THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

MINIATURE PCB BRUSHLESS MOTORS – COIL DESIGN AND SENSORLESS CONTROL DEVELOPMENT

STEPHEN PORTER
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Electrical Engineering
with honors in Electrical Engineering

Reviewed and approved* by the following:

Tim Kane
Professor of Electrical Engineering
Thesis Supervisor and Honors Adviser

David Cubanski
Associate Professor of Electrical Engineering
Faculty Reader

* Electronic approvals are on file in the Schreyer Honors College.

ABSTRACT

Traditional high-speed brushless motors, such as those found on drones, are heavy and expensive components. Miniature brushless motors made from Printed Circuit Boards (PCBs) are lightweight, inexpensive, and easy to assemble, although relatively weak. One previous attempt to develop a PCB motor for use in a drone found it too weak to create sufficient lift and impossible to control using the typical Back-EMF (BEMF) method of sensorless control. This thesis attempts to improve on the design of that PCB motor and controller to resolve those deficiencies and make it viable for use in drones, which might allow for the size and cost of miniature drones to decrease. Development efforts focused on three areas: motor design improvements based on a literature review of existing designs, the design of an Electronic Speed Controller (ESC) prototyping ecosystem to quickly and inexpensively test different motors and sensing circuitry topologies, and the design of several sensing circuitry topologies intended to provide closed-loop BEMF feedback from the motor to a microcontroller controlling the ESC. The results of the work do not suggest that such a small motor can be controlled via BEMF to lift a drone. However, the ESC prototyping environment successfully simplified the prototyping and testing process. It will be improved upon to test larger motor designs and other control topologies in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thanks first to my parents for their unwavering support in all of my pursuits, and to my grandfathers, who inspired me to be a maker and made this journey possible. Thanks to Andy Kriebel for his advice and encouragement as I searched for my thesis and a job I would love. Thanks to the community of makers, connected by the internet, for sharing their work and making it open-source so others can learn and build from it. And thanks to Dr. Kane and Dr. Cubanski for their support of this idea.

Chapter 1

Introduction

The goal of the miniature Printed Circuit Board Brushless DC motor is to contribute to the development of small, low-cost drones. Other applications that require small, high-speed motors may exist, as well. Traditional motors are relatively expensive, heavy, and require mechanical integration into a system. Printed Circuit Board (PCB) motors, however, can be extremely cheap, lightweight, infinitely customizable, and easily integrated into an electrical system.

Only one example of a miniature PCB motor intended for use in a drone was published at the time this work started [1], [2]. It was designed and built by a Maltese engineer named Carl Bugeja, who published details of his work on the websites Hackaday.io and Youtube.com. The design had several shortcomings. First, the motor's coils (in the form of traces spirals on a PCB) were not designed to optimize performance, and the motor was not powerful enough to generate lift for a drone. Second, the Electronic Speed Controller (ESC) that was designed to control the motor was forced to use a Hall sensor to detect the rotor's position because the Back-EMF (BEMF) generated by the motor was too weak. The control scheme using a Hall sensor is viable but is less reliable and has a longer Bill of Materials than the industry-standard "sensorless" control scheme that relies on the motor's BEMF [3].

The goal of this thesis is two-fold. First, a PCB motor will be designed and tested to see if it improves on Bugeja's design. Although no PCB motors besides Bugeja's have been designed for drone applications, there are numerous examples of formal research into PCB motors for

other applications. This research will be used to design an optimized coil for the motor. The optimized motor design will retain most of the characteristics of Bugeja's design so the two motors can be directly compared during the testing phase. The improved coil should result in a more powerful motor that also generates a stronger and clearer BEMF signal.

The second goal of the thesis is to design an ESC to control the improved motor using a sensorless control scheme. If necessary, amplification circuitry of some kind will be implemented on the BEMF signal so it can be observed by the microcontroller. Neither objective, if accomplished, will necessarily result in a motor system that is capable of lifting a miniature drone. However, progress toward the two objectives will at least be stepping stones to realizing cheap and lightweight drones and other applications of this type of motor.

The first section of this thesis will provide background information on the various pieces of the project, including Brushless DC (BLDC) motor theory and sensorless control theory. Later sections describe the design of the motor itself and the electronic speed controller hardware and software. Results and future work conclude the thesis.

1.1: Brushless DC Motors

Permanent magnet motors have generally replaced brushed motors in electromechanical systems. Brushed motors are very simple to construct but have disadvantages such as brush friction and large amounts of noise generated by commutation [4]. Permanent magnet motors operate without any electrical connection between the motor's rotor and stator. This results in less noisy commutation and improved reliability but requires additional control hardware and software to spin the motor.

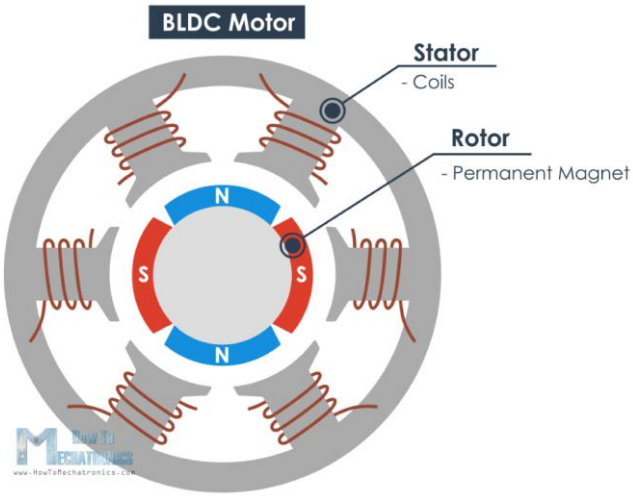

Figure 1: Parts of a BLDC Motor

BLDC motors have a series of discrete coils spaced out in a circle around the stator, meaning that each commutation shifts the source of the magnetic field (the active coil) by some angular step. In the most common configuration of BLDC motor, like the one above, that angle is 60 degrees. Such a large angular step means that BLDC motors suffer from significant torque ripple compared to other permanent magnet motors [3]. However, they are easier to control than other types of permanent magnet motors because they only require a constant-voltage power

supply and relatively simple switching hardware and software. Extension of this thesis's conclusions to other permanent magnet motor types can be the subject of further work.

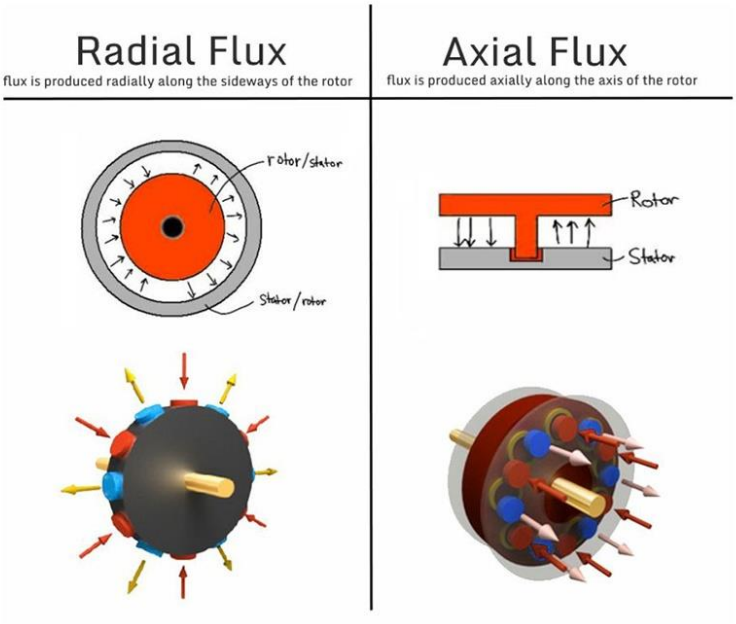

Figure 2: Rotary vs. Axial Flux Structure

Typical BLDC motors can be classified as rotary flux devices. In this category, the rotor and stator are concentric and coplanar, with the rotor's magnets mounted at a radius different from that of the stator's coils. An alternative to the rotary flux geometry is the axial flux geometry, in which the rotor and stator mounted like concentric disks separated by a small air gap. The PCB motors used in this thesis are axial flux devices that replace the traditional wire-wound stator coils with spiral traces. When the axial-flux motor is implemented on a PCB, the motor is infinitely customizable and easy to integrate into an existing electrical system. The PCB device can be lighter, smaller, and less expensive. However, PCB axial-flux motors produce relatively little torque per their size in comparison to conventional BLDC motors.

1.2: Motor Control

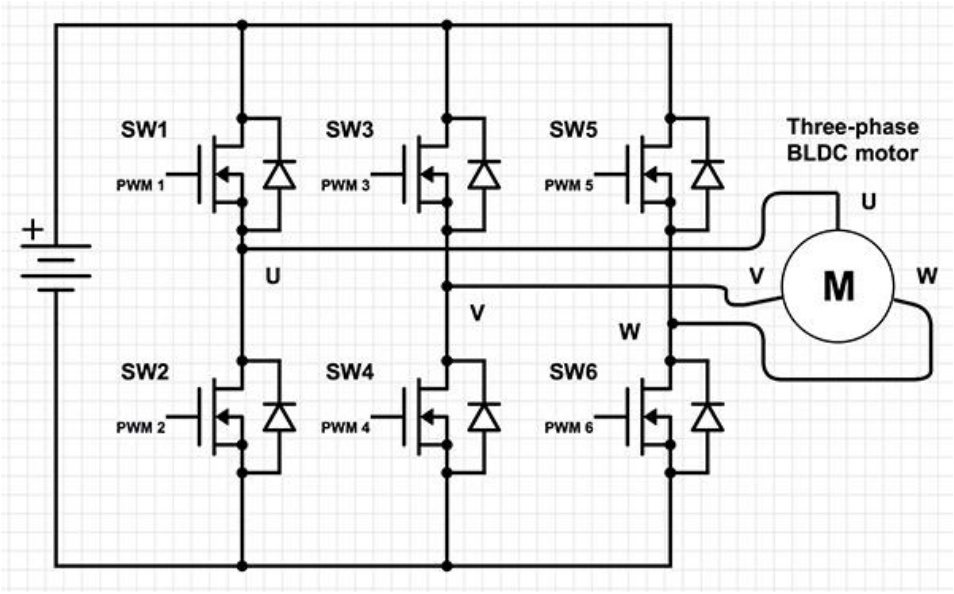

Figure 3: Triple Half-H Bridge Driver for 3-Phase Commutation

The most common architecture for a BLDC motor uses three phases, each consisting of two coils, and is driven by a circuit featuring three half-H bridges. This results in a six-step electrical commutation sequence to pass current through the coils. Only two phases are active at any time, with current entering one coil from the high voltage side and exiting another coil through the low voltage side. Current flow through the coils creates a magnetic field; the field's polarity depends on the direction of the current flow. This magnetic field interacts with the magnetic fields of the permanent magnets in the rotor, and the commutation sequence serves to pull the rotor in a circular motion.

