Design of a Brushless Servomotor for a Low-cost Compliant Robotic Manipulator

Allan Zhao

Acknowledgement

I would like to thank my advisor Professor Pieter Abbeel for his support and guidance throughout my time as a member of his group. I would also like to thank David Gealy, Dr. Stephen McKinley, Brent Yi, Philipp Wu, and Eric Zhou for their invaluable contributions in mechanical design and firmware development.

Design of a Brushless Servomotor for a Low-cost Compliant Robotic Manipulator

by

Allan Zhao

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2018

Design of a Brushless Servomotor for a Low-cost Compliant Robotic Manipulator

Abstract

Design of a Brushless Servomotor for a Low-cost Compliant Robotic Manipulator

by

Allan Zhao

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

The design of a brushless servomotor as well as its application in a low-cost, compliant 7-DOF robot arm are presented. The recent popularity of consumer drones has led to a significant reduction in the cost of high-torque brushless gimbal motors. These motors offer a unique set of advantages for robotic manipulation, but are not typically used for this purpose. A servomotor based on brushless gimbal motors is designed and implemented, including custom electronics and firmware. Cost reduction is achieved by using commodity components and taking system-level considerations into account. The servo achieves active compliance using velocity sensing and high bandwidth field oriented control. Design decisions and tradeoffs are discussed.

Contents

Acknowledgments

I would like to thank my advisor Professor Pieter Abbeel for his support and guidance throughout my time as a member of his group. I would also like to thank David Gealy, Dr. Stephen McKinley, Brent Yi, Philipp Wu, and Eric Zhou for their invaluable contributions in mechanical design and firmware development.

Chapter 1

Introduction

Compliant robot arms can safely interact with their surroundings, making them popular in collaborative and personal robotics. Most commercially available compliant robot arms use compact, high-speed brushed DC or BLDC servomotors, which require high reduction gearboxes to achieve a reasonable amount of torque. However, high reduction ratios result in poor backdrivability and low intrinsic compliance. Adding compliance usually requires additional sensors [12] or a significant increase in mechanical complexity [5], both of which increase the overall system cost.

Motors with inherently high torque minimize the need for gear reduction, but tend to be heavy, bulky, or prohibitively expensive. Examples include stepper motors, "torque" motors, and frameless motors. The recent availability of commodity brushless gimbal motors, arising from the popularity of consumer drones, resolves some of these disadvantages. Brushless gimbal motors are typically used in a direct-drive configuration to stabilize cameras on drones, and are optimized for high torque output at low speeds. They are also relatively lightweight and inexpensive. As a result, they are uniquely suited for building robotic actuators.

We present a design for a servomotor based on brushless gimbal motors, and examines its suitability for constructing a low-cost, compliant robot arm. The servomotor is composed of a brushless gimbal motor, mounting plate, and custom-designed controller board. The controller board runs firmware implementing field-oriented control (FOC) for precise torque control and active compliance. Multiple servos can communicate with a host computer over a single RS-485 multidrop serial bus, simplifying wiring.

Eight of these servos are used to drive the seven degrees of freedom and end effector of a robot arm currently under development. They contribute substantially to the compliance and low cost of the arm as a whole.

