Grid-Independent Charging Station with Power Flow Display

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the
Degree of Bachelor of Science
by:

Adrian C. Delphia

Samuel T. Veilleux

Submitted 12/17/2012

Advised by Professor Stephen Bitar
Co-advised by Professor John McNeill

Table of Contents

Table of Figures 3

Table of Tables 5

Abstract 1

Background 2

  Previous MQPs and Available Resources 2

  Design Concept and Motivation 2

Project Development 4

  Early Stages 4

  Grid Independence 4

  Finalizing Goals 5

Project Description 8

  Overview 8

  Solar Panel 9

  Charge Controller 10

  Battery 15

  Inverter 17

  Inverter Power Switch Circuit 20

    Selecting a Transient Voltage Suppression Diode 21

    Selecting a MOSFET 22

  Current Sensing 26

    Background 26

    Methods 26

    ACS712 28

  Outlets 36

  DC/DC converter 37

  Arduino 43

    Introduction 43

    The Memory Problem 45

    The Main Loop 45

    Sensing Wattage 45

    Drawing the Arrow 46

Conclusions 49

Shortcomings 49

Recommended Future Developments 50

Implementing an On/Off Switch 50

Adding Another Display Screen 51

Final Thoughts 52

Works Cited 53

Referenced Datasheets 53

Appendix A: Arduino Code 54

Appendix B: Adafruit_GFX Library 63

Appendix C: Proof of Concept Photo 72

Table of Figures

Figure 1: Power flow diagrams for hybrid electric vehicles.[5] 3
Figure 2: The Final Project. Higher resolution available in Appendix C 6
Figure 3: System level power flow chart with data-lines 8
Figure 4: Picture of AK roof taken from Salisbury Labs 9
Figure 5: Faceplate of Solar Panel 9
Figure 6: An Aerial Photograph of AK from the Southwest with Solar Panel Annotated 10
Figure 7: Result of "Renewable Energy Applications" Project 11
Figure 8: Rogue MPT-3024 Charge Controller 12
Figure 9: Outback Flex Max 60 MPPT Charge Controller 12
Figure 10: TP Box PCM-924 Charge Controller 13
Figure 11: Tracer 2210RN Solar Charge Controller 14
Figure 12: BP20-12 Sealed Lead Acid Battery 17
Figure 13: Modified Sine Wave Plot[3] (3-level) 17
Figure 14: Tripp-Lite Inverter 18
Figure 15: PowerBright PureSine Inverter 19
Figure 16: Samlex PST-30S-12A Inverter 19
Figure 17: Equivalent circuit for inverter controller. 20
Figure 18: Diode package style for the TVS protection. 21
Figure 19: Graph of $r_{ds(on)}$ as a function of temperature with a drain current of 130A. 22
Figure 20: Equivalent thermal circuit for analysis 23
Figure 21: Thermal equivalent circuit used to calculate heatsink properties. 24
Figure 22: Heatsink to ensure MOSFET switch doesn't burn out. 25
Figure 23: The two wires on the left make the physical connection and the lower ones sense the current accurately. 27
Figure 24: The current sensing hall-effect board. 29
Figure 25: The left shows the sensitivity for the ±5A and the right is for the ±20A sensor 30
Figure 26: Sensitivity for the 20A sensor against different current values and temperatures. 31
Figure 27: Sensitivity of the 20A sensor across temperature. 31
Figure 28: Left is the supply current for a 5A sensor and on the right is the supply current for the 20A sensor. 32
Figure 29: Labeled construction of the current sensor board. 34
Figure 30: Soldered circuit board schematic that the current sensor board is mounted to. 35
Figure 31: Two standard AC outlets, installed with the box exposed 36
Figure 32: USB Faceplate 37
Figure 33: AC Faceplate with USB Outlet Included 37
Figure 34: The DC/DC converter used to power the USB port from the battery. 38
Figure 35: Test circuit for CPT DC/DC Module 39
Figure 36: CPT DC/DC converter efficiency as a function of supply voltage 40
Figure 37: Ripple Voltage on CPT DC/DC Converter (with 1.47 amp load) 41
Figure 38: Circuit used to measure CPT DC/DC converter consumption at no-load 41

Figure 39: Voltage as an idling CPT DC/DC converter uses a capacitor as its source 42
Figure 40: Self discrge of capacitor in idle-state test circuit for CPT DC/DC converter 43
Figure 41: A snapshot of the Adafruit 2.8" TFT shield for Arduino Uno Rev 3 44
Figure 42: A cropped and marked-up sketch of a right-facing display-arrow 47

Table of Tables

Table 1: Value Analysis for Battery Chemistries 15
Table 2: Test data gathered on CPT DC/DC Module 39

Abstract

This paper discusses a grid-independent charging display system from conception to implementation. Everything from initial ideas and brainstorming to feasibility of implementation are thoroughly discussed, as well as challenges met along the way. A functioning prototype capable of charging a standard cellphone 24 hours a day, 7 days week, independently of the grid is built and documented. Additionally a standard 120V AC outlet for use with low-power devices is integrated into the display, sharing the same power source. The system is powered exclusively with the solar panels on the roof off the Atwater Kent building on the Worcester Polytechnic Institute (WPI) campus.

Challenges met involve energy-storage for night-time operation, system-monitoring, and power flow control. Multiple touch screen displays are integrated at the location of the charging station. The processors controlling the displays have data-feeds which provide information about the voltages and currents at certain points of the circuit. Storing this data digitally allows for complex analysis of the system's behavior and usage over time. The display board will be installed in the Atwater Kent building in such a location that students can easily interface with it, and the information it provides will be easy to understand. The availability of purely renewable energy (and the ease of viewing its consumption-statistics) will promote awareness and green-thinking among electrical engineering students.

Background

Previous MQPs and Available Resources

In 2010, Jim Dunn of Future Solar LLC donated six solar panels to the Worcester Polytechnic Institute Electrical Engineering Department. Included were two of each of three different types of panels, with power ratings close to 250 watts. Two of them have been installed on the roof of the Atwater Kent building, at a fixed angle. These panels provided motivation for a 2011 MQP entitled “Renewable Energy Applications” (JKM 5A11). The focus of that project was to design and build a board which could apply maximum power point tracking (MPPT) and apply DC/DC conversion on the harvested power based on a set of user-defined software rules. The exemplary-implementation of the finished product was the board’s ability to charge a lead-acid battery while harvesting at the solar panel’s maximum power point.

After that project’s completion, the department sought to continue project work with another MQP which focused on solar technology. The existing solar installation on the roof of the building (with the leads feeding into NECAMSID lab) was a valuable resource, because it allowed initial project research and scope definition to begin immediately. Additional resources included the work done in the previous renewable energy MQP, and the four other solar panels.

Design Concept and Motivation

The goal of this project was to create a grid-independent charging station for cell phones and laptops. Its intended purpose is to promote green energy while saving energy, by creating a noticeable display in the lobby of Atwater Kent. The device’s purpose and usage will be clear to visitors and students alike. The design is transparent but will be shielded from touch, to attract attention while discouraging theft or tampering.

Inside, every component of the system is displayed in a neat and logical (with the solar panel input conduit labeled). There are three built in screens updating in real time to show the energy flow.

This will hopefully encourage the visitor to want to use the device and subsequently plug in a phone or laptop to one of the outlets. Then he or she will be amazed as the sensing algorithms and displays start updating to show energy flowing out of the solar panels into the battery and the device plugged in.

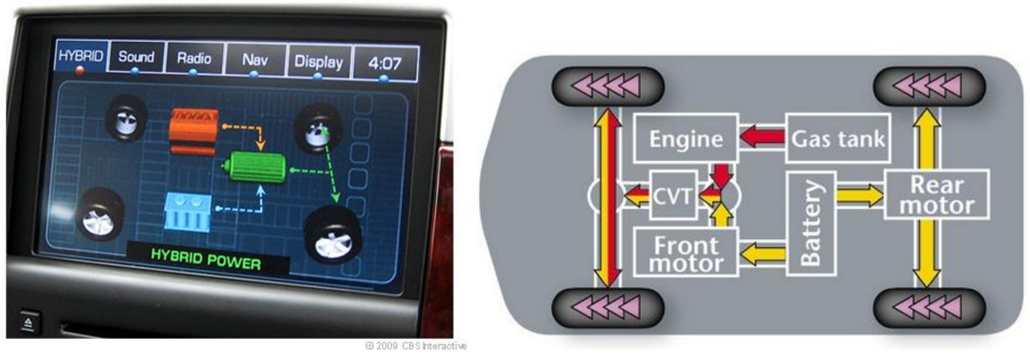


Figure 1: Power flow diagrams for hybrid electric vehicles.[5]

The displays incorporated in the solar power station show similar information, with the purpose of showing the user how the system is functioning.   Our product is unique in this sense and will hopefully stimulate green energy use and thinking for future engineers. Additionally, an up-to-date quantized display of the system power flow will show the users how much a given load is drawing (i.e. comparing a charging cell-phone to a laptop). This promotes power-usage awareness in a way that the average grid-tied distribution system does not.

Project Development

Early Stages

It was decided that the project would focus on the design of a solar powered charging station for cell phones and/or lap-tops. Several topologies and features were considered, and the best ideas from each were used in the final design. The first idea was a stand-alone cell phone charging station, like those commonly seen in theme parks and airports. Another idea was to build the same system, but instead to add USB ports to a table so cell phones could be charged while people were eating or working. Another design was for a beach-umbrella with the same technologies on a smaller scale, allowing small devices to be charged at the beach or around outdoor seating. The idea of adding an inverter to each of these designs was discussed, as it would provide the user with familiar AC power for sundry other small devices.

Some problems common to each design were the difficulties associated with assembling a functional prototype while still being able to apply the necessary time and energy to the electrical design. A portable solar beach umbrella with an AC outlet is a product which currently cannot be purchased, and initially seemed like a desirable project. However, after considering the difficulties of converting a store-bought umbrella, given the size of the components required for the design, it was decided to be infeasible. Even for the two non-portable installations, the size of the components which would be required (particularly the battery) sparked discussions about moving the energy conversion and storage elements to an electrical cabinet or storing them in the NECAMSID lab. The installation of these systems seemed too labor-intensive, it remained desirable to take measurements on all parts of the system and have them easily viewable by the user.

Grid Independence

One question in many of the designs was the decision to include a grid-tie, or to require the system to stand-alone. The grid-tie guarantees a constant delivery of power, even during times when the solar panel is not usable and the system battery is depleted. This could result from a pattern of particularly cloudy days, when solar irradiance is not abundant. It could also happen when the solar panel is obstructed by snow, or even if the system is heavily used at night. However, excluding the grid-tie would simplify the design, and allow the installation to be more portable. Despite the fact that the system is tied to a large solar panel, the absence of the grid-tie would provide the possibility for an outdoor installation in a remote location, such as in a campground or on hiking trails. Additionally, the system could operate without adding any strain to the electrical system of an RV or other automobile.

Rough estimates were made regarding the system's daily usage pattern, and the worst-case loading was calculated. It was decided that an energy storage capability of 500 watt-hours could operate with reasonable reliability in the absence of a grid-tie. Using rough estimates for the energy-storage consumer price[1,2], it was quickly decided that lithium-ion and nickel-metal hydride batteries would cost roughly $1250 and $1375 respectively. These implementations would also require a complex charge controller with the ability to monitor several cells. In contrast, 12 volt lead acid batteries with capacities in the 40 ampere-hour range are abundant, due to their demand in automotive, marine, and other vehicles. More information on battery selection can be found in the "System Components" section of this document.

Finalizing Goals

Given the considerations from the early stages of the project design, it was decided that the system would be a wall-mounted board with all the system components attached. This design shift came about for a few reasons. The primary reason is to attract more "customers". The purpose of this product is to promote green energy. A small display with an outlet built into the wall will not be very conspicuous. In contrast, having the entire project mounted on a board with all components and connections visible will make it easier to spot and, consequently, draw more attention.

People will be curious, approach the display, and find out what it does by plugging in their laptop or cell phone. Rather than having one small screen displaying limited information there are three screens showing the flow of power in the system. This will give the user a more logical idea of how the system works, from power generation to its finalized delivery into their device. Moreover, the fact that all components are visible will demonstrate its complexity and add an interesting factor.

In some of the original ideas the battery and inverter and all components would be stored away out of sight which means the user would just be plugging into what looks like a conventional outlet painted green. It would most likely be taken for granted as it would seem like an ordinary outlet. With this mobile display design the user can see the energy storage device, see the inverter and outlet wiring, and see how the charge is flowing into their phone or laptop (either from the battery or from the solar panel). This enables the user to build a connection with the display and appreciate that he or she is not drawing any power from the grid, but from this device built by students for students.

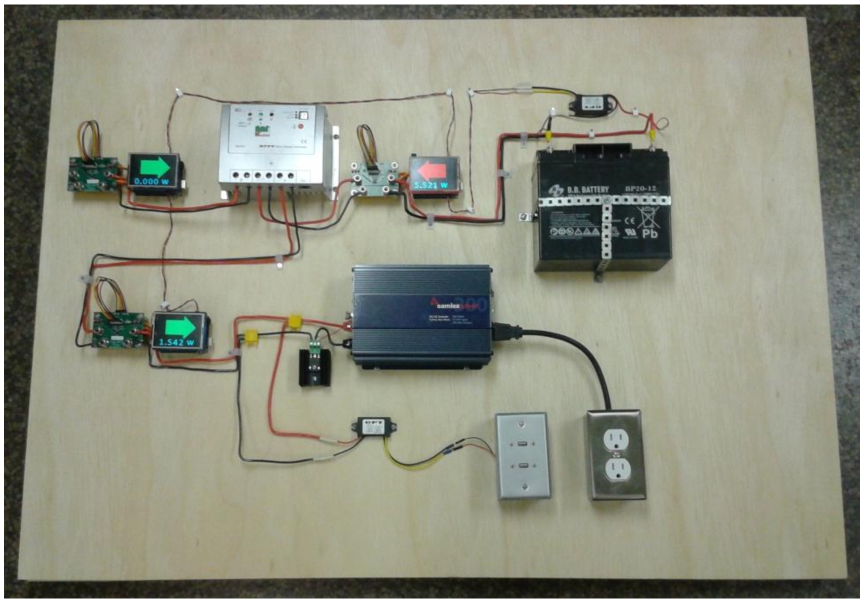


Figure 2: The Final Project. Higher resolution available in Appendix C

The mechanical design of this board (**Figure 2**) consists of a sturdy flat board containing the components fastened with the manufacturer's mounting holes. A clear Plexiglas cover will surround them for safety and theft control while allowing nothing to be hidden. The outlet portion will be positioned near an edge to allow for easy connections. One desirable quality of this design is that it allows for transferability. It can be placed anywhere and can function properly as long as you run solar panel wires down to its location. This makes for an eye-catching mobile display unit, which could be installed across the campus.

Project Description

Overview

An overview of the design’s behavior from a system-level perspective is shown in **Figure 3**.

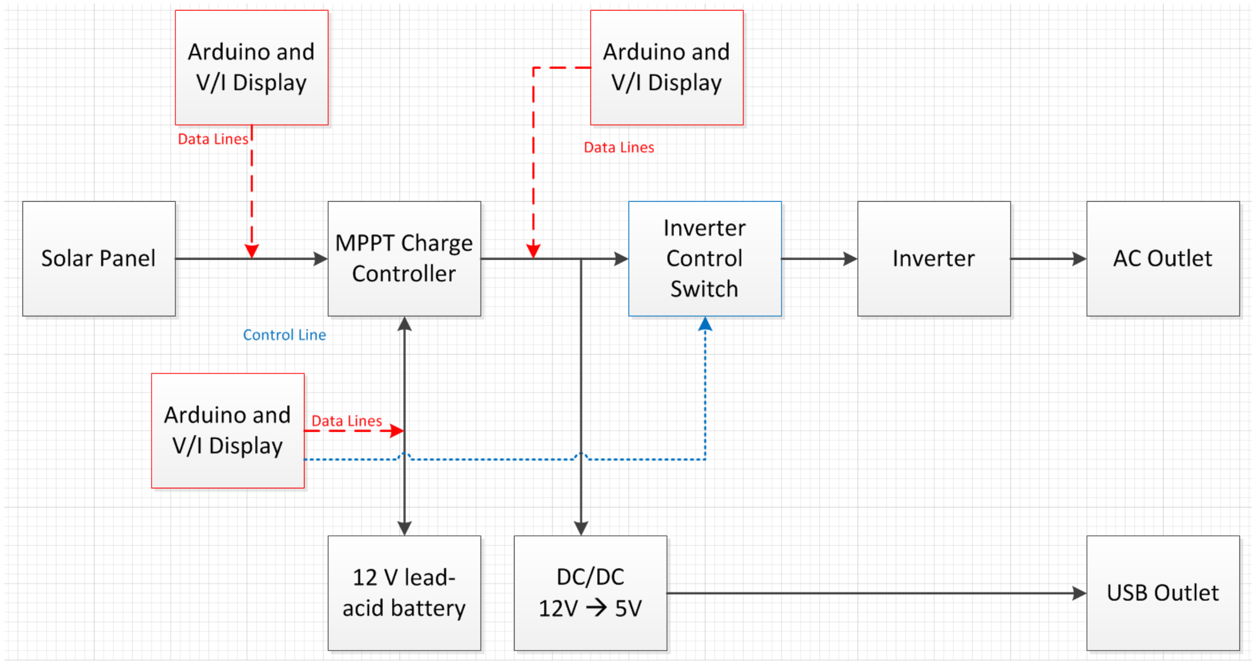


Figure 3: System level power flow chart with data-lines

Figure 3 shows that the general power flow is from the solar panel, through the charge controller, into the battery, into the inverter, then into the load. It should be noted that the charge controller is designed specifically for use in solar systems with lead-acid battery storage. It employs a maximum-power-point-tracking algorithm to idealize the power extraction from the solar panel. It also handles its own DC-to-DC conversation before its output, so it can properly charge lead-acid batteries.

Along with powering the inverter, the battery also supplies power to the DC/DC conversion for powering the microcontroller and display screen, along with the DC power for the USB charger. The chart also shows the data lines for the microcontroller. These data lines indicate the system connections at which the voltage and current are measured. This data is used in the display, and logged for analysis.

Solar Panel

The solar panels which were mounted on the roof of the Atwater Kent building in 2011 can be seen roughly in **Figure 4**, a picture taken facing north from the fourth floor of Salisbury Laboratories.

