Brushless Servo Control

A General Purpose Industrial Servo System using Hobby Brushless Motors

Ethan Russell
University of Puget Sound

ABSTRACT

This paper describes and evaluates a servo control system designed to use off-the-shelf brushless motors instead of traditional purpose-built industrial servomotors. These motors are very well suited for servo control while being much cheaper. The control system is designed to be used with a wide variety of servo applications, and drop in place of more traditional motor control systems.

This paper describes methodology for implementing brushless servo control, specifically field-oriented control, servo control systems, and outlines the hardware and software design process for developing a brushless servo controller with the STM32 microprocessor series.

1 GLOSSARY OF TERMS

- Armature - The series of electromagnetic windings in a motor.
- Stator - The stationary part of a motor. In brushless motors, this contains the armature.
- Rotor - The rotating part of the motor. In a brushless motor, this contains a ring of permanent magnets.
- Torque - A rotational force
- Commutation - The process by which a motor's windings are energized and de-energized.
- Open-Loop/Closed-Loop - closed-loop control loops, also known as feedback loops, try to maintain an output condition by comparing it against the current actual condition. Open-loops simply process a 1:1 input to an output.
- Back-EMF - The generated voltage created by magnets rushing past the windings.
- AC/DC - Alternating-Current, Direct-Current respectively. DC values are constant, while AC have a time-varying waveform.
- PWM - Pulse-Width-Modulation. Output voltage switched on and off, effectively modulating analog voltage levels in a digital way.
- Duty-Cycle - Time spent on divided by total PWM period. Duty cycle is effectively proportional to drive voltage.

2 INTRODUCTION

Industrial servomotors are fundamental to robotics. Every industrial robot arm, pick and place machine, CNC machine, and electric car traction control system use some sort of servo to move. Servo motors use position and torque sensors to feed back into a closed-loop control system to regulate the torque output of a motor to move precisely and consistently.

Despite being essential for almost any computer controlled mechanical system, there are several problems with existing servo motor systems. Primarily, the cost of commercially available servo motor systems makes servo systems out of the reach of hobbyists, pushing someone with a small budget to use more inefficient and naive alternatives to servo control - specifically stepper motors or pneumatic systems. Secondly, there is no general purpose servo control system for every application. Despite having very similar logic, there is a different servo control system for low speed position control with gear reduction, velocity regulation, and yet another for torque control.

Meanwhile, there has been an effort to make hobby DC motors for remote controlled cars more and more power-dense. Traditional brushed DC motors are limited in how much power they can dissipate without burning up, so the hobby market has moved towards using brushless motors instead. Because of this push, brushless motors have come down in cost dramatically, and these incredibly powerful motors can be picked up at a hobby shop for nearly pocket change. Brushless motor controllers need to keep track of rotor position so many of these cheap RC car motors have small hall-effect sensors built into the armature, so unlike traditional industrial servomotors, no extra mechanical systems are necessary for servo control.

This project was created to use these cheap hobby brushless motors as industrial servomotors for robotics. Outside of this controller, the primary robotics logic will send position, torque, and velocity commands, and the brushless servo control system will generate waveforms to regulate the respective motor characteristics.

3 BACKGROUND AND RELATED WORK

3.1 Existing Brushless Servo Alternatives

3.1.1 Stepper Motors. Stepper motors are typically used in hobbyist robotics projects because of their ease of use. A stepper motor is a simple open-loop motor that regulates rotor position mechanically. Every time a coil is energized, the rotor moves a fixed number of degrees and then continues to draw current to keep the rotor in place. Because the motor is always drawing stall-current, it has to be built to withstand the heat generated from continuously drawing stall-current, and therefore is many times less power dense than other DC motors. Furthermore, if anything mechanically stops the rotor from spinning, the control logic can't react, and the robotic system will fail. The motors are very electrically and mechanically

noisy as the rotors accelerate and decelerate to and from a stop hundreds of times over a single revolution.

3.1.2 Brushed DC Servos. Typical servo systems use Brushed DC motors and use an optical encoder to measure the rotor position. The electromagnets inside traditional brushed DC motors are commutated by mechanical brushes. As voltage is applied to the motor terminals, the rotor will spin automatically as the brushes contact the rotor's winding contacts. Position is regulated with a closed loop using the optical encoder as feedback, and voltage output to the brushes is ramped up and down. These motors are not very good at regulating torque because of torque ripple from the commutation process. Torque sensitive applications like camera stabilization have only become accessible due to brushless servo technology.

Cheap hobby DC servomotors (for use in model airplanes, etc.) use a potentiometer in place of an optical encoder for price reasons, and therefore can only move a maximum of a few degrees before reaching a limit. These small plastic bricks are what most people think of when mentioning the word "servo", but while these motors use the same general technology, hobby DC servo systems are much simpler than industrial servo systems.

3.2 Commercially Available Brushless Servo Systems

Several brushless servos are on the market currently. Due to their power density and consistent torque output, brushless servo systems have become very popular in industrial applications. These servo systems however, are typically sold as a unit with a controller and motor system, and are typically don't drop below $1000 per motor. These systems usually don't have torque regulation options, so they wouldn't be applicable to any servo application. They wouldn't be general-purpose enough to use on a delicate camera gimbal, for example.