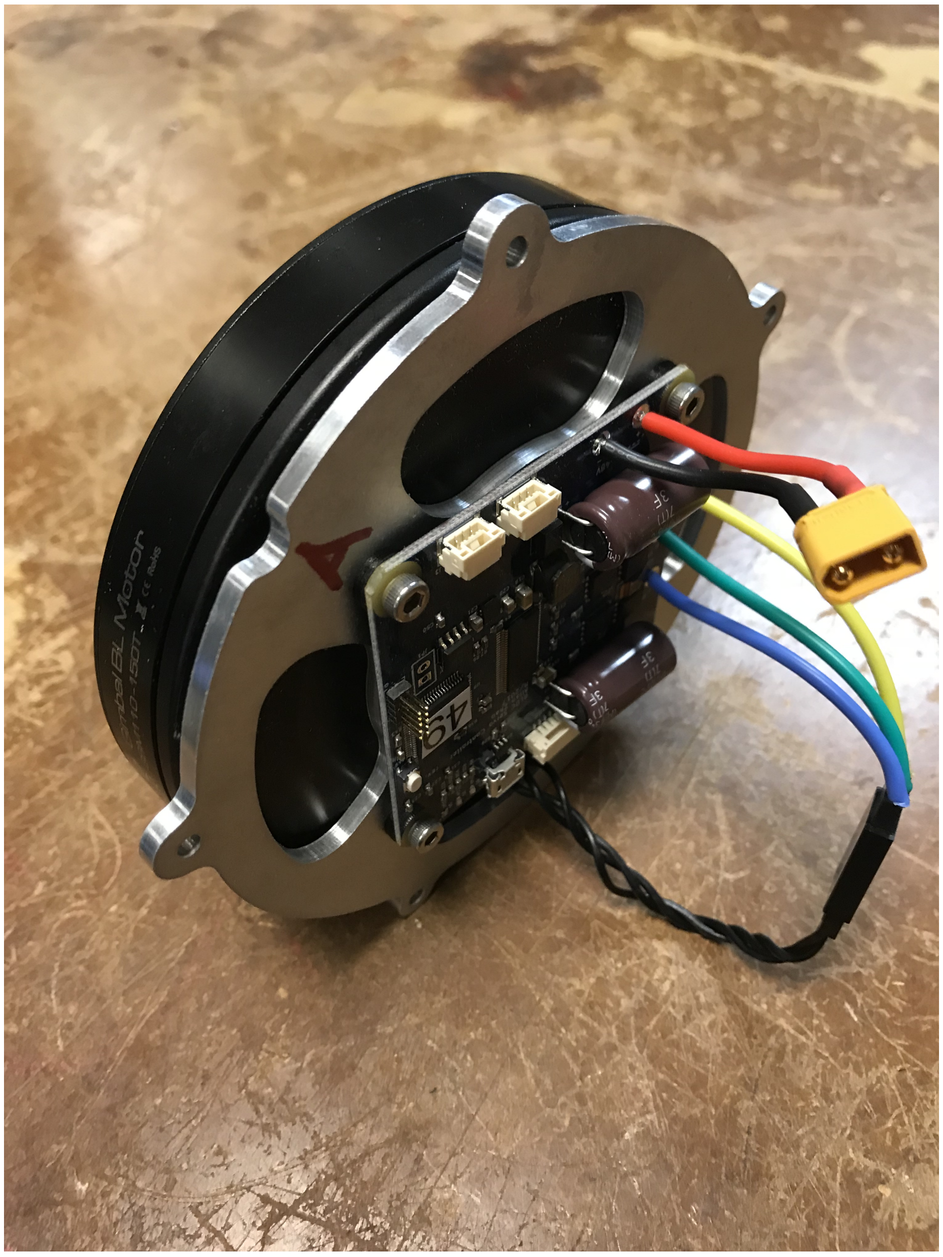


Figure 1.1: Brushless servomotor

Chapter 2

Related Work

2.1 Brushless Servomotors

The concept of using brushless servomotors with minimal gearing, in order to reduce cost and/or increase compliance, has significant precedent.

Stepper motors can be considered a type of high-torque brushless motor, and have been widely used as inexpensive quasi-direct-drive robot actuators. Examples include the low-cost compliant 7-DOF arm by Quigley et al. [6], which uses stepper motors for the proximal 4-DOF, and Walter [1], which uses stepper motors for all 6 DOFs. One significant disadvantage of using stepper motors in a robot arm is their relatively high mass, which increases the torque required to counteract gravitational forces. Additionally, high swinging mass compromises safety in the event of a collision. Quigley et al. address this issue in their design by using lightweight geared servos in the distal 3-DOF, at the cost of some compliance.

The use of three-phase brushless motors for quasi-direct-drive robotic actuation has also been explored extensively. Compared to a stepper motor with comparable torque output, a three-phase brushless motor tends to be more expensive but lighter in weight. Three-phase brushless motors also tend to have significantly lower cogging torque. Combined with a highly transparent transmission, this allows a torque control loop to be closed by sensing motor current only. The WAM Arm [9] achieves closed-loop force control without force sensors using this technique. Although they are not robot arms, the MIT Cheetah [7] and GOAT leg [3] are other examples of this approach.

