Brushless Motor Demonstration/Tachometer

Jacob Reyes & Ricardo Rodriguez

April 28th, 2025

Texas A&M University at McAllen - HECM
Multidisciplinary Engineering Department
ESET 359 - Electronic Instrumentation

Instructor: Rafael Fox

TABLE OF CONTENTS

Introduction:

Brushless direct current (BLDC) motors are widely used in various applications due to their energy efficiency and longer lifespan compared to brushed DC motors. They are commonly found in electric vehicles, medical devices, consumer electronics, and robotics. Brushless motors are composed of a stator and a rotor. The stator contains coils of wire that can be energized to generate a magnetic field. The rotor houses permanent magnets. By precisely timing the activation of the stator coils, the magnetic field interacts with the magnets in the rotor, causing the rotor assembly to rotate. We will use this concept to construct our own motor, which will serve as an educational tool to help students visualize and understand the operation of brushless motors.

The main objective of this project is to rebuild the brushless DC motor display in the lab and incorporate a method of speed control using the USB6002 Data Acquisition Module. This will be achieved by redesigning both the motor's base and rotor, allowing the motor to run efficiently. Additionally, we will be using a Hall effect sensor to detect the presence of a magnetic field, enabling precise timing for switching the electromagnet and allowing us to count the motor’s rotations through the data acquisition device.

Description of System Design:

Hardware:

The Hardware used in this project consists of 6 distinct components. These components are listed down below:

- 3D Printed Rotor
- Neodymium Magnets (12mm)
- 608ZZ / ABEC-7 - Bearing
- A3144 - Hall Effect Sensor [1]
- Copper Magnetic Coil - (800 - 1200 Turns)
- IRFZ44N - Mosfet [2]
- NI USB-6002 - DAQ USB Device [3]

We created the rotor (Figure 1) using the computer-aided design (CAD) program Creo, and printed it out using the 3D printer. The roter was designed with an outer diameter of 100 mm and an inner diameter of 22.2 mm. We also included eight slots on the outer rim to house the neodymium magnets. Each slot has a diameter of 12.25 mm and a depth of 3.3 mm, and they are evenly spaced at 45-degree intervals around the rotor. We designed the outer diameter of the wheel and the magnet slots to be slightly larger than the outside diameter of the bearing and magnets to ensure a secure, press-fit connection.

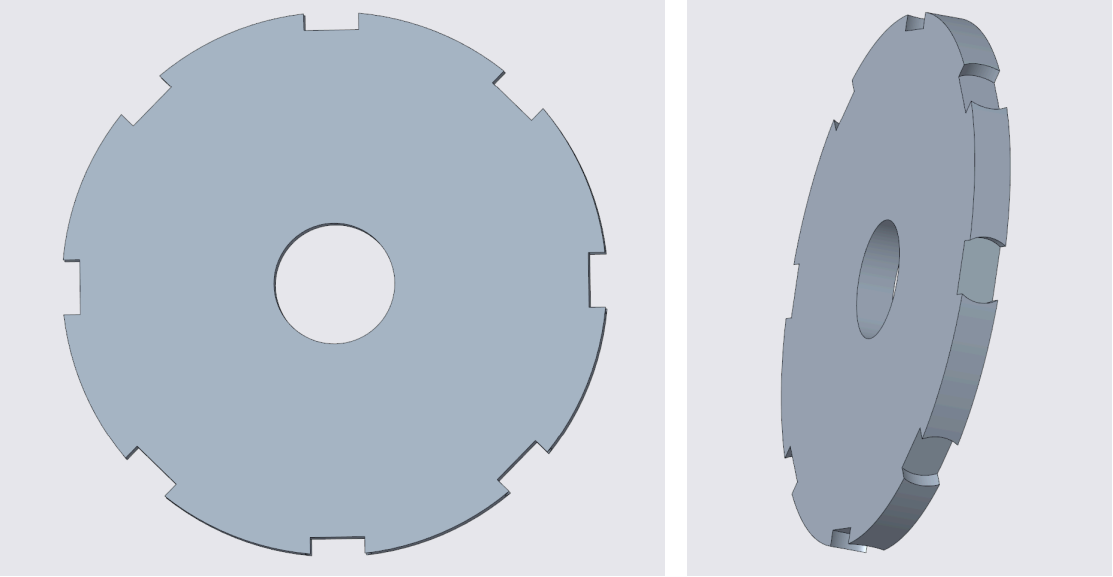


Figure 1: 3D Printed Wheel (Front & Side Views)

We chose to use neodymium magnets (Figure 2) for our project because they are the strongest type of permanent magnet and are available in compact sizes. The magnets we used have a diameter of 12 mm and a thickness of 3 mm. Initially, we planned to use eight magnets for our rotor, however, one was damaged during testing. As a result, we used only six magnets in order to keep the rotor balanced.

Figure 2: Neodymium Magnets (12mm Diameter)

The 608ZZ - ABEC-7 bearing (Figure 3) was press-fitted into the rotor and mounted onto the axle on the motor stand. We chose this bearing to reduce friction between the rotor and the axle. This type of bearing is commonly used in roller skates and skateboards due to the metal shields on both sides, which protect the internal components from dust and debris. We sourced the bearing from a fidget spinner we had on hand. Its low rolling resistance significantly improves the rotor's movement, allowing the motor to operate more efficiently.

Figure 3: 608ZZ / ABEC-7 Bearing

The A3144 Hall Effect sensor (Figure 4) is used to detect the position and presence of the permanent magnets on the rotor. When powered, the sensor typically outputs a voltage equal to the power supply voltage. However, when the sensor detects a magnetic field, the output switches to a low state. We will be monitoring the state of this sensor in order to determine the rotations per minute of the motor. This sensor is also capable of detecting the polarity of the magnet, depending on the pole of the permanent magnet, the sensor's output will either be positive or negative.