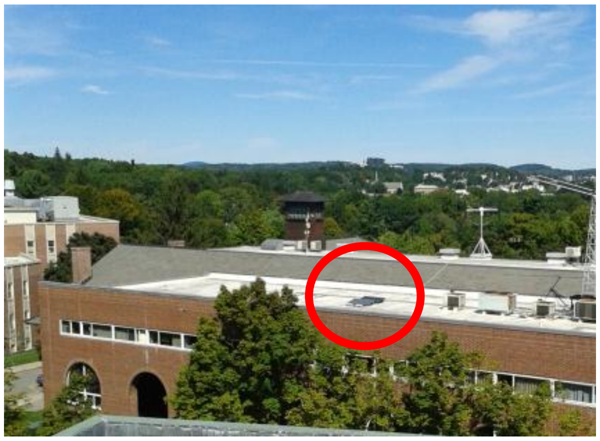
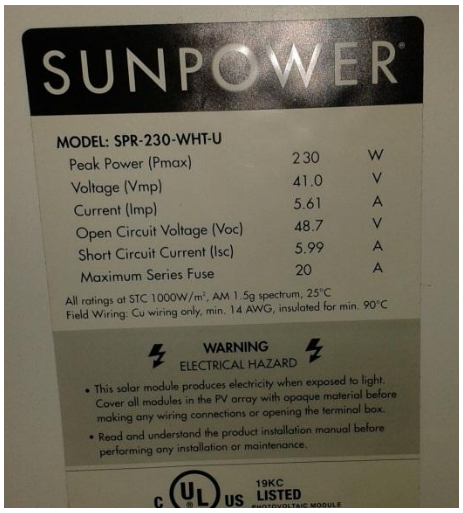


Figure 4: Picture of AK roof taken from Salisbury Labs



SUNPOWER®

MODEL: SPR-230-WHT-U

Peak Power (Pmax)	230	W
Voltage (Vmp)	41.0	V
Current (Imp)	5.61	A
Open Circuit Voltage (Voc)	48.7	V
Short Circuit Current (Isc)	5.99	A
Maximum Series Fuse	20	A

All ratings at STC 1000W/m², AM 1.5g spectrum, 25°C
Field Wiring: Cu wiring only, min. 14 AWG, insulated for min. 90°C

WARNING
ELECTRICAL HAZARD

- This solar module produces electricity when exposed to light. Cover all modules in the PV array with opaque material before making any wiring connections or opening the terminal box.
- Read and understand the product installation manual before performing any installation or maintenance.

c UL us 19KC LISTED PHOTOVOLTAIC MODULE

Figure 5: Faceplate of Solar Panel

An additional aerial photograph retrieved from Bing Maps can be seen in **Figure 6**. The photograph was taken before the solar panels were installed, but their location and wiring have been roughly annotated, so the location of the panels can be clear.

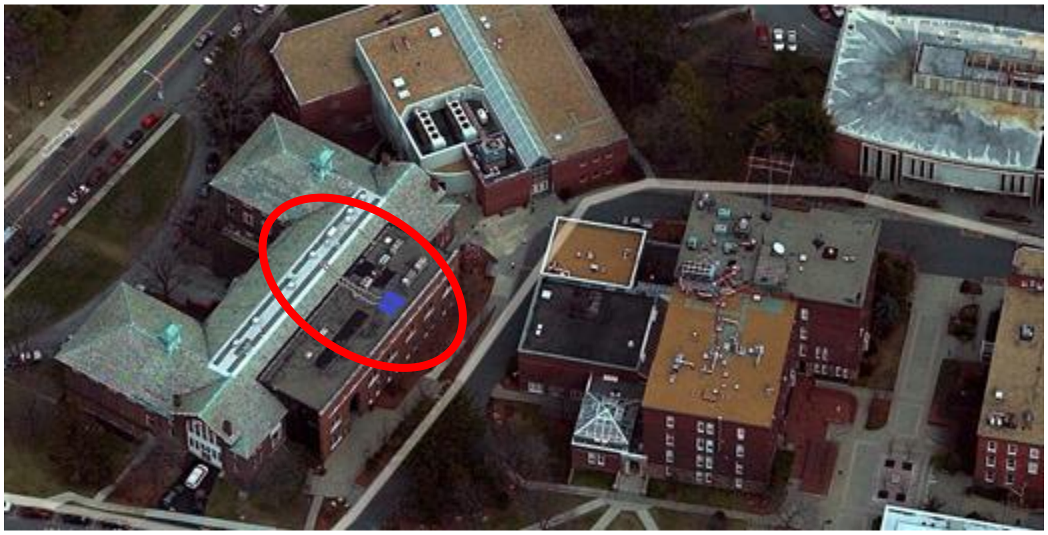


Figure 6: An Aerial Photograph of AK from the Southwest with Solar Panel Annotated

Charge Controller

Two specific features are required of the charge controller in this system. Maximum power point tracking on the solar panel input was a priority. Additionally, the charge controller is expected to be able to use its output to charge a lead-acid battery without overcharging. There were several different modules advertised to complete these tasks, varying in their price and documentation.

Maximum power point tracking, or MPPT, involves varying the voltage-current ratio used to load the solar panel to find (and remain focused on) the maximum amount of extractable power. The voltage-current relationship for a solar panel is downward-sloping. At either end of their relationship, power is zero – either a high voltage (V_{OC}) and no current, or a high current (I_{SC}) with no voltage. An approximation for the correct tradeoff between them is given on the solar panel, (I_{MP}=5.61 amps, V_{MP}=41.0 volts). However, due to varying solar irradiance levels throughout the year (and throughout the day, for that matter), these values do not always correspond to the maximum power point.

When a system employs MPPT, it can vary the voltage at which it operates. When it does so, it monitors the total power (by monitoring both the current and the voltage, and preforming a multiplication.) My constantly making small variations and monitoring their effect, the system can easily determine whether or not it is operating at the maximum power point. Once the proper voltage and current have been attained, the system (or a different one, depending on the design) can perform some kind of DC/DC conversion to bring the output to a desired level. In the case of this system, that voltage is about 12V (more during the bulk-charging phase for the battery.)

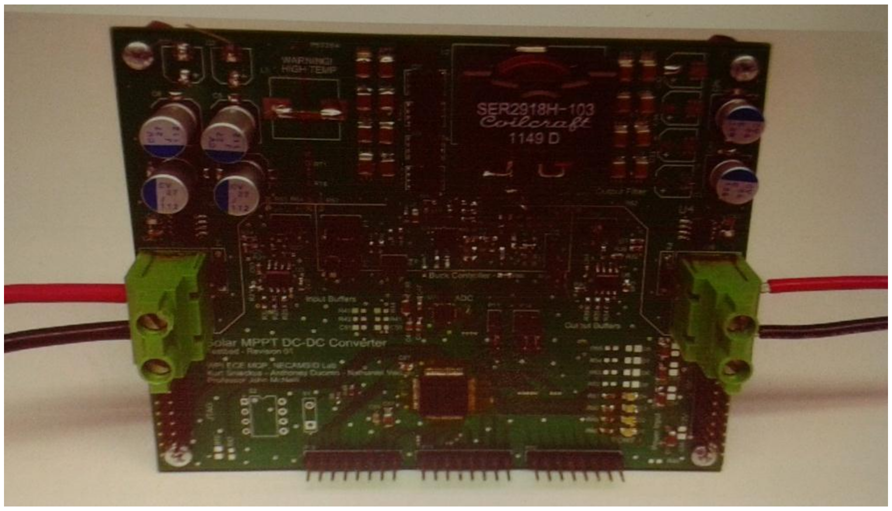


Figure 7: Result of "Renewable Energy Applications" Project

The “Renewable Energy Applications” MQP (**Figure 7**) resulted in a board which was designed to do exactly that task. However, after discussing the feasibility of implementing it in this project, it was decided that this device was not ready for use in this capacity. The focus of that project was on making the board dexterous and programmable, so it could accomplish many different tasks. Unfortunately, despite its flexibility and its many features, the project has been abandoned. Had it been pursued for commercial applications and large-scale production, it would be a very valuable tool to this project. However, little documentation regarding its use was made available, and it was not tested thoroughly. Additionally, only one was made, and it is not currently in Worcester. It became a worry that it may stop performing after a day or a week, and lack of documentation would make the problem very difficult to diagnose. Resultantly, a store-bought module built to complete the same tasks was selected.

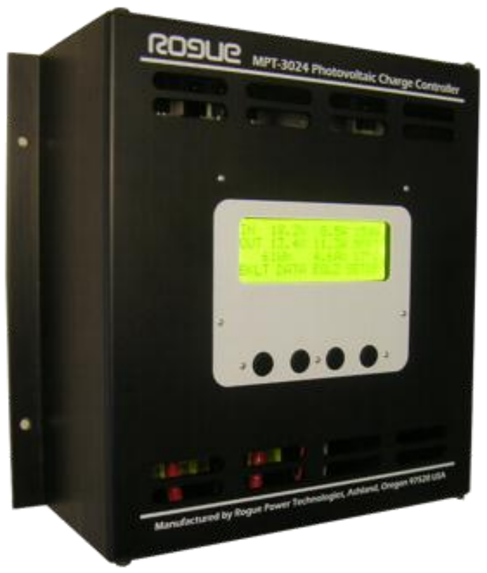

ROGUE
MPT-3024 Photovoltaic Charge Controller
Manufactured by Rogue Power Technologies, Ashland, Oregon 97520 USA

Figure 8: Rogue MPT-3024 Charge Controller

The Rogue MPT-3024 (**Figure 8**) is a charge controller for lead acid batteries which is designed to receive its energy from photovoltaic systems. It employs MPPT and has protection to keep the battery safe from overcharging. It is advertised to have high-efficiency internal DC/DC, has its own display, and logs 30 days of data. It employs temperature sensing to further protect the battery, and can communicate with PCs. While has some features which may have gone unused in this design, it would certainly do the job. Unfortunately, its cost was correspondingly high, at about $500. It was taken out of consideration when it went out of stock on the Rogue company website.

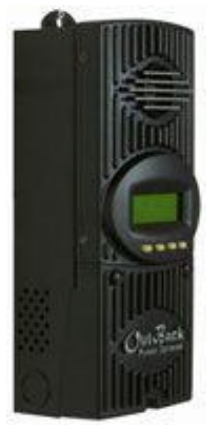


Figure 9: Outback Flex Max 60 MPPT Charge Controller

The Outback Flex Max 60 MPPT Charge Controller (**Figure 9**) has similar features to the Rouge machine. It also employs some data logging and has an LCD display. It is also compatible with 12V, 24V, 36V, 48V, and 60V lead acid battery arrays. Like the Rogue, its features (along with its price) are outside the scope of this design. Additionally, after reviewing the feasibility of adding this product to the design, the group decided that a charge controller with an LCD screen would distract from the displays already in the project. A simpler and cheaper MPPT controller was sought.

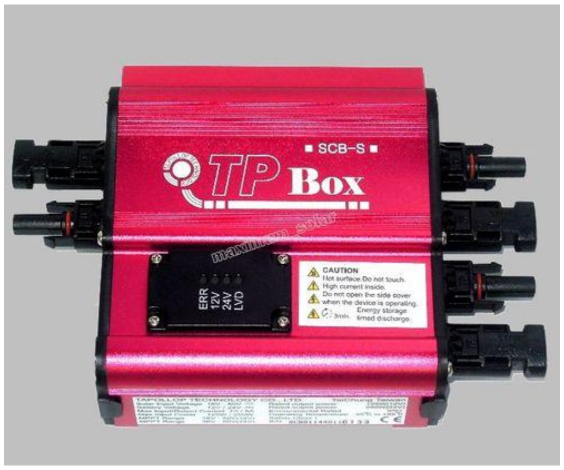


Figure 10: TP Box PCM-924 Charge Controller

The TP Box (**Figure 10**) is another MPPT charge controller designed for use with 12V or 24V lead acid batteries and solar panels with V_{OC}<80 V. It is less than one fourth the prices of the Rouge or Outback Power products, and boasts all the features required for this design. It does not have an LCD screen or any data-logging of its own, but does have output (in the form of an LED light) to describe which phase of charging the battery is in. Additionally, its outputs keep the load and the battery separate, allowing it to shut off the load if the battery's charge level falls too low. Documentation for this product is in-depth, and widely available.

For a period of time, the TP Box had been selected to be used in this system. However, it was later discovered that its power rating was for a 24V system (rather than 12), and that its output current limit is the same for both operation modes. Resultantly, its effective power rating for our system was not 250W as advertised, but rather 125W. For that reason, a higher-capacity alternative was sought. The Tracer X line of charge controllers from EPSolar Photovoltaic Co. seemed a reasonable substitute. They employ MPPT, and are compatible with 12-volt systems. According to the user's manual, they are primarily used in lighting systems, and come with some firmware which employs daylight tracking. This allows the device to be configured to enable output to the load for some time-periods surrounding sunrise and sunset. This feature was undesirable in the system, which should have its load output enabled at all times. A phone call with EPSolar revealed that one of the configuration settings for the Tracer (specifically setting #17) instructs the device to always keep the load enabled. The Tracer-2210RN (**Figure 11**) can output up to 20A, and cost $129 including shipping on Ebay (during winter 2012).

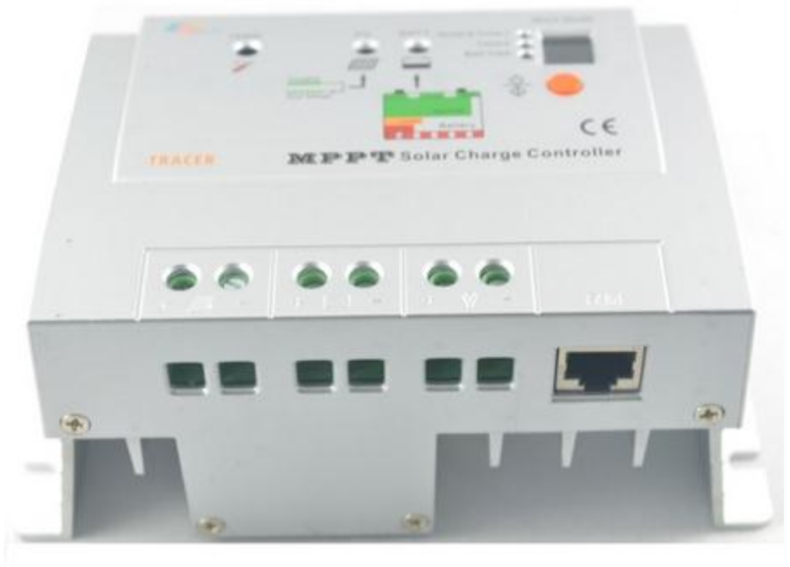


Figure 11: Tracer 2210RN Solar Charge Controller

Battery

Many types of rechargeable batteries could be used in this design. They differ in implementation-simplicity, capacity, cost, and life-expectancy. The three tables below show a value analysis preformed over four different battery chemistries.

Table 1: Value Analysis for Battery Chemistries

Battery Cemistry	Joules/gram	Joules/$	Cost for 12 V, 10 Amp-hour	Life	Difficulty (x/10)
Lithium-Ion	733	4156	$95.99	400-1200 cycles	8
Lead-Acid	119	15456	$27.95	500-800 cycle	4
Nickle-Metal-Hydride	161	9900	$180.00	500-1000 cycles	8 (will need to manage cells)
Nickle-Cadmium	180	231	$232.91	2000	10 (needs deep discharge!)

Rating	Joules/gram	Joules/$	Cost for 12 V, 10 Amp-hour	Life	Difficulty (x/10)
5	200+	>13k	<25	>1000	<3
4	170-200	10k-13k	25-50	500-1000	4-6
3	140-170	7k-10k	50-100	200-500	6-7
2	110-140	4k-7k	100-200	50-200	8-9
1	<110	<4k	>200	<50	10
Category Weight (x/100)	40	80	80	70	90

Battery Cemistry	Joules/gram	Joules/$	Cost for 12 V, 10 Amp-hour	Life	Difficulty (x/10)	SCORE
Lithium-Ion	5	5	3	4	2	**3.61**
Lead-Acid	2	5	4	4	4	**4.00**
Nickle-Metal-Hydride	3	3	2	4	2	**2.72**
Nickle-Cadmium	4	1	1	5	1	**2.11**

The first table above show data gathered for different battery chemistries converted to comparable units. The second table quantizes ranges by desirability, with 5 being the most favorable, and 1 being the least favorable. The bottom of the second table gives a weighting for each category, on a scale from 0-100, quantizing its importance. The bottom table shows the quantization of data from the first table using the ranges defined in the second table. At the right, the score for each battery chemistry given their quantizations and score weightings.

The conclusion is that a lead-acid battery is the proper storage device for our design. If this is determined to be not possible, the next most desirable option is a lithium-ion battery. The most attractive aspect of the lead acid battery is its comparably low price and implementation ease, even as capacity scales upward. Multi-cell lead-acid storage systems can be internally connected without the need for individual cell-monitoring. They are also manufactured on a large scale, for use in automotive and marine electronics.

Additional thought was given to the normal usage pattern for this implementation. It is intended that the battery installed in this system could sustain normal daily usage with no sunlight (resulting from snow-cover or a prevailing storm front) for a few days. However, the average usage will include a daily recharge. Therefore, the battery will seldom see a deep-discharge. This usage would not work effectively with a nickel-cadmium battery because of its pronounced memory-effect.

Because lead-acid batteries can be purchased in a variety of shapes and capacities, consideration was given to the decision of which to buy. Having only decided the capacity-demand of 40 amp-hours at 12 volts, many decisions remained. Lighting-backup batteries designed for wall-mount seemed ideal, but their capacities were undesirably low. Although a multi-battery system could continue to make use of the flat battery, the difficulties of designing a management circuit for that topology made the idea undesirable.

Further research revealed that 12 volt lead acid batteries around the target capacity are produced in bulk for automotive, marine, and sundry other transportation means. Those batteries overwhelmingly adhere to being roughly cubic, with two posts for connection. This topology is not ideal for the wall-mount board design, so the selected battery was compromised to be 20 amp-hour sealed lead acid battery belonging to Professor Bitar, the project advisor. Dimensions for the edges visible in **Figure 11** are roughly 17cm by 28cm. The thickness (which made it attractive for the wall-mount design topology) is only 7 cm. There is also a second one in the NECAMSID lab.

