Converter Design for Solar Powered Outdoor Mobile Robot

Josue Cruz-Lambert, Patrick Benavidez,
Jacqueline Ortiz, Jack Richey, Shane Morris, Nicolas Gallardo, and Mo Jamshidi
Department of Electrical and Computer Engineering
The University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
Email: josueilambert@yahoo.com, patrick.benavidez@utsa.edu,
jaqdandori@gmail.com, xje791@my.utsa.edu, shane.morris33@yahoo.com, hbq744@my.utsa.edu, moj@wacong.org

***Abstract*—This project presents a hybrid system implementing the use of solar panels and batteries to power a robot. The main aim is to integrate a charging system which allows the batteries to be charged from solar panels, wall outlet, and a deployable solar charging station. The proposed system is divided in three sections: design of solar panels, design of a battery charger and design of a DC-DC converter with fuzzy MPPT tracking system. This paper will cover only up to the design of the DC-DC converter, as further work is still pending design/results. For the design of the solar panels, different cell configurations were considered. Once the solar panels were designed and fitted to the robot, we determined the battery requirements to meet the robot power load and payload. Lithium Polymer batteries were chosen as the power source for the robot since they have a competitive power density to weight ratio. In order to extend the battery life, and simplify the load, we decided to take two battery banks. One battery is charging while the other battery is discharging. This allowed us to precisely control the battery charging profile without load variations interfering with our measurements. We used a fuzzy maximum power point tracking controller with a primary focus on regulating the power input for a lipo. Different topologies of DC-DC converters were considered and based on our literature research, it was concluded that a Buck-Boost converter is the most appropriate option when working with solar cells. The following paper discusses all the design specifications, component decisions and construction of the solar powered robot. Various technologies, not used in the robot, are included here as a literature review of the current state of the art. The goal of this paper is to summarize the tested methods and results to expedite future researchers in the correct direction.**

***Index Terms*—Solar, Renewable, LiPo, Lithium Polymer, Robotics, Fuzzy Controller, Energy**

I. Introduction

This paper is a summary of the various techniques and technologies used to design a solar powered robot. Although no final results are included, this paper explores the current available products, software and their limitations for this application. The aim of this paper is to bring new college up to date with the current state of our research.up to date with my research, in turn saving them the time involved with the discovery process. Section two will explain the goals and objectives of the project. The following sections are organized in the chronological order that they were explored in.

II. Design Constraints

In this project, we are interested in installing a solar panel on a mobile robot platform. The main objective is to have the robot operate outdoors without the need of external power supplies. To start off the debate, we wanted a robot that would operate 100% on solar power. However, the solar cells an individual can get a hold off are only up to 15% efficiency. Attempts to acquire solar cell from reputable manufacturers went unanswered. Since our initial focus was to acquire the most efficient cells for their weight, we had to refocus our goal on what would actually fit on the robot. Since we were limited by the physical space of the robot, we considered optimizing the structure of the cells. Some form factors are pyramid arrangement, inverted pyramid, accordion, and flexible membranes. After ordering and producing a small sample of two cell configurations, we determine that a flat solar panel would be the easier to manufacture, marginally more efficient than the other configurations. Once we decided on the solar cell structure, we began to construct a solar cell panel. As a forewarning to the reader, construction of the solar cells is a time consuming endeavor, approximately 28 hours per panel. Building a solar cell on the sole merit of saving money is not a good reason to make your own after factoring in the time, cost, and materials. Our recommendation would be to spend more on an off-the-shelf component if possible.

From the initial batch of solar cell, we ordered 100 units to create our panel. In the selection process, we picked the solar cell that claimed to have the best watts/meter rating. Normalizing the results, and then selecting the best for our robot. After receiving them, we started to characterized them, and record their performance. More details are provided in section four, solar panel modeling.

