Portable Power Wall
Formal Report

Prepared for:
COMM 2443 – Susan Woo
ELEX 4440 – Kathy Manson
ELEX 4560 – Ed Casas
Project Supervisor – Mahda Jahromi

Prepared by:
Islay Clare-Kellett -
Katharyn Taylor -
April 16, 2019

Table of Contents

List of Illustrations

Executive Summary

The Power Wall Project is a custom-piece of power equipment designed to provide an electrical source, on-demand when the equipment is away from standard electrical outlet. The team of Islay Clare-Kellet (Power) and Katharyn Taylor (Telecom) worked closely together to design, procure materials and components, create detailed design, build, test, commission, and document the project. The overall cost for the project materials went over budget of $800 by $200 due to replacement of equipment during the commissioning phase.

During the seven phases of the project plan, the team experienced accomplishments, like completing the working version of the Power Wall. The first prototype the current Power Wall performs well and provides three voltage levels and can be started and stopped simply with the two push buttons and latching relays. The physical unit is can easily be used a camping power supply and it isn't the typical and polluting generator one can find at their local Canadian Tire story. The finished unit is rugged and can be rolled over rough terrain. It will prove to be an excellent camping companion when needing to power phones, coffee makers or lights.

The team also experienced a number of challenges on the technical and human resource sides. On the technical side, the team learned the timing and requirements to carry out in-depth research on limits, capabilities and common issues related to the equipment and processes associated with the design and production of the equipment. Should we have conducted more research, we would have discovered that there was a common issue with turning the analog signal from AM712 to a digital sensor. We also discovered, the hard way, that giving too much voltage to the input pins would kill the pi.

In the end, though we were able to construct a working prototype of the Power Wall, with power monitoring capabilities, we learned important lessons on being accountable to the Project Plan deadlines, on team communication and on personal commitment to the project objectives. While these human resource challenges almost seemed insurmountable during the project implementation, it was our perseverance and our team commitment that kept us motivated and focussed on the end goal. We are proud of our work and thankful for the experience.

Introduction

Project Overview

Design, create and test the development of a portable power wall, which provides electrical power-on-demand away from a standard electrical outlet. The eventual use of the portable power wall be used in and outside of recreational vehicle, to power electrical equipment from fridges, phones and stereos.
Clients have asked for a portable source of electrical power for use on camping trips and other vacations where no standard electrical outlets are available (i.e. accessing power in the woods, on top of mountains or in the middle of the desert.)

Power Side

The purpose of this project was to design and build a Portable Power Wall to provide 5V DC, 12V DC, and 120V AC power when camping in remote areas. This is to cut down on the need to run a noisy and polluting Generator, to reduce the risk of draining your vehicle battery, and to allow you to continue running your electrical devices if others in your group need to take the vehicle away for any reason.

Telecom Side

The purpose of this project was to design and build a monitoring device to measure the Power Wall's output from a distance (away from the equipment itself). We enhance the design to make it a "smart" Power Wall, by adding a Raspberry Pi and sensors, allowing Power Wall t the ability to broadcast information about itself to a webserver on its on private wi-fi network.

Project Description

Design and Layout

Power Side

The design called for an easily portable container to house both the power and monitoring equipment, (covered in the telecom portion of the report), and for the power wall to provide 5V DC for charging small electronics, 12V DC for a small portable recreation refrigerator, and 120V AC to charge larger electronics like a laptop as well as to run camp lights. The first step was to outline loosely how all the components would connect to each other shown in Figure 1 below. This showed how we would use a main bus to connect the four main sections of the power wall together.

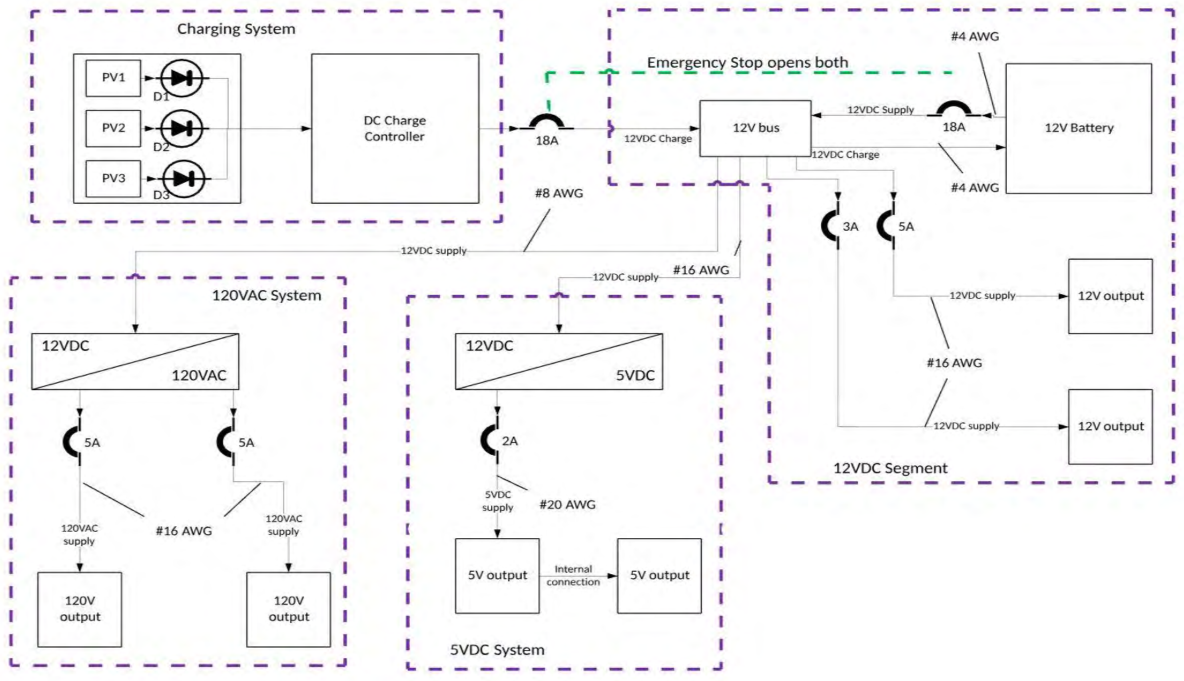


Figure 1 - System Architecture with original protection plan

From that stage we were given a Kirkland marine deep cycle battery to use as our primary power source and selected the converters we would use to turn the 12V DC supplied by the battery into 5V DC and 120V AC. The parts selected were a 15W DROK 12V to 5V buck converter setting the maximum current in the 5V circuit to 3A, and a Giandel 600W continuous and 1200W peak modified sine wave inverter to provide the 120V this set the maximum current of the AC circuit to 5A continuous. This inverter also contained the charge controller that took the input from the solar panels and fed it through the 12V bus to the battery.

Once the components had been selected, we designed the protection system with values based off of the recommended maximum values from the manufacturers and the wire run lengths inside the unit housing. We also added a start and stop button that incorporated latching relays to allow for an emergency stop without opening the unit if required.

