Online Photovoltaic Monitoring System

A Major Qualifying Project Report

Submitted to the faculty

Of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Ali Magzari

Date: April 15, 2015

Advisor:

Professor Susan M. Jarvis

Abstract

The goal of this project is to create an online photovoltaic monitoring system. This is achieved by designing a photovoltaic system, building the analog circuitry for proper voltage and current readings, and creating a webserver that displays the monitored data in a user-friendly charting interface. The webserver is in WAN (Wide Area Network) and could therefore be accessed anywhere in the world that benefits from an Internet connection. The project could potentially be adapted to service different types of photovoltaic systems to insure proper functioning and data monitoring.

Acknowledgements

I would like to thank the following:

My advisor, Professor Susan M. Jarvis for her constant encouragements and guidance throughout the project. I appreciate her patience in dealing with the most disorganized student. Her understanding and support are the only motives that granted me strength to complete this project.

My friend, Eric Willcox for helping me design the solar panels acrylic support for this project. His rendering skills are stupendous.

My friend, Velin Dimitrov for introducing the theoretical person that I am to Washburn Shops and laser cutting technology.

My Mother, Lalla Najat for her unconditional love and support. She would telepathically know when things went wrong and call me in the middle of the night to offer me comfort. Life without her would stand meaningless. Je t'aime Maman.

Table of Contents

List of Figures

List of Tables

Executive Summary

The photovoltaic system used in this project consists of two parallel strings each formed of three panels in series. To complete the circuit, a 100kΩ resistor is used as a load to the system. That way, we could measure the current going through the load rather than limiting ourselves to the 0A open circuit current. Every panel is known to have a 9V,111.11mA maximum ratings. To insure the operational safety of the microcontroller board responsible of data collection (Arduino), a voltage divider has been set up at the positive terminal of every panel to lower the photovoltaic maximum voltage range (0V-27V) to the acceptable voltage range of the analog input rant of the ADC in the Arduino (0V-5V). For the current issue, a voltage follower was placed at the measuring node of the voltage divider to isolate the photovoltaic current from the microcontroller analog pins.

An Ethernet shield was utilized to transfer the collected data to the designed webserver. The design of the webserver was simplified to consist of two major parts: Real-time Data visualized as a list in the default page of the webserver and Data History which displays the voltage and current values of the system on a chart system over the course of the day selected by the user.

So far, the results have been satisfying and the goals of the project reached. It would be however necessary in the future to ameliorate this project by taking into account other variables such as temperature. Investigating and displaying how the system I-V curve is changing during shading and other events would be a great addition to the project.

1.0 Introduction

Energy harvesting has been the subject of core scientific attention for many years now. Renewable energy was first targeted toward industrial projects where it was solely applied at the macro level. However, harmful fossil fuels emissions and extravagant oil prices increased the public awareness greatly. That caused a potential turning point where green energy became a major focus in several market sectors that directly impact the consumers. Solar energy has been particularly researched and utilized in multiple areas such as domestic power production.

Solar energy proved to be a great alternative in terms of power cost reduction and energy sustainability. On the other hand, such system requires regular maintenance to insure its proper operation. Seemingly insignificant factors such as dust and shading could lower the output power of one cell, thus reducing the power output of the entire photovoltaic system. Periodic inspections of individual components wiring is also recommended to reduce the chance of failure of the photovoltaic system.

Several products have emerged in the market to monitor the power generated by photovoltaic systems to lower the number of necessary onsite inspections. These products could be classified in two distinct groups: traditional string inverter power sensors and smart micro-inverters. The former technology suffers from a lack of precision because it ignores the health condition of every solar panel and instead conveys the information about the total string. Although this may still be helpful, pinpointing the fault will require potential onsite inspections. On the other hand, smart micro-inverters monitor the power of every panel in the system. Being fairly new to the market, this technology is very expensive and seems to be out of reach for most individuals. The high cost is explained by the necessity of placing an inverter for each solar panel and the complex software package that delivers the service.

This report presents the proof of concept of a solution to this problem. The developed unit is a customized webserver that logs the voltage data of every single module of the system along with the total voltage and current and makes them available within the Wide Area Network. This solution -compared to prior arts- is a less expensive alternative that may in the future enable any household to enjoy the advantages of solar energy to the fullest.

2.0 Background

This chapter is divided into two major sections. The first section covers some of the prior art while the second section deals with a study of the solar cell at the semiconductor level. The later section is not only reserved for this project but seems a promising base for research in the subject.

2.1 Prior Art

This section reports few of the emerging photovoltaic monitoring technology. This technology is basically developed to provide the user with a better insight of the performance of his/her photovoltaic system.

2.1.0 Fronius Datalogger easy/pro/Web

Fronius is one of the leaders in energy conversion in the United States of America. *Fronius Datalogger easy/pro* (Figure 1) collects the data from the inverter and sends it to a personal computer for further processing using a software such as *Solar.access*. The logger is large enough to save data over a period up to 3 years. Once connected to a modem, the remote monitoring featured is created (LLC, 2013). *Fronius Datalogger Web* on the other hand does need a modem. It uses Ethernet technology and acts as a web server that makes a website accessible to the public. The product price varies from $276.55 to $736.00 depending on specifications and vendors (Google, n.d.).

Figure 1: FRONIUS IG Datalogger Box (LLC, 2013)

2.1.1 BenQ/AUO Solar Monitoring System

The AUO monitoring package consists of a data logger device for real-time performance monitoring and web portal for Internet access to collected data. AUO is inclusively compatible with AC Unison solar power products. This package is sold at $616.95 by Ecodirect (ECODIRECT, n.d.).

Figure 2: BenQ AUO Solar 19.M2M01.002 (ECODIRECT, n.d.)

2.1.2 SolarEdge Zigbee-Ethernet Gateway

This product not only connects an inverter to a gateway point but also allows wireless connectvity among inverters themselves (CIVICSOLAR, wireless-communication-products).The cost of this communication product is of $451.50 (CIVICSOLAR, SolarEdge Zigbee-Ethernet Gateway , n.d.). Not that this product only provides connectivity and is not responsible for data logging.

Figure 3: SolarEdge Zigbee-Ethernet Gateway

These products are essentially used for power data logging. Some of them need an external modem for data publishing while others have communication parts charged of data transfer and monitoring. Although performing well, these products are not as attractive to the public due to their allocated prices in the market.

2.2 Photovoltaic System Modeling and Simulation

Before getting started in the design process of any monitoring system, an appropriate knowledge of the properties of the devices to be monitored is recommended. In this project, the individual modules of a photovoltaic system are the candidates to be examined. Based on the fact that a module is defined as an array of solar cells (usually connected in series) (Eckstein, 1990), the mathematical model of a solar cell is presented first. This chapter also compares few techniques used by scientists in order to extract photovoltaic parameters solely from a commercial datasheet. Those parameters are very important in the creation of a characteristic equation that would describe the I-V curve of the solar module in question. That is critical in visualizing the solar cell behavior under multiple physical conditions such as shading. Finally, a simulation of an entire photovoltaic system using MATLAB/Simulink is provided to confirm theoretical results and anticipate future system behaviors.

2.2.0 Solar Cell Model (Ideal and Realistic)

The ideal solar cell model consists of a current source connected in parallel with a diode (Rodrigues, Melicio, Mendes, & Catalao, 2011) as shown in Figure 4.

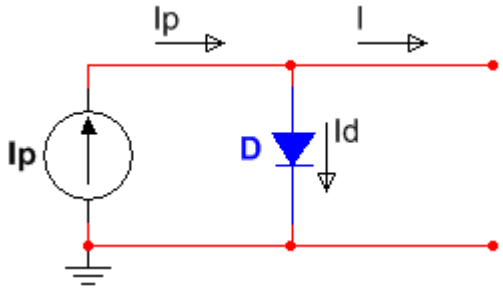


Figure 4: Ideal Solar Cell Model

The cell current I is formulated as:

$$I = I_P - I_D \tag{1}$$

where I_P is the photo-generated current, and I_D is the diode current.

A diode is simply a *pn* junction that consists of an *n*-type semiconductor containing free moving electrons being in contact with a *p*-type material where holes are dominating (Sedra & Smith, 2010). The diode current I_D is defined by the Shockley equation (Hambley, 2011) shown below.

$$I_D = I_s \left[\exp\left(\frac{qV_D}{nkT}\right) - 1\right] \tag{2}$$

where I_s is the saturation current, V_D is the voltage across the diode, n is the ideality factor, q is

the electrical charge magnitude of an electron $1.60217646 \times 10^{-19} C$, k is the Boltzmann's constant $1.386503 \times 10^{-23} J/K$, and T is the junction temperature in Kelvin.

The diode operates in three major regions. The diode is said to be forward-biased when its voltage is positive. In this region, the current increases exponentially in function of the voltage as it is depicted in equation (2). When V_D is negative, the diode is reverse-biased and maintains a relatively low and negative constant current until it enters the reverse-breakdown region where the diode current decreases exponentially (Hambley, 2011).

When a solar cell is exposed to light, photons with energy greater than the band gap energy of the diode excites the semiconductor, thus forming electron-hole pairs. These pairs are separated by the internal electric field of the diode to create an electric current that is proportional to the provided light intensity (Walker, 2001). Combining equations (1) and (2), the output cell current is represented as follows:

$$I = I_P - I_s \left[\exp\left(\frac{qV_D}{nkT}\right) - 1 \right] \tag{3}$$

From equation (1), it can be deduced that the I-V curve of a solar cell is formed by reflecting the I-V curve of its modeling diode across the horizontal axis (voltage) and up-shifting it by the value of the photo-generated current as shown in the figure below.

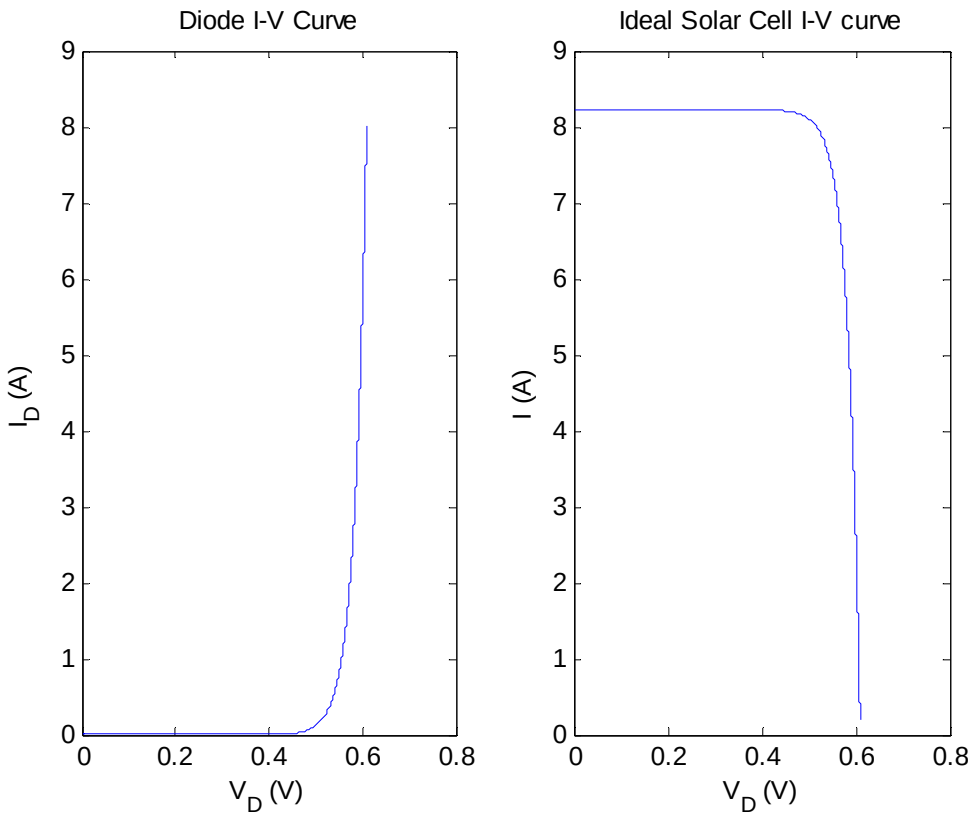


Figure 5: General I-V Curve for Diode and Solar Cell

In reality however, there are power losses due to various reasons such as the material resistivity, contact levels (Rekioua & Matagne, 2012), *p-n* junction leakages, crystal defects and impurities (Kaiser, 2010). Series and Shunt resistances are therefore added to the model to account for the voltage and current losses respectively. A more realistic model is shown in the figure below.

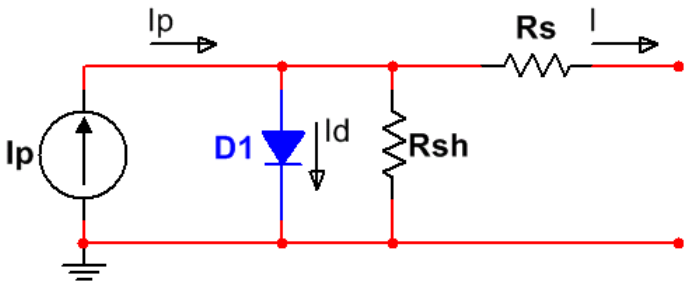


Figure 6: Realistic Solar Cell Model

Using Kirchhoff's current law, the solar cell characteristic equation (Eq. 3) becomes:

$$I = I_P - I_s\left[\exp\left(\frac{V_D + IR_s}{V_T n}\right) - 1\right] - \frac{V_D + IR_s}{R_{SH}} \quad (4)$$

Where V_T is the junction thermal voltage defined as $\frac{kT}{q}$.

2.2.1 Extraction of Solar Cell Parameters from Datasheet

In the previously developed model, the I-V characteristic of a solar cell is based on five parameters $(I_P, I_s, n, R_s, R_{SH})$. Why are these parameters so important?

Most of solar modules datasheets provide four main parameters: short-circuit current I_{sc}, open-circuit voltage V_{oc}, maximum current and voltage points (I_{mpp}, V_{mpp}) (Rekioua & Matagne, 2012). Most of the time, an I-V curve is also provided at nominal temperature and irradiance and other few selected test conditions. Therefore, predicting the value of a specific V-I point at any other condition would not be error-free. In fact, the accuracy of the approximation is limited by the defined step size of both horizontal and vertical axes of the provided I-V graph and the corecteness level of the used extrapolation method. Finding the parameters necessary to define the I-V characteristic equation solves this problem.

