Final Report

Devin Hernandez
Electrical Engineering
The University of Texas at Arlington
Arlington, Texas
devin.hernandez2@mavs.uta.edu

Jose Guzman
Electrical Engineering
The University of Texas at Arlington
Arlington, Texas
jxg3557@mavs.uta.edu

Jonathan Dolormente
Electrical Engineering
The University of Texas at Arlington
Arlington, Texas
Jonathan.dolormente@mavs.uta.edu

Nguyen Thai Binh Duong
Electrical Engineering
The University of Texas at Arlington
Arlington, Texas
bxd3589@mavs.uta.edu

***Abstract*— This report will outline key details concerning the progress made and goals achieved for our PV-Battery Integrated Energy Source project. We will first cover the introduction to our project, our procedures for the project, and the results we gather from performing certain tasks, followed by our results and discussions of results.**

Keywords—Arduino, battery, photovoltaic, MOSFET, MPPT

I. Introduction

In this project, we have been tasked with creating a PV-Battery Integrated Energy Source circuit starting from simulation and proceeding to a final product fabricated from physical components soldered to a printed circuit board (PCB). The procedure, results, and discussion will be in numerical order from two to four. We have not yet reached the final phase of the project, as our PCB design and soldering phases have yet to be completed. As of this report, we are testing the automation of the prototype and reworking the PCB design.

A crucial principle employed in this lab is Maximum Power Point Tracking (MPPT), utilized in photovoltaic (PV) integrated systems to optimize power efficiency. The power output of a PV array is influenced by several factors, including temperature, cloud cover, and irradiance. Hence, the implementation of MPPT is imperative to ensure the harnessing of maximum power regardless of these varying conditions. Figure 1 illustrates the method through which MPPT is achieved.

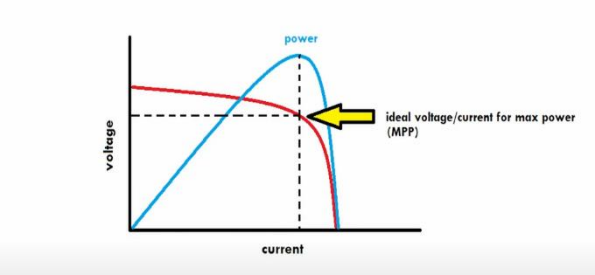

Figure 1. Typical I-V curve for a solar panel

It is essential to highlight that this representation pertains to a specific scenario and may not be universally applicable. The process involves regulating the voltage, which in turn affects the current emanating from the PV array. The maximum power point is identified by maximizing the product of voltage and current. Consequently, the optimal combination of voltage and current values is determined and adjusted to ensure peak efficiency.

MOSFETs play a significant role in allowing voltage to be stepped up or down in the buck-boost converters used. These devices are sent a PWM signal by the Arduino that tells it exactly how long to turn on and off. This will either allow the voltage to be stepped up or down in the case of buck-boost converters (in which we are using two). For buck-boost converters, the circuit will either be in buck mode when the PWM signal is below 50 percent duty cycle or be in boost mode when above 50 percent duty cycle. This will allow for the manipulation of voltage and therefore current, in turn allowing for a phone charger and battery to be charged.

II. Procedure

We started the project by doing research and simulation, as well as assigning roles to our four group members and setting up a basic schedule for our relevant tasks. We were given a large set of literature to use for the project, each containing different topics relating to the project, and other literature that contained information on components that could be used. We were also given some MultiSim files that would be useful for the project.

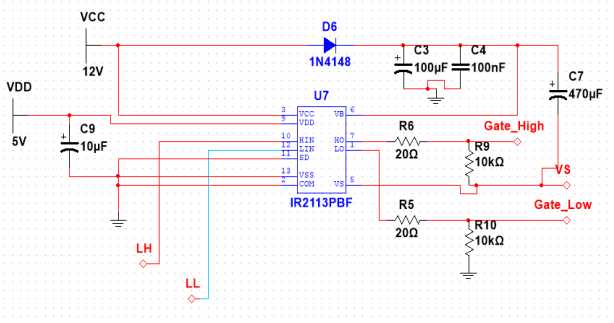

Figure 2. A circuit diagram of the gate driver that we are using. This was provided to us for simulation purposes.

After quickly reviewing our supplied materials, we decided that our group roles would be as follows:

- Jose: The Project Manager of the group; responsible for developing circuit topology, aiding in prototyping and implementation, additional research, and schedule adhesion.
- Devin: The Designer of the group; responsible for maintaining the simulation documents, programming, spearheading the creation of the PCB, and double-checking construction quality.

- Jonathan and Binh: In charge of Prototyping and Implementation. Tasked with the construction of the breadboard prototype and soldering components to the PCB.

At this point in the project, Devin made a list of parts to order for the project so that Jonathan and Binh could put together a prototype, represented in the table named "Parts List 1" appended at the end of the report. We have been given a budget of $230 by the department to order the parts that we need to complete the project. As the prototype was being built with the parts that were ordered, Jose and Devin worked on programming the Arduino to produce a PWM output required by the two MOSFETs in the Bidirectional DC-DC circuit to function properly.

A. *Breadboarding*

First, before any prototype can be created, a circuit topology must be laid out. To begin this process, research must be done to determine the best and most efficient way to connect all our major components. Article [1] gave insight into how our PV array and DC-DC converter should be connected to the rest of our components. The layout for that portion of the circuit can be seen in Figure 3.

