Senior Project:

SOLAR-POWERED HOT TUB

Designed by:

Jonathan Peterson and Anthony Zepeda

Advised by:

Dr. Ali Shaban

May 29, 2016

Electrical Engineering Department

California Polytechnic State University

TABLE OF CONTENTS

TABLE OF CONTENTS ii

TABLE OF FIGURES iii

TABLE OF TABLES v

ABSTRACT vi

1. INTRODUCTION 1

2. BACKGROUND 4

3. DESIGN 5

4. CONSTRUCTION 19

5. DATA 39

6. ANALYSIS 50

7. CONCLUSION 55

8. FUTURE WORK 57

APPENDIX A: DIVISION OF LABOR 60

APPENDIX B: SCHEDULE 61

APPENDIX C: COST ANALYSIS 63

APPENDIX D: BILL OF MATERIALS 64

APPENDIX E: ANALYSIS OF SENIOR PROJECT DESIGN 65

APPENDIX F: USER-INTERFACE SOURCE CODE 68

APPENDIX G: MATLAB SCRIPT FOR ENERGY CALCULATIONS 93

APPENDIX H: WORKS CITED 95

TABLE OF FIGURES

Figure 1.1: Study on Hot Tub and Pool Owners in the U.S. [10] 1

Figure 3.1: Hardware High Level Block Diagram 5

Figure 3.2: Hardware High Level Model 5

Figure 3.3: Hot Tub Block Diagram 7

Figure 3.4: Hot Tub Model 7

Figure 3.5: Solar Panel Assembly Block Diagram 9

Figure 3.6: Solar Panel Assembly Model 10

Figure 3.7: Solar Water Heater Assembly Block Diagram 11

Figure 3.8: Solar Water Heater Assembly Model 12

Figure 3.10: User-Interface High-Level Programming Flow-Diagram 13

Figure 3.11: User-Interface Schematic 15

Figure 3.12: Relays Schematic 17

Figure 3.13: Switch-Debounce Circuitry Schematic 18

Figure 4.1: Hot Tub with Pump and Wiring Removed 19

Figure 4.2: New Spa Pump Piping 20

Figure 4.3: Pre-Pump-Installation Plumbing Connections 21

Figure 4.4: Existing Hot Tub Plumbing (Insulation Removed) 22

Figure 4.5: Existing Ports chosen for Hot Water Heater Assembly 23

Figure 4.6: Final Plumbing for Hot Water Heater Assembly 24

Figure 4.7: Hot Tub Plumbing Leak Repairs 25

Figure 4.8: Finished Hot Tub Base 26

Figure 4.9: Pre and Post Finished Side Panel 27

TABLE OF FIGURES (CONTINUED)

Figure 4.10: Finished Spa Pump and Plumbing with Flow Control and Bleed Valves 28

Figure 4.11: Preliminary and Final Water Heater Apparatus 30

Figure 4.12: Heat Exchanger Construction 31

Figure 4.13: Completed Copper Tubing 32

Figure 4.14: Prepped Copper Tubing 33

Figure 4.15: Completed Heat Exchanger 33

Figure 4.16: Completed Housing for Electrical Components 34

Figure 4.17: Materials for Control Circuitry Housing 36

Figure 4.18: Completed Hardware for User Interface 37

Figure 4.19: Control Wiring for User Interface 38

Figure 5.1: Battery Discharge Rate 43

TABLE OF TABLES

Table 2.1 System Specifications 4

Table 3.1: High Level Block Diagram Descriptions 6

Table 3.2: Hot Tub Model Descriptions 8

Table 3.3: Solar Panel Model Descriptions 10

Table 3.4: Solar Water Heater Model Descriptions 12

Table 3.7: User Interface System Settings 14

Table 3.8: User Interface Schematic Port Descriptions 16

Table 4.1: Solar Vacuum Tube Temperature Data 29

Table 5.1: Breakdown of Power Consumption 39

Table 5.2: Test Conditions for System Power 41

Table 5.3: Battery and Spa Pump Data 42

Table 5.4: Test Conditions for Lighting Control Operation 42

Table 5.5: Test Conditions for Control of Pumps 48

Table 5.6: Test Conditions for the User Interface (Display Modes) 49

Table 6.1: Design Options Summary 52

Table C.1: Project Cost Analysis 63

ABSTRACT

This paper discusses the design and implementation of a solar-powered hot tub. The concept of this project was to design an independently-operated hot tub powered by a 12V rechargeable battery, charged during the day by a single 400W solar panel. For this purpose, a twenty-year-old name-brand hot tub was purchased, in used condition. The plumbing, AC electrical wiring, and mechanical pumps were all removed and replaced with new components to meet our design specifications. Additionally, a solar water-heater was designed and integrated into the system to directly apply the sun's heat to water pumped in and out of the tub, which significantly reduced the power budget for the system. Furthermore, the tub structure was fitted with energy efficient LED lighting for night-time use. Lastly, a user-friendly control and display unit was designed and embedded into the tub's mechanical structure to allow owners to adjust and set modes of operation, jet and heat cycle times, and lighting options. Our design allows an owner to continually power their hot tub at no additional cost every month. This project served as a channel through which much of our studies in microelectronic, embedded system, and power system design got put into practice.

1. INTRODUCTION

The hot tub industry has made huge strides in profitability since its inception in the 1960s[5]; however, over the past five-year period hot tubs sales have steadily declined (average annual rate of 9.3%) as homeowners slowly recover from the latest economic recession[7]. Figure 1.1 displays the amount of hot tub owners during the time period since the recession.

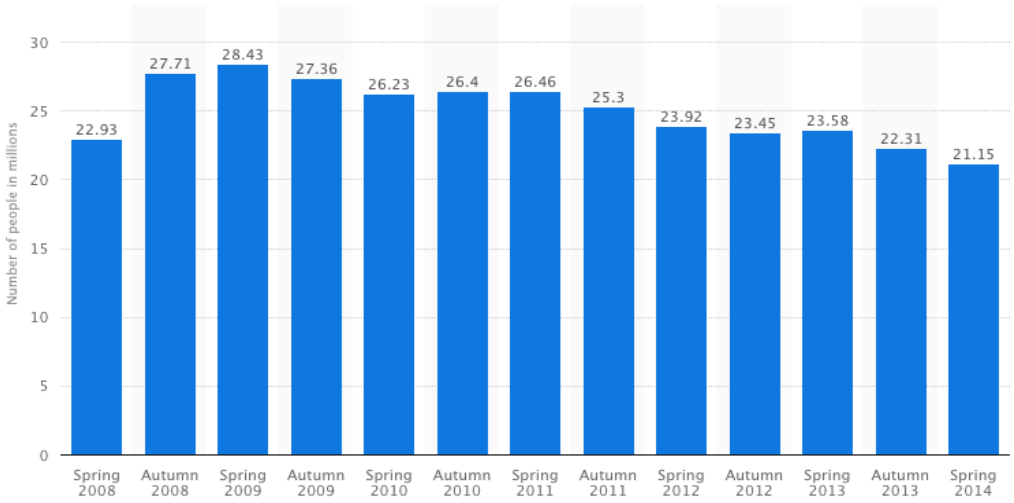


Figure 1.1: Study on Hot Tub and Pool Owners in the U.S. [10]

Aside from the initial cost of purchasing a hot tub, owners must also incur a high cost of electrical energy each month to continually power the tubs; thus there exists a new market for low-energy consumption hot tubs.

Most hot tub manufacturers have realized the need to make their designs more energy efficient. Without completely redesigning their products and overhauling their manufacturing processes, most companies have only made minor changes to existing designs, which results in minor energy savings for consumers. These minor

changes include: swapping out incandescent lighting with energy efficient LED lighting, using higher density foam for insulation, and making more airtight tub covers. Each of these improvements do cut-down the cost of energy, but not on a level that makes hot tubs once again affordable for the general public; in fact, hot tubs featuring these improvements cost more than they did beforehand.

What the hot tub market needs is a complete redesign of hot tub power-management systems. The current electrical grid is set-up to provide residential homes with AC power; thus high-energy consumption devices such as refrigerators, washing machines, and hot tubs require an AC connection in order to be powered from a residential home. However, purchasing regulated AC power from energy providers, such as electric utilities, is very costly and comes with additional fees. One way to eliminate energy costs and fees is to generate power off grid using free sources of energy, such as the sun or wind.

Interestingly enough, the solar industry has experienced extraordinary growth throughout the economic crisis (a 78% increase in sales from 2006-2011) as consumers demand more and more for "green" solutions[14].

However, providing an entire home with off-grid, or even grid-tied, DC power can be very costly: typically ranging from 15k-30k dollars. What makes the most economical sense when powering a single high energy-consumption device, is to power only the device (hot tub) with a small yet adequately sized and affordably priced off-grid solar system.

Therefore, the goal of this project is to design an all-inclusive hot tub that features a solar powered off-grid power management system which currently is not offered by any other hot tub manufacturer. This product could enter the hot tub market at its lowest point in the past decade, taking advantage of a minimal competitor situation and an ever-increasing consumer demand for sustainable appliances.

2. BACKGROUND

The hot tub will consist of the following major subsystems: a 12V 175Ah battery for system power, a 400W solar panel for battery recharging, a direct sunlight heat exchanger with a 15W DC pump for heating and circulation, a 1/8 HP AC pump for main circulation, LED lighting, and a user interface controlled by an embedded MSP430 microcontroller. Table 2.1 displays the system specifications for each subsystem.

Table 2.1 System Specifications

SUBSYSTEM	SPECIFICATION
Battery	• Nominal 12V • Capacity: 175Ah • Power: 2100Whr
Water Temperature	• Temperature Range: 80-110°F • Not selectable within range • Increased temperature is a result of heat pump run time
Heat Pump	• Nominal Operating Voltage: 12VDC • Flow Rate: 3 GPM • Rated Power: 15W
Jet Pump	• Nominal Operating Voltage: 120VAC • Max Amperage: 4A • Flow Rate: 45 GPM • Rated Power: ⅛ HP
LED Lighting	• Nominal Operating Voltage Range: 9V~14.8V • Max Current Draw: 8A • Rated Power: 4.9W per strip (10 strips total) • Average Intensity: 402 Lumens per strip • Dimmable: Yes • LED Lifetime: 40,000 hours
Solar Panels	• Max Voltage: 91V • Max Amperage: 8.39A • Rated Power: 400W • Weight: 44.1 lbs • Dimensions: 52” x 78” x 1.5” • Series Fuse Rating: 40A

3. DESIGN

The hardware of the project will consist of three major systems: The hot tub itself, a solar panel assembly, and a solar water heater assembly. The solar panel will be used to charge the battery that will power all of the pumps and electronics used in the hot tub while the solar water heater will be used to regulate the water temperature. The high-level block diagram of the entire system is shown in Figure 3.1, followed by a model of the overall system in Figure 3.2 and a table describing its components (Table 3.1).

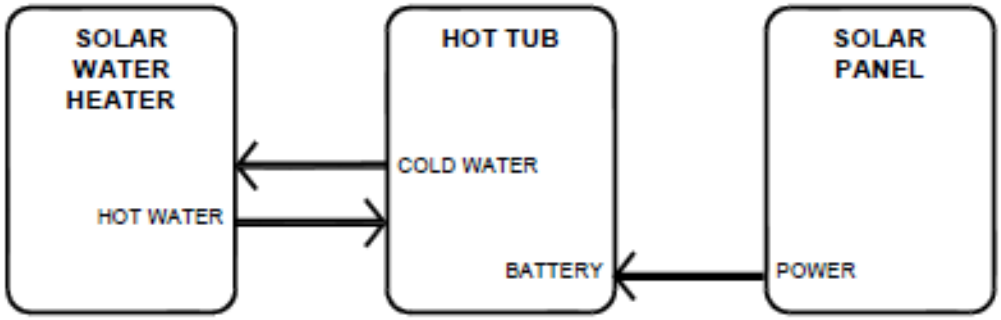


Figure 3.1: Hardware High Level Block Diagram

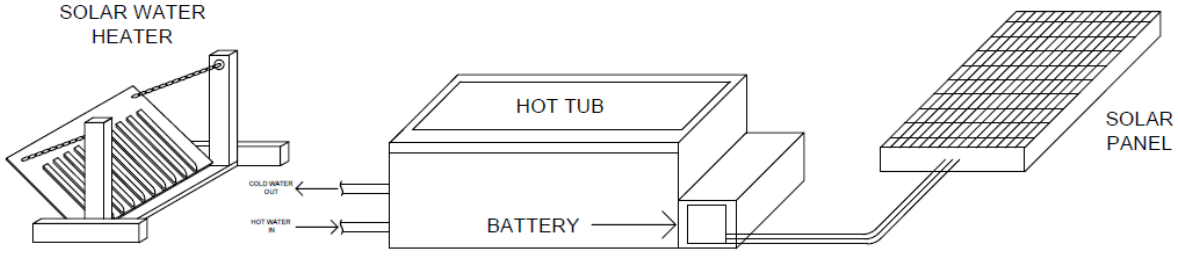


Figure 3.2: Hardware High Level Model

Table 3.1: High Level Block Diagram Descriptions

BLOCK NAME	BLOCK DESCRIPTION	FUNCTIONALITY
Hot Tub	• Hot Tub • Spa Pump • Accessories	Provides the housing of the user interface, the jets, battery, and other accessories.
Solar Panel	• Solar Panel • Charge Controller • 12V Battery • Inverter	Provides all of the necessary electrical power to the hot tub and all of its accessories.
Solar Water Heater	• Solar Water Heater • 12V DC Pump	Provides all of the hot water for the hot tub.

HOT TUB STRUCTURE

The hot tub structure portion of the high-level block diagram consists of all of the accessories associated with the housing of the tub. These include things such as the user interface, the jets, and the lighting. Although there are a lot of components involved with the hot tub portion of the project there are truly only three major inputs. These are the power for the tub, the hot water for the tub, and all of the user inputs through the user interface. A simple block diagram of this is shown in Figure 3.3 followed by a more detailed model (Figure 3.4). A table describing all of the labels is provided in Table 3.2.

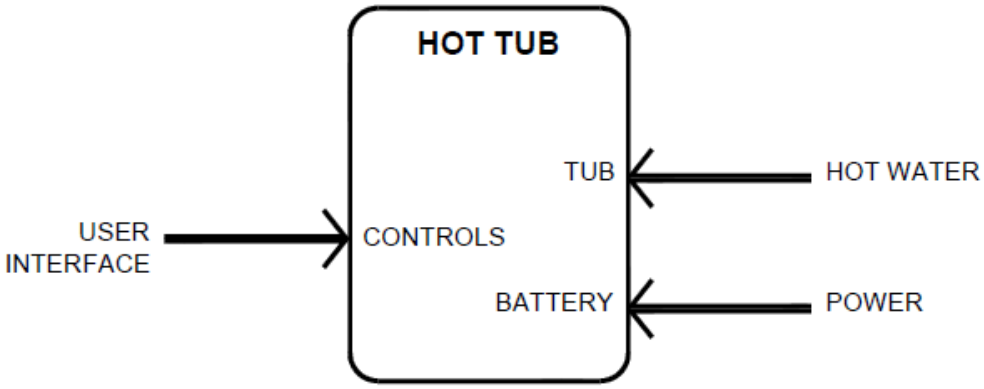


Figure 3.3: Hot Tub Block Diagram

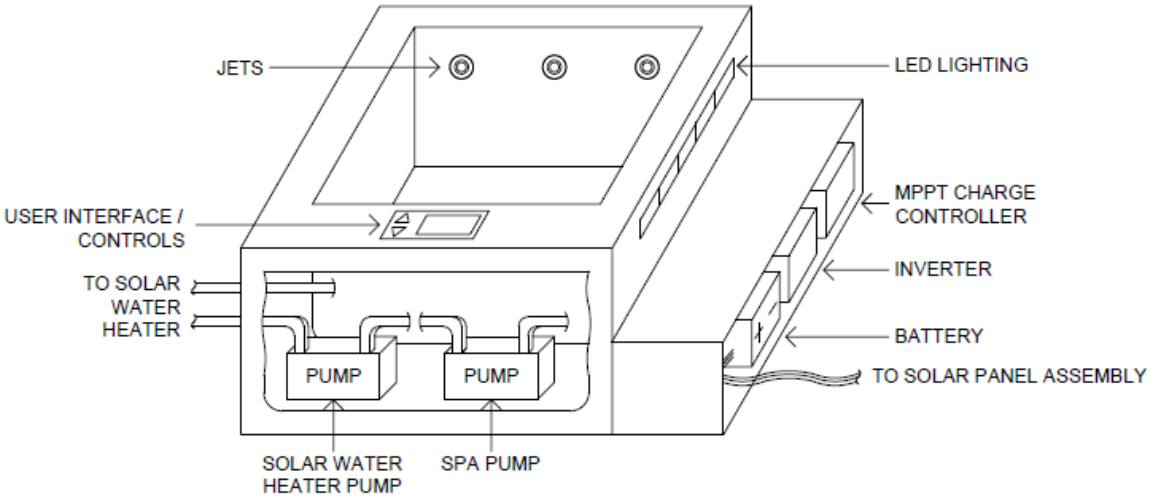


Figure 3.4: Hot Tub Model

Table 3.2: Hot Tub Model Descriptions

LABEL	FUNCTIONALITY
User Interface	• Displays temperature and menu options • Controls run modes and lighting functions
LED Lighting	• Exterior lighting and one interior light • Dimmable control through the user interface
Spa/AC Pump	• Manual control for 1, 10, and 15 minute operation • Automatic control for 40 minute operation • Automatic control for 24 hour operation
Heat/DC Pump	• Manual control for 30sec, and 5-15 minute operation • Automatic control for 40 minute operation • Automatic control for 24 hour operation
Battery	• Power source for pumps, lights, and controller • Connected to the solar panel assembly

SOLAR PANEL ASSEMBLY

The solar panel assembly is responsible for providing all of the electrical power to the system. It is comprised of three major components: a solar panel itself, a charge controller, and a battery. Note that the battery is apart of both the solar panel assembly and the hot tub structure because one powers it and the other houses it. It operates by having the solar panel produce electrical power from the sun. Once this power is harvested it is stored in the battery so that the hot tub can be used at all times of the day. The charge controller plays a vital role because it regulates the power flow in the system by ensuring that the battery is properly charged and any excess power is properly taken care of. A simple block diagram of the system is provided in Figure 3.5 while the model and associated part descriptions are shown in Figure 3.6 and Table 3.3 respectively.