One of the experimental methods used to extract those patrameters is the measurementt of the solar cell module I-V values at multiple illumination levels (Bouzidi, Cheggar, & Bouhemadou, 2007). That requires the use of laboratory equipment and the hassle of handling solar panels that could be larger depending on the application. Other common methods is to use the following relations (Technologies, 2012) to calculate the series and shunt resistances.

$$R_{SH} = \frac{\Delta V_{sc}}{\Delta I_{sc}} \quad (5)$$

$$R_S = \frac{\Delta V_{oc}}{\Delta I_{oc}} \quad (6)$$

The above expressions are calculated by estimating the slope of the datasheet I-V curve at the open-circuit and short-circuit points. This method does not require the use of laboratory equipment but is not freed of potential measurements inaccuracies. Adding to that, the photo-generated current I_P is usually approximated to the short-circuit current I_{sc} (El Tayyan, 2012) and the reverse saturation current I_s is estimated by the following equation (Villalva, Gazoli, & Filho, 2009):

$$I_s = \frac{I_{sc}}{exp\left(\frac{V_{oc}}{nV_T}\right) - 1} \quad (7)$$

All the methods mentioned above are acceptable but show a great deal of inconvenience (i.e. the necessity to take module measurements) and uncontrolled parameter estimations. The problem resides in the fact that multiple parameters are approximated. That does not seem robust especially when parameter evaluations depend on other estimated variables. To take into account those variable dependences, it is logical to evaluate a system of five equations to solve for the five unknowns parameters. Unfortunately, semiconductor physics does not offer us clean linear equations where linear matrix theory techniques could easily find the system's solutions. Instead, nonlinear and transcendental equations such as (4) were developed. The rest of this section presents a numerical solution proposed by the authors in (Villalva, Gazoli, & Ruppert, 2009) that produces an I-V curve based on datasheet parameters. The suggested method is commented along its technical description with exterior arguments that may not necessarily reflect the motivations of the authors in (Villalva, Gazoli, & Ruppert, 2009).

The ideality factor n, also known as the emission coefficient is usually bounded between the values 1 and 2 (Hambley, 2011). The factor value $n \in [1, 2]$ governs the curvature of the semiconductor characteristic curve (Carrero, Amador, & Arnaltes, 2007). Nevertheless, 1 is an adequate modeling choice (Alonso, 2005). The algorithm runs on the former condition with the idea that curve fitting techniques could be used at the end for accuracy purposes. The series resistance lowers the voltage that is originally produced by a value no higher than the difference between open-circuit and maximum-point voltages. For that reason, $R_S \in \left[0, \frac{V_{oc,n} - V_{max}}{I_{max}}\right]$. After each iteration, R_S obtains a new value and becomes $R_S + x$. Meanwhile, the parallel resistance R_P is solved so that there is only one set (R_S, R_P) that satisfies the existence of the maximum point. Equation (4) is multiplied by the maximum-point voltage V_{max} to be set equal to the experimental maximum power point (datasheet). Once arranged, the equation provides a new definition of the shunt resistance that is:

$$R_P = \frac{V_{max}(V_{max} + I_{max}R_S)}{V_{max}I_p - V_{max}I_s exp\left(\frac{V_{max} + I_{max}R_S}{N_S n}\frac{q}{kT}\right) + V_{max}I_s - P_{max}} \tag{8}$$

where N_S is the number of cells in series of the solar panel in question.

Since the characteristic equation (4) is transcendental, it is impossible to formulate an analytical solution where the voltage V is on one side, and the current I on the other side. Instead, Newton-Raphson method is used to solve for $I = Eq.(4)$ over a reasonable range of V values. Using a current divider in the circuit shown in Figure 6, the nominal photo-generated current is defined as follows:

$$I_{P,n} = \frac{(R_S + R_P)}{R_P} I_{SC} \tag{9}$$

Equation 7 is strengthened to become:

$$I_s = \frac{I_P - \frac{V_{oc}}{R_P}}{exp\left(\frac{V_{oc}}{nV_T}\right) - 1} \tag{10}$$

Note that all of the above calculations solve for the five parameters at nominal temperature and irradiance conditions ($T = 25°C + 273.15$ and $G = 1000W/m^2$). The photo-generated current is both linearly dependent on the light irradiance and affected by temperature change (De Soto, Klein, & Beckman, 2005):

$$I_P = \left(I_{P,n} + k_I \Delta_T\right) \frac{G}{G_n} \tag{11}$$

where $I_{P,n}$ is the nominal calculated light generated current, k_I the current-temperature factor, Δ_T the difference between actual and nominal temperatures, $G\ and\ G_n$ the actual and nominal irradiances.

The short circuit dependence on temperature could be seen as follow:

$$I_s = \frac{I_P k_I \Delta_T - \frac{V_{oc} k_V \Delta_T}{R_P}}{exp\left(\frac{V_{oc} k_V \Delta_T}{nV_T}\right) - 1} \tag{12}$$

where k_V is the voltage-temperature coefficient [%/°C].

2.2.2 Photovoltaic System Simulation (Kyocera KCT85T in MATLAB)

KCT85T is a solar panel manufactured by Kyocera, a multinational electronics and ceramics manufacturer headquartered in Kyoto, Japan. This section will use the numerical methodology previously described to arrive at a complete mathematical model of a photovoltaic system. The datasheet information of KC85T 87.348W max. power (Table 1) is used to simulate the I-V behavior of the KC85T panel at selected temperature and irradiance levels.

Table 1: Electric Specifications of KC85T under STC (KYOCERA)

Maximum Power (Pmax)	87.348W
Maximum Powe Voltage (Vmpp)	17.4V
Maximum Power Current (Impp)	5.02A
Open Circuit Voltage (Voc)	21.7V
Short Circuit Current (Isc)	5.34A
Temperature Coefficient of Voc	-8.21e-2V/°C
Temperature Coefficient of Isc	2.12e-3A/°C

Knowing that the present module contains 36 series cells, the I-V model of the module at nominal conditions is shown below.

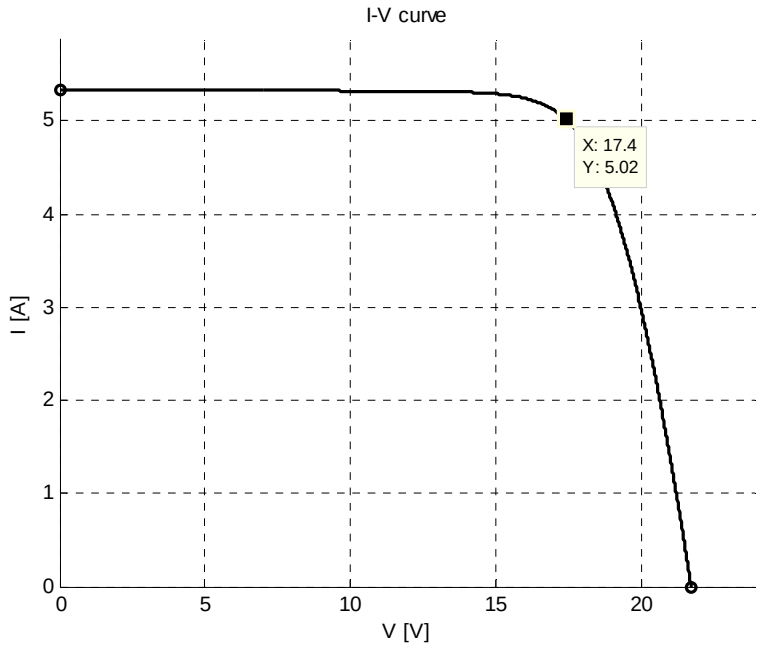


Figure 7: Calculated I-V Model of KC85T at STC

Note that the critical maximum point (17.4V, 5.02A). More precisely, not that the maximum power derived from the model is 87.348002W, a slight 2.2897e-6 percent error from the maximum power listed on the datsheet. By studying the generated curve however, both of the short and open circuit pairs appear to be exact: (0V, 5.34A) and (21.7V, 0A) respectively. At this nominal situation, the five parameters are calculated to be:

Table 2: Calculated Parameters of KC85T at STC

Rs	Rp	Ip	Is	n
0.3223Ω	620.060527Ω	5.342792V	3.43442e-10	1

Taking a step further, the I-V curve is modeled at 1000W/m^2 and other selected temperatures ($50°C$ and $75°C$) using equations (11 & 12) to recomputed the light generated and saturated currents in equation (4).

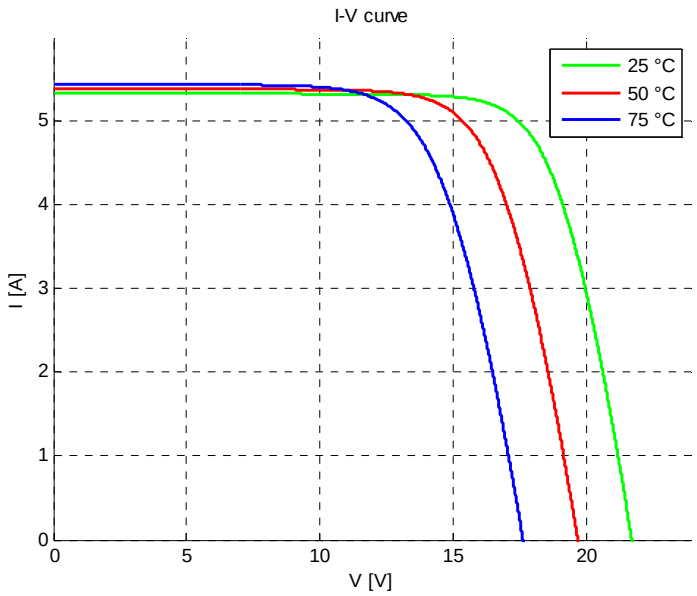


Figure 8: Calculated I-V Model of KC85T at 25, 50, and 75 degrees Celsius

The experimental I-V curve present in the datasheet (Figure 9) is very similar to the modeled curve shown above which highlights the accuracy of the used model.

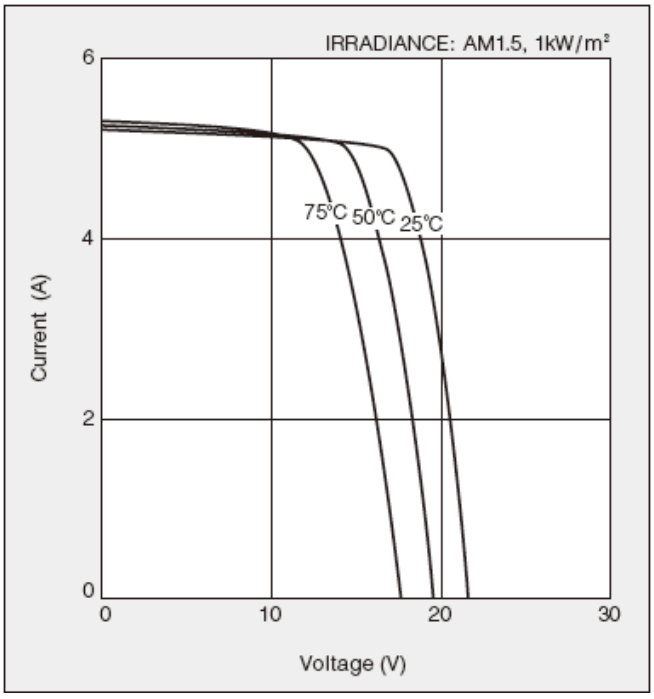


Figure 9: Experimental I-V Curve of KC85T at 25, 50, and 75 degrees Celsius (KYOCERA)

Equations (11 & 12) are now used to modify the model at $25°C$ and different irradiance levels: 1000, 800, 600, 400, and 200W/m^2.

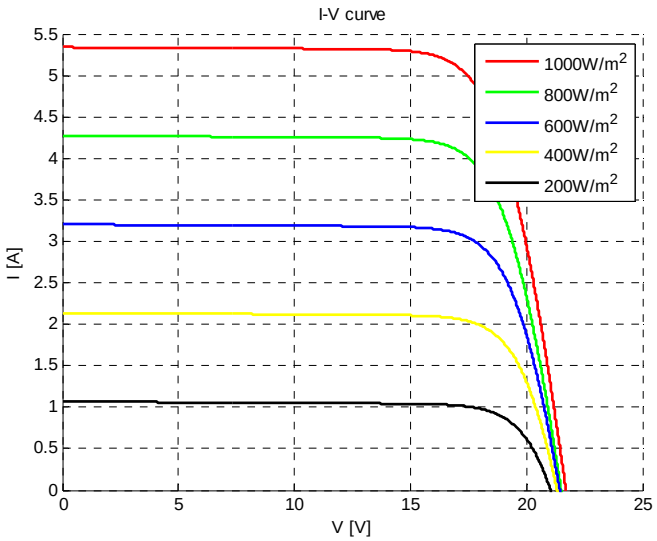


Figure 10: Calculated I-V Model of KC85T at selected irradiance levels

Again, the experimental I-V curve present in the datasheet (Figure 11) seems to be similar to the modeled curve shown above, which gives more confidence in this model.

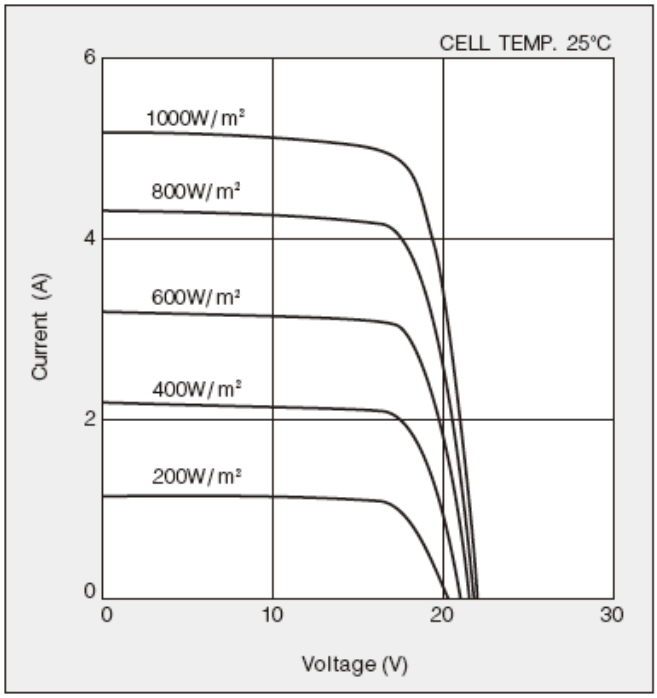


Figure 11: Experimental I-V Curve of KC85T at selected irradiance levels (KYOCERA)

This modeling technique has proven to be accurate to model solar module behaviors and characteristics. This model could be very useful in this project as it could determine and predict the I-V behavior of the photovoltaic system while exposed to different levels of temperature and irradiance.

3.0 Design

This chapter describes the design of the hardware and software aspects of the power monitoring system prototype.

3.1 Hardware: Physical and Electrical Layout

This section shows the hardware structure of the prototype. The prototype consists of a photovoltaic array whose voltage and current are measured using a microcontroller-based system. The microcontroller is also used to store the data and display them online. The data storage and web application development are discussed in a later section.

3.1.0 Solar Panels and Physical Setup

Photovoltaic panels are usually connected both in series and parallel to provide a desired voltage and current according to the application. As a physical model, 6 photovoltaic panels have been chosen to form 2 parallel strings of 3 panels connected in series as shown in the figure below.