Our next focus was the load the robot would take, having a fixed power generation, we determine the best approach was to maximize our power usage. To simplify the equations, we restricted the solar panel to exclusively charge lithium-polymer batteries. This would in turn reduce the complexity of the circuit to two components, the buck-boost converter, and a single battery to charge. To further reduce the load disturbance on a panel, we added an additional bank of batteries. The goal is to have one bank always charging, therefore only having the

load of a buck-boost converter inefficiencies, as well as the battery charge cycle. The second bank is used to be discharged into the transients of the robot, from CPU, sensors, and other miscellaneous equipment. It is at this point where we needed to start estimating the load capacity to properly size the batteries. Since the robot doesnt have a direct objective, and is rather a platform for students to use, we decided to equip it with the standard ACE Lab payload. See Table III for a breakdown on the components and expected load.

TABLE I: ACE Lab Standard Robot Payload

Item ID	Item Description	QTY	Expected Load Total
1	Motor (Left/Right)	2	$200.0W$
2	Xbox Kinect	1	$12.96W$
3	ODROID	1	$20.0W$
4	Head Rotation Stepper	1	$3.96W$
5	High Torque Servo	1	$6.6W$
6	USB Wi-Fi Adapter	1	$2.5W$
7	Microcontroller (atmel mega class)	1	$1.25W$

From the table above, we can see that the largest load is the motor/drivers. We then determined the size the batteries should be to match the motor demand,to reduce the performance impact on the robot. First we considered the working voltage of the motors based on the manufacturer specifications:

$$12.00V \pm 2.0 < V_{motor} < 24.00V \pm 2.0 \quad (1)$$

Using standard lithiumpolymer batteries, each battery has a cell nominal operational voltage of 3.7 volts. We used the industry standard of 3.5 volts to consider a cell fully depleted, and 4.2 volts fully charged. This data was taken directly from the battery distributor hobbyking.com [8]. With these parameters, we can then estimate for the operating voltage range of the batteries.

$$21V < V_{bat} < 25.2V \quad (2)$$

Lastly, we needed to calculated/estimate our watt/hours to specify our operational time. We analyzed what power modes the robot would work on and determined there are two main modes. Full operational mode where the previous table shows worse case loading of the system. The second mode we call sensory reconnaissance power mode where only the sensor which we believe to be running non-stop would be powered. The items we identified are as follows. The main controller (item 3), 3d vision (item 2), and connectivity (item 6) are always operational during idle stage and during reconnaissance. Therefore, the solar panel needs to charge our batteries equivalent to the power consumption of items 2, 3, and 6 to break even ideally, plus losses. This we determined should fit the 1 hour of operation time frame. At worst case, the draw of both motors at full power would mean $200W/Hr$. Since the main constrain on the maximum power output of the solar panel was the size of the cells vs efficiency, we were able to calculate that a $50W$ solar panel could be installed and fit within the robot dimensions, given publicly available solar cells data, and our own measured cells. This effectively gave us a target load of between $35.46W$ to $47.27W$, the max constant load derived from all but the motor drivers running. Therefore a battery of about 50 watt hours would be needed. For us, we knew we could realize this goal with two lipos rated at 5AH, 11.1 volts. Since we want to drive the motor at max speed, it would be later necessary to add an additional battery in series to increase the voltage to 22.2, within the working range of the motor.

In summary, we seek to optimize the amount of time a robot can spend outdoors. Having identified the largest and most constant load on our system, we designed a solar panel of $50W$ at $100W/m^2$ irradiance. Although this solar panel gave us 29.08% down to 5.46% buffer from the reconnaissance load to almost full load requirements, it is restricted mostly by the state of the art technology, as well as the surface area the solar panel takes up. The main limitation are, listed in order of importance, size, cost, and efficiency. Once the solar panel was sized and limited, we determine the batteries bank should be able to provided enough power for one hour of operation in reconnaissance mode. While in worse case scenario loading, the robot would only be operational for approximately 15 minutes. At this point, the robot would switch to the second battery bank and deplete that source, increasing our operation time to a little under 30 minutes in an ideal case.

III. Simulation Design

Having defined and constrained our robotics platform, we began to simulate the electronics payload and sources with the aid of Matlab. At first, we started by simulating the entire system in one model. The following screen clips are the view of our first attempt, sub-divided from left to right in Figure 1 and Figure 2.

