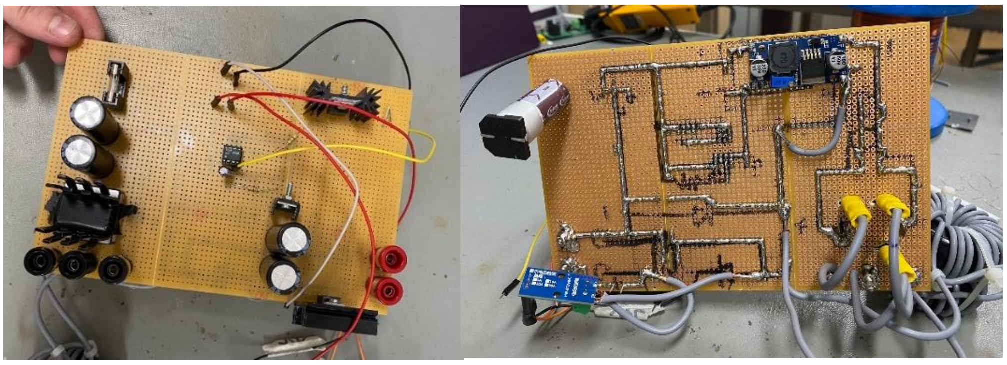
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

EE463 Term Project: Wind Turbine Battery Charger Final Report SoC



Selen Özge ÖZGÜR - 2375566

Onat ŞİMŞEK - 2375772

Canberk KAÇAN -2443240

Contents

Introduction

Converters for charging and discharging a battery have been used tremendously for years now. In this project, we have been assigned to design and implement a power electronics circuitry for charging a battery from a three-phase supply. The three-phase supply has a variable line-to-line voltage since it represents a wind turbine. In this report, you can find the design steps and design decisions, information about the chosen converter topology, simulations, real-life results and their comparison with the simulation results, controller and lastly BOM, bill of materials, will be given.

Specifications

Most of the wind turbines do not generate a constant line-to-line voltage. The voltage level varies with the change in the wind speed and wind direction. To represent this phenomenon, we were asked to make our designs for a varying line-to-line input voltage. You can see the design specifications below:

- Input voltage: 15 $V_{\text{line-to-line}}$ to 25 $V_{\text{line-to-line}}$
- Battery Capacity: 100Ah
- Battery nominal voltage: 12V
- Output current: 10A
- Output current ripple: 20%

Topology Selection

To do AC/DC conversion, we first do the AC/DC then DC/DC conversion. In this part topologies that we used will be introduced.

Three-Phase Full Wave Diode Rectifier

For AC/DC part, we selected three phase full wave diode rectifier topology for our design. The circuit topology can be seen in Figure 1. Main reason of choosing this topology is it give smoother output when we compared to half wave topologies and easier to control when we compared to thyristor circuitries. We accept output of this circuit is a DC with small ripples and implement it to the main design.

$$V_{dc} = 1.35 * V_{l-l}$$

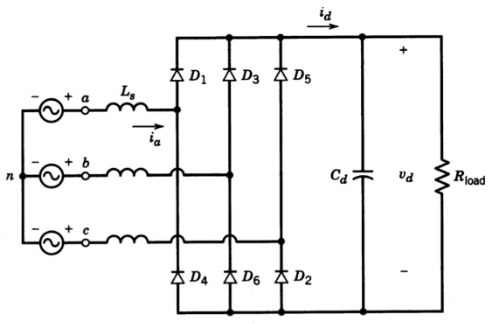


Figure 1: Three phase full wave diode rectifier

Buck Converter

Output voltge of the AC/DC converter is higher than desired DC output so that we need to lower the voltage by using DC/DC converter. In order to do that we used Buck Converter topology becaouse the topology is easy to implment and efficient. Topology can be seen in Figure 2.

$$V_o = D * V_d$$

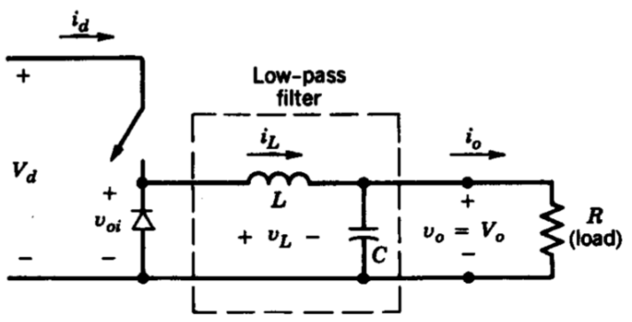


Figure 2: Buck Converter

Simulation Results

In this part of the report, the waveforms of the important components are obtained using the Simulink, which was also used for the simulation report previously. We modeled the wind turbine, or the Variac in our case, by using the AC signal sources and we arranged them so that we obtained a balance 3-phase input. Note that the presented results are obtained in a closed-loop manner, by implementing a PI controller in the Simulink environment. Simulink simulation is given in figure 3. The K_P and K_I parameters of the PI controller are obtained using the built-in “pidtune” function in MATLAB and the obtained parameters are manipulated so that the current overshoot in the transient is eliminated.

Also note that these results are obtained when the AC input is 25 $V_{l\text{-}l,\ RMS}$. Thus, in the real-life test results, since we do not increase the AC input voltage directly to 25 $V_{l\text{-}l,\ RMS}$, the observed overshoots in the transient do not actually occur. Our components are also selected in this manner. Simulation results are given in Appendix A.

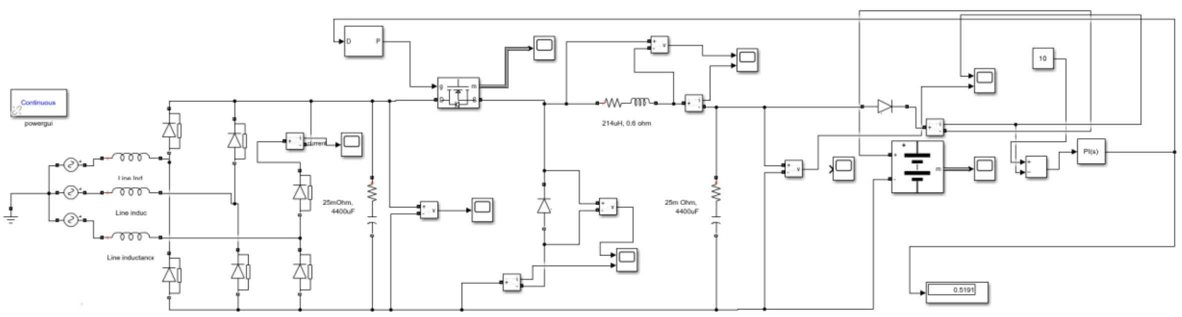


Figure 3: Simulink model for the simulation

Component Selection

We selected our components according to the simulation results. Moreover, we have also presented many of the components in the simulation report as well. However, after taking some feedbacks, some of our components are changed accordingly.

- **Diode Rectifier Selection**: Since our input is three-phase AC voltage, we decided to implement a three-phase diode bridge rectifier. Even though most of the three-phase diode bridge rectifiers have higher ratings than our operating ratings, we preferred to implement a three-phase diode bridge rectifier to have a more compact design. Thus, we chose VS-36MT160 a three-phase diode bridge rectifier as our rectifier. The datasheet of the rectifier can be found at the Appendix B section of the report.
- **Capacitor Selection**: After rectifying the input AC voltage, since we still observed voltage peaks of the sinusoids, we connected two large electrolytic capacitors parallel to the rectifier to store more energy in the capacitors. Since our maximum DC voltage at the output of the rectifier is about 35V, we selected a capacitor with a voltage rating of 63V to have some safety margin. The capacitance value of the selected capacitors is 2200µF. As a result, we have a capacitor with 4.4mF capacitance value at the output of the rectifier. Furthermore, by paralleling two capacitors, we have not only achieved a higher capacitance, but we also achieved a lower ESR rating. Moreover, we have used the same capacitors at the output stage of the buck converter to maintain a low voltage ripple at the output and also for filtering purposes.
- **MOSFET Selection**: According to our simulation results, we selected our MOSFET as IRF540, which has a relatively low $R_{DS,on}$ value with a V_{DS} value which allows us to have enough safety margin. The datasheet of the MOSFET can be found at the Appendix B section of the report.
- **Diode Selection**: We have also selected a diode according to our simulation results. Since our rated current is 10A and voltage rating is about 35V, we selected our diodes as DSS 16-0045A, which has a maximum reverse repetitive voltage rating of 45V, and average forward current rating of 16A. This diode has been used both for the buck converter stage and for placing before the battery to prevent the inrush current flow from the battery. The datasheet of the diode can be found at the Appendix B section of the report.
- **Controller Selection**: In order to control the output current more easily, we decided to implement a digital control method. For this purpose, we used “Arduino Uno” as our controller. We supplied PWM signal to the gate driver, since the output current rating of the Arduino Uno is not sufficient to turn the MOSFET on, using one of the PWM pins of the Arduino Uno.
- **Current Sensor Selection**: Since we need to supply a constant current of 10A to the battery, and since we implemented a digital controller using Arduino Uno, we decided to use a current sensor which is Arduino compatible. We selected our current sensor as ACS712ELC-30A, which is capable of measuring the current up to 30A. We chose the model which can measure 30A due to the stock problems by being aware that this is an overdesign.
- **Gate Driver Selection**: Since we have implemented a digital controller, we have also selected a gate driver to increase the voltage and current ratings of the Arduino and to drive the MOSFET in a better way, by applying a higher voltage than the threshold voltage of the MOSFET between the gate and the source of MOSFET. Moreover, since the buck converter can be designed by both implementing a high-side and low-side driver circuits, we have chosen a gate driver with separate low and high side logic inputs. Therefore, we ended up selecting the gate driver IR2106. According to the datasheet of IR2106, we have also selected a capacitor. Furthermore, by considering the gate charge and gate turn-on characteristics of the MOSFET,

we have selected a bootstrap capacitor, bootstrap diode, and a gate resistor. The datasheet of the gate driver can be found at the Appendix B section of the report.

- **Gate Driver Capacitor Selection**: As a decoupling capacitor of the gate driver IC, we have selected a 10µF electrolytic capacitor to filter out the noise and remove any power distortion.
- **Gate Driver Resistance Selection**: In order to supply high enough current and also limit the current going to the gate of the MOSFET, we have connected a series resistor with a resistance of 4.7Ω. We did not chose a higher resistance to turn the MOSFET on faster in order to decrease the switching losses.

- **Voltage Supply Selection for Lower Voltages**: In order to supply power to Arduino Uno, to the gate driver, and to the fan, we need another voltage regulator. For that purpose, we have used another buck converter named "LM2596", which is produced by Texas Instruments. LM2596 has an input voltage range of 4-35V and output voltage range of 1-30V. Thus, to supply power to Arduino Uno, to the gate driver, and to the fan, we selected LM2596 buck converter.

Design

After deciding on the overall design, we evaluated additional features for a bonus. These are:

- Compactness Bonus
- Analog Controller Bonus
- PCB Bonus
- Closed Loop Current Controller Bonus
- Reverse Current Protection
- Over Current Protection

First, we tried to implement analog controller, but we realized that this would take much more time than digital controller so that we decided to start with digital controller then try analog controller if we have time when we successfully implement the other parts. Same condition is valid for PCB design. Unfortunately, we couldn't complete the task before the demo day, so we had to give up these plans.

We also tried to drive MOSFET in high side at the beginning of the design. We implemented high side MOSFET driver circuit and showed it in demo day, then we changed it after.

Diode Rectifier Design

Our initial plan was to connect two bridge rectifiers according to our design consideration but when we check components in the lab, we found VS-36MT160 which gets three phase AC input and convert it to the DC which fully meetings our design considerations. This component helped us to save space but placing the component to the stripboard was a bit problematic. We cut the stripboard in a wrong way in our first trial then we developed a method to place it with minimum damage to the stripboard by using sharp screw. In Figure 3, upper stripboard was in the same size with below one but due to unsuccessful cuts, we broke the board and cut it to reuse it. Furthermore, you can see correctly placed rectifier between inputs from the variac (yellow cable) and capacitors from the Figure 3.