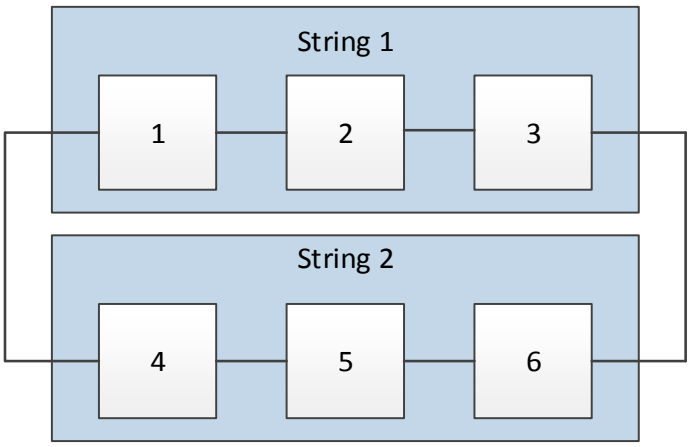


Figure 12: Structure of the Photovoltaic System Model

The solar panel used in this model is a PARALLAX 750-00031 outputting a maximum of 9 DCV at 1W with the dimensions of 135x135mm. The datasheet is attached in Appendix A1. An acrylic based support of about 508x355mm was manufactured to hold the panels in place as shown below.

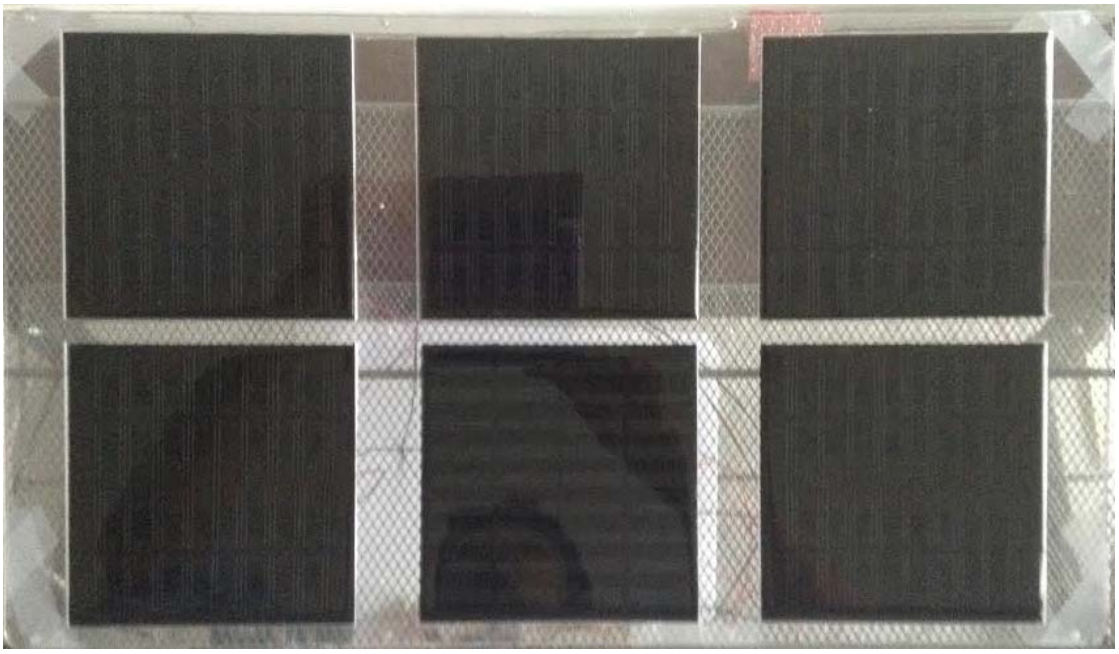


Figure 13: Solar Panels mounted on Acrylic Frame

The solar panels are connected to a breadboard circuitry while the whole prototype is supported by a rack for stability and transportation purposes.

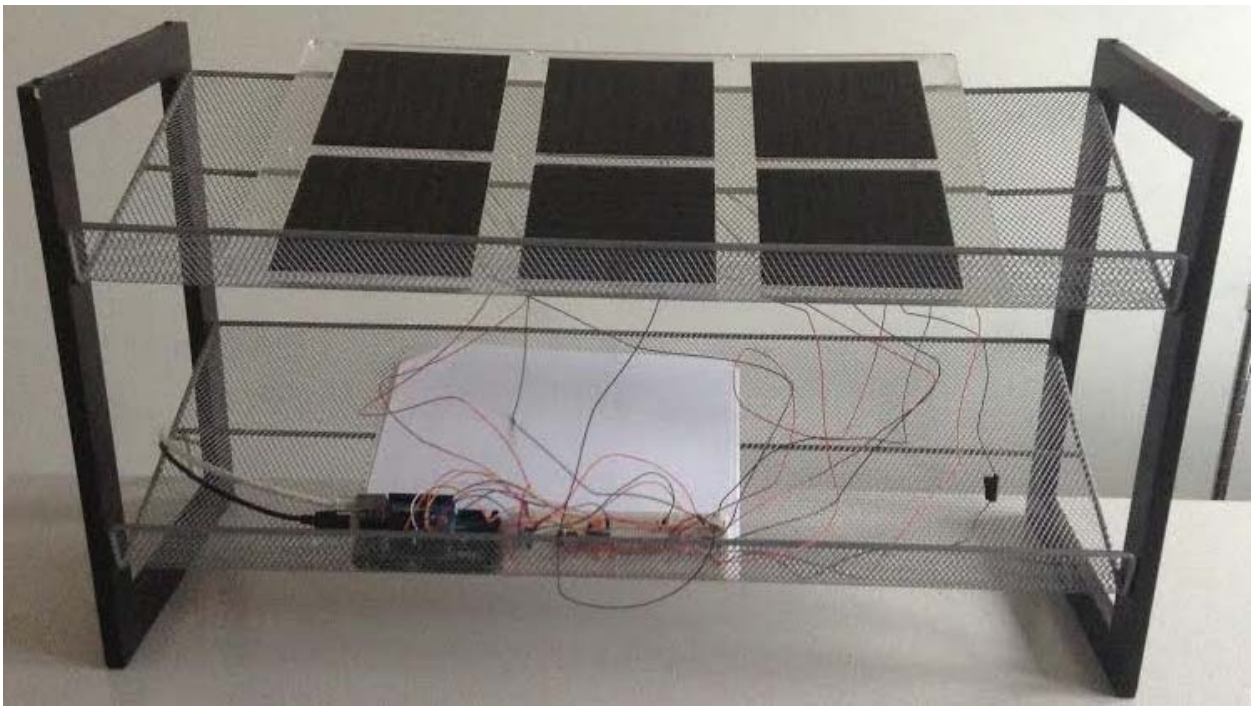


Figure 14: Experiment Rack

3.1.1 Analog Circuitry, Boards, and Parts

The overall idea is connecting the photovoltaic system to a testing load and recording the voltage levels of the individual solar panels and the current going through the load. A microcontroller is responsible of reading the system data and processing them. The microcontroller board used in this project is the Arduino Mega 2560 (Figure 15) whose specifications are shown below.

Table 3: Microcontroller Specifications (Arduino, n.d.)

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54
Analog Input Pins	16
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Analog Input Resolution	10 bits (1024 values)

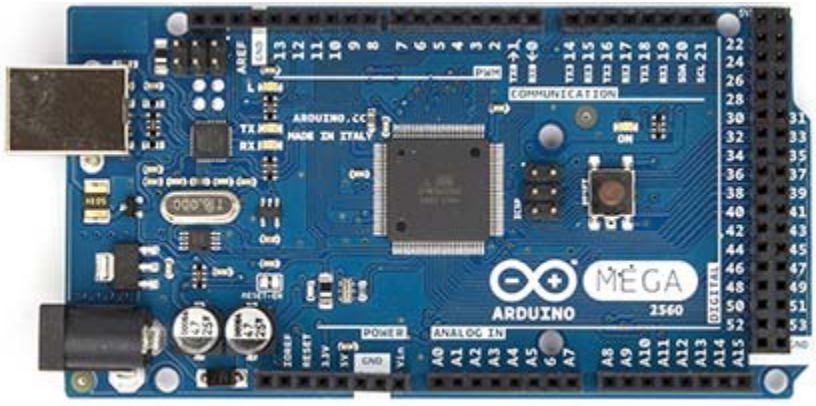


Figure 15: Arduino Mega 2560 Top View

The individual voltage of all six panels is recorded by connecting the positive terminals of the solar modules to the corresponding analog inputs of the microcontroller board. The following circuit shows the enumerated nodes at which the voltage could be captured.

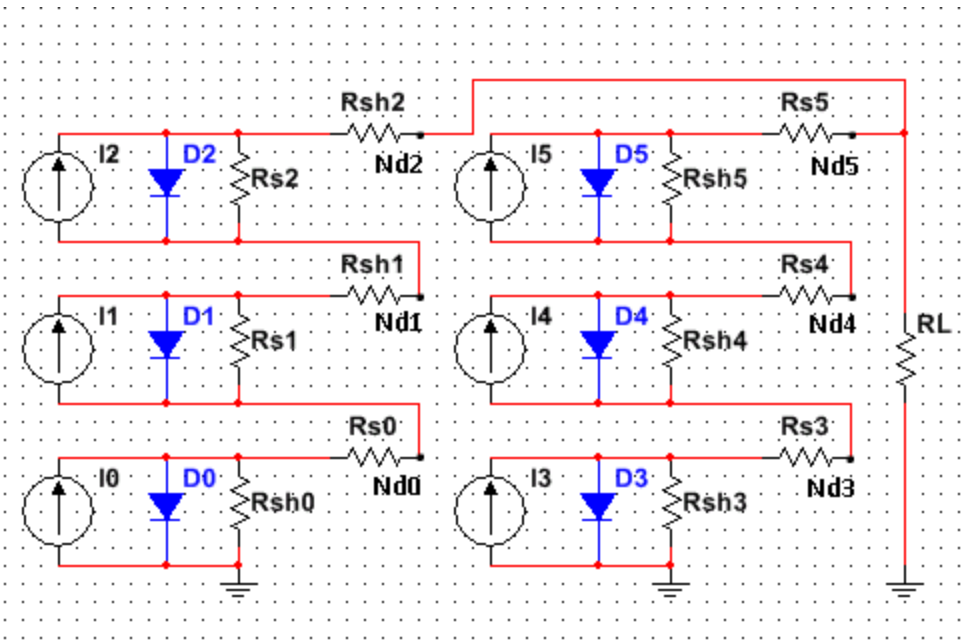


Figure 16: PV Circuit and Measurement Nodes

RL is the testing load resistance representing an actual load that could be connected to the photovoltaic system in real life. Since the Arduino analog input pins voltage maximum rating is only 5V and the maximum output voltage of a singular panel is 9V, a voltage divider should be used to avoid the device malfunction or/and ruin.

Let Vpmax be the maximum outputted voltage by each solar panel (9V).

Below are listed the maximum voltage values at the different nodes:

Table 4: Maximum Voltage at the Circuit Nodes

Node	Maximum Voltage
Nd0	9V
Nd1	2×9V=18V
Nd2	3×9V=27V
Nd3	9V
Nd4	2×9V=18V
Nd5	3×9V=27V

The voltages at the six nodes raise concern as they are higher than 5V (9V; 18V; 27V). A solution would be placing a voltage divider at those nodes (Nd0-5). Let the voltage divider consist of two resistors Rvd1 and Rvd2 with the output voltage being $V_o = V_{in} \frac{Rvd1}{Rvd1+Rvd2}$. The impedance fraction should ideally be 0.185 (27V/5V) in order to take advantage of the ADC full resolution. The following resistor values would be used according to availability:

Rvd1=100kΩ.

Rvd2=330kΩ+100kΩ+10kΩ+2.2kΩ=442.2kΩ.

The fraction value therefore becomes 0.184 (100/542.2) which is very close to the ideal value of 0.185, thus allowing us to safely measure the voltage at those high nodes while taking advantage of the full resolution range of the ADC. Note that 27V is the maximum open voltage of the three-in-series modules while exposed to maximum light irradiance. As long as there is a load connected to the photovoltaic system, 27V should never be an expected total voltage.

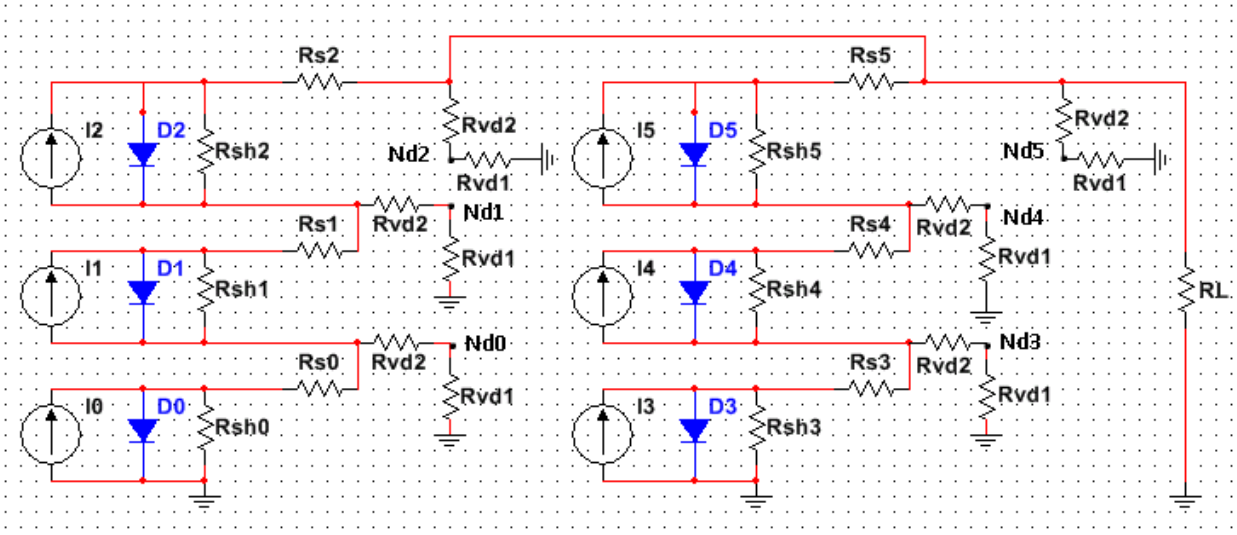


Figure 17: PV Circuit and measurement Nodes with Voltage Dividers

The voltage issue set aside, the current concern yet stands. The maximum current (short circuit) is 111.11 mA which is high compared to the DC current per I/O (40mA). A good way to solve this issue is the use of a voltage follower that would not let the current run from the photovoltaic system into the analog pins.

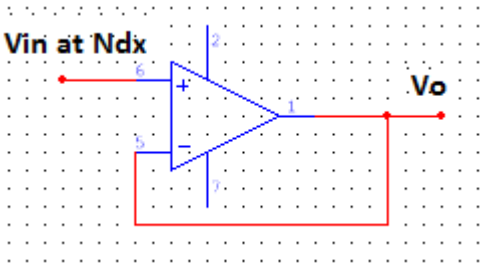


Figure 18: Voltage Follower Circuit

A voltage follower (buffer) is built by using an operational amplifier by tying its negative input to the output, thus creating a negative feedback loop. Due to high input impedance, no current (ideally) shall flow into the amplifier. The output voltage would also adjust so that both of the op-amp inputs voltages match. Now, whether, the output of the voltage divider and buffer circuit

are directly connected to the analog pins of the MCU or have to go through a shift register, isolation is insured.

