UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
INGENIERO INDUSTRIAL

PROYECTO FIN DE CARRERA

INTEGRATION OF SOLAR AND WIND POWER

AUTHOR: Juan Gabriel Ballesteros García
DIRECTOR: Paul Scott Carney

MADRID, May 2014

Index of Documents

Autorizada la entrega del proyecto del alumno:

Juan Gabriel Ballesteros García

EL DIRECTOR DEL PROYECTO

Paul Scott Carney

Fdo.: ..... Fecha: ..... / ..... / .....

V^{O} B^{O} DEL COORDINADOR DE PROYECTOS

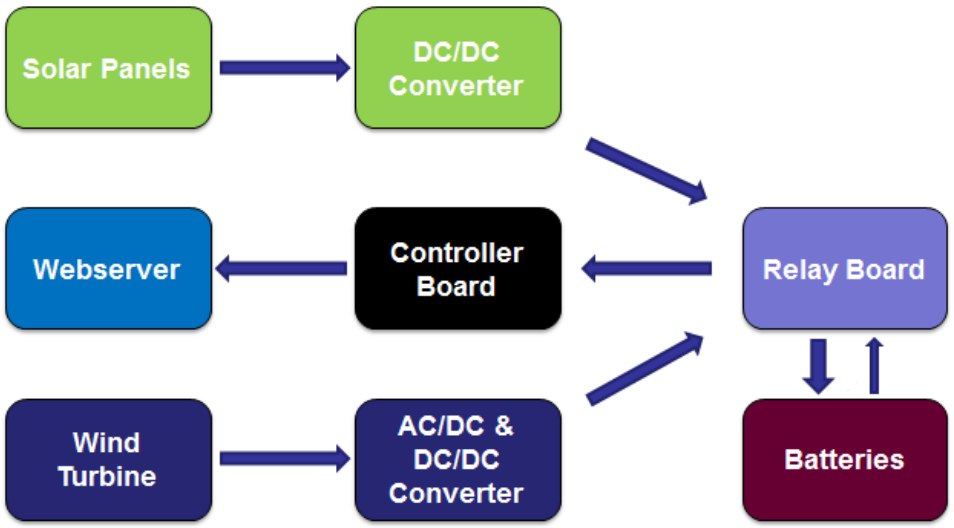
Fernando de Cuadra García

Fdo.: ..... Fecha: ..... / ..... / .....

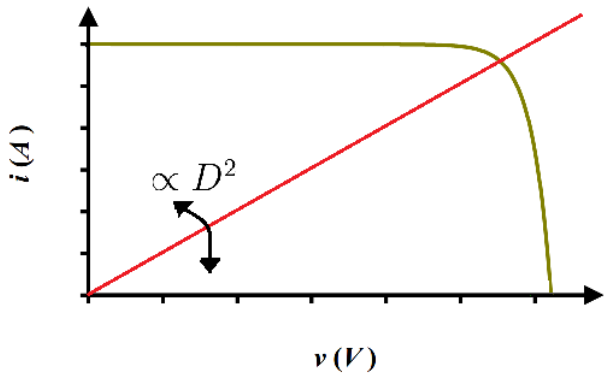
Abstract

Renewable energies are growing rapidly due to some extent to subsidies, federal tax credits, etc. There is also a growing "green"movement. An increasing number of people decide to install a rooftop solar power plant in States such as California. Net metering, on top of subsidies, is very attractive. Users get paid if they produce more energy than they consume. Small wind turbines are less popular but still becoming more present in isolated households.

This projects aims at managing a hybrid small power generating station. By combining solar and wind resources the power output is more reliable. We are provided two *Solarland SLP100-12* 100W solar panels and a *Windynation Windtura 750 PMA* 800W wind turbine. The energy produced by those devices is stored in 24V lead acid batteries. The two 12V solar panels are connected in series to form the PV array. This increases the voltage by a factor of two and maintains an output current equal to that of a single solar panel.

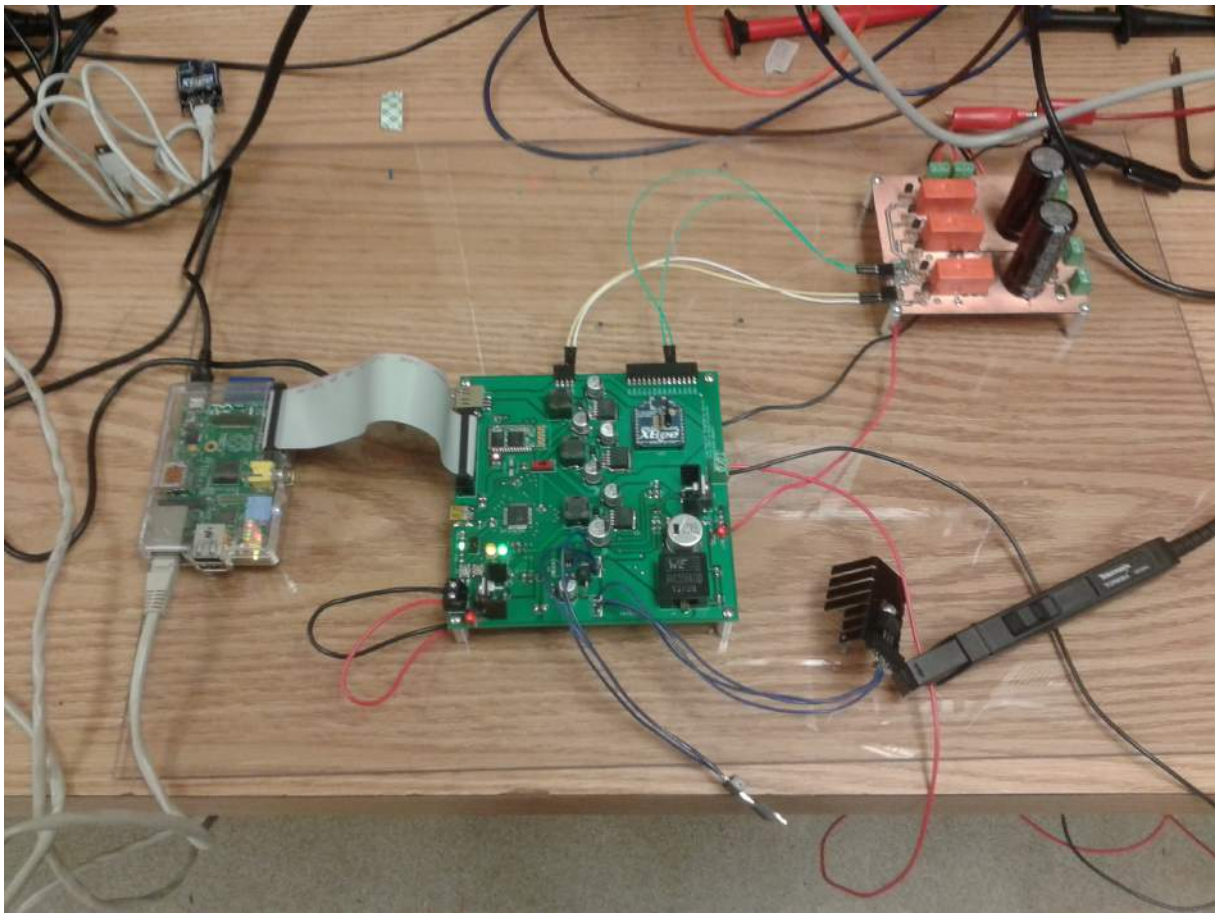


The challenge is to create the control system that manages the energy flow to the batteries. An *Atmel Atxmega128a4u* microcontroller reads the sensors which provide the input and output currents and voltages. With that information and the appropriate program it is able to adjust the duty cycle of the voltage converters (a synchronous buck converter for the solar part, and an asynchronous buck converter for the wind part) and therefore adequately charge the batteries. A maximum power point tracking algorithm is implemented in the said program to always maximize electrically the output of the PV array. This is feasible by simply adjusting the duty cycle D of the DC/DC converter.



Reverse polarity protections are implemented on the Printed Circuit Board to avoid any damage produced by a user that connects the wires the wrong way in which case red LED's would be turned on as an indicator.

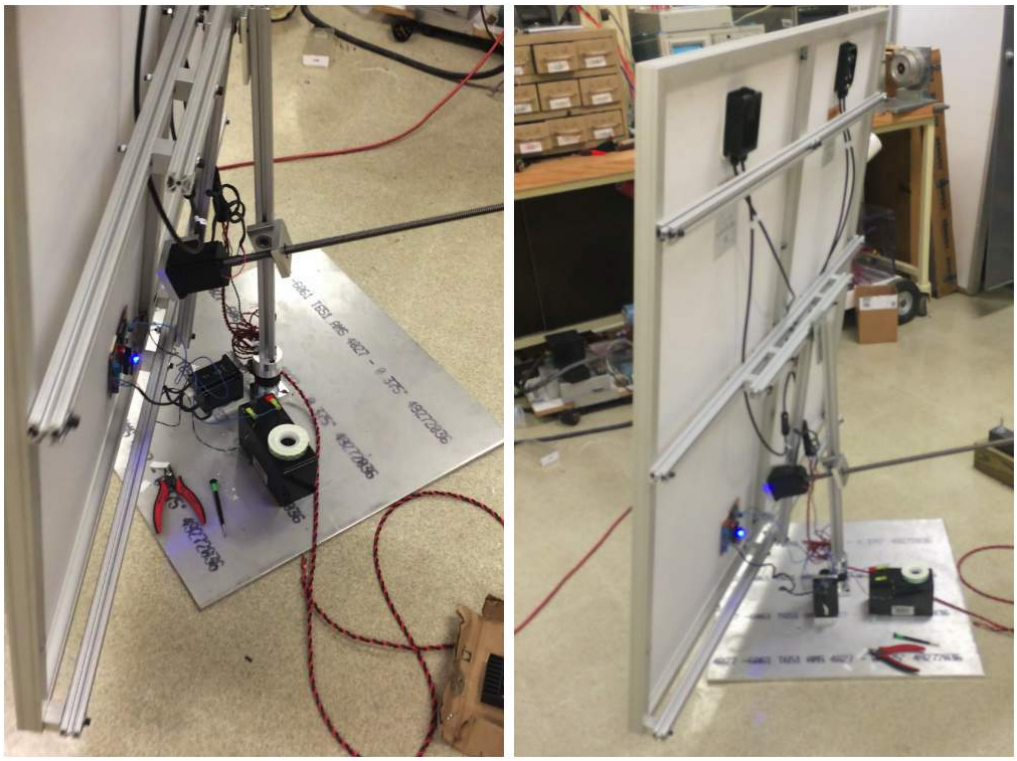
The system has its own webserver which can be connected directly to a computer or more conveniently to a router that can serve the website to any computer on the local network or to any computer on the internet (with some extra requirements such as getting a static IP address from the ISP). The webserver is implemented using a Raspberry Pi. It communicates with the microcontroller using a serial link. The Raspberry Pi sends the location of the system and the current time and in exchange receives the sensors' data.



The data is processed using a Python script that adapts the values to be human readable and stores them in a Round Robin Database in its SD card. That database is then used to generate meaningful results such as average power output, peak power output, as well as graphs with the

power output as a function of time (for both solar and wind sources), status of the batteries and more. All this information is displayed on multiple pages of a website that was developed using the Wordpress environment with an adaptive theme, which adapts the look of the website to the resolution of the screen of the device that is browsing it.

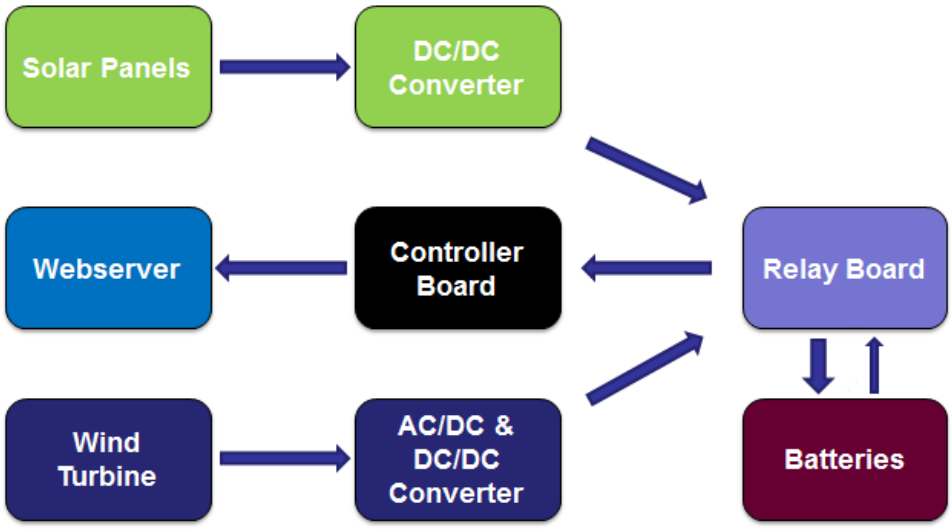
The users are able to interact with the system and not only supervise it by setting their location on one of the pages of the website. The longitude and latitude are then passed to the microcontroller which in turn sends them over a wireless link using Xbees to another part of the project which is the mechanical solar tracker. The solar tracker uses a compass and accelerometer to determine the orientation and tilt angles of the PV array. The tracker moves using stepper motors.



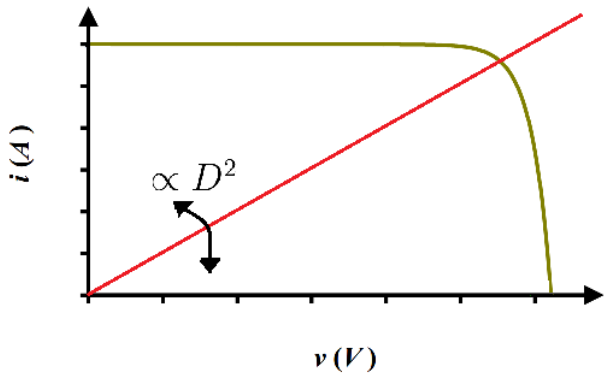
Resumen

Las energías renovables cada vez están ocupando un puesto más importante en el sector energético. Esto en parte se debe a las ayudas económicas de gobiernos. También hay una concienciación de la sociedad a adoptar un enfoque más ecológico en cuanto a consumo y gasto de energía. La medición del consumo neto de energía en algunos estados ha propiciado el crecimiento del número de instalaciones de sistemas de producción de energía renovable por parte de particulares. Los usuarios que disfrutan de dicho sistema de medición son pagados por producir la energía que inyectan en la red, es decir el exceso de producción que no consumen. Los paneles solares son los más usados en las instalaciones de particulares aunque hay un crecimiento en el uso de pequeños generadores eólicos, particularmente en sitios aislados.

Este proyecto tiene por objetivo crear el sistema de gestión de energía de una pequeña planta de producción compuesta por dos paneles solares *Solarland SLP100-12* de 100W cada uno así como un generador eólico *Windynation Windtura 750 PMA* de 800W nominales. La energía producida se almacena en unas baterías de plomo de 24V. Se ha decidido conectar los dos paneles solares en serie para así duplicar la tensión de salida y mantener la corriente baja, generando una máxima potencia de salida de 200W.

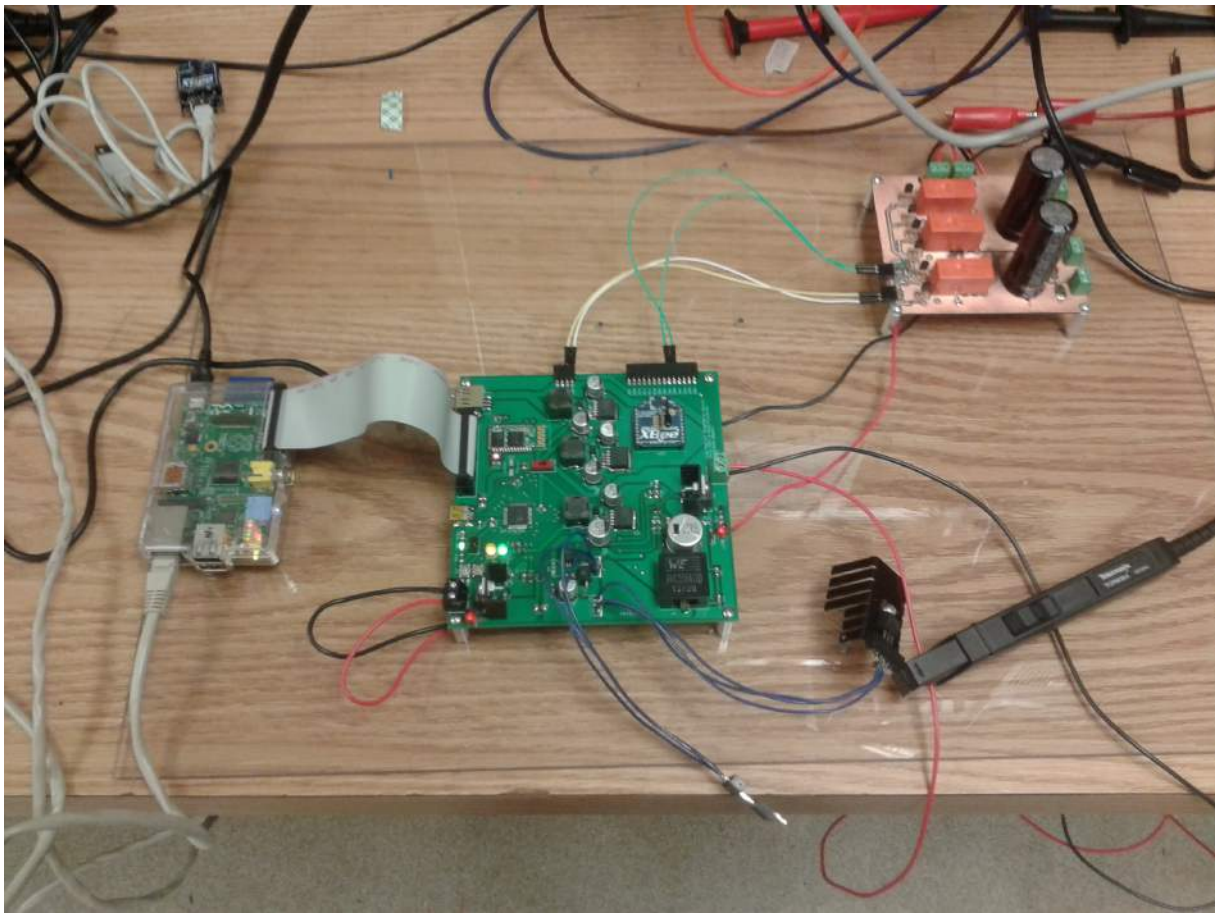


La tarea es crear el sistema de control que gestiona el flujo de energía hacia las baterías. Un microcontrolador *Atmel Atxmega128a4u* lee los sensores tanto a la entrada como a la salida del convertidor de tensión con tal de obtener tensiones y corrientes. Con esa información y un programa adecuado es posible ajustar el ciclo de trabajo D del convertidor (síncrono en el caso de los paneles solares, asíncrono en el caso de la turbina eólica) y así cargar las baterías adecuadamente. Un algoritmo de obtención del máximo punto de potencia es implementado en dicho programa para maximizar eléctricamente (modificando la carga vista por los paneles solares) la potencia entregada por la fuente solar. Esto es posible ajustando únicamente el ciclo de trabajo D.



Protecciones contra polaridad inversa han sido implementadas para evitar daño a los circuitos como a las baterías. En caso de que un usuario conecte los cables de forma invertida se encenderán unos LEDs rojos indicadores.

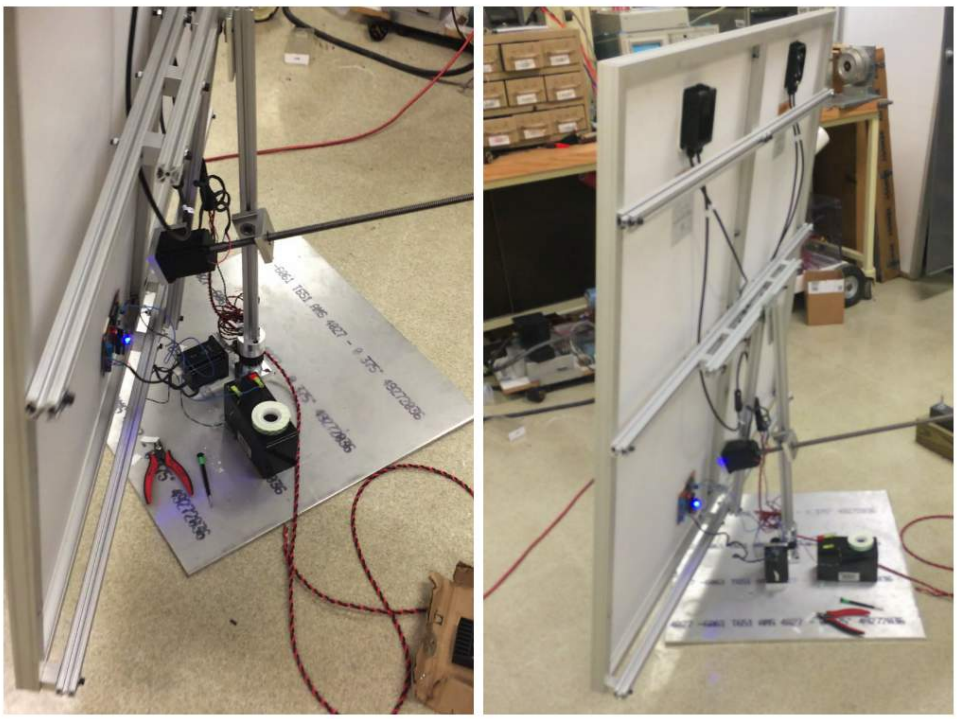
El sistema está dotado de su propio servidor que puede ser conectado con un cable ethernet a un ordenador directamente o más convenientemente a un router para servir la página web a la red local, o a cualquier ordenador conectado a internet, siempre que se configure el router del usuario para que tenga una IP estática, etc. El servidor ha sido implementado usando una Raspberry Pi. Comunica con el microcontrolador a través de conexión serial. La Raspberry Pi envía la posición geográfica del sistema y la hora y recibe los datos de los sensores.



La información es procesada usando un programa escrito en Python que adapta los valores de los sensores para que puedan ser interpretados por una persona y los almacena en una base de datos de tipo Round Robin en su tarjeta SD. Los datos son entonces extraídos de dicha base de

datos para generar resultados como la potencia media generada, la potencia de pico generada, así como gráficos en los que se ve la potencia generada en función del tiempo (tanto para fuente solar como fuente eólica), estado de las baterías y más. Toda esta información es mostrada en múltiples subpáginas de la página web que ha sido desarrollada en un entorno Wordpress con un tema adaptativo. Este permite que se adapte la página web a la resolución del dispositivo que la esté consultado, sea ordenador, tablet o smartphone.

Los usuarios pueden configurar el sistema determinando la situación geográfica en una de las subpáginas de la web. Longitud y latitud son guardadas y enviadas al microcontrolador que a su vez las envía a través de un enlace inalámbrico realizado con XBees al seguidor solar mecánico. Éste usa una brújula y acelerómetro para determinar respectivamente su orientación e inclinación de los paneles solares. El seguidor se mueve usando motores paso a paso.



The present is theirs;
the future, for which I really worked, is mine.
NIKOLA TESLA

Acknowledgements

From the **University of Illinois at Urbana-Champaign**: Professor **Paul Scott Carney** for supervising the project. ECE Lab Assistant **Kevin Colravy** for providing equipment and tools. ECE Service Shop Staff **Mark Smart** for making the PCBs and **Skot Wiedmann** for giving advice.

Texas Instruments for kindly providing a generous quantity of samples and allowing the project to enter the *Innovation Challenge* contest as well as providing a $200 coupon.

AVRfreaks Forum for providing advice using the Atxmega128a4u controller by Atmel.

DOCUMENT I

PROJECT REPORT

Index

List of Figures

List of Tables

List of Code Snippets

Acronyms

ADC	Analog to Digital Converter
ASF	Atmel Software Framework
ECE	Electrical and Computer Engineering
FOCV	Fractional Open Circuit Voltage
GPIO	General Purpose Input Output
GUI	Graphical User Interface
IC	Integrated Circuit
ICAI	Insitituto Católico de Artes e Industrias
INCON	Incremental Conductance
IP	Internet Protocol
IT	Information Technology
LAMP	Linux Apache MySQL PHP
LPSU	Logic Power Supply
MPP	Maximum Power Point
OS	Operating System
P&O	Perturb and Observe
PCB	Printed Circuit Board
PDI	Program and Debug Interface
PFC	Proyecto Fin de Carrera
PSU	Power Supply
PV	Photovoltaic
PWM	Pulse Width Modulation
RPi	Raspberry Pi
RPP	Reverse Polarity Protection
RRD	Round Robin Database
SSH	Secure Shell
TI	Texas Instruments

PART I

INTRODUCTION

1. Preamble

THE use of renewable energies is increasing. The World is now aware of global warming, as well as air and ground pollution. There is an increasing number of people that decide to implement in their power source an alternative to the grid. Also, a percentage of the population is living in rural areas, or decides to move away from the city. Having a source of energy where the grid does not reach is in those cases helpful. This project will try to give a solution by integrating both solar and wind resources as an energy source for households and organizations.

The following sections cover the aspects of the project as a whole but due to its difficulty, the project was in fact divided in three different parts. The three parts, each covered by one student, are the following:

- Controller Board, Solar DC/DC converter. Webserver.
- Wind AC/DC and DC/DC converter.
- Solar Tracker.

This document will cover in detail the first one, the other two will be briefly covered.

1.1. Features

- Combined Solar and Wind power, efficient utilization.
- Active 2-axis sun tracking system.
- Automatic blade furling of wind turbine at excessive wind speeds (no electronics involved).
- Energy storage in a battery bank.
- Current and voltage sensors for real time power measurement.
- Remote information of the system provided through a website (real time information).

1.2. Goals and Intended Function

- Provide an efficient system of using wind and solar power.
- Make the DC/DC converters (solar and wind).
- Control both converters (solar and wind) with a microcontroller.
- Make a physical sun tracker for the solar panels.
- Find and implement an efficient strategy for solar and wind power management.
- Set up a connection between the micronctroller and Internet.
- Make a website where the users will receive feedback from the system.

1.3. Benefits

- User more independent of the grid.
- Able to provide power to isolated areas.
- Cleaner energy source.
- Option of using the system as a Backup (provided by batteries).
- Interaction with the system through a website.
- Advice on website (tells user if for example increasing number of batteries would be beneficial).
- Net metering in some States (get paid by the grid!).
- Excess energy when battery charged can be used to heat water.

2. State of the Art

THE so called hybrid plants are not a novelty. Mixing solar and wind generation is in fact becoming more and more popular at a small scale. Normally the dominating product in the market is a simple hybrid lead acid battery charger that most of the time has no user friendly interface. It does the work but if something fails or the power generation is not optimized the user is not able to find out why. The price of such devices has been found to be above two times the costs of the controller board developed and covered in this project except for those made in China with dubious designs. Some manufacturers also claim that their products have a true MPPT system but they use a fixed voltage algorithm which is very limiting. This project aims not only at creating the hybrid charger but to also give the user the ability to supervise the system.

There are also some interesting features of the project that (to our knowledge) have not been found elsewhere such as using the mathematical formulas for the solar tracker instead of a light sensor, which in some cases could give erroneous data because of dirt, clouds, shades, etc. This feature involves the user in the system because the location (latitude and longitude) have to be set on the website. In the best cases, the products on the market have an alphanumeric display which gives basic information such as total KWh produced, etc. They do not give advanced information such as that presented on the website of the project, with graphs, power averages, battery status, and more.

The product developed here gives the possibility to track and monitor the system from anywhere using the Internet which fits perfectly in these changing times where remote control of smart homes is booming.

PART II

BACKGROUND AND SYSTEM BLOCK DIAGRAMS

1. Background

SOME background knowledge is required in order to develop this project. Besides knowing how to design circuits and program it is fundamental to understand how a solar panel works, otherwise the whole system would not be properly designed and accommodated to its intended function.

1.1. Solar Panels

Solar panels have a peculiar power output curve called the *IV* curve, which represents the output current as a function of the output voltage.

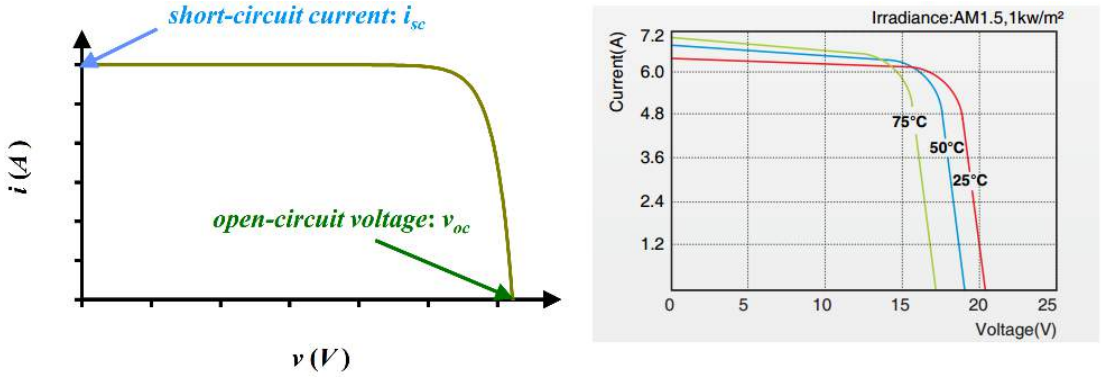


Figure 1. IV Curves, Specific to the Solar Panel used on the right

The operating point of the solar panel corresponds to one point on the curve, and thus the power output of the solar panel corresponds to the area enclosed in the rectangle which upper right corner is the said point and which bottom left corner is the origin.