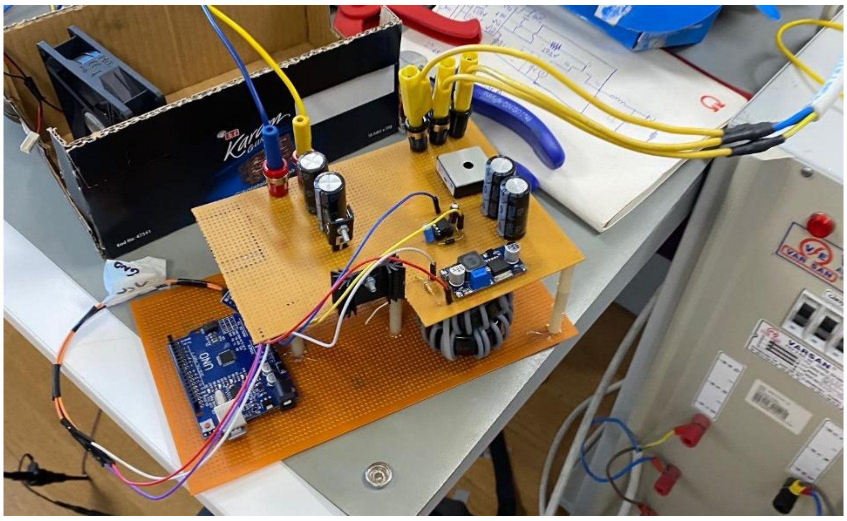


Figure 4: Reused Stripboard

Arduino, Fan and MOSFET Driver Supply Circuitry

Our design must work with single input source which is the variac; however, the output of the variac is too high to supply Arduino, fan and MOSFET driver. To supply smaller DC voltage, we used LM2596 Adjustable Voltage Step Down Power Module. The module can be seen in Figure 4.

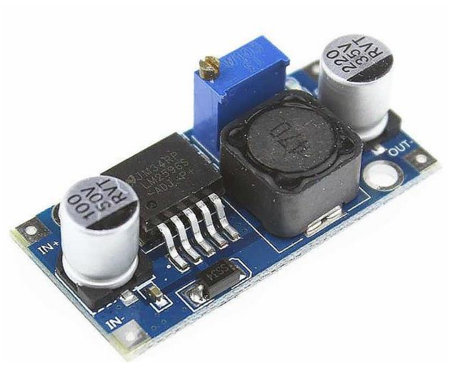


Figure 5: LM2596

We connect the rectifier output to this power module and arranged it as to get 10V output. This 10 V is used to supply power to MOSFET driver, fan, and Arduino.

Gate Driver Circuit

In order to turn the MOSFET on, as mentioned previously, we used Arduino Uno as our digital controller. Moreover, to supply the high enough voltage between the gate-source terminals and current to the gate of the MOSFET, we used a gate driver. Thus, we implemented a gate driver circuit for this purpose.

High-Side Gate Driver Circuit

First, in order to drive the MOSFET, we utilized a high-side switching circuit using our gate driver, IR2106. Furthermore, we know that we need to supply a high enough voltage, which has to be higher than the threshold voltage of the MOSFET specified in the datasheet of our MOSFET, between the gate-source pins of the MOSFET.

In the high side of the circuit, our MOSFET is placed between the node that our inductor and buck diode is connected and the node where we have the rectifier output voltage. In Figure 6, the buck converter in which a high-side switching operation is utilized is shown.

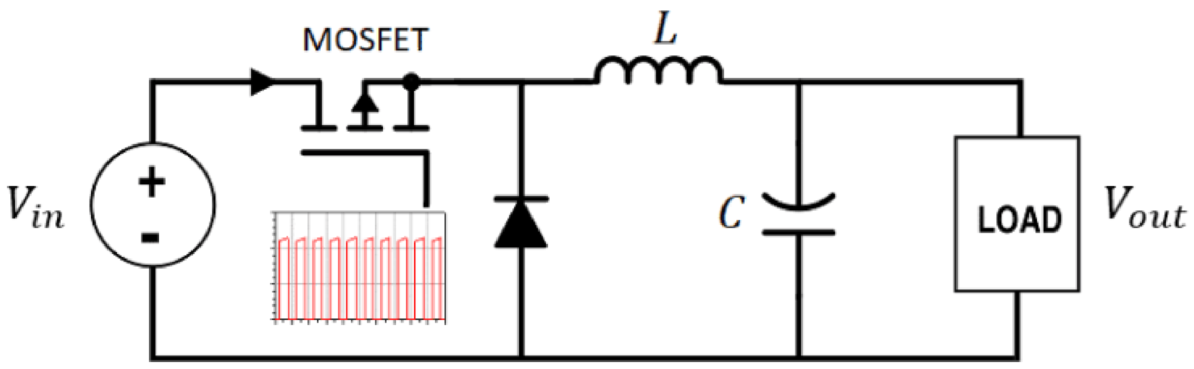


Figure 6: Buck converter with high-side switching operation

In this configuration, since the voltage should be applied between the gate-source terminals (V_{GS}) of the MOSFET to turn it on and since the rest of the buck converter components are connected to the source of the MOSFET, we need to supply a gate voltage higher than the voltage at the source of the MOSFET. To accomplish this, we constructed a bootstrap circuit.

Bootstrap Circuit Operation

Normally, the bootstrap circuit is utilized for half-bridge applications. Moreover, by constructing a half-bridge circuit, one can utilize synchronous buck converter as well, but we did not prefer this methodology since it is really hard to control two MOSFET with a necessity of a dead time implementation, which really hard to implement with a digital controller.

To construct a bootstrap circuit, a capacitor and a diode is sufficient most of the time. The most important considerations are the voltage rating of both the diode and the capacitor. The current flowing through the bootstrap circuit is not too high but the current rating of the bootstrap diode should also be checked. In addition, the voltage rating of the capacitor should be high since it will charge up to a value which is double of the supply voltage of the gate driver circuit. Finally, the selected capacitor should have a high current rating since it will charge and discharge continuously.

The capacitance value of the bootstrap capacitor is calculated using the following formula:

$$C_G = \frac{Q_G}{V_{Q1_G}} \; \& \; V_{Q1_G} = V_{DD} - V_{BootDiode}$$

Where Q_G is the total gate charge of the MOSFET, which can be found in the datasheet. And, $V_{BootDiode}$ is the forward voltage drop due to the bootstrap diode. Finally, once the gate charge is determined, the minimum value of the bootstrap capacitor can be estimated by:

$$C_{bootstrap} \geq 10 * C_G$$

Q_G of our MOSFET is given as 72nC in its datasheet. Also, we had selected 1N4007 as the bootstrap diode. Then, by assuming our input voltage to the gate driver IC is 10V, which can be adjusted and fixed since we have used a buck converter, C_G can be calculated as 8nF. Then, our minimum capacitance for the bootstrap capacitor is 80nF. Thus, an electrolytic capacitor with 0.1µF capacitance and having a voltage rating of 35V is selected as bootstrap capacitor.

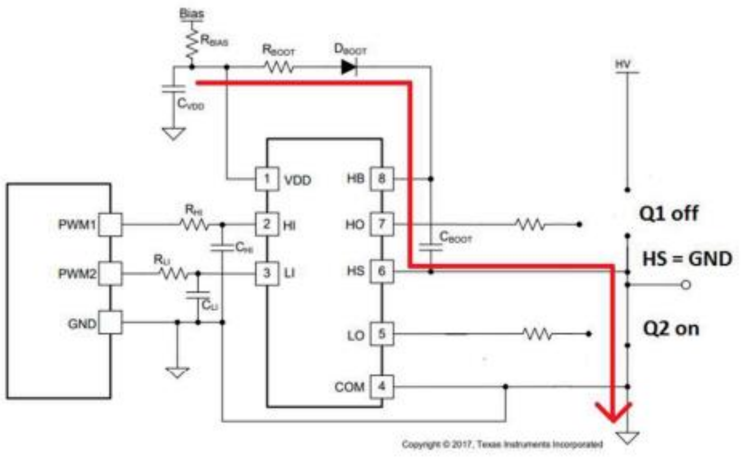


Figure 7: Bootstrap capacitor charging path in a half-bridge application

In Figure 7, the charging path of the bootstrap circuit in a half bridge application is depicted. When the high-side switch is off and the low-side switch is on, the capacitor charges using the low-side switch. However, since we have a reverse diode instead of a low-side switch in our buck converter, this path is not valid for our application.

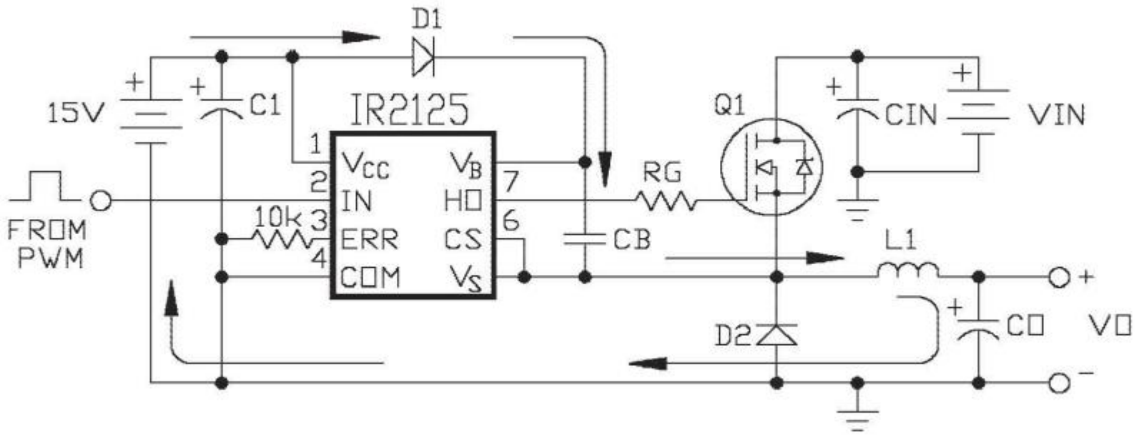


Figure 8: Bootstrap charging path with a buck converter is connected

In Figure 8, exactly the same application as the one we have built is shown. In this figure, the bootstrap capacitor is charged using the shown path, and output capacitor of the buck converter is also charged when bootstrap capacitor is charged. In the next cycle, when the switch is ON, the bootstrap capacitor discharges and the voltage is supplied to the gate-source terminals of the high-side MOSFET. Figure 9 shows the discharging path of the bootstrap capacitor.

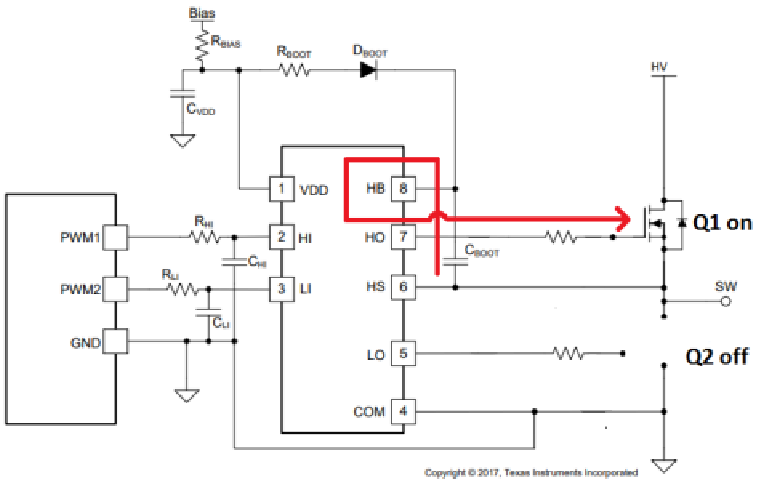


Figure 9: Bootstrap capacitor discharging path

After facing the error that the bootstrap capacitor is not charging, which later we found out that it was due to the PWM pin of the Arduino Uno, we constructed a low-side switching circuit to drive the MOSFET.

Low-Side Gate Driver Circuit

When the MOSFET is driven in the low side, since the source pin of the MOSFET is connected to the – pin of the rectifier output, which our common neutral line, the need to construct a bootstrap circuit is eliminated. Thus, it is sufficient to supply the PWM signal generated by the Arduino to the low-side input pin of the gate driver IC and having the gate drive signal from the low-side output pin of the IR2106. Figure 10 depicts an example of a buck converter design with a low-side MOSFET switching.