The driving circuitry is usually controlled by a microcontroller, FPGA, or ASIC. The motor can be driven most simply using open-loop control, where instructions from the controller to the driving circuit do not rely on any feedback from the motor. This configuration is inefficient because it does not rely on feedback, thus increasing current draw and torque ripple.

Closed-loop control, however, does rely on feedback from the motor to decide when to move to the next step in the commutation sequence, increasing efficiency.

The basic implementation of closed-loop control uses a Hall sensor or similar to sense the rotor's position, but requires more parts and is less reliable than the preferred sensorless closed-loop control scheme. Sensorless control is normally implemented using observation of BEMF. In a BLDC motor having three phases, two of the phases are active at any one time, leaving one phase inactive. Due to the movement of magnets in the rotor, a voltage is induced in the inactive coil in such a way as to resist the rotation of the motor (by Lenz's Law). This voltage can be observed by the control circuitry to infer commutation timing [5]. Other methods of sensorless control do exist, but this thesis focuses on BEMF sensing.

The strength and frequency of the BEMF signal are directly related to the speed of the rotor, so most ESCs rely on both open-loop and closed-loop sensorless control. An open-loop algorithm is used to start up the motor while the BEMF is too weak to be sensed by the control circuitry at low motor speeds. Once the motor has achieved sufficient speed the controller switches to the closed-loop algorithm.

1.3: Methods of Back EMF Sensing

Several methods can be used to analyze a BEMF waveform to determine commutation points.

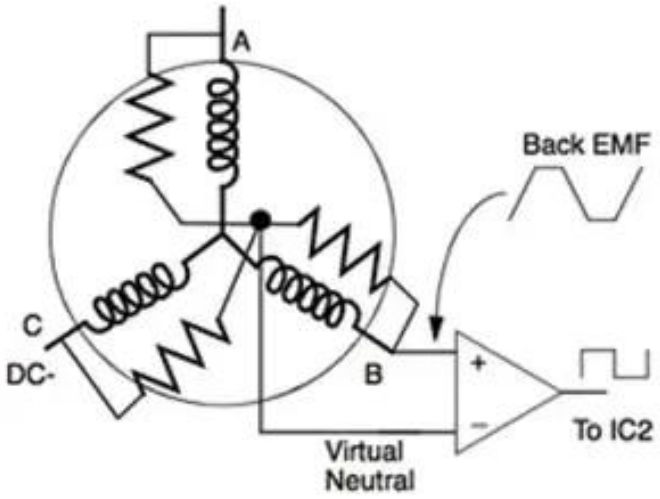

Figure 4: Zero-Crossing Method

The simplest is called the zero-crossing method. This method involves monitoring the BEMF waveform of the non-energized phase. It is compared to the voltage of a virtual neutral point, which is created by connecting the inputs of each motor phase through resistors. The point of interest is when the polarity of the observed phase changes relative to the virtual neutral point, which is related to the actual ideal commutation point by a phase shift of 30 degrees [5]. The comparison can make use of either a comparator or an ADC, both of which are usually available inside a microcontroller. This method requires high motor speeds, is sensitive to noise, and suffers from high common-mode voltages [6]. Since it is the simplest method of BEMF sensing and the PCB motor is only intended to be run at high speeds it is the first method that will be attempted to implement sensorless control for the improved PCB motor.

Other sensing methods that can be explored as necessary include the integration of the BEMF signal and observation of the BEMF waveform's third harmonic. Both methods are more complicated to implement than the zero-crossing method, but do not suffer as significantly from that method's drawbacks and do not require a phase shift to the correct commutation point [6].

1.4: PCB Coil Winding Theory

The number, arrangement, dimensions, and shape of the coils of wire that make a brushless motor function are critical design parameters. Any motor with coils arranged in a circular shape and activated sequentially to pull on the permanent magnets in the rotor is likely to achieve rotation. However, non-optimal designs will have low torque, significant torque ripple, and draw excess current. All of these factors detract from the efficiency of the motor. A well-designed coil will also produce a cleaner BEMF waveform than a sub-optimal coil design, which will be critical to achieving sensorless control with minimum additional componentry.

Exploration of the research into PCB motor coil design shows many examples of theoretical and experimental work. The applications of the documented motors are different from the application of this motor, but the theory behind coil design still applies.

The choice of the coil's shape is important because not all of the coil contributes to the motor's rotation. Only the segments of the coil roughly parallel to the radius of the motor are active or generating a useful magnetic field. The ends of the coil (roughly tangential to the circumference of the motor) are inactive. The challenge is to choose a shape that maximizes the active coil length while minimizing the inactive length to achieve the best motor efficiency [7]. Although a roughly trapezoidal coil shape is a popular design due to its efficient use of the stator

area, it does have a relatively long inactive trace length for each turn. Rhomboidal or hexagonal designs are a good alternative, significantly decreasing the inactive trace length while only adding some non-ideality to the active trace length since parts of the active length are not exactly radial [8].

The fundamental mathematics of a motor state that the BEMF generated in the coil is related to the change in flux-linkage that the coil experiences. Flux linkage, simplistically, is the product of the magnetic flux from the permanent magnet through the area contained by the coil, multiplied by the number of turns in the coil [9]. Thus, more turns lead to a greater flux linkage, which generates a larger BEMF that can be observed to control the motor.

In an ideal system, the motor's flux-linkage could be maximized by packing as many turns as possible into each coil. However, a realistic system suffers from the copper's resistance as current travels through the traces, which generates significant heat [7]. Additionally, since every turn of the coil cannot follow the same path, the required spiral shape means that each additional coil encloses less area and contributes less flux linkage to the motor. So, choosing the number of turns becomes an optimization problem that balances maximizing the flux linkage of the motor without causing unacceptable levels of copper loss.

Chapter 2

Motor Design

The goal of the motor design for this thesis is to improve the characteristics of the motor designed by Carl Bugeja. The part of the motor that involves the most intensive design work is the stator, which is made from a PCB. Improving on the design of the stator offers the best opportunity to optimize the performance of the motor. Bugeja's stator design features coils wound in a circular shape and packs in as many turns as possible to improve performance [1].

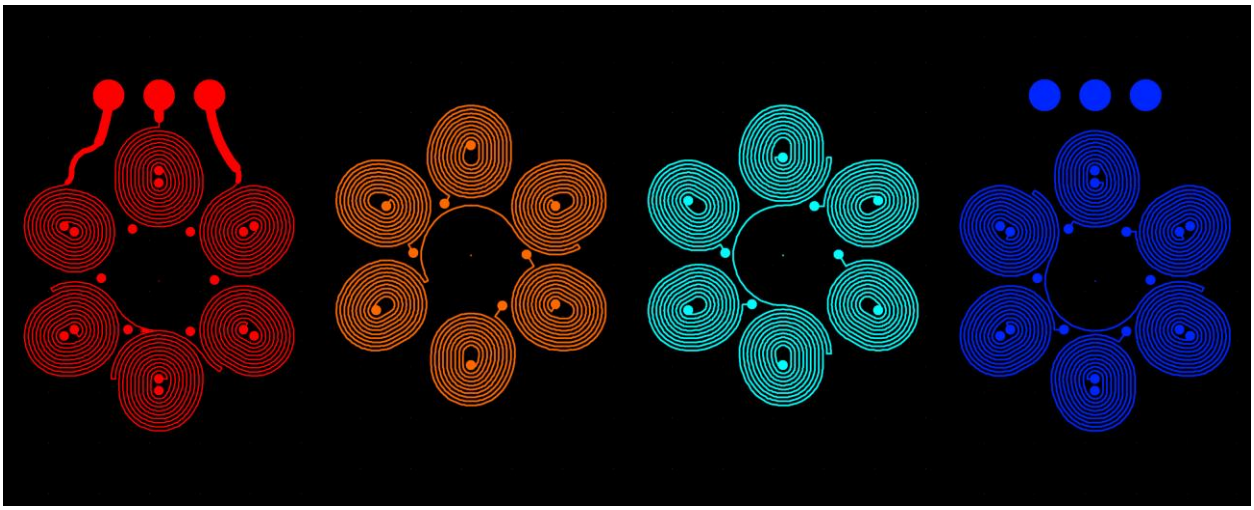

Figure 5: Bugeja's Coil Design

Bugeja's design was successful by several measures: it is simple to manufacture and assemble, its driving circuitry is easy to implement, and can spin at several thousand revolutions per minute (rpm) with open-loop control. However, the motor was not able to generate enough BEMF to implement sensorless control and its torque output is insufficient to turn a propeller fast enough to generate enough lift for a miniature drone [2]. The motor designed for this thesis will ideally achieve higher loaded speeds and generate sufficient BEMF to implement sensorless control.

2.1: Design Constraints

Before any design work occurred, the motor was constrained to only a few variables to make experimentation manageable. First, since the improved motor will be compared to Bugeja's original design, its stator had to have similar characteristics to his. The improved stator, therefore, had the same mechanical dimensions, layer count, phase arrangement, and trace dimensions. These values are listed in table form below. Beyond limiting the number of variables to manipulate, Bugeja's design characteristics worked with the well understood and documented control scheme for three-phase brushless motors. The improved motor also used Bugeja's rotor design.

Table 1: Fixed Variables for Stator Design

Characteristic	Specification
Outer Diameter	16 mm
Inner Diameter	4 mm
Phases	3
Coils per phase	2
PCB layers	4
Trace width	4 mil
Trace spacing	4 mil

2.2: Variable Manipulation

The constraints placed on the design left only two variables to manipulate. The first was the shape of the coil. In theory, the performance of Bugeja's motor was hampered because of the shape he chose for the motor coils. Examination of engineering literature did not reveal any examples of a circular coil shape being the best suited for a powerful axial-flux brushless motor. The literature typically identified a trapezoidal shape as most common and suitable, with

rhomboidal and hexagonal shapes as viable alternatives [7], [8]. An explanation of the theory behind the coil shapes was given in the previous chapter.