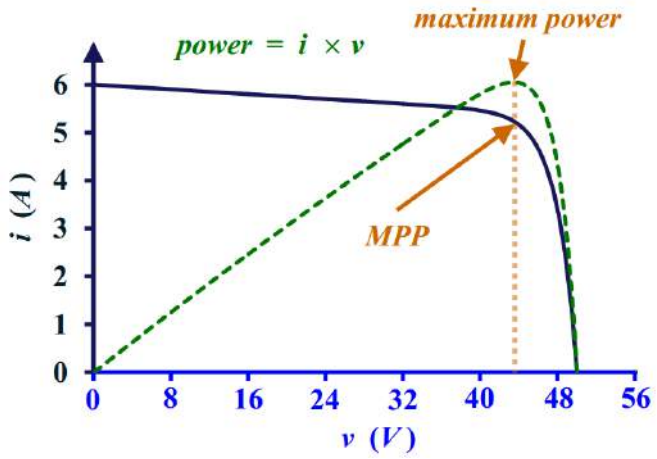


Figure 2. Solar Panel MPP ($\frac{\partial P}{\partial V}$ in green)

The insolation changes the amount of current produced but it is very important to understand that the voltage output remains almost the same. This means that when the setup is decided, i.e number of solar panels in series and parallel, the output voltage will be determined. The output also changes slightly with varying temperatures as seen on Figure 1.

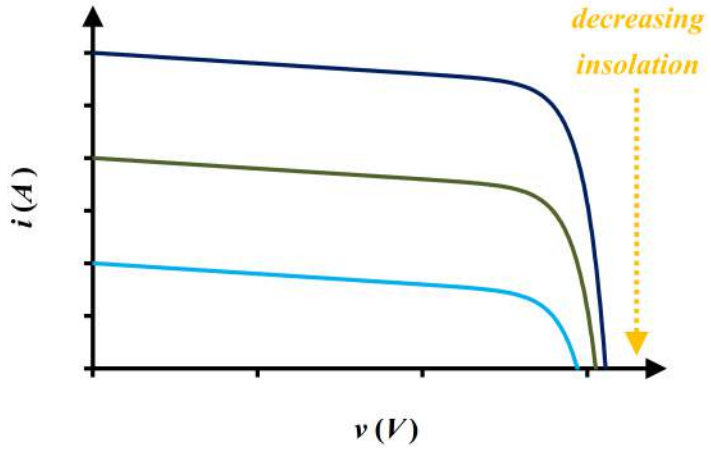


Figure 3. Power Output varies with Insolation

Adding solar panels in series proportionally increases the voltage output and adding them in parallel proportionally increases the current.

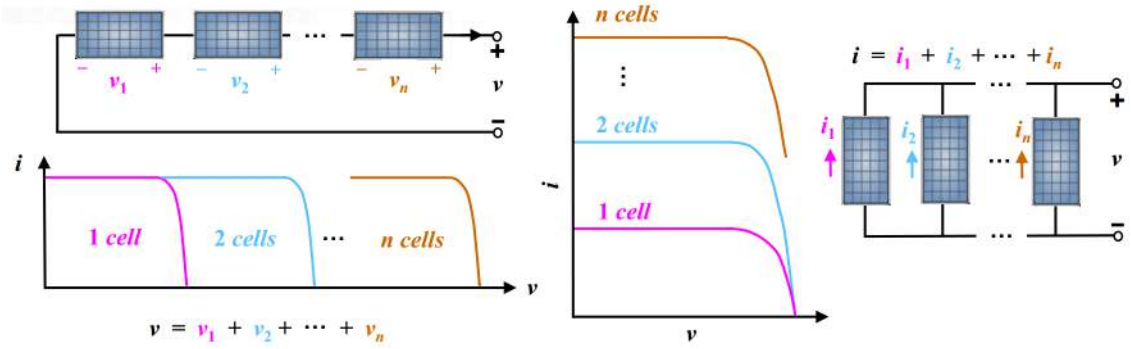


Figure 4. Panels in Series (left) and Parallel (right)

In this project it was decided that it would be more efficient to connect the two solar panels in series to get 35V at the MPP and use a battery bank of 24V (28.8V at maximum charge). Using higher voltages with an efficient converter allows smaller currents to flow through the system and therefore the ohmic losses are smaller. It also gives a higher voltage difference between the input and output than an MPP, for example for a 2x1 instead of a 1x2 PV array, of 17.5V and output of 14.4V (12V battery) guaranteeing that the buck converter will always work despite some small voltage drops from the solar panels and the circuits' ohmic resistance.

1.1.1. PV Load

When a resistive load is applied to the PV array, the operating point is determined by the intersection of the load curve with the IV curve. Because the load is behind the DC/DC buck converter, the load seen by the PV array is variable.

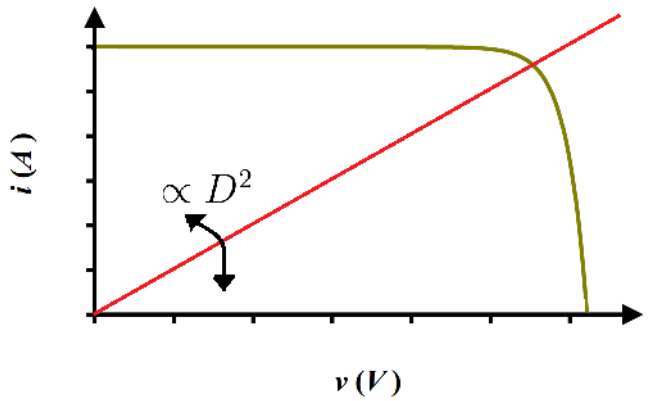


Figure 5. Intersection of Load Curve with IV Curve

The variation of the load produces a change in the slope of the curve. In order to attain the MPP, the load should always be adjusted. In other words, the duty cycle of the DC/DC converter should be adjusted so that the load seen by the PV array (marked with a prime sign : ')changes and intersects the *IV* curve at the MPP.

$$V_{out} = I_{out} R_{load} \tag{1}$$

$$V_{in} = I_{in} R_{load}\prime \tag{2}$$

For a DC/DC buck converter (as shown in next section):

$$V_{out} = D V_{in} \tag{3}$$

$$I_{out} = \frac{I_{in}}{D} \tag{4}$$

Substituting in equation (1):

$$\frac{V_{in}}{I_{in}} = \frac{R_{load}}{D^2} \tag{5}$$

Relating to equation (2):

$$R_{load}\prime = \frac{R_{load}}{D^2} \tag{6}$$

The load seen by the PV array is thus the load on the low side divided by the squared duty cycle.

1.2. Wind Turbine

Although this document does not cover the wind part, the microcontroller was intended to provide also the wind DC/DC converter control signals, hence some knowledge was required.

The characteristic curve of the three phase generator was obtained in the lab because the manufacturer did not provide one.

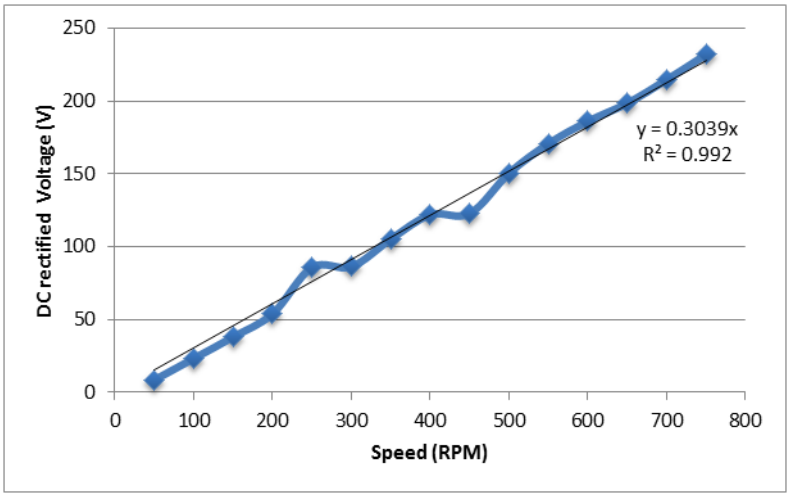


Figure 6. Voltage Speed Characteristic of the Generator

The generator is designed to output a nominal power of 800W but can output up to 1200W. The design of the DC/DC converter must have a thermal design which is an added difficulty compared to the solar DC/DC converter but the control and signal part which is implemented with the controller board is the same. A feedback loop monitors the output voltage and current and takes into account the input current and voltages.

The difference with the solar panels is that the voltage varies constantly and leaves an area where a buck/boost converter would be better than a buck converter. The student in charge of this part decided to dump the power to a resistive load when the voltage is not high enough to be used by a buck converter. That power could be used to heat water for example.

2. System Block Diagrams

THE purpose of this part is to show the different components of the system represented by blocks as well as their interactions. Solid arrows indicate physical connections. Empty arrows indicate wireless connections. The different blocks are described in the following sections of this report.

2.1. System Energy Block Diagram

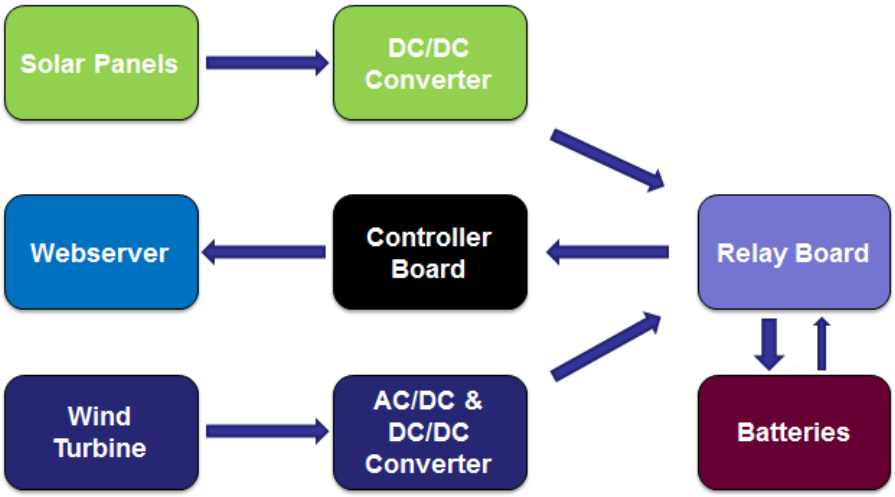


Figure 7. Energy Flow

The PV array is connected to a DC/DC buck converter to step down the voltage. The power is then sent through a relay board to the batteries. The solar panels are mounted on a mechanical 2-axis solar tracker.

The wind turbine is connected to first a rectifier and then to a DC/DC buck converter to step down the voltage. The power is then sent through a relay board to the batteries.

The batteries in turn store the energy but also act as the power supply for the control board, the webserver and the solar tracker. This is because they are a reliable source of power for all the logic even when both sources (wind and solar) are not providing input power. The battery bank is made of two 24V lead acid batteries.

The relay board swaps the energy flow to the batteries. If a battery is full and the other is empty and the power source of the former is producing more energy than the latter's then it will swap the batteries.

2.2. System Data Block Diagram

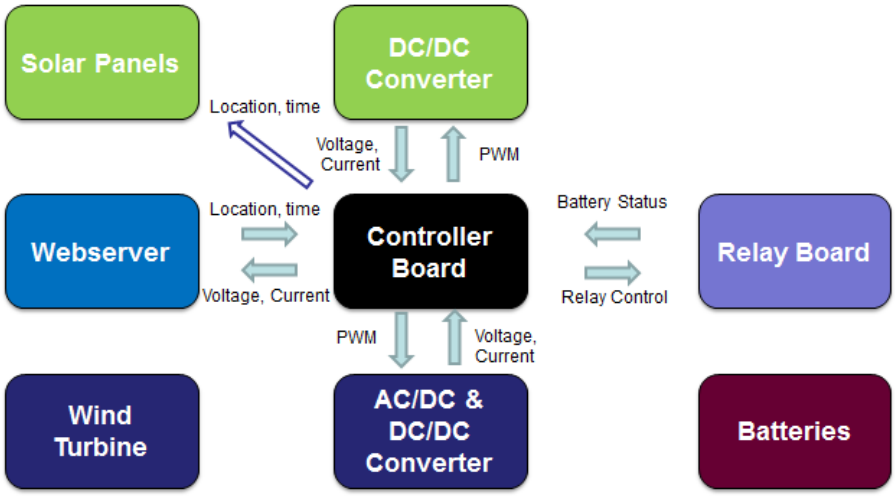


Figure 8. Data Flow

The central hub of information of the system is the controller board. The microcontroller reads the voltage and current sensors and sends the PWM signals that are required to drive properly the voltage converters.

The controller board exchanges information with the webserver. It receives the date and time as well as the location of the system (used to optimize the solar production, orienting the PV array). It also sends the filtered sensor values in order to keep track of the output power, etc.

A wireless link between the controller board and the solar tracker system allows the location to be updated and the time and date to be read by the tracker in order to optimize the solar production mechanically. In other words, the controller board acts as a bridge between the webserver and the solar tracker.

The relays on the relay board are driven by the controller board.

PART III

CONTROLLER BOARD AND SOLAR DC/DC CONVERTER

1. Overview and Block Diagram

THE block diagram here presented is specific to the PCB of the main controller and Solar DC/DC converter of the project.

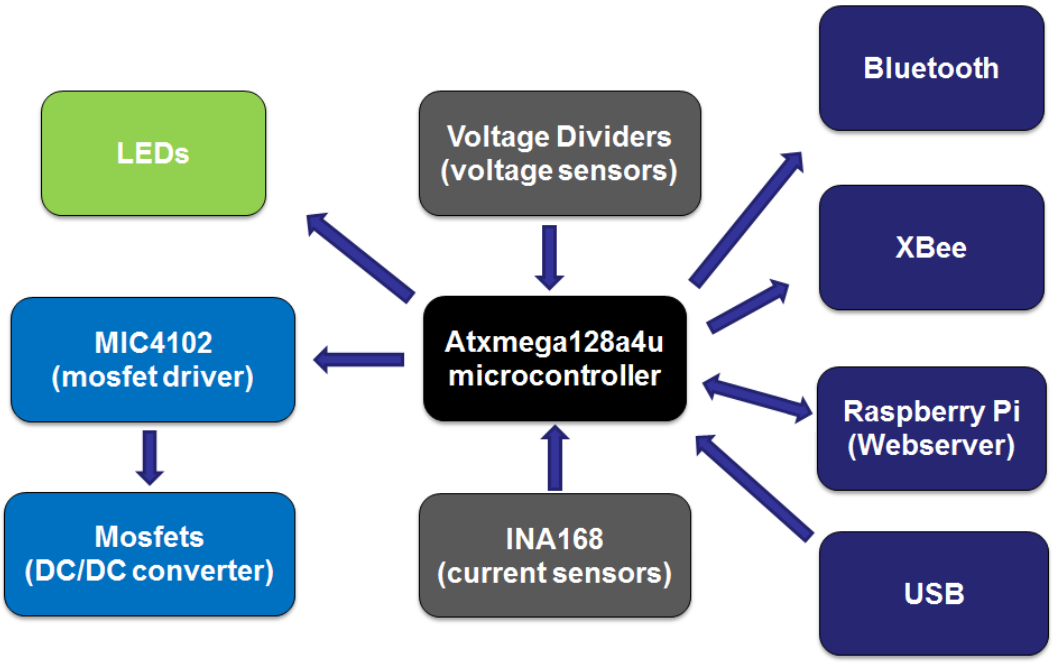


Figure 9. Control and Solar DC/DC Diagram

1.1. Description

One of the power sources in the design is a solar energy source. There are two *Solarland SLP100-12U* [27] 12 VDC solar panels connected in series with a maximum power output 100W each. The panels are mounted on a mechanism to track the sun. The output power is monitored and managed by the controller board.

1.2. Components

1.2.1. Microcontroller

An Atmel Atxmega128a4u was used to control the system. Its 32 MHz clock and numerous ADC inputs as well as USARTs make this microcontroller ideal for this project. It also has an internal USB driver which makes it possible to load a bootloader once using the PDI programming interface and then use the USB for programming the rest of the times.

1.2.2. Voltage Sensors

Voltage dividers were used to bring the voltages to safe levels for the microcontroller (0.95 V maximum) both at the input and output of the board. A capacitor on the low side of the voltage dividers suppresses most part of the noise.

1.2.3. Current Sensors

Texas Instrument's INA168 was used to monitor the current. Theses sensors are able to read a drop of a few millivolts across a resistor (sense resistor) and amplify it to a range of values that the microcontroller can interpret more easily (0 to 0.95V).

1.2.4. Bluetooth

An HC-05 bluetooth module is used to send data using Serial communication to a terminal and observe the data in real time. This feature is intended for debugging purposes, for commercialization it can be removed to reduce costs.

1.2.5. Xbee

This module is used to communicate with the mechanical solar tracking system wirelessly.

1.2.6. USB

This connector permits higher portability. The microcontroller can be programmed directly through USB without the need of a dedicated programmer, it also provides the required logic power for programming. This feature can be as well removed from the final version since it does not add extra functionality for the end user and reduces costs.

1.2.7. LEDs

Two LEDs are used to indicate the state of the battery bank (charging / charged).

1.2.8. DC/DC converter

A buck converter is used to step down the voltage. It has a synchronous mosfet bridge which aims to get higher efficiencies than those of an asynchronous converter. A proper design necessary in order to avoid the *shoot-through* condition (a short-circuit through the mosfets). Using the MIC4102 as the gate driver makes it possible. The PWM signal supplied to the gate driver is produced by the microcontroller at 125 KHz.

1.2.9. Other

There are some extra features on the board which can be observed in *Schematics*:

- A snubber branch to reduce voltage transients on the coil (ringing).
- External schottky in parallel with the Low Side mosfet which reduces switching losses.
- Reverse polarity protection using mosfets and zener diodes, as well as red LEDs to indicate the reverse polarity condition.

Both the DC/DC converter and Controller for the whole system were implemented on the same board to reduce the number of peripheral boards. Thorough heat sinking of the mosfets was not required due to the high efficiency of the synchronous buck converter.

2. Design

THE design of the circuit and choice of the electronic components is the first thing that needs to be done before starting to layout the components on the PCB. The circuit is composed by two different blocks: Logic, and Power management. The former takes care of handling all the communication links, algorithms and data, the latter is mainly composed of the DC/DC converter and handles the power conversion from the solar panel to appropriate characteristics in order to charge the battery bank.

2.1. DC/DC Buck Converter

The two 12V solar panels used are connected in series. From the datasheet we can determine that the MPP for a single panel is near 17.5V. Therefore, when the solar panels are connected in series we obtain an MPP voltage of approximately 35V (which is the intended working voltage). The nominal voltage of the battery bank is 24V. Hence, a buck converter is required.

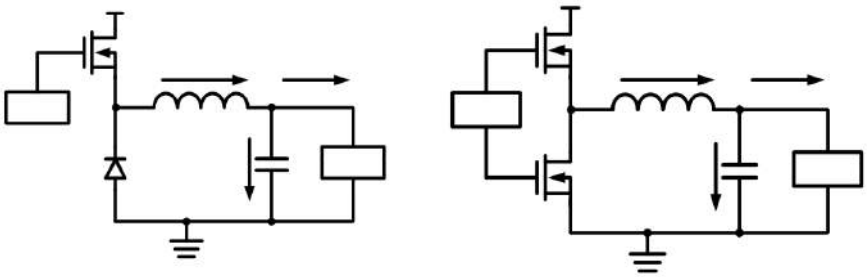


Figure 10. Asynchronous (left) and Synchronous (right) topologies

The reason why a synchronous buck DC/DC converter was chosen over an asynchronous converter is efficiency. Not only it makes the system as a whole more efficient but it also reduces considerably the heat dissipated by the power mosfets, reducing the minimum size of the heatsinks. This option is nowadays viable for systems of low and medium power.

Asynchronous converters use only a high side mosfet and a low side diode. On the other hand, Synchronous converters use mosfets on both sides which increases considerably the complexity because by adding that feature a *shoot-through* condition appears. If both mosfets are turned on at the same time, a short occurs producing the consequent damage. Proper design and choice of the mosfet driver IC is required to avoid that condition at all costs. This results in a higher cost compared to the asynchronous converter solution.

	Asynchronous	Synchronous
Pros	Simpler, Inexpensive	Efficient
Cons	Higher heat dissipation, less efficient	Shoot through condition, more complex and expensive

Table 1. Asynchronous vs. Synchronous DC/DC Converter Comparison

2.2. Synchronous DC/DC Buck Converter Design

In order to determine the correct values of the components of the DC/DC buck converter, a mathematical analysis of the DC/DC Buck converter must be conducted.

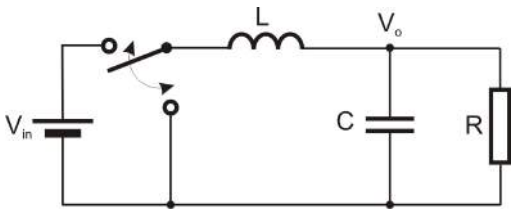


Figure 11. Simplified DC/DC buck converter

When the High Side Mosfet is ON ($t_{ON} = DT$):

$$V_L = L\frac{dI_{LON}}{dt} \tag{7}$$

$$I_{LON} = \int_0^{t_{ON}} \frac{V_L}{L}dt = \frac{V_L}{L}t_{ON} \tag{8}$$

$$V_{in} = V_L - V_{out} \tag{9}$$

$$I_L = \frac{V_{in} - V_{out}}{L}DT \tag{10}$$

When the High Side Mosfet is OFF ($t_{OFF} = T - t_{ON}$):

$$V_L = L\frac{dI_{LOFF}}{dt} \tag{11}$$

$$I_{LOFF} = \int_{t_{ON}}^{T} \frac{V_L}{L}dt = \frac{V_L}{L}(T - t_{ON}) \tag{12}$$

$$V_L = -V_{out} \tag{13}$$

Setting both currents equal we get:

$$V_{out} = DV_{in} \tag{14}$$

This expression is approximate since we don't consider voltage drops across the mosfets, or series resistances of the components, etc. Nonetheless, it gives us the linear nature of the buck converter. By adjusting the duty cycle D the output voltage can be adjusted linearly respect to the input voltage.

2.2.1. Minimum Inductance

Normally we aim for a current ripple (ΔI_L) of 30%.

$$L_{min} = \frac{(V_{in} - V_{out})D}{f\Delta I_L} \tag{15}$$

Where f is the frequency at which the buck is operated.
The parameters for the most demanding conditions are:

$$V_{out} = 21.6V \tag{16}$$

$$V_{in} = 35V \tag{17}$$

$$I_L = (200/35) * (35/21.6) = 9.26A \tag{18}$$

$$\Delta I_L = 0.3 * 9.26 = 2.78A \tag{19}$$

$$f = 125KHz \tag{20}$$

For which:

$$L_{min} = \frac{(35 - 21.6) * (21.6/35)}{125 * 10^3 * 2.78} = 23.82\mu H \tag{21}$$

We will implement a 47μH high quality inductor.

2.2.2. Minimum Capacitance

Normally we aim for a voltage ripple (ΔV_C) of 5%.
Considering the parameters described in the previous section we can calculate the minimum capacitance as:

$$C_{min} = \frac{\Delta I_L}{8f\Delta V_C} \tag{22}$$

$$C_{min} = \frac{2.78}{8 * 125 * 10^3 * 0.05 * 21.6} = 2.57\mu F \tag{23}$$

We will implement a 1000μF capacitor which will stabilize the power output and act as a backup reservoir when the relay board swaps the batteries.

2.2.3. Mosfets

The mosfets are the "switches" of the buck DC/DC converter. Their specifications need to meet the power requirements of the circuit. N-Channel Mosfets with a low R_{dsON} (low power loss, during conducting time) are sought.

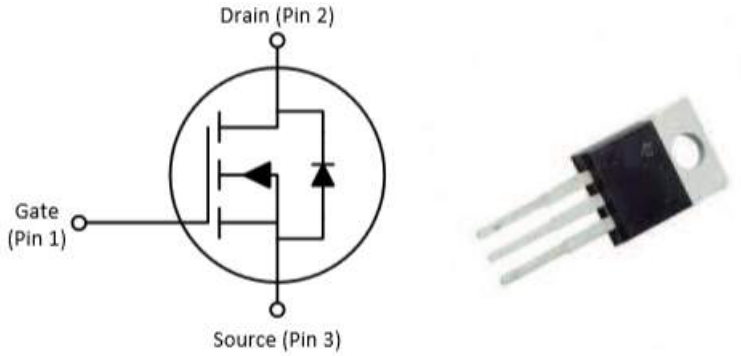


Figure 12. N-Channel Mosfet

A good choice is the CSD18534KCS Mosfet by TI [22]:

T_A = 25°C		TYPICAL VALUE		UNIT
V_{DS}	Drain-to-Source Voltage	60		V
Q_g	Gate Charge Total (10 V)	19		nC
Q_{gd}	Gate Charge Gate to Drain	3.1		nC
$R_{DS(on)}$	Drain-to-Source On Resistance	V_{GS} = 4.5 V	10.2	mΩ
		V_{GS} = 10 V	7.6	mΩ
$V_{GS(th)}$	Threshold Voltage	1.9		V

Figure 13. CSD18534KCS Specifications

It has a low R_{dsON} and its voltage rating is sufficient. The current limitation does not appear on the previous figure but it is 52A at 100ºC, so it can handle easily the current of the application.

In order to improve the efficiency of the converter, a schottky diode can be added in anti-parallel mode with the low side mosfet. It will conduct quicker than the mosfet when in the transition zone as long as its voltage drop is lower than the mosfet´s.

2.2.4. Mosfet Driver

It is not viable to connect the power mosfets directly to the output pins of the microcontroller because their gates demand a high charge quantity in a small period of time, that is in other words, a high current. A gate driver is therefore required. The singular synchronous topology has the *shoot-through* condition, it would be wise to use a driver that takes care of that. Another feature which is also important is that it should be able to switch its outputs with a 3.3V signal (PWM). There are many drivers in the market, but the simplicity of the MIC4102 by Micrel [24] makes it a good choice.

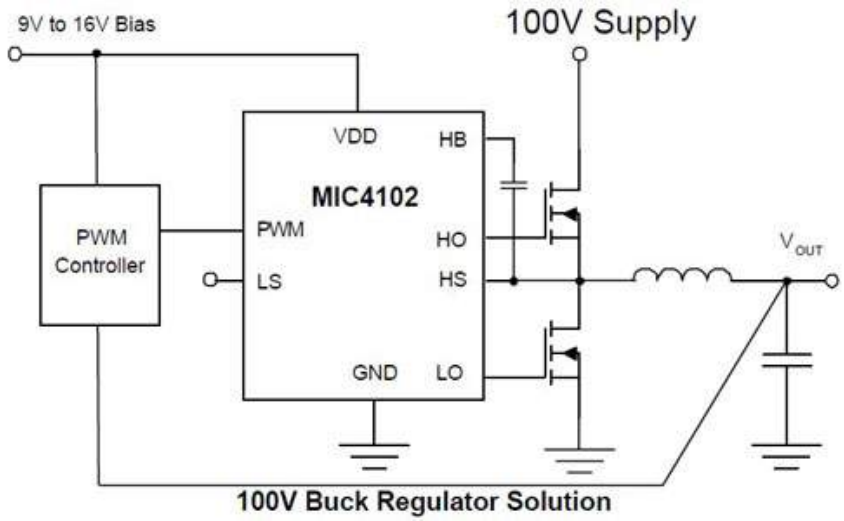


Figure 14. MIC4102 Setup

The driver has an internal bootstrap diode, but we can add an external diode should it be necessary (it would remove some heat off the driver). The bootstrap capacitor needs to be sized depending on the mosfets of the converter. From Figure 13, the total gate charge of the mosfets is 19nC. Following the driver´s datasheet:

$$C_B \geq \frac{Q_{gate}}{\Delta V_{HB}} \tag{24}$$

where:

- Q_{gate} = Total Gate Charge at ΔV_{HB} (ΔV_{HB} sould be less than 0.1V).
- ΔV_{HB} = Voltage Drop at the HB Pin.

$$C_B \geq \frac{19 * 10^{-9}}{0.1V} \tag{25}$$

A value higher than 190nF, like 1uF can be used.

2.3. Reverse Polarity Protection Design

The RPP feature is implemented to make the design as marketable and safe as possible. A user that mistakenly plugs the solar panels or the battery bank the wrong way might not only destroy all the logic components due to incorrect voltages but also create a short and in the worst case scenario damage the solar panel as well as create a battery explosion. Therefore, RPPs were implemented both at the input and output of the board as well as red LEDs that notify the user of the situation.

Multiple RPP options were studied before choosing the one that would be implemented. There are two main alternatives.

