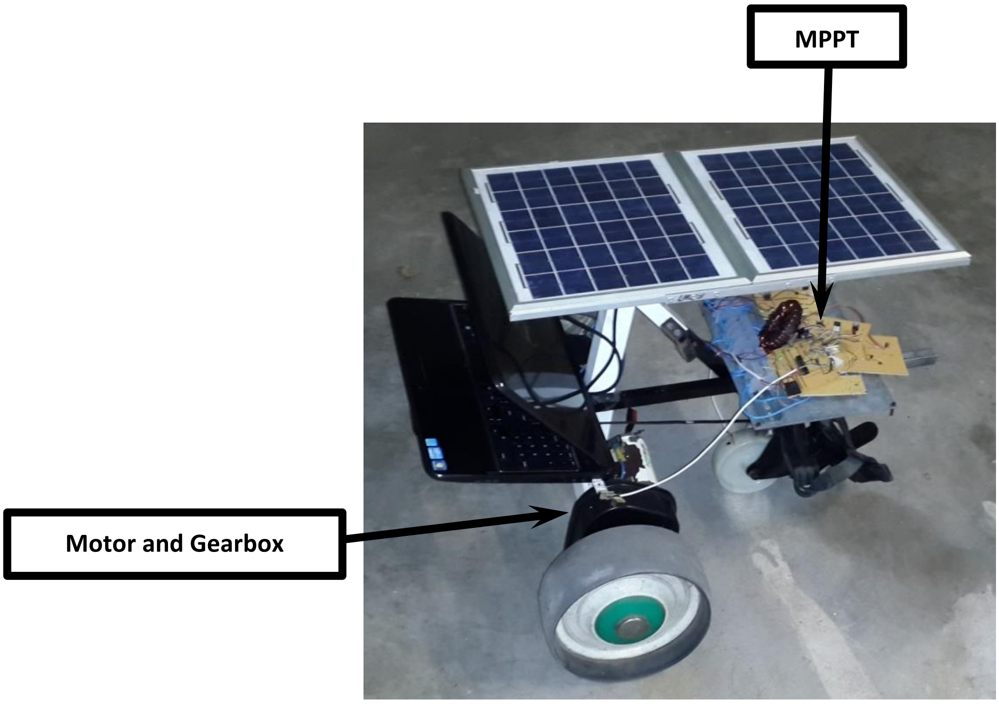


Figure 3-10: MPPT mounted on the vehicle

Electrical wiring

The electrical connections between the battery and the motor; the battery and the MPPT and the MPPT; and the solar panels and the MPPT are primarily made with 2.5mm^2 insulated braded copper wires. The current rating of the wire should be much more than adequate for this system, with a maximum current draw of about 2A.

Figure 3-11 is an image of the finished vehicle prototype.

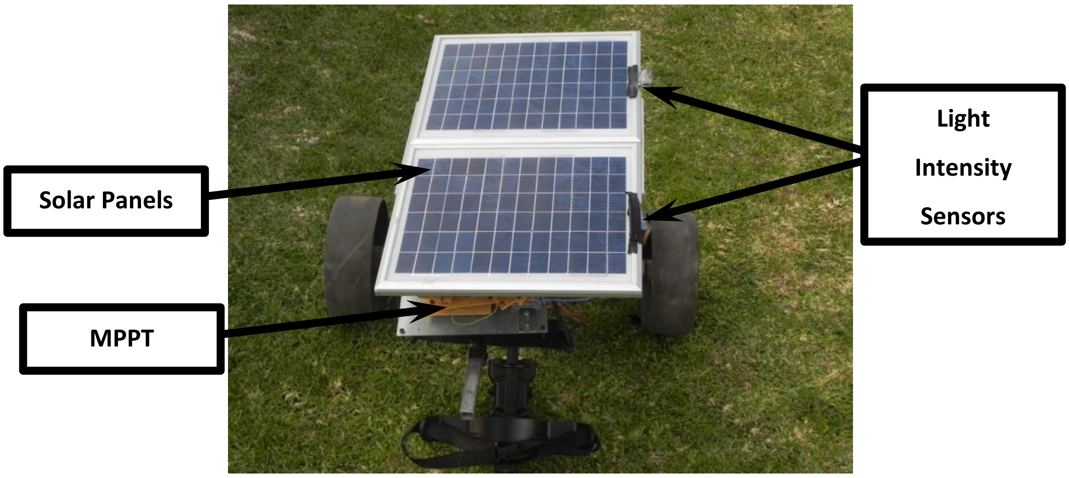


Figure 3-11: Final vehicle prototype

3.4 Conclusion: Chapter 3

This chapter highlighted the experimental setup for this research project. Although the vehicle produced is very basic, it does cover all the functionalities required to test a MPPT control system.

Chapter 4 - Prototype Maximum Power Point Tracker

Figure 4-1 is a representation of a MPPT as part of a generic solar system, illustrating its main functional behaviour. These include:

- Converting the power input from the PV panels to a different voltage and current pair as output for the battery system.
- Controlling the power conversion through the control of the switching transistor's duty cycle by means of a PWM signal, produced by the control system.
- Collecting and exporting MPPT process data.
- Collecting feedback from sensors for control system.

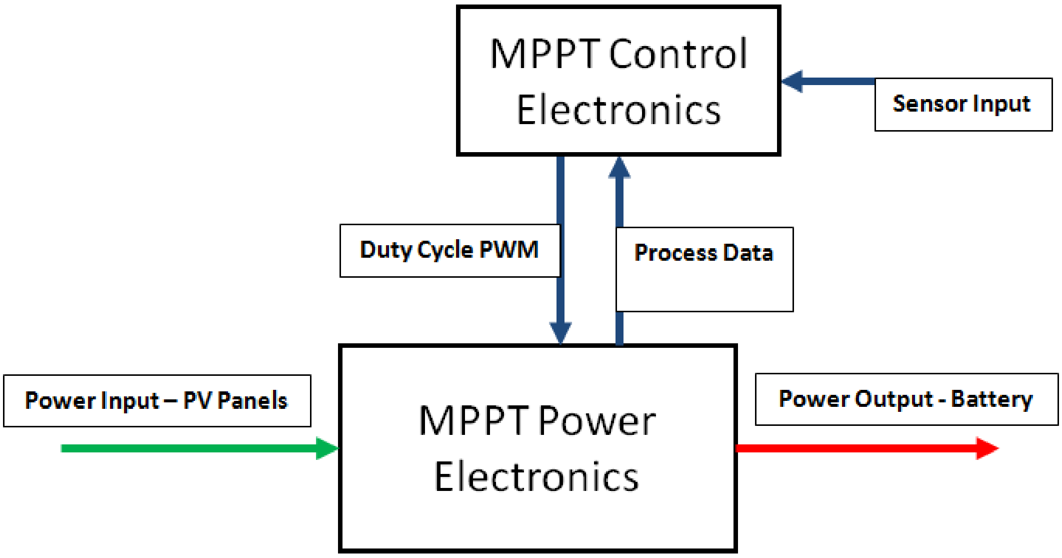


Figure 4-1: MPPT Functional Diagram

The maximum power point tracker, or MPPT, has two main functions in solar panel systems:

- Drawing the correct amount of current from the solar panels in order to operate them at their maximum power point.
- Electrically isolating the PV panels from the load. This enables the battery to determine the load voltage, while the duty cycle regulates the PV voltage relative to the load voltage.

This chapter is divided into two sections:

- MPPT power electronics circuit: Includes the design, manufacturing and testing of the MPPT circuit.

- MPPT control circuit: Includes the design, manufacturing, programming and testing of the micro controller circuit used for the MPPT.

4.1 MPPT Power Electronics Circuit

The following requirements were formulated for the MPPT power circuit:

- The circuit needs to have a power capacity that is the same or greater than that of the maximum power that can be supplied by the solar panels. The nominal maximum power of the solar panels will be surpassed with an increase in solar irradiation and a decrease in temperature, therefore the system should be designed to manage an appropriate amount of power more than the nominal.
- The MPPT must isolate the load from the PV panels electrically. This can be achieved by using suitable capacitors and inductors.

The functional constraints for the MPPT are summarized in Table 4-1.

MPPT Nominal Power Electronics Circuit Constraints	
Maximum nominal power provided by solar panels	20W
Nominal operating input voltage	17.4V
Nominal operating current input	1.14A
Maximum nominal input voltage (Open circuit)	21.6V
Maximum nominal input current (Short circuit)	1.3A
Nominal operating output voltage	Battery dependant. 12V battery is used.
Nominal operating output current	1.6A

Table 4-1: MPPT Power Electronics Circuit Constraints

From Table 4-1, it can be seen that the nominal input voltage is significantly more than the nominal output voltage used to charge the battery. It was decided to use a step-down (buck) converter as the charge control circuit of the MPPT. As a first approach it is assumed the voltage from the solar panels will not require step-up (boost) conversion in order to supply the battery while operating at the panel's maximum power point.

Calculating the absolute maximum parameters of the MPPT

Also from Table 4-1 the nominal maximum parameters for the MPPT is a good indication of what the absolute maximum specifications should be. There is one important coefficient, obtained from the solar panel manufacturer data sheets, that is used to calculate the output voltage and current of the PV panel, based on the influence of temperature. This coefficient for the open circuit voltage is given the symbol μ_{TV} and has the value of -0.075V/K [21].

From Figure 2-6 and Figure 2-8 it is seen that a change in solar irradiation constitutes the biggest effect on the PV output current. Also, a change in the PV cell temperature constitutes the biggest effect on the PV output voltage. The absolute minimum PV cell temperature is 0°C (273.15K) for the purpose of this research and the nominal temperature is 25°C (298.15K as per standard test conditions [21]). To calculate the maximum MPP voltage, the temperature coefficient is used as follows:

$$V_{MPPmax} = V_{MPPnominal} + \mu_{TV} * (T_{MINactual} - T_{nominal}) \qquad (4.1)$$

$$V_{MPPmax} = 17.4 + (-0.075) * (273.15 - 298.15)$$

$\rightarrow \boldsymbol{V_{MPPmax}} =$ **19.28V**

Where:

V_{MPPmax} – Maximum MPP output voltage of the solar panel (Input to the system)

$V_{MPPnominal}$ – Nominal MPP output voltage of the solar panel (Input to the system)

μ_{TV} – Temperature coefficient of the solar panel voltage

$T_{MINactual}$ – Actual PV cell temperature

$T_{nominal}$ – Nominal PV cell temperature

Using the same formula as above, the maximum open circuit voltage of the solar panel is calculated to yield 23.48V.

The absolute maximum solar irradiation for the scope of this research is taken as 1500W/m^2 and the nominal as 1000W/m^2 (Standard test conditions). This ration is:

$$\frac{1500}{1000} = 1.5$$

The temperature coefficient for current, μ_{TI}, is 0.05%/K [21]. From the coefficient it is deduced that there is an increase in current with an increase in temperature. The maximum temperature that the PV cells will be exposed to is 100°C (373.15K). If the current is assumed to be directly proportional to the solar irradiation, then maximum short circuit current of the panels can be calculated as:

$$I_{SCmax} = \frac{q_{max}}{q_{nominal}} * I_{SCnominal} + \frac{q_{max}}{q_{nominal}} * I_{SCnominal} * \frac{\mu_{TI}}{100} * (T_{MAXactual} - T_{nominal}) \qquad (4.2)$$

$$I_{SCmax} = \frac{1500}{100} * 1.3 + \frac{1500}{1000} * 1.3 * \frac{0.05}{100} * (373.15 - 298.15)$$

$\rightarrow \boldsymbol{I_{SCmax}} =$ **2.02A**

Temperature has an opposing effect on the output voltage and current of the solar panel. As a good estimate, maximum power that can be produced by the panel is calculated as being directly proportional to the solar irradiation incident on it. In this case 20W nominal power multiplied by the 1.5

max/nominal ratio of the solar irradiation yields 30W absolute maximum power. Table 4-2 lists these absolute maximum parameters for the MPPT.

Absolute maximum MPPT Power Electronics Circuit Constraints	
Absolute maximum power provided by solar panels	30W
Maximum input voltage (Open circuit)	23.48V
Maximum input current (Short circuit)	2.02A
Maximum operating output voltage	14.7V
Maximum operating output current	2A

Table 4-2: MPPT power electronics absolute maximum design constraints

4.1.1 Design of MPPT Power Electronics Circuit

The design of the MPPT charge control circuit is based on the design of a step-down DC-DC converter circuit. Figure 4-2 is the proposed diagram of a basic step-down single quadrant dc-dc converter [22].

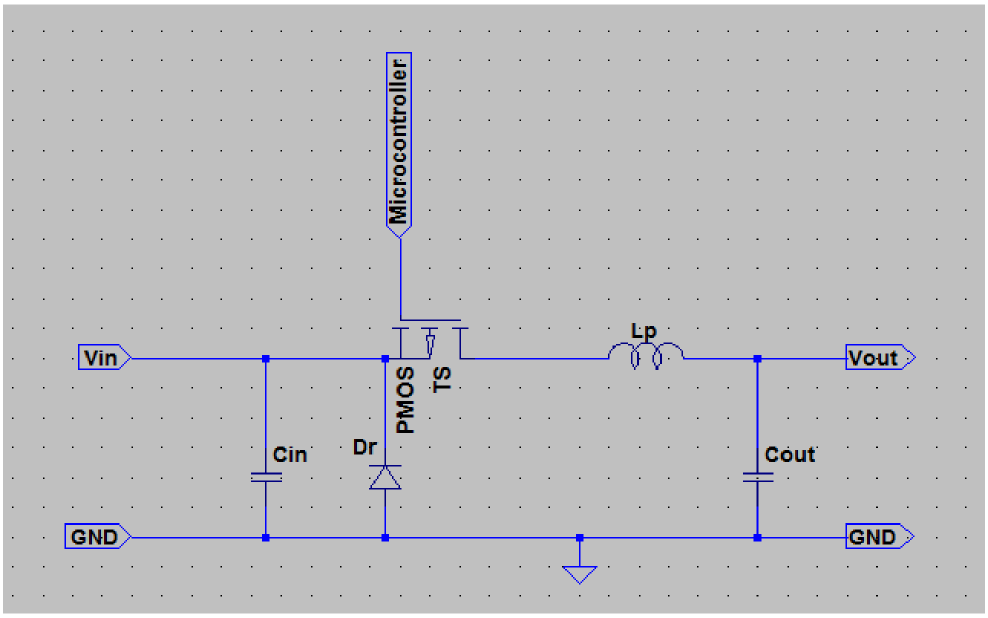


Figure 4-2: Step-down DC-DC converter circuit diagram

The main components along with their symbols and functions are listed below:

- T_s: Switching transistor
  - Controls the duty cycle of the converter and consequently the output voltage and input current of the system
- L_p: Power inductor
  - Keeps the current in the circuit from rising too high as the output capacitor is charged and also stores energy to be used during the discharge phase [23].

- $\boldsymbol{C_{out}}$: Output capacitor
    - Reduces the output voltage ripple of the circuit
- $\boldsymbol{C_{in}}$: Input capacitor
    - Reduces the output voltage ripple of the PV panels
- $\boldsymbol{D_r}$: Rectifying diode
    - Regulates flow during the discharge phase of the circuit, in order for the inductor to discharge its energy every time the voltage input source is removed.

Figure 4-3 illustrates the operation of a step-down converter. When the transistor switch is in the closed position, the voltage source charges the inductor and capacitor. As the transistor switch opens up, a voltage drop occurs and the inductor and capacitor releases the stored energy towards the output of the circuit [24].

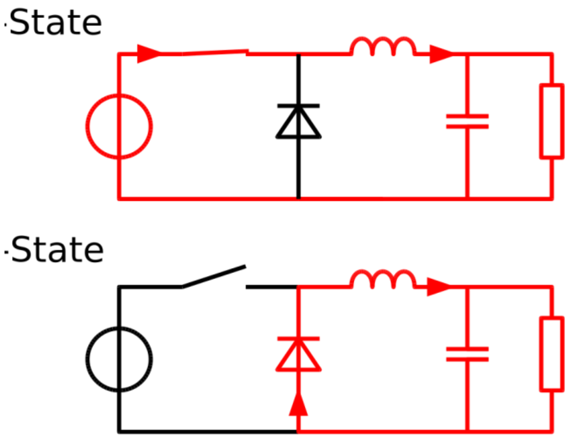


Figure 4-3: Circuit diagram illustrating the operation of a step-down DC-DC converter [24]

The following specifications were calculated for the selection of the converter's main components:

- Duty cycle of the transistor
- Inductor current
- Capacitor ripple voltage
- Transistor currents
- Minimum inductance of L_p
- Minimum capacitance of C_{out}
- Minimum capacitance of C_{in}

The following assumptions are made to simplify the calculations, without loss to the functionality of the calculated results [25]:

- The inductor, L_p, has negligible resistance and it has a constant inductance. It can be seen as an ideal inductor.

- The capacitors, C_{out} and C_{in}, have negligible resistance and they a constant capacitance. They can be seen as ideal capacitors.
- Due to the fact that L_p is seen as an ideal inductor and C_{out} is seen as an ideal capacitor, they are assumed to form an ideal LC filter. This assumption provides that the output voltage ripples and input current ripples are not significant.
- The conduction resistance of T_s is assumed negligible and thus an ideal switch.
- All of the power losses associated with the circuit are not taken into account, as the focus of the research is not in optimizing the efficiency of the MPPT power electronics.

Duty cycle of the transistor

The components should be selected based on the maximum work performance that is required. The duty cycle of the transistor is used in many of the calculations and the maximum conversion duty cycle shall be used.

A definition of the duty cycle of the circuit, is the percentage of time that the output voltage of the transistor switch is high, i.e. the percentage of time that the transistor switch is closed [26].

The minimum duty cycle of the step-down converter can be calculated as follows:

$$\delta_{min} = \frac{V_{outMin}}{V_{mpptMax}} \qquad [25] \qquad (4.3)$$

Where:

δ_{min} - Minimum duty cycle of the step-down DC-DC converter transistor switch

V_{outMin} - Minimum output voltage of the converter (V)

$V_{mpptMax}$ - Maximum output voltage of the solar panels (V)

From Table 1, the nominal battery voltage of 12V is the value of V_{outMin} and the maximum input voltage of 19.28V the value of $V_{mpptMax}$. Substituting these values into equation 7.1 yields the following:

$$\delta_{min} = \frac{12}{19.28}$$

$$\rightarrow \boldsymbol{\delta_{min} = 0.62}$$

Minimum inductance of L_p

The inductor must be able to sustain the peak current of the circuit without being saturated and the maximum peak-to-peak ripple current should be kept below 60% of the maximum output current [25].

Using Ohm’s law, the following equation is formulated for the maximum output current:

$I_{outMax} = \frac{P_{max}}{V_{outMin}}$ [25] (4.4)

4

Where:

I_{outMax} - Maximum output current of the step-down converter circuit (A)

P_{max} - Maximum power transferred by the converter (W)

The maximum peak-to-peak ripple current of the inductor is formulated below:

$I_{ppMax} = \frac{V_{outMin}}{L_p} \times \frac{1-\delta_{min}}{F_{swMin}}$ [25] (4.5)

Where:

I_{ppMax} - Maximum peak-to-peak current ripple of the inductor L_p (A)

L_p - Chosen inductance of the inductor (H)

F_{swMin} - Minimum switching frequency of the transistor T_s (Hz)

$I_{ppMax} \leq 0.6I_{outMax}$ (60% of maximum output current) [25] (4.6)

Substituting equation 4.5 into equation 4.6 yields:

$0.6I_{outMax} \geq \frac{V_{outMin}}{L_p} \times \frac{1-\delta_{min}}{F_{swMin}}$ (4.7)

Multiplying both sides of the resulting equation with L_p and dividing both sides with $0.6I_{outMax}$ results in:

$L_p \geq \frac{V_{outMin}}{0.6I_{outMax}} \times \frac{1-\delta_{min}}{F_{swMin}}$ (4.8)

Substituting equation 4.7 into equation 4.8 yields the first inductor selection criteria:

$\rightarrow \boldsymbol{L_p} \geq \frac{\boldsymbol{V^2_{outMin}}}{\boldsymbol{0.6P_{Max}}} \times \frac{\boldsymbol{1-\delta_{min}}}{\boldsymbol{F_{swMin}}}$ [25] (4.9)

The current that the inductor must accommodate without saturating can be formulated as follows:

$I_{pk} = \frac{P_{max}}{V_{outMin}} + \frac{V_{outMin}}{2.L_p} \times \frac{1-\delta_{min}}{F_{swMin}}$ [25] (4.10)

Where:

I_{pk} - The peak current that inductor L_p must accommodate (A)

L_p - The inductance of L_p as selected for trial and error method of comparing the minimum saturation current (H)

In an MPPT, the input current to the system is controlled in order to find the maximum power point of the solar panels [27]. As can be seen from equation 4.10, the peak current through the inductor is directly proportional to the duty cycle and inversely proportional to the switching frequency of the PWM pulse. The duty cycle is by definition also directly proportional to the input/output voltage ratio of the system, see equation 4.3. Therefore the output current of the solar panel is best controlled using the frequency parameter of the system, because in changing the duty cycle this will influence the output voltage of the MPPT.

Minimum capacitance of C_{out}

The formula for the maximum net charge exchange within the capacitor during a switching cycle is given as:

$$\Delta Q_{max} = \frac{1}{8} \times \frac{I_{ppMax}}{F_{swMin}} \qquad [25] \qquad (4.11)$$

Where:

ΔQ_{max} - Maximum net charge exchange within the output capacitor of the step-down converter (C)

Combining equation 4.5 and equation 4.11 results in:

$$\Delta Q_{max} = \frac{(1-\delta_{min}) * V_{outMin}}{8.F_{swMin}^2 * L_p} \qquad [25] \qquad (4.12)$$

The maximum output ripple voltage can be estimated as:

$$\Delta V_{outMaxPP} = \frac{\Delta Q_{max}}{C_{out}} \qquad [25] \qquad (4.13)$$

Where:

$\Delta V_{outMaxPP}$ – Maximum output ripple voltage of the step-down converter (V)

C_{out} – Output capacitance of C_{out} as selected to keep the output ripple voltage within the specification of the design process (conditional on the performance required from the converter) (F)

Combining equations 4.12 and 4.13; and exchanging terms will yield the capacitance value of C_{out}, for a specified preferred value of $\Delta V_{outMaxPP}$.

Minimum capacitance of C_{in}

The formula for the maximum net charge exchange within the capacitor during a switching cycle is the same as that of the output capacitor.

Switching transistor current

A formula to determine the maximum RMS current of the switching transistor is given below:

$$I_{Ts\_RMS} = \frac{P_{max}}{\sqrt{V_{outMin} * V_{mppMax}}} \qquad [25] \qquad (4.14)$$

Rectifying diode currents

The maximum current that will flow through the rectifying diode is formulated as the maximum power accommodated by the system, divided by the minimum output voltage [25].

$$I_{DrMax} = \frac{P_{max}}{V_{outMin}} \qquad [25] \qquad (4.15)$$

Selection of power inductor L_p

Equation 4.9 is used to calculate the minimum inductance value for L_p.

$$L_p \geq \frac{V_{outMin}^2}{0.6P_{Max}} \times \frac{1-\delta_{min}}{F_{swMin}} \qquad [25] \qquad (4.16)$$

The following values can be substituted in equation 4.9 in order to determine the minimum inductance of L_p:

- $\delta_{min} = 0.62$ Determined using equation 4.3
- $P_{Max} = 30W$ Maximum power supplied by solar panels
- $V_{outMin} = 12V$ Minimum voltage required to charge the vehicle's battery
- $F_{swMin} = 50kHz$ Minimum switching frequency applied to the switching transistor, as selected from the micro controller.

This produces:

$$L_p \geq \frac{12^2}{0.6*30} \times \frac{1-0.62}{50*10^3}$$

$\rightarrow \boldsymbol{L_p \geq 60.8\mu H}$

From the above, it is clear that the minimum induction value of L_p can be adjusted according to the switching frequency output selected from the micro controller.

By substituting the above parameters into equation 4.10 the maximum peak current that the inductor must accommodate is calculated for the minimum induction value of $60.8\mu H$ for L_p.

$$I_{pk} = \frac{P_{max}}{V_{outMin}} + \frac{V_{outMin}}{2 * L_p} \times \frac{1-\delta_{min}}{F_{swMin}} \qquad [25] \qquad (4.10)$$

$$I_{pk} = \frac{30}{12} + \frac{12}{2 * 60.8 * 10^{-6}} \times \frac{1-0.62}{50 * 10^3}$$

$$\rightarrow \boldsymbol{I_{pk} = 3.25A}$$

If the chosen induction value for L_p was very large or alternatively the switching frequency of the transistor very fast, then equation 4.10 can be simplified to determine the absolute minimum current that the system can be designed for. This is used as guide to select the inductor, even when the switching frequency is changed.

$$I_{pk_{absMin}} = \frac{P_{max}}{V_{outMin}} \qquad (4.13)$$

Which yields:

$$I_{pk_{absMin}} = \frac{30}{12}$$

$$\rightarrow \boldsymbol{I_{pk_{absMin}} = 2.5A}$$

According to the above results, a toroid ferrite core >200µH inductor was chosen for the MPPT power circuit. Substituting this inductance value into equation 4.10 results in:

$$I_{pk} = \frac{30}{12} + \frac{12}{2 * 200 * 10^{-6}} \times \frac{1-0.62}{50 * 10^3}$$

$$\rightarrow \boldsymbol{I_{pk} = 2.73A}$$

Copper wire with a diameter of 2mm has a maximum continuous DC current rating of 32A [28] for a single core, this is much higher than needed and 2mm transistor wire to manufacture the inductor is sufficient for this project.

The inductor is manufactured by hand. A toroid ferrite core, T-290 with an OD of 73.65mm, ID of 38.85mm and a thickness of 12.7mm was chosen. The transistor wire of diameter 2mm was coiled around the core to produce more or less the right amount of inductance. The accuracy of the inductance has fairly high tolerance, since the system is over designed by some margin, as long as the inductance is large enough to ensure that the inductor does not get saturated during operation.

The formula to calculate the number of turns required for a toroid inductor of a specific inductance is as follows:

$$number\ of\ turns = 1000 * \sqrt{\frac{L}{A_L}} \text{ [29]} \tag{4.17}$$

Where:

L - Inductance required. (mH)

A_L - Relationship between the inductance and number of turns. (mH/1000turns)

For the specific core selected, the A_L value is 4,880mH/1000turns.

Therefore:

$$number\ of\ turns = 1000 * \sqrt{\frac{0.2}{4880}}$$

$$number\ of\ turns = \mathbf{6.4\ turns} \rightarrow$$

Approximately 33 turns were used to produce the required inductor, which has an inductance in excess of 200µH.

Figure 4-4 is an image of the manufactured inductor L_p.

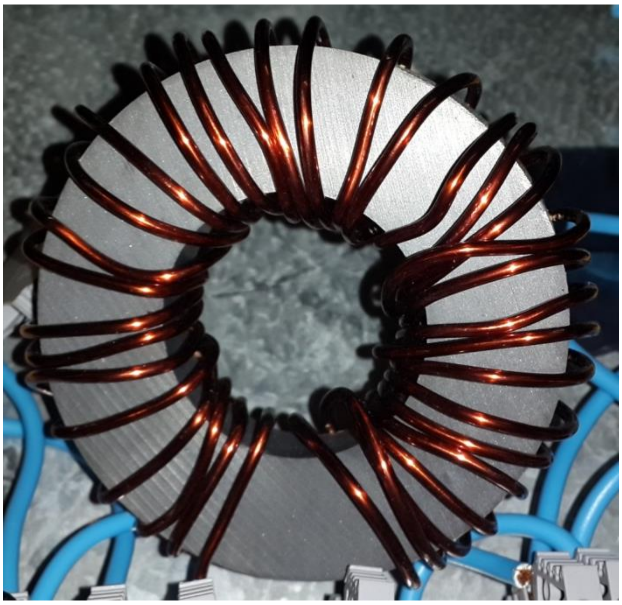


Figure 4-4: Manufactured inductor L_p for the MPPT power circuit

The specifications of L_p are summarized in Table 4-3.

Inductance	>200µH

Maximum DC current	32A

Table 4-3: Summary of power inductor L_p specifications

Selection of output capacitor C_{out}

A combination of equation 4.12 and 4.13 is used to calculate the minimum output filter capacitance, C_{outMin}.

$$C_{outMin} = \frac{(V_{MPPmax} - V_{outMin}) * V_{outMin}}{8.F_{swMin}^{2} * L_p * V_{MPPmax} * \Delta V_{outP-P}} \qquad [25] \qquad (4.14)$$

The following values can be substituted in equation 4.14 in order to determine the minimum capacitance value of C_{outMin}:

- $V_{MPPmax} = 19.28V$ Determined using equation 4.3
- $V_{outMin} = 12V$ Minimum voltage required to charge the vehicle's battery
- $F_{swMin} = 50kHz$ Minimum switching frequency applied to the switching transistor, as selected from the micro controller.
- $L_p = 200\mu H$ Inductance of the selected inductor L_p
- $\Delta V_{outP-P} = \ 5\% * V_{outMax} = 0.05 * 14.7V = 0.735V$ Peak-peak ripple voltage

Which yields:

$$C_{outMin} = \frac{(19.28-12)*12}{8.(50*10^3)^2 * 200*10^{-6} * 19.8 * 0.735}$$

$$\rightarrow \boldsymbol{C_{outMin} = 1.5\mu F}$$

From equations 4.12 and 4.13 it can be seen that the relationship between the values of L_p and C_{out} are inversely proportional to each other. This implies that if a larger inductor is selected, then a smaller capacitor can be selected and vice versa.

The selected capacitor must have a maximum voltage tolerance greater than V_{outMax} [25]. Therefore, the voltage tolerance of the output capacitor must be greater than 14.7V. A capacitor much greater than 1.5µF is required. This taken into account, an aluminium electrolytic capacitor was selected with a capacitance of 330µF and tolerance voltage of 200V. Figure 4-5 is an image of the selected capacitor C_{out}.

Figure 4-5: Selected capacitor C_out for the MPPT power circuit

The specifications of C_{out} are summarized in Table 4-4.

Panasonic ED RADIAL	
Capacitance	330µF
Maximum Voltage	200V
Tolerance	20%

Table 4-4: Summary of output capacitor C_out specifications

Selection of output capacitor C_{in}

The calculation of C_{in} is done similarly as that of C_{out}. The voltage across the capacitor is the only difference and the following formula is used:

$$C_{inMin} = \frac{V_{outMax} - V_{MPPmin}}{8.F_{swMin}^2 * L_p * \Delta V_{inP-P}} \qquad [25] \qquad (4.14)$$

The following values can be substituted in equation 4.15 in order to determine the minimum capacitance value of C_{inMin}:

- $V_{MPPmin} = 11.78V$ Determined using the maximum test temperature and the voltage temperature coefficient of the panel
- $V_{outMax} = 14.7V$ Minimum voltage required to charge the vehicle's battery
- $F_{swMin} = 50kHz$ Minimum switching frequency applied to the switching transistor, as selected from the micro controller.
- $L_p = 200\mu H$ Inductance of the selected inductor L_p
- $\Delta V_{inP-P} = 5\% * V_{MPPMin} = 0.05 * 11.78V = 0.589V$ Peak-peak ripple voltage

Which yields:

$$C_{inMin} = \frac{14.7 - 11.78}{8.(50*10^3)^2 * 200*10^{-6} * 0.589}$$

$\rightarrow \boldsymbol{C_{inMin} = 1.24\mu F}$

This is similar to the minimum value allowed for C_{outMin}. Consequently the same capacitor was selected for the input capacitor.

Selection of switching transistor $\boldsymbol{T_s}$

Equation 4.14 is used along with the following parameters to determine the maximum RMS current that the switching transistor must accommodate:

- $P_{Max} = 30W$ Maximum power supplied by solar panels
- $V_{outMin} = 12V$ Minimum voltage required to charge the vehicle's battery
- $V_{mppMax} = 19.28V$ Maximum voltage that the solar panel can provide

Substituting these parameters into equation 4.14 produces:

$$I_{Ts\_RMS} = \frac{30}{\sqrt{12 * 19.28}}$$

$\rightarrow \boldsymbol{I_{Ts\_RMS} = 1.97A}$

The voltage rating (drain-source voltage) of the transistor must be greater than the open circuit voltage of the solar panel [25]. A solar panel's open circuit voltage is its maximum voltage and in this instance it is 23.48V.

Another important parameter to consider is the gate voltage required to switch the transistor, allowing the maximum required current to flow through. The maximum pin output voltage of the selected microcontroller PWM is 5V when used in the open drain configuration.

A p-channel MOSFET was selected with a fast switching frequency of 8MHz. Figure 4-6 is an image of the selected transistor and Table 4-5 is a summary of the most important specifications of the component.

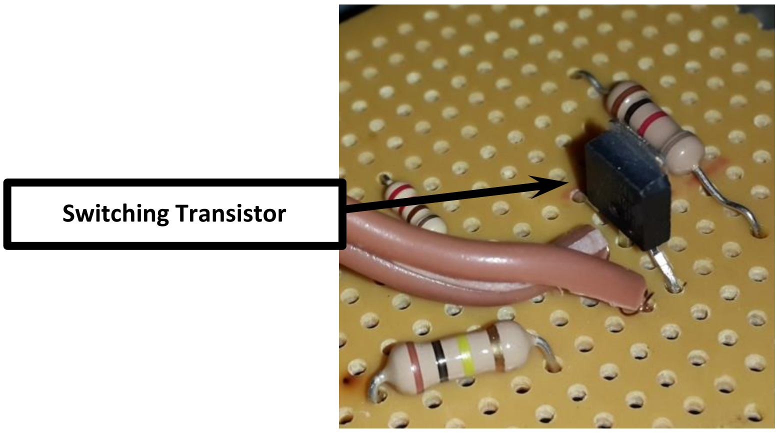


Figure 4-6: Selected switching transistor T_s for the MPPT power circuit

IRFU9024N PMOS	
Maximum Drain-Source Voltage	-60V
Maximum Drain Current	-8.8A
Switching Frequency	8MHz
Gate voltage (currents > 1.22A)	±20V

Table 4-5: Summary of the switching transistor T_s specifications

A small N-channel MOSFET was used to switch gate voltage of the P-channel MOSFET. The NMOS pulls the gate of the PMOS to ground when the switch is closed and pulls the gate up to the panel voltage when open.

Selection of rectifying diode $\boldsymbol{D_r}$

The following parameters are used along with equation 4.14 to determine the maximum current that the rectifying diode needs to accommodate:

- $P_{Max} = 30W$ Maximum power supplied by solar panels
- $V_{outMin} = 12V$ Minimum voltage required to charge the vehicle's battery

Substituting these values into equation 4.14 produces the following result:

$$I_{DrMax} = \frac{30}{12}$$

$$\rightarrow \boldsymbol{I_{DrMax} = 2.5A}$$

The maximum voltage threshold of the diode needs to be larger than the maximum output voltage of the system [25]. In this instance it is 14.7V.

A silicon junction diode was elected. Figure 4-7 is an image of the selected diode:

Figure 4-7: Selected rectifying diode D_r for the MPPT power circuit

Table 4-6 is a summary of the rectifying diode specifications.

Vishay VS-30CTH02-N3	
Forward Bias Voltage	0.92V
Maximum Current	15A
Switching Time	26ns
Maximum Voltage Threshold	200V

Table 4-6: Summary of the rectifying diode D_r specifications

4.1.2 Manufacturing of MPPT Power Electronics Circuit

A modular approach was followed to manufacture the MPPT. Different sections of the power electronics circuit was built at a time using insulated copper wire, terminal connector blocks on a rail and a few prototype circuit boards. Figure 4-8 and Figure 4-9 below are images of the circuit, mounted onto a galvanised steel base plate for structural stability. The transistor circuits are built on a prototype circuit board.

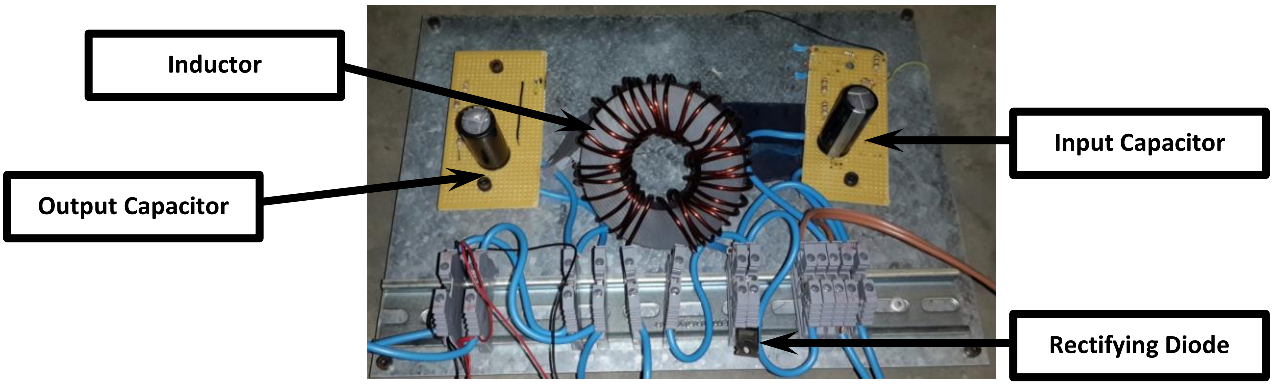


Figure 4-8: MPPT Power Electronics Circuit

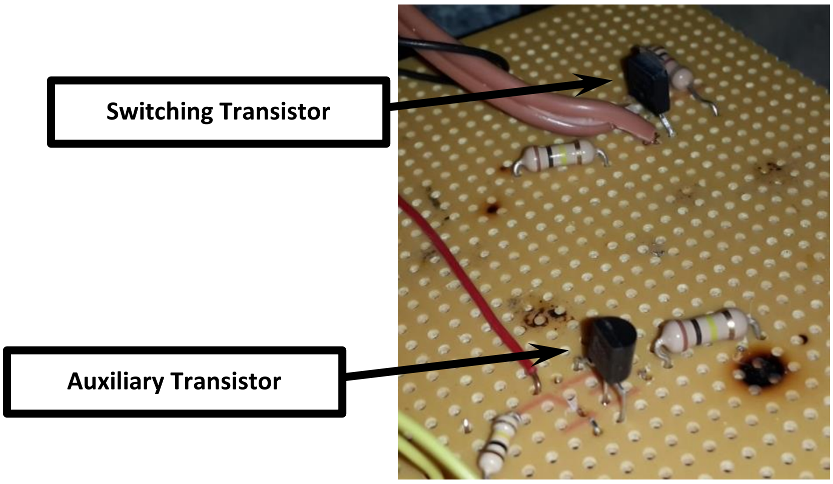


Figure 4-9: Transistor switching circuits

4.1.3 Testing of MPPT Power Electronics Circuit

After connecting the battery and solar panels, the circuit was initially tested by manually applying voltage to the gate of the NMOS transistor and switching the PMOS transistor, thus measuring that the solar panel output voltage switches from open circuit voltage to the battery voltage.