A hexagonal coil shape was chosen for the prototype of the improved motor because it significantly decreased the length of lossy inactive traces while only marginally decreasing the area that contributed to flux linkage relative to trapezoidal coils [8]. The specifications of the outermost hexagon in the coil are listed below. The numbers were calculated based on the physical parameters of the stator listed previously.

Table 2: Outermost Hexagon Specifications

Characteristic	Specification
Inner Radius (R_i)	2.6 mm
Outer Radius (R_o)	7.6 mm
Center Radius (R_c)	5.1 mm
Knee Angle (k)	30 Degrees
Coil Allotment (A_l)	57 Degrees

An image of the nested hexagons used in the Computer-Aided Design of the hexagonal coil is shown below to help visualize the hexagon specifications. The upper end of the dashed major axis line represents the stator's center. The Inner, Outer, and Center Radii are measured from that point. The Knee Angle is the angle of the diagonal lines relative to the major axis. The stator design called for six coils, meaning each Coil Allotment could ideally be 60 degrees of the stator surface. However, the coils must be separated by a distance greater than the manufacturer's copper-to-copper clearance specification, so 57 degrees was allotted for each coil.

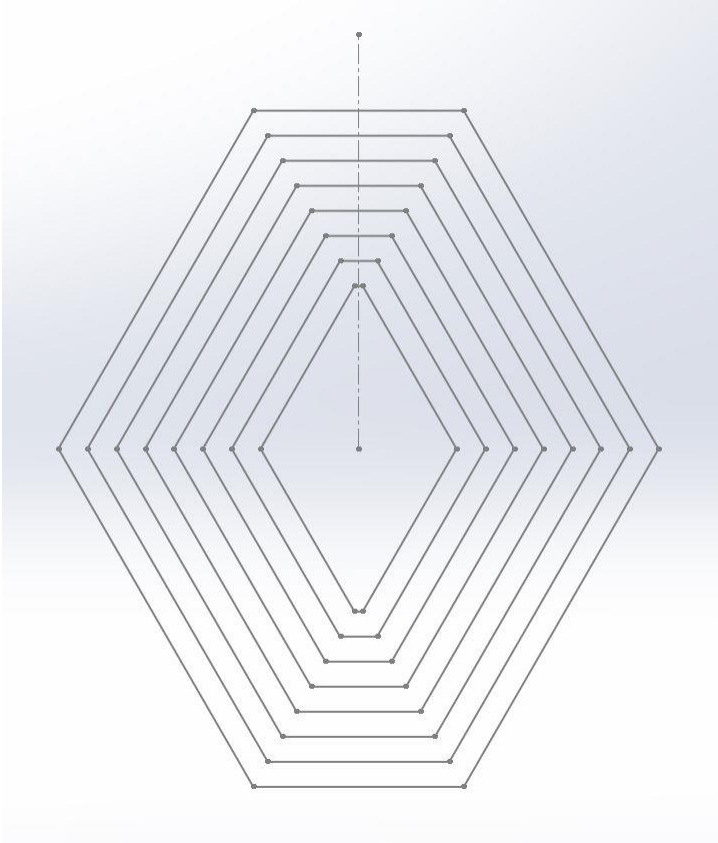

Figure 6: Nested Hexagons

The remaining variable to manipulate was the number of turns in each coil. More turns per coil results in a larger flux linkage. This creates a larger BEMF, which makes sensorless control easier to implement. Increasing the number of turns per coil also increases the coil resistance, which causes energy to be dissipated as heat. Thus, choosing the number of turns per coil became an optimization problem. The ideal number was that where further increases in flux linkage from an extra turn did not justify the increase in coil resistance. This applies no matter the coil shape.

The quantities relevant to calculating the motor's flux linkage and winding resistance were the total area and perimeter of each coil. The area was proportional to flux linkage and the perimeter was proportional to winding resistance. The area and perimeter could be quickly and

simply estimated by treating each turn of the coil as a closed hexagon. Since the coil dimensions were chosen to make use of all of the area available on the PCB (improving the performance of the motor), an unfortunate trade-off is that the resulting coils were not shaped like regular hexagons. This made the calculations of area and perimeter slightly more involved than they would be otherwise. The formulas are written in table form below and reference the hexagon characteristics defined previously.

Table 3: Area and Perimeter Formulas

Description	Formula
Hexagon Height (l_i)	R_o - R_i
Hexagon Width (g_i)	$2\ \pi\ R_c\ \frac{A_l}{360}$
Rectangle Area (R_A)	$g_i\ l_i$
Triangle Area (T_A)	$\frac{l_i^2}{2}\tan(k)$
Hexagon Area	$R_A - 4\ (T_A)$
Diagonal Length (D_L)	$\frac{l_i}{2\cos(k)}$
Horizontal Length (H_L)	$g_i - l_i \tan(k)$
Total Perimeter	$4\ (D_L) + 2\ (H_L)$

The area formula found the area of the smallest rectangle that encompassed the hexagon and subtracted the area of the four triangles inside the rectangle but outside the hexagon. The perimeter formula used basic trigonometry to calculate the length of one horizontal side and one diagonal side and added them together the appropriate number of times to find the total perimeter of the hexagon. Both formulas relied on the width and height of the hexagon. The width and height change predictably based on the trace width and spacing defined for the design, so it was simple to calculate the area and perimeter of each hexagon in the coil. The resulting area and

perimeter for each hexagon were added to the total area and perimeter numbers. Spreadsheets and Python scripts helped to run the calculations quickly.

The resistance of the winding was easily calculated from the coil perimeter using the basic formula for resistance.

$$R = \frac{\rho A}{L}$$

Where ρ is the resistivity of copper, A is the cross-sectional area of the copper traces, and L is the total length (perimeter) of the windings.

2.3: Optimization of Number of Turns

The initial effort to analyze the flux linkage (area) and resistance (perimeter) values as a function of the number of turns attempted to calculate a value proportional to "k_p," which was defined as the "generated torque per copper loss" [8].

$$k_p = \frac{k_T}{\sqrt{R_{ph}}}$$

Where k_T is the motor's torque constant and R_{ph} is the motor's phase resistance. Since k_p is a measure of the motor's performance, logic dictates that turns should be added as long as the value of k_p increases. Phase resistance was easily calculated, and the torque constant k_T was assumed to be the product of the coil area and current, which was calculated based on the phase resistance. A second look at the literature showed this assumption to be false, but the stator PCB

had already been designed and ordered. The erroneous assumption showed that the value of k_p stopped increasing with the eighth turn, so each coil on the PCB was designed with eight turns.

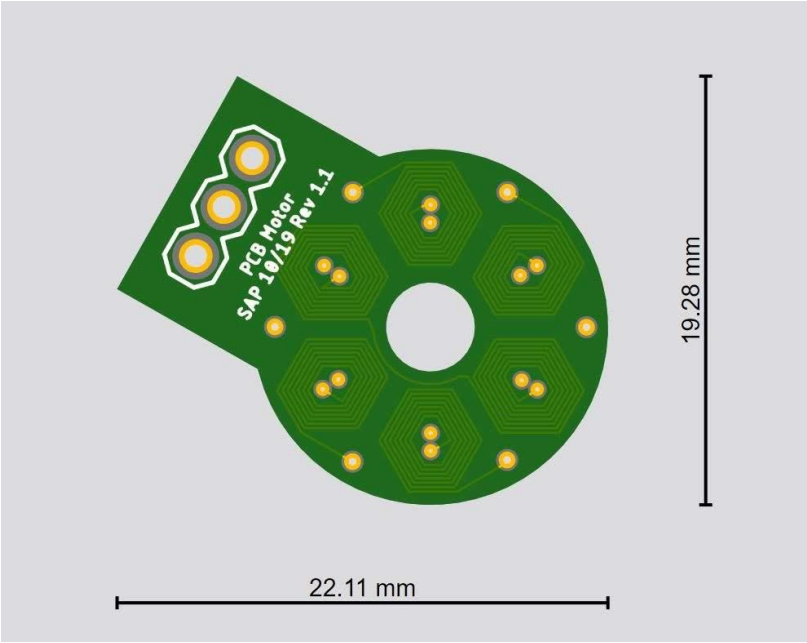

Figure 7: Rendering of the Optimized Stator Design

Despite the mathematical error, the design remained unchanged in order to focus on the ESC design. The hexagonal coil shape alone can be tested to see if it improves the motor's performance relative to the original motor. Additionally, eight turns happened to be nearly the maximum number of turns that were possible with the hexagonal coil design, so it was impractical to try to add more. Also, the coil resistance of the hexagonal motor was measured to be significantly lower than in Bugeja's motor, so heating will already be a problem for the motor. Fewer coils would make this problem worse.

2.4: Motor Assembly

The same mechanical design was used for both this thesis's motor and Bugeja's motor. It is designed to be easily assembled without the need for special tools. Parts such as the magnets, headers, and bearing are all available to consumers. The custom stator and rotor can easily be ordered by a hobbyist who has access to Computer-Aided Design software.

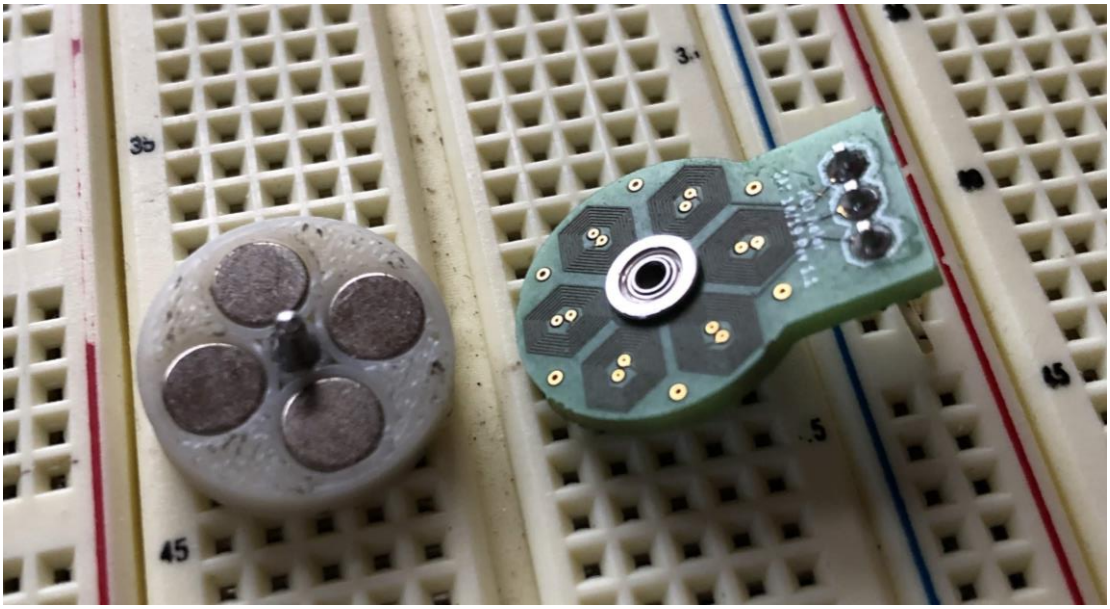

Figure 8: Motor Assembly

To begin assembly, a three-pin male header is soldered to the stator PCB and is used to connect the motor to its driving circuitry. The stator PCB is connected to the rotor via a shaft that fits into a bearing press-fit into the center of the stator. The opposite end of the shaft is press-fit into the center of a 3D-printed rotor. The rotor itself has four circular indentations spaced equally around its underside. These indentations accept circular rare-earth magnets, which are arranged with alternating polarity. When assembled, the motor has a diameter of 16mm and a height of approximately 6mm. The hexagonal motor design is pictured.