2.2 Motor Controller Boards

The VESC [10] and ODrive [11] are open-source brushless motor controllers commonly used in hobbyist robotics. The VESC is mainly targeted towards electric skateboards, while the ODrive was designed from the start to be a low-cost brushless servomotor controller. Both support servomotor operation, including torque control, with the addition of an absolute encoder.

We considered both controller boards for the robot arm project, but found neither to be suitable. Due to its high power handling capability (>1 kW) and support for two motors, the ODrive is physically large and would have to be placed in the base of the robot. One motor cable and one encoder cable for each joint would need to be routed through the base of the arm. The VESC is much more compact, and could be placed next to the motor it controls. This would allow routing only a single power cable through the arm. However, the VESC would still require a separate encoder board. Both the VESC and the ODrive have significantly higher current ratings than required for our project, increasing the size and cost of components.

The all-in-one design of our motor controller board was inspired by the Mechaduino [4], an open-source stepper motor servo controller. The Mechaduino integrates an absolute magnetic encoder on the same board as the microcontroller, reducing the number of separate circuit boards. It is also attached to the back of the motor it controls, creating a modular unit that simplifies wiring. Our motor controller board was designed with a similar form factor, but with the ability to drive a brushless motor instead.

Chapter 3

System Design

3.1 Motor Controller Board

The motor controller board (MCB) is a four-layer PCB and was designed in KiCad [2], a free and open-source EDA package. It is rigidly mounted to the brushless motor using an aluminum plate and spacers. The MCB integrates all of the electronic components necessary to control a brushless motor onto one PCB, including an absolute magnetic encoder, microcontroller, three-phase inverter, current sensors, RS-485 transceiver, inertial measurement unit (IMU), and temperature sensor. This enables significant BOM reduction and improves manufacturability.

The AS5047D 14-bit magnetic encoder detects the angular position of a diametric magnet attached to the rotor of the brushless motor. It was chosen due to its SPI interface, high resolution, and high output data rate (upwards of 40 kHz).

The STM32F405 microcontroller runs the custom firmware and control code, described in the section "Firmware and Controls". It has a clock speed of 168 MHz and a hardware floating point unit, enabling complex control algorithms to be implemented.

The three-phase inverter is based on a Texas Instruments DRV8301 gate driver IC, and drives the three phases of the brushless motor.

Three INA240 bidirectional current sense amplifiers measure the current in the three motor phases, by amplifying the voltage across a 0.002 ohm sense resistor in series with each phase. They were chosen due to their high common-mode rejection and ability to measure current in both directions.

The ST3485 RS-485 transceiver converts between the single-ended, logic level signals of the microcontroller's UART peripheral and the differential signals on the shared RS-485 bus. Because each servo can have different ground potentials due to voltage drop, the large -7 to 12 V common mode input voltage range is necessary. The ST3485 also provides built-in ESD protection, improving system robustness.

The LSM6DS3US IMU provides acceleration and angular velocity measurements, which can be read by the host computer.

An LM75B temperature sensor reports the temperature of the board in degrees Celsius. Because the MCB is placed directly against the motor, this provides a reasonable approximation of the actual motor temperature. The MCB can be configured to perform a safe