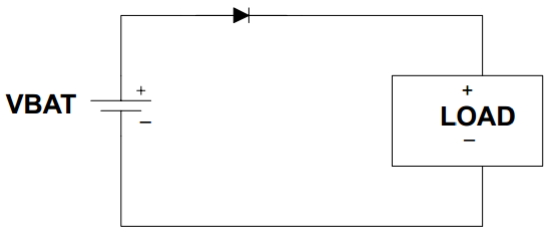


Figure 15. RPP with Diode

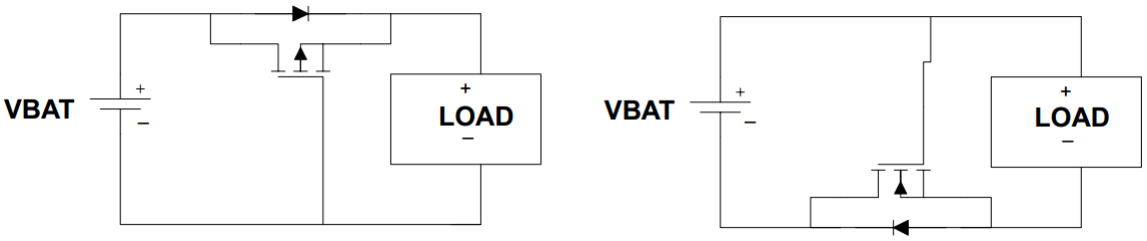


Figure 16. Simplified RPP with NMosfet (right) and PMosfet (left)

The diode is an effective and simple solution although it is inefficient for high currents. The voltage drop of 0.7V (or in the best case 0.4V for a schottky diode) leads to a high power loss. We are expecting a maximum of around 8A at the input and 10A at the output which ends up in several Watts of loss (with their inherent problem of heat sinking, etc).

A better solution is to use either an N-channel mosfet or a P-channel mosfet where the only loss is a result of the R_{dsON}. They act like switches, when the source is connected the wrong way, no electrons will flow. In order to achieve this, proper biasing of the gate of the mosfets is required. A diode and a resistor as shown in the *Schematics* will take care of this.

	Diode	**Mosfet**
Pros	Simpler, Cheaper	Efficient
Cons	Higher heat dissipation, less efficient	More complex and expensive

Table 2. Diode vs. Mosfet RPP Comparison

We are targeting maximum efficiency, therefore the best option is to implement the mosfet solution.

2.4. Sensors Design

Sensors are used to measure voltage and current both at the input (from the solar panels) and at the output (to the batteries) using the ADC of the microcontroller. These values can be multiplied to produce respectively the power input and power output of the system.

2.4.1. Voltage Sensors

The voltage sensors can be built using accurate resistors (1% should suffice), filtering the output with a 100nF capacitor.

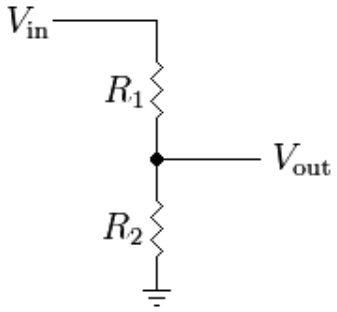


Figure 17. Voltage Divider

It is easy to derive that:

$$V_{out} = V_{in}\frac{R_2}{R_1 + R_2} \tag{26}$$

The microcontroller used in this project is the Atxmega128a4u which accepts up to 0.95V as maximum input voltage to the ADC. This must be taken into account to select the values of the resistors.

We also must consider that a good adjustment of the resistors will allow to exploit the full range of the 12 bit ADC. The upper limit (maximum possible voltage) must fall within the ADC range (below 0.95V) but close to 0.95V so as to not lose resolution within the whole range. The pins of the Atxmega128a4u are 3.3V tolerant and therefore are quite safe.

For the highest possible input voltage we have to assign the 0.95V limit:

$$V_{in-max} = 45V \tag{27}$$

There is one degree of freedom. By fixing a value, the other is computed. Let's take :

$$R_1 = 220k\Omega \tag{28}$$

For that value of R_1 any value of $R_2 \leq 4.7k\Omega$ will work. We will implement a 4.02kΩ resistor.

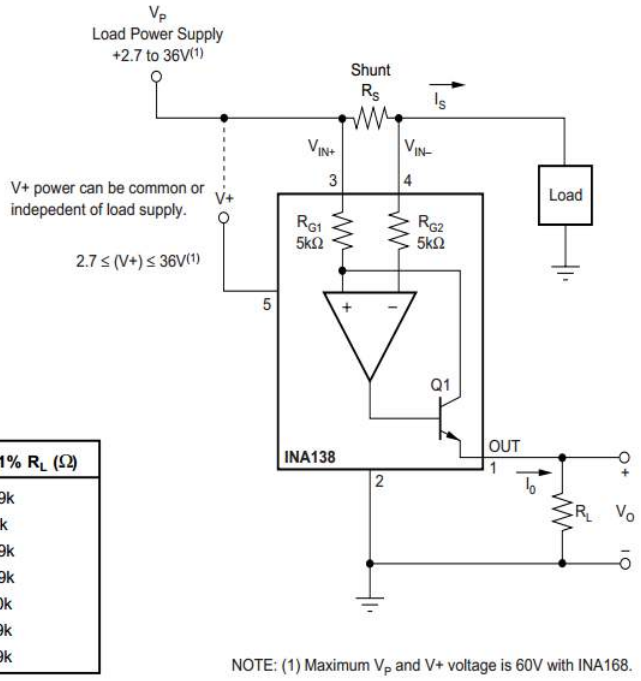
For the highest possible output voltage we have to assign the 0.95V limit:

$$V_{in-max} = 31V \tag{29}$$

Using the same resistor value for R_1 as before we obtain that $R_2 \leq 6.9k\Omega$. We will implement a 6.89kΩ resistor.

2.4.2. Current Sensors

The current sensors are built around TI's INA168 IC. The chip outputs a voltage that is proportional to the current.



VOLTAGE GAIN	EXACT R_L (Ω)	NEAREST 1% R_L (Ω)
1	5k	4.99k
2	10k	10k
5	25k	24.9k
10	50k	49.9k
20	100k	100k
50	250k	249k
100	500k	499k

Figure 18. INA168 Setup

As we can see in the figure, the INA168 is a sort of amplifier, which gain is determined by the output resistor R_L. The sense resistor R_S must be of very small value since all the current that goes through the system goes through this resistor producing a voltage drop of a few millivolts. This voltage drop is then picked up, amplified and fed to the ADC of the microcontroller. As for the voltage sensors, there is one current sensor at the input of the system and one current sensor at the output of the system.

From the datasheet we have:

$$V_{out} = kR_S I_S \tag{30}$$

From the table in Figure 18 the relation between the gain k and the resistor R_L can be determined:

$$k = \frac{R_L}{5k\Omega} \tag{31}$$

A sense resistor of 5mΩ is a good compromise between sensitivity (must have a voltage drop big enough so the INA168 can pick it up) and power lost in the form of heat.

For the highest possible input current we have to assign the 0.95V limit:

$$I_{S-max} = 8A \tag{32}$$

For which we obtain that the gain should be $k = 23.75$ with a value for $R_L \leq 118.75k\Omega$. We decide to implement a 115kΩ gain resistor.

For the highest possible output current we have to assign the 0.95V limit:

$$I_{S-max} = 10A \tag{33}$$

For which we obtain that the gain should be $k = 19$ with a value for $R_L \leq 95k\Omega$. We decide to implement a 95kΩ gain resistor.

2.5. Communication Design

The controller board is a hub. It exchanges information with the RPi, it also sends information to the solar tracker and it should also be able to send information over bluetooth. All communication peripherals interface with the microcontroller using a serial link.

2.5.1. RPi Connector

The RPi interface consists of a 20 pin (10 pins, 2 lines) connector. The ribbon cable provided with the RPi attaches all its GPIOs to the controller board through the said connector, although only a few pins are actually used. The serial protocol/interface only requires that the voltage levels be the same. In this case, both the Atxmega128a4u and the RPi operate at 3.3V. Two lines: transmission and reception, as well as ground are required.

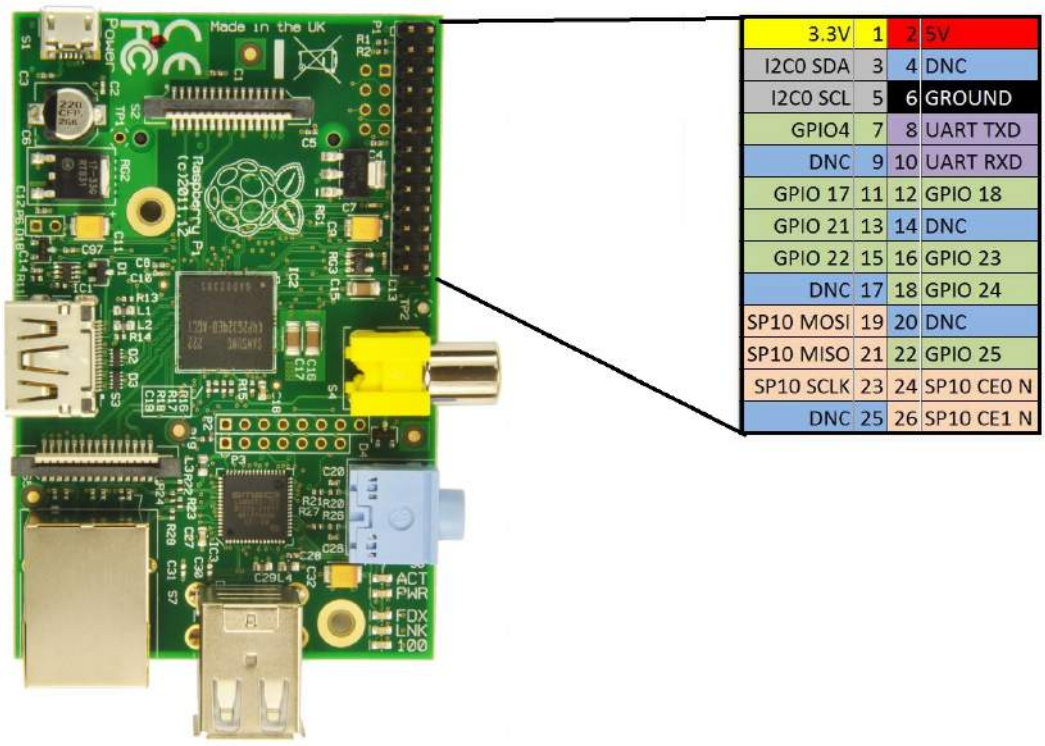


Figure 19. Raspberry Pi GPIOs

2.5.2. XBee

The XBee module was chosen over a bluetooth module to communicate with the solar tracker despite its higher cost because of its much higher range. Bluetooth modules normally have a range of 10m which is not enough if the controller board is to stay in a house, the solar tracker being in the backyard or at a considerable distance. The XBee has a range of 30m with walls or 100m line-of-sight range. A model with a wire antenna was chosen in order to avoid any interference and loss of range due to ground planes on the PCB, which trace antennas are very sensitive to. The XBee operates at 3.3V which makes it suitable for the xmega microcontroller.

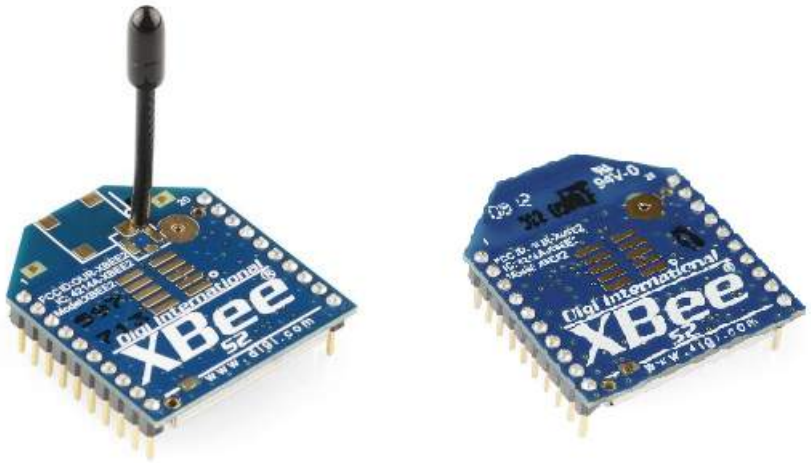


Figure 20. Wire Antenna (left) and Trace Antenna (right) XBee models

2.5.3. Bluetooth

A bluetooth module was used in order to do some debugging in the first stages of the project and easily check the readings and outputs of the microcontroller. Using a terminal on a computer such as $TeraTerm$ or $BlueTerm$ on a smartphone it is possible to debug on the go. The XBee could also have been used for this purpose but the initial configuration steps and having to carry around the receiving module, were not worth the small amount of money the HC-05 bluetooth module costs. The HC-05 operates at 3.3V which makes it suitable for the xmega microcontroller.

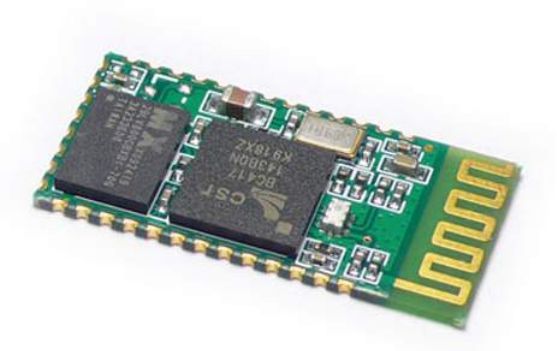


Figure 21. HC-05 Bluetooth Module

2.6. Logic Power Supply Design

The LPSU is built around TI´s LM2576 which is a buck DC/DC converter IC. It exists in four versions: 3.3V, 5V, 12V and adjustable output voltage. In this project the first three types were used. This IC is able to provide up to 3A and suits our needs, being also availabe with an SMD package.

Overview of the load distribution:

- 3.3V rail : Microcontroller, XBee, HC-05, LED, Output to peripheral boards.
- 5V rail (shared with USB power): RPi, Output to peripheral boards.
- 12V rail : INA168, MIC4102, Output to peripheral boards.

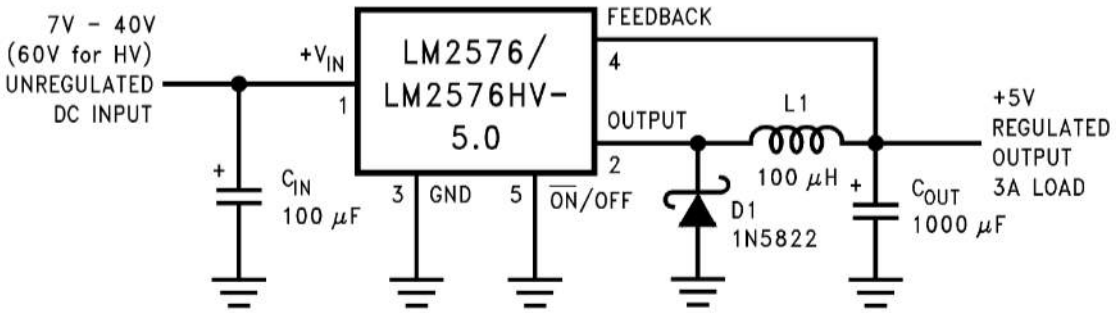


Figure 22. LM2576 Setup

Each version of the LM2576 requires different external components that also need to be adjusted considering to the current drawn by the load. The manufacturer provides a datasheet in which external components are recommended. Following those guidelines we obtain the following components for our application:

	Diode	**Input Capacitor**	**Output Capacitor**	**Inductor**
LM2576-3.3V	1N4148	100μF	470μF	150μH
LM2576-5V	1N4148	100μF	470μF	150μH
LM2576-12V	1N4148	100μF	470μF	680μH

Table 3. LM2576 External Components

3. Simulation and Prototyping

BEFORE implementing the circuit we have to make sure that the assumptions and values calculated in the design are correct. An easy way to test the power part of the design is by creating the circuit and simulating it in *Pspice* [17]. The logic part is prototyped.

3.1. Pspice Circuit

There are no logic devices in *Pspice* such as microcontrollers, therefore the signals to drive the mosfet gates must be produced by other means. A programmable signal where the HIGH and LOW portions of a period as well as the frequency are specified suffices. In our case, the frequency was set to 125 KHz and the duty cycle to 80% (which should produce a voltage output of 28V). There are also no battery models in *Pspice*, but the load can be modeled using a resistor which value is tuned to accommodate the expected load current. The RPP model needs also to be tested. This is achieved by introducing switches in the circuit which will toggle and reverse the connection, in the following example, of the battery.

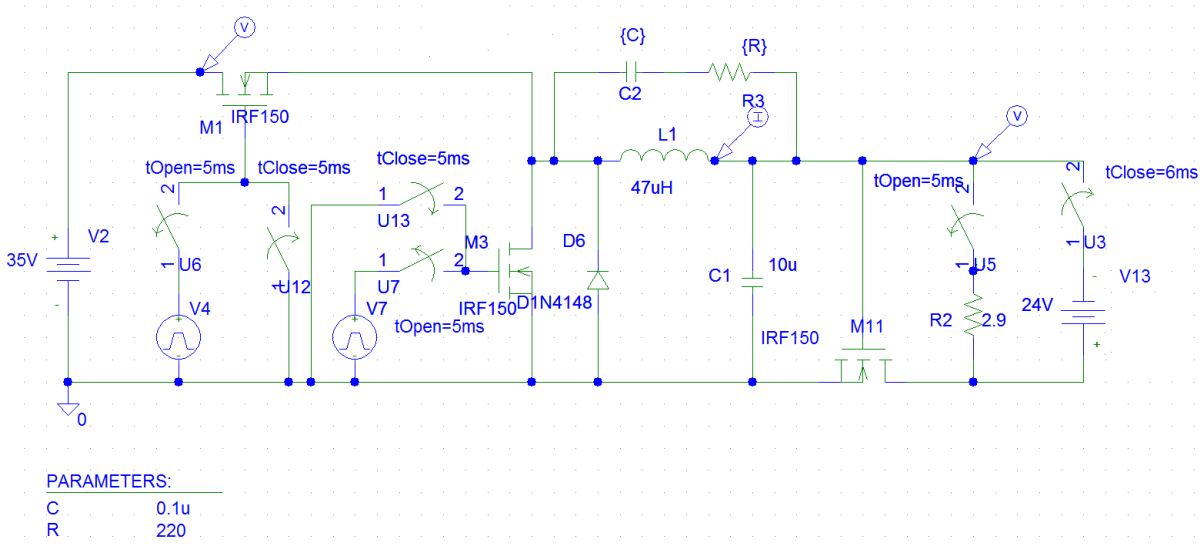


Figure 23. Circuit of the Power Part in Pspice

The simulated output current is higher than what we will be using and the circuit works, the output voltage is approximately equal to the input voltage times the duty cycle. As we can see, at $t = 5ms$ a set of switches opens and at $t = 6ms$ the other set of switches closes. The current flowing back into the circuit (blue series) is null, which indicates that the RPP works. It is worth noting that the transients are very good with almost no oscillations which means that the snubber branch is also working.

The big spikes at $t = 5ms$ do not occur in normal operation. The sets of switches were only used for demonstration purposes.

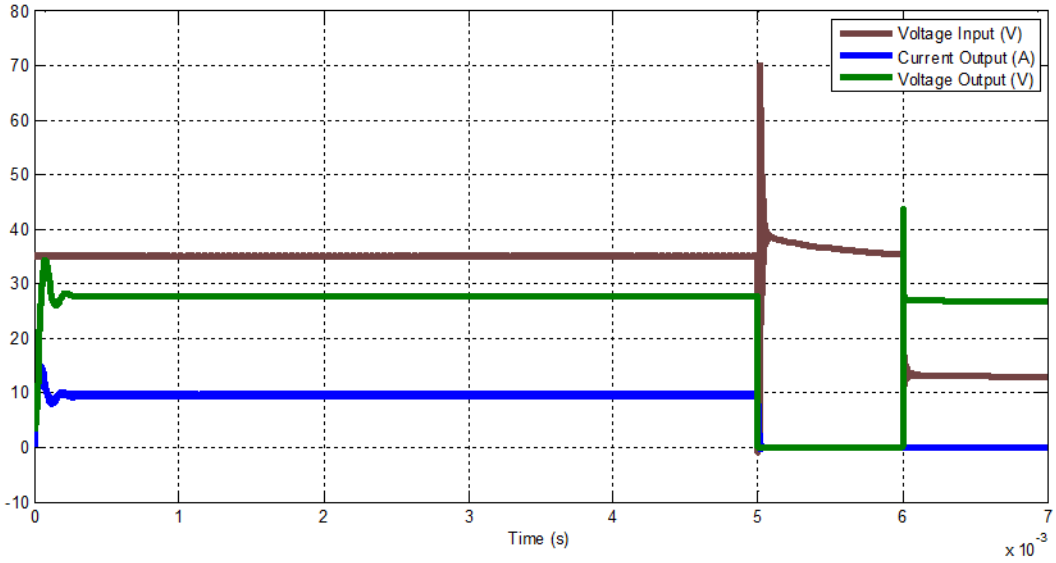


Figure 24. Simulation of the Power Part in Pspice

3.2. Prototyping

A manually made custom board for the Atxmega128a4u was used to prototype the various communication links and controls before designing the final PCB.

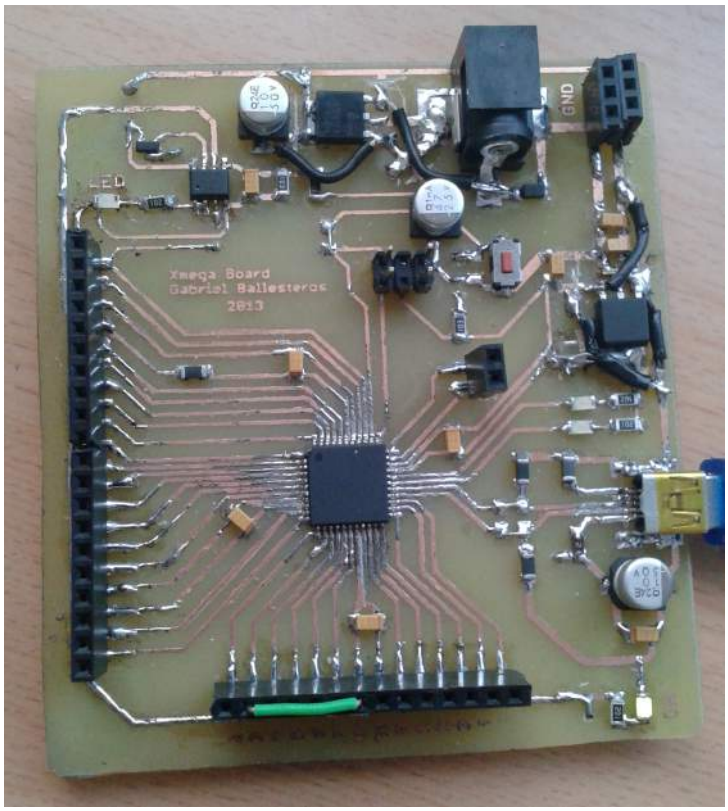


Figure 25. Atxmega128a4u Prototyping Board

4. Schematics

THE schematics were produced using *Cadsoft Eagle* [10]. The size of the board was bigger than that allowed for the student version of the software, therefore the professional version had to be used. A few components needed to be created because they did not exist in the included libraries.

4.1. Logic Power Supply

The LPSU produces the required voltage levels for all the logic parts of the circuit to work. The power source is the battery bank which seats around 24V to 28V. Hence, linear voltage regulators must be discarded. The best option is to use switch mode power supplies which are highly efficient (above 90%). The LM2576 IC by Texas Instruments [25] requires very few external components and yields enough current for this application.

- An LM2576 is used for each voltage level : 3.3V, 5V and 12V. The inductors were selected according to TI's LM2576 datasheet, as well as the diodes.
- An LED is used to show that the 3.3V rail is being powered (the microcontroller is hooked to this voltage level).
- An USB connector is placed only to conveniently power the RPi.
- All voltage levels are output in order to power other parts of the system such as the wind DC/DC converter and the relay board.

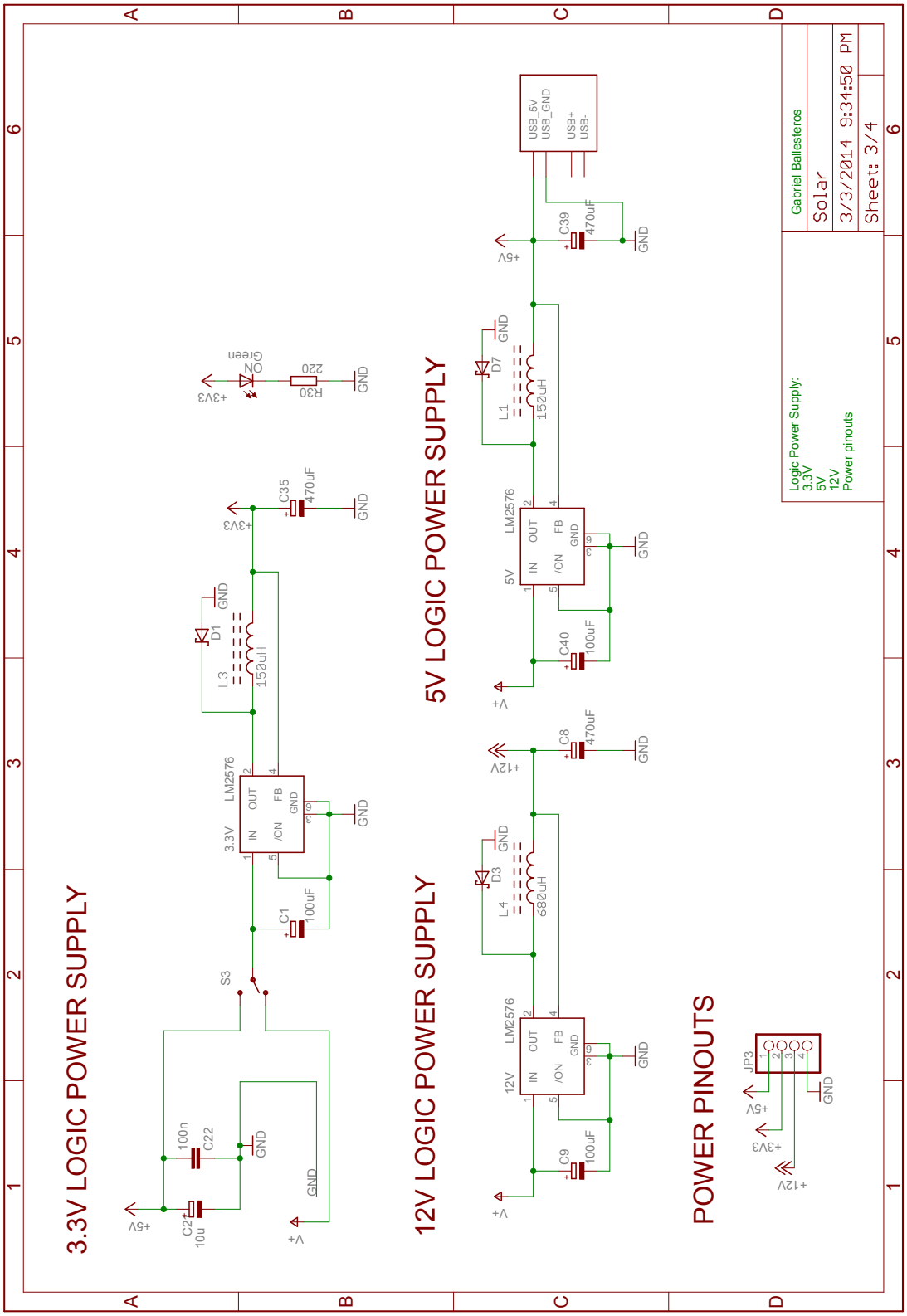


Figure 26. Logic Power Supply

4.2. Communication Modules

The ATxmega128a4u interfaces with its peripherals using multiple serial ports.

- An XBee module is used to communicate with the Solar Tracker. Besides the serial lines, an additional pin is used, which allows the XBee to enter low power consumption should it be required.
- A bluetooth module with an LED which indicates if the device is paired.
- A RPI interface, which makes it easier to connect using a 20 pin ribbon cable which also allows the common ground connection to ensure that the serial link works properly.

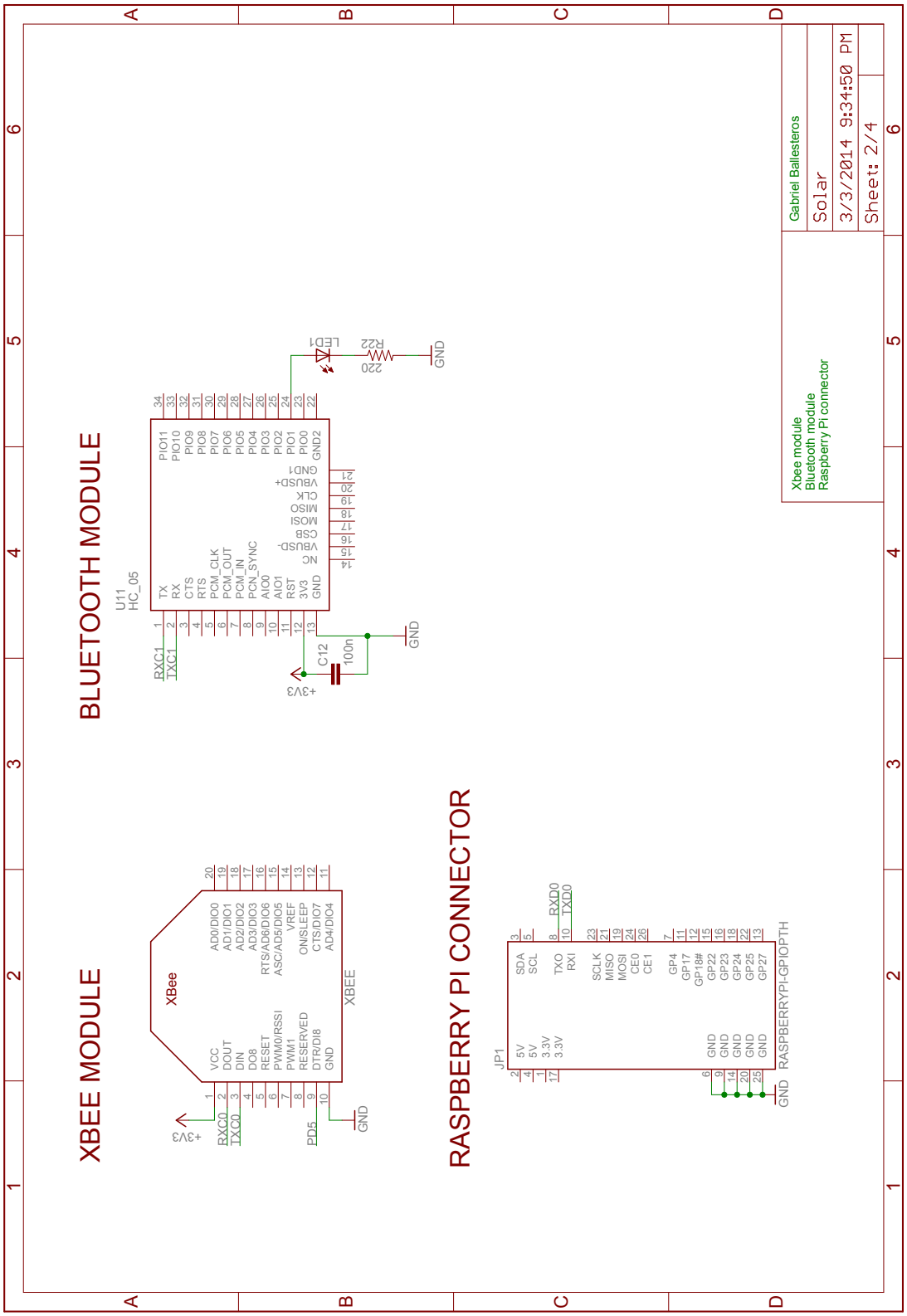


Figure 27. Communication Modules

4.3. Microcontroller

The ATxmega128a4u requires a few external components.