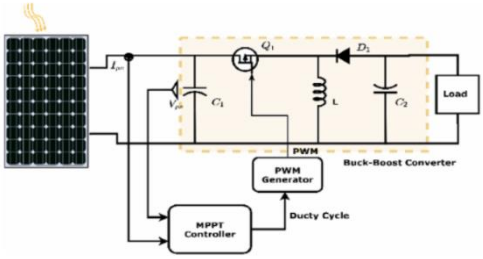

Figure 3. A general layout for an MPPT-controlled buck-boost converter.

It should be noted that in Figure 3, MPPT occurs before the DC-DC converter. This means the best voltage and current values will be input into the DC-DC converter. Since it is now determined where the load will be located, the Bidirectional DC-DC converter and battery can be connected in parallel with it. These DC-DC converters will need to have their separate gate drivers, which are then controlled by the Arduino. After a layout for the circuit was found, the final topology was created. It can be seen in Figure 4.

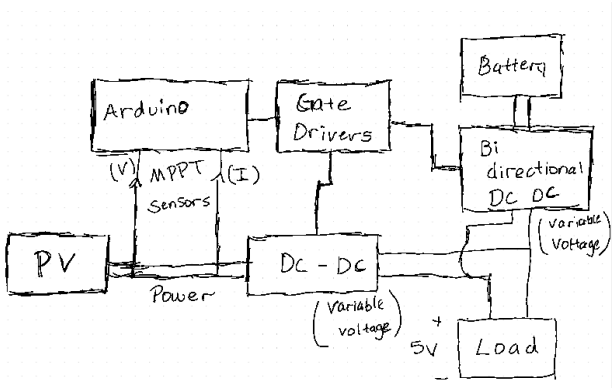

Figure 4. The proposed circuit topology for this project.

The final topology included in Figure 4 accounts for all major components in the circuit. Figure 4 is the foundation of the project; once this is completed, prototyping and implementation could begin.

Jonathan and Binh worked to create the two gate drivers, one is seen in Figure 5, on the top breadboard. They also worked to create both of the DC-DC converters, both of which are shown in Figure 5 and Figure 6.

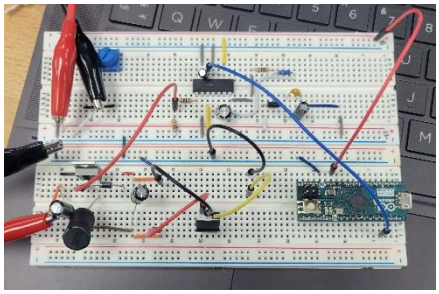

Figure 5. Two breadboards containing the buck-boost converter, its gate driver, and the Arduino Micro.

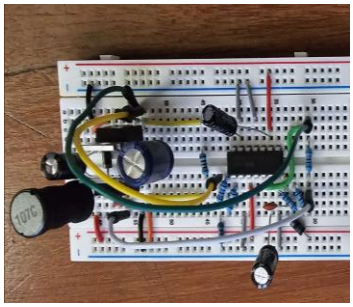

Figure 6. The Bidirectional DC-DC converter with its gate driver.

To save space, the Bidirectional DC-DC converter and its gate driver were condensed onto one breadboard, taking up only about half of the breadboard as a whole, as seen in Figure 6.

While testing the conventional buck boost converter, we immediately ran into issues with the functionality of the circuit. The output was not what was expected of the circuit, because of this, more testing was done on both the DC-DC converter and gate driver circuit. After careful testing, the group determined the issue stemmed from the gate driver. The low output signal

was not detected on the oscilloscope and therefore not received by the MOSFET, thus rendering the circuit inoperable. To remedy this, voltage measurements were taken at important nodes in the gate driver circuit to determine where the issue could be. We found multiple issues with the configuration of our circuit, and these issues were immediately fixed. After these revisions, the gate driver was able to operate as intended with both low and high frequencies detected by the oscilloscope. With a working gate driver, the team proceeded to test the DC-DC converter; however, after some brainstorming, the team determined to swap the classic buck boost converter with a non-inverting DC-DC converter. The constructed non-inverting DC-DC converter can be seen in Figure 7.

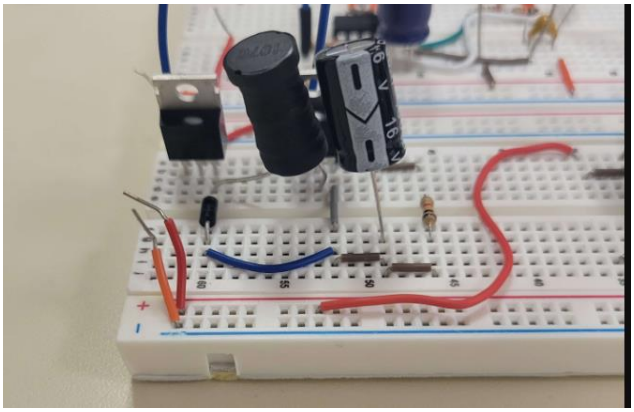

Figure 7. Constructed non-inverting DC-DC converter.

Once created, testing could begin; those results are given in the results section of the report. The last section of the procedure involved the making of the bi-directional DC-DC converter. This circuit was by far the most complex but straightforward circuit. Construction of this circuit proved simple and there were little to no setbacks for this circuit. The circuit is shown in Figure 8.