An appropriate operational amplifier would be operating at low single supply voltage since the Arduino board only offers 3.3V and 5V output pins. It should also be rail-to-rail input/output so that 0-5V remains the range of the collected voltage values. MCP600 (Appendix 3) seem to meet those qualifications. Since six nodes are to be treated, 2 quad MCP6004 chips would be used.

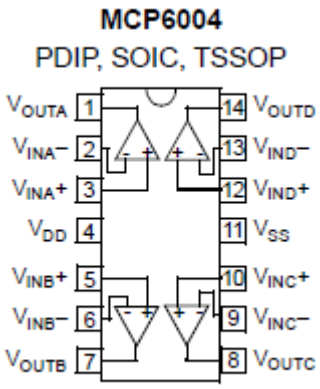


Figure 19: MCP6004 Amplifier Pin Diagram

V_{DD} being the positive voltage supply would be connected to the Arduino 5V output and V_{SS} being the negative input supply would be connected to ground (single supply). V_{IN+} would be connected to the appropriate node while V_{IN-} would be connected to V_{OUT} to create a negative feedback.

The following table shows the pins correspondence of the project prototype where the blue cells represent pins from one MCP6004 and the violet ones the second MCP6004 chip.

Table 5: PV System Monitoring Pin Connections

Solar Panel Number	Circuit Node	MCP6004 Input Pin	MCP6004 Output Pin	MCU Analog Inputs
0	Nd0	VinC+	VoutC	A0
1	Nd1	VinD+	VoutD	A1
2	Nd2	VinC+	VoutC	A2
3	Nd3	VinD+	VoutD	A3
4	Nd4	VinB+	VoutB	A4
5	Nd5	VinA+	VoutA	A5

The table above applies to the following solar module alignment.

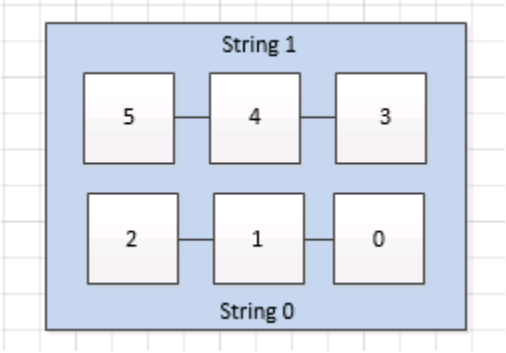


Figure 20: PV System Solar Module Alignment

A last change to the analog circuitry is the addition of bypass and blocking diodes to reduce the effects of shading. A bypass diode (BpD) is connected in parallel to every solar panel. That way, in case a panel is totally shaded, the series circuit would not be broken and current would travel through the corresponding bypass diode. A blocking diode (BlD) is connected in series with each branch (string) to ensure that the current travels in only one direction in that particular string (from the positive terminal of module 2 and 5 to the load resistor). Otherwise, in case of shaded photovoltaic strings, current could travel from one string to the other and cause electrical failure.

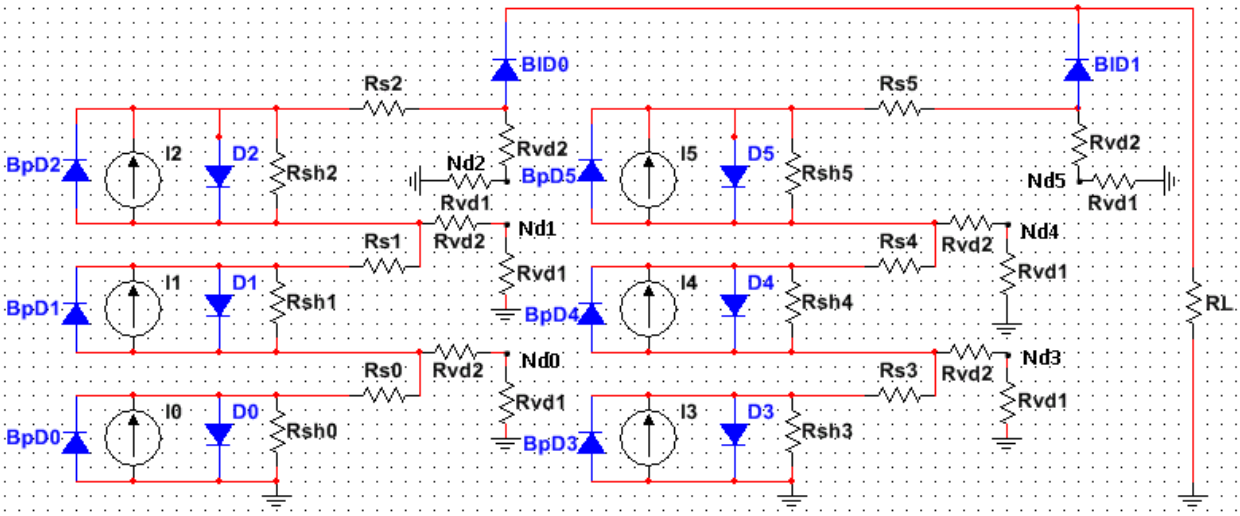


Figure 21: PV circuit with Bypass and Blocking Diodes

The bypass and blocking diodes used in this circuit are 1N4148 (Datasheet in Appendix3). 1N4148 was chosen because of its parameters:

Reverse Voltage (VR): 75V
Average Forward Current (AFC): 150mA

VR is higher than the maximum module voltage diode min. rating(DMR) (9*1.25=11.25V) and the total maximum branch voltage DMR (27*1.25=33.75V). AFC is also higher than the maximum branch current DMR (111.11*1.25=138.89mA).

3.1.2 The Internet of Things

To connect the Arduino to the Internet, an important piece is necessary which is the Arduino Ethernet Shield shown below.

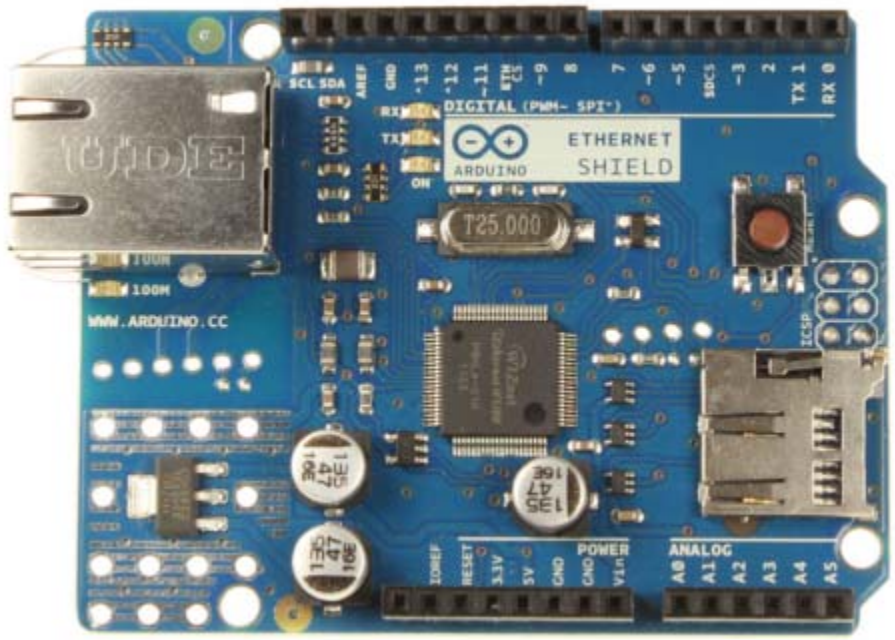


Figure 22: Arduino Ethernet Shield Top View (Arduino, Arduino Ethernet Shield, 2015)

This shield is to be connected on top of the Arduino Mega board using SPI port. It is also connected to the router through an RJ45 cable. Its operational voltage is 5V which is provided by the Arduino board itself. This shield has slot that could potentially host a micro-SD card to log all the monitored data in this project prior to online viewing.

3.2 Software: Data Processing and Monitoring

The big picture is to program the Arduino microcontroller to record the desired data from the circuitry, process it, and create a webserver to host the real-time data and a display its history graph. There are several ways to achieve this goal. An easy way is the use of a service such as Plotly: an online data visualization tool, that takes care of plotting the data once received from the Arduino through an Ethernet connection. The downside of this solution is that the user loses the luxury to design his/her own webserver since the graph is only viewed on the Plotly website. It appears best then to opt for a more self-contained option such as the one developed by (Everett, n.d.). The overall idea is to write an Arduino sketch that stores the desired data into an SD card. An html file is used to design the webserver that lists the SD data files and displays the data graph upon a click. The example operates with one sensor reading. It was modified to display all of the six needed readings. The server design has been altered as well to serve multiple web pages.

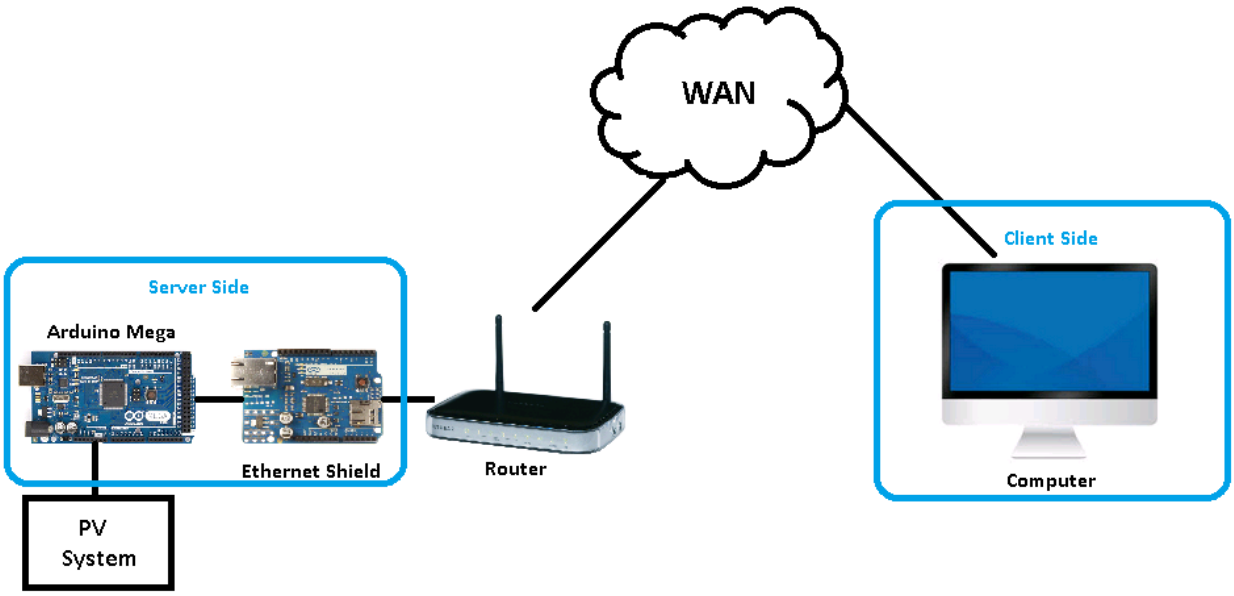


Figure 23: Communications Diagram

Two separate codes are used in this project:

- Arduino sketch: OPVMS.ino
  This code is responsible of capturing the data, string them in an SD card, and act as a webserver.
- HTML code: HC.htm
  This code is stored in the SD card and uses Highcharts.js to chart the data strored in the SD card.
- Other minor HTML codes are used to create other web pages such as the Project Description and the People involved in the project.

3.2.0 Arduino sketch summary

The Arduino sketch is composed of six main functions. The Setup function initializes the system such as the SD card, Ethernet connection defined by the MAC address of the Ethernet shield 90:A2:DA:0D:7B:B6 and the designed server IP address 192.168.1.177. Note that "177" was a random number allocated to the new "machine" on the network (the newly created webserver). This function also initiates the UDP port to allow access to an NTP (Network Time Protocol) server and get a synchronized time. The HtmlHeader functions are responsible of sending "OK" or "404" messages according to necessity. For example, if the user clicks on a link that no longer exists, an HtmlHeader404 function would be ready to send the 404 (Not Found) message. The SendNTPpacket sends requests to the NTP server address. That is necessary for the function of the getTime function that return the epoch time after parsing the UDP packed received from the NTP server. The Loop function is known to be the heart of the sketch. All major data processing occurs in it and all functions are called and used from it. More specifically, the analog pins are monitored and the individual solar modules voltages and current calculated. This information

along the epoch time is then sent as a string to a file in the SD card in the data folder. There is a file for every day. A string of data is added to its corresponding file every n seconds (in this case, n=1 for debugging and troubleshooting purposes). The Arduino sketch is available in Appendix 4.

3.2.1 HTML codes summary

HC.htm is the principal HTML code of the project as it is responsible for extracting the data from the SD card data files and parsing it into separate series (Time, V0, V1, V2, V3, V4, V5, I). It also calls Highcharts javascript to display this data into two different charts for the pleasure of the user. Home.htm is the file that displays the project description. People.htm lists the people involved in the project (Prof. Susan Jarvis: Advisor; Ali Magzari: Student WPI'15). HTML codes are available in Appendices 5-7.

3.2.2 Router Configuration

Since the Ethernet shield is connected to the home router for internet access, the router was configured to make the webserver accessible to the WAN (Wide Area Network). The home router was accessed and was set to service port forwarding at port 8081.

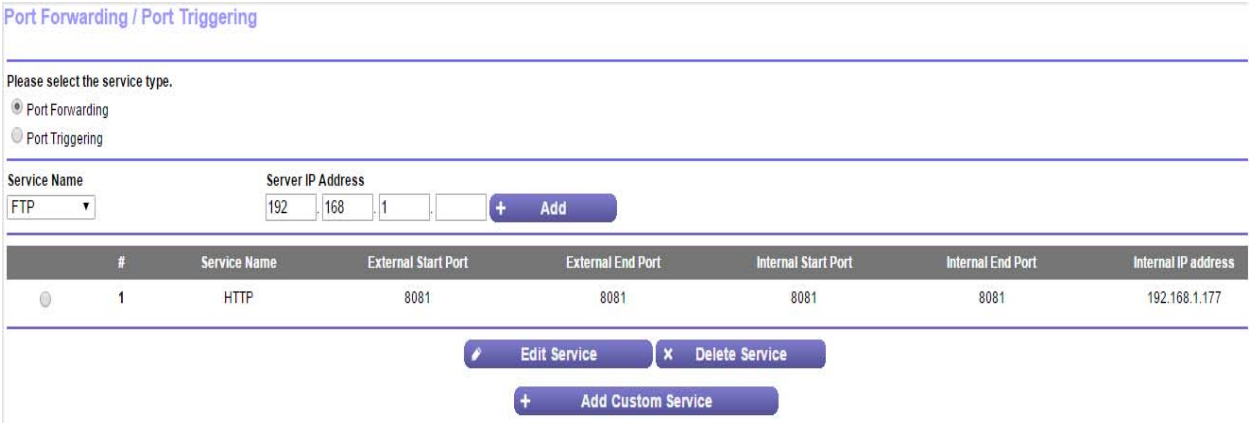


Figure 24: Router Port Forwarding Configuration

4.0 Results

This section showcases the results and achievements of this project.

