DEVELOPMENT OF SOLAR ELECTRICITY GENERATING PERFORMANCE MONITORING SYSTEM

By

SERDAR ILYASOV

FINAL PROJECT REPORT

Submitted to the Department of Electrical & Electronic Engineering
in Partial Fulfilment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

DEVELOPMENT OF SOLAR ELECTRICITY GENERATING PERFORMANCE MONITORING SYSTEM

by

Serdar Ilyasov

A project dissertation submitted to the
Department of Electrical & Electronic Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:

_____

Assoc. Prof. Dr. Balbir Singh Mahinder Singh
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

Serdar Ilyasov

ABSTRACT

The overall energy demand of the world is growing rapidly year by year. But electricity is still mainly generated by utilizing fossil fuels such as natural gas and coal. The emissions from combustion of fossil fuels have led to serious environmental and climatic problems. Implementation of renewable energy sources could solve these problems. Solar radiation is one of the renewable energy sources which have been utilized to produce electrical energy. However due to the low efficiency and high cost of solar panels and intermittent nature of the solar irradiance, it is not considered as a most reliable way to produce electricity. That is why a reliable monitoring system is required to solve the issues. This project intends to develop a reliable solar energy generating performance monitoring system. This paper highlights the types of solar panels available in the market and compares their efficiency and reliability. Also, it analyzes the components of the solar electricity generating system. The methodology to be used in this project includes identification of the parameters affecting the performance of solar panels. Moreover, meteorological and environmental factors which influence the efficiency of PV panels are investigated. By implementing the findings of experiments, the electrical components needed to build the hardware circuit are identified and programmed. Software part of the system is designed and implemented. Eventually, complete monitoring system is designed and tested. The project is the Final Year Project of a Bachelor of Engineering (Hons) Electrical & Electronics Engineering student at Universiti Teknologi PETRONAS.

ACKNOWLEDGEMENTS

First of all, I would like to express my appreciation to my parents. I have always been and will be grateful to my family for their support not only during university years, but during the whole course of my life. They are always very happy for my smallest achievements and are extremely supportive in my difficult times. My parents made me the person I am today and they are the major contributors to all my accomplishments.

I would like to express my sincere and utmost gratitude to my supervisor, Associate Professor Dr. Balbir Singh Mahinder Singh, for his guidance and help throughout the course of this project. The idea of the project that was proposed to me by my supervisor was very interesting and challenging one to be working on. His experience and suggestions were very helpful and played a vital role in successful completion of the project.

Moreover, I want to express my highest appreciation to a number of friends of mine; especially Raingsei Rhino, Erkin Ahundov, Annadurdy Sahetdurdyyev for their help during the course of this project. I approached them several times with requests, and they always found time and effort to assist me.

I would also like to thank my girlfriend, Natthika Siriwat, for her help, support and understanding in these busy and difficult times during the course of the project.

Also, I was lucky enough to have a large number of friends in this university who made my university years interesting and joyful. Though most of us are from different countries, I will remain with most of them in touch in the future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SEGS	Solar Energy Generating System
PV	Photovoltaic
FYP	Final Year Project
LCD	Liquid Crystal Display
IDE	Integrated Development Environment
VB.Net	Visual Basic.Net
I/O	Input/Output

CHAPTER 1
INTRODUCTION

1.1 Project Background

Throughout history, innovations and novelties have fueled the development and progress of technology, which resulted in even further energy and time saved for even further improvements. One of such examples is the invention of electricity. Nowadays, the electricity is not a luxury as it was previously, but an everyday necessity. More demand in electricity resulted in creation of new technologies for generating electrical energy. In order to provide sufficient energy for public and reduce the pollution, the attention shifted from conventional fossil fuels to renewable sources of energy. The list of renewable energy sources includes wind, tides, rain and sunlight. The name "renewable" means that it comes from the natural resources, thus, it is naturally replenished. **[1]**

Table 1 Top 5 countries utilizing renewable energy resources in the year 2011 **[2]**

	Renewable power capacity (incl. hydro)	Renewable power capacity (not incl. hydro)	Renewable power capacity per capita (not incl. hydro)[2]	Biomass power capacity	Geothermal power capacity	Hydropower capacity
1	China	China	Germany	United States	United States	China
2	United States	United States	Spain	Brazil	Philippines	Brazil
3	Brazil	Germany	Italy	Germany	Indonesia	United States
4	Canada	Spain	United States	China	Mexico	Canada
5	Germany	Italy	Japan	Sweden	Italy	Russia

	Solar PV capacity	Solar PV capacity per capita	Wind power capacity	Solar hot water/heat capacity[1]	Solar hot water/heat capacity per capita[1]	Geothermal heat installed capacity	Geothermal direct heat use[3]
1	Germany	Germany	China	China	Cyprus	United States	China
2	Italy	Italy	United States	Turkey	Israel	China	United States
3	Japan	Czech Rep.	Germany	Germany	Austria	Sweden	Sweden
4	Spain	Belgium	Spain	Japan	Barbados	Germany	Turkey
5	United States	Spain	India	Brazil	Greece	Japan	Japan

Table 1 depicts the list of five leading countries in certain areas of renewable energy sources utilization. These countries understand the importance of implementing available renewable resources. Solar energy has been used by our ancestors since ancient times. No doubt that the current technological progress enables us to utilize the solar energy more than before. Solar heating and solar photovoltaic are the most developed and interesting technologies in the area of utilization of solar energy **[2]**. Currently, some countries such as Germany, Italy, Spain and Japan are building solar farms in the rural areas, where solar panels installed occupy a big area of land **[2]**. It is critical in this case to have a reliable monitoring system to control the performance of equipment. If one or more solar panels fail to generate sufficient electricity or start malfunctioning, the monitoring system will be able to detect them, so that further actions could be taken. Figure 1 lists top ten countries which use solar PV panels to generate electricity.

The significance of having a monitoring system is emphasized by the fact that the input solar irradiance utilized by photovoltaic panels is not stable throughout the day time; not mentioning the night time when there is no sunlight available. The fluctuation of the input creates unwanted effects on output of the solar electricity generating system.

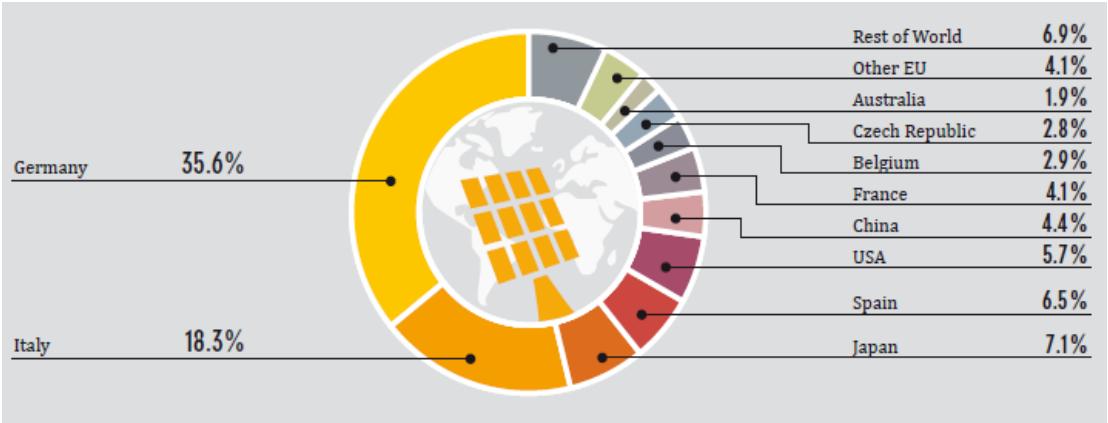


Figure 1 Top 10 countries using solar PV in the year 2011 **[2]**

1.2 Problem Statement

Generating electricity from solar radiation is considered as a clean way to produce power required as compared to conventional electricity generation by using fossil fuels. Every time the combustion reaction of fossil fuels and oxygen takes place, it emits greenhouse gases into atmosphere. Producing photovoltaic (PV) modules, on the other hand, is the only time when carbon footprint is established in solar electricity generation. Moreover, the solar radiation is available for free daily. Though this type of renewable energy has many advantages, there are still disadvantages which limit the potential growth of usage of this technology. One of the reasons is the cost and fragility of solar panels. If there is a malfunction in the system and if it is not detected on time, the solar panel can get damaged. Furthermore, the fact that the sunlight is not available during night time creates another issue. Even during the day time, the solar radiation, which is the input to the photovoltaic panel, is not stable, but fluctuates due to meteorological and other reasons. In this case, a reliable monitoring system or device would resolve the problem enabling the owners to identify the problem. Moreover, it would be even better if the monitoring device could be portable.

1.3 Objectives

The objectives that are aimed to be achieved upon the completion of this project are as follows:

i. To study the factors that affect the performance of PV based solar electricity generating system (SEGS);

ii. To design an effective SEGS performance tracking and monitoring system.

1.4 Scope of Study

This project will be focusing on monitoring of solar electricity generating system (SEGS). In order to do so, initially a literature review will be conducted on solar power and solar generating systems, as well as on available monitoring systems to understand the requirements of the future work. Moreover, initial assessment and experiments regarding output characteristics of the PV array under different conditions will be performed to study the factors affecting the performance of solar panels. The results of experiments will provide author with information needed to select the sensors to monitor the desired outputs. The hardware circuit will be designed to provide the physical presence to the project. Atmel microcontroller based Arduino Uno development board is planned to be employed in the circuit. After that, the software program will be created in order to control the hardware circuit. The program will be written in C language. Tests and simulations will be carried on to verify readings. And finally, the device will be implemented in the SEGS.

1.5 Project Relevancy

There is a rising interest in power generating systems which apply renewable sources. These types of power generating systems are very beneficial, because they do not produce any pollutants which can harm the environment and reduce the usage of fossil fuels. Solar radiation is one of the renewable energy sources which have been utilized to produce electrical energy.

This project is relevant, as it aims to develop a performance tracking and monitoring system to monitor and analyze operation of the solar panels in the SEGS. It would benefit in case of failure of solar panels, so the unnecessary spending would be avoided. Moreover, it would contribute to the general development of solar utilization technology in case of successful project completion.

1.6 Feasibility of the Project within the Scope and Time Frame

First stage of the project will be initial testing and assessment of the solar panels, where the output in terms of voltage and current will be observed and recorded during a daytime within 7 days period. It will help to understand the relationship between availability of the solar radiation and the performance of solar panel. Thus, it will be the basis for further research. After understanding what type of output factors to measure, the next stage of designing the actual circuit and controlling program will be implemented. Later on, simulations will be performed and the system will be tested in reality. Eventually, the results will be recorded and discussed. Thus, this project is expected to be completed within FYP 1 and FYP 2 time frames.

CHAPTER 2
LITERATURE REVIEW

2.1 Solar Electricity Generating System

Solar radiation is available in abundance and many consider it as a future source of energy which could solve the problem of depleting resources of fossil fuels. Moreover, there is a positive environmental impact of using solar irradiance to produce energy, because basically it has a low carbon footprint; thus only a small amount of pollutants is emitted into atmosphere compare to the pollution produced by energy generating plants implementing fossil fuels. So, the currently existing challenge is to develop technology which would maximize the utilization of this source of energy. **[3]**

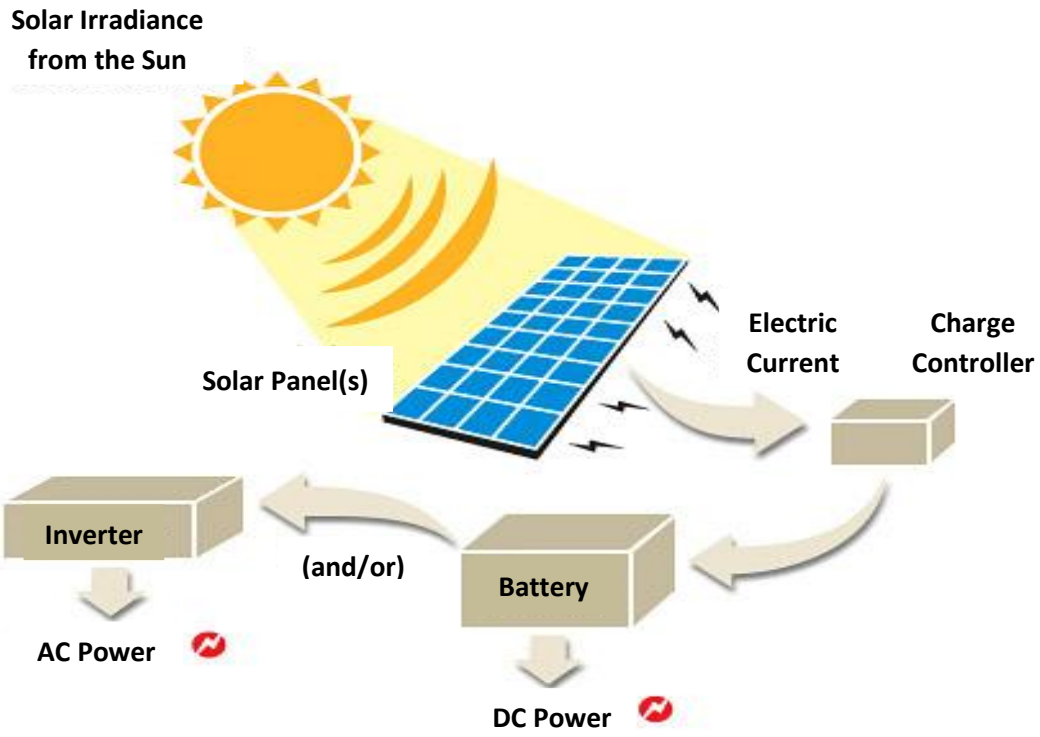


Figure 2 Set-up of solar electricity generating system **[3]**

The basic set-up of solar electricity generating system is depicted in Figure 2. As can be seen, it features equipment such as solar panels, charge controller, inverter and batteries. This is the standard configuration of the system; however, it may differ depending on the requirements and the special needs of the designers and consumers. Moreover, the sizing is another factor that can be altered in order to meet requirements.

2.2 Solar Resource

As our planet rotates around the Sun, the intensity of solar light striking any region of Earth also changes relatively. Sun actually provides very huge amount of radiation to the areas around it, which includes Earth as well. The constant for radiation from Sun is given to be 1353 W/m^2. However, the radiation from the Sun that strikes the Earth's surface is not the same as the radiation striking the atmosphere, because after entering the Earth's atmosphere it is suppressed. The reasons for suppression are absorption, scattering and reflection of electromagnetic waves by the particles in atmosphere. **[4] [9]**

The scattered radiation is named diffused radiation. The radiation that passes through atmosphere without any interference is called beam radiation. And the reflected radiation is the one which gets reflected by the ground. The global radiation can be found by adding three radiations: diffused, beam and reflected radiations. The highest amount of radiation that can be received on the surface of the Earth is approximately 1000 W/m^2, and it is usually obtained on a clear sky day. **[3] [9]**

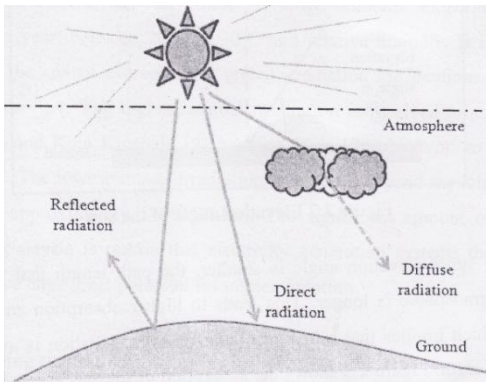


Figure 3 Radiation path through atmosphere **[5]**

2.3 Factors Affecting the Efficiency

When we decide the location where to place the solar panel we must make sure the location is proper. To decide how proper the location is, there are several factors that must be considered. These factors include availability of solar energy, meteorological conditions of the area, cloudiness of the area, angle of inclination of the panel and so on. These factors should be considered with extreme accuracy to get the most effective and efficient electricity production. **[7] [8]**

The availability of solar energy is the first consideration that needs to be analyzed. The availability of sunlight depends on the geographical location. Many areas receive a lot of sunlight during the day, while some receive extremely few amounts. The areas closer to the equator have more intense sunlight than the regions of Earth far from equator. Some areas even have no sunlight for several months, like two poles of Earth. It means placing solar power station in lesser light striking areas, will make the station inefficient. Below you may see the sunlight intensity map of the world in Figure 4. The areas close to equator get the most intensive sunlight during the day.

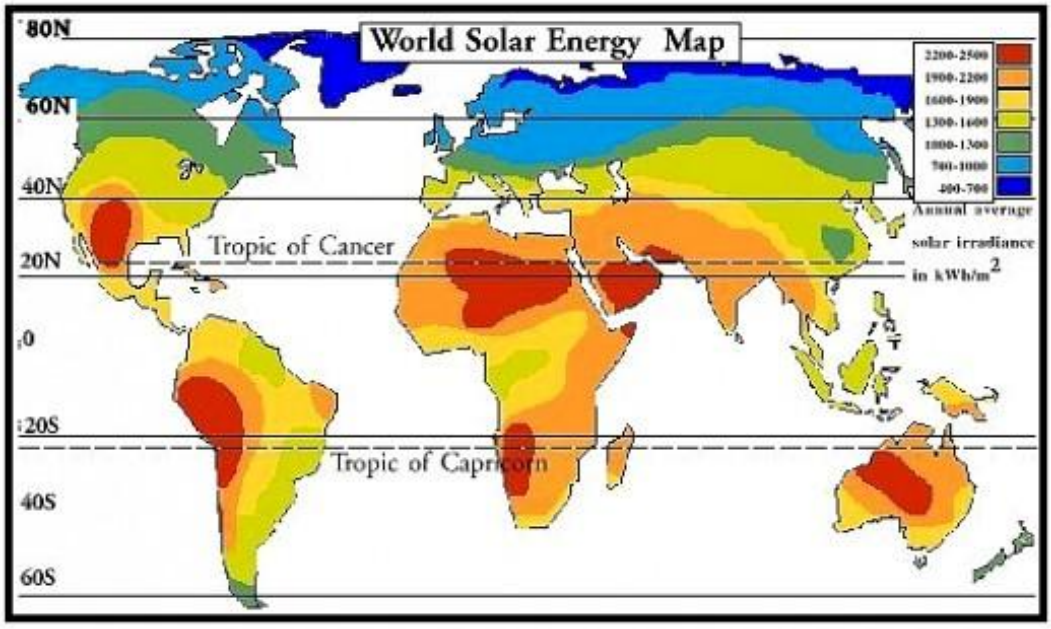


Figure 4 Sunlight intensity map of the world **[6]**

Other vital factors which affect the intensity of solar light are natural phenomena, such as rain, snow, storm and etc **[10][11].** It is very unreasonable to place solar panels in rainy areas, as these areas have less intensity of sunlight averagely during a year. For most effective production of solar energy, the direct and un-interfered light is needed. In Malaysia, unfortunately, the rate of raining is high, which means the electricity production will be lesser during the rain. The rain will make the power station inefficient.