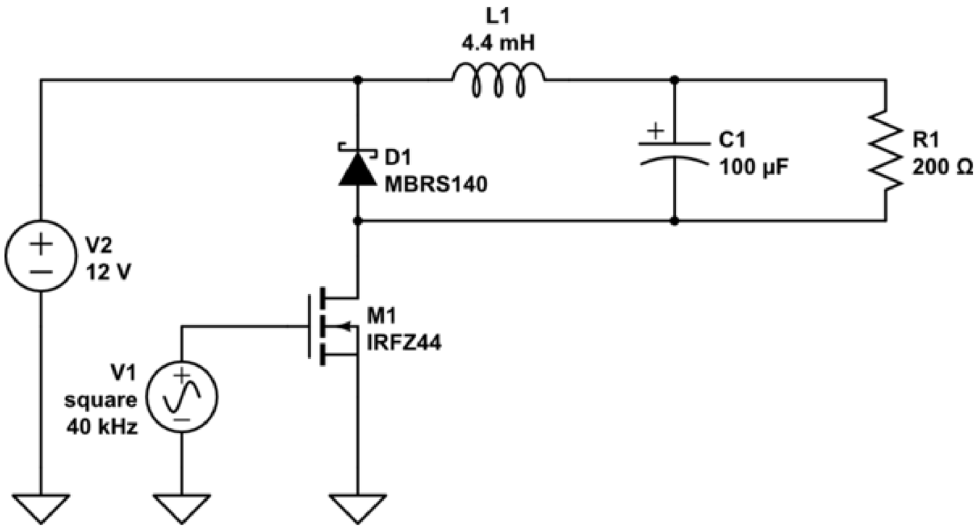


Figure 10: An example of the low side switching application for buck converter

This circuit operates exactly in the same way as when the MOSFET is at the high side. As a result, we used this kind of circuit configuration in our final design.

Inductor Design

For the inductor, our specifications were approximately 210µH with a 10A continuous operation current rating. After excessive market research, we have realized that finding an inductor with such specifications was nearly impossible. There were some options, such as AGP4233-224ME, but either they were out of stock or impossible to get in the necessary time period. Moreover, the in-stock options were pricey. Thus, we have decided to design and wind our own inductor. First, we have checked the magnetic cores in the laboratory to see if there were any cores that were compatible with our design. There were three cores that stood out: 0077111A7, 0075192A7 and 0077442A7. 0077111A7's A_L range was too small for the design, 0077442A7's performance was below 60%for the needed operation parameters, so we have settled to use 0075192A7. In Figure 11, you can see the typical DC bias performance of the chosen core.

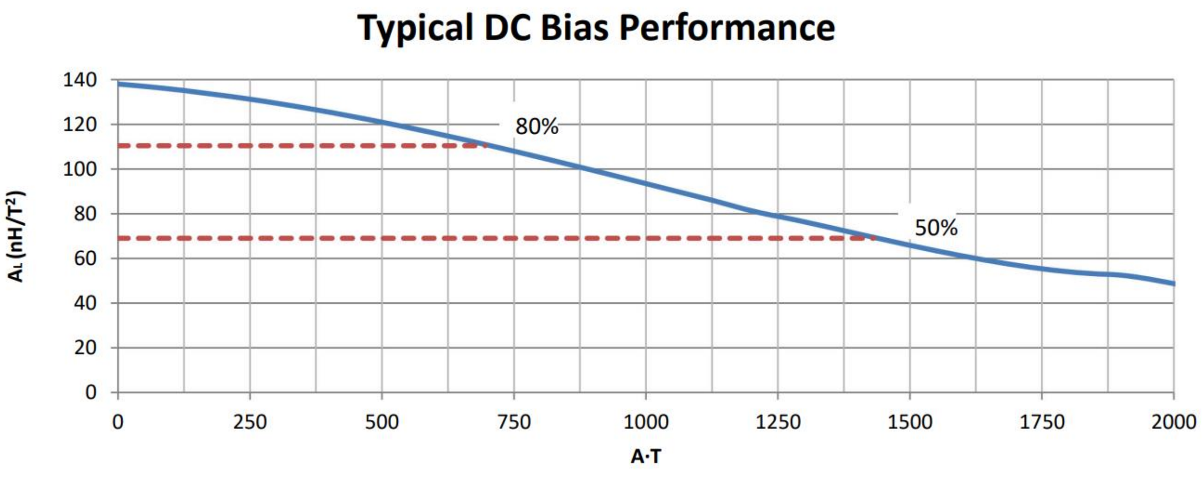


Figure 11: Typical DC performance of the core 0075192A7 [z]

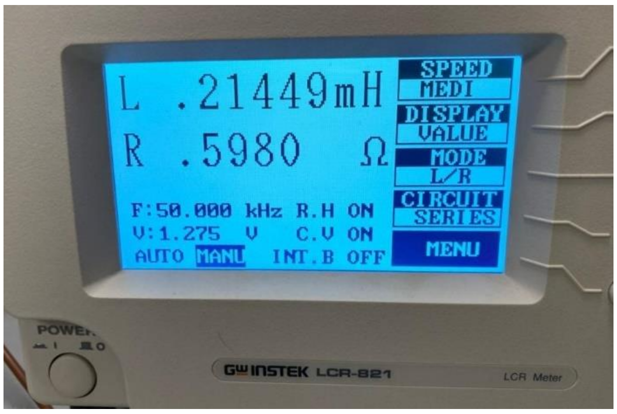
In this performance graph, the x-axis shows the A.T values which are equal to N.I, current multiplied by the number of turns. Moreover, the y-axis shows the A_L values. From a quick literature review and discussion with our classmates, we have learned the following equation holds for the A_L.

$$A_L = \frac{L}{N^2} = \frac{1}{R}$$

As previously mentioned, the current rating was 10A which means if we take the point where A.T is equal to 500 in Figure 11, we would have to make the turn number 50, and if the turn number is 50, using the equation above the inductance value becomes 300µH as shown in the above equation.

$$L = A_L N^2 = 120 * 10^{-9} * (50^2) = 300\mu H$$

Which is more than acceptable for our design. Then, we encountered a problem. Making 50 turns on only one core made the fill factor nearly 80% which is considered bad even if it is somewhat higher than 50%. Nevertheless, we did not have any other core; thus, we updated our design. We have thought to use two cores and make 23 turns on each of the cores, after that, we just connected them in series. This way, we have reduced the fill factor to 41%, reached an inductance value of 215µH as can be seen in Figure 12. Note that this value is for 50kHz, which is the MOSFET switching frequency. Moreover, by making them two inductors in series we believe we have prevented the heating of the inductor since we did not observe any heat problems at the inductor during our trials and the demo. You can see the final design of the inductor in Figure 13.



```
L .21449mH
R .5980  Ω
F:50.000 kHz R.H ON
V:1.275  V    C.V ON
AUTO MANU   INT.B OFF
```

Figure 12: Designed inductor measurement

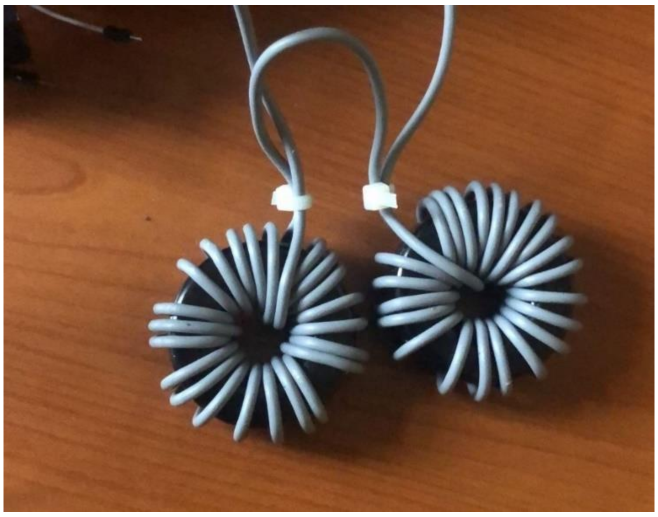


Figure 13: Inductor final design

Close-Loop Current Control

In this project, our task was to supply constant 10A to a battery that has variable voltage. Due to the battery having a variable voltage, closed-loop current control is a must for this project. In our design, the current reading is done via the ACS712 current sensor which offers reliable operation up to ±30A. The current sensor can be seen in Figure 14.

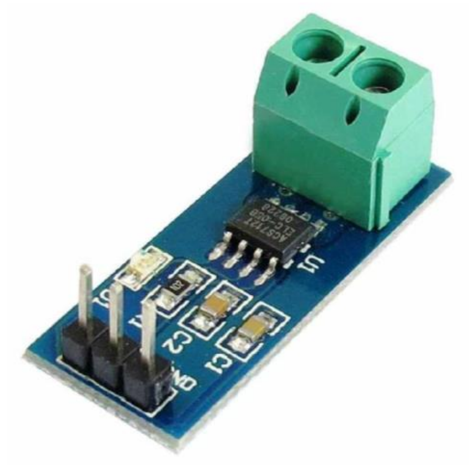


Figure 14: ACS712

We have used a digital current sensor since we decided to use Arduino UNO for the switching, due to not being able to implement an analog controller despite our efforts. The sensor has three connections to the Arduino: 5V for the V_{cc}, GND and ADC reader. We have located the sensor just before the battery to sense the current passing through the battery directly. First, we have implemented a PI controller but after some testing, we have realized that the PI controller was not a suitable controller for our aim since the internal resistance of the battery is too low, approximately 0.03Ω, even small changes in the output voltage can cause very high inrush currents to pass through the battery, and with the PI control we could not prevent overshoots that cause high currents to pass through at 10A operation. As mentioned in the circuit design, we located a diode just before the battery to prevent the discharging of the battery when it is connected to our circuit in the non-charging state, making the current reading 0A until a voltage value is reached where 10A can be supplied. Due to having a large current gap, the error accumulates and when the duty cycle value is reached, a high overshoot happens. If we were more experienced in PI control or control in general, we could probably make it work but by grappling with a tight schedule and a close deadline, we have chosen to adjust our control methodology. It should be noted that we tuned the parameters, K_P and K_i, using MATLAB but still, we were not able to solve the overshoot problem and we were afraid to use PI control for the 10A battery test.

Thus, we have adjusted our control algorithm. First, the reference current which is the current level we want to supply the battery with is specified inside the code. Then, the current measurement is taken. The code compares the specified current with the measured current. If the specified current is bigger than the measured current, the duty cycle is increased to increase the output voltage of the buck converter. However, if the measured current is bigger, then the duty cycle is decreased to reduce the output voltage of the buck converter. A block diagram showing this process can be seen in Figure 15.

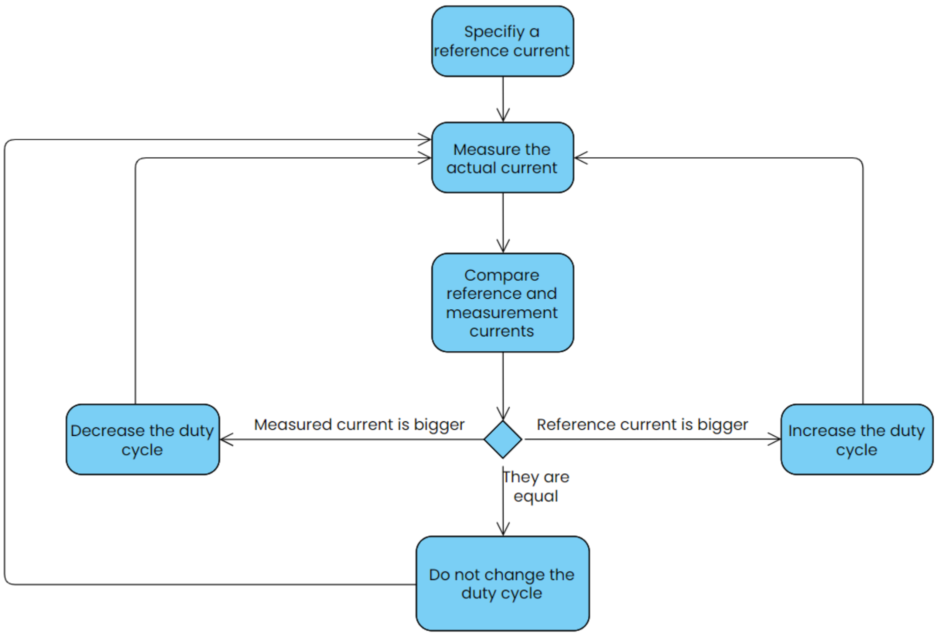

Specifiy a reference current
Measure the actual current
Compare reference and measurement currents
Decrease the duty cycle
Measured current is bigger
Reference current is bigger
Increase the duty cycle
They are equal
Do not change the duty cycle

Figure 15: Block diagram represents the control process

Using this control technique, we could charge the battery successfully. Moreover, the implemented code is given in Appendix C.