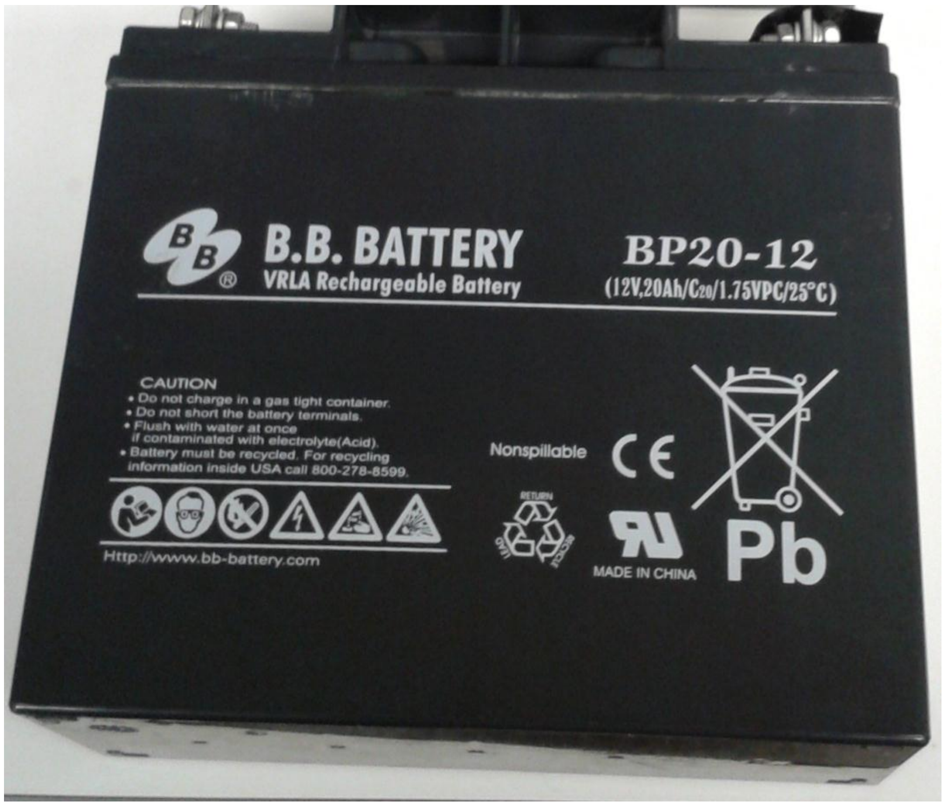

B.B. BATTERY
VRLA Rechargeable Battery
BP20-12
(12V,20Ah/C20/1.75VPC/25°C)
CAUTION
• Do not charge in a gas tight container.
• Do not short the battery terminals.
• Flush with water at once if contaminated with electrolyte(Acid).
• Battery must be recycled. For recycling information inside USA call 800-278-8599.
Http://www.bb-battery.com
Nonspillable
CE
MADE IN CHINA
Pb

Figure 12: BP20-12 Sealed Lead Acid Battery

Inverter

12 V DC to 120V$_{RMS}$, 60 Hz, inverters have become popular in recent years, because of their ability to convert energy from an automotive battery and/or alternator into the familiar AC power which runs many devices in the home and office. Despite their abundance, some key features proved to be the deciding factor the inverter-selection process for this design. One main feature is the sine-wave type. Cheaper inverters produce a modified sine-wave (**Figure 13)**, which is a less clean wave form.

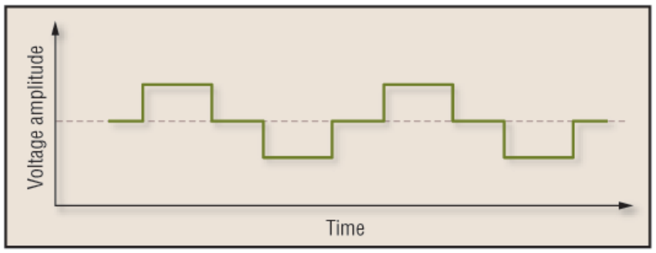


Figure 13: Modified Sine Wave Plot[3] (3-level)

Conversely to the modified sine wave plot, a true-sine wave is considerably freer of harmonics. Though the price of the modified sine wave inverters is attractive, they are known to have compatibility issues with certain appliances, including laptop chargers[4]. Because laptops are directly in the target appliance of this system, it was decided that the inverter must be a pure-sine wave generator.

It was decided that the inverter's power rating did not need to exceed 300W, but should be greater than 150W. This way it is rated to handle almost any laptop, and will employ its own safety mechanisms if a larger appliance (such as a vacuum cleaner) is mistaken plugged into this lower-energy system. Although the inverter is operated with a control-switch driven by one of the Arduinos, protection and sectionalization features were key in the selection process for the inverter.

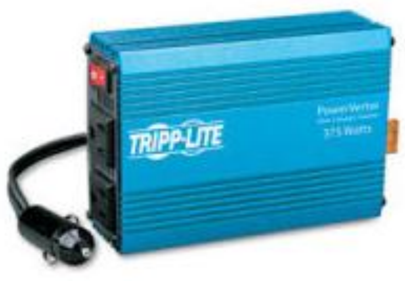


Figure 14: Tripp-Lite Inverter

The Tripplite inverter was attractive because of its high instantaneous power rating (600W) and its acceptance of automotive fuses (yellow tab at the right of **Figure 14**). However, its lack of mounting hardware and its lack of additional features made it less attractive as more inverters were researched.

Figure 15: PowerBright PureSine Inverter

The PowerBright PureSine inverter (**Figure 15**) is a pure sine wave generator rated to operate continuously at 300 watts when needed. It has a built-in cooling fan, and is designed to work with a lead-acid battery. One user-review online described it as "quiet" compared to other inverters. It is advertised to have an "overload indicator", but does not detail how the indication is done. Its cost is higher than the Tripplite, but it does have more features.

Figure 16: Samlex PST-30S-12A Inverter

The Samlex 300W inverter (**Figure 16**) is a pure-sine wave generator with mounting hardware. This made it attractive for the wall-mounted design of this project. Additionally, it comes with a 2-year warrantee, and can operate between 10.5 V and 16.5 V on its input. It executes its own low-battery shutdown, and audible overload-alarm. It also features an internal-temperature-controlled cooling fan.

It can also take input from 24 V or 48 V battery systems, so if the board is ever upgraded it can continue to function seamlessly. These features are desirable, because they add sectionalization and protection to the system. The Samlex PST-30S-12A is the inverter for this system.

Inverter Power Switch Circuit

Due to the energy storage capabilities of this system, it will not be able to provide power to the AC outlets twenty four hours a day. For this reason we must be able to disable the inverter that powers the outlets. This task is accomplished by inserting a low-side MOSFET switch in the middle of the inverters negative return path to ground. When the switch is an open circuit the inverter will not function effectively disabling the outlets to conserve the power in the battery to ensure constant USB operation during the night and cloudy days. A voltage suppression diode is located parallel to the inverter to provide a safe means of routing a transient surge in voltage due to the inverter suddenly being disconnected. Refer to the following figure.

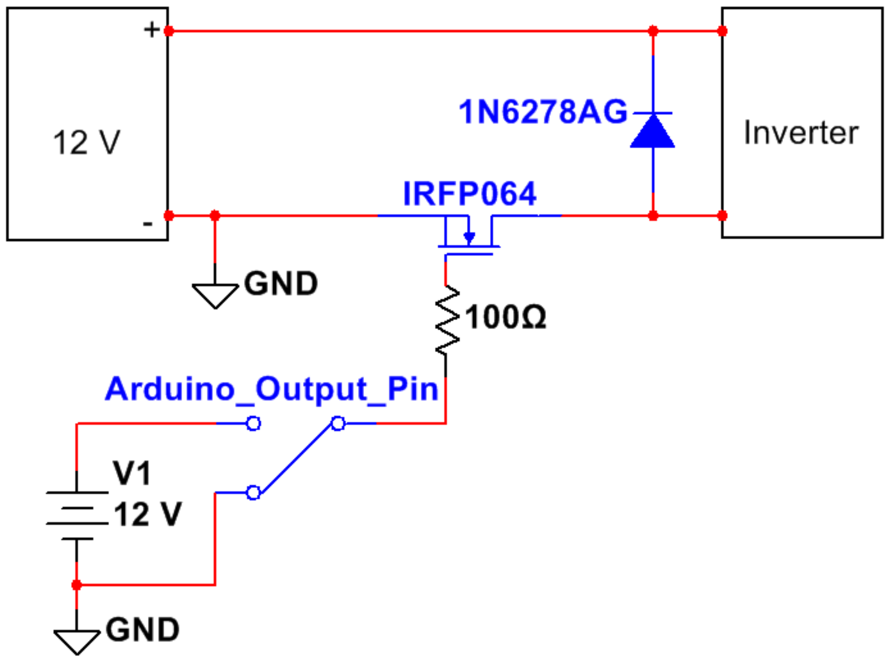


Figure 17: Equivalent circuit for inverter controller.

Selecting a Transient Voltage Suppression Diode

When the inverter is disconnected, the current in the inductors inside the inverter might not be at 0A, a likely situation. Rather than spend resources on a zero crossing detector a Transient Voltage Suppression (TVS) diode is used as shown in **Figure 18.** Since the voltage across an inductor is proportional to the inductance and the change in current over the change in time, if there's a large current through the inductor and the inverter is powered off a large transient voltage would be created and possibly damage the system. With the diode in place, the voltage can quickly settle down to zero and the diode will dissipate all the power safely.

The diode will need to have a reverse breakdown voltage greater than 12V so it stays reverse biased during normal operation. Secondly, it needs to handle the large voltage transients and currents that could be created. The peak current was estimated to be around 60A double the worst case scenario, and the peak voltage was unknown so a diode was chosen with a very high voltage rating. This led us to the following characteristics; a reverse breakdown voltage of 19 – 21V, a maximum peak pulse current of 54.2A and a maximum voltage rating of 231V. Price and availability also weighed in on this decision and the part settled on is the Vishay 1N6278AG.

Figure 18: Diode package style for the TVS protection.

Selecting a MOSFET

The MOSFET selection was based heavily on heat characteristics. First the selection of PMOS or NMOS was made. This was an easy selection since we can only drive the gate of the MOSFET with a positive voltage from an Arduino control pin so an NMOS was selected. Next, it had to be ensured that the Arduino could drive the FET into saturation so the threshold voltage must be around 2 to 3 volts typical. Finally, we had to ensure the MOSFET could operate properly at the worst case power estimations.

To estimate the worst case power that the FET needed to handle the following graph from the datasheet of the IRFP064 was used.[7]

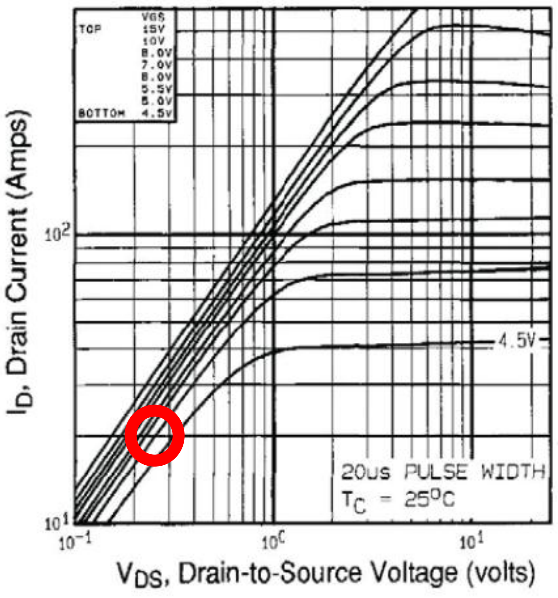


Figure 19: Graph of $r_{ds(on)}$ as a function of temperature with a drain current of 130A.

It is known that the V_{GS} of the FET will be 5 volts since that is what the Arduino outputs so the second curve is used. It is also known that the worst case drain current will be 20A, circled in **Figure 19.** This shows that the drain to source voltage will be around 1.5 volts so multiplying these two values together yields the worst case power that the MOSFET needs to handle, 30W.

$$P_{DS} = V_{DS} * I_{DS} = 20A * 1.5V = 30W$$

Next the thermal resistivity properties of the device were recorded and shown below.

$$R_{thJA} = 40\ \frac{°C}{W}$$

$$R_{thCS} = .24\frac{°C}{W}$$

$$R_{thJC} = .5\frac{°C}{W}$$

$$Max\ Junction\ Temperature = 175\ °C$$

First it was calculated to see if the device could operate reliably on its own with no heat sink. An equivalent circuit is shown below.

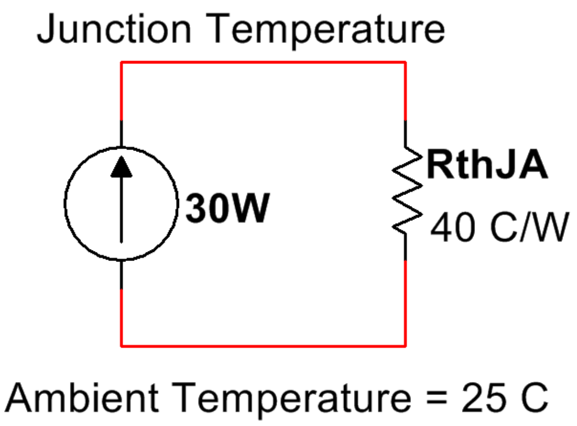


Figure 20: Equivalent thermal circuit for analysis.

To calculate the temperature the junction would be at if no heat sink was used and it dissipated 120 W is shown below.

$$30W * 40\frac{°C}{W} = 1200°C$$

That means the junction temperature would be at 1200 degrees Celsius! Clearly that would burn out the device immediately and render the switch useless. Rather than putting eight of them in parallel we can calculate the thermal resistance a heat sink would need to have in order to use just one transistor saving money, space, and time.

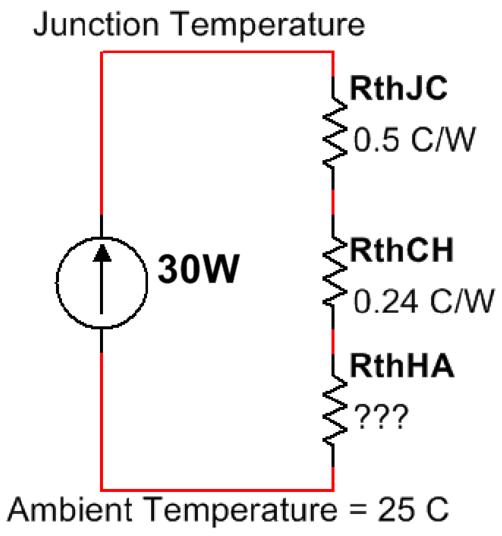


Figure 21: Thermal equivalent circuit used to calculate heatsink properties.

RthJC is the thermal resistivity from the junction to the case of the IC and RthCH is the thermal resistivity from the case to the heatsink (typical value). The thermal resistivity from the heatsink to ambient is what needs to be calculated. Limiting the junction temperature to 170 degrees Celsius to be safe (max 175 degrees Celsius) calculating the maximum value for RthHA is trivial, shown below.

$$30W * \left(0.5\frac{°C}{W} + 0.24\frac{°C}{W} + R_{thHA}\right) = 170°C - 25°C \therefore R_{thHA} \leq 4.09\frac{°C}{W}$$

Any heat sink with a thermal resistivity of 4.09 degrees C per watt or lower will ensure the junction temperature will stay under 170 degrees. Now the only other stipulation of the heatsink is that is connected to the transistor via a flat greased surface. This means that the heatsink must be flat and thermal paste must be used to attach the heatsink to the transistor back. Below is a picture of the heatsink selected.

Figure 22: Heatsink to ensure MOSFET switch doesn't burn out.

It has a part number of FA-T220-51E[6] and a thermal resistivity of 3.4 degrees Celsius per watt. This means at worst case load, the junction of the MOSFET will reach temperatures of 150 degrees Celsius, ensuring we never exceed the max case rating of 175 degrees of the chip.

Current Sensing

Background

The uniqueness of this project lies in the display system showing the power flow of the system. To show power in the unit of Watts from the solar panel, battery, and load, two parameters need to be known; the voltage times the current need to be multiplied together to calculate the instantaneous power. Measuring voltage is very easy in electrical systems as we can simply send out electrical connections to our microcontroller which can read them as an analog voltage. Sensing current proves to be a much more challenging task, and much research was done on different sensors and topologies for current measurement.

Methods

Sense Resistor

This method involves placing a low value resistor in series with the path of current to be measured. The voltage across the resistor is then measured by a microcontroller and knowing the resistance and voltage the current can be calculated very simply.

This current measuring topology has the key advantage of being very simple to implement as it just takes one component, the resistor. Connecting the resistor to the microcontroller can be difficult though. This is because the sense resistor chosen usually has a very small value to waste less power as heat. Some sense resistors in milliohms are not uncommon. This means that the connections to the microcontroller must be very low resistance or it can affect the measurement of current greatly. Sometimes, something called a kelvin connection is used shown below.

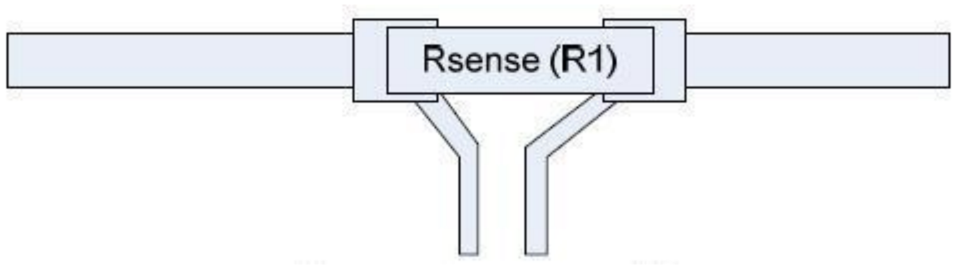


Figure 23: The two wires on the left make the physical connection and the lower ones sense the current accurately.

Here with four terminals resistance from the wires and leads can be eliminated as one set of connections measures the current, and the other pair measures the voltage across the resistor. This makes the current in the resistor independent of the leads and the voltage measurement draws no current so the measurement is more accurate.

Aside from being hard to interface with, a sense resistor has a few other disadvantages for our system application. In high current applications such as this system a high power rated resistor is needed. The projected high case current in our system is 20A which would mean the resistor would need to be massive. This can take up valuable space on our display board. Moreover, more power would be wasted as heat which is not ideal for our system whose purpose is to promote the reduction of energy. Finally, resistance value can vary with temperature greatly which is bad for this project since an accurate current measurement is key to making this display useful.

Hall-effect Sensor

The hall-effect method for measuring current involves measuring the magnetic field being produced by a current carrier which is proportional to the current. This can then be turned into a current value as the sensor acts as a transducer making a voltage proportional to the current. This method is a little more complicated and expensive compared to the sense resistor, but suits the project more.

This method is considered noninvasive unlike the sense resistor as on does not need to disrupt any cabling or insert components. Generally, the wire carrying the desired current to be sensed is passed through the sensor. This allows for the device to be electrically isolated from the carrier. Additionally, Hall Effect sensors are much friendlier to very high current unlike the series sense resistor. This will save our system power as up to 20A could be seen at any one time which won't be dissipated as heat like in the resistor.