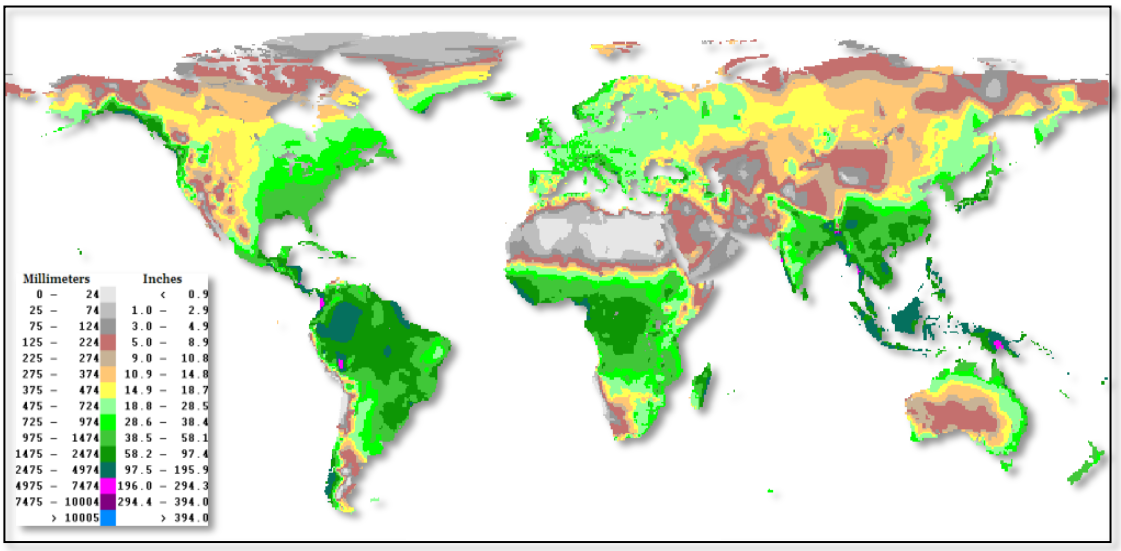


Figure 5 Total annual rainfall **[6]**

From Figure 5 above it is possible to observe that Latin America, Central and South parts of Africa, South and South East Asia, and Europe; all have significantly high rate of rainfall, which means the power production by solar panels in these areas will not be as much as the ones in for example North Africa or Australia. Cloudiness also affects the sunlight intensity. Indeed it is one of the most important factors. Even if the power station is located on equator, you might not get expected power from the plant if the area is cloudy for most of the day **[12].** We need to make sure the area is not very much cloudy, especially during the hours when sunlight has the most intensity, which is from 9 am until 4 pm **[14].** Again geographical location of the plant is very critical.

2.4 PV Hierarchy

The PV hierarchy starts with the basic cell. The cell is the fundamental component of the solar panel, and it is mainly Si atoms. Based on the arrangement of atoms in the structure, the cell is divided into several types. The arrangement of the cell decides the efficiency of the structure as well. Below is the general structure of the photovoltaic cell.

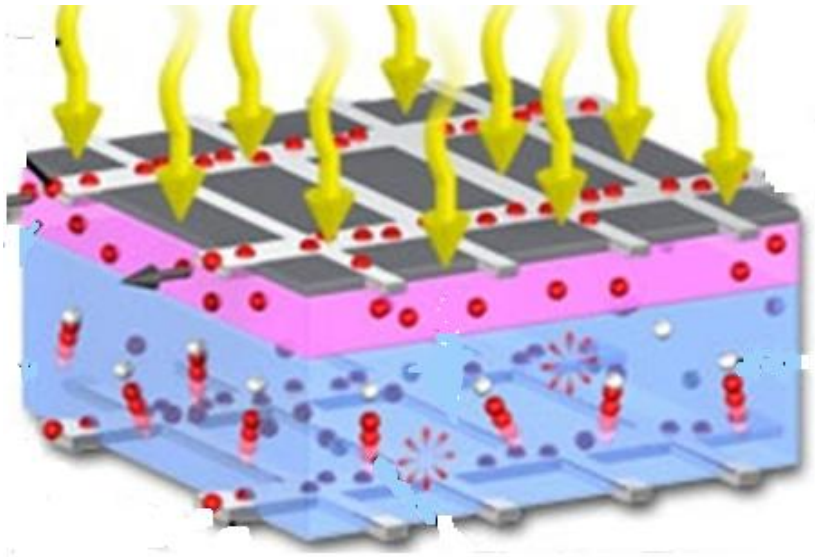


Figure 6 Photovoltaic cell **[13]**

The collection of vast amount of silicon cells make up a system called module. These modules can generate electrical energy from the solar rays hitting them. The output of such modules can be used to supply appliances with power. It can be said that the modules are elementary product from which solar energy can be used for appliances.

When several modules of solar panels are connected to each other to form a bigger panel is called array. Arrays can be several meters in cross section. Arrays are widely used in large scale application, where several thousands of modules can be connected together to form dozens of arrays. Below you may observe the hierarchy of photovoltaic cell.

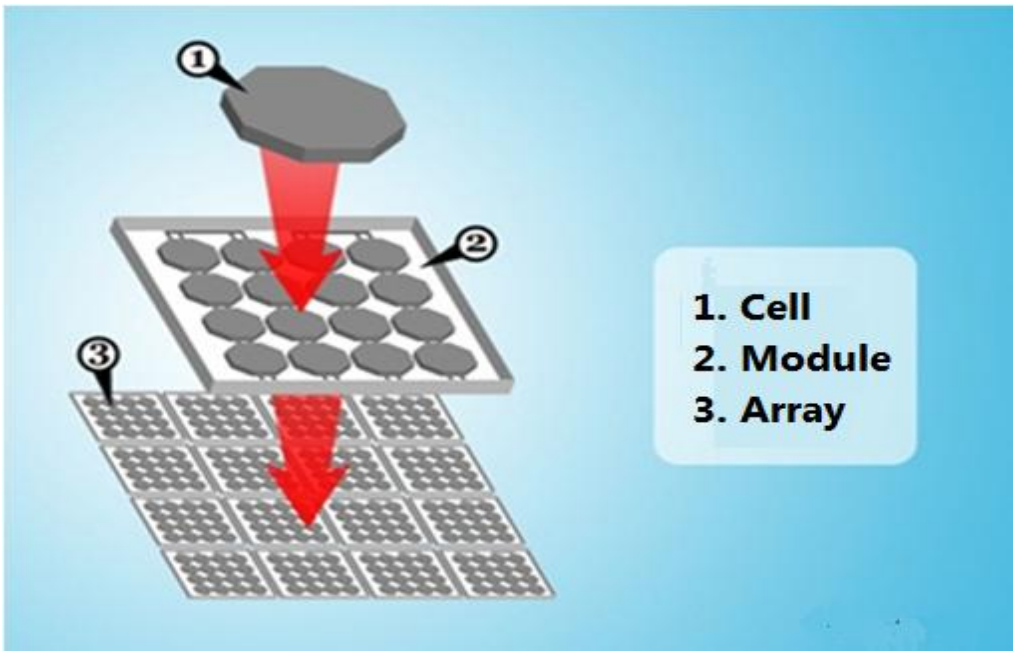


Figure 7 Photovoltaic hierarchy **[15]**

Figure 7 summarizes and depicts the PV hierarchy. The cells inside the modules as well as modules inside the arrays can be interconnected in series or in parallel between each other, or even in combination of in series and in parallel connections depending on the required specifications of the output and properties of the panel.

2.5 Types of Solar Panels

There are various types of solar panels that are in manufacture nowadays. They are divided into 3 main groups based on the structure of the panel. Those types are:

- Monocrystalline Solar Panels
- Polycrystalline Solar Panels
- Amorphous Solar Panels

There are several different types and techniques of designing the structure of the panel, like: crystalline Si silicon, CuInGa solenoid, Cd telluride solar cell, thin film, GaAr gallium arsenide multijunction, DSSC – light absorbing dyes, QDSC – quantum dot solar cell, polymer solar cell and etc. Although each of them has different theories and techniques of design, they are part of above three types **[13].** The technique that costs the least amount of money is amorphous solar panel type. It has very simple design steps and cheaper materials than other two types. They are usually manufactured by inducing a film with a conductor attached to it to the glass. After that the structure is put into a frame that can be either plastic or metal. The technique is very useful for low intensity conditions, as they provide the best performance in this condition. Crystalline structure solar panels use silicon as the main component of the panel. Silicon solar cells are attached to each other in certain geometry and then covered by the frame made of metal material. These structures give very good performance and are very useful in condition with high intensity sunlight. For this reason crystalline structure solar cells are effectively and vastly used in the off-grid and grid-tie solar panel systems **[10].**

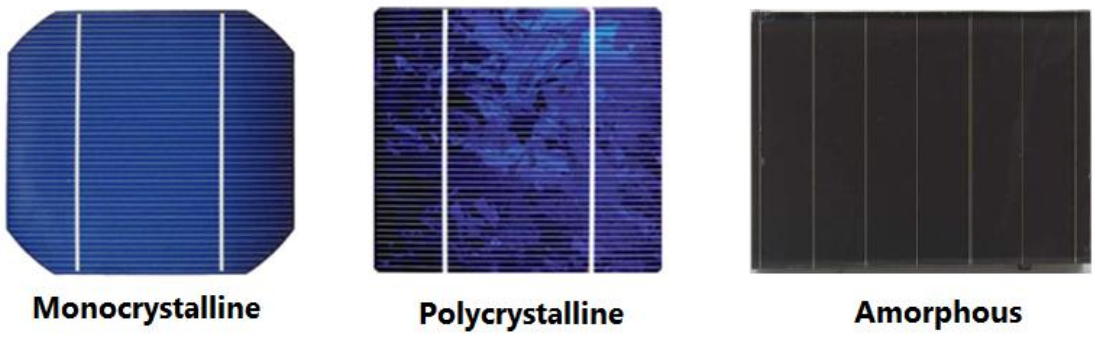


Figure 8 Different types of solar panels **[11]**

2.5.1 Monocrystalline Solar Panels

Monocrystalline solar panels are derived from special silicon wafers. These wafers are cut from Si crystals. Crystals are not natural, as they are formed under special conditions to suit the design and demands of the solar panel structure. The length of the wafer is several centimetres and several of those wafers are aligned to form a panel. The main advantage of Monocrystalline Solar Panel is that they have higher efficiency than other types of panel. The disadvantage of the structure is that the cost of this type of solar panel manufacture is very high, and unless the project is huge, it is not advisable to use Monocrystalline structure of solar panel. **[10]**

The crystals used in monocrystalline structure are very pure. The purity of the crystals is the main factor behind the high efficiency of the panel. The more pure the crystal, more efficient it is. Monocrystalline solar panel crystals are mainly of colour blue. The alignment of the crystals in this structure is very linear. That is why extreme care and precautions are required to sustain the product with efficient output power. **[13]** Monocrystalline modules are the most effective among all. They give the efficiency of 11% to 14%, making the cell most desired in many situations **[15]**. The cells in these structures do not fully fill up the space, as the shape of the cells is circular. So some space is spared in these modules.

2.5.2 Polycrystalline Solar Panels

Polycrystalline structure also in form of wafers of silicon, like monocrystalline solar panels, but the main difference is that silicon wafers do not need to be as perfectly aligned as monocrystalline. The conditions of controlling the growth of crystal do not need to be extremely careful, and for that reason this type of solar panel structure is relatively cheaper compared to the monocrystalline. In this structure crystals that are connecting the structure together grow together, unlike monocrystalline, where each interlocking Si crystal grown separately from each other **[10].** This factor significantly reduces the cost of the structure. The shortcoming of polycrystalline structure of solar panels, compared to monocrystalline is the efficiency of the panel. As the structure is less organized, it reduces the efficiency of panel by significant amount. That is why if the objective of the project for power generation is low cost, polycrystalline structure should be used, and if the objective is high efficiency of the panel, monocrystalline structure of solar panel is more suitable.

The efficiency of the polycrystalline solar panels is in the region of 10 % to 12 %, which is not very bad. But this value is lower than the Monocrystalline structure solar panels, which have efficiency of nearly 15 % **[15].** Modern approach used to measure the output power of the panel uses the constant factor that is given according to the region. If the region has the factor of 1, it means for 200 Watt polycrystalline structure 200 Watt of output power will be obtained. If the factor is 0.5, only half of 200 Watt will be obtained. The simple structure of polycrystalline Solar Panel is shown in Figure 9.

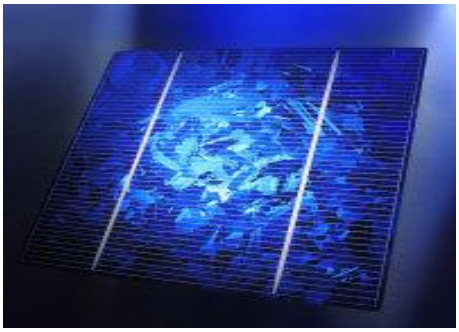


Figure 9 Polycrystalline type solar panel **[11]**

2.5.3 Amorphous Solar Panels

Amorphous type of solar panel is constructed from amorphous silicon. The main difference from crystalline structure is that the silicon atoms are not ordered in net like shape, as they are in crystalline structures. Another feature of the amorphous structure is that the production of this panel structures are not implemented by growing the atoms of Si, but instead Si atoms are carefully put on a thin layer of the substratum. The layers of silicon on the substrate are doped with different compactness so that the structure responds differently under various sun intensities **[10].** That improves the efficiency of the panel. It is very complicated manufacture technique, but they are very useful in low intensity conditions. That is why they are produced massively nowadays and the price decreases fast. The advantage of thin layer of Si atoms on the substrate has a certain benefit. It helps the panel to be flexible, so it is more adaptive to the environment that panel is used in. In case the

panel is used on the roof of the building with curvy surface, the panel will suit perfectly; as it can be bend on nonlinear areas.

The amorphous modules are least efficient among all, as they have the efficiency of 6 % under perfect conditions **[15]**. The main advantage is that the module is effective under low light conditions. Amorphous type solar panels were mainly used for small scale application like the sunlight calculators, but with the time passed and the researches made, it was found that the panel is also effective for large scale applications. So the mass production of amorphous solar panels is underway nowadays. Moreover, the amorphous type solar panels are more flexible compare to other types.

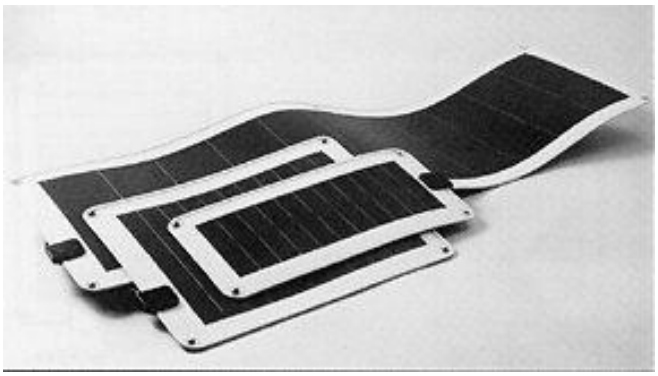


Figure 10 Amorphous type solar panel **[11]**

CHAPTER 3
METHODOLOGY

3.1 Process Flow Planning

Figure 11 describes the sequence of work load expected to be performed in order to complete the project on time. More detailed explanation of each part of the flow chart is provided below, in order to clarify the steps expected to be performed during May 2012 semester under Final Year Project 1 course and September 2012 semester under Final Year Project 2 course.

3.1.1 Research Methodology

In general, after selecting the topic for the Final Year Project the research is conducted starting with literature review on solar energy and available technologies utilizing solar energy. Literature review aims at increasing the awareness about the research topic as well as updating the author on the current state of developments in the research area. It is done through analysis of related articles in journals, books, conference papers, project papers and all other useful references.

3.1.2 Initial Assessment

In this part of the research several experiments are performed in order to verify the information obtained in literature review part. It should enable the author to get more hands-on experience and knowledge regarding performance of photovoltaic panel. The data gathered in this part will be implemented throughout the rest of the project period as a basis for further sections. Thus, it is necessary to pay more attention to correctness of the procedures and data obtained, so no mistakes would arise in the future stages.

3.1.3 Identification of Parameters and Components

After successful completion of the literature review and initial assessment parts, the information obtained should be sufficient to identify the parameters which should be monitored and analyzed. Thus, the components and sensors needed for hardware circuit are chosen according to the parameters of the solar panels to be monitored.

3.1.4 Circuit and Program Design

This is arguably the longest part of the project, where the actual electrical circuit should be compiled from the components identified in the previous step in order to give a physical presence to the project. Moreover, all these components should be integrated with a microcontroller in order to obtain the readings of measurements. That is why the hardware should be properly programmed.

3.1.5 Performing Simulation and Real Monitoring

After obtaining a fully functioning device, it is necessary to conduct simulations to verify the expected theoretical behaviour. Software simulators can be implemented to conduct the simulations in order to anticipate all possible issues which can appear later. Only after this step, the real implementation of the system in order to monitor the SEGS can be performed.

3.1.6 Analysis of Results

The results and data obtained while monitoring the components of SEGS is further analyzed in order to identify the areas for improvement. In this step the actual received data is compared with the theoretically expected values to determine the correctness of measurements.

3.1.7 Design Alteration & Conclusion

Eventually after obtaining the results of comparison and analysis of the data, it might be necessary to introduce some modifications to improve the monitoring system. Moreover, calibration and optimization should be performed, so that the

readings of sensors will be accurate. At the end, all the information obtained will be summarized and concluded in the reports to describe the work done and discuss the results.