Reverse Current Protection

When we connect charger to the battery, initially uncharged capacitors want to draw current from the battery. To prevent that we placed a diode between the battery and the charger.

Over Current Protection

For over current protection we placed a 15A fuse. The main idea of this fuse is to cut the connection of the circuit and variac to protect both battery and charger. In Figure 16 you can see the fuse left top corner of the stripboard.

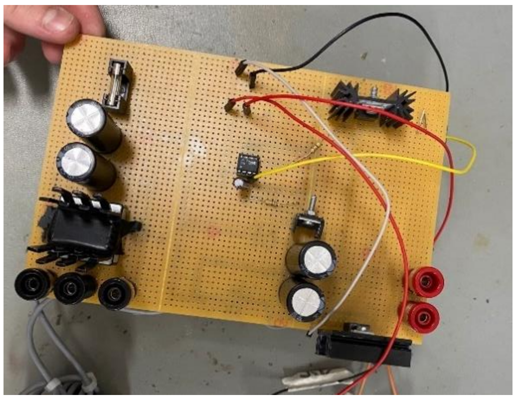


Figure 16: Front view of the stripboard

Component List & Cost Analysis

Components that we used in the project and their price are given in Table 1.

Component	Description	Ratings	Cost per Piece	Amount
VS-36MT160	Three Phase Full Wave Diode Rectifier	1600V- 35A	15.13 €	1
2200 µF	Input and Output Capacitor	63 V	0.55 $	4
LM2596	Adjustable Voltage Step Down Power Module	In: 4V-35V Out: 1.23V-30V/ 3A Max	26.10 TL	1
Fan	-	12 V- 0.15 A	130 TL	1
IR2106	MOSFET Driver	20 V- 200mA	72.07 TL	1
10 µF	Driver Capacitor	50 V	0.52 TL	1
IRF540	MOSFET	100V- 33 A	18.42 TL	1
Fuse	-	15A	0.70 TL	1
Fuse Holder	-	-	3.68 TL	1
Arduino Uno R3 SMD CH340 - Klon	Digital Controller	50 V- 50 mA	180.50 TL	1
ACS712	Current Sensor	5V- 20A	71.83 TL	1
Stripboard	-	-	45 TL	1
0075192A7	Inductor Core	-	5 $	2
AWG 14 Cable	Cable for Inductor Windings	15A	7.5 TL/m	5.3m
DSS16-0045A	Buck Converter and Battery Diode	45V- 16A	41.26 TL	2
45AS	Diode Heat Sink	-	24.68 TL	1
Heat Sink	MOSFET Heat Sink	-	27.63 TL	1
Heat Sink	Rectifier Heat Sink	-	25 TL	1
Jack	Input and Output Pins	-	4.42 TL	5
Others	Resistors, solder etc.	-	5 TL	1
Grand Total:	1640 TL/ 49.75 €			

Table 1. Component List and Cost Analysis

Charger cost 49.75 € As can be seen from the table 1; however, since we used the existent components in the lab, the cost for us is way more below than calculated one.

Test Results

To test the design, we used 1 Ω resistance to see whether we can supply 10 A or not. Then we charged the battery with 15, 20 and 25 V_{l-l} voltage and observed waveforms.

1Ω Load Resistance

The main reason to test our cirucit with resistance is to observe our circuit current waveform, checking the switching operation and measuring the heating processes. In Figure 17, you can see the MOSFET gate voltage (yellow), output current (blue) and and inductor current (green).

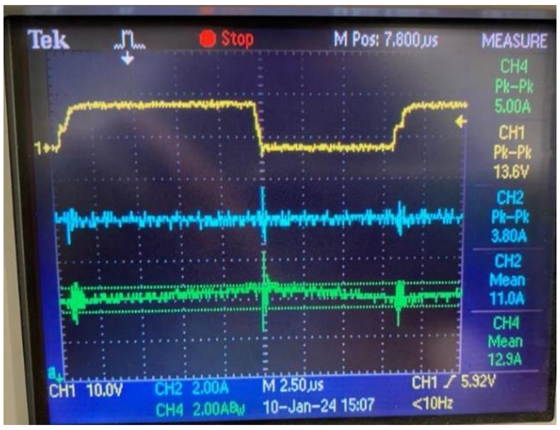


Figure 17: 1 ohm, 10 A Test Results

As we can see, MOSFET gate voltage peak value is equal to 13.6 V which is enough to open to the MOSFET. Average output and inductor currents ripple seem a bit high at the switching instants but this is not a problem thanks to our component ratings.

Charging the Battery

Just like in the above case, you can see the MOSFET gate voltage (yellow), output current (blue) and and inductor current (green) in the following figures, but this time, we are charging the battery with 10 A. Figure 18 shows wavefroms with 15 V_{l-l} input. Figure 19 and 20 show wavefroms with 20 and 25 V_{l-l} input respectively.

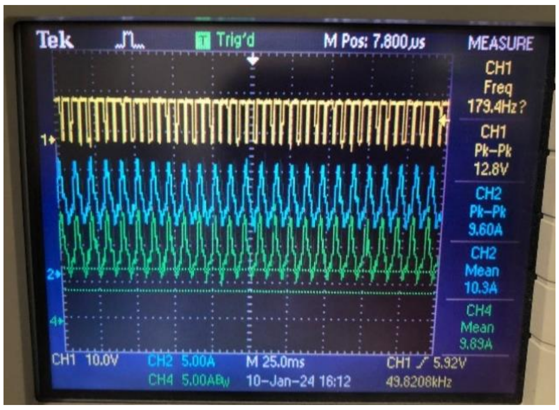


Figure 18: Wavefroms with 15 V_{l-l} input

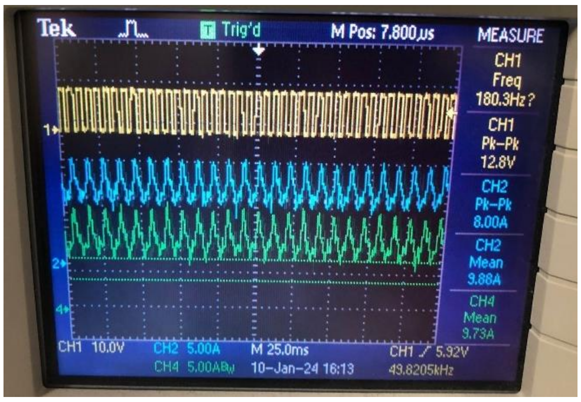


Figure 19: Waveforms with 20 V_{l-l} input

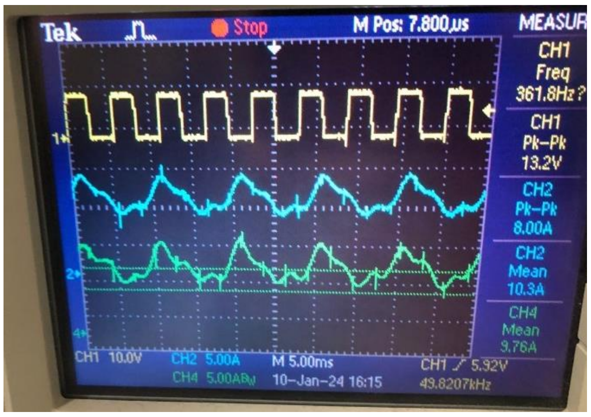


Figure 20: Waveforms with 25 V_{l-l} input

As you can see from the above figures, we can not see exactly 10 A but our mean output current error is at most %3. With this test, we saw that we are able to charge the battery with nearly 10 A.

Loss Calculation & Thermal Analysis

The design has four main loss elements:

- Three-phase rectifier diodes
- Switch which is a MOSFET
- Buck converter diode
- Battery diode

Since we have used different heatsinks for each of the components, we must derive their thermal analysis separately.

Three-phase rectifier diode

As a rectifier, we have used a three-phase rectifier, VS-36MT160. In the datasheet, it is given that the instantaneous forward voltage per junction is approximately 0.9V for 10A. It should be noted at a given time two of the junctions are in conduction mode for the rectifier to work so we must multiply the conduction loss by two. The ambient temperature was measured as 23°C at the time of our trials.

$$T_J = P_{loss}(R_{JC} + R_{CS}) + T_{ambient}$$

$$P_{loss} = 2 * 0.9 * 10 = 18\text{W}$$

$$T_J = 18(1.35 + 0.2) + 23 = 50.9°C$$

This temperature can be considered acceptable.

MOSFET

The MOSFET has two main losses: conduction losses and switching losses. As a MOSFET, we have usedIRF540N and the switching frequency was 50kHz. From this knowledge, we can calculate the losses.

- **Conduction Losses:**
  $$P_{cond} = R_{DS} I^2 D$$
  The conduction loss expression includes the duty cycle, which is a variable parameter in our project. According to our calculations, the duty cycle can have the value, at most, 0.8 so we can take the D as 0.8 to calculate the losses at the worst heating case.
  $$P_{cond} = 0.044 * 10^2 * 0.8 = 3.52\text{W}$$

- **Switching Losses:**

$$P_{sw} = V_{in} I f_{sw}\left(t_r + t_f\right) = 33 * 10 * 50000 * (35 + 35) * 10^{-9} = 1.155\text{W}$$

Then the total loss and temperature increase can be calculated.

$$T_J = (P_{cond} + P_{sw})R_{JA} + T_{ambient}$$

$$T_J = (3.52 + 1.155)62 + 23 = 312.85°C$$

As you can see, without a heatsink the temperature becomes too high; therefore, a heatsink must be implemented. You can see the new temperature with a heatsink in the equation below.

$$T_J = (P_{cond} + P_{sw})(R_{JC} + R_{CS}) + T_{ambient}$$

$$T_J = (3.52 + 1.155)(1.15 + 0.5) + 23 = 30.7°C$$

Buck Converter Diode

For the buck converter diode, we have used DSS16-0045A. The conduction loss can be considered as the main loss on the diode.

$$Pconduction = V_F I = 0.6 * 10 = 6W$$

$$T_J = P_{cond}(R_{JC} + R_{CS}) + T_{ambient}$$

$$T_J = 6(1.4 + 0.5) + 23 = 34.4°\mathrm{C}$$

As you can see a good heatsink also gets rid of the heating problem of the diode.

Battery Diode

We have chosen the battery diode to be same as the buck converter since the diode parameters were proper and we had the spare component. Since having same ratings as the buck converter, we can estimate that the losses on the battery diode are same with the losses on the buck converter diode. Thus, a heatsink would be enough to cool down the component.

However, just to be safe, we have also implemented a cooling fan to change the cooling to forced cooling. In Figure 21, you can see the final circuit component's temperatures after an operation that continues longer than 5 minutes. As you can see in Figure 21, the maximum temperature of a component is 48.1°C. The most heated component is the three-phase rectifier diode which is compatible with thermal analysis.