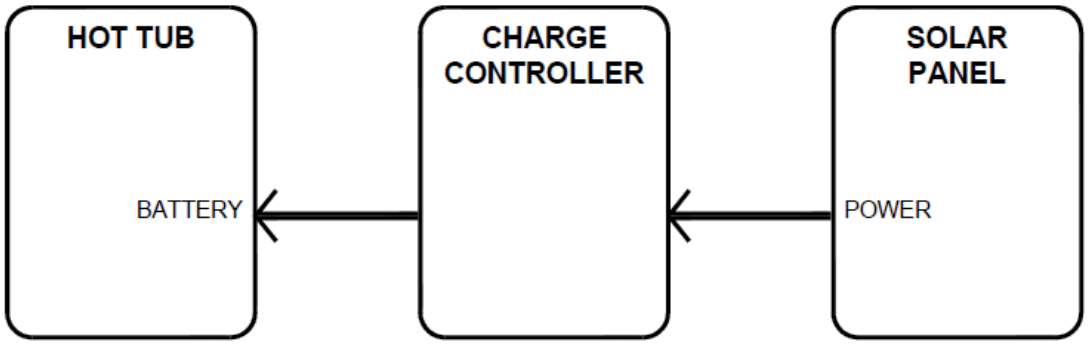


Figure 3.5: Solar Panel Assembly Block Diagram

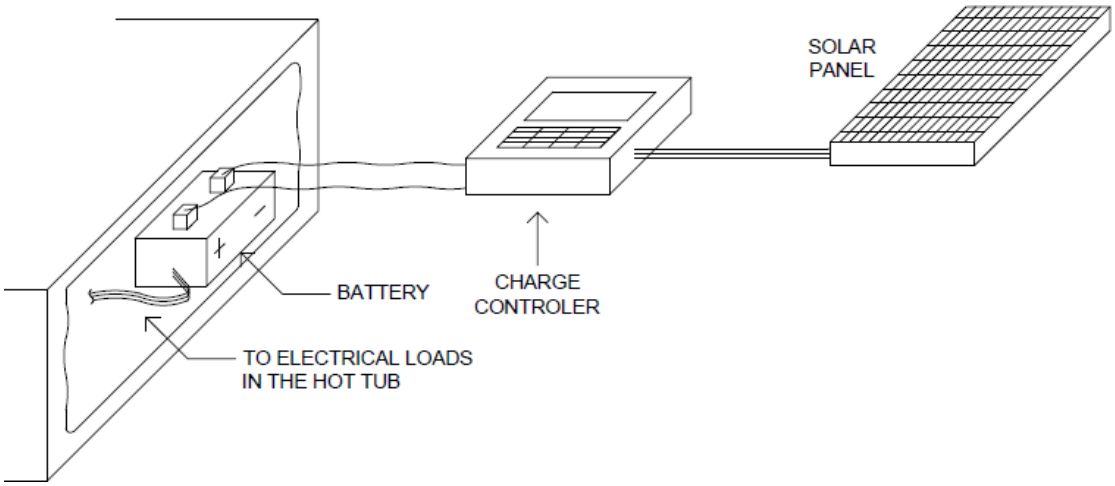


Figure 3.6: Solar Panel Assembly Model

Table 3.3: Solar Panel Model Descriptions

LABEL	FUNCTIONALITY
Solar Panel	• Provides power for the entire hot tub
Charge Controller	• Regulates the power supplied by the solar panel • Ensures proper charging of the battery
Battery	• Stores the power to be used by the hot tub

SOLAR WATER HEATER ASSEMBLY

The solar water heater assembly is responsible for providing the hot water to the system. It does this using a single 12 volt DC pump and 10 solar vacuum tubes. The pump carries the water out of the hot tub and into a heat exchanger built out of copper tubing and solar vacuum tubes. While in the heat exchanger the water is warmed up by the heat transfer from the copper pipes and is pumped back into the hot tub. By repeating this process throughout the day the water is continuously heated until an appropriate temperature is reached. Once this temperature is reached the pump can be turned off in order to keep the water from getting any hotter. The block diagram showing the system setup is provided in Figure 3.7 while a more detailed model is shown in Figure 3.8. Table 3.4 provides descriptions of the components.

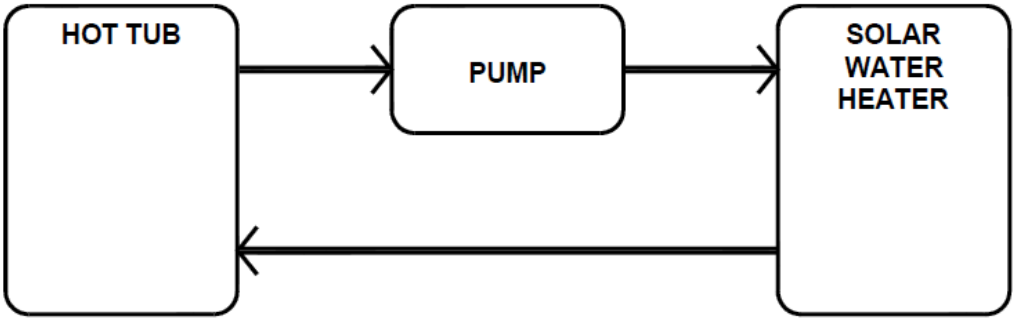


Figure 3.7: Solar Water Heater Assembly Block Diagram

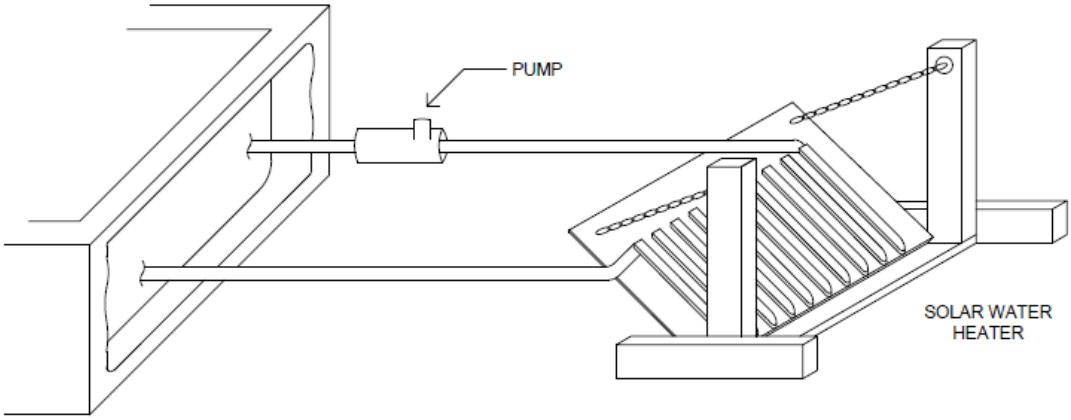


Figure 3.8: Solar Water Heater Assembly Model

Table 3.4: Solar Water Heater Model Descriptions

LABEL	FUNCTIONALITY
Solar Water Heater	• Provides the hot water for the hot tub
Pump	• Control the water flow throughout the system

USER INTERFACE

The user interface is a physical unit, which allows the user to view system diagnostics and adjust system settings. Upon power-up, the main-menu of control options as well as the current temperature reading will be displayed. From here, the user will be able to adjust the mode of operation, the jets' running-time, lighting control, and heat pump running time, as shown in Figure 3.10.

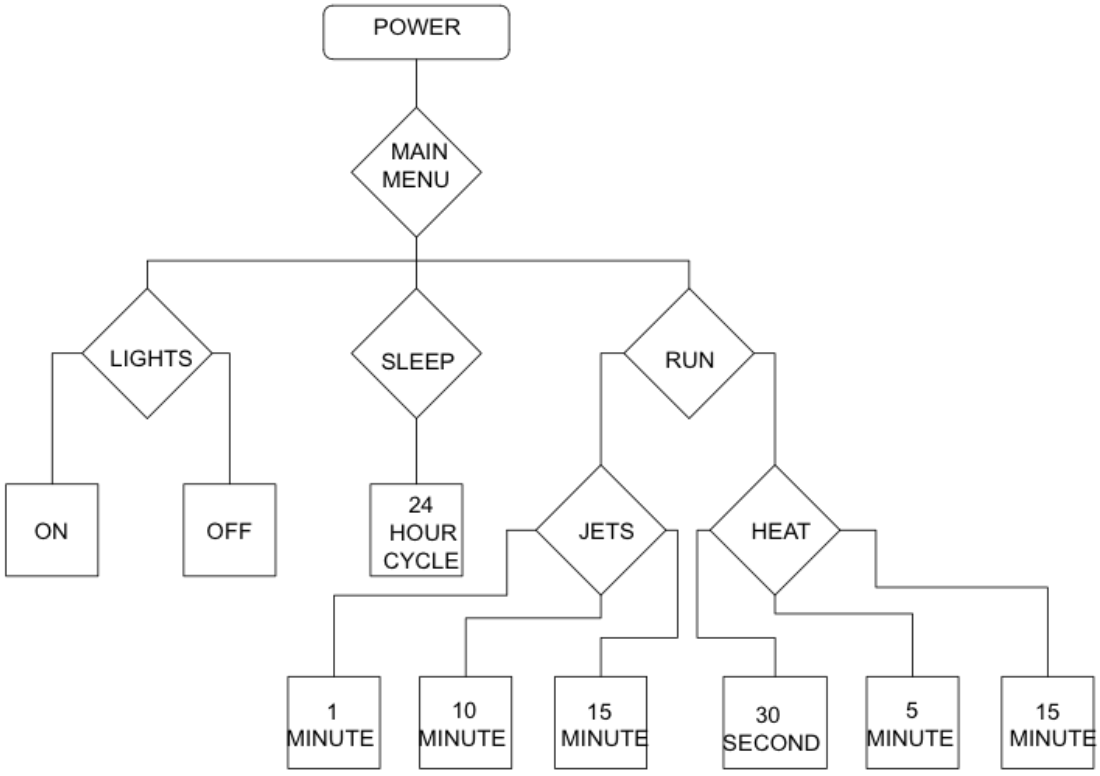


Figure 3.10: User-Interface High-Level Programming Flow-Diagram

A complete list of system settings and descriptions for each setting can be found in Table 3.7. The source code developed for the complete control and automation of the user-interface is documented in Appendix F.

Table 3.7: User Interface System Settings

SETTING NAME	SETTING DESCRIPTION
POWER	Energizes user-interface and initializes software instructions
MAIN MENU	Displays each of the main control settings
TEMPERATURE STATUS	Displays the current temperature of the water
RUN MODE SETTINGS	Displays two modes of operation: Auto-Run and Manual Control
AUTO-RUN	Displays “AUTO-RUNNING” when selected Energizes both heat (DC) and jet (AC) pumps for 40 minutes
MANUAL CONTROL	Displays two options for control: Jets and Heat
JETS	Displays 3 pre-set time settings: “1 minute”, “10 minutes”, and “15 minutes” Displays running time for selected pre-set Energizes the AC pump for selected time
HEAT	Displays 3 pre-set time settings: “30 seconds”, “1 minute”, and “15 minutes” Displays running time for the selected pre-set time Energizes the DC pump for the selected time
SLEEP	Displays “SLEEPING” when selected Energizes heat (DC) pump for 30 minutes ON and 30 minutes OFF Energizes jet (AC) pump for 1 hour ON and 2 hours OFF Cycles continuously 24 hours/day and 7 days/week until powered off
LIGHTING	Potentiometer-controlled dimmable intensity-range

The user interface (Figure 3.11) will consist of an LCD screen, a temperature display screen, 4 buttons (for control and making selections), two potentiometers, an embedded microcontroller (MSP 430), associated wiring, and support circuitry (Figures 3.12 and 3.13).

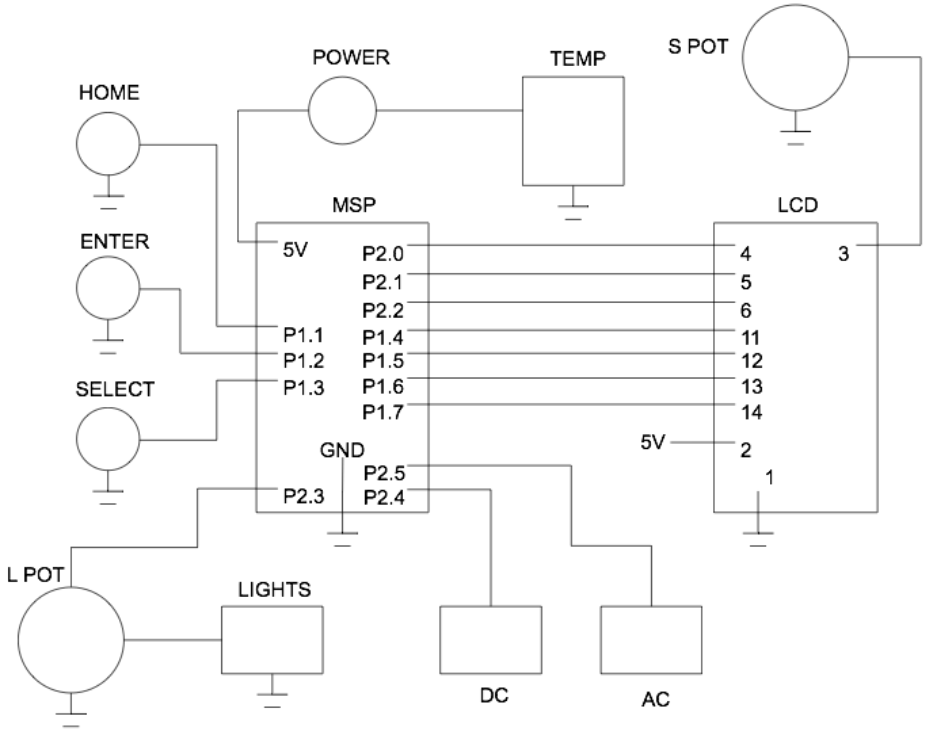


Figure 3.11: User-Interface Schematic

Table 3.8 describes each of the function blocks and input/output ports shown in Figure 3.11.