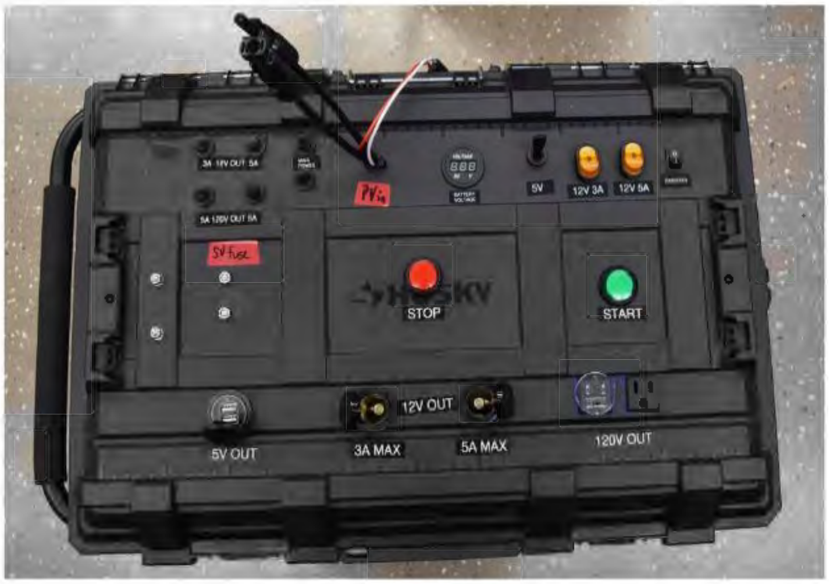


Figure 2 - Lid interface layout

With the circuitry design done the final layout of the main panel, located on the unit lid was also finalized as shown in Figure 2. It was designed so that all similar items were grouped together with protection in the top left, output switches in the top right, and outputs along the bottom. The main start and stop buttons are located in the middle and are illuminated so that you can find them when operating the unit in low light environments.

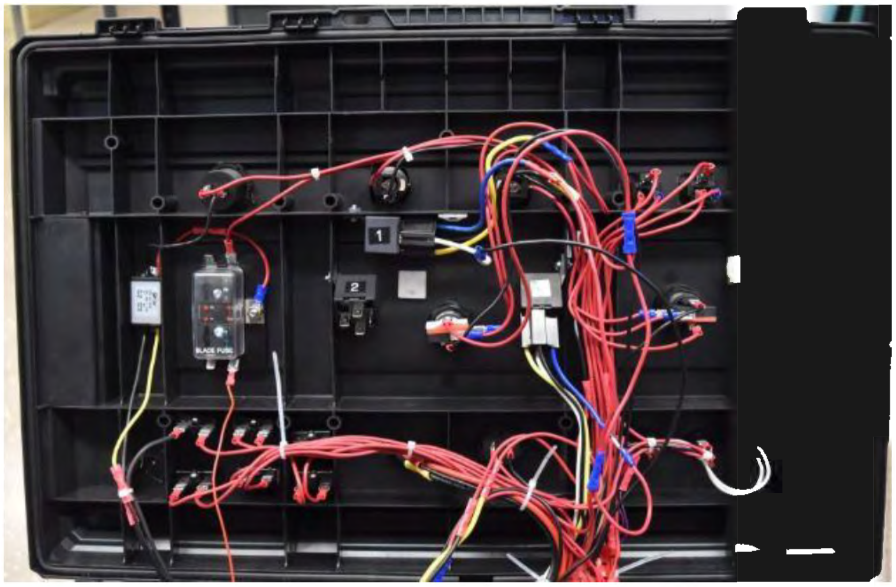


Figure 3 - Under lid wiring and components

The exact placement of the lid components was dictated by the webbing formed as a part of the manufacturing process. The above Figure also shows the relays used to connect the main 12V bus to the battery and charging system as well as the protection for the 5V system and the DROK 12V to 5V converter. Most of the components are automotive to allow for easy acquisition and replacement of components while on the road.

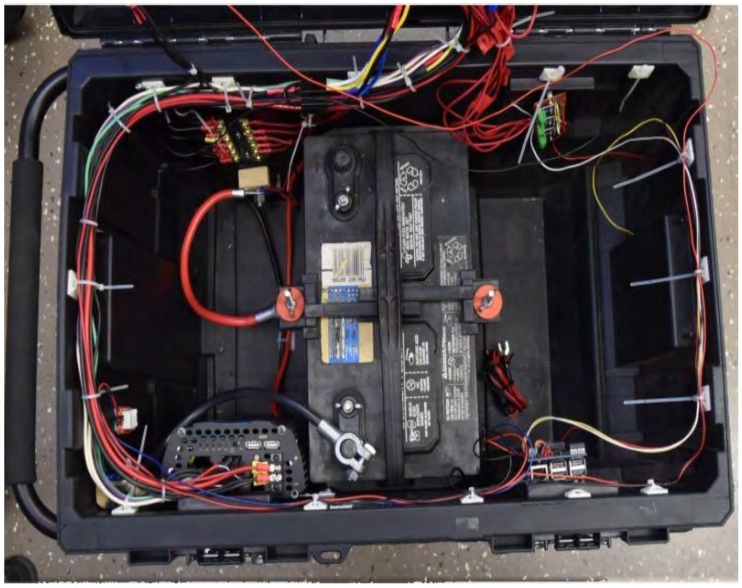


Figure 4 - Unit internals

The interior layout was dictated by the battery size compared to the overall box geometry as seen in Figure the left side holds the power equipment and the right contains the telecom monitoring sensors and raspberry pi. The mounting tray for the battery is epoxied to the base of the box and then using a standard automotive battery hold down strap and J-bolts. There are also wooden blocks used to isolate the main battery terminal lugs from the outside of the box to prevent accidental shocks or injury to the user while the battery is connected for use. Above the positive battery lead connection is the 12V terminal strip and across the box you can see the inverter and the connections to the solar panels.

Telecom Side

In order to monitor how the Powerwall is operating there must be some sort of monitoring circuit. It is broken up into 2 main parts Figure 3: the user interface, the webserver and the data collection, the sensors. The raspberry pi, being the brains of the monitoring system, is responsible for creating a private wi-fi to broadcast the webserver with the Powerwall's current status information. The Pi periodically checks the data from the sensors and automatically updates the information on the webserver. The sensors that were chosen for this project were the INA 219, the AM2302, and ASC 712. All reference material and wires guides are in Appendix IV – Raspberry Pi Documentation and Appendix V – Sensor Information respectively.