The potentiometer, that forms part of the control circuit detailed in the next section, was then used to manually adjust the duty cycle of the switching transistor. The output voltage of the solar panel was monitored to verify that the complete range of voltages were produced as the potentiometer was adjusted from minimum resistance (solar panel operating at open circuit voltage) to maximum resistance (solar panel operating at battery voltage).

As the tests were concluded it was decided that:

- The circuit isolates the battery from the solar panels.
- The full solar panel output voltage range was verified.
- The time response of the circuit was satisfactory.
- Even though some of the components were overdesigned, the optimization of weight, efficiency and response time of such a circuit falls outside the scope of this research project.

4.1.4 Design Modifications

Some of the circuit's components were damaged during testing and transportation and were consequently replaced. The same quick tests were done to verify that the circuit is restored and that the other circuitry won't be damaged of connected back into the MPPT system.

4.1.5 Final Prototype MPPT Tests

The final power electronics circuitry tests were done as part of the main MPPT system. These tests revolved more around the control circuitry and are detailed in the next section.

4.2 MPPT Control Circuit

Figure 4-10 is a diagram illustrating the MPPT control circuit.

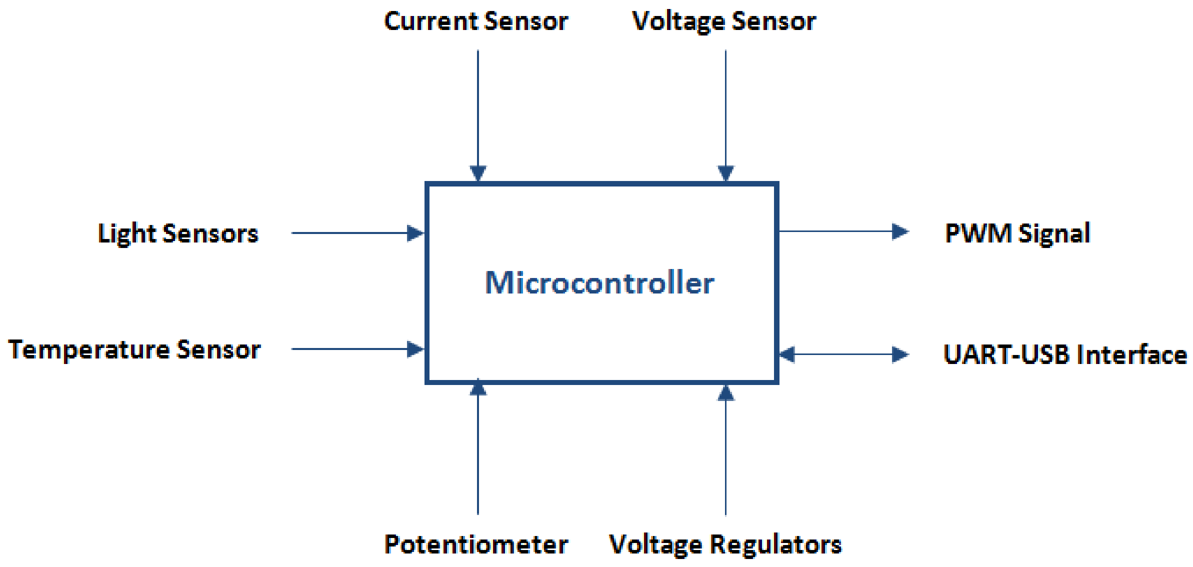


Figure 4-10: MPPT control circuit diagram

The solar vehicle requires one microcontroller unit. This unit, among other functions, controls the MPPT power circuits, which is its main function.

The MPPT control circuit contains the following components and interfaces:

- Microcontroller
    - Implementation of the MPPT algorithm
    - Analogue to digital converter for measuring inputs to the controller
    - Pulse-Width-Modulation (PWM) module to generate output control signal
    - General switching and indication via digital outputs
- Light sensor
    - Estimation of the solar irradiance incident on the solar panels through measuring the light intensity in lux.
- Temperature sensor
    - Measurement of the solar panel temperature
- Current sensors
    - Measuring the output current of the solar panel (Input current of the MPPT)
    - Measuring the output current of the MPPT
- Voltage sensors
    - Measuring the output voltage of the solar panel (Input voltage of the MPPT)
    - Measuring the output voltage of the MPPT
- USB communication interface
    - Transmits MPPT data real time to a computer (Notebook)

- Potentiometer
    - Manual control of MPPT for testing
- Voltage regulators
    - 3.3V voltage supply for the microcontroller
    - 5V voltage supply for transistor switching

4.2.1 Design of MPPT Control Circuit

This section outlines the requirements and selection of the main MPPT control circuit components. These include:

- Microcontroller
- Light sensor
- Temperature sensor
- MPPT input current sensor
- MPPT input voltage sensor
- MPPT output voltage sensor
- USB communication interface
- Potentiometer
- Voltage supply regulator for microcontroller
- Voltage supply regulator for transistor switching

Microcontroller

The microcontroller is the core of the MPPT control circuit. All of the MPPT algorithm (Except learning/teaching algorithm) and monitoring software is executed on this CPU. The analogue sensor inputs are monitored and converted by the microcontroller and the transistor switching control output (PWM) is generated by it.

With the above in mind, the following required specifications for the microcontroller were set:

1. At least one analogue-to-digital converter (ADC) module is required capable of monitoring at least seven analogue inputs in series. Real-time parallel monitoring would be ideal for high speed performance, but this is not required for the tests done in this research.
2. One PWM module to produce the control signal for the MPPT switching transistor. A minimum frequency of 50kHZ is required.
3. If the microcontroller is 3.3V based, then some of the pins must be 5V tolerant for better resolution on sensor inputs, as well as open drain source voltage for the PWM control signal.
4. Enough on-board flash memory to contain all of the control software.
5. Enough RAM to for the system to be functional.
6. Enough digital I/O pins for monitoring and switching.

A microcontroller from Microchip Technology Incorporated was selected. The specifications for the PIC24HJ128GP502 chip are summarized in Table 4-7 [30]. From Table 4-7 it is clear that the selected Microchip meets the research requirements.

PIC24HJ128GP502 Microcontroller	
ADC module	10/12-bit ADC, 13 ADC input channels
PWM module	Four output compare modules that can be implemented as PWM outputs
Voltage requirements	3.0-3.6 operating voltage 5V-tolerant pins Selectable open drain and internal pull-ups
Flash memory	128kb flash memory
RAM	8k SRAM
Digital I/O's	21 I/O pins

Table 4-7: Summary of specifications for the selected microcontroller to meet project requirements [30]

The microchip is available in a SPDIP package that simplifies the manufacturing of the control test circuit.

The microchip is also easily programmable in the C-language which is preferred by the user for this specific application. The MPLAB ICD 2™ module is available for debugging and programming of the chip. The MPLAB IDE 8.6 available from Microchip® is adequate software to develop the necessary software for the microcontroller.

Figure 4-11 is an image of the selected microcontroller.

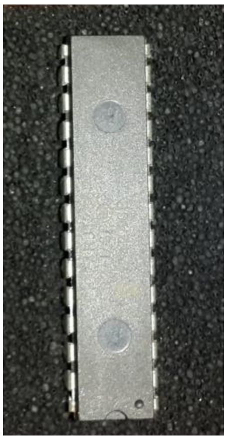


Figure 4-11: PIC24HJ128GP502 microcontroller SPDIP package

Light sensor

A light sensor is required by the control system in order to execute the MPPT control algorithm.

The light sensor is required to be small enough to fit on top of the solar panels, without casting a significant shadow over them. The sensor must also have an adequate lux range and operating voltage.

The sensor selected is the MLG5506 from MOBICON Electronic Components. Light dependant resistor (LDR). It has a resistance value of 4kΩ at 10 lux.

Figure 4-12 is an image of the selected light sensor.

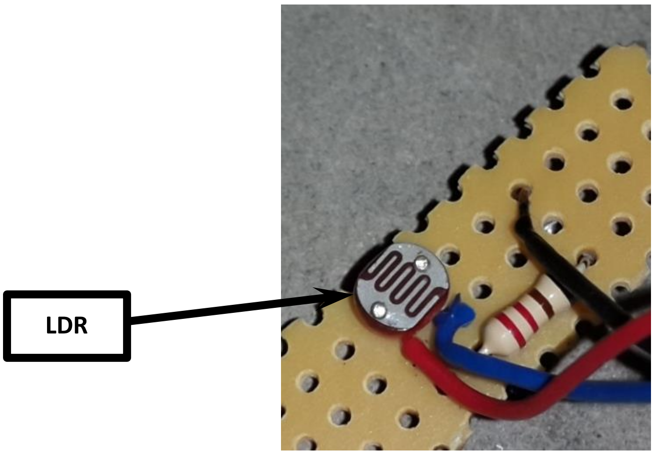


Figure 4-12: Selected light sensor

A second light sensor is required to identify partial shadowing of the solar panels.

Temperature sensor

A temperature sensor is required by the control system to input an estimate of the PV cell temperature to the MPPT control algorithm.

The sensor needs to be small enough to fit on top of the PV panel without casting a significant shadow over it. The measurable temperature range required is 20°C - 60°C for the scope of this research. The measuring tolerance can be quite large at about 1°C.

The LM35 centigrade temperature sensor was selected. Its temperature range is -55°C - 150°C.

The sensor was taped in the corner of the solar panel on top of the glass to give an estimate reading of the PV cell temperature.

The sensor was supplied with 5V and the output voltage could be read directly by microprocessor.

Figure 4-13 is an image of the selected temperature sensor.

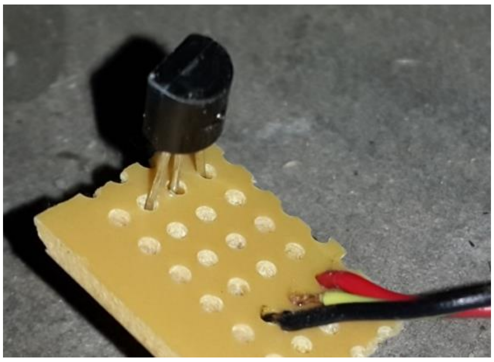


Figure 4-13: Temperature Sensor

MPPT input current sensor

The MPPT input current sensor is used to determine, in conjunction with the input voltage, the power point of the PV panel. The current measurement also indicates the PV panel point of operation on its I-V curve.

The absolute maximum output current of the PV panel is 2.02A and the absolute maximum output voltage is 23.48V. The maximum sensor specifications need to be larger than these limits.

It was decided to estimate the panel input current through measuring the voltage drop across a 1Ω 30W resistor, connected in series to the negative terminal of the solar panel. The voltage drop is measured directly by the microprocessor.

Figure 4-14 is an image of the selected current sensor for the input of the MPPT.

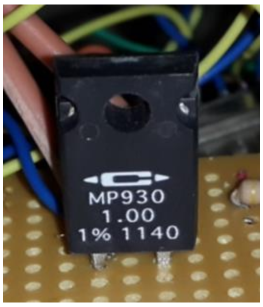


Figure 4-14: Selected current sensor

MPPT input voltage sensor

The input voltage of MPPT is the same as the output voltage of the PV panel. This voltage is used to determine the PV panel output power as well as the point of operation on its I-V curve.

A simple voltage divider circuit is adequate to measure output voltage of the PV panel. The absolute maximum voltage to be measured is 23.48V. Considering that 3.3V is the minimum pin tolerance of the microcontroller, the resolution of the measurement is:

$$\frac{3.3}{23.48} = \mathbf{0.14V}$$

This resolution is suitable for accurate measurement by the microcontroller.

Figure 4-15 is a simple diagram of a universal voltage divider circuit.

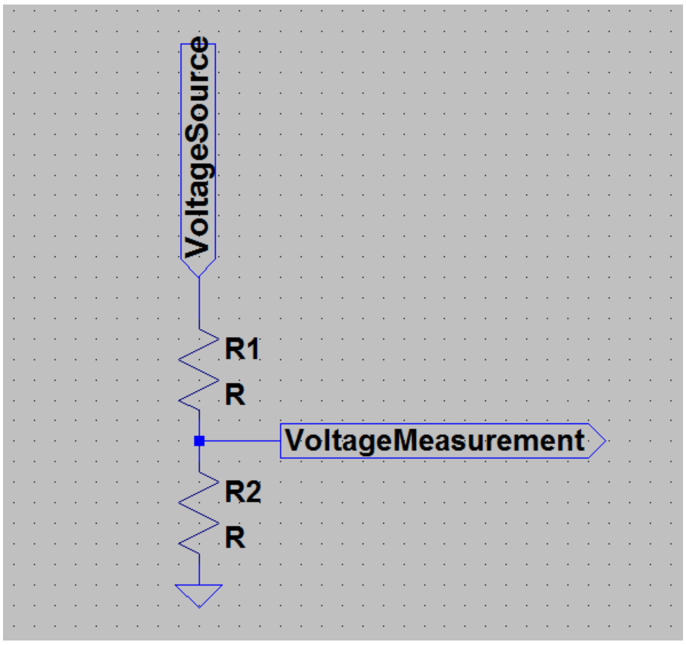


Figure 4-15: A simple universal voltage divider circuit

The current though R_1 and R_2 is same and through Ohm's law the following relationship is formulated:

$$\frac{V_1}{R_1} = \frac{V_2}{R_2} \tag{4.16}$$

Multiplying both sides of equation 7.15 by R_1 and dividing by V_2 results in:

$$\frac{V_1}{V_2} = \frac{R_1}{R_2} \tag{4.17}$$

With the voltage to be measure across R_2, the value for V_2 is set at 3.3V. Consequently the value for V_1 is:

$$23.48 - 3.3 = \mathbf{20.18V}$$

This yields a ratio of 6.1 between V_1 and V_2, which is the same as between R_1 and R_2.

The maximum power of the resistor will be reached in the one with the highest voltage across it first, V_1 in this case. If 0.25W resistors are used for the circuit, then the minimum resistance value for R_1 can be calculated according to Ohm's law:

$$R = \frac{V^2}{P} \qquad (4.18)$$

$$R_1 = \frac{20.18^2}{0.25}$$

$$\rightarrow \boldsymbol{R_1 = 1628.93\Omega}$$

If the value for R_1 is substituted into equation 7.16, then the value for R_2 is:

$$\frac{1628.93}{6.1} = \mathbf{267.04\Omega}$$

Standard resistor values that approach the values calculated above are 1800Ω for R_1 and 270Ω for R_2 [31]. This is a resistance ratio of 6.67 as opposed to 6.1. The value for R_2 is more or less the same, thus with R_1 being larger, the voltage over R_2 is slightly less than 3.3V. The standard values for R_1 and R_2 is acceptable for this application.

MPPT output voltage sensor

The output voltage of the MPPT is the same as the input voltage of the battery. This measurement is used to calculate the output power of the MPPT. The voltage sensor is also used to monitor the battery voltage, especially with regards to over-voltage protection.

A similar voltage divider circuit is used to the one implemented for the MPPT input voltage. The measuring resolution with a maximum voltage of 14.7V is:

$$\frac{3.3}{14.7} = \boldsymbol{0.22V}$$

With the value for V_2 set at $3.3V$ and $V_1 = 14.7 - 3.3 = 11.4V$, the voltage ratio is 3.45.

Using equation 4.17 and 0.25W resistors the minimum resistance value for R_1 is calculated:

$$R_1 = \frac{11.4^2}{0.25}$$

$$\rightarrow \boldsymbol{R_1 = 519.84\Omega}$$

Substituting this value into equation 4.17 to calculate R_2 yields:

$$\frac{519.84}{3.45} = \mathbf{150.68\Omega}$$

Standard resistor values that approach the values calculated above are 560Ω for R_1 and 150Ω for R_2 [31]. This is a resistance ratio of 3.73 as opposed to 3.45. The value for R_2 is more or less the same, thus with R_1 being larger, the voltage over R_2 is slightly less than 3.3V. The standard values for R_1 and R_2 is acceptable for this application.

USB Communication Interface

A communication module is needed between the UART interface of the microprocessor and the USB interface of a computer for serial communication. This is not as practical as wireless communication, but it is inexpensive and adequate for this specific system.

The communication module selected, is based on the UMFT 230 XB-01 microcontroller from FTDI Chip and manufactured by them. Except for the drivers required, the module can be used as is.

Figure 4-16 is an image of the selected UART to USB conversion module for this project.

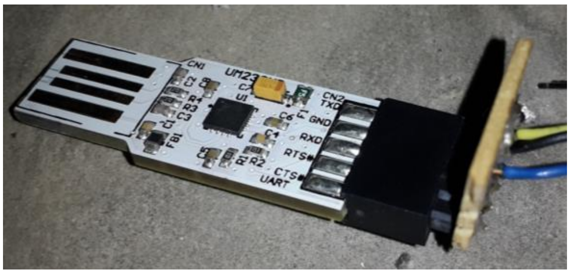


Figure 4-16: Selected UART to USB conversion module for serial communication

Potentiometer

A potentiometer is required to adjust the operating power point of the PV panel manually. This is a useful feature during initial tests, as well as calibrating the control system.

The potentiometer is used in a voltage divider circuit. A 10kΩ potentiometer is used in series with a 1kΩ resistor. When the potentiometer is tuned to maximum resistance, then the voltage ratio is 10. With a 3.3V input to the voltage divider circuit, the voltage range across the potentiometer is 0V – 2.97V.

Figure 4-17 is an image of the potentiometer chosen.

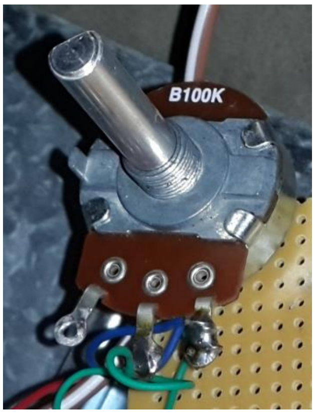


Figure 4-17: 10kΩ Potentiometer

Voltage supply regulator for microcontroller

A standard voltage regulator IC with a fixed output of 3.3V is needed to supply the control unit from the 12V battery.

The 41MK53E IC from Texas Instruments was selected and is presented in Figure 4-18.

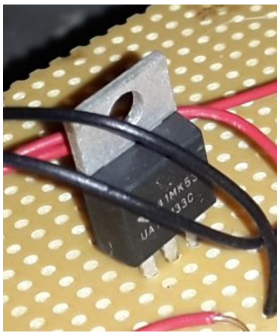


Figure 4-18: Selected 3.3V supply regulator

Voltage supply regulator for transistor switching

A standard voltage regulator IC with a variable output for an intended 5V is needed to supply the control unit open drain circuit from the 12V battery. The output voltage is controlled via a small potentiometer.

The LM317T IC from STMicroelectronics was selected and is presented in Figure 4-19.

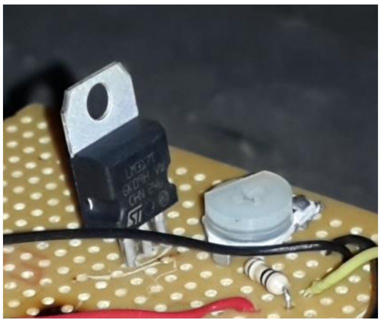


Figure 4-19: Selected 5V supply regulator

4.2.2 Manufacturing of MPPT Control Circuit

Several basic prototype circuit boards were used to manufacture the MPPT control circuit. The reason for this modular approach was to build and test each individual circuitry and maintaining this ability through the whole development process of the MPPT. The manufacturing process can be grouped into the following stages:

- Basic microcontroller implementation
- Implementation of light sensor
- Implementation of temperature sensor
- Implementation of current sensors
- Implementation of voltage sensors
- Implementation of USB communication interface

Microcontroller implementation

Being the central component for all the MPPT decisions, the microcontroller was implemented first and basic functionalities checked.

Figure 4-20 is an image after the basic RC components and programming interface.

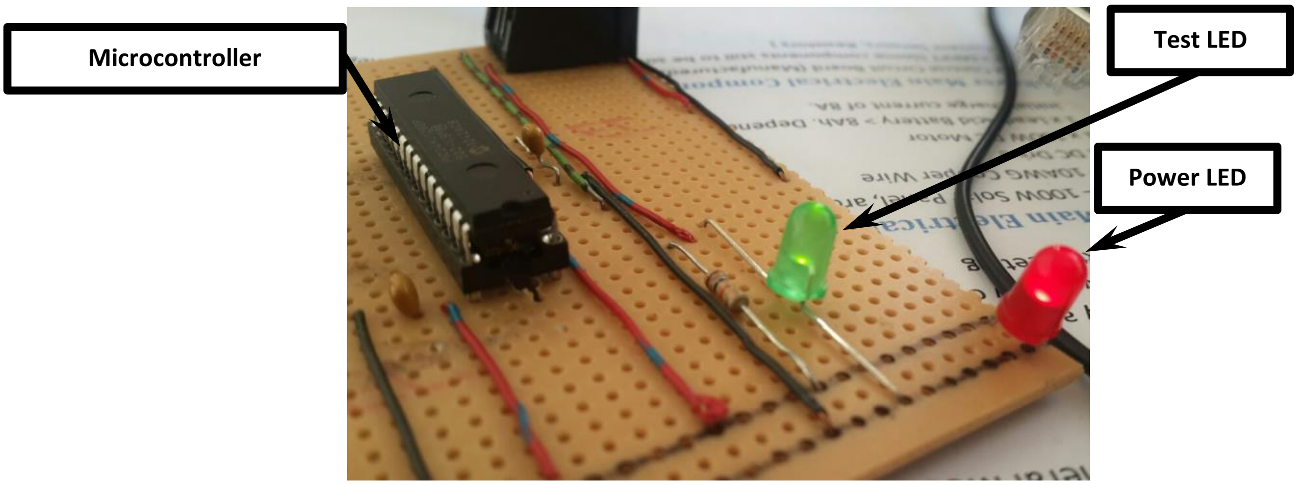


Figure 4-20: Basic microcontroller circuitry

The first program executed was to setup the internal processor clock and to test the digital outputs and a green LED was switched on to indicate basic microcontroller operation.

Table 4-8 is the final pin assignment of the microprocessor.

Pin Number	Function
1	MCLR
2	Hello World LED
3	ADC Negative Voltage Reference
4	PGED1
5	PGEC1
6	ADC input – LDR1
7	ADC input – LDR2
8	VSS
9	
10	
11	
12	
13	VDD
14	
15	
16	
17	UART+
18	UART-
19	VSS

20	VCAP
21	OC1 output - PWM
22	
23	ADC input – Potentiometer/TEMP
24	ADC input – Vsolar
25	ADC input - Isolar
26	ADC input - Vbat
27	AVSS
28	AVDD

Table 4-8: Microprocessor pin assignment

Light Sensor

Each light sensor is connected in series with a 2.2kΩ resistor. One pin of the light dependant resistor is connected to 3.3V, the other to the one pin of the 2.2kΩ resistor, which in turn is connected to ground. A voltage measurement is taken across the 2.2kΩ resistor via the ADC module of the microcontroller.

All of the other sensors and transistor were connected to the ADC module of the microprocessor for sequential monitoring of process inputs. The PWM output signal for the transistor was tested with an oscilloscope to ensure that frequency was correct.

After connecting the USB communication module as well, the complete control circuitry is presented in Figure 4-21.

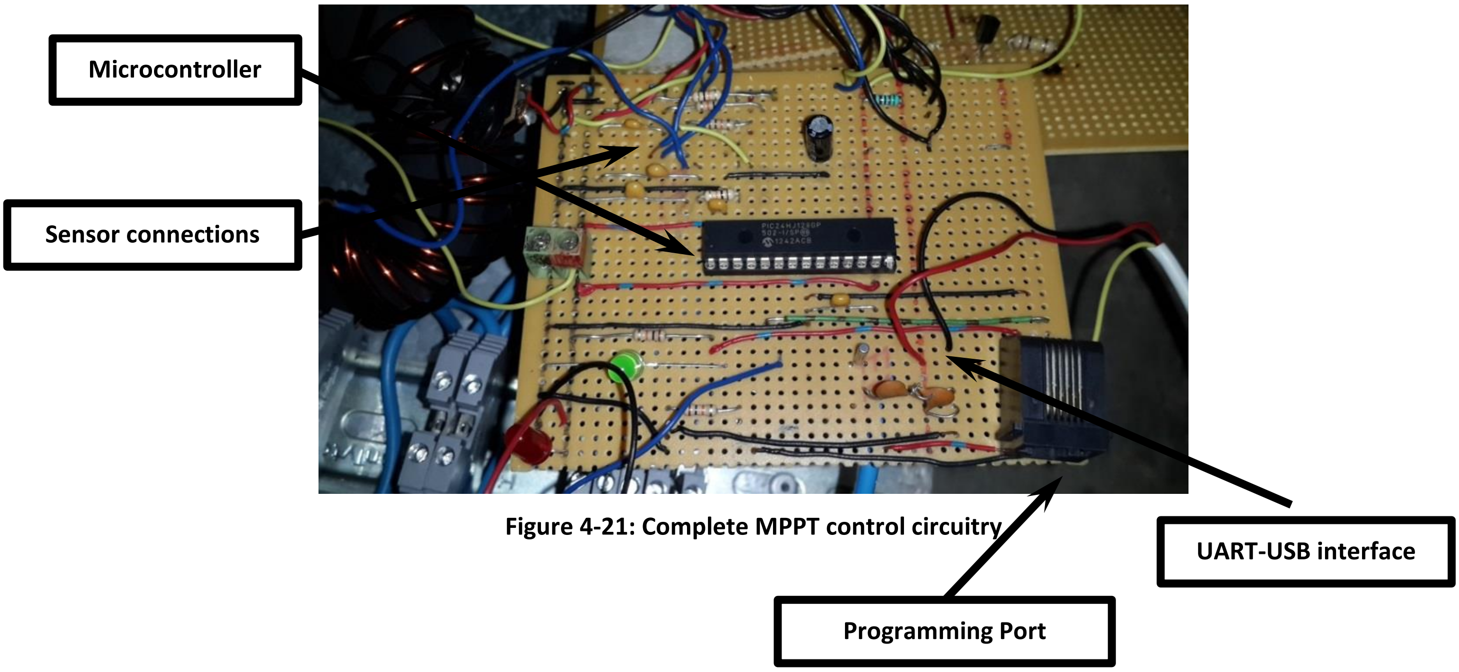


Figure 4-21: Complete MPPT control circuitry

4.2.3 Hardware Configuration Software

The software that is required to perform a basic setup of the microcontroller as well as the senor inputs, PWM output and UART communication consist of a large amount of program registers that need to be set. The following functions were created for this setup, excluding the initial coding to setup interrupts and the processor clock:

- IOInit(); //Initialize the I/O ports
- ADCInit(); //Initialize the A/D converter
- DMAInit(); //Initialize DMA
- ADCStart(); //Turn ADC on
- OC1Init(); //Initialize Output compare (PWM)
- UARTinit(); //Initialize UART

To view the setup of the registers, the full program software can be accessed on the DVD attached.

The microprocessor was configured to operate at an internal clock speed 23MHz.

4.2.4 Software – Input and Output Control

Basic input and output software was developed to test the ADC inputs and PWM output (One port each).

To test the ADC software the potentiometer was connected and active debugging was used with break points to read the processor shift register values to determine if the input is received.

The PWM software was tested by measuring the output with an oscilloscope to determine if the signal profile is correct.

Due to the fact that these software blocks were duplicated and used in the final MPPT software, it is not individually included in the software package.

4.2.5 Software – DC-DC Converter Software

The potentiometer tested in the previous section was used to manipulate the PWM output signal. This signal was used to drive the switching transistor of the MPPT in order to control the output voltage of the solar panel.

The output voltage of the panel was varied from open circuit to that of the battery.

4.2.6 Software – MPPT Test Software

After the hardware was confirmed with the control coming from the potentiometer, the software was changed to cycle through the duty cycle range of the MPPT.

Up to this point every section of the MPPT had been tested and the incorporation of each into a MPPT system could progress.

4.2.7 Testing of MPPT Control Circuits

All of the electronics were assembled and connected into the solar vehicle system. The MPPT was continuously looped through the duty cycle range and the values from the sensors were sent to the notebook and stored in a .txt file.

4.3 Conclusion: Chapter 4

This chapter details the design of the prototype test MPPT. The MPPT was designed to meet testing requirements and not to be a highly efficient product. The efficiency is similar for most test conditions, thus it was not required as a test variable.

The MPPT was manufactured and mounted on the solar vehicle.

Chapter 5 - Proposed MPPT Algorithm

Figure 5-1 is a schematic representation of the MPPT controller. After collecting the sensor inputs, the MPPT tracking function estimates the new MPP value and consequently the PWM is adjusted for the MPPT to operate at a new duty cycle. The MPPT data collection function in turn samples process parameters to the laptop where the learning algorithm can be used to generate new controller parameters. The MPPT can then be updated with these new parameters to see if the accuracy has improved.

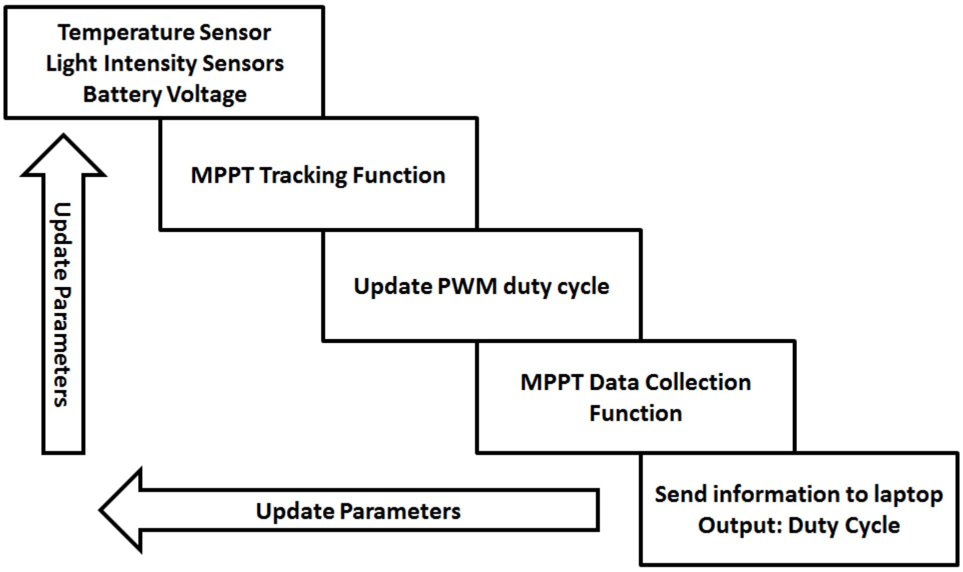


Figure 5-1: Schematic representation of MPPT controller

An overview to the development of a MPPT control algorithm is detailed in Figure 5-2. The following four steps were required for development:

- IV and PV curve data collection
    - Data collection gives insight into the functioning and performance of the acquired solar panels.
    - It also acts as a database of actual MPP locations for different external conditions and forms a good learning platform for the proposed control algorithm.
- Control system simulation
    - The proposed control system was simulated first in order to have a functioning algorithm before it is implemented on the hardware and tested.
    - This ensures better development time and safer testing.
- Update of control system parameters

  - Control system parameters are updated iteratively in order to improve its effectiveness.
- Implement control system on MPPT
  - After a good level of confidence in the algorithm was reached through simulation, the next step was to implement it on a real scaled solar vehicle prototype for feasibility tests.

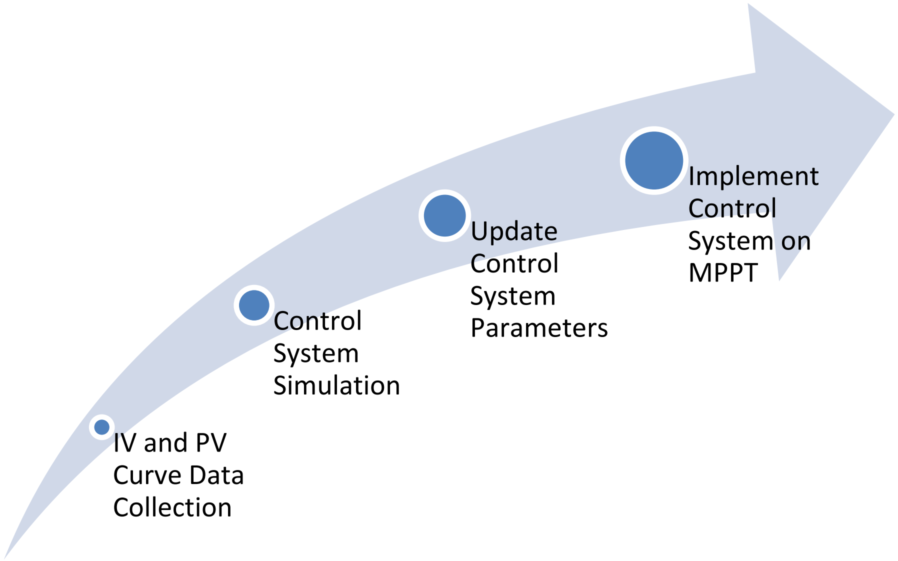

IV and PV Curve Data Collection
Control System Simulation
Update Control System Parameters
Implement Control System on MPPT

Figure 5-2: Development of MPPT control algorithm

A MPPT (Maximum Power Point Tracker) controller is needed to ensure the PV panel operates at its maximum output power capacity for its current system environment solar irradiance and the ambient temperature.

The purpose of the proposed MPPT algorithm is to:

1. Determine the initial maximum power point of a solar panel.
2. Track the maximum power point across a variance in ambient and PV cell temperatures; as well as a variance in solar irradiance from the sun.
3. Manage an optimum response to temporary shadow disturbances (Shadow flicker).
4. Locate and manage the effect of possible damaged and/or other sectional efficiency disturbances across the PV array.

In order to find a solution for the above outcomes 1 and 2, the following assumptions and results are adopted from a previous study [32]. These assumptions form a good summary of MPPT theory and highlights core concepts for the design of a MPPT control algorithm:

- The maximum power produced by a PV cell/panel is dependent on solar irradiance and the temperature of the cell/s only. This implies that the load of the system does not have an influence on the location of the maximum power point on the characteristic power curve of the solar cell/panel [32].
- The power produced by a PV panel can be varied for a specific system load through a change in the duty-cycle of the DC-DC converter of the MPPT system [32].
- For a constant set of solar irradiance and PV cell temperatures, the duty-cycle active for that specific instance, increases with an increase in the incident system load [32].
- For a constant incident system load, the duty-cycle active for that specific instance, increases with an increase in the solar irradiance [32].

It is clear that in order to be able to determine the maximum power point of a PV panel and track it accurately, the solar irradiance and PV cell temperatures need to be known. Even though the system load does not influence the position of the MPP (maximum power point), it does determine the duty-cycle needed to operate at the optimal power curve point. Careful monitoring of the PV panel output current and voltage, as well as the output voltage are needed by the system to drive the load, since both are dependent on the duty cycle.

Further requirements of the MPPT algorithm require the control system to have a fast response time to temporary shadows imposed on the PV panels for short periods of time. This criterion is directly related to the application of PV panels in motorised vehicles. The effect of this so called shadow flicker on a vehicle adapted to accommodate it and on one that is not is to be investigated to determine the further usefulness of using a control system with a fast response time.

The last requirement is to determine the usefulness of locating and managing the effects of damaged or otherwise performance impaired solar cells (partial shadow effects) in the overall PV panel system.

Chapter 5.1 details the proposal, formulation and design of a suitable MPPT control algorithm with the specific focus on moving PV panels (Implementation in solar vehicles).

A PV panel system is simulated and the proposed control MPPT algorithm implemented to determine quick and useful results to the research. The details of the simulation model, program code and results are detailed in Section 5.2.

5.1 Proposed MPPT Control Algorithm Design

For the purpose of creating a fast response control system in order to manage the dynamic solar vehicle system, it was decided to focus on a fuzzy logic control algorithm. A fuzzy logic control system is less complex than most control systems because it is not dependant on a mathematical model needing real-time calculation resources [12].

In [11] a fuzzy logic controller is designed based on the solar panel behaviour when functioning under optimum experimental conditions. This method greatly simplifies the real-time calculation resources, as well as the initial design of the algorithm.

Fuzzy logic controllers do however rely strongly on expert system knowledge for optimal performance. The parameters of the control system need to be guessed initially, based on this expert knowledge [11]. If needed, the parameters of the control system can be updated as the system is tested to increase its efficiency. In order to have a control system that can continuously improve itself (control parameters), a neural network is designed to update these control parameters of the fuzzy logic algorithm. These parameters do not require real-time updates and the neural network is implemented during tests to improve the control system for real-time use. The aim is to decrease the trial and error time needed to setup the fuzzy logic controller parameters.

Section 5.1.1 details the design of the fuzzy logic controller needed to locate the MPP and tracking it real-time. Section 5.1.2 formulates a neural network control system to efficiently update the fuzzy logic control parameters.

A further purpose of the MPPT control system is to actively compensate for possible damage to PV cells or other performance deficiencies such as shadow flicker. Section 5.1.3 formulates a proposed solution for the highly dynamic occurrences of shadow disturbances to the PV panel. The more permanent or longer lasting disturbances such as damaged or dirty PV cells are analysed in Section 5.1.4.

5.1.1 Fuzzy Logic Controller

Information is structured into knowledge through learning and experience; and logic is the analysis of this knowledge to determine appropriate response, known as the rule of inference [33]. Fuzzy logic is the analysis of linguistic terms (Fuzzy terms) as the knowledge base and logically analysing it by means of a rule base [33].

The fuzzy logic controller design methodology as adapted from [11] and [33] is listed below as a framework to formulate the control system:

1. Definition of controller inputs and outputs as fuzzy sets
2. Knowledge and rule base of the controller
3. Defuzzification from an inference diagram

1. Definition of controller inputs and outputs as fuzzy sets

A classical set of values will have a crisp boundary [34]. For example:

$$A = \{x | x < 20\} \qquad (5.1)$$

Where x belongs to the set A only if x is smaller than 20.

Linguistic inputs and outputs are known as fuzzy sets. In contrast with classical sets, the transition to belong to a set is gradual [34]. Below is a definition of a fuzzy set:

$$A = \{(x, \mu_A(x)) | x \in X\} \qquad [34] \qquad (5.2)$$

μ_A – Membership function for x in A

Each element of X is mapped to a continuous membership value by the membership function. This membership value is between 0 and 1 [34]. If all the membership values were just 0 or just 1, then the set is just a normal classical set in X. Figure 5-3 is an example of a classical set, versus a fuzzy set [35].