To the best of my knowledge, there is no industrial servo system that uses off-the-shelf hobby motors without additional mechanical components (usually optical encoders) attached. Using hobby motors dramatically decreases the cost, because hobby brushless motors are becoming increasingly available.

4 BRUSHLESS SERVO CONTROL METHODOLOGY

Unlike traditional DC or asynchronous AC motors, brushless motors respond very differently depending on the input waveform. This controller project aims to drive a brushless motor optimally to enable applications that require very constant torque control.

4.1 General Motor Model

Figure 1 diagrams the electrical constituents at play when driving power through a motor armature. Only R_a has a fixed series resistance, created by losses in the copper windings. All other impedance is created by a) The coils in the armature which make a large inductor L_a, converting the armature current to and from magnetic fields, effectively smoothing out time-varying current with large voltage spikes when current changes. Also, b) the voltage source e in parallel with the drive voltage source. This is known as the back-EMF, which is produced when the magnets in the rotor are spinning past the armature windings. This voltage is in phase with the drive voltage, and minimizes armature current I_a at higher RPMs.

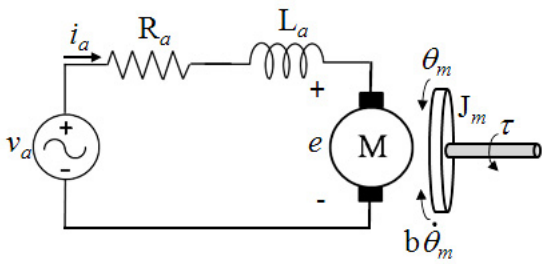

Figure 1: Electrical motor model

4.2 Brushless Motors

Brushless motors, also known as Permanent-Magnet Synchronous Motors (PMSM) are controlled by closed-loop logic systems that monitor the rotational position of the motor and generate a synchronous AC waveform whose period defines the rotor speed, and whose shape affects the torque production over the course of its revolution. In these motors, commutation logic is used in lieu of carbon brushes and copper contacts. Use of solid state electronics reduces friction, increases reliability, decreases motor cost, and increases the winding current that the motor can withstand. Even cheap hobby brushless motors can many times more power dense than any other existing motor technology. The trade-off is more complex and expensive controllers. In a servo system however, the controllers are necessary regardless of motor design.

4.3 Trapezoidal Commutation

Traditional hobby brushless control systems generate a very simple square-wave waveform emulating the waveform seen by the armature of a traditional DC brushed motor. This commutation technique is known as trapezoidal (or six-step) commutation. Trapezoidal commutation is the simplest brushless motor control strategy. At any point in time, one motor phase is sourcing current, one is sinking current, and one is disconnected. This leaves six possible states, which are divided evenly based on the rotor position, so each state is active for 60°(electrical). Using three hall-effect sensors to derive the rotor position, the controller can be driven by a simple state look-up table. This is shown in Table 1.

This commutation technique also makes sensorless rotor measurement easier. When a phase is turned off, the controller can listen to the the Back-EMF voltage generated from that phase. Because the back-EMF is synchronous with the rotor, the controller can use the back-EMF waveform to find the rotor position. Hobby brushless motors used for helicopter propellers don't need torque at low RPMs, so they can simply drive the motor like a stepper motor until it generates enough rotor speed to pick up the back-EMF. Unlike RC helicopter propellers however, servo control is very dependent on low RPM torque so sensorless control wasn't feasible for this project.

Table 1: Trapezoidal commutation lookup table

Electrical Angle	Hall-Effect State	Phase A	Phase B	Phase C
0°-60°	{0,0,1}	+	-	Off
60°-120°	{0,1,1}	+	Off	-
120°-180°	{0,1,0}	Off	+	-
180°-240°	{1,1,0}	-	+	Off
240°-300°	{1,0,0}	-	Off	+
300°-360°	{1.0,1}	Off	-	+

4.4 Field-Oriented Control

Because some servo applications are very sensitive to torque fluctuations and efficiency, this project puts more care into producing a waveform that minimizes heat and stabilizes torque production. Traditional trapezoidal commutation can not regulate the torque going into the motor, because the current is constantly changing. Field-oriented control aims to a) regulate the current going through the motor, and b) make sure that the motor is driven optimally.

If we think of the magnetic field generated by the armature as a vector quantity centered between the three coils, trapezoidal commutation would jump 60°every time it changes state. We can also think of the rotor's magnetic field as a vector quantity - the rotor's permanent magnets move around the stator once per revolution. Ideally, the stator's magnetic field should line up with the rotor's magnetic field to produce torque to the rotor. Because there is always a component of these vectors that don't line up, some of the current driven by a trapezoidal commutation controller goes into heating up the motor instead of producing torque - the non-torque producing component of armature current goes into weakening the rotor's magnetic field instead.