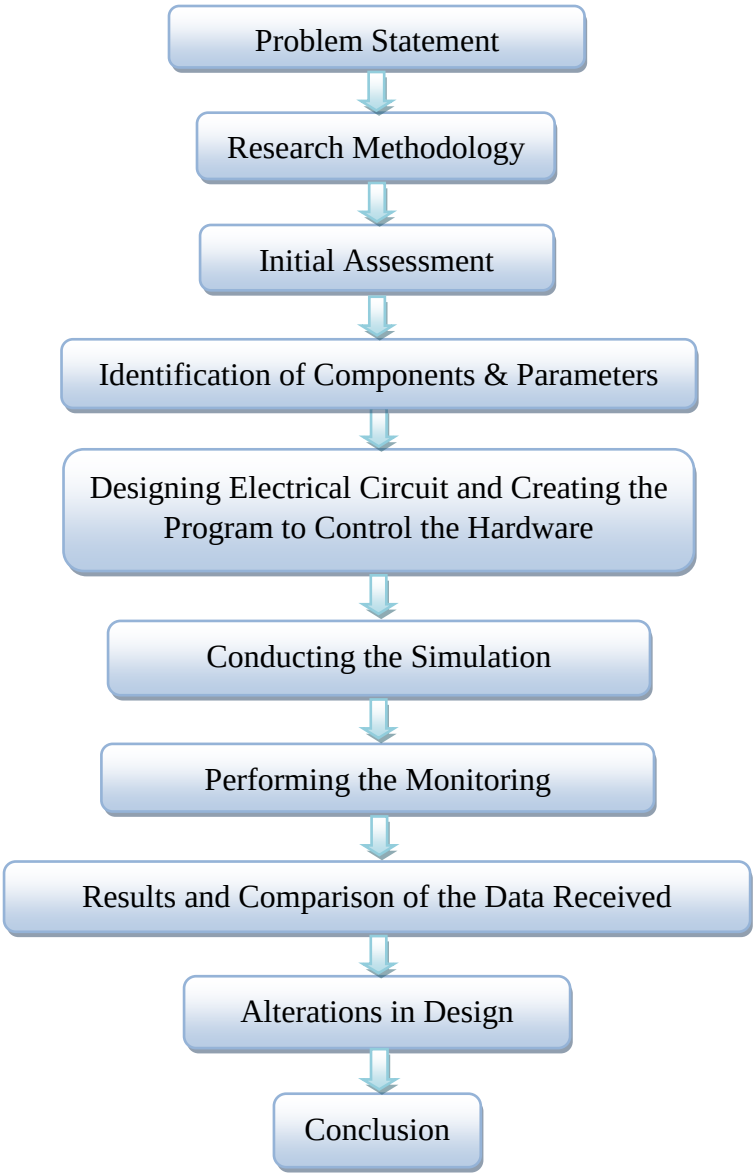


Figure 11 The project process flowchart

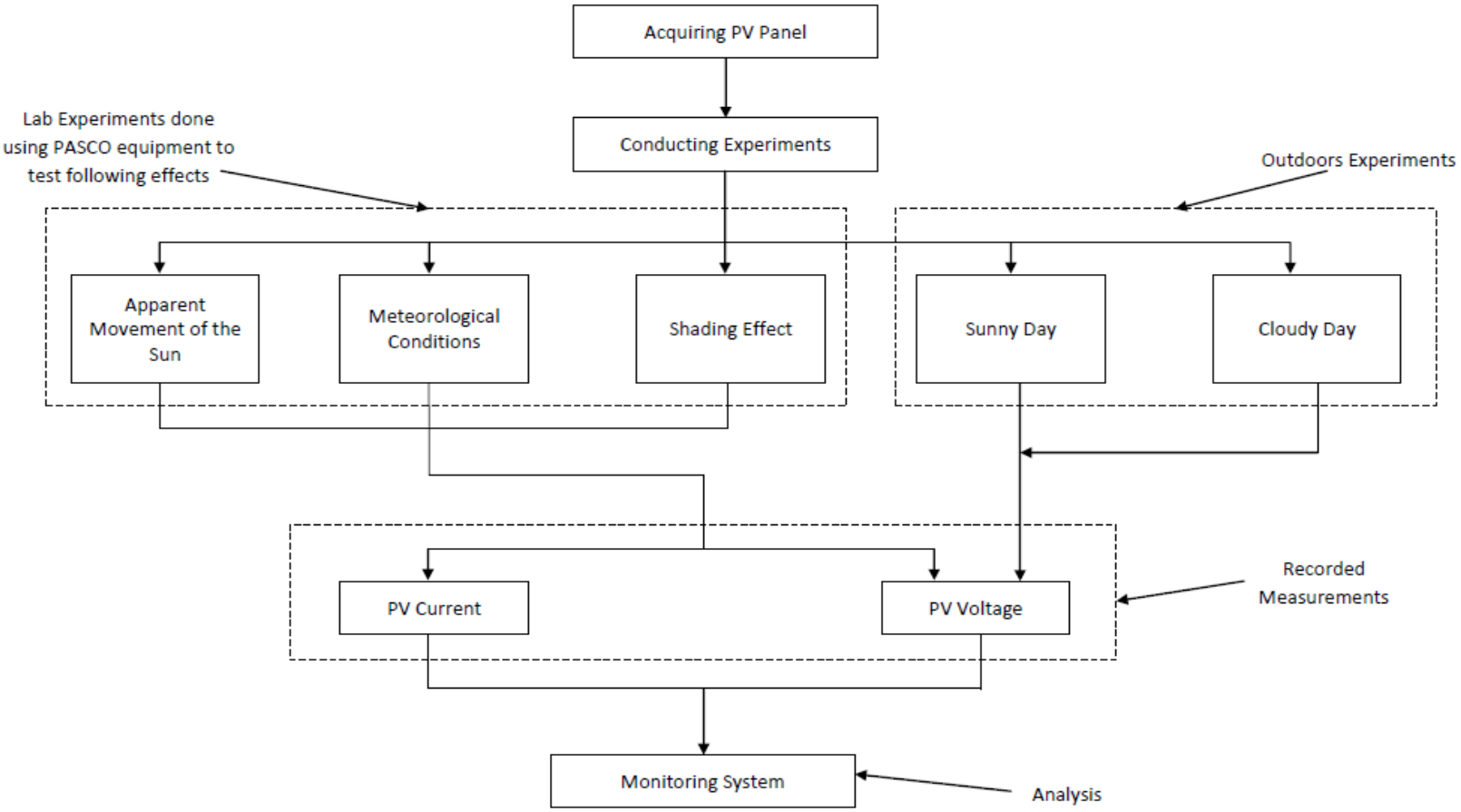


Figure 12 Activities completed in FYP1

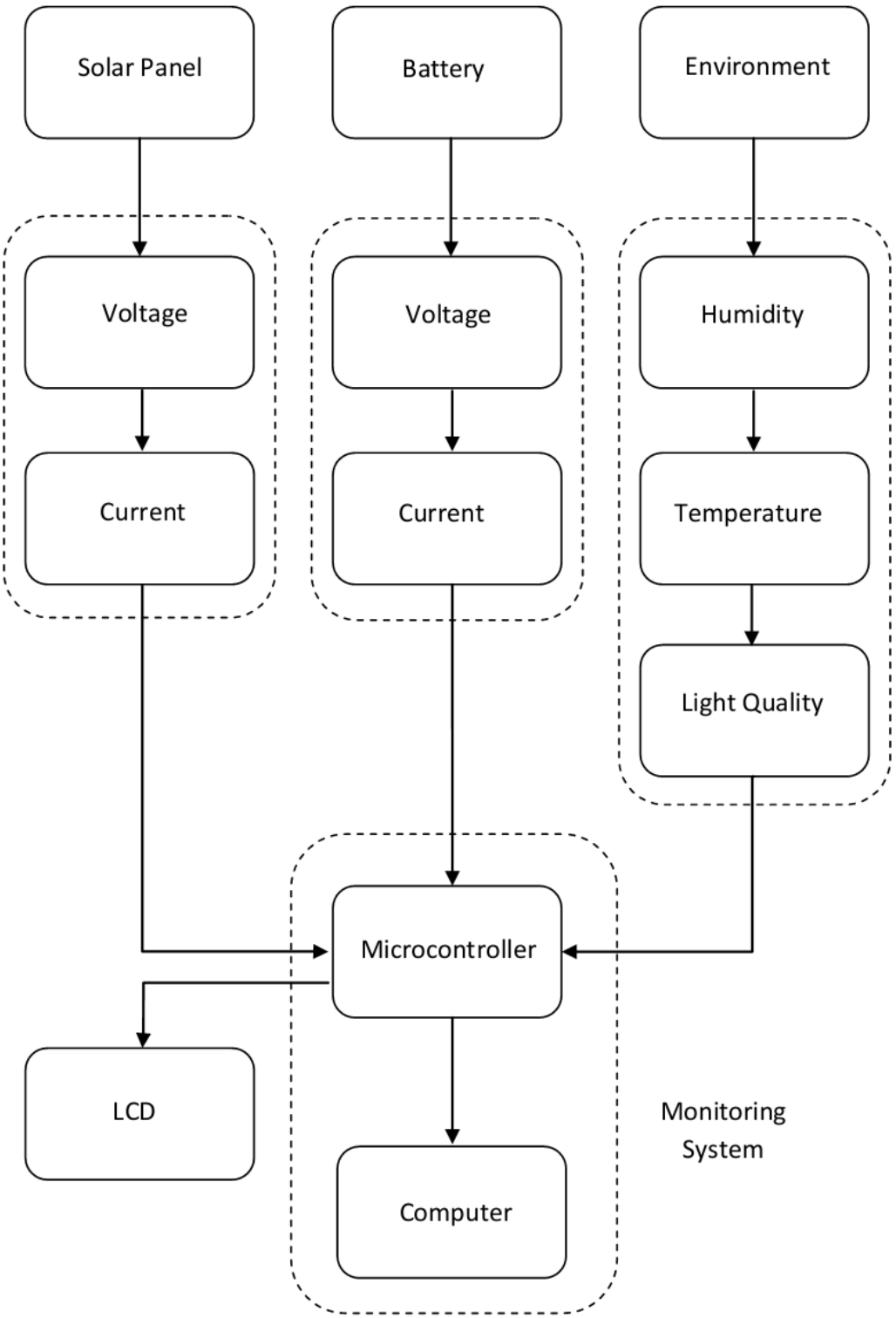


Figure 13 Activities completed in FYP2

3.2 Gantt Chart and Key Milestones

Process

Milestone

Final Year Project I (May 2012)

No	Task Name	Week 1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Topic Selection								MIDSEM BREAK							
2	FYP Briefing															
3	Literature Review															
4	Submission of Extended Proposal															
5	Meeting with FYP supervisor															
6	Proposal Defense															
7	Initial Testing/Parameters & Components Identification															
8	Submission of Interim Draft Report															
9	Submission of Interim Report															

Table 2 Gantt chart for FYP1

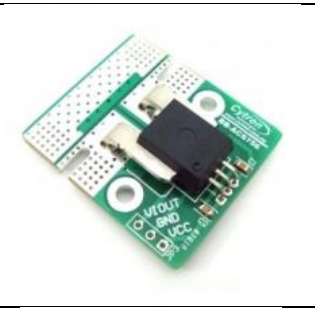
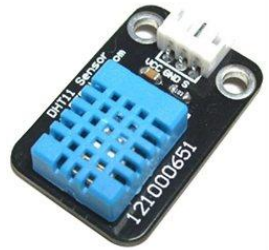
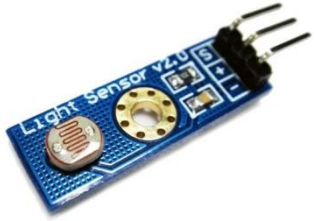
Final Year Project II (September 2012)

No	Task Name	Week 1	2	3	4	5	6	7		8	9	10	11	12	13	14	15
1	Designing the electrical circuit and creating a program								MIDSEM BREAK								
2	Progress Report Submission																
3	Pre - EDX																
4	Submission of Draft Report																
5	Submission of Dissertation (Soft Bound)																
6	Submission of Technical Paper																
7	Oral Presentation																
8	Submission of Project Dissertation (Hard Bound)																

Table 3 Gantt chart for FYP2

3.3 Tools Required

No	Name of the equipment	Short Description	Picture
1.	Solar Panel	The solar panel generates electrical energy from the solar irradiance. The actual solar panel is required in order to perform the initial assessment experiments. Moreover, this panel will be used for monitoring purposes.	
2.	Battery	Battery is needed to store the energy obtained from the solar panel. 12 Volt battery will be implemented to build the small prototype of the SECS.	
3.	Arduino Uno	Arduino Uno is the microcontroller development board which can be programmed to perform different tasks. In this project this board will collect the readings from sensors and render data to send to PC.	
4.	Computer (either PC or laptop)	Computer is needed to receive the data from the Arduino board and analyze it.	

5.	50A Current Sensor (Bb-ACS756)	Current sensor which provides accurate measurements of AC and DC currents. Can be applied in many fields of engineering.	
6.	Temperature and Humidity Sensor (DHT11)	Sensor which provides accurate and reliable readings of the temperature and humidity of the environment.	
7.	LCD Display	LCD is used to display the values which microcontroller decodes and computes from readings obtained from the sensors.	
8.	Light Sensor	Light sensor is used to measure the quality of the light supplied by the environment	

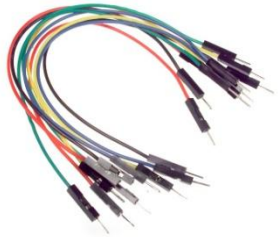
9.	Potentiometer Based Voltage Sensor	Potentiometer is used as a part of the voltage sensor which is needed to read the voltage values of battery and solar panel.	
10.	Jumpers	Jumpers are used to connect the all the components together, thus provide the media for information and energy transmission within the system.	
11.	Resistors	Resistors are implemented to lower the current and voltage coming in the I/O ports of microcontroller. Thus, resistors help to prevent the possible damage.	

Table 4 List of sensors and equipment needed

3.4 Proposed Set-Up of the System

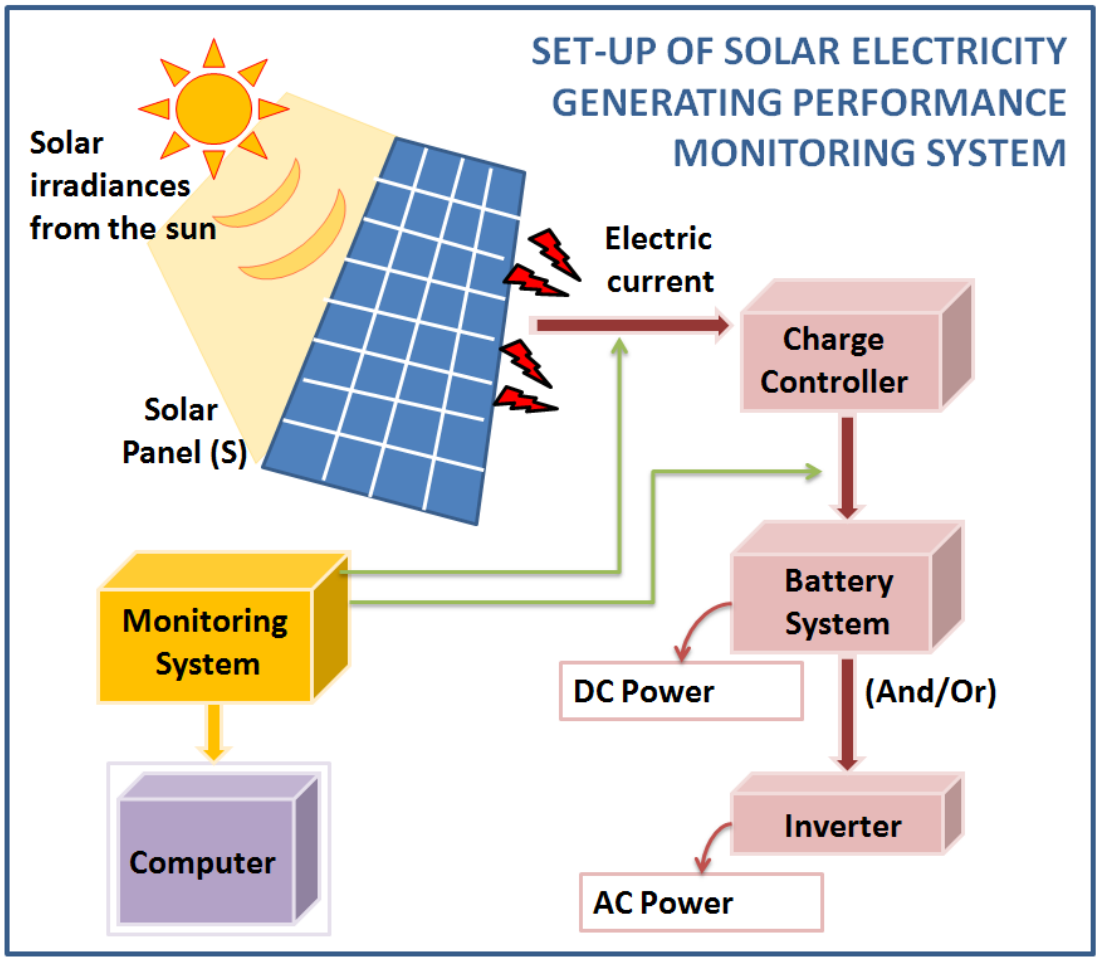


Figure 14 Expected layout of the system

As depicted in Figure 14, the system under design is expected to be able to monitor the output of the solar panel system and battery system. It would enable users to identify the changes in output of the PV panels due to the different reasons, such as environmental and meteorological factors. Furthermore, it would be able to trace the possible changes in the battery system as well due to the inconsistent supply of energy from the solar system. In general, it would contribute to the overall development of the solar energy utilizing technology.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Initial Assessment

Two experiments were performed in order to obtain data about output behavior of the photovoltaic panel under different specified conditions. The data was successfully recorded and analytical explanation based on literature review was made. Figure 15 depicts the PV panel which was implemented in both experiments and Table 5 provides the specifications of that particular panel:

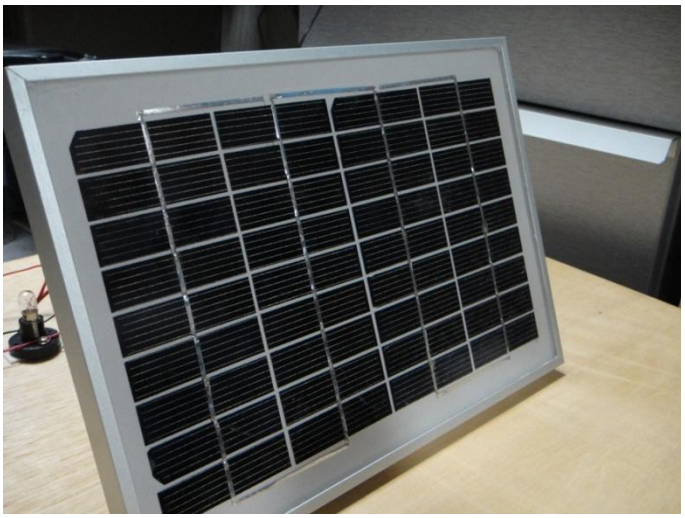


Figure 15 Photovoltaic panel used

Table 5 Characteristics of PV panel

Cell Type:	Mono
Rated Max Power:	5 W (peak)
Rated Voltage:	17.28 V
Rated Current:	0.29 A
Opened-Circuit Voltage:	21.24 V
Short-Circuit Current:	0.33 A

4.1.1 Initial Testing of PV's Voltage Characteristics on Sunny Days

First of all, it was necessary to obtain the output voltage readings of the solar panel during sunny days. The experiments were conducted inside Universiti Teknologi PETRONAS (UTP), from 23rd of June till 29th of June, 2012. This was a perfect time to measure voltage characteristics of PV panel, because there were no cloudy days within this period. It can be explained by the fact that June is considered as a dry month in Peninsula Malaysia. The output voltage was recorded every thirty minutes within above mentioned period of days between 8:00 and 18:30. It is necessary to note that PV panel was directed straight to the Sun, so the output readings recorded were maximum possible. At the end, the average values of the readings obtained at the same time during 7 days were calculated. Table 6 shows the results obtained:

Table 6 Voltage characteristics of PV on sunny days

Time (hours)	Voltage, V
800	15.21
830	16.34
900	16.92
930	17.56
1000	18.24
1030	18.48
1100	18.69
1130	18.75
1200	19.32
1230	19.89
1300	20.44
1330	20.37
1400	20.39
1430	20.29
1500	20.23
1530	20.14
1600	19.2
1630	18.84
1700	18.53
1730	18.41
1800	17.56
1830	17.13

Microsoft Office Excel 2007 software was used later in order to plot the graph of voltage output versus time. Figure 16 depicts the red line which represents the average voltage readings within seven days versus time. We can see the tendency that the voltage output increases in the first half of the day from 8:00 till 14:00 approximately and drops between 15:30 and 18:30. The maximum output was achieved between 13:00 and 15:30 every day. It means that it is preferred to adjust the solar panel in such a way, that it would face the Sun directly between 13:00 and 15:30 period of time every day.

Moreover, the blue line on the graph represents the sixth order equation which was derived to simulate the output line. The accuracy of the equation is around 98%. This equation can be used in the future to program the monitoring device, so it could predict the expected value of the output voltage readings in order to use the values for the purpose of comparison with an actual output. Later, monitoring system would be able to suggest a certain action based on the outcome of comparison made.