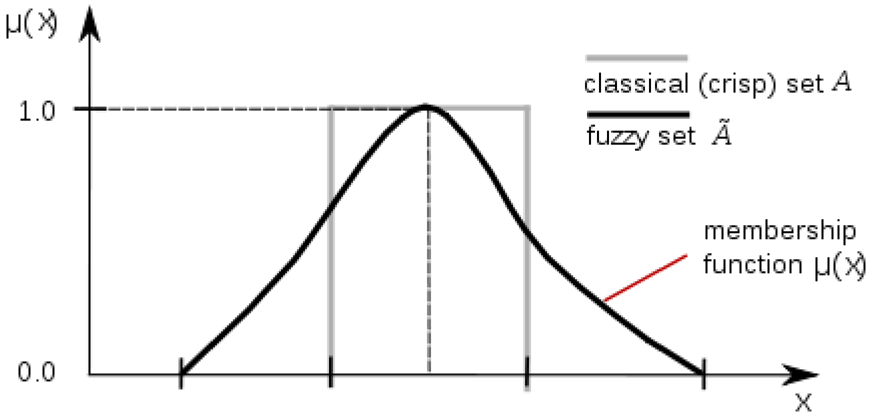


Figure 5-3: Fuzzy set versus classical set [35]

Two fuzzy inputs are defined:

- T_{pv} : PV cell temperature
- G_{solar} : Solar irradiation

The output of the control system is defined as:

- d_{MPPT} : Duty cycle to control the power circuit of the MPPT

The design frames for each fuzzy input/output (fuzzy sets) are defined next and each one consists of:

- Fuzzy sunsets (Linguistic values)
- Universe of discourse
- Membership function

As recommended in [33], an odd number of fuzzy subsets (linguistic values) are chosen for each fuzzy input/output to ensure that there is a fuzzy set in the middle. In order to simplify the processing resources needed for the controller, each fuzzy set is limited to five fuzzy sets as a first approach.

Fuzzy subsets for T_{pv} are defined as:

- Low
- Moderate
- High
- VHigh – Very High
- XHigh – Extremely High

$\rightarrow \mathcal{L}T_{pv}$

The universe of discourse T for $\mathcal{L}T_{pv}$ is chosen as:

T = [0°C, 100°C]

The chosen temperature range is large on purpose in order to try and stabilize the algorithm learning process.

Fuzzy subsets for G_{solar} are defined as:

- XLow – Extremely Low
- VLow – Very Low
- Low
- Moderate
- High

$\rightarrow \mathcal{L}G_{solar}$

The universe of discourse G for $\mathcal{L}G_{solar}$ is chosen as:

G = [0W/m^2, 1500W/m^2]

The irradiance range is chosen to large on purpose in order to try and stabilize the algorithm learning process.

Even though the MPPT measures rather the sunlight intensity in lux, an approximation is used to obtain the irradiance in W/m^2.

Fuzzy subsets for d_{MPPT} are defined as:

- VSmall – Very Small
- Small
- Medium
- Large
- VLarge – Very Large

$\rightarrow \mathcal{L}d_{MPPT}$

The universe of discourse D for $\mathcal{L}d_{MPPT}$ is chosen as:
D = [0.2, 0.8]

The design of the membership function for each fuzzy set consists of assigning a crisp input range to every fuzzy subset as well as choosing a suitable membership function profile.

Gaussian membership functions were chosen as a suitable profile which will yield a smooth variation of the output signal [12].

The formula for a Gaussian function can be defined as [34]:

$$gaussian(x;\sigma,c) = e^{[-((x-c)/\sqrt{2}\sigma)^2]} \qquad (5.3)$$

Where:
σ : Determines the width of the membership function
c : Represent the centre of the membership function

Figure 5-4 is an example of a Gaussian membership function shape.

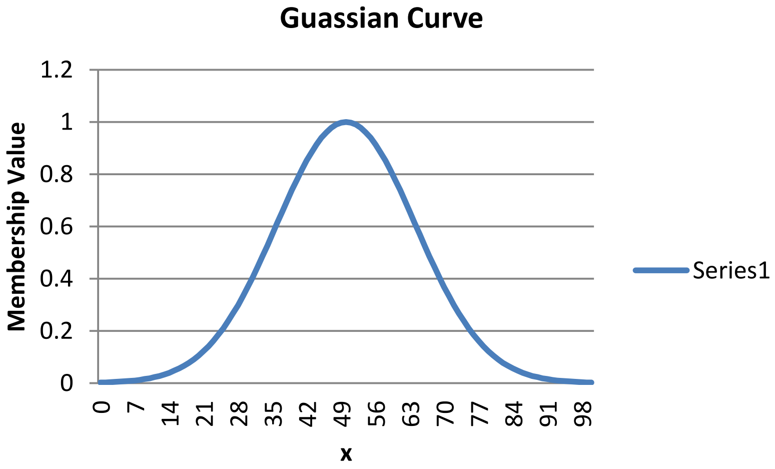


Figure 5-4: Example of a Gaussian membership function

Table 5-1, Table 5-2 and Table 5-3 contain the crisp input range, as assigned for each membership function (Fuzzy subset) for every fuzzy set. The input range was selected with aim of overlapping the Gaussian membership functions by at least 15%. The overlapping of the membership functions creates a gradual transgression from one fuzzy subset to the other. The input range is also chosen as a first approach and is later adjusted for fine tuning of the control system.

The universe of discourse for the cell temperature, T, ranges from 0°C to 100°C. If the temperature coefficient for polycrystalline PV cells is 0.12V/°C [36], then it is clear that the cell temperature has a huge effect on the PV panel performance. The coefficient means that for every 1°C increase in cell temperature from the standard test temperature of 25°C, the open circuit voltage, V_{OC}, of the solar panel will decrease by 0.12V. Because of this linear dependency of cell efficiency on cell temperature, the fuzzy subset ranges will be almost evenly divided for the first setup of the controller.

Fuzzy Subset	Input Range	c	σ
Low	0°C - 20°C	0°C	9
Moderate	0°C - 50°C	25°C	10
High	25°C - 65°C	45°C	9
VHigh	40°C - 100°C	70°C	12
XHigh	55°C - 100°C	100°C	20

Table 5-1: Crisp input range assignment for fuzzy set T_pv, PV cell temperature

Figure 5-5 is the resultant Gaussian membership function μ_T. Table 5-1 also indicates the c and σ values chosen as to allow for at least a 15% overlap of the fuzzy subset functions.

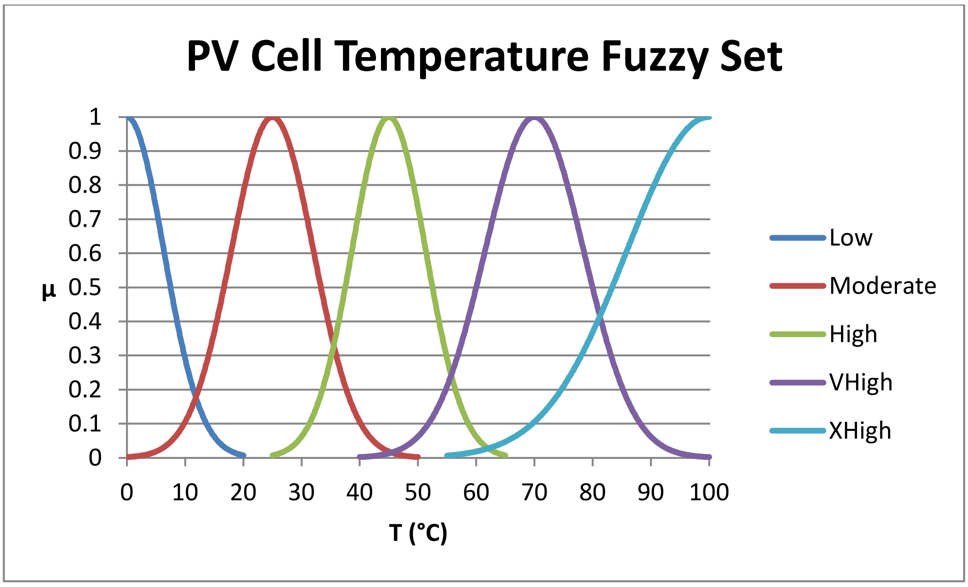


Figure 5-5: Gaussian membership functions for fuzzy set T_pv

Table 5-2 similarly contains the input range as well as the Gaussian formula parameters c and σ for the solar irradiation fuzzy set. The rated solar irradiance for a PV panel is 1000W/m^2. Even though the resultant overlap between the Moderate subset and High subset is only 8%, it is deemed as functional enough for a first approach.

Fuzzy Subset	Input Range	c	σ
XLow	0W/m^2 - 400W/m^2	0	180
VLow	150W/m^2 - 650W/m^2	400	110
Low	450W/m^2 - 950W/m^2	700	110
Moderate	700W/m^2 - 1300W/m^2	1000	135
High	1100W/m^2 - 1500W/m^2	1500	180

Table 5-2: Crisp input range assignment for fuzzy set G_solar, solar irradiation

Figure 5-6 is the resultant Gaussian membership function μ_G.

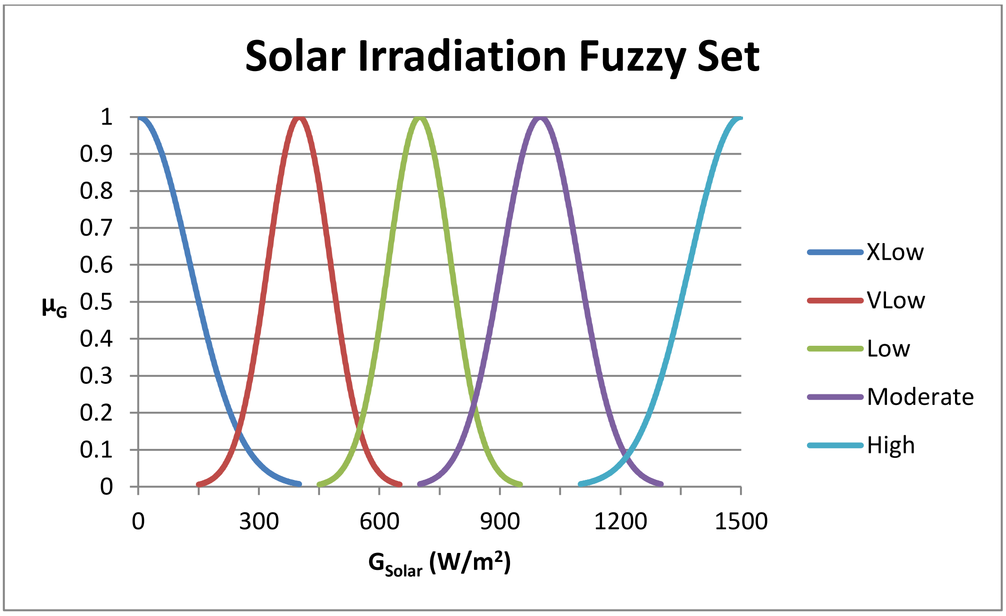


Figure 5-6: Gaussian membership functions for fuzzy set G_solar

Table 5-3 contains the input range as well as the Gaussian formula parameters c and σ for the duty cycle fuzzy set. The absolute minimum duty cycle was calculated in previous chapters. The absolute maximum duty cycle is when the PV panel's output voltage is pulled down to the same voltage as that of the battery, thus a duty cycle of 1. With these constraints in mind, the output range for the duty cycle fuzzy set was formulated.

Even though the resultant overlap between the VSmall subset and Small subset; and between the Small and Medium subset is only 15%, it is deemed as functional enough for a first approach.

Fuzzy Subset	Output Range	c	σ
VSmall	0.37 – 0.5	0.37	0.06
Small	0.4 – 0.66	0.53	0.06
Medium	0.56 – 0.82	0.69	0.06
Large	0.72 – 1.0	0.86	0.07
VLarge	0.87 – 1.0	1.0	0.04

Table 5-3: Crisp input range assignment for fuzzy set d_MPPT , duty cycle to control the power circuit of the MPPT

Figure 5-7 is the resultant Gaussian membership function μ_d.

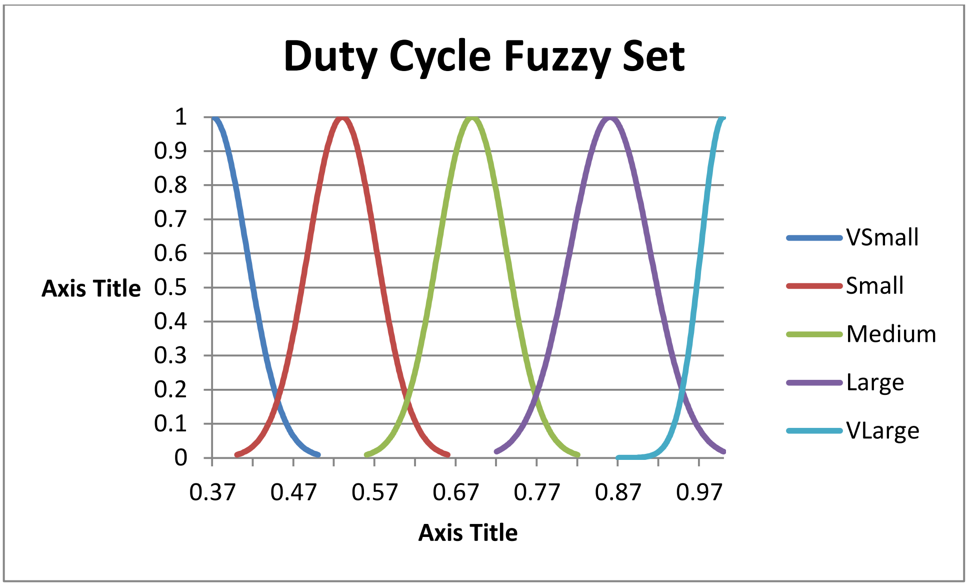


Figure 5-7: Gaussian membership functions for fuzzy set d_MPPT

2. Rule base for the controller

An initial rule base is created for the fuzzy logic controller. These If-Then rules are summarized as a matrix in Table 5-4. Later in this chapter this original approach is compared to the final rule base, after some parameter changes were made in order to fine tune the controller.

From Figure 2-5 it is clear that the higher the solar irradiation is, the more current must be drawn from the PV panel in order to stay on the MPP. Figure 2-7 illustrates that the current does not change much with temperature changes; however, a slight increase of the current is detected with an increase in cell temperature. Since the designed MPPT is controlled by varying the load on the PV panel, these assumptions are enough to formulate the fuzzy logic rule base.

T_{pv} → G_{solar} ↓	**Low**	**Moderate**	**High**	**VHigh**	**XHigh**
XLow	Medium	Large	Large	VLarge	VLarge
VLow	Small	Medium	Large	VLarge	VLarge
Low	Small	Medium	Large	VLarge	VLarge
Moderate	Small	Medium	Large	VLarge	VLarge
High	Small	Medium	Medium	VLarge	VLarge

Table 5-4: Fuzzy rule base for MPPT control system

3. Defuzzification from an inference diagram

The Mamdani fuzzy model as found in [34] is a popular way to infer a result from multiple rules with multiple antecedents. Figure 5-8 illustrates how this principle can be implemented, using the Min-Max (AND and OR operators) conform [34].

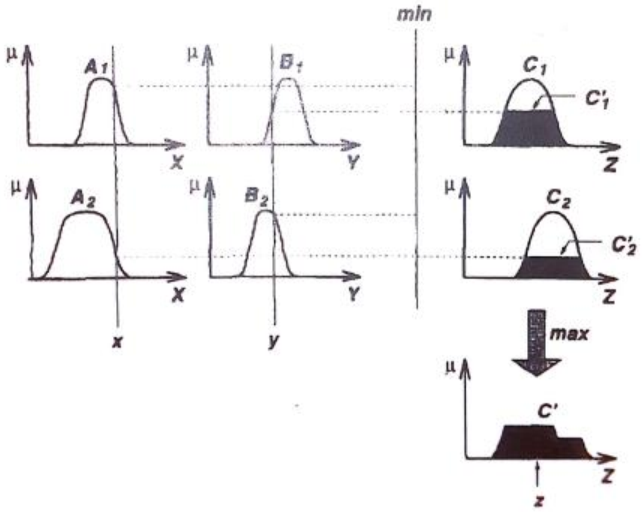


Figure 5-8: Illustration of the Mamdani method, using the Min-Max principle [34]

With reference to Figure 50:

- $C'_1 = A_1 \cap B_1$
- $C'_2 = A_2 \cap B_2$
- $C' = C'_1 \cup C'_2$

During trial and error tests, it was found that the Minimum (OR) principle to select a value between two input membership functions did not give accurate enough results. It was decided that this is due to the fact that the ambient temperature (one input) affects the MPP voltage much more than the solar irradiation (other input). Figure 2-5 and Figure 2-7 clearly illustrate this when compared.

As a first approach, it was decided that the output membership value to the output fuzzy subset should be obtained by combining the membership values of both fuzzy input subsets. It was decided to use an 80/20 % ration between the temperature and solar irradiation input membership function results. Thus C'_1 and C'_2 can be obtained as follows:

- $C'_1 = 0.8\mu_{A1} + 0.2\mu_{B1}$
- $C'_2 = 0.8\mu_{A2} + 0.2\mu_{B2}$

A popular method to obtain a crisp value from C'_1 and C'_2 is to take the centroid of area [34]. The centroid of area is defined for an arbitrary function as:

$$z_R = \frac{\int_z \mu_{C'}(z) z dz}{\int_z \mu_{C'}(z) dz} \text{ [37]} \qquad (5.4)$$

The actual functions of each min and max operation do not have to be calculated. Coordinates representing the functions shall be used in the calculation of the centroid of area. Given enough coordinates for a high resolution, this method is adequate for accurate results.

$$C'_n = \{(C'_{ni}, \mu_{C'n}(C'_{ni})) | C'_{ni} \in \min(T_{pv}, G_{solar})\} \qquad (5.5)$$

Where i is the ith coordinate set of the AND function. 500 Coordinates are chosen as an adequate resolution with the benefit of low processing resources required. C'_n is the set of coordinates comprising the AND function for every If-Then rules pairing. n Is the number of If-Then rules pairings that were adjusted to be valid.

Each coordinate set . C'_n is a function of d_{MPPT} and the centroid of area can be estimated as follows:

$$d_R = \frac{\sum_{500} \mu_{C'}(d) d}{\sum_{500} \mu_{C'}(d)} \qquad (5.6)$$

Where $\mu_{C'} = \max(C'_1, C'_n)$ (5.8)

Further fine tuning of the fuzzy logic controller is possible through scaling the input variables. This creates a shift of the horizontal axis and is also a method of prioritizing the input variables. The method for selecting appropriate scaling values may need future adjustments, even after a decent set is selected through trial and error due to:

- Different climate areas of operation
- Suspect results
- Optimization of the algorithm

A smart method for adapting the fuzzy logic controller, based on its scaling values, is by applying a neural network algorithm. Chapter 5.1.2 details the formulation of this method.

5.1.2 Neural Network Parameter Update Algorithm

A neural network (Adaptive network) can be defined as a network structure that can determine its input-output behaviour through a set of parameters that can be adjusted [34]. Figure 5-9 is an example of such a network.

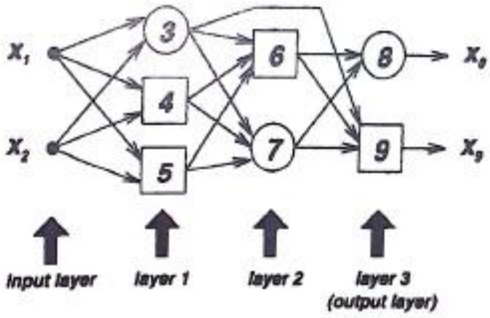


Figure 5-9: Example of a neural network [34]

x_1 and x_2 are the input parameters to the network and x_8 and x_9 are the output parameters.

A neural network consists of several neural layers. In the example above there is an input and output layer, as well as 2 hidden layers (layers 1 and 2). Figure 5-10 represents a single neuron; illustrating the functions that it can perform which includes:

- Adjustment of input signals via weight parameters. These parameters are also adjustable through learning algorithms.
- Determination of whether or not an input signal should be transferred to the output of the node and at what magnitude. This is accomplished through a transfer function.

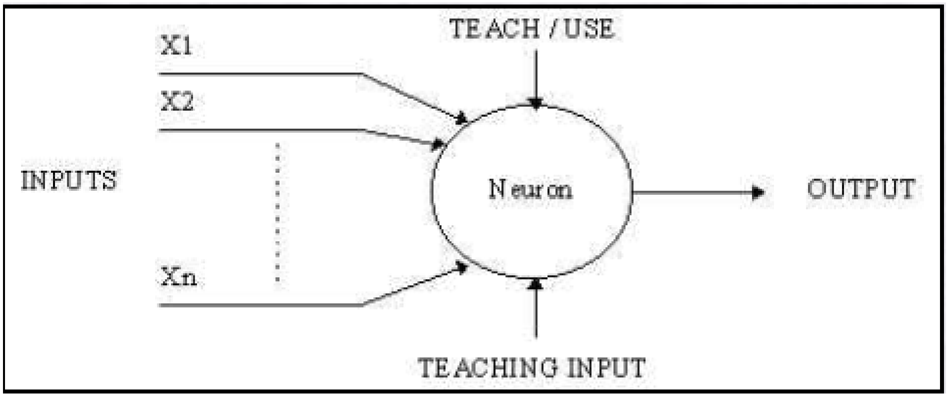


Figure 5-10: Representation of a single node and its functions in a network of nodes [38]

The concept to be tested is to have a fuzzy logic controller with the ability to adapt its fuzzy sets in order to produce a better algorithm through application of an offline learning algorithm. The idea is to have control system that can change if a part of it is deemed not to be efficient enough. Following this idea, the following parameters for a neural network is proposed:

- Inputs:
    - PV cell temperature T_{pv}
    - Solar irradiation G_{solar}
- Outputs:
    - Duty cycle of the MPPT controller d_{MPPT}

The network design chosen is a feed-forward type [34], on the basis that output signal of a node is necessary as an input to another node as this would increase the processing resources needed.

The fuzzy logic control system as developed has the advantage that it is a very simple algorithm with much flexibility by varying the parameters of the fuzzy subsets. These parameters need to be optimized through trial-and-error or expert knowledge. This does hamper the scope of the control system. An example is a PV cell that is damaged to such an extent as to impair the performance of solar panel.

If data of the real system input-output behaviour can be collected, then learning algorithms can be applied to a control system to update its parameters. A neural network is capable of teaching itself either offline or online, supervised or unsupervised [34]. For the purpose of this research, a neural network for the control system is proposed that can be updated through supervised offline learning. Data for the learning process is collected through manual searching for the PV panel's MPP and logging the temperature, solar irradiation and controller output for that specific instance.

The designed neural network has the primary function of updating the fuzzy inference system (fuzzy controller) in such a way that the fuzzy controller becomes a 'smart' controller, capable of updating itself in order to optimize its output. A combination of these two technologies is possible by creating a neural network with input fuzzy subsets (one per neuron) as the first neural level. An accepted acronym for this methodology is ANFIS: "Adaptive-network-based Fuzzy Inference System" [39].

Figure 5-11 is an illustration of the proposed ANFIS controller.

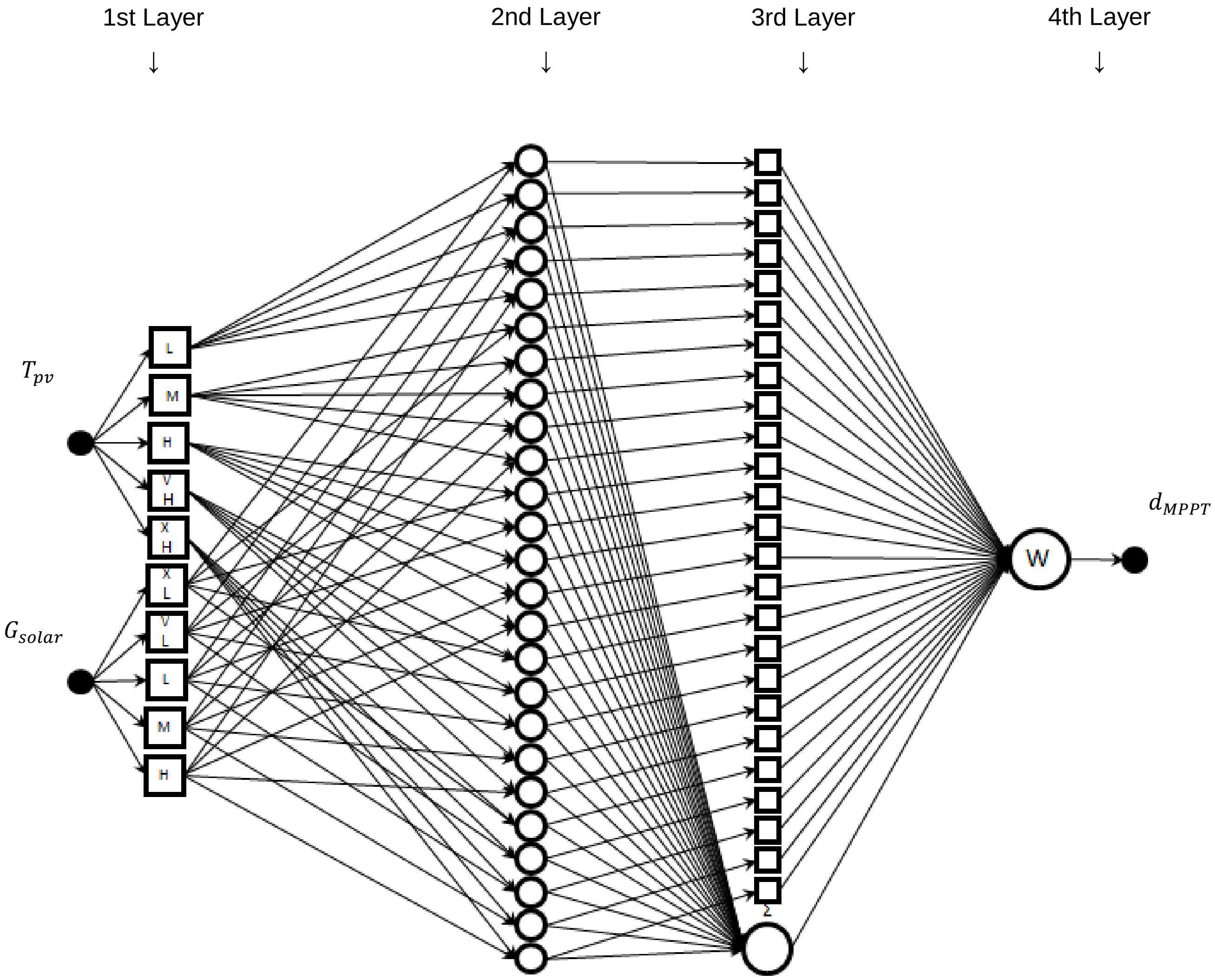


Figure 5-11: The proposed ANFIS controller diagram

The developed ANFIS consists of an input layer, output layer and 4 hidden layers. For ease of constructing the ANFIS model and to simplify the teaching algorithm for the ANFIS, the Mamdani method of fuzzy inference will be discarded and the Takagi and Sugeno's method implemented.

An example of the Takagi and Sugeno method for fuzzy inference is illustrated in Figure 5-12. The result generated by each fuzzy rule is used as weighted parameter that is used to determine the weighted average of linear functions. These are linear functions of the input variables. The parameters of these linear functions are updated during the learning process of the control system. These parameters of the linear functions will be chosen at random for each fuzzy result.

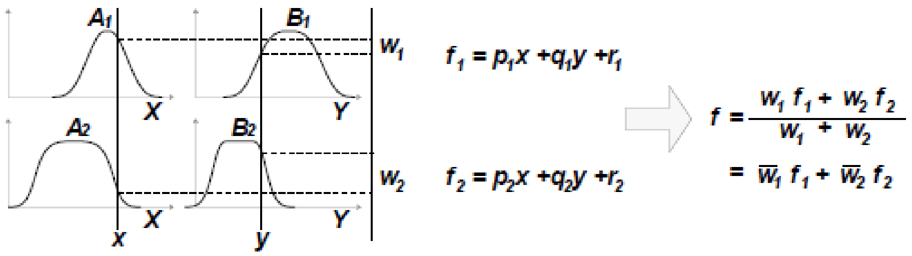


Figure 5-12: Example of the Takagi and Sugeno method for fuzzy inference [39]

The Following functionalities are assigned to each node layer:

- Input Layer: The crisp input values are fed into the controller and relayed to the appropriate fuzzy input subsets.
- 1st Layer: The first hidden layer consists of a fuzzy input set per node. No weights are assigned to the crisp input values and the output generated by each node is the membership function value.
- 2nd Layer: The second hidden layer determines the result of each input pair (only two input variables are used). As with the tested fuzzy control system, the output of this node layer is determined on the basis of the temperature have a larger effect on the MPP voltage than the solar irradiation. A simple function applying a fix ratio of 80/20% to the weights and combining the result is used, with the rule that both membership results must be greater than zero.
- 3rd Layer: The third hidden layer consists of one set of nodes that multiplies each 2nd layer result with a linear function $f_i = p_i T_{pv} + q_i G_{solar} + z_i$ and a single node that summates all of the results from the previous layer.
- 4th Layer. The fourth hidden layer calculates the weighted average of all the results and outputs a crisp value.
- Output Layer: The output layer holds the crisp result calculated by the control system.

The fuzzy sets used in developing the fuzzy logic control system, is used as initial membership functions for the ANFIS control system. The initial parameters for the third hidden layer linear functions are chosen arbitrarily. The resulting 25 functions are listed below:

1. $f_1 = 0.004725T_{pv} + 0.000105G_{solar} + 0.37$ (5.8)
2. $f_2 = 0.005040T_{pv} + 0.000084G_{solar} + 0.37$ (5.9)
3. $f_3 = 0.005355T_{pv} + 0.000063G_{solar} + 0.37$ (5.10)
4. $f_4 = 0.005670T_{pv} + 0.000042G_{solar} + 0.37$ (5.11)
5. $f_5 = 0.005985T_{pv} + 0.000021G_{solar} + 0.37$ (5.12)
6. $f_6 = 0.004725T_{pv} + 0.000105G_{solar} + 0.37$ (5.13)
7. $f_7 = 0.005040T_{pv} + 0.000084G_{solar} + 0.37$ (5.14)
8. $f_8 = 0.005355T_{pv} + 0.000063G_{solar} + 0.37$ (5.15)
9. $f_9 = 0.005670T_{pv} + 0.000042G_{solar} + 0.37$ (5.16)

10. $f_{10} = 0.005985T_{pv} + 0.000021G_{solar} + 0.37$ (5.17)

11. $f_{11} = 0.004725T_{pv} + 0.000105G_{solar} + 0.37$ (5.18)

12. $f_{12} = 0.005040T_{pv} + 0.000084G_{solar} + 0.37$ (5.19)

13. $f_{13} = 0.005355T_{pv} + 0.000063G_{solar} + 0.37$ (5.20)

14. $f_{14} = 0.005670T_{pv} + 0.000042G_{solar} + 0.37$ (5.21)

15. $f_{15} = 0.005985T_{pv} + 0.000021G_{solar} + 0.37$ (5.22)

16. $f_{16} = 0.004725T_{pv} + 0.000105G_{solar} + 0.37$ (5.23)

17. $f_{17} = 0.005040T_{pv} + 0.000084G_{solar} + 0.37$ (5.24)

18. $f_{18} = 0.005355T_{pv} + 0.000063G_{solar} + 0.37$ (5.25)

19. $f_{19} = 0.005670T_{pv} + 0.000042G_{solar} + 0.37$ (5.26)

20. $f_{20} = 0.005985T_{pv} + 0.000021G_{solar} + 0.37$ (5.27)

21. $f_{21} = 0.004725T_{pv} + 0.000105G_{solar} + 0.37$ (5.28)

22. $f_{22} = 0.005040T_{pv} + 0.000084G_{solar} + 0.37$ (5.29)

23. $f_{23} = 0.005355T_{pv} + 0.000063G_{solar} + 0.37$ (5.30)

24. $f_{24} = 0.005670T_{pv} + 0.000042G_{solar} + 0.37$ (5.31)

25. $f_{25} = 0.005985T_{pv} + 0.000021G_{solar} + 0.37$ (5.32)

Since the minimum duty cycle of the MPPT is 0.37, the initial value for r_i will be set at 0.37 for all the linear functions of the 3rd node layer. This should ensure that the output of the controller does not fall below 0.37.

The maximum duty cycle that the controller may produce is 1. Consequently the values for p_i and q_i were chosen such as to scale the respective input values to range from 0 to 1 as well as maintaining $p_iT_{pv} + q_iG_{solar} \leq 0.63$.

The transfer function needed for the second hidden layer can be expressed as follows:

$$w_i = 0.8\mu_T + 0.2\mu_G \quad (5.33)$$

Where:
0.8 and 0.2 are weights that can also be adjusted. They shall be denoted for every fuzzy output j as w_{jT} and w_{jG}.

Where μ_T and μ_G are the resulting membership function values for a specific temperature of solar irradiation fuzzy subset. The result is w_i, which is the weighted parameter applied to the fuzzy inference linear function for the ith fuzzy rule.

The bottom node in the third layer is used to calculate the sum of all the weights:

$$w_t = \sum_{i=1}^{25} w_i \qquad (5.34)$$

The summated weights for all the fuzzy rules are defined by the symbol w_t.
The function used in the single node fourth hidden layer is defined as:

$$d_{mppt} = \frac{\sum_{i=1}^{25} w_i f_i}{w_t} \qquad (5.35)$$

In order to update the parameter base of the neural network, it was decided to make use of the back-propagation learning rule [33] and [34].

A suitable weight update algorithm is found in [33]:

$$w_{ji}(k+1) = w_{ji}(k) - \eta \frac{\partial E(k)}{\partial w_{ji}(k)} \quad [33] \qquad (5.36)$$

Where:

$w_{ji}(k)$: The weighted parameter for the ith input to the node in the jth neural layer for a specific iteration k.

η: Learning rate of the rule with a value of less than 1. The smaller the value of this parameter, the quicker the calculation will converge to a value for w_{ji}.

$E(k)$: The application dependable cost function, in this instance chosen as the least-squares output error function for a specific iteration k.

$$E(k) = \frac{1}{2}\sum_{j=1}^{q}(y_j - ¥_j(k))^2 \quad \text{for a single node output } j \quad [33] \qquad (5.37)$$

Where:

y_j: Desired output value of the node.
$¥_j(k)$: Actual output value of the node for iteration instance k.
$y_j - ¥_j(k)$ is denoted is e_j and is the output error.
Through the application of the product rule and the chain rule:

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = -e_j(k) f'_{ji}\left[\sum_{i=1}^{n} w_{ji}(k)\, a_i + w_0(k)\right] a_i \quad [33] \qquad (5.38)$$

$$\frac{\partial E(k)}{\partial w_0(k)} = -e_j(k) f'_{ji}\left[\sum_{i=1}^{n} w_{ji}(k)\, a_i + w_0(k)\right] \quad [33] \qquad (5.39)$$

Where:
$w_0(k)$: Bias term of the result

a_i: Input variable

$f_{ji}(a_i)$ is chosen as a linear function (ex. the output node) such that $f_{ji}(a_i) = a_i$ and this result in $f'_{ji}(a_i) = 1$ and:

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = -e_j(k)a_i \qquad (5.40)$$

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = -e_j(k) \qquad (5.41)$$

Substituting equation 5.20 and 5.21 into 5.16 yields the following result:

$$w_{ji}(k+1) = w_{ji}(k) + \eta e_j(k)a_i \quad [33] \qquad (5.42)$$

$$w_0(k+1) = w_{ji}(k) + \eta e_j(k) \quad [33] \qquad (5.43)$$

Equation 8.24 shall be used to update the weighted parameters for the neural network for a specific known group of desired input-output pairs.

To calculate the new weights for nodes other than the output node, the error value has to be propagated back to be adjusted by the previous node's weights and become the next node's error value; hence the name of back propagation. Consequently the next output error needs to be calculated by multiplying it with the previous node's weights as well as the new nodes input and the previous nodes derivative.

A simple method of updating (teaching) the neural network weight parameters are to scan through a specific power curve and the finding the MPP on that curve manually. Once the MPP is located, it can be used as the benchmark for the learning and upgrading of neural network algorithm for a specific set of temperature and solar irradiation values.

5.1.3 Dynamic Response to Shadow Flicker

The PV panels of a moving vehicle will be subjected to a variety of shadows or shadow flicker. Shadow flicker may vary in its duration and size of shadowed area. If the duration of the shadow over the PV panels is very short, then it is better for the MPPT not to track this new maximum power point (MPP), because of the delay in switching back to normal once the disturbance is gone. During the shadowed period, the irradiance is much lower, compared to that of direct sunlight and it makes sense not to jeopardize the harvesting during full sunlight.

By its nature, the fuzzy logic controller is quick in decision making and it does not need to search for the MPP. This feature makes it convenient to change the MPP as long as the flicker is not an extremely short disturbance. If this very short instance is too quick for the controller, then it is assumed to be an insignificant disturbance and the shadow is ignored.

When a shadow covers only part of the PV panel array then it is very ambiguous for the control system to make its decision based on the input from only one light sensor which might be situated in the shadow or not. This causes the light sensor input to be very dependent on its location on the PV array. If the system contains more than one sensor, it is possible to determine the presence of a large shadow over one of the two test PV panels used. Knowing about the presence of a shadow that only partially covers the PV panel array can be used to improve the performance of the controller as follows:

- The controller will be assigned to track the MPP as if the shadow was not present on the one panel by using the input of the sensor which is not covered by the shadow.

5.1.4 Dynamic Response to PV Cell Damage and Efficiency Degradations

The neural network, as defined in Section 5.1.2, is used to fine tune the fuzzy logic controller. PV cell damage causes efficiency degradations to the panel as well as changing its MPP. This requires changes to be made to the parameters of the fuzzy logic controller (tuning).

In order for the neural network to know when tuning is required, symptoms of permanent cell damage needs to be identified. For the neural network to work outside a lab environment, the MPPT will scan manually through the power curve of the PV panel to locate the MPP. Once the new MPP is located (due to the degraded/deficient cells), the weight parameters of the neural network algorithm is updated for the desired control result. It stands to reason, that if one of the two test PV panels are not damaged, then the normal MPP of that specific panel is the one used for the control system and no update of the neural network is needed.

From the experiments done to test the MPPT control system, clarity can be obtained as to how often such a system needs to check whether the panels are damaged/degraded or not. However, it should be noted that such a test is rather time consuming and interrupts the duties of the MPPT as well as not producing good power output from the system.

In order to check individual panels for damage, future research may include extra sensors to monitor each panel.

5.2 Simulation of Proposed MPPT Control Algorithm

Experimental tests are ideal for real-time/real-life results to verify a given system. These experiments do however require considerable resources such as time, money and labour. A good simulation can yield reasonable parameters for the fuzzy logic and neural network controllers, hence decreasing trial-and-error time during experimentation to get the final control parameters. A further benefit of a simulation is that tests can be conducted any time of the day.