If the current production from the brushless controller were sinusoidal and completely in-phase with the rotor's back-EMF, torque ripple is removed, overall torque is increased, while the total armature current stays the same. Furthermore, the inductive voltage spikes from pulsing coils in a square wave would be minimized, decreasing heat dissipated by the power inverter. Field-oriented control does exactly this - it uses current sensors on each of the phases to drive fully sinusoidal phase current to the armature

Field oriented control takes advantage of all of the benefits of brushless motors. Because it isolates the torque producing component from the field-weakening component, it creates a fully independent torque and field controller. For very torque sensitive applications, torque is completely controllable, even at extremely low speeds.

4.4.1 D-Q Reference Frame. Fundamental to understanding field oriented control is understanding the armature's magnetic field vector with respect to the rotor's reference frame. The torque-producing and field-weakening components of the armature current are fixed with respect to the rotor. As the rotor spins, only current in the torque-producing axis provides any torque on the rotor. In practice, the torque-producing axis is 90°ahead of the magnetic field vector from the rotor's permanent magnets.

Through relatively simple trigonometry, the magnetic field vector can be transformed into the reference frame of the rotor. Figure 3 shows this transformation. In the rotor reference frame, called the d-q reference frame, the torque-producing axis is known as the quadrature (or q) axis. The field-weakening axis is known as the direct (or d) axis.

In practice, this transformation is done in two steps. The three phase current is mapped into X and Y components, and then rotated to line up with the rotor. These two steps are known as the Park transform and Clarke transform, respectively.

4.4.2 Synchronous Current Regulator. A subtlety of this description is the assertion of driving current current instead of driving voltage. The motor model describes how a motor's impedance varies with time - the coils are inductive loads which resist changes to current, and back-EMF is generated as the rotor's magnets spin past the stator. Current and voltage are proportional only with a constant impedance, and a motor's impedance can't be predicted at any given point in time. Regulating current requires a closed-loop control system that senses current and varies output voltage.

In practice, we can only pulse full drive voltage on and off instead of actually varying the voltage. This is known as PWM, or Pulse-Width-Modulation. Inductive loads like motor windings, resist the change of current and average out pulse-width-modulated output and generate a relatively constant voltage proportional to the PWM duty-cycle.

Instead of having three control loops that independently control PWM of the three phases to regulate current, current is transformed to and from the d-q reference frame and regulated from there. While the math can be relatively computationally-intensive, the control systems won't be regulating AC values, making the control system much more responsive, even when the control loop is cycling slowly.

This is the fundamental part of field oriented control - known as the synchronous current regulator, diagrammed in Figure 2. The synchronous current regulator keeps track of the current running through all three phases and varies the PWM to all three phases to regulate a current vector as defined by the user.

4.5 PID Control Systems

Much of the science of controlling any physical system comes down to closed-loop control systems that can carefully ramp an output up and down to achieve an output sensor reading. In practice, this usually ends up being a PID control loop. PID (acronym for Proportional, Integral, Derivative) systems find the difference in a measured sensor value and a target setpoint, called the `error`. The output control variable from a PID controller is the sum of three terms: the proportional term, the integral term, and the derivative term.

The proportional term (P-term, or gain) simply multiplies the error by a fixed gain. A proportional-only control system would cause oscillation in a mechanical system. If a motor was set to a certain setpoint, the P-term would ramp motor torque to move the position to the setpoint, and would have to ramp it back as inertia kept the rotor spinning past the setpoint.

The Integral term integrates the previous errors over time. If there was a steady error too small for the proportional gain to respond adequately, the integral term would get bigger and bigger, providing the motor with the push it needs to make it to a setpoint.

