FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of a Servo Motor Optimised for Robotic Applications

João David Soares Silva

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Paulo José Cerqueira Gomes da Costa

Second Supervisor: José Alexandre Carvalho Gonçalves

February 24, 2017

Abstract

Servo motors are actuators that allow for position control. They consist of a DC motor coupled to a reduction gearbox, a position feedback sensor and the electronic circuit responsible for decoding the pulse width signal reference and driving the motor itself. They move from one position to another nearly at maximum speed which can cause abrupt movements and current spikes in the power supply. By using potentiometers as position sensors, they offer a narrow sense range and usually can't be used in continuous rotation mode. For that reason, most can only position the motor shaft between 0° and 180°.

SMORA (**S**ervo **M**otor **O**ptimised for **R**obotic **A**pplications) consists of custom hardware and firmware that includes a microcontroller and a series of sensors, allowing for the motor's current, temperature and voltage to be measured in real-time, as well as precise position feedback thanks to the included hall-effect magnetic position encoder. It also includes an accelerometer and a gyroscope to measure the servo's body relative position and rotation.

This thesis focuses on the development of SMORA from a hardware, firmware and software perspective. The implemented hardware and communication protocol will be described and the PID and speed compensation algorithms used for the control explained and demonstrated.

Acknowledgements

I would like to start by thanking my advisor Paulo Costa for his constant support, availability and dedication, as well as experience and advise. He was the main proponent for this thesis. Also to my co-advisor José Gonçalves for his helpful tips and support.

This work would not have been possible without the constant support of friends and family. To the Harding, Silvéria and Mason family, who were personally invested in the success of this thesis. To the Goldstraw family who day in and day out supported me and my quirks. To all the Riverside family, who's constant prayer and encouragement were invaluable. To my parents for giving me this priceless opportunity. Thank you all for your friendship and support.

Thank you Fia for all this years of patience, endurance and loyalty.

Thank you Lord.

João Silva

“A wise man will hear and increase learning,
And a man of understanding will attain wise counsel”

Proverbs 1:5

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog to Digital Converter
API	Application Programming Interface
BOM	Bill of Materials
CDC	Communication Device Class
DC	Direct Current
DMP	Digital Motion Processor
DXF	Drawing Exchange Format
EEPROM	Electrically Erasable Programmable Read-Only Memory
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
ICSP	In-Circuit Serial Programming
LDO	Low-Dropout
LED	Light Emitting Diode
MEMS	Micro-Electro-Mechanical Systems
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PCB	Printed Circuit Board
PID	Proportional Integral Derivative
PIV	Proportional Integral Velocity
PWM	Pulse-Width Modulation
QFN	Quad Flat No leads
RC	Radio Controlled
RGB	Red Green Blue
RTS	Ready-to-Send
SCARA	Selective Compliance Articulated Robot Arm
SCL	Serial Clock Line
SDA	Serial Data Line
SMORA	Servo Motor Optimised for Robotic Applications
SRAM	Static Random Access Memory
STL	STereoLithography
SWD	Serial Wire Debug
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Chapter 1

Introduction

This chapter will give a context to RC servo motors in robotic applications as well as explain the motivation for creating an improved version.

1.1 Context

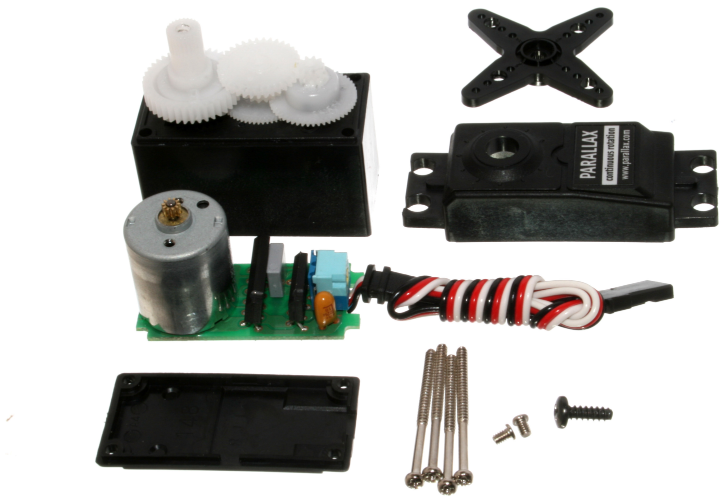


Figure 1.1: RC Servo disassembled showing its main elements [1]

A servo motor (Fig. 1.1) is a rotary/linear actuator, the purpose of which is to allow precise control of angular/linear position. It is comprised of a DC motor coupled with a reduction gear system and a position feedback sensor for closed-loop control. Although sometimes employed in industrial applications, their most common use case is in radio-controlled models such as planes and cars. These RC servo motors are distinct from the industrial type mostly by their lower torque, size and cost and more basic interface capabilities.

Their ease of use, high torque quality, convenient package and low price allow for a wide adoption in the field of robotics. Tasked to move and maintain itself at a particular position, most servos have a range from 0° to 180° when unmodified. By their self-integrated nature, they eliminate the need to: design a control system; analyse the transient response; fine tune the feedback loop; determine the proper gear ratio for a specific speed and efficiency; choose an appropriate motor; build the amplifier and motor driver.

Servo motors generally have three connection wires: power, ground and signal. The signal line is fed with a PWM signal (pulse width modulation). The width of the pulse is then used to determine the angle to which the device should move. A short one millisecond pulse sets it to 0° while a longer two millisecond pulse sets it to 180°.

1.2 Motivation

Even though servos have excellent qualities when used in the field of robotics, several properties can still be enhanced. These devices are known to have nonlinear properties and dynamic factors such as dead-zones, backlash and friction that might hinder their precision and accuracy. They usually do not provide position, velocity or current feedback to the device controlling them. Factors such as a change in the load attached to the system increase the complexity of the analysis even further. Controlling these servos using a PWM signal might also become complicated if the number of servo motors to control increases.

The intent of this thesis is to rebuild a servo motor comprising several sensors and higher level addressing/communication capabilities. Also studying it's characteristics through experimental analysis.

Chapter 2

Literature Review

In this chapter, some methods for model parameter extraction necessary for an accurate control of the DC motor, controller architectures, as well as existing hardware which deploy some of these technologies will be reviewed.

2.1 Controller Architecture

When choosing a servo motor, it is important to have a good understanding of the system to be controlled. The better that knowledge, the higher the chances of accomplishing a desired behaviour. However, the control algorithm is paramount to the improvement of transient response times and reduction of steady state errors and sensitivity to load parameters.

Servo control can generally be broken down into two classes of problems [3]: Command tracking and disturbance rejection. The first pertains to how well the actual motion follows the position, velocity, acceleration and torque commands. The second, can include anything from torque disturbance on the motor shaft to supply voltage variations.

PID (Proportional Integral and Derivative) 2.1.1 and PIV (Proportional Integral and Velocity) 2.1.2 loops are the most common controller strategies employed. Unlike Feed-Forward control, which can anticipate the internal commands for zero error following, disturbance rejection reacts to unknown disturbances and modelling errors. To improve the overall performance, both of these control types can be combined.

2.1.1 PID

The typical servo motion controller topology is depicted in Fig. 2.1. ***G(s)*** represents the DC motor driver transfer function.

The servo drive receives a signal command that represents a motor current. The following equation relates motor shaft torque $\boldsymbol{T}$ with motor armature current $\boldsymbol{I}$ by a torque constant factor $\boldsymbol{K_t}$ (2.1).

$$T = K_t I \tag{2.1}$$

The parameters for the motor are the rotor and load moment of inertia $\boldsymbol{J}$, viscous friction constant (dampening) $\boldsymbol{b}$ and torque constant factor $\boldsymbol{K_t}$. The actual position $\boldsymbol{\theta(s)}$ can be measured by an encoder coupled to the motor shaft. ***Td*** models the load disturbance while $\boldsymbol{\theta_r(s)}$ represents the desired position command.

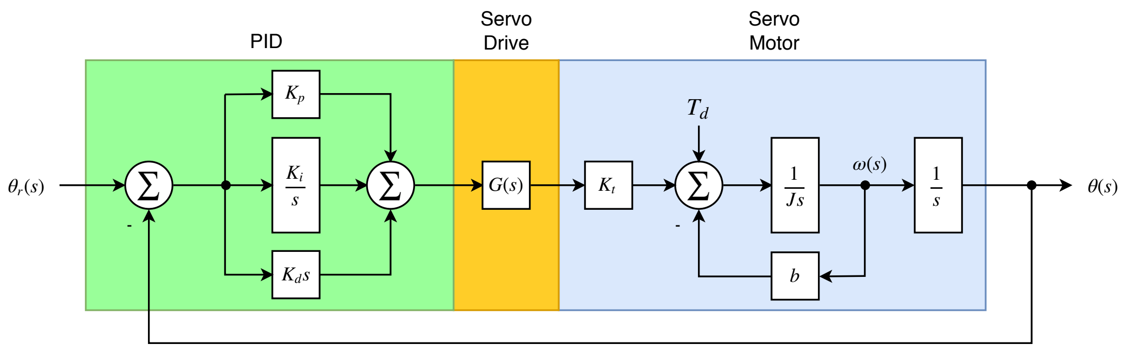


Figure 2.1: Basic PID Controller

The PID operates on the position error and outputs a torque command to the servo driver. Three gains need to be adjusted in the PID controller ($\boldsymbol{K_p}$, $\boldsymbol{K_i}$ and $\boldsymbol{K_d}$). They act on the error between desired and present position (2.2).

$$e(t) = \theta_r(s) - \theta(s) \tag{2.2}$$

The signal output from the PID is 2.3.

$$u(t) = K_p e(t) + K_i \int e(t)\delta t + K_d \frac{d}{dt} e(t) \tag{2.3}$$

There are two main ways of selecting the gains for the PID controller. Trial-and-error based on the operators own experience or an analytical approach. The first has the downside that there is no physical insight into what the gains mean or even if they are optimal. The second can be accomplished by, for example, following *Ziegler and Nichols* [4] [5] method based on setting $\boldsymbol{K_i}$ and $\boldsymbol{K_d}$ to zero, analysing the system's step response while slowly increasing $\boldsymbol{K_p}$ until the shaft position oscillates and recording both $\boldsymbol{K_p}$ and the oscillation frequency. That data can then be used to calculate the final values of $\boldsymbol{K_p}$, $\boldsymbol{K_i}$ and $\boldsymbol{K_d}$. This last approach doesn't usually yield many benefits to a high-performance system. The settling time can generally be further improved.

2.1.2 PIV

An easier to tune alternative to the PID that has a better to predict system response is the PIV topology (Fig. 2.2). This controller combines a position and velocity loop. A velocity correction command results from the position error multiplied by $\boldsymbol{K_p}$. $\boldsymbol{K_i}$ now operates directly on the velocity error as opposed to the position error in the PID. $\boldsymbol{K_d}$ is replaced by $\boldsymbol{K_v}$.

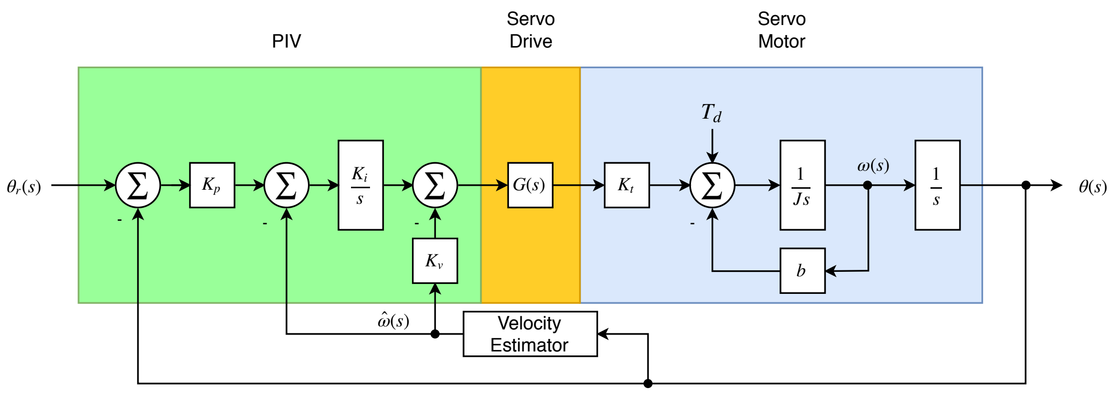


Figure 2.2: Basic PIV Controller

To accurately use this topology, a good knowledge of the motor velocity is required. To tune this system, two control parameters are needed: The bandwidth(BW) and damping ratio(ζ). An estimate of the rotor and load moment of inertia $\hat{J}$ and damping constant $\hat{b}$ are also required. Higher bandwidth is related to quicker rise and settling times while damping pertains mainly to overshoot. The K_p, K_i and K_v constants can be calculated from equations 2.4, 2.5 and 2.6.

$$K_p = \frac{2\pi BW}{2\zeta + 1}, \qquad (\mathrm{s}^{-1}) \tag{2.4}$$

$$K_i = (2\pi BW)^2(1+2\zeta)\hat{J}, \qquad (\mathrm{N\,m\,rad}^{-1}) \tag{2.5}$$

$$K_v = (2\pi BW)((1+2\zeta)\hat{J}) - \hat{b}, \qquad (\mathrm{N\,m\,rad}^{-1}\,\mathrm{s}) \tag{2.6}$$

2.1.3 Feed-Forward

To accomplish near zero following and tracking errors, Feed-Forward is often used in combination with a PID or PIV loop. For this topology, the controller needs access to a position $\theta_r(s)$, velocity $\omega_r(s)$ and acceleration $\alpha_r(s)$ commands (Fig. 2.3). Total inertia $\hat{J}$ and damping ration $\hat{b}$ estimates should also be available.

To calculate the estimated torque needed to make the desired motion, equation 2.7 is used.

$$U(s) = \hat{J}(\alpha_r(s)) + \hat{b}(\omega_r(s)) \tag{2.7}$$

Feed-Forward control reduces settling time and minimizes overshoot. In any case, to increase the velocity estimation accuracy, an encoder with enough resolution for the feedback is also required.