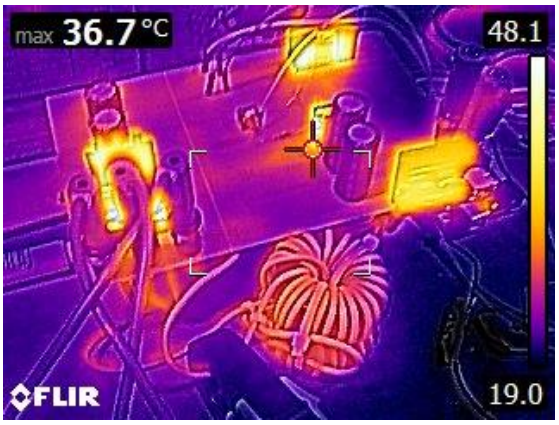


Figure 21: Thermal photograph of the final circuit

Conclusion

In this project we designed an AC-DC converter capable of producing a 10A DC output for battery charging and this report is the explanation of the whole process. From beginning to end we encountered lots of problems like placing the components on stripboard, miscalculations in controller design, time issues, not working components etc. At the end we encountered these problems as a team. We learned the importance of each team member being knowledgeable about every subsystem, the significance of starting the project early as a key point in resolving faulty component and design issues, the variation in thermal requirements for each circuit, and the ease and enjoyment in solving process-related problems with a cohesive team.

We would like to express our gratitude to all members of the Power Electronics Laboratory and our classmates who have been supportive and helpful through the process.

References

- [z] Magnetics, 0075192A7 datasheet, Jun. 2021
- https://keysan.me

Appendix A

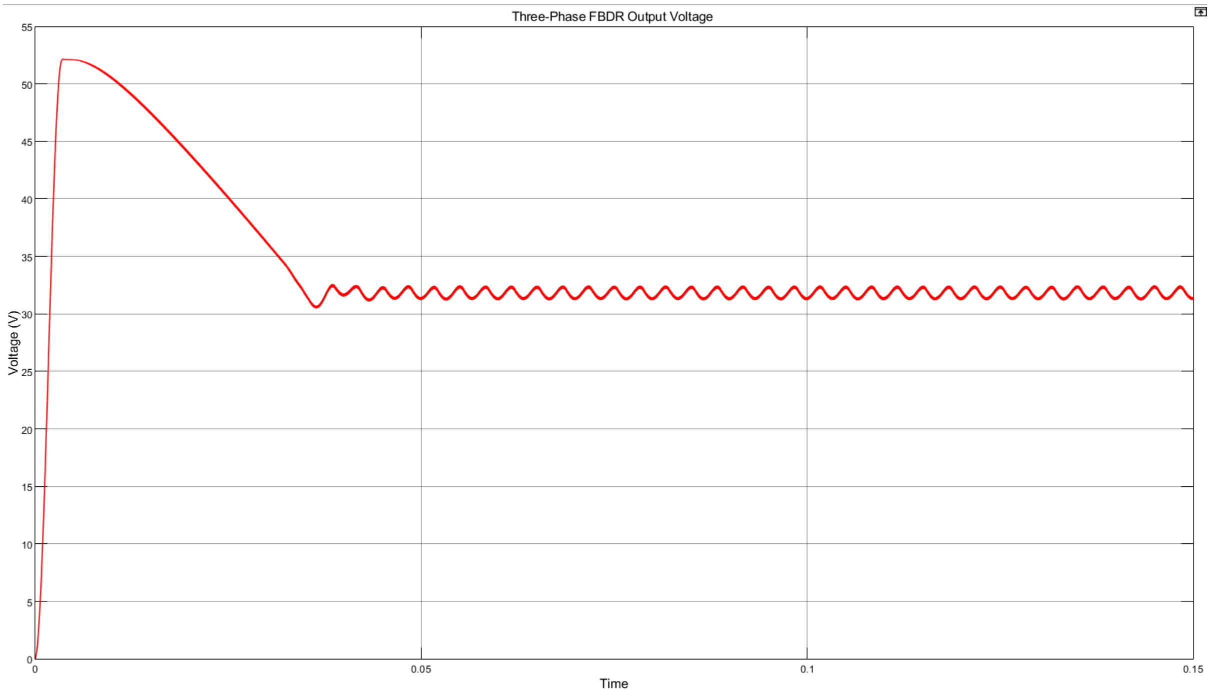


Figure a. Three-phase FBDR output voltage waveform

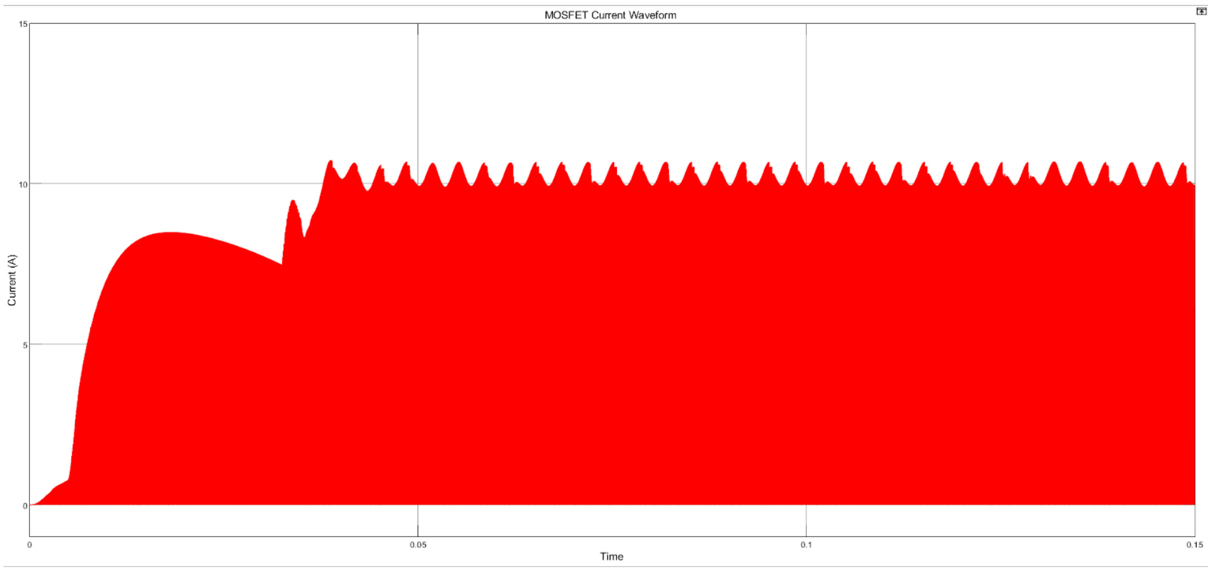


Figure a+1. MOSFET current waveform

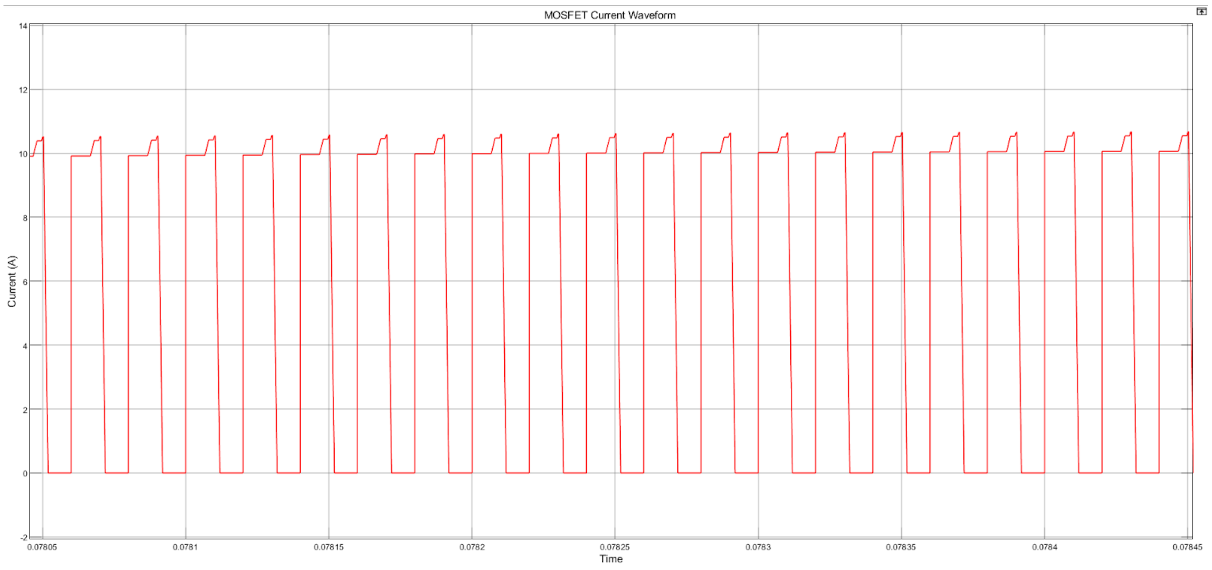


Figure a+2. MOSFET current waveform (zoomed-in in the steady state)

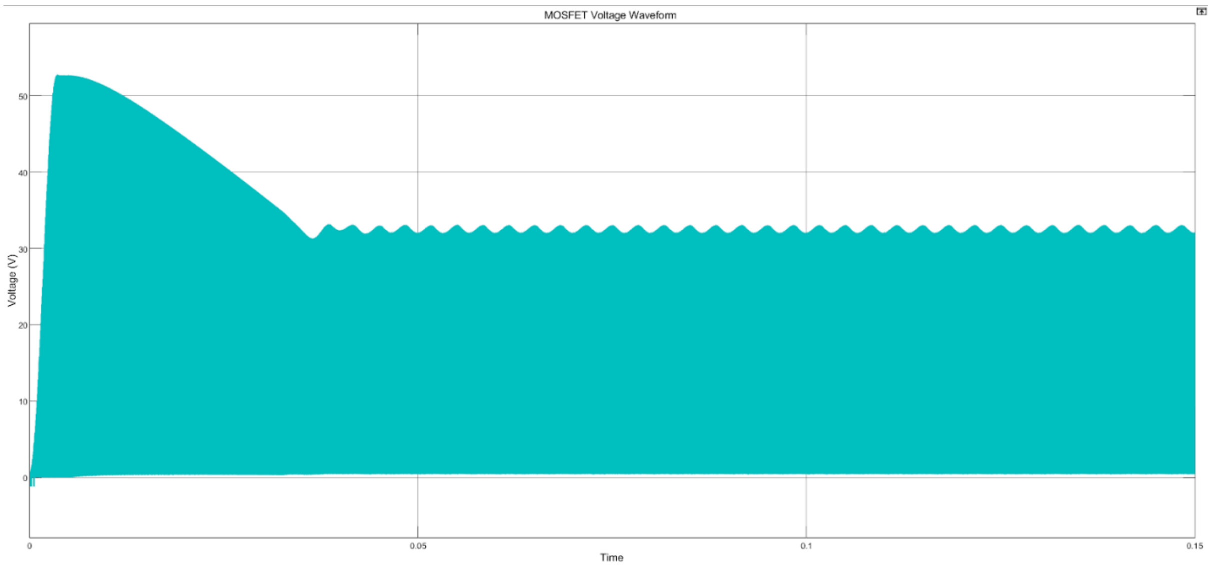


Figure a+3. MOSFET drain-source voltage waveform

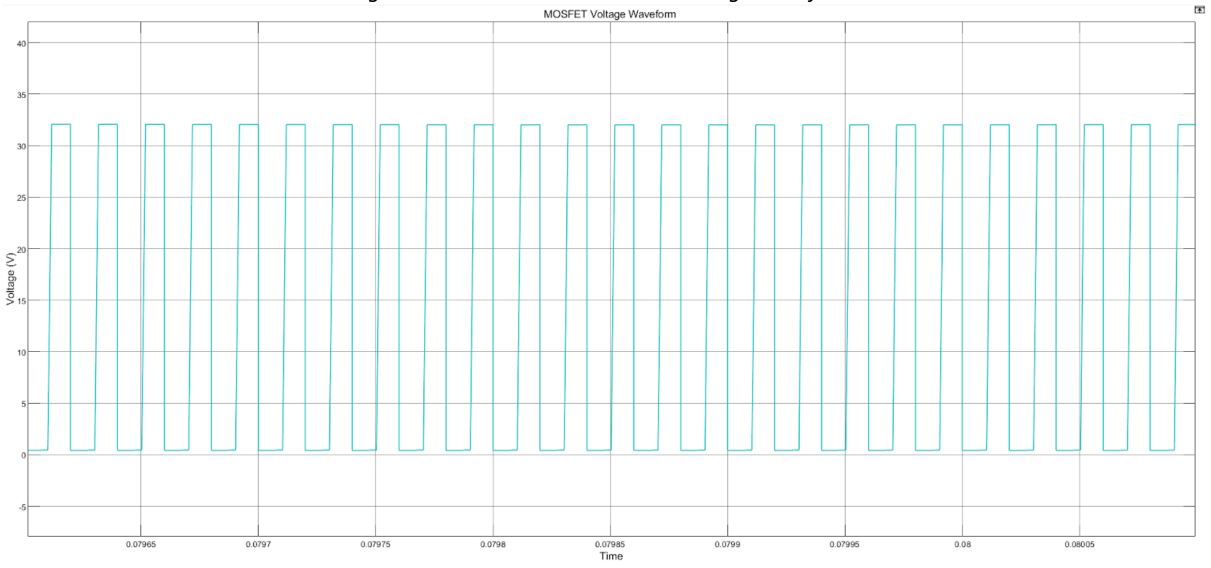


Figure a+4. MOSFET drain-source voltage waveform (zoomed-in in the steady state)