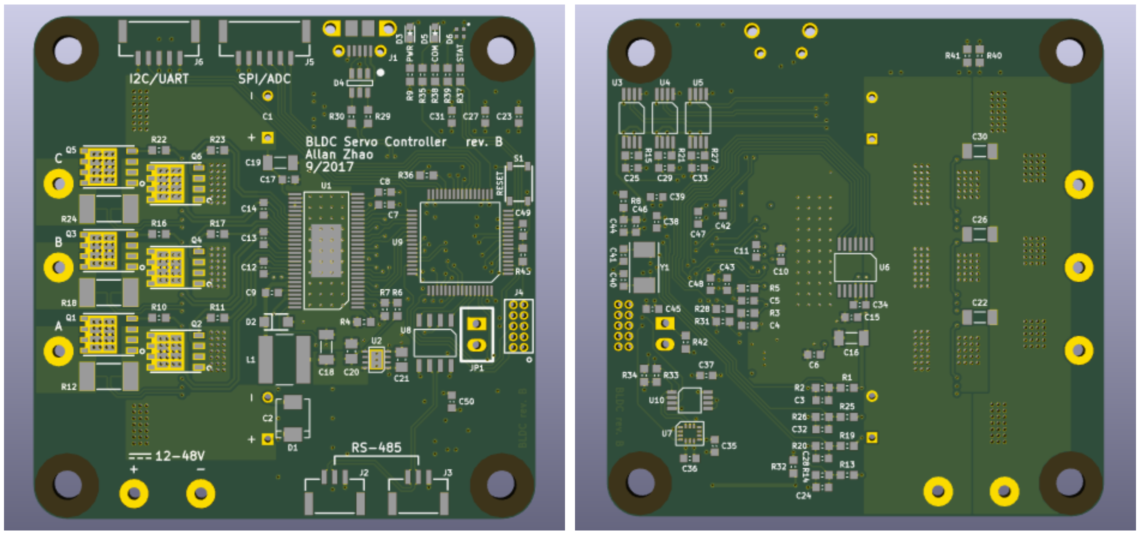


Figure 3.1: Motor controller board renderings (front and back)

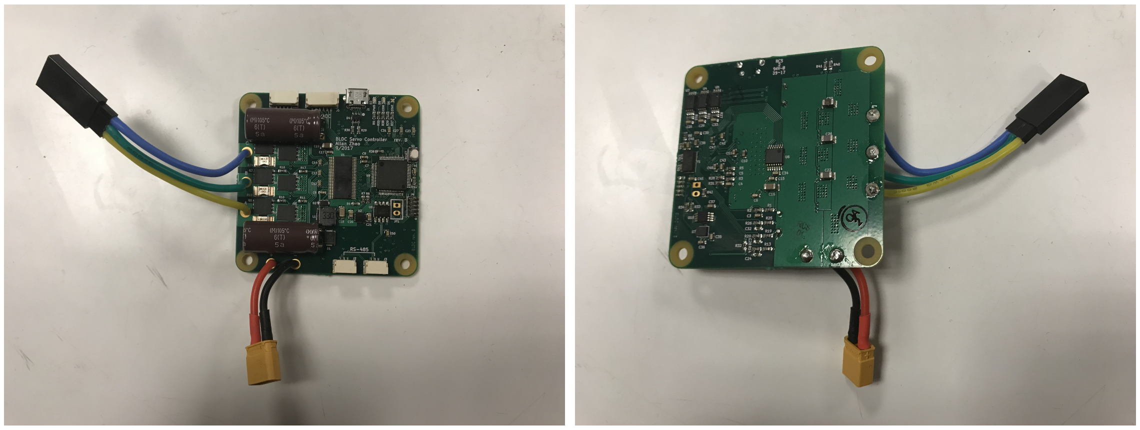


Figure 3.2: Assembled motor controller board (front and back)

shutdown if the temperature reading becomes too high.

By placing the encoder on the same board as the microcontroller, no separate encoder board and cable are required. Additionally, the close proximity of the MCB to the motor allows the motor cable to be trimmed to only a few centimeters in length, reducing electromagnetic interference (EMI).

3.2 Cabling

Each servo requires only two outside connections: DC power and RS-485 communications. Additional expansion ports are provided for connecting buttons, LEDs, etc. Multiple servos can be connected together in a linear bus topology, minimizing the amount of wiring needed. Power supply voltage drop is automatically compensated by measuring the DC bus voltage and adjusting PWM duty cycles accordingly. Twisted pair cabling is used for the RS-485