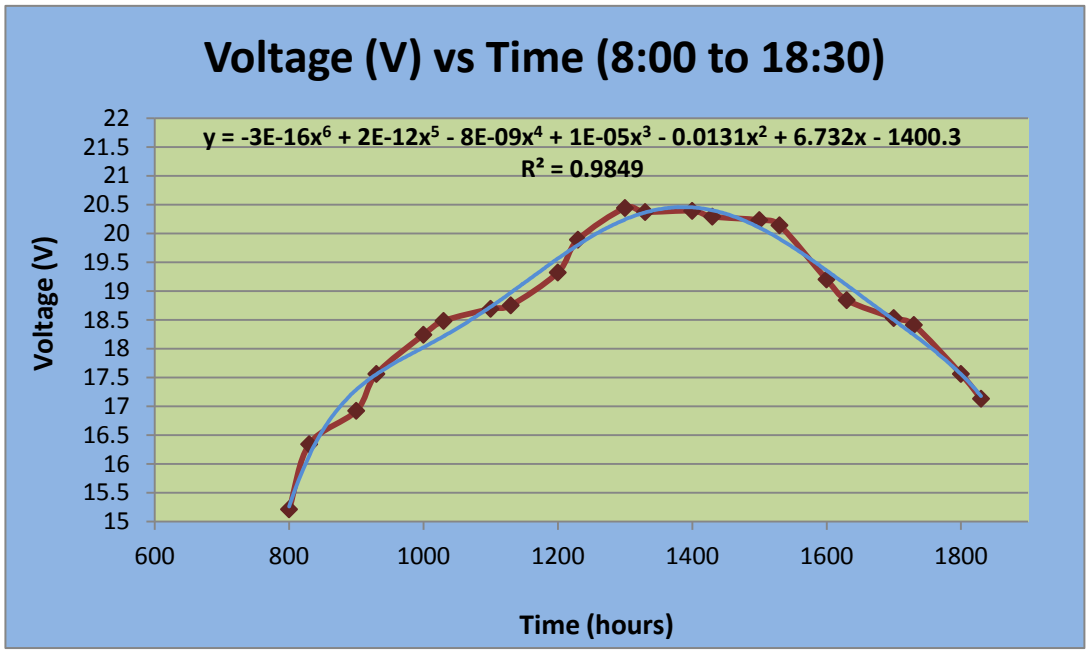


Figure 16 Voltage characteristics of PV panel on sunny days

4.1.2 Testing Environmental and Meteorological Impacts on I-V Characteristics of PV Panel

The second experiment was performed at UTP's Physics Lab at Block 19, on 10th of July, 2012. Figure 17 shows the set-up of the experiment. During this experiment the PASCO Data Acquisition System connected to the PC was used to obtain the output results of voltage and current.

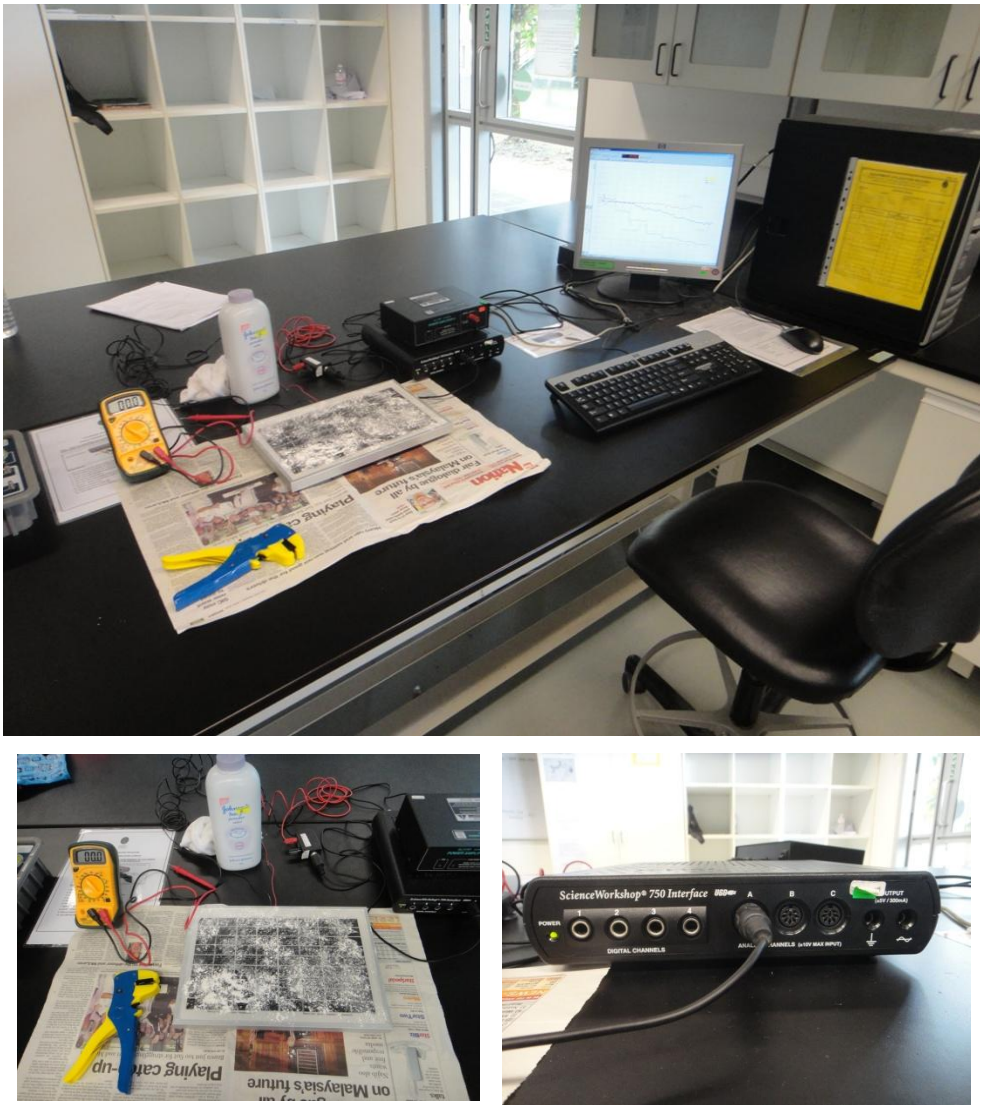


Figure 17 Set-up of second experiment

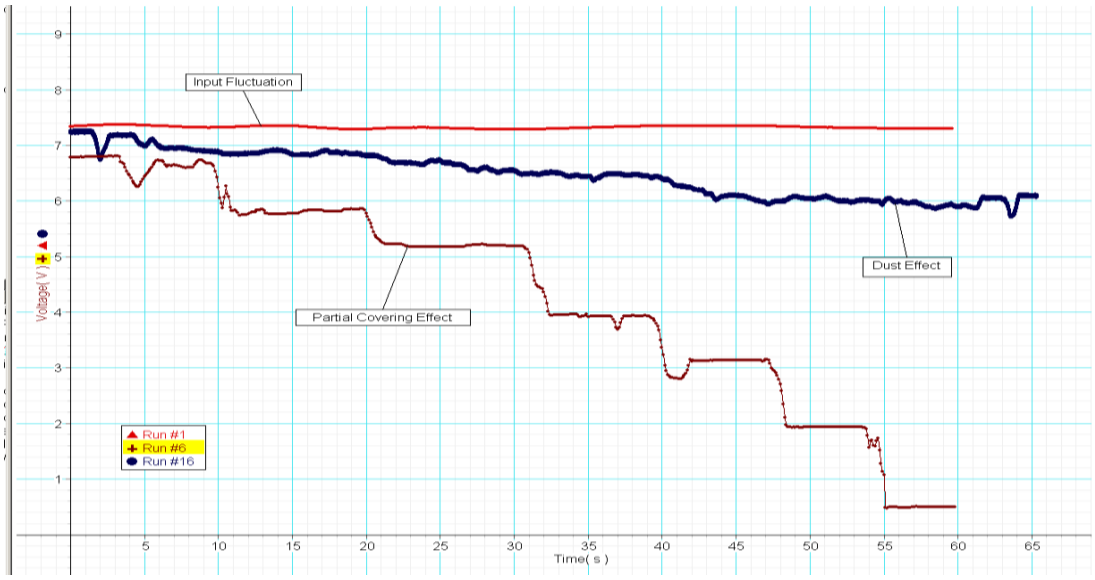


Figure 18 Opened-Circuit voltage characteristics

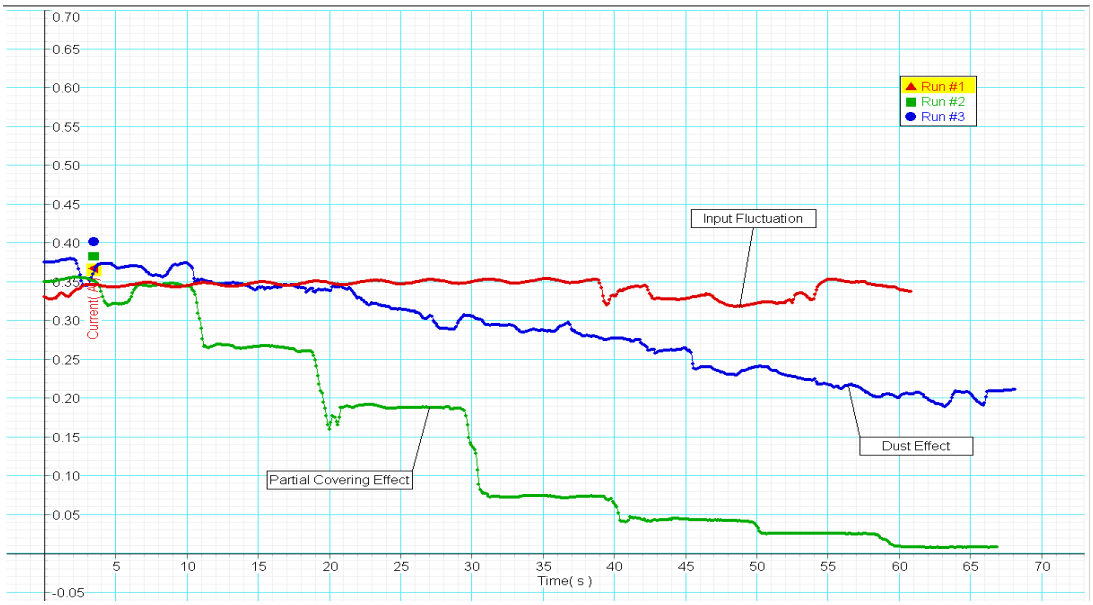


Figure 19 Short-Circuit current characteristics

Figure 18 and Figure 19 represent the test results of opened-circuit voltage versus time and short-circuit current versus time graphs, respectively. First of all it is noticed that, the reading values are low because of the fact that the experiment was performed in a laboratory when there was no direct solar beam. Instead, the irradiation supplied by the light sources inside the room was employed. Still the results obtained are sufficient to use them in order to understand the behavior of the output current and voltage under certain conditions.

As can be noticed from Figure 18 and Figure 19, there are three lines on each of the graph representing the results of three possible scenarios tested:

- ***Input Fluctuation***: the voltage and current output readings were recorded within a one minute time. No external factors were changed during that minute.
- ***Dust Effect*****:** the voltage and current output readings were recorded during one minute time while the PV panel was gradually covered by the baking flour to simulate the dust that can be accumulated on the panel during some period of time.
- ***Partial Covering Effect***: the voltage and current output readings were recorded within a one minute time while approximately 15% of the surface area of the photovoltaic panel was covered with a sheet of paper to imitate the shading effect.

As we can see from the results, even though no external factors were changed during the first experiment, the lines representing output voltage and current are not straight, proving that the fluctuation exists. From the results of the second experiment, it is obvious that the output readings decrease eventually after some time when the dust gets accumulated on the surface of the solar panel, thus blocking some cells in the panel. From the graphs obtained, we can conclude that gradual decrease of output current and voltage can reach the value between 15% and 30% due to the dust effect.

The results of the third experiment suggest that partial covering of the surface by 15% of the area results in a drop of the output characteristics by 10% to 15%. Thus shading affects the performance of the solar panel dramatically.

4.2 Electrical Circuit Design

After obtaining the results of the experiments related to the initial assessment, the factors which are needed to be monitored and analyzed have been identified. The main parameters which are affected by meteorological and environmental conditions are voltage and current of PV panel. Both of the factors directly influence the power of the energy generated by solar panel. Thus, these are the main factors which should be monitored.

These findings enabled authors to identify and select the electrical components needed to build the hardware prototype of the device. These components are listed in the Section 3.3 of this report. Figure 20 shows the schematics of the circuit with the pin connections to the Arduino Uno development board.

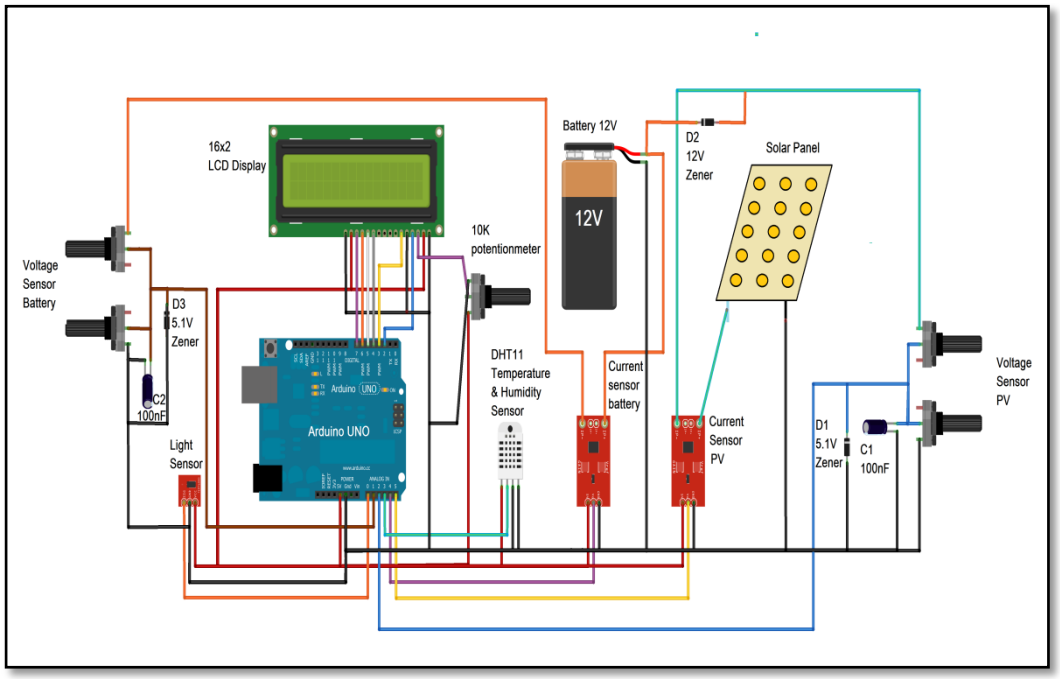


Figure 20 Schematics of proposed circuit

4.2.1 Assembling the Circuit

As we can see from the diagram displayed in Figure 21, voltage and currents readings are monitored for both battery and PV panel by implementing 50A current sensors and potentiometer based voltage sensory circuit. Moreover, humidity, temperature and light quality, which represent the environmental factors, are monitored as well to provide the users with explanation of fluctuation or change of energy generated by solar panel. The heart of the circuit is the Atmel microprocessor based Arduino Uno microcontroller board. Arduino Uno board has 14 digital input/output pins and 6 analogue input pins. The main function of the board is to collect the readings from all the sensors and decode them into data which can be further implemented. Furthermore, microcontroller sends simultaneously the readings both to 16x2 liquid crystal display (LCD) and computer. LCD displays all the values obtained and decoded, thus providing visual aid to users. Computer, on the other hand, utilizes the application for monitoring and analyzing the readings received.

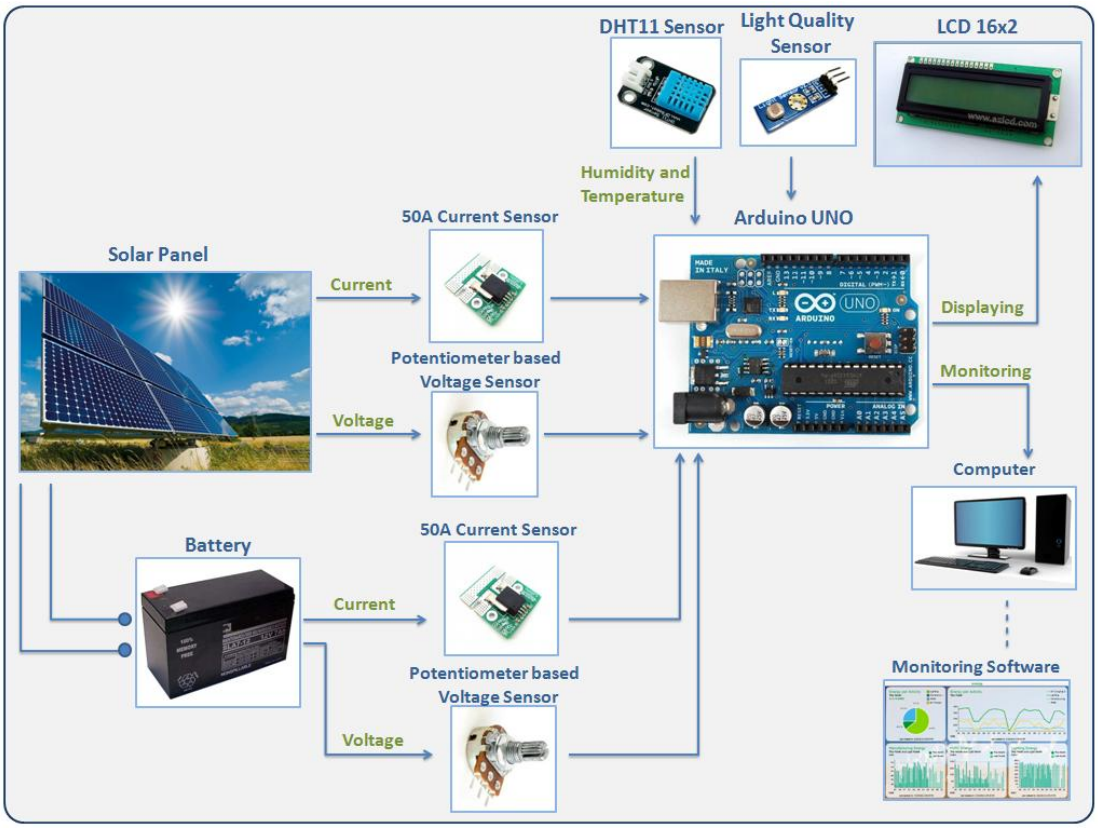


Figure 21 Diagram of set up of SEGS monitoring system

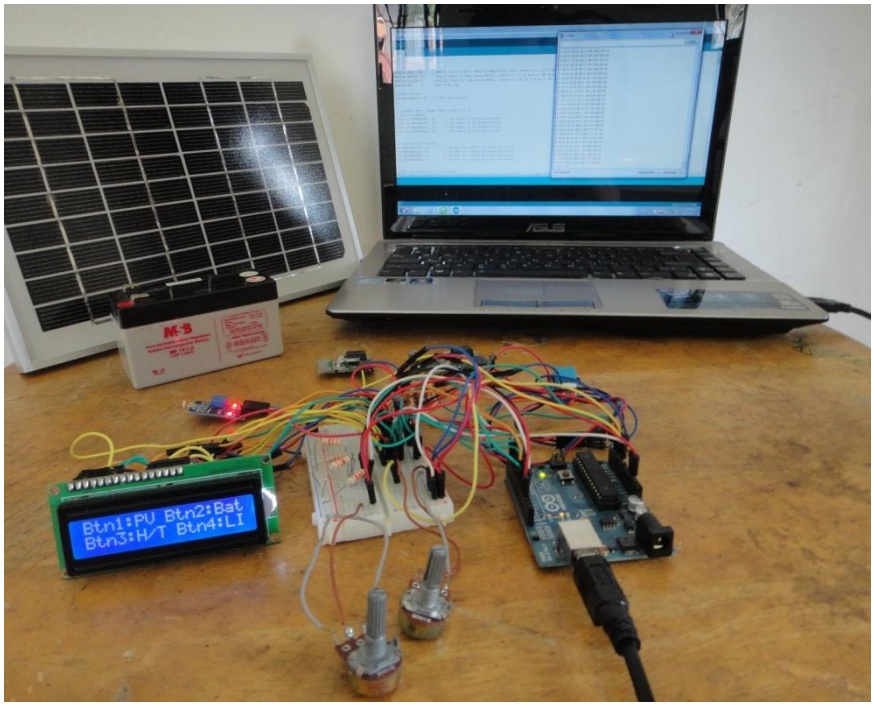


Figure 22 Components of SEGS monitoring system

Figure 22 depicts partially assembled components of the SEGS monitoring system. All the components were obtained and combined together.