Table 3.8: User Interface Schematic Port Descriptions

BLOCK NAME	BLOCK DESCRIPTION	PORT NAME	PORT DESCRIPTION
POWER	Energizes the unit	5V	Switch between USB and 5V devices
SELECT BUTTON	Scroll through options	P1.3	Connects to MSP input
ENTER BUTTON	Command to select an item	P1.2	Connects to MSP input
HOME BUTTON	Return to Main-Menu	P1.1	Connects to MSP input
MSP	Microcontroller used for processing	P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7 P2.0 P2.1 P2.2 P2.3 P2.4 P2.5	GPIO Input → POWER GPIO Input → HOME GPIO Input → ENTER GPIO Input → SELECT GPIO Input → LCD 11 GPIO Input → LCD 12 GPIO Input → LCD 13 GPIO Input → LCD 14 GPIO Input → LCD 4 GPIO Input → LCD 5 GPIO Input → LCD 6 GPIO Output → LIGHTS GPIO Output → DC Pump GPIO Output → AC Pump
LCD	Display unit	1, 16 2, 15 3 4 5 6 11 12 13 14	LCD ground → MSP ground LCD power → 5v power switch LCD intensity control → S POT R/S control → MSP P2.0 R/W control → MSP P2.1 Enable control → MSP P2.2 Data bit 0 → MSP P1.4 Data bit 1 → MSP P1.5 Data bit 2 → MSP P1.6 Data bit 3 → MSP P1.7

Table 3.8 (Continued): User Interface Schematic Port Descriptions

BLOCK NAME	BLOCK DESCRIPTION	PORT NAME	PORT DESCRIPTION
TEMP	Display unit	5V	Connected to 5V switch
L POT	Lights Potentiometer	P2.3	12V power for lights
S POT	LCD screen potentiometer	3	Control arm of potentiometer

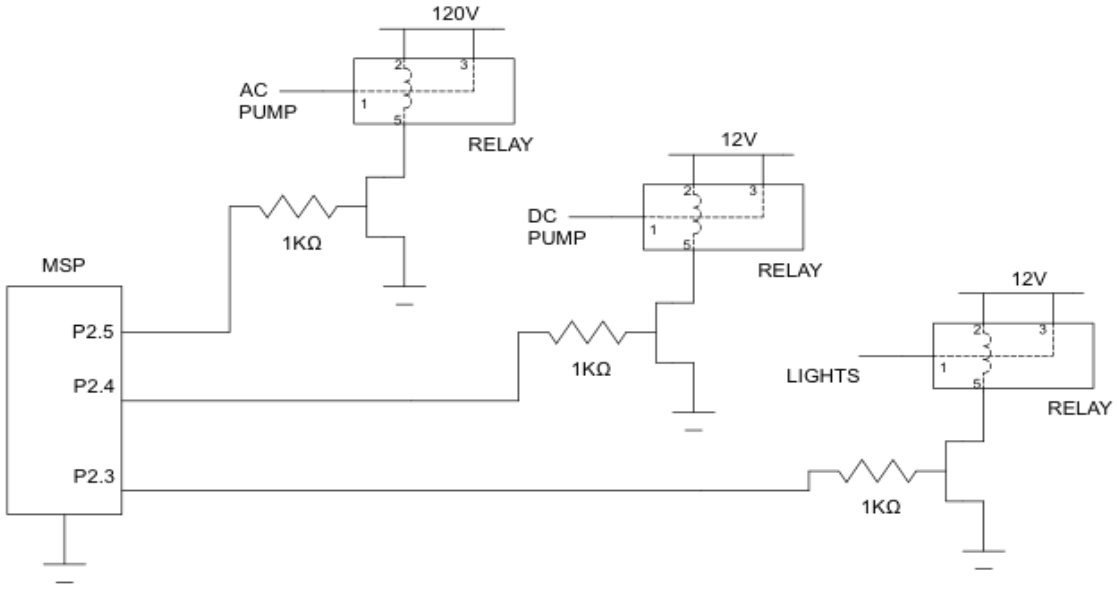


Figure 3.12: Relays Schematic

Figure 3.12 shows how digital output pins from the microcontroller are used to energize each relay, allowing the required amount of power to safely flow from each source to each load.

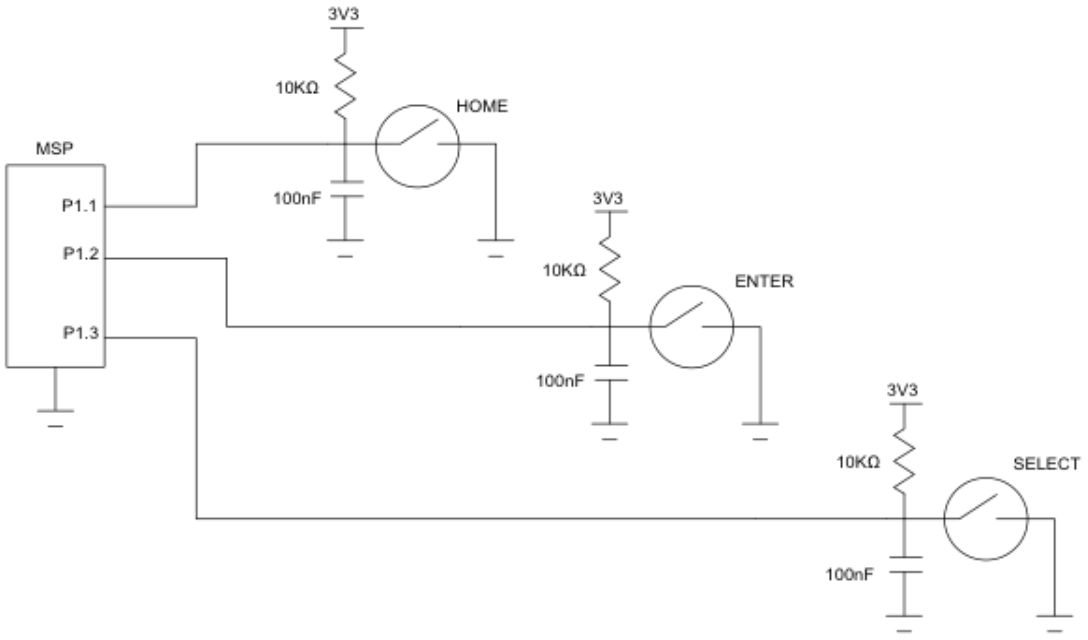


Figure 3.13: Switch-Debounce Circuitry Schematic

Figure 3.13 shows how low-pass filters were used to mitigate bouncing of the mechanical switches.

4. CONSTRUCTION

HOT TUB STRUCTURE

To begin construction on this project an existing two-person hot tub was purchased to work with. The first step consisted of stripping the old hot tub of all of its existing mechanical and electrical parts. This included removing the existing 3 horsepower motor along with all of its electrical control wiring, which would be replaced with the earlier discussed design. The cleared out electrical compartment is shown in Figure 4.1.

Figure 4.1: Hot Tub with Pump and Wiring Removed

After the hot tub had been cleared the design for the new pump and associated piping was installed. The construction of this involved sizing the existing pipes (2" piping) that were going to be utilized and making sure the appropriate connections were bought so that the new spa pump (1 ½" ports) could be properly installed.

Figure 4.2 shows the preliminary piping model.

Figure 4.2: AC Pump Plumbing (Preliminary Model)

After putting more thought into the piping it was decided to modify the preliminary design in order to add valves on either end of the spa pump. This would allow the user to isolate the spa pump in case any repairs needed to be made, without having to drain the entire tub. An additional valve was also added just above the input port of the spa pump so that the user can add water directly into the spa pump. This allows the user to ensure that the pipes leading into the pump are full of water and not air; therefore, avoiding any possibility of having air lock when starting the pump. The final design that was used is shown in Figure 4.

Figure 4.3: Pre-Pump-Installation Plumbing Connections

With the main connections to the new spa pump complete it was time to find piping that could be utilized for the hot water heater assembly. This involved removing a majority of the insulation on the bottom of the tub in order to get a good look at the rest of the existing plumbing. Figure 4.4 shows the bottom of the hot tub with most of the insulation removed. After determining the function of each pipe it was decided to repurpose two existing drain ports (Figure 4.5) in the bottom of the tub as the input and output of the hot water heater. These ports were chosen because they didn't effect the current functionality of the tub and because they provided a short, clean route for the plumbing of the hot water heater.

The next step in the construction of the tub was to install all of the hot water heater plumbing. The same general design that was used for the spa pump was used for the hot water heater pump. Two valves on either end of the pump were included so that the hot water heater itself could be isolated from the pump and the internal plumbing connecting it to the hot tub. Therefore, the user could remove the hot water heater as they pleased, allowing them to store it in a safer place when not in use.

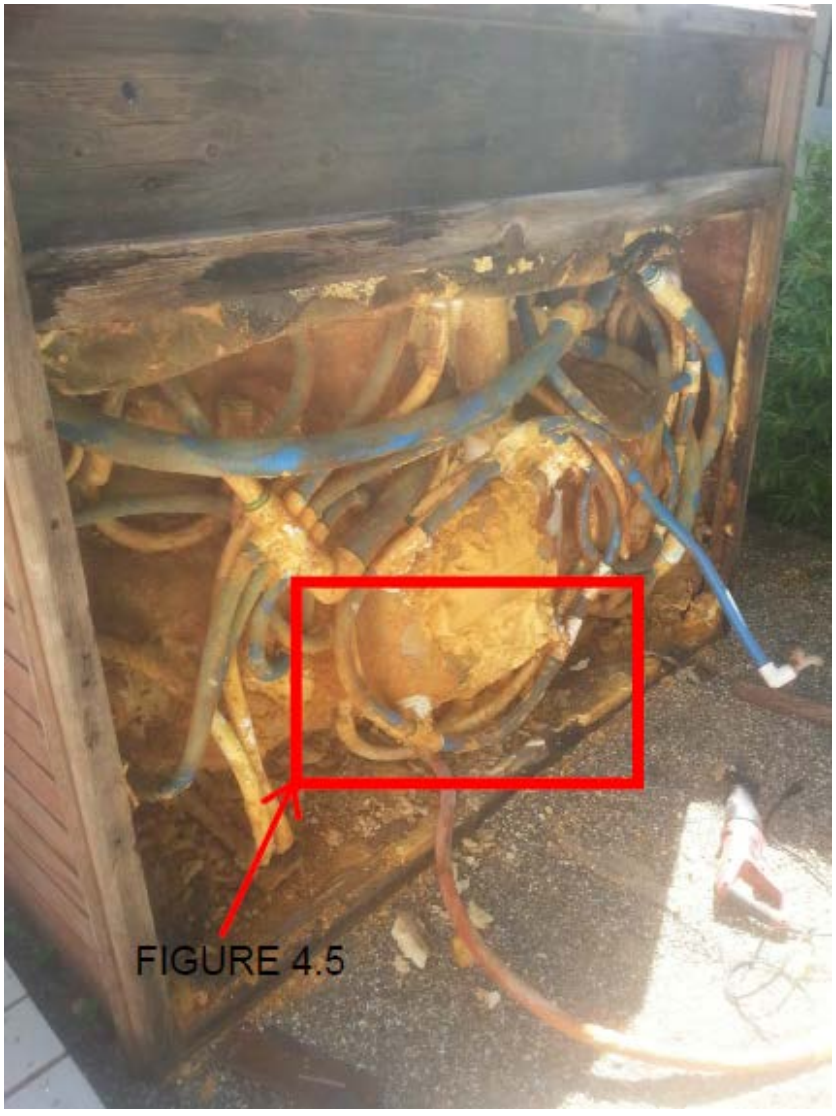


Figure 4.4: Existing Hot Tub Plumbing (Insulation Removed)

Figure 4.5: Existing Port chosen for Hot Water Heater Assembly

After designing the plumbing for the water heater, it was installed along with the associated pump and valves. Figure 4.6 shows the valves and the mounted pump for the finished plumbing.

Figure 4.6: Final Plumbing for Hot Water Heater Assembly

With the design and construction of the plumbing and the pumps complete, the next step was to test the existing plumbing that was going to be kept in order to ensure that there were no leaks. After dropping the hot tub into its normal position and filling it with insulation and water it was found that there were many leaks in the existing pipes. This lead to a long process of tracking the leaks, cutting out the faulty pipes, and replacing them with new PVC pipes. Pictures of the repairs made are provided in Figure 4.7. Once all of the leaks were repaired the base of the tub was filled with insulation and covered with two by fours and plywood to give it some

structural integrity. The side panels of the hot tub were also covered with plywood in order to give the finished product a clean look. The last step to completing all of the hardware on the tub itself consisted of mounting the spa pump and connecting it to the rest of the plumbing. The finished pictures of the hot tub are provided in Figures 4.8, 4.9, and 4.10.

Figure 4.7: Hot Tub Plumbing Leak Repairs

Figure 4.8: Finished Hot Tub Base

Figure 4.9: Pre and Post Finished Side Panel

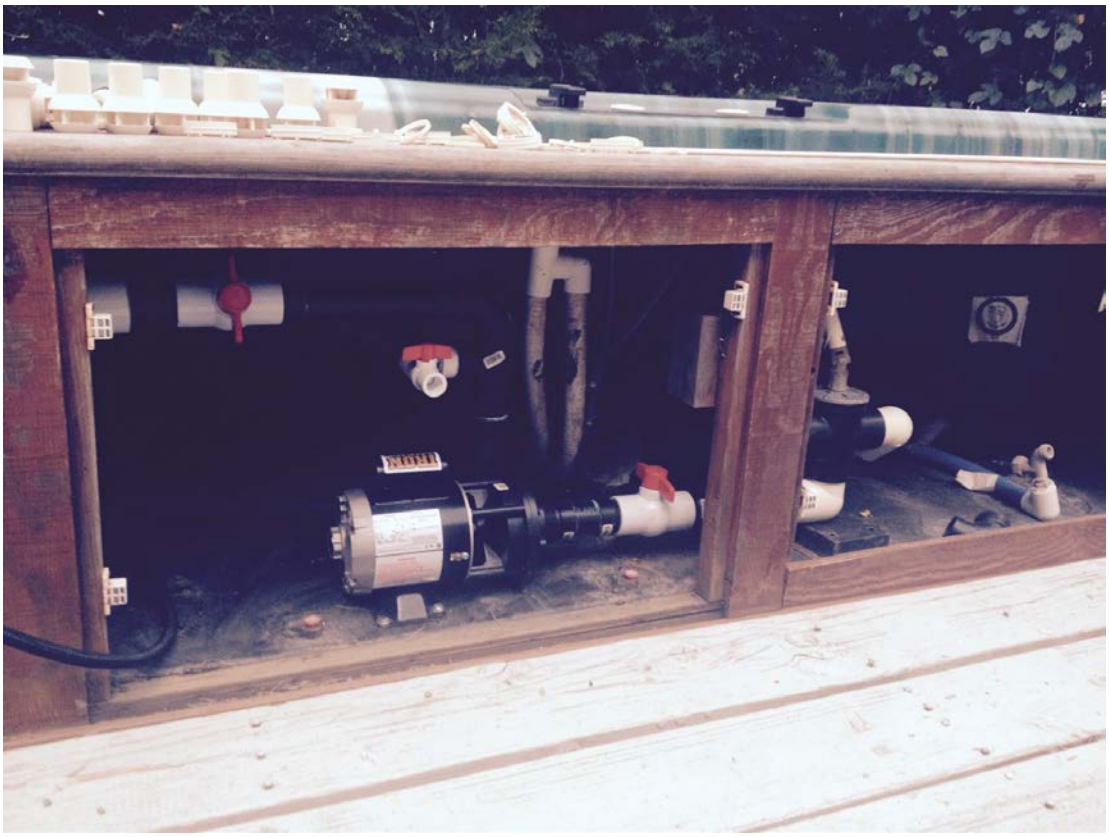


Figure 4.10: Finished Spa Pump and Plumbing with Flow Control and Bleed Valves

SOLAR WATER HEATER ASSEMBLY

It was decided to use solar vacuum tubes to heat the water in order to reduce the amount of electrical energy needed. The selected solar vacuum tubes collect sunlight and use reflective layers to trap the light within the tubes. This allows for a very efficient use of the sun's heat. Table 4.1 shows the measured water temperature in one of these tubes over a period of time.

Table 4.1: Solar Vacuum Tube Temperature Data

Time Elapsed (minutes):	0	15	30	45	60	75
Water Temp. (°F):	73	89	100	109	119	126

In order to use the solar vacuum tubes an apparatus needed to be built, which could hold them. In addition to that the apparatus needed to be adjustable so that the solar vacuum tubes could be repositioned throughout the day to remain perpendicular with the sun. The final product is shown in Figure 4.11. One thing to note is that a reflective styrofoam backing was used on the apparatus for two reasons: the reflective surface provided more sun for the solar tubes and the styrofoam provided additional padding between the wood backing and the glass tubes.