Chapter 3

Electronic Speed Controller Hardware Design

A brushless motor has no electrical connection between its rotor and stator, so an Electronic Speed Controller is required to compel the motor to spin. As was explained previously, the driving circuitry in the ESC uses logic to manipulate a set of transistor H-bridges that pass current through the phases of the motor to create rotation in a six-step commutation cycle. Each full commutation cycle corresponds to one-half turn of the rotor.

After the PCB motor prototype was designed and fabricated, a series of prototypes of the ESC driving hardware and the logic software behind it were developed to test the motor and observe its performance relative to Bugeja's control motor. Both the circuit and code prototypes started very simply and advanced with each iteration. The goal is to eventually have a motor that can be controlled by a user, start independently, and run efficiently using closed-loop control.

Consideration was given to the experimental nature of the project and the desire for speedy and cost-effective development. Therefore, the circuitry was designed to be modular to allow parts such as the motor PCB and BEMF observation circuitry to be swapped and updated without affecting other parts of the circuitry. Once a viable design is observed the design can then be packaged into a very small PCB that might be suitable for mounting on a drone. This section focuses on hardware development.

3.1: First Hardware Prototype

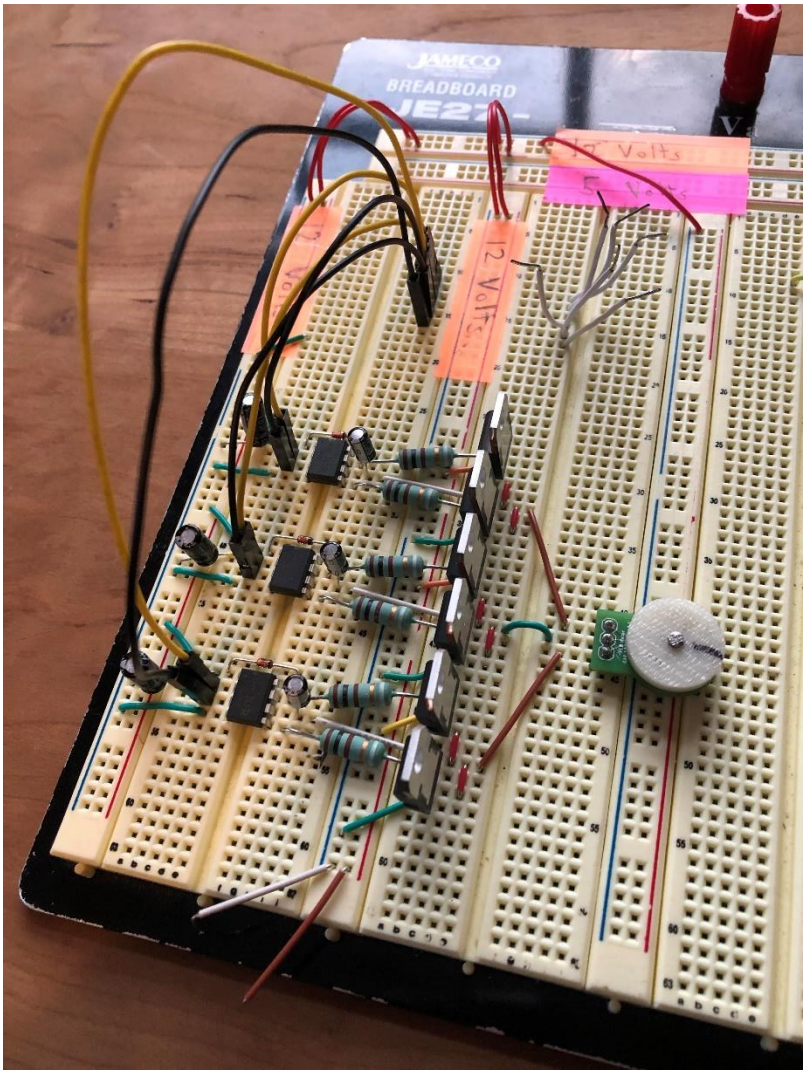

Figure 9: ESC Prototype 1

In the interest of validating the PCB motor without a large investment in time or money, the first prototype motor controller was based around an open-source ESC from a maker called ElectroNoobs [10]. His controller was designed for radial-flux BLDC motors typically found on drones and was capable of open-loop motor starting and sensorless closed-loop control using an ATmega328 microcontroller. In his version, everything was implemented on a PCB. The design

was appealing because it was simple, open-source, and known to be capable of driving a BLDC motor. Also, the ATmega328 is a familiar microcontroller that is compatible with the Arduino development environment, making it easy to understand ElectoNoobs' code and write custom software.

Significant changes were made as to how the circuit was implemented to speed up development. First, since all of the major components were available as through-hole packages, it was very simple to wire everything on a breadboard. This decision meant that no time was needed to create a design and review a PCB in Computer-Aided Design software, eliminated the lead time related to having a PCB fabricated, and allowed for any necessary wiring corrections or additions to be made immediately at no cost.

The first prototype of the controller was built very quickly and used only the minimum components needed to drive the motor. Further simplifying the prototype was the replacement of the bare ATmega328 microcontroller with an Arduino Nano board. The Arduino Nano uses the ATmega328 but also implements power regulation circuitry, a serial programmer, and comes with the Arduino bootloader installed. So, no finicky breadboarding of the bare microcontroller chip and its associated components were required. Also, there was no need to build a method of programming the chip, since the Arduino Nano comes with a Mini USB connector that interfaces with a host computer.

A downside to the design of this controller was that it required both 12 V and 5 V power. 5 V was used for the logic of the H-Bridge and was compatible with the Arduino. 12 V was used to supply the H-Bridge and transistors that guide current through the motor. It was simple to implement a dual power supply architecture in prototype form because a benchtop power supply was easily accessible. However, such architecture was not viable for later iterations of the ESC,

since the intended application is a miniature drone that should be based around 5 V only for simplicity. This design also used very large transistors capable of carrying several amps of current at 12 V, which was an amount of power far greater than what the PCB motor requires.

With the ESC built on the breadboard a very simple commutation sequence was written in Arduino code. Running that code on the Arduino Nano to control the H-Bridges proved that both Bugeja's and the newly designed motor were capable of rotation. They each required a manual kick start, the rotation was slow, and excessive heat built up very quickly in the motors. Still, building such simple hardware and software prototypes allowed for the design of more permanent and sophisticated ESCs to move forward with confidence.

3.2: Second Hardware Prototype

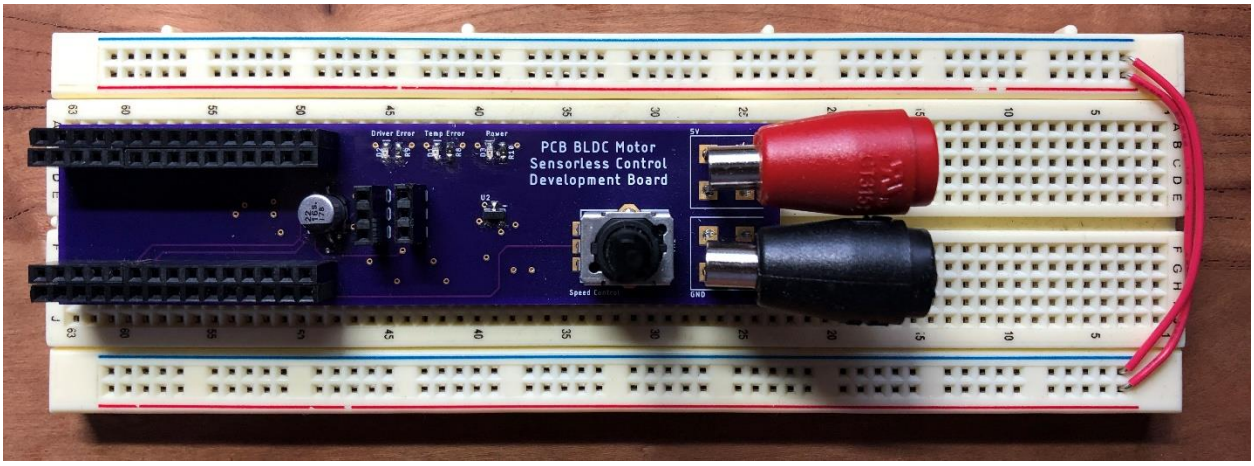

Figure 10: Sensorless Control Development Board, Top Side

The second prototype of the motor controller hardware called the "PCB BLDC Motor Sensorless Control Development Board," was designed to be a universal platform for the remaining thesis experiments. It is modular, so different components can be added to the base

circuit board the design develops. The development board consists of the parts of the ESC that are not anticipated to change throughout the design process. Specifically, the motor driving circuitry, overtemperature protection circuitry, a benchtop power supply connection, and a potentiometer for speed control are implemented permanently.