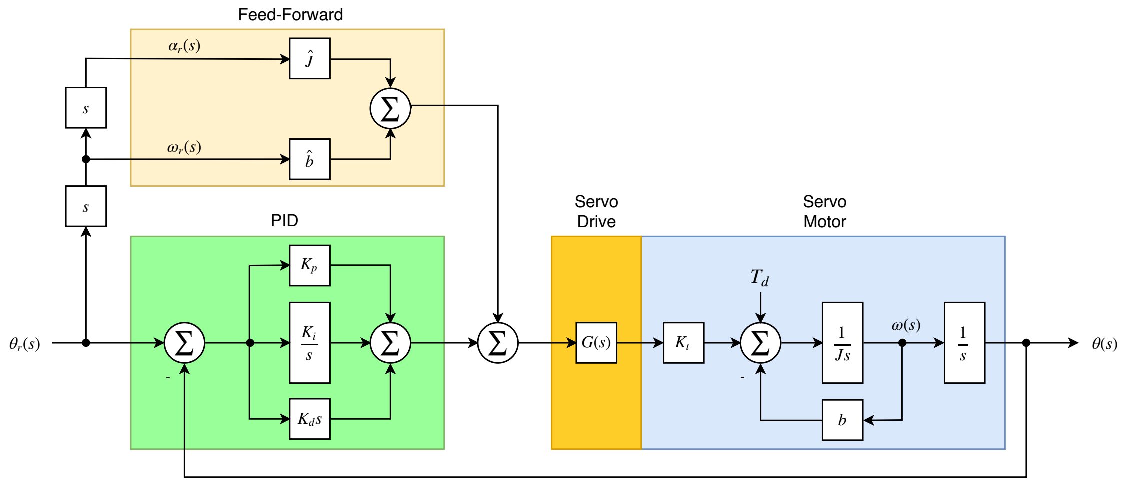


Figure 2.3: Basic Feed-Forward Controller

2.2 Parameter Estimation

RC servo motors are comprised of three main components: a DC motor, driving circuitry with potentiometer encoder for position feedback and gear reduction system. When modelling the controller, certain parameters from the DC motor need to be appropriately measured or estimated for an improved control [6]. They can be extracted from the transitory response to a step on the input and steady state response for different input voltages [7]. The model in Fig. 2.4 can be defined by equation 2.8, where $\boldsymbol{i_a}$ is the armature current and the voltage $\boldsymbol{V}$ is the sum of the voltage drop across the equivalent resistance $\boldsymbol{R}$, equivalent inductance $\boldsymbol{L}$ and back electromotive force $\boldsymbol{e}$.

$$V = Ri_a + L\frac{\partial i_a}{\partial t} + e \tag{2.8}$$

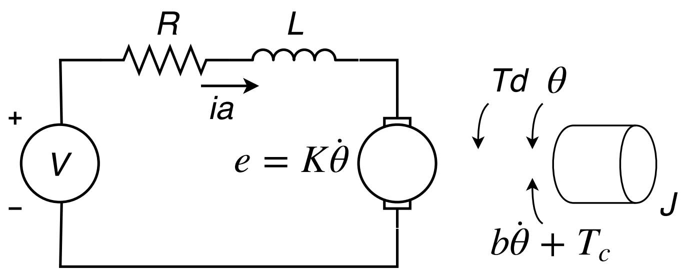


Figure 2.4: DC motor equivalent circuit

Equation 2.9 shows the resulting torque $\boldsymbol{T_r}$ as it relates to the developed torque $\boldsymbol{T_d}$, static friction torque $\boldsymbol{T_c}$ and viscous friction $b\dot{\theta}$.

$$T_r = T_d - b\dot{\theta} - T_c \tag{2.9}$$

The following equations express the relationship between developed torque $\boldsymbol{T_d}$ and current $\boldsymbol{i_a}$ (2.10), back electromotive force $\boldsymbol{e}$ and angular velocity ω (2.11) and load torque $\boldsymbol{T_L}$ with moment of inertia $\boldsymbol{J}$ and angular acceleration $\dot{\omega}$ (2.12).

$$T_d = Ki_a \tag{2.10}$$

$$e = K\omega \tag{2.11}$$

$$T_L = J\dot{\omega} \tag{2.12}$$

From this equations we can obtain the angular acceleration $\dot{\omega}$ (2.13) which can then be discretised into 2.14 where $\triangle T$ is the sampling time.

$$\dot{\omega} = \frac{Ki_a - T_c - b\omega}{J} \tag{2.13}$$

$$\omega[k] = \omega[k-1] + \triangle T\frac{Ki_a[k-1] - T_c - b\omega[k-1]}{J} \tag{2.14}$$

Equation 2.15 can then be attained by minimising the sum of the absolute error between the estimated (2.14) and actual transitory response, assuming zero inductive voltage drop across $\boldsymbol{L}$ and initial known values of $\boldsymbol{T_c}$ and $\boldsymbol{K}$.

$$J\dot{\omega} = \frac{K}{R}(V - K\omega) - b\omega - T_c \tag{2.15}$$

Solving 2.15 yeilds 2.16:

$$\omega(t) = \frac{c}{d}(1 - e^{-dt}) \tag{2.16}$$

where:

$$c = \frac{KV - RT_c}{RJ} \tag{2.17}$$

$$d = \frac{K^2 + Rb}{RJ} \tag{2.18}$$

In steady state $\omega = \frac{c}{d}$ resulting in:

$$\omega = \frac{K}{K^2 + Rb}V - \frac{RT_c}{K^2 + Rb} \tag{2.19}$$

By repeating the procedure starting from an estimated value for $\boldsymbol{R}$ to obtain $\boldsymbol{T_c}$ and $\boldsymbol{K}$ and replacing them with the new estimations to get $\boldsymbol{R}$ back, the parameters will converge to the real values.

2.3 Existing hardware

There are already several smart servo motors which include an array of sensors and communication protocols that attempt to shorten the prototyping phase by giving the designer tools to aid the process and increase development efficiency and speed.

2.3.1 Open Servo

First presented in 2005, Open Servo [8] was an attempt to develop an open community-based project with the goal of creating a drop-in replacement digital servo controller for robotics. In it's 3$^{\text{rd}}$ revision (Fig. 2.5), the circuit boasts features such as speed and temperature sensing, independent H-bridge allowing for breaking, ATMega168 at 20MHz, 400khz I^{2}C/TWI interface, 6V to 7.5V voltage input and 3A continuous current output. But the angle of rotation is still bound by the limits of the mechanical potentiometer. It seems that for the last few years it's development has abated.

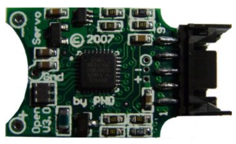


Figure 2.5: Open servo V3 circuit

2.3.2 SuperModified

The SuperModified v3.0 [9] (Fig. 2.6) circuit offers a complete DC motor controller. Like the Open Servo, it's designed to fit inside of a regular RC servo motor. The circuit includes a 15-bit absolute position encoder, continuous 360° range, position and velocity profiled movements, multiple bus interfaces such as RS-485, UART and I^{2}C and 5V to 24V voltage input at 5A continuous current output.

However, the hardware and firmware are proprietary, thus preventing the development community from improving it's functionality. Depending on the application, it's cost of 55 € might prove uneconomical considering that a servo to apply it to also needs to be purchased.

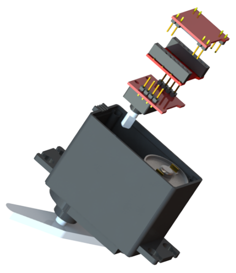


Figure 2.6: SuperModified v3.0 circuit

2.3.3 Moti

Appearing for the first time at the end of 2014, Moti [10] (Fig. 2.7) promised to be a *"smart servo motor that makes it easier to build robots"*. Like Open Servo and SuperModified, Moti aims to be a drop-in replacement to the typical controller. The major difference being that the developer wanted it to *"play well with the web and mobile devices, so there's a RESTful API for developing apps"*.

Unfortunately Moti is still not available to the public after an unsuccessful Kickstarter campaign and it doesn't seem like the hardware and firmware will be open-source.

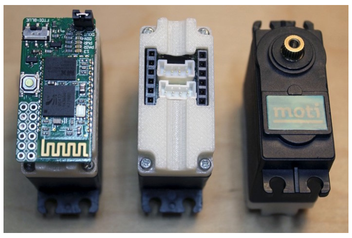


Figure 2.7: Moti servomotor

2.3.4 AX-12A

The Robotis AX-12A [11] (Fig. 2.8) *"is the most advanced actuator on the market in this price range and has become the defacto standard for the next generation of hobby robotics"* [2]. By including a half-duplex asynchronous serial communication interface with 1Mbit baud rate, this servo has the ability to track its position, speed, temperature, voltage and load while allowing for the strength and speed of the motor's response to be controlled. It's enclosure is designed so that various robot forms can be accomplished through the use of specially designed frames that allow multiple devices to be mechanically linked together.

Nevertheless, the limitations of a mechanical position sensor are still present. Despite having a larger position sense range when compared to most RC servo motors, there is yet a 60° deadzone where the position is unknown. Like the SuperModified, the hardware and firmware are proprietary.

Figure 2.8: Robotis AX-12A smart servo motor [2].

2.4 Conclusion

Disturbance rejection can be obtained from PID and PIV controllers. Because the overshoot and rise times are tightly coupled, a change in any of the gains greatly influences the overall system. PIV control provides for a decoupled management of overshoot and rise times, while allowing easier tuning and high disturbance rejection qualities. Feed-forward helps to mitigate tracking errors.

Parameter estimation can be accomplished by extracting the transitory response to a step on the input and steady state response for different input voltages. By starting from an initial set of estimations, $\boldsymbol{R}$, $\boldsymbol{T_c}$ and $\boldsymbol{K}$ can be further iterated to converge to their true values.

Chapter 3

Implementation

RC servo motors allow for position control. However, they move from one position to another at maximum speed which can cause abrupt movements and current spikes in the power supply. By using potentiometers as position sensors, they offer a narrow sense range and their resolution is limited by the ADC responsible for reading their voltage output. For that reason, most can only position the motor shaft between 0° and 180°. Fig. 3.1 shows a simplified diagram of a typical servo motor.

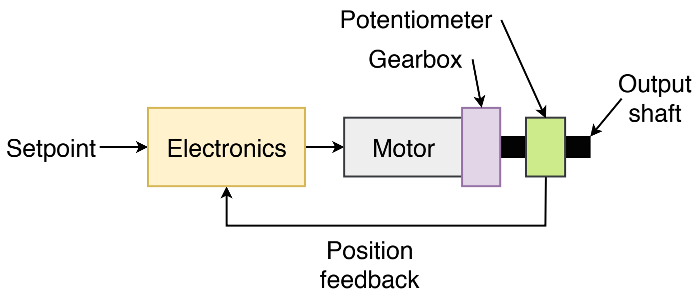


Figure 3.1: Simplified diagram of a servo motor

Certain modifications can be performed, such as removing the limit pin from the output shaft. Nonetheless, the movement is still constrained by the mechanical range of the potentiometer itself. Furthermore, there is no feedback on the present position so the user can only guess where the shaft actually is.

This chapter will describe the design process for SMORA from a hardware and software perspective.

3.1 Hardware

SMORA is an open-source attempt at sidestepping the shortcomings of previous devices while adding more features at a similar price range. By replacing the mechanical potentiometer with a hall-effect magnetic encoder, SMORA allows for high resolution position and speed feedback while having true 360° rotation capabilities. Additionally, it includes current, voltage, temperature and attitude sensors. Not only can a user retrieve their data through the included half-duplex serial interface but also use them to build new algorithms and features that the other devices are not capable of, such as retrieving the present yaw, pitch and roll of the device itself. By taking advantage of the original MG996R metal gears, SMORA is more rigid and endures higher loads without breaking.

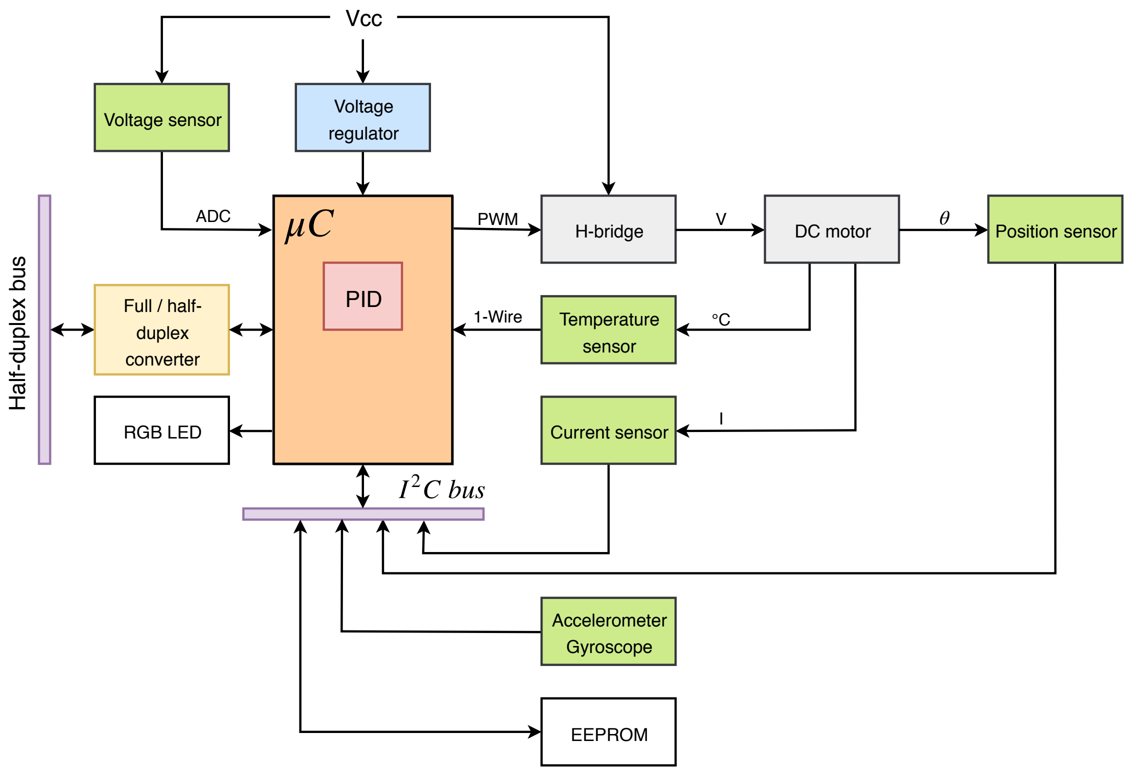
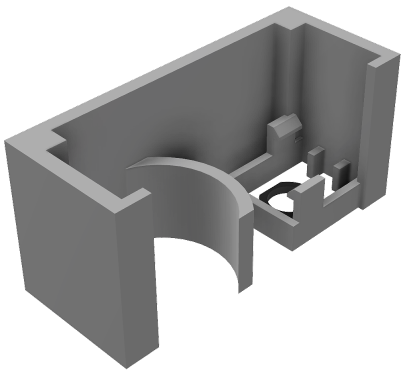


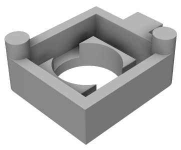
Figure 3.2: SMORAs diagram

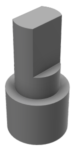
Fig. 3.2 shows the diagram of the architecture chosen to base the development of the circuit on.

3.1.1 MG996R Reverse engineering

The development of the SMORA prototype started with reverse engineering a MG996R servo motor. The inner dimensions were measured with a digital caliper and used to build a simplified 3D model of the inside (Fig. 3.3a). As the selected position sensor needs a diametrically magnetised neodymium magnet, to hold it in place and keep it centered while the shaft rotates, two additional models were created (Fig. 3.3b and Fig. 3.3c). With the dimensions constrained, a DXF file with the outline for the magnetic encoder PCB was exported.


(a) Section cut


(b) Magnet holder


(c) Shaft

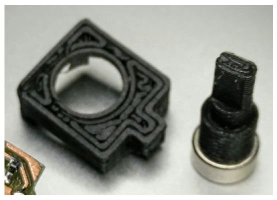
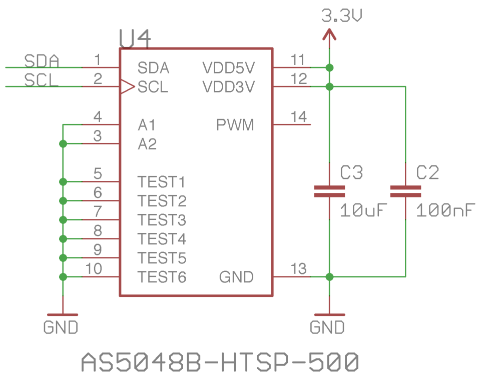

(d) 3D printed

Figure 3.3: 3D rendering of the inside of a MG996R servo enclosure, magnet holder and shaft.

3.1.2 Magnetic Encoder

Unlike most of the RC servo motors commonly available, to measure the shaft position SMORA uses a AMS AS5048B [12] hall-effect magnetic encoder instead of a potentiometer. It works by measuring the magnetic field of a diametrically magnetised neodymium magnet (Fig. 3.4b) and calculating it's angle of rotation. With 14-bit resolution provided through I^2C, this sensor is capable of outputting 16384 positions per revolution while allowing for a full 360° rotation (Fig. 3.4a). To that end, the limit pin from gearbox output gear had to be removed (Fig. A.9).



(a) Schematic.

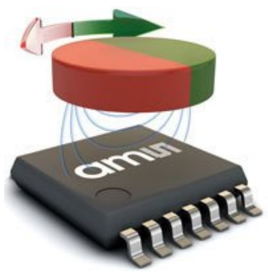

(b) Magnet diagram.

Figure 3.4: AS5048B.

3.1.3 EEPROM

To allow for some flexibility, and considering that some microcontrollers don't have an internal EEPROM, SMORA includes an external Microchip 24LC16BT 2Kbyte EEPROM [13] (Fig. 3.5). This memory can be used to store, for example, configuration parameters such as PID gains and position, speed and alarm limits.