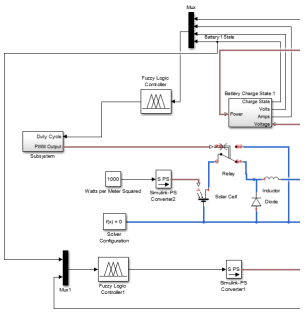

Fig. 1: Buck Converter Circuit

Without going into too much detail, making a simulation of this scale work on the first few trials prove to be fruitless. This is mostly attributed to errors in one section of the project propagating to the outputs on other sections, therefore occluding the real cause of the problem in the first place. It then became necessary to isolate the problems, and tackle them individually. The next sections will cover each sub-system, following the traditional divide and simulate strategy. []

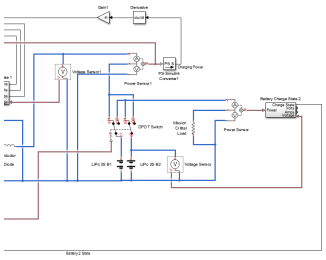

Fig. 2: Battery switches and state function

IV. Solar Panel Modeling

We started to seclude the simulation to individual sections by first starting with the solar panel. We used the PowerSim library for matlab in order to have all the advance pspice simulation components already tested by a trusted source. To create the solar panel, we measured our panels performance on a sunny day, and inputted those values into the PowerSim solar cell model. Leaving only the default parameters were we couldnt measure or otherwise quantify the need to replace default values.

The following parameters were used for our simulation, based on the measured values of our cells.

TABLE II: Solar-Cell Characteristics

Item ID	Parameter	Notation	Value
1	Short Circuit Current	I_{SC}	$4.08A$
2	Open circuit voltage	V_{oc}	$0.6V$
3	Irradiance used for measurements	I_{r0}	$1000.0W/m^2$
4	Quality Factor	N	$1.5W$
5	Series Resistance	R_s	0Ω

For those unfamiliar with the powersim library, the following equation was used to solve for the output current of the solar cell.

$$I = I_{ph} - I_S * \exp(\frac{V + I * R_S}{N * V_t}) - (V + I * R_S)R_p$$

where

$I_{ph} = I_r * (\frac{I_{ph0}}{I_{r0}}) \rightarrow$

$$\left\{ \begin{array}{c} I_{ph0} \text{ = solar generated current} \\ I_{r0} \text{ = measure irradiance for } I_{ph0} \end{array} \right\}$$

I_S = diode saturation current

V_t = thermal voltage

N = quality factor = diode emissions coefficients

In order to verify our model match our measured values, we designed the following test circuit in Matlab/Simulink as shown in Figure 3.

After running the simulation for 10 seconds, we obtained the following results during simulation in Figure 4.

These results showed our panel should be able to output a peak of 37.75 W. Just enough power to fulfill our requirements stated in section 1, but on the low end of our expected 50

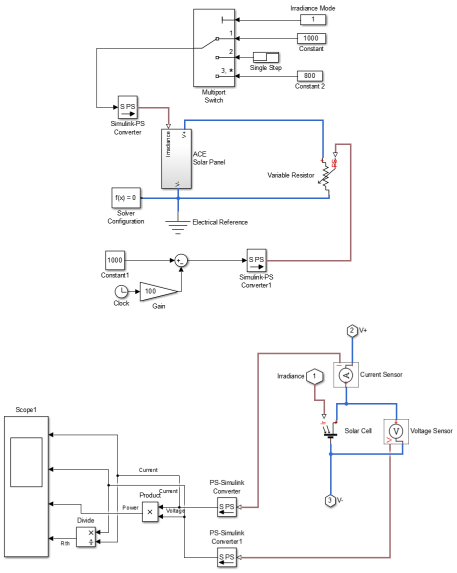

Fig. 3: Inside the Solar Panel sub-system. A solar cell with many probes and scopes