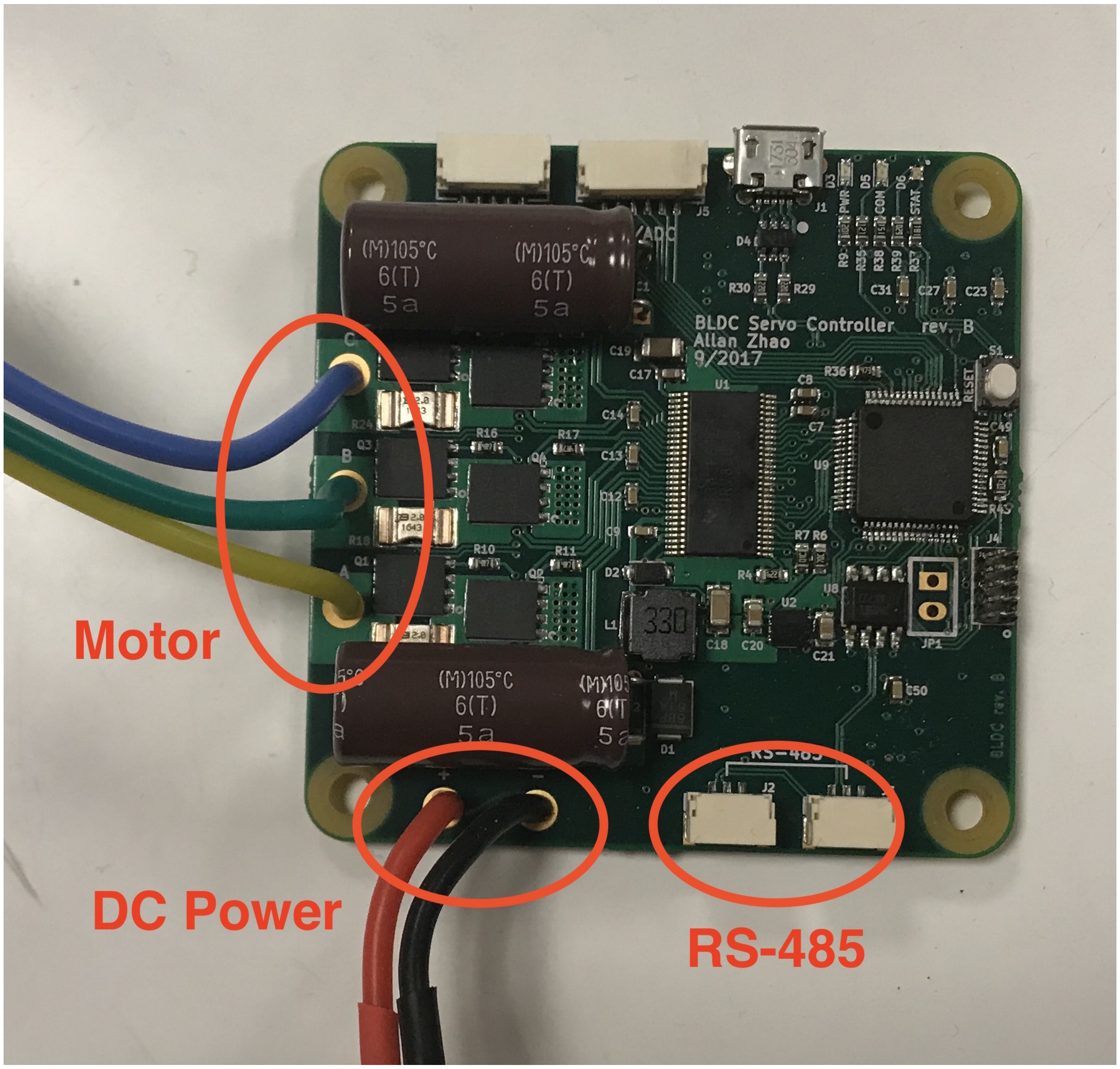


Figure 3.3: Motor controller board with connections labeled

bus to improve signal integrity.

3.3 Firmware and Controls

The motor controller firmware is written in C and C++ and based on the ChibiOS real-time operating system [8]. One thread handles serial communications, while a high-priority thread runs the main control loop.

Motor calibration values are stored in flash memory, and are not lost when power is removed. They include the offset of the encoder magnet relative to the phases in the motor stator, the number of electrical revolutions per mechanical revolution[1], motor resistance[2], motor torque constant, and other parameters that may vary between motors.

Firmware can be updated over the RS-485 bus using a custom bootloader, making physical access to the motor controller boards unnecessary. The bootloader can also be updated in a similar fashion.

The main control loop runs at 20 kHz (same as the PWM frequency), and performs cascaded PI current, velocity, and position control. Using the technique of field oriented control, the phase currents are transformed into two orthogonal components in the rotor's reference frame. The direct component, aligned with the magnetic flux in the motor, is controlled to zero[3]. The quadrature component, perpendicular to the magnetic flux in the motor, affects the torque.