The formulation and realization of this simulation is divided into the following subsections:

- Mathematical model of PV panel array
- MPPT duty cycle
- Simulation software used
- Program for PV panel simulation
- Program for fuzzy logic controller
- Program for neural network controller (fine tune) and ANFIS
- Integration of MPPT controller, shadow detection and deficient panel detection
- Simulation results

Mathematical model of PV panel array

The objective of this mathematical model is to obtain the parameters for an I-V curve of a specific PV panel and PV array. This I-V curve can be used to calculate the MPP of the panel/array for a specific set of solar irradiation and cell temperature conditions.

PV cells are a form of semiconductor diodes [40]. The P-N junctions of these semiconductor diodes are exposed to light in order for them to generate a current [40]. Figure 5-13 represents a PV cell as a semiconductor diode and current source [40]. The diagram shows the ideal PV cell, however it also shows the equivalent parallel and series resistances of the cell. These resistance values are significant as they aggregate when cells are connected into an array to form a PV panel.

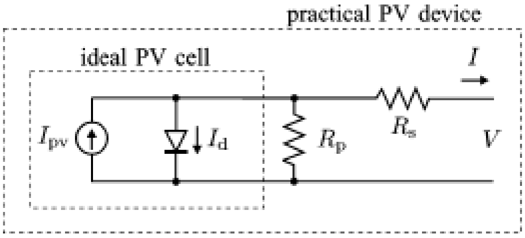


Figure 5-13: The representation of a PV cell as a semiconductor diode and current source

The equation for an ideal PV cell is:

$$I_{out} = I_{PVcell} - I_{0cell}[\exp\left(\frac{qV_{out}}{akT}\right) - 1] \qquad [40] \qquad (5.44)$$

Where:

I_{out} : Output current of the PV cell [A]

I_{PVcell} : Current produced within the PV cell, directly proportional to solar irradiation [A]

I_{0cell} : Diode reverse saturation current (leakage current) of the cell [A]

q : Electron charge constant of $1.60217646 \times 10^{-19}C$

k : Boltzmann constant of $1.3806503 \times 10^{-23}J/K$

T : P-N junction temperature [K]

a : Diode ideality constant, which is totally empirical. The value for a is chosen arbitrary and influences fitment of the model curve. The value for a is normally $1 \leq a \leq 1.5$ and can be adjusted later if further improvements is deemed necessary according to experimental results [40].

V_{out} : Output voltage of the PV cell [V]

$I_{0cell}[\exp\left(\frac{qV}{akT}\right) - 1]$ is known as the Shockley diode equation [40].

When PV cells are connected in an array to form a panel, the following equation more accurately models the I-V curve relationship [40]:

$$I_{out} = I_{PV} - I_0\left[\exp\left(\frac{V_{out}+R_S I_{out}}{V_t a}\right) - 1\right] - \frac{V_{out}+R_S I_{out}}{R_p} \quad [40] \qquad (5.45)$$

Where:

I_{out} : Output current of the PV panel [A]

I_{PV} : Current produced within the PV panel, directly proportional to solar irradiation [A]

I_0 : Diode reverse saturation current (leakage current) of the panel [A]

V_{out} : Output voltage of the PV panel [V]

R_s : Equivalent series resistance of the PV panel [Ω]

R_p : Equivalent parallel resistance of the PV panel [Ω]

V_t : Array thermal voltage defined as $V_t = \frac{N_s kT}{q}$ where N_s represents the number of PV cells connected in series [V] (5.46)

$$I_{PV} = I_{PVcell} N_p \quad [40] \qquad (5.47)$$

$$I_0 = I_{0cell} N_p \quad [40] \qquad (5.48)$$

Where N_p represents the number of cells (solar strings) connected in parallel to form the array.

The current that is produced within the PV cell, due to the incident solar irradiation on it, is a linear relationship [35]. The following equation is used to determine I_{PV} for a specific temperature and solar irradiation set:

$$I_{PV} = (I_{PV,nominal} + K_I \Delta T)\frac{G}{G_{nominal}} \quad [40] \qquad (5.49)$$

Where:

$I_{PV,nominal}$: Nominal current produced within the PV cell at 25°C and 1000W/m^2 [A]

ΔT : Temperature differential between the actual and the nominal (273K) temperatures $(T - T_{nominal})$ [K]

K_I : Short-circuit current over temperature coefficient as can be found in the specification sheet of a PV panel [A/K]

G : Actual solar irradiation [W/m^2]

$G_{nominal}$: Nominal solar irradiation (1000W/m^2) [W/m^2]

The next parameter for equation 8.27 to be calculated is the diode saturation current I_0.

$$I_0 = \frac{I_{SC,nominal}+K_I\Delta T}{\exp((V_{OC,nominal}+K_V\Delta T)/aV_t)-1} \quad [40] \qquad (5.50)$$

Where:

$I_{SC,nominal}$: Nominal short-circuit current of the PV panel [A]

$V_{OC,nominal}$: Nominal open-circuit voltage of the PV panel [V]

K_V : Open-circuit voltage over temperature coefficient as can be found in the specification sheet of a PV panel [V/K]

As suggested in [35], the values for R_s and R_p are best found when it is kept in mind that the given experimental nominal maximum power of the panel should match that of the mathematical model. The two resistance values can then be iteratively calculated until the two power output values matches exactly. Therefore the current calculated by means of equation 8.27 for the maximum power point multiplied by the MPP voltage will produce the MPP power.

The value for $I_{PV,nominal}$ is calculated as follows:

$$I_{PV,nominal} = \frac{R_s+R_p}{R_p} I_{SC,nominal} \quad [40] \qquad (5.51)$$

After some iterative calculations, the following resistance values were obtained for the selected PV panels:

- R_s = 420Ω
- R_p = 0.2Ω

From the PV panel specification sheet, the following parameters are obtained:

- K_I = 0.05%/K of $I_{SC,nominal}$
- K_V =-0.075 V/K
- $I_{SC,nominal}$ = 1.3A
- $V_{OC,nominal}$ = 21.6V
- $I_{MPP,nominal}$ = 1.14A
- $V_{MPP,nominal}$ = 17.4V

With the above parameters known, the curves in Figure 5-14 and Figure 5-15 are calculated using the equations for an example ambient (cell) temperature of 25°C and 1000W/m^2 solar irradiation. Figure 5-14 is the resultant I-V curve and Figure 5-15 is the corresponding power curve.

The red star on the curve indicates the maximum power point of the PV panel at 25°C and 1000W/m^2. This is equal to 17.4V, which is the same as that of the actual panel.

The power produced by the simulation PV panel is 19.9W, compared to the 20W of the real panel. Obviously one good result does not qualify the simulation as an accurate representation of the real panel and behaviour does differ slightly from the real panel. However, the curves are very similar and the purpose of the simulation is just to receive meaningful test data for the control simulation and not to setup the controller for the real system. Experimental data shall be used to train the control system simulation.

Again note the red star on the power curve indicating the MPP. It is clear from its position at the top of the curve that this is the MPP of the curve.

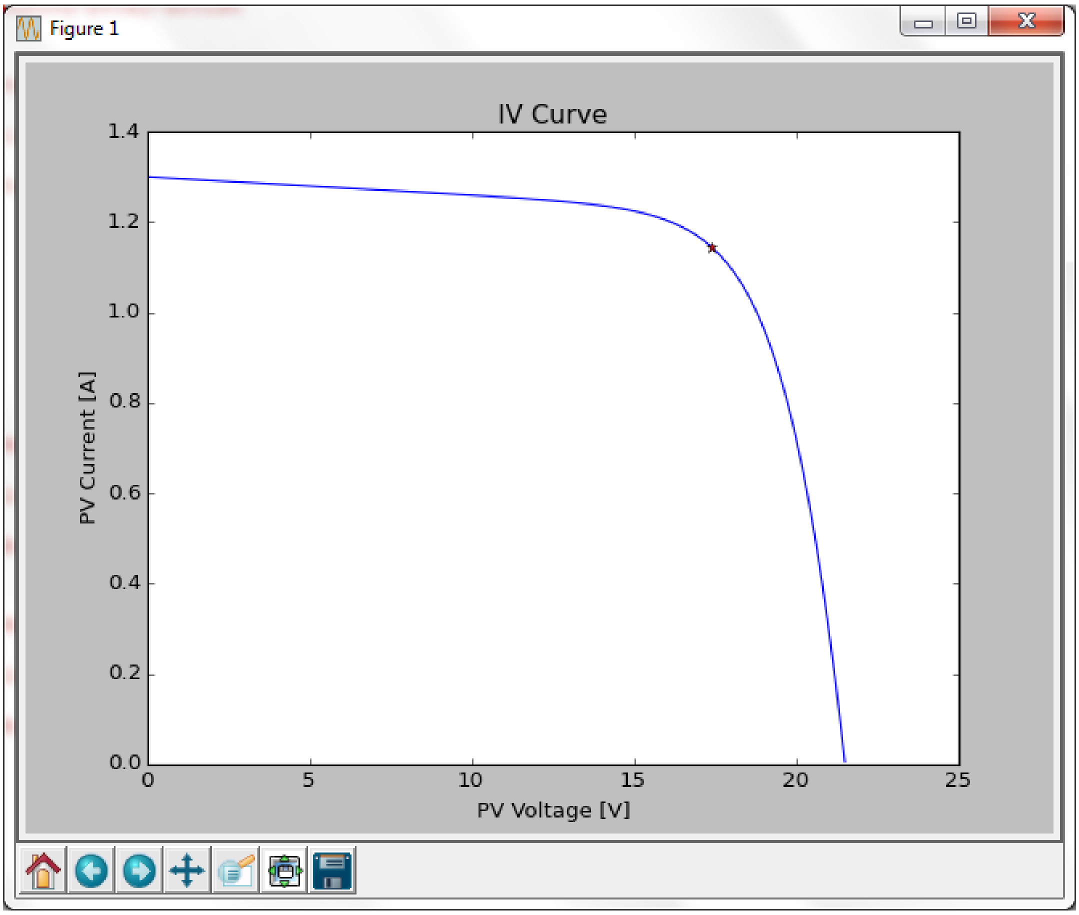


Figure 5-14: The I-V curve for the test PV panel as produced by the mathematical model at 25°C and 1000W/m2

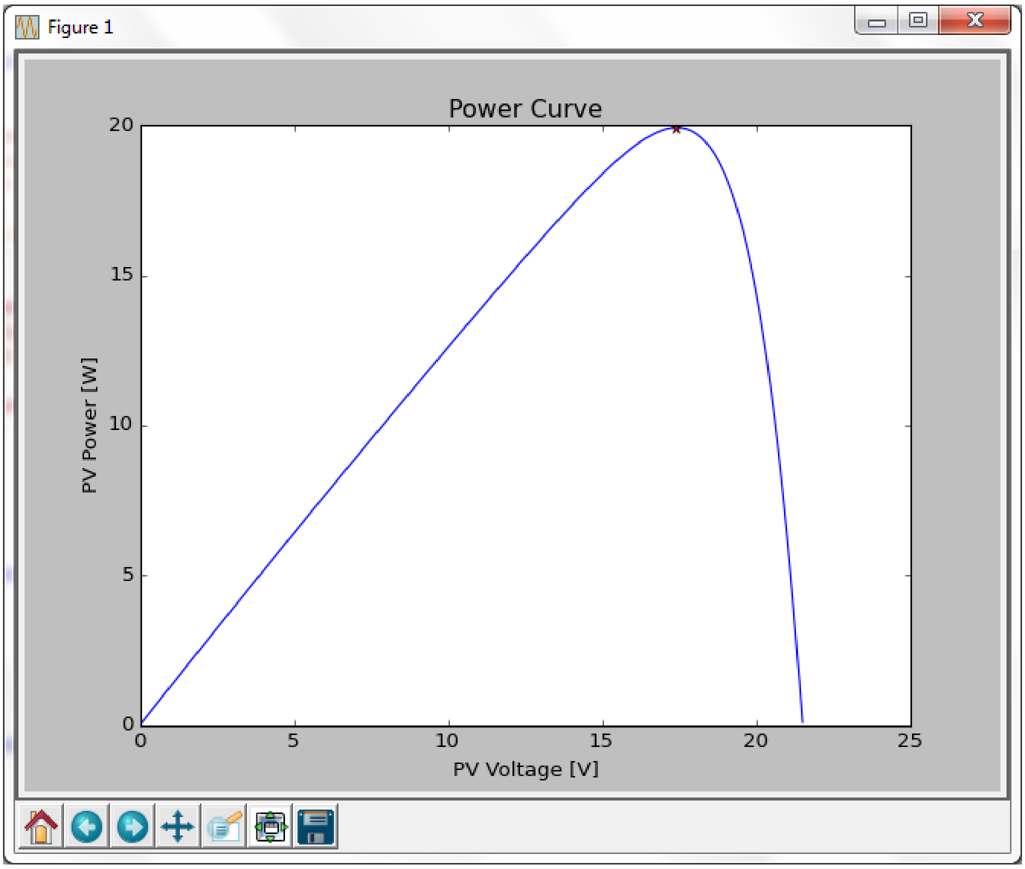


Figure 5-15: The power curve for the test PV panel as produced by the mathematical model at 25°C and 1000W/m2

The curves fit the theoretical I-V and power curves for PV panels and is deemed as a good estimation of actual experimental curves and data. The mathematical model of the PV panels is thus considered good enough to use to determine whether or not the proposed control systems can be applied to an actual system.

MPPT duty cycle

The duty cycle of the MPPT PWM transistor driver is defined as the operating output voltage of the PV panel, divided by the MPPT output voltage [41].

The output voltage of the MPPT is operated at the battery voltage. This voltage is monitored on the real MPPT as it increases as the battery is charged especially at the first stage of charging. Figure 5-16 below illustrates the charge cycle of a lead-acid battery cell [42].

For the purpose of the simulation this increase in voltage shall be ignored as it only influences the duty cycle for finding the MPP and not the MPP itself. An easy way to adapt to a change in battery voltage is to scale the duty cycle fuzzy set accordingly in the real MPPT. The simulation's aim is directed more at testing the MPPT control system over short periods of time.

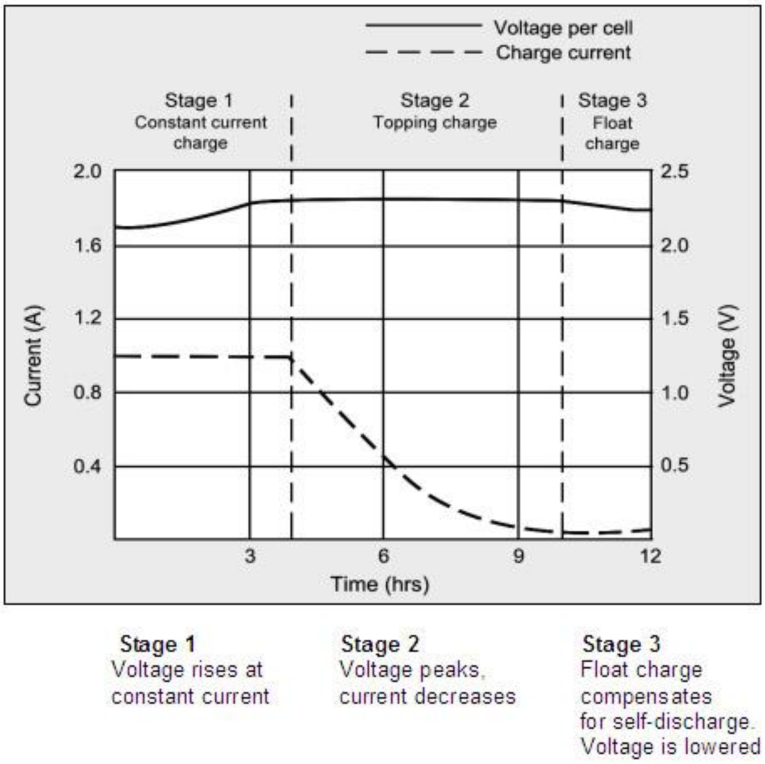


Figure 5-16: Charge curve of a lead-acid battery cell [42]

A constant battery voltage of 12.5V is used in the simulation. Using the duty cycle computed by the MPPT algorithm in tandem with the mathematical PV panel model, the accuracy of the control system can be estimated to a relatively accurate degree.

Simulation software used

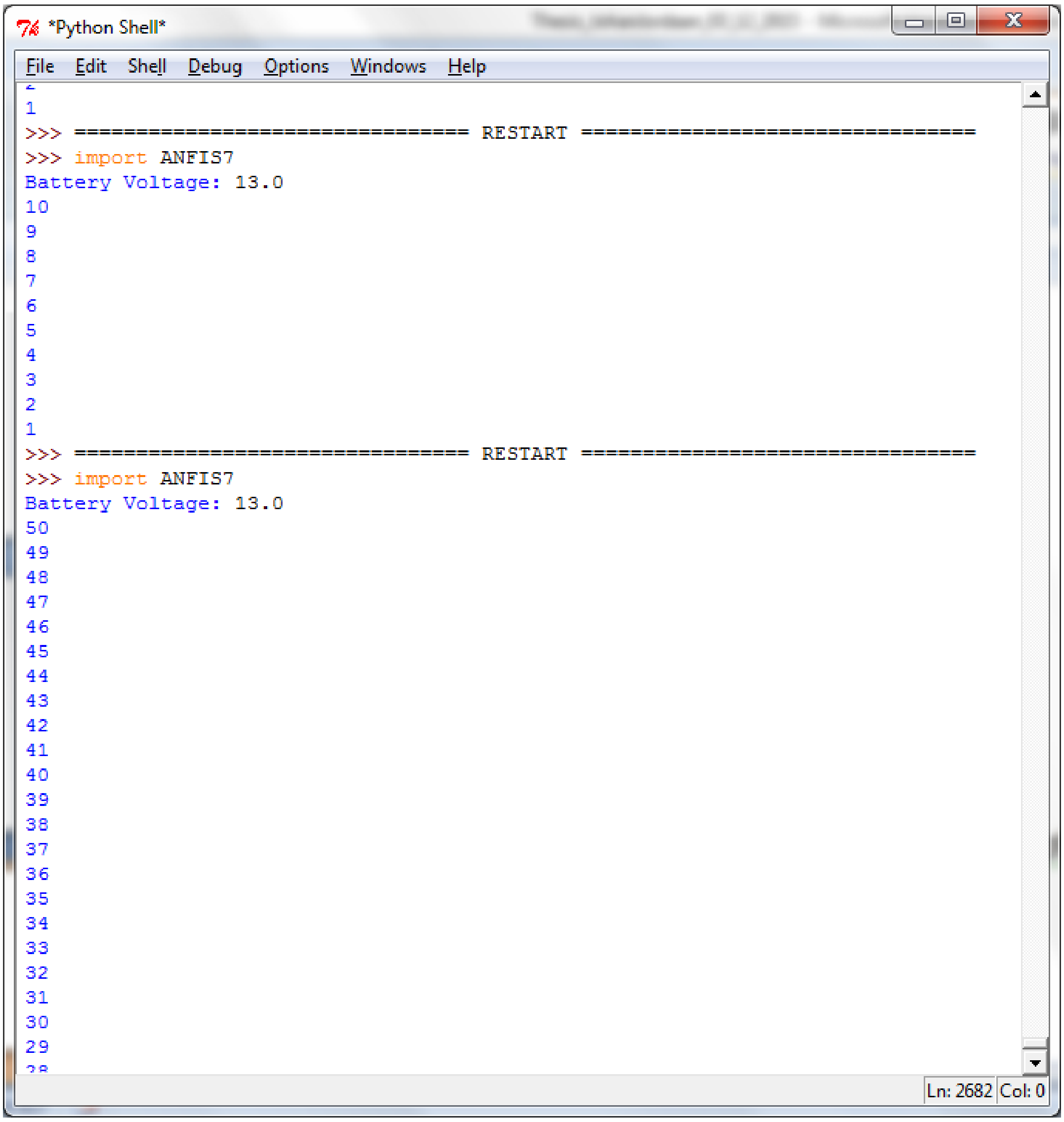
The Python language was chosen arbitrarily for the simulation on the basis of finding many helpful interface and mathematical packages to calculate meaningful data. The software packages include:

- Python® Idle 2.7 as the IDE
- Numpy® base N-dimentional array package
- Scipy® for scientific calculations
- Matplot® for creating accurate 2D graphs from results

All of these software packages are available on the internet for free.

Python is often compared to languages such as C#, Java and Visual Basic, which makes the software easy to adapt to and use properly [43].

The software is developed in neat functions and separate modules with comprehensive comments all round to increase future use of the software. Figure 5-17 is an example of the Python Idle 2.7® interface.



```
2
1
>>> ================================ RESTART ================================
>>> import ANFIS7
Battery Voltage: 13.0
10
9
8
7
6
5
4
3
2
1
>>> ================================ RESTART ================================
>>> import ANFIS7
Battery Voltage: 13.0
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
```

Figure 5-17: An example of the program design interface using the Python Idle 2.7 IDE

The interface is very basic and text base, except for the ability to plot graphs.

Program for PV panel simulation

The module for the PV panel simulation is named *PV_Sim.py* and it has the following functionalities:

- Producing the output power versus voltage curve of a PV panel for a specific pair or collection of ambient temperatures and solar irradiation to illustrate the effect of each.
- Producing the output current versus voltage curve of a PV panel for a specific pair or collection of ambient temperatures and solar irradiation to illustrate the effect of each.
- Calculation of the maximum power point for a specific power curve.

It is prompt based and the user needs to supply the panel temperature as well as the incident solar irradiation.

The following functions were developed:

- Nominal_PVcurrent ()
- Leakage_current ()
- PVcurrent ()
- Output_current (IV,V0)
- Output_current2 (V1)
- LoopPlot ()

The Nominal_PVcurrent () function calculates the internal current that is generated withing the PV panel at 25°C and 1000W/m^2.

Leakage_current () and PVcurrent () calculates the reverse saturation current and internal current of the PV panel, dependent on the cell temperatures as well as the incident solar irradiation.

Output_current (IV,V0) and Output_current2 (V1) establishes the relationship between the output current and voltage of the panel and solves the equation.

LoopPlot () generates the plots required of the data calculated.

The main function within this module is Output_current (IV,V0) and Output_current2 (V1) and it is inserted below:

```
def Output_current(IV,V0):
        Iout,Vout = IV
         return (PVcurrent() - Leakage_current()*(math.exp((Vout + Rs*Iout)/(Vt*a)) - 1) - (Vout +
                 Rs*Iout)/Rp - Iout,Vout - V0)

def Output_current2(V1):
         Iout,Vout = fsolve(Output_current,(0,15),V1)
        return (Iout)
```

This PV simulation is used separately from the controller simulation to generate I/O data in order to set up the controller parameters in offline learning. This is to prove the functionality of the controller and its learning base, but real I/O data is still used to verify the controller for experimental tests to be done.

Program for fuzzy logic controller

This program is used as a first approach to determine the effectiveness of a fuzzy controller implemented on a MPPT. It is prompt based and the user needs to supply the panel temperature as well as the incident solar irradiation.

The functions for the temperature fuzzy subsets are:

- Tpv_Low (T)
- Tpv_Moderate (T)
- Tpv_High (T)
- Tpv_VHigh (T)
- Tpv_XHigh (T)

These functions all receive the same temperature input and apply it to each respective membership function and output the membership function result.

The functions for the solar irradiation fuzzy subsets are:

- Gsolar_XLow (G)
- Gsolar_VLow (G)
- Gsolar_Low (G)
- Gsolar_Moderate (G)
- Gsolar_High (G)

These functions all receive the same irradiation input and apply it to each respective membership function and output the membership function result.

The function below is an example of a fuzzy subset:

```
def Tpv_Low(T): #Fuzzy subset LOW (1.1)
  if  T <= 20:
     uTLow = math.exp(-1*math.pow(((T-0)/9),2))
  else:
     uTLow = 0.0
  return uTLow
```

Fire_Temperature (T) and Fire_SolarG (G) collects all the results from the rules that fire (not equal to zero).

def RuleBase(T,G) contains the controller rule base and calculates the results for each input/output pair. Since there are five temperature fuzzy subsets and five solar irradiation subsets, 25 rules are required in the rule base.

FuzzyOutput_Max (T,G) infers the calculated results and outputs a crisp duty cycle value. This function is displayed in full:

```
def FuzzyOutput_Max(T,G):
      FireResults = RuleBase(T,G)
        i = 0
        VS = 0.0
        S = 0.0
        M = 0.0
        L = 0.0
        VL = 0.0
        while (i<25):
          fuz,fval = FireResults[i]
          if ((fuz == 'VS') & (fval > VS)):
            VS = fval
          elif ((fuz == 'S') & (fval > S)):
            S = fval
          elif ((fuz == 'M') & (fval > M)):
            M = fval
          elif ((fuz == 'L') & (fval > L)):
            L = fval
          elif ((fuz == 'VL') & (fval > VL)):
            VL = fval
          i = i + 1
        FuzOut = [VS,S,M,L,VL]
        print FuzOut
        print FireResults

        #Create list of coordinates for every output fuzzy subset
        dlist = numpy.arange(0.37,1,0.01)
        VSlist = numpy.arange(0.37,1,0.01)
        Slist = numpy.arange(0.37,1,0.01)
        Mlist = numpy.arange(0.37,1,0.01)
        Llist = numpy.arange(0.37,1,0.01)
        VLlist = numpy.arange(0.37,1,0.01)
        Flist = numpy.arange(0.37,1,0.01)
```

```
dArg1 = numpy.arange(0.37,1,0.01)
j = 0

while (j<63):
    if ((math.exp(-1*math.pow(((dlist[j]-0.37)/0.06),2))<VS) & (0.37<=dlist[j]<=0.5)):
        VSlist[j] = math.exp(-1*math.pow(((dlist[j]-0.37)/0.06),2))
    elif ((math.exp(-1*math.pow(((dlist[j]-0.37)/0.06),2))>=VS) & (0.37<=dlist[j]<=0.5)):
        VSlist[j] = VS
    else:
        VSlist[j] = 0.0

    if ((math.exp(-1*math.pow(((dlist[j]-0.53)/0.06),2))<S) & (0.4<=dlist[j]<=0.66)):
        Slist[j] = math.exp(-1*math.pow(((dlist[j]-0.53)/0.06),2))
    elif ((math.exp(-1*math.pow(((dlist[j]-0.53)/0.06),2))>=S) & (0.4<=dlist[j]<=0.66)):
        Slist[j] = S
    else:
        Slist[j] = 0.0
    if ((math.exp(-1*math.pow(((dlist[j]-0.69)/0.06),2))<M) & (0.56<=dlist[j]<=0.82)):
        Mlist[j] = math.exp(-1*math.pow(((dlist[j]-0.69)/0.06),2))
    elif ((math.exp(-1*math.pow(((dlist[j]-0.69)/0.06),2))>=M) & (0.56<=dlist[j]<=0.82)):
        Mlist[j] = M
    else:
        Mlist[j] = 0.0
    if ((math.exp(-1*math.pow(((dlist[j]-0.86)/0.07),2))<L) & (0.72<=dlist[j]<=1)):
        Llist[j] = math.exp(-1*math.pow(((dlist[j]-0.86)/0.07),2))
    elif ((math.exp(-1*math.pow(((dlist[j]-0.86)/0.07),2))>=L) & (0.72<=dlist[j]<=1)):
        Llist[j] = L
    else:
        Llist[j] = 0.0
    if ((math.exp(-1*math.pow(((dlist[j]-1)/0.04),2))<VL) & (0.87<=dlist[j]<=1)):
        VLlist[j] = math.exp(-1*math.pow(((dlist[j]-1)/0.04),2))
    elif ((math.exp(-1*math.pow(((dlist[j]-1)/0.04),2))>=VL) & (0.87<=dlist[j]<=1)):
        VLlist[j] = VL
    else:
        VLlist[j] = 0.0
    j = j + 1

#Create one list representing the output fuzzy result
k = 0
while (k<63):
    Flist[k] = max(VSlist[k],Slist[k],Mlist[k],Llist[k],VLlist[k])
    dArg1[k] = Flist[k]*dlist[k]
    k = k + 1
dArg2 = sum(dArg1)
dArg3 = sum(Flist)
dr = dArg2/dArg3
print "Duty Cycle: ", dr
print "Target MPP: ", 12/dr
return dr
```

Program for ANFIS controller

The program is also prompt based and the user needs to supply the panel temperature as well as the incident solar irradiation.

Figure 5-18 illustrates a basic functional diagram for the simulation of the ANFIS controller. The simulation consists of an ANFIS network, dependant on several membership functions and fuzzy rules.

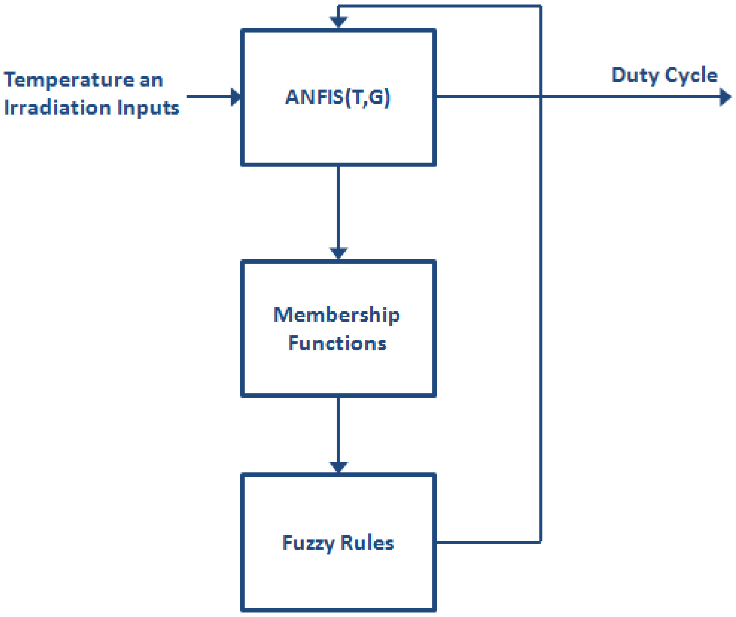


Figure 5-18: Functional diagram of simulated ANFIS controller

The program is more or less divided into groups representing the different node layers. The first group of functions are individual functions, one for each node in the first hidden layer. These functions are the fuzzy subsets of the controller. Listed below are these functions:

- Tpv_Low (T)
- Tpv_Moderate (T)
- Tpv_High (T)
- Tpv_VHigh (T)
- Tpv_XHigh (T)
- Gsolar_XLow (G)
- Gsolar_VLow (G)
- Gsolar_Low (G)
- Gsolar_Moderate (G)
- Gsolar_High (G)

An example of such a function is:

```
def Tpv_Low(T):            #Fuzzy subset LOW (1.1)
   uTLow = math.exp(-1*math.pow(((T-0)/9),2))
   return uTLow
```

These simple functions use the user input values for temperature and irradiation to output their respective membership values. These values are then combined for each if-then rule (each determined according to the node connections) output to produce weights for the next node layer. Consequently the functions for the next group, second hidden layer, are defined as:

- rule1 ()
- rule2 ()
- rule3 ()
- …..
- ……
- rule25 ()

The last function ANFIS (T,G), calculates the weighted sum of the initial weights multiplied by the linear inference functions. The resultant duty cycle is returned by the function. These are the representations of the third and fourth hidden layers of the ANFIS and conclude the active part of the controller.

For complete simulation software functions, refer to Appendix B.

Learning Algorithm for ANFIS Controller

The learning algorithm for the ANFIS is calculated using the back propagation method as described previously. An input set of five input/output pairs per fuzzy subset is used to train the controller. The program reads this data set from a normal comma separated variable (CSV) file and propagates the result of each back through the network to obtain the optimal parameters.

Figure 5-19 is diagram illustrating the process of collecting new data from the MPPT and used off-line to calculate new control parameters. After collecting sensor inputs for a number of cycles, the average value is calculated for more accurate readings. This set of results is then exported to the laptop where the MPP is calculated for the specific curve after the full curve data is collected. After every output set, the duty cycle is adjusted in order to get process values for the power curve at a specific set of conditions.

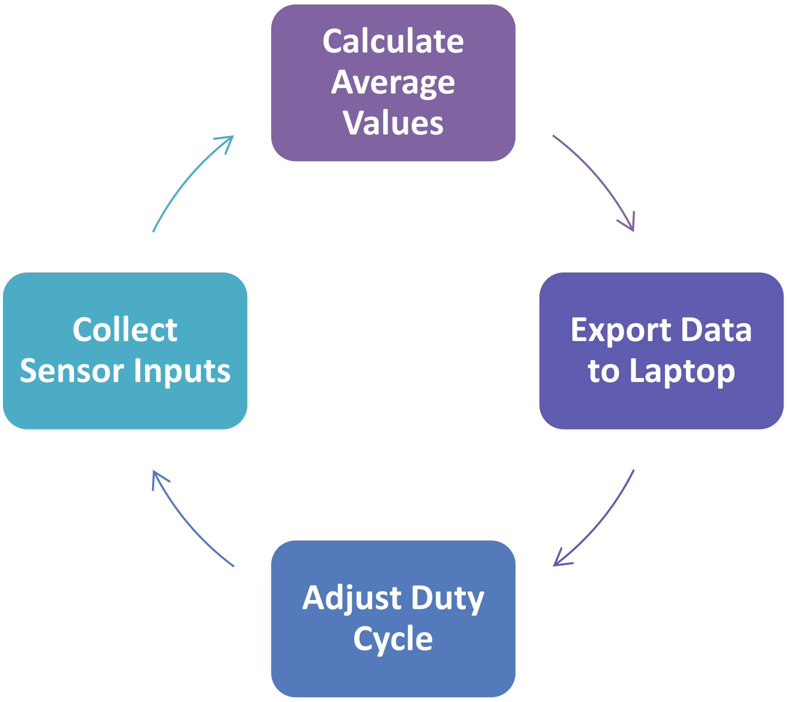


Figure 5-19: Collection of new data diagram

The error of each output is obtained as the difference between the PV simulation MPP duty cycle and the result of the ANFIS controller.

The following functions are used to update the first, second and third layer parameters of the ANFIS network respectively:

- learnInput()
- learnwT()
- learnP()

For every MPP and input set, that is read into the ANFIS, there is an update of the program parameters of the network through the back propagation algorithm. The update methodology is to subtract the change in error of the program output with regards to the parameter that needs to be updated, multiplied by the learning rate of the update, from the parameter value after the previous update [44]. The error is therefore literary back propagated through the network to determine what effect a specific parameter has on the output error.

Figure 5-20 is a functional flow diagram of the learning algorithm program. It is an iterative process that cycles until the output error approaches zero for a specific set of parameters. This procedure is followed for every set of parameters and the whole process repeated for better accuracy and stability.

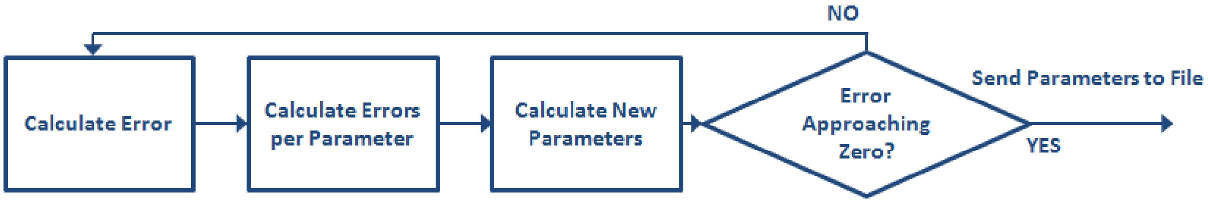


Figure 5-20: Functional flow diagram of learning algorithm

For complete software functions, refer to Appendix D.

The changes in error with respect to the parameters that is required are listed below:

- 1st layer: $\frac{E_d}{\sigma_{jT}}, \frac{E_d}{\sigma_{kG}}, \frac{E_d}{c_{jT}}, \frac{E_d}{c_{kG}}$
- 2nd layer: $\frac{E_d}{w_{jkT}}, \frac{E_d}{w_{jkG}}$
- 3d layer: $\frac{E_d}{p_{jk}}, \frac{E_d}{q_{jk}}, \frac{E_d}{r_{jk}}$

The error E_d can be defined as:

$$E_d = \frac{1}{2}(targetDutyCycle - outputDutyCycle)^2 \quad [44] \tag{5.52}$$

Figure 5-21is a small section of the ANFIS algorithm. Please note the following numbering convention:

L_xN_y, where L_x is the layer number and N_y is the node number. This diagram was used to develop the differential equations needed to produce a back propagation learning algorithm.

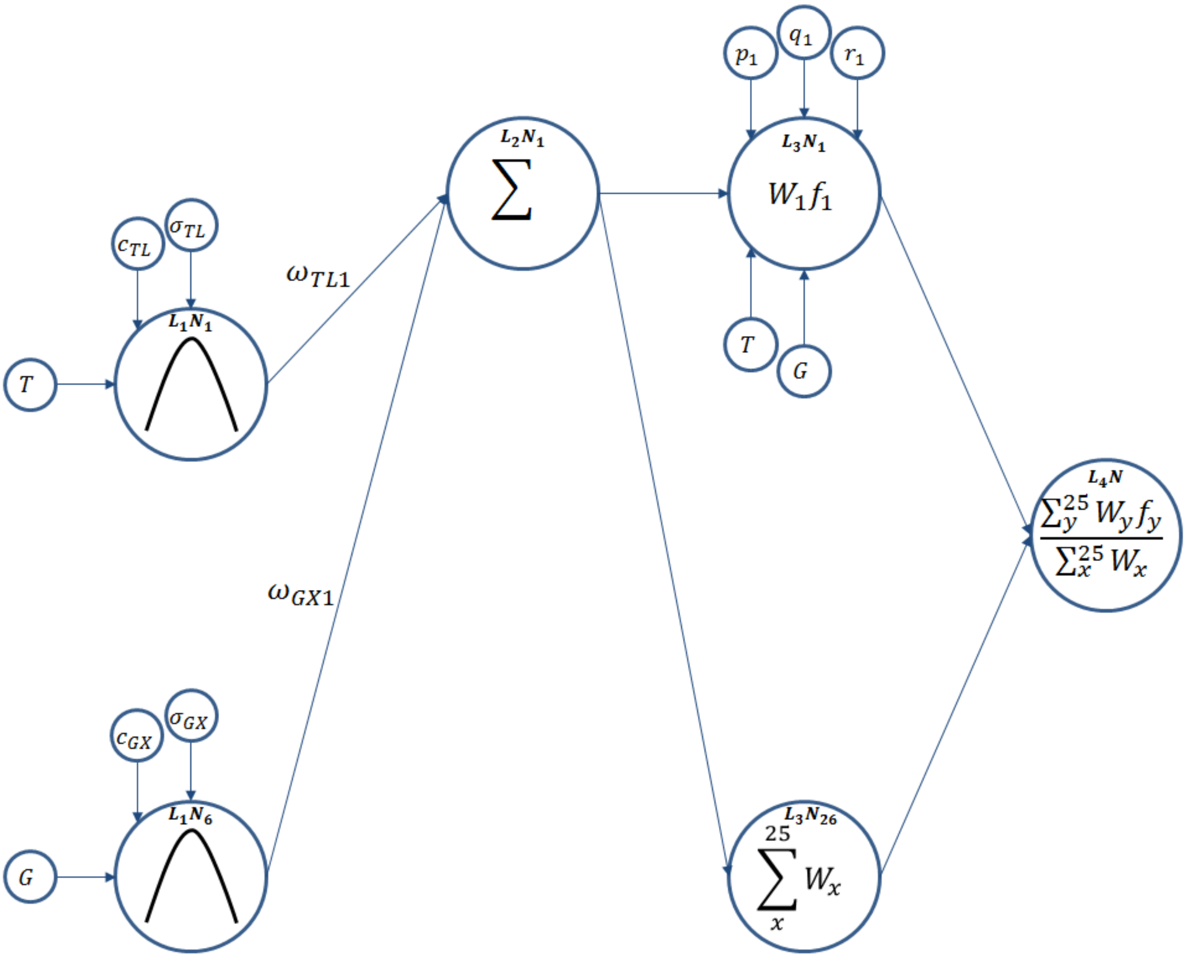


Figure 5-21: Section of the ANFIS algorithm

The first set of equations was developed to update the p_x, q_x and r_x parameters. The individual steps are shown for p_1 only. For the other linear function parameters, because the process is very similar, with only small changes in each end result needed to produce a formula for each parameter.