Figure 4.11: Water Heater Apparatus

With the apparatus built, the next step was to create the heat exchanger that would sit on the apparatus. It was decided to build the heat exchanger out of pressurized copper pipes that would sit inside of the solar vacuum tubes. Another set of copper pipes, for the water to flow through, would be soldered across the tops of the solar vacuum tubes in order to allow the heat from the pressurized tubes to be transferred into the flowing water. The construction began with soldering together the pressurized copper pipes that would sit inside of the solar vacuum tubes along with the two pipes that would sit on top of them. Figure 4.12 shows this construction while Figure 4.13 shows the final product.

One thing to note about the completed copper tubing is that they were pressurized using acetone. By doing this it allows for the copper tubing to be more efficient when transferring heat. While the pressurized pipes are sitting inside of the vacuum

tubes the acetone will begin to boil and focus the heat up towards the top of the pipe, closer to where the water will flow.

Figure 4.12: Heat Exchanger Construction

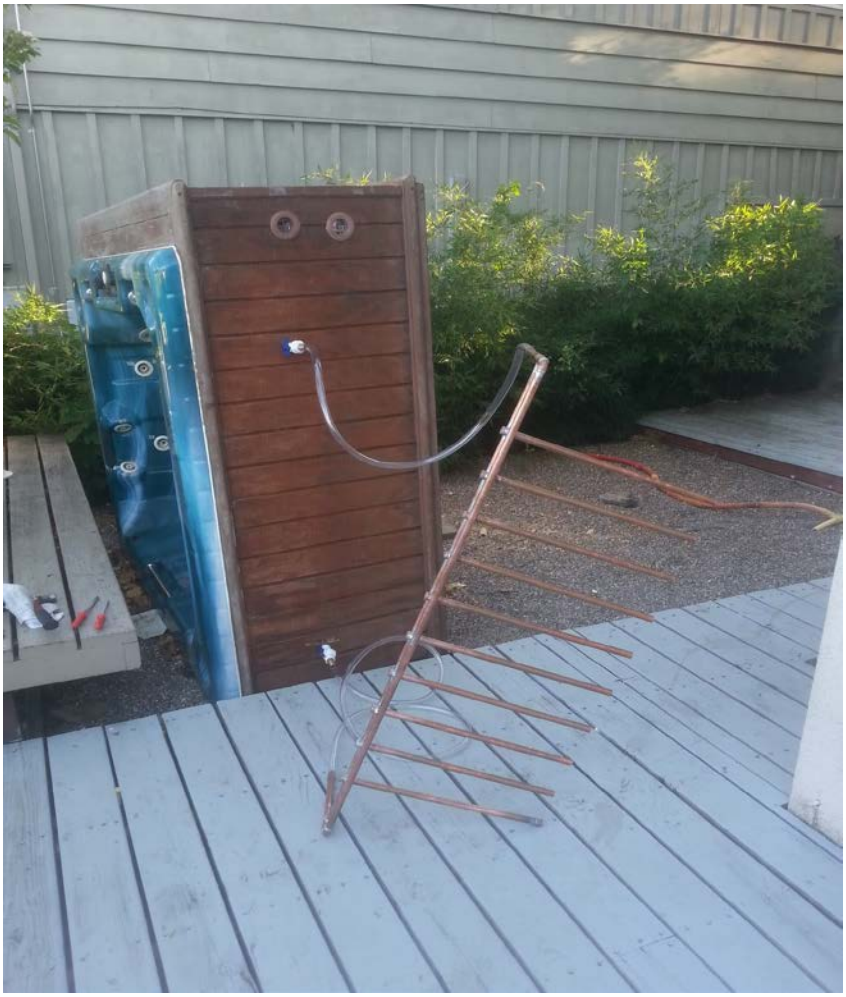


Figure 4.13: Completed Copper Tubing for Heat Exchanger

With the heat exchanger complete, the copper piping needed to be prepped, in order to be fit into the solar vacuum tubes. This was done by wrapping each individual pipe in steel wool (Figure 4.14), which will allow it to transfer heat more effectively. The last step to finish the heat exchanger was to mount the solar vacuum tubes to the bottom of the apparatus. The completed heat exchanger unit is shown in Figure 4.15.

Figure 4.14: Prepped Copper Tubing

Figure 4.15: Completed Heat Exchanger

SOLAR PANEL ASSEMBLY

The solar panel assembly is the last major component of the project. One thing that was needed was a storage box that could house all of the necessary equipment: the battery, the inverter, and the MPPT charge controller. Therefore, one was designed to fit in between the legs of the heat exchanger so that it would allow for a clean look when the whole system was put away.

Figure 4.16: Completed Housing for Electrical Components

The pictures in Figure 4.16 above show the completed box that will be used to house the battery, charge controller and inverter. It was built out of 2x4s and plywood. The design features 3" of ground clearance to protect the equipment from any water that may be on the ground, while also allowing for better heat dissipation. The box was tested to ensure that it could handle the 175-200 pound weight of the equipment that it will house.

With the housing for all of the electrical components complete, the last step was to create the appropriate wire connections between each component. This involved assessing the loads that each run was going to see and sizing the wire appropriately. Using the load calculations and the NEC allowed the correct wire size to be chosen.

After determining wire sizes, all of the wires were cut to the desired length, tinned, and soldered to their appropriate connectors. Conduit was ran from the spa pump and water heater pump to the storage box so that the wires could be safely ran between the controls and the pumps. Once all of the connecting wires were finished there was only the matter of grounding the system that needed to be handled. In order to make the system a truly stand-alone one an 8 foot copper rod would have had to be driven into the earth in order to provide ground for the system. For the sake of this project, a solid copper ground wire was run from the house panel to the ground connection on the spa pump.

USER INTERFACE

With all of the major components complete, the construction shifted to building the user interface. This section highlights the steps involved with building the hardware of the system, while the *User Interface* section under the Design heading goes through the software processes (Note that the code for the user interface can be found in Appendix F). To start, an electrical box and a piece of sheet metal (Figure 4.17) were used to house the user interface and all of the control circuitry needed. Once the layout of the screens and buttons were figured out the sheet metal was cut to allow all of the pieces to fit nicely (Figure 4.18).

Figure 4.17: Materials for Control Circuitry Housing

Figure 4.18: Completed Hardware for User Interface

With all of the hardware finished it was time to solder all of the power/control wires to their appropriate terminals. This included soldering together all of the schematics shown in Figures 3.11, 3.12, and 3.13. The pinout shown in Table 3.8 was used as a guide. Figure 4.19 shows a picture of some of the connections being made.

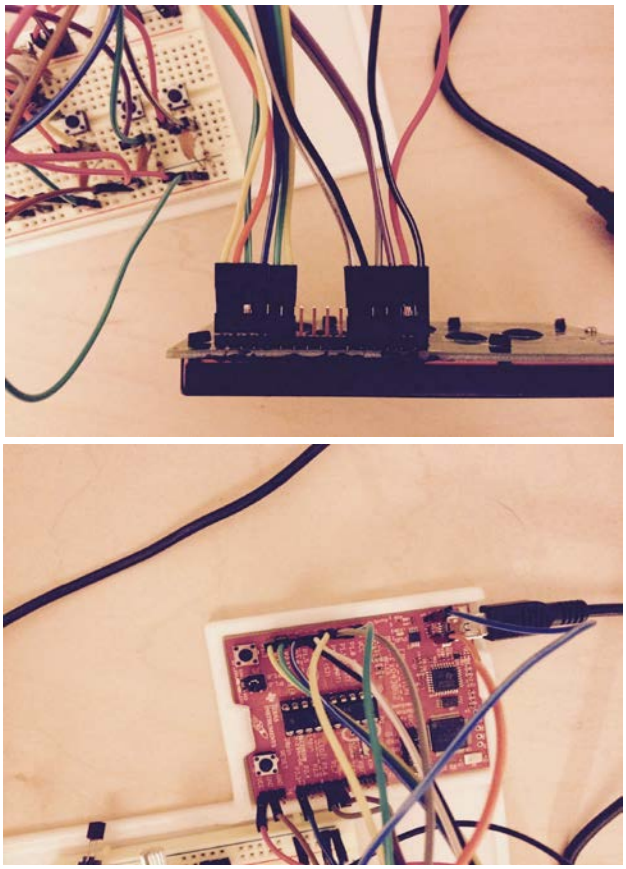


Figure 4.19: Control Wiring for User Interface

With the final touches of the hardware finished, the electrical box was sealed and installed on the wooden lip of the hot tub (Figure 4.20). Conduit was then ran from the underside of the control unit to the wooden box housing all of the electrical components to finish up all of the necessary control wiring, thus completing the construction of the hot tub. The only remaining construction from this point on was purely cosmetic.

5. DATA

Power Calculations:

In order to verify that the 175Ah 12V battery was large enough to power the whole system it was necessary to calculate the total load of all of the components. This included the power draw from the spa pump, the hot water pump, the lighting, and the control panel. Table 5.1 shows these components and their total power draw.

Table 5.1: Breakdown of Power Consumption

LOAD	OPERATING CONDITIONS	POWER CONSUMPTION
SPA PUMP	1.3A at 120V AC (13A at 12V DC)	(1.3A * 120V) = 156W
HOT WATER PUMP	1.25A at 12V DC	(1.25A * 12V) = 15W
LIGHTING	4A at 12V DC	(4A * 12V) = 48W
CONTROL PANEL	~5mA at 5V (Powered through USB)	(5mA * 5V) = 25mW (Negligible)
Total Wattage Consumed:		220W
Total Wattage Stored in Battery:		(175Ah * 12V) = 2100Wh

With the battery's ability to store 2100Wh and a system power demand of 220W it can be expected to get about 9.5 hours (2100Wh / 220W = 9.55 hours) of use before the battery is completely drained. In order to avoid a complete drain on the battery the system cycle schedule was designed for 8 hours of use per day. Since the system

should be available for use at all times of the day the pumps will be on a regular schedule of being on for 1 hour and then off for 2 hours. This allows the tub to consistently circulate the water throughout the day. With the hot tub holding approximately 400 gallons of water and the spa pump being able to push 45 GPM it is expected that all of the water can be circulated through the system 6.75 times an hour (45 GPM * 60 minutes = 2700 gallons) (2700 gallons / 400 gallons per tub = 6.75 tubs). This is good because stagnant water is never a good thing to have for a long period of time, so being able to circulate the water through the system 6.75 times an hour should avoid this.

System Testing:

Each of the system specifications listed in the Specifications Section was tested and verified in accordance with the following test methods and procedures.

BATTERY POWER:

The hot tub system shall operate off of a 12V rechargeable battery. The battery shall be able to charge to a minimum of 95% of the 12V capability (11.4V). Procedure outlined in Table 5.2. Battery voltage data along with voltage and amperage data from the spa pump were taken over a period of time to monitor the performance of the system. Table 5.3 shows the results while Figure 5.1 displays them in a plot.

Table 5.2: Test Conditions for System Power

OPERATION	TEST	RESULT	EXPECTED
OFF MODE	Open Circuit Voltage Test: Monitor voltage across terminals	12.85V	Shall be greater than 11.4V
SLEEP MODE	Partial load test: Monitor voltage across terminals	12.4V	Shall be greater than 11.4V
RUN MODE	Full load test: Monitor voltage across terminals	12.2V	Shall be greater than 11.4V

Table 5.3: Battery and Spa Pump Data

TIME	BATTERY VOLTAGE	SPA PUMP VOLTAGE	SPA PUMP CURENT	SPA PUMP WATTAGE
1:30 PM	12.85V DC	-	-	-
1:45 PM	12.65V DC	116.2V AC	1.267A	147.23 W
2:00 PM	12.48V DC	116.2V AC	1.267A	147.23 W
3:00 PM	12.34V DC	116.2V AC	1.76A	204.51 W
3:15 PM	12.29V DC	116.3V AC	1.66A	193.06 W
3:30 PM	12.25V DC	116.3V AC	1.67A	194.22 W
3:45 PM	12.22V DC	116.4V AC	1.69A	196.72 W
4:00 PM	12.20V DC	116.4V AC	1.68A	195.55 W
4:15 PM	12.16V DC	116.3V AC	1.62A	188.41 W
4:30 PM	12.12V DC	116.3V AC	1.62A	188.41 W

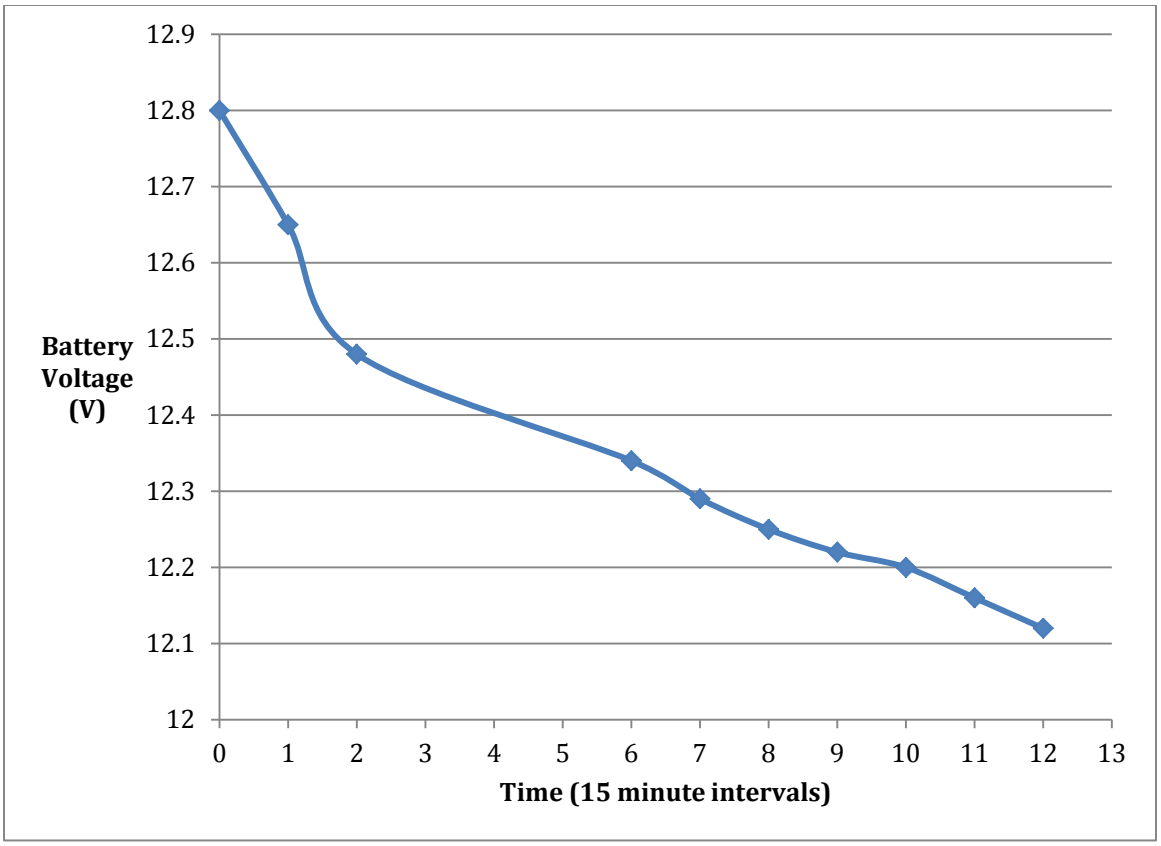


Figure 5.1: Battery Discharge Rate

Looking at Figure 5.1 it can be seen that after the initial startup of the pumps the battery discharge rate is rather linear. Following this trend it can be concluded that the system could run for 7.5 hours straight on a full battery charge (assuming a complete drain to 11.4V). These calculations are shown on the next page, following the trend of losing 0.4V every 15 minutes.

12.12V – 11.4V = 0.72V remaining

0.72V / 0.04V = 18 periods until completely drained

18 * 15 min = 270 min remaining (4.5 hours)

3 hours running + 4.5 hours remaining = 7.5 hours of total run time

Comparing these results with the expected 9.55 hours of total run time shows that the initial expectations were reasonable. One reason that they were incorrect was the fact that the spa pump was assumed to use 156W but actual measurements showed that it consumed around 200W. Re-calculating the expected run time with the actual wattage used by the spa pump produces an expected run time of 7.98 hours which is a lot closer to what was seen. The ½ hour of run time difference between expected and actual is most likely due to the inefficiencies of the solar water heater pump and the inverter.