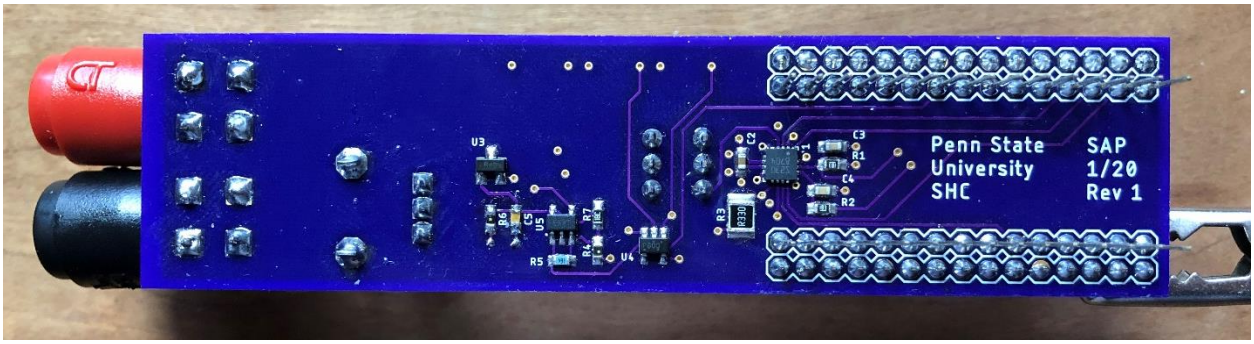

Figure 11: Sensorless Control Development Board, Bottom Side

Connectors facilitate the modular nature of the board, allowing interface with peripheral devices that may change, such as the microcontroller, motor, and closed-loop control or amplification circuitry. The development board was designed to mount to a standard breadboard so additional circuitry can be prototyped quickly and at little cost. The breadboard connection has the additional benefit of providing a wide and stable base of support and protecting the tiny components soldered to the underside of the board.

The Integrated Circuits (ICs) and MOSFETs used in the previous prototype were not viable for the second prototype due to size and voltage constraints. Searches for a BLDC motor controller that uses 5 V for both logic and driving and is available in a small surface-mount package led to the same IC that was used by Bugeja. It seems to be the best option for space-constrained low-power applications. It features integrated protection circuitry against short circuits.

Temperature protection circuitry was implemented on the development board because of experience from the first hardware prototype, which generated so much heat in the stator PCB that it could be dangerous for the stator if the circuit was left on for too long. The output of an analog temperature sensor was fed to a comparator circuit that triggered an error if the temperature of the stator got too high and used hysteresis to hold the error until the temperature fell below an acceptable level.

The error signals from both the motor driver IC and the temperature sensor were broken out to red LEDs on the top of the circuit board for visual error indication. The two error signals were ORed together and fed to the microcontroller so the software could be aware of a problem immediately.

The development board was compatible with the Arduino Nano series of microcontrollers (MCU). The Nano was chosen because it was extremely easy to interface with, just like any Arduino board, and it was small enough that its footprint fit within that of the development board. The electrical connections between the circuitry built into the development board and the specific pins of the Nano connector were chosen very intentionally due to the port structure of the Nano [11].

Table 4: Arduino Nano Port Functions

Port	Basic Function	Additional Function
B	Digital I/O	PWM, Internal Comparator, External Interrupts
C	Analog I / Digital I/O	Analog to Digital Converter
D	Digital I/O	PWM

The Arduino Nano has three accessible Input/Output (I/O) ports. Ports B and D correspond to digital-only I/O. Some of these pins are capable of Pulse Width Modulation

(PWM). Port D also contains the two external interrupt pins and internal comparator pins available on the Nano. Port C is connected to the ATmega328's Analog to Digital Converter (ADC) [11].

The motor driver required six inputs, three low and three high, that control the transistors in the H-Bridge that drive the three-phase motor. It was desirable to control all the inputs using only a single code instruction so that current changes within the motor happen essentially simultaneously. All six inputs could be put on one port to accomplish this. However, having all the inputs on one port would limit the amount of experimentation that is possible.

Some motor drivers use a PWM signal on the high or low (or both) input signals to the motor driver to modify the motor's performance by essentially decreasing the applied voltage to the motor. Such a control scheme is not currently part of the design plan, but it should be an option in the future. So, the three high inputs were put on Port B pins capable of PWM, and the low inputs were put on Port C pins that can be operated as basic digital I/O. Four pins of Port C were still available to be used as inputs to the ATmega328's ADC, which could useful for implementing closed-loop control. Port D was not used to connect to the motor inputs because it has interrupts and a comparator could also be useful for closed-loop control.

The most important feature of the development board was that all the pins of the motor and the microcontroller are broken out to separate headers. This allowed experimental shield boards to be plugged into the development board that carried additional circuitry necessary for the sensorless closed-loop control of the motor. The shield boards could take input from the motor and the MCU directly, manipulate it, and then pass information back to the microcontroller. The shield format saved money and time since the basic driving circuitry was unlikely to change. Only the small shield boards needed to be designed, ordered, and built.

The development board paired with an MCU alone was capable of open-loop motor control, and its first purpose was the development of a control algorithm to get the motor spinning more quickly and more efficiently than was possible with the first hardware prototype.

3.3: First Sensorless Control Shield

The first Sensorless Control Shield was an exploratory initial step to develop the circuitry necessary to implement sensorless control for the PCB motor. This shield was a proof-of-concept, demonstrating that such a shield was compatible with the development board system. The shield was designed to connect to the extra headers broken out from all of the pins of the Arduino Nano and the three phases of the motor.

Figure 12: Motor Control Shield 1

This first shield had very simple onboard circuitry and many test points. It was designed with the most common method of BEMF sensing in mind, called zero-crossing detection, which was explained in Chapter 1. Zero-crossing detection derives a virtual neutral point relative to the

three motor phases and then compares the actual voltage of each phase to that virtual neutral point. The change in polarity of the phase relative to the neutral point allows the commutation timing to be inferred. The shield created a neutral point by connecting the three phases through a high-value resistor. Comparison was done by feeding the neutral point and each phase to the inputs of three comparators. Three comparators were used for simplicity in this design, but a production system would likely use one comparator and multiplex the three phases to a single input. The three phases, the neutral point, the comparator outputs, and multiple ground connections were broken out to test points to allow easy and reliable connection to an oscilloscope for observation.

3.4: Second Sensorless Control Shield

The second Sensorless Control Shield was designed to test a possible configuration of the BEMF amplification circuitry. It used the same mechanical structure as the first shield, connecting to all of the microcontroller and motor pins. It also created a virtual neutral point relative to the three phases of the motor.

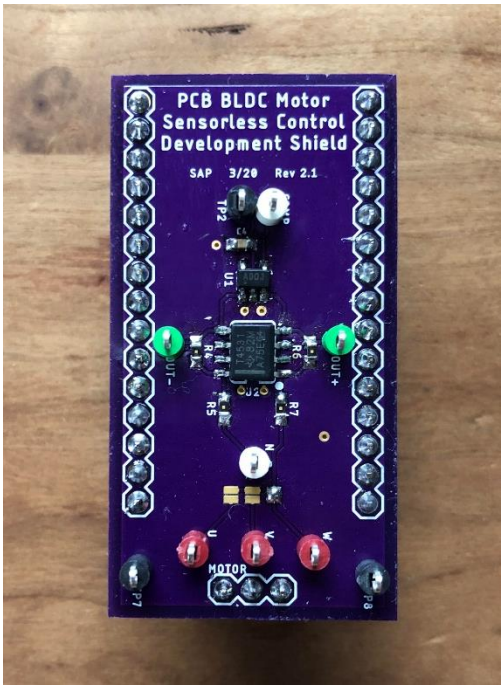

Figure 13: Motor Control Shield 2

The zero-crossing method of sensorless control observes the polarity of the BEMF signal relative to the virtual neutral point, where the change in polarity is called the zero-crossing point. The magnitude of the BEMF, and therefore the ease of observing its zero-crossing point, is directly related to the speed and strength of the motor. A PCB motor as small as those observed in this thesis was a relatively weak motor, so even though it could be spun at several thousand rpm the generated BEMF was still very weak and needed to be amplified for sensing to be viable.

The specific quantity of interest to the designer of this amplification system was the signed difference between the neutral point voltage and the phase BEMF being observed. A comparator such as the one used in the previous shield design could be used to detect the polarity of the two signals relative to each other. However, the unamplified signal had such a small peak-to-peak magnitude that noise in the system could easily be misinterpreted as a polarity change. So, before being fed to the comparator the difference between the two signals should amplified to make the true zero-crossing point easier to observe.

For a discrete circuit designer, operational amplifiers (op-amps) provide the simplest method of constructing analog amplifiers. The typical op-amp is a differential input, single output device whose output when amplifying alternating voltages swings about the system ground. There are two reasons that this simple system was not viable for this particular development board environment. First, only a single power supply was available to the system. To get symmetric swing about the system ground from the amplifier output, a dual power supply architecture was required. The second issue was the output voltage needed to have a common-mode component approximately equal to that of the virtual neutral point so that the swing of the amplified BEMF could the neutral point. The DC component of the input BEMF signal was already approximately 2.5 V, and it can be difficult to amplify the AC voltage without affecting the DC voltage in a standard op-amp. If the amplifier circuit had any DC gain the resulting output could be offset from the neutral point such that no zero-crossing occurs.

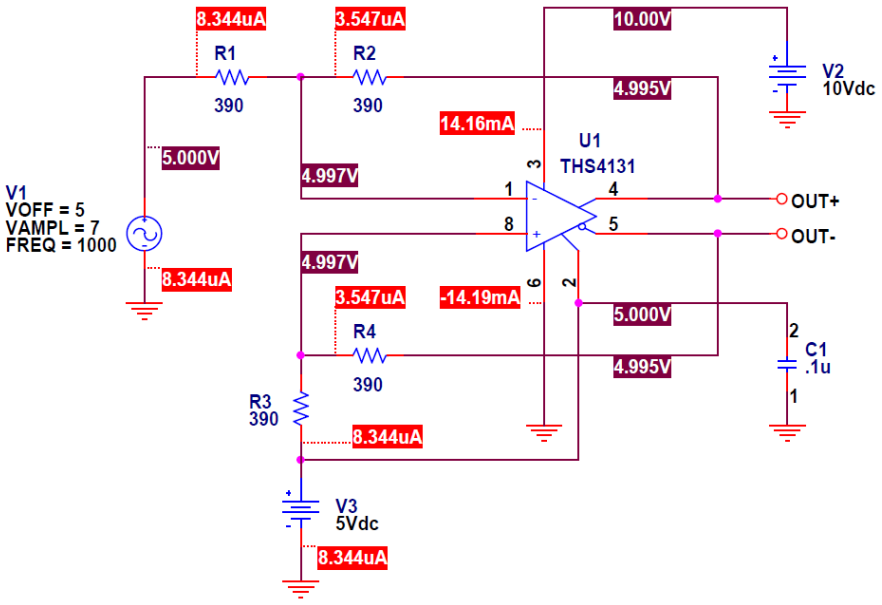

Figure 14: Differential Amplifier with Common-Mode Voltage

A solution to these issues was to use a purely differential op-amp, whose inputs and outputs were both differential. Thus, the differential inputs were the two quantities whose difference is of interest - the motor phase BEMF and the neutral point. The outputs were amplified versions of those inputs which were fed directly to the comparator. Such devices can be used with single power supply architectures because they often include a common-mode voltage input, which sets the DC value of the output that the AC outputs swing around.