4.1 Default Page

The webserver is accessible at LocalIpAddress:ForwardingPort: XX.XXX.XX.XXX:8081. The local IP address was hidden for security reasons. Once that address is typed, the following webpage appears.

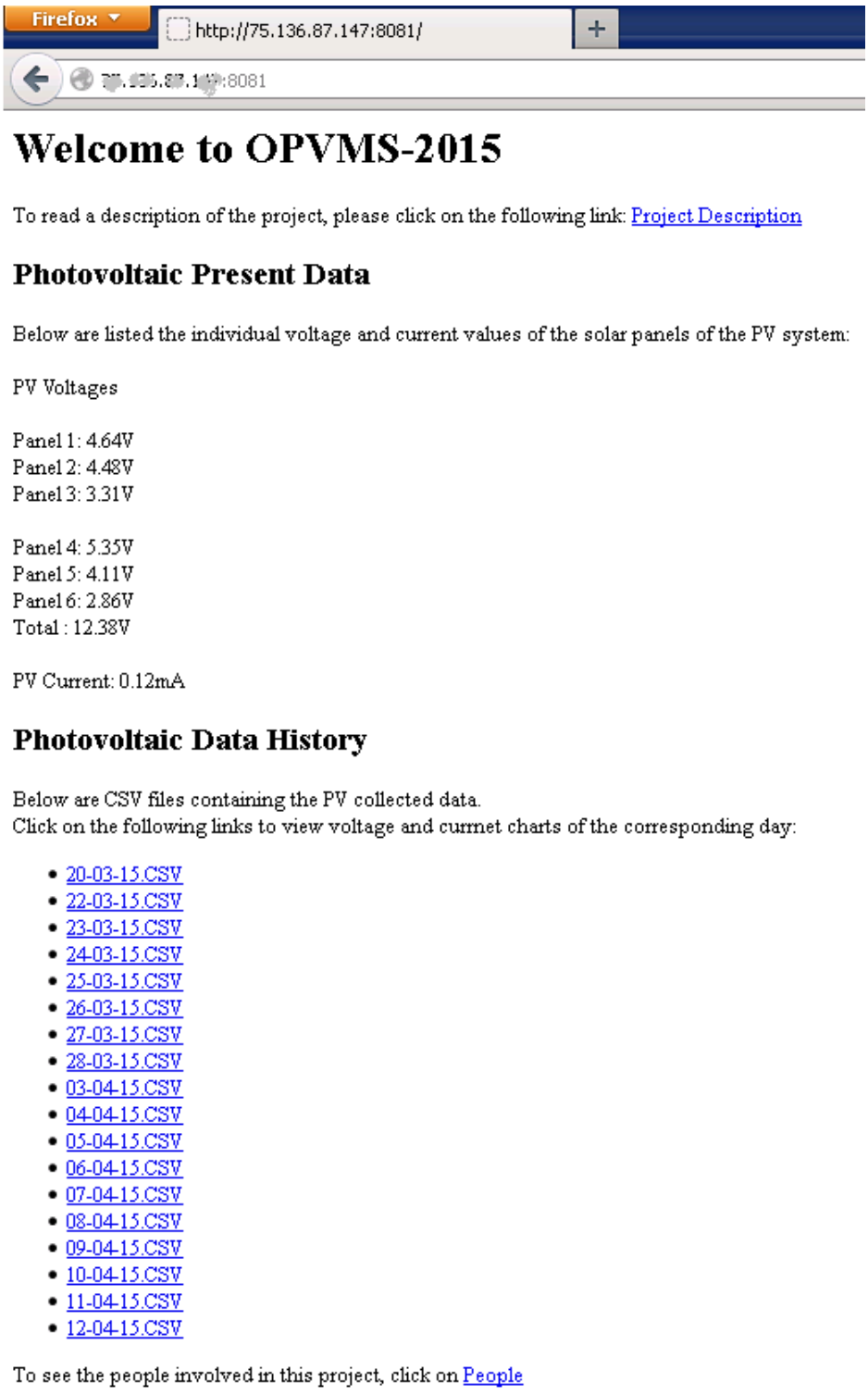


Figure 25: Webserver Default Page

The default page has two main parts: the photovoltaic present data and data history. The present data lists the real-time voltages and current of the PV system whereas the Data history lists daily CSV data files. Upon a click, the history charts are displayed.

4.2 Photovoltaic Data History Page

As a proof of concept, an experiment was run. The photovoltaic panels were set facing the windows of the home in which this project saw light for an entire 24 hour period on the 04/08/2015 from 00:00 to 23:59.

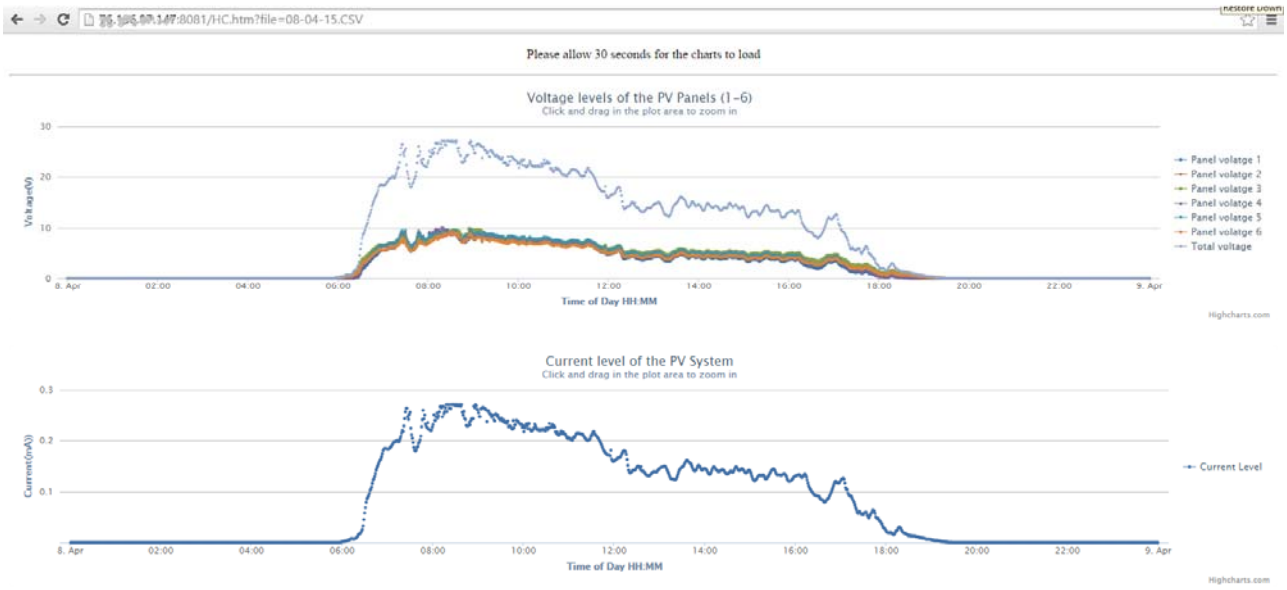


Figure 26: 04/08/2015 Data Chart

As shown in the figure above, two charts are loaded into the page. The top one display the individual voltages of the solar panel and the total voltage versus time. The second chart displays the current of the PV system versus time.

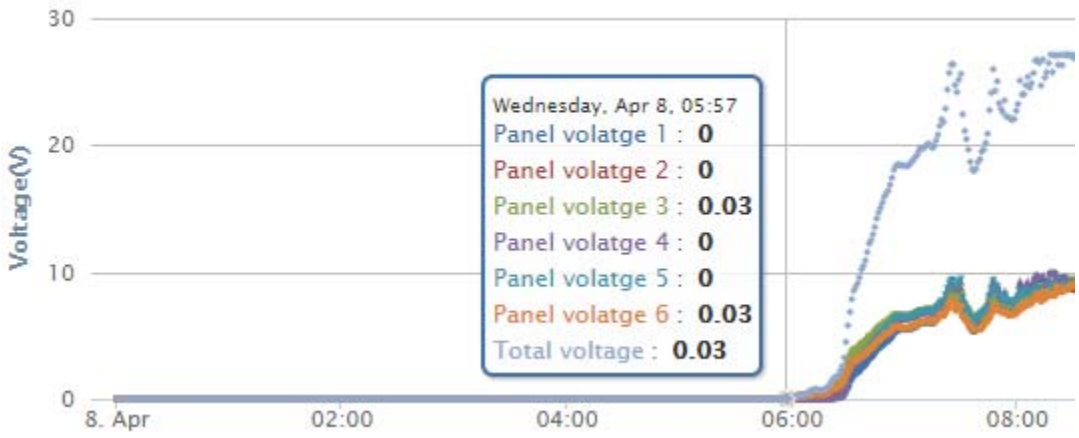


Figure 27: Voltage Values on 04/08/2015 at 05:57

Note that the voltage values from 00:00 where all zero due to the darkness until 05:57 where two panels started producing low voltages. That could be explained by the start of civil twilight that occurred at 05:50 on that day (timeanddate, 2015). The voltage values then rise to finally reach the 0.00V mark at 19:29, eight minutes after sunset (19:21) and twenty minutes before the end of civil twilight (19:49). All voltages after that point are null due to darkness.

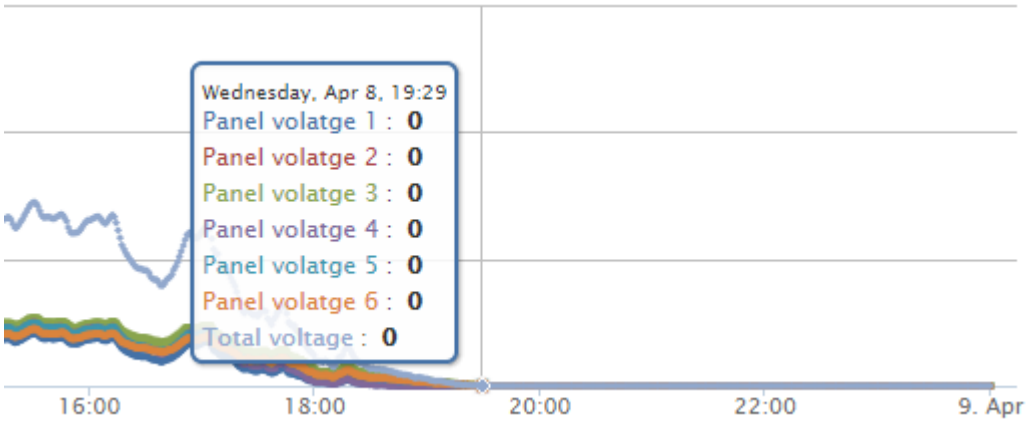


Figure 28: Voltage Values on 04/08/2015 at 19:29

The highest recorded total voltage value is recorded at 08:19 (27.11V). overall the highest voltage values are recorded between 08:00 and 09:00.

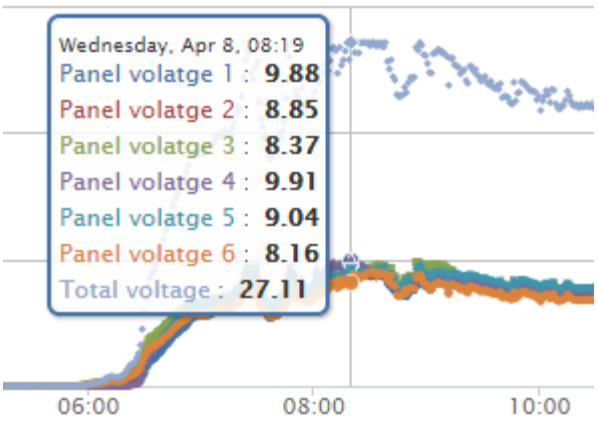


Figure 29: Voltage Values on 04/08/2015 at 08:19

As mentioned, the solar panels were kept inside the house (due to rain that could ruin the unprotected breadboard wiring) facing the east windows. It explains why the highest voltage values were not recorded around solar noon as it could have been if the panels were installed on the roof. Due to the roof obstruction, the highest solar irradiance level that would penetrate the room through the east windows would be in the range of 8:00 and 9:00 as shown in the diagram below (SunEarthTools, 2015).