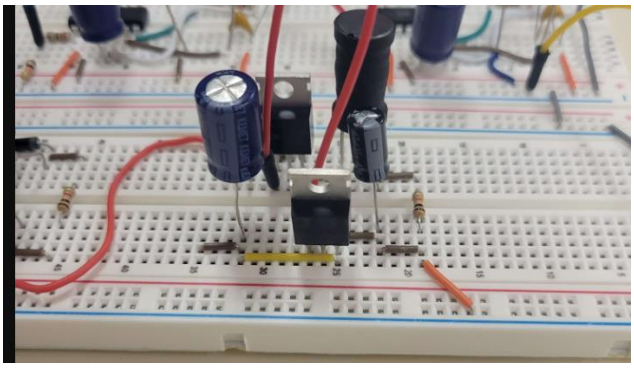

Figure 8. Bi-Directional DC-DC circuit

With the completion of the bi-directional circuit, testing for this circuit could now be done. Those results are put into the results section as well. The operation of this circuit is controlled by two MOSFETs, it switches directions based on which switch is on and which is off. It should also be noted that boost and buck modes are configured to be in a certain direction, so where the voltage sources are input are important. Once we were able to verify that the bidirectional circuit worked, we could then begin to combine all our circuits. This would be done with both converters in parallel with each other, with two gate drivers in total driving each converter, with there being two in total. The next section will provide more detail with the switch of the Arduino Micro and the Uno. With the entire set up, tests could be completed, and the results are again given in the results section. The finished prototype is shown in figure 9.

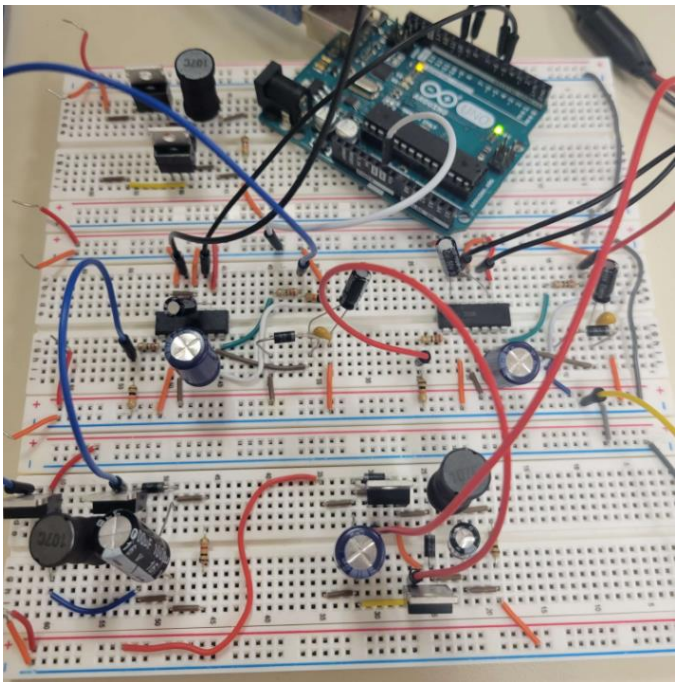

Figure 9. Finished prototype.

B. Programming

The programming has been especially difficult. Devin worked through four different prototypes of the code, with each prototype using a different implementation of a timer. The prototype attempted to use digitalWrite(); to make a PWM signal directly, though we have not yet tested code to perform the MPPT calculations. This first method did not work because, while it is simple to create two static complementary PWM signals with the delay(); method, we could not find a way to implement adjustable timing to the code as required by the project to function properly. The second version of the code used analogWrite(); to generate the PWM signal, but we found that the map(); method could not generate the signal at the bit rate required, as it performs integer math to make the mapping. The third version of the code takes advantage of the timer registers supplied by the ARM64 processor on the Arduino. We do not truly understand how the timer registers work, but we do know that the code works as it should. We made a fourth version of the code that uses the TimerOne library, but we ran into issues with timing and reverted to version three.

The first two versions of the code had an issue where the period of the compliment signal was not aligned with the period of the main signal. This resulted in what appeared to be a half-period phase shift, which is not ideal, as two MOSFETs cannot be on at the same time. Version four of the code simply did not

work the way it was intended. This was more than likely due to a lack of understanding.