- Decoupling capacitors.
- A reset button.
- A programming button (allows bootloader).
- A programming interface (which uses the PDI protocol).
- A small ferrite bead to filter noise for accurate analog readings.
- A USB interface is used here for increased portability; it enables the microcontroller to be programmed without a specific programmer once a bootloader has been uploaded to its memory. This feature is beneficial for development purposes but can be removed for commercialization.
- Unused pins are redirected to the peripheral boards using a female connector.
- LEDs are used to indicate the state of the battery bank.

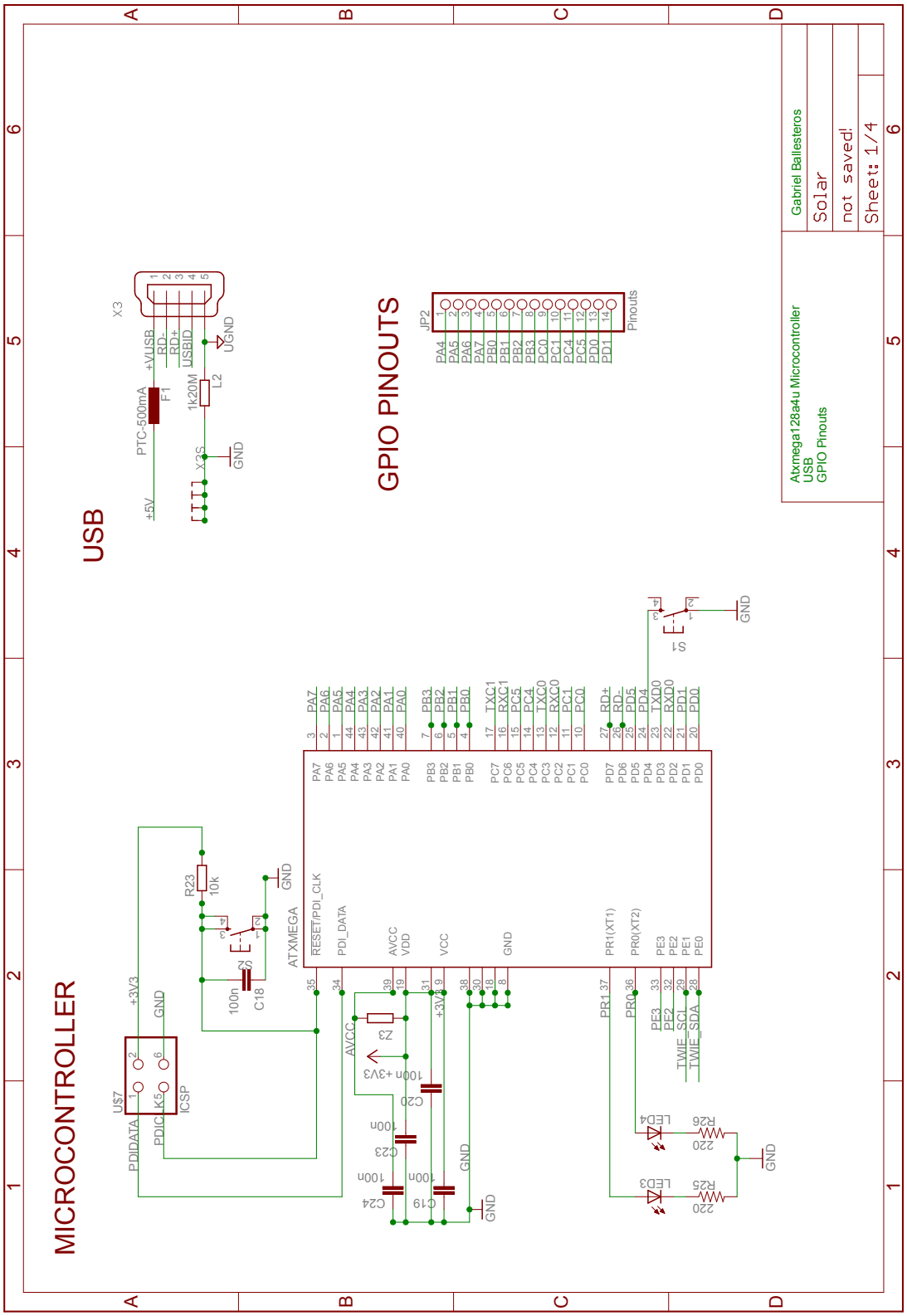


Figure 28. Microcontroller

4.4. Solar DC/DC Converter

The power management part of the circuit is handled by the solar DC/DC converter. It is a synchronous buck DC/DC converter. It has additionally some improving extra futures other than the converter itself.

- RPP both at the input and output.
- TI INA168 as current sensor at the input and output.
- Voltage dividers with output filtering capacitors.
- Micrel MIC4102 as a synchronous mosfet driver.
- A snubber circuit in parallel with the coil.
- The diode D4 is optional (it can avoid overheating of the mosfet driver), the driver has an internal diode.
- Capacitors before the DC/DC converter to filter possible noise.
- Higher electrolytic capacitor value than required to smooth out the output with a small parallel ceramic capacitor to eliminate switching noise.

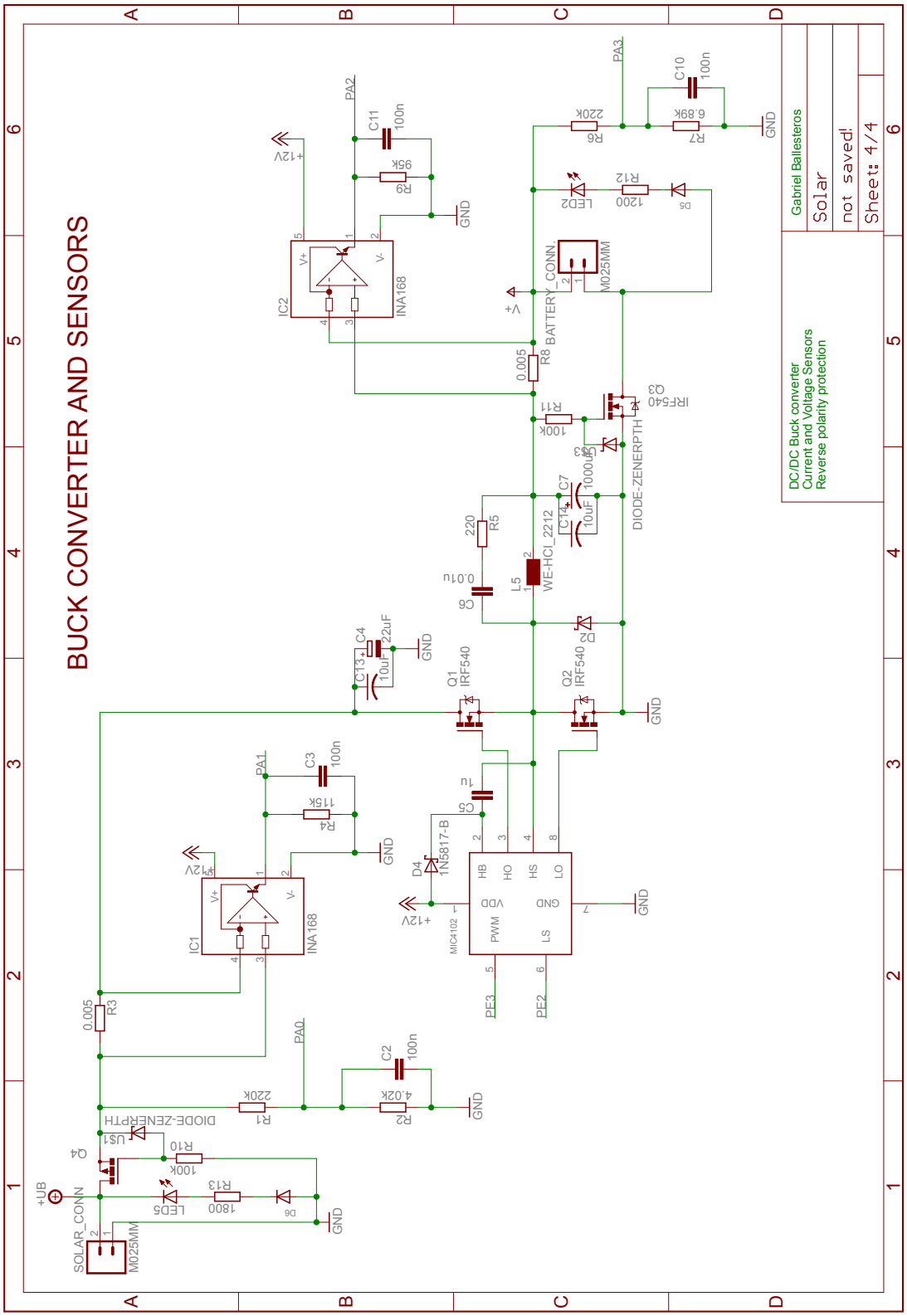


Figure 29. Solar DC/DC Converter

5. PCB

THE PCB of the controller board was designed using *Eagle* as well. It was decided that it would be manufactured professionally by *Advanced Circuits*. They have a student promotion where it is possible to order a PCB for $33, and it is shipped in less than a week. In order for them to make the PCB it was necessary to follow their requirements, that is the Gerber files format, tolerances, trace width, etc. This was a good experience to see how professional manufacturing companies work.

5.1. Eagle Layout

The layout was done following the main directives of PCB design. Right angles were avoided, manufacturer layout hints for ICs was followed using the datasheets whenever possible, number of vias was minimized, etc. The power part (DC/DC converter) was kept on one side of the board while the logic circuitry and signals were on the other side of the board. This was done in order to get unaltered signals by the switching high currents.

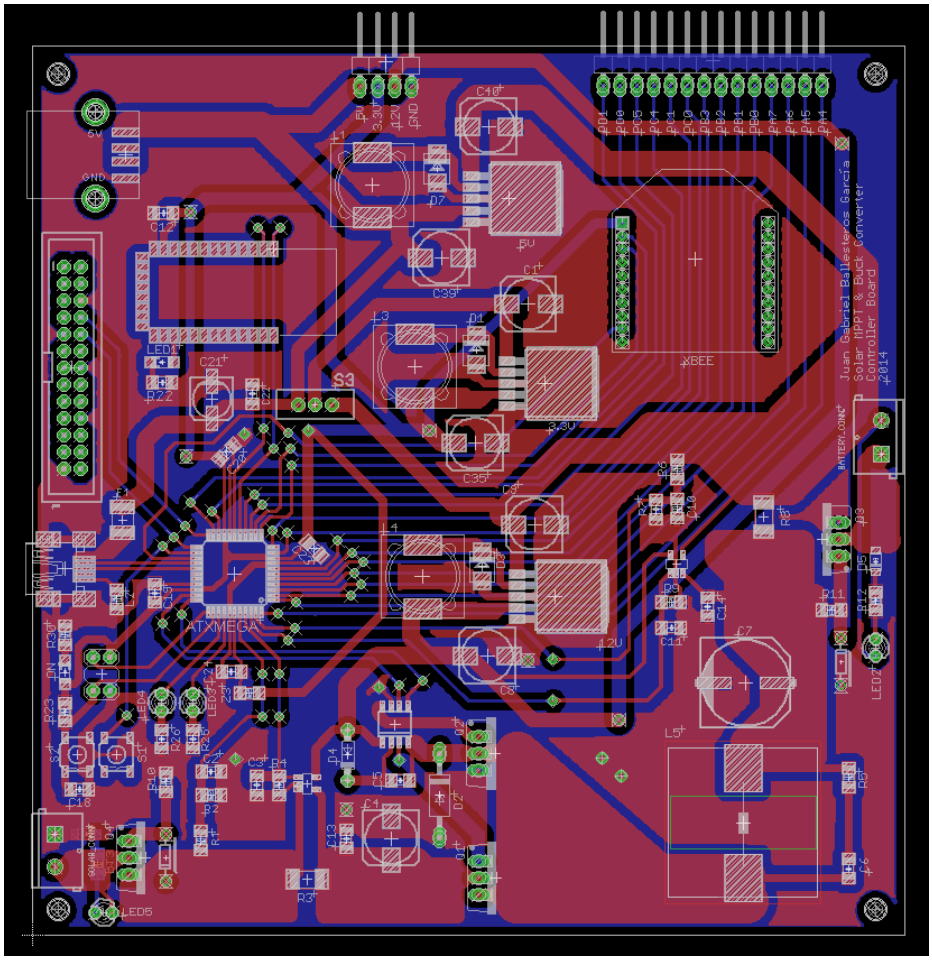


Figure 30. Solar Controller and Converter PCB

5.2. Manufactured and Assembled PCB

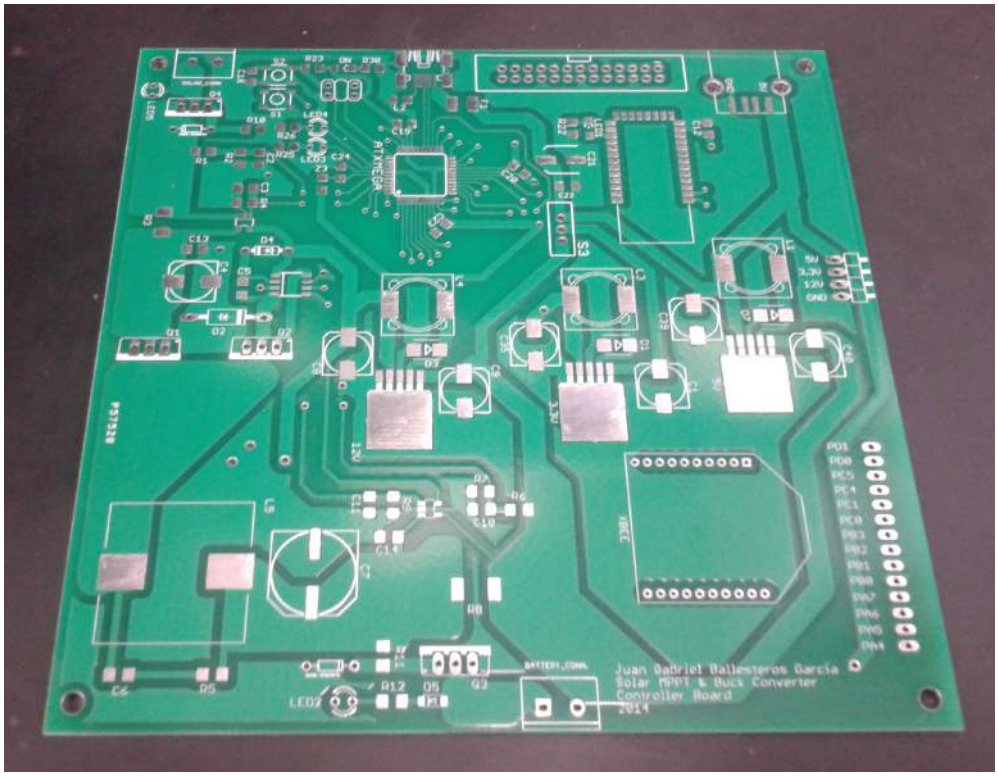


Figure 31. PCB Manufactured by *Advanced Circuits*

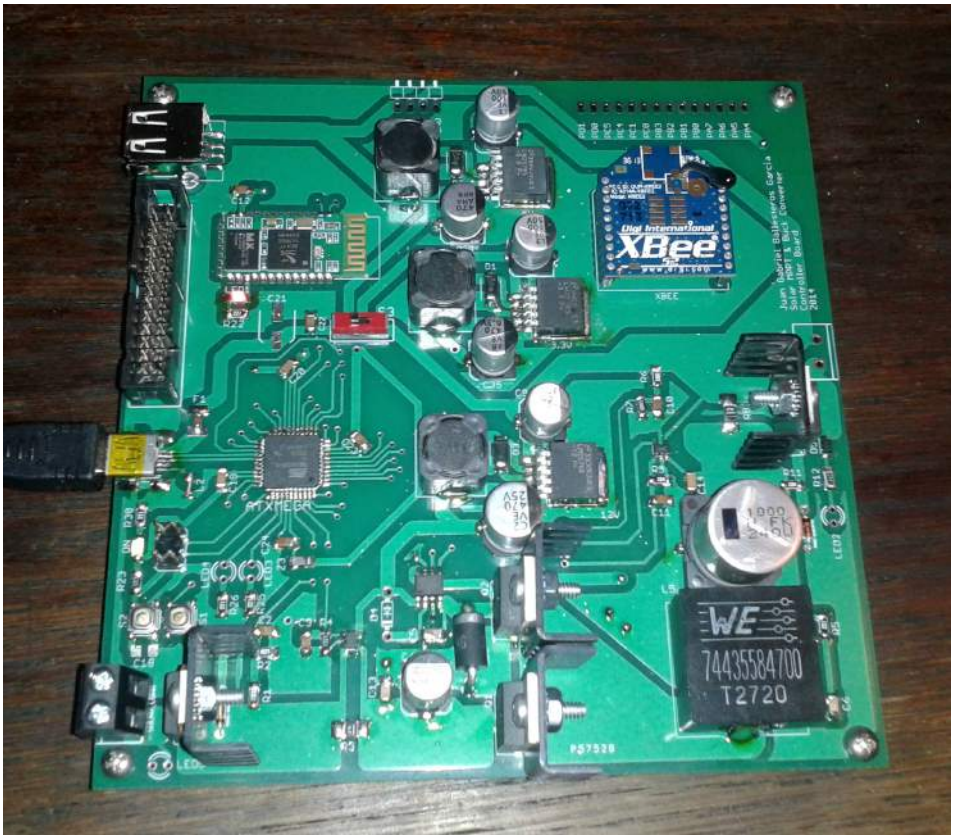


Figure 32. Assembled (few components missing) PCB

6. Software

THE program of the microcontroller is extensive and it is developped in *Part VI Code*. Besides the typical configuration that needs to be done on a microcontroller, there are the specific functions for this project such as the optimizing algorithm, the closed loop control for the PWM signals, etc.

6.1. Bootloader

The bootloader is what allows the microcontroller to be programmed using USB. It is provided by *Atmel* but was changed because the default pin used to enter the bootloader could not be used. The bootloader is long and can only be compiled using specific tools such as *IAR Embedded Workbench*. The following lines had to be changed in *conf_isp.h*:

```
%This portion was changed
#define ISP_PORT_DIR PORTC_DIR
#define ISP_PORT_PINCTRL PORTC_PIN3CTRL
#define ISP_PORT_IN PORTC_IN
#define ISP_PORT_PIN 3

%To this
#define ISP_PORT_DIR PORTD_DIR
#define ISP_PORT_PINCTRL PORTD_PIN4CTRL
#define ISP_PORT_IN PORTD_IN
#define ISP_PORT_PIN 4
```

Code 1. Change Installation File

6.2. MPPT Algorithm

An important component of the code for the solar part is the MPPT algorithm. There are a few algorithms currently in use in solar optimizers: "Fractional Open Circuit Voltage", "Perturb and Observe" and "Incremental Conductance".

In this project the "Incremental Conductance" algorithm was chosen over the other two due to its superiority.

	FOCV	**P&O**	**INCON**
Pros	Simple, Inexpensive	Simple	Accurate, Finds the optimum and stays
Cons	Requires computing of constants, disconnects periodically PV array	Oscillates naturally around the optimum, never stays still	Requires more computing power and sensors

Table 4. MPPT Algorithm Comparison

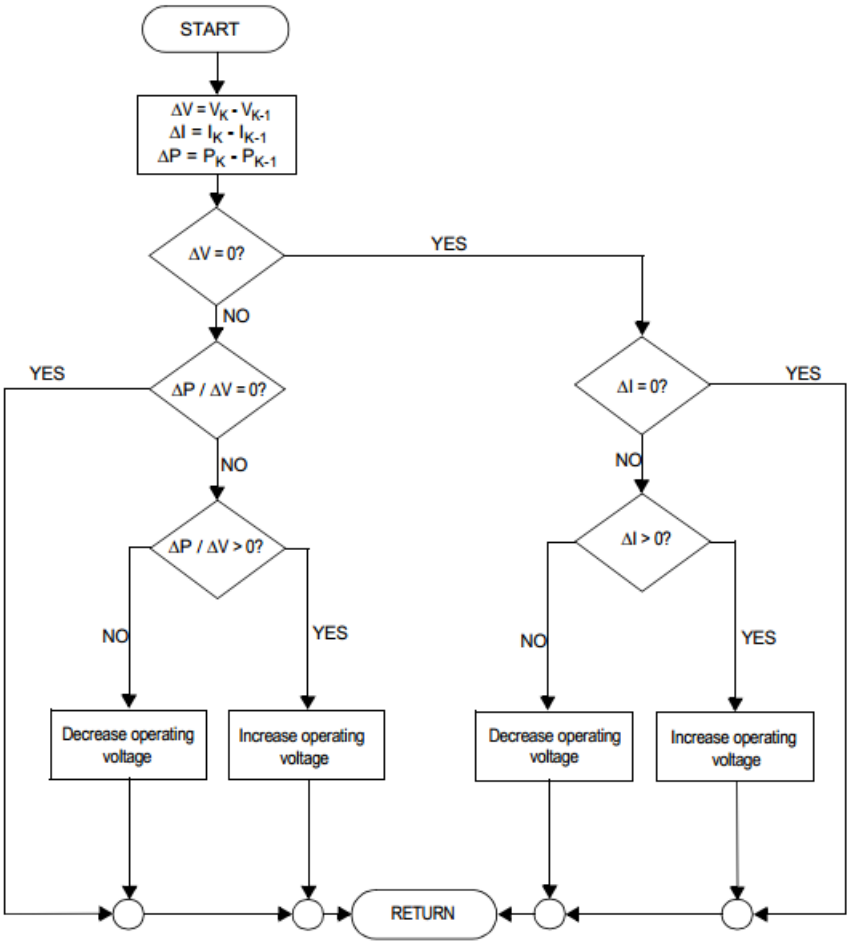


Figure 33. Incremental Conduction Algorithm Diagram [7]

6.3. Program Flow

6.3.1. Main

After the initial setup of the microcontroller the program enters the main loop in which it executes over and over the same functions. The microcontroller reads the sensors and process the acquired data. It calculates the battery voltage to determine the status (charged or discharged). With the sensors's data it is also capable of determining the appropriate PWM duty cycle for both solar and wind parts. Using the information received from the RPi and stored in variables it is able to set the connections (whether the PV array should be connected and if the batteries should be swapped). There are more things to the program that can be seen in *Part VI Code* such as value averaging and filtering, logic conditions to take decisions, etc.

The MPP algorithm resides in the function where the solar PWM duty cycle is set. The algorithm only makes sense when the battery is discharged. When the battery is charged, even if the MPP algorithm finds the optimum point of operation, setting a duty cycle higher than required would overload the battery and damage it. Therefore, when the battery is charged the microcontroller takes care of keeping it at a constant voltage of approximately 28.8V, bypassing the MPP algorithm.

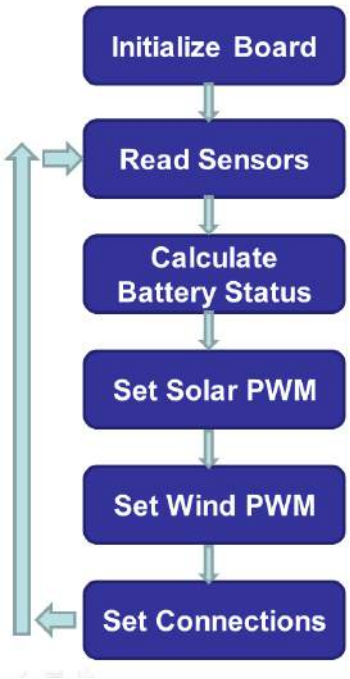


Figure 34. Main Program Synthesis on Atxmega128a4u

6.3.2. Interrupt Routine

The interrupt routine takes care of the communication with the RPi. It is an asynchronous link in the way that the clock of the RPi and the clock of the Atxmega128a4u are not shared and the Atxmega128a4u listens whenever the RPi sends information and stops to execute the current program to take care of that information. The microcontroller then uses the time and location information when it goes back to the main program loop.

The interrupt routine consists of receiving the time and location, sending the filtered sensor values to the RPi and sending the time and location to the solar tracker over XBee.

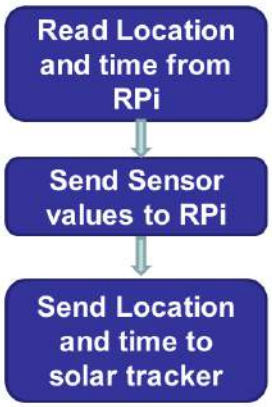


Figure 35. Interrupt Routine Synthesis on Atxmega128a4u

PART IV

RELAY BOARD

1. Design

THE relay board has two functions. The first one is to swap the energy flow to the batteries. If a battery is full and the other is empty and the power source of the former is producing more energy than the latter's then it will swap the batteries. The second function is to disconnect the PV array between the sunset and the sunrise to avoid any possible backward flow of energy.

1.1. Relay Type

Relays are easy to use. They require a simple driving circuit to operate. A typical configuration is made using a low side transistor that acts as a switch controlled by a signal (generated by a microcontroller). The diode in parallel with the coil creates a safe path of discharge when de-energizing it.

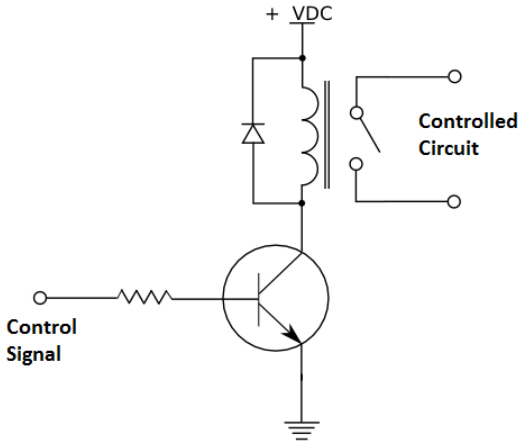


Figure 36. Simple Relay Control

This simple model can be improved in order to reduce power consumption. A simple relay constantly consumes energy when working in one of its two positions. This is not ideal when the objective is to create an efficient system. A solution to this problem is to use latching relays.

Latching relays only consume energy when switching because they mechanically snap to whichever position they are set. This of course leads to added complexity, more components, and increased cost but it is the option that was implemented in this project.

1.2. Control Signals

There are two main functions and therefore there are two groups of signals:

- The PV array connection signals. Connection and Disconnection signals.
- The battery swapping signals (shared by two relays). Connection and Reverse Connection signals.

1.3. Switching time

The relays take 10 ms to switch. In other words, for 10 ms the whole system has no power supply. The solution was to use big capacitors that are energized in parallel with the batteries but are placed in the circuit before the switching contacts of the relays. The capacitors are able to supply power to the system for around 100 ms before the voltage falls below 1.6V which is the lower working limit for the Atxmega.

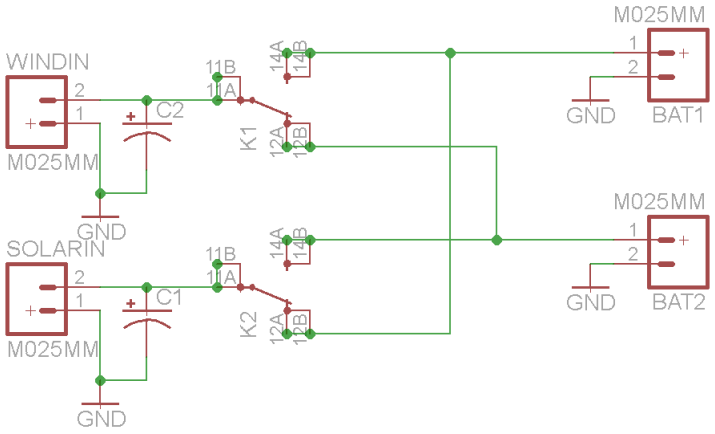


Figure 37. Capacitors on Relay Board

2. Schematics

THE schematics were produced using *Cadsoft Eagle* [10]. The size of the board was bigger than that allowed for the student version of the software, therefore the professional version had to be used.