SOLAR PANEL POWER:

The next calculations include the solar panel and recharging the battery. In order for the system to work properly the battery must be able to be charged (at the very least) the same amount that it had been drained that day. Since the battery can store 2100W a solar panel large enough to produce that in one day is necessary. Using a MatLab script (Appendix G) from a previous course (EE 420 – Sustainable Electric Energy Conversion) allowed for the calculations of average kilowatt hours produced per square meter on any given day along with the amount of daylight hours for that

day. Choosing a day in mid-December as a baseline for worst possible energy output yielded the result that there are approximately 9.61 hours of daylight that produce about 852.81 watt-hours per meter squared. Dividing this by the number of daylight hours gives the result that an average of 88.74 watts can be produced per square meter of solar panel throughout the day. These calculations were ran assuming a 20% efficiency of the solar panel used.

Using the dimensions of the solar panel found in Table 2.1 it can be concluded that the area of the solar panel is 2.588 meters squared. Multiplying this by the average watts per square meter provides an estimated 229.71 watts can be produced per hour by the 400 watt solar panel. Multiplying this again by the number of daylight hours produces a daily yielding of approximately 2208 watts per day. Therefore, even on a short winter day the 400W solar panel could completely recharge the battery of the system if needed. In order to ensure that the solar panel is run as efficiently as possible a max power point tracking charge controller was purchased to allow the solar panel to perform at a higher rate.

PUMP OPERATION:

Both jet and heat pumps are controlled by user selected power modes on the user interface. The pumps shall only operate in the SLEEP and RUN power modes. The circulation pump shall operate within 10% of the manufacturer's rated output in both modes. Procedure outlined in Table 5.4.

Table 5.4: Test Conditions for Control of Pumps

OPERATION	TEST	RESULT	EXPECTED
SLEEP MODE	No load test: Measure current	☑	Zero current
	DC pump test: Measure current Monitor running time Monitor cycling	1.32A ☑	1.2A 30 min ON / 30 min OFF Cycle continuously until powered off
	AC pump test: Measure current Monitor running time Monitor cycling	~1.7A ☑	1.3A 1 hour ON / 2 hours OFF Cycle continuously until powered off

Table 5.4 (Continued): Test Conditions for Control of Pumps

OPERATION	TEST	RESULT	EXPECTED
RUN MODE	<u>No load test:</u> Measure current	☑	Zero current
	<u>DC pump test:</u> Measure current Monitor running time Monitor cycling	1.32A ☑	1.2 A 30 min ON / 30 min OFF Cycle continuously until powered off
	<u>AC pump test:</u> Measure current Monitor running time Monitor cycling	~1.7A ☑	1.3A 1 hour ON / 2 hours OFF Cycle continuously until powered off
	<u>Pre-set time test:</u> Monitor 30 second Monitor 1 minute Monitor 5 minute Monitor 10 minute Monitor 15 minute	☑	30 second ON then OFF, no cycle 1 minute ON then OFF, no cycle 5 minute ON then OFF, no cycle 10 minute ON then OFF, no cycle 15 minute ON then OFF, no cycle

LIGHTING CONTROL OPERATION:

The hot tub features 10 LED light bulbs. All 10 LEDs shall illuminate upon user selection on the user interface. Electrical current being supplied to each LED shall be tested and shall not exceed the manufacturer's rated input current value. Procedure outlined in Table 5.5.

Table 5.5: Test Conditions for Lighting Control Operation

OPERATION	TEST	RESULT	EXPECTED
RUN MODE	LED test: Illuminate all LEDs by cycling through full range	☑	Dim to full intensity
	Current Test: Measure current to lighting throughout range	☑	Not to exceed 8A

USER INTERFACE OPERATION:

The user interface shall control every operation of the hot tub as well as display temperature, charge percentage, and each feature within the system settings of the user interface itself. Procedure outlined in Table 5.6.

Table 5.6: Test Conditions for the User Interface (Display Modes)

OPERATION	TEST	RESULT
DISPLAY TEMPERATURE	Observe: Temperature on temperature screen	☑
	Verify temperature with secondary measurement device (acceptable within 1°F)	☑
DISPLAY MAIN MENU	Observe: Main menu options and verify that selection of each option navigates to each sub setting options	☑
DISPLAY POWER MODES	Observe: Run, Sleep, Lighting	☑
DISPLAY FULL JET TIME-RANGE	Observe: Values (1, 10, 15 minutes)	☑
DISPLAY FULL LIGHTING-RANGE	Observe: Intensity change throughout potentiometer range	☑
CLEAR DISPLAYS	Observe: Clear screens when powered off	☑

6. ANALYSIS

PRELIMINARY DESIGN CONSIDERATIONS

There are multiple ways to go about designing a solar powered hot tub however they all must include three major subsystems. These subsystems include: the tub itself, a way to power it, and a way to heat the water. Within these three subsystems is where the variation in design will occur. The following paragraphs go through some of the more common design variations in detail while Table 6.1 sums up all of the ideas.

The first major design variation includes how to power the hot tub. It can either be completely solar powered or grid tied. The advantages of having the system completely solar powered include not having to plug it into an outlet and not having any added expenses on the monthly electric bill. These two advantages are weighted more heavily than others because they are some of the main reasons for why people don't buy hot tubs. The cons of using solar panels are the fact that they can take up a lot of space. Whereas a plug-in takes up practically no space a solar panel is about 3'x5' on average and needs to be in direct sunlight. This means that it must be elevated above the rest of the system; however, this can be used as an advantage. By putting the solar panel on top of a cabinet it will allow it to remain in direct sunlight while also providing storage space for the user.

The alternative for powering the hot tub includes tying the system to the grid. By doing this the user will never have to worry about running out of power; however, it

will cost them a lot more to install the system. By grid tying the hot tub the user will need to include an inverter to transform the DC voltage into AC voltage so that it can be put back onto the grid. The user will also have to install protection equipment in order to ensure that they are properly attached to the grid and that they are not back powering anything if the power were to go out. Lastly, there is the possibility that the user will have to pay a monthly electric bill if the system is grid tied.

The next major design variation includes the pumps used in the system. They can either be DC powered or AC powered. If they are DC powered then the system will be able to run directly off of the 12V battery source from the solar panel assembly. This is convenient because the battery is necessary in the system anyways. The only downfall to the DC pumps is the fact that they are more expensive to buy. However, since an inverter would be needed to power AC pumps it is safe to say that using DC pumps wouldn't cost significantly more than an AC setup. The only clear advantage to using AC pumps is the fact that they are cheaper; however, as stated, additional hardware would be necessary in order to make them run.

The last major design variation includes heating the water. It can either be done using a solar water heater or an electric water heater. The advantage of a solar water heater is the fact that it uses a lot less electrical power than an electric heater. The major disadvantage to the solar water heater is the fact that, like the solar panel, it is going to take up a lot of extra space. However, if this extra space can be utilized then it will reduce the negative effects of it.

Preliminary design decision includes going completely solar powered while using DC pumps and a solar water heater. By utilizing the space that the solar panels and solar water heater need the design will be completely self-sustainable while providing nice amenities to the user. We saw that the advantages of completely eliminating any monthly electric bill outweighed the size disadvantages. As for the pumps we decided to use the DC powered ones in order to maximize the energy achieved using the solar panels. This way there is no worry about running out of power. The design variations are summarized in Table 6.1 while the current design aspects are in bold.

Table 6.1: Design Options Summary

SUBSYSTEM	VARIATION	PROS	CONS
Powering the System	• **Completely Solar Powered**	• No plug necessary • No added monthly cost to the electric bill	• Need room for a solar panel • Overall system is larger
	• Grid Tied System	• No worry about running out of power	• Adds more parts to the system (inverter and protection circuits) • More expensive
Pumps	• **All DC Powered Pumps**	• More energy efficient	• More expensive
	• All AC Powered Pumps	• Cheaper	• Inverter needed to convert power
Heating the Water	• **Solar Water Heater**	• More energy efficient than an electric water	• Need room for a solar panel • Overall system is larger
	• Electric Water Heater	• Takes up less space	• Use a lot of power

FINAL DESIGN

After extensively weighing out the pros and cons addressed in our preliminary analysis, we decided to go with an off-grid DC powered system with solar heat tubes as opposed to using electric heating. These two choices afforded us the ability to run the system off of only one solar panel and one 12V battery, significantly reducing the overall physical size and complexity of the system.

Additionally, we ended up going with one DC pump and one AC pump. The DC pump was selected to circulate water between the tub and the heater because this is a low flow dependent operation requiring very little power (15W) to operate. Operation of the main circulation pump (jets pump) on the other hand, requires a considerable amount of power (⅛ HP) to operate, which demands approximately 13A of current from a 12V battery...taking the efficiency of the pump into consideration. The increased safety risk associated with the higher current is one major reason why we decided to go with an AC pump. The other reason we decided against a DC pump for main circulation (jets) is because there weren't any ⅛ HP DC pumps available for general purchase, it would have to be special ordered driving up the cost significantly.

FINAL PRODUCT

Unfortunately we ran out of time and budget before we could integrate the solar panel into the system. The hot tub runs solely off of the 12V battery with the solar panel only serving as a mean to recharge the battery, so we didn't lose out on proof of concept by not incorporating it. Even still, it would have been beneficial to monitor the charging of the battery in real-time to verify that initial power consumption calculations (shown in DATA section with MATLAB code displayed in Appendix G) were within acceptable accuracy. That being said, the maximum power-point tracker automatically monitors any solar panel connected to it and displays, records, and adjusts supply voltage as necessary. Therefore, if we had connected a solar panel to the MPPT and determined that the system draws more power from the battery than the solar panel can recharge during any one day…then we would have simply adjusted the duty cycle of each pump (duty cycle adjusted in software) thereby reducing the energy consumption to meet recharging capability.

7. CONCLUSION

Overall the project was successful in proving that a completely stand-alone solar powered hot tub is achievable. All of the individual systems worked as expected. The amount of energy provided by the 175Ah 12V battery was very close to what was expected and it had no problem powering the whole system for a continuous 7 hour stretch of time. The heat exchanger was very effective in heating up moderate amounts of water between 100-104°F (typical for a hot tub) and the user interface provided a clean, easy way for the user to control all aspects of the hot tub.

Although the individual components worked as expected they weren't as effective as they could have been once they were all combined into one system. The main problems seemed to be with the heating of the tub and the pressure of the jets. The size of the tub played a large role in both of these problems. With such a large amount of water in the hot tub the heat exchanger had to be run for a longer period than expected in order to get the water temperature up to an acceptable level. The large size of the tub also meant that there were a lot more jet ports to push water through which the smaller spa pump had some trouble with. Even though the spa pump was powerful enough to circulate the water it didn't produce as much pressure as expected. Additionally, all insulation was removed during system testing due to water leaks in the plumbing, which had a significant effect on the tub's ability to retain heat. A smaller tub would provide an easy fix to both of these problems. This analysis, along with other possible design changes, are visited in the next section titled Future Work.

Another large problem that was encountered was the poor condition of the existing plumbing of the hot tub. Initially the existing plumbing was kept in order to save money on having to replace it; however, this backfired because of the numerous leaks that were encountered in the existing tubes. This lead to a large amount of time being spent tracking down and repairing leaks.

In the end the project came out over budget. It was expected to have spent about $1150 total but the major components ended up costing around $1250. Including all of the tools and small items that were necessary to complete the job it is estimated that the complete project cost around $1500 to produce. While this was over budget it was still in a respectable range considering many modern hot tubs cost anywhere from $2000 - $5000 plus the additional monthly electric bill to run them.

In conclusion the project was a success and provided an excellent platform to build off of in the future. With a few design changes the efficiency of the hot tub could be maximized in order to produce a completely stand-alone solar powered hot tub that has the same functionality as commercially produced products however uses a fraction of the energy.

8. FUTURE WORK

In order to create a better version of the solar powered hot tub there are a few design changes that could be implemented. To begin, a smaller hot tub would be ideal. The one used in this project was a 3 person tub that held somewhere around 400 gallons of water. Due to the method of heating that was used it was very difficult to get the entire hot tub up to the desired temperature. A 1-2 person tub that holds about 100-150 gallons would work the best with the current setup. Another recommendation would be to install all new plumbing. Leaking pipes were consistently a problem throughout this entire project and installing all new plumbing would have saved a lot of time. Insulation is another key feature that must be addressed during the initial stages of design, it is recommended that a spray on foam be used to completely seal the space between the outside of the tub and the structure used to house the tub; this being done only after the plumbing has been leak and pressure checked.

Electrically there were a couple of changes that could be made. One problem seen in the current setup was that there was a lack of pressure in the jets. This was due to the amount of jets and the smaller sized spa pump (1/8 HP). Since the spa pump was the largest possible with the battery used the only other option was to cut out some of the existing jets. However, it was seen that the smaller 15W water heater pump provided a good amount of pressure to the 2 ports it was connected to. One major change that could be made is to remove the larger spa pump (1/8 HP) completely and install an additional 15W water pump for each pair of jets. With the

200W consumed by the spa pump (1/8 HP) an extra 12 15W DC water pumps could have been used, providing much more pressure for the jets without using any extra energy. This would also provide much more selectivity for the user when controlling which jets to turn on and off.

Making this design change would initially save $200 because there would be no need to buy an inverter to run the spa pump. This money could be used to buy the additional 15W water pumps leading to the conclusion that 6-7 extra water pumps would be ideal in order to break even on the cost of everything while reducing the overall energy used by the system. With the system using less overall energy the idea of a smaller battery could be entertained, providing even more savings. With more 15W water pumps the controller could also be modified to provide even more control over the entire system. By adding additional relays for the extra water pumps the user could potentially control each jet individually in order to get the system to perform exactly how they want it.

Another change that could be made was in the design of the heat exchanger. The current deign consisted of only running water along the tops of the solar vacuum tubes; however, an alternate option would be to run flex copper piping into and out of the vacuum tubes so that the water would actually travel the length of each tube. The setback with this design is that smaller diameter piping would be needed to run through the tubes, meaning that less water is flowing through the system and might therefore lead to it taking longer to heat the entire tub of water. However, it would

be worth building and testing in order to see which method performs better for a smaller tub application.

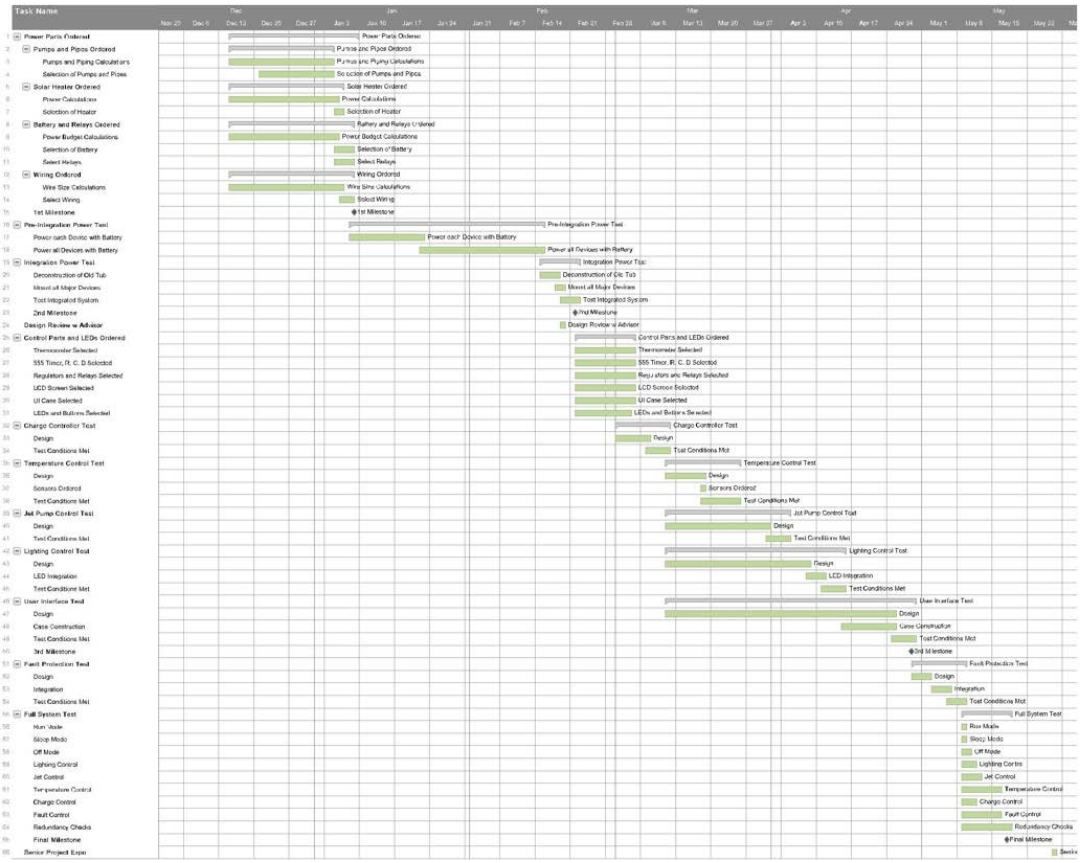
With a smaller tub and a more efficient heat exchanger the controller could be improved in order to allow more control over the water temperature of the tub. This could be done by using the readout of the temperature sensor and using that to control the relay of the single 15W water heater pump. By allowing the user to select a target temperature the system can turn on the water heater pump and leave it on until the temperature sensor reads the desired output.