While testing the circuit in its entirety, the team ran into issues with outputting 4 PWM signals at a high enough frequency. This involves direct manipulation of the timer registers in the Arduino; coding the Arduino Micro for this specific purpose proved to be difficult. Instead of attempting to accomplish this, the team opted for an easier approach, which was using an alternative Arduino. In this case, an Arduino Uno R3. With the exchange of the Micro to the Uno, coding became much easier and allowed for the output of 4 PWM signals. 2 pins here were set at a frequency of 62khz (bidirectional DC-DC) and the other 2 were set at 31khz (noninverting DC-DC). The specific code is shown in Figure 10.

```
#define s1_pin 5  // Pin for S1 (High)
#define s2_pin 6  // Pin for S2 (LOW)
#define s3_pin 9  // Pin for S3 (connected to buck mode in the diagram) HIGH
#define s4_pin 10  // Pin for S4 (connected to boost mode in the diagram) LOW

void setup() {
  // Set fast frequency for 62khz
  TCCR0B = TCCR0B & B11111000 | B00000001;
  TCCR1B = TCCR1B & B11111000 | B00000001;

  // Initialize both pins to low
  analogWrite(s1_pin, 0);
  analogWrite(s2_pin, 0);
  analogWrite(s3_pin, 0);
  analogWrite(s4_pin, 0);

  // Set PWM pins as outputs
  pinMode(s1_pin, OUTPUT);
  pinMode(s2_pin, OUTPUT);
  pinMode(s3_pin, OUTPUT);
  pinMode(s4_pin, OUTPUT);
}

void loop() {
  // Boost mode: S1 closed, S2 switching
  analogWrite(s1_pin, 255);   // S1 closed
  analogWrite(s2_pin, 128);     // S2 switching at 50% duty cycle

  // Boost mode: S3 open, S4 switching (power runs from right to left)
  analogWrite(s3_pin, 0);   // S3 open
  analogWrite(s4_pin, 128);     // S4 switching
}
```

Figure 10. Final Arduino Uno Code

C. PCB Prototype

The last thing that we started work on was the PCB. We intended to have a fully working PCB design that was ready to be milled out by the time breadboarding and programming were completed, however, we could not decide on a design that fit our purposes (especially considering the time loss from changing the schematic) and the PCB was not completed.

To make the PCB, Devin started by recreating the electrical schematic in KiCad, a PCB design software similar to the likes of Autodesk EAGLE or NI Ultiboard. Devin also added in some extra components that were not included with the simulation files and made sure that all the devices that were placed had parity with the Multisim counterpart.

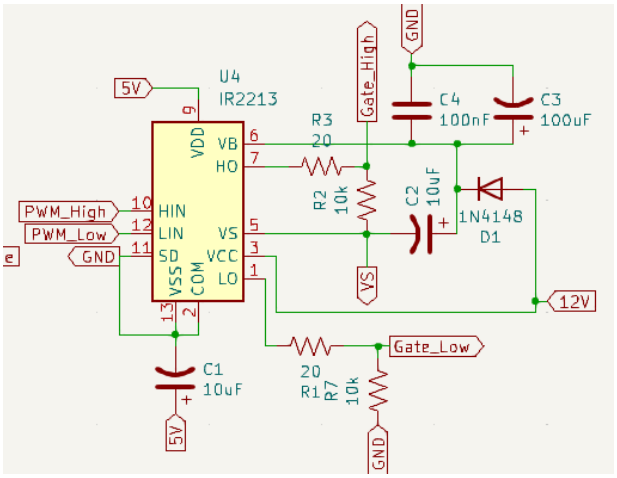

Figure 11. One of two gate drivers recreated within the KiCad schematic file.

Once this was done, the parts then had to have their footprints assigned so that KiCad knew what to show when the PCB layout was being created. For the more specialized devices, like the Arduino Micro (Figure 8), their footprint was already assigned.

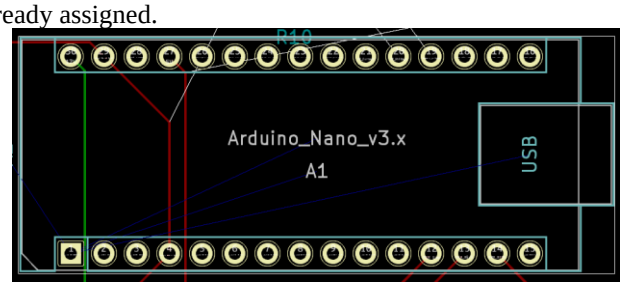

Figure 12. The footprint of the Arduino Micro. The Arduino Nano and the Arduino Micro have the same footprint.

There is, however, ambiguity for more general components, like resistors, capacitors, and inductors, that need to be measured and given their footprints appropriately. These components don't have a proper dimensional measurement on their datasheets, simply because they are so generic, so Devin had to measure their dimensions with a set of calipers and make a rough comparison with the provided footprints in KiCad to find which ones could work properly. Once all the required devices were placed into the schematic file and given their footprints, all the devices could then be directly imported into the PCB file and placed. From here, all the parts can then be moved freely around the sheet. Note the white lines that connect all the components within KiCad. These lines make it easier to see what devices connect where and are accurate to the connections made in the schematic.

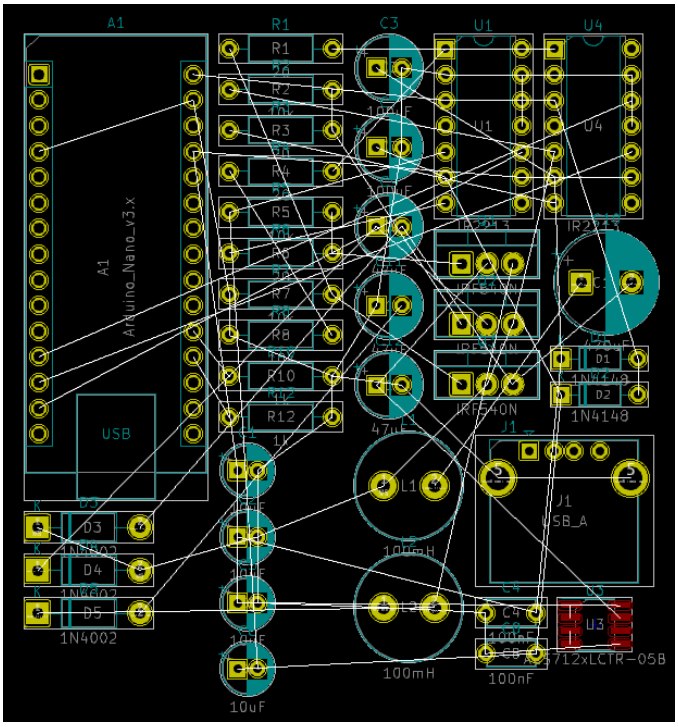

Figure 13. The whole circuit placed in the PCB design suite.