In torque control mode, the setpoint of the quadrature controller is directly set by the host computer. In velocity control mode, it is set by the velocity PI controller. In position control mode, both the torque and velocity setpoints are generated internally.

Step responses for the position and velocity controllers, with an unloaded motor, are shown in Figure 3.4. Note that the controller outputs are saturated during the ramp-up phase, requiring integral anti-windup.

The psuedocode in Figure 3.5 describes the basic structure of the control loop.

The motor generates back-EMF voltage when it rotates, producing a current (and torque) that opposes its motion. This results in dampening and negatively affects the backdrivability of the motor. Although the FOC current controller is able to cancel out this back-EMF-induced current fairly quickly, it still helps to have a feed-forward correction term. Back-EMF compensation is implemented by adding a term to the quadrature current PI controller's feed-forward voltage that is proportional to velocity:

$$V_{q,ff} = I_{q,sp} R_{motor} + \omega K_T$$

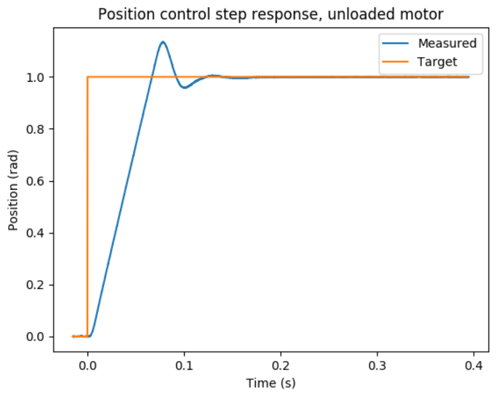
where $I_{q,sp}$ is the FOC quadrature current setpoint, R_{motor} is the motor resistance, ω is the angular velocity, and K_T is the motor torque constant.

As shown in a spin down test (Figure 3.6), back-EMF compensation reduces damping significantly. A motor is accelerated to a velocity of 10 rad/s in velocity control mode, followed by a torque command of zero. With back-EMF compensation, the motor continues

[1]The number of poles in the motor divided by 2.

[2]Resistance between the terminals of an equivalent ideal DC motor

[3]The speed of the motor was sufficient for our application, so field weakening operation was not implemented.



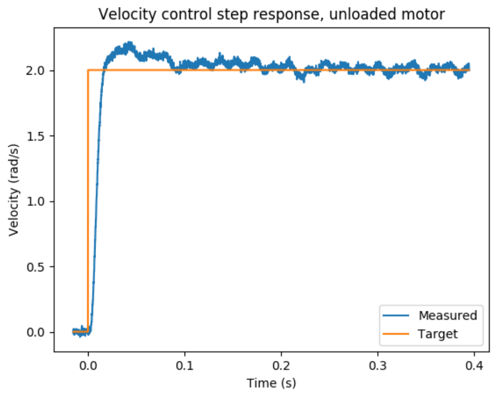


Figure 3.4: Step responses with an unloaded motor

```
function RunInnerControlLoop
    while true do
        ReadEncoder;
        if no recent communication then
            BrakeMotor;
        end if
        EstimateState;
        RunPositionControl;
        RunVelocityControl;
        RunCurrentControl;
    end while
end function
```

Figure 3.5: Main control loop psuedocode

to coast for more than half a second. Without back-EMF compensation, the motor stops almost immediately. Reduced damping due to back-EMF compensation makes the servo easier to backdrive, improving compliance.

3.4 Calibration

Each servomotor goes through an initial calibration procedure after assembly to produce motor calibration values. The main purpose of calibration is to find the relationship between encoder values and the position of the rotor, as the encoder magnet can be mounted in an arbitrary orientation. The calibration procedure involves stepping through the phases of the motor in open-loop mode, as if it were a stepper motor. The encoder value is recorded at each step. From the encoder values, it is possible to derive the encoder magnet offset and the number of poles in the motor.