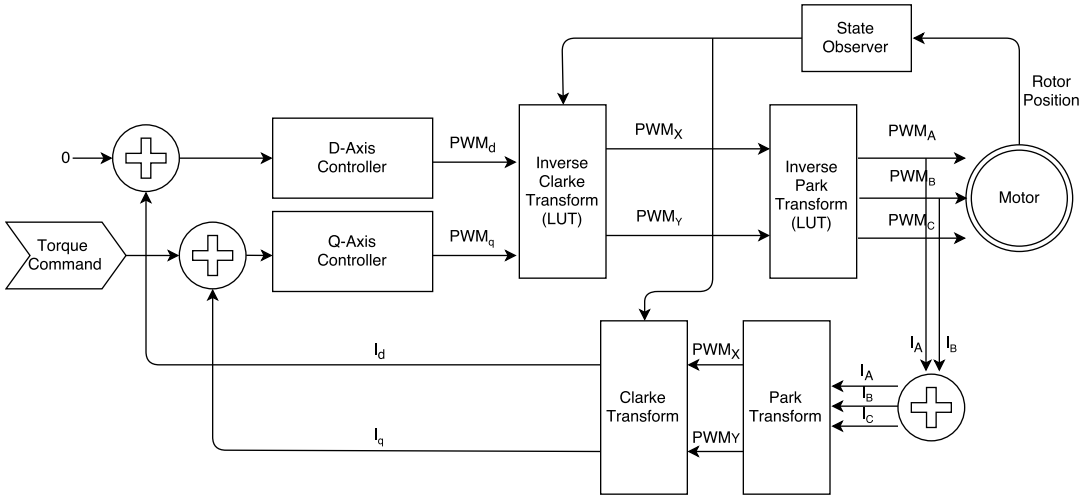

Figure 2: Block Diagram for the Synchronous Current Regulator

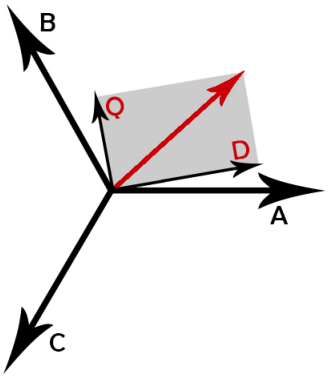

Figure 3: Transformed vectors from phases ABC to the d-q reference frame

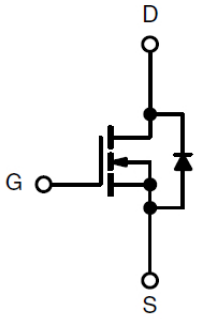

Figure 4: MOSFET schematic symbol

The derivative term pays attention to the change in error over time. If the error begins to get smaller and smaller, the derivative term is there to make sure the system doesn't overshoot, and ramps the output down as the measured value gets closer to the target. Effectively, the D-term dampens the output ringing.

By adjusting the weights of the three terms, a servo system can be tuned to jump to a position as quickly as possible without moving past the target point[4].

In this project, PID control systems are used in two places: the outer servo controller to vary `torqueCommand` to make the rotor move to a desired position or velocity, and the control loops inside of the synchronous current regulator to vary the output PWM duty cycles to regulate I_q and I_d currents to `torqueCommand` and 0, respectively.

5 IMPLEMENTATION

The final project for the Brushless Servo Driver is a single electrical board with on-board CPU that handles all of the commutation and servo control of a brushless motor. To use it, a robotics control computer can pass it commands through simple electrical interfaces. This section details the design process for creating a system to meet this goal.

5.1 Hardware

All of the electronics for the Brushless Servo Controller were designed and built from scratch apart from the microcontroller development board from ST Microelectronics. Section 5.1 outlines the electrical design as required by the software.

5.1.1 MOSFETs. Any motor control system will need to both source and sink relatively high currents on every coil winding without burning up. MOSFETs (Metal-Oxide Semiconductor Field-Effect Transistors) were chosen to fit this requirement because they are increasingly the closest thing to a electrically controlled solid-state switch that that are available now. Every year, MOSFET technology continues to expand, bringing both cost and power capability. I chose an Infineon part number IRFB7537 due to its incredibly low series resistance for the price. Two of these MOSFETs were used in parallel to robustly handle the power requirements of high power hobby motors. Field-Effect Transistors like MOSFETs,

don't have a saturation voltage like most transistors. Instead, when fully on, they have constant series resistance, R_{ds}. This is important because under normal operation, they can conduct many amps of current while only dissipating standard resistive power losses into heat. Despite all of this, MOSFETs bring several other important factors when programming a motor controller.

First of all, inside the MOSFET package, there exists a parallel body diode. Figure 4 shows this in the MOSFET schematic symbol. This diode is will always conduct current from source to drain if the voltage of the source is higher than that of the drain. When motor windings are energized and de-energized, induction creates large reverse voltage spikes which get conducted through the body diode and get converted to heat. The body diode has a much larger conducting resistance than R_{ds}, so much more of this current gets converted to heat. These diode losses make the MOSFET much less efficient, so minimizing inductive spikes with software is important to avoid burning up the high current circuitry. The PID loops in the synchronous current regulator capped the I_d and I_q errors to a maximum error value so that the voltage wouldn't ramp too quickly causing inductive spikes from the motor armature.

Secondly, MOSFETs include another important thermal loss factor. Field-Effect Transistors have a gate capacitance, so turning fully-on takes time, as a function of how much current can be driven to the gate. While partially on, the drain-source resistance gets ramped to R_{ds} exponentially, but dissipates much more drain-source current during this time into heat. The percent of time spent switching must be minimized to increase power capacity of the inverter. At the same time, however, the windings must be switched fast enough for the armature inductance to average out PWM to avoid conducting full current even at low duty-cycles - the PWM period must be carefully chosen to minimize switching losses while still varying the armature voltage. The PWM period was empirically chosen to be around 3khz.

Finally, MOSFETs are switched by the gate voltage with respect to drain. Figure 5 shows the schematic for a single phase of the MOSFET bridge. The top MOSFET is switched with reference to its output phase rather than V-. To get around this, the gate driver (IRS2184) charges a capacitor in a circuit known as a charge pump. When this capacitor is discharged by the gate, it needs time to recharge again before trying to drive the high-side MOSFET. If the PWM duty cycle is too high, not enough time is given to the gate driver to recharge its charge pump. The field-oriented control PWM lookup tables had to be generated so that the duty cycles would always be low enough so that the MOSFET would always be fully on. This max duty cycle value was found empirically with an oscilloscope, but would be different with different drive circuitry.