The PCB is intended to be a standard two-layer PCB that will have the devices loosely grouped by what they are: gate drivers, DC-DC converters, and the Arduino, along with the sensors and the outside connections for the battery, solar panel, and our load. With that in mind, we can follow a basic rule for laying out traces that helps to prevent errant trace paths: the upper layer traces (in red) will mostly travel vertically, and the lower layer traces (in green) will travel horizontally.

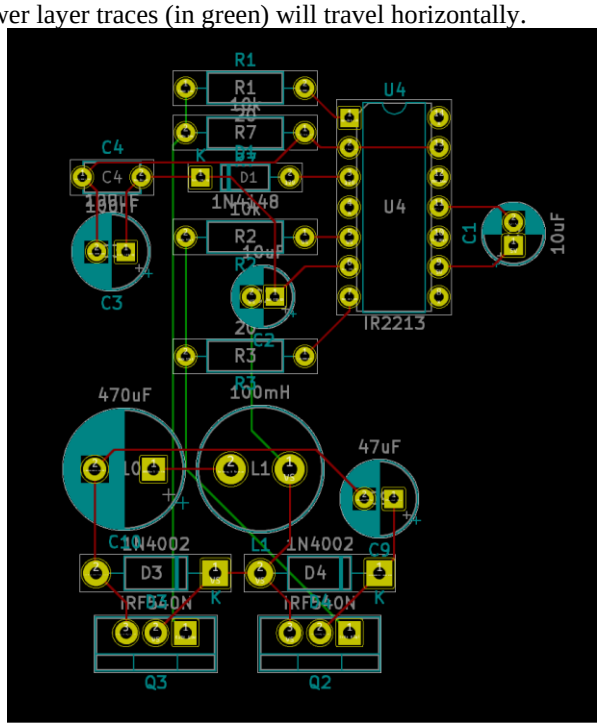

Figure 14. Proposed PCB layout for the Bidirectional DC-DC converter, along with its associated gate driver.

Following the issues we had with programming the Arduino Micro, the schematic had to be changed to allow for the Arduino Uno R3 that took its place. The schematic also had to be changed to include the non-inverting DC-DC converter that replaced the buck-boost converter.

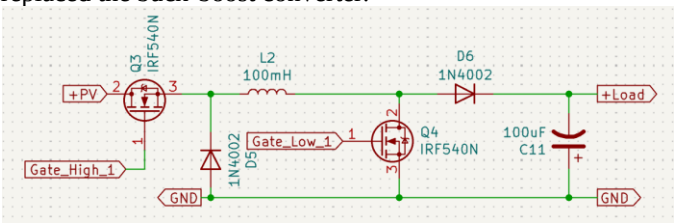

Figure 15. The schematic for the non-inverting buck-boost converter

Part of the problem with making the schematic that we faced was readability. To ensure that portions schematic were readable, several tools in KiCad were used to refine and condense the various nets within the schematic:

1. The Electrical Rules Check (ERC)
2. Highlighting the wires and pins of a net
3. The Symbol Editor

The ERC shows us what errors, if any, there are in the schematic. It picks up issues such as unconnected pins, power delivery issues, misconnected pins, and so on. This was used to make sure that all the connections were properly made and helps with resolving issues that came up with custom symbols in the symbol editor. Highlighting the wires and pins of a net (which can be selected as a tool in the right-hand sidebar, it does not appear to have a true name) allows the user to condense their global labels so that reading the schematic is not difficult. The Symbol Editor was used to create custom symbols for the Gate Driver IC and the ACS712 module to enhance readability.

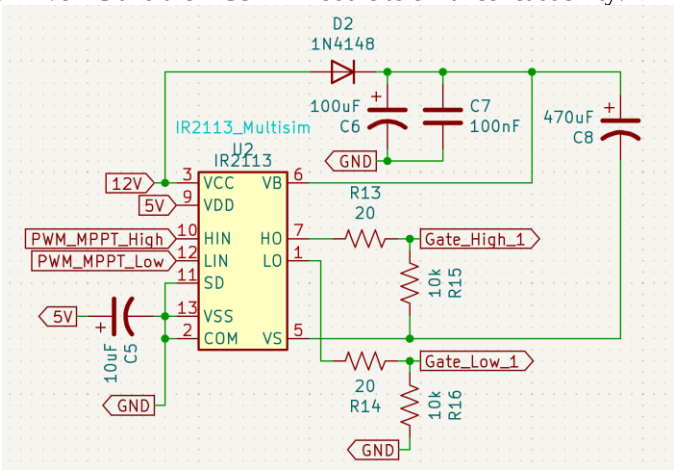

Figure 16. One of the Gate Drivers with a custom symbol. Meant to represent the Gate Driver circuit as seen in Figure 2.