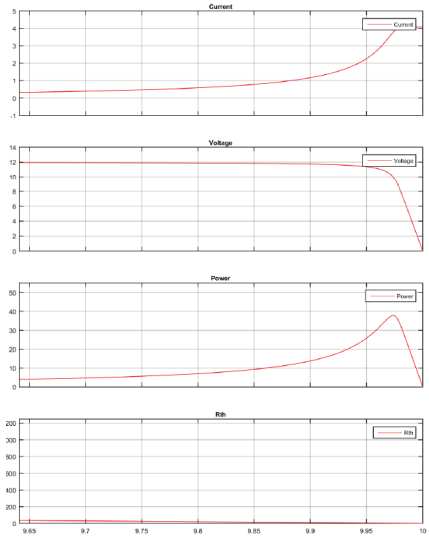

Fig. 4: Simulation results

watts. We reviewed the solar cell sellers description and found that we were expecting in the neighborhood of six amps. Re-running the simulation with the short circuit current set to 6.0 A resulted in a 55.5 watt panel. At this point in time, we still did not have a full panel built, therefore it was only estimates we were comparing. After we constructed the panel, we determined that 5.8 Amps was the short circuit current. This resulted in a 53.5 watt estimated panel.

After reviewing the simulation results, we determined functional representation of our real world panel, and continued adjusting the parameters as more measurement data became available.

V. DC-DC Converter Modeling

Initally, we thought we could use a buck converter. Once we modeled it, we realized that the buck worked, but lacked a principal component. The solar panel is expected to output about 10 volts during peak power [1]. A single battery under peak demand would need about 10.5 volts at 5 amps [2]. Since it wouldnt be possible to meet the battery demand and solar cell characteristics, the converter would need to step-up the voltage, at reduced current output. It was then necessary to use a boost converter, but with a dropout voltage of at least 1 volt, it wouldnt be feasible either. Thats why we ended up with a buck-boost converter. Since the boost converter uses the inductor independently to power the circuit, it would be theoretically possible to produce any voltage we needed for the batteries. Since our input voltage is within 2 volts from our output voltage, most of our losses would be due to current losses in the inductor, and not on voltage conversion. This is how we conceptually justified using a buck-boost converter. Further testing is still being developed to verity this claim. The following is a screen shot of the simulation model we used on matlab to create a buck-boost converter.

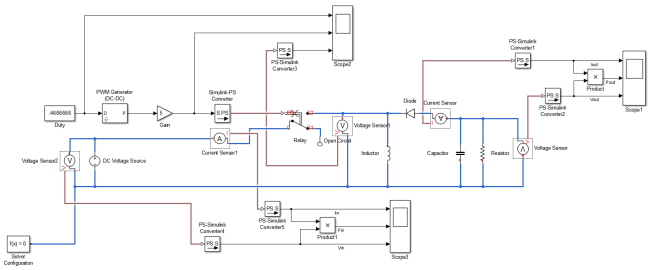

Fig. 5: Matlab Buck-Boost circuit [3]

Here is were the majority of the time is spent for this project. Although matlab comes with built-in DC-DC converters, they lack controlability of external systems. Most off the shelf converters, and the matlab pre-made models have built in controllers that regulate the voltage and current internally. For our project, we wanted to build a fuzzy controller that directly drives the pwm duty cycle of the buck-boost converter. Using the circuit shown on the previous image, we decided to implement the verbatim as shown on the buck-boost converters chapter from the textbook. [3]

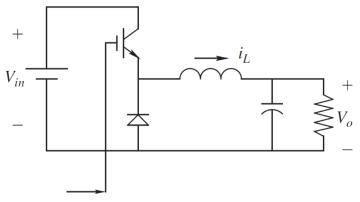

Fig. 6: Buck boost circuit

The following values were used for our passive components. We determined the inductor value, L1, by using the equation