Overall, the tub size, pump setup, and heat exchanger design are the main components that could use some adjustments in order to provide a solar powered hot tub that could perform at a higher level than the current one. All of these changes would allow for further advancements in the user interface to allow for more control over the system as a whole.

APPENDIX A: DIVISION OF LABOR

TASK DESCRIPTION	ANTHONY	JONATHAN
POWER SYSTEM DESIGN:		
SOLAR PANEL SIZING	X	X
THERMAL HEATER SIZING	X	X
PUMP SIZING	X	X
ELECTRICAL CONDUCTOR SIZING	X	
POWER BUDGET	X	X
PIPING DESIGN	X	
WIRE HARNESS DESIGN	X	X
CAD	X	X
AUTOMATION AND EMBEDDED SYSTEM DESIGN:		
USER INTERFACE SOFTWARE DESIGN		X
USER INTERFACE HARDWARE DESIGN		X
LIGHTING CONTROLLER DESIGN	X	X
TEMPERATURE CONTROLLER DESIGN	X	X
SYSTEM INTEGRATION:		
SOLAR PANEL INSTALLATION	X	X
THERMAL HEATER INSTALLATION	X	X
LIGHTING INSTALLATION	X	
PUMPING UNITS INSTALLATION	X	X
WIRE HARNESS INSTALLATION	X	X
BATTERY INSTALLATION	X	X
USER INTERFACE INSTALLATION		X

APPENDIX B: SCHEDULE



PRELIMINARY MILESTONE BREAKDOWN:

1st Milestone:	January 9, 2016	All power system parts on order.
2nd Milestone:	February 22, 2016	Test completion of integrated power-system components.
3rd Milestone:	April 29, 2016	Test completion of all user-interface controls.
Final Milestone:	May 20, 2016	Test completion of full-system run.
Project Expo:	May 27, 2016	Live Demonstration

FINAL MILESTONE ANALYSIS:

The first milestone slid back all the way to February 20th due to difficulties finding pumps and electronics that met our system specifications. Originally we wanted to go with a DC pump for both circulation and heating, but DC pumps under ½ HP and above 1/16 HP do not exist unless specially ordered, which costs more than we could budget.

The second milestone slid back to May 5th due to significant issues with the hot tub leaking due to age and poor condition of the used tub. This was the biggest issue we encountered during the project because the system could not be tested until the tub could hold water and be pressurized.

The third milestone slid back to May 15th due to the prior milestones sliding back.

The final milestone slid back to May 27th, which caused us to fail to meet out project expo milestone.

APPENDIX C: COST ANALYSIS

Table C.1 displays a breakdown of the expected cost of the project and a description for how each cost will be paid for. A complete itemized breakdown of every component's cost can be found in the Bill of Materials Section.

Table C.1: Project Cost Analysis

ITEM	COST	PAID FOR BY	
System Components	$1050	Cal Poly EE Department	$400
		Jon Peterson	$325
		Anthony Zepeda	$325
Tools	$100	Jon Peterson	$50
		Anthony Zepeda	$50
TOTAL: $1150			

APPENDIX D: BILL OF MATERIALS

BOM	By: Jonathan Peterson & Anthony Zepeda				
Part Name	Part Number	Description	Quantity	Unit Cost	Cost
Solar Panel	N/A	400 watt solar panel	1	$0.00	$0.00
Battery	LFP12175D	12V battery (175Ah)	1	$250.00	$250.00
Inverter	PWRI100012S	1000W Pure Sine Wave Inverter	1	$200.00	$200.00
MPPT Charge Controller	Tracer4210A	Max 100V PV input 12/24V ouput	1	$170.00	$170.00
Solar Vacuum Tubes	N/A	(10) Solar Collecting Glass Tubes	1	$99.00	$99.00
Spa Pump	3410030-1E	120VAC 1/15 hp Spa Pump	1	$148.00	$148.00
Circulation Pump	TS5 15PV	Solar DC Circulation Pump	1	$30.00	$30.00
Insulation	N/A	Kraft Faced Insulation Batts	2	$48.58	$97.16
Microcontroller	MSP430	MSP Based microcontroller	1	$2.84	$2.84
LCD Screen	ADM102K-NSW-FBS	16x4 LED Screen	1	$10.53	$10.53
Buttons	N/A	SPST Push Button Switch	3	$3.99	$11.97
Temperature Sensor	N/A	Waterproof temp. sensor w/ screen	1	$5.95	$5.95
Relay	HY41PN12DC	12V industrial power relay	1	$4.57	$4.57
LED Lighting	N/A	12V Dimmable Stip LED's	TBD	$0.00	$0.00
Copper Tubing	N/A	10' Length of 1/2" Copper Tubing	3	$12.99	$38.97
Copper Caps	N/A	1/2" Copper Caps	10	$0.62	$6.20
Copper Fittings	N/A	1/2" Copper Fittings	10	$1.38	$13.80
Wood - 2x4	N/A	2x4 for Wood Framing	10	$2.53	$25.30
Plywood	N/A	Plywood Backing	3	$10.65	$31.95
Screws	N/A	1 1/2" & 3" Screws	1	$6.47	$6.47
PVC Pipe	N/A	10' Length of 1 1/2" & 2" PVC Pipe	2	$7.49	$14.98
PVC Pipe Fittings	N/A	1 1/2" PVC Pipe Fittings	10	$1.35	$13.50
PVC Ball Valve	N/A	1 1/2" PVC Ball Valve	2	$10.73	$21.46
PVC Ball Valve	N/A	1/2" PVC Ball Valve	3	$2.52	$7.56
Wire	N/A	1' Length of 10 Gage THHN Stranded Wire	30	$0.59	$17.70
Wire	N/A	100' Length of 12 Gage THHN Stranded Wire	1	$12.99	$12.99
Wire	N/A	1' Length of 6 Gage THHN Stranded Wire	20	$0.99	$19.80
				Total:	$1,260.70

APPENDIX E: ANALYSIS OF SENIOR PROJECT DESIGN

Project Title: Solar Powered Hot Tub
Student's Name: Jon Peterson **Signature:**_____.
Student's Name: Anthony Zepeda **Signature:**_____.
Advisor's Name: Dr. Ali Shaban **Initials:**_____.
Date:_____.

Summary of Functional Requirements

The solar powered hot tub operates off of a rechargeable 12V battery which is charged during the day by a 400W solar panel, allowing it to operate independently from the grid. The hot tub features a user interface which provides the user with full control of the hot tub's temperature, lighting, jet operation time, and power mode.

Economic Impacts

Natural Capital:
One of the key objectives of this project was to reduce the reliance on electrical power from commercial utilities, which create pollution and harm to the environment during the generation and transmission of power.

This project meets that objective in two ways: the first is by establishing a system that operates independently from the electrical grid, and the second is by using a natural resource (the sun) to power the system.

Human Capital:
Hot tubs are used by many for therapeutical purposes. Heated water relaxes tense and aching muscles, which helps relieve physical and mental stresses.

By developing a less expensive to operate hot tub, more consumers may take advantage of these stress relieving benefits and may become more effective at work, allowing them to be more productive and therefore more marketable..

Commercial Manufacturing Financials

This project is not intended to be sold for profit as a marketable product; however, for the right entrepreneur, this could be a profitable venture.

The cost to build a single unit is high, due to all the expensive subsystem components: solar panels, solar heaters, DC pumps, the tub, etc. Therefore in order to earn a profit, the hot tub would need to be manufactured in a facility that is equipped to construct several units in the same time that one unit normally could be constructed. This would require an automated manufacturing system and a large warehouse to store completed units, which would be a very large production costs.

Additionally, it would be imperative for the entrepreneur to establish business partners in the solar equipment industry and pump manufacturing industry so to buy units in bulk at a reduced price.

With the correct business model, it may be possible to reduce manufacturing costs from $1300 per unit down to $500 per unit. The expected retail price for a solar powered hot tub is $5500, which suggests that there is money to be made here. However, it would require a lot of capital to get production started.

Environmental Impacts

As mentioned earlier, this project was designed with environmental concerns as the driving factors. By operating off-grid these hot tubs are helping reduce the pollution created by electrical generation and distribution providers, as well as utilizing natural resources.

However, the system does operate off of rechargeable batteries which typically have a lifetime of 4-6 years, which means that a battery would have to be disposed of at those time intervals, adding to the build up of hazardous chemical waste at landfills.

Manufacturability

As mentioned earlier, there is a high cost associated with the manufacturing of hot tubs and it is imperative to partner with other companies in order to integrate their products at a reduced cost.

Sustainability

Maintenance of the hot tub is relatively simple, other than the battery change every 4-6 years, the system only requires regular water treatment.

The system could be designed to be more sustainable if the battery was replaced by some form of energy harvesting device, which does not contain hazardous chemicals and require to be replaced periodically.

Additionally, if salt water were used in place of fresh water then there would be no need to routinely treat the water.

Ethical Concerns

Due to the size of a hot tub it may be ethically questionable whether or not it is right to purchase a product which creates pollution due to manufacturing processes and adds harmful chemical waste (batteries) to the environment, when an extra long soak in a bathtub may suffice.

Regular bathtubs can be purchased that feature soothing jets comparable to those found in a hot tub, so there is a case to be made here. However, when considering

the extra water usage required to routinely be filling up and draining a bathtub, ethical concerns of water usage abuse arise.

Health and Safety Concerns

Hot tubs present a drowning and ingestion of overly chlorinated water hazard when used by unsupervised children. Additionally, special personal protective equipment and care must be taken when changing out batteries.

Social Concerns

Hot tubs are a luxury item used for recreation and relaxation by those who can afford the high ticket price. As such, the hot tub creates inequality among different communities with citizens earning drastically different incomes than others.

The average hot tub consumer is a middle class home-owner. Often times, hot tubs are purchased as a status symbol, just as purchasing green products has become because it cost more to buy these items. Thus, by developing an expensive status symbol, this product has added to the widening inequality gap.

Personal Development

This project allowed us to use much of the knowledge we had gained during our academic career and then some! Aside from using computer engineering skills gained in CPE courses to design the software of the user-interface, microelectronic course skills to design debounce and transistor-switch circuits in support of the user-interface hardware, and power systems course skills to size higher wattage electronics and design a safe system free from overcurrent conditions...we also got to learn a long list of new skills as well. Some of the new skills we gained include: plumbing design and repair, welding, carpentry, cost analysis to meet budget constraints, metalsmithing, and thermodynamics associated with heat exchange.