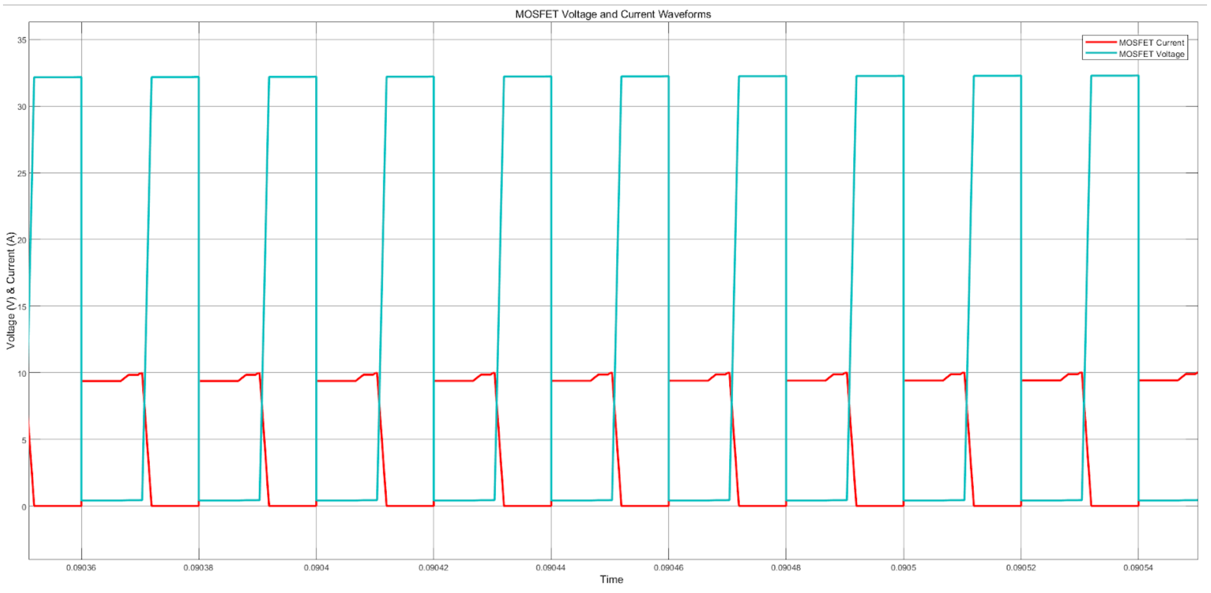


Figure a+5. MOSFET drain-source voltage and current waveforms (zoomed-in in the steady state)

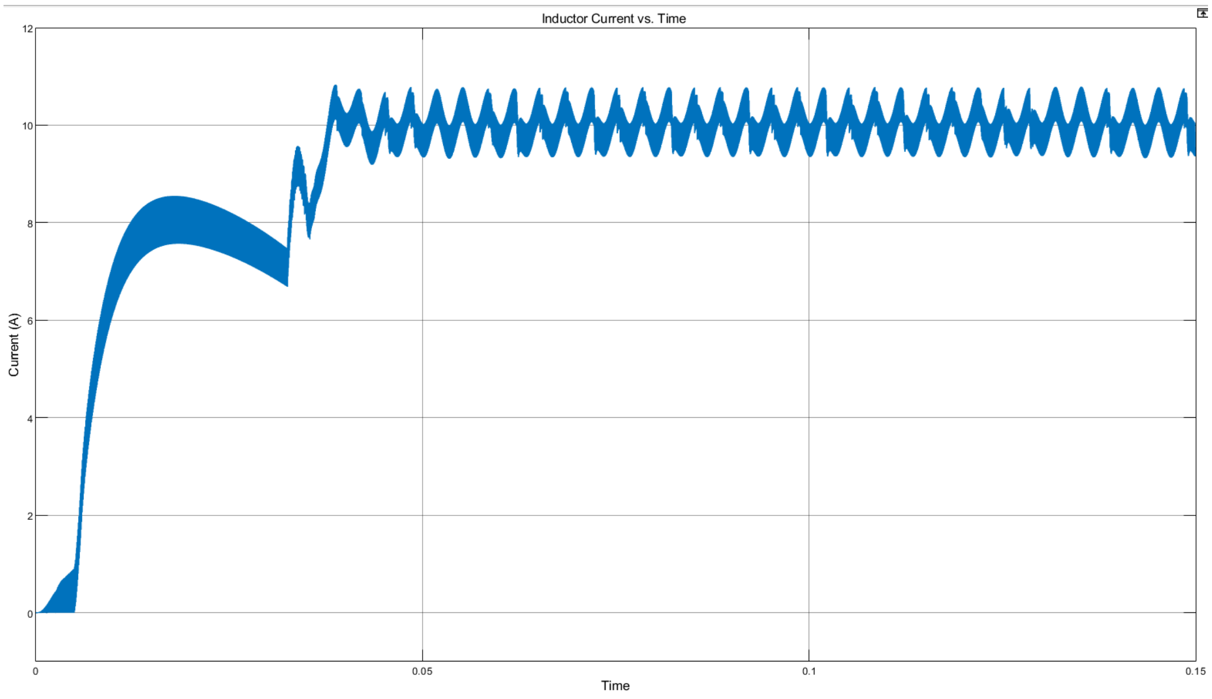


Figure a+6. Inductor current vs. time graph

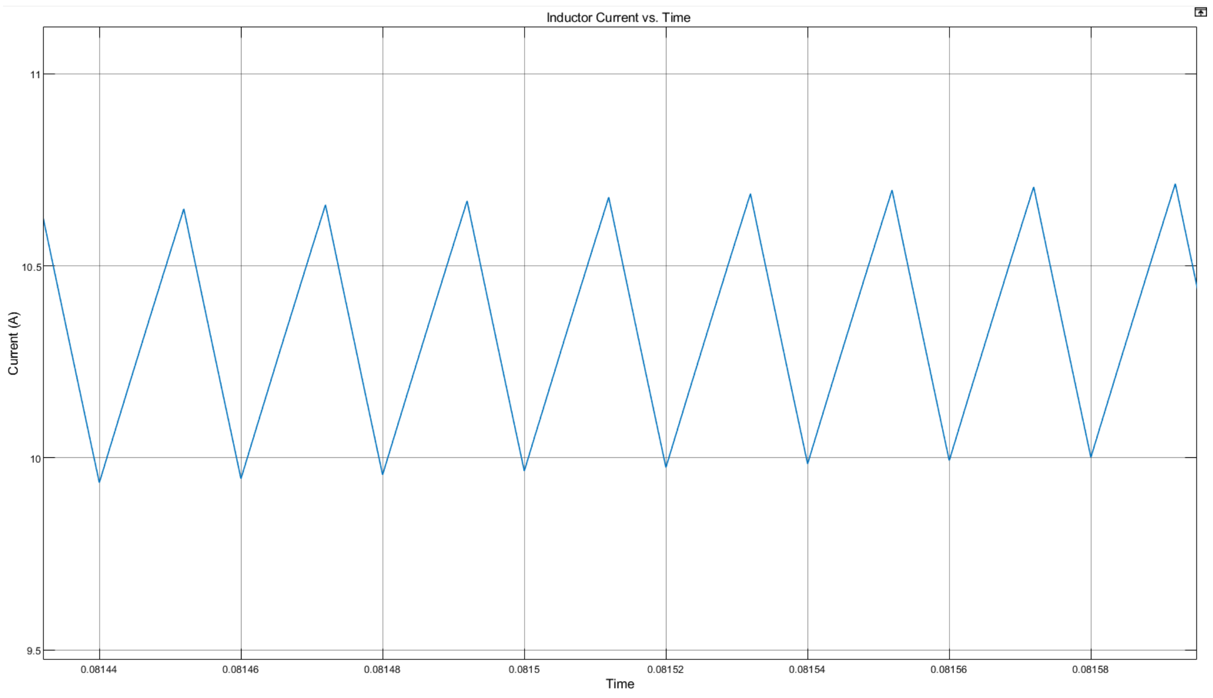


Figure a+7. Inductor current waveform (zoomed-in in the steady state)