TABLE III: Buck-Boost Parameters

Item ID	Description	Value	Units
L1	Fixed Inductor	1	mH
C1	Fixed Capacitor	2000	uF
u1	PWM Generator	400	kHz
D	Duty Cycle	0.9	N/A

provided by Ned Mohan [3] textbook:

$$L = \frac{Vin}{\Delta_{i_L}}(D * Ts) \tag{3}$$

Where

$$Vin = 12V$$

Δ_{i_L} = $25mA$ to $5mA$

D = chosen arbitrarily for the simulation, later determined by the fuzzy controller.

$$Ts = 1/f_s = 1/400k$$

In the literature, the iductor value was specifically chosen for the task. IN contrast, the capacitance value consideration is omitted from the textbook [3]. As a starting point, we found a boost converter in the lab that had similar performance to our buck-boost. From it, we found an capacitor rated at 2000 uF. Therefore, we assumed this would be a good starting point for our simulation. The frequency value was chosen from examples in the textbook.

Once the simulation was configured, we ran it and found that the simulation clock would stop after 2.5 uS. Although we repeated the simulation using various duty cycles, and other stepping methods, every simulation took more than 5 hours to run, and stopped after a couple micro-seconds. This was unacceptable for controller design, since we needed to test and calibrate various controllers. We decided matlab simulink was unable to deliver the performance we needed. After attempting to optimize our circuit in matlab, we decided to test another program. We then recreated the circuit in Proteus 8 professional. The following is a screen clip of the circuit built to simulate the buck-boost converter.

After building the circuit shown in the previous picture, we ran the simulation and found it actually ran significantly faster. However, a quick simulation was only the beginning of our problems, as the output from the simulation did not show desirable results, shown in the next picture.

From the oscilloscope output, we see near zero voltage in blue (second line from the top) and a significant voltage spikes. Since we don't want a ripple voltage on the batteries, we attempted to mitigate the problem by altering the capacitance value. We tried 1 UF, 10uF and even 100uF. For each simulation, the results did not change significantly. From our equation earlier, we expect a 1.08mH inductor to give us a ripple current of about 25mA. Having a voltage of nearly zero became our biggest hurdle. Further literature reading is necessary in order to properly size the capacitance. In a last attempt to get a better understanding of the system, an android application named EveryCircuit was used to test the system response. With the aid of EveryCircuit, we gain

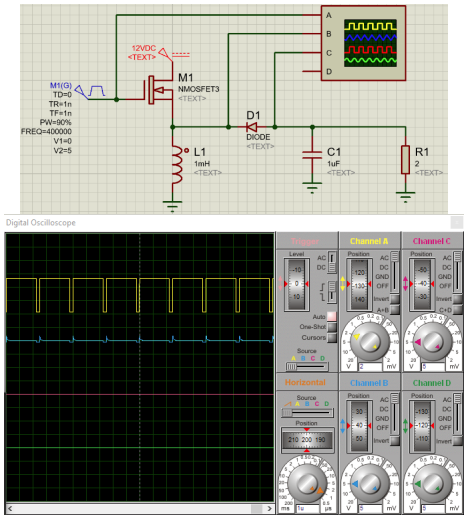

Fig. 7: Proteus model output of buck-boost converter, TOP/YEL PWN, BELOW/BLU: Voltage Output of inductor. Middle/RED: Output Voltage

an understanding that the capacitor was being charged when the inductor was discharging, the capacitance was not large enough to remove the ripple on the output. Every simulation attempt made to change the capacitance value resulted in either the programs crashing, or undesireable results. Therefore, a more extensive literature review is needed to understand the requirement for matching capacitance to the inductance.