The hall-effect sensor does come with drawbacks. As mentioned earlier the sensor is more expensive than most sense resistors. Furthermore, they can be inaccurate at lower current. This shouldn't be a problem for our system though as whenever there is at least one load connected, the current will be large and therefore accurate. For these reasons we decided to go with the hall-effect sensor to measure current in our design.

ACS712

The Hall Effect sensor chosen for our system is ACS712 from Allegro Microsystems. It is a standard SOIC-8 package. This integrated circuit contains the hall-effect sensor itself and an included circuit to convert this into a proportional voltage. For this reason our system is using the ASEK712ELC circuit board (an evaluation board from Allegro Microsystems for the ACS712) which includes screw-terminal connectors for the power line being measured, along with headers to provide power and transmit the sensor signal.

The circuit boards can measure current from -30 to +30 Amps, and the output indicates the flow direction. Sensitivities from 66 mV/A to 185 mV/A are available in the evaluation boards, and are dependent on the gain in some analog amplification on the board. This varied current rating allowed increased measurement accuracy for smaller currents (where applicable), while still allowing other high-current measurements to be done where needed.

Operation

The operation of the circuit is relatively simple. The following figure shows the evaluation board that is being used. It is a positive that it is large because it will be easily discernible to onlookers of the project, and help show power flow as one can "follow" the current through the sensor to load and see it update on the screens in real time.

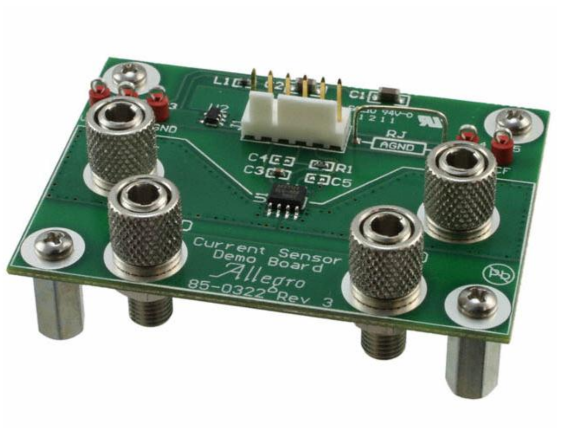
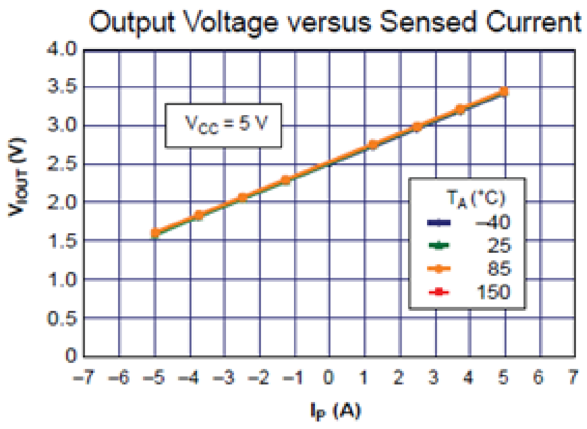


Figure 24: The current sensing hall-effect board.

The two large cylindrical connectors in the middle of the board are where the current desired to be sensed is routed through. The large wide traces can be seen on either side of the SOIC from these connectors. This allows it to handle up to 30A acting as a heatsink. These traces are routed through the hall-effect sensor located within the IC from allegro. As the current flows through this the magnetic field is measured and converted to a voltage directly proportional to the current in line with the following graph. We can then send this voltage into the microcontroller to do the proper math to output the appropriate current. It should be noted that depending on the board model, that the proportionality constant changes. It has three discrete values, either 66mV/A, 100mV/A or 185mV/A. The microcontroller can easily account for these differences with math depending on the model used.



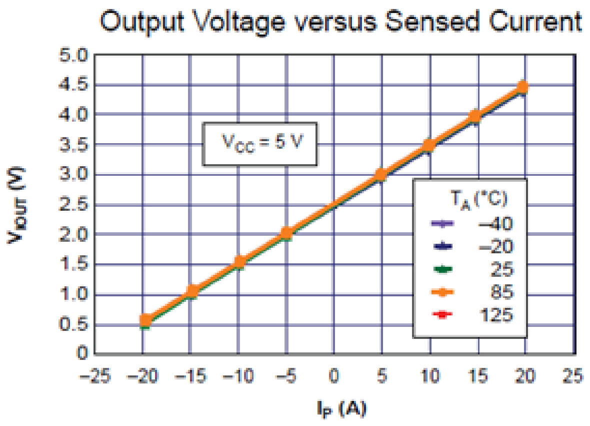


Figure 25: The left shows the sensitivity for the ±5A and the right is for the ±20A sensor.

Doing a quick slope calculation it is easy to find the sensitivity of the sensor from the datasheet. The left one corresponds to the 185mV/A sensor as that is the slope of the line, and the right graph has a slope of 100mV/A.

A very important aspect of these sensors is that 0A is centered around 2.5 volts. This enables the sensor to not only sense magnitude but also direction of the current. This is crucial for the project as the direction of certain current paths can be changing and need to be reflected on the screens (such as current flowing into the battery to charge it, or out of the battery to supply a load during the night). If a voltage measured from the sensor is less than 2.5 volts than we can multiply it by the sensitivity of the device and know it is negative. If greater than 2.5 volts then it is obviously positive.

Accuracy of the sensor

The hall effect sensor is very accurate, even against temperature. The plot below indicates sensitivities of the 20A sensor over the sensor-range, for four different operating temperatures. Sensing 0A at 150 degrees Celsius leaves a sensitivity of 99.5 mV/A instead of 100mV/A, which is an error of less than 1%.

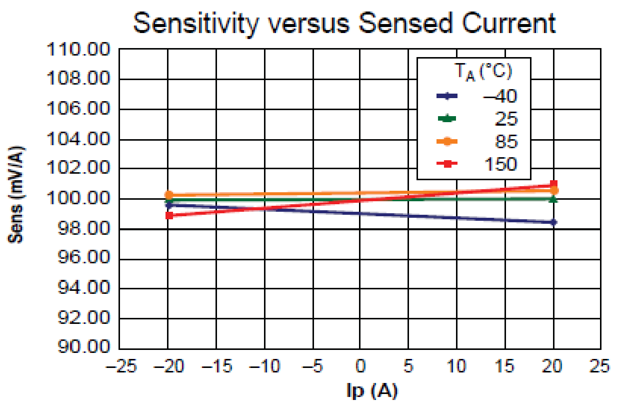


Figure 26: Sensitivity for the 20A sensor against different current values and temperatures.

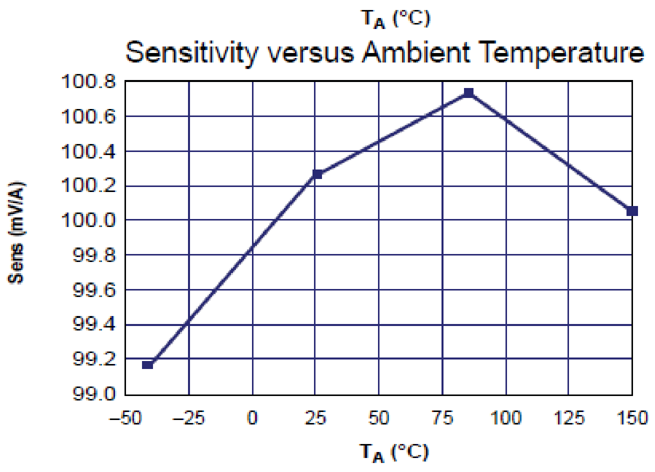
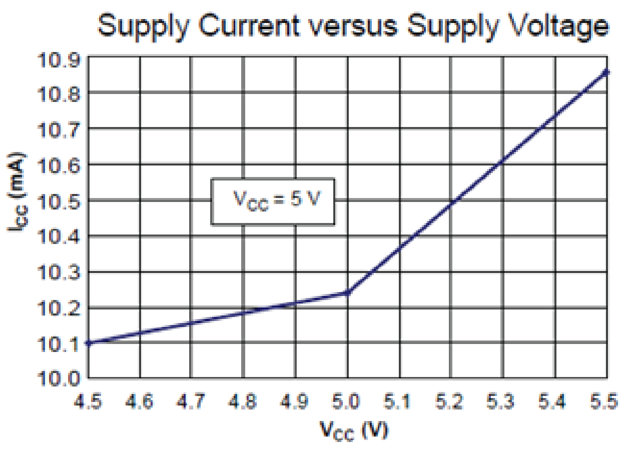


Figure 27: Sensitivity of the 20A sensor across temperature.

This graph again demonstrates how accurate this sensor is. At first it looks poor across temperature, however the y-axis is only changing by 1.8mV/A (a negligible amount for this application). Although the measurements lose some accuracy, it is not a bother to the user that the display reads “10.18 W” instead of “10.00 W”, because the display screens aren’t meant to be a laboratory-quality measurement instrument. Thus, a small offset does not detract from the purpose of the system. Power flow will still be shown and the user will get a feel for how much power is being conserved by taking it from the sun or the battery.

Power wasted versus ideal sense resistors

Using worst case load values the power wasted in the hall effect method versus a sense resistor method are compared. The worst case load that needs to be measured is 20 amperes.



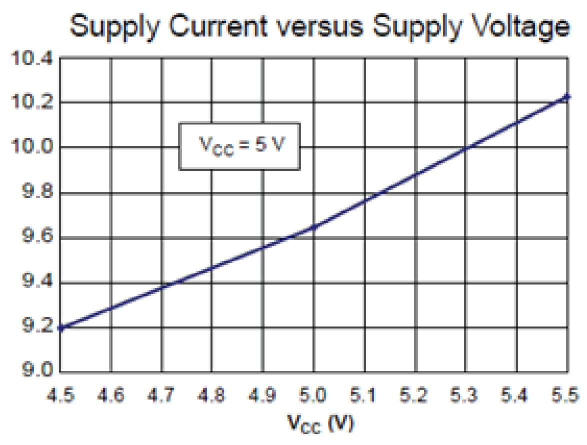


Figure 28: Left is the supply current for a 5A sensor and on the right is the supply current for the 20A sensor.

According to datasheet graphs from Allegro the hall-effect boards will consume around 10.5mA worst case when running off of a 5 volt supply rail.

$$P_{hall} = 5V * 10.5mA = 52.5mW$$

This is a small amount of power used at worst case loads. The great thing about the hall-effect sensor is it dissipates approximately the same power to measure 20A and 2A.

Now looking at reasonably sized power transistors capable of handling 20A the prices and resistor value rise quickly. A quick search yielded a 100mOhm 20W power resistor.[8] At 20A this would only see 2W which is fine, but the power wasted is about 40 times greater than the hall-effect board. For a project that focuses on saving energy, the current sense topology would not be ideal.

Current Topology Conclusion

For this project a hall-effect sensor was chosen as the proper method for measuring current. It uses a considerable less amount of power during loading conditions compared to alternative methods and has reasonable accuracy suitable for this application.

The hall-effect sensor has 2 terminals in which the current to be measures passes through, and two terminals that output a voltage proportional to the measured current. The board needs to be powered as well and current is ready to be measured. It has one more connection than a sense resistor, but wastes much less power across all loads that are nonzero. As current increases, the power wasted in the chip does not increase as load increases unlike the power resistor topology. It is easy to implement and accurate enough to give reasonable measurements to communicate through the display screen as well as be accurate across different temperatures and loading conditions.

Using the Sensor in the System

This section details the circuit and operation of the sensor in the project. There are seven connections from the current sensor board between inputs and outputs, and some are used solely for cosmetic reasons.

The important input connections are *IP+* and *IP+GND* which are the current entrance and return path. The pins *5V* and *AGND* is what power the sensor which receive 5 volts from the DC/DC converter and its ground respectively. The output pins *Vout* and *Vsns* are the two signal wires that let the Arduino microcontroller calculate the power in watts. *Vout* is proportional to the current and *Vsns* is proportional to the voltage. The microcontroller then multiplies these together to get a power reading. The *Vsns* pin was artificially created on a circuit board located underneath the Allegro current sensor which allows for easy and neat connections from the sensor to the microcontroller. Finally, the output *IP-GND* is strictly for cosmetic reasons as it is shorted to *IP+GND* internally by the current sensor board. The only reason this wire is included is to make the layout more understanding to the reader. As presented this allows the user to follow the current plus path, as well as its return path much easier. The following shows an example of the setup of the project.

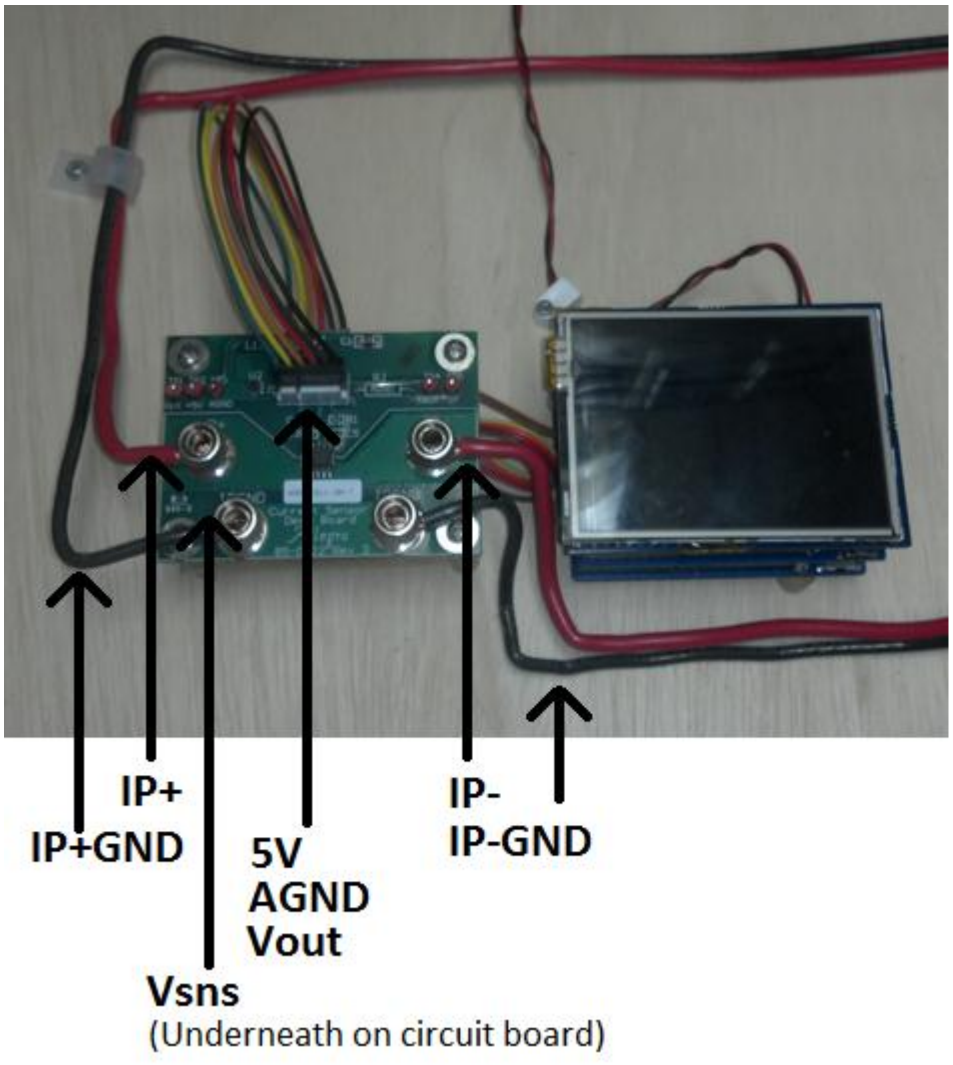


Figure 29: Labeled construction of the current sensor board.

Notice how it is immediately clear where the current is flowing. Following the red wire, one can clearly see it passes through the connector, through the IC which is measuring it, and back out through the right connector. Additionally, the return path is right alongside it making it easy to see the current loop. The other signals exist underneath the board from connectors going from the top to the bottom of the circuit board not shown. These then get routed across to the Arduino circuit board located on the right. The following figure represents the schematic for the circuit board beneath the current sensor.

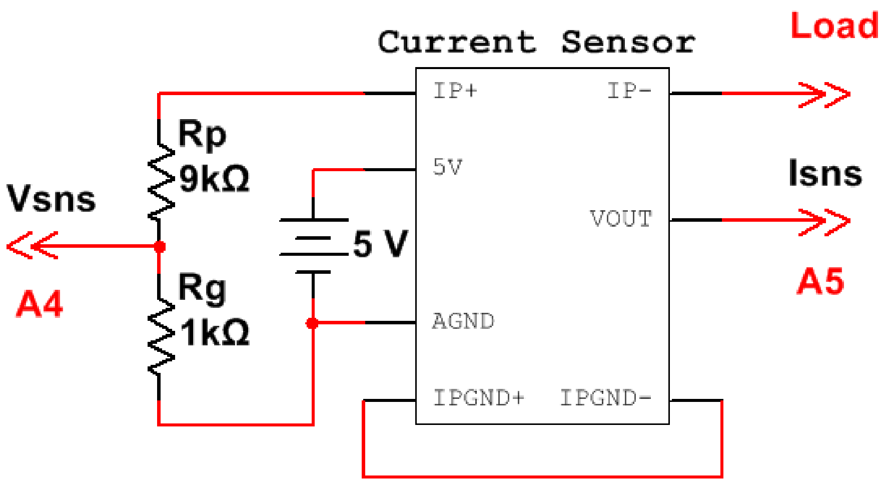


Figure 30: Soldered circuit board schematic that the current sensor board is mounted to.

The voltage is divided by some scalar depending on what current is being measured and this signal goes to pin A4 on the Arduino. For the solar panel, a divide by ten voltage divider is used because the max voltage input to an Arduino pin is 5 volts and the max voltage from the solar panel is 50 volts. Dividing this ensures that this limit is never exceeded. The software simply multiplies the voltage by ten to return an accurate measurement. The other two sensors implore divide by 3 circuits to keep the voltage within 5 volts max.

The sensor outputs a voltage proportionally to the measured current from IP+ to IP- which is routed to the Arduino pin A5 which is then scaled according to the current sensor and then multiplied by the voltage from Vsns to output an accurate power usage in watts.

Outlets

At the lower-right corner of the wall-mounted board there is two outlet boxes. Their design is similar to those seen on concrete or metal structures, with the ground-box exposed (as shown in **Figure 31**). Two standard AC outlets are included, both powered by the Samlex inverter. They share a ground connection with the earth-ground wire in AK, and the negative terminal of the solar panel on the input.