Application notes from Texas Instruments were found to help correctly design the circuitry around the fully differential op-amp IC [12]. The gain of the op-amp was set by two sets of two resistors that provide negative feedback between output and input. The desired common-mode output voltage was set to 2.5 V which was half of the supply. In addition to the virtual neutral point that was present on the first shield, the new shield also allowed any of the three motor phases to be mapped to the input of the op-amp, broke out the op-amp's differential

outputs and the comparator's output to test points, and the comparator output was supplied to one of the microcontroller's external interrupt pins for future use in closed-loop control.

3.5: Complete Development System

The complete system of the development board, Arduino Nano, a PCB motor, and shield prototypes allows the user to spin the motor and observe its phase voltages in the interest of deciding if the zero-crossing method of BEMF sensing might be viable for closed-loop control. Further shields can be developed on that conclusion, whether or not such a method is found to be viable.

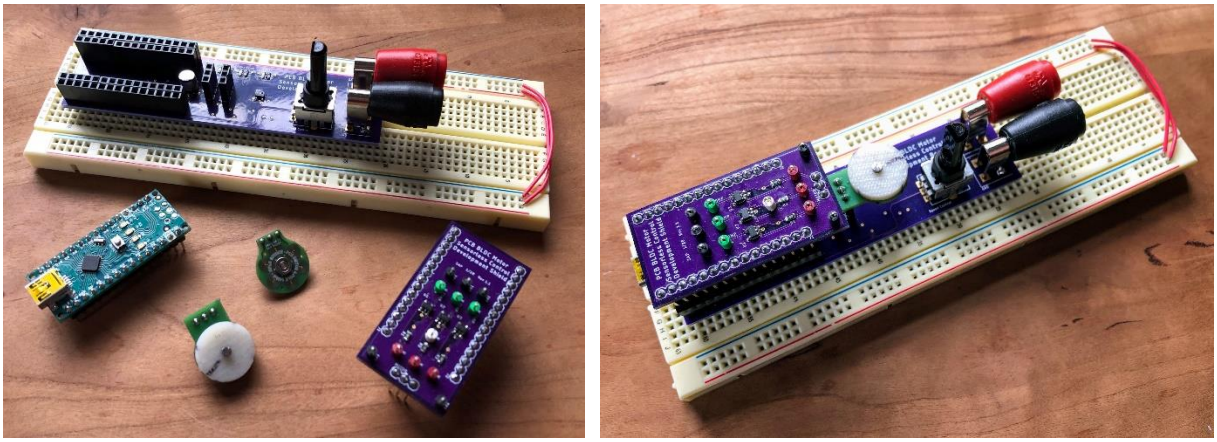

Figure 15: Complete Development System

Chapter 4

Electronic Speed Controller Software Design

The ESC hardware required software to dictate when the motor driver commutes the motor. The prototype software developed in a very continuous manner, with a basic building block being modified slightly and repeatedly to test and implement improvements. This differed dramatically from the hardware prototyping process, which was very discrete since individual circuit boards had to be designed and ordered. Therefore, this section describes the evolution of the software prototype and major design characteristics more than specific code instances.

4.1: Design Decisions

From the beginning, the development board's hardware and software were designed around the Arduino microcontroller and development environment. This simplified the hardware design because Arduino boards implement power, reset, I/O, and programming circuitry by default, greatly decreasing the amount of effort that goes into the design of a microcontroller-based system.

This decision also simplified the software design because Arduino boards can be programmed with Arduino's hardware-centered adaptation of the C programming language, which is written, compiled, and exported from a free and simple Integrated Development Environment. This makes programming the microcontroller quite simple by eliminating most of the tedious and confusing register manipulation normally required when programming an MCU.

4.2: Basic Building Block

The building block for the software was the most basic possible routine to achieve commutation. It consisted of code to define the direction of the I/O pins of the Arduino Nano, and then a six-step commutation sequence that set the state of the I/O pins to control the motor driver appropriately. Each step in the commutation sequence was separated by a set time delay. This building block was first used with the breadboarded hardware prototype to confirm that the PCB motor could spin. It was then used with slight modifications to confirm that the Sensorless Control Development Board had been designed and assembled correctly. After that, the basic code building block was modified to improve performance and add features.

4.3: Software Iterations

The initial code building block proved that the PCB motor was able to spin, but it also highlighted how important a quality acceleration algorithm would be to self-start the motor, bring it reliably to a steady-state speed, and reach a steady-state speed high enough to potentially observe BEMF. Since the rotor of the PCB motor had very little mass, acceleration could happen quite quickly in theory. However, writing the software to decrease the time delay between steps proved to be the greatest challenge of the thesis for this electrical engineer.

The first few iterations of the software attempted to implement an acceleration algorithm in stages that caused the step delay value to very roughly mimic an exponential decay function. The motor would start with a very large step delay that was decreased very quickly until a cutoff step delay was reached. Another stage of acceleration took over and decreased the step delay by a factor of 10 less per step than in stage one. Three stages were used in the last attempt at this

method, which accelerated the motor successfully but was very finicky and couldn't reach a very high speed.

The issue with this simple method was that changes to the step delay did not adapt continuously to the motor's speed. The faster the motor spun, the smaller the change in the step delay should be. The staged algorithm very roughly approximated this with three decrement values separated by a factor of 10, but the transition between stages was abrupt and the smallest decrement value could not accelerate the motor beyond a certain speed. The motor would frequently stop suddenly because the rotor could not maintain synchrony with the magnetic field produced by the stator.

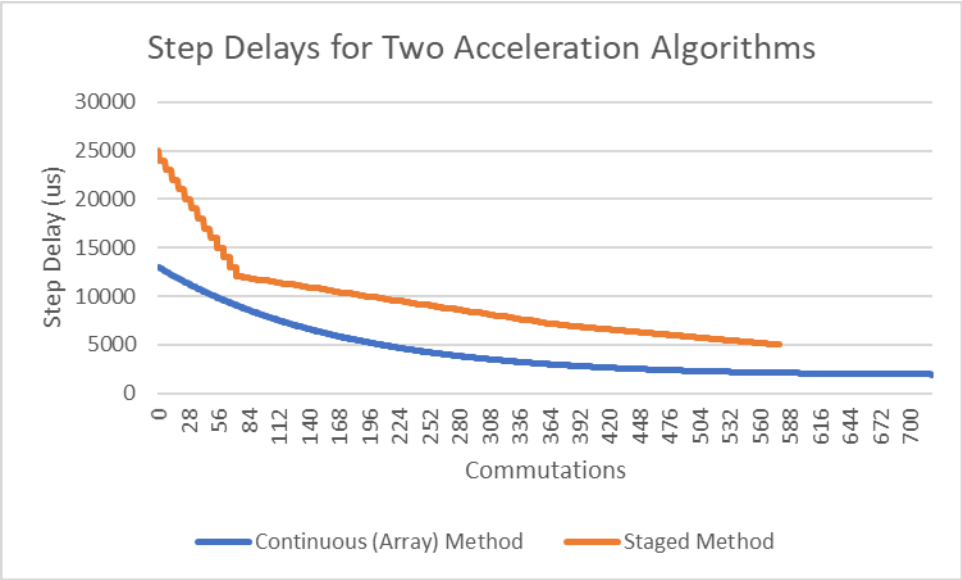

Figure 16: Graphical Comparison of Acceleration Algorithms

The obvious solution was to implement step delay values according to an actual exponential decay function into the acceleration algorithm. This could have been done by writing an exponential function directly into the software and calculating a new step delay number along with every step. However, since the time between steps is currently implemented with a delay

function rather than an interrupt, the concern was that the computational intensity of calculating values from an exponential function would cause unintended differences in the timing of commutations.

Instead, the values from the exponential function were calculated in Excel, formatted as an array, and put in the memory of the Arduino. The acceleration algorithm simply retrieved sequential delay values from memory with each commutation. The issue with this method was that storing a large array in the Arduino consumed most of its available memory, limiting the number of commutation steps that could be taken by the acceleration algorithm. As can be seen in the graph above, though, the values used in the array allowed the motor to accelerate smoothly and reliably and reach a higher speed than was possible with the staged algorithm. This last acceleration method was used to spin the motor so that results from the hardware could be observed.

Chapter 5

Results and Future Work

5.1: Motors

The two motors described in Chapter 2 were fabricated for use in this thesis. The original intent was to compare the performance of the two motors to see if the research-based hexagonal design improved on Bugeja's original design. Unfortunately, due to this engineer's error, the copy of Bugeja's motor was ordered with critical shorts across the entire bottom layer. Its performance was so degraded that it couldn't be tested on the hardware built for this thesis. The best comparison that could made was to Bugeja's descriptions of his results on YouTube and Hackaday.

The hexagonal motor achieved a top speed of approximately 2,100 rpm using the open-loop acceleration algorithm described previously. That algorithm started the motor consistently, accelerated it to its top speed within approximately three seconds, and could maintain the top speed successfully. This result was a great improvement over initial results and came out of significant iteration on the acceleration algorithm. Further improvements to the algorithm to improve the top speed were attempted, but none were reliable. The highest motor speed observed was 3,500 rpm. The process of modifying the acceleration algorithm (which involves generating values in Excel and loading them in an array format into Arduino) was very time-consuming. Unfortunately, the open-loop structure meant that iteration was the only way to improve the motor's acceleration, and it was found that iteration did not consistently result in improvements because small changes to the algorithm often damaged the motor's performance. Bugeja's motor

was able to achieve a top speed of about 14,000 rpm when equipped with a Hall sensor for closed-loop feedback [1]. The slow top speed of the hexagonal motor was likely due in part to shoddy software written by this electrical engineer. It is also likely that the motor's acceleration and top speed could be significantly improved if closed-loop Hall sensor feedback was implemented, having now observed the difficulty of open-loop acceleration. The current motor and ESC could not use closed-loop feedback for acceleration because the BEMF is far too weak at low motor speeds.