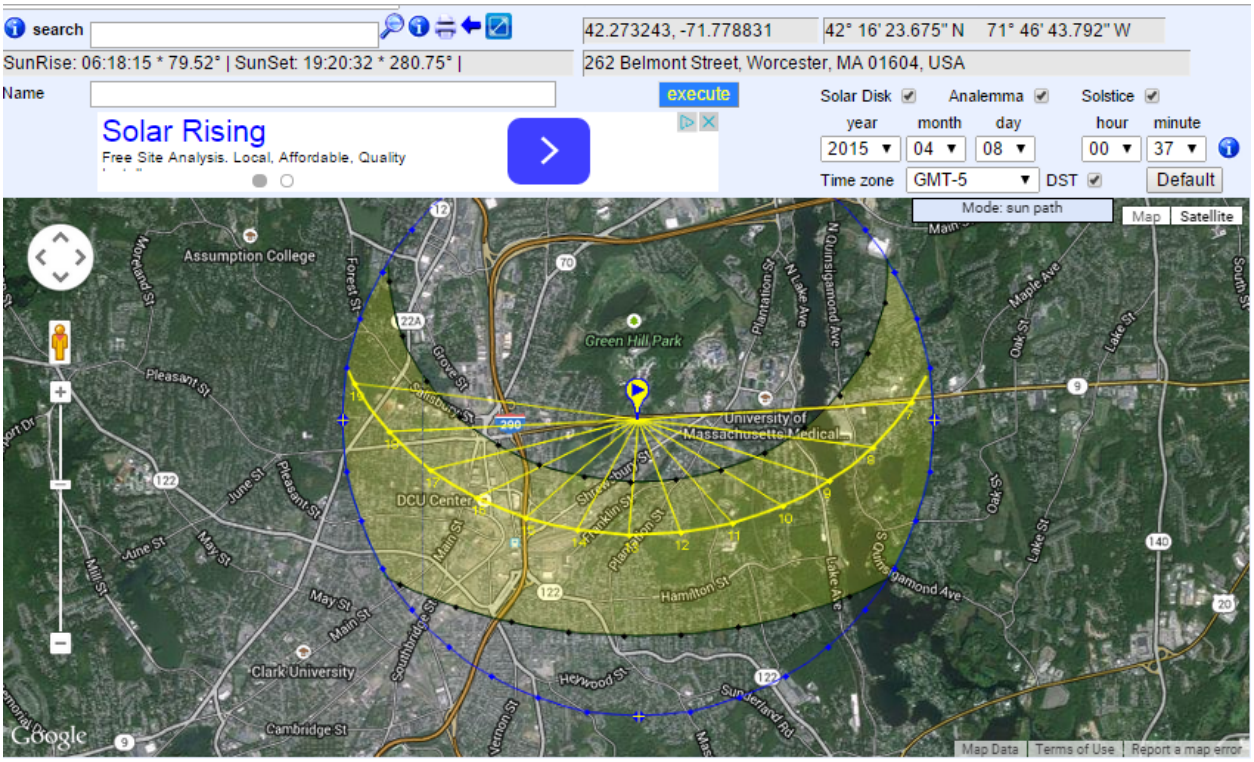
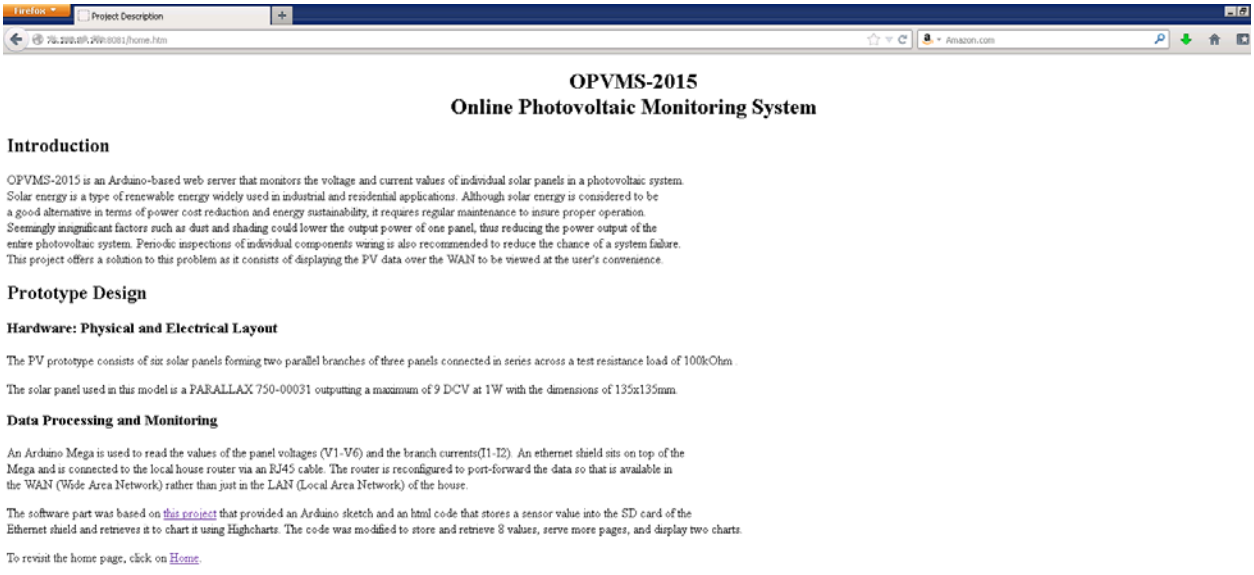


Figure 30: Solar Path on 04/08/2015

4.3 Project Description Page

From the default pages, a ink could be clicked to view the project description.



OPVMS-2015
Online Photovoltaic Monitoring System

Introduction

OPVMS-2015 is an Arduino-based web server that monitors the voltage and current values of individual solar panels in a photovoltaic system. Solar energy is a type of renewable energy widely used in industrial and residential applications. Although solar energy is considered to be a good alternative in terms of power cost reduction and energy sustainability, it requires regular maintenance to insure proper operation. Seemingly insignificant factors such as dust and shading could lower the output power of one panel, thus reducing the power output of the entire photovoltaic system. Periodic inspections of individual components wiring is also recommended to reduce the chance of a system failure. This project offers a solution to this problem as it consists of displaying the PV data over the WAN to be viewed at the user's convenience.

Prototype Design

Hardware: Physical and Electrical Layout

The PV prototype consists of six solar panels forming two parallel branches of three panels connected in series across a test resistance load of 100kOhm.

The solar panel used in this model is a PARALLAX 750-00031 outputting a maximum of 9 DCV at 1W with the dimensions of 135x135mm.

Data Processing and Monitoring

An Arduino Mega is used to read the values of the panel voltages (V1-V6) and the branch currents(I1-I2). An ethernet shield sits on top of the Mega and is connected to the local house router via an RJ45 cable. The router is reconfigured to port-forward the data so that is available in the WAN (Wide Area Network) rather than just in the LAN (Local Area Network) of the house.

The software part was based on this project that provided an Arduino sketch and an html code that stores a sensor value into the SD card of the Ethernet shield and retrieves it to chart it using Highcharts. The code was modified to store and retrieve 8 values, serve more pages, and display two charts.

To revisit the home page, click on Home.

Figure 31: Project Description Page

4.4 People Page

From the default and project description page, the following people page could be viewed.

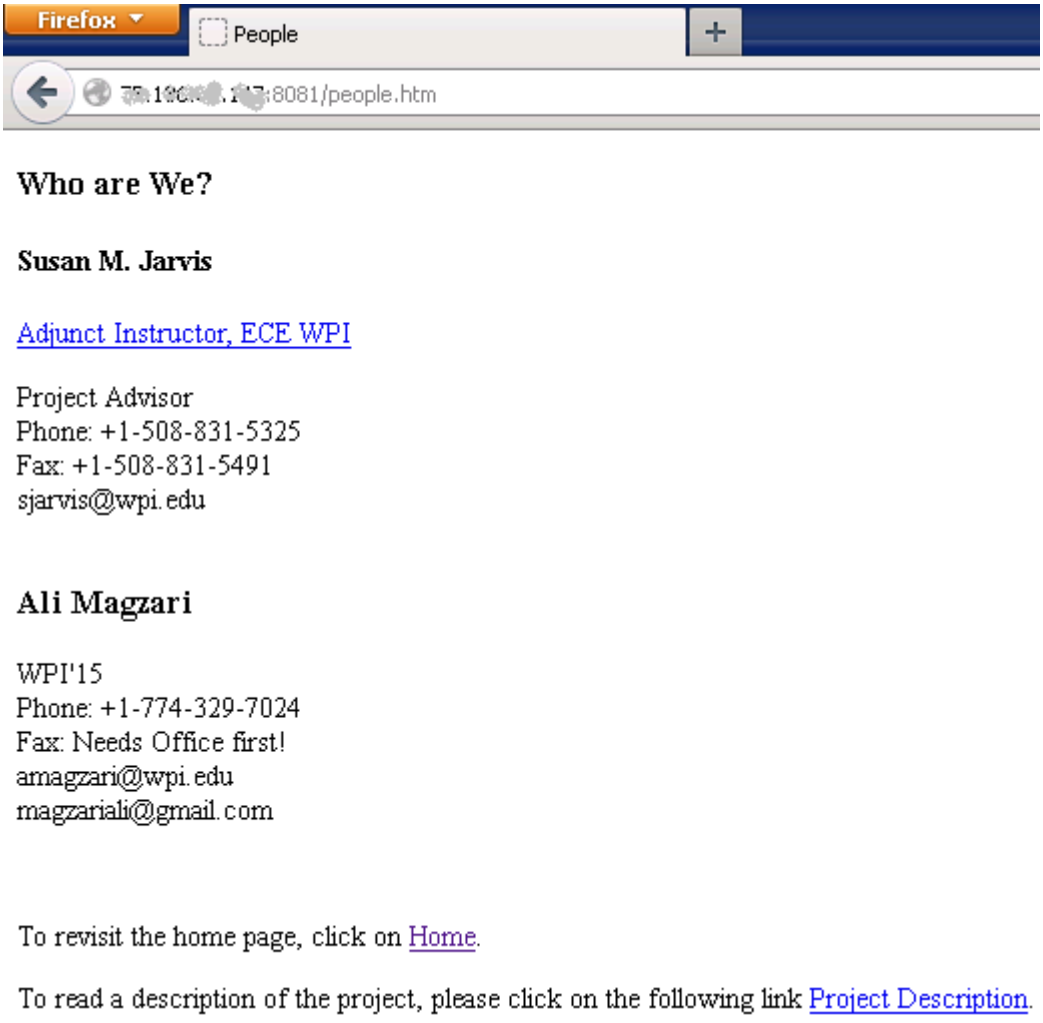


Figure 32: People Page

From this page, the WPI Professor page could be viewed for further detail.

4.5 Shading Effects and Measurements

In this experiment, the solar modules are individually fully shaded and the voltages are recorded. Using Figure 16, the following table shows the shaded cells and the corresponding module voltages.

Table 6: Shaded Panel and Voltage Consequences

Shaded Panel	VPanel0 (V)	VPanel1 (V)	VPanel2 (V)	VPanel3 (V)	VPanel4 (V)	VPanel5 (V)
None	3.98	4.85	5.14	5.01	4.64	4.35
0	0.00	4.93	5.64	3.82	3.92	3.76
1	4.72	-0.40	5.70	4.37	3.60	3.76
2	4.64	5.41	-0.40	4.29	3.90	3.50
3	2.70	3.55	4.16	0.00	5.09	5.11
4	2.92	3.66	4.19	5.94	-0.48	5.14
5	2.97	3.68	4.24	5.83	5.51	-0.61

Note that when fully shaded, Panel0 and Panel3 produce 0.00V. Since the negative terminals of these panels are connected to ground, once they are shaded and the output voltage is set to null, the voltage drop is logically 0.00V. The rest of the panels, however, show a negative voltage drop across them when fully shaded. The main reason of that occurring is that their negative terminal is not connected to ground but to the positive terminal of an unshaded module. When the panel is shaded, the current goes through the bypass diode that is known to have a certain voltage drop. In this case the observed voltage drops are: 0.40V; 0.48V; 0.61V.

5.0 Conclusions and Future Work

The greater aspect of this project was the successful combination of several areas of electrical engineering to produce an adequate proof of concept of an online photovoltaic monitoring system. Microelectronics and semiconductor science were necessary to predict the behavior of the photovoltaic system, hence the use of techniques such as bypass/blocking diodes and voltage divers to insure the operational safety of the prototype. Knowledge of embedded systems and computer science was very useful when it came to programming the Arduino board whose language is based on C and C++. Html and Javascript were also languages used in formatting and displaying data on the webserver.

One of the ways this project could be improved is to dedicate deeper attention to the different building blocks of the project. This project was completed by one student that used a system engineering approach to build the necessary building blocks. Due to time constraints, some areas have received less attention than they deserve. For example, the photovoltaic modeling and simulation explained in Section 2.2 is of great importance, but sadly could not be used in the project.

The SD card on to the Ethernet shield stores the data and will eventually saturate one day. A better option could be sending the collecting data to a Mysql database. Another improvement would be the creation of a proper PCB (Printed Circuit Board) and packaging to protect the underlying circuitry from weather conditions such as rain and snow that could cause the malfunction of the prototype.

This project could serve as a starting point for other students in the future. They could choose to either improve the whole project following a system engineering point of view or focus on a particular aspect of the project (i.e. effects of shading).

Works Cited

Alonso, M. (2005). *Caracterización y modelado de asociaciones de dispositivos fotovoltaicos.* Madrid: Centro de Investigaciones energéticas, Medioambientales y Tecnológicas (Ciemat).

Arduino. (2015). *Arduino Ethernet Shield*. Retrieved from Qrduino: http://www.arduino.cc/en/Main/ArduinoEthernetShield

Arduino. (n.d.). *Arduino Uno*. Retrieved from ARDUINO: http://arduino.cc/en/Main/arduinoBoardUno

Bouzidi, K., Cheggar, M., & Bouhemadou, A. (2007). Solar cells parameters evaluation considering the series and shunt resistance. *91*(18).

Carrero, C., Amador, J., & Arnaltes, S. (2007). A single procedure for helping PV designers to select silicon PV module and evaluate the loss resistances. *32*(15).

CIVICSOLAR. (n.d.). *SolarEdge Zigbee-Ethernet Gateway* . Retrieved 04 26, 2013, from CIVICSOLAR: http://www.civicsolar.com/product/solaredge-zigbee-interface-gateway?utm_source=google_shopping&utm_medium=google_shopping&utm_campaign=google_shopping&gclid=CJWR2Pud6LYCFY87MgodUzUAkQ

CIVICSOLAR. (n.d.). *wireless-communication-products.* Retrieved 04 26, 2013, from CIVICSOLAR: http://www.civicsolar.com/sites/default/files/documents/wwwsolaredgeusfilespdfsproductsinverterszigbee-wireless-communication-products-65748.pdf

De Soto, W., Klein, S. A., & Beckman, W. A. (2005). Improvement and validation of a model for photovoltaic array performance. *Solar Energy 80*(2006).

Eckstein, J. H. (1990). *Detailed Modelling of Photovoltaic System Components* . University of Wisconsin - Madison.

ECODIRECT. (n.d.). *BenQ AUO Solar 19.M2M01.002*. Retrieved 04 26, 2013, from ECODIRECT, Clean Energy Solutions: http://www.ecodirect.com/ProductDetails.asp?ProductCode=BenQ-AUO-19-M2M01-002&gclid=CLvd2tue6LYCFQdgMgodBGIAOg

El Tayyan, A. A. (2012). PV system behavior based on datasheet. *9*.

Everett. (n.d.). *Arduino: Super Graphing Data Logger*. Retrieved from Everett's Pjocts: http://everettsprojects.com/2012/12/31/arduino-super-graphing-data-logger/

Google. (n.d.). *Google Shopping*. Retrieved 04 26, 2013, from Google: http://www.google.com/shopping

Hambley, A. R. (2011). *Elecrical Engineering Principles anApplications.* Upper Saddle River: Pearson Education.

Kaiser, T. J. (2010, 05 24). *Lecture 08: Solar Cell Characterization.* Retrieved 04 12, 2012, from EE 580 - Solar Cell Basics for Teachers: http://www.coe.montana.edu/ee/tjkaiser/ee580/Notes/MSUEE580-08Characterization.pdf

KYOCERA. (n.d.). *KCT85T High efficiency multicrystal photovoltaic module.* Retrieved 01 15, 2013, from Backwoods Solar Electric Systems: http://www.backwoodssolar.com/catalog/Spec_Sheets/KC85T.pdf

LLC, F. U. (2013). *Fronius Datalogger easy/pro*. Retrieved 04 26, 2013, from FRONIUS INTERNATIONAL: http://www.fronius.com/cps/rde/xchg/SID-16D84AA2-C939179D/fronius_international/hs.xsl/83_16098_ENG_HTML.htm

PARALLAX. (n.d.). *http://www.parallax.com/sites/default/files/downloads/750-00031-9V-1W-Solar-Panel-Datasheet.pdf.* Retrieved from http://www.parallax.com.

Rekioua, D., & Matagne, E. (2012). *Optimization of Photovoltaic Power Systems: Modelization, Simulation and Control.* Springer.

Rodrigues, E., Melicio, R., Mendes, V., & Catalao, J. (2011). Simulation of a Solar Cell considering Single-Diode Equivalent Circuit Model. *ICREPQ.* Bilbao.

