Prototype Electric Vehicle Phase II

M. Ellingson, A. Tkacz, S. Patel
Department of Electrical Engineering
Northern Illinois University, Dekalb IL 60115

***Abstract*—The primary purpose of our project was to continue the development of the benchtop prototype electric vehicle through the application of a bi-directional converter which drives an electric motor using a battery pack. The key features of our project are as follows. There is a bi-directional converter which boosts and bucks voltage to transfer energy between the batteries, motor, and capacitors. The client will use a terminal to control the motor speed and be notified of measurements and fault-conditions associated with running the motor. The design chosen to control the bi-directional converter uses Proportional-Integral (PI) control. This was implemented with an Arduino microcontroller and an analog voltage sensor. The motor drive is inputted with DC from the converter and outputs three-phase AC to run the induction motor. The finished project is able to turn the motor using the battery with proper communication from the Arduino in order to do so. This project will give a future prototype electric vehicle team the opportunity to further develop this design.**

I. Introduction

The main part of this project is the bi-directional converter. This converter is a combination of a boost and buck converter circuit. A boost converter is created by connecting an inductor, IGBT (insulated-gate bipolar transistor), and diode. When the IGBT is turned on, the current builds up in the inductor. When turned off, the current is released from the inductor through the diode to the capacitor and load. A buck converter is created by connecting the same components in a different manner. When the IGBT is turned on, current flows through the inductor. The inductor should be large enough to maintain continuous conduction for the load. This means that when the IGBT is turned off, current will continue flowing to the load (i.e. the voltage source) by looping through the diode.

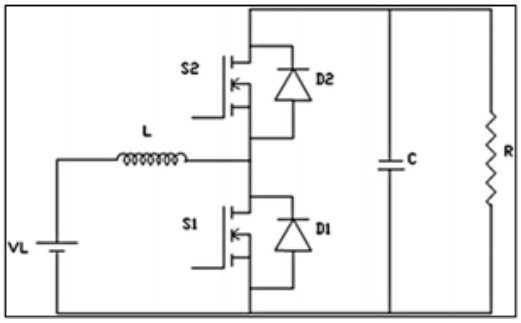

Figure 1: Bi-directional converter schematic

The bi-directional converter, Figure 1, is created using the same circuit as the boost converter apart from replacing the diode with a second IGBT. The boost converter portion is controlled using the PI method. A block diagram of this controller can be seen in Figure 2.

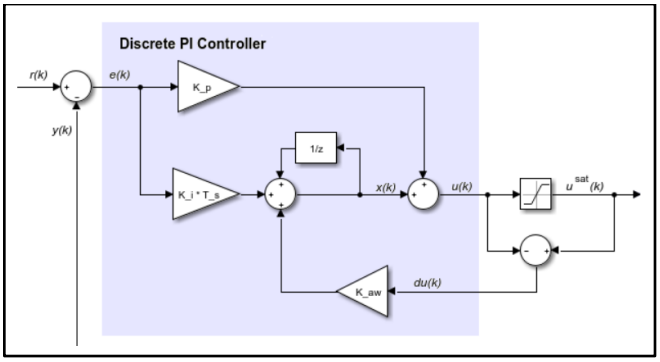

Figure 2: Discrete PI Controller Block Implementation (shaded) [1]

The Discrete PI Controller block uses the backward Euler discretization method to calculate the control signal u(k) as shown below.

$$u(k) = \left[K_p + (K_i + du(k)K_{aw})\frac{T_s z}{z-1}\right] \mathrm{e(k)},$$

where

- u is the control signal.
- K_p is the proportional gain coefficient.
- K_i is the integral gain coefficient.
- K_{aw} is the anti-windup gain coefficient.
- T_s is the sampling period.
- e is the error signal.

Figure 3: Discrete PI Controller Equation [1]

II. Materials and Methods

The boost part of the bi-directional converter was simulated with Mathworks Simulink. This was accomplished by drawing a boost converter with a PI controller in Simulink as shown in Figure 4. The implementation and equation shown in Figures 2 and 3 were included in the Simulink schematic via the Discrete PI Controller block. The output waveform of the boost converter is analyzed in the Results and Discussion section. The actual boost converter controlled by the Arduino microcontroller, using the PI control method, that drives the motor with the battery supply voltage is discussed in the next paragraph.