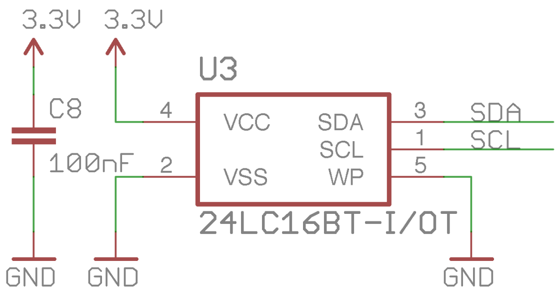
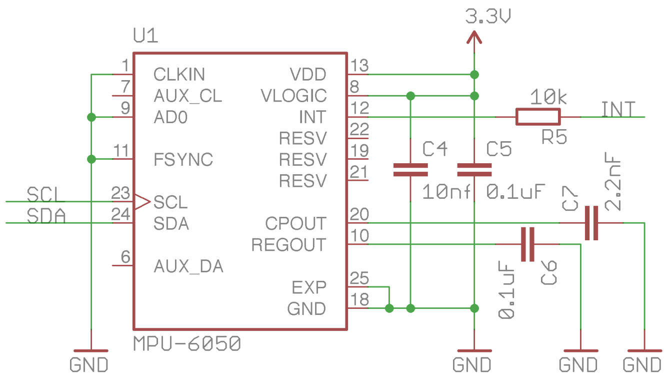


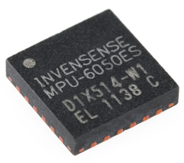
Figure 3.5: 24LC16BT circuit schematic

3.1.4 MPU-6050 accelerometer and gyroscope

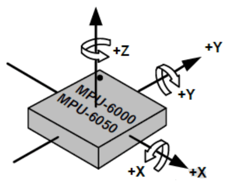
SMORA is the only servo that allows the user to get feedback on the devices attitude. A InvenSense MPU-6050 [14] (Fig. 3.6a and 3.6b) containing a 16-bit MEMS accelerometer and gyro can give feedback on the pitch, roll and yaw (Fig. 3.6c), either by supplying the raw data or by using the integrated DMP (Digital Motion Processor) to calculate it.



(a) Schematic.



(b) QFN IC.



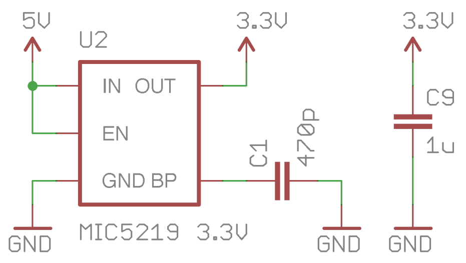
(c) Axes.

Figure 3.6: MPU-6050.

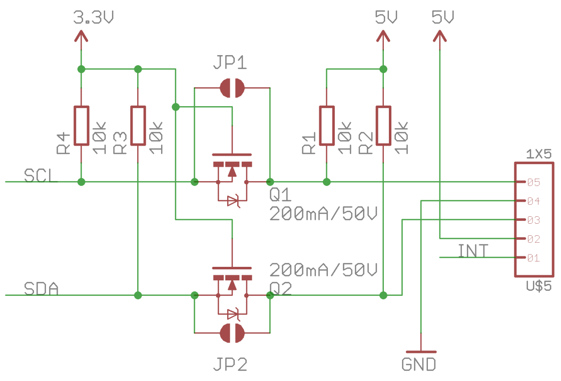
3.1.5 Magnetic encoder: voltage regulation and translation

To allow the magnetic encoder PCB to operate at either 3.3V or 5V supplies and interface logic levels, a MIC5219-3.3 LDO voltage regulator [15] (Fig. 3.7a) was added, as well as two BSS138 N-channel MOSFETs [16] acting as voltage translators (Fig. 3.7b) for the SDA and SCL lines.

This level shifting is necessary if the host and master operate at different voltages, lest the device with the lower supply has a voltage on it's lines higher than it tolerates, potentially damaging it.



(a) 3.3V voltage regulator schematic



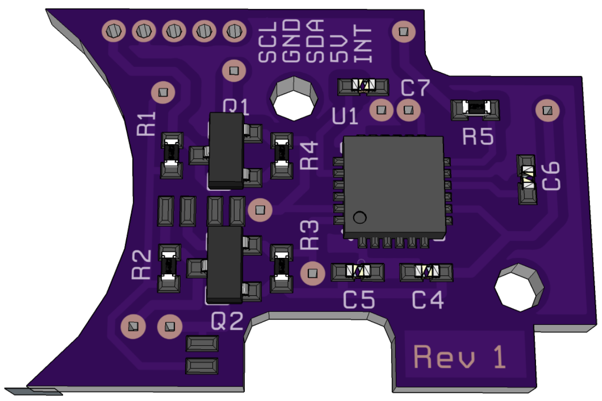
(b) Level translator schematic

Figure 3.7: Position encoder supply and level translator.

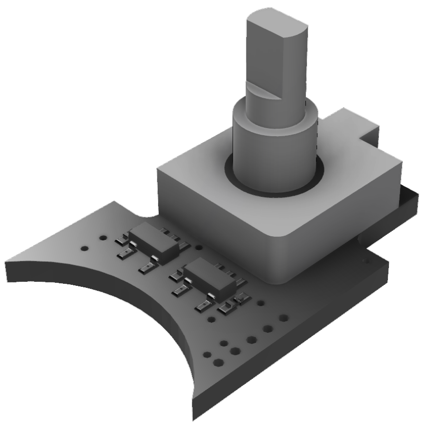
Jumpers J1, J2 and J3 should be shorted to configure it to operate at 3.3V. In that case, the BSS138 MOSFETs and MIC5219-3.3 LDO voltage regulator can be omitted.

3.1.6 Position encoder PCB

A 3D model of the position encoder PCB was created to test the fitting against the initial 3D model before being fabricated (Fig. 3.8a and Fig. 3.8b).



(a) Top.



(b) Bottom with holder and shaft.

Figure 3.8: 3D rendering of the position encoder PCB.

Fig. 3.9 shows a picture of the position encoder PCB with the MPU-6050 accelerometer/gyro populated next to a 5 cent coin for size comparison.

The full schematic for this PCB can be found in Fig. A.1.

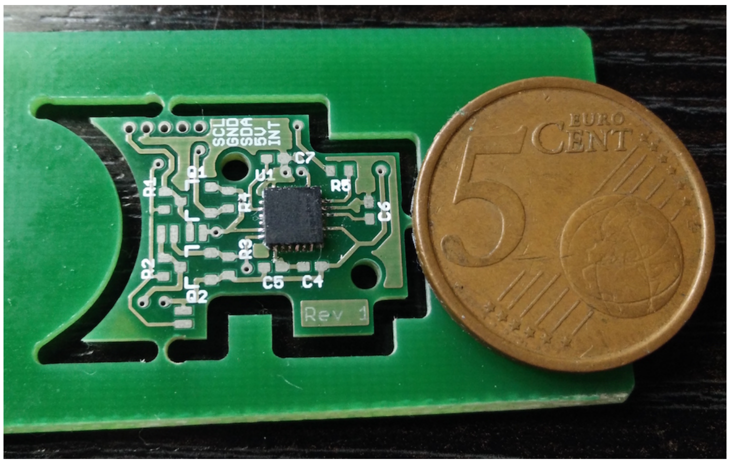
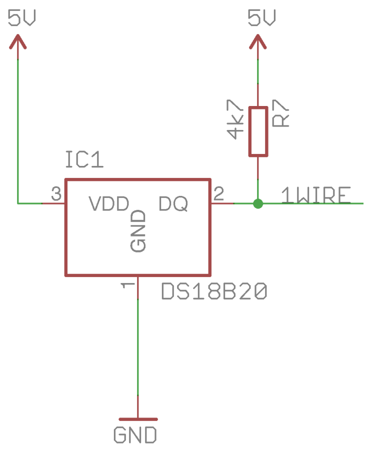


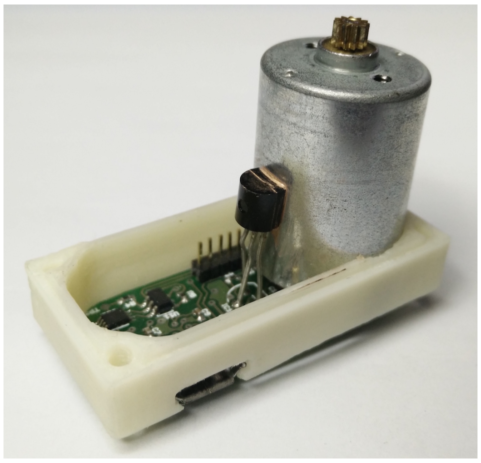
Figure 3.9: Position encoder PCB prototype.

3.1.7 Temperature sensor

In order to preserve the proper operation of the DC motor, the temperature needs to be monitored and kept below a certain level. To that end, a Maxim DS18B20 [17] (Fig. 3.10a) 9-bit to 12-bit resolution digital thermometer is adjoined to it by thermal paste (Fig. 3.10b) and communicates over a 1-Wire bus, thus requiring only one data line. It is powered from the same source as the microcontroller although the sensor itself can derive power directly from this data line ("parasitic power"). Providing an alarm function with non-volatile user-programmable upper and lower points allows for the microcontroller to offload the over-temperature triggering.



(a) Schematic.



(b) Adjoined to the motor.

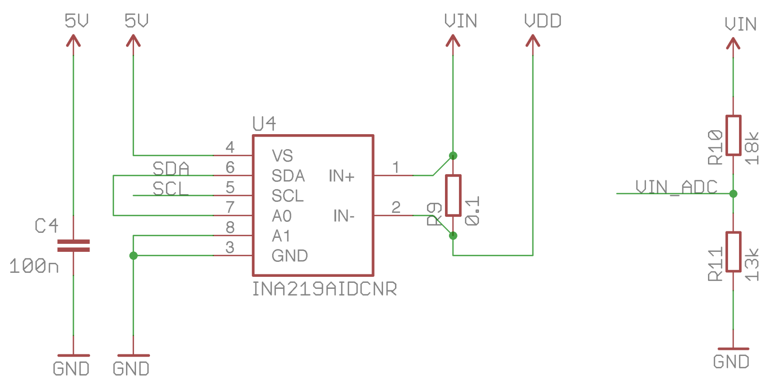
Figure 3.10: Temperature sensor.

3.1.8 Current and voltage sensor

Having the ability to measure the motor current allows for it's different parameters and torque to be estimated. Torque limitation and control can also be accomplished. A Texas Instruments

INA219A [18] (Fig. 3.11a) is used to monitor the motor's current through a 0.1 ohm shunt resistor. This sensor reports shunt voltage drop, bus supply voltage and power with programmable conversion times and filtering. It enables direct current readouts in milli-Amperes and voltage in Volts through the I^2C interface.

Additionally, the power supply voltage can also be monitored through a resistive voltage divider connected to an ADC pin (Fig. 3.11b).



(a) Current (b) Voltage

Figure 3.11: Current and voltage sensor schematics.

3.1.9 Half-duplex interface

SMORA connects to the outside world through a half-duplex serial interface. As opposed to full-duplex, it is devised to only transmit or receive at a time. However, it only requires one line instead of two.

The same cable is used for communication and power. The Molex PicoLock 4 pin connectors allow for up to 3A of supply current. Multiple SMORA's can be daisy-chained together (Fig. 3.12) and communicate at speeds of over 1Mbit/s.

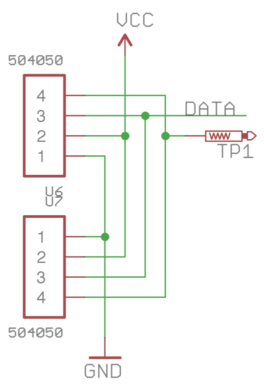
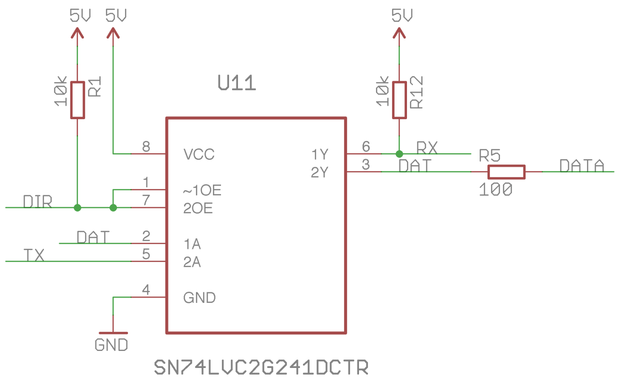
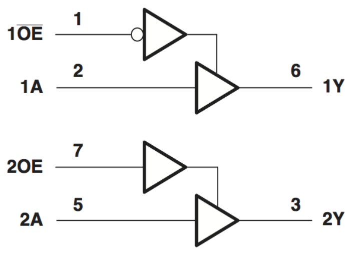


Figure 3.12: Half-duplex connectors schematic.

A Texas Instruments SN74LVC2G241 [19] (Fig. 3.13) dual-buffer and driver with tri-state outputs is responsible for integrating the half-duplex interface and the full-duplex serial interface of the microcontroller.



(a) Schematic.



(b) Diagram.

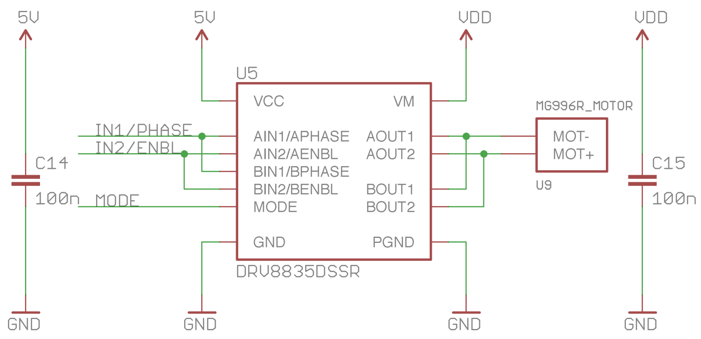
Figure 3.13: Half-duplex circuit.

By controlling the output-enable inputs of the buffers, the microcontroller can regulate which pin of the full-duplex serial interface (receive or transmit) should be connected to the half-duplex data pin. Setting the DIR pin high will enable transmission while setting it low enables reception. A 100 ohm resistor in series with the data line is used to limit the buffers current, lest there is a collision if multiple devices try to transmit at the same time.

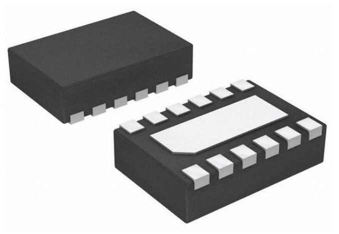
When the buffers are disabled, their output are in high impedance. To assure that the RX line stays at a logic high level when this happens, a 10K resistor pull-up was added. Similarly, a pull-up resistor in the DIR line guarantees that it stays at a known logic level while the microcontrollers' pins are being configured during startup.

3.1.10 Motor driver

The Texas Instruments DRV8835DSSR [20] (Fig. 3.14a) is an integrated dual low-voltage H-bridge capable of up to 3A drive current (when both H-bridges are connected in parallel).



(a) Schematic.



(b) WSON-12 IC packaging.

Figure 3.14: Motor driver.

It operates on a motor supply from 0V to 11V and a device supply from 2V to 7V. The microcontroller uses this driver to control the motor's torque using a PWM signal and a direction signal. This driver was mainly chosen for it's ultra-compact WSON form-factor (Fig. 3.14b) and for having a current supply capability above what the motor requires. This current was measured to be below 2A with the DC motor shaft locked.

3.1.11 RGB Led

To show the user if there is an alarm situation, SMORA employs a common-cathode RGB LED (Fig. 3.15). This LED is also used for the bootloader status.

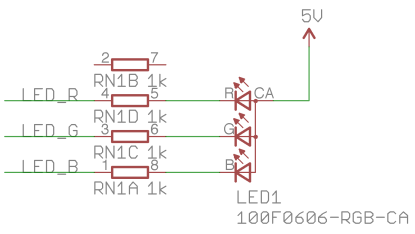
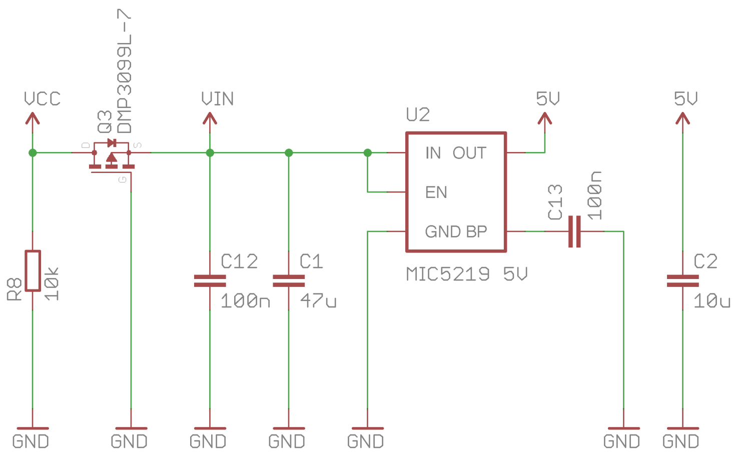


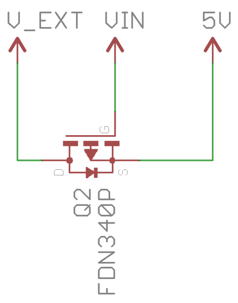
Figure 3.15: RGB LED schematic.