When the hexagonal motor was received from the fabricator its coil resistance was observed to be about 8 Ohms, which was less than half of what Bugeja observed on his original motor. This was probably because the hexagonal motor has fewer turns overall. As a result, the hexagonal motor drew 2.5 W from a 5 V power supply. Since the motor was unloaded and spun relatively slowly, most of that power dissipated as heat in the rotor and stator. This limited the amount of testing that could be done at one time because the motor had to cool down after about a minute of continuous use. Future revisions on the hexagonal motor will attempt to increase the coil resistance so it can spin for longer without overheating.

5.2: Hardware

The basic ESC board was functional immediately after being soldered. It allowed the motor, MCU, and shields to be swapped out as necessary, and the extra headers and connection to the breadboard made observing the operation of the motor and debugging very easy. The error detection features were never triggered, but they worked as intended when tested.

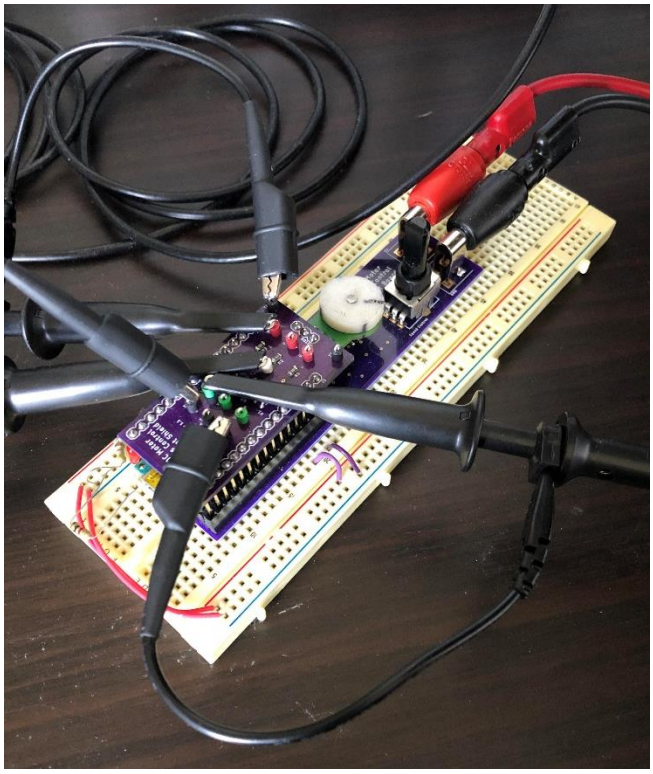

Figure 17: Typical Experimental Setup

The overtemperature detection circuit would have been very useful except the large air gap between the motor and the sensor was not accounted for, so the sensed temperature never reached the trigger threshold. Future revisions of this board will fix the error. The typical experimental setup, as shown above, looks a little messy but was very easy to work with. Oscilloscope probes connected directly to the signals of interest using test points.

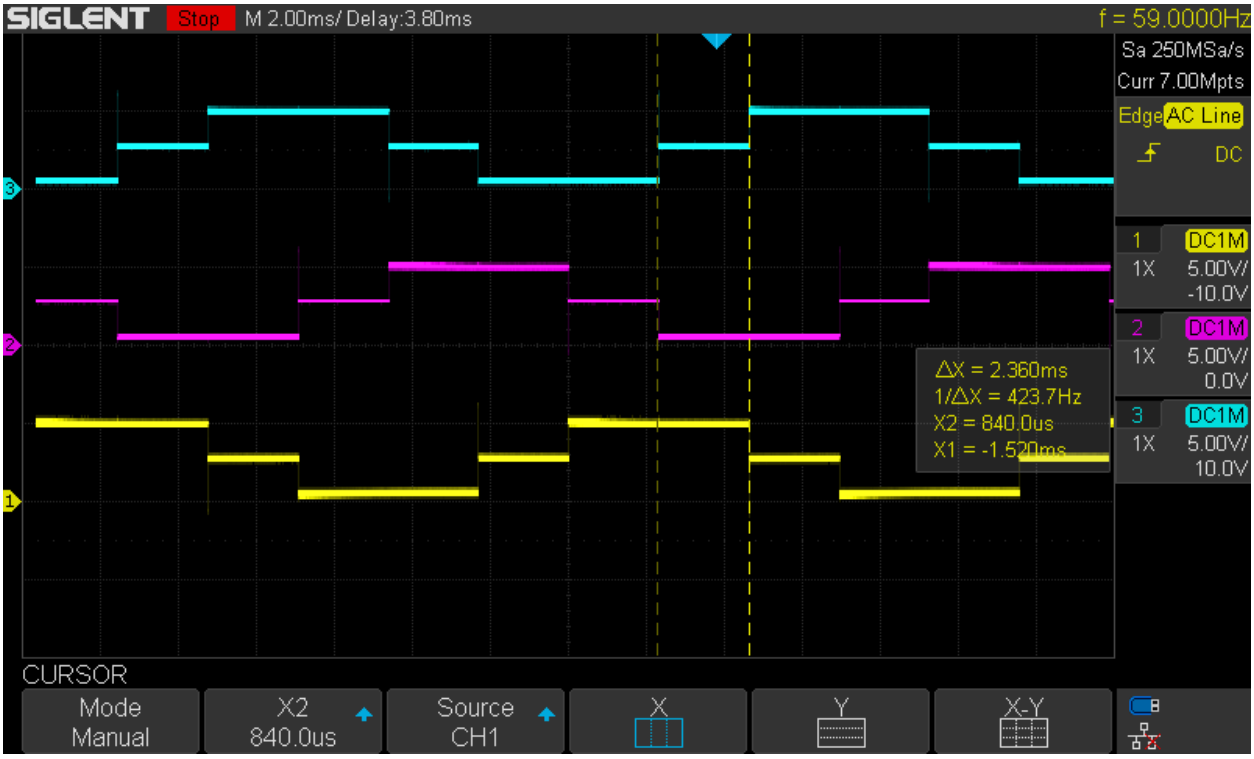

Figure 18: Motor Phases at 2,100 rpm

Observation of all the motor's phases using an oscilloscope clearly showed the six-step commutation sequence. Each of the six steps drives the three phases into a unique combination of high, floating, and low states. The waveform shown in Figure 18 corresponds to the motor spinning steadily at about 2,100 rpm. This speed, the highest that can be accelerated to reliably, was used to show the output of the two shield boards.

The purpose of the first sensorless control shield was to observe each of the three phases of the motor compared to a virtual neutral point, as would be done in the zero-crossing method of closed-loop control. The shield mapped each phase to a comparator for simplicity, but a real system would only map the phase of interest to a single comparator at any given time.

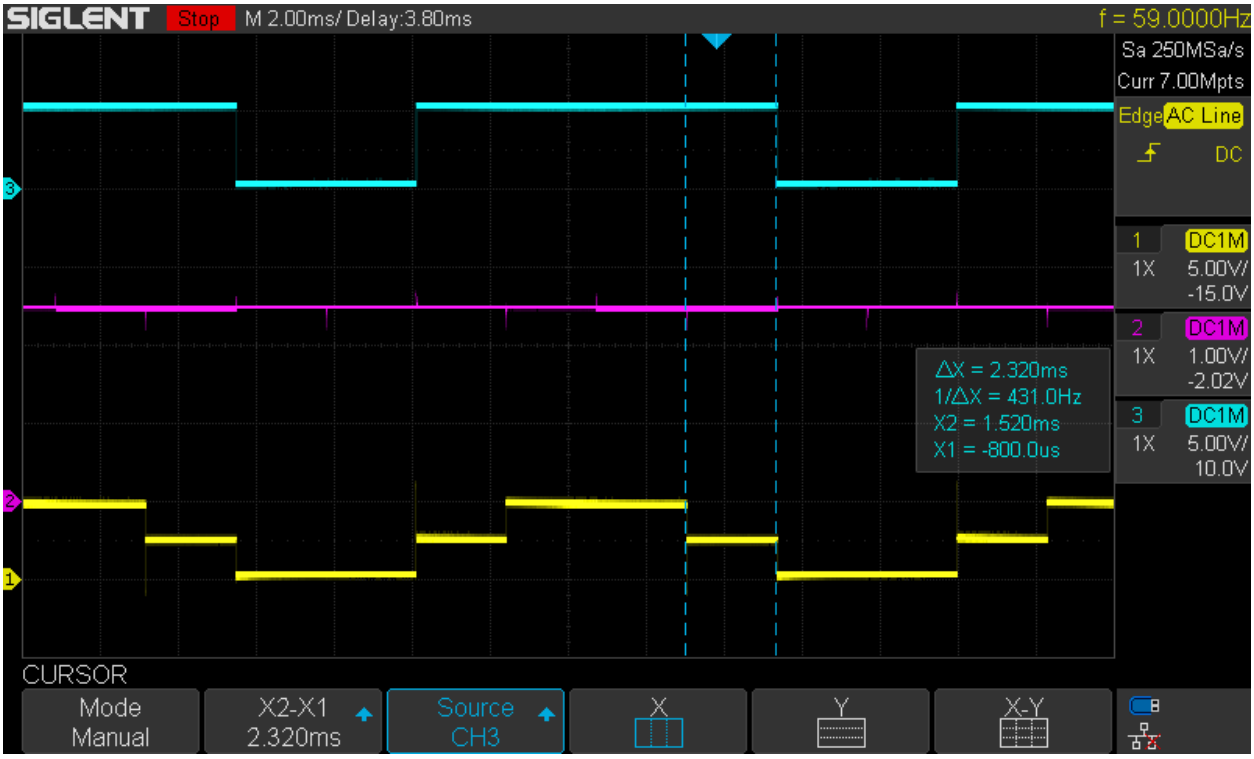

Figure 19: Phase, Neutral Point, and Comparator Outputs from Shield 1

The oscilloscope capture above shows one phase of the motor (yellow), the virtual neutral point (pink), and the output of the comparator (blue) connected to that phase. The neutral point rested steadily at 2.5 V. It showed some inductive spiking with each commutation, but that was not relevant for this shield. The comparator output worked well when the observed phase is driven high or low. The region of interest, however, was when the phase is left floating (highlighted by the cursors). The movement of magnets in the rotor should generate some BEMF that appears as a sloped voltage line in that region. Ideally, the voltage would cross the neutral point and the state of the comparator output would change inside that region of interest. This was not the case, however. The floating phase voltage was observed to be higher than the neutral point at about 2.7 V, and the slight slope of its graph was present whether or not the motor was spinning, so it did not indicate generated BEMF. Even if sufficient BEMF were generated it

would likely not be enough to cross the neutral voltage due to the voltage offset, so the comparator output would not switch.