Sedra, A. S., & Smith, K. C. (2010). *Microelectronic Circuits.* New York Oxford: Oxford University Press.

SunEarthTools. (2015, april 08). Retrieved from http://www.sunearthtools.com/dp/tools/pos_sun.php?lang=en

Technologies, A. (2012). *IV and CV Characterizations of Solar/Photovoltaic Cells Using the B1500A.* Retrieved 12 04, 2012, from http://cp.literature.agilent.com/litweb/pdf/5990-4428EN.pdf

timeanddate. (2015, april 12). *Worcester, U.S.A. — Sunrise, sunset and daylength, April 2015*. Retrieved from timeanddate: http://www.timeanddate.com/sun/usa/worcester

Villalva, M. G., Gazoli, J. R., & Ruppert, E. F. (2009). Comprehensive approach to modeling and simulation of photovoltaic arrays. *IEEE Transactions on Power Electronics, 25*(5), 1198--1208.

Villalva, M. G., Gazoli, R. J., & Filho, E. R. (2009). MODELING AND CIRCUIT-BASED SIMULATION OF PHOTOVOLTAIC ARRAYS. Sao Paulo: IEEE.

Walker, G. (2001). Evaluating MPPT Converter Topologies Using A MATLAV PV Model. *Journal of Electrical & Electronic Engineering*, 49-56.

Appendices

A1. PARALLAX 750-00031 Datasheet

Specification for 1W PV Module

(Test condition: 1000W/m², AM1.5, 25℃)

Type / Silicon / Parameters		1W
		Mono-Crystalline Silicon
Maximum Power	Watt	1
Production Tolerance		±10%
Maximum Power voltage	V	9V
Maximum Power current	A	0.11
Open circuit voltage	V	10.4
Short circuit current	A	0.12
Cells thickness		0.18mm±20μm
size of panel (wide and high)		135*135*3mm
Number of cells		18
Weight per piece (Kgs)		0.23

Figure 33: PARALLAX 750-00031 Specifications (PARALLAX)

A2. MCP6001/2/4 Datasheet

MCP6001/2/4

1 MHz, Low-Power Op Amp

Features

- Available in SC-70-5 and SOT-23-5 packages
- Gain Bandwidth Product: 1 MHz (typ.)
- Rail-to-Rail Input/Output
- Supply Voltage: 1.8V to 5.5V
- Supply Current: I_Q = 100 µA (typ.)
- Phase Margin: 90° (typ.)
- Temperature Range:
  - Industrial: -40°C to +85°C
  - Extended: -40°C to +125°C
- Available in Single, Dual and Quad Packages

Applications

- Automotive
- Portable Equipment
- Photodiode Amplifier
- Analog Filters
- Notebooks and PDAs
- Battery-Powered Systems

Description

The Microchip Technology Inc. MCP6001/2/4 family of operational amplifiers (op amps) is specifically designed for general-purpose applications. This family has a 1 MHz Gain Bandwidth Product (GBWP) and 90° phase margin (typ.). It also maintains 45° phase margin (typ.) with a 500 pF capacitive load. This family operates from a single supply voltage as low as 1.8V, while drawing 100 µA (typ.) quiescent current. Additionally, the MCP6001/2/4 supports rail-to-rail input and output swing, with a common mode input voltage range of V_{DD} + 300 mV to V_{SS} – 300 mV. This family of op amps is designed with Microchip's advanced CMOS process.

The MCP6001/2/4 family is available in the industrial and extended temperature ranges, with a power supply range of 1.8V to 5.5V.

The rest of datasheet could be found at: http://www.mouser.com/ds/2/268/21733e-41017.pdf

A3. 1N4148 Datasheet

VISHAY
www.vishay.com

1N4148

Vishay Semiconductors

Small Signal Fast Switching Diodes

FEATURES

- Silicon epitaxial planar diode
- Electrically equivalent diodes: 1N4148 - 1N914
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

Pb e3 RoHS COMPLIANT HALOGEN FREE

APPLICATIONS

- Extreme fast switches

MECHANICAL DATA

Case: DO-35
Weight: approx. 105 mg
Cathode band color: black
Packaging codes/options:
TR/10K per 13" reel (52 mm tape), 50K/box
TAP/10K per ammopack (52 mm tape), 50K/box

PARTS TABLE

PART	ORDERING CODE	TYPE MARKING	INTERNAL CONSTRUCTION	REMARKS
1N4148	1N4148-TAP or 1N4148TR	V4148	Single diode	Tape and reel/ammopack

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Repetitive peak reverse voltage		V_{RRM}	100	V
Reverse voltage		V_R	75	V
Peak forward surge current	t_p = 1 µs	I_{FSM}	2	A
Repetitive peak forward current		I_{FRM}	500	mA
Forward continuous current		I_F	300	mA

The rest of datasheet could be found at: http://www.vishay.com/docs/81857/1n4148.pdf

A4. Arduino Sketch: OPVMS.ino

```
/*************************************************************
 * ***                        OPVMS                        ***
 * ************************************************************
 * Author: Ali Magzari
 * Date: April 2015
 * This code is based on the Super Graphing Data Logger code
 * written by Everett Robinson in December 2012
*/

// Libraries
#include <SD.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <SPI.h>
#include <string.h>
#include <Time.h>
#include <EEPROM.h>
#include <EEPROMAnything.h>
#include <avr/pgmspace.h>

// Ethernet setup
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x7B, 0xB6 };
byte ip[] = { 192,168,1, 177 };
EthernetServer server(8081);

// NTP setup
unsigned int localPort = 8888;
IPAddress timeServer(132, 163, 4, 101);
//NTP time stamp is in the first 48 bytes of the message
const int NTP_PACKET_SIZE= 48;
byte packetBuffer[ NTP_PACKET_SIZE];
EthernetUDP Udp;

// Data logger and Timer Control
const int analogPin = 0;
//The time the last measurement occurred.
unsigned long lastIntervalTime = 0;
//1 minute interval between measurements
#define MEASURE_INTERVAL 60000
//The time at which we should create a new week's file
unsigned long newFileTime;
#define FILE_INTERVAL 300
char charBuf[5];   // To store float data into a string

//A structure that stores file config variables from EEPROM
typedef struct{
   //Keeps track of when a newfile should be made.
    unsigned long newFileTime;
    //The path and filename of the current week's file
    char workingFilename[19];
} configuration;

configuration config;
//Actually make our config struct
```

```
// Strings stored in flash mem for the Html Header (saves ram)
prog_char HeaderOK_0[] PROGMEM = "HTTP/1.1 200 OK";
prog_char HeaderOK_1[] PROGMEM = "Content-Type: text/html";
prog_char HeaderOK_2[] PROGMEM = "";

// A table of pointers to the flash memory strings for the header
PROGMEM const char *HeaderOK_table[] = {
  HeaderOK_0,
  HeaderOK_1,
  HeaderOK_2
};

// A function for reasy printing of the headers
void HtmlHeaderOK(EthernetClient client) {

    char buffer[30]; //A character array to hold the strings from the flash
mem

    for (int i = 0; i < 3; i++) {
      strcpy_P(buffer, (char*)pgm_read_word(&(HeaderOK_table[i])));
      client.println( buffer );
    }
}

// Strings stored in flash mem for the Html 404 Header
prog_char Header404_0[] PROGMEM = "HTTP/1.1 404 Not Found";
prog_char Header404_1[] PROGMEM = "Content-Type: text/html";
prog_char Header404_2[] PROGMEM = "";
prog_char Header404_3[] PROGMEM = "<h2>File Not Found!</h2>";

// A table of pointers to the flash memory strings for the header
PROGMEM const char *Header404_table[] = {
  Header404_0,
  Header404_1,
  Header404_2,
  Header404_3
};

// Easy peasy 404 header function
void HtmlHeader404(EthernetClient client) {

    char buffer[30]; //A character array to hold the strings from the flash
mem

    for (int i = 0; i < 4; i++) {
      strcpy_P(buffer, (char*)pgm_read_word(&(Header404_table[i])));
      client.println( buffer );
    }
}

void setup() {
  Serial.begin(9600);
  // set the SS pin as an output (necessary!)
  pinMode(10, OUTPUT);
```

```
  digitalWrite(10, HIGH);  // but turn off the W5100 chip!

  // see if the card is present and can be initialized:
  if (!SD.begin(4)) {
    //Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");

  // The SD card is working, start the server and ethernet related stuff!
  Ethernet.begin(mac, ip);
  server.begin();
  Udp.begin(localPort);
  // make sure our config struct is syncd with EEPROM
  EEPROM_readAnything(0,config);
}

// A function that takes care of the listing of files for the
// main page one sees when they first connect to the arduino.
// it only lists the files in the /data/ folder. Make sure this
// exists on your SD card.
void ListFilesVoltage(EthernetClient client) {

  File workingDir = SD.open("/data");

  client.println("<ul>");

    while(true) {
      File entry =  workingDir.openNextFile();
       if (! entry) {
         break;
       }
       client.print("<li><a href=\"/HC.htm?file=");
       client.print(entry.name());
       client.print("\">");
       client.print(entry.name());
       client.println("</a></li>");
       entry.close();
    }
  client.println("</ul>");
  workingDir.close();
}

// A function to get the Ntp Time. This is used to make sure that the data
// points recorded by the arduino are referenced to some meaningful time
// which in our case is UTC represented as unix time (choosen because it
// works simply with highcharts without too much unecessary computation).
unsigned long getTime(){
// send an NTP packet to a time server
sendNTPpacket(timeServer);

  // wait to see if a reply is available
  delay(1000);
  if ( Udp.parsePacket() ) {
    // We've received a packet, read the data from it
```

```
    Udp.read(packetBuffer,NTP_PACKET_SIZE);
    // the timestamp starts at byte 40 of the received packet and is four
    // bytes or two words.
    // First, extract the two words:

  unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    // combine the four bytes (two words) into a long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // subtract seventy years:
    unsigned long epoch = secsSince1900 - seventyYears - 14400; //-14
    // return Unix time:
    return epoch;
  }
}

// send an NTP request to the time server at the given address,
// necessary for getTime().
unsigned long sendNTPpacket(IPAddress& address){

  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123); //NTP requests are to port 123
  Udp.write(packetBuffer,NTP_PACKET_SIZE);
  Udp.endPacket();
}


// How big our line buffer should be for sending the files over the ethernet.
// 75 has worked fine for me so far.
#define BUFSIZ 75

void loop(){
  if ((millis() % lastIntervalTime) >= MEASURE_INTERVAL){ //Is it time for a
new measurement?

    char dataString[68] = "";
    int count = 0;
    unsigned long rawTime;
```

```
    rawTime = getTime();

    while((rawTime == 39) && (count < 12)){
      delay(5000);
      rawTime = getTime();
      count += 1;
    }
    if (rawTime != 39){

   //Decide if it's time to make a new file or not. Files are broken
      //up like this to keep loading times for each chart bearable.
      //Lots of string stuff happens to make a new filename if necessary.
      if (rawTime >= config.newFileTime){
        int dayInt = day(rawTime);
        int monthInt = month(rawTime);
        int yearInt = year(rawTime);
        char newFilename[18] = "";
        char dayStr[3];
        char monthStr[3];
        char yearStr[5];
        char subYear[3];
        strcat(newFilename,"data/");
        itoa(dayInt,dayStr,10);
        if (dayInt < 10){
          strcat(newFilename,"0");
        }
        strcat(newFilename,dayStr);
        strcat(newFilename,"-");
        itoa(monthInt,monthStr,10);
        if (monthInt < 10){
          strcat(newFilename,"0");
        }
        strcat(newFilename,monthStr);
        strcat(newFilename,"-");
        itoa(yearInt,yearStr,10);
        //we only want the last two digits of the year
        memcpy( subYear, &yearStr[2], 3 );
        strcat(newFilename,subYear);
        strcat(newFilename,".csv");

        //make sure we update our config variables:
        config.newFileTime += FILE_INTERVAL;
        strcpy(config.workingFilename,newFilename);
        //Write the changes to EEPROM. Bad things may happen if power is lost
midway through,
        //but it's a small risk we take. Manual fix with EEPROM_config sketch
can correct it.
        EEPROM_writeAnything(0, config);
      }

      //get the values and setup the string we want to write to the file
      int sensor0 = analogRead(0);
      int sensor1 = analogRead(1);
      int sensor2 = analogRead(2);
      int sensor3 = analogRead(3);
      int sensor4 = analogRead(4);
```

```
   int sensor5 = analogRead(5);
   int sensor6 = analogRead(6);
   int sensor7 = analogRead(7);

   char timeStr[12];
   char sensorStr0[6];
   char sensorStr1[6];
   char sensorStr2[6];
   char sensorStr3[6];
   char sensorStr4[6];
   char sensorStr5[6];
   char sensorStr6[6];
   char sensorStr7[6];
   //char sensorStr8[6];
   float RdvCf = 542.2/100;
   float BtV = 5.0/1023.0;
   float VCf = RdvCf*BtV;

   ultoa(rawTime,timeStr,10);
   strcat(dataString,timeStr);
   strcat(dataString,",");

   // PV Voltages in branch 1
   float V1 = analogRead(A0)*VCf;
   dtostrf(V1, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");
   float V2 = (analogRead(A1)-analogRead(A0))*VCf;
   dtostrf(V2, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");
   float V3 = (analogRead(A2)-analogRead(A1))*VCf;
   dtostrf(V3, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");

   // PV Voltages in branch 2
   float V4 = analogRead(A3)*VCf;
   dtostrf(V4, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");
   float V5 = (analogRead(A4)-analogRead(A3))*VCf;
   dtostrf(V5, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");
   float V6 = (analogRead(A5)-analogRead(A4))*VCf;
   dtostrf(V6, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");
   float VT = (V1+V2+V3+V4+V5+V6)/2;
   dtostrf(VT, 4, 2, charBuf);
   strcat(dataString,charBuf);
   strcat(dataString,",");


// Pv Currents in branch 1 and 2
```

```
      float I = VT/(1*100);
      dtostrf(I, 6, 4, charBuf);
      strcat(dataString,charBuf);

      //open the file we'll be writing to.
      File dataFile = SD.open(config.workingFilename, FILE_WRITE);

      // if the file is available, write to it:
      if (dataFile) {
        dataFile.println(dataString);
        dataFile.close();
        // print to the serial port too:
        //Serial.println(Str4);
        Serial.println(dataString);
      }
      // if the file isn't open, pop up an error:
      else {
        //Serial.println("Error opening datafile for writing");
      }
    }
    else{
      Serial.println("Couldn't resolve a time from the Ntp Server.");
    }
    //Update the time of the last measurment to the current timer value
    lastIntervalTime = millis();
  }
  //No measurements to be made, make sure the webserver is available for
connections.
  else{
    char clientline[BUFSIZ];
    int index = 0;

    EthernetClient client = server.available();
    if (client) {
      // an http request ends with a blank line
      boolean current_line_is_blank = true;

      // reset the input buffer
      index = 0;

      while (client.connected()) {
        if (client.available()) {
          char c = client.read();

     // If it isn't a new line, add the character to the buffer
          if (c != '\n' && c != '\r') {
            clientline[index] = c;
            index++;
            // are we too big for the buffer? start tossing out data
            if (index >= BUFSIZ)
              index = BUFSIZ -1;

            // continue to read more data!
            continue;
          }
```

```
          // got a \n or \r new line, which means the string is done
          clientline[index] = 0;

          // Print it out for debugging
          Serial.println(clientline);

     // Look for substring such as a request to get the root file
          if (strstr(clientline, "GET / ") != 0) {
            // send a standard http response header
            HtmlHeaderOK(client);
            client.println("<h1>Welcome to OPVMS-2015</h1>");
            client.print("To read a description of the project, please click
on the following link: <a href=\"/home.htm?file=description\">Project
Description</a>");
            client.println("<h2>Photovoltaic Present Data</h2>");
            client.println("Below are listed the individual voltage and
current values of the solar panels of the PV system:<br><br>");

             int sensor0 = analogRead(0);
             int sensor1 = analogRead(1);
             int sensor2 = analogRead(2);
             int sensor3 = analogRead(3);
             int sensor4 = analogRead(4);
             int sensor5 = analogRead(5);
             int sensor6 = analogRead(6);
             int sensor7 = analogRead(7);

             float RdvCf = 542.2/100;
             float BtV = 5.0/1023.0;
             float VCf = RdvCf*BtV;

             float V1 = analogRead(A0)*VCf;
             float V2 = (analogRead(A1)-analogRead(A0))*VCf;
             float V3 = (analogRead(A2)-analogRead(A1))*VCf;
             float V4 = analogRead(A3)*VCf;
             float V5 = (analogRead(A4)-analogRead(A3))*VCf;
             float V6 = (analogRead(A5)-analogRead(A4))*VCf;
             float VT = (V1+V2+V3+V4+V5+V6)/2;
             float I = VT/100;

            client.print("PV Voltages<br><br>");
            client.print("Panel 1: "); client.print(V1);
            client.print("V<br>");
            client.print("Panel 2: "); client.print(V2);
            client.print("V<br>");
            client.print("Panel 3: "); client.print(V3);
            client.print("V<br>");
            client.print("Panel 4: "); client.print(V4);
            client.print("V<br>");
            client.print("Panel 5: "); client.print(V5);
            client.print("V<br>");
            client.print("Panel 6: "); client.print(V6);
            client.print("V<br>");
            client.print("Total  : "); client.print(VT);
            client.print("V<br>");
            client.print("PV Current: "); client.print(I);
```

```
            client.print("V<br>");

            client.println("<h2>Photovoltaic Data History</h2>");
            client.println("Below are CSV files containing the PV collected
data.<br>");
            client.println("Click on the following links to view voltage and
currnet charts of the corresponding day:");
            ListFilesVoltage(client);
            client.print("To see the people involved in this project, click
on <a href=\"/people.htm?file=people\">People</a>");

          }
          else if (strstr(clientline, "GET /") != 0) {
            // this time no space after the /, so a sub-file!
            char *filename;

            filename = strtok(clientline + 5, "?"); // look after the "GET /"
(5 chars) but before
            // the "?" if a data file has been specified. A little trick,
look for the " HTTP/1.1"
            // string and turn the first character of the substring into a 0
to clear it out.
            (strstr(clientline, " HTTP"))[0] = 0;

            // print the file we want
            Serial.println(filename);
            File file = SD.open(filename,FILE_READ);
            if (!file) {
              HtmlHeader404(client);
              break;
            }

            Serial.println("Opened!");

            HtmlHeaderOK(client);

            int16_t c;
            while ((c = file.read()) > 0) {
                // uncomment the serial to debug (slow!)
                //Serial.print((char)c);
                client.print((char)c);
            }
            file.close();
          }
          else {
            // everything else is a 404
            HtmlHeader404(client);
          }
          break;
        }
      }
      // give the web browser time to receive the data
      delay(1);
      client.stop();
    }
  }}
```