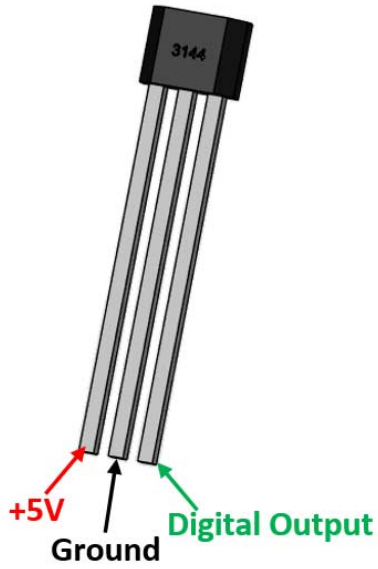


Figure 4: A3144 Hall Effect Sensor

The copper coil (Figure 5) will serve as the stator in our motor and will be used to create an electromagnet that repels the neodymium magnets mounted on the outer rim of the rotor. Electromagnets are formed by passing an electric current through a coil of wire, generating a magnetic field within the coil and turning it into a magnet. We are using an electromagnet because it allows us to control the magnetic field activating or deactivating it as needed. The coil will be integrated into a circuit that activates the magnetic field whenever a permanent magnet is detected. By timing the activation of the electromagnet, we can continuously push the rotor in one direction.

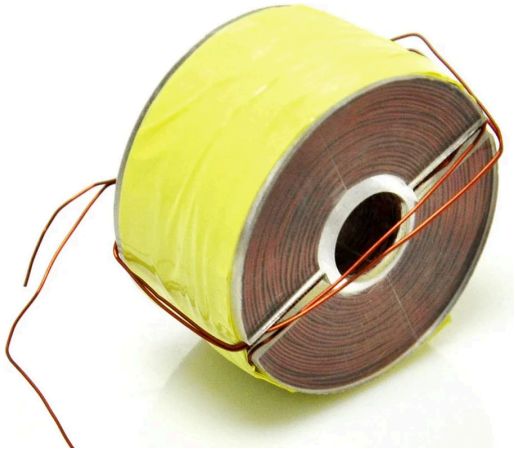


Figure 5: Copper Magnetic Coil (800 - 1200 Turns)

The IRFZ44N MOSFET (Figure 6) will be used to switch the electromagnet on and off based on the signal from the Hall effect sensor. We chose to use a MOSFET instead of a standard transistor due to its faster switching speed and higher efficiency. The output from the Hall effect sensor is connected to the gate of the MOSFET. Depending on the sensor's output, the MOSFET will either activate or deactivate the electromagnet.

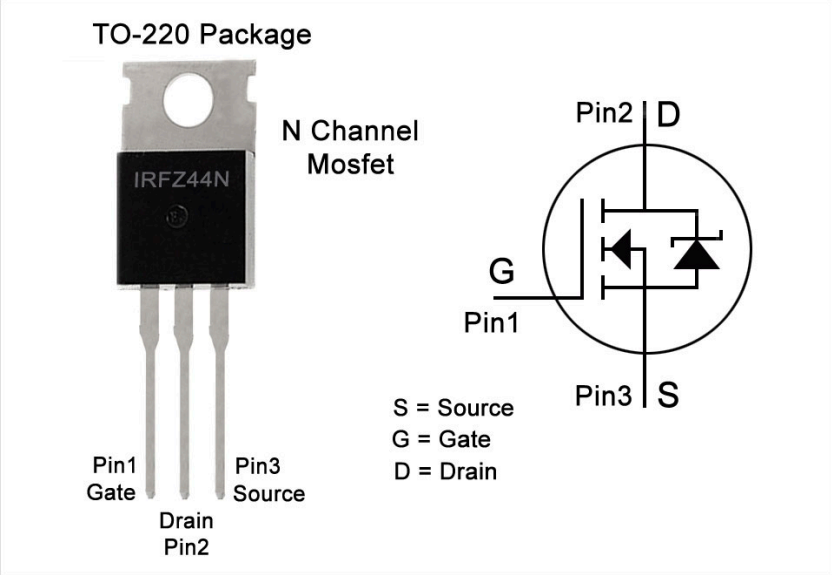


Figure 6: IRFZ44N Mosfet

The NI USB-6002 Data Acquisition System (DAQ) (Figure 7) will be used to collect signal data from the Hall effect sensor in LabVIEW. The DAQ will monitor the sensor’s voltage output, which, as mentioned previously, is typically at a digital high level and drops to a digital low when a magnetic field is detected. Using the DAQ, we will capture these transitions and create a Virtual Instrument (VI) in LabVIEW to count the number of high-to-low changes. This count will then be used to calculate the RPM (revolutions per minute) of the motor.

Figure 7: IRFZ44N Mosfet

Software:

The software we utilized for this Brushless Motor Demonstration/ Tachometer would be the graphical programming environment and language, specifically a dataflow programming language or otherwise known as the software of LabVIEW. For this project we were required to measure the brushless motor's revolutions per minute in a semi-accurate format using the aforementioned software LabVIEW. This proved rather arduous and time consuming due to LabVIEW's unique graphical programing. However, we first calculated what the required steps would need to be to start the system in response to the user's wanted revolutions per minute. Thus we were able to create a flowchart that would be able to describe the required process to successfully replicate the wanted objective. This flowchart is shown in Figure 8.