3.1.12 Power supply regulation and protection

SMORA can be powered from 5V to 11V (versus 9V to 12V of the AX-12A) through the half-duplex interface cable. A MIC5219-5V [15] LDO regulator is employed to regulate this voltage down to 5V while a high-side DMP3099L-7 [21] P-channel MOSFET protects the circuit from reverse polarity [22] (Fig. 3.16a). A 47uF ceramic capacitor ensures that the supply stays as noise-free as possible.



(a) Regulation and reverse-polarity protection.



(b) Voltage selector.

Figure 3.16: Power supply schematic.

Additionally, due to SMORA being programmable through an external interface (USB or serial) that can also power it, a FDN340P P-channel MOSFET (Fig. 3.16b) ensures that only one power supply is connected at a time. Priority is given to the power received from the half-duplex interface cable.

3.1.13 SMORA Variants

SMORA comes in two variants: SMORA-A8 and SMORA-A32 (Fig. 3.17).

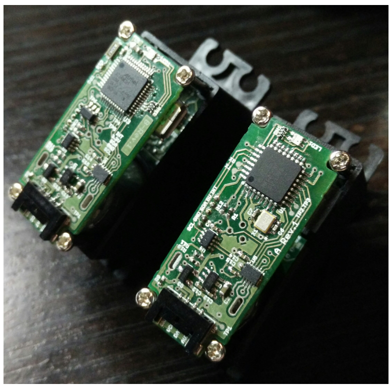


Figure 3.17: Actual picture of the 2nd prototypes of SMORA-A32 to the left and SMORA-A8 to the right.

SMORA-A8 (full schematic in Fig. A.2) includes a 8-bit ATmega328p with 32KB of flash memory and 2KB of SRAM running at 5V and 16MHz. It was chosen in this variant for being hugely popular amongst the developer and maker community. The amount of open-source project created with this microcontroller translates into a wealth of information when it comes to available source-code and hardware designs. Thus, allowing for faster firmware development.

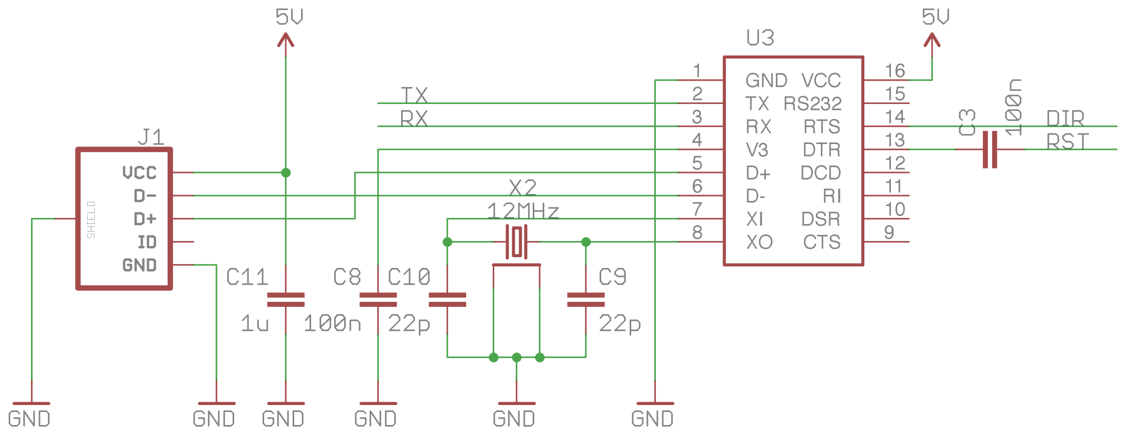


Figure 3.18: USB-to-Serial interface schematic.

To program the SMORA-A8 variant, a USB-to-Serial interface was created based on the CH340G integrated circuit (Fig. 3.18). A SN74LVC2G241 [19] was also included that allows the same board to both program and interface with the half-duplex bus with the communication direction managed by a computer through the CH340G RTS pin (Fig. 3.19).

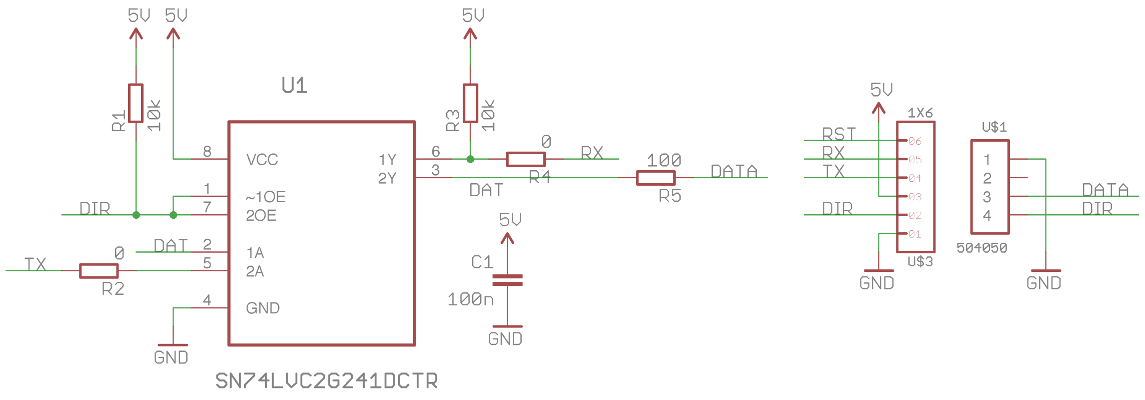
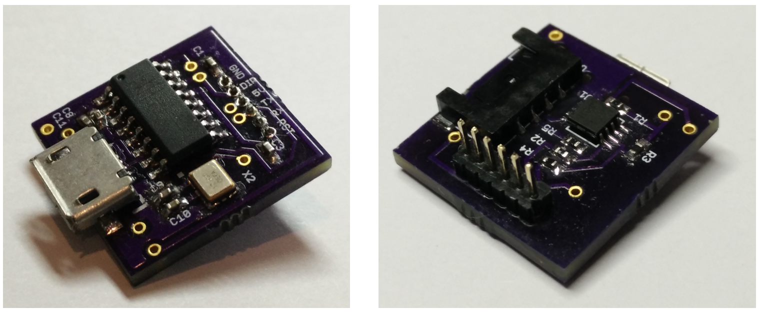


Figure 3.19: Half-duplex programmer's schematic.

The final programmer prototype is shown in Fig. 3.20 and it's full schematic in Fig. A.4.



(a) Top. (b) Bottom.

Figure 3.20: SMORA-A8 programmer and half-duplex interface prototype.

SMORA-A32 (Fig. 3.21) includes a 32-bit ATSAMD21G18A ARM Cortex-M0+ with 256KB of flash memory and 32KB of SRAM running at 3.3V and 48MHz with a 32.768KHz crystal. Having 8 times more flash, 16 times more RAM and thrice the speed, allows this microcontroller to include more complex algorithms than it's ATmega328p counterpart.

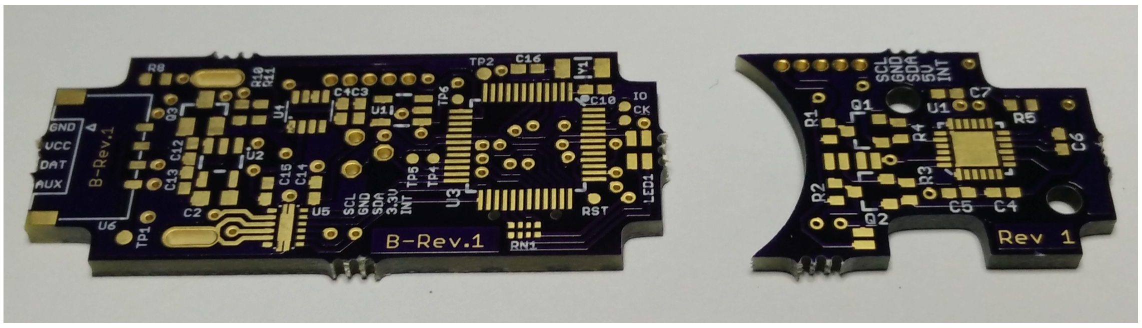


Figure 3.21: Actual picture of SMORA-A32 final PCBs manufactured by OSHPark.

The SMORA-A32 variant can be programmed directly through the included USB interface. When plugged into a computer, it presents a CDC serial interface capable of speeds of over 2.5Mbit/s that can also be used for debugging and to allow it to act as a master in the half-duplex bus network. A resettable fuse protects the circuit from currents over 500mA through the USB port.

A MIC5219-3.3V [15] LDO regulator is employed to regulate the voltage down to 3.3V. To ensure that the half-duplex bus always runs at 5V, a Texas Instruments TXS0101 [23] (Fig. 3.22) 1-bit bidirectional voltage level translator is included.

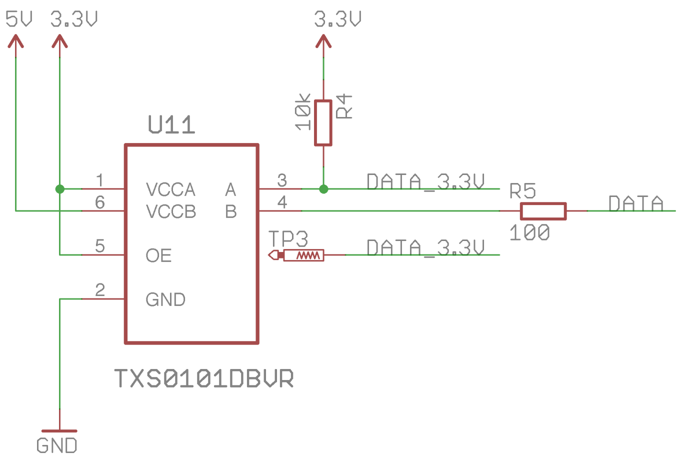
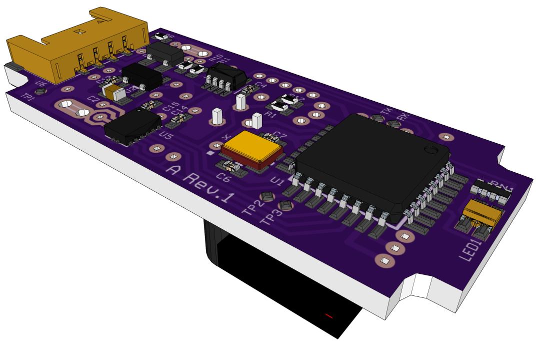


Figure 3.22: TXS0101 schematic.

3.1.14 External cover 3D model

Because SMORA's custom PCB is larger than the one used in a MG996R servo, in order for the device to be self contained a custom 3D cover model was designed. To ensure the PCBs would fit, they were first modelled in 3D (Fig. 3.23a and Fig. 3.23b) so that a cover could be created around them (Fig. 3.24).


(a) SMORA-A8.

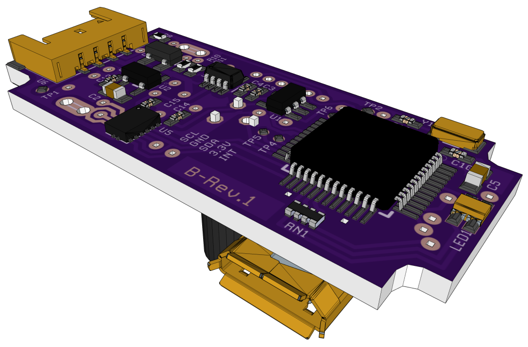

(b) SMORA-A32.

Figure 3.23: 3D rendering of the SMORA PCB variants.

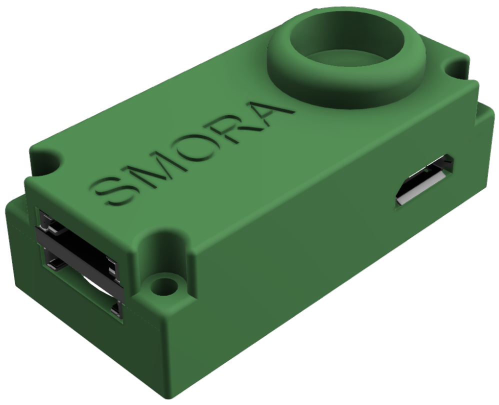


Figure 3.24: 3D rendering of SMORA A32 cover

A ball-bearing insert was also added in line with the shaft to add stability and rigidity when used in the construction of other robotic devices. Fig. 3.25 shows a picture of the elbow of a custom built SCARA robotic arm fitted with a SMORA-A32.

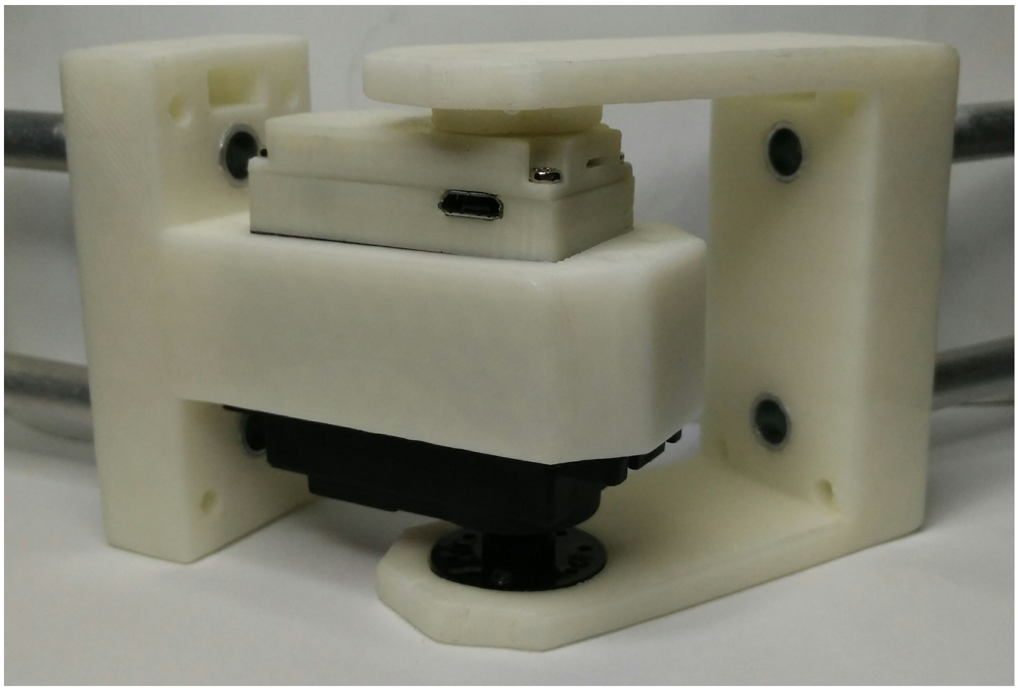
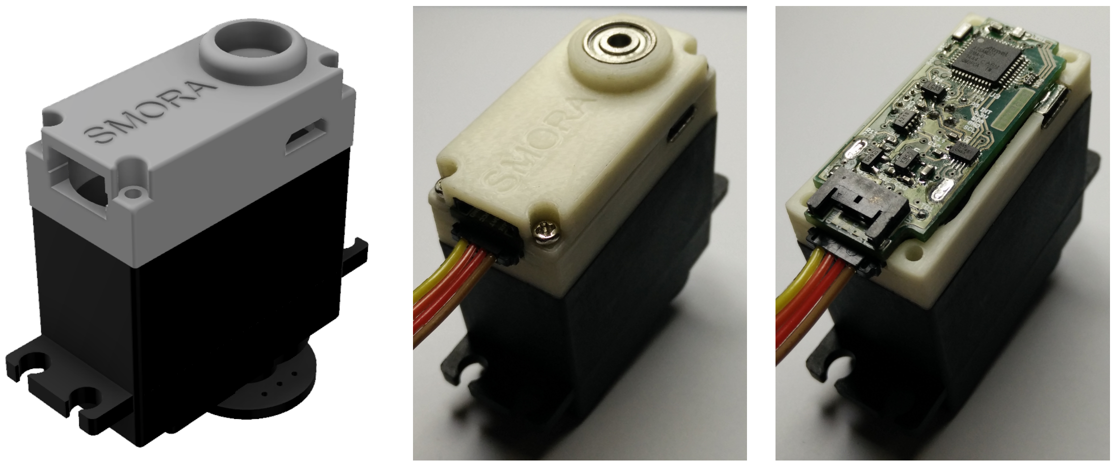


Figure 3.25: SCARA elbow.