4.2.2 Functionality of the Monitoring Device

The device receives the measured readings from all sensors and converts them into appropriate values in order to display these values in LCD and transfer the values to computer. The device communicates with PC by means of serial port interfacing (usually called COM port). Meaning the data can be transmitted only by one bit at a time. The device is programmed the way that it combines all decoded values for one instant of time and sends it as a string of bits to computer. However, a short delay is needed between the sets of strings to enable the applications on computer recognize and separate every string. Moreover, in order to separate the values of each factor within the string, the special character "|" is added. It simplifies the extraction and separation of values corresponding to the parameter they represent. Any COM port monitoring application can be used to observe the data received. Figure 23 shows the example of the received values for each observed parameter combined into a string and the separation of strings observed in Arduino IDE software.

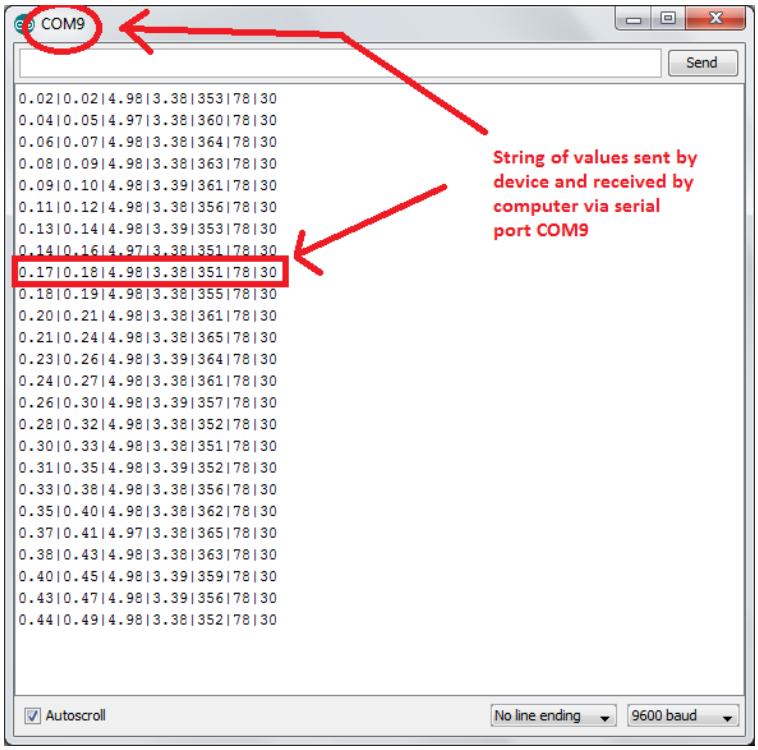


Figure 23 Example of data received by computer

The front view of the monitoring tool is illustrated in Figure 24. There are 4 black push buttons which are used to select the parameters with the real time values displayed on LCD. Initially LCD displays the hint for purpose of each button.

Figure 24 Front view of the device

If button 1 is pressed PV panel's voltage and current readings will be displayed, if button 2 is pressed battery's voltage and current readings are displayed, if button 3 then humidity and temperature values, and if button 4 then sun light intensity is displayed. All four possibilities are shown in the Figure 25.

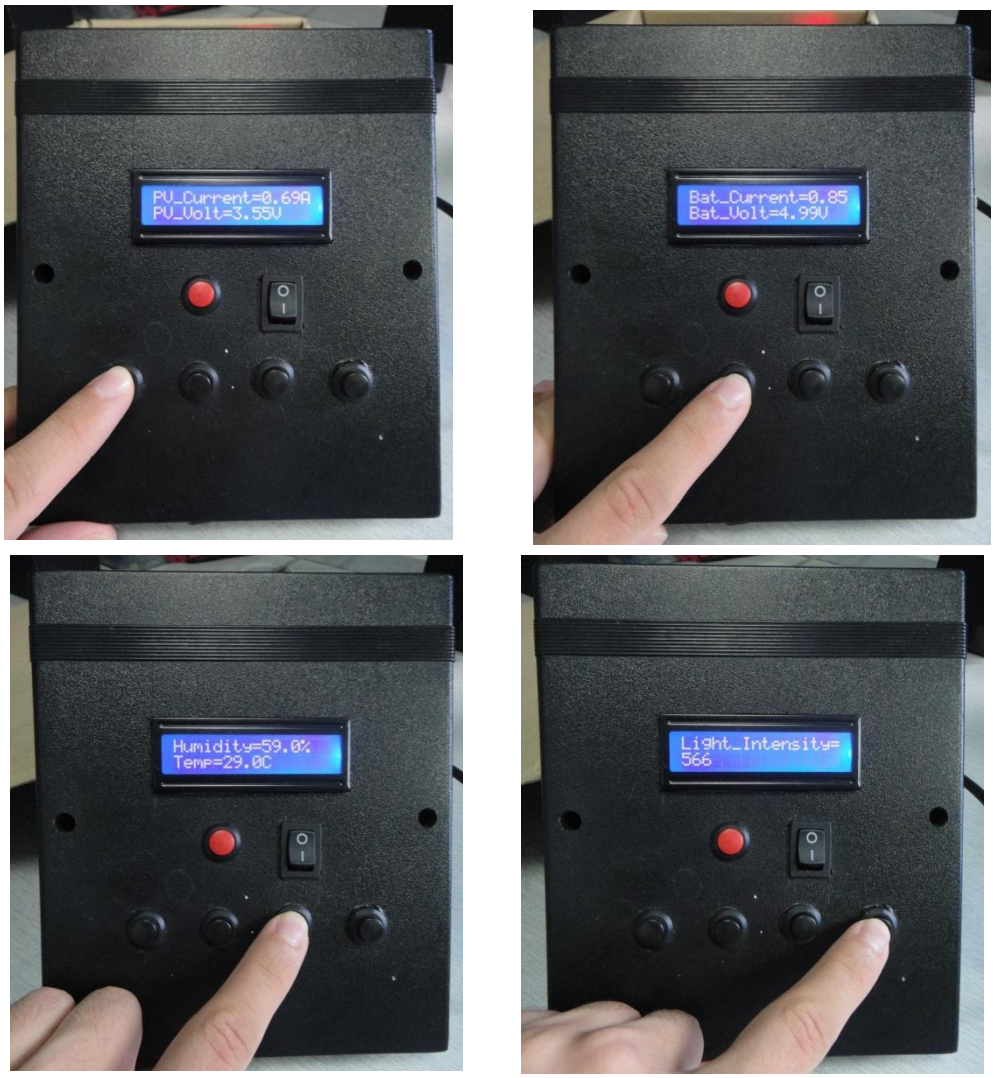


Figure 25 4 possible data displayed

4.3 Program Development and Design

After receiving measurements and converting them into string of data, Arduino Uno board transfers this data to PC using the USB 2.0 serial B type cable. Special program is required to provide the interface for devices connected to computers via serial port. The values sent to computer can be retrieved and displayed

on the screen by using that application.

The software, which could provide the communication media between computer and monitoring device, was designed in Microsoft Visual Studio 2010 IDE. The Visual Basic.Net programming language was implemented to program the code. There were several versions designed before the final version was obtained. Description of types and features of each of them is provided in following sub chapters.

4.3.1 Initial Version and Design of the SEGS Monitoring Program

Initially, the basic layout template available in the Microsoft Visual Basic 2010, combined with open source codes available in the Internet, was used to create a simple design for the program. The created design included the area for choosing and setting the configurations of COM port and the window displaying the string of values. Moreover, two buttons, "Connect" and "Disconnect", were added to control the connection of the program to the COM port, because only one program can access the particular serial port at a time. The illustration of the program's main window is shown in Figure 26.

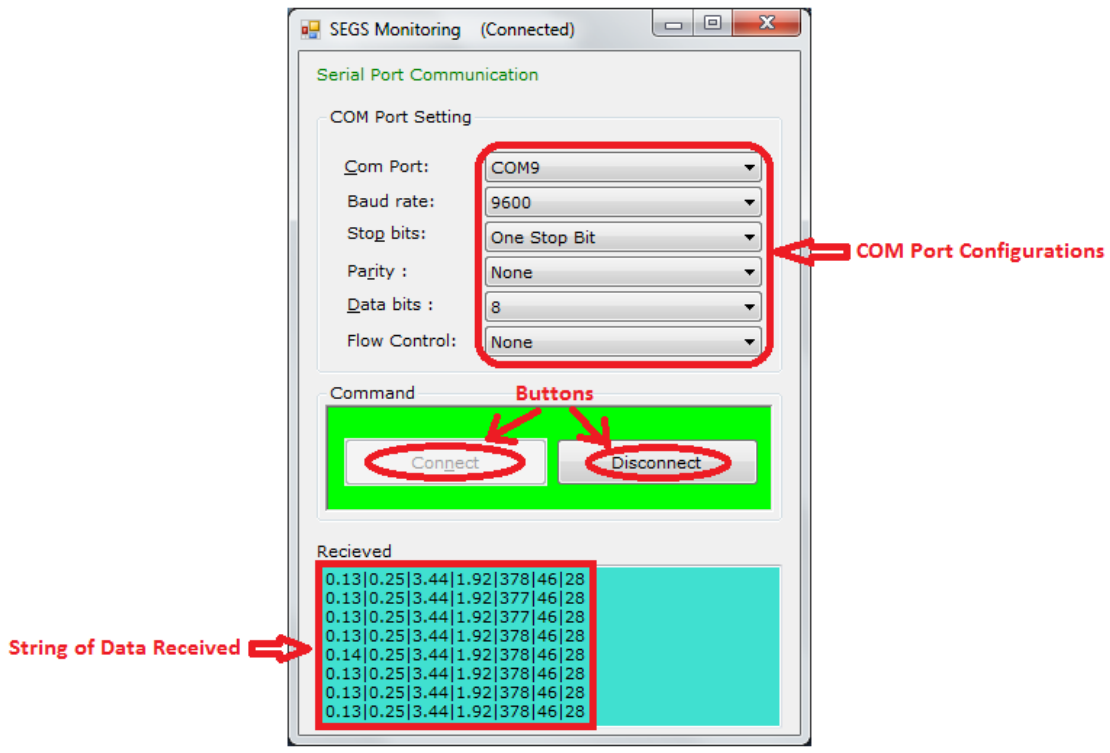


Figure 26 Main window of initial program

The main challenge in the beginning was to separate each value in the string and to assign the values to the type of measurement their represent. After improving the code of the program, this challenge was overcome. In the improved version of the initial application, the values could be displayed in the displaying window of the program. Figure 27 depicts the improved version of the initially designed program.

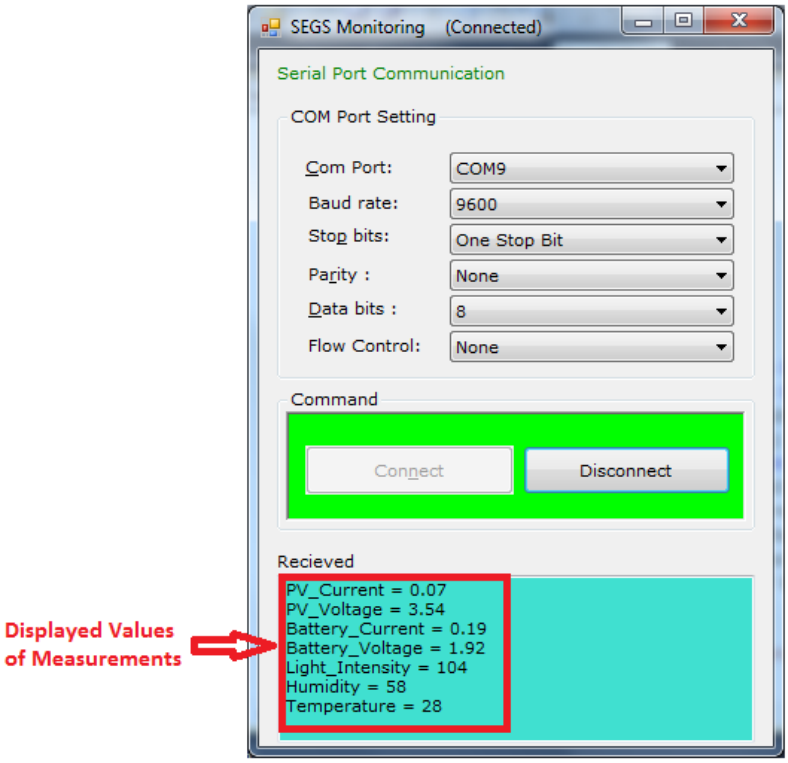


Figure 27 Main window of modified initial program

4.3.2 Intermediate Version of the SEGS Monitoring Program

In order to provide better functionality of the program and meet the expectations of users, the graphical feature was included in the intermediate version of the SEGS Monitoring software. The graphs would provide users with visual aid to understand and analyze the changes of monitored parameters. Measuring the values of environmental factors and creating graphs from the readings would also enable users to conduct researches and create statistical data of meteorological factors for each month. Moreover, this information would be useful to analyze and determine the average number of sunny days for each month or during a year in that particular location, so it would be easier to approximately estimate possible energy generated by solar energy generating system of a certain size and efficiency.

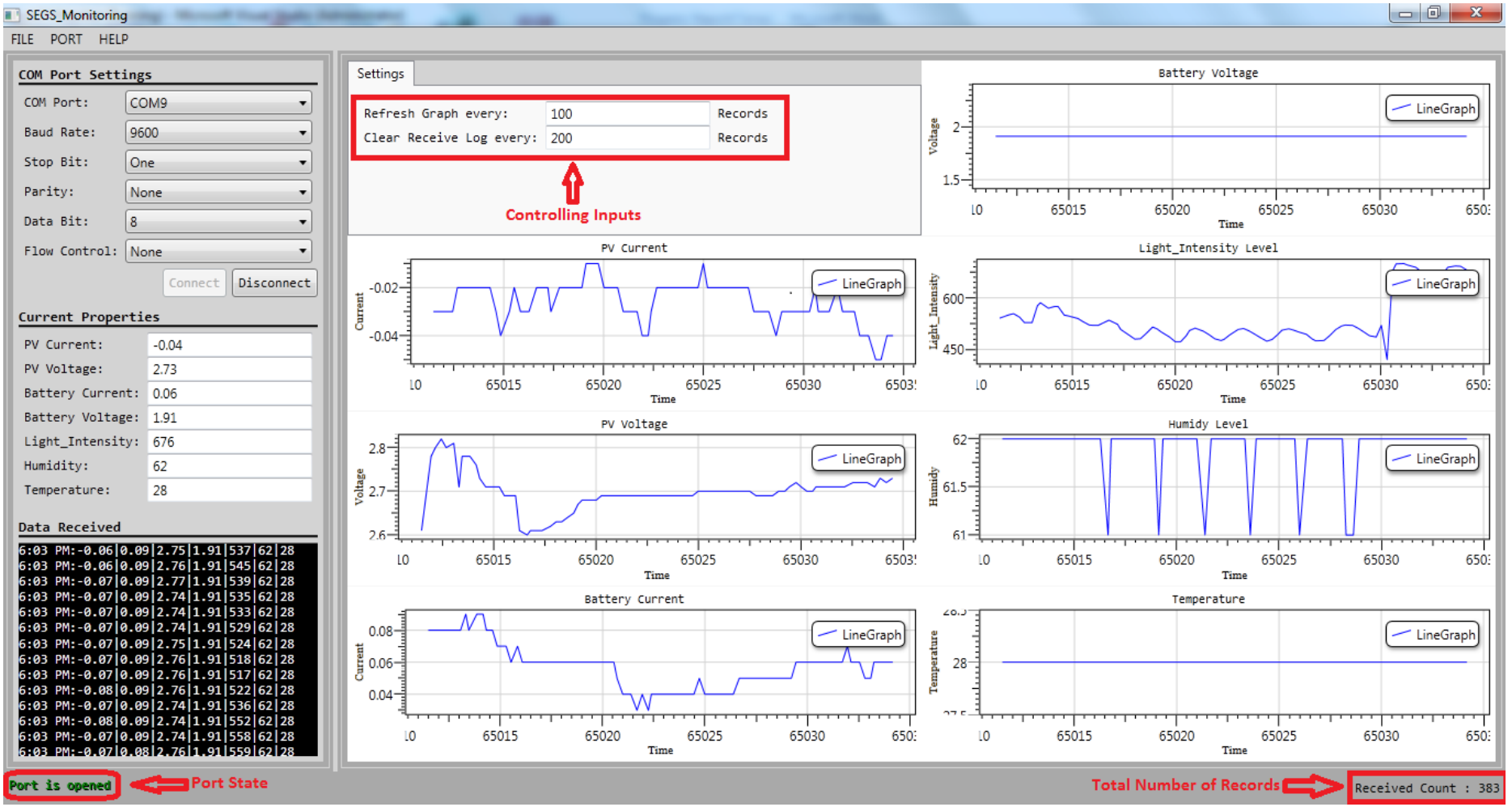


Figure 28 Main window of intermediate version of SEGS Monitoring program

Figure 28 illustrate the main window of intermediate version of SEGS Monitoring program. It is clear that all previous features, such as COM port settings, current readings of measurements and the list of data strings received, remain in the program. However, seven additional windows to display the "readings vs time" graphs for each sensor separately are added to the main window of the program. One more important addition is the feature to control and change the settings. Both controlling inputs enable users to set the specific number of reading records taken before refreshing the graph, thus providing more flexibility and choice to users. From Figure 28 it is understood that graphs will be refreshed after one hundred records are received and shown on the graphs. It is useful in case of need of limited number of records over a short period of time to obtain the necessary information to perform calculations or analysis. In order to keep track of total number of records received since the start of monitoring the right bottom corner of the application displays the output of the counter which provides this information. At the same time, the sign on the left bottom corner of the main window provides the information about state of the serial port. Users receive a hint about availability of the port.

4.3.3 Final Version of the SEGS Monitoring Program

The final and complete version of SEGS Monitoring software includes the analyzing part for readings of voltage values generated by PV panel. According to the Section 4.1 (Initial Assessment) of the report the readings of the voltage output of the solar panel were recorded for duration of several days, and the graph of average voltage value versus time of the day was obtained. Later, formula describing the curve of the graph was generated in Microsoft Excel software. After that, the influences of the dust effect and shading effect were observed in the lab. From the results, it was concluded that there is a direct linear relationship between percentage of the area of solar panel covered with shadow or dust and the output voltage drop of the panel. Taking into account these experimental results, additional PV generated voltage observing and analyzing feature was added to the program. This feature enables users to compare the PV voltage readings obtained from the sensors with the expected values of the measurements; thus, it enables to predict possible causes of the difference between these values.