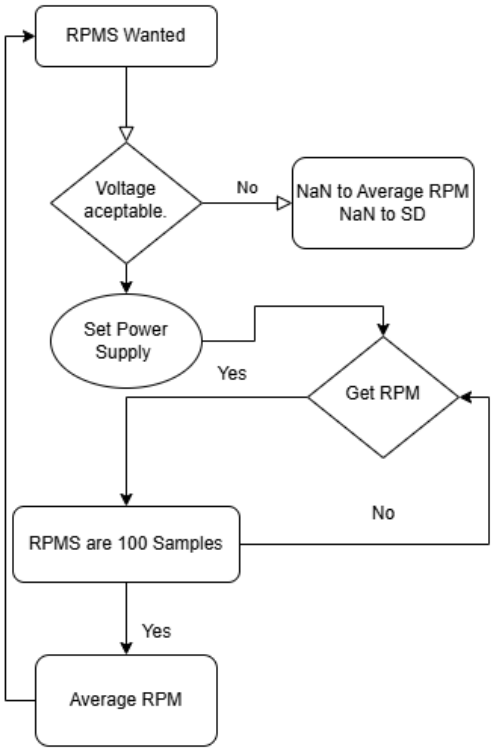


Figure 8: Total Simplified Flow Chart Schematic

As we can see from the flowchart shown above, we would first require the software to get the input RPM's and convert it into a voltage level that we can then send to our Keysight Power Supply to power the magnetic coil and the Hall Effect Sensor. This conversion rate is not available to us until we have developed a system that can first gather the RPM's from an arbitrary voltage level with a constant limiting current of 1 Ampere. Therefore, we will first have to develop a system that can gather the RPMs from our brushless motor demonstration before we can set up a conversion rate between voltage and the revolutions per minute of the system. Thus we were able to create the following LabVIEW program seen below to be able to determine a singular RPM measurement in a while loop.

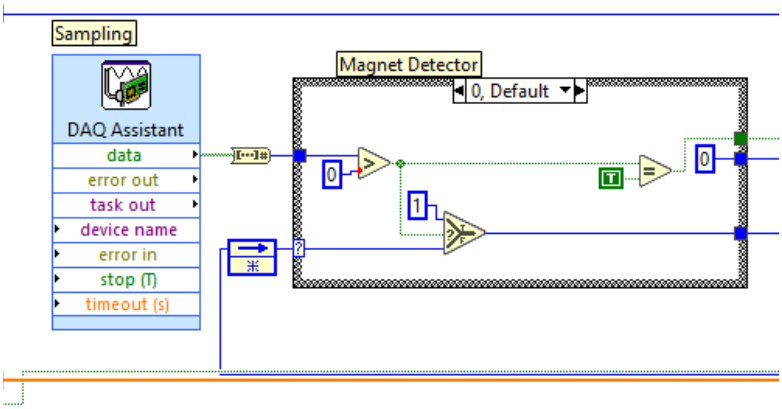


Figure 9: DAQ Digital Input & State Detector

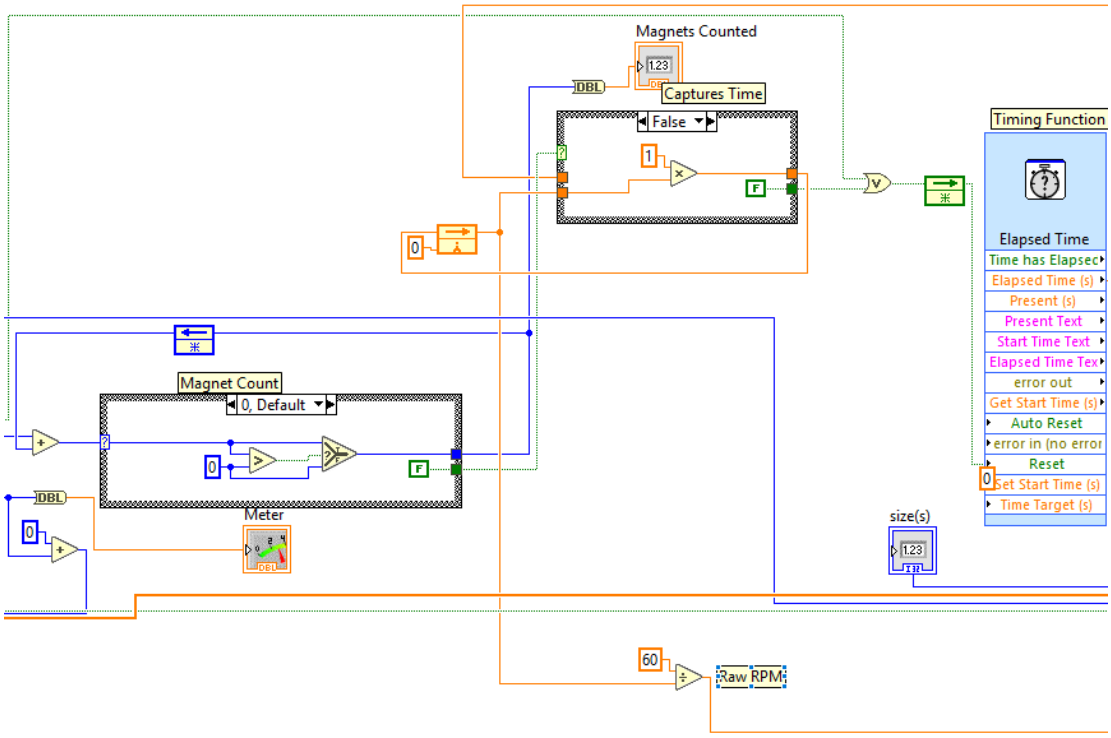


Figure 10: Magnet Counter, Time Function, Time Capture, Raw RPM

The first step of our LabVIEW software program is to first set our DAQ assist as a digital input. This means we will see a “True” if the signal is high (magnet not detected) and a “False” when the line is low, (magnet is detected). We see that first we must convert our boolean output of the DAQ assist input to