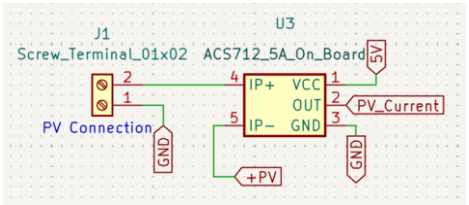

Figure 17. The custom symbol of the ACS712. Is simply a graphical representation of the modules that we purchased, which contain other parts.

We eventually decided that the PCB design we had was not sufficient and opted to change it to better fit the maximum five-by-four-inch board that we could mill. Devin realized that the traces were not large enough to support the power that would be moving through the PCB, so they were changed to one millimeter track width, with the traces that carry the high frequency PWM signal having 0.5-millimeter track width.

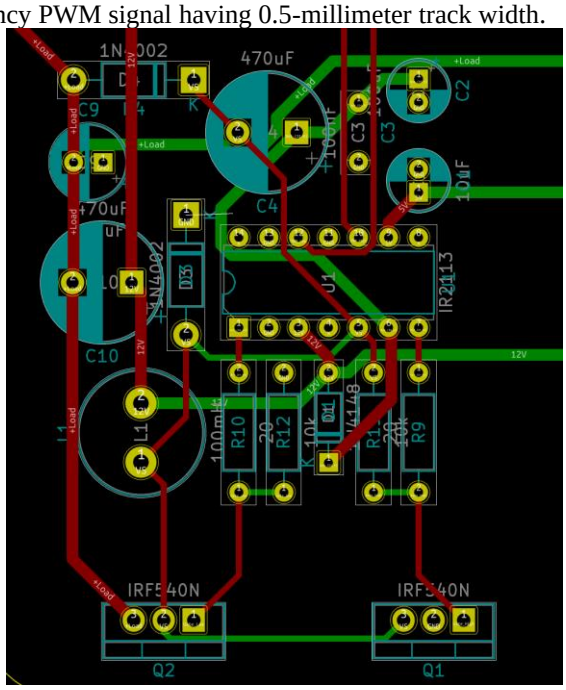

Figure 18. A newly proposed PCB layout for the Bidirectional DC-DC converter, along with its associated gate driver.

The new design of the PCB, partially pictured above in Figure 11, includes similar design choices to the original design. It, however, has been modified to take up less empty space on the PCB. The two MOSFETs have been placed a distance apart to keep the high frequency PWM lines at an even distance, which is intended to function as a form of differential pair. An image of the new design has been included in the Appendix section of this report.

III. Results

The first results of importance were the values of our input and output voltages from the PV arrays' DC-DC converter. The table for the given values can be shown in Table 1.

Table 1. Classic DC-DC converter testing

Trials	Input Voltage (V)	PWM Signal (%)	Output Voltage (V)
Trial 1	5V	25%	1.9
Trial 2	5V	50%	2.6
Trial 3	5V	75%	2.9

Table 1 demonstrates the results of multiple testing trials for the DC-DC converter. It should be mentioned that values were tested at these PWM values because of how these converters operate; to operate under buck mode the duty cycle will need to be < 50%, and to activate boost mode the duty cycle will need to be > 50%. Therefore, the three values 25, 50, and 75 percent will give a clear indication of whether the buck and boost modes are working as desired. It can be seen in these results that the buck mode of the converter is working, but that the boost mode is not. These values are not the values that our group intended to achieve; because of this, a separate set of trials was set up for further testing. These results can be seen in Table 2.

Table 2. Further classic DC-DC converter testing

Trials	Input Voltage (V)	PWM Signal (%)	Output Voltage (V)
Trial 1	12V	25%	4.14
Trial 2	12V	50%	6.13
Trial 3	12V	75%	7.13

Table 1 provides additional results when the input voltage is changed. The results shown from both Tables 1 and 2 depict a working buck converter instead of a buck-boost converter. Subsequent analysis revealed that the root cause of this discrepancy was the suboptimal frequency of the PWM signal generated by the Arduino, which was measured at a mere 490Hz—significantly lower than the target frequency of approximately 20kHz necessary for optimal operation. This discrepancy necessitates additional experimentation to achieve the desired frequency and converter functionality.