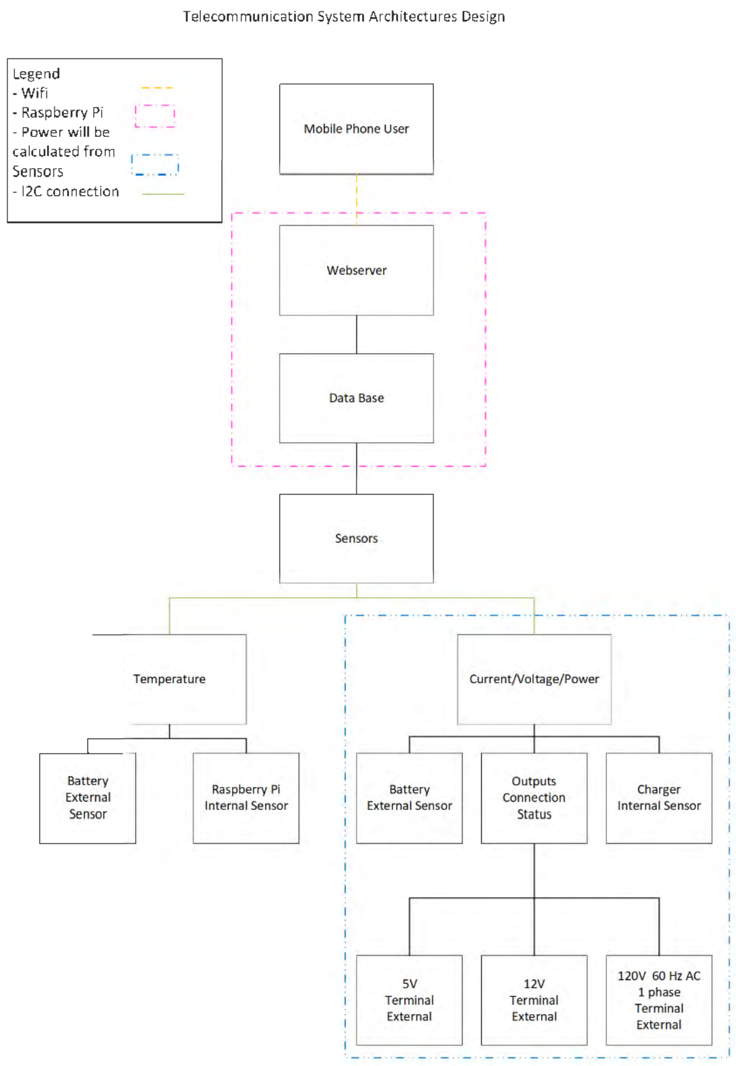


Figure 3 - Powerwall monitoring system

The INA 219 Figure 4 is a current sensor that can calculate the voltage across its shunt resistor. It uses I2C (I^2C, IIC or Inter-Integrated Circuit) protocol to communicate with the raspberry pi. We used the I2C pins on the raspberry pi to send data from the sensors to the pi. We planned to use the sensors to monitor the 5-volt output, the 12-volt output and the 120-volt 60 Hz AC. Unfortunately, we did not read the data sheet of the sensor carefully and found out when testing that the sensor can only handle up to 32-votls. As a result of supplying too much voltage to the sensor it blew up. We were forced to abandon monitoring the AC power since we did not leave enough time to source, purchase and code for a new sensor.

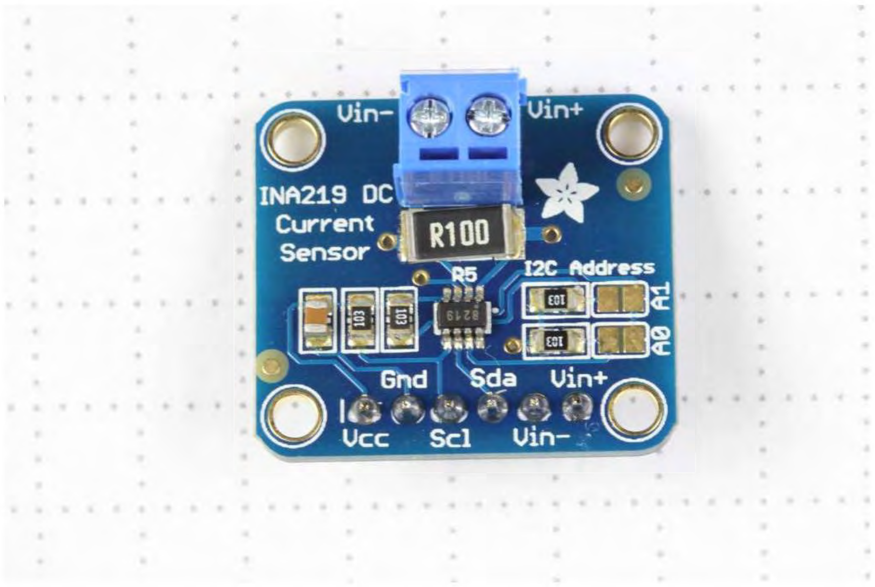


Figure 4 - INA 219 Current and Voltage Sensor

The AM2302 Figure 4 is a temperature and humidity sensor. It measures the temperature and humidity and sends the information via a digital signal to the pi. We chose to connect the AM2302 to GPIO pin number 27, it can be connected to any GPIO pin as long as it is configured properly on the pi. All we had to do for this sensor is specify how many numbers we wanted to display on the webserver.

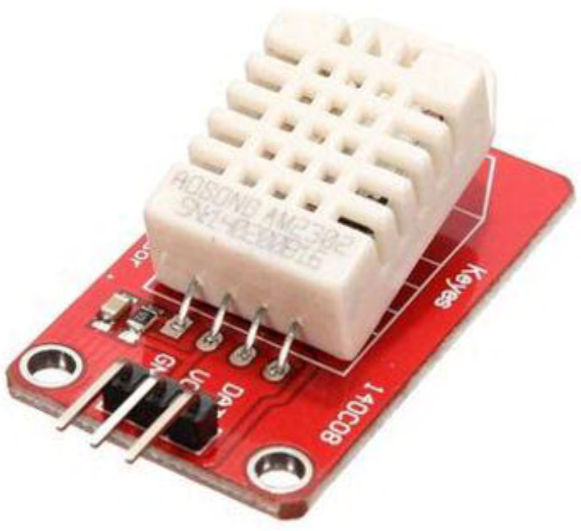


Figure 5 - AM2303 Temperature and Humidity Sensor

ASC 712 Figure 6 is a 20A current sensor. We indented to use this sensor to monitor the battery itself. When programing the sensor, we found out that this sensor outputs analog data, which the raspberry pi will not read. After doing some research we found most people were using the MCP 3008 IC chip, analog to digital converter. However, even with the ADC chip we were unable to get the sensor to send meaningful data to the pi and we had to abandon the battery status monitoring.

Figure 6 - ASC 712 20A Current Sensor

Operation

Power Side

The operation of the power side of the power wall was designed to be very straight forward for the end user. All they need to do is press the green start button seen in Figure 2 until it lights up, they then release the button and can turn on any or all of the output switches as needed. When power is applied at the same time as the start and stop buttons illuminate the battery voltage indicator will also begin to display the current battery voltage at the 12V terminal strip. It should display between 12.3V and 12.8V when charged and up to 14.2 when charging, if the meter indicates 11.8 or less turn off the device and connect the PV panels if not already attached and allow time in full sun to recharge or remove the battery from the box and attach to a plug in lead acid battery charger.

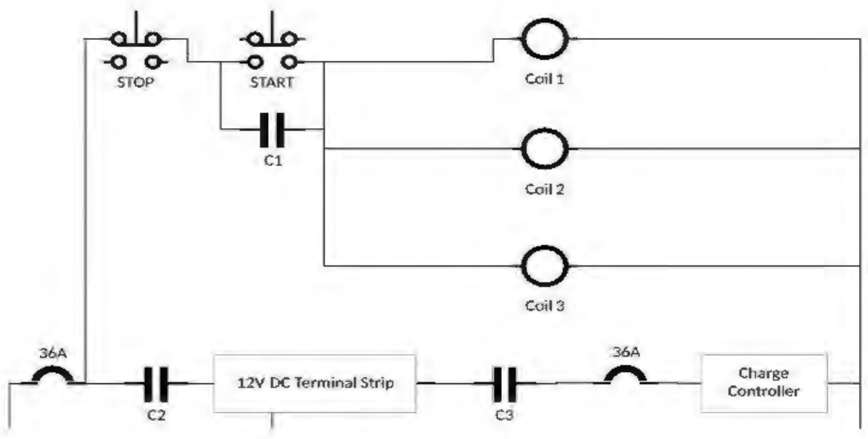


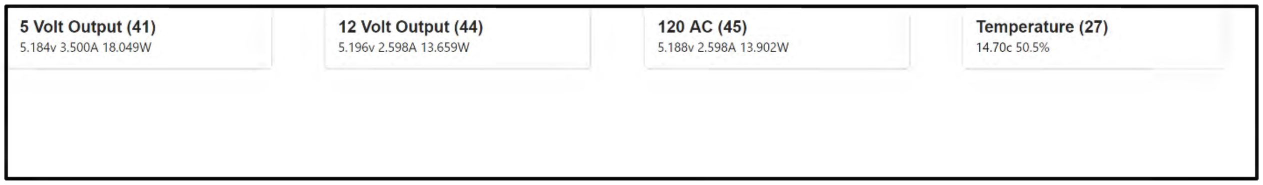
Figure 7 - Main Power Control Circuit

To achieve the latching on start there are three relays shown installed in Figure and indicated in the schematic excerpt in Figure 7 the full wiring diagram can be found in Appendix II – Full Wiring Diagram and Discussion. When the start button is pressed all three coils are energized and coil 1 seals in the start until the stop button is pressed opening the circuit. Coil 2 and 3 close the contacts connecting the battery and charging circuits to the main 12V terminal strip respectively. Once the system is powered the output switches in the top right of figure 2 can be turned on to provide whatever combination of the outputs the user requires.