that of a scalar int value of 1 or 0 for each respective possible input, as shown in Figure 9. That input would then be used to determine in which state it would be. There exist three possible “Magnet Detector States”, which include state 0 which is when the first magnet is detected and we output a signal that will allow us to start our time function. State 1 will determine if the system is on (meaning no magnet is present) it will be maintained here unless a magnet pulse is generated. State 2 is when the system sees a magnet which then results in us outputting a value that will then feed into the magnet counter seen in *Figure 10*. This magnet counter, seen in *Figure 10*, will only work as a case statement that will determine how many magnets the system has seen, which in our case would be six magnets to see and the seventh magnet should stop the system and get the elapsed time from our Timing Function. Once we have enough magnets counted, we will then get the elapsed time before we reset the timing function to start from zero which can only be done by holding this value in a case statement (Capture Time Case Statement seen in *Figure 10)* using our very limited knowledge in LabVIEW. From there we will be able to divide 60 seconds by our captured elapsed time to calculate the amount of revolutions per minute if we know how long it takes for one revolution.

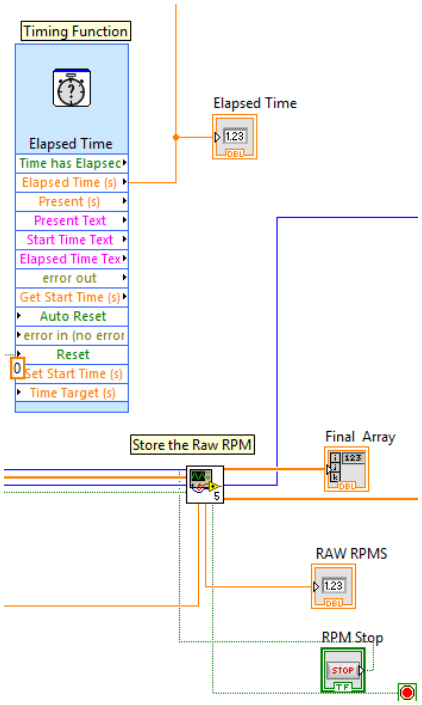


Figure 11: Storing Sub-Vi Array

The next step as we can see in our flowchart in *Figure 8,* would be placing the raw RPMs in an array until an array size of a hundred iterations was complete. To accomplish this we created a sub-vi *Figure 11* that would receive the empty array from the system and would add the raw RPM while only adding the value that did not repeat itself consecutively as well as if the number was either zero or infinity. We made every number in the array unique due to the raw RPM loading into the array through one interaction of the while loop, meanwhile the RPM value changed after a few iterations of the while loop had occurred. The sub-vi would check for the value if it was repeated and for incorrect values of 0 and inf to not be present in our system. The sub-vi also measured the array size in such a way that it would stop our while loop if the size of the array reached 100 iterations. This can be seen in *Figure 12*, where Numeric 2 acts as the input of the Raw RPMs and array out acts as the appended array with all the correct values that will then be fed to the shift register of our original while loop.

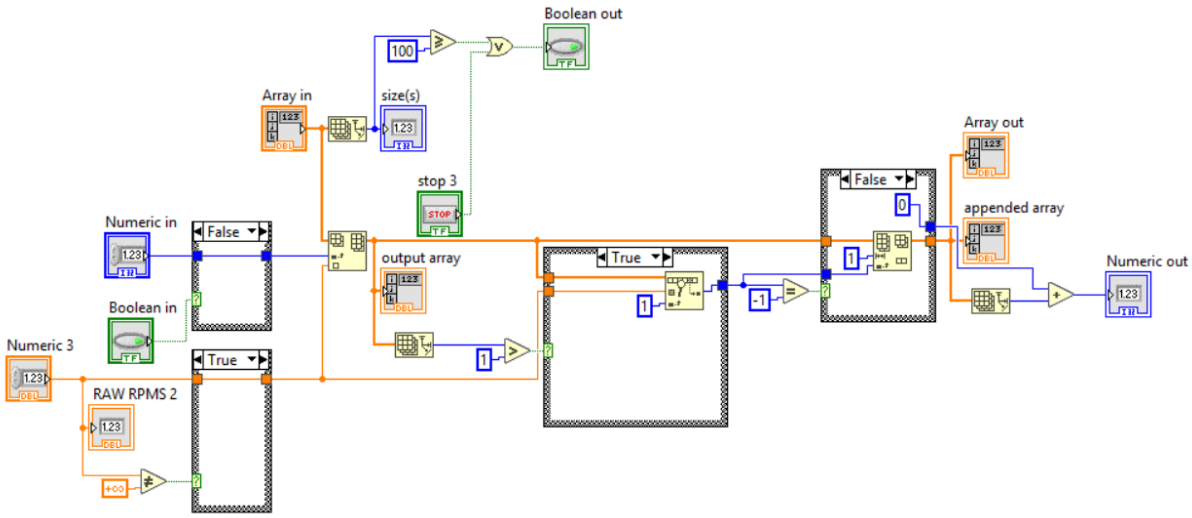


Figure 12: Sub-Vi - Storing Values into the Array

Once the size of the array has reached the desired limit we, we then proceed to end the while loop with our completed array as our output of the while statement. This allowed us to then set up the Statistic Function provided by LabVIEW through the "statistical and probability" menu page to automatically determine the arithmetic average of an array of whatever size desired as long as the array is converted to a

dynamic type of data. We also were able to determine the standard deviation of these measurements due to the fact that the system itself is not as accurate due to several factors. With this we were able to determine the Average RPMs of the system from 100 data samples alongside its own standard deviation. This function alongside the conversion paths are shown in *Figure 13*.