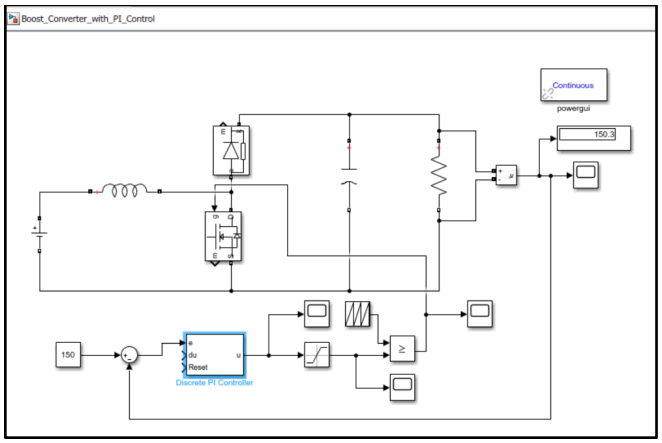

Figure 4: Boost Converter with PI Control Simulink Schematic

```
float s = 140.0;
float Kp = 0.6;
float Ki = 0.008;
y = analogRead(INPUT1) / 4;
e = s - y;
ae = ae + e;
ae = bound(ae, -10000.0, 10000.0);
u = 128 + ( e * Kp ) + ( ae * Ki );
u = bound(u, 0, 210);
analogWrite(PWM_BOOST, u);
digitalWrite(7, !digitalRead(7));
```

Figure 5: Select portion of Arduino code

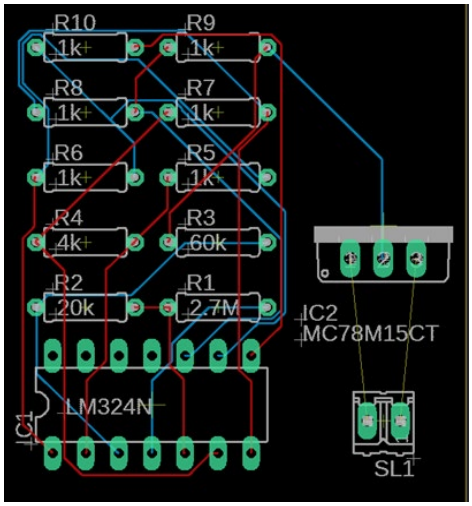

Figure 6: Analog Voltage Sensor PCB

It was possible to implement a tunable PI controller using the Arduino programming environment. Figure 5 shows an excerpt of the implemented code. The first three lines are from the top and control the setpoint for voltage (s), proportional constant (Kp), and integral constant (Ki). Lines four through eleven are from the main program loop. Line four reads and scales the measurement from the voltage sensor (Figure 6). Line five computes the error between that measurement and the setpoint. Line six adds that error together which accomplishes continuous integration. Line seven limits the error to ±10000. Line eight computes the control effort by multiplying the error by Kp and the accumulated error by Ki. Line nine limits the output duty cycle to 210/255 to avoid damaging the IBJT. Line ten writes the output value to the PWM pin. Line eleven toggles an output pin which when connected to an oscilloscope can be used to verify the integral step size is constant. When looped continuously, this program performs the function of a PI controller.

III. Results and Discussion

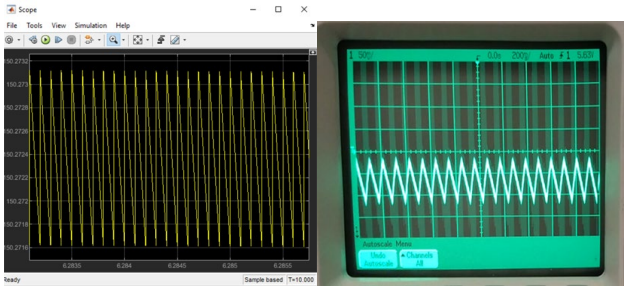

Figure 7: Simulink Output (left) and Actual Output (right)

The Simulink simulation of the boost converter with PI control yielded an output voltage with 0.2 % error and a ripple voltage of 1.5 mV. The actual boost converter yielded results within a couple of volts of the setpoint with little ripple voltage. Figure 7 shows how the output voltages for both the Simulink simulation and the actual output measured on the oscilloscope have the same sawtooth waveform pattern. This confirms that the actual PI-controlled boost converter operates like the one simulated in Simulink.

IV. Conclusions

Development for the bi-directional converter included implementing boost functionality to run the motor. PI control was used and implemented using MathWorks Simulink and the Arduino IDE (integrated development environment). The issues encountered during development, including those involving the Arduino communication and reset cycle, were solved. Documentation has been left for future development work on the Prototype Electric Vehicle.

Acknowledgments

The authors would like to thank Dr. Donald Zinger for his guidance throughout the project design process. We would also like to thank Marcin Ciszek for releasing alarm code for the Arduino monitoring system. Finally, we appreciate having Fahad Alqahtani give his suggestions for improvement throughout the development process.

References

[1] "Discrete PI Controller," Discrete-time PI controller with external anti-windup input - Simulink, 21-Sep-2017. [Online]. Available: https://www.mathworks.com/help/physmod/sps/ref/discretepicontroller.html. [Accessed: 23-Apr-2020].