- General purpose npn transistor used for every low side of a coil.
- 2K2 resistors used to drive the base of every transistor.
- 1n4004 diode used for every coil discharge.
- 4700μF capacitor on batterys rails.

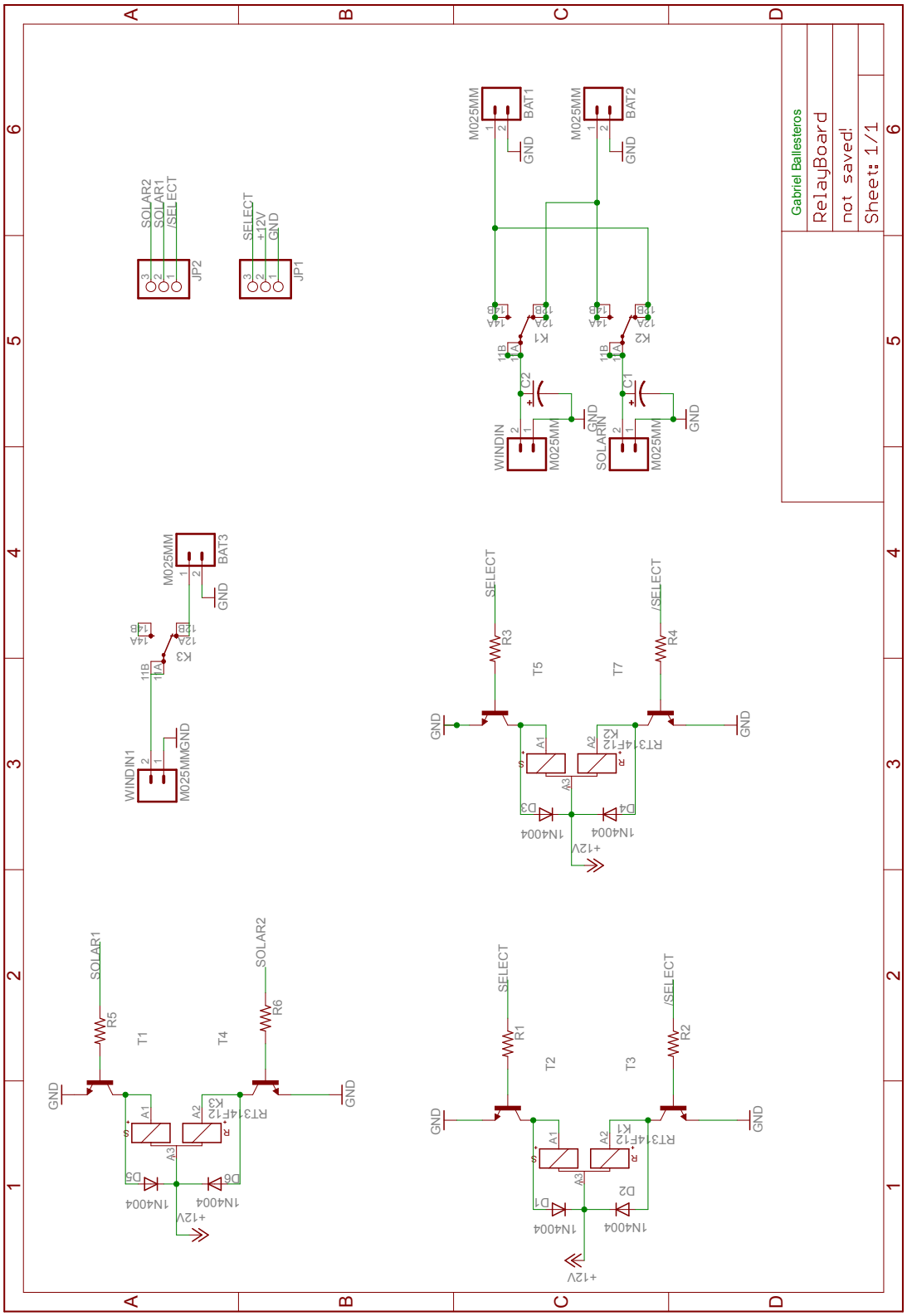


Figure 38. Relay Board

3. PCB

THE PCB of the controller board was designed using *Eagle* as well. It was manufactured in the service shop of the ECE department at the *Univeristy of Illinois*. The downside is that it does not have solder mask and the holes are not plated which is an added difficulty when soldering, the relays in particular.

3.1. Eagle Layout

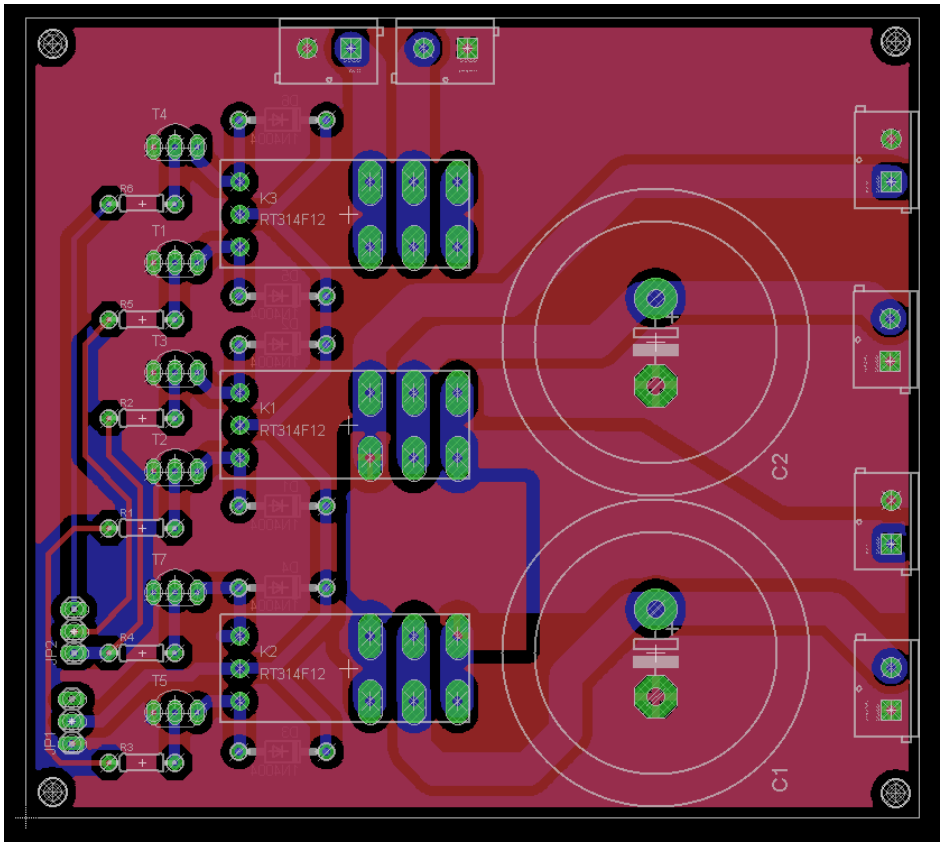


Figure 39. Relay PCB

This layout is the layout that should be used. Unfortunately some errors were present in the first version and instead of resubmitting the Gerber files to the service shop the board was modified cutting traces and resoldering components. Note that some traces are not equal in the picture in next section to those on the layout above, nonetheless the function of the board is of course the same.

3.2. Manufactured and Assembled PCB

Figure 40. Manufactured and Assembled Relay PCB

PART V

WEBSERVER

1. Webserver on Raspberry Pi

"

THE Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games." [9]

In our case we will use the RPi to host the webserver of the project. A microcontroller is able to serve websites to clients but it is slow and also it is not capable of serving a nice GUI. The RPi has a microprocessor instead of a microcontroller, in particular, built around a *700 MHz ARM1176JZF-S core*. It is not extremely fast but it is more than enough to serve decently the website the RPi will host and do some tasks, and all of that using a Linux distro.

The RPi has an ethernet port which is very convenient, it can be plugged to the modem/router of the user and the website containing all the data of the system can be accessed from any computer on the local network. Some more IT work and a static IP address would allow the RPi to be accessed from anywhere around the World.

The selected OS for the RPi is *Raspbian 'Wheezy'*, which is an optimized *Debian* distro for the RPi. The RPi being an open source platform, makes it fairly straight forward to get it working and learn how to implement and run programs on it. The OS is downloaded from *http://www.raspberrypi.org/downloads/* and installed on and SD card using *Win32DiskImager* if on Windows.

Figure 41. Raspbian

It is easy to make the RPi remotely accessible using SSH, which allows us to program it over ethernet after the SSH function has been enabled for the first time using a keyboard.

Another advantage of the RPi compared to a computer besides the cost is its extremely low power consumption of 1W to 2W during normal operation which makes it worth to monitor the system all day long and still produce a net positive flow of energy.

1.1. Webserver Solution Stack

After downloading and installing the OS on the SD card we are able to proceed to the installation of the webserver. Many possibilities have been examined before choosing which webserver base would be applied. The popular *LAMP* stack is somewhat demanding for the RPI, and it was decided that the *LIGHTTPD* [13] would be the best option.

As for the database, *MYSQL* is the most common choice but again, on the RPi it won't work as well so the database system that will be used is *SQLITE* [14] which is lighter on resources.

We will also need to implement *PHP*.

Figure 42. The Webserver Solution Stack : LightTPD, SQLite and php

1.2. Installation

Before installing the components mentioned in the previous part, it is recommended to "update" (synchronize the database of available software packages and the versions available) as well as "upgrade" (it will cause any packages with newer versions available to be updated).

```
sudo apt-get update
sudo apt-get upgrade
```

Code 2. Update and Upgrade Raspbian

Then we can install the solution stack:

```
sudo apt-get install lighttpd
sudo apt-get install php5-common php5-cgi php5
lighty-enable-mod fastcgi
lighty-enable-mod fastcgi-php
service lighttpd force-reload
sudo apt-get install sqlite3
sudo apt-get install php5-sqlite
```

Code 3. Installing Solution Stack

2. WordPress Website

WORDPRESS [11] is a free and open source content management system which runs on a web hosting service (the solution stack described previously, in this project).

Figure 43. WordPress

A nice feature of Wordpress is its GUI which allows us to modify the website without having to write lines of code. Website themes can be downloaded and installed in the Wordpress folder of the webserver. Also, some plugins (which are website enhancements) can be downloaded and activated graphically.

2.1. Installation

The latest wordpress file has to be downloaded and extracted. The owner of the folder can be changed.

```
cd /var/www
chown pi: .
rm *
wget http://wordpress.org/latest.tar.gz
tar xzf latest.tar.gz
mv wordpress/* .
rm -rf wordpress latest.tar.gz
```

Code 4. Download Wordpress

In the *wp-content* folder, the *wp-config.php* must be changed adding the following code after . This tells Wordpress that SQLite will be used.

```
define('DB_TYPE', 'sqlite');
```

Code 5. Change Configuration File

Another change has to be made in the same folder. Change the following lines of the *wp-install.php*:

```
%Change this line
$message = __('<strong><em>Note that password</em></strong> carefully! It is a
    <em>random</em> password that was generated just for you.');
%For this line
$message = __('<strong><em>Note that password</em></strong> carefully! It is a
    <em>random</em> password that was generated just for you.<br><br> ' .
    $random_password);
```

Code 6. Change Installation File

When all the previous steps are done, the Wordpress installer can be run from a web browser navigating to the Raspberry Pi's IP address with the */wp-admin/install.php* extension. In our particular case, this address happens to be *http://192.168.137.2/wp-admin/install.php*. After entering all the personal information the installer will proceed.

2.2. Website Design

The website can be configured after the login *http://192.168.137.2/wordpress/wp-login.php*. The theme that was used for the website is *Responsive* [12], i.e. it adapts the website to the screen resolution of the device that is browsing it, making it look nice on computers and smartphones.

The structure is the following:

- **Home Page**: This is the main page of the website. It is where the user starts when the address of the website is typed in a browser.
- **Solar Generation**: This page displays all the power output of the solar panels that flow to the battery bank. It displays the last hour, last day, last month, and last year's power production.
- **Wind Generation**: This page displays all the power output of the wind turbine that flow to the battery bank. It displays the last hour, last day, last month, and last year's power production.
- **Battery Status**: This page displays the voltage of the batteries that are on the bank with a range of one week.
- **Efficiency**: This page displays the efficiency of the system's energy flow from both the wind and solar sources to the batteries for the last year.
- **Set Location**: This page is where the user can set the location (latitude and longitude) where the system is installed. This is crucial in order for the solar tracker to work properly.
- **Advice**: This page displays some advice for the user such as : "check if panels are correctly connected", "the capacity of the battery bank could be increased", etc.

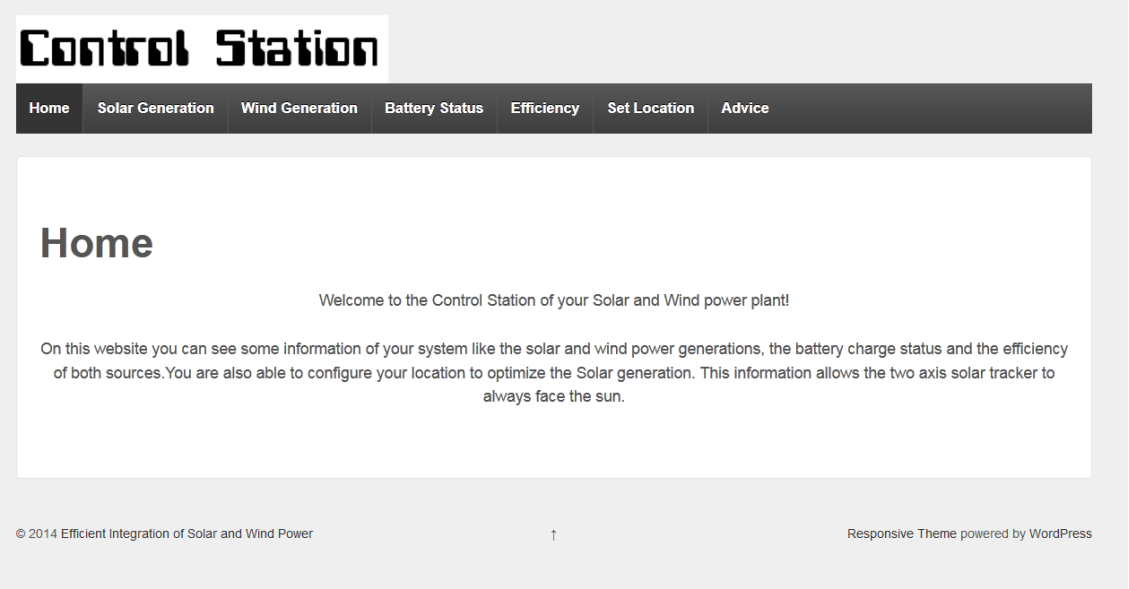


Figure 44. Home Page

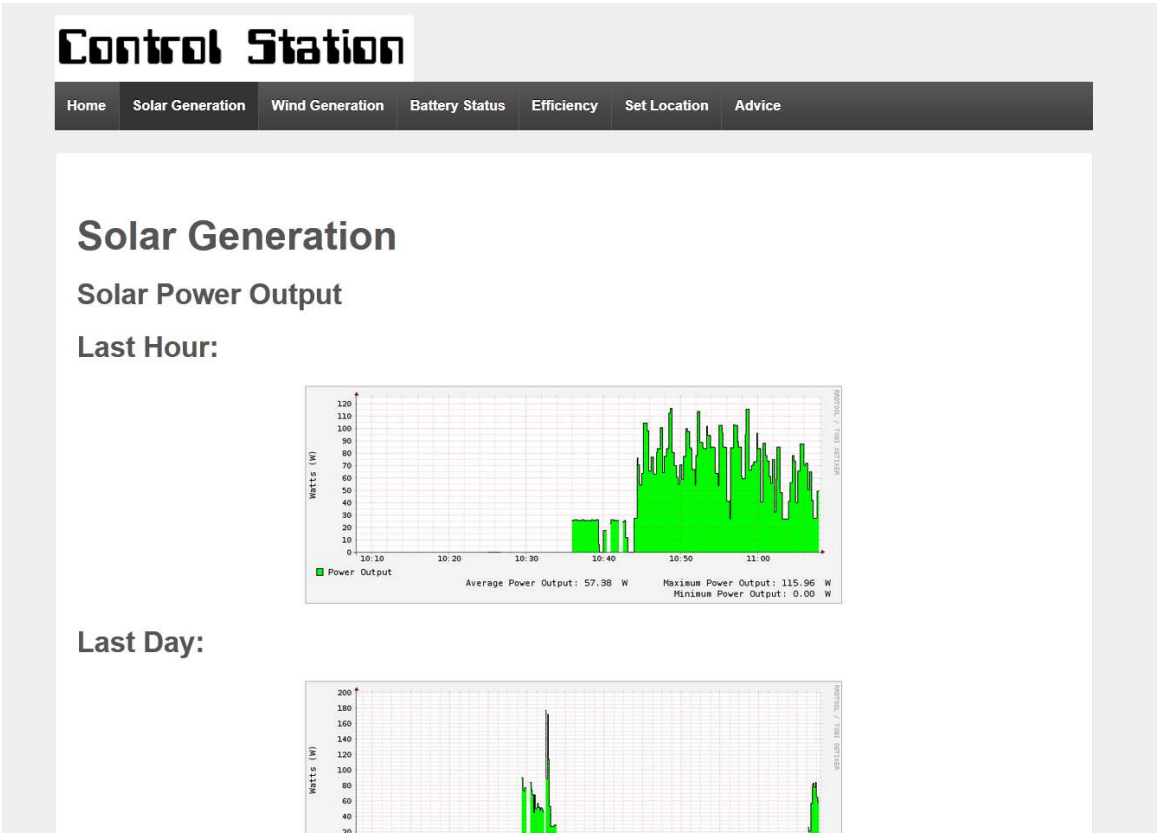


Figure 45. Solar Generation Page

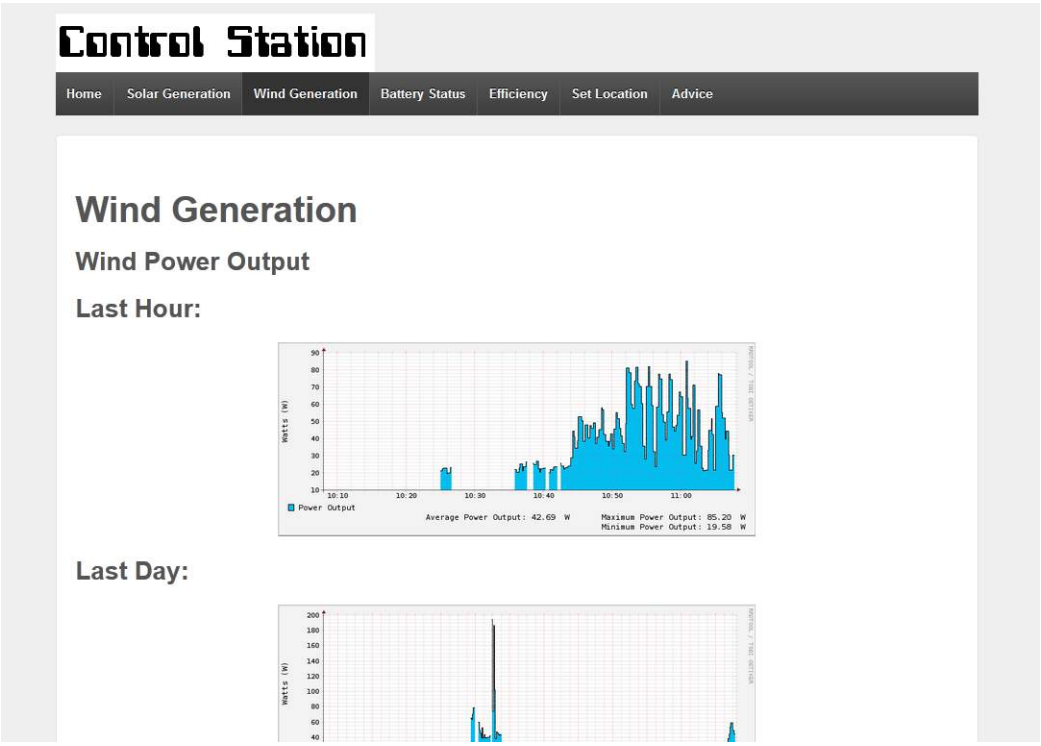


Figure 46. Wind Generation Page

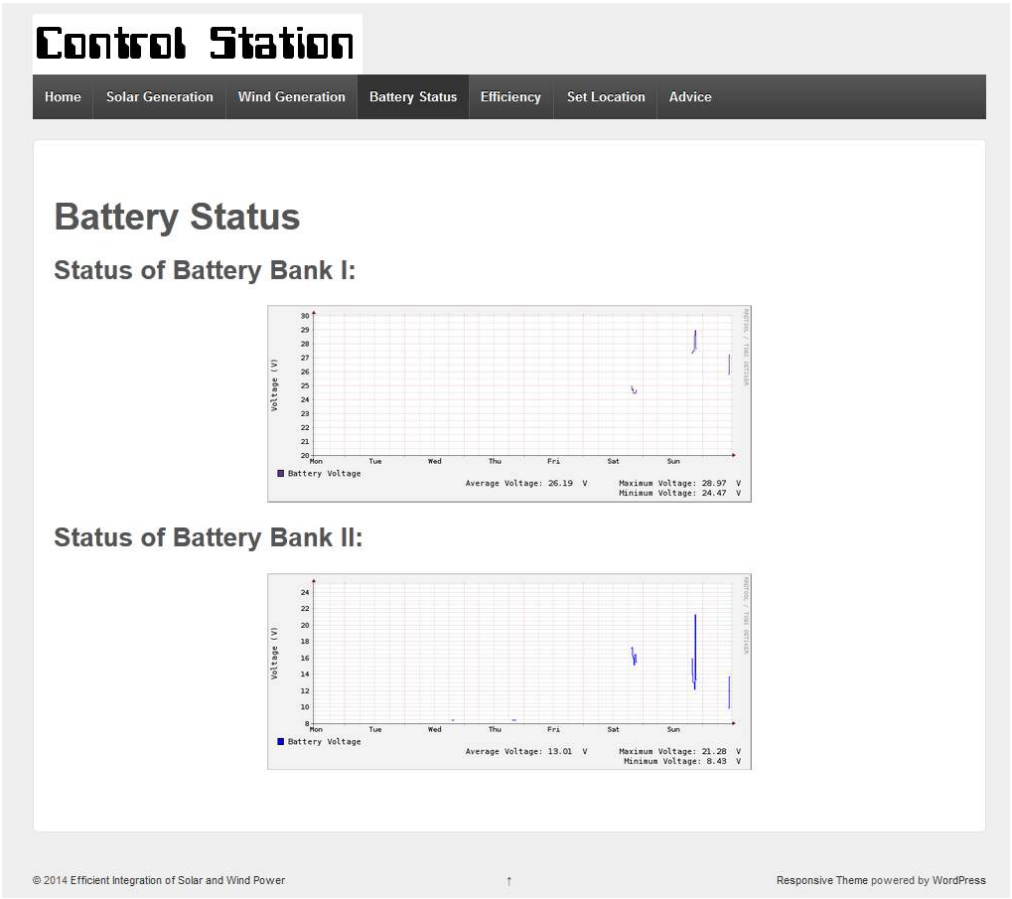


Figure 47. Battery Status Page

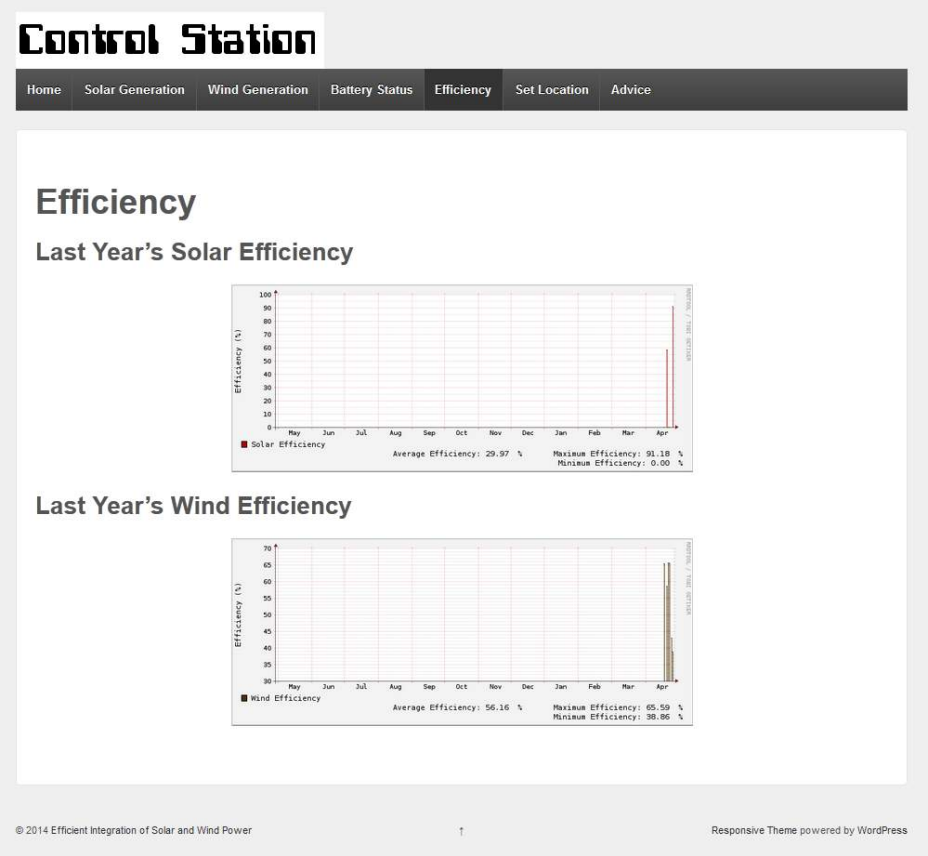


Figure 48. Efficiency Page

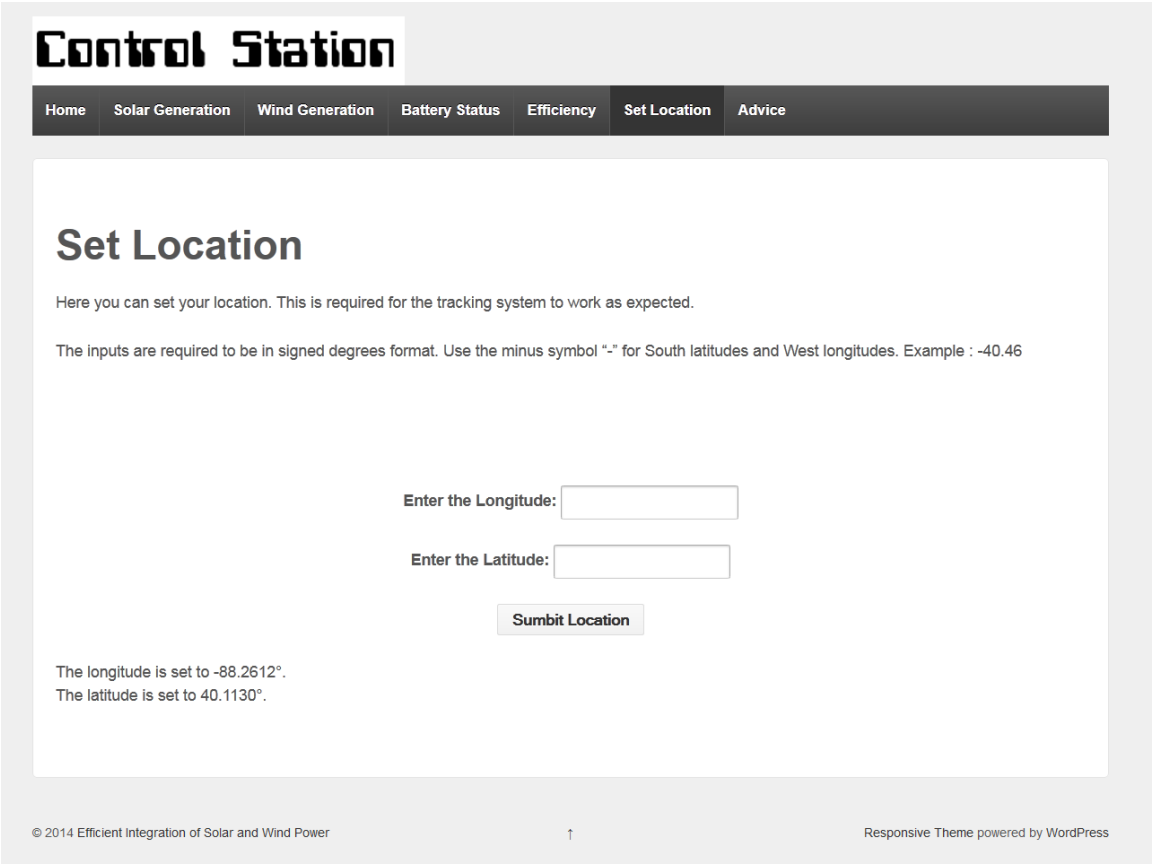


Figure 49. Location Page

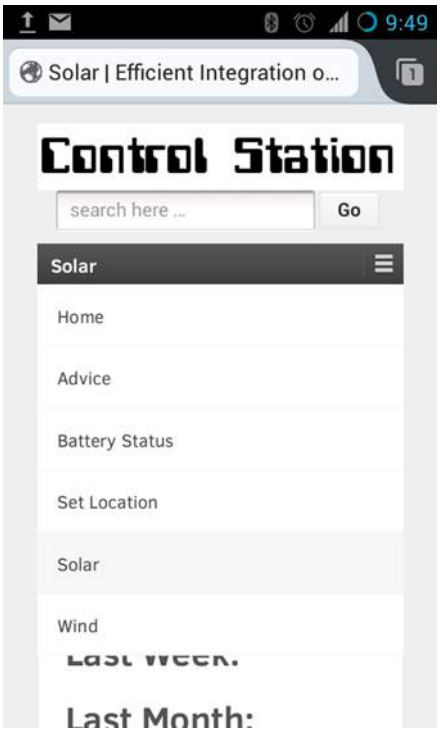


Figure 50. Website Appearance on Smartphone

2.2.1. Image Zoom

The best feature of the website is its ability to provide the system information graphically to the user. In order to improve the visibility of the graphs, a plugin called *Image Zoom* [15] was used. As it's name states, it enables the user to zoom in the graphs by just clicking on them.