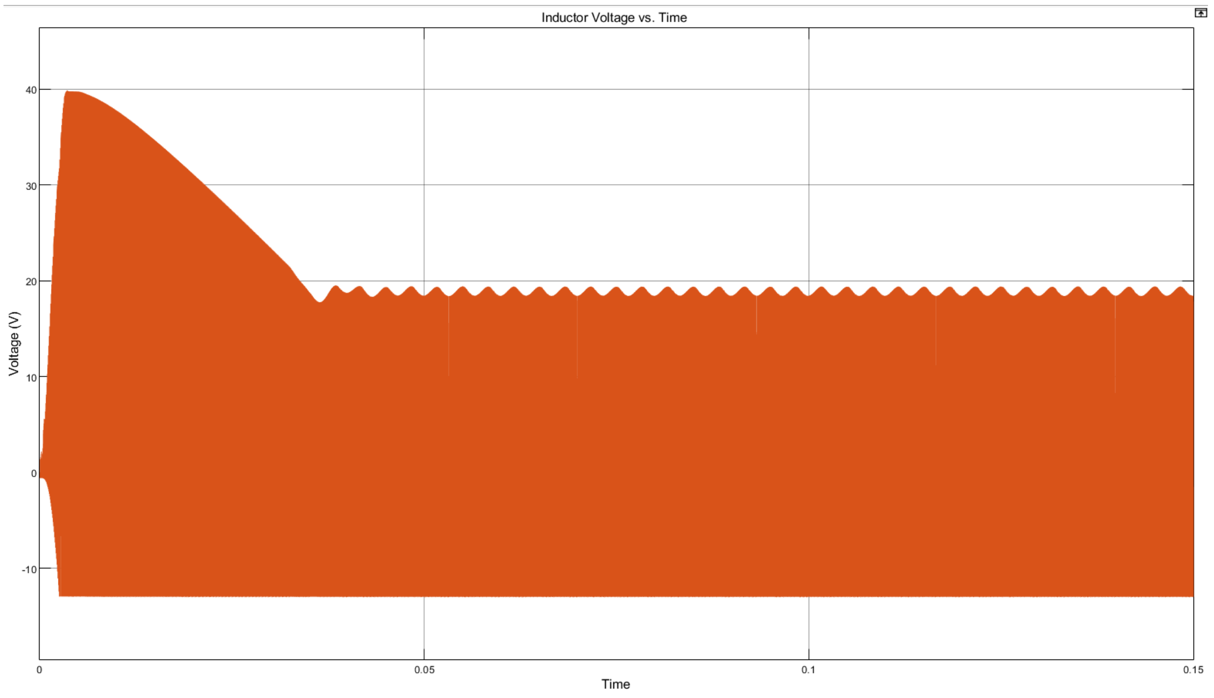


Figure a+8. Inductor voltage waveform

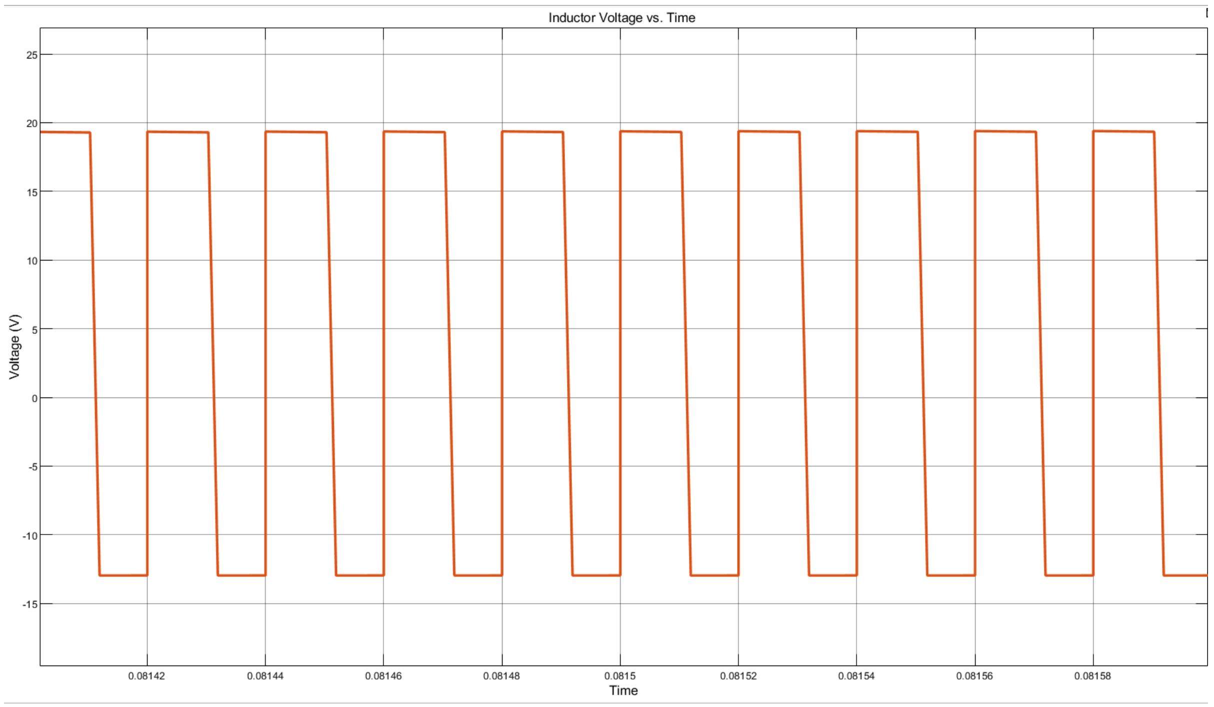


Figure a+9. Inductor voltage waveform (zoomed-in in the steady state)

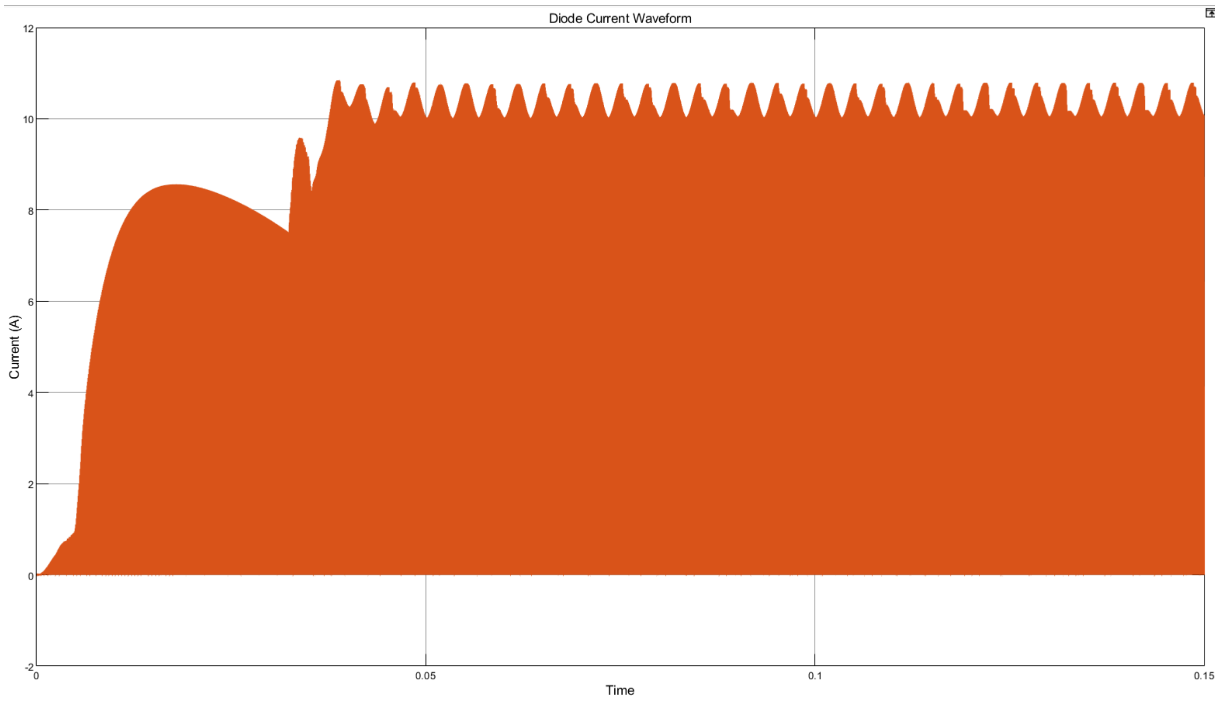


Figure a+10. Diode current waveform

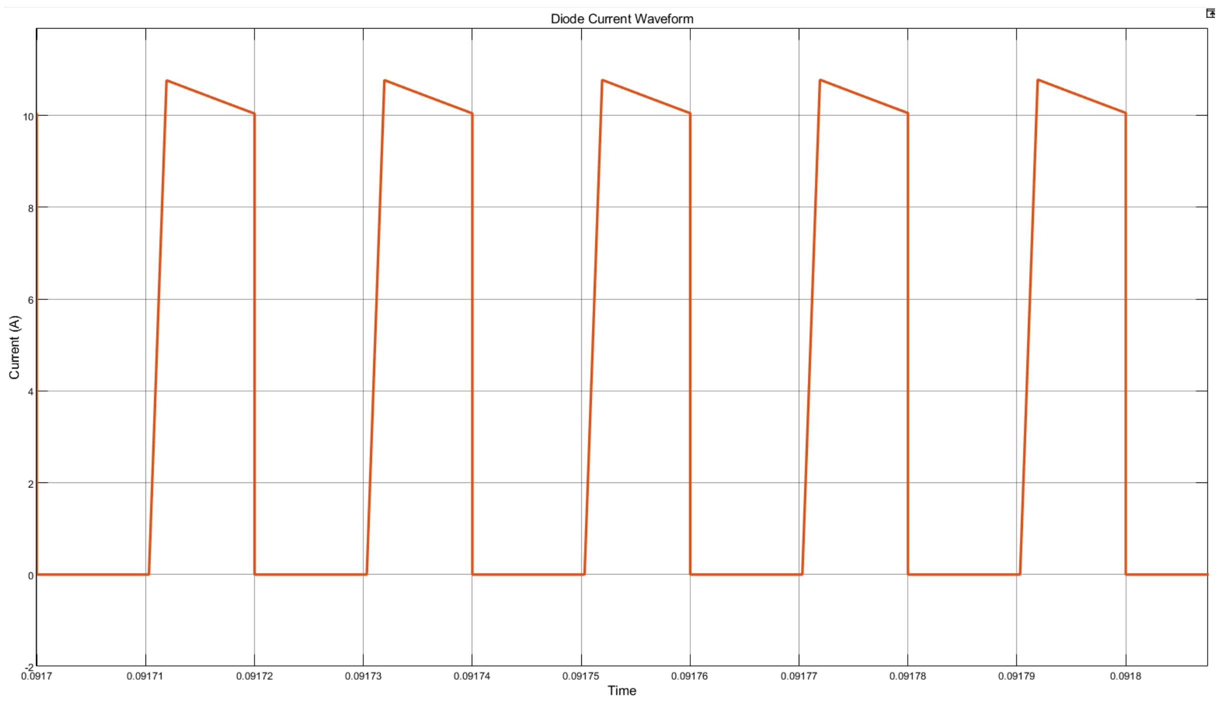


Figure a+11. Diode current waveform (zoomed-in in the steady state)

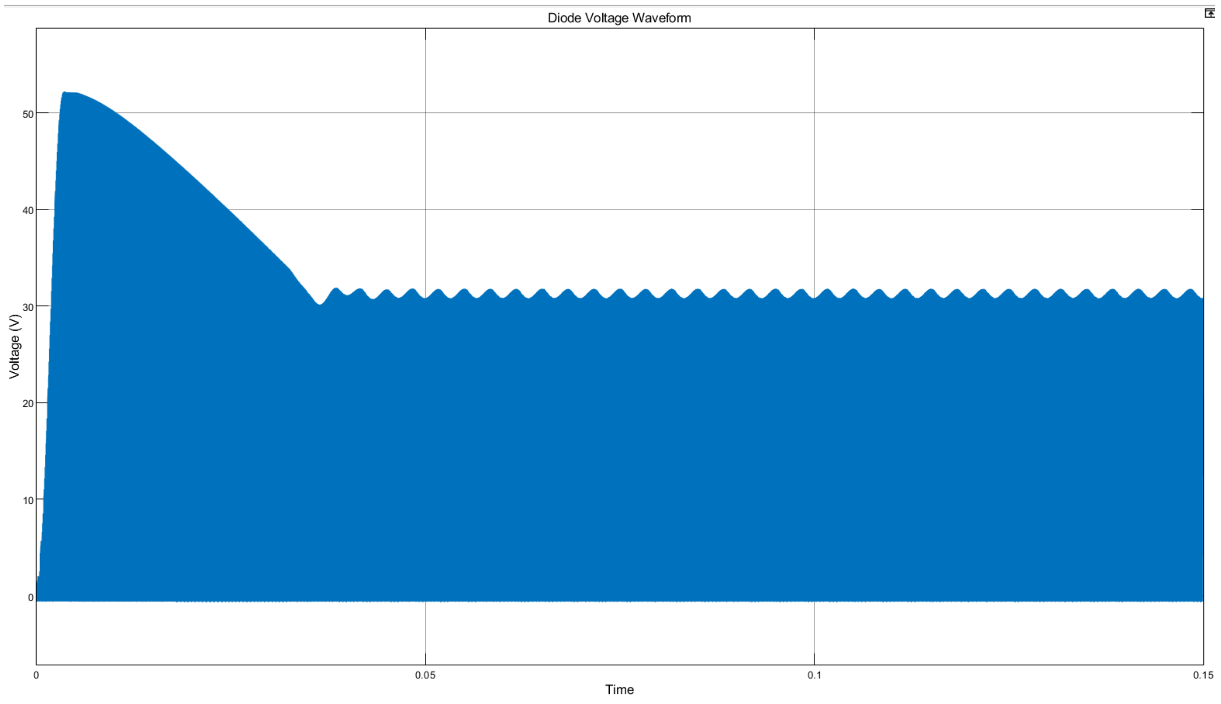


Figure a+12. Diode voltage waveform

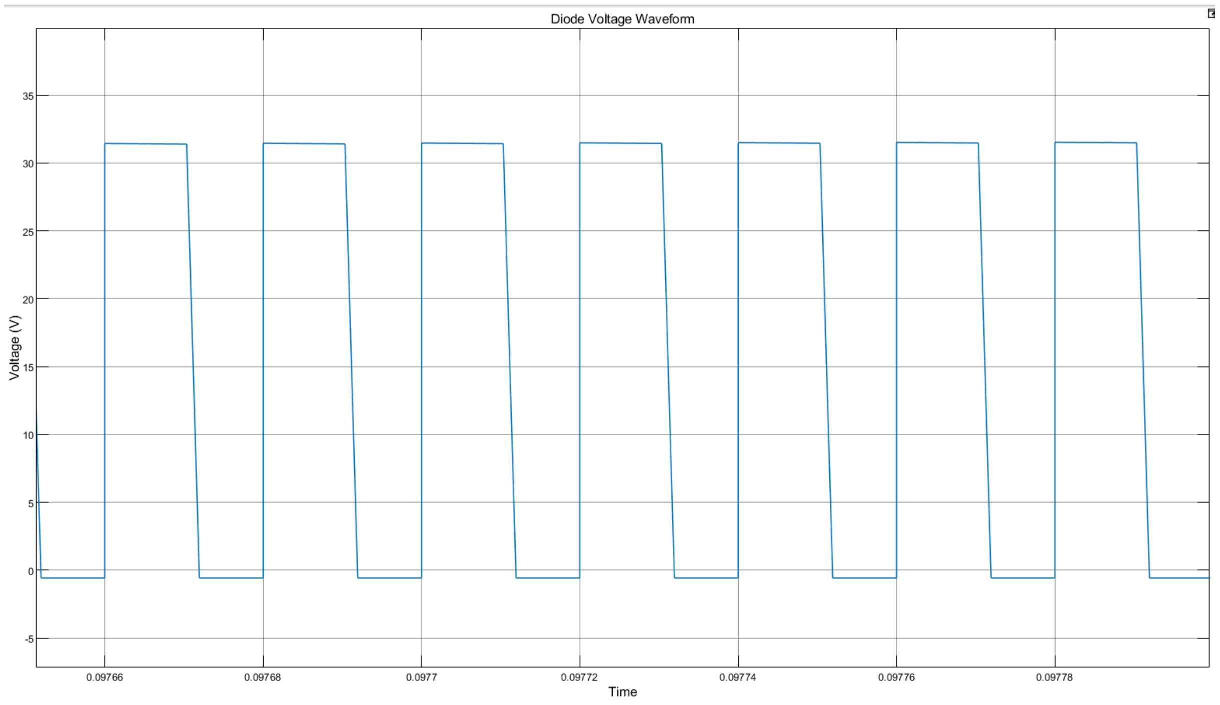


Figure a+13. Diode voltage waveform (zoomed-in in the steady state)