The purpose is to determine how the output error of the control system changes with regards to modifications done to parameter p_1. This term, $\frac{\partial E_d}{\partial p_1}$, can be calculated by propagating the error back through the control system up to the variable/parameter of interest.

$$\frac{\partial E_d}{\partial p_1} = \frac{\partial E_d^i}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial p_1} \tag{5.53}$$

$$\frac{\partial E_d^i}{\partial out_{L_3N_1}} = \frac{\partial E_d}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} \tag{5.54}$$

$$\frac{\partial E_d^i}{\partial out_{L_3N_1}} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}}$$

$$\frac{\partial E_d}{\partial p_1} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial p_1}$$

The output of the fourth layer node is equal to the output of the control system, i.e. the MPPT duty cycle. The error E_d can then be defined as:

$$E_d = \frac{1}{2}(targetDutyCycle - outputDutyCycle)^2$$

$$\partial out_{L_4N} = outputDutyCycle$$

$$E_d = \frac{1}{2}(targetDutyCycle - out_{L_4N})^2$$

The following parameters were calculated to complete the differential equation:

$$\frac{\partial E_d}{\partial out_{L_4N}} = \frac{\partial(\frac{1}{2}targetDutyCycle^2 - targetDutyCycle.\partial out_{L_4N} + \frac{1}{2}out_{L_4N}{}^2)}{out_{L_4N}}$$

$$\frac{\partial E_d}{\partial out_{L_4N}} = out_{L_4N} - targetDutyCycle$$

$$\frac{\partial out_{L_4N}}{\partial net_{L_4N}} = 1$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} = \frac{\partial(out_{L_3N_1} + \cdots + out_{L_3N_{25}})/\sum_x^{25} W_x}{out_{L_3N_1}}$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} = \frac{1}{\sum_x^{25} W_x}$$

$$\frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} = 1$$

$$\frac{\partial net_{L_3N_1}}{\partial p_1} = \frac{\partial[W_1(p_1T + q_1G + r_1)]}{\partial p_1}$$

$$\frac{\partial net_{L_3N_1}}{\partial p_1} = W_1T$$

When all of the above parameters are combined in the original differential equation, the following equation is obtained:

$$\frac{\partial E_d}{\partial p_1} = (out_{L_4N} - targetDutyCycle) \times \frac{1}{\sum_x^{25} W_x} \times W_1 T \tag{5.55}$$

This equation is valid for all p_x, q_x and r_x parameters when rewritten as follows:

$$\frac{\partial E_d}{\partial p_z} = (out_{L_4N} - targetDutyCycle) \times \frac{1}{\sum_x^{25} W_x} \times W_z T \tag{5.56}$$

$$\frac{\partial E_d}{\partial q_z} = (out_{L_4N} - targetDutyCycle) \times \frac{1}{\sum_x^{25} W_x} \times W_z G \tag{5.57}$$

$$\frac{\partial E_d}{\partial r_z} = (out_{L_4N} - targetDutyCycle) \times \frac{1}{\sum_x^{25} W_x} \times W_z \tag{5.58}$$

The error rate for each parameter can then be used to update the actual parameter, based on the gradient of change in error. Hence, the larger effect a parameter has on the output error, the more it shall change to reduce the error. This rate of change of a parameter can however be controlled to stabilize the learning algorithm through the setting of the learning rate parameter, η.

$$p_{z+1} = p_z - \eta \frac{\partial E_d}{\partial p_z} \tag{5.59}$$

$$q_{z+1} = q_z - \eta \frac{\partial E_d}{\partial q_z} \tag{5.60}$$

$$r_{z+1} = r_z - \eta \frac{\partial E_d}{\partial r_z} \tag{5.61}$$

The second set of equations was developed to update the weights for each of the inputs to calculate the weighted average between temperature and irradiation fuzzy set membership function results. This totals five weights per fuzzy set.

The following differential equation was developed to calculate the effect that the input weight, ω_{TL1}, has on the output error. This is the weight for the low temperature fuzzy set that is input into the first weighted average node:

$$\frac{\partial E_d}{\partial \omega_{TL1}} = \frac{\partial E_d}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} \times \frac{\partial net_{L_2N_1}}{\partial \omega_{TL1}} \tag{5.62}$$

The propagation of the output error back to the second layer of the network yielded the following equation:

$$\frac{\partial E_d}{\partial out_{L_2N_1}} = \frac{\partial E_d{}^{i}}{\partial out_{L_2N_1}} + \frac{\partial E_d{}^{ii}}{\partial out_{L_2N_1}} \tag{5.63}$$

As can be seen in the network, the error can be propagated back to the second layer through two paths. The differential equation for the first path is defined as:

$$\frac{\partial E_d{}^{i}}{\partial out_{L_2N_1}} = \frac{\partial E_d{}^{i}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} \tag{5.64}$$

$$\frac{\partial E_d{}^{i}}{\partial out_{L_2N_1}} = \frac{\partial E_d{}^{i}}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} \tag{5.65}$$

$$\frac{\partial E_d{}^{i}}{\partial out_{L_2N_1}} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} \tag{5.66}$$

For the first path error propagation, the following parameters were equated:

$$\frac{\partial E_d}{\partial out_{L_4N}} = out_{L_4N} - targetDutyCycle$$

$$\frac{\partial out_{L_4N}}{\partial net_{L_4N}} = 1$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} = \frac{1}{\sum_x^{25} W_x}$$

$$\frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} = 1$$

$$\frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} = \frac{\partial [W_1(p_1T+q_1G+r_1)]}{\partial W_1}$$

$$\frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} = p_1T + q_1G + r_1$$

The differential equation for the second path is defined as:

$$\frac{\partial E_d{}^{ii}}{\partial out_{L_2N_1}} = \frac{\partial E_d{}^{ii}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \tag{5.67}$$

$$\frac{\partial {E_d}^{ii}}{\partial out_{L_2N_1}} = \frac{\partial {E_d}^{ii}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \qquad (5.68)$$

$$\frac{\partial {E_d}^{ii}}{\partial out_{L_2N_1}} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \qquad (5.69)$$

For the first path error propagation, the following parameters were equated:

$$\frac{\partial E_d}{\partial out_{L_4N}} = out_{L_4N} - targetDutyCycle$$

$$\frac{\partial out_{L_4N}}{\partial net_{L_4N}} = 1$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} = \frac{\partial(out_{L_3N_1} + \cdots + out_{L_3N_{25}})/out_{L_3N_{26}}}{\partial out_{L_3N_{26}}}$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} = -\left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right).{out_{L_3N_{26}}}^{-2}$$

$$\frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} = 1$$

$$\frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} = \frac{\sum_x^{25} W_x}{W_1}$$

$$\frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} = 1$$

$$\frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} = 1$$

$$\frac{\partial net_{L_2N_1}}{\partial \omega_{TL1}} = \frac{\partial(\omega_{TL1} out_{L_1N_1} + \omega_{GX1} out_{L_1N_6})}{\partial \omega_{TL1}}$$

$$\frac{\partial net_{L_2N_1}}{\partial \omega_{TL1}} = out_{L_1N_1}$$

$$\frac{\partial net_{L_2N_1}}{\partial \omega_{TL1}} = e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}$$

By combining the differential equations for each path, the following equation was developed:

$$\frac{\partial E_d}{\partial \omega_{TL1}} = \left(out_{L_4N} - targetDutyCycle\right). e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}. [\frac{1}{\sum_x^{25} W_x}(p_1T + q_1G + r_1) - \left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right). out_{L_3N_{26}}{}^{-2}]$$

Similarly the following equations are examples for a few of the other parameters:

$$\frac{\partial E_d}{\partial \omega_{TX5}} = \left(out_{L_4N} - targetDutyCycle\right). e^{[-((T-c_{TX})/\sqrt{2}\sigma_{TX})^2]}. [\frac{1}{\sum_x^{25} W_x}(p_{25}T + q_{25}G + r_{25}) - \left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right). out_{L_3N_{26}}{}^{-2}]$$

$$\frac{\partial E_d}{\partial \omega_{GX1}} = \left(out_{L_4N} - targetDutyCycle\right). e^{[-((G-c_{GX})/\sqrt{2}\sigma_{GX})^2]}. [\frac{1}{\sum_x^{25} W_x}(p_1T + q_1G + r_1) - \left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right). out_{L_3N_{26}}{}^{-2}]$$

$$\frac{\partial E_d}{\partial \omega_{GH5}} = \left(out_{L_4N} - targetDutyCycle\right). e^{[-((G-c_{GH})/\sqrt{2}\sigma_{GH})^2]}. [\frac{1}{\sum_x^{25} W_x}(p_{25}T + q_{25}G + r_{25}) - \left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right). out_{L_3N_{26}}{}^{-2}]$$

The rate parameters are once again updated through the rate of change in error due to the effect of the specific parameter.

$$\omega_{TL1+1} = \omega_{TL1} - \eta\frac{\partial E_d}{\partial \omega_{TL1}} \tag{5.70}$$

The last set of equations was developed to calculate the membership function parameters for each of the fuzzy sets.

The following differential equation was developed to calculate the effect that the fuzzy set parameter, c_{TL}, has on the output error. This is the membership function for the low temperature fuzzy set that determines the centre point of the curve:

$$\frac{\partial E_d}{\partial c_{TL}} = \frac{\partial E_d}{\partial out_{L_1N_1}} \times \frac{\partial out_{L_1N_1}}{\partial net_{L_1N_1}} \times \frac{\partial net_{L_1N_1}}{\partial c_{TL}} \tag{5.71}$$

The propagation of the output error back to the first layer of the network yielded the following equation:

$$\frac{\partial E_d}{\partial out_{L_1N_1}} = \frac{\partial {E_d}^{i}}{\partial out_{L_1N_1}} + \frac{\partial {E_d}^{ii}}{\partial out_{L_1N_1}} + \frac{\partial {E_d}^{iii}}{\partial out_{L_1N_1}} + \frac{\partial {E_d}^{iv}}{\partial out_{L_1N_1}} + \frac{\partial {E_d}^{v}}{\partial out_{L_1N_1}} \tag{5.72}$$

As can be seen in the network, the error can be propagated back to the first layer through five paths. The differential equation for the first path is defined as:

$$\frac{\partial {E_d}^{i}}{\partial out_{L_1N_1}} = \frac{\partial {E_d}^{i}}{\partial net_{L_2N_1}} \times \frac{\partial net_{L_2N_1}}{\partial out_{L_1N_1}} \tag{5.73}$$

$$\frac{\partial {E_d}^{i}}{\partial net_{L_2N_1}} = \frac{\partial {E_d}^{iA}}{\partial net_{L_2N_1}} + \frac{\partial {E_d}^{iB}}{\partial net_{L_2N_1}} \tag{5.74}$$

$$\frac{\partial {E_d}^{iA}}{\partial net_{L_2N_1}} = \frac{\partial {E_d}^{iA}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} \tag{5.75}$$

$$\frac{\partial {E_d}^{iA}}{\partial net_{L_2N_1}} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} \tag{5.76}$$

$$\frac{\partial {E_d}^{iB}}{\partial net_{L_2N_1}} = \frac{\partial {E_d}^{iB}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} \tag{5.77}$$

$$\frac{\partial {E_d}^{iB}}{\partial net_{L_2N_1}} = \frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} \tag{5.78}$$

$$\frac{\partial {E_d}^{i}}{\partial out_{L_1N_1}} = \left[\left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} \times \frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} \times \frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}}\right) + \left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}}\right)\right] \times \frac{\partial net_{L_2N_1}}{\partial out_{L_1N_1}} \tag{5.79}$$

Similarly the equations for the other four paths were obtained:

$$\frac{\partial {E_d}^{ii}}{\partial out_{L_1N_1}} = \left[\left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_2}} \times \frac{\partial out_{L_3N_2}}{\partial net_{L_3N_2}} \times \frac{\partial net_{L_3N_2}}{\partial out_{L_2N_2}} \times \frac{\partial out_{L_2N_2}}{\partial net_{L_2N_2}}\right) + \left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_2}}{\partial net_{L_2N_2}}\right)\right] \times \frac{\partial net_{L_2N_2}}{\partial out_{L_1N_1}} \tag{5.80}$$

$$\frac{\partial {E_d}^{iii}}{\partial out_{L_1N_1}} = \left[\left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_3}} \times \frac{\partial out_{L_3N_3}}{\partial net_{L_3N_3}} \times \frac{\partial net_{L_3N_3}}{\partial out_{L_2N_3}} \times \frac{\partial out_{L_2N_3}}{\partial net_{L_2N_3}}\right) + \left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_3}}{\partial net_{L_2N_3}}\right)\right] \times \frac{\partial net_{L_2N_3}}{\partial out_{L_1N_1}} \tag{5.81}$$

$$\frac{\partial {E_d}^{iV}}{\partial out_{L_1N_1}} = \left[\left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_4}} \times \frac{\partial out_{L_3N_4}}{\partial net_{L_3N_4}} \times \frac{\partial net_{L_3N_4}}{\partial out_{L_2N_4}} \times \frac{\partial out_{L_2N_4}}{\partial net_{L_2N_4}}\right) + \left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_4}}{\partial net_{L_2N_4}}\right)\right] \times \frac{\partial net_{L_2N_4}}{\partial out_{L_1N_1}} \tag{5.82}$$

$$\frac{\partial {E_d}^{V}}{\partial out_{L_1N_1}} = \left[\left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_5}} \times \frac{\partial out_{L_3N_5}}{\partial net_{L_3N_5}} \times \frac{\partial net_{L_3N_5}}{\partial out_{L_2N_5}} \times \frac{\partial out_{L_2N_5}}{\partial net_{L_2N_5}}\right) + \left(\frac{\partial E_d}{\partial out_{L_4N}} \times \frac{\partial out_{L_4N}}{\partial net_{L_4N}} \times \frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} \times \frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} \times \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} \times \frac{\partial out_{L_2N_5}}{\partial net_{L_2N_5}}\right)\right] \times \frac{\partial net_{L_2N_5}}{\partial out_{L_1N_1}} \tag{5.83}$$

The parameters below were equated and plugged into the above differential equations:

$$\frac{\partial out_{L_1N_1}}{\partial net_{L_1N_1}} = 1$$

$$\frac{\partial net_{L_1N_1}}{\partial c_{TL}} = \frac{\partial e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}}{\partial c_{TL}}$$

$$\frac{\partial net_{L_1N_1}}{\partial c_{TL}} = (\frac{T}{{\sigma_{TL}}^2} - \frac{1}{{\sigma_{TL}}^2} c_{TL})e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}$$

$$\frac{\partial E_d}{\partial out_{L_4N}} = out_{L_4N} - targetDutyCycle$$

$$\frac{\partial out_{L_4N}}{\partial net_{L_4N}} = 1$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_1}} = \frac{\partial net_{L_4N}}{\partial out_{L_3N_2}} = \frac{\partial net_{L_4N}}{\partial out_{L_3N_3}} = \frac{\partial net_{L_4N}}{\partial out_{L_3N_4}} = \frac{\partial net_{L_4N}}{\partial out_{L_3N_5}} = \frac{1}{\sum_x^{25} W_x}$$

$$\frac{\partial out_{L_3N_1}}{\partial net_{L_3N_1}} = \frac{\partial out_{L_3N_2}}{\partial net_{L_3N_2}} = \frac{\partial out_{L_3N_3}}{\partial net_{L_3N_3}} = \frac{\partial out_{L_3N_4}}{\partial net_{L_3N_4}} = \frac{\partial out_{L_3N_5}}{\partial net_{L_3N_5}} = 1$$

$$\frac{\partial net_{L_3N_1}}{\partial out_{L_2N_1}} = p_1T + q_1G + r_1$$

$$\frac{\partial net_{L_3N_2}}{\partial out_{L_2N_2}} = p_2T + q_2G + r_2$$

$$\frac{\partial net_{L_3N_3}}{\partial out_{L_2N_3}} = p_3T + q_3G + r_3$$

$$\frac{\partial net_{L_3N_4}}{\partial out_{L_2N_4}} = p_4T + q_4G + r_4$$

$$\frac{\partial net_{L_3N_5}}{\partial out_{L_2N_5}} = p_5T + q_5G + r_5$$

$$\frac{\partial out_{L_2N_1}}{\partial net_{L_2N_1}} = \frac{\partial out_{L_2N_2}}{\partial net_{L_2N_2}} = \frac{\partial out_{L_2N_3}}{\partial net_{L_2N_3}} = \frac{\partial out_{L_2N_4}}{\partial net_{L_2N_4}} = \frac{\partial out_{L_2N_5}}{\partial net_{L_2N_5}} = 1$$

$$\frac{\partial net_{L_4N}}{\partial out_{L_3N_{26}}} = -\left(out_{L_3N_1} + \cdots + out_{L_3N_{25}}\right). out_{L_3N_{26}}{}^{-2}$$

$$\frac{\partial out_{L_3N_{26}}}{\partial net_{L_3N_{26}}} = 1$$

$$\frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_1}} = \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_2}} = \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_3}} = \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_4}} = \frac{\partial net_{L_3N_{26}}}{\partial out_{L_2N_5}} = 1$$

$$\frac{\partial net_{L_2N_1}}{\partial out_{L_1N_1}} = \frac{\partial(\omega_{TL1} out_{L_1N_1} + \omega_{GX1} out_{L_1N_6})}{\partial out_{L_1N_1}}$$

$$\frac{\partial net_{L_2N_1}}{\partial out_{L_1N_1}} = \omega_{TL1}$$

$$\frac{\partial net_{L_2N_2}}{\partial out_{L_1N_1}} = \omega_{TL2}$$

$$\frac{\partial net_{L_2N_3}}{\partial out_{L_1N_1}} = \omega_{TL3}$$

$$\frac{\partial net_{L_2N_4}}{\partial out_{L_1N_1}} = \omega_{TL4}$$

$$\frac{\partial net_{L_2N_5}}{\partial out_{L_1N_1}} = \omega_{TL5}$$

For the other parameter in the membership function (determines the width of the function), σ_{TL}, the following equation applies:

$$\frac{\partial E_d}{\partial \sigma_{TL}} = \frac{\partial E_d}{\partial out_{L_1N_1}} \times \frac{\partial out_{L_1N_1}}{\partial net_{L_1N_1}} \times \frac{\partial net_{L_1N_1}}{\partial \sigma_{TL}} \qquad (5.84)$$

This equation only differs by one parameter and can be equated as follows:

$$\frac{\partial net_{L_1N_1}}{\partial \sigma_{TL}} = \frac{\partial e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}}{\partial \sigma_{TL}} \qquad (5.85)$$

$$\frac{\partial net_{L_1N_1}}{\partial \sigma_{TL}} = (\frac{(T-c_{TL})^2}{\sigma_{TL}{}^3}) e^{[-((T-c_{TL})/\sqrt{2}\sigma_{TL})^2]}$$

The rate of change in error can then be used to update these two parameters:

$$c_{TL+1} = c_{TL} - \eta \frac{\partial E_d}{\partial c_{TL}} \tag{5.86}$$

$$\sigma_{TL+1} = \sigma_{TL} - \eta \frac{\partial E_d}{\partial \sigma_{TL}} \tag{5.87}$$

The same equations, including a few modifications, apply to all of the other parameters located in the other membership functions.

Integration of MPPT controller, shadow detection and deficient panel detection

Simulation results

The results of the fuzzy controller were very stable, even though the values were only accurate for temperatures close to 25°C. With some trial-and-error adjustments the controller can be perfected, however the very fact that time consuming adjustments are needed for such a controller was the result needed to justify the development of the ANFIS controller.

Figure 5-22 is a graphic representation if the duty cycle offset from MPP duty cycle per data set with only the random initial ANFIS parameters entered.

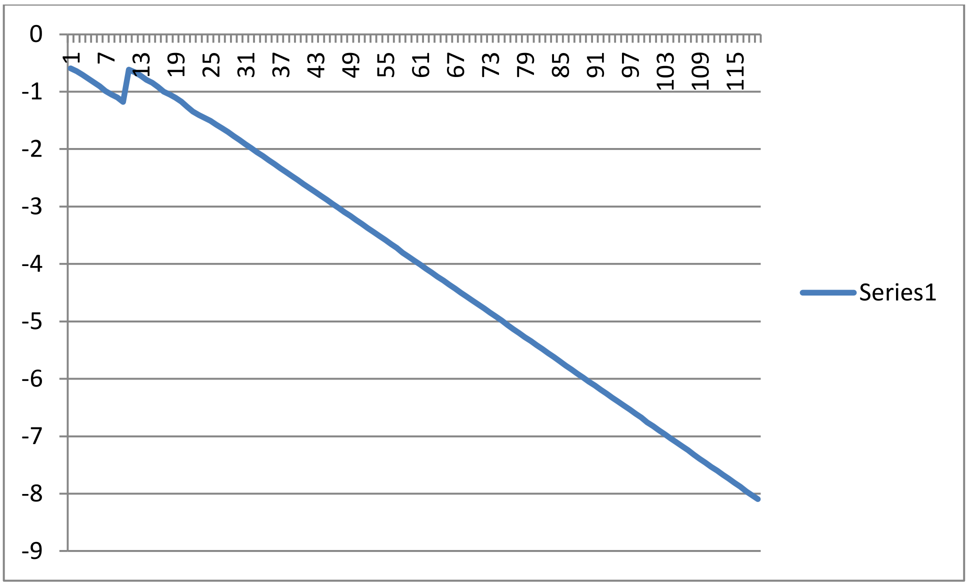


Figure 5-22: Duty cycle offset from MPP duty cycle

It is clear that the system is highly unstable and useless without a learning algorithm.

The ANFIS controller with after one cycle of running the ANFIS learning algorithm gave rather stable results. The initial simulated results (same example used as for the fuzzy controller) are shown in Figure 5-23. 99 Sets of data obtained from the solar panel were used to train the controller.

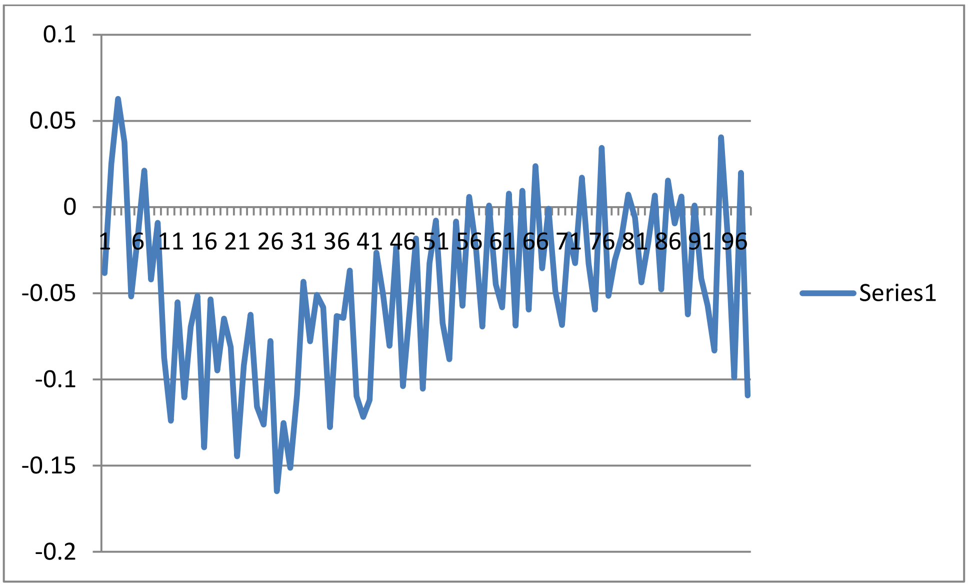


Figure 5-23: Initial results after implementing the ANFIS controller

The ANFIS controller was then updated using the same data set, but running the learning algorithm for 5 cycles. The results are shown in Figure 5-24.

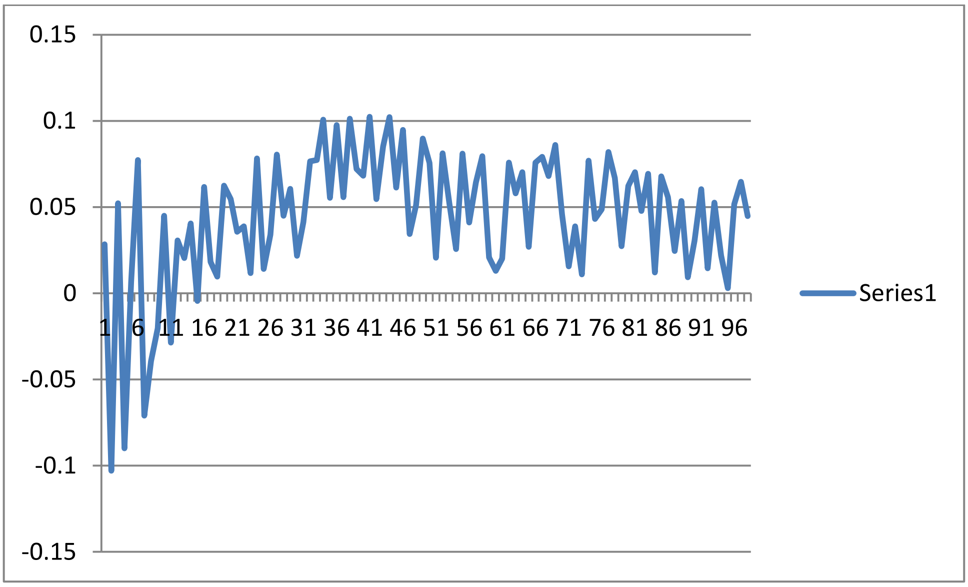


Figure 5-24: ANFIS controller results after 5 iterations of learning were applied

The overall results look good and the same principles can be applied to update the MPPT parameters offline. During lower temperatures the controller is less accurate, but still adequate enough to enable good test results. The numbers in the left pane indicates the difference between the actual MPP and the controller proposed MPP and the x-axis represents the number of the data set.

In order to implement the ANFIS controller in software on the microprocessor, ADC example code obtained from www.microchip.com was used [45].

Application of ANFIS Controller on Microprocessor

Figure 5-25 is a functional flow diagram of the software executed on the microprocessor of the MPPT.

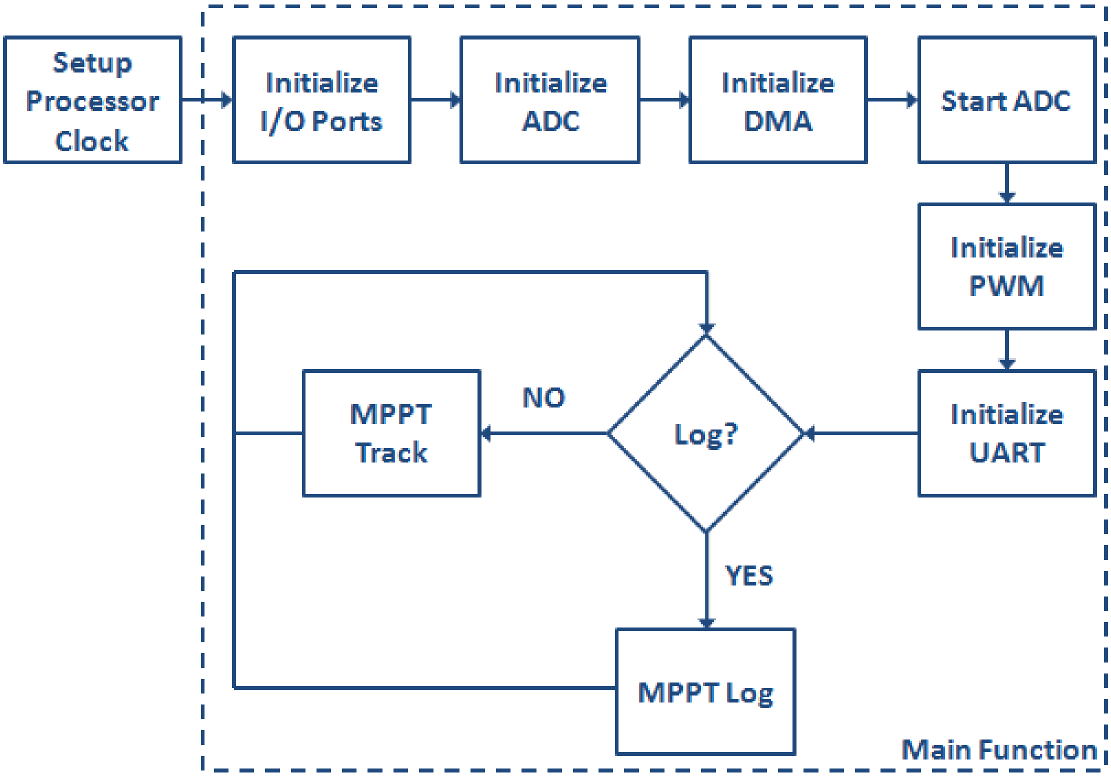


Figure 5-25: Functional flow diagram microprocessor software overview

After the setup of the microprocessor clock, the software enters the main function, main(). All of the other functions are executed within this function.

The following functions are used to setup ports and modules:

- IOInit()
    - Initialize digital input and output ports
- ADCInit()
    - Initialize analogue-to-digital converter module
- DMAInit()
    - Initialize direct memory access for storage of ADC values
- ADCStart()
    - Start the ADC module
- OC1Init()
    - Initialize output compare module for PWM signal
- UARTInit()

    - Initialize UART module for communication

After initialization, the program enters an infinitive loop with two executions possible:

- MPPT_Track()
    - Control function to calculate MPP
- MPPT_Curve()
    - Log function to send curve data to the laptop for processing

Figure 5-26 is a functional flow diagram of the MPPT_Track() function:

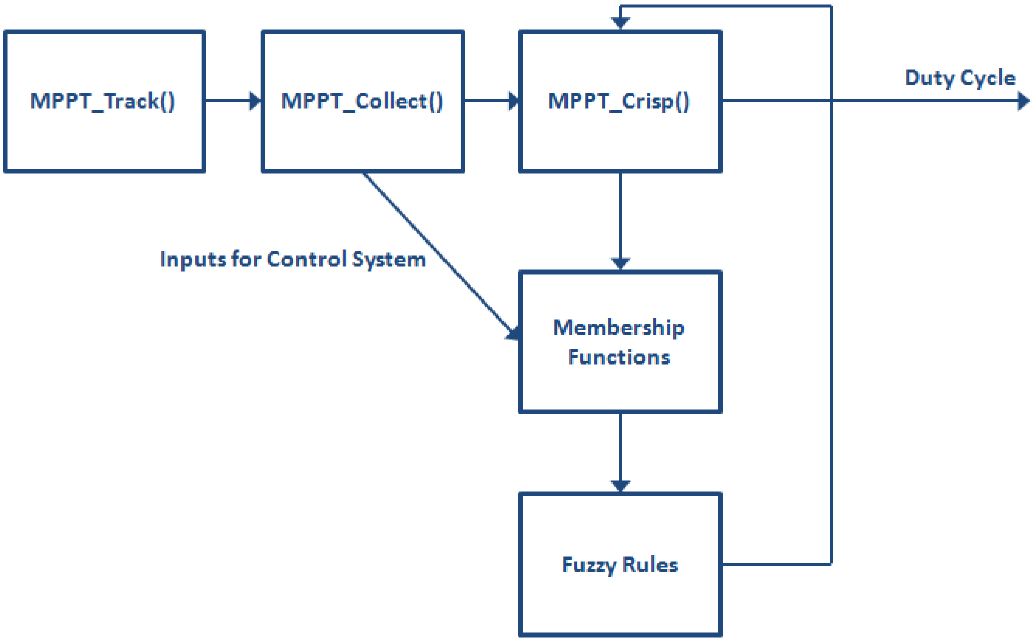


Figure 5-26: Functional flow diagram of MPPT tracking function

The first function called is the MPPT_Collect() function to source inputs from sensors. The MPPT_Crisp() function is then executed, containing the ANFIS algorithm and referencing membership functions and fuzzy rules.

For complete software functions, refer to Appendix C.

5.3 Conclusion: Chapter 5

The control system was developed and mathematically simulated to get a good feel of the accuracy of such a model. The software was then developed for the actual MPPT controller.

Chapter 6 - Integration and Testing of Prototype MPPT

With all the individual modules designed, manufactured and tested, the next step is to assemble all of them into one working system.

After integration a test schedule was set up. This schedule was however not adhered to due to fluctuating weather conditions. A better approach was then implemented by setting up a list of tests to be done, along with the expected outcome and the real outcome.

In some instances weights had to be applied to the vehicle in order to prevent it from being blown over by the wind. This extra weight was assumed to have a negligible effect on the test results and consequently ignored.

The same software used on the compute to determine the ANFIS result was used in the microprocessor. Due to the language differences this code had to be manually coded from python to c as well as real sensor values added.

6.1 Integration of Prototype MPPT

The asembled vehicle is easy to transport and the speed can be set low enough for the vehicle to be controlable without distance controle.

All functionalites were tested and the vehicle was deemed ready to begin tests with.

In terms of software, the ANFIS algorithm and IV profile logger programs were installed on the microprocessor of the MPPT. The learning algorithm of the ANFIS was done offline on a computer with the collected data in due to the high processing and memory requirements. After the ANFIS parameters were calculated on the computer, they were manually transferred to the microprocessor to update the software.

6.2 Prototype MPPT Experiments

The following procedures were followed in order to test the MPPT algorithm:

1. All sensors were calibrated to give accurate measurements.
2. IV curve data was collected in order to create a knowledge base from which to train the ANFIS.
3. The data collected was then implemented on the offline ANFIS system to create new parameters for the control system.

4. The programmer was used to update the microcontroller parameters in order to test for any system response improvements.
5. A number of experiments were then performed after the control system was deemed stable enough.
6. As new data became available, the control system parameters were updated again do try and improve the accuracy of the results.

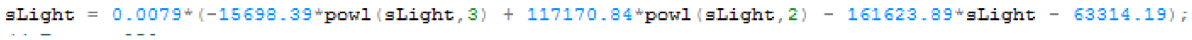
Sensor calibration

The following sensors are linear with regards to the voltage levels measured. In order to calibrate them a multi-meter was used to determine what the conversion factor is between the measured value on the processor and the actual reading:

- MPPT input current – 24.96
- MPPT input voltage - 21
- MPPT output/battery voltage – 35.17

The conversion numbers used in software is listed next to the sensor names.

The light dependant resistors' results were found to best for a second order linear function when a curve fit was done between the readings and actual measurement with a light meter. A small program as part of the ANFIS package was code in order to do an easy conversion on a computer. This program produced a linear third order equation with was used for conversion, along with the 0.0079 conversion factor to get the final values in W/m^2. An extract from the controller code is included to indicate the conversion.



```
sLight = 0.0079*(-15698.39*powl(sLight,3) + 117170.84*powl(sLight,2) - 161623.89*sLight - 63314.19);
```

The temperature sensor output was confirmed with a thermometer for low temperatures and the result was within 5% of the rated specifications.

IV curve collection

About 99 curves had been collected as a knowledge base for the ANFIS control system. Most of the curves fall within the set scope of the project, namely between 20°C – 60°C and 12,000lx – 120,000lx.

All of the data can be found on the data disc that is attached to this dissertation and Appendix A. Figure 6-1 illustrates a few IV curves at a constant panel temperature of 42°C.

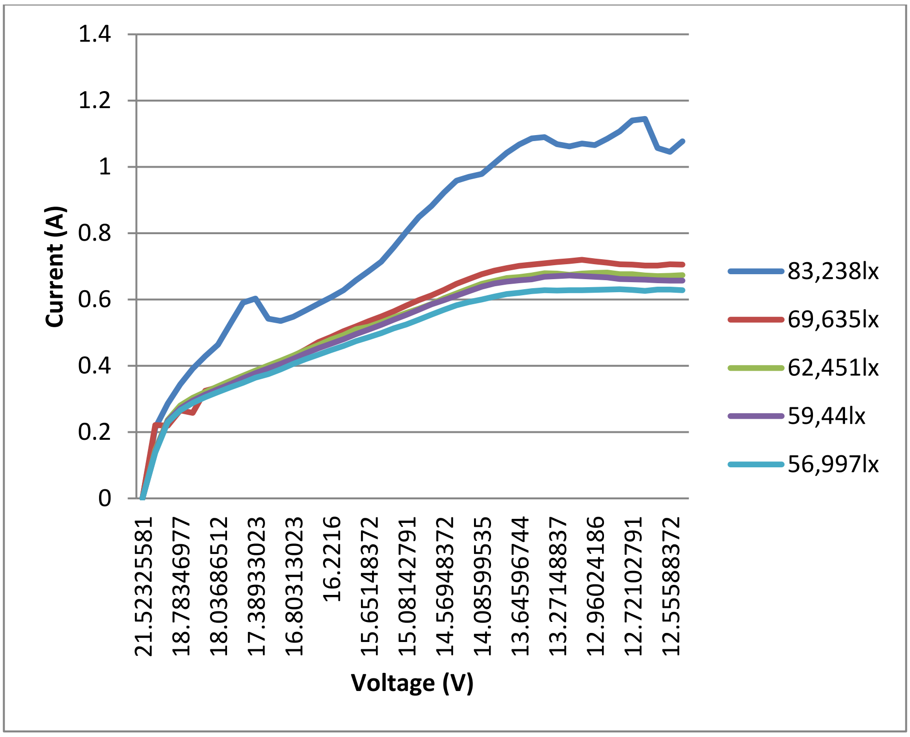


Figure 6-1: Current versus voltage for a range of 5 light intensities at constant 42°C

Figure 6-1 it can be observed that the:

- Current reduces as the light intensity decreases.
- Current becomes constant as the voltage approaches lower values.