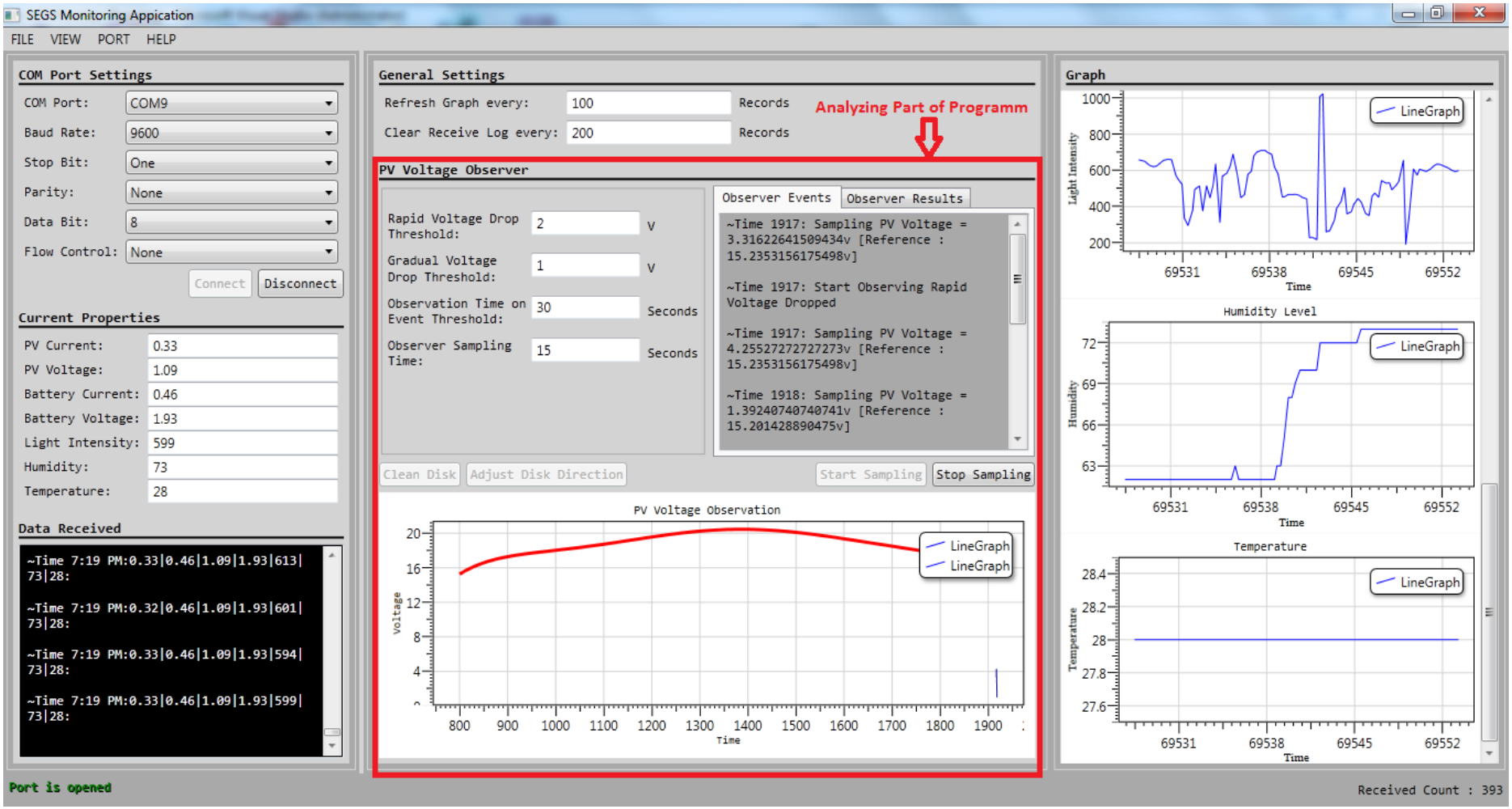


Figure 29 Main window of the final version of SEGS Monitoring program

The newly added feature enables users to select the value of the rapid voltage drop threshold in order to detect the rapid voltage drop of the actual value compare to expected value and which can be associated with partial covering effect or increased cloudiness. Gradual voltage drop threshold is used to set the value of voltage drop which would correspond to the difference between expected voltage value and the actual voltage reading during a specified period of time due to the effect of the dust accumulated on the solar panel. Observation time on event threshold is the optional input for users to specify the amount of time needed by analyzing part in order to define the conclusive decision whether rapid voltage drop or gradual voltage drop happened. Observer sampling time is the optional input for users to specify the amount of time required for analyzing part to calculate the average voltage between values in the beginning and at the end of the specified period of time. And this value is compared later to the expected value of output voltage to determine the rapid voltage drop, gradual voltage drop or normal voltage value.

The PV Voltage Observation graph depicts the curve of the experimentally obtained average values of voltage generated by PV panel at a specific point of time during a day with a red line and the actual values of voltage readings obtained at a specified time interval with a blue line. Start sampling and stop sampling buttons control the analyzing part. If the analyzer is on, all of the above described functions will be activated. However, if analyzer is off, the program only performs monitoring. Observer events and observer results windows provide the results and explanations of the analyzer.

Additional feature of the final version of the program is the data logging. As shown in Figure 30, the LOG folder is created in the same location where the main executable application file is placed.

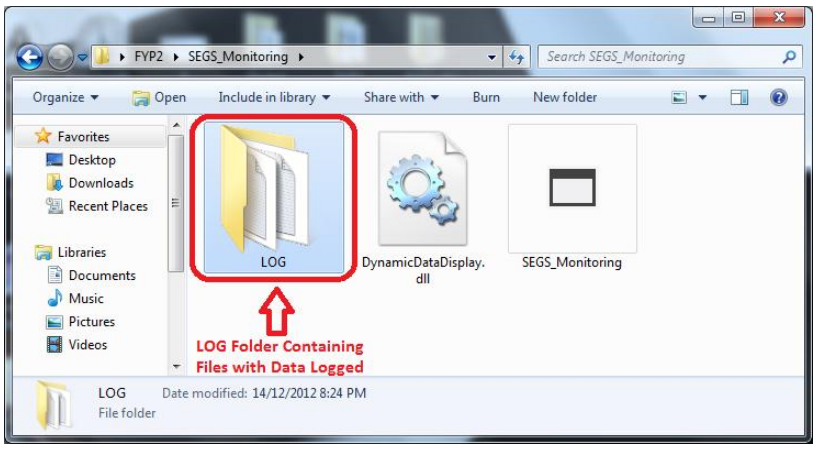


Figure 30 LOG folder location

All the readings obtained during monitoring are stored in the files with .txt format inside the LOG folder, as shown in Figure 31; thus, can be opened by Notepad software available almost on every computer. Every new file, depending on the time of creation, has the following format of labelling the title "SEGS_Monitoring_yyyy_mm_dd_hh_mm_ss".

Figure 31 Example of data log files created

As shown in Figure 32, each line begins with the corresponding date and time of the day when the readings were taken. Moreover, from the Figure 32, it is understood that averagely 3-4 strings of data are transferred from the monitoring device to the computer, thus providing almost real time measurements with high accuracy.

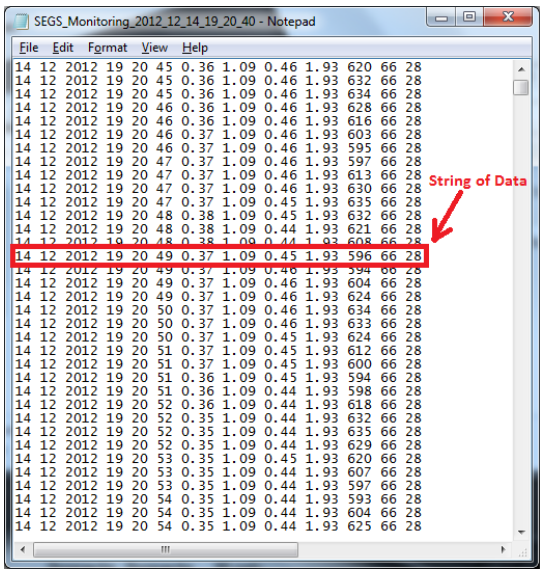


Figure 32 Content of data log file

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project presents the details and results of activities done in order to develop the solar electricity performance monitoring system. Vast knowledge was gained regarding solar energy and photovoltaic panels which are implemented to utilize this type of energy from literature review. Solar radiation is one of the renewable and clean sources of energy available. That is why many countries are investing in development and implementation of solar energy by building solar electricity generating plants. Despite the advantage of solar energy being free, there are still disadvantages in utilization this type of renewable energy. Photovoltaic panels which generate electricity from solar radiation have low efficiency and high cost. Apparent movement of the sun and meteorological conditions are the factors which reduce the efficiency of PV panels. Reliable performance tracking and monitoring system is one of the solutions which would help improve the efficiency of solar electricity generating systems.

Several experiments were conducted to bridge the gap of information lacking from literature review. The results of experiments show that there is a certain tendency in the voltage output of the solar panel depending on the availability of the sun, and tendency in change of output voltage during the day time. Maximum output can be achieved in the afternoon from 13:00 till 15:30. Moreover, another experiment was conducted to prove that the output current and voltage of PV panels are not stable but fluctuate. Certain factors such as dust effect and partial covering effect were manipulated while testing the solar panel's output characteristics to understand the influence. Thus, all these experiments were performed in order to meet one of the objectives of this project, which is to study the factors that affect the performance of PV based solar electricity generating system (SEGS). After acquiring all results, it can be concluded that this objective was successfully completed.

Information obtained from conducted experiments was used to design and create the analyzing and monitoring system. This information was used to choose sensors and other hardware parts needed to build the circuit of the monitoring system. Moreover, knowing the range of the output characteristics of tested PV panel under several conditions enabled author to select the variables to be measured while monitoring the components of solar electricity generating system. As soon as the circuit was finalized, the next stage of programming the software application to monitor the readings of sensors started. Several versions of the software were programmed eventually combining into the final and complete version, which has all features required for a reliable SEGS monitoring system. After combining hardware and software parts, tests and simulations were conducted in order to verify the correct functionality of the system in total. The results have met the requirements of the project. Thus, all objectives of the project have been achieved.

5.2 Recommendation for Project Continuation

Further improvement of the monitoring system can be achieved by expanding the capability of the system to entertain greater number of solar panels. However, in order to do so, the hardware part of the system should be modified significantly. This is due to the fact that all analog and digital I/O pins of Arduino Uno development board were used to connect the sensors and LCD. Thus, another model of the Arduino family boards should be implemented; for example Arduino Mega which has greater number of pins. More than that, the software application should be modified as well to provide the graphical illustration of the measurements obtained from additional solar panels.

Another useful suggestion is to integrate the wireless transmission of the readings feature. It would significantly improve and enhance the system in general.

The system can be implemented in a hybrid energy generating system as well which implements different sources of renewable energy. However, in this case additional experiments should be conducted initially to understand the nature of other energy resources and the factors affecting performance of generators utilizing that particular source of energy.

REFERENCES

[1] Renewable Energy. (2010). *Renewables 2010: Global Status Report.* Retrieved 2nd July, 2012, from http://www.ren21.net/Portals/97/documents/GSR/REN21_GSR_2010_full_revised%20Sept2010.pdf

[2] Listed Countries Using Renewable Energy. (2011). *Renewables 2011: Global Status Report.* Retrieved July 2, 2012, from http://www.ren21.net/Portals/97/documents/GSR/GSR2011_Master18.pdf

[3] Photovoltaic Module Performance. (2011). Retrieved July 2, 2012, from http://www.pres.org.pk/category/re-technologies/solar-energy/

[4] J.A. Duffie and W.A. Beckman, *Solar Engineering of Thermal Processes,* 2nd ed. New York: John Wiley & Sons, 1991.

[5] The German Energy Society, *Planning and Installing Photovoltaic Systems: A Guide for Installers, Architects and Engineers.* London, UK: James and James Science Publishers, 2005

[6] Climate Chart, "World Climate Maps", Retrieved July 3, 2012, from http://www.climate-charts.com/World-Climate-Maps.html

[7] D.R. Clark, S.A. Klein, and W.A. Beckman, "A Method of Estimating the Performance of Photovoltaic System," *Solar Energy*, vol.26, no. 5, pp. 413-418, 1981.

[8] Sandia National Laboratories, *Stand Alone Photovoltaic Power Systems: A Handbook of Recommended Design Practices.* USA, 1991.

[9] T. Markvart, *Solar Electricity.* Chichester, England: John Wiley & Sons Ltd, 1994.

[10] G.M. Masters, *Renewable and Efficient Electric Power Systems.* New Jersey: John Wiley & Sons, 2004.

[11] A. Luque and S. Hegedus, Eds., *Handbook of Photovoltaic Science and Engineering.* England: John Wiley & Sons, 2003.

[12] Luqu S.S. Soulayman, "On the Optimum Tilt of Solar Absorber Plates," *Renewable Energy,* vol. 1, no. 3-4, pp. 551-554, 1991.

[13] P.J. Lunde, *Solar Engineering of Thermal Processes.* New York: Wiley, 1980.

[14] G. Lewis, "Optimum Tilt of Solar Collectors," *Solar and Wind Tech,* vol. 4, pp. 407-410, 1987.

[15] D.L. Evans, "Simplified Method for Predicting Photovoltaic Array Output," *Solar Energy,* vol. 27, no. 6, pp. 555-560,1981

[16] Arduino Support Site, "Arduino Uno Reference Design", Retrieved December 2, 2012, from http://arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

[17] Arduino Support Site, "ATmega168/328-Arduino Pin Mapping", Retrieved December 2, 2012, from http://arduino.cc/en/Hacking/PinMapping168

APPENDICES

APPENDIX A

C LANGUAGE BASED CODING FOR ARDUINO UNO BOARD

```
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//   Author:  SERDAR ILYASOV
//   Project: Development of Solar Electricity Generating Performance Monitoring System
//   Version: v1.8
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


#define DHT11_PIN 1     // Analog output of DHT11 Humidity/Temperature sensor connected to A1 of Arduino
                        //UNO Board
#define Analog  A5      //Analog output of Light Sensor Module connected to A5 of Arduino UNO Board


//Global variables
int AnalogValue = 0;     //to store analog value
int lightVal = 0;

// constants won't change
//They're used here to set pin numbers:
const int buttonPin1 = 9;      // the number of the pushbutton pin
const int buttonPin2 = 10;     // the number of the pushbutton pin
const int buttonPin3 = 11;     // the number of the pushbutton pin
const int buttonPin4 = 12;     // the number of the pushbutton pin


// variables will change:
int buttonState1 = 0;        // variable for reading the pushbutton status
int buttonState2 = 0;        // variable for reading the pushbutton status
int buttonState3 = 0;        // variable for reading the pushbutton status
int buttonState4 = 0;        // variable for reading the pushbutton status




//DHT11 declaration and initialization
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
byte read_dht11_dat()
{
  byte i = 0;
  byte result=0;
  for(i=0; i< 8; i++)
  {
    while(!(PINC & _BV(DHT11_PIN)));  // wait for 50us
    delayMicroseconds(30);
    if(PINC & _BV(DHT11_PIN))
      result |=(1<<(7-i));
    while((PINC & _BV(DHT11_PIN)));  // wait '1' finish
    }
    return result;
}
```

```
//LCD declaration and initialization
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
// include the library code:
#include <LiquidCrystal.h>

//initialize the library with the numbers of the interface pins
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);


//PV Current Sensor declaration and initialization
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
const int numReadings1 = 30;
float readings1[numReadings1];      // the readings from the analogue input
int index1 = 0;                     // the index of the current reading
float total1 = 0;                   // the running total
float average1 = 0;                 // the average

float currentValue1 = 0;



//BATTERY Current Sensor declaration and initialization
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
const int numReadings = 30;
float readings[numReadings];        // the readings from the analogue input
int index = 0;                      // the index of the current reading
float total = 0;                    // the running total
float average = 0;                  // the average

float currentValue = 0;


void setup()
{

 //initialize the pushbutton pin as an input:
 pinMode(buttonPin1, INPUT);
 pinMode(buttonPin2, INPUT);
 pinMode(buttonPin3, INPUT);
 pinMode(buttonPin4, INPUT);


 //set up the LCD's number of columns and rows:
 lcd.begin(16, 2);


 //DHT11 Humidity/Temperature Sensor
 DDRC |= _BV(DHT11_PIN);
 PORTC |= _BV(DHT11_PIN);
 Serial.begin(9600);


 //PV_Current
 for (int thisReading1 = 0; thisReading1 < numReadings1; thisReading1++)
 readings1[thisReading1] = 0;


 //Battery_Current
 for (int thisReading = 0; thisReading < numReadings; thisReading++)
 readings[thisReading] = 0;

 //setup the input for Light Sensor
 pinMode(Analog, INPUT);

}
```

```
void loop()

{

  //Start of PV Current Sensor loop routine
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  total1= total1 - readings1[index1];
  readings1[index1] = analogRead(A0); //Raw data reading
  readings1[index1] = (readings1[index1]-510)*5/1024/0.04+0.13;   //Data processing:510-raw data from
                                                                  //analogRead when the input is 0; 5-5v; the
                                                                  //first 0.04-0.04V/A(sensitivity); the second
                                                                  //0.04-offset val;
  total1= total1 + readings1[index1];
  index1 = index1 + 1;
  if (index1 >= numReadings1)
  index1 = 0;
  average1 = total1/numReadings1;                                 //Smoothing algorithm
  currentValue1= average1;
  Serial.print(currentValue1);                                    //Sending PV current reading to PC
  Serial.print("|");                                              //separator
  delay(30);


  //Start of BATTERY Current Sensor loop routine
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   total= total - readings[index];
  readings[index] = analogRead(A4);                               //Raw data reading
  readings[index] = (readings[index]-510)*5/1024/0.04+0.13
   total= total + readings[index];
  index = index + 1;
  if (index >= numReadings)
    index = 0;
  average = total/numReadings;                                    //Smoothing algorithm
  currentValue= average;
  Serial.print(currentValue);                                     //Sending battery current readings to PC
  Serial.print("|");                                              //separator
  delay(30);



  //Start of PV Voltage Sensor loop routine
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    int sensorValue1 = analogRead(A2);
   float voltage1 = sensorValue1 * (23.0 / 1023.0);          // Convert the analog reading (which goes from
                                                             // 0 - 1023) to a voltage (0 - 23V)
   Serial.print(voltage1);                                   // send the PV voltage reading to PC
   Serial.print("|");                                        // separator

  //Start of BATTERY Voltage Sensor loop routine
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  int sensorValue2 = analogRead(A3);
  float voltage2 = sensorValue2 * (15.0 / 1023.0);           // Convert the analog reading (which goes from
                                                                    // 0-1023) to a voltage (0 - 15V)
  Serial.print(voltage2);                                   // print out the value of battery voltage
  Serial.print("|");                                        // separator
```

```
//Start of Light Sensor loop routine
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
AnalogValue = analogRead(Analog);
lightVal = 1023 - AnalogValue;
Serial.print(lightVal);                                  // display analogue value from light sensor
Serial.print("|");                                       // separator
delay(30);



//Start of DHT11 Temp/Humid Sensor loop routine
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
byte dht11_dat[5];
byte dht11_in;
byte i;                                                  // start condition
PORTC &= ~_BV(DHT11_PIN);                                // 1. pull-down i/o pin from 18ms
delay(18);
PORTC |= _BV(DHT11_PIN);
delayMicroseconds(40);
DDRC &= ~_BV(DHT11_PIN);
delayMicroseconds(40);

dht11_in = PINC & _BV(DHT11_PIN);
if(dht11_in)
{
  Serial.println("dht11 start condition 1 not met");
  return;
}
delayMicroseconds(80);
dht11_in = PINC & _BV(DHT11_PIN);
if(!dht11_in)
{
  Serial.println("dht11 start condition 2 not met");
  return;
}

delayMicroseconds(80);                                   // now ready for data reception
for (i=0; i<5; i++)
  dht11_dat[i] = read_dht11_dat();
DDRC |= _BV(DHT11_PIN);
PORTC |= _BV(DHT11_PIN);
byte dht11_check_sum = dht11_dat[0]+dht11_dat[1]+dht11_dat[2]+dht11_dat[3];         // check check_sum
if(dht11_dat[4]!= dht11_check_sum)
{
  Serial.println("DHT11 checksum error");
}

//Humidity readings sent to PC
Serial.print(dht11_dat[0], DEC);
Serial.print("|");

//Temperature readings sent to PC
Serial.println(dht11_dat[2], DEC);

delay(100);



// read the state of the pushbutton value:
buttonState1 = digitalRead(buttonPin1);
buttonState2 = digitalRead(buttonPin2);
buttonState3 = digitalRead(buttonPin3);
buttonState4 = digitalRead(buttonPin4);
```

```
 //Start of LCD loop routine
 //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 //>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   if (buttonState1 == HIGH && buttonState2 == LOW && buttonState3 == LOW && buttonState4 == LOW) {
        lcd.clear();                    //clear the LCD display
        lcd.setCursor(0,0);             //start form first column and first raw
        lcd.print("PV_Current=");
        lcd.print(currentValue1);       //display PV current readings
        lcd.print("A");
        lcd.setCursor(0,1);
        lcd.print("PV_Volt=");          //display PV voltage readings
        lcd.print(voltage1);
        lcd.print("V");
        delay(50); }

   else if (buttonState1 == LOW && buttonState2 == HIGH && buttonState3 == LOW && buttonState4 ==
LOW) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Bat_Current=");
        lcd.print(currentValue);        //display battery current readings
        lcd.print("A");
        lcd.setCursor(0,1);
        lcd.print("Bat_Volt=");
        lcd.print(voltage2);            //display battery voltage readings
        lcd.print("V");
        delay(50); }

   else if (buttonState1 == LOW && buttonState2 == LOW && buttonState3 == HIGH && buttonState4 ==
LOW) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Humidity=");
        lcd.print(dht11_dat[0]);        //display environmental humidity level readings
        lcd.print(".");
        lcd.print(dht11_dat[1]);
        lcd.print("%");
        lcd.setCursor(0,1);
        lcd.print("Temp=");
        lcd.print(dht11_dat[2]);        //display environmental temperature level readings
        lcd.print(".");
        lcd.print(dht11_dat[3]);
        lcd.print("C");
        delay(50); }

   else if (buttonState1 == LOW && buttonState2 == LOW && buttonState3 == LOW && buttonState4 ==
HIGH) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Light_Intensity=");
        lcd.setCursor(0,1);
        lcd.print(lightVal);            //display environmental light intensity level readings
        lcd.print(" ");
        delay(50); }

   else {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Btn1:PV ");          //press button1 to display voltage and current for PV
        lcd.print("Btn2:Bat");          //press button2 to display voltage and current for battery
        lcd.setCursor(0,1);
        lcd.print("Btn3:H/T ");         //press button3 to display humidity and temperature of environment
        lcd.print("Btn4:LI");           //press button4 to display environmental light intensity
        delay(50); }


}
```