3.1.15 SMORA prototype

With the 3D model components tested for fitting, the STL files were exported and realised in a 3D printer. Fig. 3.26a, 3.26b and 3.26c show a 3D rendering of the final model as well as pictures of the actual SMORA-A32 variant prototype.



(a) 3D model. (b) With cover. (c) Without cover.

Figure 3.26: Pictures of the final SMORA-A32 prototype.

3.1.16 BOM list

SMORA-A8 (Table A.2 and Table A.3) and SMORA-A32 (Table A.4 and Table A.5) have a bill of materials of under 34 € in single quantities excluding PCBs and 3D printer material. Most components can be obtained through most online retailers. By procuring them directly from the manufacturer, the cost can be substantially reduced. For the development of the prototypes in this thesis, the components were either directly acquired through their manufacturer's sample program or sourced from over-seas resellers.

3.2 Software

Both SMORA-A8 and SMORA-A32 have a common firmware base compatible with the Arduino environment. This grants the open-source community an easy way to develop their own algorithms and communication protocols.

3.2.1 Bootloader

SMORA-A8 has a standard OptiBoot bootloader with a baud rate of 115200, common in most Arduino development boards. SMORA-A32 has a modified SAM-BA bootloader to use the LED as a status indicator. Both can be reprogrammed through test pads exposed in the main PCB (Fig. 3.27).

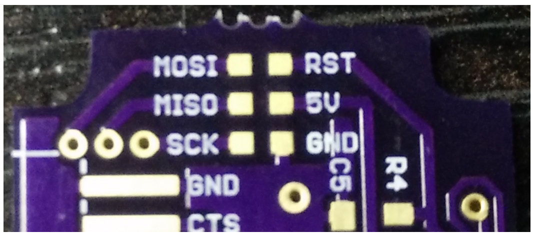


Figure 3.27: SMORA-A8 ICSP.

To program the ATmega328p bootloader, a ICSP programmer is required. Open-source hardware designs like the USBasp are compatible. The ATSAMD21G18A however, needs to be programmed through SWD (Serial Wire Debug). This can be mitigated by using, for example, a Raspberry PI Zero running the OpenOCD software.

3.2.2 Communication Protocol

To simplify the communication through the half-duplex serial interface, a protocol similar to the one incorporated in the AX-12A servo motor was written.

Communication is based on frames composed of a sequence of bytes. Each frame starts with a 5 bytes header, followed by a payload of arbitrary size and a one byte checksum. The header always starts with two OxFF bytes that allow the receiver to pinpoint the start of a new frame. Next come the identification byte, frame length byte and instruction byte. Every message ends with a checksum calculated by summing all the bytes except the first 2, taking the least significant byte and performing a NOT operation (Eq. 3.1). The first byte of the payload indicates the register to where the rest of the payload is to be written.

$$\sim \left(\sum_{n=2}^{length-1} byte[n] \right) \&\texttt{0xFF} \tag{3.1}$$

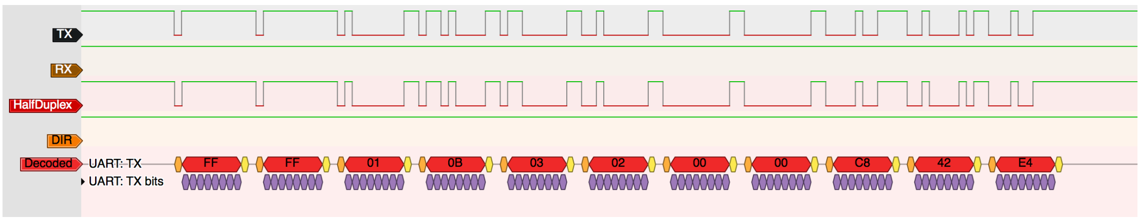


Figure 3.28: Half-duplex transmitted frame at 1Mbit/s.

Fig. 3.28 displays the signal in the RX, TX, HalfDuplex and DIR lines, as well as the decoded bytes. The computer sets the DIR line high through the USB-to-Serial RTS pin to configure the half-duplex interface to transmission mode. The same bytes can be seen also in the TX line. The first 5 bytes contain the frame start bytes (`0xFFFF`), followed by ID (`0x01`), length (`0x0B`, `11` in decimal) and instruction (`0x03`, WRITE_REGISTER). The payloads first byte (`0x02`) tells the microcontroller to write 4 bytes (`0x0000C842`) to the GOAL_POSITION register, which is decoded as a float with the value `100,0` in little endian. The last byte (`0xE4`) is the checksum obtained by Eq. 3.2.

$$checksum = \sim(\texttt{0x01}+\texttt{0x0B}+\texttt{0x03}+\texttt{0x02}+\texttt{0x00}+\texttt{0x00}+\texttt{0xC8}+\texttt{0x42})\ \&\ \texttt{0xFF} \quad (3.2)$$

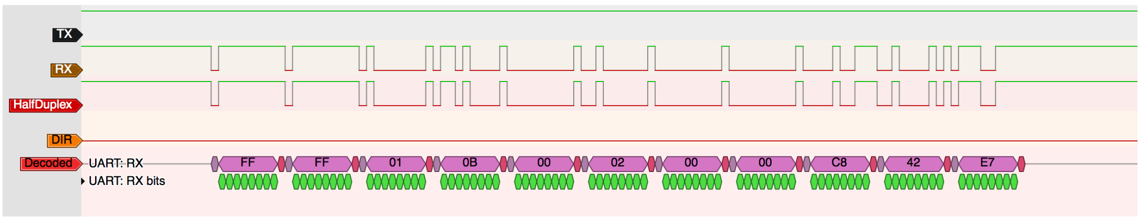
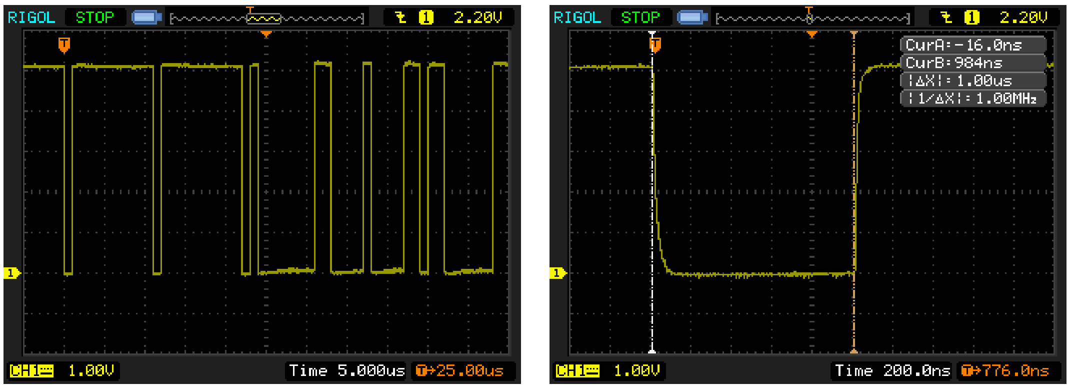


Figure 3.29: Half-duplex received status frame at 1Mbit/s.

Likewise, the optional status frame in Fig. 3.29 shows the received bytes in the HalfDuplex and RX lines with the DIR line set low. The 5th byte (`0x00`) representing the instruction byte in the transmitted frame is replace with a status byte. This byte allows to diagnose failures such as checksum errors and communication faults or alarm situation such as under-voltage or over-heating. The value `0x00` indicates that there was no error. This frame is configured to echo the previously received payload so that a corruption in the transmission can be spotted.

Unlike the AX-12A, this protocol is not limited to 1 byte size registers. Having a structure that is aware of the size of each register allows the protocol to be expanded into writing to all registers forthwith from a single frame while still being able to decode their size properly.

Fig. 3.30 shows an oscilloscope capture of the beginning of a half-duplex frame at 1Mbit/s.



(a) Beginning of a frame. (b) Shortest bit.

Figure 3.30: Oscilloscope view of the half-duplex communication at 1Mbit/s.

3.2.3 Host side software

To aid in debugging, data logging and graph plotting, a host side script and library were written in Python. This script abstracts the communication layer by exposing simple commands that take care of the complexities of encoding and decoding a frame and selecting the direction of the half-duplex serial interface. The following script is sufficient to configure the position PID gains and frequency and set the desired position.

```python
from smora.smora import *
s = Smora(port='/dev/cu.wchusbserial1420', baudrate=1000000, debug=1)
s.open()
status = s.setPIDParams(id=0x01, pid=0, Kp=0.38, Ki=0.0055, Kd=0.007, Kf=0.0,
  ↪  frequency=50)
status = s.write_data(id=0x01, register.GOAL_POSITION, 100.0)
s.action(id=0x01) # commit changes
s.close()
```

In order to plot the variables of interest in each test, the script uses a thread to gather the data transmitted by SMORA while the commands are being issued. When the test is complete, this data is logged to a file and plotted using the Python library *matplotlib*. Other mathematical functions like the Butterworth filter employed to calculate the compensator array rely on the *numpy* library.

3.2.4 Arduino firmware

On the firmware side, a C++ class is instantiated. This class is responsible for initialising all the microcontroller pins, sensors and interfaces and retrieving the configuration parameters (shown in the following structure) from the external EEPROM. It includes parameters such as the half-duplex identification and baud rate, PID gain parameters and the registers that can be read and written.

```c
typedef struct STORAGE {
  unsigned char model = 0x02;
  unsigned char id = 0x01;
  unsigned long baudrate = 1000000;
  unsigned long version = CONFIG_VERSION;
  PID position;
  PID speed;
  CONTROL_T control;
} STORAGE;
```

The hot loop responsible for the core functionality of SMORA runs in the Arduino loop function. By always being aware of the current time in microseconds, this loop matches the frequency to which the PID was configured, thus reducing the jitter (only minimally disturbed by interrupt routines) and making sure that values like current, voltage, temperature and present position are always retrieved periodically. It is also responsible for calculating the speed, switching between the different operation modes (position and speed control), computing the PID output, actuate on the DC motor by setting the PWM value, transmitting debugging data (if requested) and taking care of the communication.

The time required for retrieving the data from the position encoder and current, voltage and temperature sensors can be seen in Table 3.1.

Table 3.1: Sensor timing

Position	Current	Voltage	Temperature
$180\mu s$	$188\mu s$	$188\mu s$	$760ms$

As stated by the DS18B20 temperature sensor datasheet [17], the average time it takes for a temperature conversion is $750ms$ with 12-bit resolution. This of course is unacceptable if the loop function is to run at frequencies of $50Hz$ and above even if the temperature is only fetched once a second. To solve this issue, a non-blocking function was created that separates the *start_conversion* and *request_temperature* commands. The maximum loop time was reduced to about $6ms$ which allows for a PID frequency of above $150Hz$.

For the communication, a state-machine function was created that is responsible for gathering the bytes sent through the half-duplex interface until their length matches the frame size, comparing the instruction to a list of know instructions and validating the checksum. If the frame is accepted, the ID field is verified and the payload decoded and applied to the appropriate registers. In the event that a status frame is required, a payload with the reply is encoded and the bytes transmitted. Once the transmission is complete, the half-duplex direction is switched to receiving mode. To prevent a deadlock in the loop, if there is a fault in the communication a timeout mechanism discards the frame and restarts the state-machine.

3.3 Algorithms

To control the motor, different algorithms were employed. They are responsible for calculating the appropriate PWM values so that SMORA can exhibit a certain desired behaviour such as setting a position or speed.

3.3.1 PID equation

The equation in 2.3 was discretised into 3.3.

$$u(k) = K_p e(k) + K_i \sum_{0}^{k} \frac{e(k)}{F} + K_d(e(k) - e(k-1))F \tag{3.3}$$

where:

$$F = PID\ frequency, \tag{3.4}$$

$$e = r(k) - y(k) \tag{3.5}$$

To prevent over-shooting from the integrator windup, an anti-windup mechanism is deployed that limits the error accumulation to a set maximum. A feed-forward loop is then combined with the PID output (3.6).

$$u(k) = u(k) + K_f r(k) \tag{3.6}$$

The output is then scaled considering the bus voltage, constrained to the PWM allowed range and applied to it. The following code shows the compute function used to calculate the PID output.

```
float PID::compute(float ref, float error){
  state.Reference = ref;
  state.previous_error = state.error;
  state.error = error;
  state.previous_integrator = state.integrator;
  state.integrator += state.error / pid.frequency;

  // Anti-windup
  if (state.integrator * pid.Ki > pid.limit_max){
    state.integrator = state.previous_integrator;
  } else if (state.integrator * pid.Ki < pid.limit_min){
    state.integrator = state.previous_integrator;
  }

  // PID Output
  state.Output  = pid.Kp * state.error;
  state.Output += pid.Ki * state.integrator;
  state.Output += pid.Kd * (state.error - state.previous_error) * pid.frequency;
   ↪  // same as /dt
```

```

    // Feed-forward Output
    state.Output += pid.Kf * state.Reference;

    return state.Output;
}
```

3.3.2 Position PID

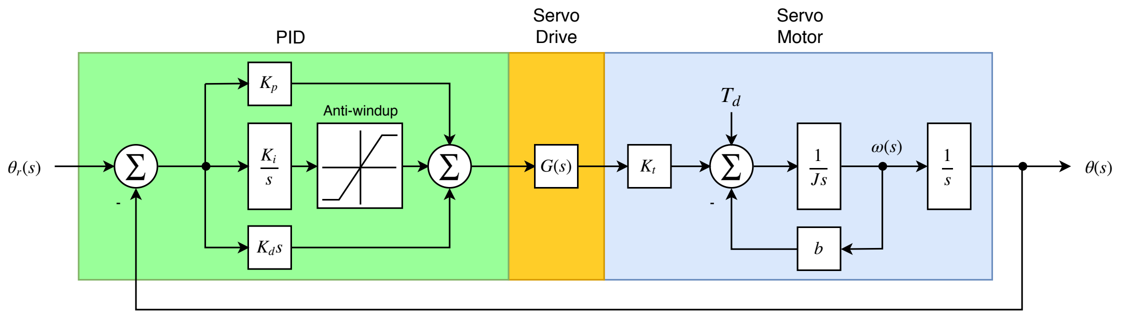


Figure 3.31: Diagram of the position PID.

Fig. 3.31 depicts the diagram for the position PID. To ensure that the shaft closely follows the position command sent by the user, the PID proportional, integral and derivative gains had to be tuned. This was accomplished by online analysis of the step-response to a variation from 100.0° to 110.0° using the SMORA-A32 variant.

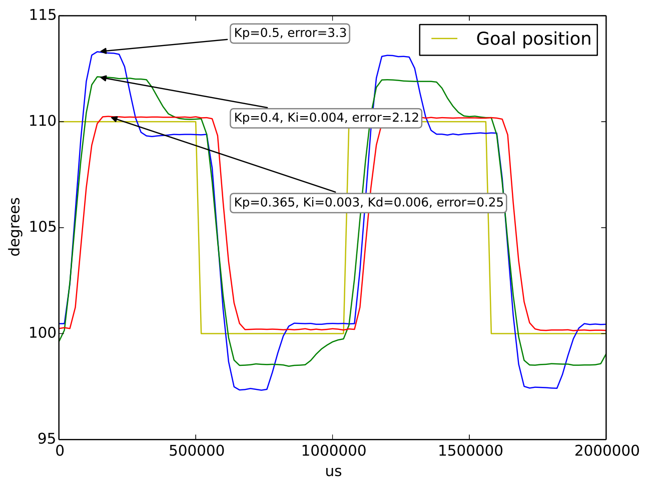


Figure 3.32: Position PID step-response with different gains.

Fig. 3.32 shows the step-response to different PID gains as well as the maximum position error for each one of them. With Kp=0.365, Ki=0.003 and Kd=0.006, a maximum error of 0.25° was accomplished.