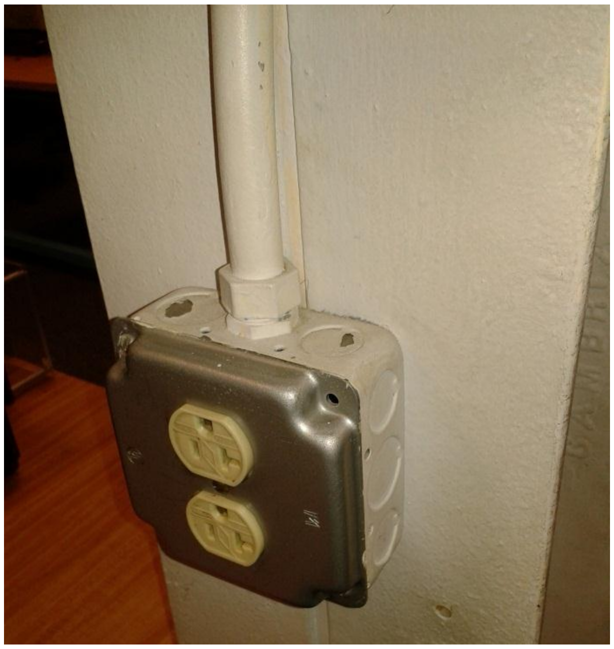


Figure 31: Two standard AC outlets, installed with the box exposed

The USB ports are mounted on a similar box, next to the AC outlet box. A single distributor for these style face-plates has been identified. A company called DataPro manufactures the plate shown in **Figure 32**. It should be noted that the female USB ports and connector screws are not sold with the face-plate, but must be selected separately on the DataPro website.

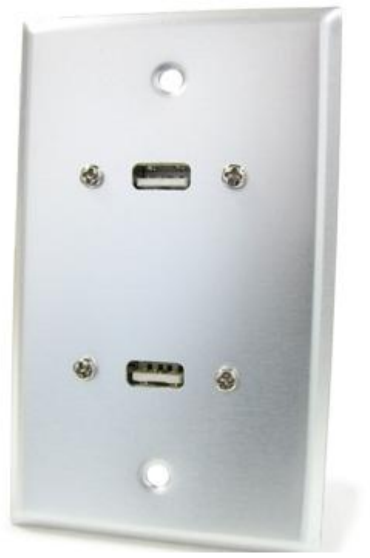


Figure 32: USB Faceplate

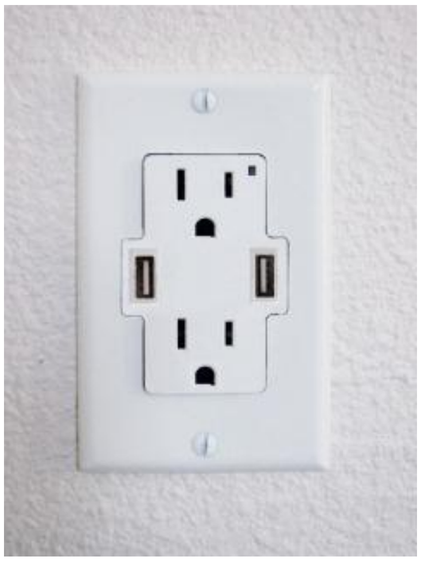


Figure 33: AC Faceplate with USB Outlet Included

It had been the intention of this project to include an outlet similar to the one shown in **Figure 33**. Unfortunately, every commercially produced AC outlet with USB connectors which could be found online at the time of our research includes its own rectification and step-down to 5V. The option to supply it with a separate 5 volt power bus is not given. This is likely because the average house does not have a 5-volt wire run, and it is cheaper to include the energy conversion in the outlet. However, because this project focuses on efficiency, the job is better down by the DC/DC module. This circumvents the alternative path, from the battery's DC to the inverter's AC and back to DC again for the USB outlet.

DC/DC converter

The Universal Serial Bus (USB) port is a 5 volt direct current device. Seeing how the voltage coming from the solar panel is variable and the voltage supplied by the battery is around 12 volts nominal we needed to step down this voltage to 5 volts with some sort of regulator. Seeing how efficiency was a major goal of this product we started searching by aiming for 90% efficient topologies. This quickly eliminated low drop out regulators and left us with the choice of a buck DC/DC converter.

The component chosen for the task was a synchronous 15W 12V to 5V buck converter. Its conversion efficiency (as specified by the manufacturer) is 90% at our designated load and can operate up to 80 degrees Celsius, which is more than sufficient for our application. It comes with mounting hardware for ease of placement on the display board and its input range (8 to 20 volts) covers that of our useable battery output (approximately 11 to 13 volts).

Figure 34: The DC/DC converter used to power the USB port from the battery.

After the CPT 12V-5V module was selected, the following circuit was constructed to test its operation. Varying demands on the converter were simulated by varying the resistance R_{LOAD}. To simulate the various voltages a lead-acid battery can have through its discharge curve, the supply voltage $V_{BATTERY}$ was also fluctuated and monitoring over the different loads. Data was gathered on the input current (using the DC current sensor in the Agilent 34405A) and on the output voltage with a handheld voltage meter (RSR MA830). These measurements were performed over $11V<V_{BATTERY}<13V$, and $0.6A<I_{LOAD}<2.4A$ to evaluate the converter's efficiency over the loading range required by the display board.

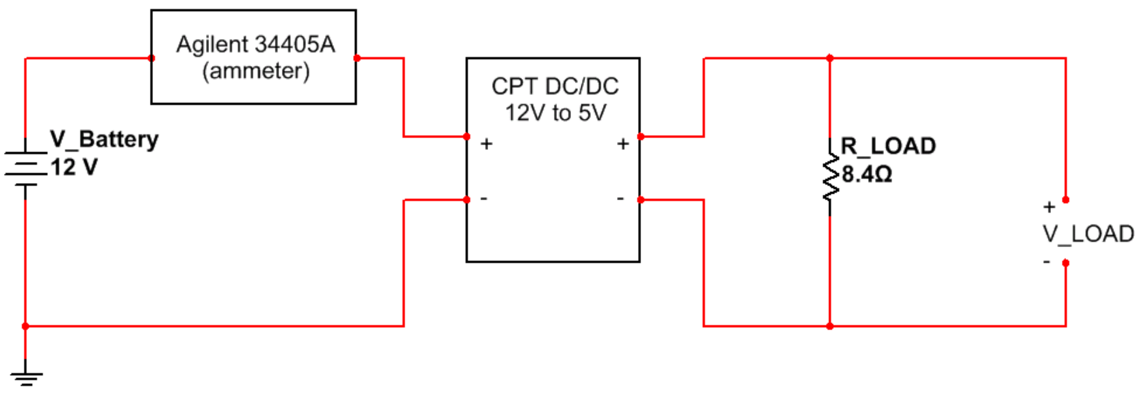


Figure 35: Test circuit for CPT DC/DC Module

Table 2: Test data gathered on CPT DC/DC Module

DC to DC 12-to-5 Test

R_LOAD [Ω]	V_SUPPLY [V]	I_IN [A]	V_OUT [V]	I_OUT [A]	P_IN [W]	P_OUT [w]	η	P_waste [W]
8.418	15.3	0.2218	5.1800	0.6153	3.394	3.188	0.9393	0.206
8.418	13.4	0.2535	5.1900	0.6165	3.397	3.200	0.9420	0.197
8.418	12.59	0.2684	5.1900	0.6165	3.379	3.200	0.9469	0.179
8.418	11.9	0.2837	5.1900	0.6165	3.376	3.200	0.9478	0.176
8.418	11.54	0.2924	5.1900	0.6165	3.374	3.200	0.9483	0.174
8.418	11.05	0.3054	5.1900	0.6165	3.375	3.200	0.9482	0.175
8.418	9.89	0.3412	5.1800	0.6153	3.374	3.188	0.9446	0.187
2.051	12	1.2720	5.0268	2.4509	15.264	12.320	0.8071	2.944
2.051	11	1.3854	5.0280	2.4515	15.239	12.326	0.8088	2.913
2.051	13	1.1600	5.0263	2.4507	15.080	12.318	0.8168	2.762
3.396	13.1	0.6363	5.0484	1.4866	8.336	7.505	0.9003	0.831
3.396	12.1	0.6969	5.0508	1.4873	8.432	7.512	0.8908	0.921
3.396	11	0.7660	5.0517	1.4875	8.426	7.515	0.8918	0.911

Using the data shown in **Table 2** (measured values in green, calculated values in black) the plot in **Figure 36** was generated. It shows a plot of converter efficiency as a function of the supply voltage. The three different curves represent different load currents, as annotated at the right.

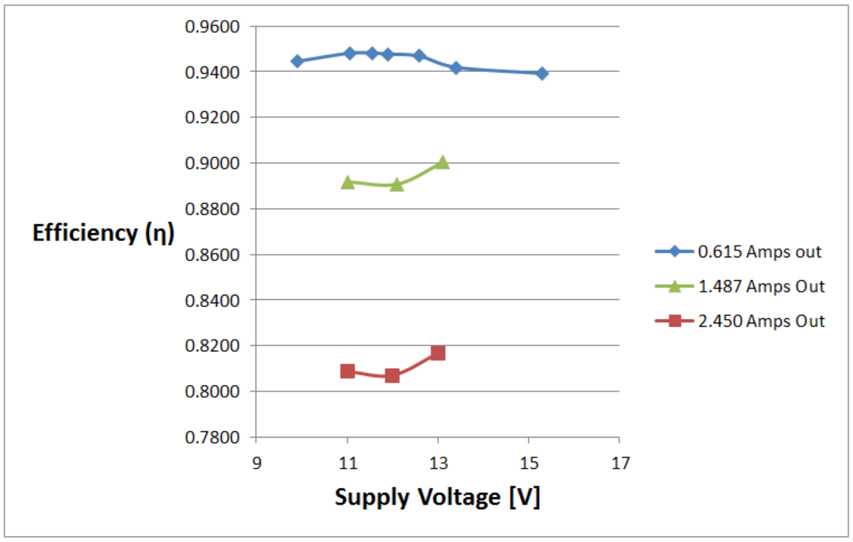


Figure 36: CPT DC/DC converter efficiency as a function of supply voltage

Figure 36 shows that at the typical converter loading (for just the Arduinos and display screens), which is between 0.5A and 1A, the efficiency is over 88%. This number is respectable for this application, because the battery being loaded has a capacity great enough to sustain the screens for two days without refilling the battery. Although a weather pattern which disables solar panels for more than two days is expectable, a grid-independent system with a two-day sustainability is acceptable.

In addition to the efficiency of the converter, it was also deemed important to determine the magnitude of the ripple voltage on the output. With a purely resistive 1.47A load on the converter's output, a Tektronix TDS 2004B oscilloscope was used with AC coupling to record the ripple voltage on the output. This is shown in **Figure 37**. The ripple had the shape of a ramp-wave with a peak-to-peak voltage that did not exceed 80mV, with the exception of some downward spikes at the bottom of the wave, which added as much as 40mV more. The wave's frequency was approximately 312.5 kilohertz. A coupling capacitor was added to the converter's output, to attempt to smooth this wave further.

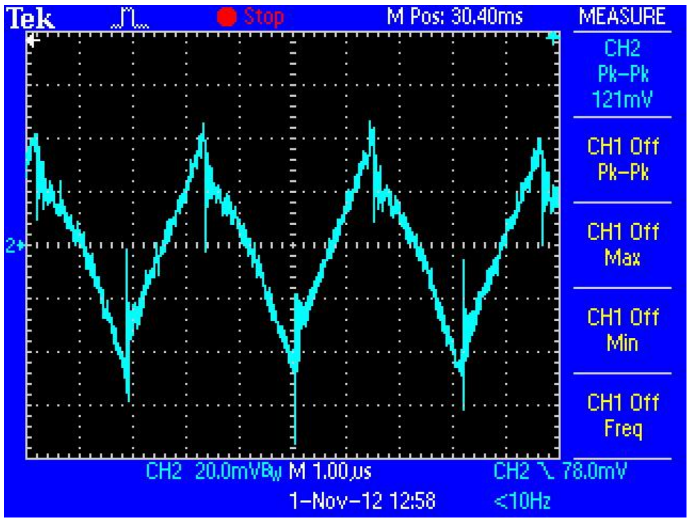


Figure 37: Ripple Voltage on CPT DC/DC Converter (with 1.47 amp load)

Another desirable piece of data was the consumption of the converter at no-load. Because one converter constantly drives the screens, its efficiency can be approximated by inspecting the plot in **Figure 36**. However, another DC/DC converter is used exclusively to drive the cell-phone charger. When no phones are connected, this converter has power supplied but no load to drive. Because that supply current is so small, the Agilent 34405A was deemed insufficient to measure it accurately. The following circuit was constructed:

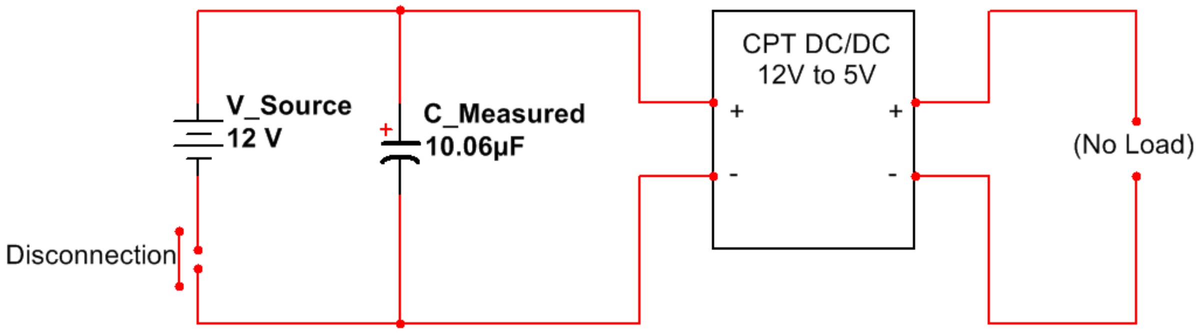


Figure 38: Circuit used to measure CPT DC/DC converter consumption at no-load

The circuit in **Figure 38** was used to measure the idle-consumption of the CPT DC/DC converter. The oscilloscope was used to view the capacitor voltage after the source was disconnected. With the source voltage at 12V and the capacitor connected, the switch was opened. The resulting waveform is shown in **Figure 39**.

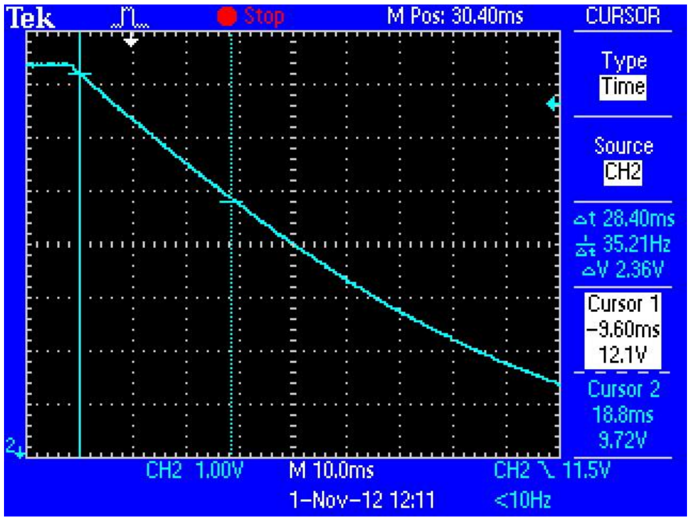


Figure 39: Voltage as an idling CPT DC/DC converter uses a capacitor as its source

The exponential-decay shape of this discharge results from the varying power consumption of the device as its source voltage falls below its specified operating range. A constant-current discharge of a capacitor should show linear voltage decay. Further, because the relevant idle-current for this system is for battery voltages higher than 11 V, a linear portion of the discharge was analyzed from the beginning of the curve. The screen cursors indicate that a voltage drop of 2.36V occurred over the time interval of 28.40 milliseconds.

$$Q = CV \ and \ i = \frac{dQ}{dt} \rightarrow \ i = \frac{d(CV)}{dt}$$

For a linear discharge, the equation above can be rewritten with the derivatives replaced with Δ statements. This allows the simple quantities obtained by oscilloscope cursors to be used to calculate the current.

$$i = \frac{\Delta(CV)}{\Delta T} = \frac{C\Delta V}{\Delta T} = 10.06\mu F * \frac{2.36V}{28.4ms} = 835.9\mu A$$

Given that the calculated current is less than 1mA, the self-discharge rate of the source-capacitor was questioned. Next, with the converter disconnected, the switch was closed to allow the capacitor to charge. The Tektronix oscilloscope was set to probe the voltage of the capacitor (with DC coupling). The switch was again opened to allow the capacitor's self-discharge current. The corresponding waveform was measured, and is shown in **Figure 40**.

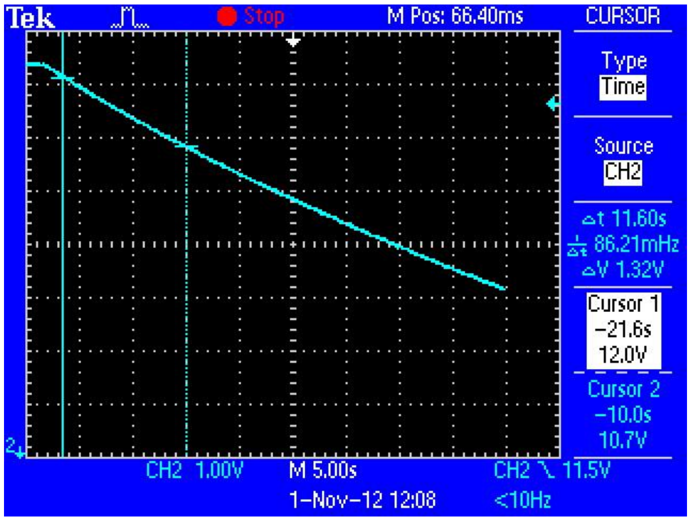


Figure 40: Self discrge of capacitor in idle-state test circuit for CPT DC/DC converter

Figure 40 shows the self-discharge of the 10.06 μF capacitor. Note that the timescale is 5 seconds/division. Using cursor to view the (roughly) first volt of discharge, it was determined that a voltage drop of 1.32 V occurred over 11.6 seconds. Therefore:

$$i_{self} = \frac{C\Delta V}{\Delta T} = 10.06\mu F * \frac{1.32V}{11.6\ s} = 1.14\mu A$$

The resulting calculation is that when fed by a 12-volt source, the CPT DC/DC converter module's input current at no-load is $835.9\mu A - 1.14\mu A \approx 834.7\mu A$. The power consumed by this operation is $12\ V * 834.7\mu A = 10.0\ mW$, which is an acceptably small value for this application.