In order to shut down the unit it is best to remove and turn off all outputs before pressing and holding the stop button until you hear the contacts of the relay click. That is not required as the stop button can be pressed whenever it is required and will stop the system, but for safety reasons it is best to turn off any and all outputs not in use in non-emergency stop situations. It is also recommended that the battery terminals not be connected when not in use or during transportation as an extra precaution to prevent drain on the battery or an accidental start while driving. Appendix I – Powerwall Testing Procedures if needed.

Telecom Side

When the raspberry pi is operational, and the sensors are feeding it data the pi makes a wi-fi called sensor pi. After logging into the pi's wi-fi you must open an Internet Brower and go to 10.0.0.1. The webserver will automatically resize its layout, display the sensor information horizontally or vertically, itself base on whether it is viewed on a computer Figure 8 or it is views on a mobile phone Figure 9. It will also resize base on the size of the window you are using to view the information on a computer. All the coding for this project is found in Appendix VI – Code for Raspberry Pi.



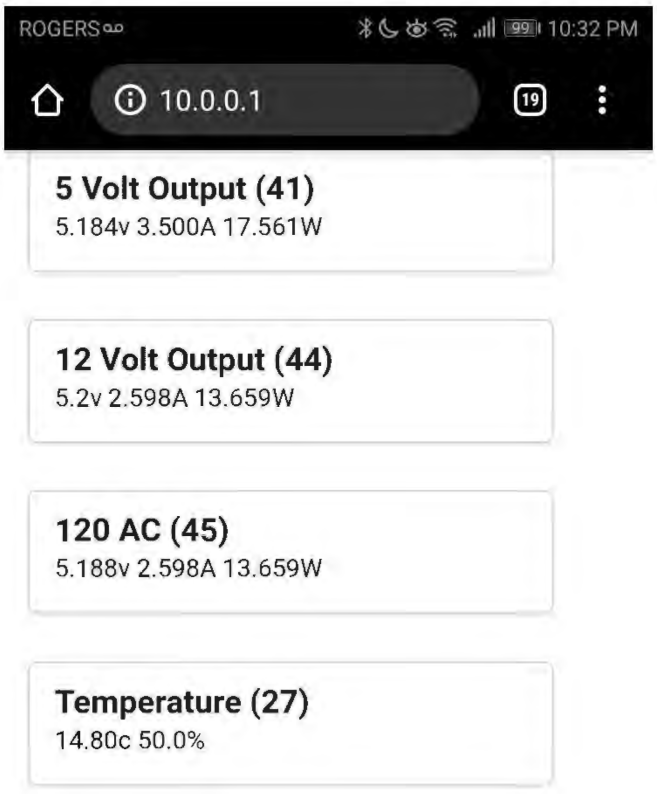
5 Volt Output (41)
5.184v 3.500A 18.049W