Figure 6-2 is the power curve of the panel for the same conditions.

From Figure 6-2 it can be observed that the:

- Power start to drop after the “knee” of the curve.
- Power reduces as the light intensity decreases.

Table 6-1 lists the 5 MPP’s under these conditions.

	Max Power	Voltage
83,238lx	14.67	13.52
69,635lx	9.58	13.78
62,635lx	9.16	13.78
59,44lx	9.03	13.93
56,997lx	8.49	13.77

Table 6-1: MPP for every test curve

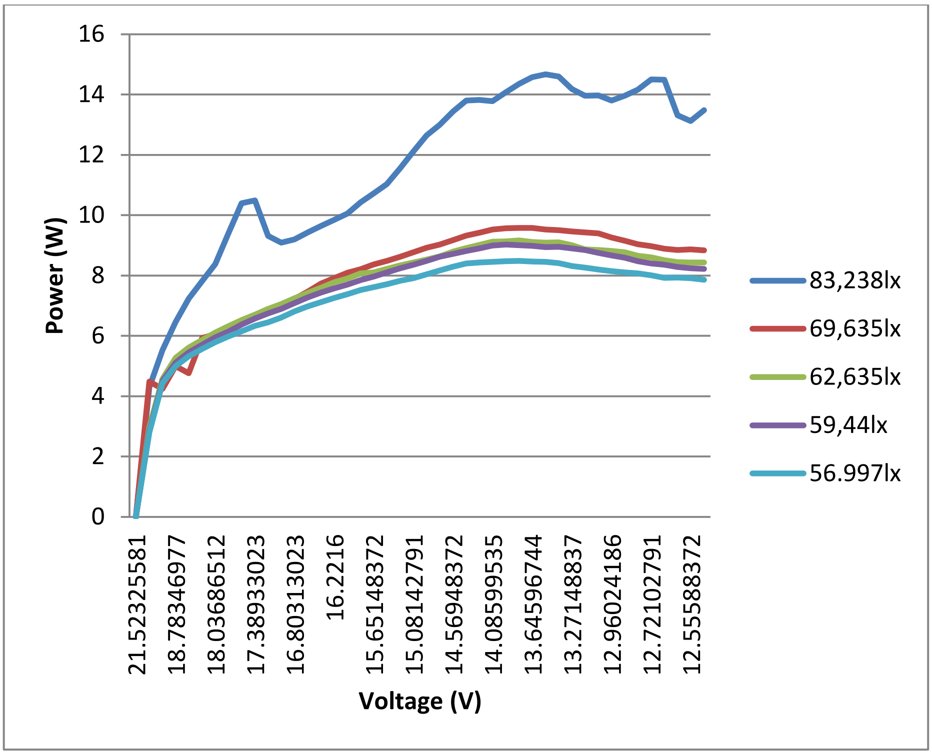


Figure 6-2: Power versus voltage for a range of 5 light intensities at constant 42°C

Figure 6-3 is a current voltage curve while at an average of 84,943lx and 2 different temperature measurements: 23.4°C and 42°C.

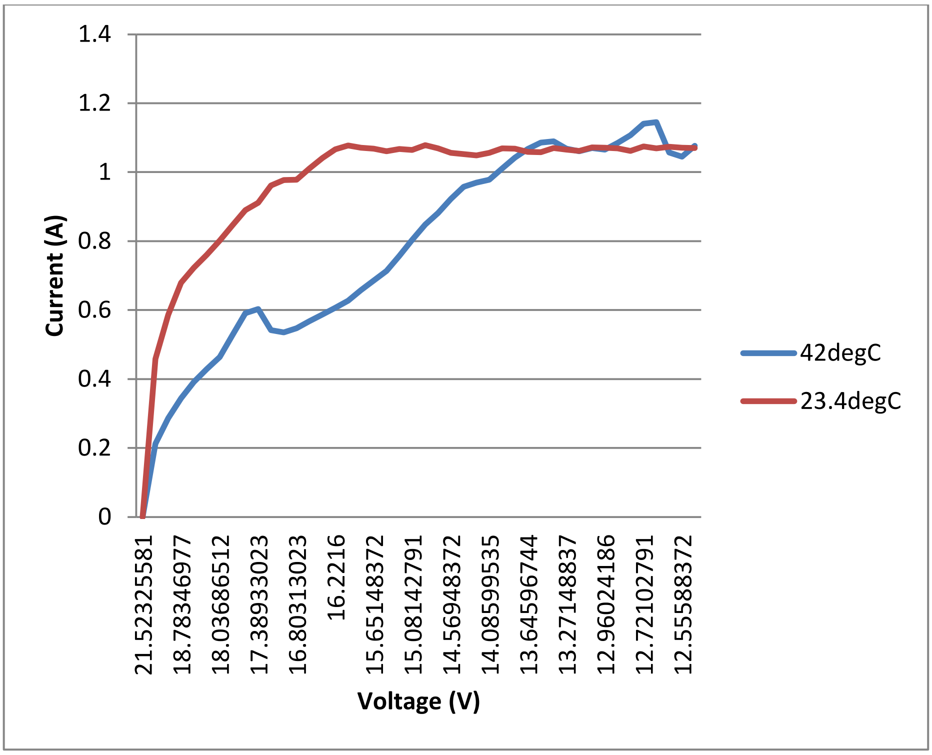


Figure 6-3: Current versus voltage for a range of 2 PV cell temperatures at constant light intensity

Figure 6-4 is the power curve for the above measurements.

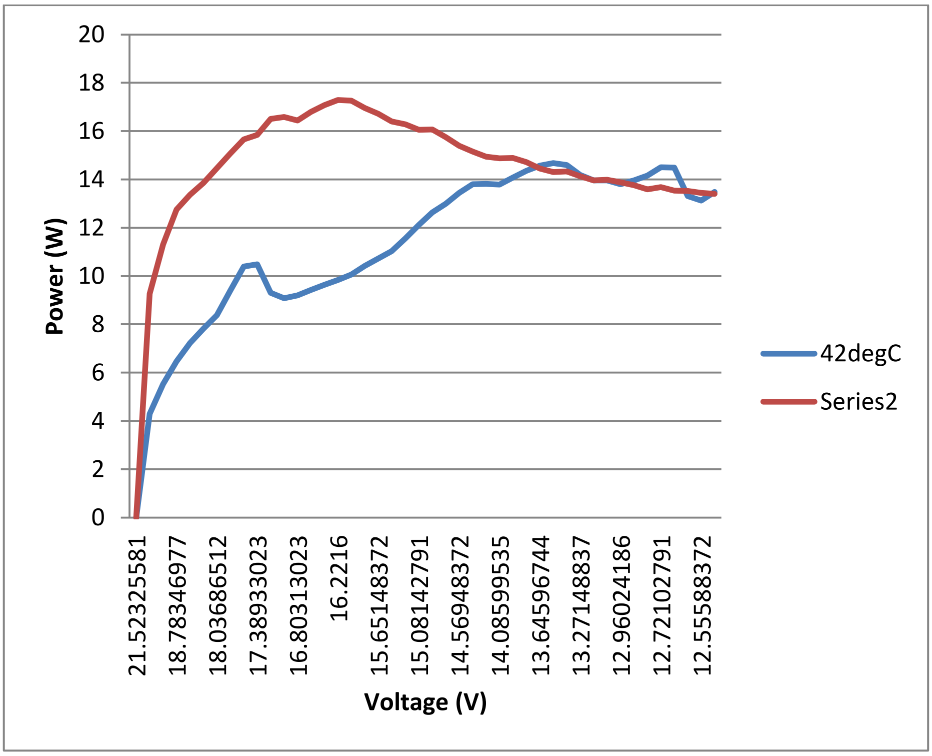


Figure 6-4: Power versus voltage for a range of 2 PV cell temperatures at constant light intensity

As expected the MPP moves forward and the new voltage is 16.22V. The maximum power level also rises to 17.29W.

What was clearly noted through all of the tests was the fast response time of the control system. Due to the fact that no tracking is involved, the MPPT can move directly to the estimated MPP. This also excludes the possibility of getting “trapped” in a local maximum point on the curve. Instead the result should always be within the vicinity of the absolute maximum point.

99 MPP’s were calculated from the data received during testing. In order to simplify the system, the duty cycle was scaled as if the battery voltage was 13V. The duty cycle produced by the ANFIS can now be scaled in turn to suit the current battery voltage at that time.

Update of control system parameters on computer

The data from Appendix A was fed into the ANFIS learning program on a computer and a list of new optimized parameters was created.

The parameters are basically adjusted for an individual duty cycle (output), until the desired duty cycle is produced. This is done in turn for each data set.

The duration of the learning algorithm is only about 12min, which is more than adequate for the scope of this project.

Update of control system parameters on microprocessor

After the new parameters are created, they are manually programmed onto the microprocessor. This process is a bit tedious, but with the stability shown in the control system it is not necessary to update the parameters the whole time.

Control system tests

Several tests were conducted to validate the MPPT control system and are discussed in detail in the next few sections.

New data and control system improvements

Especially in the beginning of the project the control system was very unstable or even had troubles converging to a new set of parameters. Continuous improvement was required to solve some of the problems picked up in calculations and hardware.

6.2.1 MPPT Experiment 1: Static basic test

The first step was to get a solid test foundation to which the other experiments could be referenced to. It was decided that the fewer system variables present for this test, the better.

The following external conditions were chosen:

- Vehicle must be stationary
- Solar panel must not be effected by shadows
- Solar panel first heats up a bit in the sun to avoid a big temperature jump
- No cloud cover
- Little or no wind
- Must be early in the day when it is not to hot
- The vehicle is jacked up in order for the motor to run while the vehicle is stationary. This is important to try and keep the battery voltage more constant. However due to the short duration of this test it was found to have negligible effects on the results.

The ANFIS system selects the assumed MPP and this information along with the IV curve information is sent through to the computer. The curve in turn is then analysed on the computer and the actual MPP is calculated.

Figure 6-5 is the power curve of the basic test at 42°C cell temperature and 49,100lx:

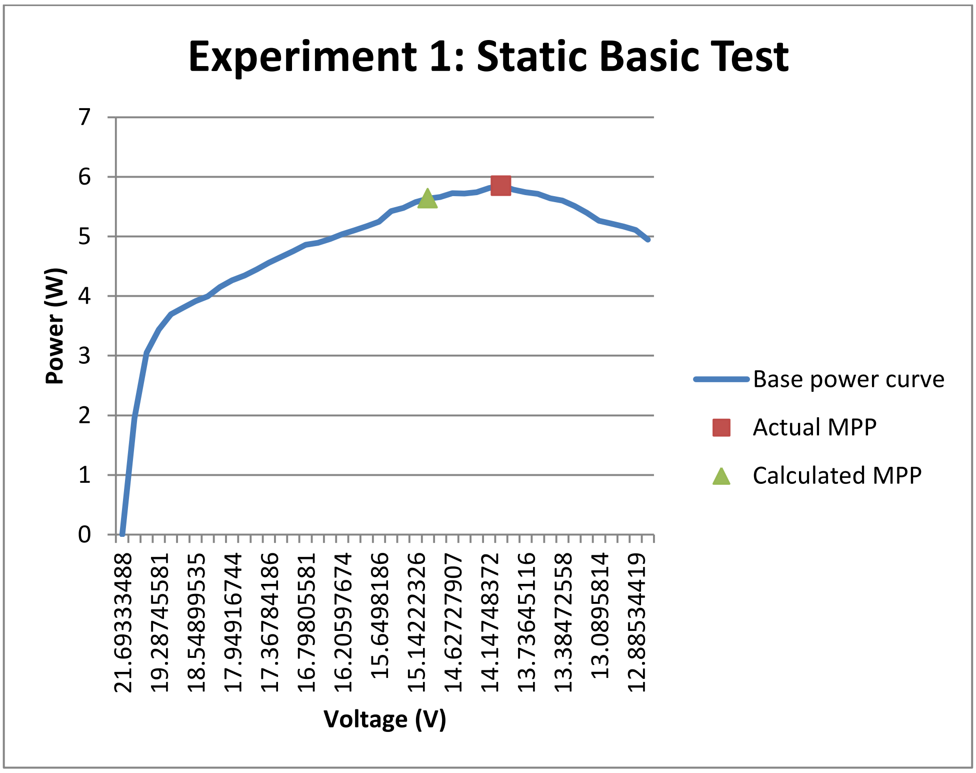


Figure 6-5: Base experimental power curve

The actual MPP was 5.85W at 14.00V. The output of the ANFIS control system was 0.87 (duty cycle). The battery voltage at this stage was 13V.

$$\frac{13}{0.87} = 14.94V$$

This amounts to 5.64W which is due to an error of:

$$\frac{14.94 - 14.00}{14.00} = 6.7\%$$

The error rate was very small and the experiment was marked as successful.

6.2.2 MPPT Experiment 2: Dynamic basic test

In order to investigate the effects of vehicle movement on the base results it was decided to keep the conditions of Experiment 2 the same as far as possible as that of Experiment 1, except for the difference the vehicle shall be in motion.

The test conditions for Experiment 2 are listed below:

- Vehicle must be in motion, but with the panel orientation constant and as close as possible to that of Experiment 1.
- Solar panel must not be effected by shadows
- Solar panel first heats up a bit in the sun to avoid a big temperature jump
- No cloud cover
- Little or no wind
- Must be early in the day when it is not to hot

Figure 6-6 is the power curve of the test result, with the measured panel voltage dropping to 42.8°C and the sunlight intensity averaging at 47,342lx.

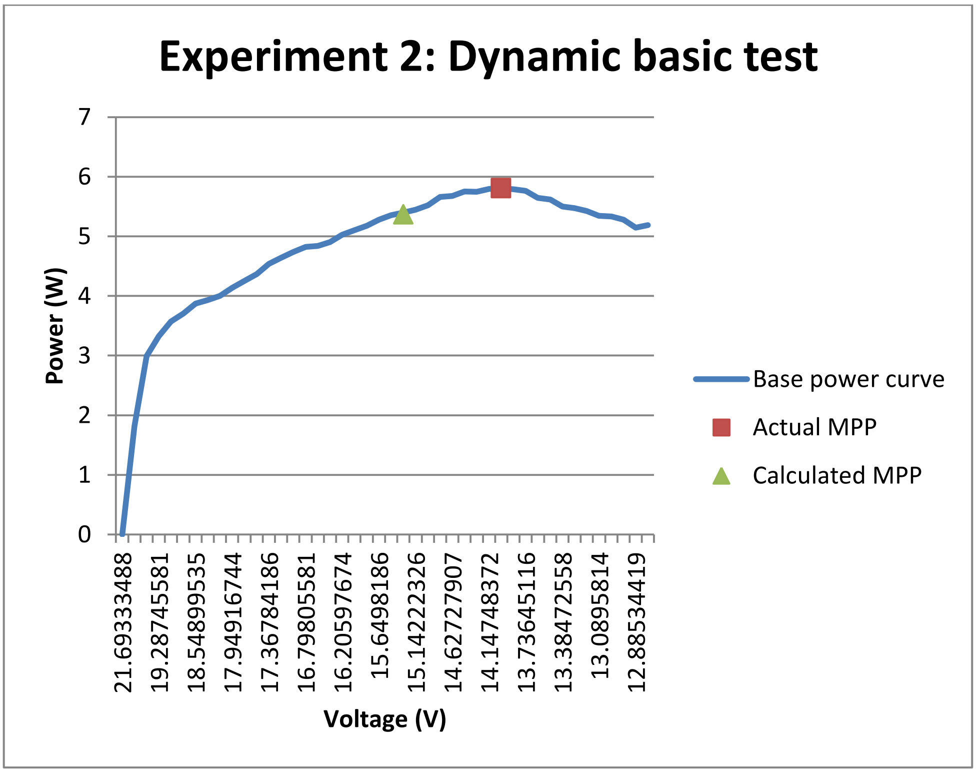


Figure 6-6: Base experimental power curve of while vehicle is in motion

The slight increase in panel temperature was against expectations due to the extra air movement of the vehicle in motion. This does however mean that the temperature increase rate of the panel due to baking in the sun might have just been slowed down. The basic conclusion is though that for the scope of this research the temperature effects due to the vehicle moving can be seen as negligible.

The drop in sunlight intensity could be attributed to the panel moving in terms of orientation, even though a flat surface was chosen to try and minimize this effect.

The actual MPP was located at 5.80W with the panel voltage being 13.99V. The actual MPP did not really move much due to the temperature of the panel temperature not changing much. The drop in power is due to the lower light intensity.

6.2.3 MPPT Experiment 3: Effect of sunlight intensity on MPPT accuracy

In order to investigate the accuracy of the MPPT at a more or less constant panel temperature, a fan was used to prevent the panel from heating up to much.

The following conditions were set for this test:

- Vehicle should be stationary with only manual movements as required to orientate the PV panel, which is detached.
- Solar panel must not be affected by shadows.
- Solar panel first heats up a bit in the sun to avoid a big temperature jump.
- No cloud cover.
- The vehicle is jacked up in order for the motor to run while the vehicle is stationary. This is important to try and keep the battery voltage more constant.

The panel orientation was angled during the experiment in order to create a light intensity range of 11,330lx to 95,129lx, while keeping the temperature relatively constant. The angle of orientation was changed after each data set was received on the vehicle.8 such sets were sourced.

The temperature was held constant at 49.3°C.

Table 6-2 details the voltage of actual MPP versus the calculated MPP for every data set.

Light Intensity	Actual MPP	Controller MPP	Error
11,330lx	13.12V	12.8V	2.4%
27,367lx	13.15V	14.5V	10.3%
41,466lx	13.17V	14.7V	11.6%
56,343lx	13.6V	14.8V	8.8%
61,470lx	13.61V	15.0V	10.2%
75,200lx	13.94V	15.3V	9.8%
89,034lx	14.13V	15.4V	9.0%
95,129lx	14.28V	15.7V	9.9%

Table 6-2: Results of tests at constant temperature

Even though the estimated MPP's are sometimes over and sometimes under the actual MPP values, the errors produced were regarded as rather small and the control system is seen as successful with regards to handling changes in light intensity.

6.2.4 MPPT Experiment 4: Effect of PV panel temperature on MPPT accuracy

The following conditions were set for this test:

- Vehicle should be stationary to ensure stable panel orientation
- Solar panel must not be effected by shadows
- Solar panel first heats up a bit in the sun to avoid a big temperature jump
- Slight cloud cover is permissible if constant
- The vehicle is jacked up in order for the motor to run while the vehicle is stationary. This is important to try and keep the battery voltage more constant.

The panel was put in a cool place to register a temperature of 13.2°C. It was consequently put on the vehicle and data was sampled as soon as the panel was mounted on the vehicle. The temperature range during the data sampling was 21.2°C to 32.9°C. A cool day was chosen so that the temperature of the panel does not increase to fast (during a measurement set). This did however decrease the maximum temperature limit of this test.

The light intensity averaged at 78,589lx.

Table 6-3 details the voltage of actual MPP versus the calculated MPP for every data set.

Panel Temperature	Actual MPP	Controller MPP	Error
21.2°C	17.39V	17.63V	7.6%
28.0°C	16.48V	16.92V	5.7%
32.9°C	15.9V	16.71V	6.1%

Table 6-3: Results of tests at constant light intensity

The control system is accurate under 10% with the effect of a changing panel temperature.

6.2.5 MPPT Experiment 5: Effect of shadow flicker

In order to calculate the system stability during shadow flicker, the following assumptions were made:

- Shadows that are only present for less than 2 second are not worth changing the MPP for due to transient electrical effects. This is, however, relative to large time periods vacant of shadows.

- Shadows that are cyclic, 3 seconds present and 3 seconds removed should be worth tracking on average.

The following conditions were set for this test:

- Vehicle should be stationary to enable manual shadow flicker
- Solar panel first heats up a bit in the sun to avoid a big temperature jump
- Slight cloud cover is permissible if constant
- The vehicle is jacked up in order for the motor to run while the vehicle is stationary. This is important to try and keep the battery voltage more constant.

A piece of thin cloth suspended between two light wooden poles was used to create shadows that fully cover the solar panels. The material was chosen to still allow the panel to be exposed to decent light intensities. This cyclic shadow flicker was repeated 20 times.

The two intermittent light intensities were an average of 34,232lx and 94,220lx.

The MPPT was able to track fast enough in order to create an average error rate of 10.9% with the shadow present and 9.7% with the shadow removed.

The average temperature of the PV panels was 42.1°C and was not significantly affected by the shadows.

6.2.6 MPPT Experiment 6: Effect of partial shadows

The solar vehicle has two PV panels connected in parallel. If only part of one panel is shadowed, then the whole of that panel is effect.

On each panel there is a light sensor installed.

If one panel is cast in shadow, it does not affect the performance of the other panel.

The following was decided with regards to adapting the MPPT to partial shadows:

- A shadow is classified in this instance as a difference in light intensity between the two light sensors.
- A significant difference was taken as 10,000lx.
- The MPPT should always track the panel with the highest light intensity.

Basic tests were performed to see whether the MPPT responds to this condition. The system responded well, however the obvious benefit of having more than two sensors were evident as slight shadowing of a few series cells had large effects on the panel’s performance.

6.3 Summary and Conclusions of MPPT Results

The experiments were all deemed as successful and the following points were noted:

- The time response of the MPPT was within 1 second. More expensive equipment should be able to detail more accurately what the exact time response is, however 1 second was considered successfully fast.
- More light intensity sensors are required if the effect of partial shadows are to be controlled more accurately. It was felt that within the scope of this research the effects of partial shadowing were to some extent tested and serves as a proof of concept for more sophisticated systems.
- The total umbrella average accuracy of the MPPT is 90%. With more accurate sensors and more data available this figure can be improved on even more. However, if a hybrid system of P&O and ANFIS algorithms are used, then very accurate results can be obtained, along with the response advantages of the ANFIS control system.

Chapter 7 - Summary

7.1 Conclusions

The importance of having a MPPT in a solar vehicle system was clearly noted as the various results under different environmental conditions would normally not be possible. This is because the MPPT prevents the solar panel from being pulled to the same voltage level as that of the battery and regulates at what voltage the panel should is operated at. This basic but very important aspect was achieved during testing and therefore hardware used can be classified as that of a functioning MPPT.

All of the project objectives for this MPPT were met:

- As “smart” energy strategy was created to harvest the maximum power from the solar panels. This was achieved by being able to track the MPP of the solar panel under various external conditions. The tracking algorithm is based on a Neuro-Fuzzy logic control system, termed ANFIS.
- This ANFIS controller was implemented on the MPPT and the learning algorithm for it was done offline on a computer. Except for this creating more work during testing, it is still a good implementation and serves as a proof of concept for a more complete system where all the software is run on the microcontroller and data is wirelessly logged on a computer.
- The MPPT adapted well to shadow flicker due to the use of artificial intelligence rather than manually tracking the MPP. The desired MPP is calculated and immediately implemented. a response time of 1 second was successfully achieved for continuous shadow flicker.
- The MPPT was proven rather accurate with changes in panel temperature, which has by far the biggest influence on the MPP.
- Changes in the light intensity had a small effect on the MPP and the MPPT was able to perform well.

Within the scope of this research this project is deemed successful and future possibilities exists to further develop and improve this type of MPPT and expand the scope of this field.

7.2 Recommendations

For the purpose of future research and development, the following observations were made:

- It is possible to improve the accuracy of the MPPT control algorithm significantly, through the hybrid combination of P&O and ANFIS algorithms. The idea is to have a control system that can react quickly to changes in solar irradiance and temperature, after which it can implement a normal P&O tracking algorithm to track the MPP. This type of system will also still prevent the tracking of a false MPP around a local maximum point in a power curve.

- The control system can be implemented on a full scale vehicle, able to transport humans. The effectiveness of the control system can hence be tested at larger power capacity systems.

- A further direction of research would be the detecting of a panel that malfunctions and the effective management of such an occurrence. For a simple two panel system in parallel this effect is easy to detect by means of individual panel monitoring. This problem does however get more complex when many panels are connected per MPPT.

Chapter 8 Bibliography

[1]. S.a. (2008, August 21). “Introduction: Solar radiation and irradiance”. [Online]. Available: https://www.uoguelph.ca/plant/courses/pbio-3110/lectures/lec05_08_000.pdf. [7 November 2015]

[2]. S.a. (2013, May). Solar power by country. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Solar_power_by_country. [12 May 2013] (Solar power by country, Wikipedia)

[3]. Lorenz, P., Pinner, D., Seitz, T. “The economics of solar power,” in *The McKinsey Quarterly*. June 2008, p. 1. (Lorenz *et al*, 2008: 1)

[4]. S.a. (2015, December 1). Artificial intelligence. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [1 December 2015]

[5]. S-kei. (2012, Oct.). GlobalREPowerCapacity-exHydro-Eng.png. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/File:GlobalREPowerCapacity-exHydro-Eng.png. [13 May 2013]. 21 October 2012. (S-kei, 2012)

[6]. S.a. (2013, May). Solar vehicle. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Solar_vehicle. [13 May 2013] (Solar vehicle, Wikipedia)

[7]. Available: https://en.wikipedia.org/wiki/File:PV_solar_roof_mount_and_rack.jpg. [21 October 2015]

[8]. S.a. (2015, October 22). Solar Panel. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Solar_panel. [22 October 2015]

[9]. Moyer, E. (2011, May).Solar Photovoltaics. GEOS24705. [Online]. Available: http://geosci.uchicago.edu/~moyer/GEOS24705/2011/Notes/SolarPhysics.pdf. [28 October 2015]

[10]. Freeman, D. (2010, November). Introduction to Photovoltaic Systems Maximum Power Point Tracking. Texas Instruments. Application Report: SLVA446. [Online]. Available: http://www.ti.com/lit/an/slva446/slva446.pdf. [28 October 2015]

[11]. [Salah, C.B., Ouali, M. (2010, July 10). “Comparison of fuzzy logic and neural network maximum power point tracker for PV systems”. ELSEVIER: Electronic Power Systems Research, Vol. 81 (2011) pp. 43 – 50.

[12]. Letting, L. K., Munda, J. L., Haman, Y. (2011). “Optimization of Fuzzy Logic Controller Design for Maximum Power Point Tracking in Photovoltaic Systems”. K. Gopalakrishnan et al. (Eds.): Soft Comput. in Green & Renew. Ener. Sys. STUDFUZZ 269, pp. 233 -260. Springer-Verlag Berlin Heidelberg.

[13]. S.a. (2015, December 11). Artificial intelligence. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [11 December 2015]

[14]. Conger, C. How can solar panels power a car? How Stuff Works. [Online]. Available: http://auto.howstuffworks.com/fuel-efficiency/vehicles/solar-cars.htm. [17 October 2014]

[15]. Whitwam, R. (2014, January). Ford’s new solar-powered hybrid car can charge up without plugging in. Extreme Tech. [Online]. Avalable: http://www.extremetech.com/extreme/173863-fords-new-solar-powered-hybrid-car-can-charge-up-without-plugging-in. [17 October 2014]

[16]. S.a. (2014, October). Multijunction photovoltaic cell. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Multijunction_photovoltaic_cell. [17 October 2014]

[17]. S.a. C-Max Specifications. Ford. [Online]. Available: http://www.ford.com/cars/cmax/specifications/view-all/. [18 October 2014]

[18]. S.a. Li-Ion vs. Lead Acid. Ultra Life. [Online]. Available: http://www.ultralifecorporation.com/download/275/. [18 October 2014]

[19]. S.a. (2014, August). Polycrystalline silicone photovoltaics. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Polycrystalline_silicon_photovoltaics. [19 October 2014]

[20]. S.a. (2008, August).Suntech Power STP005-12/Db, STP010-12/Kb, STP020-12/Cb. IBC SOLAR.

[21]. S.a. (2008, August). Solar Modules – Polycrystalline, Suntech Power STP005-12/Db, STP010-12/Kb, STP020-12/Cb. IBC Solar. Available: http://www.fibat.ro/energie_neconv/panouri_ibc_solar/5.pdf. [13 April 2015]

[22]. Wildi, T. (2006). Electrical Machines, Drives, and Power Systems. Sixth Edition. ISBN: 0-13-196918-8. pp. 527 – 530.

[23]. S.a. (2001, November 29). DC-DC Converter Tutorial. Maxim Integrated Products Inc. [Online]. Available: http://www.maximintegrated.com/en/app-notes/index.mvp/id/2031. [22 October 2014]

[24]. S.a. (2008, February 13). Buck Operating. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Buck_converter#mediaviewer/File:Buck_operating.svg. [29 October 2014]

[25]. S.a. (2013, May). AN-2124 Power Circuit Design for SolarMagic™ SM3320. SNOSB84C. Texas Instruments Inc. [Online]. Available: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=snosb84&fileType=pdf. [23 October 2014]

[26]. Neaman, D.A. (2007). Microelectronics: Circuit Analysis and Design. Third Edition. ISBN: 007-125443-9. p. 1092.

[27]. Charais, J. (2010). Maximum Power Solar Converter. AN1211. Microchip Technology Inc. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/01211B.pdf. [29 October 2014]

[28]. Lund, M.W. (2015, April 6). Wire Gauge and Current Limits Including Skin Depth and Strength. Power Stream. [Online]. Available: http://www.engineeringtoolbox.com/wire-gauges-d_419.html. [15 April 2015]

[29]. S.a. (1992). ARRL Handbook. Chapter 2. American Radio Relay League.

[30]. S.a. (2012). *PIC24HJ32GP302/304, PIC25HJ64GPX02/X04 and PIC24HJ128GPX02/X04*. Microchip Technology Incorporated. ISBN: 978-1-62076-238-7.

[31]. Balttenberger, K. (2015). Standard Resistor Values. RF Cafe. [Online]. Available: http://www.rfcafe.com/references/electrical/resistor-values.htm. [23 April 2015]

[32]. Iskender, I. (2005, January). "A fuzzy logic controlled power electronics system for maximum power point detection of a solar energy panel". COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, Vol. 24 Iss: 4 pp. 1164 – 1179.

[33]. Gopal, M. (2004). Digital Control and State Variable Methods. Conventional and Neuro- Fuzzy Control Systems, second edition. ISBN: 0-07-048302-7. pp. 718 – 851.

[34]. Jang, J. R. (1995, March). “Neuro-Fuzzy Modelling and Control”. Proceedings of the IEEE, Vol. 83, No. 3 (1995), pp. 378 – 406.

[35]. Jankuloski, B. (2009, August 31). Fuzzy crisp.svg. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/File:Fuzzy_crisp.svg. [31 January 2014]

[36]. Surles, W., Baum, J., Watrous, A., Johnson, S., Horanyi, E., Zarske, M. S. (2014, December 31). “Photovoltaic Efficiency: The Temperature Effect. Fundamentals Article”. Teach Engineering. [Online]. Available: https://www.teachengineering.org/collection/cub_/lessons/cub_pveff/Attachments/cub_pv eff_lesson02_fundamentalsarticle_v6_tedl_dwc.pdf. [1 January 2015]

[37]. S.a. (2014, January). Electricity generation. Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Electricity_generation. [07January 2013]

[38]. Stergiou, C., Siganos, D. “Neural Networks”. Imperial College London. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#A simple neuron. [9 January 2015]

[39]. Jang, J.R. (1992, September). “Self-Learning Fuzzy Controllers Based on Temporal Back Propagation”. IEEE Transactions on Neural Networks, Vol.3 No.5 (September 1992), pp. 714-723.

[40]. Villalva, M.G., Gazoli, J.R., Filho, E.R. (2009, May 5). “Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays”. IEEE Transactions on Power Electronics, Vol.24 No.5 (2009), pp. 1198 – 1208.

[41]. Rizzo, G. (2010). “Automotive Applications of Solar Energy”. IFAC Symposium Advances in Automotive Control. pp. 1 – 12.

[42]. Buchmann, I. (2015). “Charge methods – BU-403: Charging Lead Acid”. Battery University. [Online]. Available: http://batteryuniversity.com/learn/article/charging_the_lead_acid_battery. [16 January 2015]

[43]. Lemburg, M.A. (2014, December 19). “The Python Wiki”. Python™. [Online]. Available: https://wiki.python.org/moin/FrontPage. [16 January 2015]

[44]. S.a. (2015, October 18). Backpropagation. Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation. [30 October 2015]

[45]. Pallones, M. (2011, February 25). Analogue to digital conversion example code. Microchip. [Online]. Available: http://www.microchip.com/search/searchapp/searchhome.aspx?id=2&q=CE357. [13 December 2015]

Appendix A: Data for Learning Algorithm

Temperature (°C)	Temperature (K)	Light Intensity (lx)	Solar Irradiation (W/m^2)	MPP	Maximum Power	Duty Cycle
20.30	293.30	45809.00	361.89	16.93	6.35	0.77
21.20	294.20	34233.00	270.44	16.34	4.35	0.80
21.20	294.20	83045.00	656.06	17.48	12.78	0.74
21.30	294.30	97656.00	771.48	17.66	15.54	0.74
21.60	294.60	12015.00	94.92	13.20	0.92	0.98
22.50	295.50	101234.00	799.75	17.60	15.95	0.74
22.60	295.60	56211.00	444.07	16.80	8.12	0.77
22.70	295.70	23886.00	188.70	15.43	2.68	0.84
23.00	296.00	109345.00	863.83	17.71	17.32	0.73
23.20	296.20	72049.00	569.19	17.20	10.71	0.76
23.30	296.30	62054.00	490.23	16.93	8.96	0.77
23.90	296.90	82302.00	650.19	17.05	12.45	0.76
24.20	297.20	93321.00	737.24	17.24	14.31	0.75
24.50	297.50	12388.00	97.87	13.17	0.97	0.99
24.60	297.60	22543.00	178.09	15.09	2.51	0.86
24.60	297.60	51039.00	403.21	16.56	7.21	0.79
24.70	297.70	102348.00	808.55	17.32	15.81	0.75
24.80	297.80	64203.00	507.20	16.78	9.31	0.77
25.10	298.10	70012.00	553.09	17.02	10.41	0.76
25.30	298.30	39956.00	315.65	15.97	5.71	0.81
25.30	298.30	119382.00	943.12	17.51	18.59	0.74
25.60	298.60	48022.00	379.37	16.13	6.35	0.81
28.60	301.60	12560.00	99.22	13.16	0.89	0.99
28.90	301.90	33060.00	261.17	15.21	4.12	0.85

29.00	302.00	73222.00	578.45	16.40	10.51	0.79
29.10	302.10	23857.00	188.47	14.65	2.61	0.89
29.30	302.30	60012.00	474.09	16.21	8.26	0.80
29.50	302.50	53394.00	421.81	16.01	6.92	0.81
29.50	302.50	81993.00	647.74	16.48	11.83	0.79
30.00	303.00	97334.00	768.94	16.49	14.31	0.79
30.10	303.10	42933.00	339.17	15.46	5.51	0.84
30.90	303.90	118443.00	935.70	16.71	17.72	0.78
31.10	304.10	105102.00	830.31	16.58	15.79	0.78
34.60	307.60	80043.00	632.34	15.81	11.63	0.82
34.80	307.80	72002.00	568.82	15.61	9.62	0.83
34.90	307.90	55405.00	437.70	15.22	7.25	0.85
35.00	308.00	62254.00	491.81	15.43	8.36	0.84
35.00	308.00	91255.00	720.91	15.88	12.96	0.82
35.10	308.10	102984.00	813.57	15.97	13.95	0.81
35.10	308.10	116334.00	919.04	16.19	16.92	0.80
35.40	308.40	12699.00	100.32	13.17	0.99	0.99
35.40	308.40	41102.00	324.71	14.88	5.02	0.87
35.50	308.50	26500.00	209.35	14.13	2.89	0.92
35.50	308.50	33209.00	262.35	14.59	3.92	0.89
39.50	312.50	12734.00	100.60	13.21	0.91	0.98
39.50	312.50	21122.00	166.86	13.19	2.22	0.99
39.70	312.70	64330.00	508.21	14.79	8.31	0.88
39.80	312.80	93325.00	737.27	15.47	12.59	0.84
40.10	313.10	73599.00	581.43	15.06	9.84	0.86
40.10	313.10	119745.00	945.99	15.61	16.71	0.83
40.40	313.40	82824.00	654.31	15.23	11.05	0.85
40.60	313.60	53266.00	420.80	14.56	6.62	0.89
40.70	313.70	40013.00	316.10	14.16	4.69	0.92

40.90	313.90	100552.00	794.36	15.24	13.65	0.85
41.20	314.20	36565.00	288.86	13.93	4.12	0.93
44.60	317.60	56225.00	444.18	14.18	6.71	0.92
45.20	318.20	34125.00	269.59	13.36	3.58	0.97
45.30	318.30	12464.00	98.47	13.19	0.69	0.99
45.30	318.30	24452.00	193.17	13.18	2.29	0.99
45.50	318.50	42263.00	333.88	13.95	4.68	0.93
45.50	318.50	105841.00	836.14	14.87	14.36	0.87
45.60	318.60	117443.00	927.80	14.88	15.56	0.87
45.80	318.80	91430.00	722.30	14.69	11.81	0.88
46.20	319.20	84350.00	666.37	14.47	10.63	0.90
46.30	319.30	62014.00	489.91	14.25	7.45	0.91
46.30	319.30	70123.00	553.97	14.36	8.69	0.91
48.90	321.90	28554.00	225.58	13.20	2.78	0.98
49.60	322.60	36465.00	288.07	13.17	3.91	0.99
49.80	322.80	11986.00	94.69	13.19	0.62	0.99
49.90	322.90	103758.00	819.69	14.31	13.10	0.91
50.00	323.00	41123.00	324.87	13.19	4.32	0.99
50.20	323.20	115312.00	910.96	14.30	14.69	0.91
50.30	323.30	54351.00	429.37	13.56	6.20	0.96
50.60	323.60	61315.00	484.39	13.59	7.10	0.96
50.80	323.80	91315.00	721.39	14.15	11.32	0.92
51.00	324.00	72864.00	575.63	13.79	8.65	0.94
51.10	324.10	86584.00	684.01	13.92	10.62	0.93
54.70	327.70	94012.00	742.69	13.74	11.28	0.95
54.80	327.80	81354.00	642.70	13.56	9.62	0.96
54.80	327.80	101489.00	801.76	13.81	12.26	0.94
54.90	327.90	62698.00	495.31	13.18	7.00	0.99
55.00	328.00	12327.00	97.38	13.17	0.38	0.99