Arduino

Introduction

To control certain signal pins and data lines as well as display power flow information in the display, a microcontroller is used. This method was chosen over other display topologies for a variety of reasons including cost, ease of implementation, and capabilities. The Arduino Uno (Rev 3) was selected due to its impressive web support that hosts a variety of tutorials and community support, and the software's fame for being easy to use for rapid prototyping.

There are three screens on the display, one displaying power flow from the solar panel, one displaying power flow from the battery, and one displaying power flow from the outlet. When surveying the display the user can see the flow of power. A snapshot of one screen in operation is shown in **Figure 41**.

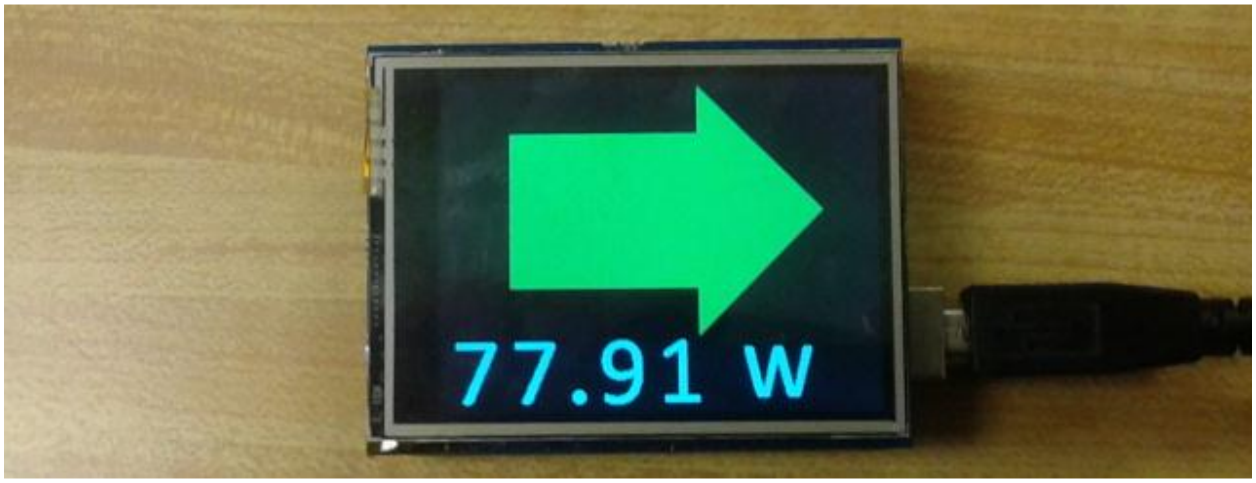


Figure 41: A snapshot of the Adafruit 2.8" TFT shield for Arduino Uno Rev 3

The resolution of the display screen is 320x240 pixels, and it supports 18-bit color (262000 colors).   Its power supply can be either 5V or 3.3V, and it has an onboard LDO regulated backlight. It is also a resistive touchscreen, but that feature was not utilized in this project. This is partially because these screens are not well reinforced, and are resultantly somewhat breakable. Additionally, time-constraints and microcontroller memory limitations made the implementation of the touch-screen an infeasible task.

An unabridged version of the C-style Arduino code can be found in **Appendix A**. Some of the more important functions are discussed in this section.

The Memory Problem

The Arduino Uno Rev 3 (which operates on the ATMEGA328P microcontroller) has 32 kilobytes of RAM. However, during the design process, an interesting memory problem surfaced. If the compiled size of the program exceeded 14.5 kB, it would not run correctly on the device. After consulting countless reference manuals and seeking help from various internet communities focusing on Arduino circuits, a solution remains unfound. Some static data was specifically stored in the program-memory (PROGMEM) to free up more space for code, but the rest of the total build was still unable to exceed 14.5 kB. For this reason, many of the functions discussed in this section had to be optimized for device-space rather than computation speed.

The Main Loop

The program's main loop calls the other functions discussed in this section, applying a logical order to the tasks performed by the Arduino.

```
01.  void loop(void){
02.
03.      float wattage=senseWattage();                 // call VI sense function.  Result is a signed float with unit [Watts]
04.      tft.fillScreen(BLACK);                        // clear the screen from last pass.
05.        if (wattage<0) drawArrow(  RED, true );     // negative wattage -- point arrow towards left, make it red.
06.        else           drawArrow(GREEN, false);     // else: positive wattage         "         right     "   green.
07.      wattage=abs(wattage);                         // arrow indicates direction -- only print positive magnitude
08.      printWatts(1000000 * wattage);                // print it (note: unit for printWatts() function is microwatts.
09.      delay(2300);                                  // wait a while - display updates once every ~2.3 seconds.
10.  } /// end of main loop
```

First (line 3), sensors are polled and the wattage of the power line is determined. Next (line 4), the screen is reset for the new display information to be drawn to it. Lines 5 and 6 determine the direction that the arrow should be pointing, and call the *drawArrow* function to draw it to the screen. In lines 7 and 8, the absolute value of the sensed wattage (with the unit of microwatts) is passed into the *printWatts* function, to be printed to the display. Finally a 2300 millisecond delay is employed (line 9) to slow the update-rate of the screen and make the display easier to read.

Sensing Wattage

The purpose of the *senseWattage* function is to read the analog pins A4 and A5 on the Arduino, using them as sensors for the voltage and current in the test-line. A voltage divider between the power line and ground is employed to step down the input voltage to be within the Arduino ADC limits (0V-5V). The current sensor (on pin A5) uses the Arduino's V_{CC} and returns voltage proportional to the current in the power line. These values are multiplied and returned as a floating point number, with the unit of watts.

```
float senseWattage(){
  int voltageBits = analogRead(A4);                // read the input on analog pin 0:
  float voltageVolts = voltageBits*0.00488759;     // converts bits to volts
  float sensedVoltage = voltageVolts*8.5965;       // Scalar determined by the voltage divider to pin A0

  int currentValue = analogRead(A5);
  float sensedCurrent = currentValue * 0.00488759; // convert current sensor voltage output from bits to volts
  sensedCurrent = (sensedCurrent-2.5) * 10 + 0.3;  // for -30 use 15.15+0.3 for -20 use 10.0+0.3

  many++; // increment the global counter.
  // Battery control - this code can be included on all three displays,
  // since the value of pin D3 doesn't matter on the other modules
  if (many>26){ // if 60 seconds has passed since the last time the battery voltage got tested,
    many=0; // reset the global integer 'many' for the next pass.
    if (sensedVoltage<=11)  digitalWrite(3, LOW); // BATTERY TOO LOW.
                            // Bring NMOS control switch's gate down to zero to turn off the inverter
    else                    digitalWrite(3, HIGH);  // BATTERY OPERATIONAL.
                            //Print NMOS control switch back to 5 V.
  }
  return sensedCurrent * sensedVoltage;
}
```

The *senseWattage* also controls the inverter-control switch, which is an NMOS device used to turn off the inverter when the battery voltage falls too low, to prolong the cellphone-charging capability of the system and protect the battery from deep discharge. The global variable *many* is incremented every time the *senseWattage* function is called, which is once every ~2.3 seconds. When it reaches 26 (~once per minute) it brings the digital pin 3 (connected to the gate of the NMOS switch) to 0V if the battery voltage has fallen below 11V, and up to 5V if the battery voltage exceeds 11V.

Drawing the Arrow

The *drawArrow* function has two input parameters. One is a 16-bit integer indicating the arrow color, and the other is a Boolean indicating if the arrow is pointing to the left (TRUE) or not (FALSE). In the first iteration of the coding process, this function utilized two drawing functions from the *Adafruit_GFX* library (**Appendix B**). Due to chip memory problems, these functions were abandoned and the process was replaced with a series of calls to the *drawFastVLine* and *drawFastHLine* functions from the same library. This was done because those functions were required for the *printChar* function, and were therefore already being loaded onto the device. A marked-up sketch of the arrow can be found in **Figure 42**.

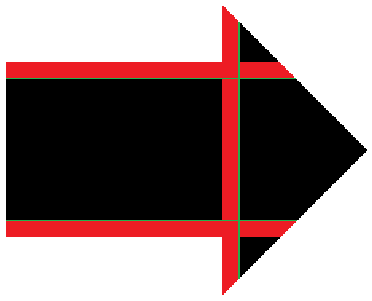


Figure 42: A cropped and marked-up sketch of a right-facing display-arrow

The red sections of **Figure 42** show sections are those which have already been drawn before the 11th pass through the draw-loop. The black sections illustrate where the arrow will be drawn, and the green lines show the pixels which are filled in the 11th pass through the draw-loop. The code for the *drawArrow* function shows the loop, which handles both left-facing and right-facing arrow.

```
void drawArrow(int color, boolean left){
  for (byte i=0; i<=55; i++){
    if (left){
      tft.drawFastHLine(  89-i,  41+i,   170+i, color);  // draws the horizontal lines of the top    of the arrow tail
      tft.drawFastHLine(  89-i, 151-i,   170+i, color);  // draws                  "             bottom        "
      tft.drawFastVLine( 124-i,   6+i, 181-i-i, color);  // draws the v-lines of the arrow point, starting at the right
      }
    else{
      tft.drawFastHLine(   55,  41+i,   170+i, color);
      tft.drawFastHLine(   55, 151-i,   170+i, color);
      tft.drawFastVLine(190+i,   7+i, 179-i-i, color);
    }
  }
}
```

For each pass through the loop, exactly three lines are drawn. The first two are horizontal lines which start at the top and bottom of the arrow's tail and stretch to its point, and the last one is the vertical line which makes up the flat part of the arrow's head.   One part of the code which is noticeably inefficient is that in each iteration through the loop, an if-statement is evaluated to determine if the arrow is left-facing or right-facing. This inefficient design was chosen to minimize the program-size on the Arduino. If the if-statement was evaluated once, outside the loop, two loop statements would be needed, which would increase the size of the program in the Arduino's memory. Processing-power is not lacking in the

Arduino or the screen, and resultantly the arrow is not drawn slowly enough that the human eye can see this process happen.

Conclusions

Shortcomings

The purpose of this section is to discuss the building of the green energy display board, make shortcomings of the project known, and what future improvements can be made to the system by future MQP students at WPI.

One of the obvious shortcomings to this project is the limitation of its power capability. It is only able to provide one AC outlet box, and cannot be guaranteed to run forever once sunlight is gone due to the energy storage limitations on the battery. Furthermore, to guarantee uninterrupted use of the USB outlets, only two outlets could be included.

A larger battery (or similar storage device) would need to fit on the display to fix this. Capacity is limited by the cost and size of that installation. Larger capacity batteries exist to fit our needs however they cost much more than allowed in our budget. Furthermore, due to the nature of our display we want the battery to be compact to fit in the display board appropriately. Right now the battery was chosen due to its sleek nature and thin profile.

Another downfall to the project is the way data is communicated. We chose to display the data on three LCD screens at logical locations in the system. This provides for a unique way to represent where power is coming from and going to. However, the screens are considerably smaller than the display board. Being only 2.8 inches, they are not the best way to communicate overall stats and use of the system. They are excellent for showing bi-directional power flow direction in the system as the arrow is quite visible from a user's standpoint but we also included some facts about overall use of the system such as "this many kilowatt hours saved so far" or "used this many joules of energy from the sun". The text may end up being too small to display clearly. One improvement a future project group could make would be transferring this data to a larger space such as the television screens located at the entrance of the Atwater Kent building. The data could be transferred wirelessly or by Ethernet, and the data about power usage could be added to the scrolling banners already in place on these screens. This could promote the board further because of the TV's eye-catching nature.   However, the system functions as it is currently designed, and a scrolling data stream is employed on the 2.8 inch screens.

One final major shortcoming of this MQP is cost. Right now the entire setup is relatively expensive to build if compared to the cost of grid-electricity. Additionally, its installation could be considered undesirable because it can disable itself when the battery is depleted. However, financial incentives for alternative-energy installations exist on both the federal and state levels in all 50 states, both for individuals and businesses. The federal internal revenue code chapter 26 USC § 136 states that taxable "[g]ross income shall not include the value of any subsidy provided (directly or indirectly) by a public utility to a customer for the purchase or installation of any energy conservation measure." However, despite governmental incentive programs, the cost of this system and its installation will still leave some people uninterested.

These are the three main suggested problems to address with this current design to make an overall better display board to promote green energy in colleges and students alike. Aside from reducing costs, adding longer functionality, and displaying information better there are many more ways this MQP could be improved upon as far as features go. A permanent outdoor installation in a remote location seems more attractive with respect to these problems, because an increased cost is expected, usage would likely be lighter, and information display is not a great priority.   Despite the drawbacks identified here, the design has applications.

Recommended Future Developments

Implementing an On/Off Switch

While a successful prototype and proof of concept was designed and tested, a few major improvement areas have identified themselves. The most obvious one is a on/off switch for the system. As we were repeatedly testing and debugging the system many times it was required that we disable the power. Seeing how the battery used ring terminals with a screw and nut this task became very time consuming and inefficient. Had a switch been implemented we would not need to undo such a tough connection dozens and dozens of times.

Moreover, every system should have a way to disable the power source. For example, if this project needed to be transported 200 miles to a different college campus there would be no need to have the battery just drain itself over that time so when it arrives the system can't be used until the sun has charged up the cell. This is a very easy fix, as any single pole single throw switch can be mounted in line with the battery's plus rail. One thing we thought of after completion was a further safety feature to protect the charge controller. Rather than just have a switch, place an easy to remove fuse and fuse holder in line with the battery. This way, a user can simply pluck out the fuse when the system should be shut off, and this also adds a layer of protection to the MPPT controller whose max input current from the battery to the load is 20 amperes. For this reason a 20A fuse should be used.

One important caveat to this system to note is that the solar panel should always be unplugged before disconnecting the battery. The datasheet of the MPPT controller indicates that plugging should happen in that order, and that unpredictable performance could occur if those steps are not followed. This results from the fact that this system can operate on either a 12V battery bank or a 24V battery bank, and the load output voltage is decided after a test on the battery voltage is complete. Therefore, to ensure a reliable system make sure there is no solar panel connected to the charge controller when the battery is disconnected from the system.

Adding Another Display Screen

After the prototype was completed another improvement was quickly realized beyond the timeframe of this MQP. Considering the USB plug will mainly be used for charging cell phones, and the AC outlet plug will mainly be used for charging laptops there is a huge difference in power consumption. As discussed earlier, a typical smart phone consumes about 3W peak to charge, whereas a typical laptop can range from 60 to 100W peak. For this reason, a separate display for USB loads and AC plug loads should have been implemented.

Since the loads vary by an entire order of magnitude, two screens would have enabled the user to get more helpful data. If one student is charging a cell phone, and a different one a laptop then the cell phone user won't really know what sort of charge his cell phone is taking. The majority of the display will be the laptop load, whereas his phone just blends in, defeating the purpose of the project goal to allow the user to easily "see" where the power is flowing in the system. It is known it is going this way, but it forks to USB and AC outlets and there should be a screen for each versus both. This is easy to implement, just time consuming.

Moreover, a standard cell phone charges at a much lower current (around 500mA) than a laptop (around 4A) and the current sensing method we used is not ideal for low currents. In fact, hall-effect sensors are not recommended for low DC currents! In the future design, the current sensor topology should be a small sense resistor which sends the output to the newly added screen so when a laptop and cellphone are plugged in, each will know exactly how much power they are consuming.

Final Thoughts

Overall this project resulted in a physical functional prototype, which is very close to being ready for public display in Atwater Kent. A few minor fixes (listed in this paper), which require little-to-no further design work, can be implemented easily to ready the prototype for its debut in the public eye. The system could be a valuable talking-point for tours, and may possibly attract more ECE students to learn about renewable energy applications. Further, the design could spread to other locations on campus to further promote green energy awareness. Overall, this product and idea is new to the market, has the potential to promote a greener Earth.

Works Cited

1) Lund, Mark W. "NiMH Battery Charging Basics." *PowerStream*. PowerStream Technologies, n.d. Web. 16 Oct. 2012. <http://www.powerstream.com/NiMH.htm>.

2) "How to Store Batteries." *Battery University*. N.p., n.d. Web. 16 Oct. 2012. <http://batteryuniversity.com/learn/article/how_to_store_batteries>.

3) Hahn, James H. "Modified Sine-Wave Inverter Enhanced." University of Missouri-Rolla Engineering Education Center, St. Louis, 1 Aug. 2006. Web. 16 Oct. 2012. <http://powerelectronics.com/power_semiconductors/power_mosfets/power_modified_sinewave_inverter/index2.html>.

4) "Inverter FAQ - DonRowe.com - Frequently Asked Questions about Power Inverters." *Inverter FAQ - DonRowe.com - Frequently Asked Questions about Power Inverters*. N.p., n.d. Web. 16 Oct. 2012. <http://www.donrowe.com/inverters/inverter_faq.html>.

5) "What Is: Hybrid Car Terminology Definitions." *Examiner.com*. N.p., n.d. Web. 17 Dec. 2012.