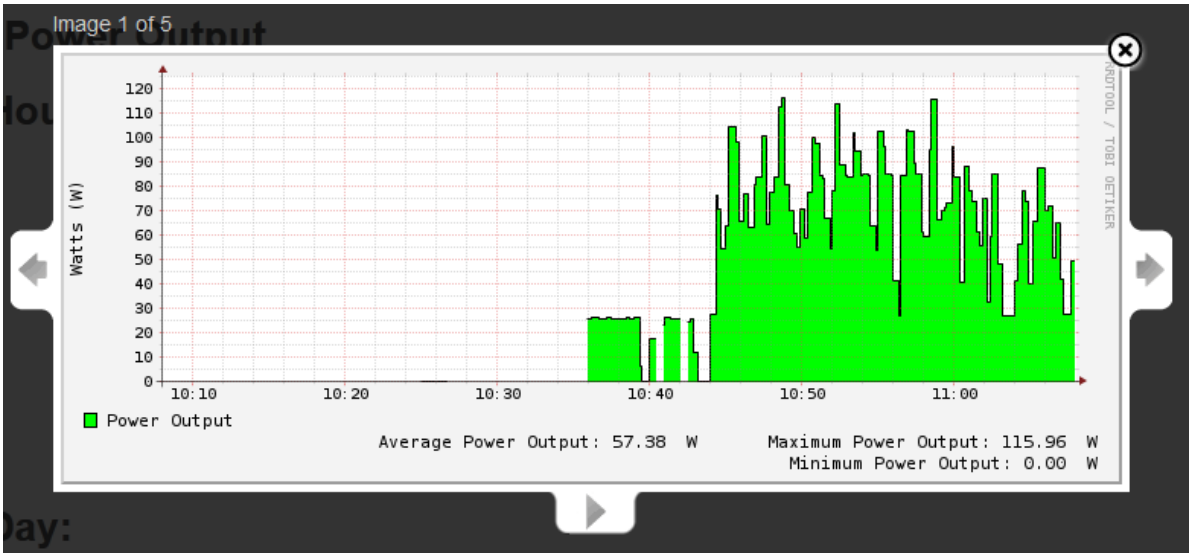


Figure 51. Image Zoom Plugin

2.2.2. PHP Script

A php script was written to take care of the update of the location. The user inputs the latitude and longitude in signed degrees and submits them. The webserver reads those values and confirms that they have been updated. At the same time, it saves them in a .txt file in its

database. This allows the whole system to run out of power (which should not happen) and when powered back, still know the location. The .txt file is read by a Pyhton script and the location information sent over the XBee link to the solar tracker.

```
Here you can set your location. This is required for the tracking system to work as
    expected.
The inputs are required to be in signed degrees format. Use the minus symbol "-" for South
    latitudes and West longitudes. Example : -40.46.

<div id="LocationForm" align="center"><form action="print-message.php"
    method="POST"><b>Enter the Longitude: </b> <input type="text" name="longitude"
    /><br><br><b>Enter the Latitude: </b> <input type="text" name="latitude" /><br><br>
<input type="submit" value="Sumbit Location" /></form></div>

[insert_php]
$file = "/var/www/wordpress/wp-content/uploads/2014/txt/location.txt";
$document = file_get_contents($file);
$lines = explode("\n", $document);

echo 'The longitude is set to ' . $lines[0]. '°.
';
echo 'The latitude is set to ' . $lines[1]. '°.
';
[/insert_php]
```

Code 7. Location Page Script

```
?php
$file = "/var/www/wordpress/wp-content/uploads/2014/txt/location.txt";
$longitude = $_POST['longitude'];
$latitude = $_POST['latitude'];

if (is_numeric($longitude) && is_numeric($latitude))
{
        if(((-180 <= $longitude) && ($longitude <= 180)) && ((-80 <= $latitude) &&
            ($latitude <= 80)))
        {
                $document = file_put_contents($file, $longitude);
                $document = file_put_contents($file, "\n", FILE_APPEND);
                $document = file_put_contents($file, $latitude, FILE_APPEND);

                echo 'The longitude is now set to ' . $longitude . ' degrees. <br>';
                echo 'The latitude is now set to ' . $latitude . ' degrees.<br><br>';
        }

        else    echo 'Please enter valid longitude and latitude values.<br>';
}
else    echo 'Please enter numbers.<br>';
?>

 <form><input type="button" value="Return to website"
     onClick="window.location.href='//192.168.137.2/wordpress/?page_id=9'"></form>
```

Code 8. Confirm Location Script

3. Round Robin Database

An interesting database system to collect data that is a time series is the RRD. It is cyclic and therefore never runs out of space. Its size is defined when it is created. When it runs out of space it goes back to the beginning and overwrites the oldest data. This is a key feature for the RPi which is using a 4 GB SD card.

Tobias Oetiker has created the *RRDtool* [3] which is an open source tool based on the RRD that has many features. The most important which are used in this project are the ability to extract the maximum, minimum and average of a time series as well as to plot nice graphs which are customizable.

The RRDtool has to be installed in the RPi first as well as the python module which is also required because the main script is written in python:

```
sudo apt-get install rrdtool
sudo apt-get install python-rrd
```

Code 9. Installing RRDtool

3.1. RRD Creation

Two databases are created, one for the solar part of the project and the other for the wind part of the project. Both collect the same amount of data with the same frequency. All parameters are adjusted following the guidelines of the author of the tool.

The step for both databases (period of data reception) is 10s. Which means that the RRD will store a day ($\frac{10*8640}{3600}$ hours), a week, a month and a year series.

The database will store the minimum, maximum and average of each of the following values:

- Voltage Output : Corresponds to the battery voltage
- Power Input : Input current times output current read by the microcontroller
- Power Output : Output current times output current read by the microcontroller
- Efficiency : Ratio of Power Output over Power Input in percentage.

```
#!/usr/local/bin/bash

rrdtool create Solar.rrd \
--start N --step 10 \
DS:SolarVoltageOut:GAUGE:30:0:40 \
DS:SolarPowerIn:GAUGE:30:0:400 \
DS:SolarPowerOut:GAUGE:30:0:400 \
DS:SolarEfficiency:GAUGE:30:0:100 \
RRA:MIN:0.5:1:8640 \
RRA:MIN:0.5:2:60480 \
RRA:MIN:0.5:4:267840 \
RRA:MIN:0.5:10:3153600 \
RRA:MAX:0.5:1:8640 \
RRA:MAX:0.5:2:60480 \
RRA:MAX:0.5:4:267840 \
RRA:MAX:0.5:10:3153600 \
RRA:AVERAGE:0.5:1:8640 \
RRA:AVERAGE:0.5:2:60480 \
RRA:AVERAGE:0.5:4:267840 \
RRA:AVERAGE:0.5:10:3153600

rrdtool create Wind.rrd \
--start N --step 10 \
DS:WindVoltageOut:GAUGE:30:0:40 \
DS:WindPowerIn:GAUGE:30:0:2000 \
DS:WindPowerOut:GAUGE:30:0:2000 \
DS:WindEfficiency:GAUGE:30:0:100 \
RRA:MIN:0.5:1:8640 \
RRA:MIN:0.5:2:60480 \
RRA:MIN:0.5:4:267840 \
RRA:MIN:0.5:10:3153600 \
RRA:MAX:0.5:1:8640 \
RRA:MAX:0.5:2:60480 \
RRA:MAX:0.5:4:267840 \
RRA:MAX:0.5:10:3153600 \
RRA:AVERAGE:0.5:1:8640 \
RRA:AVERAGE:0.5:2:60480 \
RRA:AVERAGE:0.5:4:267840 \
RRA:AVERAGE:0.5:10:3153600
```

Code 10. Creating RRD with RRDtool

4. Program Flow

THE RPi executes a **Python** script continuously to make the exchange of information with the controller board possible. The RPi sets the pace and the controller board replies using interrupts in order not to loose information.

The script starts the serial link and then checks the *location.txt* file to see if it has been updated. It computes the sunset and sunrise times to give the order to disconnect the PV array from the system (if the current time is between sunset and sunrise). The RPi then sends the time and date as well as the location over serial and when the controller board reads them it replies back with the sensors values. These values (12 bit ADC values) are then transformed in the RPi to be human readable. Power values as well as efficiencies are computed and the relevant data is stored in the database.

Graphs are generated with the data stored in the RRD. They display the time series of the particular data but also its minimum, maximum and average.

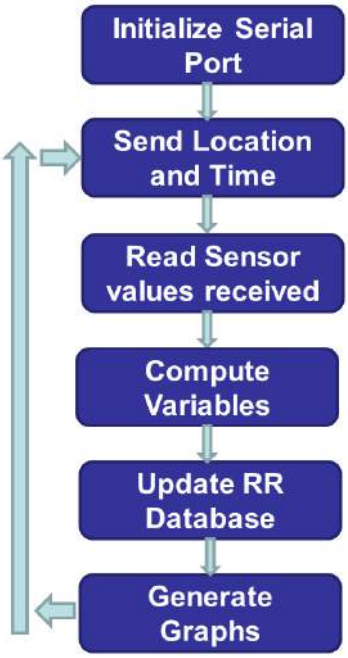


Figure 52. Program Synthesis on the Raspberry Pi

PART VI

CODE

1. Controller Board Code

THE code for the Atxmega128a4u microcontroller has been written in **C** using *Atmel Studio 6* [16]. It has many parts that are linked together using *ASF*. The advantage of using Atmel´s software is that many drivers and examples are available.

1.1. Configuration

1.1.1. Board GPIOs

```
#ifndef CONF_BOARD_H
#define CONF_BOARD_H

//LED pins
#define LED0      IOPORT_CREATE_PIN(PORTR, 0)
#define LED1      IOPORT_CREATE_PIN(PORTR, 1)

//Mosfet driver pins
#define LS        IOPORT_CREATE_PIN(PORTE, 2)

//Serial comm pins
#define TX0       IOPORT_CREATE_PIN(PORTC, 3)
#define RX0       IOPORT_CREATE_PIN(PORTC, 2)
#define TX1       IOPORT_CREATE_PIN(PORTC, 7)
#define RX1       IOPORT_CREATE_PIN(PORTC, 6)
#define TX2       IOPORT_CREATE_PIN(PORTD, 3)
#define RX2       IOPORT_CREATE_PIN(PORTD, 2)
#define XBEESLEEP IOPORT_CREATE_PIN(PORTD, 5)

//Relay pins
#define PVRELAYON    IOPORT_CREATE_PIN(PORTB, 0)
#define PVRELAYOFF   IOPORT_CREATE_PIN(PORTB, 1)
#define BRELAYSB1    IOPORT_CREATE_PIN(PORTC, 4)
#define BRELAYSB2    IOPORT_CREATE_PIN(PORTC, 5)

//Button pin
#define BUTTON    IOPORT_CREATE_PIN(PORTD, 4)

//Solar ADC
#define ADC   ADCA

#endif // CONF_BOARD_H
```

1.1.2. USART

```
#ifndef CONF_USART_SERIAL_H_INCLUDED
#define CONF_USART_SERIAL_H_INCLUDED

//Xbee USART
#define USART_SERIAL_0                    &USARTC0
#define USART_SERIAL_0_BAUDRATE           9600
#define USART_SERIAL_0_CHAR_LENGTH        USART_CHSIZE_8BIT_gc
#define USART_SERIAL_0_PARITY             USART_PMODE_DISABLED_gc
#define USART_SERIAL_0_STOP_BIT           false

//Bluetooth Module USART
#define USART_SERIAL_1                    &USARTC1
#define USART_SERIAL_1_BAUDRATE           9600
#define USART_SERIAL_1_CHAR_LENGTH        USART_CHSIZE_8BIT_gc
```

```
#define USART_SERIAL_1_PARITY              USART_PMODE_DISABLED_gc
#define USART_SERIAL_1_STOP_BIT            false

//Raspberry Pi USART
#define USART_SERIAL_2                     &USARTD0
#define USART_SERIAL_2_BAUDRATE            9600
#define USART_SERIAL_2_CHAR_LENGTH         USART_CHSIZE_8BIT_gc
#define USART_SERIAL_2_PARITY              USART_PMODE_DISABLED_gc
#define USART_SERIAL_2_STOP_BIT            false

#endif /* CONF_USART_SERIAL_H_INCLUDED */
```

1.1.3. ADC

```
#ifndef CONF_ADC_H
#define CONF_ADC_H

/* Refer to the ADC driver for detailed documentation. */
#define CONFIG_ADC_CALLBACK_ENABLE
#define CONFIG_ADC_CALLBACK_TYPE uint16_t

#define CONFIG_ADC_INTLVL        ADC_CH_INTLVL_HI_gc

#endif /* CONF_ADC_H */
```

1.2. Initialization

```
#include <asf.h>
#include <board.h>
#include <conf_board.h>
#include <avr/interrupt.h>
#include <pmic.h>

static uint8_t current_scan_channel = 0;

int sin_voltage;
int sin_current;
int sout_voltage;
int sout_current;
int win_voltage;
int win_current;
int wout_voltage;
int wout_current;

//Xbee USART
static usart_serial_options_t USART_SERIAL_0_OPTIONS = {
    .baudrate = USART_SERIAL_0_BAUDRATE,
    .charlength = USART_SERIAL_0_CHAR_LENGTH,
    .paritytype = USART_SERIAL_0_PARITY,
    .stopbits = USART_SERIAL_0_STOP_BIT
};

//Bluetooth Module USART
static usart_serial_options_t USART_SERIAL_1_OPTIONS = {
    .baudrate = USART_SERIAL_1_BAUDRATE,
    .charlength = USART_SERIAL_1_CHAR_LENGTH,
    .paritytype = USART_SERIAL_1_PARITY,
    .stopbits = USART_SERIAL_1_STOP_BIT
};

//Raspberry Pi USART
static usart_serial_options_t USART_SERIAL_2_OPTIONS = {
    .baudrate = USART_SERIAL_2_BAUDRATE,
    .charlength = USART_SERIAL_2_CHAR_LENGTH,
    .paritytype = USART_SERIAL_2_PARITY,
    .stopbits = USART_SERIAL_2_STOP_BIT
};

//ADC callback function
static void adc_handler(ADC_t *adc, uint8_t ch_mask, adc_result_t result)
{
```

```
    // Store the ADC results from the scan in the variables
    if (ch_mask & ADC_CH0)
    {
        switch(current_scan_channel)
        {
            case 0:
                sin_voltage = result;
                break;
            case 1:
                sin_current = result;
                break;
            case 2:
                sout_current = result;
                break;
            case 3:
                sout_voltage = result;
                break;
            case 4:
                win_voltage = result;
                break;
            case 5:
                win_current = result;
                break;
            case 6:
                wout_voltage = result;
                break;
            case 7:
                wout_current = result;
                break;
            default:
                return;
        }

        current_scan_channel++;

        // When 8 pins have been scanned the SCAN OFFSET wraps to zero
        if (current_scan_channel == 8) current_scan_channel = 0;
    }
}

static void adc_init(void)
{
    struct adc_config adc_conf;
    struct adc_channel_config adcch_conf;

    ADCA.CALL = nvm_read_production_signature_row(ADCACAL0);
    ADCA.CALH = nvm_read_production_signature_row(ADCACAL1);

    adc_read_configuration(&ADC, &adc_conf);
    adcch_read_configuration(&ADC, ADC_CH0, &adcch_conf);

    adc_set_conversion_parameters(&adc_conf, ADC_SIGN_OFF, ADC_RES_12, ADC_REF_BANDGAP);
    adc_set_clock_rate(&adc_conf, 5000UL);
    adc_set_conversion_trigger(&adc_conf, ADC_TRIG_EVENT_SWEEP, 1, 0);
    adc_set_current_limit(&adc_conf, ADC_CURRENT_LIMIT_HIGH);
    adc_set_gain_impedance_mode(&adc_conf, ADC_GAIN_HIGHIMPEDANCE);
    adc_enable_internal_input(&adc_conf, ADC_INT_BANDGAP);

    adc_write_configuration(&ADC, &adc_conf);
    adc_set_callback(&ADC, &adc_handler);

    adcch_set_input(&adcch_conf, ADCCH_POS_PIN0, ADCCH_NEG_NONE, 1);
    adcch_set_interrupt_mode(&adcch_conf, ADCCH_MODE_COMPLETE);
    adcch_enable_interrupt(&adcch_conf);
    adcch_set_pin_scan(&adcch_conf, 0, 7);

    adcch_write_configuration(&ADCA, ADC_CH0, &adcch_conf);

    // Enable the Event System to use as trigger source
    sysclk_enable_module(SYSCLK_PORT_GEN, SYSCLK_EVSYS);
    // Conversion 62500 times a second
    EVSYS.CH0MUX = EVSYS_CHMUX_PRESCALER_512_gc;
}
```

```
void board_init(void)
{
    /* This function is meant to contain board-specific initialization code
     * for, e.g., the I/O pins. The initialization can rely on application-
     * specific board configuration, found in conf_board.h.
     */

    //Disable global interrupts
    cpu_irq_disable();

    //Configure and Calibrate 32MHz clock
    OSC.DFLLCTRL &= ~(OSC_RC32MCREF_gm);
    OSC.DFLLCTRL |= OSC_RC32MCREF_RC32K_gc;
    OSC.CTRL |= OSC_RC32MEN_bm|OSC_RC32KEN_bm;
    while(!(OSC.STATUS & OSC_RC32MRDY_bm));
    while(!(OSC.STATUS & OSC_RC32KRDY_bm));
    DFLLRC32M.CTRL |= DFLL_ENABLE_bm;
    CCP = CCP_IOREG_gc;
    CLK.PSCTRL = 0;
    sysclk_init();

    //Configure GPIO Pins
    ioport_init();
    ioport_set_pin_dir(LED0, IOPORT_DIR_OUTPUT);
    ioport_set_pin_dir(LED1, IOPORT_DIR_OUTPUT);

    ioport_set_pin_dir(LS, IOPORT_DIR_OUTPUT);

    ioport_set_pin_dir(PVRELAYON, IOPORT_DIR_OUTPUT);
    ioport_set_pin_dir(PVRELAYOFF, IOPORT_DIR_OUTPUT);
    ioport_set_pin_dir(BRELAYSB1, IOPORT_DIR_OUTPUT);
    ioport_set_pin_dir(BRELAYSB2, IOPORT_DIR_OUTPUT);

    ioport_set_pin_dir(BUTTON, IOPORT_DIR_INPUT);
    ioport_set_pin_mode(BUTTON, IOPORT_MODE_PULLUP);


    //Configure USART
    ioport_set_pin_dir(TX0, IOPORT_DIR_OUTPUT);
    ioport_set_pin_level(TX0, HIGH);
    ioport_set_pin_dir(RX0, IOPORT_DIR_INPUT);

    ioport_set_pin_dir(TX1, IOPORT_DIR_OUTPUT);
    ioport_set_pin_level(TX1, HIGH);
    ioport_set_pin_dir(RX1, IOPORT_DIR_INPUT);

    ioport_set_pin_dir(TX2, IOPORT_DIR_OUTPUT);
    ioport_set_pin_level(TX2, HIGH);
    ioport_set_pin_dir(RX2, IOPORT_DIR_INPUT);

    stdio_serial_init(USART_SERIAL_0, &USART_SERIAL_0_OPTIONS);
    stdio_serial_init(USART_SERIAL_1, &USART_SERIAL_1_OPTIONS);
    stdio_serial_init(USART_SERIAL_2, &USART_SERIAL_2_OPTIONS);

    usart_set_rx_interrupt_level(USART_SERIAL_2, USART_INT_LVL_LO);

    //Confiure ADC
    PR.PRPA = 0x05; // POWER ON ADC
    adc_init();
    adc_enable(&ADC);

    //Enable global interrupts
    pmic_init();
    cpu_irq_enable();
}
```

1.3. Main and Interrupt Routine

```
#include <asf.h>
#include <conf_usart_serial.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <pmic.h>

//counters
int i=0;
int j=0;
int k=0;

//PWM variables
int solar_duty = 80;
int wind_duty = 30;
float solar_duty_charged = 90;
float wind_duty_charged;
int solar_duty_max;
int wind_duty_max;

//time variables
volatile int day;
volatile int month;
volatile int year;
volatile int hours;
volatile int minutes;
volatile int seconds;
volatile int night = 0;

//location variables
volatile int longitude;
volatile int latitude;

//Power variables
int sin_voltage;
int sin_current;
int sout_voltage;
int sout_current;
int sout_voltage_now = 0;
int sout_voltage_before = 0;
int sout_current_now = 0;
int sout_current_before = 0;
int inc_voltage = 0;
int inc_current = 0;
int inc_power = 0;
int spout_now = 0;
int spout_before = 0;
uint32_t spout_average = 0;
int sout_current_max = 4050;
int sout_voltage_min = 2000;
int sout_voltage_max = 3900;

int win_voltage;
int win_current;
int wout_voltage;
int wout_current;
int wout_current_max = 4050;
int wout_voltage_min = 2000;
int wout_voltage_max = 3900;
uint32_t wpout_average = 0;

int svoltage_charged = 3900;
int wvoltage_charged = 4000;

float sin_voltage_normalized;
float sout_voltage_normalized;
int sin_voltage_normalizedi;
int sout_voltage_normalizedi;

//Averaging variables
int sin_voltage_sum = 0;
int sin_current_sum = 0;
```

```
int sout_current_sum = 0;
int sout_voltage_sum = 0;
int win_voltage_sum = 0;
int win_current_sum = 0;
int wout_current_sum = 0;
int wout_voltage_sum = 0;

int sin_voltage_average = 0;
int sin_current_average = 0;
int sout_current_average = 0;
int sout_voltage_average = 0;
int win_voltage_average = 0;
int win_current_average = 0;
int wout_current_average = 0;
int wout_voltage_average = 0;


//Relay variables
bool PV_status = 1; //connected or disconnected
bool last_battery = 0;

//Battery variables
bool sbattery_status = 0;
bool wbattery_status = 0;

//PWMs
struct pwm_config pwm_cfg_solar;
struct pwm_config pwm_cfg_wind;

//Serial comms
static int uart_putchar(char c, FILE *stream);
static int uart_getchar(FILE *stream);


struct uart_port {
    volatile uint8_t *udr;
    volatile uint8_t *ucsra;
};

static FILE usart0 = FDEV_SETUP_STREAM (uart_putchar, uart_getchar, _FDEV_SETUP_RW);
static FILE usart1 = FDEV_SETUP_STREAM (uart_putchar, uart_getchar, _FDEV_SETUP_RW);
static FILE usart2 = FDEV_SETUP_STREAM (uart_putchar, uart_getchar, _FDEV_SETUP_RW);

struct uart_port uart_port0 = {
    .udr = &USARTC0_DATA,
    .ucsra = &USARTC0_STATUS
};

struct uart_port uart_port1 = {
    .udr = &USARTC1_DATA,
    .ucsra = &USARTC1_STATUS
};

struct uart_port uart_port2 = {
    .udr = &USARTD0_DATA,
    .ucsra = &USARTD0_STATUS
};

//functions
void set_solar_PWM(void);
void set_wind_PWM(void);
void get_battery_status(void);
void track_MPPT(void);
void PV_connection(void);
void swap_batteries(void);
void read_inputs(void);

int main (void)
{
     board_init();

     fdev_set_udata(&usart0, &uart_port0);
     fdev_set_udata(&usart1, &uart_port1);
     fdev_set_udata(&usart2, &uart_port2);
```

```
     ioport_set_pin_level(PVRELAYON, HIGH);
     delay_ms(200);
     PV_status = 1;
     ioport_set_pin_level(PVRELAYON, LOW);


     ioport_set_pin_level(BRELAYSB1, HIGH);
     last_battery = 1;
     delay_ms(200);
     ioport_set_pin_level(BRELAYSB1, LOW);

     ioport_set_pin_level(LED0, HIGH);
     ioport_set_pin_level(LED1, HIGH);
     ioport_set_pin_level(XBEESLEEP, LOW);

     //Configure PWM
     ioport_set_pin_level(LS, HIGH);
     pwm_init(&pwm_cfg_solar, PWM_TCE0, PWM_CH_D, 125000); //PWM on PE3
     pwm_init(&pwm_cfg_wind, PWM_TCC0, PWM_CH_B, 125000);  //PWM on PC1


    while(1)
    {
        read_inputs();
        get_battery_status();
        set_solar_PWM();
        //Waiting Wind Student to complete her part...
        //set_wind_PWM();
        PV_connection();

        fprintf(&usart1, "%d %d %d %d %d %d %d %d\r\n",
        sin_voltage_average, sin_current_average, sout_voltage_average, sout_current_average,
        win_voltage_average, win_current_average, wout_voltage_average, wout_current_average);

        fprintf(&usart1, "%d %d %d %d\r\n", solar_duty, wind_duty, sbattery_status,
            PV_status);
    }

}

void track_MPPT()
{
    //actual values of current, voltage and power
    sout_current_now = sout_current_average;
    sout_voltage_now = sout_voltage_average;
    spout_now = (uint32_t) sout_current_now * sout_voltage_now;

    //compute increments
    inc_current = sout_current_now - sout_current_before;
    inc_voltage = sout_voltage_now - sout_voltage_before;
    inc_power = spout_now - spout_before;

    //go towards maximum power and stay when found
    if(inc_voltage == 0)
    {
        if(inc_current != 0)
        {
            if(inc_current > 0 && sout_current_now < sout_current_max) solar_duty++;
            else solar_duty--;
        }
    }

    else
    {
        if((inc_power/inc_voltage) != 0)
        {
            if((inc_power/inc_voltage) > 0 && sout_current_now < sout_current_max)
                solar_duty++;
            else solar_duty--;
        }
    }

    sout_current_before = sout_current_now;
    sout_voltage_before = sout_voltage_now;
    spout_before = spout_now;
```

```
}

void set_solar_PWM()
{
    solar_duty_charged = (float) (100 * 28.8) / (0.01396*sin_voltage_average - 3.287);
    if(solar_duty_charged > 96) solar_duty_charged = 96;
    if(solar_duty_charged < 80) solar_duty_charged = 80;

    switch(sbattery_status)
    {
        case 0:
            track_MPPT();
            break;
        case 1:
            solar_duty = (int) solar_duty_charged;
            //if((wbattery_status == 0) && (spout_average > wpout_average)) swap_batteries();
            break;
        default:
            return;
    }

    if(solar_duty > 96) solar_duty = 96;
    if(solar_duty < 80) solar_duty = 80;

    pwm_start(&pwm_cfg_solar, solar_duty);
    //pwm_start(&pwm_cfg_solar, 80);
}

void set_wind_PWM()
{
    //wind_duty_charged = (float) (100 * 28.2) / (0.013608*win_voltage_average - 2.7863);
    if(wind_duty_charged > 55) wind_duty_charged = 55;
    if(wind_duty_charged < 8) wind_duty_charged = 8;

    switch(wbattery_status)
    {
        case 0:
            if((wout_current <= wout_current_max) && (wind_duty < 92)) wind_duty++;
            else wind_duty--;
            break;
        case 1:
            wind_duty = (int) wind_duty_charged;
            if((sbattery_status == 0) && (wpout_average > spout_average)) swap_batteries();
            break;
        default:
            return;
    }

    if(wind_duty > 55) wind_duty = 55;
    if(wind_duty < 8) wind_duty = 8;

    pwm_start(&pwm_cfg_wind, wind_duty);
}

void get_battery_status()
{
    //Battery I
    if(sout_voltage_average < svoltage_charged)
    {
        sbattery_status = 0; // discharged, needs to be charged
        ioport_set_pin_level(LED0, HIGH);
    }

    else if(sout_voltage_average >= svoltage_charged)
    {
        sbattery_status = 1; // partially charged
        ioport_set_pin_level(LED0, LOW);
    }


    //Battery II
    if(wout_voltage_average < wvoltage_charged)
    {
        wbattery_status = 0; // discharged, needs to be charged
        ioport_set_pin_level(LED1, HIGH);
```

```
    }

    else if(wout_voltage_average >= wvoltage_charged)
    {
        wbattery_status = 1; // partially charged
        ioport_set_pin_level(LED1, LOW);
    }
}

void PV_connection()
{
    if(ioport_get_pin_level(BUTTON) == 0)
        {
            night = !night;
            delay_ms(300);
        }

    if(night && PV_status)
    {
        ioport_set_pin_level(PVRELAYOFF, HIGH);
        PV_status = 0;
        delay_ms(200);
        ioport_set_pin_level(PVRELAYON, LOW);
    }

    else if (!night && !PV_status)
    {
        ioport_set_pin_level(PVRELAYON, HIGH);
        PV_status = 1;
        delay_ms(200);
    }

    ioport_set_pin_level(PVRELAYON, LOW);
    ioport_set_pin_level(PVRELAYOFF, LOW);
}

void swap_batteries()
{
    if(last_battery == 0)
    {
        ioport_set_pin_level(BRELAYSB1, HIGH);
        last_battery = 1;
    }

    else
    {
        ioport_set_pin_level(BRELAYSB2, HIGH);
        last_battery = 0;
    }

    delay_ms(200);
    ioport_set_pin_level(BRELAYSB1, LOW);
    ioport_set_pin_level(BRELAYSB2, LOW);

}

void read_inputs()
{
    sin_voltage_sum += sin_voltage;
    sin_current_sum += sin_current;
    sout_current_sum += sout_current;
    sout_voltage_sum += sout_voltage;

    win_voltage_sum += win_voltage;
    win_current_sum += win_current;
    wout_current_sum += wout_current;
    wout_voltage_sum += wout_voltage;

    i++;

    if(i == 7)
    {
        sin_voltage_average = sin_voltage_sum / 7;
        sin_current_average = sin_current_sum / 7;
        sout_current_average = sout_current_sum / 7;
```

```
        sout_voltage_average = sout_voltage_sum / 7;

        win_voltage_average = win_voltage_sum / 7;
        win_current_average = win_current_sum / 7;
        wout_current_average = wout_current_sum / 7;
        wout_voltage_average = wout_voltage_sum / 7;

        spout_average = (uint32_t) sout_current_average * sout_voltage_average;
        wpout_average = (uint32_t) wout_current_average * wout_voltage_average;

        i = 0;
        sin_voltage_sum = 0;
        sin_current_sum = 0;
        sout_current_sum = 0;
        sout_voltage_sum = 0;

        win_voltage_sum = 0;
        win_current_sum = 0;
        wout_current_sum = 0;
        wout_voltage_sum = 0;
    }
}

static int uart_putchar(char c, FILE *stream)
{
    struct uart_port *udata = (struct uart_port *)fdev_get_udata(stream);

    if (c == '\n') uart_putchar('\r', stream);
    while (!(*udata->ucsra & USART_DREIF_bm));
    *udata->udr = c;
    return 0;
}

static int uart_getchar(FILE *stream)
{
    struct uart_port *udata = (struct uart_port *)fdev_get_udata(stream);
    while (!(*udata->ucsra &  USART_RXCIF_bm));

    return *udata->udr;
}


ISR(USARTD0_RXC_vect)
{
    //Time information transmitted
    fscanf(&usart2,"%d", &night);
    fscanf(&usart2,"%d:%d:%d", &hours, &minutes, &seconds);
    fscanf(&usart2,"%d/%d/%d", &day, &month, &year);
    fscanf(&usart2,"%d|%d", &longitude, &latitude);

    fprintf(&usart0, "%d:%d:%d\r\n", hours, minutes, seconds);
    fprintf(&usart0, "%d/%d/%d\r\n", day, month, year);
    fprintf(&usart0,"%d|%d\r\n", longitude, latitude);

    //Power variables
    fprintf(&usart2,"%d\r\n", sin_voltage_average);
    fprintf(&usart2,"%d\r\n", sin_current_average);
    fprintf(&usart2,"%d\r\n", sout_voltage_average);
    fprintf(&usart2,"%d\r\n", sout_current_average);
    fprintf(&usart2,"%d\r\n", win_voltage_average);
    fprintf(&usart2,"%d\r\n", win_current_average);
    fprintf(&usart2,"%d\r\n", wout_voltage_average);
    fprintf(&usart2,"%d\r\n", wout_current_average);

    while(usart_rx_is_complete(&USARTD0)) uart_getchar(&usart2);
}
```

2. Webserver Code

THE code for the RPi has been written in **Python** using *WebIDE* by *Adafruit* [19]. The RPi was connected to the network and programmed and accessed over SSH. The task is tedious using basic editors such as *nano* or *vi*. Adafruit created a nice solution to this problem where writing code and programming the RPi is much easier and flexible using *WebIDE*.