12 Volt Output (44)
5.196v 2.598A 13.659W

120 AC (45)
5.188v 2.598A 13.902W

Temperature (27)
14.70c 50.5%

Figure 9 - Webserver on a computer



5 Volt Output (41)
5.184v 3.500A 17.561W

12 Volt Output (44)
5.2v 2.598A 13.659W

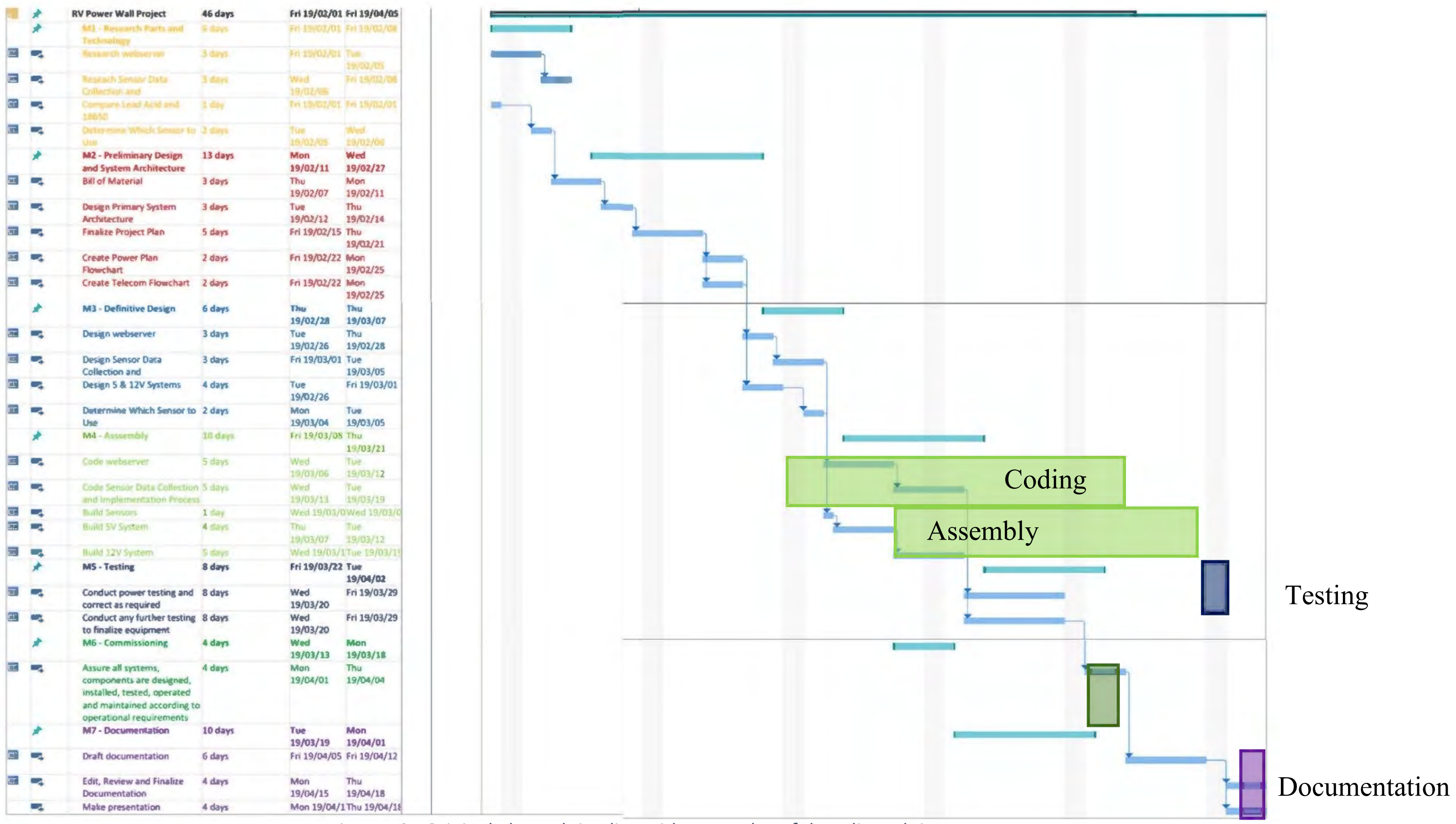
120 AC (45)
5.188v 2.598A 13.659W

Temperature (27)
14.80c 50.0%

Figure 8 - Webserver on a mobile phone

Schedule

Gant Chart



RV Power Wall Project	46 days	Fri 19/02/01	Fri 19/04/05
M1 - Research Parts and Technology	6 days	Fri 19/02/01	Fri 19/02/08
Research webserver	3 days	Fri 19/02/01	Tue 19/02/05
Research Sensor Data Collection and	3 days	Wed 19/02/06	Fri 19/02/08
Compare Lead Acid and 18650	1 day	Fri 19/02/01	Fri 19/02/01
Determine Which Sensor to Use	2 days	Tue 19/02/05	Wed 19/02/06
M2 - Preliminary Design and System Architecture	13 days	Mon 19/02/11	Wed 19/02/27
Bill of Material	3 days	Thu 19/02/07	Mon 19/02/11
Design Primary System Architecture	3 days	Tue 19/02/12	Thu 19/02/14
Finalize Project Plan	5 days	Fri 19/02/15	Thu 19/02/21
Create Power Plan Flowchart	2 days	Fri 19/02/22	Mon 19/02/25
Create Telecom Flowchart	2 days	Fri 19/02/22	Mon 19/02/25
M3 - Definitive Design	6 days	Thu 19/02/28	Thu 19/03/07
Design webserver	3 days	Tue 19/02/26	Thu 19/02/28
Design Sensor Data Collection and	3 days	Fri 19/03/01	Tue 19/03/05
Design 5 & 12V Systems	4 days	Tue 19/02/26	Fri 19/03/01
Determine Which Sensor to Use	2 days	Mon 19/03/04	Tue 19/03/05
M4 - Assembly	10 days	Fri 19/03/08	Thu 19/03/21
Code webserver	5 days	Wed 19/03/06	Tue 19/03/12
Code Sensor Data Collection and Implementation Process	5 days	Wed 19/03/13	Tue 19/03/19
Build Sensors	1 day	Wed 19/03/0	Wed 19/03/0
Build 5V System	4 days	Thu 19/03/07	Tue 19/03/12
Build 12V System	5 days	Wed 19/03/1	Tue 19/03/1
M5 - Testing	8 days	Fri 19/03/22	Tue 19/04/02
Conduct power testing and correct as required	8 days	Wed 19/03/20	Fri 19/03/29
Conduct any further testing to finalize equipment	8 days	Wed 19/03/20	Fri 19/03/29
M6 - Commissioning	4 days	Wed 19/03/13	Mon 19/03/18
Assure all systems, components are designed, installed, tested, operated and maintained according to operational requirements	4 days	Mon 19/04/01	Thu 19/04/04
M7 - Documentation	10 days	Tue 19/03/19	Mon 19/04/01
Draft documentation	6 days	Fri 19/04/05	Fri 19/04/12
Edit, Review and Finalize Documentation	4 days	Mon 19/04/15	Thu 19/04/18
Make presentation	4 days	Mon 19/04/1	Thu 19/04/1

Figure 10 - Original planned timeline with an overlay of the adjusted time

Like in every project, things did not go according to plan. Assembly and coding took longer than expected so we needed to adjust our schedule, this resulted is 1-2 days or testing and 1-2 days for documentation. The overlay is the adjustment timeline.

Costs (Materials and Labour)

Bill of Materials (BOM)

The total cost for this project is $2400. The work wage is calculated at $1600, based on the estimate of two student technicians at $20/hr. for a 10-week project period, (February 1 to April 1). Total parts cost is calculated at $800 which includes a deep cycle battery, Raspberry Pi B+ and supporting electrical components required to assemble the power wall. For a more detail breakdown of BOM please refer to Appendix III – Bill of Materials.

Part Description	Cost
Control Parts	$ 32.37
Conductors & Relays	$ 108.72
Assembly Parts	$ 477.92
Circuit Protection Parts	$ 26.11
Sensors & Pi	$ 157.67
Total Part Cost	**$ 802.79**
Labour	
2 Students $\frac{\$20}{Hrs} * 4\frac{hr}{week} * 10\frac{weeks}{Project}$	$1600.00
Total Cost Equipment & Parts and Labour	**$2402.97**

Table 1 - Summary of Cost

Problems Encountered & Lessons Learned

Power Side

Problem: Giandel Inverter Mounting bracket
The Giandel inverter mounting instructions dictate that you use the molded mounting bracket on the back of the inverter however the model ordered doesn't include any mounting bracket molded to the housing or otherwise.
Solution: Fabricate mounting points using small L-brackets from Rona
The solution was simple however the problem should never have existed and after conversations with the part manufacturer the error is in the included instructions as the exact model we purchased doesn't include any mounting points and what they recommended was to do as we did or to use a shelf and straps to hold the unit in place.

Problem: Relay 2 contacts not closing
In the relay control circuit 12V were present across the coil of relay 2 however the contacts did not close, and the battery could not connect to the 12V terminal strip and provide power to the system.
Solution: fix socket wiring and replace component with a new 12V automotive relay
We removed the suspect relay from the box which was easy thanks to the standard connectors and checked both the connector wiring and used a bench top power supply and multimeter to check the operation of the relay. It was then discovered that there had been both a small mistake

in wiring the relay connector but with bench top tests it was also found that the coil inside the relay was open circuit and it was replaced.

Problem: Lost exhaust and cooling fans
Between delivery and assembly time the exhaust and cooling fans purchased were stored and lost.
Solution: New fans sourced but no time to install
A replacement set of 12V computer fans were sourced and are now waiting to be installed but due to time constraints the install will occur after the completion of the project. So, for now to prevent overheating and potential gas build up from the operation of the lead acid battery the unit is tested in short intervals and the lid is left open to allow for air and heat exchange freely.

Problem: Mismatch of Solar panel connectors
Since we used our own already owned solar panels, we encountered a mismatch of input connector types. Two panels used the standard connectors found on most solar panels but the third had been converted to use a SAE two pin automotive connector.
Solution: Build Adapter Cable
When paralleling panels, we built an adapter cable to allow the SAE connected panel to connect to the charge controller. A long-term solution would be to select a connector type and use that for all of the panels negating the need for the adapter.

Telecom Side

Problem: Realizing the AM712 20A current sensor send analog data
AM712 is an analog sensor, the raspberry pi can only except digital signals.
Solution: Converting the analog signal to a digital signal
We use the MCP3008 analog to digital converter integrated chip to turn the analog signal from the AM712 to a digital signal, which the raspberry pi can except and utilize. Unfortunately, we still were unable to use the sensor. After more research we found that a lot of people having the same problem using the ADC chip with a raspberry, as well as another group using the same IC chip to convert an analog signal to digital.

Problem: Raspberry Pi stopped working
After setting the raspberry pi all set up to read the sensors and display the data on a webserver, attaching it to the power part of the project and having a sensor blow up, the raspberry pi died. I bought a new pi and tried again. The pi still died.
Solution: Do not give the pi 5V at its input pins
I consulted Ed Casas and we figured out, after doing some research, that giving 5V to the input pins of the pi could kill it. So, I rewired my sensors so that they were only using 3.3V and could not short out the Pi again.

Conclusions

Power Side

As a proof of concept and a first prototype the current power wall performed well, it provides the three voltage levels required and can be started and stopped simply with the two push buttons and latching relays. The physical unit is also acceptable for the intended use as a camping power supply that doesn't rely on a loud and polluting generator or risk draining your vehicle's battery. The unit is capable of being rolled on rough terrain and though long drops or rolling it over are not recommended it is robust enough to bounce down off of curbs and down small ledges comparable to the height of standard stair risers. With the concept suitably proven for the desired task other additions and recommendations can be added and updated as desired without worry that the base design wouldn't function.