Tuning was achived by setting a proportional gain with zero integral and derivative. Then increasing the integral gain until the steady-state error was minimised followed by a rise in the derivative gain until there was no over-shoot. Changing one gain can affect the properties that the others try to improve. For that reason, when setting one, the others had to be slightly tweaked. By the inherent physical limitations of the system, the PID output and motor temperature were monitored during the tuning process to ensure that the current peaks and temperature were constrained.

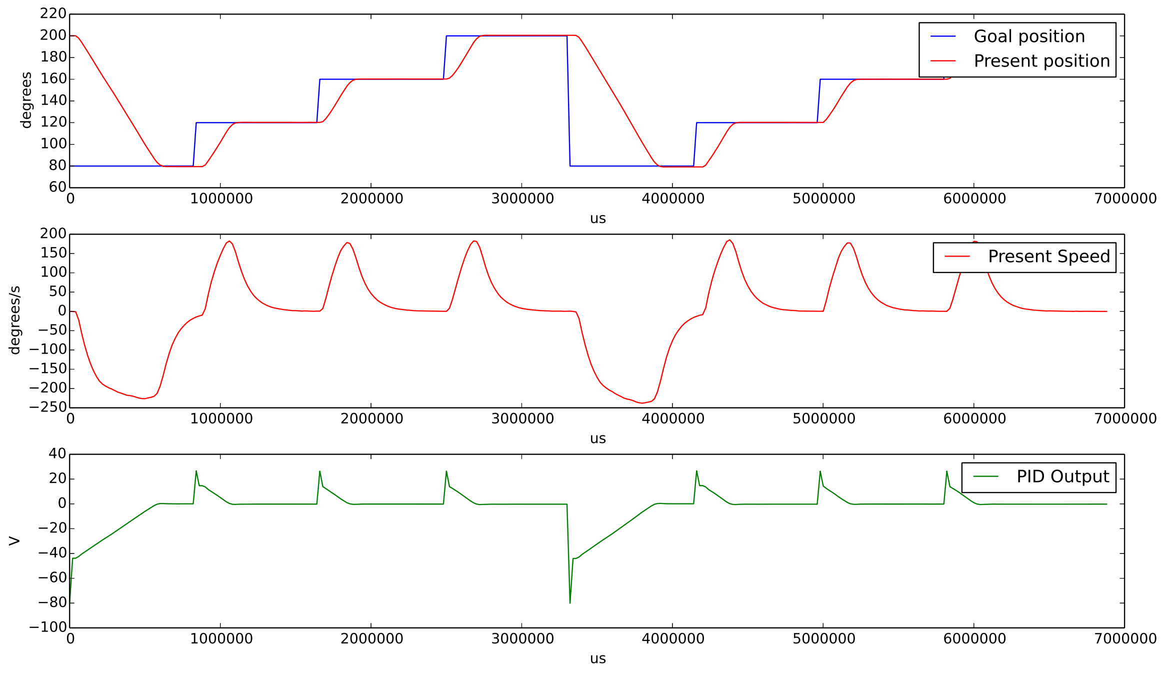


Figure 3.33: Position PID step-response to different position commands.

Fig. 3.33 shows the step-response to multiple position commands, along with the PID output in Volts and the present speed.

3.3.3 Speed compensator

While testing the performance of the speed PID, a non-linear but periodic error was identified (Fig. 3.34).

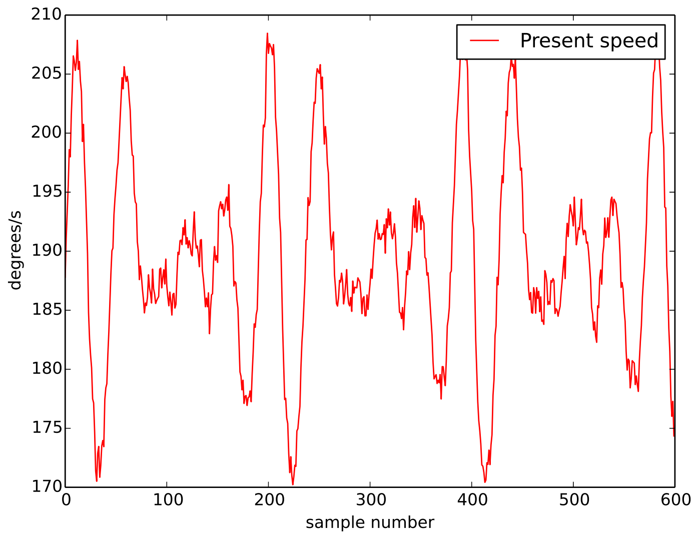


Figure 3.34: Present speed with a speed command of 200°/s.

When the speed was logged over multiple rotations and ordered by angle, it became clear that the error was always similar at specific positions (Fig. 3.35). The same picture shows the gathered data filtered with a 1st order Butterworth filter at the same frequency as the PID ($50Hz$).

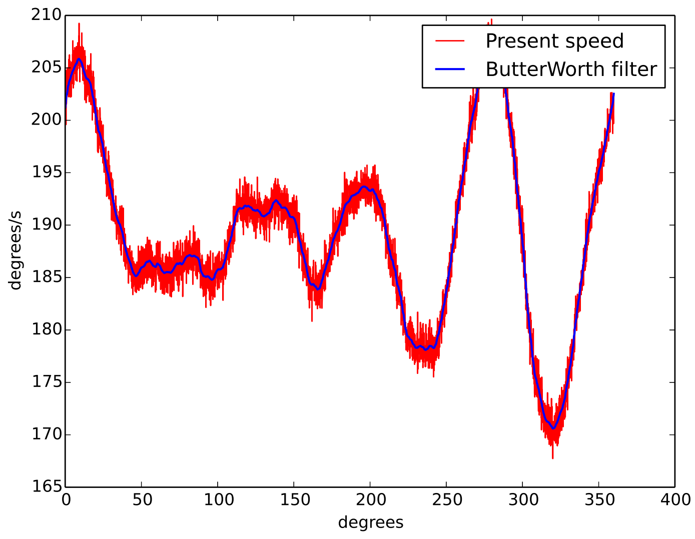


Figure 3.35: Present speed vs angle with a speed command of 200°/s.

This error was attributed to a misalignment in the gearbox system, as well as friction between the gears themselves. To correct for it, the concept of a Hammerstein model [24] (Fig. 3.36) was used. This model aims at improving the systems response by identifying the static nonlinear block and constructing another one that is it's inverse function. By combining the two, the error is "canceled" and the system becomes linear dynamic.

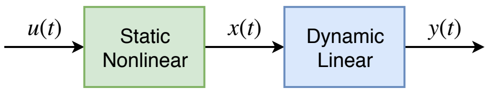


Figure 3.36: Hammerstein model.

To reduce the overhead of a lookup table for multiple speeds and positions, a single array with 360 integer values (one for each degree) was obtained from the error at a reference speed of 200.0°/s. The value in the array index corresponding to the integer value of the present position is then subtracted from the PWM output scaled by a gain calculated from the ratio between the goal and reference speed.

Fig. 3.37 shows multiple speeds with and without the compensation array applied to them. The array values can be found in Table A.1.

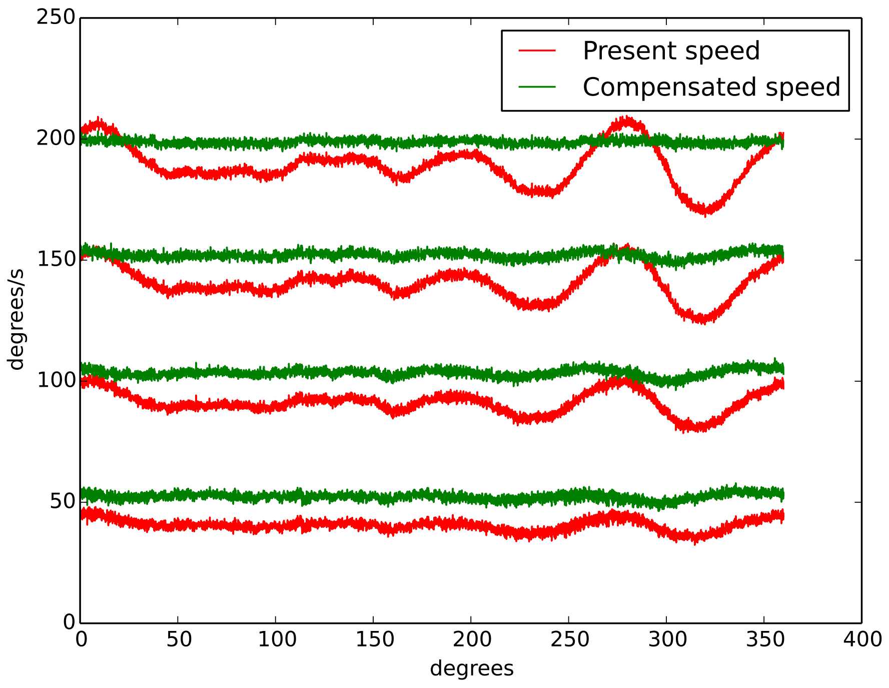


Figure 3.37: Plot of multiple speeds compensated.

In Fig. 3.38, the symmetry in the error for the different plotted speeds at, for example, 315.0° reflects the gears friction at that particular position. By applying a silicon based lubricant directly to the gears, the error was slightly lessened but not eliminated.

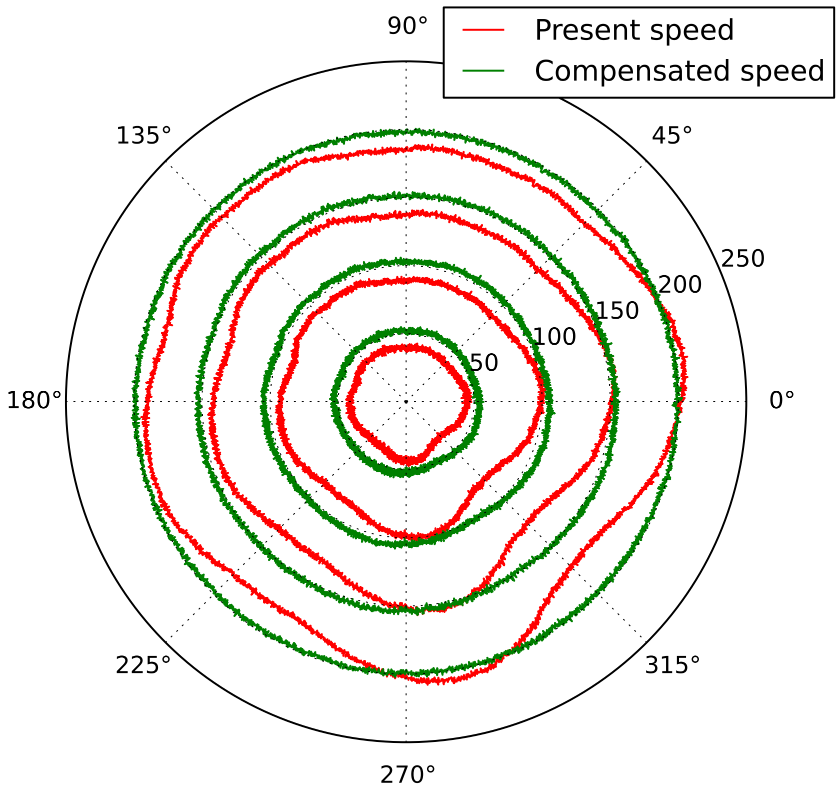


Figure 3.38: Polar plot of multiple speeds compensated.

3.3.4 Speed PID

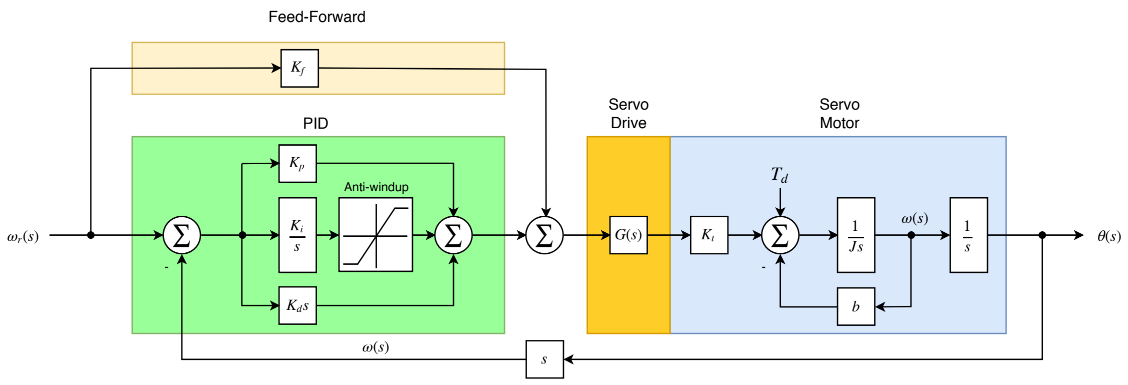


Figure 3.39: Diagram of the speed PID.

Fig. 3.39 depicts the diagram for the speed PID. Tuning it followed the same procedure as the one used for the position PID. However, a feed-forward gain was also included.

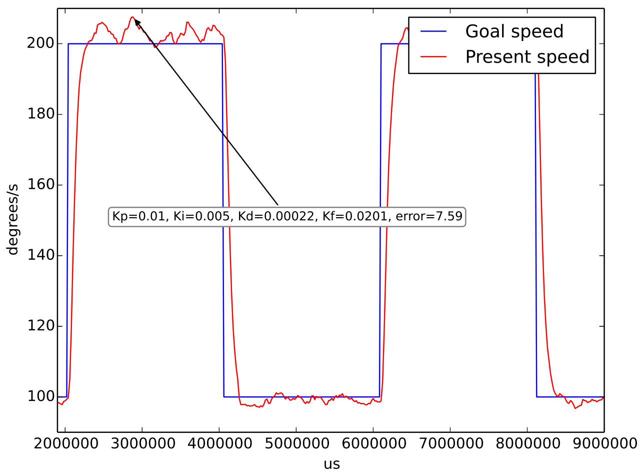


Figure 3.40: Speed PID step-response.

Fig. 3.40 shows the step-response to a variation in speed from 100.0°/s to 200.0°/s with Kp=0.01, Ki=0.005, Kd=0.00022 and Kf=0.0201. With this gains, a maximum error of 7.59°/s was accomplished.

The speed is calculated using equation 3.7.

$$speed = diffAngle(p(k-1),\ p(k))F \tag{3.7}$$

where:

$$F = PID\ frequency, \tag{3.8}$$

$$p = position \tag{3.9}$$

The function $diffAngle$ ensures that the angle difference is constrained between 180° and -180° degrees.

```
float diffAngle(float prevAngle, float newAngle){
  float diff = newAngle - prevAngle;
  if (diff > 180.0)
    diff -= 360.0;
  if (diff < -180.0)
    diff += 360.0;
  return diff;
}
```

The steady-state noise in Fig. 3.40 and 3.41 can be explained by the derivative of the noise from the position sensor as the speed is being calculated in the control loop, as well as the integer and discrete nature of the speed compensator. By using an array of floats instead of integer values, this compensation can be further improved at the cost of larger overhead.