A. Hardware Review

Since simulations were holding us back, we considered building the circuit and testing the controller on an actual live circuit. Although this case was not ideal, the documentation and papers regarding buck-boost controllers made us confident that we could create an inexpensive circuit. At first, we started using the AVR451: BC100. This is a prototype circuit that has three buck converters, each built to specific power levels. Although the circuit proved to be fully working, it had a few limitations. Most importantly, the support for this hardware is practically non-existent. Outside the included technical library, limited external/update documentation was available [4] [5]. This became a huge driver since all the code was written for atmels avr studio from 2007, it became pretty challenging to import the code into the new IDE from atmel. However, thanks to the work of Rosenberg [6], most of the code had been ported to somewhat compatible version of AVR Studio. However, as another BC100 user noted ”perhaps I should have chosen a more recent reference design.” [7] As we started to understand and attempted to update the code to accept new functions, it became more apparent that the original author, AVR, spent considerable amount of time optimizing the code. Using modern libraries and code conflicted with the existing, dated code. We decided to abandon this platform and create our own using the more widely supported, arduino platform. Within a matter of days, we already had the output PWM signal of 1.38 Khz, and a adjustable duty cycle via serial communications. While we were still away from a fuzzy controller, we had created the core foundation for our micro-controller to accept higher level control signals. With this, we started to design the core of the fuzzy controller. Although the tuning would have to wait until we had a full simulation.

VI. Controller Design - Preliminary

To start our controller, the fuzzy membership functions and rule set were first created. The Fuzzy system is divided into two controllers. One determines which battery needs to be charged, based on the charge state of each battery, and usage. The other fuzzy controller, directly controls the PWM signal of the buck converter. Each controller is named accordingly to their function, and as such we have battery_selector.fis and battery_charger.fis. The battery selector fuzzy controller consists of taking the following state during a battery charge cycle:

1) Empty The battery voltage is at or below 3.5 and considered empty.
2) Maximum Current The battery may accept up to 1C to recharge its initial 80-90%.
3) Fixed Voltage The battery voltage is now at 4.15 per cell, and requires a constant voltage to reach 100% capacity.
4) Full Battery the Battery is considered full when the voltage has reach 4.20V per cell.

Using these four states as an input, our battery selector fuzzy controller determines which batteries will benefit the most from charging, assuming a maximum irradiance is available. The output is a logic true/false from toggling a DPDT relay. Below are some screen shots of the fuzzy toolbox.

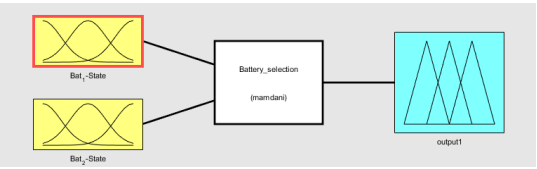

Fig. 8: Fuzzy inference system – SystemBatterySelector.fis

Next, the battery charging controller was designed based on the following constraints:

- How much current the battery is being charged at, and voltage.
- The power output of the solar panel after the buck converter.
- The state which the battery charging cycle is at.

Since we can only control the output voltage of the buck converter using conventional techniques, it became apparent that this would be the most complicated portion of the controller design. Below are some of the initial controller designs memberships functions, rules and outputs.

VII. Controller Design

Overall, we had partial success in simulation individual sections of the project. The solar cell and panels were simulated according to our actual measure values. Although the