2.1. Python Script

```
#!usr/bin/python

import time
import serial
import sys
import io
import rrdtool
import ephem
import datetime

night = False

# configure the serial connections
ser = serial.Serial(
    port='/dev/ttyAMA0',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)

ser.flushInput();



while True:
    y = []

    file = open('/var/www/wordpress/wp-content/uploads/2014/txt/location.txt','r')
    longitude = file.readline().rstrip()
    latitude = file.readline()
    file.close()
    o = ephem.Observer()
    o.lat = latitude
    o.long = longitude
    s=ephem.Sun()
    s.compute()
    longitude = int(float(longitude) * 100)
    latitude = int(float(latitude) * 100)

    if (ephem.localtime(o.next_rising(s)) - datetime.datetime.now()) <=
        datetime.timedelta(minutes = 2) :
        night = False
    if (ephem.localtime(o.next_setting(s)) - datetime.datetime.now()) <=
        datetime.timedelta(minutes = 2) :
        night = True

    ser.write (str(night).encode())
    ser.write("\r\n")

    while (ser.inWaiting() != 0):
```

```
    ser.flushInput();

ser.write(time.strftime("%H:%M:%S"))
ser.write("\r\n")
ser.write(time.strftime("%d/%m/%Y"))
ser.write("\r\n")
ser.write(str(longitude).encode())
ser.write("|")
ser.write(str(latitude).encode())
ser.write("\r\n")
print(time.strftime("%H:%M:%S"))
print(time.strftime("%d/%m/%Y"))
print str(longitude).encode()
print str(latitude).encode()

for i in range (0,8):
    x = ser.readline()
    x = x.rstrip('\r\n')
    y.append(int(x))


sin_voltage = (y[0] - 205) / 73.48
sin_current = (y[1] - 205) / 470.9
sout_voltage = (y[2] - 205) / 125.9
sout_current = (y[3] - 205) / 470.9

win_voltage = (y[4] - 205) / 73.48
win_current = (y[5] - 205) / 470.9
wout_voltage = (y[6] - 205) / 125.9
wout_current = (y[7] - 205) / 470.9

svout = round(sout_voltage, 2)
wvout = round(wout_voltage, 2)

spin = round((sin_voltage * sin_current), 2)
if spin < 0:
    spin = 0
spout = round((sout_voltage * sout_current), 2)
if spout < 0:
    spout =0
if spin == 0:
    seff = 0
else:
    seff = round(((spout / spin) * 100), 2)

wpin = round((win_voltage * win_current), 2)
if wpin < 0:
    wpin = 0
wpout = round((wout_voltage * wout_current), 2)
if wpout < 0:
    wpout = 0
if wpin == 0:
    weff = 0
else:
    weff = round(((wpout / wpin) * 100), 2)


print(svout, spin, spout, seff, wvout, wpin, wpout, weff)

ret = rrdtool.update('Solar.rrd', 'N:%s:%s:%s:%s' %(svout, spin, spout, seff))
ret = rrdtool.update('Wind.rrd', 'N:%s:%s:%s:%s' %(wvout, wpin, wpout, weff))

#Solar graphs
ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/HSPowerOut.png","-w600","-h200",
                        "--start", "-1h", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Solar.rrd:SolarPowerOut:AVERAGE",
                        "AREA:PowerOut#00FF00:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")
```

```
ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/DSPowerOut.png","-w600","-h200",
                        "--start", "-1d", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Solar.rrd:SolarPowerOut:AVERAGE",
                        "AREA:PowerOut#00FF00:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/WSPowerOut.png","-w600","-h200",
                        "--start", "-1w", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Solar.rrd:SolarPowerOut:AVERAGE",
                        "AREA:PowerOut#00FF00:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/MSPowerOut.png","-w600","-h200",
                        "--start", "-1m", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Solar.rrd:SolarPowerOut:AVERAGE",
                        "AREA:PowerOut#00FF00:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/YSPowerOut.png","-w600","-h200",
                        "--start", "-1y", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Solar.rrd:SolarPowerOut:AVERAGE",
                        "AREA:PowerOut#00FF00:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/YSEff.png","-w600","-h200",
                        "--start", "-1y", "--vertical-label=Efficiency (%)",
                        "DEF:SolarEff=Solar.rrd:SolarEfficiency:AVERAGE",
                        "AREA:SolarEff#B20000:Solar Efficiency",
                        "LINE:SolarEff#000000",
                        "COMMENT:\\n",
                        "GPRINT:SolarEff:AVERAGE:Average Efficiency\: %3.2lf %S%%",
                        "COMMENT:  ",
                        "GPRINT:SolarEff:MAX:Maximum Efficiency\: %3.2lf %S%%\\r",
                        "GPRINT:SolarEff:MIN:Minimum Efficiency\: %3.2lf %S%%\\r")
#Wind graphs

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/HWPowerOut.png","-w600","-h200",
                        "--start", "-1h", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Wind.rrd:WindPowerOut:AVERAGE",
                        "AREA:PowerOut#00BFF2:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/DWPowerOut.png","-w600","-h200",
```

```
                        "--start", "-1d", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Wind.rrd:WindPowerOut:AVERAGE",
                        "AREA:PowerOut#00BFF2:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/WWPowerOut.png","-w600","-h200",
                        "--start", "-1w", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Wind.rrd:WindPowerOut:AVERAGE",
                        "AREA:PowerOut#00BFF2:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/MWPowerOut.png","-w600","-h200",
                        "--start", "-1m", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Wind.rrd:WindPowerOut:AVERAGE",
                        "AREA:PowerOut#00BFF2:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/YWPowerOut.png","-w600","-h200",
                        "--start", "-1y", "--vertical-label=Watts (W)",
                        "DEF:PowerOut=Wind.rrd:WindPowerOut:AVERAGE",
                        "AREA:PowerOut#00BFF2:Power Output",
                        "LINE:PowerOut#000000",
                        "COMMENT:\\n",
                        "GPRINT:PowerOut:AVERAGE:Average Power Output\: %3.2lf %SW",
                        "COMMENT:  ",
                        "GPRINT:PowerOut:MAX:Maximum Power Output\: %3.2lf %SW\\r",
                        "GPRINT:PowerOut:MIN:Minimum Power Output\: %3.2lf %SW\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/WSEff.png","-w600","-h200",
                        "--start", "-1y", "--vertical-label=Efficiency (%)",
                        "DEF:WindEff=Wind.rrd:WindEfficiency:AVERAGE",
                        "AREA:WindEff#4C3100:Wind Efficiency",
                        "LINE:WindEff#000000",
                        "COMMENT:\\n",
                        "GPRINT:WindEff:AVERAGE:Average Efficiency\: %3.2lf %S%%",
                        "COMMENT:  ",
                        "GPRINT:WindEff:MAX:Maximum Efficiency\: %3.2lf %S%%\\r",
                        "GPRINT:WindEff:MIN:Minimum Efficiency\: %3.2lf %S%%\\r")

#Battery graphs
ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/Bat1.png","-w600","-h200",
                        "--start", "-1w", "--vertical-label=Voltage (V)",
                        "DEF:Battery1=Solar.rrd:SolarVoltageOut:AVERAGE",
                        "LINE:Battery1#663096:Battery Voltage",
                        "COMMENT:\\n",
                        "GPRINT:Battery1:AVERAGE:Average Voltage\: %3.2lf %SV",
                        "COMMENT:  ",
                        "GPRINT:Battery1:MAX:Maximum Voltage\: %3.2lf %SV\\r",
                        "GPRINT:Battery1:MIN:Minimum Voltage\: %3.2lf %SV\\r")

ret = rrdtool.graph(
    "/var/www/wordpress/wp-content/uploads/2014/graphs/Bat2.png","-w600","-h200",
                        "--start", "-1w", "--vertical-label=Voltage (V)",
                        "DEF:Battery2=Wind.rrd:WindVoltageOut:AVERAGE",
                        "LINE:Battery2#0000FF:Battery Voltage",
```

```
                        "COMMENT:\\n",
                        "GPRINT:Battery2:AVERAGE:Average Voltage\: %3.2lf %SV",
                        "COMMENT:  ",
                        "GPRINT:Battery2:MAX:Maximum Voltage\: %3.2lf %SV\\r",
                        "GPRINT:Battery2:MIN:Minimum Voltage\: %3.2lf %SV\\r")

    time.sleep(6)
```

DOCUMENT II

COST ANALYSIS

Index

1. Purchased Materials

THESE components were sourced from *Mouser*, *Amazon* and *eBay*. Their cost fluctuates and is probably different by now. The list is therefore intended to give a rough idea of the total cost and materials used. The project is a prototype. It is hard to exactly tell what the cost would be for the commercialized version because of diminishing costs when ordered in bulk, etc. Also, more components than required were ordered in many cases due to their possible loss when assembling the boards manually.

Product Description	Manufacturer #	Quantity	Price (USD)	Ext. (USD)
Fixed Inductors 680uH 20% SMD 1210	SRR1210-681M	1	1.19	1.19
Fixed Inductors 150uH 20% SMD 1210	SRR1210-151M	2	1.20	2.40
Schottky Diodes & Rectifiers 40 Volt 3.0 Amp 75 Amp IFSM	SSA34HE3/61T	3	0.262	0.79
USB Connectors 4P RECEPTACLE TYPE A	87583-2010BLF	1	0.894	0.89
EMI Filter Beads, Chips & Arrays MULTILAYER CHIP BEAD Z=1K OHM @100MHz 25%	2512061027Y1	1	0.076	0.08
Zigbee/802.15.4 Modules XBee ZB w/WiredWhip AT Router F/W	XB24-Z7WIT-004	2	18.54	37.08
Schottky Diodes & Rectifiers 5.0 Amp 60 Volt 150A IFSM @ 8.3ms	SB560A-E3/73	1	0.461	0.46
Fixed Inductors WE-HCI SMD Flat Wire HighCurrent Inductor	74435584700	1	10.69	10.69
Tactile Switches 50 mAmps at 12 Volts	SKQGAKE010	2	0.153	0.31
Slide Switches SPDT ON-NONE-ON TOP	SS12SDP2	1	1.24	1.24
MOSFET P-Ch 55 Volt 80 Amp	STP80PF55	1	3.35	3.35
Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.01uF 25volts X7R 10%	VJ1206Y103KXX CW1BC	2	0.065	0.13
Thick Film Resistors - SMD 1/4watts 10Kohms 5%	RK73B2BTTD103J	10	0.019	0.19
Multilayer Ceramic Capacitors MLCC - SMD/SMT 50volts 0.47uF 10%	C1206C474K5 NACTU	2	0.382	0.76
Aluminium Electrolytic Capacitors - SMD 10uF 16V 85C	AVE106M16B12T-F	2	0.403	0.81

Diodes - General Purpose, Power, Switching 100 Volt 500mA 4ns	1N4148W-E3-18	10	0.124	1.24
Voltage Regulators - Switching Regulators 3A SD Vtg Reg	LM2576S-12/NOPB	1	3.08	3.08
Fixed Terminal Blocks 5MM FIX PCB TERM BLK 2 CIRCUITS	39543-3002	1	0.905	0.91
Thick Film Resistors - SMD 100K OHM 5%	ERJ-8GEYJ104V	10	0.03	0.3
Zener Diodes 10 Volt 0.5W 5%	BZX55C10-TAP	2	0.109	0.22
Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.1uF 50volts Y5V +80-20%	VJ1206V104ZX APW1BC	50	0.03	1.50
Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 10uF 50volts X5R 10%	C3216X5R1H106 K160AB	3	1.09	3.27
Thick Film Resistors - SMD 1206 220ohms 5% Tolerance	ERJ-8GEYJ221V	10	0.03	0.30
Standard LEDs - SMD Hyper Red, 645nm 15mcd, 20mA	LH N974-KN-1	1	0.109	0.11
Standard LEDs - SMD Green, 570nm 45mcd, 20mA	LG N971-KN-1	1	0.076	0.08
Resettable Fuses - PPTC 0.20A 9V 0.65ohm	MF-PSMF020X-2	2	0.372	0.74
Thick Film Resistors - SMD 1/4watt 6.98Kohms 1%	CRCW12066K9 8FKEA	10	0.04	0.40
Thick Film Resistors - SMD 1206 220Kohms 1% Tolerance	ERJ-8ENF2203V	10	0.041	0.41
Thick Film Resistors - SMD 1206 4.02Kohms 1% Tolerance	ERJ-8ENF4021V	10	0.041	0.41
Thick Film Resistors - SMD 1/4watt 115Kohms 1%	CRCW1206115 KFKEA	10	0.04	0.40
Current Sense Resistors - SMD 1watt .005ohms 1%	WSL20105L000FEA18	2	2.59	5.18
Aluminium Electrolytic Capacitors - SMD 1000uF 50V	EEV-FK1H102M	2	2.13	4.26
USB Connectors USB Mini-B Recept On-The-Go Rt.Angle	67503-1020	1	1.57	1.57
Gate Drivers 100V HalfBridge MOSFET Driver with Anti-Shoot-through Circuitry(Lead Free)	MIC4102YM	2	4.97	9.94
Aluminium Electrolytic Capacitors - SMD 25 Volts 470uF 20% 10x10	VE-471M1ETR-1010	1	0.60	0.60
Aluminium Electrolytic Capacitors - SMD 470uF 10V 85C	AVE477M10G24T-F	1	0.916	0.92
Aluminium Electrolytic Capacitors - SMD 470uF 6.3V 85C Case 8 x 10	AVE477M06F24T-F	2	0.741	1.18

Aluminium Electrolytic Capacitors - SMD 100uF 50V 85C Case 8 x 10	AVE107M50F24T-F	4	0.741	2.96
Headers & Wire HousingsWR-PHD Socket Header 2.54mm 32p	613032143121	3	0.34	1.02
General Purpose RelaysSPDT 12VDC 240Ohm 16A GEN PURPOSE RLY	RT314F12	4	3.90	15.60
Raspberry Pi - Model B	DEV-11546	1	39.95	39.95
Raspberry Pi - GPIO Ribbon Cable (6")	CAB-11489	1	2.95	2.95
Pi Tin for the Raspberry Pi - Clear	PRT-11623	1	7.95	7.95
Raspberry Pi - GPIO Shrouded Header (2x13)	PRT-11490	1	0.95	0.95
Wireless Bluetooth Transceiver Module RS232 / TTL HC-05	EIM361	1	5.24	5.24
8-bit Microcontrollers - MCU 44TQFP, IND TEMP GREEN, 1.6-3.6V	ATXMEGA128A4U-AU	1	4.91	4.91
Controller PCB	-	1	33.00	33.00

2. Free of Charge Materials

THESE components were sourced from *TI* and the ECE service shop of the *University of Illinois* free of charge. The estimate price is shown so as to compute the total cost of the project in the next section.

2.1. TI Samples

Product Description	Manufacturer #	Quantity	Price (USD)	Ext. (USD)
MOSFET 60V N-Chnl NexFET Pwr MOSFET	CSD18534KCS	4	1.81	7.24
Voltage Regulators - Switching Regulators 3A SD VTG Reg	LM2576S-3.3/NOPB	1	3.08	3.08
Voltage Regulators - Switching Regulators 3A SD VTG Reg	LM2576S-5.0/NOPB	1	3.08	3.08
Current & Power Monitors & Regulators Hi-Sd Msmnt Current Shunt Mntr Crnt Otp	INA168NA/3K	3	2.51	7.53

2.2. ECE Service Shop

Product Description	Manufacturer #	Quantity	Price (USD)	Ext. (USD)
Carbon Film Resistors - Through Hole 2.2Kohms 0.05	291-2.2K-RC	6	0.131	0.79
Rectifiers Vr/400V Io/1A T/R	1N4004	6	0.153	0.92
Transistors Bipolar - BJT 600mA 75V NPN	P2N2222AG	6	0.458	2.78
Current & Power Monitors & Regulators Hi-Sd Msmnt Current Shunt Mntr Crnt Otp	INA168NA/3K	3	2.51	7.53
2mm 10pin XBee Socket	PRT-08272	2	0.51	1.02
Standard LEDs - Through Hole Red 460mcd 640nm 40 deg Diffused	WP710A10SRD/F	2	0.153	0.31
Standard LEDs - Through Hole GREEN DIFFUSED	WP7113GD	2	0.185	0.37
Relay PCB	-	1	10.00	10.00

3. Labor

THE labor must be taken into account in the development of the product although it is a one time expense (or for the most part of it because there are normally revisions to do), once the product is developed it can be sent to a manufacturing plant limiting further costs mainly to hardware expenses.

Work	Hours	Wage (USD/hour)	Cost (USD)
Hardware Design	100	35	3500
Hardware Prototyping and Simulation	100	35	3500
Hardware Assembly	10	15	150
Software Development	80	35	2800

4. Total Costs

THE total costs of the project are the hardware costs and the labor costs. There are other costs that were not included such as the solar panel costs and the wind turbine because the developed project could be used with any devices that meet the power specifications which of course are subject to significant price variations.

Description	**Cost (USD)**
Hardware	256.57
Labor	9,950.00
Grand Total	**10,206.57**

4.1. Brief Analysis

The total labor costs can be seen as the development costs of the product. They are a one time expense (fixed costs), also know as *Non-recurring engineering*.

The total hardware costs can be much lower than those described in the previous table. Many components have duplicates that were not used, they were purchased in case some of them got lost or damaged. When purchased in bulk many electronic components's price goes down by a factor of four times or more for many of them. The estimate cost of hardware if the product is to be manufactured in significant quantities would be of around half the cost of the project.

DOCUMENT III

APPENDICES

Index

1. Extra Developed Work

THE student in charge of the solar tracker had trouble designing his circuit and helped was offered. A summary of the work that was done for that student is presented.

1.1. Solar Tracker Description

The solar tracker is a mechanism that allows the PV array to follow the sun throughout the day and optimized the total power production. In order to accomplish this task, the solar panels are mounted on a structure that can tilt and pivot around its axis using two stepper motors.

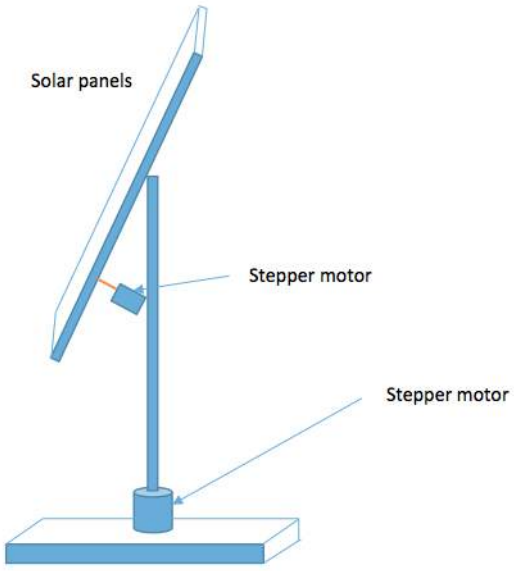


Figure 1. Solar Tracker Mechanism

The structures was designed and developed by the machine shop of the ECE department at the *University of Illinois* but the control circuit had to be designed.

The solar tracker is able to orientate and tilt the PV array using an accelerometer and compass : the LSM303D breakout board by *Pololu*. The main microcontroller is an Arduino which is able to receive information (location and time) from the main controller board through an XBee link. By programming the relevant solar angle functions it is possible to determine the optimum tilt and orientation angles.