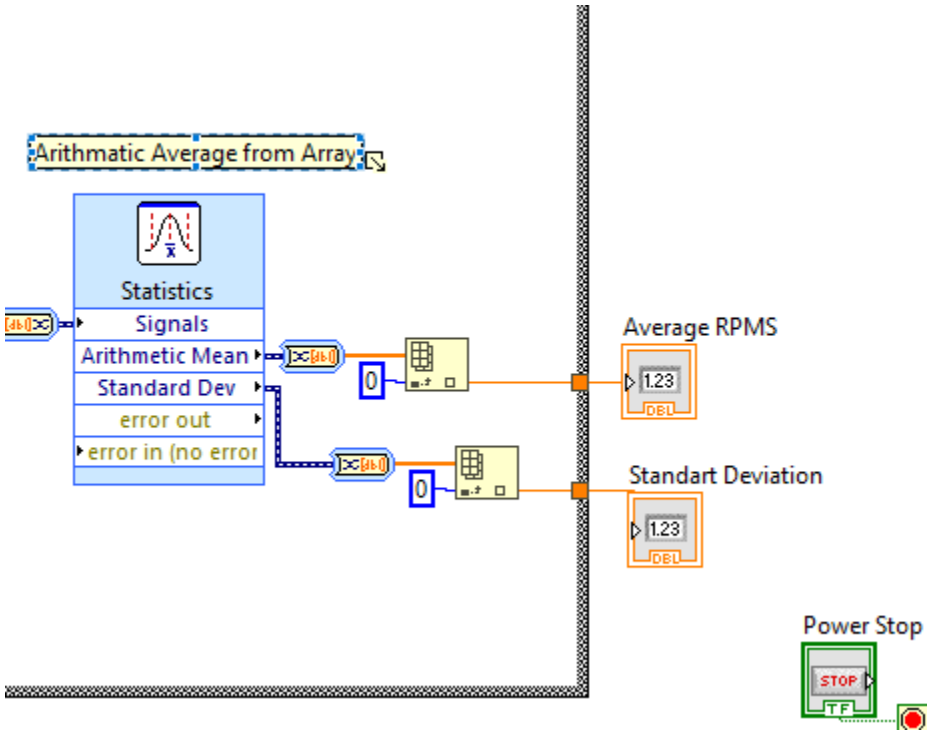
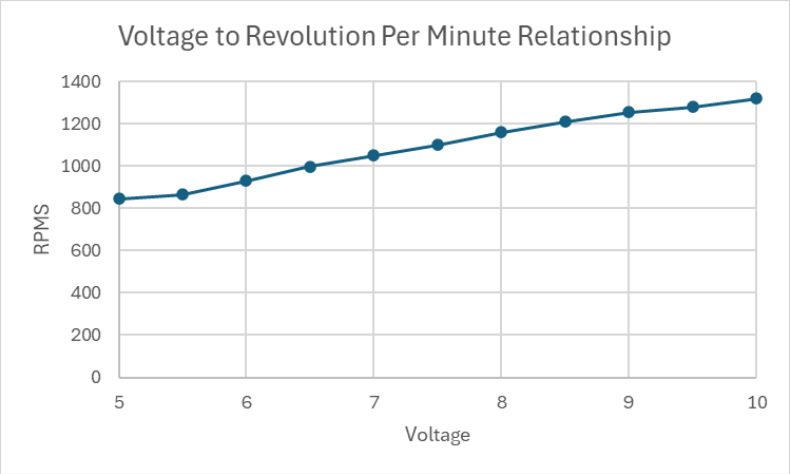


Figure 13: Statics and Probability Function

From this we were able to determine the relationship between the voltage of the system in relation to the Revolution Per Minute. Which was getting the Average RPMs from the 5v to 10v mark with a data step size of 0.5v, all this after the system had achieved a steady state (*One Minute*). These values can be seen in Graph 1, where we were able to plot the relationship between voltage and RPM and can be represented in *Equation 1*.

Graph 1: Voltage to RPM Relationship



$$RPMS = 100.84 \cdot V + 335.45$$

Equation 1: Relationship of Line Fitting of Graph 1

From this relationship and equation we were able to determine a conversion rate that will follow the trend relationship seen in *Graph 1.* We will first get the input RPMs desired and subtract 335.45, then we will get that value and divide the 100.84 to be able to generate the appropriate voltage for that desired RPM value. However, there are some limitations, such as having the system not being powered by a voltage value lower than 5 volts and higher than 10 volts since the system sees unimaginable RPM noice due to the brushless motor being too fast for the labview software to operate accurately (*Figure 14*). If the voltage is between this sweet range we will then be able to feed it into our power supply, which is the sub-vi provided by Keysight *(Figure 16*) once we load the appropriate drivers to properly control it remotely (*Figure 15*).