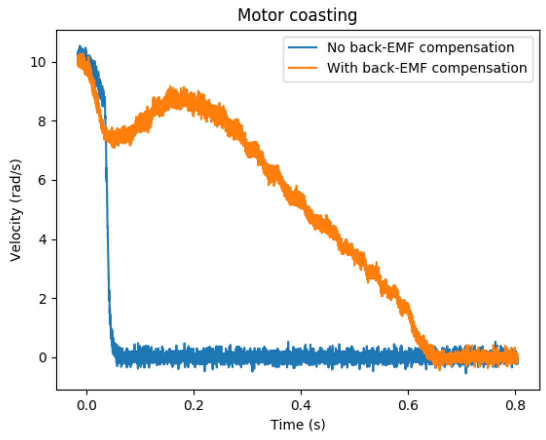


Figure 3.6: Motor spin down test

3.5 Communications

The motor controller boards communicate with a host computer, over an RS-485 bus, using a custom serial protocol. Each board is assigned a unique ID, which can be used to address it individually. This unique ID is stored in flash memory in a similar way to the motor calibration values. All values that can be written to or read from a board are abstracted into virtual registers. This includes commands (current setpoint, duty cycle, etc.), motor calibration values, and sensor readings. Any number of registers can be written or read in a single command, and both reading and writing can be done in a single request. Currently, only one servo can be addressed per packet.

Using a FT232RL-based USB to RS-485 adapter, roughly a thousand packets can be sent per second from a computer running Ubuntu 16.04 Linux. For a robot arm with eight servos, this translates to a whole-arm update rate of 125 Hz.

The structure of communication packets is shown in Figure 3.7.

Request Packet							
	Sync Flag	Protocol Version	Message Length	Board ID	Function Code	Payload	CRC
Size (bytes)	1	1	2	1	1	n	2

Response Packet								
	Sync Flag	Protocol Version	Message Length	Board ID	Function Code	Errors	Payload	CRC
Size (bytes)	1	1	2	1	1	2	n	2

Figure 3.7: Communications packet structure

Chapter 4

Applications

4.1 Robot Arm

These servos are used in a low-cost, compliant robot arm currently under development. The robot consists of a base unit containing one servo, three nearly-identical links containing two servos each, and an end effector containing one servo for a total of eight servos per arm. The arm has seven degrees of freedom, excluding the end effector.

The eight servos are connected to a 48 V power supply using 18 AWG silicone-insulated wire. They communicate with a host computer running the Robot Operating System (ROS) over a single twisted pair RS-485 bus.

The host computer interfaces with the RS-485 bus using an off-the-shelf USB serial adapter. A ROS node implements our custom serial protocol, and translates between ROS messages and packets on the bus.

The components on the motor controller boards were carefully placed so that two boards can nest within each other, as shown in Figure 4.1. This allows two servos to be placed back-to-back in a compact fashion, reducing the arm's profile.