APPENDIX B

ARDUINO UNO REFERENCE DESIGN

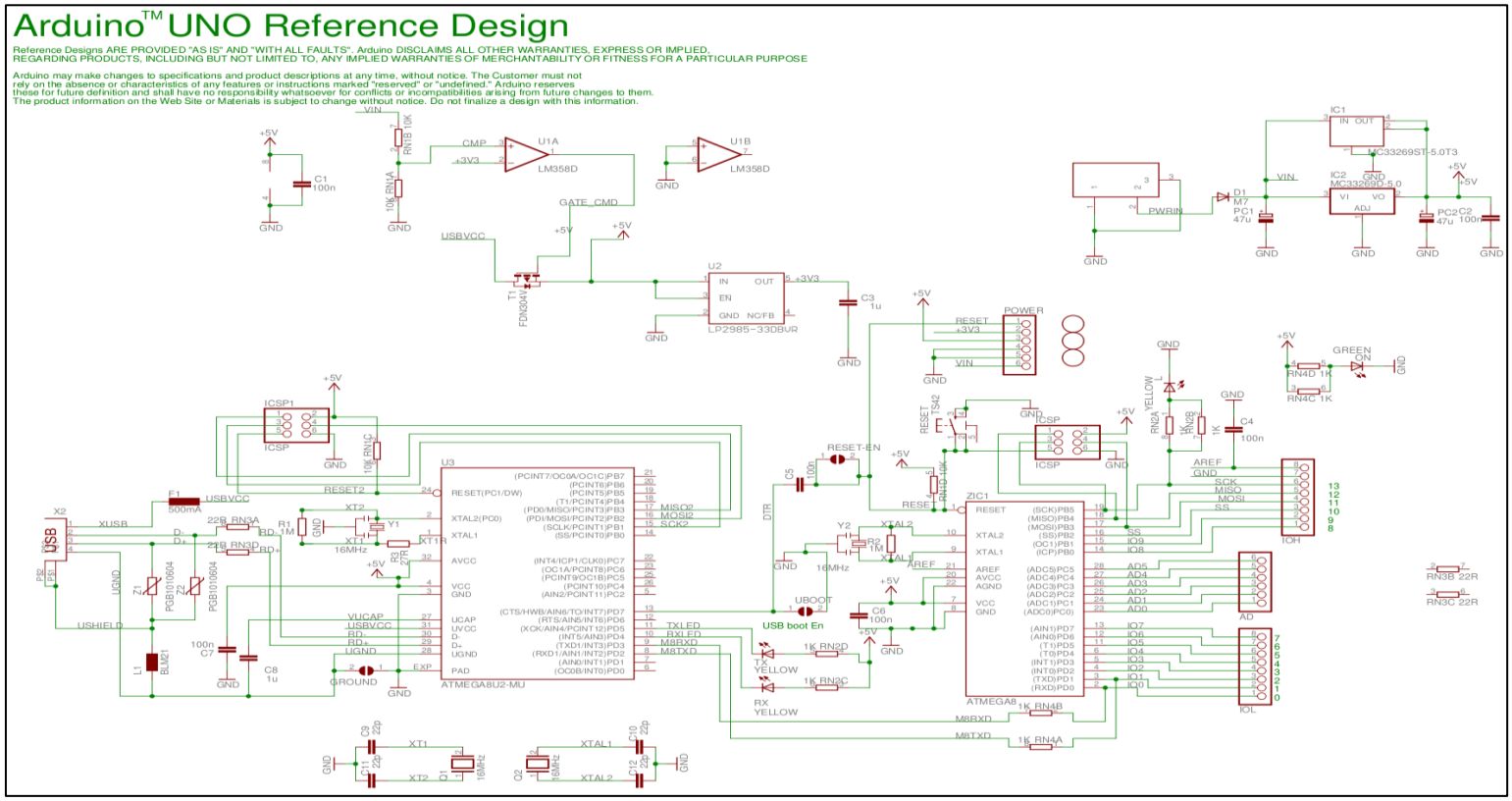


Figure 33 Arduino Uno board schematics **[16]**

APPENDIX C

ATMEGA328 PIN MAPPING

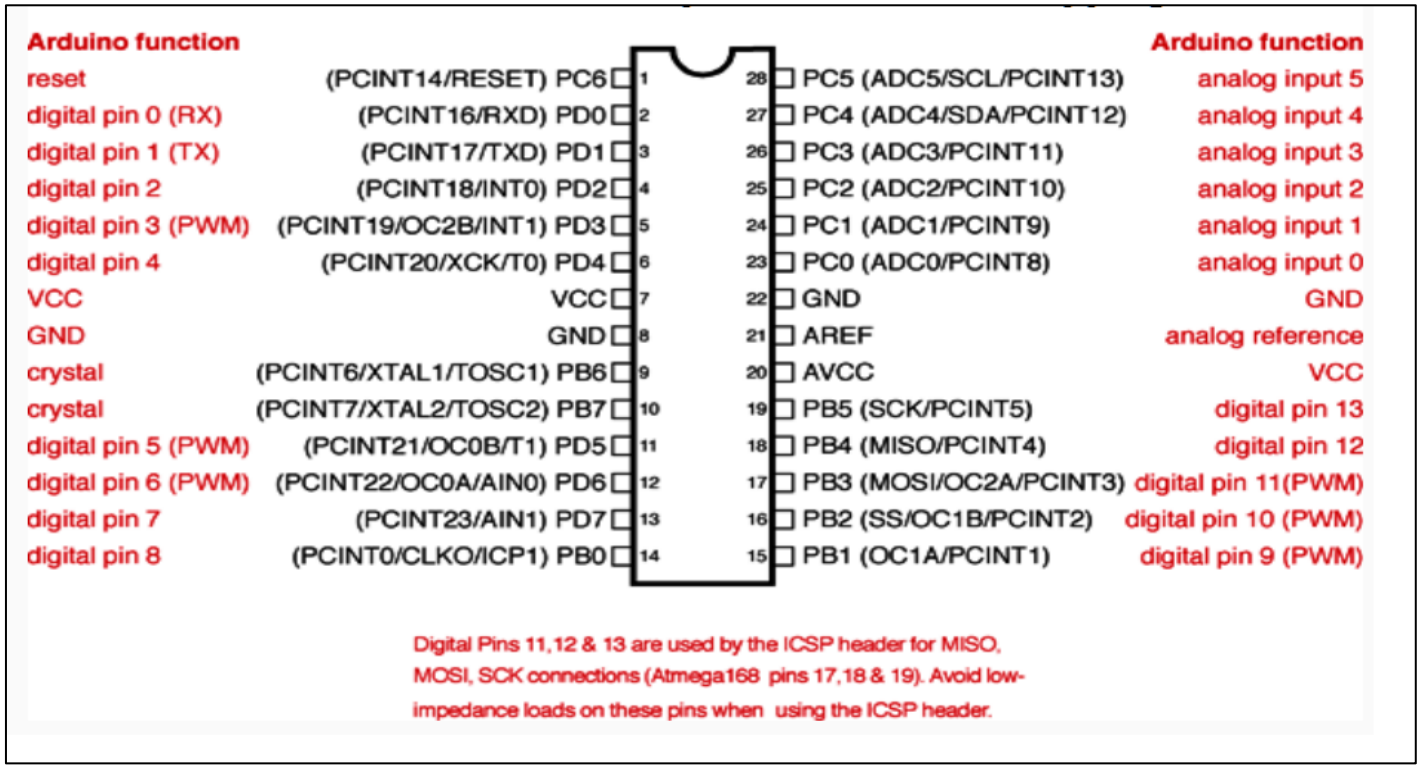


Figure 34 ATmega328P-PU chip to Arduino pin mapping **[17]**

APPENDIX D

VB.NET CODE FOR MONITORING & ANALYZING APPLICATION

```
Imports System.ComponentModel
Imports System.IO.Ports
Imports System.Text
Imports System.Collections.ObjectModel
Imports Microsoft.Research.DynamicDataDisplay.DataSources

Class MainWindow
    Implements INotifyPropertyChanged

#Region "StandardData"

    Public ReadOnly StandardBautRate As List(Of String) = New List(Of
String)(New String() {"2400", "4800", "9600", "14400", "19200", "38400",
"56000", "57600", "115200", "128000", "256000"})
    Public ReadOnly StandardDataBit As List(Of Integer) = New List(Of
Integer)(New Integer() {7, 8})

#End Region

#Region "Private UI Properties"

    Private _bautRateList As List(Of String)
    Private _selectedBautRate As String
    Private _comPortList As String()
    Private _selectedCOMPort As String
    Private _stopBitList As List(Of StopBits)
    Private _selectedStopBit As StopBits
    Private _parityList As List(Of Parity)
    Private _selectedParity As Parity
    Private _dataBitList As List(Of Integer)
    Private _selectedDataBit As Integer
    Private _handShakeList As List(Of Handshake)
    Private _selectedHandShake As Handshake
    Private _isPortAvailable As Boolean
    Private _currentPortState As PortState
    Private _PVVolatage As Double
    Private _PVCurrent As Double
    Private _batteryCurrent As Double
    Private _batteryVoltage As Double
    Private _humidity As Double
    Private _temperature As Double
    Private _darkness As Double
    Private _PVVoltageSeries As ObservableDataSource(Of Point)
    Private _PVCurrentSeries As ObservableDataSource(Of Point)
    Private _batteryVoltageSeries As ObservableDataSource(Of Point)
    Private _batteryCurrentSeries As ObservableDataSource(Of Point)
    Private _humiditySeries As ObservableDataSource(Of Point)
    Private _darknessSeries As ObservableDataSource(Of Point)
    Private _temperatureSeries As ObservableDataSource(Of Point)
    Private _clearGraphRecordNum As Integer
    Private _clearLogRecordNum As Integer
    Private _referencePVVoltageSeries As ObservableDataSource(Of Point)
```

```
    Private _averagePVVoltageSeries As ObservableDataSource(Of Point)
    Private _rapidVoltageDropTheshold As Double
    Private _gradualVoltageDropTheshold As Double
    Private _observationTimeOnEventThreshold As Integer
    Private _samplingTime As Integer
    Private _isPVVoltageGraphVisible As Boolean
    Private _isPVCurrentGraphVisible As Boolean
    Private _isBatterVoltageGraphVisible As Boolean
    Private _isBatterCurrentGraphVisible As Boolean
    Private _isHumidityGraphVisible As Boolean
    Private _isTemperatureGraphVisible As Boolean
    Private _isDarknessGraphVisible As Boolean

#End Region

#Region "Private Members"

    Private WithEvents _serialPort As SerialPort
    Private _readBuffer As String
    Private _monitoredData As MonitorData
    Private _rawData As StringBuilder
    Private _dataReceivedCount As Long
    Private _aboutView As AboutView

#End Region

#Region "MVVM Based"

    ''' <summary>
    ''' Property Change Event Handler
    ''' </summary>
    ''' <remarks>Do not remove this, if so all of the UI elements will have
problems :D</remarks>
    Public Event PropertyChanged As PropertyChangedEventHandler Implements
INotifyPropertyChanged.PropertyChanged

    ''' <summary>
    ''' Raise property change
    ''' </summary>
    ''' <param name="name">Property name</param>
    ''' <remarks>Do not remove this, if so all of the UI elements will have
problems :D</remarks>
    Protected Sub RaisePropertyChanged(ByVal name As String)

        RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(name))

    End Sub

#End Region

#Region "Private Methods"

    Private Sub Window_Initialized_1(sender As Object, e As EventArgs)

        _rawData = New StringBuilder()
        _aboutView = New AboutView()

        Me.PVVoltageSeries = New ObservableDataSource(Of Point)()
        Me.PVCurrentSeries = New ObservableDataSource(Of Point)()
        Me.BatteryVoltageSeries = New ObservableDataSource(Of Point)()
        Me.BatteryCurrentSeries = New ObservableDataSource(Of Point)()
        Me.TemperatureSeries = New ObservableDataSource(Of Point)()
        Me.DarknessSeries = New ObservableDataSource(Of Point)()
```

```
        Me.HumiditySeries = New ObservableDataSource(Of Point)()
        Me.ReferencePVVoltageSeries = New ObservableDataSource(Of Point)()
        Me.AveragePVVoltageSeries = New ObservableDataSource(Of Point)()

        Prepare()

        DataLogging.Instance.Initialise()

        AddHandler PVVoltageObserver.Instance.OnPVVolatageReport, AddressOf
PVVoltageObserverReport
        AddHandler PVVoltageObserver.Instance.OnEventReport, AddressOf
PVVoltageEventReport
        AddHandler PVVoltageObserver.Instance.OnRapidVoltageDroppedDetect,
AddressOf PVVoltageRapidDroppedDetected
        AddHandler PVVoltageObserver.Instance.OnConstantVoltageDroppedDetect,
AddressOf PVVoltageGradualDroppedDetected

        PVVoltageObserver.Instance.Initialise(100, 100)

        Dim intLoopIndex As Integer
        For intLoopIndex = 800 To 1830 Step 15
            Me.ReferencePVVoltageSeries.AppendAsync(Me.Dispatcher, New
Point(intLoopIndex,
PVVoltageObserver.Instance.GetReferenceVolatage(intLoopIndex)))
        Next intLoopIndex


        DataContext = Me

    End Sub

    Private Sub Prepare()

        Me.BautRateList = (From s In StandardBautRate
                     Select s).ToList()

        Me.SelectedBautRate = BautRateList(2)

        Me.ComPortList = IO.Ports.SerialPort.GetPortNames()
        If Me.ComPortList.Count <> 0 Then
            Me.SelectedCOMPort = Me.ComPortList.First()
            Me.IsPortAvailable = True
        Else
            Me.IsPortAvailable = False
        End If


        Me.StopBitList = [Enum].GetValues(GetType(StopBits)).Cast(Of
StopBits).ToList()
        Me.SelectedStopBit = Me.StopBitList(1)

        Me.ParityList = [Enum].GetValues(GetType(Parity)).Cast(Of
Parity).ToList()
        Me.SelectedParity = Me.ParityList(0)

        Me.DataBitList = (From s In StandardDataBit
                          Select s).ToList()
        Me.SelectedDataBit = Me.DataBitList(1)

        Me.HandShakeList = [Enum].GetValues(GetType(Handshake)).Cast(Of
Handshake).ToList()
        Me.SelectedHandShake = Me.HandShakeList(0)
```

```
    _serialPort = New SerialPort()

    Me.ClearGraphRecordNum = 100
    Me.ClearLogRecordNum = 200
    Me.RapidVoltageDropTheshold = 5
    Me.GradualVoltageDropTheshold = 2
    Me.ObservationTimeOnEventThreshold = 600
    Me.SamplingTime = 300
    Me.IsBatteryCurrentGraphVisible = True
    Me.IsBatteryVoltageGraphVisible = True
    Me.IsPVCurrentGraphVisible = True
    Me.IsPVVoltageGraphVisible = True
    Me.IsHumidityGraphVisible = True
    Me.IsDarknessGraphVisible = True
    Me.IsTemperatureGraphVisible = True


End Sub

Private Sub OpenPort()

    With _serialPort

        .PortName = _selectedCOMPort
        .BaudRate = _selectedBautRate
        .Parity = _selectedParity
        .DataBits = _selectedDataBit
        .StopBits = _selectedStopBit
        .Handshake = _selectedHandShake
        .RtsEnable = False
        .ReceivedBytesThreshold = 1
        .NewLine = vbCr
        .ReadTimeout = 10000

    End With

    ' Close if it is not close properly
    ClosePort()

    Try
        'Open port
        _serialPort.Open()

        'Update port state
        Me.CurrentPortState = PortState.PortOpen

        'Enable the sampling button
        btnStartSampling.IsEnabled = True


    Catch ex As Exception

        'Update port state
        Me.CurrentPortState = PortState.PortClose

        MessageBox.Show("Unable to open port!")

    End Try


End Sub

Private Sub RefreshPort()
```

```
        Me.ComPortList = IO.Ports.SerialPort.GetPortNames()
        If Me.ComPortList.Count <> 0 Then
            Me.SelectedCOMPort = Me.ComPortList.First()
            Me.IsPortAvailable = True
        Else
            Me.IsPortAvailable = False
        End If


    End Sub

    Private Sub ClosePort()

        If _serialPort.IsOpen Then

            'Turn off sampling
            PVVoltageObserver.Instance.StopSampling()

            'Disable the buttons
            btnStopSampling.IsEnabled = False
            btnStartSampling.IsEnabled = False


            _serialPort.Close()

            'Update port state
            Me.CurrentPortState = PortState.PortClose

            '

        End If

    End Sub

    Private Sub OnDataReceived(ByVal sender As System.Object, _
                               ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) _
                               Handles _serialPort.DataReceived

        Dim monitoredData As MonitorData
        Dim receiveTime = Date.Now

        If _serialPort.IsOpen Then

            Try

                _readBuffer = _serialPort.ReadLine()
                DataReceivedCount += 1

                Try
                    monitoredData = New MonitorData(_readBuffer,
receiveTime.Ticks)


                    If (PVVoltageObserver.Instance.IsStartedToObserve()) Then

PVVoltageObserver.Instance.Observe(monitoredData.PVVoltage)
                    End If

                    DataLogging.Instance.Log(monitoredData)
```

```
                    Me.Dispatcher.BeginInvoke(New Action(Of
MonitorData)(AddressOf UpdateCurrentMonitorData), monitoredData)
                    Me.Dispatcher.BeginInvoke(New Action(Of String)(AddressOf
UpdateDataToUI), String.Format("{0}:{1}", receiveTime.ToShortTimeString(),
_readBuffer.Trim()))

                    UpdateSeries(monitoredData)

                Catch ex As Exception

                End Try

            Catch inv As InvalidCastException

            Catch ex As Exception
                'MessageBox.Show("Time out!")
            End Try

        End If

    End Sub

    Private Sub UpdateDataToUI(ByVal rawData As String)

        If rawData <> String.Empty Then
            If DataReceivedCount Mod Me.ClearLogRecordNum = 0 Then
                UIRawData.Document.Blocks.Clear()
            End If

            UIRawData.AppendText(String.Format("~Time {0}:{1}", rawData.Trim(),
Environment.NewLine))
            UIRawData.ScrollToEnd()

        End If

    End Sub

    Private Sub UpdateCurrentMonitorData(ByVal monitoredData As MonitorData)

        If monitoredData Is Nothing Then
            End
        End If

        PVCurrent = monitoredData.PVCurrent
        PVVolatage = monitoredData.PVVoltage
        BatteryCurrent = monitoredData.BatteryCurrent
        BatteryVoltage = monitoredData.BatteryVoltage
        Darkness = monitoredData.Darkness
        Humidity = monitoredData.Humidity
        Temperature = monitoredData.Temperature

    End Sub

    Private Sub UpdateSeries(ByVal monitoredData As MonitorData)

        Dim time = New Date(monitoredData.Time)

        Dim refZeroTime = New Date(time.Year, time.Month, time.Day)
        Dim observeTime = time.Subtract(refZeroTime)
        Dim stTime = observeTime.TotalSeconds

        If DataReceivedCount Mod ClearGraphRecordNum = 0 Then
            Me.Dispatcher.BeginInvoke(New Action(AddressOf ClearGraphSeries))
```

```
        Else
            Me.PVVoltageSeries.AppendAsync(Me.Dispatcher, New Point(stTime,
monitoredData.PVVoltage))
            Me.PVCurrentSeries.AppendAsync(Me.Dispatcher, New Point(stTime,
monitoredData.PVCurrent))
            Me.BatteryCurrentSeries.AppendAsync(Me.Dispatcher, New
Point(stTime, monitoredData.BatteryCurrent))
            Me.BatteryVoltageSeries.AppendAsync(Me.Dispatcher, New
Point(stTime, monitoredData.BatteryVoltage))
            Me.HumiditySeries.AppendAsync(Me.Dispatcher, New Point(stTime,
monitoredData.Humidity))
            Me.DarknessSeries.AppendAsync(Me.Dispatcher, New Point(stTime,
monitoredData.Darkness))
            Me.TemperatureSeries.AppendAsync(Me.Dispatcher, New Point(stTime,
monitoredData.Temperature))
        End If

    End Sub

    Private Sub PVVoltageObserverReport(ByVal sender As Object, ByVal arg As
EventArgs)

        Dim aegr = DirectCast(arg, VolatageEventArg)

        If aegr Is Nothing Then
            Return
        End If

        Dim refVoltage =
PVVoltageObserver.Instance.GetReferenceVolatage(aegr.Time)
        Me.AveragePVVoltageSeries.AppendAsync(Me.Dispatcher, New
Point(aegr.Time, aegr.AverageVoltage))
        rtxObserverReport.AppendText(String.Format("~Time {0}: Sampling PV
Voltage = {1}v [Reference : {2}v]{3}", aegr.Time, aegr.AverageVoltage,
refVoltage, Environment.NewLine))

    End Sub

    Private Sub PVVoltageEventReport(ByVal sender As Object, ByVal arg As
EventArgs)

        Dim rptEgr = DirectCast(arg, ReportEventArg)

        If rptEgr Is Nothing Then
            Return
        End If

        rtxObserverReport.AppendText(String.Format("~Time {0}: {1}{2}",
rptEgr.Time, rptEgr.Report, Environment.NewLine))
        rtxObserverReport.ScrollToEnd()

    End Sub

    Private Sub PVVoltageRapidDroppedDetected(ByVal sender As Object, ByVal arg
As EventArgs)

        Dim rptEgr = DirectCast(arg, ReportEventArg)

        If rptEgr Is Nothing Then
            Return
        End If
```

```
        rtxAnalysisReport.AppendText(String.Format("~Time {0}: {1}{2}",
rptEgr.Time, rptEgr.Report, Environment.NewLine))
        rtxObserverReport.ScrollToEnd()

        btnMovePanel.IsEnabled = True

    End Sub

    Private Sub PVVoltageGradualDroppedDetected(ByVal sender As Object, ByVal
arg As EventArgs)

        Dim rptEgr = DirectCast(arg, ReportEventArg)

        If rptEgr Is Nothing Then
            Return
        End If

        rtxAnalysisReport.AppendText(String.Format("~Time {0}: {1}{2}",
rptEgr.Time, rptEgr.Report, Environment.NewLine))
        rtxAnalysisReport.ScrollToEnd()

        btnCleanDust.IsEnabled = True

    End Sub

    Private Sub ClearGraphSeries()

        Me.PVVoltageSeries.Collection.Clear()
        Me.PVCurrentSeries.Collection.Clear()
        Me.BatteryCurrentSeries.Collection.Clear()
        Me.BatteryVoltageSeries.Collection.Clear()
        Me.HumiditySeries.Collection.Clear()
        Me.DarknessSeries.Collection.Clear()
        Me.TemperatureSeries.Collection.Clear()

    End Sub

    Private Sub OnCloseCommand()

        DataLogging.Instance.Close()

        Me.Close()

    End Sub

#End Region

#Region "Properties"