55.10	328.10	20198.00	159.56	13.23	1.65	0.98
55.20	328.20	33546.00	265.01	13.22	2.90	0.98
55.20	328.20	50123.00	395.97	13.22	5.54	0.98
55.30	328.30	46465.00	367.07	13.19	4.92	0.99
55.30	328.30	72493.00	572.69	13.39	8.37	0.97
55.90	328.90	120564.00	952.46	13.71	14.80	0.95
57.80	330.80	105315.00	831.99	13.48	12.52	0.96
57.90	330.90	72495.00	572.71	13.19	8.15	0.99
58.30	331.30	90134.00	712.06	13.17	10.45	0.99
58.50	331.50	117473.00	928.04	13.56	14.12	0.96
58.60	331.60	51249.00	404.87	13.18	5.33	0.99
58.60	331.60	84610.00	668.42	13.18	9.71	0.99
58.80	331.80	67415.00	532.58	13.17	7.40	0.99
59.60	332.60	25048.00	197.88	13.18	1.76	0.99
60.20	333.20	44441.00	351.08	13.20	4.29	0.98
60.30	333.30	31485.00	248.73	13.20	2.58	0.98
60.40	333.40	12485.00	98.63	13.21	0.07	0.98

Appendix B: MPPT Simulation Software

```
#First hidden layer of ANFIS network
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++
#Fuzzy set functions: Temperature fuzzy set (1)
def Tpv_Low(T): #Fuzzy subset LOW (1.1)
    uTLow = math.exp(-1*math.pow(((T-cTL)/sTL),2))
    return uTLow

def Tpv_Moderate(T): #Fuzzy subset MODERATE (1.2)
    uTModerate = math.exp(-1*math.pow(((T-cTM)/sTM),2))
    return uTModerate

def Tpv_High(T): #Fuzzy subset HIGH (1.3)
    uTHigh = math.exp(-1*math.pow(((T-cTH)/cTH),2))
    return uTHigh

def Tpv_VHigh(T): #Fuzzy subset VHIGH (1.4)
    uTVHigh = math.exp(-1*math.pow(((T-cTV)/sTV),2))
    return uTVHigh

def Tpv_XHigh(T): #Fuzzy subset XHIGH (1.5)
    uTXHigh = math.exp(-1*math.pow(((T-cTX)/sTX),2))
    return uTXHigh

#Fuzzy set functions: Solar irrdiation fuzzy set (2)
def Gsolar_XLow(G): #Fuzzy subset XLOW (2.1)
    uGXLow = math.exp(-1*math.pow(((G-cGX)/sGX),2))
    return uGXLow

def Gsolar_VLow(G): #Fuzzy subset VLOW (2.2)
    uGVLow = math.exp(-1*math.pow(((G-cGV)/sGV),2))
    return uGVLow

def Gsolar_Low(G): #Fuzzy subset LOW (2.3)
    uGLow = math.exp(-1*math.pow(((G-cGL)/sGL),2))
    return uGLow
```

```
def Gsolar_Moderate(G): #Fuzzy subset MODERATE (2.4)
    uGModerate = math.exp(-1*math.pow(((G-cGM)/sGM),2))
    return uGModerate

def Gsolar_High(G): #Fuzzy subset HIGH (2.5)
    uGHigh = math.exp(-1*math.pow(((G-cGH)/sGH),2))
    return uGHigh

#Second hidden layer of ANFIS network
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++
#Determine firing strength of each rule
def rule1():
    w1 = wT1*(Tpv_Low(Tin)) + wG1*(Gsolar_XLow(Gin))
    return w1

def rule2():
    w2 = wT2*(Tpv_Low(Tin)) + wG2*(Gsolar_VLow(Gin))
    return w2

def rule3():
    w3 = wT3*(Tpv_Low(Tin)) + wG3*(Gsolar_Low(Gin))
    return w3

def rule4():
    w4 = wT4*(Tpv_Low(Tin)) + wG4*(Gsolar_Moderate(Gin))
    return w4

def rule5():
    w5 = wT5*(Tpv_Low(Tin)) + wG5*(Gsolar_High(Gin))
    return w5

def rule6():
    w6 = wT6*(Tpv_Moderate(Tin)) + wG6*(Gsolar_XLow(Gin))
    return w6

def rule7():
    w7 = wT7*(Tpv_Moderate(Tin)) + wG7*(Gsolar_VLow(Gin))
    return w7
```

```
def rule8():
    w8 = wT8*(Tpv_Moderate(Tin)) + wG8*(Gsolar_Low(Gin))
    return w8

def rule9():
    w9 = wT9*(Tpv_Moderate(Tin)) + wG9*(Gsolar_Moderate(Gin))
    return w9

def rule10():
    w10 = wT10*(Tpv_Moderate(Tin)) + wG10*(Gsolar_High(Gin))
    return w10

def rule11():
    w11 = wT11*(Tpv_High(Tin)) + wG11*(Gsolar_XLow(Gin))
    return w11

def rule12():
    w12 = wT12*(Tpv_High(Tin)) + wG12*(Gsolar_VLow(Gin))
    return w12

def rule13():
    w13 = wT13*(Tpv_High(Tin)) + wG13*(Gsolar_Low(Gin))
    return w13

def rule14():
    w14 = wT14*(Tpv_High(Tin)) + wG14*(Gsolar_Moderate(Gin))
    return w14

def rule15():
    r15 = wT15*(Tpv_High(Tin)) + wG15*(Gsolar_High(Gin))
    return r15

def rule16():
    w16 = wT16*(Tpv_VHigh(Tin)) + wG16*(Gsolar_XLow(Gin))
    return w16

def rule17():
    w17 = wT17*(Tpv_VHigh(Tin)) + wG17*(Gsolar_VLow(Gin))
    return w17
```

```
def rule18():
    w18 = wT18*(Tpv_VHigh(Tin)) + wG18*(Gsolar_Low(Gin))
    return w18

def rule19():
    w19 = wT19*(Tpv_VHigh(Tin)) + wG19*(Gsolar_Moderate(Gin))
    return w19

def rule20():
    w20 = wT20*(Tpv_VHigh(Tin)) + wG20*(Gsolar_High(Gin))
    return w20

def rule21():
    w21 = wT21*(Tpv_XHigh(Tin)) + wG21*(Gsolar_XLow(Gin))
    return w21

def rule22():
    w22 = wT22*(Tpv_XHigh(Tin)) + wG22*(Gsolar_VLow(Gin))
    return w22

def rule23():
    w23 = wT23*(Tpv_XHigh(Tin)) + wG23*(Gsolar_Low(Gin))
    return w23

def rule24():
    w24 = wT24*(Tpv_XHigh(Tin)) + wG24*(Gsolar_Moderate(Gin))
    return w24

def rule25():
    w25 = wT25*(Tpv_XHigh(Tin)) + wG25*(Gsolar_High(Gin))
    return w25

def BatteryComp():
    global fbat
    fbat = 1/(1 + math.exp(-1*Vbat))
    return wTbat*fbat

#Thrird and fourth hidden layers of ANFIS network + output layer
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++
def ANFIS(T,G):
    global
res1,res2,res3,res4,res5,res6,res7,res8,res9,res10,res11,res12,res13,res14,res15,res16,res17,res18,
res19,res20,res21,res22,res23,res24,res25,res26
    #Linear functions for third hidden layer nodes
    f1 = p1*T + q1*G + r1
    f2 = p2*T + q2*G + r2
    f3 = p3*T + q3*G + r3
    f4 = p4*T + q4*G + r4
    f5 = p5*T + q5*G + r5
    f6 = p6*T + q6*G + r6
    f7 = p7*T + q7*G + r7
    f8 = p8*T + q8*G + r8
    f9 = p9*T + q9*G + r9
    f10 = p10*T + q10*G + r10
    f11 = p11*T + q11*G + r11
    f12 = p12*T + q12*G + r12
    f13 = p13*T + q13*G + r13
    f14 = p14*T + q14*G + r14
    f15 = p15*T + q15*G + r15
    f16 = p16*T + q16*G + r16
    f17 = p17*T + q17*G + r17
    f18 = p18*T + q18*G + r18
    f19 = p19*T + q19*G + r19
    f20 = p20*T + q20*G + r20
    f21 = p21*T + q21*G + r21
    f22 = p22*T + q22*G + r22
    f23 = p23*T + q23*G + r23
    f24 = p24*T + q24*G + r24
    f25 = p25*T + q25*G + r25

    #Results for first 25 nodes in the third hidden layer
    res1 = f1*rule1()
    res2 = f2*rule2()
    res3 = f3*rule3()
    res4 = f4*rule4()
    res5 = f5*rule5()
    res6 = f6*rule6()
    res7 = f7*rule7()
    res8 = f8*rule8()
    res9 = f9*rule9()
    res10 = f10*rule10()
    res11 = f11*rule11()
    res12 = f12*rule12()
    res13 = f13*rule13()
    res14 = f14*rule14()
    res15 = f15*rule15()
    res16 = f16*rule16()
    res17 = f17*rule17()
    res18 = f18*rule18()
    res19 = f19*rule19()
    res20 = f20*rule20()
    res21 = f21*rule21()
    res22 = f22*rule22()
    res23 = f23*rule23()
    res24 = f24*rule24()
    res25 = f25*rule25()

    #Result of 26th node in third hidden layer
    res26 = rule1() + rule2() + rule3() + rule4() + rule5()\
          + rule6() + rule7() + rule8() + rule9() + rule10()\
          + rule11() + rule12() + rule13() + rule14() + rule15()\
          + rule16() + rule17() + rule18() + rule19() + rule20()\
          + rule21() + rule22() + rule23() + rule24() + rule25()

    #Result of single node in layer 4 (calculating the weighted average and send result to output node)
    #d = BatteryComp()*((res1 + res2 + res3 + res4 + res5 + res6 + res7 + res8 + res9 + res10 + res11
+ res12 + res13 + res14 + res15 + res16 + res17 + res18 + res19\
    #     + res20 + res21 + res23 + res24 + res25)/res26)

    d = ((res1 + res2 + res3 + res4 + res5 + res6 + res7 + res8 + res9 + res10 + res11 + res12 + res13
+ res14 + res15 + res16 + res17 + res18 + res19\
         + res20 + res21 + res23 + res24 + res25)/res26)
```

Appendix C: MPPT Control Software

```
int main(void)
{
        asm volatile (
        "MOV.W #%0, W3\n\t"
        "MOV.B #%1, W0\n\t"
        "MOV.B #%2, W1\n\t"
        "MOV.B #%3, W2\n\t"
        "MOV.B W1, [W3+0]\n\t"
        "MOV.B W2, [W3+0]\n\t"
        "MOV.B W0, [W3+0]"
        : /* no outputs */
        : "U"(OSCCON), "i"(0x2), "i"(0x46), "i"(0x57)
        : /* clobber */ "w0", "w1", "w2", "w3");

        // Setup bits for FRC with PLL, Fp/Fcy = 23MHz
        OSCCONbits.NOSC = 0b001;
        CLKDIVbits.DOZEN = 0b0;
        CLKDIVbits.FRCDIV = 0b000;
        CLKDIVbits.PLLPOST = 0b01;
        CLKDIVbits.PLLPRE = 0b00000;
        PLLFBDbits.PLLDIV = 0b000110000;
        OSCTUNbits.TUN = 0b000000;

        // Configure RA0 for 'Hallo World' LED
        TRISAbits.TRISA0 = 0b0; // Set RA0 as output pin
        PORTA = 0x00;
        LATAbits.LATA0 = 0b1; // Latch RA0

        int a;
        static unsigned int LogTimer1 = 0;
        static unsigned int LogTimer2 = 0;
        static unsigned int LogTimer3 = 0;
        static unsigned int Log1I[0x1CB];
        static unsigned int Log1V[0x1CB];

  IOInit();          //Initialize the I/O ports
  ADCInit();          //Initialize the A/D converter
  DMAInit();          //Initialize DMA
  ADCStart();                              //Turn ADC on
```

```
    OC1Init();
    UARTinit();                                     //Initialize UART
    c3 =0;

    while(1)
    {
        //MPPT_Curve();
        MPPT_Track();
    }
}

void MPPT_Track()
{
        MPPT_Collect(); //Collect inputs

        sTemp = sPOT;
        sTemp = sTemp*0.2; //Convert register value to degrees C
        sLight = sLight/195*0.78;
        sLight2 = sLight2/195*0.78;

        if (sLight < sLight2)
        {
            sLight = sLight2;
        }

        sLight = 0.0079*(-15698.39*powl(sLight,3) + 117170.84*powl(sLight,2) - 161623.89*sLight - 63314.19); //Convert register value to W/m^2
        //sTemp = 250;
        //sLight = 1000;
        //sTemp = 50;
        sVout = sVout/521.90*14.14;

        dutyC2 = dutyC;
        MPPT_Crisp();
        dutyC = dutyC*sVout/13.0;


        if (dutyC > dutyC2 || dutyC > dutyC2)
        {
            while(U1STAbits.TRMT != 0x1);
```

```
                snprintf(c12,15,"%f",dutyC);
                U1TXREG = c12[0];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[1];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[2];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[3];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[4];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[5];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[6];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = c12[7];
                while(U1STAbits.TRMT != 0x1);
                U1TXREG = '\n';
        }
        //dutyC = dutyC*13.0/13.0;
        tempTest = (dutyC-(sVout/21.6))/(1 - (sVout/21.6))*459.0;
        if (tempTest > 459.0)
        {
                tempTest = 459.0;
        }
        //OSCCC = tempTest;

        T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                OC1RS = tempTest;
        T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
}

void MPPT_Collect()
{
        long int del;
        esVout = 0;
        esIout = 0;
        esVin = 0;
        esIin = 0;
        esTemp = 0;
```

```
    esLight = 0;
    esPOT = 0;
    del = 0;

    tempDelay = min_sec;
            RTCCRead();
            if (tempDelay != min_sec)
            {
                tempDelay2++;
                if (tempDelay2 > 10)
                {
                    tempDelay2 = 0;
                }
            }
            while (del <= 10000)
            {
                if (AD1CSSL == 0x10)
                {
                    ADCStop();
                    AD1CSSL = 0x200;
//Select AN9 as sample input
                    //AD1CHS0bits.CH0SA = 0b01001;
//Positive input -> AN9
                    ADCStart();
                    c1 = 0;
                    while (c1<1000)
                    {
                        c1++;
                    }
                    sVout = ADCAvg();
                    esVout = esVout + sVout;
                }

                else if (AD1CSSL == 0x200)
                {
                    ADCStop();
                    AD1CSSL = 0x400;
//Select AN10 as sample input
                    ADCStart();
                    c1 = 0;
```

```
                        while (c1<1000)
                        {
                            c1++;
                        }
                        while(IFS0bits.AD1IF != 1);  //waiting
                        sPOT = ADCAvg();
                        esPOT = esPOT + sPOT;

                        //T2CONbits.TON = 0b0; // Switch off Timer2 for
OC1 setup
                        //OC1RS = sPOT/0x0008;
                        //T2CONbits.TON = 0b1; // Switch on Timer2, OC1
setup completed
                    }

                    else if (AD1CSSL == 0x400)
                    {
                        ADCStop();
                        AD1CSSL = 0x800;
    //Select AN11 as sample input
                        ADCStart();
                        c1 = 0;
                        while (c1<1000)
                        {
                            c1++;
                        }
                        while(IFS0bits.AD1IF != 1);  //waiting
                        sVin = ADCAvg();
                        esVin = esVin + sVin;
                    }

                    else if (AD1CSSL == 0x800)
                    {
                        ADCStop();
                        AD1CSSL = 0x1000;
    //Select AN12 as sample input
                        ADCStart();
                        c1 = 0;
                        while (c1<1000)
                        {
```

```
                        c1++;
                    }
                    while(IFS0bits.AD1IF != 1);  //waiting
                    sIin = ADCAvg();
                    esIin = esIin + sIin;
                }

                else if (AD1CSSL == 0x1000)
                {
                    ADCStop();
                    AD1CSSL = 0x20;
//Select AN5 as sample input
                    ADCStart();
                    c1 = 0;
                    while (c1<1000)
                    {
                        c1++;
                    }
                    while(IFS0bits.AD1IF != 1);  //waiting
                    sLight2 = ADCAvg();
                    esLight2 = esLight2 + sLight2;
                }

                else if (AD1CSSL == 0x20)
                {
                    ADCStop();
                    AD1CSSL = 0x10;
//Select AN4 as sample input
                    ADCStart();
                    c1 = 0;
                    while (c1<1000)
                    {
                        c1++;
                    }
                    while(IFS0bits.AD1IF != 1);  //waiting
                    sLight = ADCAvg();
                    esLight = esLight + sLight;
                }

                if (sPOT>0x0030)
```

```
                    {
                        LATAbits.LATA0 = 0b0; // Unlatch RA0
                    }

                    else
                    {
                        LATAbits.LATA0 = 0b1; // Latch RA0
                    }

                    del = del +1;
                }
                del =0;
                sVout = esVout/10000;
                sIout = esIout/10000;
                sVin = esVin/10000;
                sIin = esIin/10000;
                sLight2 = esLight2/10000;
                sLight = esLight/10000;
                sPOT = esPOT/10000;
                sVout = esVout/10000;

                esVout = 0;
                esIout = 0;
                esVin = 0;
                esIin = 0;
                esTemp = 0;
                esLight = 0;
                esPOT = 0;
}

long double MPPT_Membership_Tlo()
{
    u_Tlo = expl(-1*powl(((sTemp-c_Tlo)/s_Tlo),2));
    return u_Tlo;
}

long double MPPT_Membership_Tmo()
{
    u_Tmo = expl(-1*powl(((sTemp-c_Tmo)/s_Tmo),2));
    return u_Tmo;
```

```
}

long double MPPT_Membership_Thi()
{
            u_Thi = expl(-1*powl(((sTemp-c_Thi)/s_Thi),2));
            return u_Thi;
}

long double MPPT_Membership_Tvh()
{
            u_Tvh = expl(-1*powl(((sTemp-c_Tvh)/s_Tvh),2));
            return u_Tvh;
}

long double MPPT_Membership_Txh()
{
            u_Txh = expl(-1*powl(((sTemp-c_Txh)/s_Txh),2));
            return u_Txh;
}

long double MPPT_Membership_Gxl()
{
            u_Gxl = expl(-1*powl(((sLight-c_Gxl)/s_Gxl),2));
            return u_Gxl;
}

long double MPPT_Membership_Gvl()
{
            u_Gvl = expl(-1*powl(((sLight-c_Gvl)/s_Gvl),2));
            return u_Gvl;
}

long double MPPT_Membership_Glo()
{
            u_Glo = expl(-1*powl(((sLight-c_Glo)/s_Glo),2));
            return u_Glo;
}

long double MPPT_Membership_Gmo()
{
        u_Gmo = expl(-1*powl(((sLight-c_Gmo)/s_Gmo),2));
        return u_Gmo;
}

long double MPPT_Membership_Ghi()
{
        u_Ghi = expl(-1*powl(((sLight-c_Ghi)/s_Ghi),2));
        return u_Ghi;
}

long double MPPT_rule1()
{
        u1 = wT1*(MPPT_Membership_Tlo()) + wG1*(MPPT_Membership_Gxl());
        return u1;
}

long double MPPT_rule2()
{
        u2 = wT2*(MPPT_Membership_Tlo()) + wG2*(MPPT_Membership_Gvl());
        return u2;
}

long double MPPT_rule3()
{
        u3 = wT3*(MPPT_Membership_Tlo()) + wG3*(MPPT_Membership_Glo());
        return u3;
}

long double MPPT_rule4()
{
        u4 = wT4*(MPPT_Membership_Tlo()) + wG4*(MPPT_Membership_Gmo());
        return u4;
}

long double MPPT_rule5()
{
        u5 = wT5*(MPPT_Membership_Tlo()) + wG5*(MPPT_Membership_Ghi());
        return u5;
}
```

```
long double MPPT_rule6()
{
	u6 = wT6*(MPPT_Membership_Tmo()) + wG6*(MPPT_Membership_Gxl());
	return u6;
}

long double MPPT_rule7()
{
	u7 = wT7*(MPPT_Membership_Tmo()) + wG7*(MPPT_Membership_Gvl());
	return u7;
}

long double MPPT_rule8()
{
	u8 = wT8*(MPPT_Membership_Tmo()) + wG8*(MPPT_Membership_Glo());
	return u8;
}

long double MPPT_rule9()
{
	u9 = wT9*(MPPT_Membership_Tmo()) + wG9*(MPPT_Membership_Gmo());
	return u9;
}

long double MPPT_rule10()
{
	u10 = wT10*(MPPT_Membership_Tmo()) + wG10*(MPPT_Membership_Ghi());
	return u10;
}

long double MPPT_rule11()
{
	u11 = wT11*(MPPT_Membership_Thi()) + wG11*(MPPT_Membership_Gxl());
	return u11;
}

long double MPPT_rule12()
{
	u12 = wT12*(MPPT_Membership_Thi()) + wG12*(MPPT_Membership_Gvl());
	return u12;
```

```
}

long double MPPT_rule13()
{
        u13 = wT13*(MPPT_Membership_Thi()) + wG13*(MPPT_Membership_Glo());
        return u13;
}

long double MPPT_rule14()
{
        u14 = wT14*(MPPT_Membership_Thi()) + wG14*(MPPT_Membership_Gmo());
        return u14;
}

long double MPPT_rule15()
{
        u15 = wT15*(MPPT_Membership_Thi()) + wG15*(MPPT_Membership_Ghi());
        return u15;
}

long double MPPT_rule16()
{
        u16 = wT16*(MPPT_Membership_Tvh()) + wG16*(MPPT_Membership_Gxl());
        return u16;
}

long double MPPT_rule17()
{
        u17 = wT17*(MPPT_Membership_Tvh()) + wG17*(MPPT_Membership_Gvl());
        return u17;
}

long double MPPT_rule18()
{
        u18 = wT18*(MPPT_Membership_Tvh()) + wG18*(MPPT_Membership_Glo());
        return u18;
}

long double MPPT_rule19()
{
        u19 = wT19*(MPPT_Membership_Tvh()) + wG19*(MPPT_Membership_Gmo());
        return u19;
}

long double MPPT_rule20()
{
        u20 = wT20*(MPPT_Membership_Tvh()) + wG20*(MPPT_Membership_Ghi());
        return u20;
}

long double MPPT_rule21()
{
        u21 = wT21*(MPPT_Membership_Txh()) + wG21*(MPPT_Membership_Gxl());
        return u21;
}

long double MPPT_rule22()
{
        u22 = wT22*(MPPT_Membership_Txh()) + wG22*(MPPT_Membership_Gvl());
        return u22;
}

long double MPPT_rule23()
{
        u23 = wT23*(MPPT_Membership_Txh()) + wG23*(MPPT_Membership_Glo());
        return u23;
}

long double MPPT_rule24()
{
        u24 = wT24*(MPPT_Membership_Txh()) + wG24*(MPPT_Membership_Gmo());
        return u24;
}

long double MPPT_rule25()
{
        u25 = wT25*(MPPT_Membership_Txh()) + wG25*(MPPT_Membership_Ghi());
        return u25;
}
```

```
void MPPT_Crisp()
{
        uR1 = (p1*sTemp + q1*sLight + r1)*MPPT_rule1();
        uR2 = (p2*sTemp + q2*sLight + r2)*MPPT_rule2();
        uR3 = (p3*sTemp + q3*sLight + r3)*MPPT_rule3();
        uR4 = (p4*sTemp + q4*sLight + r4)*MPPT_rule4();
        uR5 = (p5*sTemp + q5*sLight + r5)*MPPT_rule5();
        uR6 = (p6*sTemp + q6*sLight + r6)*MPPT_rule6();
        uR7 = (p7*sTemp + q7*sLight + r7)*MPPT_rule7();
        uR8 = (p8*sTemp + q8*sLight + r8)*MPPT_rule8();
        uR9 = (p9*sTemp + q9*sLight + r9)*MPPT_rule9();
        uR10 = (p10*sTemp + q10*sLight + r10)*MPPT_rule10();
        uR11 = (p11*sTemp + q11*sLight + r11)*MPPT_rule11();
        uR12 = (p12*sTemp + q12*sLight + r12)*MPPT_rule12();
        uR13 = (p13*sTemp + q13*sLight + r13)*MPPT_rule13();
        uR14 = (p14*sTemp + q14*sLight + r14)*MPPT_rule14();
        uR15 = (p15*sTemp + q15*sLight + r15)*MPPT_rule15();
        uR16 = (p16*sTemp + q16*sLight + r16)*MPPT_rule16();
        uR17 = (p17*sTemp + q17*sLight + r17)*MPPT_rule17();
        uR18 = (p18*sTemp + q18*sLight + r18)*MPPT_rule18();
        uR19 = (p19*sTemp + q19*sLight + r19)*MPPT_rule19();
        uR20 = (p20*sTemp + q20*sLight + r20)*MPPT_rule20();
        uR21 = (p21*sTemp + q21*sLight + r21)*MPPT_rule21();
        uR22 = (p22*sTemp + q22*sLight + r22)*MPPT_rule22();
        uR23 = (p23*sTemp + q23*sLight + r23)*MPPT_rule23();
        uR24 = (p24*sTemp + q24*sLight + r24)*MPPT_rule24();
        uR25 = (p25*sTemp + q25*sLight + r25)*MPPT_rule25();

        uR26 = MPPT_rule1() + MPPT_rule2() + MPPT_rule3() + MPPT_rule4() + MPPT_rule5() +
MPPT_rule6()+ MPPT_rule6() + MPPT_rule7() + MPPT_rule8() + MPPT_rule9() +
                        MPPT_rule10() + MPPT_rule11() + MPPT_rule12() + MPPT_rule13() +
MPPT_rule14() +MPPT_rule15() + MPPT_rule16() + MPPT_rule17() + MPPT_rule18() +
                                MPPT_rule19() + MPPT_rule20() + MPPT_rule21() + MPPT_rule22()
+ MPPT_rule23() + MPPT_rule24() + MPPT_rule25();

        dutyC = (uR1 + uR2+ uR3 + uR4 + uR5 + uR6 + uR7 + uR8 + uR9 + uR10 + uR11 + uR12 +
uR13 + uR14 + uR15 + uR16 + uR17 + uR18 + uR19 + uR20 + uR21 + uR22 + uR23 + uR24 +
uR25)/uR26;
}
```

```
void MPPT_Curve()
{
        OSCCC = 0x0;
        do
        {
                long int del;
                esVout = 0;
                esIout = 0;
                esVin = 0;
                esIin = 0;
                esTemp = 0;
                esLight = 0;
                esPOT = 0;
                del = 0;

                tempDelay = min_sec;
                                RTCCRead();
                                if (tempDelay != min_sec)
                                {
                                        tempDelay2++;
                                        if (tempDelay2 > 10)
                                        {
                                                tempDelay2 = 0;
                                        }
                                }
                                while (del <= 10000)
                                {
                                        if (AD1CSSL == 0x10)
                                        {
                                                ADCStop();
                                                AD1CSSL = 0x200;
        //Select AN9 as sample input
                                                //AD1CHS0bits.CH0SA = 0b01001;
        //Positive input -> AN9
                                                ADCStart();
                                                c1 = 0;
                                                while (c1<1000)
                                                {
                                                        c1++;
                                                }
```

```
                        sVout = ADCAvg();
                        esVout = esVout + sVout;
                    }

                    else if (AD1CSSL == 0x200)
                    {
                        ADCStop();
                        AD1CSSL = 0x400;
    //Select AN10 as sample input
                        ADCStart();
                        c1 = 0;
                        while (c1<1000)
                        {
                            c1++;
                        }
                        while(IFS0bits.AD1IF != 1);  //waiting
                        sPOT = ADCAvg();
                        esPOT = esPOT + sPOT;

                        //T2CONbits.TON = 0b0; // Switch off Timer2 for
OC1 setup
                        //OC1RS = sPOT/0x0008;
                        //T2CONbits.TON = 0b1; // Switch on Timer2, OC1
setup completed
                    }

                    else if (AD1CSSL == 0x400)
                    {
                        ADCStop();
                        AD1CSSL = 0x800;
    //Select AN11 as sample input
                        ADCStart();
                        c1 = 0;
                        while (c1<1000)
                        {
                            c1++;
                        }
                        while(IFS0bits.AD1IF != 1);  //waiting
                        sVin = ADCAvg();
                        esVin = esVin + sVin;
```

```
                }

                else if (AD1CSSL == 0x800)
                {
                    ADCStop();
                    AD1CSSL = 0x1000;
//Select AN12 as sample input
                    ADCStart();
                    c1 = 0;
                    while (c1<1000)
                    {
                        c1++;
                    }
                    while(IFS0bits.AD1IF != 1);  //waiting
                    sIin = ADCAvg();
                    esIin = esIin + sIin;
                }

                else if (AD1CSSL == 0x1000)
                {
                    ADCStop();
                    AD1CSSL = 0x20;
//Select AN5 as sample input
                    ADCStart();
                    c1 = 0;
                    while (c1<1000)
                    {
                        c1++;
                    }
                    while(IFS0bits.AD1IF != 1);  //waiting
                    sLight2 = ADCAvg();
                    esLight2 = esLight2 + sLight2;
                }

                else if (AD1CSSL == 0x20)
                {
                    ADCStop();
                    AD1CSSL = 0x10;
//Select AN4 as sample input
                    ADCStart();
```

```
        c1 = 0;
        while (c1<1000)
        {
            c1++;
        }
        while(IFS0bits.AD1IF != 1);  //waiting
        sLight = ADCAvg();
        esLight = esLight + sLight;
    }

    if (sPOT>0x0030)
    {
        LATAbits.LATA0 = 0b0; // Unlatch RA0
    }

    else
    {
        LATAbits.LATA0 = 0b1; // Latch RA0
    }

    del = del +1;
}
del =0;
sVout = esVout/10000;
sIout = esIout/10000;
sVin = esVin/10000;
sIin = esIin/10000;
sLight2 = esLight2/10000;
sLight = esLight/10000;
sPOT = esPOT/10000;
//if (del == 100)
//{
LogIVcurve();
//}
//if (AD1CSSL == 0x1000)
//{
//  LogIVcurve(sVin,sIin);
//}
sVout = esVout/10000;
```

```
                        esVout = 0;
                        esIout = 0;
                        esVin = 0;
                        esIin = 0;
                        esTemp = 0;
                        esLight = 0;
                        esPOT = 0;
                //LogPVcurve()
                //LogIVPTLt()
        } while (OSCCC < 0x1CB);
}

void LogIVcurve()
{
                //while(IFS0bits.AD1IF != 1);  //waiting
                //a = ADCAvg();
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                OC1RS = OSCCC;
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
                OSCCC = OSCCC + 10;
                if (OSCCC >= 0x1CB)
                {
                        OSCCC = 0x0;
                }

                //T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
                //OC1RS = 0x130;
                //T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed

                while(U1STAbits.TRMT != 0x1);
```

```
//snprintf(c4,50,"%f",sPOT); //0 -> 5.47, bypass resistor -> 5.47
snprintf(c4,15,"%f",sVout); //0 -> 3990, bypass resistor -> 4095
snprintf(c6,15,"%f",sVin); //7 -> 20, bypass resistor -> 4095
snprintf(c7,15,"%f",sIin); // 45 -> 58, bypass resistor -> 4095
snprintf(c8,15,"%f",sLight); //
snprintf(c9,15,"%f",OSCCC);
snprintf(c10,15,"%f",sLight2); //
snprintf(c11,15,"%f",sPOT); //
U1TXREG = c4[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[7];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[8];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[9];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[10];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c4[11];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[2];
while(U1STAbits.TRMT != 0x1);
```

```
U1TXREG = c6[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c6[7];
while(U1STAbits.TRMT != 0x1);
//U1TXREG = c6[8];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c6[9];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c6[10];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c6[11];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c7[7];
while(U1STAbits.TRMT != 0x1);
//U1TXREG = c7[8];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c7[9];
//while(U1STAbits.TRMT != 0x1);
```

```
//U1TXREG = c7[10];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c7[11];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c8[7];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c8[8];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c8[9];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c8[10];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c8[11];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[3];
while(U1STAbits.TRMT != 0x1);
```

```
U1TXREG = c9[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c9[7];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c9[8];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c9[9];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c9[10];
//while(U1STAbits.TRMT != 0x1);
//U1TXREG = c9[11];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[1];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c10[7];
while(U1STAbits.TRMT != 0x1);
U1TXREG = ',';
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[0];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[1];
while(U1STAbits.TRMT != 0x1);
```

```
U1TXREG = c11[2];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[3];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[4];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[5];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[6];
while(U1STAbits.TRMT != 0x1);
U1TXREG = c11[7];
while(U1STAbits.TRMT != 0x1);
U1TXREG = '\n';

//if (U1STAbits.URXDA == 1)
//{
//		testRX = U1RXREG;
//		while(U1STAbits.TRMT != 0x1);
//		U1TXREG = testRX;
//}
/*if (U1STAbits.TRMT = 0x1)
{
		while(U1STAbits.TRMT != 0x1);
		//snprintf(c4,50,"%f",c5);
		sprintf(c4,50,"%i",sPOT);
		U1TXREG = c4[0];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = c4[1];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = c4[2];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = c4[3];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = c4[4];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = c4[5];
		while(U1STAbits.TRMT != 0x1);
		U1TXREG = '\n';

		//c5 = c5 + 0.12334;
```

```
        }*/


}

void RTCCInit(void)
{
    RCFGCALbits.RTCEN = 0x0; //Disable RTCC
    NVMKEY = 0x55;
    NVMKEY = 0xAA;
    RCFGCALbits.RTCWREN = 0x1; //Enable RTCC wrt
    RCFGCALbits.RTCOE = 0x0;
    RCFGCALbits.RTCPTR = 0x3;
    RTCVAL = 0x0015; //(xxYY)
    RTCVAL = 0x0728; //(MMDD)
    RTCVAL = 0x0000; //(xWHH)
    RTCVAL = 0x0000; //(MMSS)
    RCFGCALbits.CAL = 0b01111111;
    RCFGCALbits.RTCEN = 0x1; //Enable RTCC
    RCFGCALbits.RTCWREN = 0x0; //Disable RTCC wrt
}

void RTCCRead(void)
{
    while(RCFGCALbits.RTCSYNC==1);
    RCFGCALbits.RTCPTR=0x3;
    // Collect values
    year=RTCVAL;

    month_day=RTCVAL;

    wday_hour=RTCVAL;

    min_sec=RTCVAL;
}

void IOInit(void) //Initialize the I/O ports
{
    TRISAbits.TRISA1 = 1;                    //Set PORTA<1> as input -> AN1 is MUXA-
```

```
        TRISBbits.TRISB3 = 1;               //Set PORTB<3> as input -> AN5 is MUXA+ First
scan input 20
        TRISBbits.TRISB2 = 1;               //Set PORTB<2> as input -> AN4 is MUXA+ First
scan input 10
        TRISBbits.TRISB12 = 1;                      //Set PORTB<12> as input -> AN12 is
MUXA+ First scan input 1000
        TRISBbits.TRISB13 = 1;                      //Set PORTB<13> as input -> AN11 is
MUXA+ First scan input 800
        TRISBbits.TRISB14 = 1;                      //Set PORTB<14> as input -> AN10 is
MUXA+ First scan input 400
        TRISBbits.TRISB15 = 1;                      //Set PORTB<15> as input -> AN09 is
MUXA+ First scan input 200
}

void ADCInit(void) //Initialize the A/D converter
{
        AD1CON1bits.AD12B = 0b1;            //12-bit operation
        AD1CON2bits.VCFG = 0b000;           //Voltage references -> ADREF+: Avdd & ADREF-:
Avss
        AD1CON3bits.ADRC = 0b0;                     //Select system clock
        AD1CON3bits.ADCS = 0b00111111;      //TCY · (ADCS<7:0> + 1) = 64 · TCY = TAD
        AD1PCFGL = 0x1CD;                           //Select AN12, AN11, AN10, AN9, AN5, AN4
and AN1 as analog pin
        AD1CHS0bits.CH0NA = 0b0;            //Negative input -> AN1
        AD1CON2bits.CSCNA = 0b1;            //Select SCAN mode
        AD1CON2bits.CHPS = 0b00;            //Select only CHO, single channel mode
        AD1CON2bits.ALTS = 0b0;                     //Sample A only
        AD1CON1bits.SIMSAM = 0b0;           //Sequencial sampling
        AD1CSSL = 0x20;                             //Select AN10 and AN9 as sample input
        AD1CON1bits.ASAM = 0b1;                     //Auto sampling
        AD1CON1bits.ADDMABM = 0b1;                  //DMA buffers are written in order of
conversion
        AD1CON1bits.SSRC = 0b111;           //Auto sampling
        AD1CON1bits.FORM = 0b00;            //Conversion results -> Unsigned Integer
        AD1CON2bits.SMPI = 0b0000;          //DMA buffer pointer increment rate -> Increment
every 2 conversions
        AD1CON4bits.DMABL = 0b000;          //Number of samples in DMA buffer per input -> 1

        IFS0bits.AD1IF = 0b0;               //Clear ADC1 interrupt flag
        IPC3bits.AD1IP = 0b000;                     //Set ADC1 interrupt priority 1
        IEC0bits.AD1IE = 0b1;                   //Enable interrupt
}

void DMAInit (void) //Initialize DMA
{
        DMA0REQbits.IRQSEL = 0b0001101;    //Interrupt request code for ADC1
        DMA0PAD = 0x0300;                           //Select ADC1 for DMA0
        DMA0CONbits.SIZE = 0b0;                     //Buffer words
        DMA0CONbits.DIR = 0b0;                      //Read from peripheral to DMA buffer
        DMA0CONbits.HALF = 0b0;                     //Generate DMA interrupt at end of block
sample
        DMA0CONbits.NULLW = 0b0;           //Select DMA for normal operation
        DMA0CONbits.AMODE = 0b00;          //Register indirect with post-increment mode
        DMA0CONbits.MODE = 0b00;           //Continuous with Ping-Pong disabled
        DMA0REQbits.FORCE = 0b0;           //No manual force instruction
        DMA0STA        = 0x0000;                             //Start address A -> _DMA_BASE
(0x2000) + DMA0STA
        DMA0STB = 0x0010;                           //Only used for Ping-Pong
        DMA0CNT = 0;                                //Increments after 2 count cycles

        IFS0bits.DMA0IF = 0b0;             //Clear DMA0 interrupt flag
        IPC1bits.DMA0IP = 0b000;           //Priority 1
        IEC0bits.DMA0IE = 0b1;                      //Enable DMA0 interrupt flag
}

void ADCStart(void) //Turn DMA and ADC on
{
        DMA0CONbits.CHEN = 1;                       //Enable DMA0 buffer
        AD1CON1bits.ADON = 1;                       //Turn on ADC1
}

void ADCStop(void) //Turn DMA and ADC on
{
        DMA0CONbits.CHEN = 0;                       //Enable DMA0 buffer
        AD1CON1bits.ADON = 0;                       //Turn on ADC1
}

void OC1Init(void) //Setup OC1 as PWM
{
                // Configure RB15 for OC1
        T2CONbits.TON = 0b0; // Switch off Timer2 for OC1 setup
        T2CONbits.T32 = 0b0; // Use 16bit timer
        T2CONbits.TCS = 0b0; // Select Fosc/2 as timer clock source
        T2CONbits.TGATE = 0b0;
        T2CONbits.TCKPS = 0b11; // Prescaler: 1:256
        OC1CON = 0x0000; // Disable OC1
        PR2 = 0x1CB; // Timer2 Period, Value 44898
        OC1R = 0x0000; // Primary compare register for OC1, Value:0
        OC1RS = 0x15; // Secondary compare register for OC1, Value:40401
        // RPOR7bits.RP15R = 0b10010; // Setup pin RB15 as OC1 output pin
        RPOR3bits.RP7R = 0b10010; // Setup pin RB7 as OC1 output pin
        OC1CONbits.OCTSEL = 0b0; // Select Timer2 as source for OC1
        OC1CONbits.OCM = 0b110; // OC1 in PWM mode without fault protection
        T2CONbits.TON = 0b1; // Switch on Timer2, OC1 setup completed
        //TRISBbits.TRISB15 = 0b0; // Setup pin RB15 as output
        TRISBbits.TRISB7 = 0b0; // Setup pin RB7 as output

        // Configure RB7/RP7 as Open Drain
        ODCBbits.ODCB7 = 0b1;
}

void UARTinit(void)
{
        RPOR4bits.RP9R = 0b00011;
        RPINR18bits.U1RXR = 0b01000;

        U1MODEbits.UARTEN = 0x0;
        U1MODEbits.UEN = 0x0;
        U1MODEbits.BRGH = 0x0;
        U1MODEbits.PDSEL = 0x0;
        U1MODEbits.STSEL = 0x0;
        U1STAbits.UTXINV = 0x0;
        U1BRG = 0x95;
        U1MODEbits.UARTEN = 0x1;
        U1STAbits.UTXEN = 0x1;
        //U1STAbits.URXEN = 0x1;
}
```