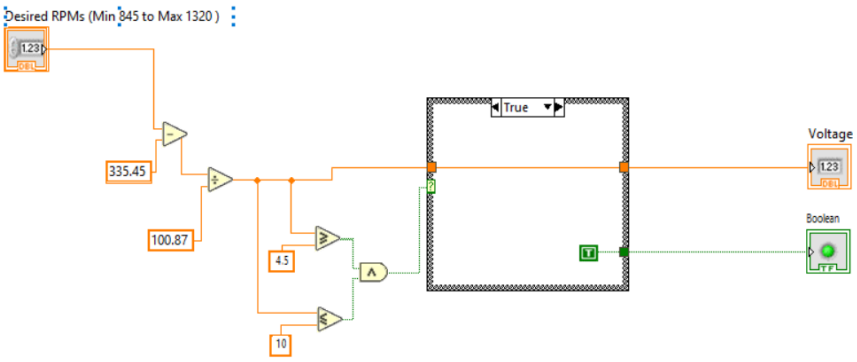


Figure 14: Conversion Ratio of RPM to Voltage

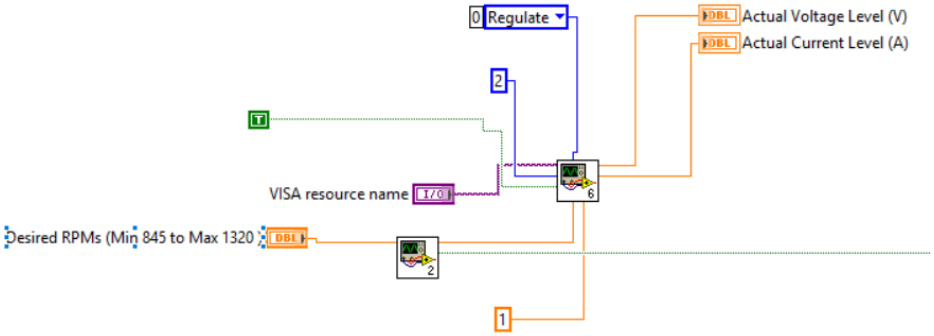


Figure 15: Conversion Ratio Subvi powered to the Power Supply.

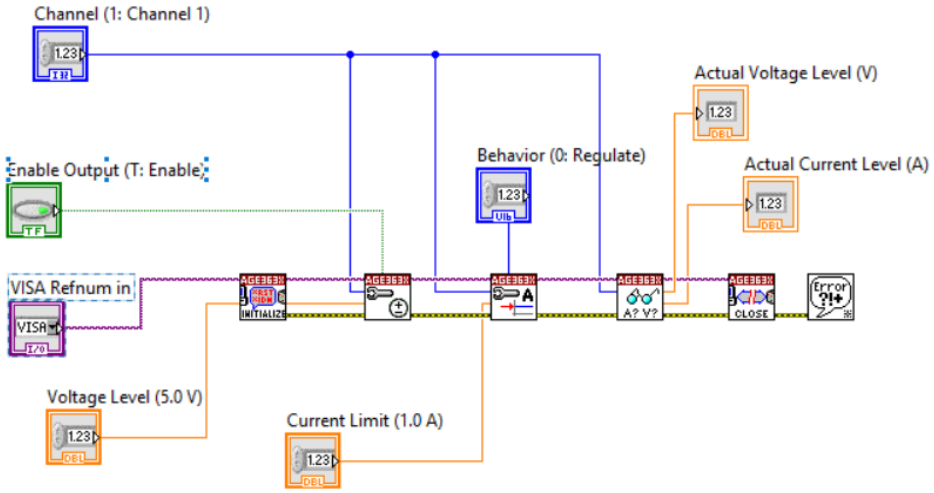


Figure 16: Power Supply Vi - Keysight.

This is all the software that we were able to create to properly run our system with the desired RPM with just the help of a "starter" push to the brushless motor. All this code accumulated allowed us to control the Power Supply to the appropriate RPMs after capturing the Raw RPMs through the utilization of case statements as well as gathering the average RPM's through the use array storage. The final piece of software we developed would be the user interface of LabVIEW, which we tried to the best of our ability to make as user friendly. This LabVIEW interface can be seen in *Figure 17*.