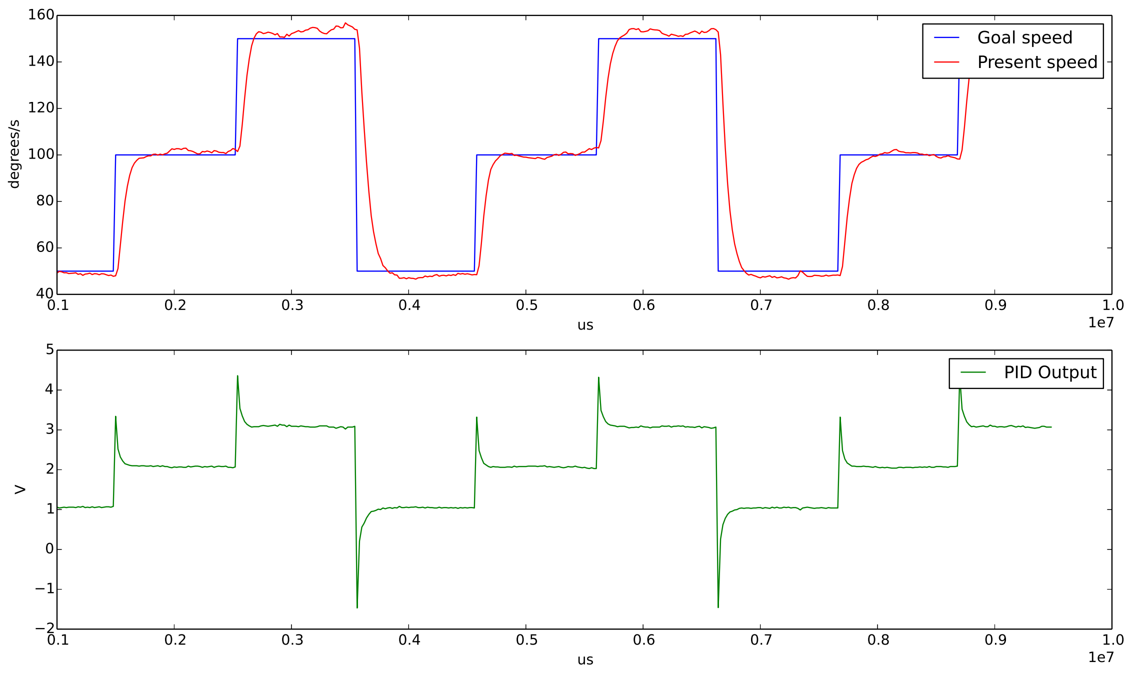


Figure 3.41: Speed PID step-response to different speed commands.

Chapter 4

Conclusion and future work

In this last chapter, the fulfilment of the objectives for this thesis is assessed and some suggestions for future work are postulated.

4.1 Goal accomplishments

The objective of this thesis was to create a servo motor optimised for robotic applications. Being such a broad subject meant that the research focus had to be kept to fields like design and prototyping of 3D models and PCBs, as well as the development of the software and firmware that could perform unit tests for each of the components, control the DC motor and prove the implemented features.

At first, a literature review about servo motors and DC motor control was conducted and different techniques and approaches gathered. Research was done on similar servo motors that were designed to aid in the development of robotics.

Through reverse-engineering of an already established servo design, a 3D model was drawn to constrain the dimensions of the electronic circuit that was to ultimately replace the original one. Features present in the original enclosure like the pins that hold the potentiometer in place were repurposed to hold the PCB with the magnetic encoder circuit. After researching the current technologies used for controlling small DC motors, the final components were chosen and sourced either through the manufacturer's sample program or other resellers.

To minimise the amount of hardware iterations and 3D printed parts until the design was finalised, the PCB was created in Eagle and modelled in 3D. This approach allowed for the overall dimensions to be verified before the PCBs and plastic parts were fabricated. The initial prototype was then manufactured by isolation routing a copper clad using a homemade CNC machine and soldering the integrated circuits. This allowed for the individual components like motor driver and current sensor to be tested and validated.

With the initial hardware verified, firmware features like controlling the shaft position started taking shape. Limitations on the bandwidth of the half-duplex circuit and the need to reposition some of the components so that the 3D enclosure would be feasible to be produced using a 3D printer led to the development of a second PCB prototype. With a PCB manufactured professionally, this prototype proved to be of much better quality than the first one.

Using the second prototype, additional firmware features like speed control and a proper communication protocol allowed for the tweaking of the PID parameters and monitoring of the motors behaviour to become swifter. With the improved software showing data that was previous unavailable, it became apparent that there was a quality issue with the original gearbox, which led to the research of methods to mitigate it. This in turn prompt the creation of the speed compensator that substantially minimised the speed error, albeit not eliminating it completely.

The firmware was consolidated so that the position and speed control modes could be integrated in the same control loop as the communication protocol. A third and last iteration of the PCBs in SMORA-A8 and SMORA-A32 to correct for minor oversights was then manufactured by OSHPark.

Even though it is not included in this thesis, the initial code for a motion control algorithm with trapezoidal velocity profile was written and a SCARA robotic arm was designed and fabricated so that a proof-of-concept with two SMORA devices daisy-chained together could be accomplished.

Overall, and even though the subject of the thesis was rather broad, the proposed objectives were successfully achieved with features that showed to be equivalent or even surpass the current devices available in the same price range.

4.2 Future work

This section describes some of the improvements that this thesis can undergo.

- **Improvement of the control algorithms through parameter estimation:** For the development of the proposed position and speed PIDs, the model for the motor was considered to be linear. To improve the accuracy of the control, the procedure described in the Literature Review for parameter estimation should be followed. This parameters can then be used in the feed-forward loop. Additionally, this calculations can be performed online, allowing for the control algorithms to compensate for the DC motor wear'n'tear over time or to identify the presence of different loads;
- **Motion control:** In order for the motion of multiple SMORA devices daisy-chained together to be synchronous, a motion control algorithm based on trapezoidal velocity profile or S-curve can be incorporated [25][26]. Along with a good estimation of the DC motor parameters, this algorithms will allow for precise position and velocity control and smooth motion while minimising jerk;
- **Improved gearbox:** Even though the MG996R servo used in this thesis has metal gears, the original enclosure is made of plastic. Coupled with poor axis alignment, the friction in different positions burdens the PID with trying to correct for error that could be avoided. Machining a metal enclosure with the axis pins correctly aligned or choosing a different donor servo altogether could minimise this errors;
- **Attitude control:** Although the hardware designed for this thesis allows for the inquiry of the servos present attitude through the included MPU-6050 accelerometer and gyroscope, the firmware doesn't take advantage of them for the control. Being a novelty in servo motor technology, this feature could be further explored so that new behaviours could be obtained. As an example, the gyroscope could be used to calibrate the $0°$ position reference for the different segments in a robotic arm. The shaft position of the servo attached to one joint could be used in conjunction with the gyroscope in the servo from the adjacent joint. By controlling the position of one while reading the data provided by the other, the whole arm can be calibrated online in a very short amount of time;
- **Port hardware to other servos:** The current prototypes were designed to fit into the MG996R enclosure and similar form-factor. By repositioning the components in the PCB, a generic board could be made so that other types and sizes of servos could take advantage of SMORA;

Appendix A

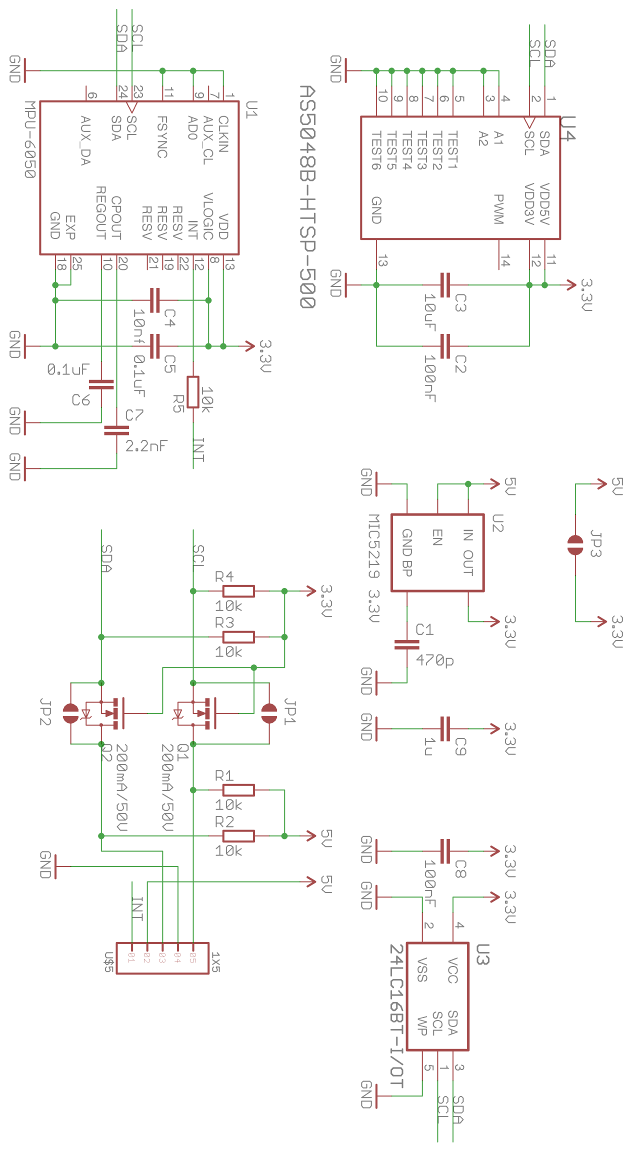


Figure A.1: Position encoder full schematic

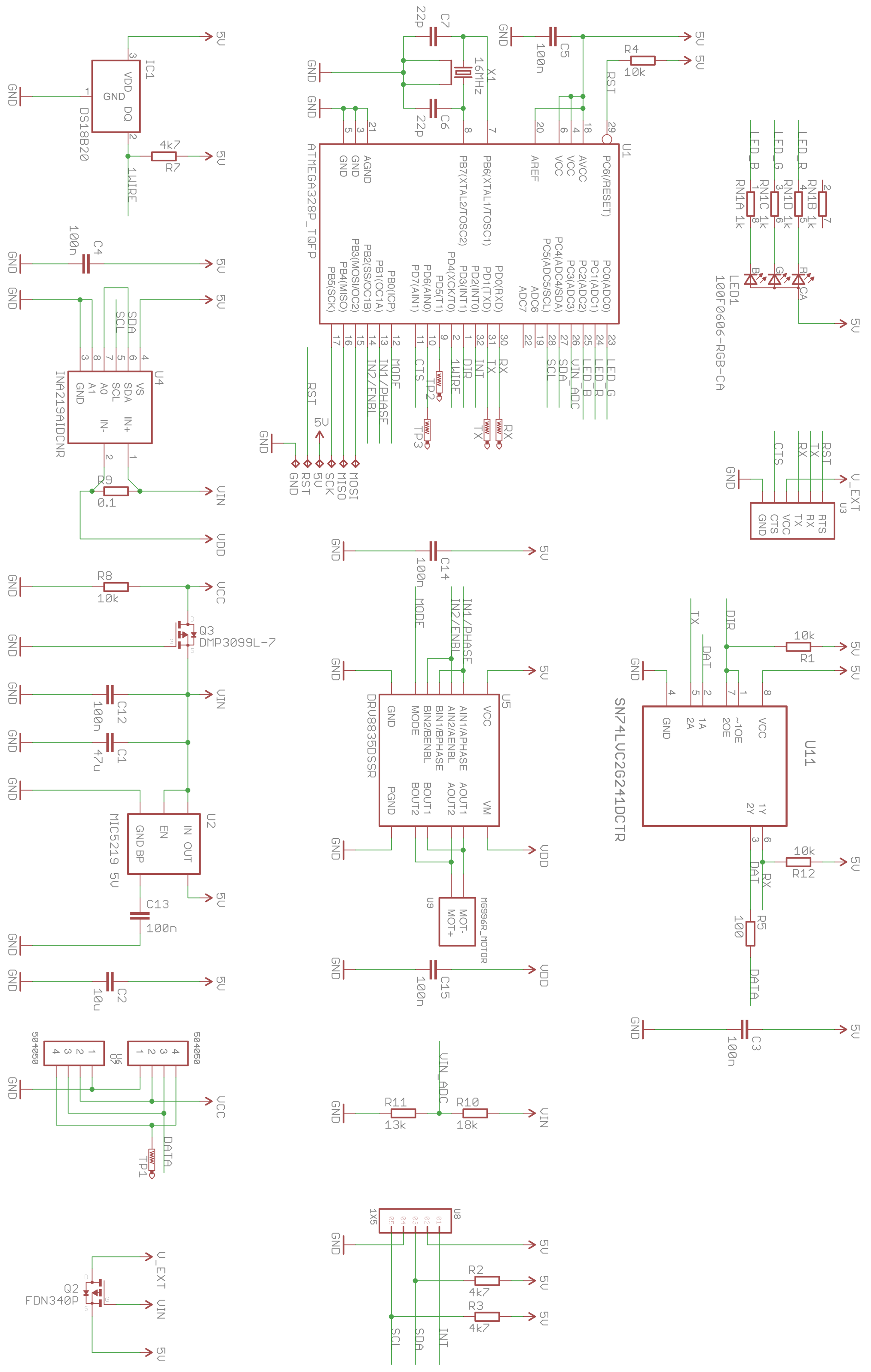


Figure A.2: SMORA-A8 main board schematic.

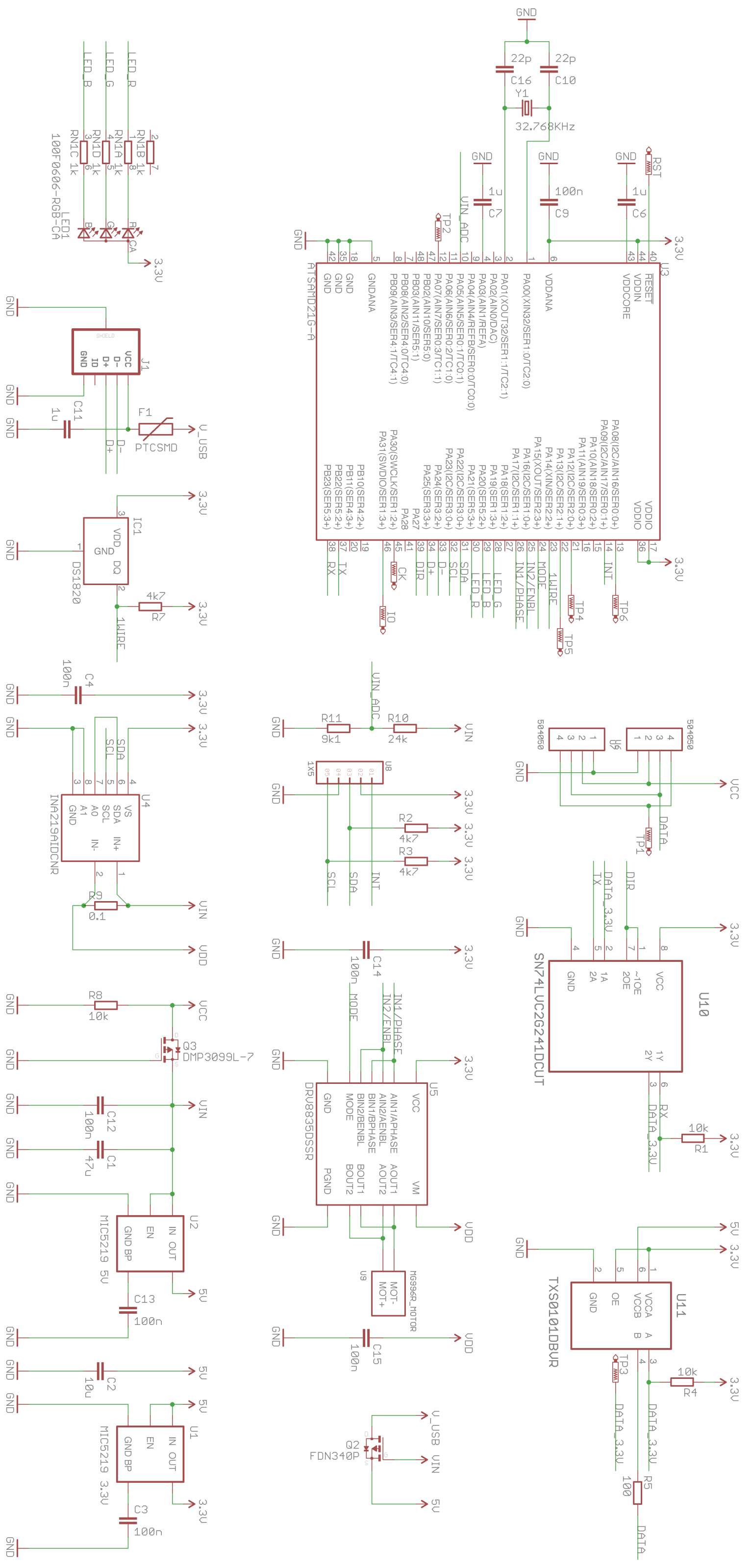


Figure A.3: SMORA-A32 main board schematic.