The second sensorless control shield was designed based on the expectation that the motor would not generate sufficient BEMF to be observed, so it should be amplified. It featured a fully differential op-amp that amplified the difference between the phase voltage and the neutral voltage by a factor of ten.

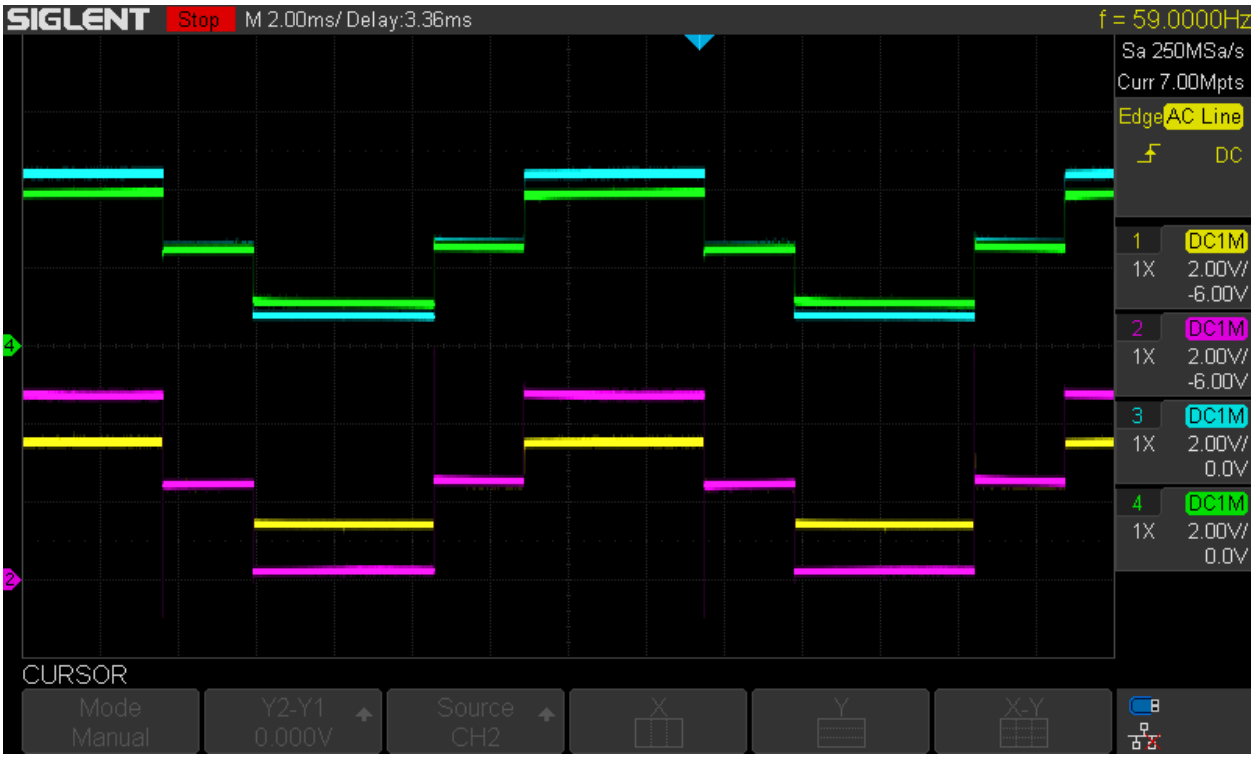

Figure 20: Amplifier Inputs and Outputs from Shield 2

The results of the test are shown in the scope capture above. The inputs to the op-amp were one of the motor phases (pink) and the neutral point (yellow). The blue output corresponds to the phase input and the green output corresponds to the neutral point, each with a gain of 10 V/V. The neutral voltage on this device was not similar to that on the first shield – it was an attenuated version of the observed phase voltage itself. This was because of the interaction of the

neutral point with the op-amp's feedback path. The disconnection of that path returned the neutral point to its expected state. Interactions with the feedback path were not considered in the design of this shield, and it appears that the inputs should have been buffered before reaching the op-amp. The outputs in the regions where the motor phase is driven high or low are not centered around the common mode, but that is to be expected because the inputs are near the limit of the supply voltage. The outputs in the region of interest are centered around a common-mode voltage of 2.5 V, as was designed, but the difference in the inputs was not multiplied by the op-amp's gain of 10 V/V. The reason for this was unclear. Buffering the inputs would improve on the circuit design and show if the fully differential op-amp topology is viable for observing the phase and neutral voltages.

5.3: Software Architecture

Whether or not Hall sensors are implemented in future versions of the motor controller, the software will certainly be restructured to use interrupts to drive the commutation sequence. Interrupts will allow for higher motor speeds and in-software computations to occur, while also driving the motor more efficiently because interrupts create a closed-loop feedback path from the motor to the MCU.

If the motor can be driven fast enough to generate some BEMF and that BEMF can be amplified by the second motor control shield, an external interrupt can be implemented based on the output of that shield's comparator, which senses the motor's zero-crossing point. Every trigger of the interrupt would cause the commutation sequence to increment.

Chapter 6

Conclusion

The scope of this thesis changed as progress was made, but the results are very interesting and much was learned that can be put to use as this project continues beyond the formal thesis. Although no direct comparison was able to be made between the hexagonal motor and Bugeja's original, the difficulty of using open-loop control to accelerate the hexagonal motor and the relatively low speed that was achieved indicate that sensorless control is not likely viable for PCB motors of this size. Even if the motor was driven to a speed fast enough to observe BEMF, the slight voltage offsets between the motor's floating voltage and the virtual neutral point may make the circuitry design for BEMF amplification and comparison prohibitively complex.

The biggest success of the thesis was the design of the ESC prototyping environment. Swapping out shield boards and motors from the basic motor driving hardware made development and comparison quicker and less expensive than would be possible if entirely new circuitry had to be designed and assembled for each change. In the future, the design of the prototyping environment will be updated based on the experience of using it for this thesis. The biggest change will be the addition of a Hall sensor, which will allow closed-loop control schemes to be tested in addition to open-loop schemes. As more motors are designed, they can be easily tested with both types of control, allowing stronger conclusions about the viability of sensorless BEMF control to be reached.

BIBLIOGRAPHY

[1] C. Bugeja, *Motor made from a PCB!*, Youtube.com, February 25, 2018. [Online]. Available: Youtube.com. [Accessed August 20, 2019].

[2] C. Bugeja, *PCB Motor*, Hackaday.io, 2018. [Online]. Available: Hackaday.io. [Accessed August 20, 2019].

[3] J. Zhao and Y. Yu, *Brushless DC Motors Application Note*, Monolithic Power Systems Application Note, 2011. [Online]. Available: Monolithic Power Systems. [Accessed September 12, 2019].

[4] S. Keeping, *An Introduction to Brushless DC Motor Control*, Digikey Electronics Article Library, 2013. [Online]. Available: Digikey. [Accessed November 10, 2019].

[5] S. Keeping, *Controlling Sensorless, BLDC Motors via Back EMF*, Digikey Electronics Article Library, 2013. [Online]. Available: Digikey. [Accessed November 15, 2019].

[6] G. Zacharia and A. Raina, "A Survey on Back EMF Sensing Methods for Sensorless Brushless DC Motor Drives," *International Journal of Emerging Trends in Engineering Research,* vol. 2, no. 2, 2014. [Online]. Available: Semantic Scholar. [Accessed October 20, 2019].

[7] D. Gambetta and A. Ahfock, "Designing printed circuit stators for brushless permanent magnet motors," *IET Electric Power Applications,* vol 3, no. 5, p. 482-490, 2009. [Online]. Available: Compendex. [Accessed August 21, 2019].

[8] X. Wang, C. Li, and F. Lou, "Geometry Optimize of Printed Circuit Board Stator Winding in Corless Axial Field Permanent Magnet Motor," *IEEE Vehicle Power and Propulsion Conference*. [Online]. Available: Compendex. [Accessed August 16, 2019].

[9] J. Hendershot and T. Miller, *Design of Brushless Permanent-Magnet Motors*. Oxford: Clarendon Press, 1994.

[10] ElectroNoobs, *My Open Source ESC*, electronoobs.com, 2019. [Online]. Available: https://www.electronoobs.com/eng_arduino_tut91.php. [Accessed October 29, 2019].

[11] Arduino, *Arduino Nano*, arduino.cc, 2016. [Online]. Available: https://store.arduino.cc/usa/arduino-nano. [Accessed November 15, 2019].

[12] Texas Instruments, *Differential Op-Amp Single-Supply Design Techniques*, ti.com, 2001. [Online]. Available: https://www.ti.com/lit/an/sloa072/sloa072.pdf. [Accessed March 1, 2020].

ACADEMIC VITA

Stephen Porter

Education

The Pennsylvania State University, Schreyer Honors College — May 2020

B.S. Electrical Engineering

Relevant Work Experience

Electrical Engineering Intern at Key Tech — January 2019 – May 2019

- *Debugged a variety of alpha-level PCBs and systems for a major active project.*
- *Identified hardware and software issues in a critical PCB, then redesigned it.*
- *Worked with a client, vendors, and engineers in the concept phase of a new device.*
- *Designed a PCB to test ultra-low current measurement circuitry.*
- *Built and tested code and hardware to automate air control for a client system.*

Teaching Assistant at Penn State Learning Factory — August 2018 – May 2020

- *Practiced and machining and other fabrication skills with design and build projects.*
- *Taught students machining and woodworking and assisted them with projects.*

Engineering Intern at Human Motion Technologies, LLC — July 2016 – August 2016

- *Designed and constructed a rack-mounted interface to connect a simple prosthetic ankle to its computer control and actuator units. Resulted in $10,000 in savings and is still in use.*

Relevant Project Experience

Student Training Program – Student Space Programs Lab — January 2017 – April 2017

- *Designed a PCB for a rocket to measure and transmit position and other data.*
- *Payload worked on the bench but failed at launch due to a shorted power connector.*

Weather Balloon Launch — November 2014 – December 2015

- *Project and Electronics Team leader, coordinated efforts of subsystem teams*
- *Successful launch and recovery of a weather balloon that burst at 96,000 feet.*
- *Developed atmospheric sensor and data storage circuitry and code for the scientific mission.*

Technical Skills & Proficiencies

- *Altium* | *Eagle* | *LT Spice* | *Electrical Testing & Prototyping* | *LabView* | *Arduino* | *Java* | *Python* | *C++* | *SolidWorks* | *Inventor* | *Woodworking* | *Metalworking* | *CNC*