A5. HC.htm code

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Super Graphing Data Logger!</title>

        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></scrip
t>
        <script type="text/javascript">
function getDataFilename(str){
    point = str.lastIndexOf("file=")+4;

    tempString = str.substring(point+1,str.length)
    if (tempString.indexOf("&") == -1){
    return(tempString);
    }
    else{
        return tempString.substring(0,tempString.indexOf("&"));
    }

}

query  = window.location.search;

var dataFilePath = "/data/"+getDataFilename(query);

$(function () {
    var chart1;
      var chart2;

      $(document).ready(function() {

        // define the options
        var options = {

            chart: {
                renderTo: 'chart1',
                zoomType: 'x',
                spacingRight: 20
            },

            title: {
                text: 'Voltage levels of the PV Panels (1-6)'
            },

            subtitle: {
                text: 'Click and drag in the plot area to zoom in'
            },

            xAxis: {
                        title: {
                    text: 'Time of Day HH:MM'
                },
                type: 'datetime',
```

```
            maxZoom: 2 * 3600000
        },

        yAxis: {
            title: {
                text: 'Voltage(V)'
            },
            min: 0,
        },

        legend: {
                    layout: 'vertical',
                    align: 'right',
                    verticalAlign: 'middle',
                    borderWidth: 0
        },

        tooltip: {
            shared: true,
            crosshairs: true
        },

        plotOptions: {
            series: {
                cursor: 'pointer',
                lineWidth: 1.0,
                point: {
                    events: {
                        click: function() {
                            hs.htmlExpand(null, {
                                pageOrigin: {
                                    x: this.pageX,
                                    y: this.pageY
                                },
                                headingText: this.series.name,
                                maincontentText:
Highcharts.dateFormat('%H:%M - %b %e, %Y', this.x) +':<br/> '+
                                                        this.y,
                                width: 200
                            });
                        }
                    }
                },
            }
        },

        series: [{
            name: 'Panel volatge 1',
            marker: {
                radius: 2
            }
                    }
                    ,
                    {name: 'Panel volatge 2',
            marker: {
                radius: 2
                    }
```

```
     }
           ,
                {name: 'Panel volatge 3',
        marker: {
            radius: 2
                }
     }
           ,
           {name: 'Panel volatge 4',
        marker: {
            radius: 2
                }
     }
           ,
           {name: 'Panel volatge 5',
        marker: {
            radius: 2
                }
     }
           ,
           {name: 'Panel volatge 6',
        marker: {
            radius: 2
                }
     },
           {name: 'Total voltage',
        marker: {
            radius: 2
                }
     }

           ]
};


// Load data asynchronously using jQuery. On success, add the data
// to the options and initiate the chart.
// http://api.jquery.com/jQuery.get/
jQuery.get(dataFilePath, null, function(csv, state, xhr) {
    var lines = [],
        date,

        // set up the two data series
        voltageLevels1 = [];
                voltageLevels2 = [];
                voltageLevels3 = [];
                voltageLevels4 = [];
                voltageLevels5 = [];
                voltageLevels6 = [];
                voltageTotal = [];
    // inconsistency
    if (typeof csv !== 'string') {
        csv = xhr.responseText;
    }

    // split the data return into lines and parse them
    csv = csv.split(/\n/g);
```

```
        jQuery.each(csv, function(i, line) {

            // all data lines start with a double quote
            line = line.split(',');
            date = parseInt(line[0], 10)*1000;

            voltageLevels1.push([
                date,
                parseFloat(line[1])
            ]);
                    voltageLevels2.push([
                date,
                parseFloat(line[2])
            ]);
                    voltageLevels3.push([
                date,
                parseFloat(line[3])
            ]);
                    voltageLevels4.push([
                date,
                parseFloat(line[4])
            ]);
                    voltageLevels5.push([
                date,
                parseFloat(line[5])
            ]);
                    voltageLevels6.push([
                date,
                parseFloat(line[6])
            ]);
                    voltageTotal.push([
                date,
                parseFloat(line[7])
            ]);

        });

        options.series[0].data = voltageLevels1;
              options.series[1].data = voltageLevels2;
              options.series[2].data = voltageLevels3;
              options.series[3].data = voltageLevels4;
              options.series[4].data = voltageLevels5;
              options.series[5].data = voltageLevels6;
              options.series[6].data = voltageTotal;

        chart1 = new Highcharts.Chart(options);
    });
});

  $(document).ready(function() {

    // define the options
    var options = {

        chart: {
            renderTo: 'chart2',
            zoomType: 'x',
```

```
        spacingRight: 20
    },

    title: {
        text: 'Current level of the PV System'
    },

    subtitle: {
        text: 'Click and drag in the plot area to zoom in'
    },

    xAxis: {
                    title: {
            text: 'Time of Day HH:MM'
        },
        type: 'datetime',
        maxZoom: 2 * 3600000
    },

    yAxis: {
        title: {
            text: 'Current(mA))'
        },
        min: 0,
        startOnTick: false,
        showFirstLabel: false
    },

    legend: {
        //enabled: false
                layout: 'vertical',
                align: 'right',
                verticalAlign: 'middle',
                borderWidth: 0
    },

    tooltip: {
          shared: true,
        crosshairs: true
    },

    plotOptions: {
        series: {
            cursor: 'pointer',
            lineWidth: 1.0,
            point: {
                events: {
                    click: function() {
                        hs.htmlExpand(null, {
                            pageOrigin: {
                                x: this.pageX,
                                y: this.pageY
                            },
                            headingText: this.series.name,
                            maincontentText:
Highcharts.dateFormat('%H:%M - %b %e, %Y', this.x) +':<br/> '+
                                Highcharts.numberFormat(this.y, 4),
```

```
                            width: 200
                        });
                    }
                }
            },
        }
    },

    series: [{
        name: 'Current Level',
        marker: {
            radius: 2
        }
                }


           ]
};


// Load data asynchronously using jQuery. On success, add the data
// to the options and initiate the chart.
// http://api.jquery.com/jQuery.get/
jQuery.get(dataFilePath, null, function(csv, state, xhr) {
    var lines = [],
        date,

        // set up the two data series
        currentLevel = [];


    // inconsistency
    if (typeof csv !== 'string') {
        csv = xhr.responseText;
    }

    // split the data return into lines and parse them
    csv = csv.split(/\n/g);
    jQuery.each(csv, function(i, line) {

        // all data lines start with a double quote
        line = line.split(',');
        date = parseInt(line[0], 10)*1000;

        currentLevel.push([
            date,
            parseFloat(line[8])
        ]);

    });

    options.series[0].data = currentLevel;


    chart2 = new Highcharts.Chart(options);
});
});
```

```
});
        </script>
    </head>
    <body>
        <p style="text-align:center;">Please allow 30 seconds for the charts
to load </p>
        <hr/>
<script
src="http://cdnjs.cloudflare.com/ajax/libs/highcharts/2.3.5/highcharts.js"></
script>

<!-- Additional files for the Highslide popup effect -->
<script type="text/javascript"
src="http://www.highcharts.com/highslide/highslide-full.min.js"></script>
<script type="text/javascript"
src="http://www.highcharts.com/highslide/highslide.config.js" charset="utf-
8"></script>
<link rel="stylesheet" type="text/css"
href="http://www.highcharts.com/highslide/highslide.css" />

<!--<div id="container" style="min-width: 400px; height: 400px; margin: 0
auto"></div>-->

<div id="chart1" style="min-width: 400px; height: 300px; margin: 0
auto"></div> <!-- Container for Chart A -->
<div class="spacer" style="height: 30px"></div>
<div id="chart2" style="min-width: 400px; height: 300px; margin: 0
auto""></div> <!-- Container for Chart B -->

    </body>
</html>
```

A6. Home.htm code

```
<!DOCTYPE html>
<html>
<head>
  <title>Project Description</title>
  <style type="text/css">
  h1 {
  text-align: center;
  font-family: Times New Roman, Times, serif;
  font-size: 20pt;
 }
  p {
  font-family: Times New Roman, Times;
  font-size: 12pt;
  font-weight: normal;
  }
  </style>
</head>
<body>
  <h1>OPVMS-2015</br>Online Photovoltaic Monitoring System</h1>
  <h2>Introduction<h2>
  <p>OPVMS-2015 is an Arduino-based web server that monitors the voltage and
current values of individual solar panels in a photovoltaic system.</br>
     Solar energy is a type of renewable energy widely used in industrial and
residential applications. Although solar energy is considered to be</br>
       a good alternative in terms of power cost reduction and energy
sustainability, it requires regular maintenance to insure proper
operation.</br>
       Seemingly insignificant factors such as dust and shading could lower
the output power of one panel, thus reducing the power output of the</br>
       entire photovoltaic system. Periodic inspections of individual
components wiring is also recommended to reduce the chance of a system
failure.</br>
       This project offers a solution to this problem as it consists of
displaying the PV data over the WAN to be viewed at the user's
convenience.</p>
  <h2>Prototype Design</h2>
  <h3>Hardware: Physical and Electrical Layout</h3>
  <p>The PV prototype consists of six solar panels forming two parallel
branches of three panels connected in series across a test resistance load of
100kOhm . </p>
  <p>The solar panel used in this model is a PARALLAX 750-00031 outputting a
maximum of 9 DCV at 1W with the dimensions of 135x135mm.</p>

  <h3>Data Processing and Monitoring</h3>
  <p>An Arduino Mega is used to read the values of the panel voltages (V1-V6)
and the branch currents(I1-I2). An ethernet shield sits on top of the</br>
     Mega and is connected to the local house router via an RJ45 cable. The
router is reconfigured to port-forward the data so that is available in</br>
       the WAN (Wide Area Network) rather than just in the LAN (Local Area
Network) of the house.</p>
  <p>The software part was based on <a
href="http://everettsprojects.com/2012/12/31/arduino-super-graphing-data-
logger/">this project</a>
  that provided an Arduino sketch and an html code that stores a sensor value
into the SD card of the</br>
```

```
  Ethernet shield and retrieves it to chart it using Highcharts. The code was
modified to store and retrieve 8 values, serve more pages, and display two
charts.</p>


  <p>To revisit the home page, click on <a
href="http://75.136.87.147:8081/">Home</a>.</p>
  <p>To see the people involved in this project, click on <a
href="people.htm">People</a>.</p>
</body>
</html>
```

A7. People.htm code

```
<!DOCTYPE html>
<html>
    <head>
        <title>People</title>
    </head>
    <body>
        <h3>Who are We?</h3>
      <h4>Susan M. Jarvis</h4>
      <a href="http://www.wpi.edu/academics/facultydir/smj.html">
      Adjunct Instructor, ECE WPI</a>
      <p>Project Advisor<br>
      Phone: +1-508-831-5325<br>
      Fax: +1-508-831-5491<br>
      sjarvis@wpi.edu</br></br></p>

      <h3>Ali Magzari</h3>
      <p>WPI'15<br>
      Phone: +1-774-329-7024<br>
      Fax: Needs Office first!<br>
      amagzari@wpi.edu<br>
      magzariali@gmail.com</p></br>

        <p>To revisit the home page, click on <a
href="http://75.136.87.147:8081/">Home</a>.</p>
            <p>To read a description of the project, please click on the
following link <a href="home.htm">Project Description</a>.</p>
    </body>
</html>
```