APPENDIX F: USER-INTERFACE SOURCE CODE

```
/*************************************************************************
  Senior Project: Solar Powered Hot Tub
  Designed by: Jonathan Peterson and Anthony Zepeda
  May 20, 2016

  This program was designed to both display and control
  the operation of the hot tub.
*************************************************************************/

#include <msp430.h>
#include <stdio.h>

/*****************************CONSTANTS*****************************/

int sec1 = 0;
int min1 = 0;
int hr1 = 0;
int sec2 = 0;
int min2 = 0;
int hr2 = 0;

int counter = 0;
int counter_DC_sleep = 0;
int counter_AC_sleep = 0;
int counter_jets = 0;
int counter_heat = 0;

int Auto_counter_30 = 1800;

int DC_sleep_counter_30 = 1800;
int AC_sleep_counter_hour = 3600;
int AC_sleep_counter_2 = 7200;


int AC_sleep_loop = 0;

int Jet_1_counter = 60;
int Jet_10_counter = 600;
int Jet_15_counter = 900;

int Heat_30_counter = 30;
int Heat_5_counter = 300;
int Heat_15_counter = 900;


int set_auto = 0;
int set_sleep = 0;

int set_jet_1 = 0;
int set_jet_10 = 0;
int set_jet_15 = 0;

int set_heat_30 = 0;
int set_heat_5 = 0;
int set_heat_15 = 0;

/*************************************************************************/
```

```
/*****************************MAIN PROGRAM*************************/
int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;        // Stop watchdog timer

  // set ports
  P1DIR |=  0xF1;  //set upper four bits of P1 to ouputs will act as data bus (nibble mode)
            //and set LED on P1.0 as output
  P1DIR &=  ~0x0E; //set as inputs (BTNS)
  P2DIR |=  0x3F;  //set first 3 bits of P2 as output to drive RS, R/W, E for p2.0 p2.1 p2.2 respectively,
            //and 2.3 as lights, 2.4 as DC pump, 2.5 as AC pump
  P2OUT = 0x00;  //make sure outputs are cleared

  //set clk to be used to 16MHz
  if (CALBC1_16MHZ==0XFF)
     {
        while(1);
     }
  DCOCTL = 0;
  BCSCTL1 = CALBC1_16MHZ;
  DCOCTL = CALDCO_16MHZ;

  CCTL0 = CCIE;  //Parameters for Interrupts
  P1OUT &= 0x00;
  CCR0 = 32768;
  BCSCTL3 = XT2S_0 + LFXT1S_0;
  TACTL = TASSEL_1 + MC_1;

  //Screensaver();

  for(;;)
  {
     MAIN_MENU();  //call systems main menu infinitely
  }
}

/*************************************************************************/



/*****************************INTERRUPT-BASED TIMER*****************/

#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{

/* Test Blinking LED
            if ((P1OUT & 0x01) == 0x01)
  {
            P1OUT &= ~0x01;
  }
  else
  {
            P1OUT |= 0x01;
  }
*/

            static char OutString[9];
  Clear_Disp();

//Convert integer variables to string variables inside a character array,
//which will then be sent to LCD Driver to display time.
  sprintf(OutString,"%d%d:%d%d:%d%d", hr2, hr1, min2, min1, sec2, sec1);


  counter++;
  counter_DC_sleep++;
  counter_AC_sleep++;
  counter_jets++;
  counter_heat++;
```

```
//Establish seconds, minutes, hours variables for operation of a digital clock
//Using two variables for each two-digit second, minute, and hour display.

  if(sec1 < 9) //Count seconds
  {
     sec1++;
  }
  else
  {
     sec1 = 0;
     if(sec2 < 5)
     {
        sec2++;
     }
     else
     {
        sec2 = 0;
        if( min1 < 9) //Count minutes
        {
              min1++;
        }
        else
        {
           min1 = 0;
           if (min2 < 5)
           {
              min2++;
           }
           else
           {
              min2 = 0;
              if(hr1 < 9) //Count hours
              {
                           hr1++;
              }
              if ((hr1 == 3) & (hr2 == 2)) //Reset clock at 23:59
              {
                 sec1 = 0;
                 min1 = 0;
                 hr1 = 0;
                 sec2 = 0;
                 min2 = 0;
                 hr2 = 0;
              }
              else
              {
                 hr1 = 0;
                 if (hr2 < 2)
                 {
                    hr2++;
                 }
                 else
                 {
                    sec1 = 0; //Additional handling for overrun hour
                    min1 = 0;
                    hr1 = 0;
                    sec2 = 0;
                    min2 = 0;
                    hr2 = 0;
                 }
              }
           }
        }
     }
  }


  if (set_auto == 1)
  {
```

```
            Init_Big();  //initializing func to reset screen

    char auto_message[] = "    AUTO-RUNNING    ";

    Clear_Disp(); //clear out disp
    Set_Curs_loc( 0x40 );
    Write_word( auto_message, sizeof(auto_message) );


    if (counter > Auto_counter_30)
    {
            P1OUT &= ~0x01;  //P2OUT &= ~0x10;
            //P2OUT &= ~0x20;
            set_auto = 0;
            counter = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                    __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
    }
    else
    {
            P1OUT |= 0x01;  //P2OUT |= 0x10;
            //P2OUT |= 0x20;
    }

                    if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                    {
            P1OUT &= ~0x01;  //P2OUT &= ~0x10;
            //P2OUT &= ~0x20;
            set_auto = 0;
            counter = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

            __delay_cycles(8000000); // 0.5s
                    __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                    }
}
if (set_sleep == 1)
{

    char sleep_message[] = "     SLEEPING     ";

    Clear_Disp(); //clear out disp
    Set_Curs_loc( 0x40 );
    Write_word( sleep_message, sizeof(sleep_message) );

    if (counter_DC_sleep > DC_sleep_counter_30)
    {
              if ((P2OUT & 0x10) == 0x10)
              {
                      P1OUT &= ~0x01;  //P2OUT &= ~0x10;
              }
       else
       {
            P1OUT |= 0x01;  //P2OUT |= 0x10;
       }
              counter_DC_sleep = 0;
    }

    if (AC_sleep_loop == 0)
```

```
    {
            if (counter_AC_sleep > AC_sleep_counter_hour)
            {
                    P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                    AC_sleep_loop = 1;
            }
            else
            {
                    P1OUT |= 0x01;  //P2OUT |= 0x20;
            }
    }
    if (counter_AC_sleep > AC_sleep_counter_2 && AC_sleep_loop == 1)
    {
       counter_AC_sleep = 0;
       AC_sleep_loop = 0;
    }

                    if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                    {
            P1OUT &= ~0x01;  //P2OUT &= ~0x10;
            //P2OUT &= ~0x20;
            set_sleep = 0;
            counter_DC_sleep = 0;
            counter_AC_sleep = 0;
            AC_sleep_loop = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

            __delay_cycles(8000000); // 0.5s
                    __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                    }

}

if (set_jet_1 == 1 )
{
            Init_Big();  //initializing func to reset screen

    char timer_message[] = "    Running Time:     ";

    Clear_Disp(); //clear out disp
    Set_Curs_loc( 0x40 );
    Write_word( timer_message, sizeof(timer_message) );

            Set_Curs_loc( 0x1A );
            Write_word(OutString, 9);


            if (counter_jets > Jet_1_counter)
            {
                    P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                    set_jet_1 = 0;
                    counter_jets = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                    __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();

            }
            else
            {
                    P1OUT |= 0x01;  //P2OUT |= 0x20;
```

```
        }

                if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_jet_1 = 0;
                counter_jets = 0;
        sec1 = 0;
        min1 = 0;
        hr1 = 0;
        sec2 = 0;
        min2 = 0;
        hr2 = 0;

                __delay_cycles(8000000); // 0.5s
                __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                }
}

if (set_jet_10 == 1 )
{
        Init_Big();  //initializing func to reset screen

   char timer_message[] = "    Running Time:     ";

   Clear_Disp(); //clear out disp
   Set_Curs_loc( 0x40 );
   Write_word( timer_message, sizeof(timer_message) );

        Set_Curs_loc( 0x1A );
        Write_word(OutString, 9);


        if (counter_jets > Jet_10_counter)
        {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_jet_10 = 0;
                counter_jets = 0;
        sec1 = 0;
        min1 = 0;
        hr1 = 0;
        sec2 = 0;
        min2 = 0;
        hr2 = 0;

                __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
        }
        else
        {
                P1OUT |= 0x01;  //P2OUT |= 0x20;
        }

                if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_jet_10 = 0;
                counter_jets = 0;
        sec1 = 0;
        min1 = 0;
        hr1 = 0;
        sec2 = 0;
        min2 = 0;
        hr2 = 0;

                __delay_cycles(8000000); // 0.5s
                __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                }

}
```

```
if (set_jet_15 == 1 )
{
        Init_Big();  //initializing func to reset screen

   char timer_message[] = "    Running Time:     ";

   Clear_Disp(); //clear out disp
   Set_Curs_loc( 0x40 );
   Write_word( timer_message, sizeof(timer_message) );

        Set_Curs_loc( 0x1A );
        Write_word(OutString, 9);


        if (counter_jets > Jet_15_counter)
        {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_jet_15 = 0;
                counter_jets = 0;
        sec1 = 0;
        min1 = 0;
        hr1 = 0;
        sec2 = 0;
        min2 = 0;
        hr2 = 0;

                __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
        }
        else
        {
                P1OUT |= 0x01;  //P2OUT |= 0x20;
        }

                if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_jet_15 = 0;
                counter_jets = 0;
        sec1 = 0;
        min1 = 0;
        hr1 = 0;
        sec2 = 0;
        min2 = 0;
        hr2 = 0;

                __delay_cycles(8000000); // 0.5s
                __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                }

}

if (set_heat_30 == 1 )
{
        Init_Big();  //initializing func to reset screen

   char timer_message[] = "    Running Time:     ";

   Clear_Disp(); //clear out disp
   Set_Curs_loc( 0x40 );
   Write_word( timer_message, sizeof(timer_message) );

        Set_Curs_loc( 0x1A );
        Write_word(OutString, 9);


        if (counter_heat > Heat_30_counter)
        {
                P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                set_heat_30 = 0;
                counter_heat = 0;
```

```
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
            }
            else
            {
                        P1OUT |= 0x01;  //P2OUT |= 0x20;
            }

                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
                        P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                        set_heat_30 = 0;
                        counter_heat = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __delay_cycles(8000000); // 0.5s
                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                        }

}

if (set_heat_5 == 1 )
{
            Init_Big();  //initializing func to reset screen

   char timer_message[] = "    Running Time:     ";

   Clear_Disp(); //clear out disp
   Set_Curs_loc( 0x40 );
   Write_word( timer_message, sizeof(timer_message) );

            Set_Curs_loc( 0x1A );
            Write_word(OutString, 9);


            if (counter_heat > Heat_5_counter)
            {
                        P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                        set_heat_5 = 0;
                        counter_heat = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
            }
            else
            {
                        P1OUT |= 0x01;  //P2OUT |= 0x20;
            }

                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
                        P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                        set_heat_5 = 0;
                        counter_heat = 0;
```

```
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __delay_cycles(8000000); // 0.5s
                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                        }

  }

  if (set_heat_15 == 1 )
  {
            Init_Big();  //initializing func to reset screen

    char timer_message[] = "    Running Time:     ";

    Clear_Disp(); //clear out disp
    Set_Curs_loc( 0x40 );
    Write_word( timer_message, sizeof(timer_message) );

            Set_Curs_loc( 0x1A );
            Write_word(OutString, 9);


            if (counter_heat > Heat_15_counter)
            {
                        P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                        set_heat_15 = 0;
                        counter_heat = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
            }
            else
            {
                        P1OUT |= 0x01;  //P2OUT |= 0x20;
            }

                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
                        P1OUT &= ~0x01;  //P2OUT &= ~0x20;
                        set_heat_15 = 0;
                        counter_heat = 0;
            sec1 = 0;
            min1 = 0;
            hr1 = 0;
            sec2 = 0;
            min2 = 0;
            hr2 = 0;

                        __delay_cycles(8000000); // 0.5s
                        __bic_SR_register_on_exit(LPM0_bits + GIE);   //__disable_interrupt();
                        }

  }
  return;
}

/*********************************************************************/
```

```
/*************************PRINT SCREENSAVER*************************/
int Screensaver()
{
  Init_Big();  //initializing func to reset screen

  char screensaver[] = "     A & J  Spas     ";
  Clear_Disp(); //clear out disp
  Set_Curs_loc( 0x40 );
  Write_word( screensaver, sizeof(screensaver) );
  __delay_cycles(80000000); // 5sec wait

  return 0;
}

/*********************************************************************/

/************************MAIN MENU SUBROUTINE************************/
int MAIN_MENU()
{
        __delay_cycles(8000000); // 0.5s

  Init_Small();

  int loop_flag2 = 0;

  char line1[] = "**** MAIN MENU ****";
  char line2[] = "RUN";
  char line3[] = "SLEEP";
  char line4[] = "LIGHTS";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );
  Set_Curs_loc( 0x55 );
  Write_word( line4, sizeof(line4) );

  int op_marker = 0;
  int Op_Sel = 0;

  while (loop_flag2 != 2)
  {
                if ((P1IN & 0x08) != 0x08 ) //Selection Button
                {
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 3

                          Op_Sel = 1;          //option two selected
                          op_marker = 1;       //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 1
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x7E ); //write arrow to line 3
```

```
                                Op_Sel = 2;          //option two selected
                                op_marker = 2;       //remember that arrow is on line 3, will loop back to line 1 next
                                while((P1IN & 0x08) != 0x08 );
                              }
                              else if (op_marker == 2)
                              {
                                Set_Curs_loc( 0x54 );
                                Write_Letter( 0x10 ); //clear arrow from line 3
                                Set_Curs_loc( 0x40 );
                                Write_Letter( 0x7E ); //write arrow back to line 1
                                        Set_Curs_loc( 0x14 );
                                Write_Letter( 0x10 ); //make sure cleared arrow from line 2

                                Op_Sel = 0;          //option two selected
                                op_marker = 0;       //remember that arrow is on last line and should loop to top next
                                while((P1IN & 0x08) != 0x08 );
                              }

                    }


                    if((P1IN & 0x04) != 0x04 )    //Enter button
                    {
                              loop_flag2 = 2;

                              if (Op_Sel == 0)          //selects option one
                              {
        RUN();
                              }
                              else if (Op_Sel == 1)          //selects option two
                              {
        SLEEP();
                              }
                              else
                              {
        LIGHTS();
                              }
                    }
  }
  return 0;
}

/*****************************************************************************/
```

```
/****************************RUN SUBROUTINE*************************/
int RUN()
{
        __delay_cycles(8000000); // 0.5s

  int Op_Sel = 0;

  int loop_flag3 = 0;

  Init_Small();
  char line1[] = "*** RUN OPTIONS ***";
  char line2[] = "AUTO-RUN";
  char line3[] = "MANUAL CONTROL";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );

  int op_marker = 0;


  while (loop_flag3 != 3)
  {
                if ((P1IN & 0x08) != 0x08 ) //Selection Button
                {
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2

                          Op_Sel = 1;          //option two selected
                          op_marker = 1;       //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
                        {
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x7E ); //write arrow to line 1

                          Op_Sel = 0;          //option two selected
                          op_marker = 0;       //remember that arrow is on line 2, will loop back to line 1 next
                          while((P1IN & 0x08) != 0x08 );
                        }
                }


                if((P1IN & 0x04) != 0x04 )     //Enter button
                {
                        loop_flag3 = 3;
                        if (Op_Sel == 0)          //selects option one
                        {
        AUTO_RUN();
                        }
                        else
                        {
        MANUAL_CONTROL();
                        }
                }
```

```
                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
        return;  //MAIN_MENU();
                        }
  }
  return 0;
}

/*********************************************************************/




/**********************AUTO-RUN SUBROUTINE**************************/
int AUTO_RUN()
{
            __delay_cycles(8000000); // 0.5s

            set_sleep = 0;
            set_auto = 1;
            set_jet_1 = 0;
            set_jet_10 = 0;
            set_jet_15 = 0;
            set_heat_30 = 0;
            set_heat_5 = 0;
            set_heat_15 = 0;
  counter = 0;


  __bis_SR_register(LPM0_bits + GIE);

  return 0;
}



int MANUAL_CONTROL()
{
            __delay_cycles(8000000); // 0.5s

  int Op_Sel = 0;

  int loop_flag4 = 0;

  Init_Small();
  char line1[] = "** MANUAL CONTROL **";
  char line2[] = "JETS";
  char line3[] = "HEAT";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );

  int op_marker = 0;


  while (loop_flag4 != 4)
  {
                        if ((P1IN & 0x08) != 0x08 ) //Selection Button
                        {
```

```
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2

                          Op_Sel = 1;          //option two selected
                          op_marker = 1;       //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
                        {
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x7E ); //write arrow to line 1

                          Op_Sel = 0;          //option two selected
                          op_marker = 0;       //remember that arrow is on line 2, will loop back to line 1 next
                          while((P1IN & 0x08) != 0x08 );
                        }
                }


                if((P1IN & 0x04) != 0x04 )     //Enter button
                {
                        loop_flag4 = 4;
                        if (Op_Sel == 0)          //selects option one
                        {
        JETS();
                        }
                        else
                        {
        HEAT();
                        }
                }


                if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                {
      return;  // MAIN_MENU();
                }
  }
   return 0;
}

/*****************************************************************************/
```

```
/**************************JETS SUBROUTINE**************************/

int JETS()
{
        __delay_cycles(8000000); // 0.5s

  int Op_Sel = 0;

  int loop_flag5 = 0;

  counter = 0;
  counter_DC_sleep = 0;
  counter_AC_sleep = 0;
  counter_jets = 0;
  counter_heat = 0;

  Init_Small();
  char line1[] = "******* JETS *******";
  char line2[] = " 1 Minute";
  char line3[] = "10 Minutes";
  char line4[] = "15 Minutes";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );
  Set_Curs_loc( 0x55 );
  Write_word( line4, sizeof(line4) );

  int op_marker = 0;


  while (loop_flag5 != 5)
  {
                if ((P1IN & 0x08) != 0x08 ) //Selection Button
                {
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 3

                          Op_Sel = 1;          //option two selected
                          op_marker = 1;       //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 1
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x7E ); //write arrow to line 3

                          Op_Sel = 2;          //option two selected
                          op_marker = 2;       //remember that arrow is on line 3, will loop back to line 1 next
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 2)
                        {
                          Set_Curs_loc( 0x54 );
```

```
                                        Write_Letter( 0x10 ); //clear arrow from line 3
                                        Set_Curs_loc( 0x40 );
                                        Write_Letter( 0x7E ); //write arrow back to line 1
                                                Set_Curs_loc( 0x14 );
                                        Write_Letter( 0x10 ); //make sure cleared arrow from line 2

                                        Op_Sel = 0;           //option two selected
                                        op_marker = 0;        //remember that arrow is on last line and should loop to top next
                                        while((P1IN & 0x08) != 0x08 );
                                      }
                        }


                        if((P1IN & 0x04) != 0x04 )     //Enter button
                        {
                                        loop_flag5 = 5;
                                        set_sleep = 0;
                                        set_auto = 0;

                                        if (Op_Sel == 0)          //selects option one
                                        {
        set_jet_1 = 1;
                                        }
                                        else if (Op_Sel == 1)          //selects option one
                                        {
        set_jet_10 = 1;
                                        }
                                        else
                                        {
        set_jet_15 = 1;
                                        }


                          __bis_SR_register(LPM0_bits + GIE);

       //();
                        }


                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
      return; // MAIN_MENU();
                        }
  }
  return 0;
}

/*********************************************************************/
```

```
/*************************HEAT SUBROUTINE*************************/

int HEAT()
{
        __delay_cycles(8000000); // 0.5s

  int Op_Sel = 0;

  int loop_flag6 = 0;

  counter = 0;
  counter_DC_sleep = 0;
  counter_AC_sleep = 0;
  counter_jets = 0;
  counter_heat = 0;

  Init_Small();
  char line1[] = "******* HEAT *******";
  char line2[] = "30 Seconds";
  char line3[] = " 5 Minutes";
  char line4[] = "15 Minutes";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );
  Set_Curs_loc( 0x55 );
  Write_word( line4, sizeof(line4) );

  int op_marker = 0;


  while (loop_flag6 != 6)
  {
                if ((P1IN & 0x08) != 0x08 ) //Selection Button
                {
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 3

                          Op_Sel = 1;          //option two selected
                          op_marker = 1;       //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //make sure cleared arrow from line 1
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x54 );
                          Write_Letter( 0x7E ); //write arrow to line 3

                          Op_Sel = 2;          //option two selected
                          op_marker = 2;       //remember that arrow is on line 3, will loop back to line 1 next
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 2)
                        {
                          Set_Curs_loc( 0x54 );
```

```
                                        Write_Letter( 0x10 ); //clear arrow from line 3
                                        Set_Curs_loc( 0x40 );
                                        Write_Letter( 0x7E ); //write arrow back to line 1
                                                Set_Curs_loc( 0x14 );
                                        Write_Letter( 0x10 ); //make sure cleared arrow from line 2

                                        Op_Sel = 0;           //option two selected
                                        op_marker = 0;        //remember that arrow is on last line and should loop to top next
                                        while((P1IN & 0x08) != 0x08 );
                                      }
                        }


                        if((P1IN & 0x04) != 0x04 )     //Enter button
                        {
                                        loop_flag6 = 6;
                                        set_sleep = 0;
                                        set_auto = 0;

                                        if (Op_Sel == 0)          //selects option one
                                        {
        set_heat_30 = 1;
                                        }
                                        else if (Op_Sel == 1)          //selects option one
                                        {
        set_heat_5 = 1;
                                        }
                                        else
                                        {
        set_heat_15 = 1;
                                        }


                          __bis_SR_register(LPM0_bits + GIE);

      //();

                        }


                        if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                        {
      return;  //MAIN_MENU();
                        }
  }
  return 0;
}

/*********************************************************************/
```

```
/*************************SLEEP SUBROUTINE*************************/

int SLEEP()
{
        __delay_cycles(8000000); // 0.5s

        set_sleep = 1;
        set_auto = 0;
        set_jet_1 = 0;
        set_jet_10 = 0;
        set_jet_15 = 0;
        set_heat_30 = 0;
        set_heat_5 = 0;
        set_heat_15 = 0;
  counter_AC_sleep = 0;
  counter_DC_sleep = 0;


  __bis_SR_register(LPM0_bits + GIE);

  //();
  return 0;
}



int LIGHTS()
{
        __delay_cycles(8000000); // 0.5s


  Init_Small();

  int loop_flag7 = 0;

  char line1[] = "****** LIGHTS ******";
  char line2[] = "ON";
  char line3[] = "OFF";


  //print out menu
  Clear_Disp(); //clear out disp
  Write_word( line1, sizeof(line1) );
  Set_Curs_loc( 0x40 );
  Write_Letter( 0x7E );
  Write_word( line2, sizeof(line2) );
  Set_Curs_loc( 0x15 );
  Write_word( line3, sizeof(line3) );

  int Op_Sel = 0;
  int op_marker = 0;


  while (loop_flag7 != 7)
  {
                if ((P1IN & 0x08) != 0x08 ) //Selection Button
                {
                        if (op_marker == 0)
                        {
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x10 ); //clear arrow from line 1
                          Set_Curs_loc( 0x14 );
                          Write_Letter( 0x7E ); //write arrow to line 2

                          Op_Sel = 1;           //option two selected
                          op_marker = 1;        //remember that arrow is on line 2
                          while((P1IN & 0x08) != 0x08 );
                        }
                        else if (op_marker == 1)
```

```
                        {
                                Set_Curs_loc( 0x14 );
                          Write_Letter( 0x10 ); //clear arrow from line 2
                          Set_Curs_loc( 0x40 );
                          Write_Letter( 0x7E ); //write arrow to line 1

                          Op_Sel = 0;           //option two selected
                          op_marker = 0;        //remember that arrow is on line 2, will loop back to line 1 next
                          while((P1IN & 0x08) != 0x08 );
                        }
                }


                if((P1IN & 0x04) != 0x04 )     //Enter button
                {
                        loop_flag7 = 7;
                        if(Op_Sel == 0)          //selects option one
                        {
                                P1OUT |= 0x01;          //turn lights on P2 2.3 as control x08...or 1.0 x01 for testing
                        }
                        else
                        {
                                P1OUT &= ~0x01;         //turn lights off
                        }
                }


                if ((P1IN & 0x02) != 0x02 ) // toggle arrow to top, clear bottom arrow
                {
        return;  //MAIN_MENU();
                }
  }
  return 0;
}

/*****************************************************************************/
```

```
/***************WRITE WORD TO LCD SCREEN SUBROUTINE*****************/
int Write_word(char *word, int l_word )
{
        int i = 0; //counter
        for (i = 0; i < l_word - 1; i++ )
        {
                char up_let = 0b11110000 & word[i];
                char lo_let = 0b00001111 & word[i];
                lo_let = lo_let << 4; //bit shift 4 to upper 4

                //write a letter
                P1OUT &= 0x0F; // clear Data bus
                P2OUT |= 0x01; //E = 0, R/W = 0, RS = 1
                P2OUT |= 0x05; //E = 1, R/W = 0, RS = 1
                P1OUT |= up_let; // put data out (function set - upper nible)
                __delay_cycles(16); // 1us -tw delay
                P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
                P1OUT &= 0x0F; // clear Data bus

                P2OUT |= 0x01; //E = 0, R/W = 0, RS = 1
                P2OUT |= 0x05; //E = 1, R/W = 0, RS = 1
                P1OUT |= lo_let; // put data out (function set - lower nible)
                __delay_cycles(16); // 1us -tw delay
                P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
                P1OUT &= 0x0F; // clear Data bus
                __delay_cycles(960); // 60us
        }
        return 0;
}

/*********************************************************************/


/***************SET CURSOR ON LCD SCREEN SUBROUTINE*****************/

int Set_Curs_loc(char loc)
{
        char up_loc = 0b11110000 & loc;
        char lo_loc = 0b00001111 & loc;
        up_loc |= 0x80; //mask upper bit to set it toa DDRAM write
        lo_loc = lo_loc << 4; //bit shift 4 to upper 4

        //set to write to  DDRAM
        P1OUT &= 0x0F; // clear Data bus
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
        P1OUT |= up_loc; // put data out (function set - upper nible)
        __delay_cycles(16); // 1us -tw delay
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P1OUT &= 0x0F; // clear Data bus

        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
        P1OUT |= lo_loc; // put data out (function set - lower nible)
        __delay_cycles(16); // 1us -tw delay
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P1OUT &= 0x0F; // clear Data bus
        __delay_cycles(960); // 60us
        return 0;
}

/*********************************************************************/
```

```
/***************WRITE LETTER TO LCD SCREEN SUBROUTINE*****************/

int Write_Letter(char letter)
{
  char up_let = 0b11110000 & letter;
  char lo_let = 0b00001111 & letter;

  lo_let = lo_let << 4; //bit shift 4 to upper 4

              //write a letter
              P1OUT &= 0x0F; // clear Data bus
              P2OUT |= 0x01; //E = 0, R/W = 0, RS = 1
              P2OUT |= 0x05; //E = 1, R/W = 0, RS = 1
              P1OUT |= up_let; // put data out (function set - upper nible)
              __delay_cycles(16); // 1us -tw delay
              P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
              P1OUT &= 0x0F; // clear Data bus

              P2OUT |= 0x01; //E = 0, R/W = 0, RS = 1
              P2OUT |= 0x05; //E = 1, R/W = 0, RS = 1
              P1OUT |= lo_let; // put data out (function set - lower nible)
              __delay_cycles(16); // 1us -tw delay
              P2OUT &= ~0x04; //E = 0, R/W = 0, RS = 1
              P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
              P1OUT &= 0x0F; // clear Data bus

              __delay_cycles(960); // 60us

              return 0;
}

/*********************************************************************/




/*************SHIFT LETTER ON LCD SCREEN SUBROUTINE*****************/

int shift_Disp_Left(void)
{
              //Display Clear
              P1OUT &= 0x0F; // clear Data bus
              P2OUT |= 0x01; //E = 1, R/W = 0, RS = 0
              P1OUT |= 0x00; // put data out (Display Clear - upper nible)
              __delay_cycles(16); // 1us -tw delay
              P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
              P1OUT &= 0x0F; // clear Data bus

              P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
              P1OUT |= 0x10; // put data out (Display Clear - lower nible)
              __delay_cycles(16); // 1us -tw delay
              P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
              P1OUT &= 0x0F; // clear Data bus

__delay_cycles(48000); // 3ms

return 0;
}

/*********************************************************************/
```

```
/***************CLEAR THE LCD SCREEN SUBROUTINE*****************/

int Clear_Disp(void)
{
        //Display Clear
        P1OUT &= 0x0F; // clear Data bus
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
        P1OUT |= 0x00; // put data out (Display Clear - upper nible)
        __delay_cycles(16); // 1us -tw delay
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P1OUT &= 0x0F; // clear Data bus

        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
        P1OUT |= 0x10; // put data out (Display Clear - lower nible)
        __delay_cycles(16); // 1us -tw delay
        P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
        P1OUT &= 0x0F; // clear Data bus

        __delay_cycles(48000); // 3ms

return 0;
}

/*************************************************************************/
```

```
/*****INITIALIZE THE LCD SCREEN FOR 5X8 DOT MATRIX SUBROUTINE*******/

int Init_Small(void) {
            //start up sequence
            //unt on
  __delay_cycles(960); // 60us

  //Functions set
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x20; // put data out (function set - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x80; // put data out (function set - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus
            __delay_cycles(960); // 60us

            //Display Function
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x00; // put data out (Display set - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0xC0; // put data out (Display set - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus
            __delay_cycles(960); // 60us

            //Display Clear
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x00; // put data out (Display Clear - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x10; // put data out (Display Clear - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus
            __delay_cycles(48000); // 3ms

            return 0;
}


/*********************************************************************/
```

```
/*****INITIALIZE THE LCD SCREEN FOR 5X11 DOT MATRIX SUBROUTINE*******/

int Init_Big(void) {
            //start up sequence
            //unt on
  __delay_cycles(960); // 60us

  //Functions set
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x20; // put data out (function set - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0xC0; // put data out (function set - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus
            __delay_cycles(960); // 60us

            //Display Function
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x00; // put data out (Display set - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0xC0; // put data out (Display set - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus
            __delay_cycles(960); // 60us

            //Display Clear
            P1OUT &= 0x0F; // clear Data bus
  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x00; // put data out (Display Clear - upper nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

  P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
  P2OUT |= 0x04; //E = 1, R/W = 0, RS = 0
            P1OUT |= 0x10; // put data out (Display Clear - lower nible)
            __delay_cycles(16); // 1us -tw delay
            P2OUT &= ~0x07; //E = 0, R/W = 0, RS = 0
            P1OUT &= 0x0F; // clear Data bus

            __delay_cycles(48000); // 3ms

            // init complete
            return 0;
}
/*************************************************************************/
```