5.1.2 Power Inverter PCB. Field-oriented control brings along several electrical requirements. In addition to simply switching on and off the three motor phases: it needed circuitry to be able to listen to the current running through at least two of the three phases. Figure 5 shows the schematic of one of these phases, including MOSFET bridge, gate driver, Hall-Effect current sensor, and voltage sense lines. The inverter also needed to regulate power to the gate drives, the hall effect current sensor, and the microcontroller.

I built and designed the power inverter board to be isolated from the logic circuitry to iron out issues relating to power circuitry,

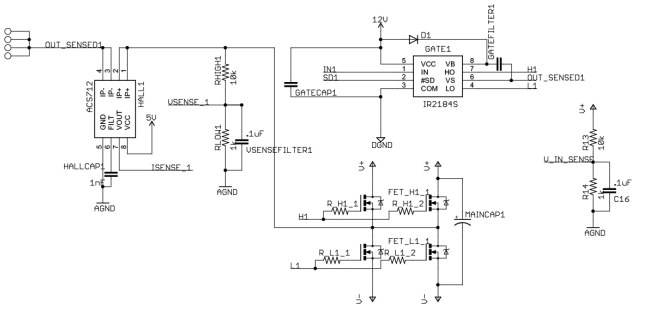

Figure 5: Schematic of one of the phases of the power inverter

separately from issues arising from microcontroller requirements. The printed circuit board was fabricated by `oshpark.com`. Now that the prototype circuitry exists, a future circuit board would incorporate both halves.

As seen from the microcontroller, for each phase there are two logic control lines for the gate driver (`IN`, for high/low sides of the bridge, and `#SD`, for enabling/disabling the bridge), and two 3.3v scaled analog outputs, (`V_SENSE` and `I_SENSE`, respectively).

5.1.3 Mechanical testbed. For development and testing, I used a 400 watt rated 7-pole hobby quadrotor motor (Turnigy Quanum MT Series 4108) mechanically connected to a 1024 position absolute optical encoder. I used the high-resolution optical encoder in place of standard hobby brushless motor hall-effect sensors because field-oriented control requires accurate interpolation between sensor readings, so testing how accurately the state observer *actually* interpolated between low resolution sensor readings was very important to the project.

5.1.4 STM32 Discovery. A typical hobby brushless motor can spin at 50,000RPM. If that motor has 7 poles like the one used on the mechanical testbed, it runs through a single electrical revolution nearly 6000 times per second. Field-oriented control needs to transform to and from a d-q reference frames and run a state observer at this speed, with enough time-step resolution to spare to generate a smooth waveform. For a reasonable output, the fast commutation loop needs to run many hundreds of thousands of times per second. This requires a powerful processor. Small 8-bit microcontrollers like the ones found in hobby brushless motor controllers would not be up to the task. Furthermore, it needs to have enough internal hardware timers to run several high speed PWM outputs, talk to the desktop computer, and run several independent PID loops. It needed many analog to digital converters to read the sensor values. To complicate everything, this project was designed to be expanded so that multiple motors could be controlled by the same processor. The STM32 series microcontrollers were chosen to balance cost and the the quantity of hardware peripherals and processing power.

For prototyping, the STM32-Discovery Board developed by ST Microelectronics was used. This development board uses an STM32-F407 chip, based on an ARM Cortex M4 CPU core, with many additional peripherals helpful for motor control. Although the development board uses up many of the available peripherals for on-board demos, including audio signal processors, accelerometers,

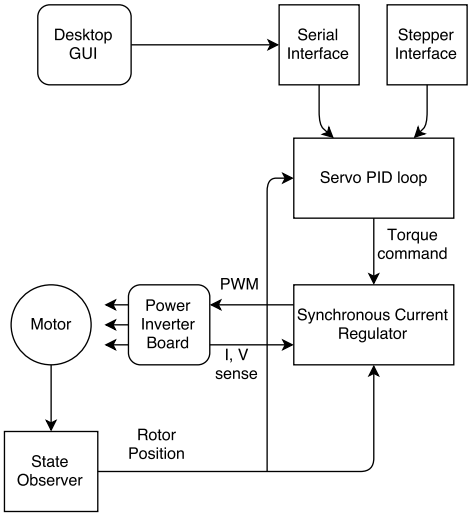

Figure 6: High level system block diagram

etc. unrelated to the motor control project, there was enough usable I/O to drive field-oriented control on one motor using the 10-bit rotary encoder on the mechanical testbed.

5.2 STM32

The STM32 series microcontrollers are tremendously powerful in spite of their price and energy requirements. They are based on an ARM Cortex M4 CPU core, which is typically used in computing systems that run an operating system. Choosing to write software for these CPUs without an operating system turned out to be a bold strategy for such a powerful chip. Much of the time spent in development involved reading through the CPU datasheet to find register addresses to talk to the CPU peripherals.