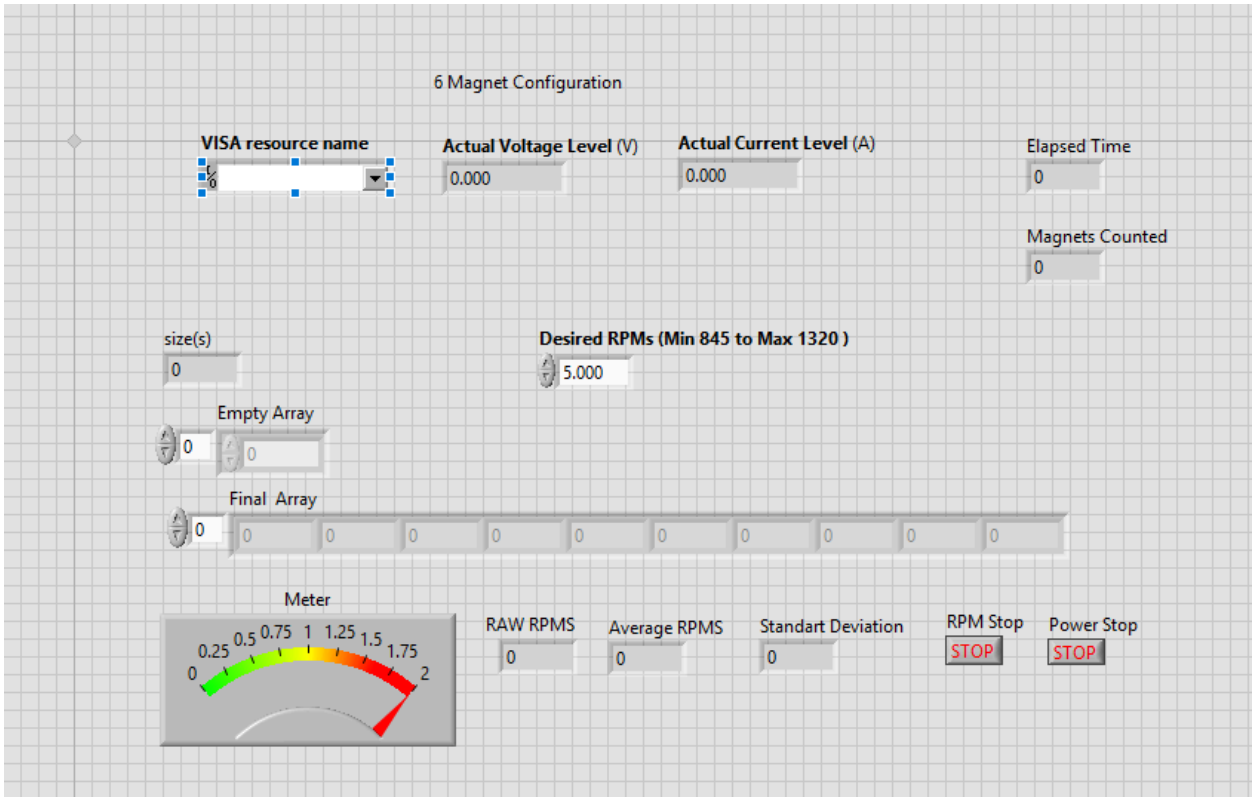


Figure 17: User Interface of LabVIEW.

Test Plan and Results:

The first thing we focused on was dismantling the previous configuration of the brushless motor demonstration and taking measurements of the acrylic rotor that had been previously installed. We decided to create a custom rotor using the CAD (Computer Aided Design) program Creo, and print it using our 3D printer with ABS plastic filament. We kept the rotor's diameter the same and included a central hole to house the 608ZZ - ABEC-7 bearing, as well as additional holes to accommodate the neodymium magnets. The design of the rotor went through several iterations. In the first iteration, the inner diameter of the rotor was too large, causing the bearing to fit loosely. We addressed this issue in the second iteration, however, during testing, we discovered that placing the magnets on the face of the rotor did not allow the motor to function properly. Since the magnets were not directly aligned with the magnetic field of the coils, the rotor had difficulty spinning. We attempted to fix this by modifying the existing rotor, but ultimately decided to design a new rotor with slots on its outer rim, positioning the magnets directly in front of the copper coils. This became our final rotor design. The slots were slightly smaller than the magnets, ensuring a tight and secure fit without the need for adhesives.

The second step of modification was rewiring the circuit to fit compactly on a smaller breadboard, this was mainly a cosmetic modification since we did not alter the circuit in any other way. We also mounted the Hall effect sensor onto a prototyping board by soldering the sensor onto the board as well as wires in order to connect the sensor to our circuit. We also created an adjustable stand that will secure the sensor to the board and place the sensor close to the rotor where it can detect the magnetic fields of the permanent magnets. The sensor stand was created using a leftover piece of plastic and formed into a 90 degree bend by applying heat while benign the plastic into shape. Adjustable slots were also cut into the stand which allows adjustability in the placement of the stand.

Finally we developed the LabView portion of the brushless motor which we had major difficulties due to it being a new programming software that used graphical programming. We had the most challenging section of the LabVIEW section being the actual storing and averaging of the RPMs. This

was due to the value being placed and stored into the system for each iteration of the while loop and the RPM value would require several hundred iterations of the while loop to actually change ist value. We were able to solve this problem by deleting the same values as the input value for the previous ten data points in the array.

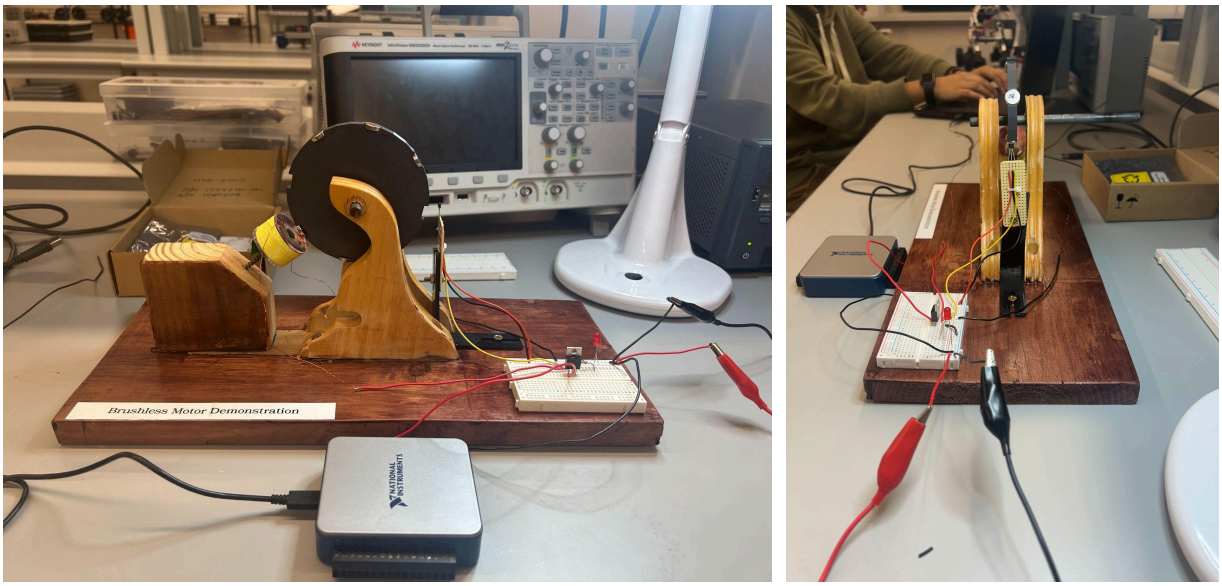


Figure 18: Completed Motor (Front & Side Views)