Telecom Side

Programming the Raspberry pi proved to be challenging on a number of different levels. For example, we had originally designed the pi to turn the analog signal from AM712 to a digital sensor. During the design phase, we should have conducted more research, then we probably would have learned that this problem was common, and we might have been able to design an alternative method or excluded this feature from the overall design. Though time was spent discovering this during the development, this was an excellent lesson to learn for future projects. During the commissioning of the equipment, we unfortunately discovered through trial and error that giving too much voltage to the input pins would kill the pi. I believe, again, if we had conducted more research on the pi’s limits and requirements, we could have avoided burning out two Pis. It may seem counterintuitive, but I am actually pleased that we did encounter these problems. It helped us understand as a team the importance of project planning, executing phases and spending more time during the planning and research phases to limit the depth and range of problems that occurred further down the project phases.

Recommendations

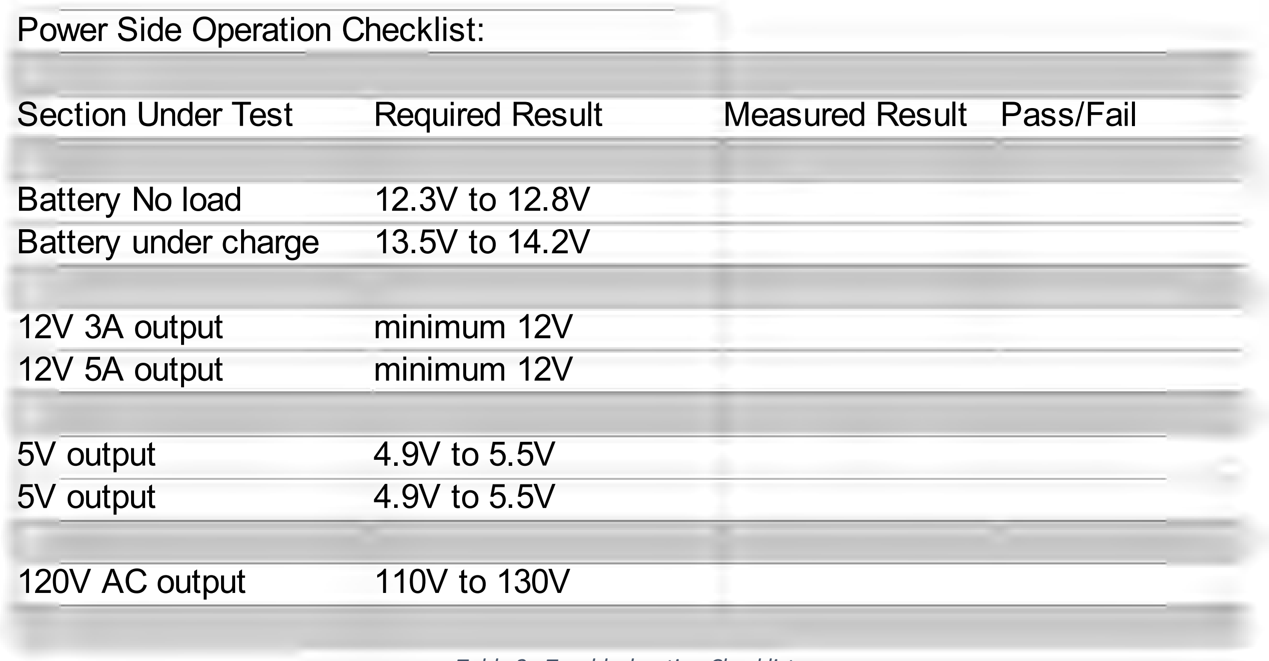
Power Side

1. Replace Flooded Lead Acid battery with comparable Lithium Ion to reduce weight and size of housing as well as to remove gas build up hazard from the operation of the lead acid battery.
2. Change Charge controller to standalone unit for Lithium charging.
3. Upgrade 12V to 5V buck converter from 15W unit to larger 25W unit. 3A max is too low to be useful for charging more than one device while the raspberry pi is active.
4. Select a single type of connector to use all solar panels and convert input to use only one line to connect to charge controller.

Telecom Side

1. User all digital output sensors when using a Raspberry Pi so that no ADC is needed.
2. Use an Arduino, that way you are not limited to using only digital out sensors. Also, the Arduino has a low power mode, which would reduce its power consumption when not executing code. There is more example code for sensor connected to Arduinos then for sensors connected to Raspberry Pis, for the applications we are using them for.
3. Going to all information about sensors whether it is: analog or digital, there is example code for a Raspberry Pi or Arduino and if the operating conditions of the sensors is compatible with your set up.
4. Add some sort of alerts on the webserver, to be triggered when a fault is detected.

Appendix I – Powerwall Testing Procedures



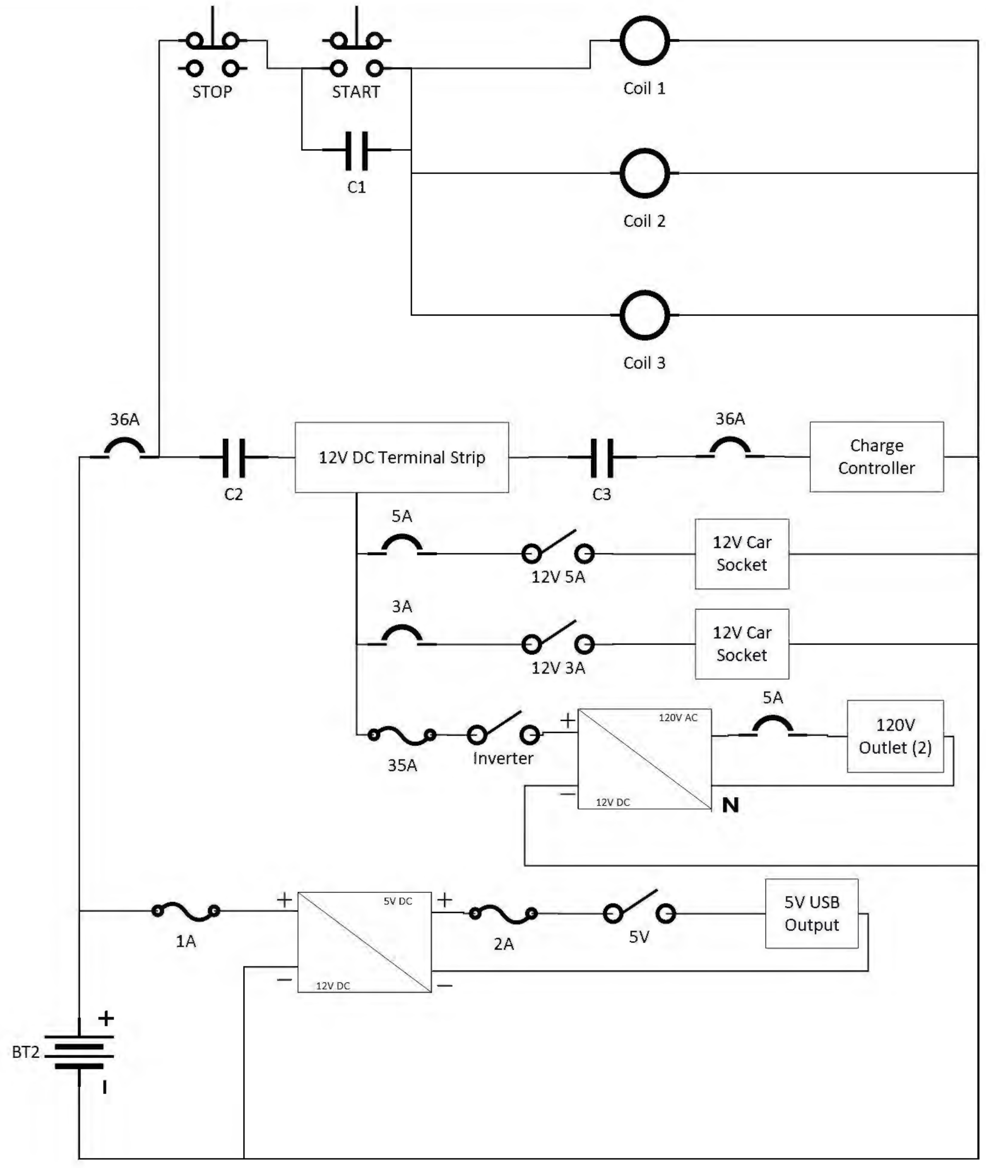
Power Side Operation Checklist:

Section Under Test	Required Result	Measured Result	Pass/Fail
Battery No load	12.3V to 12.8V		
Battery under charge	13.5V to 14.2V		
12V 3A output	minimum 12V		
12V 5A output	minimum 12V		
5V output	4.9V to 5.5V		
5V output	4.9V to 5.5V		
120V AC output	110V to 130V		

Table 2 - Troubleshooting Checklist

The above Table 2 - Troubleshooting Checklist Table 2 is a sample of the table used to verify the basic function of the power side of the power wall project. The tests were simple voltage checks also used to verify that the sensors that report the data to the raspberry pi web server are reporting correctly. The 12V to 5V converter has a small potentiometer on the back so it can be adjusted based on the results gathered during the basic function test to bring it back to where it should be for proper operation.

Appendix II – Full Wiring Diagram and Discussion



The above wiring diagram shows how all the different systems were integrated inside the box with the protection equipment and switches. At the top is the control circuit that connects the battery and charging system to the terminal strip. In the lower half of the diagram the connections between the terminal strip and the converters is shown. It is also shown at the bottom of the page how the 12V to 5V converter provides a hot bus for the raspberry pi for testing that could easily be removed and connected into the terminal bus later with only the replacement of one crimp on ring terminal.

Appendix III – Bill of Materials

Detailed Bill of Material

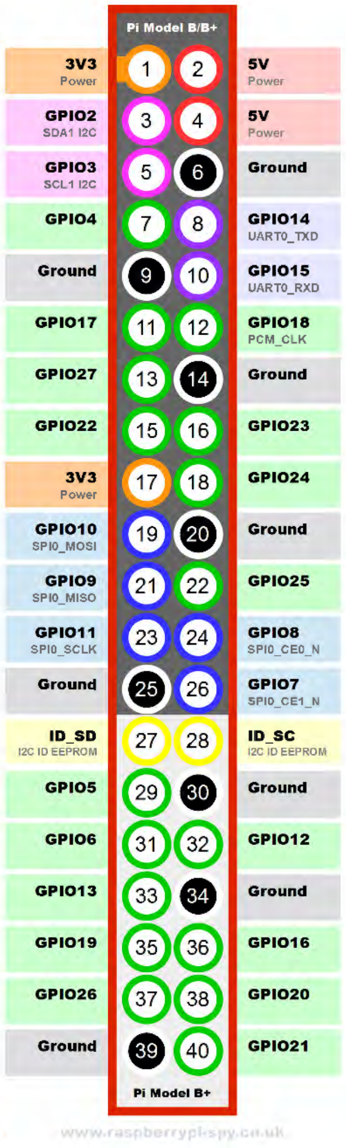
Nomenclature	Part Number	Qty.	Price	Extended
		Control		
Relay	933332011	1	10.29	10.29
Relay – SPST	50-835H-0	1	3.48	3.48
Relay – SPSD	50-835H-1	1	3.48	3.48
Switch – Amber Rocker	9411-7-11	2	7.56	15.12
				Subtotal 32.37
		Conductors & Related		
Relay Socket	RS40	1	19.98	19.98
Heat Shrink 3/8”	20-6997-4	1	3.99	3.99
Female Faston – Blu (box)	20-6919-2	1	2.49	2.49
Female Faston – Ylw (box)	020-6923-0	1	3.49	3.49
Female Faston – Red (box)	020-6967-6	1	9.90	9.90
Ring Terminal – Red #8 (box)	1704-15	1	14.37	14.37
Battery Lead – Red	6201-0-BP	1	7.46	7.46
Battery Lead – Black	6201-5-BP	1	7.46	7.46
USB 12V switch		1	11.20	11.20
12V Power Outlet	56482	2	14.19	28.38
				Subtotal 108.72
		Assembly		
Battery Tray	865-E	1	3.59	3.59
Huskey Chest		1	79.98	79.98
Battery J-Bolts – 8”	W1694C	1	4.54	4.54
12V DC-DC Buck		1	10.99	10.99
DROK Waterproof DC Buck Converter Voltage Regulator		1	13.18	13.18
600W 120V AC inverter		1	48.99	48.99
MC 4 Connectors (M/F)		1	8.16	8.16

12 AWG wire		1	8.49	8.49
Solar Panels (2pack)		1	300.00	300.00
				Subtotal 477.92
		Circuit Protection		
Fuse – 2A ATC (box)	BP/ATC-2-RP	1	2.63	2.63
Circuit Breaker 18A	30-6018	2	4.55	9.10
Circuit Breaker 3A	30-6003	2	3.00	6.00
Circuit Breaker 5A	30-6005	2	4.19	8.38
				Subtotal 26.11
		Sensors & Pi		
INA 219 - Current/Voltage		3		
Raspberry Pi B+		1	70.00	70.00
Protype Pi Shield		1	28.00	28.00
AM2302 - Temperature		1	12.99	12.99
ASC 712 - 20A current sensor		1	16.68	16.68
22 AWG wire pack		1	30.00	30.00
				Subtotal 157.67
				Total for all parts 802.79
				Labour 1600
				Grant total 2403

Appendix IV – Raspberry Pi Documentation

Datasheet for Raspberry Pi: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf
How to start using your Raspberry Pi: https://www.raspberrypi.org/documentation/installation/

Figure 11 - Raspberry Pi 3 B+


Pi Model B/B+

3V3 Power	1	2	5V Power
GPIO2 SDA1 I2C	3	4	5V Power
GPIO3 SCL1 I2C	5	6	Ground
GPIO4	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SPI0_MOSI	19	20	Ground
GPIO9 SPI0_MISO	21	22	GPIO25
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CE0_N
Ground	25	26	GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21

Pi Model B+

www.raspberrypi-spy.co.uk

Figure 12 - GPIO pins for the Raspberry Pi

Appendix V – Sensor Information

INA219

Datasheet: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf
Sample Code: https://pypi.org/project/pi-ina219/
Wiring diagram was taken from the datasheet.

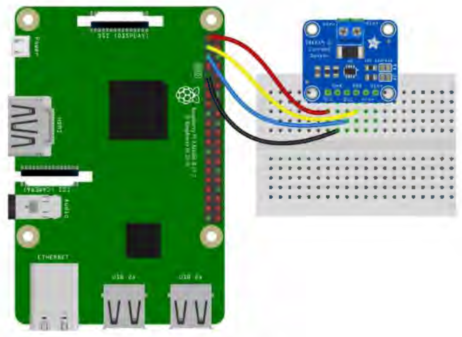


Figure 13 - How to wire up INA219 to Pi

AM2302

Datasheet: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf
Sample Code: https://github.com/adafruit/Adafruit_Python_DHT
Wiring diagram: https://raspberrytips.nl/dht22-temperatuursensor-raspberry-pi/