The firmware was written in C, using the ARM-GCC toolchain, and built using ST Microelectronics StdPeriphLib peripheral library for the STM32.

5.3 Position Sensor Readings

Unlike hall effect sensors around the rotor, the optical encoder used in the mechanical testbed measures physical rotor position instead of electrical angle, so the angle is divided by the number of poles of the specific motor. With the testbed's motor, this gives an electrical resolution of 146 steps, compared to 6 steps with a hall effect sensor. A state observer is necessary for interpolating these sensor readings into something that the synchronous current regulator can work with.

5.3.1 Gray Code. The rotary encoder used for testing doesn't output the shaft position in traditional binary. Instead, as the shaft rotates, only one bit is changed at a time. Gray code is the most common encoding scheme for similar systems. It is especially useful for error correction because a misread would occur if some of the bits change before others in traditional binary. The encoder used in the test-bed used an unknown modified version of gray code, so a lookup table was required to convert to computer readable shaft position.

5.3.2 Kalman Filter. A Kalman filter was initially chosen to observe low resolution/noisy sensor readings, although it was scrapped for a simpler solution. Kalman filters take in sensor data with associated uncertainty, and then predict the state of the system by the next time step. When the sensor data arrives, it performs a weighted average between the new sensor data and the prediction which is then fed back into the state observer by the next time step. Measurements closer to the predicted value are given more weight. Kalman filters are incredibly good at removing measurement noise from a system, but they need a relatively accurate model of how the state will change over time. In a motor control system, this includes acceleration commands and an estimate of the motor's moment of inertia. This model was too computationally intensive to estimate rotor position, especially when spinning at high speeds.

5.3.3 Alpha-Beta observer. An alpha beta filter is also very effective at smoothing out measurement noise, but it doesn't need a detailed model of the system. An alpha beta observer uses two internal states to approximate the system state. One internal state is derived by integrating the value of the second state. The output of an alpha beta state observer uses a weighted average of this estimated system state and the sensor values, which get plugged back into the state observer for the next time step. Alpha Beta filters can be almost as accurate as Kalman filters for systems that involve mechanical position, because they can be easily estimated by integrating velocity. The Alpha Beta state observer I implemented ended up being much less computationally intensive, simpler, and indistinguishable from the Kalman filter output.

5.4 Tuning

Tuning separates a successful servo system from an unusable one. A well-tuned control system reaches the target setpoint quickly and without oscillation. For timing critical applications like CNC Machines, position control must be consistent so that multiple servos move synchronously. For this project, lots of time was spent empirically finding adequate values of P, I, and D terms of both the outer PID loops and inner synchronous current regulator loops. Also, finding adequate values of alpha and beta in the state observer were necessary to make smooth, interpolated rotor position estimation.

Also, because the sensors built into the a motor might not be completely aligned with the stator, it is important to find a correct `TIMING` constant to advance/retard the timing where all of the current is torque-producing.

5.5 Optimization

5.5.1 Modified Synchronous Current Regulator. The inverse Park and Clarke transform inside of the synchronous current regulator needs to be inside of the fastest timing loop for waveform generation, although this process is very computationally intensive. For computational efficiency, it can be replaced by a lookup table. For this project, I implemented Shane Colton's Modified Synchronous Current Regulator[2]. In this solution, an inverse Park Transform is replaced with a simple sine wave generator with a phase advance. Because V_d generated by the D-Axis controller implies a phase shift of the output waveform, we can simply phase shift a sine wave to regulate d-q current. The forward Park and Clarke transform

can take place in the slower timing loop with the inner current regulator controllers, so that only a sine wave generator is inside of the fast commutation loop. This can increase max motor speed many times.

With an oscilloscope, I measured the fast control loop to be running on the order of 7500Khz. At this point, PWM period becomes the limiting factor instead of processing power for motor speed. This also leaves more than enough processing power to commutate multiple different motors at the same time at very high speeds.

5.5.2 ADC DMA. The STM32 comes with many optimization features that help move some operations away from the main CPU loop to free processing power for more important logic[8]. This motor driver needs to continuously listen to seven analog values. The analog-to-digital converter (ADC) takes time to read analog values which can use up lots of processing time. Instead, I utilize the direct-memory-access register(DMA) of the chip. The DMA can continuously drive the on-chip ADC and move the converted value to seven distinct locations in memory - in the background. With the DMA, I can read the most recent converted analog value from any of the seven sensors, instantly.

5.5.3 Timers. Timing the control flow of the Brushless Servo Controller was necessary for conserving processing power for things that need it. Some parts of the Brushless Control System need to be executed faster than others. Commutation, for example needs to be able to always be updating the waveform at the fastest speed possible, and the state observer needs to accurately interpolate rotor position for the synchronous current regulator to generate a waveform. The inner PID loops don't need to be regulating I_d and I_q quite as quickly however, because I_d and I_q are DC values. The outer PID loop to control torque is much slower still, because the physical mechanical system is much slower to respond. Logic to control the serial input and output buffers are also running in this slower timer because serial communication is not critical for timing.