APPENDIX G: MATLAB SCRIPT FOR ENERGY CALCULATIONS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Senior Project: Solar Powered Hot Tub
%Designed by: Jonathan Peterson and Anthon Zepeda
%May 20, 2016
%
%This script was created in order to perform data
%calculations aiding in determining the power output
%from a 400W solar panel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Latitude, Longitude, and Day Definitions
Lat = 35.484;
Long = 120.672;
Day = 359;

%Declination Calculation
Dec = (23.45 * sind((360/365)*(Day-81)));

%Hour Angle Calculation
t = (0:1:1440);
Hour = (t/60);
H_B4_N = -(Hour-12);
HA = ((15).*H_B4_N);

%Part 1
%Altitude Calculation
Alt = asind((sind(Dec)*sind(Lat))+(cosd(Dec)*cosd(Lat)*cosd(HA)));

%Azimuth Calculation
Azm = asind((cosd(Dec).*sind(HA))./(cosd(Alt)));

for i = 1:1441
   if cosd(HA(i)) >= (tand(Dec)/ tand(Lat))
      Azm(i) = Azm(i);
   else
      if Azm(i) <= 0
         Azm(i) = (-180-Azm(i));
      else
         Azm(i) = (180-Azm(i));
      end
   end
end

Alt(Alt<0.1)=0.1;
%plot(Azm,Alt,'b');



%Part 2 - Irradaiance versus Time
AMR = (1./(sind(Alt)));

I = (1377 .* (0.7.^(AMR.^0.678)));

%plot(t,I2,'b');



%Part 3 - Incident Energy
W = t(2:1440)-t(1:1439);
H = ((I(2:1440)+I(1:1439))/2);
IE_Array = W.*H;
IE = sum(IE_Array); %Wmin/m2
IE = (IE/(60000)) %kWh/m2

%Day Light Hours
HSR = acosd((-tand(Lat))*(tand(Dec)));
SR = (HSR/15);
Day_Light_Hour = (SR*2);
```

```
%Average Incident Energy per Daylight Hour
Avg_IE = (IE/Day_Light_Hour);


%Part 4 - Power from Panels
PWR_Array = (IE_Array .* 0.2 .* t(1:1439) .* (1/1000)); %kW/m2
PWR = sum(PWR_Array); %Total kW

%plot(t(1:1439),PWR_Array,'b');


%Part 5 - Temperature Problem
Sum_Temp = (((30*sin(0.0043633.*(t-200)))+75)-32)*(5/9); %Degrees C
Wint_Temp = (((30*sin(0.0043633.*(t-50)))+51)-32)*(5/9); %Degrees C

Sum_Pan_Temp = (((52.5*sin(0.0043633*(t-200)))+97.5)-32)*(5/9); %Degrees C
Wint_Pan_Temp = (((24*sin(0.0043633*(t-50)))+56)-32)*(5/9); %Degrees C

Operating_Temp = 25;

Sum_Eff = 0.20*(1-((Sum_Pan_Temp - Operating_Temp).*0.0038));
Wint_Eff = 0.20*(1-((Wint_Pan_Temp - Operating_Temp).*0.0038));

PWR_Array_Eff = (IE_Array .* Wint_Eff(1:1439) .* t(1:1439) .* (1/1000)); %kW/m2
PWR = sum(PWR_Array_Eff); %Total kW

plot(t(1:1439),PWR_Array,'b',t(1:1439),PWR_Array_Eff,'k');


%Part 6 - Total Energy Generated for both Part 4 and 5
PW = t(2:1439)-t(1:1438);
PH = ((PWR_Array(2:1439)+PWR_Array(1:1438))/2);
Tot_Energy_Array = PW.*PH;
Tot_Energy = sum(Tot_Energy_Array); %Wmin/m2
Tot_Energy = (Tot_Energy/(60)) %Wh/m2

PWE = t(2:1439)-t(1:1438);
PHE = ((PWR_Array_Eff(2:1439)+PWR_Array_Eff(1:1438))/2);
Tot_Energy_Array_Eff = PWE.*PHE;
Tot_Energy_Eff = sum(Tot_Energy_Array_Eff); %Wmin/m2
Tot_Energy_Eff = (Tot_Energy_Eff/(60)) %Wh/m2

%Average Generated Energy per Daylight Hour
Avg_Gen_Energy = (Tot_Energy/Day_Light_Hour)
Avg_Gen_Energy_Eff = (Tot_Energy_Eff/Day_Light_Hour)
```

APPENDIX H: WORKS CITED

1. "Best Hot Tubs for 2015 - ConsumerAffairs." *ConsumerAffairs*. N.p., n.d. Web. 26 Oct. 2015.
2. "Consumer Perspectives on Sustainability Choices." *EcoFocus Worldwide*. N.p., n.d. Web. 26 Oct. 2015.
3. Deziel, Chris. "How to Change My Hot Tub From Electric to Solar." *Home Guides*. SF Gate, n.d. Web. 26 Oct. 2015.
4. "Evolving Smart Technologies Across Home Appliance and Consumer Electronics Markets." *Appliance Design Magazine RSS*. Intertek Testing Services. Web. 26 Oct. 2015.
5. Hammel, Cailley. "A History of Innovation: Hot Tubs." *AQUA*. AQUA Magazine, 1 Sept. 2013. Web. 26 Oct. 2015.
6. "How And Where To Buy a Swimming Pool Solar Heater." *Solar Energy For Homes*. N.p., n.d. Web. 26 Oct. 2015.
7. "Industry Reports." *Market Research Reports*. N.p., n.d. Web. 26 Oct. 2015.
8. "Jacuzzi Hot Tubs & Spas | Jacuzzi.com." *Jacuzzi Hot Tubs*. N.p., n.d. Web. 26 Oct. 2015.
9. "The Most Eco Friendly Hot Tub Spas." *Energy Efficient Hot Tubs*. Jacuzzi® Hot Tubs, n.d. Web. 26 Oct. 2015.
10. "People Living in Households That Own a Pool, Hot Tub or Spa, USA 2014." *Statista*. N.p., n.d. Web. 26 Oct. 2015.
11. Siegal, RP. "Study Shows Bright Green Future for Smart Appliances." *Triple Pundit People Planet Profit*. N.p., 09 Mar. 2010. Web. 26 Oct. 2015.

12. "Solar Hot Tub." *SOLAR HOT TUB.COM*. ABC Solar Incorporated, n.d. Web. 26 Oct. 2015.

13. "Solar Markets Around The World." *Solar Markets*. N.p., n.d. Web. 26 Oct. 2015.

14. Wedgeworth, Cicely. "Where in America You'll Find Homes With Hot Tubs, in 2 Maps." *Real Estate News and Advice Realtorcom*. N.p., 21 May 2015. Web. 26 Oct. 2015.