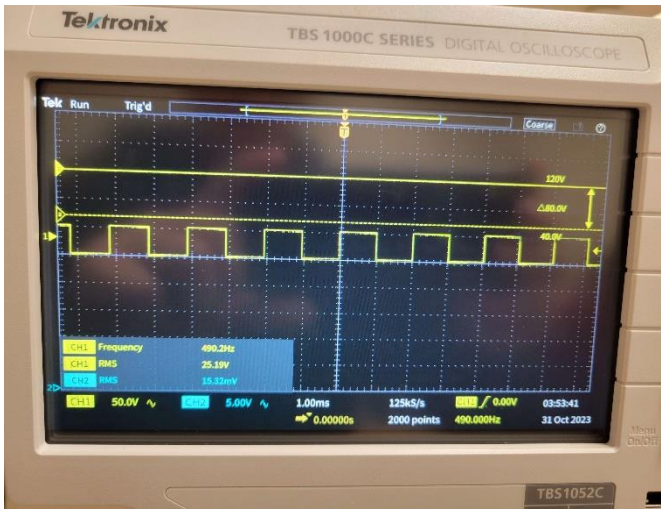

Figure 19. PWM Signal at 490Hz

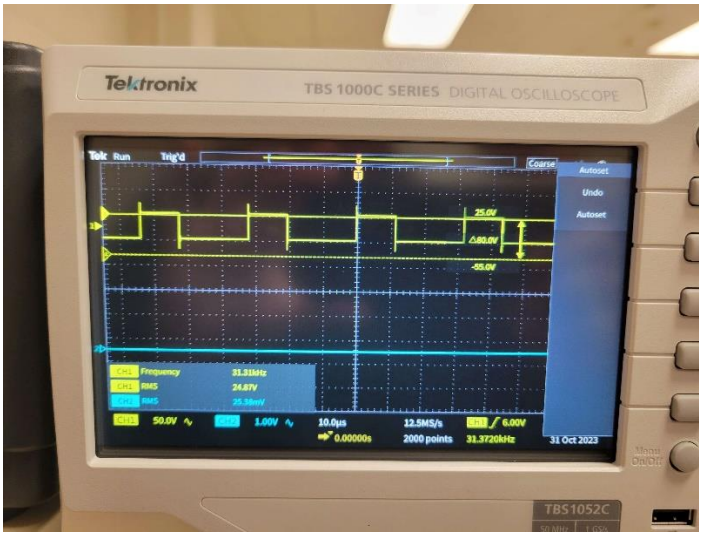

Figure 20. PWM Signal at 31kHz

Figure 11 shows a renewed PWM signal with the desired frequency we need for our circuit. Further results, including the load voltage with the renewed PWM signal and different inductor values, have not yet been tested. In addition, the bi-directional DC-DC has also not been tested. Therefore, there are no additional results to be displayed until further testing can be conducted with these new variables.

After the swap from the conventional DC-DC buck boost converter to a non-inverting DC-DC converter, the circuit was tested with a standard set of parameters. An input of 6V and 1A was kept constant through the tests as this is a simulation of the input from the PV array. The voltage measured across the load is referenced off a 1k ohm resistor. The table of values is shown below in table 3.

Table 3. non-inverted DC-DC converter boost mode

V_{in}	Duty Cycle	V_R
6V	25%	6.8V
6V	50%	10.1V
6V	75%	16.9V

Table 3 shows the boost mode results for the non-inverting DC-DC. In boost mode, this configuration is operated by turning switch 1 fully on and by switching switch 2. The results given are effective results and will be talked about briefly more in the discussion tab. Table 4 reveals the results obtained by the circuit while operation under buck mode.

Table 4. non-inverted DC-DC converter buck mode

V_{in}	Duty Cycle	V_R
6V	25%	0.51V
6V	50%	1.78V
6V	75%	3.1V

With the tables provided above, it is shown that the boost and buck operations of the circuit operate at the voltage levels at which they are intended to. The next tables show the results of the bidirectional DC-DC converter when operating in both boost and buck modes. The constant values set for these tests are 5V input for the boost mode and 12V input for buck mode as these are the inputs we will likely be using. The resistance we will be using as our load is going to be 10k ohms.

Table 5. bidirectional DC-DC converter boost mode

V_{in}	Duty Cycle	V_R
5V	25%	7.07V
5V	50%	10.40V
5V	75%	17.54V

Table 6. bidirectional DC-DC converter buck mode

V_{in}	Duty Cycle	V_R
12V	25%	1.76V
12V	50%	3.91V
12V	75%	6.04V

IV. Discussion of Results

As of current, we have conducted 6 separate trials with our circuit. For the first 3 trials, the circuit received 5V with the only varying parameter being the pulse width signal. The first trial begins with a 25% signal and is increased by 25% for each following trial. The resulting output voltage is recorded in the above table. The remaining 3 trials were tested in the same fashion, except the input was raised to 12V. We see from the table that each increase in pulse width increases the output voltage, effectively showcasing a working buck converter. As noted previously in "Results", all six trials were tested under the old Arduino code which would output a PWM signal at 490Hz instead of 31kHz. Although we have done some testing with the updated frequency, we still do not have clear results that give us an indication of a better-functioning circuit. There will still need to be more testing of both the updated PWM signal and varying

inductor values. The results from these specific tests are presumed to be accomplished by the end of this week.

It is notable to mention that the values from our DC-to-DC converter are not the values we want. The DC-DC converter that our group aimed to create was a buck-boost converter, however after discussing our results, what we had achieved was a functional buck converter. Although this was not what we had intended, our team agreed that a buck converter could still be used to accomplish the goal we had set. The power input from the PV array is expected to reach 100W under ideal conditions; this is more than enough to supply both our load and battery with power. Because of this, a buck converter can be placed in conjunction with our PV array and always buck our output to 5V, therefore charging the phone (load).

During the procedure it was mentioned that the group swapped the conventional DC-DC converter with a non-inverting DC-DC converter; while reviewing the topology of the circuit, it seemed that we would need two positive nodes in parallel with each other so that power can flow smoothly. This brought some complexities because of the introduced MOSFET; however, it is more efficient and that can be seen in both buck and boost tables located in the results section.