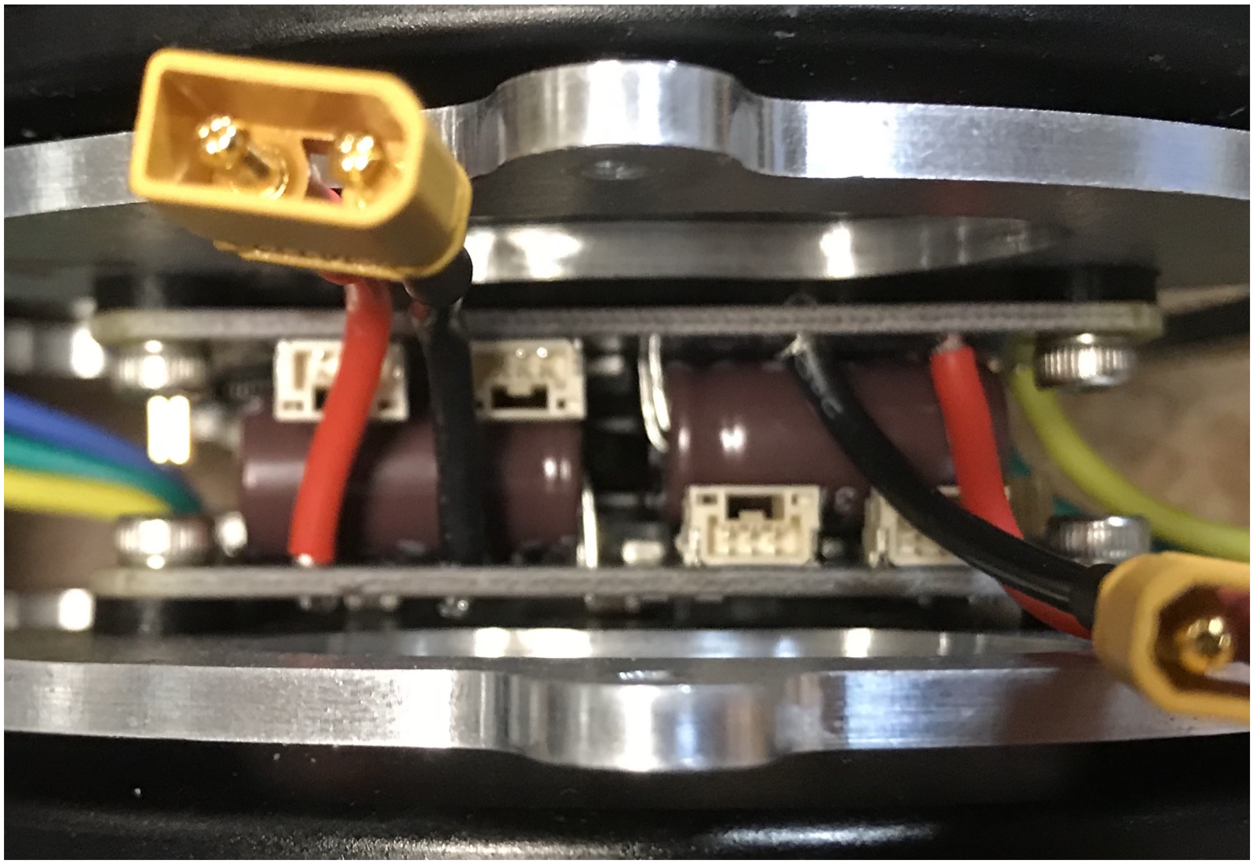


Figure 4.1: Two servomotors nested together

Chapter 5

Conclusions

The design of a brushless servomotor and its application in a low-cost, compliant 7-DOF robot arm was described. The servos are constructed using brushless gimbal motors, taking advantage of their high torque and recent affordability. Integrated into a 7-DOF robot arm, the servo contributes significantly to the arm's compliance and reduces overall system cost. Active compliance is achieved using feed-forward back-EMF compensation, while cost reduction is achieved by using commodity components and taking system-level considerations into account.

Bibliography

[1] Jochen Alt. Walter. http://walter.readthedocs.io/en/latest/. Accessed: 2018-05-17.

[2] KiCad Developers. Kicad eda. http://kicad-pcb.org/. Accessed: 2018-05-16.

[3] Simon Kalouche. Design for 3d agility and virtual compliance using proprioceptive force control in dynamic legged robots. Master's thesis, Carnegie Mellon University, 2016.

[4] Tropical Labs. Mechaduino - tropical labs. http://tropical-labs.com/index.php/mechaduino. Accessed: 2018-05-17.

[5] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995.

[6] M. Quigley, A. Asbeck, and A. Ng. A low-cost compliant 7-dof robotic manipulator. In *2011 IEEE International Conference on Robotics and Automation*, pages 6051–6058, May 2011.

[7] Sangok Seok, Albert Wang, David Otten, and Sangbae Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1970–1975. IEEE, 2012.

[8] Giovanni Di Sirio. Chibios free embedded rtos. www.chibios.org. Accessed: 2018-05-17.

[9] Barrett Technology. About the wam arm barrett technology. https://www.barrett.com/wam-arm/. Accessed: 2018-05-17.

[10] Benjamin Vedder. Vesc open source esc — benjamin's robotics. http://vedder.se/2015/01/vesc-open-source-esc/. Accessed: 2018-05-17.

[11] Oskar Weigl. Odrive. https://odriverobotics.com/. Accessed: 2018-05-17.

[12] Chi-haur Wu and Richard P Paul. Manipulator compliance based on joint torque control. In *Conference on Decision and Control including the Symposium on Adaptive Processes*, volume 19, pages 88–94. IEEE, 1980.