In firmware, the high speed commutation and state observer logic is running at full CPU load in the main loop, and there are two timers that control slower program flow. A 1Khz timer regulates the inner PID loop, and a 244Hz timer keeps track of timing and the outer servo loop.

5.6 Desktop App

Servo target velocity, position, and torque values can be sent over the serial port to command the motor. Real time motor data is also sent back as the motor is spinning to evaluate efficiency, monitor position/velocity and current. A Java desktop app was written to analyze this data and to test the servo. Figure 7 shows a screenshot of the main window. The bottom right corner shows a scatter plot of actual phase currents with respect to the electrical angle of the rotor. With Field-oriented control, this should be a perfect sinusoid with the minimum/maximum centered around 180°. This scatter plot is helpful for fine-tuning timing as well as testing the optimality of the drive current waveform. Total drive current is a signed sum of I_d and I_q, where negative values indicate a reverse torque.

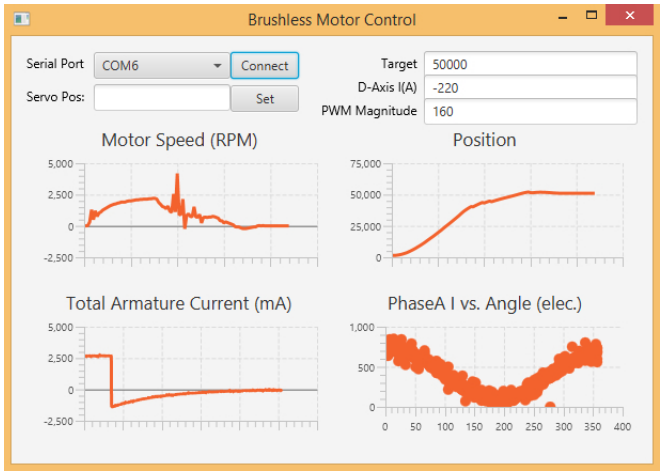

Figure 7: Brushless Motor Control GUI

The GUI was written using JavaFX and uses the Java-Simple-Serial-Connector(JSSC) for serial communication to the microcontroller. JSSC fires an event listener when bytes are received from the serial port, where comma-separated values from the development board are parsed and plotted on graphs.

5.7 External Interfaces

To send commands to the Brushless Servo Controller, a user can choose one of two interfaces.

5.7.1 Stepper Interface. The Brushless Servo Controller is designed to seamlessly replace stepper motor systems. Stepper motor drivers typically use a two-wire digital interface - `STEP` and `DIR`. The `STEP` line moves a fixed number of degrees every time the line is pulled high. The `DIR` line defines the direction a step is taken. The size of a step is defined in the `STEP_DISTANCE` constant. Unlike actual stepper drivers, this interface can also be used to control velocity or torque setpoints.

5.7.2 Serial Interface. A user can also talk to the controller using standard 9600 baud UART serial. A servo command always starts with a `S` character and ends with a newline. For example, to move the servo to position 1000, a user would enter `S1000`. To select between torque, position, or velocity mode, a user would enter `ST`, `SP`, or `SV`, respectively.

The `tx` line will always be outputting real time motor data with a series of comma-separated values in the form of: `>RPM, POS,` I_{tot}`, ANGLE,` I_a`, TARGET,` I_d`, PWM_MAGNITUDE`.

6 EVALUATION

The purpose of this project is to dramatically bring down the cost and ease of use of existing industrial servo systems. Using off-the-shelf hobby motors in place of purpose-built servo motors definitely achieves this goal, but more work would be needed to make this controller a viable alternative to commercial servo systems.

6.1 State Observer evaluation

To emulate the sensor readings from a hall-effect sensor built into hobby motors, a state counter was incremented every 60 electrical degrees. With some trial-and-error state observer adjustment, the state observer estimation of rotor position was indistinguishable from the full high-resolution measured rotor position, even during acceleration. The driver will be able to accurately work with low-resolution hall effect sensors, meaning that off the shelf hobby motors will be able to be used as industrial servos.

6.2 Tuning evaluation

Without more tuning, this system would not be very helpful in many timing-critical applications like CNC machines. The servo loop currently doesn't reach target position quickly and overshoots its target by a few degrees. More tuning would be necessary to make this project actually competitive with existing servo systems. Compounding this problem, the power supplies I used for testing could not make generate much torque on the motor, so the servo PID weights had to be artificially low to avoid hitting the limits of the power supply. Tuning with new power supplies would be necessary for improving this system.

7 DISCUSSION

The brushless control methodology and servo logic used in this project is nothing new. Controllers regulating torque output of brushless motors use field-oriented control and the process is relatively well documented. This project makes a general purpose servo control system using these well-understood control methodologies and brings them into the realm of the hobbyist by using cheaply available hardware. Using hobby brushless motors is new for industrial servo control, but there is nothing stopping these motors from performing precise servo control like more expensive servo motors. This project bridges that gap. Although it still needs work to bring it to the level of commercially available servo controllers, it has potential to be very useful to the hobbyist community for its cost and wide variety of applications.