Description of System Operation:

The brushless DC motor display will rotate at a speed set by a user in the Virtual Instrument (VI) we created. To set up the motor you will need to open up the Virtual Instrument on a laptop and connect the National Instruments USB-6002 DAQ to the laptop as well as the usb cable coming from the power supply. After, the initial setup is complete the user is prompted to input a desired speed for the motor, keep in mind that due to limitations of the DAQ the speed range of the motor is 830 RPM - 1320 RPM if a speed that is outside of this range is entered into the VI the program will not activate the power supply. Once an acceptable speed is entered into the VI the power supply will output a voltage that corresponds with the set speed. An initial push is required to get the motor to start spinning, once the motor is spinning the speed will steadily increase until it reaches the target speed. From a dead stop the motor's rise time is around two minutes, and if the motor is already spinning, the rise time drops to around one minute and a half. The motor will then continue to spin until the user presses the stop button on the front panel of the VI.

The brushless motor (*Figure 19)* circuit functions by switching the electromagnetic field of the stator on and off in response to the position of the permanent magnets located on the outer rim of the rotor. When the Hall effect sensor detects the presence of a magnet on the rotor, it outputs a digital low signal to the gate of the MOSFET, which turns off the coil's electromagnetic field. With the coil off, the magnet opposite the Hall effect sensor can pass by without interference. Once the magnet moves past the coil, the Hall effect sensor no longer detects a magnetic field and sends a digital high signal to the MOSFET, turning the coil back on. The magnetic field of the coil repels the passing magnet, accelerating the rotor. This cycle continuously repeats, with the coil repelling the magnets at precisely timed intervals, causing the motor to spin.

The software first requires an RPM value from the user that falls between the ranges of 830 RPMs and 1320 RPMs (5.0V to 10V respectively in their RPM format). The software will initialize the system with a voltage correlated to that specific user input RPM. After we initialize the starter to make the

wheel rotate initialize, the system will generate the RPM by counting how many magnets the DAQ has seen in the digital port and then getting the elapsed time if the required seven magnets have been seen in total.This will get us simply one RPM value which we will then store into an array, and replicate this process until we have 100 data samples in said array Once we have enough data samples in the array then we will be able to inform the user what the average RPM is after getting the arithmetic average of the complete array from the statistic module. This process will continue indefinitely until the program is terminated.

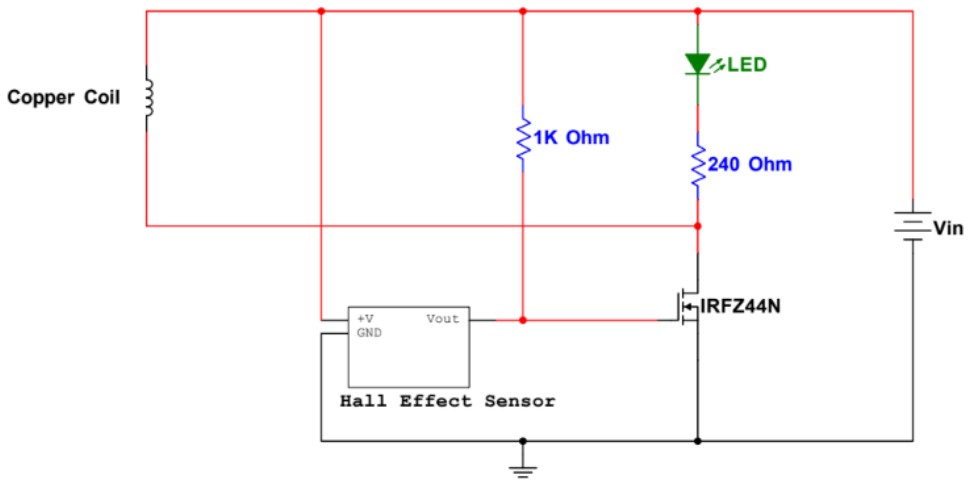


Figure 19: Brushless Motor Circuit

Conclusion:

In conclusion, this project was successfully completed in alignment with the objectives set for the brushless motor demonstration and tachometer. Through this project, we gained a deeper understanding of how electromagnetic currents interact with circuits, as well as how to calculate revolutions per minute (RPM) using LabVIEW software. Overall, this project enhanced our grasp of key concepts such as automated systems and closed-loop control systems.

Some improvements we would like to make in future iterations include the implementation of a PI (Proportional-Integral) controller to more effectively reach the desired RPM. A PI controller would help stabilize the system by minimizing the error in feedback. Additionally, we plan to incorporate the two remaining magnets into the brushless motor demonstration. This will improve the system's weight distribution and result in a smoother RPM reading.

Despite challenges related to time constraints, this was an ideal final project for the course. It covered a wide range of topics and provided valuable hands-on experience in designing and developing a closed-loop system. We solidified our understanding of constructing and programming the brushless motor display and tachometer.

References:

[1] Components101, “A3144 Hall effect Sensor,” Components101, Jan. 03, 2018. https://components101.com/sensors/a3144-hall-effect-sensor

[2] HEXFET, “IRFZ44NPbF HEXFET ® Power MOSFET.” Available: https://www.infineon.com/dgdl/Infineon-IRFZ44N-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a40153563b3a9f220d

[3] National Instruments, “USB-6002 Specifications - NI,” https://www.ni.com, 2023. https://www.ni.com/docs/en-US/bundle/usb-6002-specs/resource/374371a.pdf?srsltid=AfmBOopLYwtzaPJZpcclRkhTvwaVCd1Xk668Bfzd6NLr4S-EwZEXHP06 (accessed Apr. 29, 2025).