Referenced Datasheets

6) http://www.digikey.com/product-detail/en/FA-T220-51E/FA-T220-51E-ND/2416493

7) http://www.digikey.com/product-detail/en/IRFP064PBF/IRFP064PBF-ND/811664

8) http://www.mouser.com/Passive-Components/Resistors/Current-Sense-Resistors/_/N-7fjcf?P=1z0vkpb

Appendix A: Arduino Code

```
// IMPORTANT: Adafruit_TFTLCD LIBRARY MUST BE SPECIFICALLY
// CONFIGURED FOR EITHER THE TFT SHIELD OR THE BREAKOUT BOARD.
// SEE RELEVANT COMMENTS IN Adafruit_TFTLCD.h FOR SETUP.

 #include <Adafruit_GFX.h>    // Core graphics library
 #include <Adafruit_TFTLCD.h> // Hardware-specific library
// #include <madData.h>         // Font Driver by Sam and Adrian
// #include <avr/io.h>
// #include <avr/pgmspace.h>

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// Assign human-readable names to some common 16-bit color values:
#define BLACK   0x0000
#define RED     0xF800
#define GREEN   0x07E0


// #define  BLUE    0x001F
// #define CYAN    0x07FF
// #define MAGENTA 0xF81F
// #define YELLOW  0xFFE0
// #define WHITE   0xFFFF

Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
// If using the shield, all control and data lines are fixed, and
// a simpler declaration can optionally be used:
// Adafruit_TFTLCD tft;

#define wrap   TRUE;  // text wrapping ON

PROGMEM unsigned char fontArray[][84]={
  //0
  {7 , 8 , 9 , 10, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, 17, 18,
18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 23, 24, 24, 25, 25, 26, 26, 27, 27,
28, 28, 29, 29, 30, 30, 31, 32, 33, 34, 35, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {16, 12, 10, 8 , 7 , 6 , 33, 5 , 37, 4 , 39, 4 , 40, 3 , 41, 3 , 42, 3 ,
42, 3 , 42, 3 , 42, 3 , 42, 3 , 42, 3 , 42, 3 , 42, 3 , 41, 3 , 40, 4 , 39, 4
, 38, 5 , 35, 5 , 29, 6 , 8 , 10, 12, 18 ,0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {19, 26, 30, 34, 36, 10, 11, 8 , 8 , 7 , 6 , 5 , 6 , 5 , 5 , 5 , 4 , 4 , 4
, 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 4 , 5 , 5 , 6 , 5 , 6 , 6 , 7 ,
6 , 9 , 9 , 16, 14, 36, 32, 28, 24, 12 ,0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //1
  {8 , 9 , 9 , 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17,
17, 18, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {11, 9 , 41, 9 , 41, 8 , 41, 7 , 41, 7 , 41, 6 , 41, 5 , 41, 5 , 41, 4 ,
41, 3 , 41, 3 , 2 , 2 , 2 , 2 , 3 , 41, 41, 41, 41, 41, 41, 41, 41, 41, 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {2 , 5 , 5 , 5 , 5 , 6 , 5 , 6 , 5 , 6 , 5 , 6 , 5 , 6 , 5 , 6 , 5 , 6 , 5
, 7 , 5 , 43, 44, 44, 44, 44, 43, 5 , 5 , 5 , 5 , 5 , 5 , 5 , 5 , 5 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //2
  {16, 13, 11, 9 , 8 , 8 , 8 , 22, 8 , 24, 8 , 25, 25, 26, 26, 26, 26, 26,
26, 25, 25, 25, 24, 24, 23, 22, 22, 21, 20, 19, 18, 17, 17, 16, 15, 14, 13,
12, 11, 10, 9 , 8 , 8 , 7 , 7 , 7 , 8 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {2 , 3 , 4 , 5 , 6 , 7 , 8 , 8 , 9 , 9 , 10, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {9 , 14, 18, 21, 23, 24, 7 , 10, 5 , 8 , 3 , 8 , 8 , 7 , 7 , 7 , 7 , 7 , 7
, 8 , 7 , 7 , 8 , 7 , 7 , 8 , 7 , 8 , 8 , 8 , 8 , 8 , 7 , 7 , 8 , 8 , 8 , 8 ,
8 , 8 , 8 , 8 , 27, 28, 28, 28, 27, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //3
  {16, 13, 11, 9 , 9 , 8 , 8 , 23, 8 , 24, 8 , 25, 26, 26, 26, 26, 26, 25,
25, 24, 23, 21, 12, 12, 11, 11, 12, 12, 23, 25, 26, 27, 27, 27, 28, 28, 27,
27, 27, 8 , 26, 7 , 25, 7 , 24, 7 , 20, 7 , 8 , 9 , 11, 14, 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {2 , 3 , 4 , 5 , 6 , 7 , 8 , 8 , 9 , 9 , 10, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 38, 39, 39, 40, 40, 41, 41, 42, 43, 44, 45, 46, 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {9 , 14, 18, 21, 22, 24, 7 , 9 , 5 , 8 , 3 , 8 , 7 , 7 , 7 , 7 , 7 , 7 , 7
, 8 , 8 , 9 , 17, 16, 16, 18, 19, 20, 10, 8 , 8 , 7 , 8 , 8 , 7 , 7 , 8 , 7 ,
7 , 1 , 8 , 4 , 9 , 6 , 9 , 11, 12, 25, 22, 20, 16, 10, 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //4
  {4 , 5 , 6 , 7 , 8 , 9 , 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16,
16, 17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {31, 29, 27, 26, 24, 22, 21, 19, 31, 17, 31, 16, 31, 14, 31, 12, 31, 11,
31, 9 , 31, 7 , 31, 5 , 31, 4 , 31, 3 , 31, 3 , 31, 2 , 2 , 2 , 2 , 3 , 3 , 3
, 31, 31, 31, 31, 31, 32, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {5 , 7 , 10, 11, 13, 15, 16, 10, 6 , 11, 6 , 10, 6 , 10, 6 , 11, 6 , 10, 6
, 10, 6 , 11, 6 , 11, 6 , 10, 6 , 9 , 6 , 8 , 6 , 43, 44, 45, 45, 44, 43, 43,
6 , 6 , 6 , 6 , 6 , 4 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //5
  {10, 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 17, 9
, 9 , 9 , 9 , 10, 23, 25, 26, 27, 28, 28, 28, 28, 28, 28, 28, 27, 27, 26, 7 ,
25, 7 , 24, 7 , 19, 7 , 7 , 8 , 10, 13, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {3 , 4 , 5 , 6 , 7 , 8 , 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 39, 40, 40, 41, 41, 42, 43, 44, 45, 46, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {22, 23, 23, 23, 23, 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 6 , 4 ,
18, 20, 21, 23, 22, 10, 9 , 8 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 8 , 3
, 8 , 5 , 9 , 10, 13, 24, 23, 20, 17, 10, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //6
  {6 , 7 , 8 , 9 , 10, 11, 12, 13, 13, 13, 14, 14, 14, 15, 15, 15, 16, 16,
16, 17, 17, 17, 18, 18, 18, 19, 19, 19, 20, 20, 20, 21, 21, 21, 22, 22, 22,
23, 23, 23, 24, 24, 24, 25, 25, 25, 26, 26, 26, 27, 27, 27, 28, 28, 28, 29,
29, 30, 30, 31, 31, 32, 32, 33, 34, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {20, 15, 12, 10, 9 , 7 , 6 , 5 , 21, 35, 5 , 21, 38, 4 , 20, 40, 4 , 20,
40, 3 , 20, 41, 3 , 20, 41, 3 , 19, 42, 2 , 19, 42, 2 , 19, 42, 2 , 19, 41, 2
, 19, 41, 2 , 19, 41, 2 , 19, 40, 2 , 20, 39, 2 , 20, 38, 2 , 20, 34, 3 , 21,
3 , 21, 3 , 22, 4 , 23, 25, 27, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {13, 23, 28, 32, 34, 37, 39, 11, 6 , 11, 8 , 5 , 8 , 8 , 6 , 6 , 6 , 5 , 7
, 7 , 5 , 6 , 6 , 5 , 6 , 5 , 6 , 5 , 6 , 6 , 5 , 6 , 6 , 5 , 5 , 6 , 6 , 5 ,
6 , 6 , 5 , 6 , 6 , 5 , 6 , 6 , 6 , 6 , 7 , 6 , 7 , 7 , 6 , 11, 11, 5 , 23, 6
, 22, 6 , 20, 5 , 18, 15, 10, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //7
  {6 , 6 , 6 , 6 , 6 , 28, 27, 27, 26, 26, 25, 25, 24, 24, 24, 23, 23, 22,
22, 21, 21, 20, 20, 20, 19, 19, 18, 18, 17, 17, 16, 16, 15, 15, 15, 14, 14,
13, 13, 12, 12, 11, 11, 13, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {3 , 4 , 5 , 6 , 7 , 8 , 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {28, 29, 29, 29, 29, 6 , 7 , 6 , 7 , 7 , 7 , 7 , 7 , 7 , 6 , 7 , 7 , 7 , 7
, 7 , 7 , 7 , 7 , 6 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 7 , 8 , 7 , 7 , 7 , 7 , 7 ,
7 , 8 , 7 , 8 , 7 , 3 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //8
  {16, 13, 11, 10, 9 , 8 , 23, 8 , 25, 7 , 26, 7 , 27, 7 , 27, 7 , 27, 7 ,
27, 7 , 27, 7 , 27, 7 , 26, 8 , 25, 8 , 24, 9 , 23, 10, 21, 11, 12, 14, 14,
12, 11, 9 , 21, 8 , 23, 8 , 25, 7 , 26, 6 , 27, 6 , 28, 6 , 28, 5 , 28, 5 ,
29, 5 , 28, 5 , 28, 6 , 28, 6 , 27, 6 , 26, 7 , 23, 8 , 8 , 10, 11, 15},
  {2 , 3 , 4 , 5 , 6 , 7 , 7 , 8 , 8 , 9 , 9 , 10, 10, 11, 11, 12, 12, 13,
13, 14, 14, 15, 15, 16, 16, 17, 17, 18, 18, 19, 19, 20, 20, 21, 22, 23, 24,
25, 26, 27, 27, 28, 28, 29, 29, 30, 30, 31, 31, 32, 32, 33, 33, 34, 34, 35,
35, 36, 36, 37, 37, 38, 38, 39, 39, 40, 40, 41, 41, 42, 43, 44, 45, 46},
  {9 , 15, 19, 21, 23, 10, 9 , 7 , 8 , 7 , 7 , 7 , 6 , 6 , 7 , 6 , 7 , 6 , 6
, 6 , 6 , 7 , 6 , 7 , 7 , 7 , 7 , 9 , 8 , 9 , 8 , 10, 9 , 18, 15, 12, 14, 17,
20, 9 , 11, 9 , 10, 7 , 8 , 7 , 8 , 7 , 7 , 7 , 7 , 6 , 7 , 7 , 7 , 7 , 6 , 7
, 7 , 7 , 7 , 7 , 7 , 7 , 7 , 8 , 8 , 10, 10, 24, 24, 20, 17, 10},

  //9
  {6 , 7 , 8 , 8 , 9 , 9 , 10, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14, 14,
15, 15, 15, 16, 16, 16, 17, 17, 17, 18, 18, 18, 19, 19, 19, 20, 20, 20, 21,
21, 21, 22, 22, 22, 23, 23, 23, 24, 24, 24, 25, 25, 25, 26, 26, 26, 27, 27,
27, 28, 28, 28, 29, 30, 31, 32, 33, 34, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {14, 10, 8 , 40, 7 , 40, 6 , 40, 5 , 41, 4 , 41, 4 , 20, 41, 3 , 22, 41, 3
, 23, 42, 3 , 24, 42, 2 , 24, 42, 2 , 24, 42, 2 , 25, 41, 2 , 25, 41, 2 , 24,
41, 2 , 24, 40, 2 , 24, 40, 2 , 24, 39, 3 , 24, 38, 3 , 23, 36, 3 , 23, 34, 4
, 22, 31, 5 , 5 , 7 , 8 , 10, 13, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {5 , 13, 17, 5 , 19, 6 , 21, 6 , 23, 5 , 24, 6 , 8 , 9 , 6 , 7 , 7 , 6 , 6
, 6 , 5 , 5 , 6 , 5 , 6 , 6 , 5 , 6 , 6 , 5 , 5 , 5 , 6 , 5 , 5 , 6 , 5 , 6 ,
6 , 6 , 6 , 6 , 6 , 5 , 6 , 6 , 5 , 7 , 6 , 5 , 7 , 7 , 6 , 9 , 9 , 5 , 10,
13, 6 , 12, 37, 36, 33, 30, 25, 19, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //dp
  {8 , 6 , 6 , 5 , 5 , 5 , 5 , 6 , 6 , 8 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {4 , 8 , 8 , 10, 10, 10, 10, 8 , 8 , 4 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

  //W
  {4 , 5 , 6 , 7 , 8 , 9 , 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {7 , 7 , 7 , 7 , 7 , 7 , 8 , 12, 17, 21, 25, 30, 34, 35, 32, 28, 23, 19,
15, 11, 8 , 7 , 7 , 7 , 7 , 7 , 7 , 8 , 11, 15, 19, 23, 27, 31, 35, 35, 31,
26, 22, 18, 13, 9 , 7 , 7 , 7 , 7 , 7 , 7 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },
  {5 , 9 , 13, 17, 21, 25, 28, 27, 25, 22, 18, 13, 9 , 8 , 11, 15, 20, 23,
23, 23, 22, 19, 15, 12, 14, 18, 22, 25, 25, 25, 23, 20, 16, 12, 8 , 8 , 12,
17, 21, 23, 25, 25, 23, 19, 15, 12, 8 , 4 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }
};

/*
```

```
int many = 26; // a global used to track how long it has been since the last
refresh.
               // used only in the senseWatts() function.
*/

void setup(void) {
  Serial.begin(9600);
  Serial.println("TFT LCD test");
#ifdef USE_ADAFRUIT_SHIELD_PINOUT
  Serial.println("Using Adafruit 2.8\" TFT Arduino Shield Pinout");
#else
  Serial.println("Using Adafruit 2.8\" TFT Breakout Board Pinout");
#endif

  tft.reset();
  uint16_t identifier = tft.readID();
  tft.begin(identifier);
  tft.setRotation(3);
  tft.fillScreen(BLACK);              // screen background is black
  Serial.begin(9600);
  digitalWrite(3, HIGH);

} // end of setup




void loop(void){

    float wattage=senseWattage();                  // call VI sense function.
Result is a signed float with unit [Watts]
    tft.fillScreen(BLACK);                         // clear the screen from
last pass.
      if (wattage<0) drawArrow(  RED, true );      // negative wattage --
point arrow towards left, make it red.
      else           drawArrow(GREEN, false);      // else: positive wattage
"        right     "   green.
    wattage=abs(wattage);                          // arrow indicates
direction -- only print positive magnitude
    printWatts(1000000 * wattage);                 // print it (note: unit
for printWatts() function is microwatts.
    delay(2300);                                   // wait a while - display
updates once every ~2.3 seconds.
} /// end of main loop



float senseWattage(){




  int voltageBits = (analogRead(A4) + analogRead(A4) + analogRead(A4))/3;
                // read the input on analog pin 4, average three reads
  float V_SOURCE = 4.9995; //1: 4.9995,   2: 5.0514,   3: 4.9719
  float voltageVolts = voltageBits*(V_SOURCE/1024);     // converts bits to
volts
  float Rg=0.99330;  //1:  0.99330,    2: 0.99102,    3: 0.99472
  float Rp=9.08570;  //1:  9.08570,    2: 1.96780,    3: 1.87160
  float sensedVoltage = voltageVolts*(Rg+Rp)/(Rg);
               // Voltage divider ratio multiplier calculated during
compiling



  int currentBits = (analogRead(A5) + analogRead(A5) + analogRead(A5))/3;
                // read current sensor value, three times, and average them
  float zeroOutputVoltage = 2.5082; // whatever the V_OUT is when there's
zero current
                                   //1: 2.5082,    2: 2.5496,    3: 2.4985
  float sensedCurrent =  ( currentBits * V_SOURCE / 1024 ) -
zeroOutputVoltage ;
        // convert current sensor voltage output from bits to volts
  float ampsPerVolt = 0.0619;  //1: 0.0619,    2: 0.1789,    3: 0.09245

  sensedCurrent = sensedCurrent / ampsPerVolt;

  if (sensedCurrent<0) sensedCurrent=0;
/*
  many++; // increment the global counter.

// Battery control - this code can be included on all three displays, since
the value of pin D3 doesn't matter on the other modules
  if (many>26){ // if 60 seconds has passed since the last time the battery
voltage got tested,
    many=0; // reset the global integer 'many' for the next pass.
    if (sensedVoltage<=11)  digitalWrite(3, LOW); // BATTERY TOO LOW. Bring
NMOS control switch's gate down to zero to turn off the inverter
    else                    digitalWrite(3, HIGH);  // BATTERY OPERATIONAL.
Print NMOS control switch back to 5 V.
  }
*/



  return sensedCurrent * sensedVoltage;
}



void drawArrow(int color, boolean left){
  for (byte i=0; i<=55; i++){
    if (left){
      tft.drawFastHLine(  89-i,  41+i,   170+i, color);  // draws the
horizontal lines of the top    of the arrow tail
```