8 FUTURE WORK

The brushless servo system I have developed has lots of room for improvement. This system is not a competitive servo controller yet. Apart from tuning the control loops, several architecture changes would be required before this product could be useful to the public.

8.1 Hardware Improvements

The first fundamental change that needs to be developed to make this project a finished product is to finalize the circuit board development. Currently the STM32 development board is connected with a prototyping board and ribbon cable to the power electronics. A final circuit board layout would include all logic circuitry and power electronics in the same layout. The only inputs would be interfaces to the controller, and the only outputs would be motor and power connections.

Secondly, combining multiple power inverters into the same PCB with one logic system would dramatically decrease the cost to drive multiple motors.

8.2 Upgraded PC interface

The serial interface provides a convenient system for driving position, velocity, and torque, but it is not sensitive to timing. There is no clock that keeps multiple servo systems synchronized. Many industrial servo systems have proprietary interfaces that control target position, velocity, and torque at specified times. While the stepper interface correctly works with timing, it requires the external controller to be constantly commanding new setpoints. A computer interface that keeps timing consistent would be a useful improvement.

8.3 Auto-Tuning

Several commercially available servos have auto-tuning routines that vibrate the rotor with white-noise current and observe how the system responds. These systems can establish PID gains for the servo loop and for the commutation loop, as well as filter out resonant frequencies. A routine like this would save hours of trial-and-error tuning for a specific mechanical system.

8.4 Desktop App changes

Currently the Desktop application for monitoring and controlling servomotors is a minimal proof of concept. A final product would allow you to change configuration parameters and tune the control system for different mechanical systems.

The desktop app could also include programmable-logic-controller (PLC) functionality. A user could set up simple timing sequences to drive multiple motor controllers at once.

9 CONCLUSION

Although servo motors are the primary means of actuating industrial robots, they are largely unavailable to the hobbyist. Industrial Servo motors are typically sold in two parts - a controller and dedicated motor/sensor unit. This project aims to provide a general purpose servo controller designed to use brushless motors designed for hobby RC cars in place of the dedicated motor/sensor unit to save costs. Despite being ridiculously cheap, these motors are incredibly power-dense, efficient, low on friction, and can have an incredibly flat torque response. Most importantly, they are also sold with small hall-effect position sensors cast into the armature for commutation. These sensors can double as position sensors for a servo motor. Unfortunately, as far as I know, no commercially available servo driver can drive these motors.

This project aims to provide consistent, precise control of the position, angular velocity, and torque of these motors for robotics applications. It is designed to be modular and controllable to work with a large variety of applications. While existing industrial CNC servo controllers couldn't drive the motors in auto-stabilizing camera gimbals, which require precise torque control at incredibly low angular velocities, this project utilizes the wide variety of hobby motors available to enable use in applications like this as well.

In general, for hobby robotics applications (CNC machines, plotters, etc.), stepper motors are used in place of servos. These motors have many shortcomings when compared to servos, but can be found for relatively cheap compared to an industrial servo system. Motors designed for RC cars can be even cheaper. A brushless motor and general purpose servo driver like the one outlined in this

project could make a much more powerful system for less money than a stepper motor and controller, and could drop in place of an existing stepper setup.

This project has been an incredible learning experience for me. In spite of many frustrating hiccups, it has been extremely rewarding and I believe I have come up with a system that is very useful for low-cost industrial robotics.

10 ACKNOWLEDGEMENTS

Due to incomplete documentation of motor control theory, this project would not be possible without the help of Shane Colton, who did his master's thesis work on field-oriented control. He was very responsive over email, and helped clear up many conceptual issues I had with brushless motor control.

The authors of `visualgdb.com` were also indispensable in providing otherwise lacking STM32 documentation and tutorials.

I'd also like to thank Prof. Brad Richards for overseeing my Capstone research this semester. The opportunity to work on a technical engineering project of this scale as part of my education has been a great experience.

REFERENCES

[1] Shane W. Colton *Design and Prototyping Methods for Brushless Motors and Motor Control*. Massachusetts Institute of Technology, 2010.
[2] Shane W. Colton *A Modified Sychronous Current Regulator for Brushless Motor Control*. Massachusetts Institute of Technology, 2011.
[3] James Robert Mevey *Sensorless Field Oriented Control of Brushless Permanent Magnet Synchronous Motors*. Kansas State, 2009.
[4] Principles of PID Control and Tuning, `http://www.eurotherm.com/temperature-control/principles-of-pid-control-and-tuning`
[5] Bilal Akin and Manish Bhardwaj *Sensorless Field Oriented Control of 3-Phase Permanent Magnet Synchronous Motors*. Texas Instruments, 2013.
[6] Greg Welch and Gary Bishop *An Introduction to the Kalman Filter*. University of North Carolina, 1995.
[7] VisualGDB: Developing STM32 projects with Visual Studio, `https://visualgdb.com/tutorials/arm/stm32/`
[8] ST microelectronics *STM32F407xx Datasheet - production data*. 2014. DM00037051