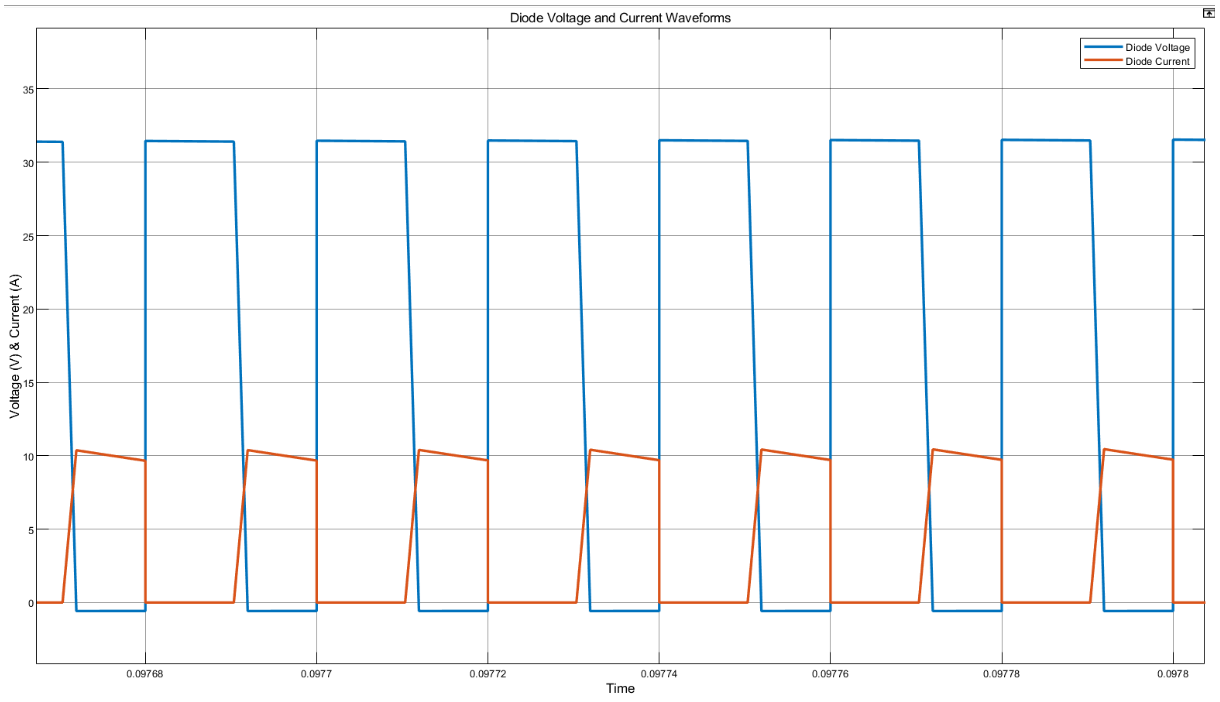


Figure a+14. Diode voltage and current waveforms (zoomed-in in the steady state)

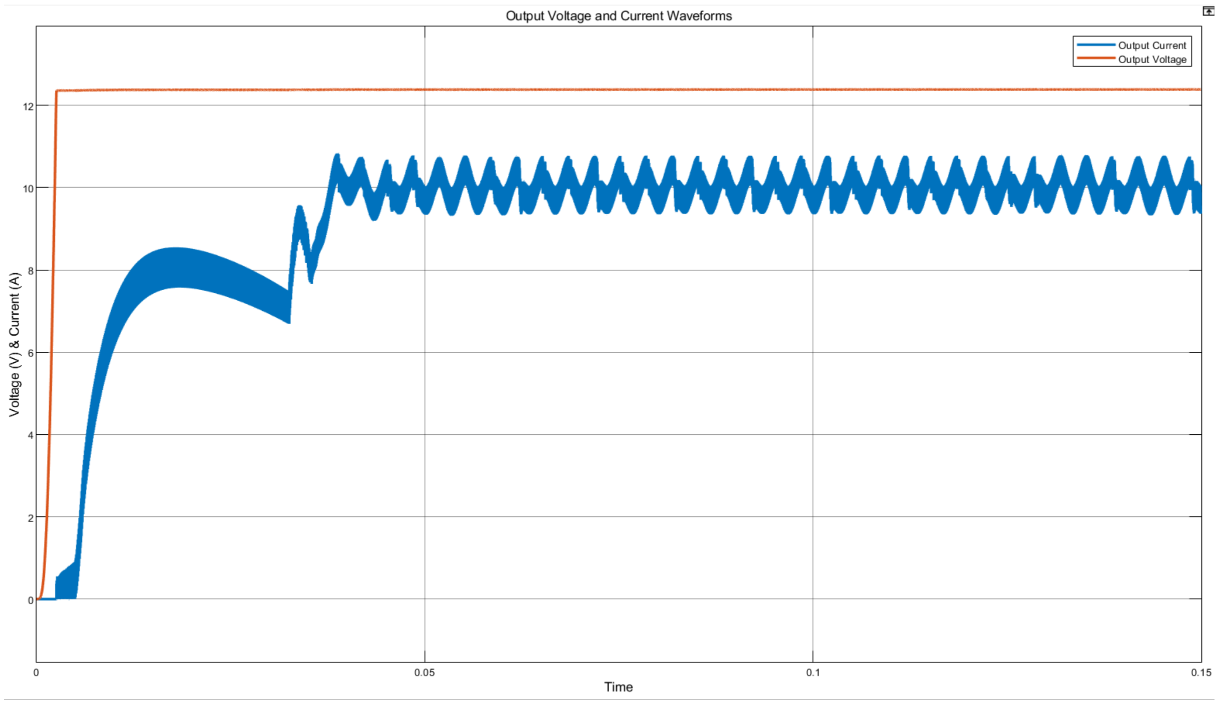


Figure a+15. Buck converter output (battery input) voltage and current waveforms

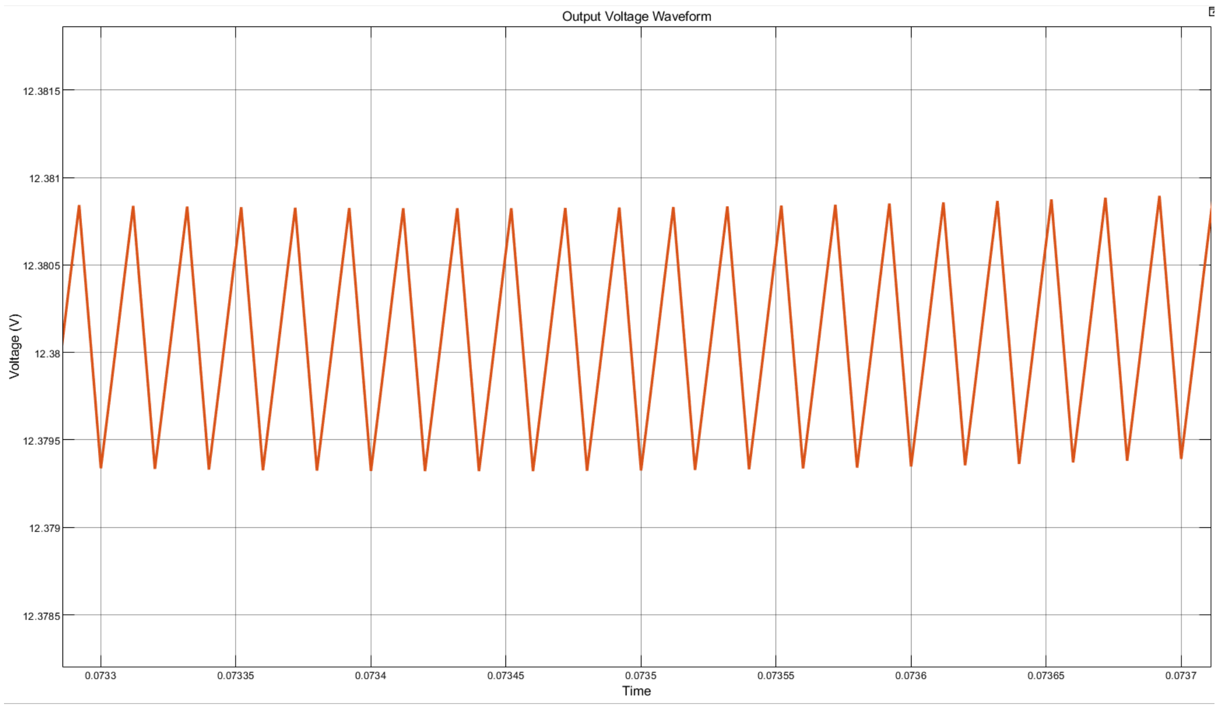


Figure a+16. Buck converter output voltage waveform

Appendix B

Link to diode datasheet:
https://cdn.ozdisan.com/ETicaret_Dosya/352467_9371492.pdf

Link to MOSDET datasheet:
https://www.vishay.com/docs/91021/irf540.pdf

Link to gate driver IC datasheet:
https://www.infineon.com/dgdl/Infineon-ir2106-DS-v01_00-EN.pdf?fileId=5546d462533600a4015355c7cfc51673

Appendix C

```
#include <PWM.h>

int pwm_pin = 10;   // the pin that the PWM is attached to
int32_t frequency = 50000; //frequency (in Hz)
int duty = 6; // in (%)
float duty_adjust = 255*duty/100;
float ref_current = 10.0; //Reference current (in A)
float current_error = 0;
void setup()
{
  Serial.begin(9600);
  InitTimersSafe();
  //sets the frequency for the specified pin
  bool success = SetPinFrequencySafe(pwm_pin, frequency);
  if(success) {
    pinMode(13, OUTPUT);
    digitalWrite(13, HIGH);
  }
}
void loop()
{
unsigned int x=0;
float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,Current_reading =0.0;
  for (int x = 0; x < 50; x++){ //Get 50 samples
  AcsValue = analogRead(A0);     //Read current sensor values
  Samples = Samples + AcsValue;  //Add samples together
  delay (3); // let ADC settle before next sample 3ms
}
AvgAcs=Samples/50.0;//Taking Average of Samples
Current_reading = (2.5 - (AvgAcs * (5.0 / 1024.0)) )/0.066;
delay(5);
pwmWrite(pwm_pin, duty_adjust);
current_error = ref_current - Current_reading;
if (AcsValueF<ref_current){
  duty_adjust=duty_adjust+1;
}
if (AcsValueF>ref_current){
  duty_adjust=duty_adjust-1;
}
if (duty_adjust>254){
  duty_adjust=230;
}
if (duty_adjust<10){
  duty_adjust=15;
}
delay(30);
}
```