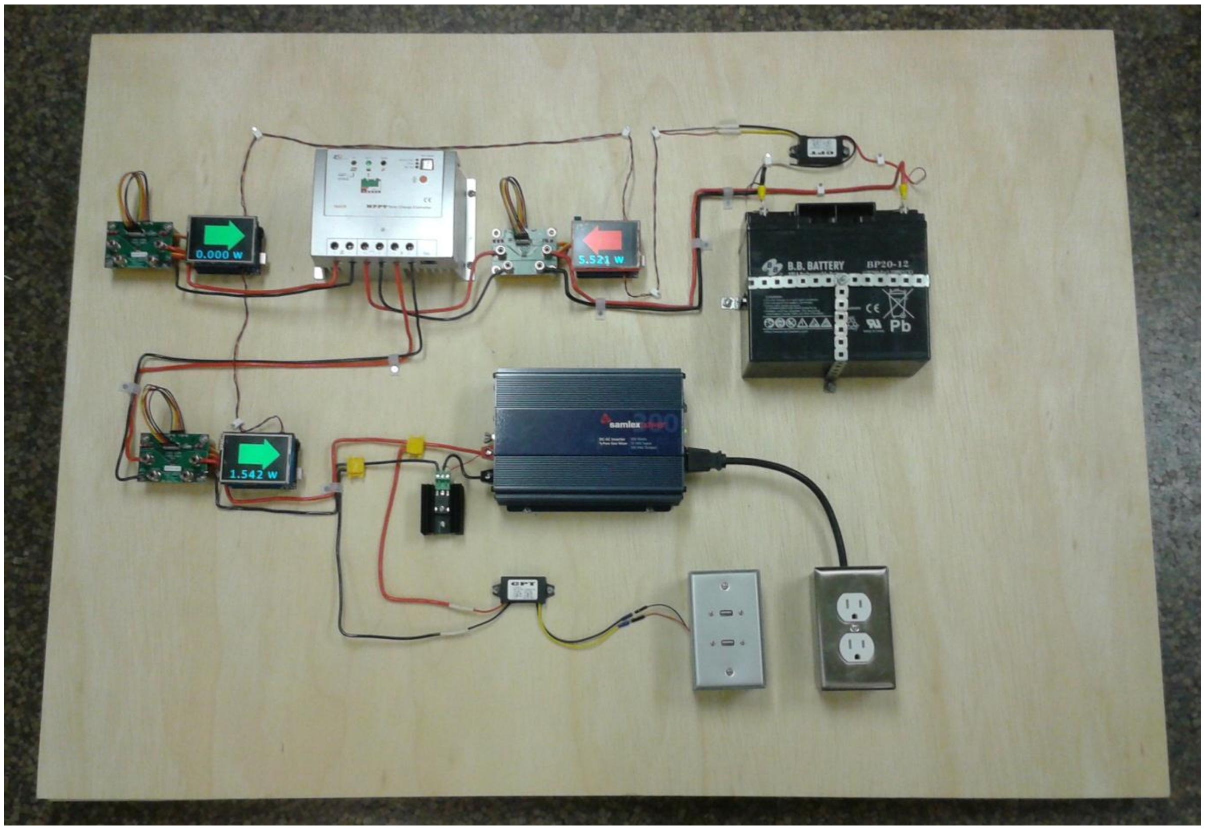
```
      tft.drawFastHLine(  89-i, 151-i,   170+i, color);  // draws
"               bottom        "
      tft.drawFastVLine( 124-i,   6+i, 181-i-i, color);  // draws the v-lines
of the arrow point, starting at the right
      }
    else{
      tft.drawFastHLine(   55,  41+i,   170+i, color);
      tft.drawFastHLine(   55, 151-i,   170+i, color);
      tft.drawFastVLine(190+i,   7+i, 179-i-i, color);
    }
  }
}






// print char input: signed char number, int x_coord, int y_coord, int color)
;
   //  XY are top left of char print spot
// print char output: unsigned int - increment the cursor by this many pixels
unsigned int printChar(signed char input, unsigned int x, unsigned int y,
unsigned int color ){

// 0   1   2   3   4   5   6   7   8   9   .   W
// 0   3   6   9   12  15  18  21  24  27  30  33
// V   V   H   H   V   H   V   H   H   V   H   V

  unsigned int charWidth=42;
  if (input==-16){ // input is a space.
    return 20;}   // draw nothing, return 20 so the calling function knows to
increment its cursor
  else if(input<0){   //decimal point. starts at 30, is H
    input=10;    // make the next bit of lookup table work in my favor.
    charWidth=20;}
  else if(input>9){// W.  starts at 33, is drawn with H
   input=11;     // make the next bit of lookup table work in my favor.
   charWidth=56; }

  byte row_start=input*3;          // holds the offset for the font vector in
pgmem
  byte dir_lookup[] = {1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1}; // a lookup table
for the draw-directions
  byte draw_dir=dir_lookup[input]; // indicates the draw function to draw
horizontal (0) or vertical (1) for each character

  // now draw_dir is either 0 (indicating horizontal) or 1 (indicating
vertical)
  // and row_start is the row in the pgmem table at which the current piece
of the font vector is stored.

  int i=0;  // conumn pointer for fetching vector information from pgmem.
  while (pgm_read_byte(&fontArray[row_start][i])>0){ // loop until the zero
point is found.
    if (draw_dir==1){
      tft.drawFastVLine(
        x+pgm_read_byte(&fontArray[row_start  ][i]),
        y+pgm_read_byte(&fontArray[row_start+1][i]),
        0+pgm_read_byte(&fontArray[row_start+2][i]),
        color);}
    else {
      tft.drawFastHLine(
        x+pgm_read_byte(&fontArray[row_start  ][i]),
        y+pgm_read_byte(&fontArray[row_start+1][i]),
        0+pgm_read_byte(&fontArray[row_start+2][i]),
        color);}
    i++;  // increment the column pointer i
  }

  return charWidth;
}




// Function to print a calculated wattage value on the screen
// Pass the value in in microwatts.  MICROWATTS
// the integer 'digits' is how many digits to print.  Decimal point is not
included.
void printWatts(signed long wattage){  // that long has the unit of
MICROWATTS.  Don't you forget it.

  // now the input wattage (unit: uW) is to be converted into a string
  // the conversion from long-int to string is implicit

    String wattage_left   = String(wattage/1000000);                    // get
numbers from the left of the dp
    String wattage_right  = String(wattage%1000000);                    // get
the stuff after the decimal point, seperately

    while (wattage_right.length()<6)
    {
      wattage_right = wattage_right+"0";                                // fix
the problem of the dropped zero after the dp
    }

    String dp = String(".");                                            //
Decimal point string to be used in concatenation
    String wattage_string = wattage_left + dp + wattage_right;          //
concatentate strings, add the decimal point

    String units = String(" W");
    wattage_string = wattage_string.substring(0,5) + units;

    if (wattage==0) wattage_string=String("0.000 W");

    unsigned int myCursor[]={10, 187};                                  //
holds the location to print to

    for (int i=0; i<7; i++){
      myCursor[0] = myCursor[0] + printChar((signed
char)byte(wattage_string.charAt(i))-48, myCursor[0], myCursor[1], 0x07FF);
    }
}
```

Appendix B: Adafruit_GFX Library

```
/***********************************
This is a our graphics core library, for all our displays.
We'll be adapting all the
existing libaries to use this core to make updating, support
and upgrading easier!

Adafruit invests time and resources providing this open source code,
please support Adafruit and open-source hardware by purchasing
products from Adafruit!

Written by Limor Fried/Ladyada  for Adafruit Industries.
BSD license, check license.txt for more information
All text above must be included in any redistribution
****************************************/


#include "Adafruit_GFX.h"
#include "glcdfont.c"
#include <avr/pgmspace.h>

void Adafruit_GFX::constructor(int16_t w, int16_t h) {
  _width = WIDTH = w;
  _height = HEIGHT = h;

  rotation = 0;
  cursor_y = cursor_x = 0;
  textsize = 1;
  textcolor = textbgcolor = 0xFFFF;
  wrap = true;
}


// draw a circle outline
void Adafruit_GFX::drawCircle(int16_t x0, int16_t y0, int16_t r,
                  uint16_t color) {
  int16_t f = 1 - r;
  int16_t ddF_x = 1;
  int16_t ddF_y = -2 * r;
  int16_t x = 0;
  int16_t y = r;

  drawPixel(x0, y0+r, color);
  drawPixel(x0, y0-r, color);
  drawPixel(x0+r, y0, color);
  drawPixel(x0-r, y0, color);

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;

    drawPixel(x0 + x, y0 + y, color);
    drawPixel(x0 - x, y0 + y, color);
    drawPixel(x0 + x, y0 - y, color);
    drawPixel(x0 - x, y0 - y, color);
    drawPixel(x0 + y, y0 + x, color);
    drawPixel(x0 - y, y0 + x, color);
    drawPixel(x0 + y, y0 - x, color);
    drawPixel(x0 - y, y0 - x, color);

  }
}

void Adafruit_GFX::drawCircleHelper( int16_t x0, int16_t y0,
               int16_t r, uint8_t cornername, uint16_t color) {
  int16_t f     = 1 - r;
  int16_t ddF_x = 1;
  int16_t ddF_y = -2 * r;
  int16_t x     = 0;
  int16_t y     = r;

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f     += ddF_y;
    }
    x++;
    ddF_x += 2;
    f     += ddF_x;
    if (cornername & 0x4) {
      drawPixel(x0 + x, y0 + y, color);
      drawPixel(x0 + y, y0 + x, color);
    }
    if (cornername & 0x2) {
      drawPixel(x0 + x, y0 - y, color);
      drawPixel(x0 + y, y0 - x, color);
    }
    if (cornername & 0x8) {
      drawPixel(x0 - y, y0 + x, color);
      drawPixel(x0 - x, y0 + y, color);
    }
    if (cornername & 0x1) {
      drawPixel(x0 - y, y0 - x, color);
      drawPixel(x0 - x, y0 - y, color);
    }
  }
}

void Adafruit_GFX::fillCircle(int16_t x0, int16_t y0, int16_t r,
                  uint16_t color) {
  drawFastVLine(x0, y0-r, 2*r+1, color);
  fillCircleHelper(x0, y0, r, 3, 0, color);
}

// used to do circles and roundrects!
```

```
void Adafruit_GFX::fillCircleHelper(int16_t x0, int16_t y0, int16_t r,
                    uint8_t cornername, int16_t delta, uint16_t color) {

  int16_t f     = 1 - r;
  int16_t ddF_x = 1;
  int16_t ddF_y = -2 * r;
  int16_t x     = 0;
  int16_t y     = r;

  while (x<y) {
    if (f >= 0) {
      y--;
      ddF_y += 2;
      f     += ddF_y;
    }
    x++;
    ddF_x += 2;
    f     += ddF_x;

    if (cornername & 0x1) {
      drawFastVLine(x0+x, y0-y, 2*y+1+delta, color);
      drawFastVLine(x0+y, y0-x, 2*x+1+delta, color);
    }
    if (cornername & 0x2) {
      drawFastVLine(x0-x, y0-y, 2*y+1+delta, color);
      drawFastVLine(x0-y, y0-x, 2*x+1+delta, color);
    }
  }
}

// bresenham's algorithm - thx wikpedia
void Adafruit_GFX::drawLine(int16_t x0, int16_t y0,
                int16_t x1, int16_t y1,
                uint16_t color) {
  int16_t steep = abs(y1 - y0) > abs(x1 - x0);
  if (steep) {
    swap(x0, y0);
    swap(x1, y1);
  }

  if (x0 > x1) {
    swap(x0, x1);
    swap(y0, y1);
  }

  int16_t dx, dy;
  dx = x1 - x0;
  dy = abs(y1 - y0);

  int16_t err = dx / 2;
  int16_t ystep;

  if (y0 < y1) {
    ystep = 1;
  } else {
    ystep = -1;
  }
  for (; x0<=x1; x0++) {
    if (steep) {
      drawPixel(y0, x0, color);
    } else {
      drawPixel(x0, y0, color);
    }
    err -= dy;
    if (err < 0) {
      y0 += ystep;
      err += dx;
    }
  }
}


// draw a rectangle
void Adafruit_GFX::drawRect(int16_t x, int16_t y,
                int16_t w, int16_t h,
                uint16_t color) {
  drawFastHLine(x, y, w, color);
  drawFastHLine(x, y+h-1, w, color);
  drawFastVLine(x, y, h, color);
  drawFastVLine(x+w-1, y, h, color);
}

void Adafruit_GFX::drawFastVLine(int16_t x, int16_t y,
                 int16_t h, uint16_t color) {
  // stupidest version - update in subclasses if desired!
  drawLine(x, y, x, y+h-1, color);
}


void Adafruit_GFX::drawFastHLine(int16_t x, int16_t y,
                 int16_t w, uint16_t color) {
  // stupidest version - update in subclasses if desired!
  drawLine(x, y, x+w-1, y, color);
}

void Adafruit_GFX::fillRect(int16_t x, int16_t y, int16_t w, int16_t h,
                uint16_t color) {
  // stupidest version - update in subclasses if desired!
  for (int16_t i=x; i<x+w; i++) {
    drawFastVLine(i, y, h, color);
  }
}


void Adafruit_GFX::fillScreen(uint16_t color) {
  fillRect(0, 0, _width, _height, color);
}

// draw a rounded rectangle!
void Adafruit_GFX::drawRoundRect(int16_t x, int16_t y, int16_t w,
  int16_t h, int16_t r, uint16_t color) {
  // smarter version
  drawFastHLine(x+r  , y    , w-2*r, color); // Top
  drawFastHLine(x+r  , y+h-1, w-2*r, color); // Bottom
  drawFastVLine(  x    , y+r  , h-2*r, color); // Left
  drawFastVLine(  x+w-1, y+r  , h-2*r, color); // Right
  // draw four corners
  drawCircleHelper(x+r    , y+r    , r, 1, color);
  drawCircleHelper(x+w-r-1, y+r    , r, 2, color);
  drawCircleHelper(x+w-r-1, y+h-r-1, r, 4, color);
  drawCircleHelper(x+r    , y+h-r-1, r, 8, color);
}

// fill a rounded rectangle!
void Adafruit_GFX::fillRoundRect(int16_t x, int16_t y, int16_t w,
                 int16_t h, int16_t r, uint16_t color) {
  // smarter version
  fillRect(x+r, y, w-2*r, h, color);

  // draw four corners
  fillCircleHelper(x+w-r-1, y+r, r, 1, h-2*r-1, color);
  fillCircleHelper(x+r    , y+r, r, 2, h-2*r-1, color);
}

// draw a triangle!
void Adafruit_GFX::drawTriangle(int16_t x0, int16_t y0,
                int16_t x1, int16_t y1,
                int16_t x2, int16_t y2, uint16_t color) {
  drawLine(x0, y0, x1, y1, color);
  drawLine(x1, y1, x2, y2, color);
  drawLine(x2, y2, x0, y0, color);
}

// fill a triangle!
void Adafruit_GFX::fillTriangle ( int16_t x0, int16_t y0,
                  int16_t x1, int16_t y1,
                  int16_t x2, int16_t y2, uint16_t color) {

  int16_t a, b, y, last;

  // Sort coordinates by Y order (y2 >= y1 >= y0)
  if (y0 > y1) {
    swap(y0, y1); swap(x0, x1);
  }
  if (y1 > y2) {
    swap(y2, y1); swap(x2, x1);
  }
  if (y0 > y1) {
    swap(y0, y1); swap(x0, x1);
  }

  if(y0 == y2) { // Handle awkward all-on-same-line case as its own thing
    a = b = x0;
    if(x1 < a)      a = x1;
    else if(x1 > b) b = x1;
    if(x2 < a)      a = x2;
    else if(x2 > b) b = x2;
    drawFastHLine(a, y0, b-a+1, color);
    return;
  }
  int16_t
    dx01 = x1 - x0,
    dy01 = y1 - y0,
    dx02 = x2 - x0,
    dy02 = y2 - y0,
    dx12 = x2 - x1,
    dy12 = y2 - y1,
    sa   = 0,
    sb   = 0;

  // For upper part of triangle, find scanline crossings for segments
  // 0-1 and 0-2.  If y1=y2 (flat-bottomed triangle), the scanline y1
  // is included here (and second loop will be skipped, avoiding a /0
  // error there), otherwise scanline y1 is skipped here and handled
  // in the second loop...which also avoids a /0 error here if y0=y1
  // (flat-topped triangle).
  if(y1 == y2) last = y1;   // Include y1 scanline
  else         last = y1-1; // Skip it

  for(y=y0; y<=last; y++) {
    a   = x0 + sa / dy01;
    b   = x0 + sb / dy02;
    sa += dx01;
    sb += dx02;
    /* longhand:
    a = x0 + (x1 - x0) * (y - y0) / (y1 - y0);
    b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
    */
    if(a > b) swap(a,b);
    drawFastHLine(a, y, b-a+1, color);
  }

  // For lower part of triangle, find scanline crossings for segments
  // 0-2 and 1-2.  This loop is skipped if y1=y2.
  sa = dx12 * (y - y1);
  sb = dx02 * (y - y0);
  for(; y<=y2; y++) {
    a   = x1 + sa / dy12;
    b   = x0 + sb / dy02;
    sa += dx12;
    sb += dx02;
    /* longhand:
    a = x1 + (x2 - x1) * (y - y1) / (y2 - y1);
    b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
    */
    if(a > b) swap(a,b);
    drawFastHLine(a, y, b-a+1, color);
  }
}

void Adafruit_GFX::drawBitmap(int16_t x, int16_t y,
                  const uint8_t *bitmap, int16_t w, int16_t h,
                  uint16_t color) {

  int16_t i, j, byteWidth = (w + 7) / 8;
  for(j=0; j<h; j++) {
    for(i=0; i<w; i++ ) {
      if(pgm_read_byte(bitmap + j * byteWidth + i / 8) & (128 >> (i & 7))) {
    drawPixel(x+i, y+j, color);
      }
    }
  }
}


#if ARDUINO >= 100
size_t Adafruit_GFX::write(uint8_t c) {
#else
void Adafruit_GFX::write(uint8_t c) {
#endif
  if (c == '\n') {
    cursor_y += textsize*8;
    cursor_x = 0;
  } else if (c == '\r') {
    // skip em
  } else {
    drawChar(cursor_x, cursor_y, c, textcolor, textbgcolor, textsize);
    cursor_x += textsize*6;
    if (wrap && (cursor_x > (_width - textsize*6))) {
      cursor_y += textsize*8;
      cursor_x = 0;
    }
  }
#if ARDUINO >= 100
  return 1;
#endif
}

// draw a character
void Adafruit_GFX::drawChar(int16_t x, int16_t y, unsigned char c,
                uint16_t color, uint16_t bg, uint8_t size) {

  if((x >= _width)            || // Clip right
     (y >= _height)           || // Clip bottom
     ((x + 5 * size - 1) < 0) || // Clip left
     ((y + 8 * size - 1) < 0))   // Clip top
    return;

  for (int8_t i=0; i<6; i++ ) {
    uint8_t line;
    if (i == 5)
      line = 0x0;
    else
      line = pgm_read_byte(font+(c*5)+i);
    for (int8_t j = 0; j<8; j++) {
      if (line & 0x1) {
        if (size == 1) // default size
          drawPixel(x+i, y+j, color);
        else {  // big size
          fillRect(x+(i*size), y+(j*size), size, size, color);
        }
      } else if (bg != color) {
        if (size == 1) // default size
          drawPixel(x+i, y+j, bg);
        else {  // big size
          fillRect(x+i*size, y+j*size, size, size, bg);
        }
      }
      line >>= 1;
    }
  }
}

void Adafruit_GFX::setCursor(int16_t x, int16_t y) {
  cursor_x = x;
  cursor_y = y;
}


void Adafruit_GFX::setTextSize(uint8_t s) {
  textsize = (s > 0) ? s : 1;
}


void Adafruit_GFX::setTextColor(uint16_t c) {
  textcolor = c;
  textbgcolor = c;
  // for 'transparent' background, we'll set the bg
  // to the same as fg instead of using a flag
}

 void Adafruit_GFX::setTextColor(uint16_t c, uint16_t b) {
   textcolor = c;
   textbgcolor = b;
 }

void Adafruit_GFX::setTextWrap(boolean w) {
  wrap = w;
}

uint8_t Adafruit_GFX::getRotation(void) {
  rotation %= 4;
  return rotation;
}

void Adafruit_GFX::setRotation(uint8_t x) {
  x %= 4;  // cant be higher than 3
  rotation = x;
  switch (x) {
  case 0:
  case 2:
    _width = WIDTH;
    _height = HEIGHT;
    break;
  case 1:
  case 3:
    _width = HEIGHT;
    _height = WIDTH;
    break;
  }
}

void Adafruit_GFX::invertDisplay(boolean i) {
  // do nothing, can be subclassed
}


// return the size of the display which depends on the rotation!
int16_t Adafruit_GFX::width(void) {
  return _width;
}

int16_t Adafruit_GFX::height(void) {
  return _height;
}
```

Appendix C: Proof of Concept Photo



This photo shows the system in steady-state, with no solar panel attached. The display screens indicate that the battery is powering the system at 5.521W, and that the power delivered to the inverter and the DC/DC converter (at idle) totals 1.542W. From this, it can be inferred that the idle-power consumption of the EP Solar Charge Controller is 3.979W.