    Public Property DataReceivedCount() As Double
        Get
            Return _dataReceivedCount
        End Get
        Set(ByVal value As Double)
            _dataReceivedCount = value
            RaisePropertyChanged("DataReceivedCount")
        End Set
    End Property

    Public Property BautRateList() As List(Of String)
        Get
            Return _bautRateList
        End Get
```

```
    Set(ByVal value As List(Of String))
        _bautRateList = value
        RaisePropertyChanged("BautRateList")
    End Set
End Property

Public Property SelectedBautRate() As String
    Get
        Return _selectedBautRate
    End Get
    Set(ByVal value As String)
        _selectedBautRate = value
    End Set
End Property

Public Property ComPortList() As String()
    Get
        Return _comPortList
    End Get
    Set(ByVal value As String())
        _comPortList = value
    End Set
End Property

Public Property SelectedCOMPort() As String
    Get
        Return _selectedCOMPort
    End Get
    Set(ByVal value As String)
        _selectedCOMPort = value
    End Set
End Property

Public Property StopBitList() As List(Of StopBits)
    Get
        Return _stopBitList
    End Get
    Set(ByVal value As List(Of StopBits))
        _stopBitList = value
    End Set
End Property

Public Property SelectedStopBit() As StopBits
    Get
        Return _selectedStopBit
    End Get
    Set(ByVal value As StopBits)
        _selectedStopBit = value
    End Set
End Property

Public Property ParityList() As List(Of Parity)
    Get
        Return _parityList
    End Get
    Set(ByVal value As List(Of Parity))
        _parityList = value
    End Set
End Property

Public Property SelectedParity() As Parity
    Get
        Return _selectedParity
```

```
    End Get
    Set(ByVal value As Parity)
        _selectedParity = value
    End Set
End Property

Public Property DataBitList() As List(Of Integer)
    Get
        Return _dataBitList
    End Get
    Set(ByVal value As List(Of Integer))
        _dataBitList = value
    End Set
End Property

Public Property SelectedDataBit() As Integer
    Get
        Return _selectedDataBit
    End Get
    Set(ByVal value As Integer)
        _selectedDataBit = value
    End Set
End Property

Public Property HandShakeList() As List(Of Handshake)
    Get
        Return _handShakeList
    End Get
    Set(ByVal value As List(Of Handshake))
        _handShakeList = value
    End Set
End Property

Public Property SelectedHandShake() As Handshake
    Get
        Return _selectedHandShake
    End Get
    Set(ByVal value As Handshake)
        _selectedHandShake = value
    End Set
End Property

Public Property IsPortAvailable() As Boolean
    Get
        Return _isPortAvailable
    End Get
    Set(ByVal value As Boolean)
        _isPortAvailable = value
    End Set
End Property

Public Property CurrentPortState() As PortState
    Get
        Return _currentPortState
    End Get
    Set(ByVal value As PortState)
        _currentPortState = value
        RaisePropertyChanged("CurrentPortState")
    End Set
End Property

Public ReadOnly Property RawData() As String
    Get
```

```
            Return _rawData.ToString()
        End Get
    End Property

    Public Property PVVolatage() As Double
        Get
            Return _PVVolatage
        End Get
        Set(ByVal value As Double)
            _PVVolatage = value
            RaisePropertyChanged("PVVolatage")
        End Set
    End Property

    Public Property PVCurrent() As Double
        Get
            Return _PVCurrent
        End Get
        Set(ByVal value As Double)
            _PVCurrent = value
            RaisePropertyChanged("PVCurrent")
        End Set
    End Property

    Public Property BatteryCurrent() As Double
        Get
            Return _batteryCurrent
        End Get
        Set(ByVal value As Double)
            _batteryCurrent = value
            RaisePropertyChanged("BatteryCurrent")
        End Set
    End Property

    Public Property BatteryVoltage() As Double
        Get
            Return _batteryVoltage
        End Get
        Set(ByVal value As Double)
            _batteryVoltage = value
            RaisePropertyChanged("BatteryVoltage")
        End Set
    End Property

    Public Property Humidity() As Double
        Get
            Return _humidity
        End Get
        Set(ByVal value As Double)
            _humidity = value
            RaisePropertyChanged("Humidity")
        End Set
    End Property

    Public Property Temperature() As Double
        Get
            Return _temperature
        End Get
        Set(ByVal value As Double)
            _temperature = value
            RaisePropertyChanged("Temperature")
        End Set
    End Property
```

```
Public Property Darkness() As Double
    Get
        Return _darkness
    End Get
    Set(ByVal value As Double)
        _darkness = value
        RaisePropertyChanged("Darkness")
    End Set
End Property

Public Property PVVoltageSeries() As ObservableDataSource(Of Point)
    Get
        Return _PVVoltageSeries
    End Get
    Set(ByVal value As ObservableDataSource(Of Point))
        _PVVoltageSeries = value
        RaisePropertyChanged("PVVoltageSeries")
    End Set
End Property

Public Property PVCurrentSeries() As ObservableDataSource(Of Point)
    Get
        Return _PVCurrentSeries
    End Get
    Set(ByVal value As ObservableDataSource(Of Point))
        _PVCurrentSeries = value
        RaisePropertyChanged("PVCurrentSeries")
    End Set
End Property

Public Property BatteryVoltageSeries() As ObservableDataSource(Of Point)
    Get
        Return _batteryVoltageSeries
    End Get
    Set(ByVal value As ObservableDataSource(Of Point))
        _batteryVoltageSeries = value
        RaisePropertyChanged("BatteryVoltageSeries")
    End Set
End Property

Public Property BatteryCurrentSeries() As ObservableDataSource(Of Point)
    Get
        Return _batteryCurrentSeries
    End Get
    Set(ByVal value As ObservableDataSource(Of Point))
        _batteryCurrentSeries = value
        RaisePropertyChanged("BatteryCurrentSeries")
    End Set
End Property

Public Property HumiditySeries() As ObservableDataSource(Of Point)
    Get
        Return _humiditySeries
    End Get
    Set(ByVal value As ObservableDataSource(Of Point))
        _humiditySeries = value
        RaisePropertyChanged("HumiditySeries")
    End Set
End Property

Public Property DarknessSeries() As ObservableDataSource(Of Point)
    Get
```

```
            Return _darknessSeries
        End Get
        Set(ByVal value As ObservableDataSource(Of Point))
            _darknessSeries = value
            RaisePropertyChanged("DarknessSeries")
        End Set
    End Property

    Public Property TemperatureSeries() As ObservableDataSource(Of Point)
        Get
            Return _temperatureSeries
        End Get
        Set(ByVal value As ObservableDataSource(Of Point))
            _temperatureSeries = value
            RaisePropertyChanged("TemperatureSeries")
        End Set
    End Property

    Public Property ClearGraphRecordNum() As Integer
        Get
            Return _clearGraphRecordNum
        End Get
        Set(ByVal value As Integer)
            _clearGraphRecordNum = value
            RaisePropertyChanged("ClearGraphRecordNum")
        End Set
    End Property

    Public Property ClearLogRecordNum() As Integer
        Get
            Return _clearLogRecordNum
        End Get
        Set(ByVal value As Integer)
            _clearLogRecordNum = value
            RaisePropertyChanged("ClearLogRecordNum")
        End Set
    End Property

    Public Property ReferencePVVoltageSeries() As ObservableDataSource(Of
Point)
        Get
            Return _referencePVVoltageSeries
        End Get
        Set(ByVal value As ObservableDataSource(Of Point))
            _referencePVVoltageSeries = value
            RaisePropertyChanged("ReferencePVVoltageSeries")
        End Set
    End Property

    Public Property AveragePVVoltageSeries() As ObservableDataSource(Of Point)
        Get
            Return _averagePVVoltageSeries
        End Get
        Set(ByVal value As ObservableDataSource(Of Point))
            _averagePVVoltageSeries = value
            RaisePropertyChanged("AveragePVVoltageSeries")
        End Set
    End Property

    Public Property RapidVoltageDropTheshold() As Double
        Get
            Return _rapidVoltageDropTheshold
        End Get
```

```
        Set(ByVal value As Double)
            _rapidVoltageDropTheshold = value


PVVoltageObserver.Instance.SetRapidVoltageDropThreshold(_rapidVoltageDropThesho
ld)

            RaisePropertyChanged("RapidVoltageDropTheshold")
        End Set
    End Property

    Public Property GradualVoltageDropTheshold() As Double
        Get
            Return _gradualVoltageDropTheshold
        End Get
        Set(ByVal value As Double)
            _gradualVoltageDropTheshold = value


PVVoltageObserver.Instance.SetGradualVoltageDropThreshold(_gradualVoltageDropTh
eshold)

            RaisePropertyChanged("GradualVoltageDropTheshold")
        End Set
    End Property

    Public Property ObservationTimeOnEventThreshold() As Integer
        Get
            Return _observationTimeOnEventThreshold
        End Get
        Set(ByVal value As Integer)
            _observationTimeOnEventThreshold = value

PVVoltageObserver.Instance.SetSamplingTime(_observationTimeOnEventThreshold)
            RaisePropertyChanged("ObservationTimeOnEventThreshold")
        End Set
    End Property

    Public Property SamplingTime() As Integer
        Get
            Return _samplingTime
        End Get
        Set(ByVal value As Integer)
            _samplingTime = value
            PVVoltageObserver.Instance.SetSamplingTime(_samplingTime)
            RaisePropertyChanged("SamplingTime")
        End Set
    End Property

    Public Property IsPVVoltageGraphVisible() As Boolean
        Get
            Return _isPVVoltageGraphVisible
        End Get
        Set(ByVal value As Boolean)
            _isPVVoltageGraphVisible = value
            RaisePropertyChanged("IsPVVoltageGraphVisible")
        End Set
    End Property

    Public Property IsPVCurrentGraphVisible() As Boolean
        Get
            Return _isPVCurrentGraphVisible
        End Get
```

```
            Set(ByVal value As Boolean)
                _isPVCurrentGraphVisible = value
                RaisePropertyChanged("IsPVCurrentGraphVisible")
            End Set
        End Property

        Public Property IsBatteryVoltageGraphVisible() As Boolean
            Get
                Return _isBatterVoltageGraphVisible
            End Get
            Set(ByVal value As Boolean)
                _isBatterVoltageGraphVisible = value
                RaisePropertyChanged("IsBatteryVoltageGraphVisible")
            End Set
        End Property

        Public Property IsBatteryCurrentGraphVisible() As Boolean
            Get
                Return _isBatterCurrentGraphVisible
            End Get
            Set(ByVal value As Boolean)
                _isBatterCurrentGraphVisible = value
                RaisePropertyChanged("IsBatteryCurrentGraphVisible")
            End Set
        End Property

        Public Property IsHumidityGraphVisible() As Boolean
            Get
                Return _isHumidityGraphVisible
            End Get
            Set(ByVal value As Boolean)
                _isHumidityGraphVisible = value
                RaisePropertyChanged("IsHumidityGraphVisible")
            End Set
        End Property

        Public Property IsTemperatureGraphVisible() As Boolean
            Get
                Return _isTemperatureGraphVisible
            End Get
            Set(ByVal value As Boolean)
                _isTemperatureGraphVisible = value
                RaisePropertyChanged("IsTemperatureGraphVisible")
            End Set
        End Property

        Public Property IsDarknessGraphVisible() As Boolean
            Get
                Return _isDarknessGraphVisible
            End Get
            Set(ByVal value As Boolean)
                _isDarknessGraphVisible = value
                RaisePropertyChanged("IsDarknessGraphVisible")
            End Set
        End Property

    #End Region

    #Region "Commads"

        Private _closeCommand As RelayCommand
        Public ReadOnly Property CloseCommand As ICommand
            Get
```

```
            If _closeCommand Is Nothing Then
                _closeCommand = New RelayCommand(AddressOf Me.OnCloseCommand)
            End If

            Return _closeCommand
        End Get
    End Property

    Private _openPortCommand As RelayCommand
    Public ReadOnly Property OpenPortCommand As ICommand
        Get
            If _openPortCommand Is Nothing Then
                _openPortCommand = New RelayCommand(AddressOf Me.OpenPort)
            End If

            Return _openPortCommand
        End Get
    End Property

    Private _closePortCommand As RelayCommand
    Public ReadOnly Property ClosePortCommand As ICommand
        Get
            If _closePortCommand Is Nothing Then
                _closePortCommand = New RelayCommand(AddressOf Me.ClosePort)
            End If

            Return _closePortCommand
        End Get
    End Property

    Private _refreshPortCommand As RelayCommand
    Public ReadOnly Property RefreshPortCommand As ICommand
        Get
            If _refreshPortCommand Is Nothing Then
                _refreshPortCommand = New RelayCommand(AddressOf
Me.RefreshPort)
            End If

            Return _refreshPortCommand
        End Get
    End Property

#End Region

#Region "Manual UI Events"

    Private Sub btnStartSampling_PreviewMouseDown_1(sender As Object, e As
MouseButtonEventArgs)

        PVVoltageObserver.Instance.StartSampling()
        btnStopSampling.IsEnabled = True
        btnStartSampling.IsEnabled = False

    End Sub

    Private Sub btnStopSampling_PreviewMouseDown_1(sender As Object, e As
MouseButtonEventArgs)

        PVVoltageObserver.Instance.StopSampling()
        btnStartSampling.IsEnabled = True
        btnStopSampling.IsEnabled = False

    End Sub
```

```
    Private Sub btnCleanDust_PreviewMouseDown_1(sender As Object, e As
MouseButtonEventArgs)
        btnCleanDust.IsEnabled = False
    End Sub

    Private Sub btnMovePanel_PreviewMouseDown_1(sender As Object, e As
MouseButtonEventArgs)
        btnMovePanel.IsEnabled = False
    End Sub

    Private Sub menuAbout_PreviewMouseDown_1(sender As Object, e As
MouseButtonEventArgs)
        _aboutView.ShowDialog()
    End Sub

#End Region

End Class
```