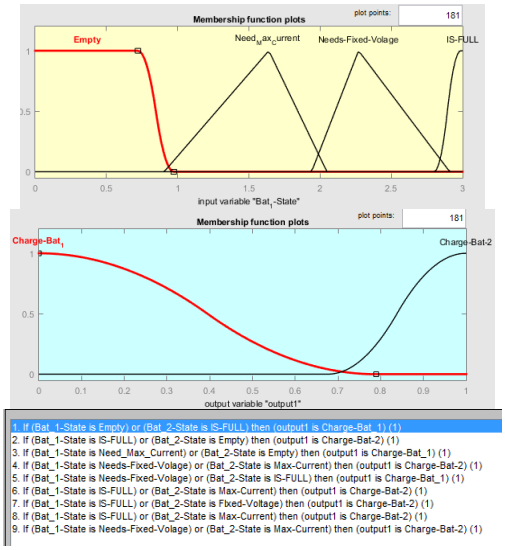

Fig. 9: Input states, output state and rules for the fuzzy inference system

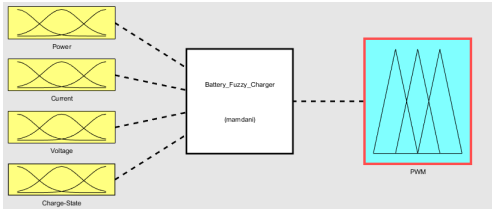

Fig. 10: System: Fuzzy Battery Charging

panel still needs to be fully characterized in various irradiance levels, initial results are very promising. simulation of a buck-boost controller has proven to be more challenging than anything else. After trying various software, we only have achieved superficial results. The decision to implement the hardware before the simulation is done is not recommended, and therefore the hardware testing has been suspended until more literature review is evaluated. Since our system isn't fully developed, the fuzzy controller has been created with an understanding of the system goals. Actual implementation of the controller will be done via a linux computer, and micro-controller for digital signal processing. As we continue to refine the individual sub-systems simulations, it is becoming increasingly difficult to add more complexity to the system without having a direct impact on the simulation performance. Therefore, each sub-system will be tested independently of each other. For initial simulation, we are treating all systems as linear time invariant system, each system will be configured in a controlled environment, test and evaluated, one variable at a time. Once we are acquire more reasonable results, an actual circuit will be implemented for the final, all system inclusive test.

VIII. Conclusions and Future Work

Overall, the project is moving at a slower pace than initially estimated. Attempts to use existing hardware proved challenging due to support/dated equipment. While various software and hardware were tested, we have determined that the best approach to designing a custom power solar robot is to design all the components from the ground up. With the exception of solar panels, the electronics should be sized and built according to the demand of the robot. In our current state, further simulation is needed before we can proceed with a high level of confidence to the embedded hardware. However, as shown, it is possible to start initial hardware testing and software coding. Another underestimated challenge that was encountered while trying to get the robot ready for implementation was that most of the hardware was in factory like state. It was then necessary to also focus on manufacturing the robot, and not just electronics programming. More than 50 hours were spent designing CAD files for 3D printing. Since manufacturing has become an iterative process, many parts were printed a couple of times to get the fit/functionality right.

We then connected our circuit in the matter shown below. From the previous figure, the goal is to have one battery ded-

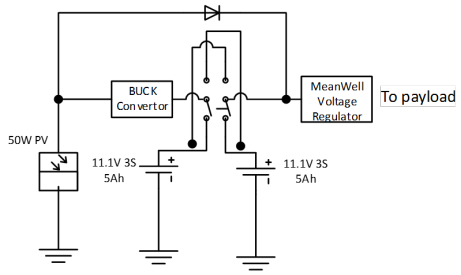

Fig. 11: Switching buck-boost converter

icated to charging, while having the other battery discharging. This allows the solar panel to be optimized for only the battery load. Since we are charging near unity, in a full sunny day we expect to find the robot switching between batteries every hour, and recharging the other battery every hour. Effectively, causing our estimated load of 46.02 W to be continuously powered. For future work, the goal is to have the robot fully assembled and partially wired for implementing and testing the power controller.

References

[1] B. Subudhi and R. Pradhan, "A comparative study on maximum power point tracking techniques for photovoltaic power systems," *Sustainable Energy, IEEE transactions on*, vol. 4, no. 1, pp. 89–98, 2013.

[2] J. Fattal and P. Bou Dib Nabil Karami, "Review on different charging techniques of a lithium polymer battery," in *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 Third International Conference on*. IEEE, 2015, pp. 33–38.

[3] N. Mohan, *Power electronics: a first course*. Wiley, 2012.

[4] "AVR451: BC100 Hardware User's Guide - doc8088.pdf." [Online]. Available: http://www.atmel.com/images/doc8088.pdf

[5] "AVR458: Charging Lithium-Ion Batteries with ATAVRBC100 - doc8080.pdf." [Online]. Available: http://www.atmel.com/images/doc8080.pdf

[6] "Projects/avr_bc100.git." [Online]. Available: http://git.kpe.io/?p=avr_bc100.git

[7] "Avr microcontrollers forums topic - megaavr and tinyavr bc100." [Online]. Available: http://www.avrfreaks.net/forum/bc100