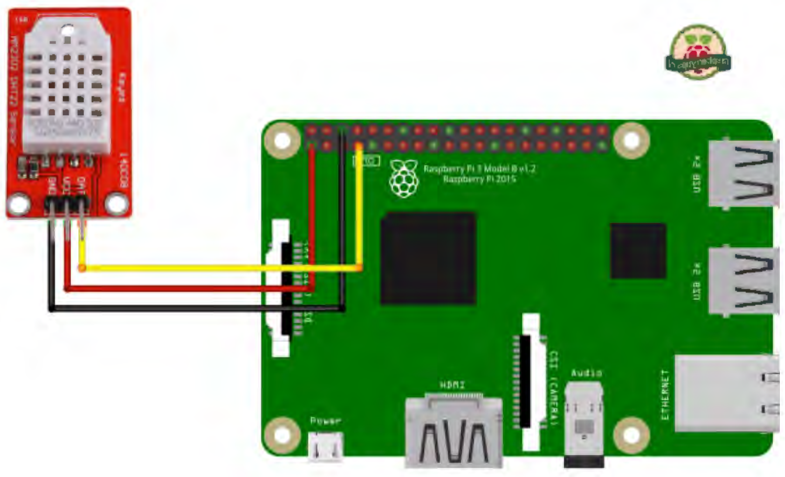


Figure 14 - How to wire up AM2303 to Pi

ASC712

Datasheet: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf
Datasheet for MCP3008: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf
Sample Code & Wiring Diagram: https://www.raspberrypi.org/forums/viewtopic.php?t=173831

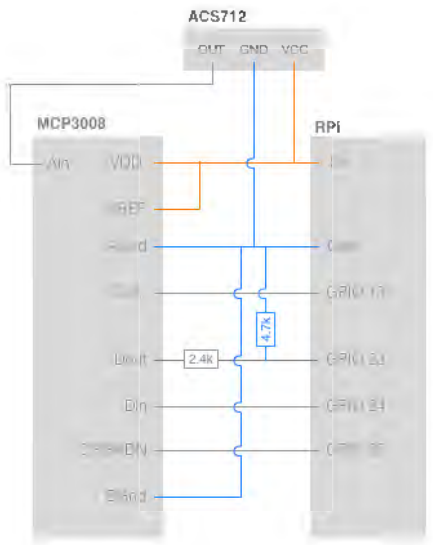


Figure 15 - How to wire up ASC712 & MCP3008 to Pi

Appendix VI – Code for Raspberry Pi

Reference Material:

Getting messages from the sensor daemons and relaying it to webserver: https://mosquitto.org/
How to make python script run as daemons: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files
Building UI: https://facebook.github.io/create-react-app/
Hosting UI: http://nginx.org/
Setting up Pi into an access point:
https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

Webserver:

```
import
React, {
Component
} from
"react";
            import "./App.css";
            import INASensor from "./INASensor";
            import AM2302Sensor from "./AM2302Sensor";
            import { Grid } from 'semantic-ui-react';

            var mqtt = require('mqtt');

            class App extends Component {

              constructor(props){
```

```
    super(props);
    this.state = {};
  }

  render() {
    return (
      <Grid stackable columns={4}>
        <Grid.Column key={1}>
        <INASensor mqtt_host={"ws://10.0.0.1:9001/"} sensorid={41}
topic={"ina219/41/info"} title={"5 Volt Output"}/>
        </Grid.Column>
        <Grid.Column key={2}>
        <INASensor mqtt_host={"ws://10.0.0.1:9001/"} sensorid={44}
topic={"ina219/44/info"} title={"12 Volt Output"}/>
        </Grid.Column>
        <Grid.Column key={3}>

        <INASensor mqtt_host={"ws://10.0.0.1:9001/"} sensorid={45}
topic={"ina219/45/info"} title={"120 AC"} />
        </Grid.Column>

        <Grid.Column key={4}>

        <AM2302Sensor mqtt_host={"ws://10.0.0.1:9001/"} sensorid={27}
topic={"am2302/27/info"} title={"Temperature"} />
        </Grid.Column>

      </Grid>
    );
  }
}

export default App;
```

Daemon for INA219:

```
#!/usr/bin/env python
from ina219 import INA219
import time
SHUNT_OHMS = 0.1

sensor_addresses = [0x40, 0x41, 0x44, 0x45]
inas = []

# Configure sensor objects
for sensor_address in sensor_addresses:
  print("Configuring {0:02X}".format(sensor_address))
  try:
    ina = INA219(SHUNT_OHMS, address=sensor_address)
    ina.configure()
    inas.append((sensor_address, ina))
  except:
    print("Unable to configure.")

# Iterate and print power

for ina in inas:
  addr,sensor = ina
  sensor.wake()
  print("{0:02X} {1} mW".format(addr,sensor.power()))
  sensor.sleep()
```

JSX for INA219:

```
 import
 React
 from
 "react";
            import { Container, Card } from "semantic-ui-react";
            import { subscribe } from "mqtt-react";

            export default class INASensor extends React.Component {
             constructor(props) {
              super(props);
                  this.state = {};
             }

             componentDidMount() {
                  var mqtt = require('mqtt');
                  this.client = mqtt.connect(this.props.mqtt_host);
                  this.client.subscribe("ina219/" + this.props.sensorid + "/info");
                  this.client.on('message', (topic, message) => {
                          var realMessage = JSON.parse(new
            TextDecoder().decode(message));
                          this.setState(realMessage);
                  })
             }

             render() {
                  let content = <Card.Content>Loading...</Card.Content>

                          if(Object.keys(this.state).length != 0) {
```

```
                          content = <Card.Content>{this.state.voltage}v
{this.state.current.toFixed(3)}A {this.state.power.toFixed(3)}W </Card.Content>
                }



    return (
      <Card>
        <Card.Content>
          <Card.Header>{this.props.title} ({this.props.sensorid})</Card.Header>
                    {content}
        </Card.Content>
      </Card>
    );
  }
}
```

JSX for AM2302:

```
import
React
from
"react";
            import { Container, Card } from "semantic-ui-react";

            export default class AM2303 extends React.Component {
              constructor(props) {
                super(props);
                this.state = {};
              }

              componentDidMount() {
                var mqtt = require("mqtt");
                this.client = mqtt.connect(this.props.mqtt_host);
                this.client.subscribe("am2302/" + this.props.sensorid + "/info");
                this.client.on("message", (topic, message) => {
                  var realMessage = JSON.parse(new TextDecoder().decode(message));
                  this.setState(realMessage);
                });
              }

              render() {
                let content = <Card.Content>Loading...</Card.Content>;
                let high_temperature = 25;
                if (Object.keys(this.state).length === 0) {
                } else {
                    let span_tag = this.state.temperature > high_temperature ? "red" : "black";
```

```
      content = (
      <Card.Content>
        <span style={{color: span_tag}}
>{this.state.temperature.toFixed(2)}c</span> {this.state.humidity.toFixed(1)}%{"
"}
      </Card.Content>
    );
   }

   return (
    <Card>
      <Card.Content>
        <Card.Header>
          {this.props.title} ({this.props.sensorid})
        </Card.Header>
        {content}
      </Card.Content>
    </Card>
   );
  }
}
```

Daemon for AM2302:

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt
import Adafruit_DHT
import time
import sys
from daemonize import Daemonize
import json

# Configuration
sensor_pins = [27]
sleep_amount = 1.0 # In seconds
max_read_failures = 10 # How many times can a read fail before giving up.

def main():
    client = mqtt.Client()
    client.connect("127.0.0.1")
    read_failure = 0
    while True:
        if read_failure >= max_read_failures:
            print("[ERROR]: Unable to read sensor {0} times. Shutting
down.".format(max_read_failures))
```

```
            break
        try:
            humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 21)
            client.publish("am2302/{0}/info".format(sensor_pins[0]),json.dumps({"humidity":
humidity, "temperature": temperature, "timestamp": int(round(time.time()))}))
        except:
            read_failure += 1
            print("[WARNING]: Unable to read sensor. Waiting 5 seconds.")
            time.sleep(5)

        time.sleep(sleep_amount)

if __name__ == "__main__":
    main()
```