1.2. Schematics

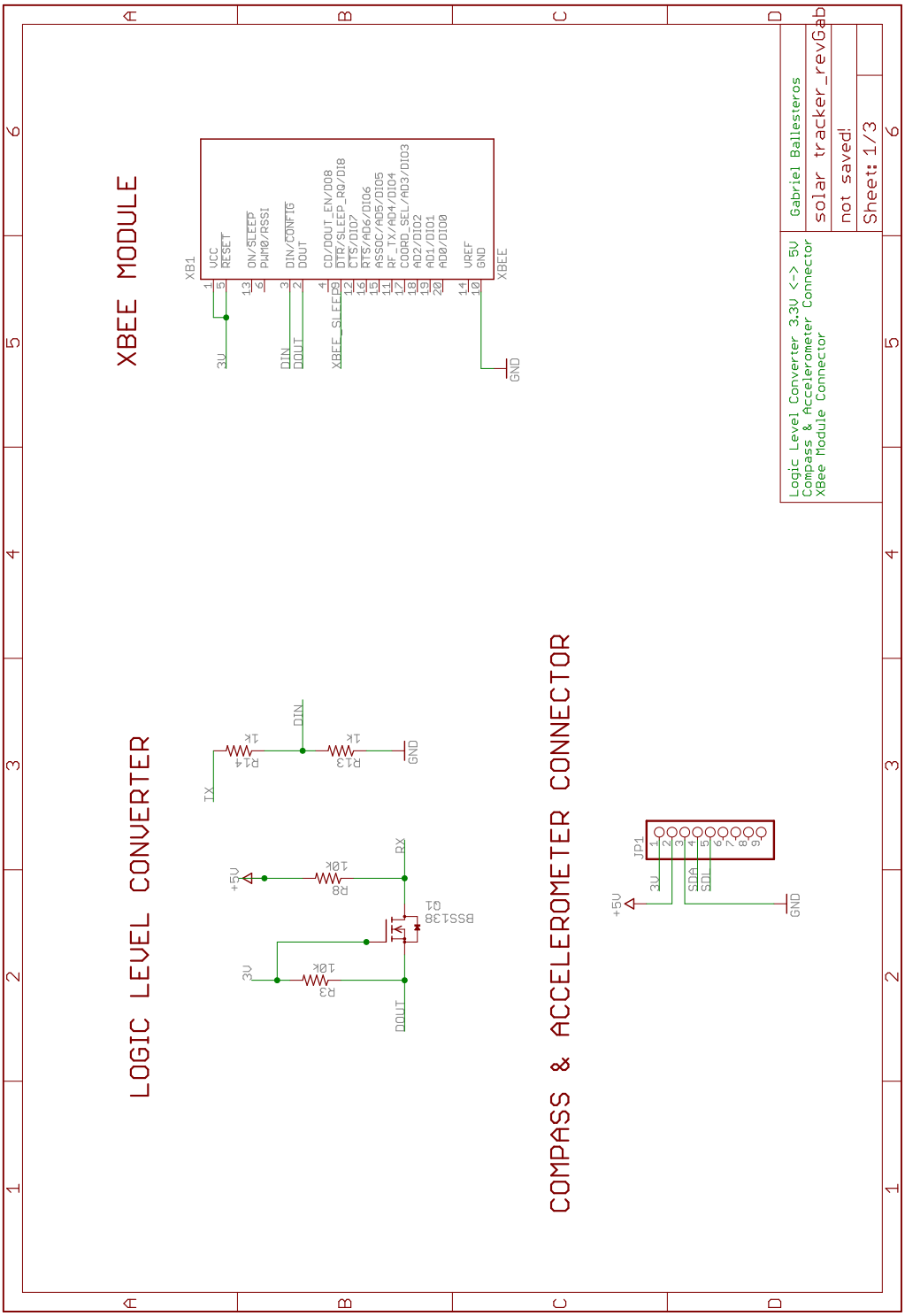


Figure 2. Sensor and Communication

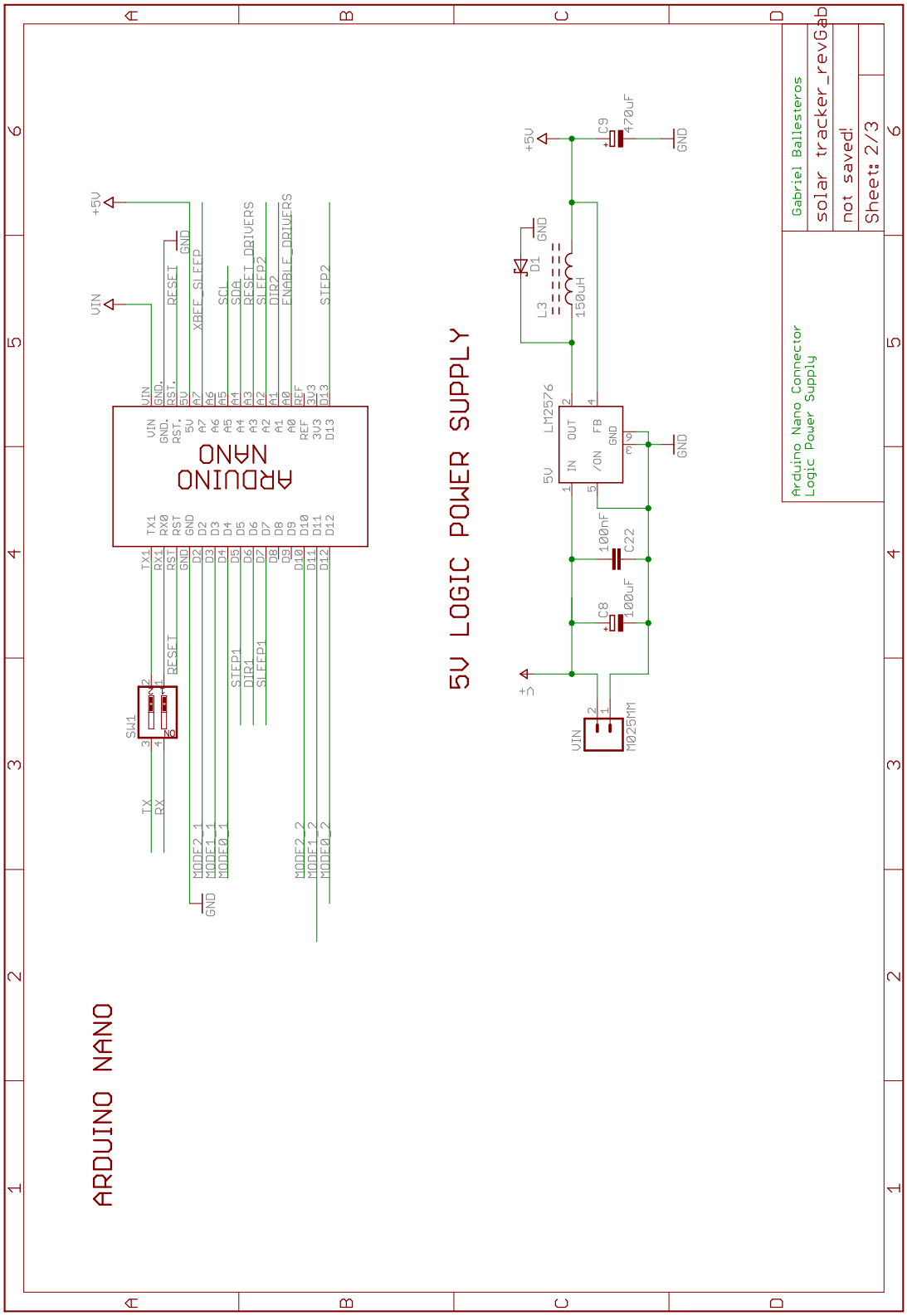


Figure 3. Arduino Nano and Logic Power Supply

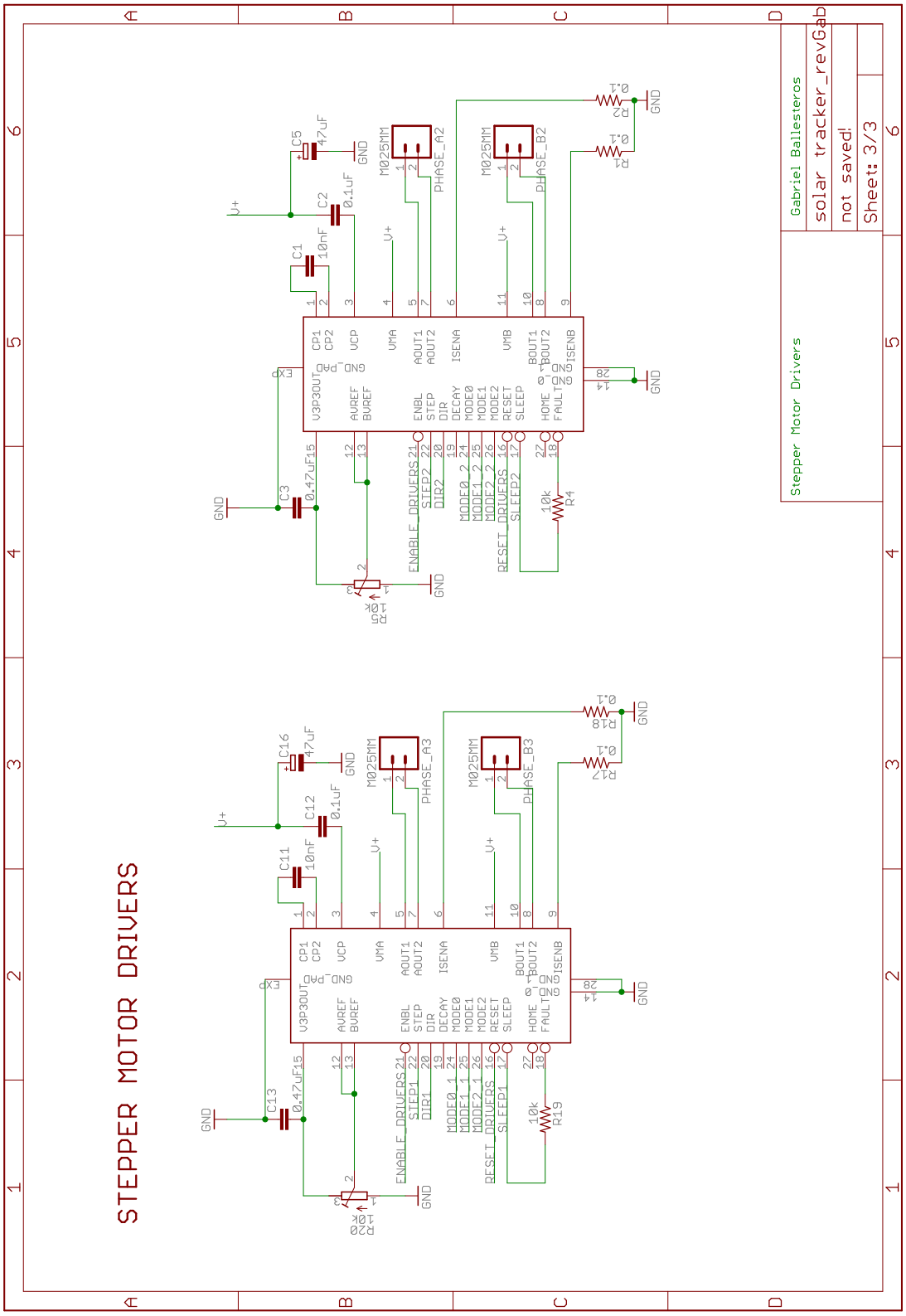


Figure 4. Stepper Motor Drivers

1.3. Eagle Layout

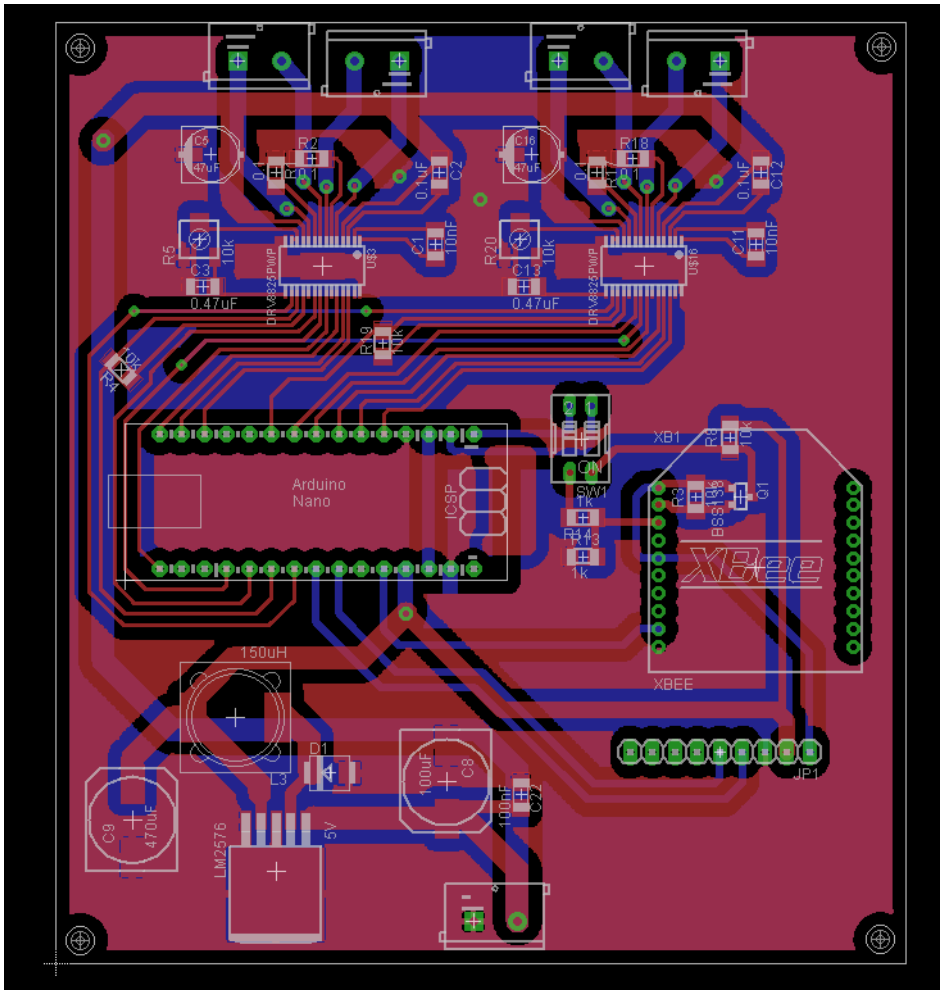


Figure 5. Solar Tracker Eagle Layout

1.4. PCB

The PCB was assembled by Azamat Zhunussov, the student in charge of the solar tracker.

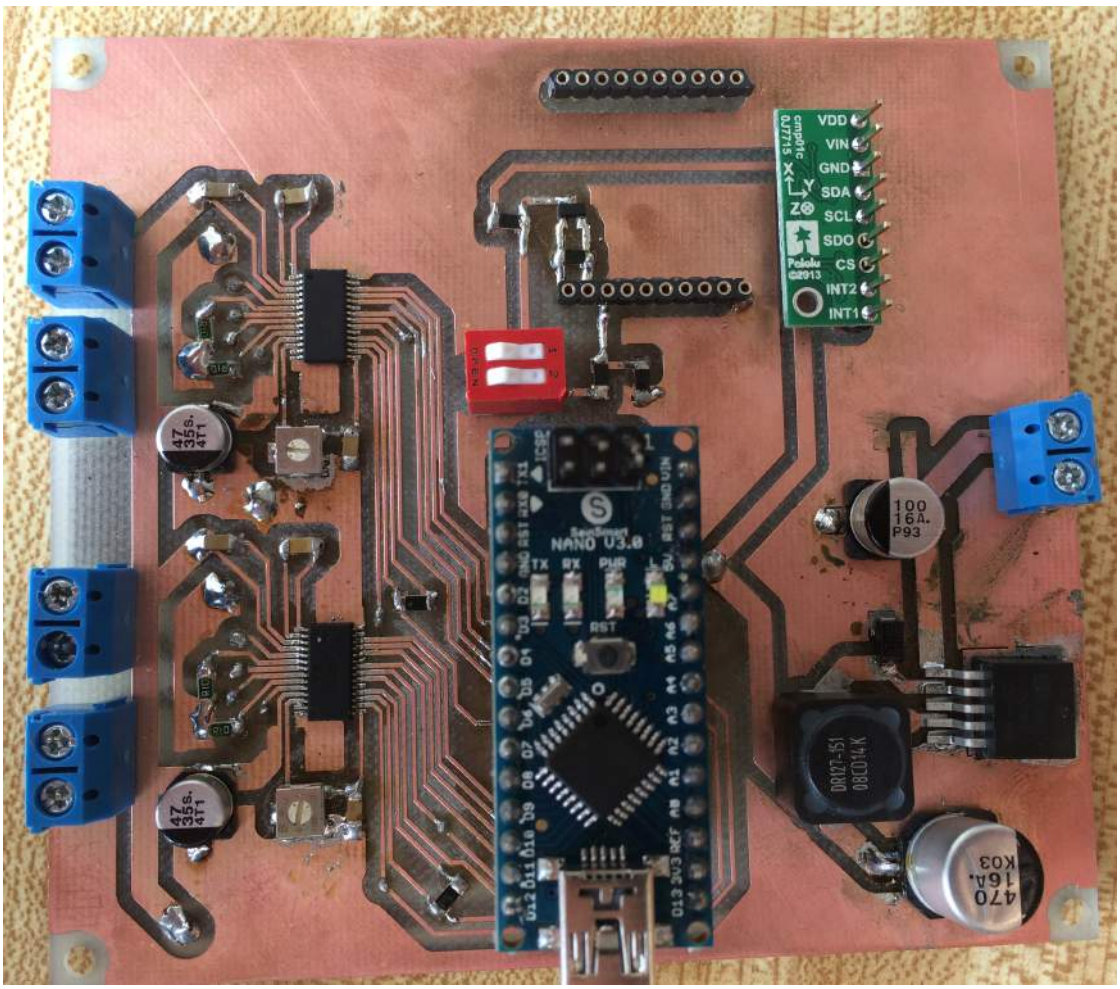


Figure 6. Assembled Solar Tracker PCB

2. Pictures

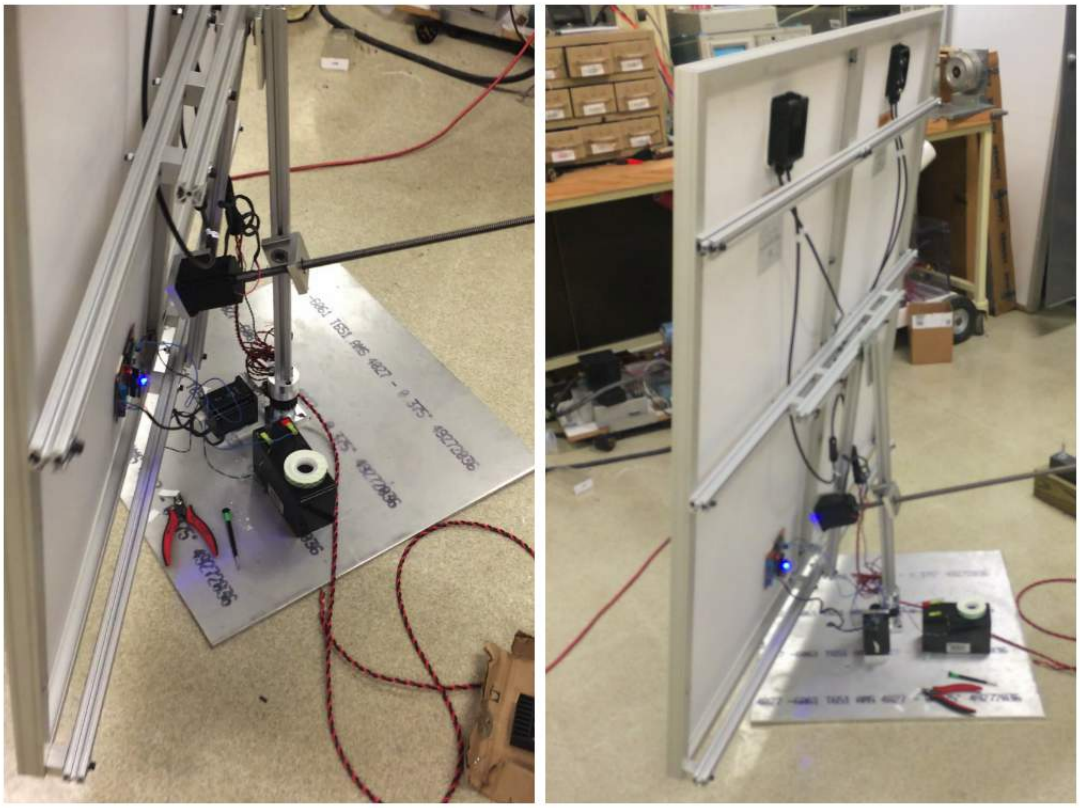


Figure 7. Solar Tracker

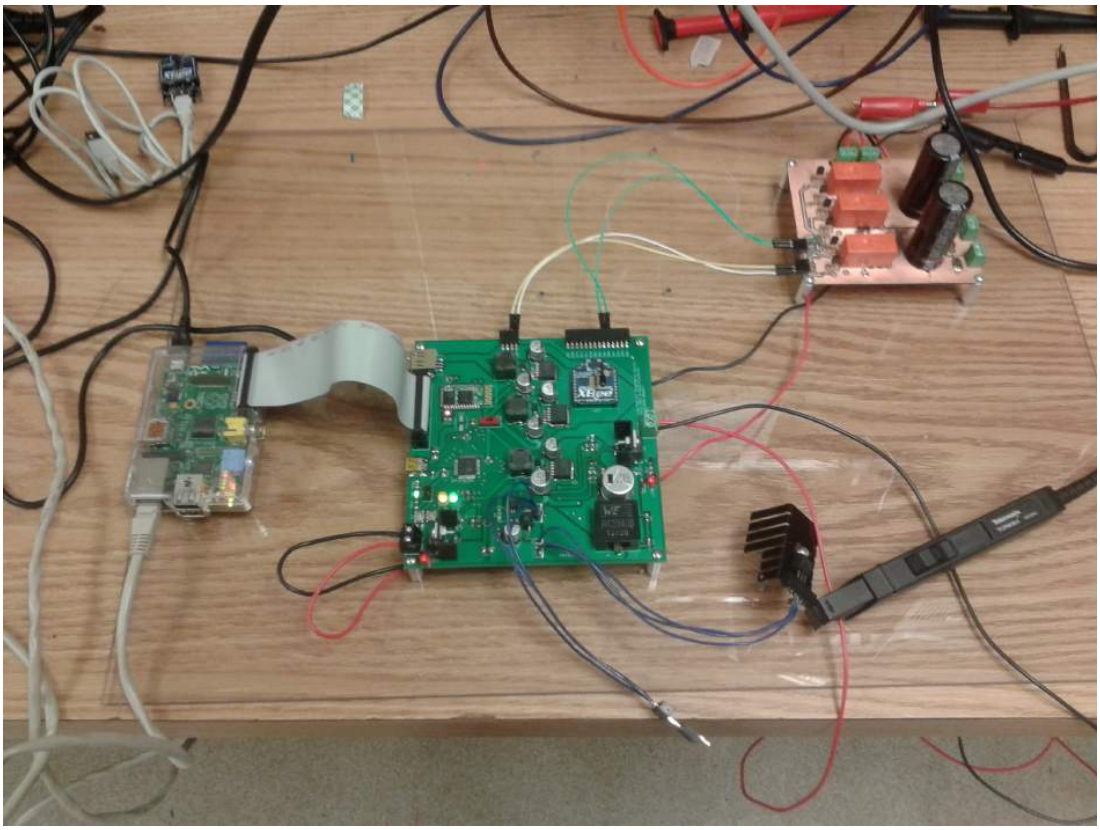


Figure 8. Project Mounted on a Plexiglass Sheet

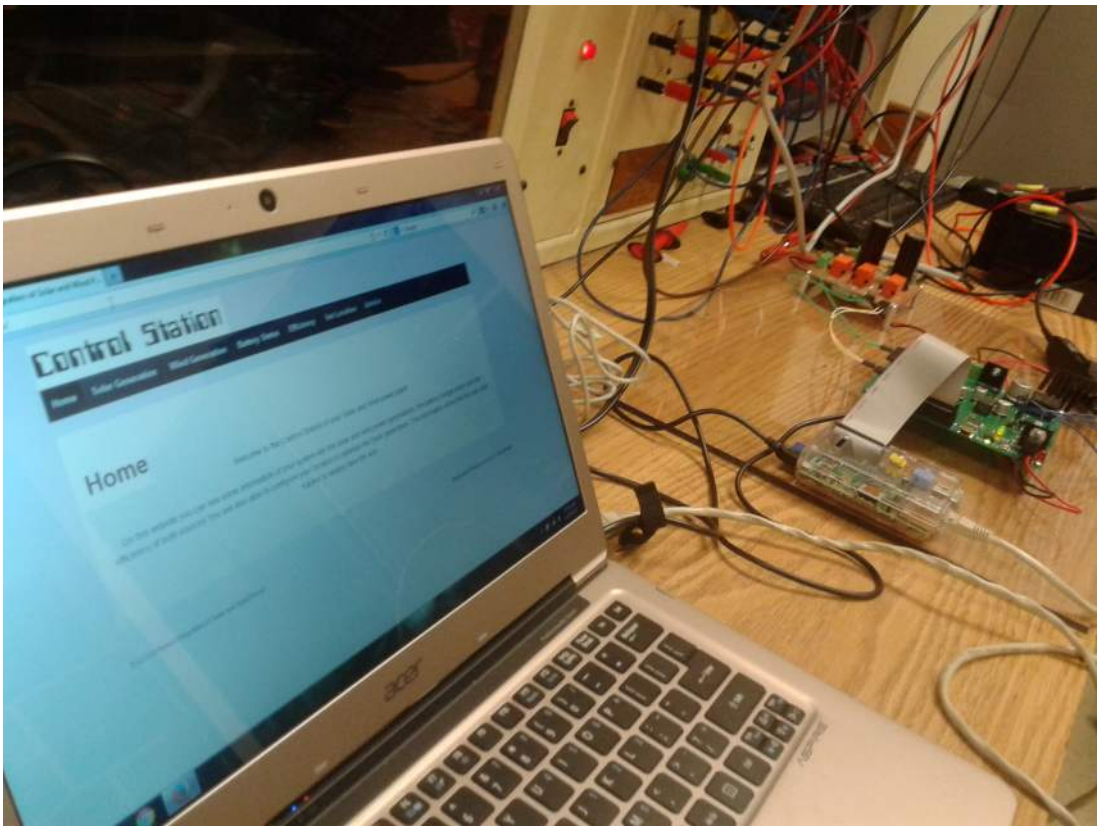


Figure 9. Working System

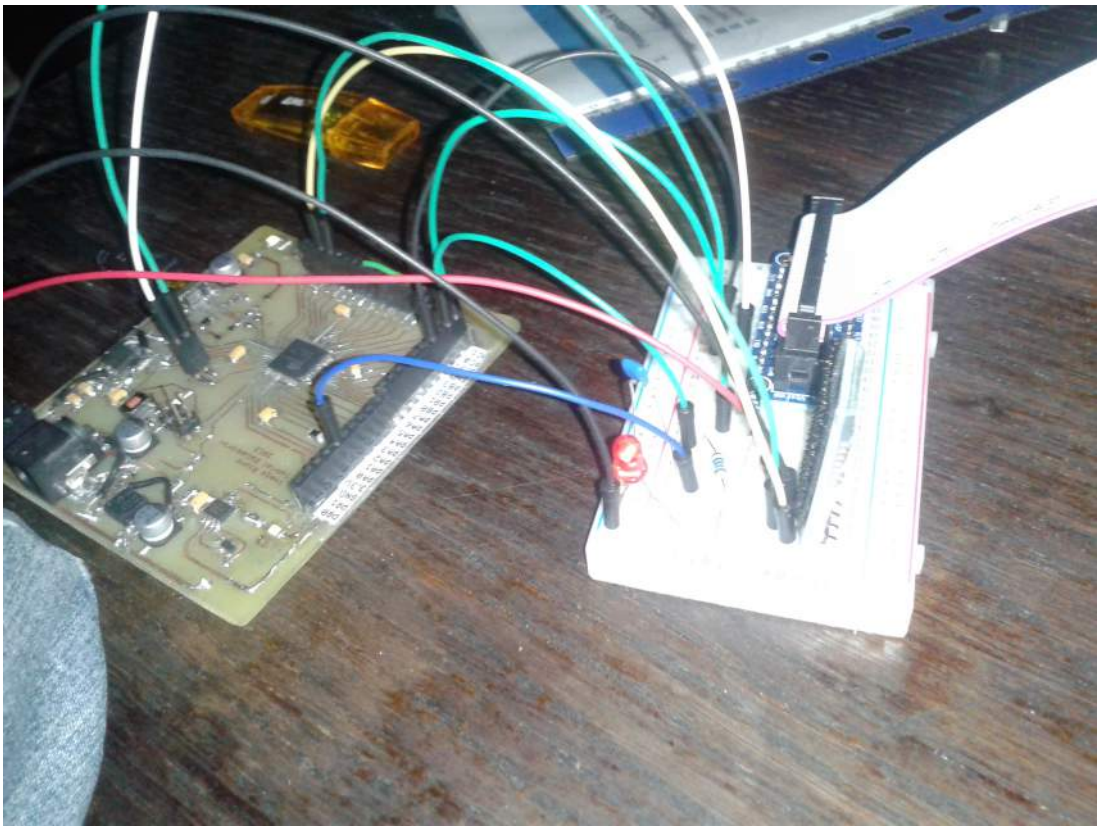


Figure 10. Prototyping

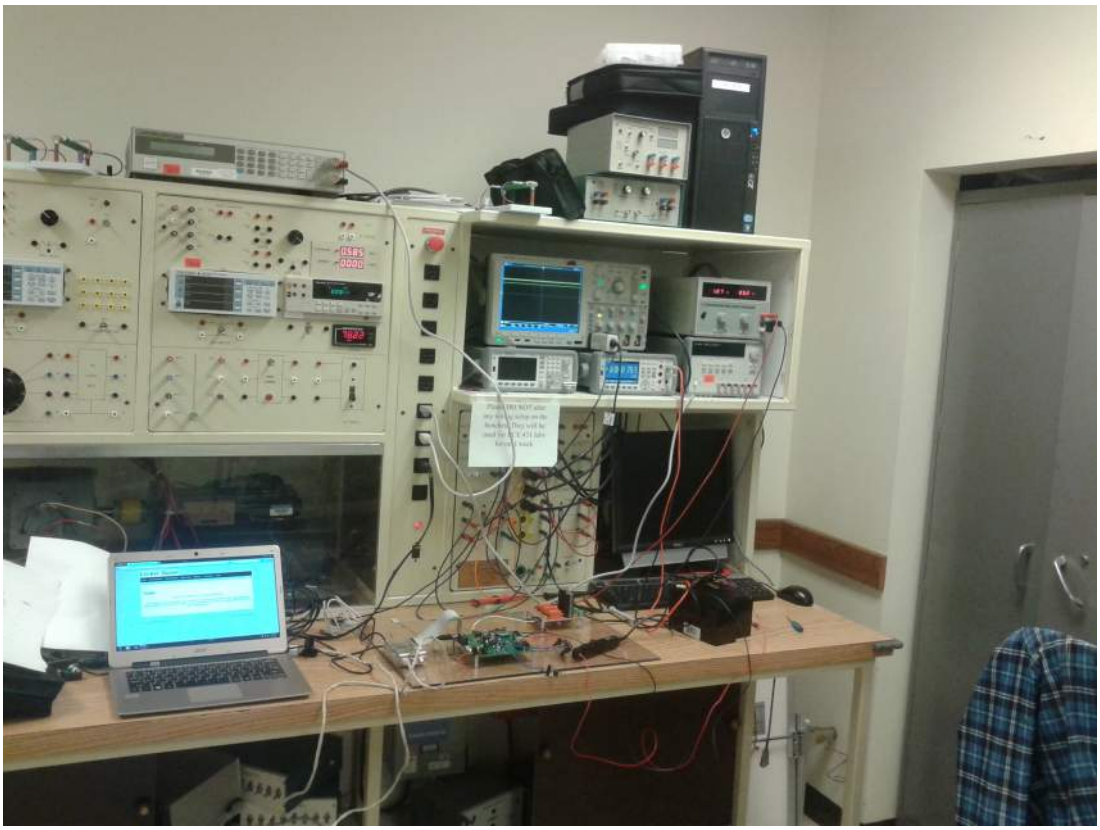


Figure 11. Test Bench

Bibliography

[1] **Oetiker, T., Partl, H., Hyna, I., Schlegl, E.**, *The Not So Short Introduction to LaTeX 2ε*, Diciembre 2009.

[2] **Universidad Pontificia Comillas**, *Página web de Proyectos Fin de Carrera*. http://www.iit.upcomillas.es/pfc

[3] **Oetiker, T.**, *RRDTool Website*. http://oss.oetiker.ch/rrdtool/

[4] **Rashid, M. H.**, *Power Electronics: Circuits, Devices, and Applications*, August 2003.

[5] **Masters, G. M.**, *Renewable and Efficient Electric Power Systems, 2nd Edition*, August 2013.

[6] **Atmel**, *Atmel AVR1300: Using the Atmel AVR XMEGA ADC*. http://www.atmel.com/images/atmel-8032-using-the-atmel-avr-xmega-adc_application-note_avr1300.pdf

[7] **Rosu-Hamzescu, M., Oprea, S., Microchip Technology Inc.**, *Practical Guide to Implementing Solar Panel MPPT Algorithms*. http://ww1.microchip.com/downloads/en/AppNotes/00001521A.pdf

[8] **Falin. J., Texas Instruments**, *Reverse Current/Battery Protection Circuits*. http://www.ti.com/lit/an/slva139/slva139.pdf

[9] **Raspberry Pi Foundation**, *Raspberry Pi Website*. http://www.raspberrypi.org/

[10] **Cadsoft**, *Eagle*. http://www.cadsoftusa.com/

[11] **Wordpress**, *Wordpress Website*. https://wordpress.com/

[12] **CyberChimps**, *Responsive Theme*. https://wordpress.org/themes/responsive

[13] **LIGHTTPD**, *LIGHTTPD Website*. http://www.lighttpd.net/

[14] **SQLite**, *SQLite Website*. http://www.sqlite.org/

[15] **SedLex**, *Image Zoom Plugin*. http://wordpress.org/plugins/image-zoom/

[16] **Atmel**, *Atmel Studio 6*.
http://www.atmel.com/microsite/atmel_studio6/

[17] **OrCAD**, *Pspice*.
http://www.electronics-lab.com/downloads/schematic/013/

[18] **IAR**, *IAR Embedded Workbench*.
http://www.iar.com/Products/IAR-Embedded-Workbench/

[19] **Adafruit**, *WebIDE*.
https://learn.adafruit.com/webide/overview

Datasheets

[20] **Atmel**, *Página web de Proyectos Fin de Carrera*.
http://www.atmel.com/Images/Atmel-8387-8-and16-bit-AVR-Microcontroller-XMEGA-A4U_Datasheet.pdf

[21] **Atmel**, *Atxmega128a4u Manual*.
http://www.atmel.com/Images/Atmel-8331-8-and-16-bit-AVR-Microcontroller-XMEGA-AU_Manual.pdf

[22] **Texas Instruments**, *Mosfet Datasheet*.
http://www.ti.com/lit/ds/symlink/csd18534kcs.pdf

[23] **Texas Instruments**, *INA168 Current Sensor Datasheet*.
http://www.ti.com/lit/ds/symlink/ina168.pdf

[24] **Micrel**, *MIC4102 Mosfet Driver Datasheet*.
www.sekorm.com/Ectm/23950

[25] **Texas Instruments**, *LM2576 Datasheet*.
http://www.ti.com/lit/ds/symlink/lm2576.pdf

[26] **TE Connectivity**, *Bistable Relay Datasheet*.
http://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FRT1%7F0913%7Fpdf%7FEnglish%7FENG_DS_RT1_0913_RT1.pdf%7FN-A

[27] **Solarland**, *Solar Panels Datasheet*.
http://www.sunshineworks.com/downloads/12-volt-solar-panels/SLP100-12U.pdf