Appendix D: Learning Algorithm Software

```
def learnP():
    global res26
    global
p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,p21,p22,p23,p24,p25
    global
q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20,q21,q22,q23,q24,q25
    global r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20,r21,r22,r23,r24,r25

    Dp1 = (dn - ds)*(BatteryComp()/res26)*rule1()*Tin
    Dp2 = (dn - ds)*(BatteryComp()/res26)*rule2()*Tin
    Dp3 = (dn - ds)*(BatteryComp()/res26)*rule3()*Tin
    Dp4 = (dn - ds)*(BatteryComp()/res26)*rule4()*Tin
    Dp5 = (dn - ds)*(BatteryComp()/res26)*rule5()*Tin
    Dp6 = (dn - ds)*(BatteryComp()/res26)*rule6()*Tin
    Dp7 = (dn - ds)*(BatteryComp()/res26)*rule7()*Tin
    Dp8 = (dn - ds)*(BatteryComp()/res26)*rule8()*Tin
    Dp9 = (dn - ds)*(BatteryComp()/res26)*rule9()*Tin
    Dp10 = (dn - ds)*(BatteryComp()/res26)*rule10()*Tin
    Dp11 = (dn - ds)*(BatteryComp()/res26)*rule11()*Tin
    Dp12 = (dn - ds)*(BatteryComp()/res26)*rule12()*Tin
    Dp13 = (dn - ds)*(BatteryComp()/res26)*rule13()*Tin
    Dp14 = (dn - ds)*(BatteryComp()/res26)*rule14()*Tin
    Dp15 = (dn - ds)*(BatteryComp()/res26)*rule15()*Tin
    Dp16 = (dn - ds)*(BatteryComp()/res26)*rule16()*Tin
    Dp17 = (dn - ds)*(BatteryComp()/res26)*rule17()*Tin
    Dp18 = (dn - ds)*(BatteryComp()/res26)*rule18()*Tin
    Dp19 = (dn - ds)*(BatteryComp()/res26)*rule19()*Tin
    Dp20 = (dn - ds)*(BatteryComp()/res26)*rule20()*Tin
    Dp21 = (dn - ds)*(BatteryComp()/res26)*rule21()*Tin
    Dp22 = (dn - ds)*(BatteryComp()/res26)*rule22()*Tin
    Dp23 = (dn - ds)*(BatteryComp()/res26)*rule23()*Tin
    Dp24 = (dn - ds)*(BatteryComp()/res26)*rule24()*Tin
    Dp25 = (dn - ds)*(BatteryComp()/res26)*rule25()*Tin

    p1 = p1 - Np*Dp1
    p2 = p2 - Np*Dp2
    p3 = p3 - Np*Dp3
    p4 = p4 - Np*Dp4
    p5 = p5 - Np*Dp5
```

```
p6 = p6 - Np*Dp6
p7 = p7 - Np*Dp7
p8 = p8 - Np*Dp8
p9 = p9 - Np*Dp9
p10 = p10 - Np*Dp10
p11 = p11 - Np*Dp11
p12 = p12 - Np*Dp12
p13 = p13 - Np*Dp13
p14 = p14 - Np*Dp14
p15 = p15 - Np*Dp15
p16 = p16 - Np*Dp16
p17 = p17 - Np*Dp17
p18 = p18 - Np*Dp18
p19 = p19 - Np*Dp19
p20 = p20 - Np*Dp20
p21 = p21 - Np*Dp21
p22 = p22 - Np*Dp22
p23 = p23 - Np*Dp23
p24 = p24 - Np*Dp24
p25 = p25 - Np*Dp25

Dq1 = (dn - ds)*(BatteryComp()/res26)*rule1()*Gin
Dq2 = (dn - ds)*(BatteryComp()/res26)*rule2()*Gin
Dq3 = (dn - ds)*(BatteryComp()/res26)*rule3()*Gin
Dq4 = (dn - ds)*(BatteryComp()/res26)*rule4()*Gin
Dq5 = (dn - ds)*(BatteryComp()/res26)*rule5()*Gin
Dq6 = (dn - ds)*(BatteryComp()/res26)*rule6()*Gin
Dq7 = (dn - ds)*(BatteryComp()/res26)*rule7()*Gin
Dq8 = (dn - ds)*(BatteryComp()/res26)*rule8()*Gin
Dq9 = (dn - ds)*(BatteryComp()/res26)*rule9()*Gin
Dq10 = (dn - ds)*(BatteryComp()/res26)*rule10()*Gin
Dq11 = (dn - ds)*(BatteryComp()/res26)*rule11()*Gin
Dq12 = (dn - ds)*(BatteryComp()/res26)*rule12()*Gin
Dq13 = (dn - ds)*(BatteryComp()/res26)*rule13()*Gin
Dq14 = (dn - ds)*(BatteryComp()/res26)*rule14()*Gin
Dq15 = (dn - ds)*(BatteryComp()/res26)*rule15()*Gin
Dq16 = (dn - ds)*(BatteryComp()/res26)*rule16()*Gin
Dq17 = (dn - ds)*(BatteryComp()/res26)*rule17()*Gin
Dq18 = (dn - ds)*(BatteryComp()/res26)*rule18()*Gin
Dq19 = (dn - ds)*(BatteryComp()/res26)*rule19()*Gin
```

```
Dq20 = (dn - ds)*(BatteryComp()/res26)*rule20()*Gin
Dq21 = (dn - ds)*(BatteryComp()/res26)*rule21()*Gin
Dq22 = (dn - ds)*(BatteryComp()/res26)*rule22()*Gin
Dq23 = (dn - ds)*(BatteryComp()/res26)*rule23()*Gin
Dq24 = (dn - ds)*(BatteryComp()/res26)*rule24()*Gin
Dq25 = (dn - ds)*(BatteryComp()/res26)*rule25()*Gin

q1 = q1 - Nq*Dq1
q2 = q2 - Nq*Dq2
q3 = q3 - Nq*Dq3
q4 = q4 - Nq*Dq4
q5 = q5 - Nq*Dq5
q6 = q6 - Nq*Dq6
q7 = q7 - Nq*Dq7
q8 = q8 - Nq*Dq8
q9 = q9 - Nq*Dq9
q10 = q10 - Nq*Dq10
q11 = q11 - Nq*Dq11
q12 = q12 - Nq*Dq12
q13 = q13 - Nq*Dq13
q14 = q14 - Nq*Dq14
q15 = q15 - Nq*Dq15
q16 = q16 - Nq*Dq16
q17 = q17 - Nq*Dq17
q18 = q18 - Nq*Dq18
q19 = q19 - Nq*Dq19
q20 = q20 - Nq*Dq20
q21 = q21 - Nq*Dq21
q22 = q22 - Nq*Dq22
q23 = q23 - Nq*Dq23
q24 = q24 - Nq*Dq24
q25 = q25 - Nq*Dq25

Dr1 = (dn - ds)*(BatteryComp()/res26)*rule1()
Dr2 = (dn - ds)*(BatteryComp()/res26)*rule2()
Dr3 = (dn - ds)*(BatteryComp()/res26)*rule3()
Dr4 = (dn - ds)*(BatteryComp()/res26)*rule4()
Dr5 = (dn - ds)*(BatteryComp()/res26)*rule5()
Dr6 = (dn - ds)*(BatteryComp()/res26)*rule6()
Dr7 = (dn - ds)*(BatteryComp()/res26)*rule7()
```

```
Dr8 = (dn - ds)*(BatteryComp()/res26)*rule8()
Dr9 = (dn - ds)*(BatteryComp()/res26)*rule9()
Dr10 = (dn - ds)*(BatteryComp()/res26)*rule10()
Dr11 = (dn - ds)*(BatteryComp()/res26)*rule11()
Dr12 = (dn - ds)*(BatteryComp()/res26)*rule12()
Dr13 = (dn - ds)*(BatteryComp()/res26)*rule13()
Dr14 = (dn - ds)*(BatteryComp()/res26)*rule14()
Dr15 = (dn - ds)*(BatteryComp()/res26)*rule15()
Dr16 = (dn - ds)*(BatteryComp()/res26)*rule16()
Dr17 = (dn - ds)*(BatteryComp()/res26)*rule17()
Dr18 = (dn - ds)*(BatteryComp()/res26)*rule18()
Dr19 = (dn - ds)*(BatteryComp()/res26)*rule19()
Dr20 = (dn - ds)*(BatteryComp()/res26)*rule20()
Dr21 = (dn - ds)*(BatteryComp()/res26)*rule21()
Dr22 = (dn - ds)*(BatteryComp()/res26)*rule22()
Dr23 = (dn - ds)*(BatteryComp()/res26)*rule23()
Dr24 = (dn - ds)*(BatteryComp()/res26)*rule24()
Dr25 = (dn - ds)*(BatteryComp()/res26)*rule25()

r1 = r1 - Nr*Dr1
r2 = r2 - Nr*Dr2
r3 = r3 - Nr*Dr3
r4 = r4 - Nr*Dr4
r5 = r5 - Nr*Dr5
r6 = r6 - Nr*Dr6
r7 = r7 - Nr*Dr7
r8 = r8 - Nr*Dr8
r9 = r9 - Nr*Dr9
r10 = r10 - Nr*Dr10
r11 = r11 - Nr*Dr11
r12 = r12 - Nr*Dr12
r13 = r13 - Nr*Dr13
r14 = r14 - Nr*Dr14
r15 = r15 - Nr*Dr15
r16 = r16 - Nr*Dr16
r17 = r17 - Nr*Dr17
r18 = r18 - Nr*Dr18
r19 = r19 - Nr*Dr19
r20 = r20 - Nr*Dr20
r21 = r21 - Nr*Dr21
```

```
    r22 = r22 - Nr*Dr22
    r23 = r23 - Nr*Dr23
    r24 = r24 - Nr*Dr24
    r25 = r25 - Nr*Dr25

def learnwT():
    global res1,res2,res3,res4,res5,res6,res7,res8,res9,res10,res11,res12,res13,res14,res15,res16,res17,res18,res19,res20,res21,res22,res23,res24,res25,res26
    global wT1,wT2,wT3,wT4,wT5,wT6,wT7,wT8,wT9,wT10,wT11,wT12,wT13,wT14,wT15,wT16,wT17,wT18,wT19,wT20,wT21,wT22,wT23,wT24,wT25
    global wG1,wG2,wG3,wG4,wG5,wG6,wG7,wG8,wG9,wG10,wG11,wG12,wG13,wG14,wG15,wG16,wG17,wG18,wG19,wG20,wG21,wG22,wG23,wG24,wG25
    DwT1 = (dn - ds)*((BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) - (dn/res26))*Tpv_Low(Tin)
    DwT2 = (dn - ds)*((BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) - (dn/res26))*Tpv_Low(Tin)
    DwT3 = (dn - ds)*((BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) - (dn/res26))*Tpv_Low(Tin)
    DwT4 = (dn - ds)*((BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) - (dn/res26))*Tpv_Low(Tin)
    DwT5 = (dn - ds)*((BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) - (dn/res26))*Tpv_Low(Tin)
    DwT6 = (dn - ds)*((BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) - (dn/res26))*Tpv_Moderate(Tin)
    DwT7 = (dn - ds)*((BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) - (dn/res26))*Tpv_Moderate(Tin)
    DwT8 = (dn - ds)*((BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) - (dn/res26))*Tpv_Moderate(Tin)
    DwT9 = (dn - ds)*((BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) - (dn/res26))*Tpv_Moderate(Tin)
    DwT10 = (dn - ds)*((BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) - (dn/res26))*Tpv_Moderate(Tin)
    DwT11 = (dn - ds)*((BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) - (dn/res26))*Tpv_High(Tin)
    DwT12 = (dn - ds)*((BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) - (dn/res26))*Tpv_High(Tin)
    DwT13 = (dn - ds)*((BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) - (dn/res26))*Tpv_High(Tin)
    DwT14 = (dn - ds)*((BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) - (dn/res26))*Tpv_High(Tin)
    DwT15 = (dn - ds)*((BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) - (dn/res26))*Tpv_High(Tin)
    DwT16 = (dn - ds)*((BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) - (dn/res26))*Tpv_VHigh(Tin)
    DwT17 = (dn - ds)*((BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) - (dn/res26))*Tpv_VHigh(Tin)
    DwT18 = (dn - ds)*((BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) - (dn/res26))*Tpv_VHigh(Tin)
    DwT19 = (dn - ds)*((BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) - (dn/res26))*Tpv_VHigh(Tin)
    DwT20 = (dn - ds)*((BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) - (dn/res26))*Tpv_VHigh(Tin)
    DwT21 = (dn - ds)*((BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) - (dn/res26))*Tpv_XHigh(Tin)
    DwT22 = (dn - ds)*((BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) - (dn/res26))*Tpv_XHigh(Tin)
    DwT23 = (dn - ds)*((BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) - (dn/res26))*Tpv_XHigh(Tin)
    DwT24 = (dn - ds)*((BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) - (dn/res26))*Tpv_XHigh(Tin)
```

```
DwT25 = (dn - ds)*((BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) - (dn/res26))*Tpv_XHigh(Tin)

wT1 = wT1 - Nt*DwT1
wT2 = wT2 - Nt*DwT2
wT3 = wT3 - Nt*DwT3
wT4 = wT4 - Nt*DwT4
wT5 = wT5 - Nt*DwT5
wT6 = wT6 - Nt*DwT6
wT7 = wT7 - Nt*DwT7
wT8 = wT8 - Nt*DwT8
wT9 = wT9 - Nt*DwT9
wT10 = wT10 - Nt*DwT10
wT11 = wT11 - Nt*DwT11
wT12 = wT12 - Nt*DwT12
wT13 = wT13 - Nt*DwT13
wT14 = wT14 - Nt*DwT14
wT15 = wT15 - Nt*DwT15
wT16 = wT16 - Nt*DwT16
wT17 = wT17 - Nt*DwT17
wT18 = wT18 - Nt*DwT18
wT19 = wT19 - Nt*DwT19
wT20 = wT20 - Nt*DwT20
wT21 = wT21 - Nt*DwT21
wT22 = wT22 - Nt*DwT22
wT23 = wT23 - Nt*DwT23
wT24 = wT24 - Nt*DwT24
wT25 = wT25 - Nt*DwT25

DwG1 = (dn - ds)*((BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) - (dn/res26))*Gsolar_XLow(Gin)
DwG2 = (dn - ds)*((BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) - (dn/res26))*Gsolar_VLow(Gin)
DwG3 = (dn - ds)*((BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) - (dn/res26))*Gsolar_Low(Gin)
DwG4 = (dn - ds)*((BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) - (dn/res26))*Gsolar_Moderate(Gin)
DwG5 = (dn - ds)*((BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) - (dn/res26))*Gsolar_High(Gin)
DwG6 = (dn - ds)*((BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) - (dn/res26))*Gsolar_XLow(Gin)
DwG7 = (dn - ds)*((BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) - (dn/res26))*Gsolar_VLow(Gin)
DwG8 = (dn - ds)*((BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) - (dn/res26))*Gsolar_Low(Gin)
DwG9 = (dn - ds)*((BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) - (dn/res26))*Gsolar_Moderate(Gin)
```

```
DwG10 = (dn - ds)*((BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) - (dn/res26))*Gsolar_High(Gin)
DwG11 = (dn - ds)*((BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) - (dn/res26))*Gsolar_XLow(Gin)
DwG12 = (dn - ds)*((BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) - (dn/res26))*Gsolar_VLow(Gin)
DwG13 = (dn - ds)*((BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) - (dn/res26))*Gsolar_Low(Gin)
DwG14 = (dn - ds)*((BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) - (dn/res26))*Gsolar_Moderate(Gin)
DwG15 = (dn - ds)*((BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) - (dn/res26))*Gsolar_High(Gin)
DwG16 = (dn - ds)*((BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) - (dn/res26))*Gsolar_XLow(Gin)
DwG17 = (dn - ds)*((BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) - (dn/res26))*Gsolar_VLow(Gin)
DwG18 = (dn - ds)*((BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) - (dn/res26))*Gsolar_Low(Gin)
DwG19 = (dn - ds)*((BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) - (dn/res26))*Gsolar_Moderate(Gin)
DwG20 = (dn - ds)*((BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) - (dn/res26))*Gsolar_High(Gin)
DwG21 = (dn - ds)*((BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) - (dn/res26))*Gsolar_XLow(Gin)
DwG22 = (dn - ds)*((BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) - (dn/res26))*Gsolar_VLow(Gin)
DwG23 = (dn - ds)*((BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) - (dn/res26))*Gsolar_Low(Gin)
DwG24 = (dn - ds)*((BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) - (dn/res26))*Gsolar_Moderate(Gin)
DwG25 = (dn - ds)*((BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) - (dn/res26))*Gsolar_High(Gin)

wG1 = wG1 - Ng*DwG1
wG2 = wG2 - Ng*DwG2
wG3 = wG3 - Ng*DwG3
wG4 = wG4 - Ng*DwG4
wG5 = wG5 - Ng*DwG5
wG6 = wG6 - Ng*DwG6
wG7 = wG7 - Ng*DwG7
```

```
    wG8 = wG8 - Ng*DwG8
    wG9 = wG9 - Ng*DwG9
    wG10 = wG10 - Ng*DwG10
    wG11 = wG11 - Ng*DwG11
    wG12 = wG12 - Ng*DwG12
    wG13 = wG13 - Ng*DwG13
    wG14 = wG14 - Ng*DwG14
    wG15 = wG15 - Ng*DwG15
    wG16 = wG16 - Ng*DwG16
    wG17 = wG17 - Ng*DwG17
    wG18 = wG18 - Ng*DwG18
    wG19 = wG19 - Ng*DwG19
    wG20 = wG20 - Ng*DwG20
    wG21 = wG21 - Ng*DwG21
    wG22 = wG22 - Ng*DwG22
    wG23 = wG23 - Ng*DwG23
    wG24 = wG24 - Ng*DwG24
    wG25 = wG25 - Ng*DwG25

def learnInput():
    global res26
    global
wT1,wT2,wT3,wT4,wT5,wT6,wT7,wT8,wT9,wT10,wT11,wT12,wT13,wT14,wT15,wT16,wT17,wT18,w
T19,wT20,wT21,wT22,wT23,wT24,wT25
    global
wG1,wG2,wG3,wG4,wG5,wG6,wG7,wG8,wG9,wG10,wG11,wG12,wG13,wG14,wG15,wG16,wG17,w
G18,wG19,wG20,wG21,wG22,wG23,wG24,wG25
    global
cTL,cTM,cTH,cTV,cTX,sTL,sTM,sTH,sTV,sTX,cGX,cGV,cGL,cGM,cGH,sGX,sGV,sGL,sGM,sGH
    DcTL = (((dn - ds)*(BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) + (dn -
ds)*BatteryComp()*(dn/res26))*wT1 + ((dn - ds)*(BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) + (dn -
ds)*BatteryComp()*(dn/res26))*wT2/
         + ((dn - ds)*(BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) + (dn -
ds)*BatteryComp()*(dn/res26))*wT3 + ((dn - ds)*(BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) + (dn -
ds)*BatteryComp()*(dn/res26))*wT4/
         + ((dn - ds)*(BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) + (dn -
ds)*BatteryComp()*(dn/res26))*wT5)*(Tin/(math.pow(sTL,2))-1/(math.pow(sTL,2)))*math.exp(-
1*math.pow((Tin-cTL),2)/(2*math.pow(sTL,2)))
```

```
  DsTL = (((dn - ds)*(BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) + (dn -
ds)*BatteryComp()*(dn/res26))*wT1 + ((dn - ds)*(BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) + (dn -
ds)*BatteryComp()*(dn/res26))*wT2/
        + ((dn - ds)*(BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) + (dn -
ds)*BatteryComp()*(dn/res26))*wT3 + ((dn - ds)*(BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) + (dn -
ds)*BatteryComp()*(dn/res26))*wT4/
        + ((dn - ds)*(BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) + (dn -
ds)*BatteryComp()*(dn/res26))*wT5)*(math.pow((Tin-cTL),2)/(math.pow(sTL,3)))*math.exp(-
1*math.pow((Tin-cTL),2)/(2*math.pow(sTL,2)))
  DcTM = (((dn - ds)*(BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) + (dn -
ds)*BatteryComp()*(dn/res26))*wT6 + ((dn - ds)*(BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) + (dn -
ds)*BatteryComp()*(dn/res26))*wT7/
        + ((dn - ds)*(BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) + (dn -
ds)*BatteryComp()*(dn/res26))*wT8 + ((dn - ds)*(BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) + (dn -
ds)*BatteryComp()*(dn/res26))*wT9/
        + ((dn - ds)*(BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) + (dn -
ds)*BatteryComp()*(dn/res26))*wT10)*(Tin/(math.pow(sTM,2))-1/(math.pow(sTM,2)))*math.exp(-
1*math.pow((Tin-cTM),2)/(2*math.pow(sTM,2)))
  DsTM = (((dn - ds)*(BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) + (dn -
ds)*BatteryComp()*(dn/res26))*wT6 + ((dn - ds)*(BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) + (dn -
ds)*BatteryComp()*(dn/res26))*wT7/
        + ((dn - ds)*(BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) + (dn -
ds)*BatteryComp()*(dn/res26))*wT8 + ((dn - ds)*(BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) + (dn -
ds)*BatteryComp()*(dn/res26))*wT9/
        + ((dn - ds)*(BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) + (dn -
ds)*BatteryComp()*(dn/res26))*wT10)*(math.pow((Tin-cTM),2)/(math.pow(sTM,3)))*math.exp(-
1*math.pow((Tin-cTM),2)/(2*math.pow(sTM,2)))
  DcTH = (((dn - ds)*(BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) + (dn -
ds)*BatteryComp()*(dn/res26))*wT11 + ((dn - ds)*(BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) +
(dn - ds)*BatteryComp()*(dn/res26))*wT12/
        + ((dn - ds)*(BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) + (dn -
ds)*BatteryComp()*(dn/res26))*wT13 + ((dn - ds)*(BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) +
(dn - ds)*BatteryComp()*(dn/res26))*wT14/
        + ((dn - ds)*(BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) + (dn -
ds)*BatteryComp()*(dn/res26))*wT15)*(Tin/(math.pow(sTH,2))-1/(math.pow(sTH,2)))*math.exp(-
1*math.pow((Tin-cTH),2)/(2*math.pow(sTH,2)))
  DsTH = (((dn - ds)*(BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) + (dn -
ds)*BatteryComp()*(dn/res26))*wT11 + ((dn - ds)*(BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) +
(dn - ds)*BatteryComp()*(dn/res26))*wT12/
```

```
        + ((dn - ds)*(BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) + (dn -
ds)*BatteryComp()*(dn/res26))*wT13 + ((dn - ds)*(BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) +
(dn - ds)*BatteryComp()*(dn/res26))*wT14/
        + ((dn - ds)*(BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) + (dn -
ds)*BatteryComp()*(dn/res26))*wT15)*(math.pow((Tin-cTH),2)/(math.pow(sTH,3)))*math.exp(-
1*math.pow((Tin-cTH),2)/(2*math.pow(sTH,2)))
  DcTV = (((dn - ds)*(BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) + (dn -
ds)*BatteryComp()*(dn/res26))*wT16 + ((dn - ds)*(BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) +
(dn - ds)*BatteryComp()*(dn/res26))*wT17/
        + ((dn - ds)*(BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) + (dn -
ds)*BatteryComp()*(dn/res26))*wT18 + ((dn - ds)*(BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) +
(dn - ds)*BatteryComp()*(dn/res26))*wT19/
        + ((dn - ds)*(BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) + (dn -
ds)*BatteryComp()*(dn/res26))*wT20)*(Tin/(math.pow(sTV,2))-1/(math.pow(sTV,2)))*math.exp(-
1*math.pow((Tin-cTV),2)/(2*math.pow(sTV,2)))
  DsTV = (((dn - ds)*(BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) + (dn -
ds)*BatteryComp()*(dn/res26))*wT16 + ((dn - ds)*(BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) +
(dn - ds)*BatteryComp()*(dn/res26))*wT17/
        + ((dn - ds)*(BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) + (dn -
ds)*BatteryComp()*(dn/res26))*wT18 + ((dn - ds)*(BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) +
(dn - ds)*BatteryComp()*(dn/res26))*wT19/
        + ((dn - ds)*(BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) + (dn -
ds)*BatteryComp()*(dn/res26))*wT20)*(math.pow((Tin-cTV),2)/(math.pow(sTV,3)))*math.exp(-
1*math.pow((Tin-cTV),2)/(2*math.pow(sTV,2)))
  DcTX = (((dn - ds)*(BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) + (dn -
ds)*BatteryComp()*(dn/res26))*wT21 + ((dn - ds)*(BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) +
(dn - ds)*BatteryComp()*(dn/res26))*wT22/
        + ((dn - ds)*(BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) + (dn -
ds)*BatteryComp()*(dn/res26))*wT23 + ((dn - ds)*(BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) +
(dn - ds)*BatteryComp()*(dn/res26))*wT24/
        + ((dn - ds)*(BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) + (dn -
ds)*BatteryComp()*(dn/res26))*wT25)*(Tin/(math.pow(sTX,2))-1/(math.pow(sTX,2)))*math.exp(-
1*math.pow((Tin-cTX),2)/(2*math.pow(sTX,2)))
  DsTX = (((dn - ds)*(BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) + (dn -
ds)*BatteryComp()*(dn/res26))*wT21 + ((dn - ds)*(BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) +
(dn - ds)*BatteryComp()*(dn/res26))*wT22/
        + ((dn - ds)*(BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) + (dn -
ds)*BatteryComp()*(dn/res26))*wT23 + ((dn - ds)*(BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) +
(dn - ds)*BatteryComp()*(dn/res26))*wT24/
```

```
          + ((dn - ds)*(BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) + (dn -
ds)*BatteryComp()*(dn/res26))*wT25)*(math.pow((Tin-cTX),2)/(math.pow(sTX,3)))*math.exp(-
1*math.pow((Tin-cTX),2)/(2*math.pow(sTX,2)))

   cTL = cTL -NcL1*DcTL
   cTM = cTM -NcM1*DcTM
   cTH = cTH -NcH1*DcTH
   cTV = cTV -NcV1*DcTV
   cTX = cTX -NcX1*DcTX
   sTL = sTL -NsL1*DsTL
   sTM = sTM -NsM1*DsTM
   sTH = sTH -NsH1*DsTH
   sTV = sTV -NsV1*DsTV
   sTX = sTX -NsX1*DsTX

   DcGX = (((dn - ds)*(BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) + (dn -
ds)*BatteryComp()*(dn/res26))*wG1 + ((dn - ds)*(BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) + (dn -
ds)*BatteryComp()*(dn/res26))*wG6/
          + ((dn - ds)*(BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) + (dn -
ds)*BatteryComp()*(dn/res26))*wG11 + ((dn - ds)*(BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) +
(dn - ds)*BatteryComp()*(dn/res26))*wG16/
          + ((dn - ds)*(BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) + (dn -
ds)*BatteryComp()*(dn/res26))*wG21)*(Tin/(math.pow(sGX,2))-1/(math.pow(sGX,2)))*math.exp(-
1*math.pow((Tin-cGX),2)/(2*math.pow(sGX,2)))
   DsGX = (((dn - ds)*(BatteryComp()/res26)*(p1*Tin + q1*Gin + r1) + (dn -
ds)*BatteryComp()*(dn/res26))*wG1 + ((dn - ds)*(BatteryComp()/res26)*(p6*Tin + q6*Gin + r6) + (dn -
ds)*BatteryComp()*(dn/res26))*wG6/
          + ((dn - ds)*(BatteryComp()/res26)*(p11*Tin + q11*Gin + r11) + (dn -
ds)*BatteryComp()*(dn/res26))*wG11 + ((dn - ds)*(BatteryComp()/res26)*(p16*Tin + q16*Gin + r16) +
(dn - ds)*BatteryComp()*(dn/res26))*wG16/
          + ((dn - ds)*(BatteryComp()/res26)*(p21*Tin + q21*Gin + r21) + (dn -
ds)*BatteryComp()*(dn/res26))*wG21)*(math.pow((Tin-cGX),2)/(math.pow(sGX,3)))*math.exp(-
1*math.pow((Tin-cGX),2)/(2*math.pow(sGX,2)))
   DcGV = (((dn - ds)*(BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) + (dn -
ds)*BatteryComp()*(dn/res26))*wG2 + ((dn - ds)*(BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) + (dn -
ds)*BatteryComp()*(dn/res26))*wG7/
          + ((dn - ds)*(BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) + (dn -
ds)*BatteryComp()*(dn/res26))*wG12 + ((dn - ds)*(BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) +
(dn - ds)*BatteryComp()*(dn/res26))*wG17/
```

```
          + ((dn - ds)*(BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) + (dn -
ds)*BatteryComp()*(dn/res26))*wG22)*(Tin/(math.pow(sGV,2))-1/(math.pow(sGV,2)))*math.exp(-
1*math.pow((Tin-cGV),2)/(2*math.pow(sGV,2)))
   DsGV = (((dn - ds)*(BatteryComp()/res26)*(p2*Tin + q2*Gin + r2) + (dn -
ds)*BatteryComp()*(dn/res26))*wG2 + ((dn - ds)*(BatteryComp()/res26)*(p7*Tin + q7*Gin + r7) + (dn -
ds)*BatteryComp()*(dn/res26))*wG7/
          + ((dn - ds)*(BatteryComp()/res26)*(p12*Tin + q12*Gin + r12) + (dn -
ds)*BatteryComp()*(dn/res26))*wG12 + ((dn - ds)*(BatteryComp()/res26)*(p17*Tin + q17*Gin + r17) +
(dn - ds)*BatteryComp()*(dn/res26))*wG17/
          + ((dn - ds)*(BatteryComp()/res26)*(p22*Tin + q22*Gin + r22) + (dn -
ds)*BatteryComp()*(dn/res26))*wG22)*(math.pow((Tin-cGV),2)/(math.pow(sGV,3)))*math.exp(-
1*math.pow((Tin-cGV),2)/(2*math.pow(sGV,2)))
   DcGL = (((dn - ds)*(BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) + (dn -
ds)*BatteryComp()*(dn/res26))*wG3 + ((dn - ds)*(BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) + (dn -
ds)*BatteryComp()*(dn/res26))*wG8/
          + ((dn - ds)*(BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) + (dn -
ds)*BatteryComp()*(dn/res26))*wG13 + ((dn - ds)*(BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) +
(dn - ds)*BatteryComp()*(dn/res26))*wG18/
          + ((dn - ds)*(BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) + (dn -
ds)*BatteryComp()*(dn/res26))*wG23)*(Tin/(math.pow(sGL,2))-1/(math.pow(sGL,2)))*math.exp(-
1*math.pow((Tin-cGL),2)/(2*math.pow(sGL,2)))
   DsGL = (((dn - ds)*(BatteryComp()/res26)*(p3*Tin + q3*Gin + r3) + (dn -
ds)*BatteryComp()*(dn/res26))*wG3 + ((dn - ds)*(BatteryComp()/res26)*(p8*Tin + q8*Gin + r8) + (dn -
ds)*BatteryComp()*(dn/res26))*wG8/
          + ((dn - ds)*(BatteryComp()/res26)*(p13*Tin + q13*Gin + r13) + (dn -
ds)*BatteryComp()*(dn/res26))*wG13 + ((dn - ds)*(BatteryComp()/res26)*(p18*Tin + q18*Gin + r18) +
(dn - ds)*BatteryComp()*(dn/res26))*wG18/
          + ((dn - ds)*(BatteryComp()/res26)*(p23*Tin + q23*Gin + r23) + (dn -
ds)*BatteryComp()*(dn/res26))*wG23)*(math.pow((Tin-cGL),2)/(math.pow(sGL,3)))*math.exp(-
1*math.pow((Tin-cGL),2)/(2*math.pow(sGL,2)))
   DcGM = (((dn - ds)*(BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) + (dn -
ds)*BatteryComp()*(dn/res26))*wG4 + ((dn - ds)*(BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) + (dn -
ds)*BatteryComp()*(dn/res26))*wG9/
          + ((dn - ds)*(BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) + (dn -
ds)*BatteryComp()*(dn/res26))*wG14 + ((dn - ds)*(BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) +
(dn - ds)*BatteryComp()*(dn/res26))*wG19/
          + ((dn - ds)*(BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) + (dn -
ds)*BatteryComp()*(dn/res26))*wG24)*(Tin/(math.pow(sGM,2))-1/(math.pow(sGM,2)))*math.exp(-
1*math.pow((Tin-cGM),2)/(2*math.pow(sGM,2)))
```

```
  DsGM = (((dn - ds)*(BatteryComp()/res26)*(p4*Tin + q4*Gin + r4) + (dn -
ds)*BatteryComp()*(dn/res26))*wG4 + ((dn - ds)*(BatteryComp()/res26)*(p9*Tin + q9*Gin + r9) + (dn -
ds)*BatteryComp()*(dn/res26))*wG9/
        + ((dn - ds)*(BatteryComp()/res26)*(p14*Tin + q14*Gin + r14) + (dn -
ds)*BatteryComp()*(dn/res26))*wG14 + ((dn - ds)*(BatteryComp()/res26)*(p19*Tin + q19*Gin + r19) +
(dn - ds)*BatteryComp()*(dn/res26))*wG19/
        + ((dn - ds)*(BatteryComp()/res26)*(p24*Tin + q24*Gin + r24) + (dn -
ds)*BatteryComp()*(dn/res26))*wG24)*(math.pow((Tin-cGM),2)/(math.pow(sGM,3)))*math.exp(-
1*math.pow((Tin-cGM),2)/(2*math.pow(sGM,2)))
  DcGH = (((dn - ds)*(BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) + (dn -
ds)*BatteryComp()*(dn/res26))*wG5 + ((dn - ds)*(BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) +
(dn - ds)*BatteryComp()*(dn/res26))*wG10/
        + ((dn - ds)*(BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) + (dn -
ds)*BatteryComp()*(dn/res26))*wG15 + ((dn - ds)*(BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) +
(dn - ds)*BatteryComp()*(dn/res26))*wG20/
        + ((dn - ds)*(BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) + (dn -
ds)*BatteryComp()*(dn/res26))*wG25)*(Tin/(math.pow(sGH,2))-1/(math.pow(sGH,2)))*math.exp(-
1*math.pow((Tin-cGH),2)/(2*math.pow(sGH,2)))
  DsGH = (((dn - ds)*(BatteryComp()/res26)*(p5*Tin + q5*Gin + r5) + (dn -
ds)*BatteryComp()*(dn/res26))*wG5 + ((dn - ds)*(BatteryComp()/res26)*(p10*Tin + q10*Gin + r10) +
(dn - ds)*BatteryComp()*(dn/res26))*wG10/
        + ((dn - ds)*(BatteryComp()/res26)*(p15*Tin + q15*Gin + r15) + (dn -
ds)*BatteryComp()*(dn/res26))*wG15 + ((dn - ds)*(BatteryComp()/res26)*(p20*Tin + q20*Gin + r20) +
(dn - ds)*BatteryComp()*(dn/res26))*wG20/
        + ((dn - ds)*(BatteryComp()/res26)*(p25*Tin + q25*Gin + r25) + (dn -
ds)*BatteryComp()*(dn/res26))*wG25)*(math.pow((Tin-cGH),2)/(math.pow(sGH,3)))*math.exp(-
1*math.pow((Tin-cGH),2)/(2*math.pow(sGH,2)))

  cGX = cGX - NcX2*DcGX
  cGV = cGV - NcV2*DcGV
  cGL = cGL - NcL2*DcGL
  cGM = cGM - NcM2*DcGM
  cGH = cGH - NcH2*DcGH
  sGX = sGX - NsX2*DsGX
  sGV = sGV - NsV2*DsGV
  sGL = sGL - NsL2*DsGL
  sGM = sGM - NsM2*DsGM
  sGH = sGH - NsH2*DsGH

count3 = 50
```

```
while (count3 > 0):
    print count3
    count3 = count3 - 1
    count = 1
    count2 = 99
    countAverage = count*count2
    Tin = 15.0
    Gin = 300.0
    ds = 0.7
    fSim_d = open('C:\Python27\ANFIS_txt\Sim_d.txt', 'r')
    fSim_T = open('C:\Python27\ANFIS_txt\Sim_T.txt', 'r')
    fSim_G = open('C:\Python27\ANFIS_txt\Sim_G.txt', 'r')

    while count > 0:
        count = count - 1
        while count2 > 0:
            count2 = count2 - 1
            Gin = float(fSim_G.readline())
            Tin = float(fSim_T.readline()) - 273.0
            ds = float(fSim_d.readline())
            while round(dn, 3) != round(ds, 3):
                dn = ANFIS(Tin,Gin)
                #wTbat = learnBAT(wTbat)
                #dn = ANFIS(Tin,Gin)
                learnP()
                dn = ANFIS(Tin,Gin)
                learnwT()
                dn = ANFIS(Tin,Gin)
                learnInput()
                dn = ANFIS(Tin,Gin)
```

Appendix E: DVD data

The following data is located on the attached DVD:

- Electronic copy of dissertation
- Video clips of experimental demonstration
- Data collected from MPPT in excel format
- MPPT simulation software
- MPPT control software
- Learning algorithm software
- PV panel simulation software