V. Conclusions

In conclusion, our group has made substantial progress on the PV-Battery Integrated Energy Source project. Thus far, we have assigned roles, ordered necessary components, developed Arduino code to produce the required PWM signals, and begun prototyping the bidirectional DC-DC converter circuit. While debugging the Arduino code presented some challenges, we now have a working version that properly generates the PWM signals to control the converter's MOSFETs. Our initial testing of the DC-DC converter verifies that it can operate in both buck and boost modes as intended. However, the output power levels do not yet match the target specifications. Our next steps will be to adjust the inductor and capacitor values in the converter circuit to achieve the desired voltage and current outputs. We will also finalize and test the overall system integration, including the PV array, battery storage, and load. Pending successful results from the breadboard prototype, we will then design the PCB and solder the components to create the final integrated device. With the majority of tasks completed in these initial phases, we are on track to meet the project timeline. Through continued teamwork, troubleshooting, and iterative improvements, we are confident the product will meet all requirements. This project has already provided useful hands-on experience in integrating electrical components, programming microcontrollers, circuit simulation, and prototyping techniques.

VI. References

[1] Dileep. G and S. N. Singh, “Selection of non-isolated DC-DC converters for solar photovoltaic system,” *Renewable and Sustainable Energy Reviews*, vol. 76, pp. 1230–1247, 2017. doi:10.1016/j.rser.2017.03.130

VII. Appendix

Parts List 1				
Part Name	Part #	Quantity	Unit Price	Total Price
MOSFET (10pcs)	LRF540	1	$7.99	$7.99
Diode	863-IN4002G	10	$0.19	$1.90
Inductor 100mH	580-19R107C	10	$1.47	$14.70
Capacitor 47 µF	80-ESW476M035AE3AA	10	$0.27	$2.70
Capacitor 470 µF	80-ESK477M035AH2EA	10	$0.50	$5.00
Voltage Regulator 5V	511-L7805CV	5	$0.69	$3.45
USB 2.0 Port	530-SS-52100-001	5	$0.56	$2.80
Gate Driver	942-IR2110PBF	2	$2.92	$5.84
Arduino Micro	A000053	1	$23.95	$23.95
Breadboard (10pcs)	pcb5	1	$19.99	$19.99
ACS712 Current Sensor Module (5pcs)	TS-US-158-CA	1	$9.49	$9.49
Overture PETG Filament 1.75mm Grey, 1kg	OVPETG175	1	$21.99	$21.99
Arduino Uno REV3	A000066	1	$27.60	$27.60
			Gross Cost	$156.16
			Net Cost	$169.04
			Remaining Budget	$60.96

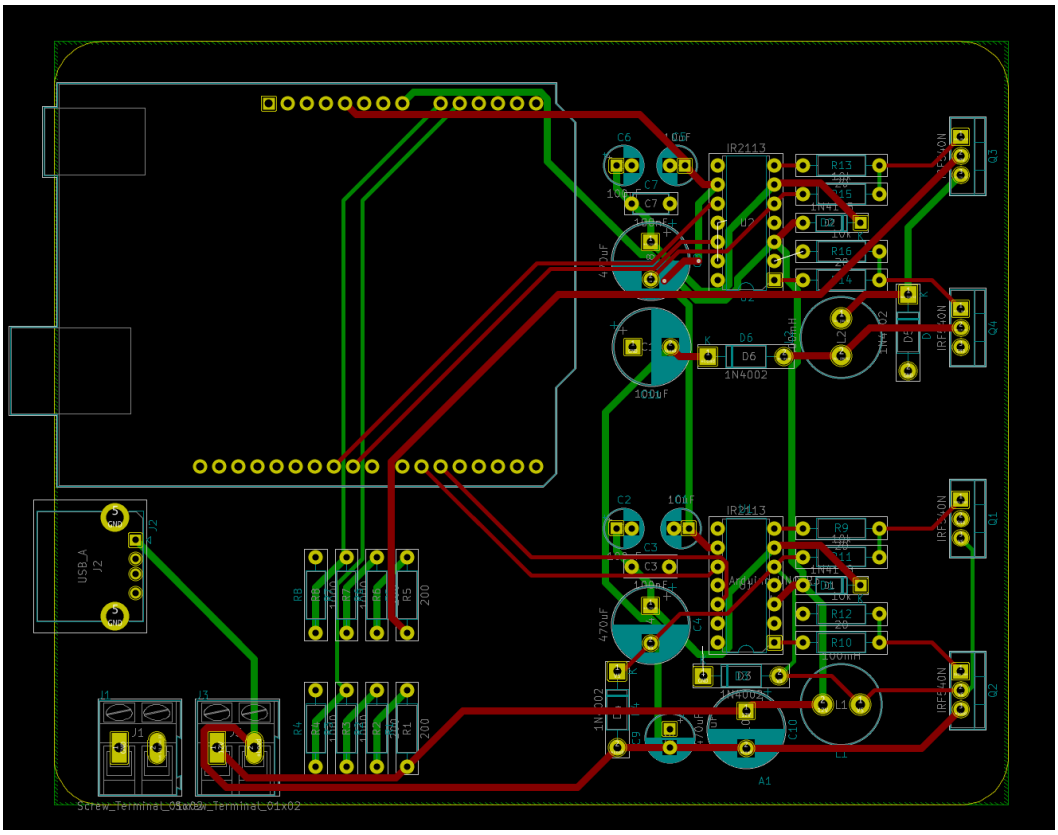

Figure 21. The new PCB Design. Note that the ACS712 current sensor is not on the board yet.