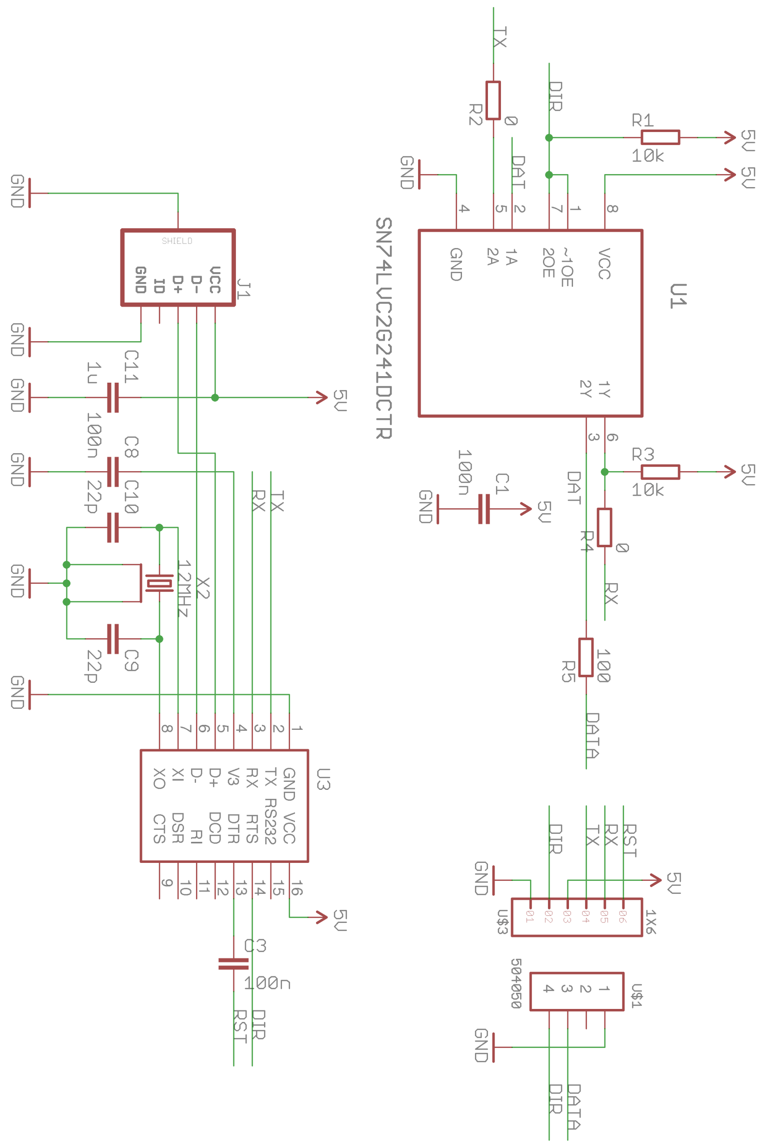
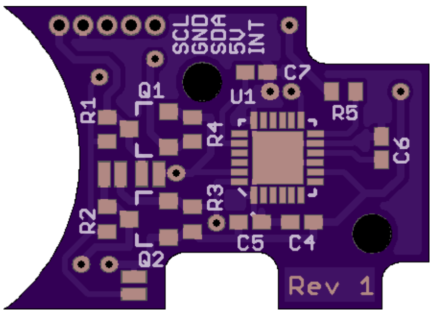
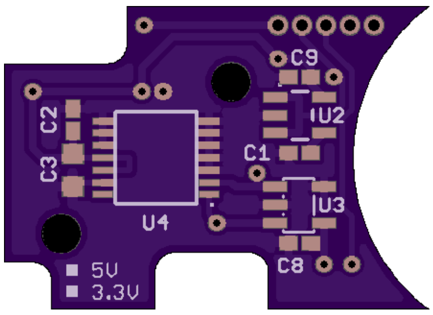


Figure A.4: SMORA-A8 programmer and interface circuit schematic.

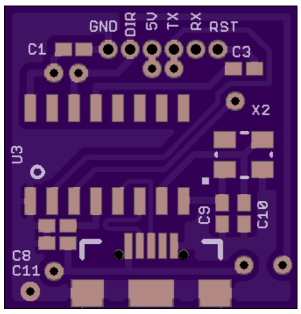


(a) Top.

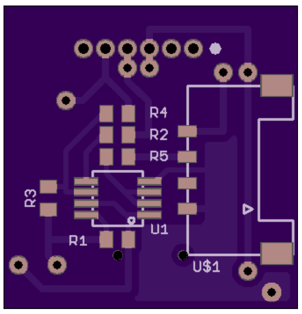


(b) Bottom.

Figure A.5: Position encoder board.

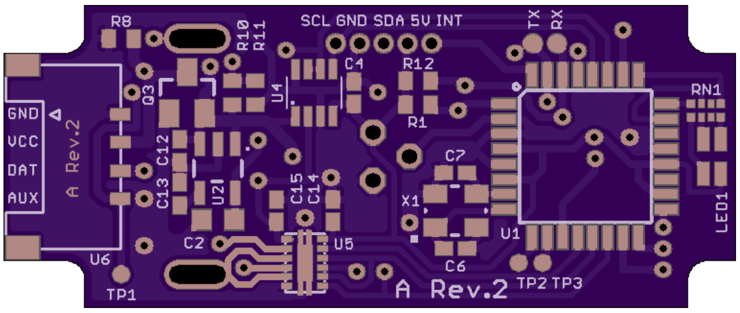


(a) Top.



(b) Bottom.

Figure A.6: SMORA-A8 programmer board.


(a) Top.

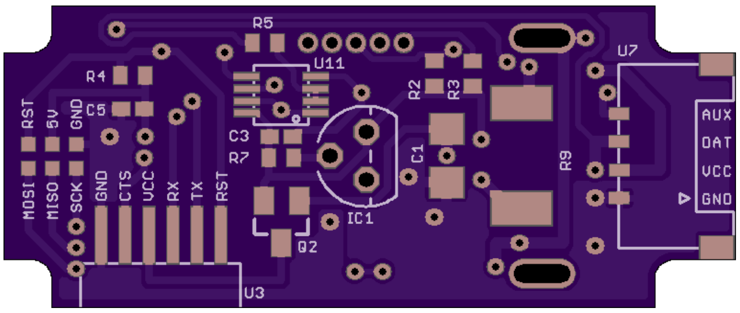
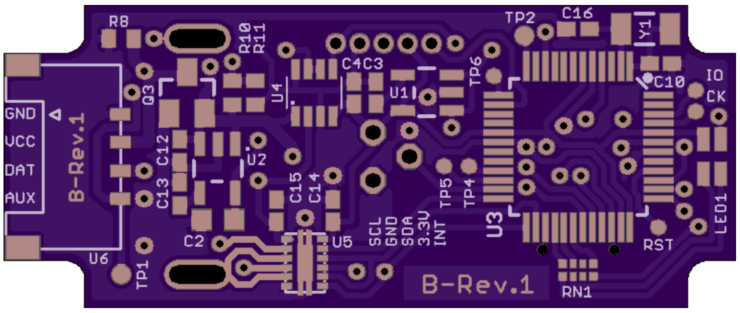

(b) Bottom.

Figure A.7: SMORA-A8 board.


(a) Top.

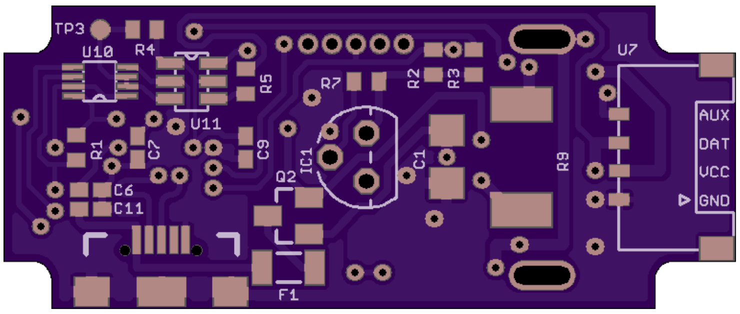

(b) Bottom.

Figure A.8: SMORA-A32 board.

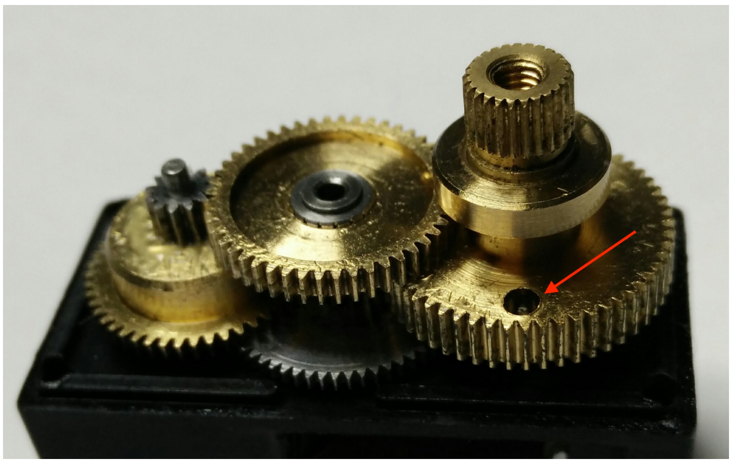


Figure A.9: MG996R gearbox with limit pin removed.

Table A.1: Speed compensation array

1-36	37-72	73-108	109-144	145-180	181-216	217-252	253-288	289-324	325-360
12	0	-3	0	2	1	-3	-3	13	-17
13	0	-3	0	2	1	-4	-2	12	-16
13	0	-3	0	1	1	-5	-1	11	-16
14	-1	-2	1	1	2	-5	0	10	-15
14	-1	-2	1	1	2	-6	0	9	-14
15	-2	-2	1	1	2	-7	2	8	-14
15	-2	-2	1	0	2	-8	3	7	-13
15	-3	-2	2	0	3	-8	4	5	-12
15	-3	-2	2	0	3	-9	4	4	-11
15	-3	-2	2	0	3	-9	5	2	-10
15	-3	-2	2	-1	3	-10	6	1	-9
15	-3	-2	2	-1	3	-10	7	0	-7
15	-3	-2	2	-2	3	-10	8	-1	-6
14	-3	-2	2	-2	3	-10	9	-3	-5
14	-3	-2	2	-3	4	-10	10	-5	-4
14	-3	-2	2	-3	4	-10	11	-6	-3
13	-3	-3	2	-4	4	-10	11	-8	-2
13	-2	-3	2	-4	4	-10	12	-9	-1
12	-2	-3	2	-4	4	-10	13	-11	0
11	-2	-3	1	-4	3	-10	14	-12	0
10	-2	-4	1	-4	3	-11	14	-13	1
10	-3	-4	1	-4	3	-10	15	-14	2
9	-3	-4	1	-4	3	-10	15	-14	2
8	-3	-4	1	-4	3	-10	16	-15	3
8	-3	-4	1	-4	3	-10	16	-16	4
7	-3	-4	1	-3	2	-10	16	-16	5
6	-3	-3	2	-3	2	-10	17	-17	5
5	-3	-3	2	-2	1	-10	17	-17	6
5	-3	-3	2	-2	1	-9	17	-17	7
4	-3	-3	2	-2	0	-9	17	-18	7
3	-3	-3	2	-1	0	-8	16	-18	8
3	-3	-2	2	-1	0	-7	16	-18	9
2	-3	-2	2	0	-1	-6	16	-18	10
1	-3	-2	2	0	-2	-5	15	-18	11
1	-3	-1	2	0	-2	-4	15	-17	11
0	-3	-1	2	0	-3	-4	14	-17	12

Table A.2: SMORA-A8 main PCB BOM list as of January, 2017.

Quantity	Value	Reference	Seller	Single Price	Price
1	MG996R_MOTOR	MG996R	Aliexpress	3,52	3,52
1	DS1820	DS18B20+PAR	Mouser	2,62	2,62
1	DRV8835DSSR	DRV8835DSSR	Mouser	1,74	1,74
1	INA219AIDCNR	INA219AIDCNR	Mouser	2	2
1	SN74LVC2G241DCTR	SN74LVC2G241DCTR	Mouser	0,651	0,651
1	ATMEGA328P_TQFP	ATMEGA328P-AU	Mouser	3,04	3,04
1	MIC5219 5V	MIC5219-5,0YM5-TR	Mouser	0,905	0,905
1	DMP3099L-7	DMP3099L-7	Mouser	0,311	0,311
1	FDN340P	FDN340P	Mouser	0,349	0,349
1	100F0606-RGB-CA	EL-19-337/R6GHBHC-A01	Mouser	0,575	0,575
1	16MHz	ABM8G-16,000MHZ-4Y-T3	Mouser	0,698	0,698
2	Molex PicoLock	504050-0491	Mouser	0,943	1,886
1	47u	EMK316BBJ476ML-T	Mouser	0,792	0,792
1	10u	GRM188R61C106MAALD	Mouser	0,321	0,321
7	100n		Mouser	0,01	0,07
2	22p		Mouser	0,01	0,02
1	18k	CPF0402B18KE1	Mouser	0,2	0,2
1	13k	CPF0402B13KE1	Mouser	0,665	0,665
6	10k	AF0402JR-0710KL	Mouser	0,104	0,624
1	4k7	WR04X4701FTL	Mouser	0,108	0,108
1	1k	WR04X1001FTL	Mouser	0,108	0,108
1	0.1R	RL73K3AR10	Mouser	0,124	0,124
					21,327 €

Table A.3: SMORA-A8 encoder PCB BOM list as of January, 2017.

Quantity	Value	Reference	Seller	Single Price	Price
1	AS5048B-HTSP-500	AS5048B-HTSP-500	Mouser	6,08	6,08
1	MPU-6050	MPU-6050	Aliexpress	1,62	1,62
1	MIC5219 3,3V	MIC5219-3,3YM5-TR	Mouser	0,905	0,905
1	24LC16BT-I/OT	24LC16BT-I/OT	Mouser	0,302	0,302
2	200mA/50V	BSS138	Mouser	0,179	0,358
1	10uF	GRM188R61C106MAALD	Mouser	0,321	0,321
1	1u	CC0402KRX5R7BB105	Mouser	0,094	0,094
4	100nF		Mouser	0,01	0,04
1	10nf		Mouser	0,01	0,01
1	2,2nF		Mouser	0,01	0,01
1	470p	VJ0402Y471KXACW1BC	Mouser	0,066	0,066
5	10k	AF0402JR-0710KL	Mouser	0,104	0,52
					10,326 €

Table A.4: SMORA-A32 main PCB BOM list as of January, 2017.

Quantity	Value	Reference	Seller	Single Price	Price
1	MG996R_MOTOR	MG996R	Aliexpress	3,52	3,52
1	DS1820	DS18B20+PAR	Mouser	2,62	2,62
1	DRV8835DSSR	DRV8835DSSR	Mouser	1,74	1,74
1	INA219AIDCNR	INA219AIDCNR	Mouser	2	2
1	SN74LVC2G241DCUT	SN74LVC2G241DCUT	Mouser	0,651	0,651
1	ATSAMD21G-A	ATSAMD21G18A-AU	Mouser	4,58	4,58
1	MIC5219 5V	MIC5219-5,0YM5-TR	Mouser	0,905	0,905
1	MIC5219 3,3V	MIC5219-3,3YM5-TR	Mouser	0,905	0,905
1	TXB0101DBVR	TXS0101DBVR	Mouser	0,622	0,622
1	DMP3099L-7	DMP3099L-7	Mouser	0,311	0,311
1	FDN340P	FDN340P	Mouser	0,349	0,349
1	100F0606-RGB-CA	EL-19-337/R6GHBHC-A01	Mouser	0,575	0,575
1	32,768KHz	FX135A-327	Mouser	0,368	0,368
1	PTCSMD	MF-NSMF050-2	Mouser	0,313	0,313
2	Molex PicoLock	504050-0491	Mouser	0,943	1,886
1	47u	EMK316BBJ476ML-T	Mouser	0,943	0,943
2	10u	GRM188R61C106MAALD	Mouser	0,321	0,642
3	1u	GRM155R61C105KA12D	Mouser	0,094	0,282
7	100n		Mouser	0,01	0,07
2	22p		Mouser	0,01	0,02
1	24k	RG1005P-912-D-T10	Mouser	0,217	0,217
4	10k	AF0402JR-0710KL	Mouser	0,104	0,416
1	9.1k	RG1005P-912-D-T10	Mouser	0,217	0,217
1	4k7	WR04X4701FTL	Mouser	0,108	0,108
1	1k	WR04X1001FTL	Mouser	0,108	0,108
1	0.1R	RL73K3AR10	Mouser	0,124	0,124
					24.492 €

Table A.5: SMORA-A32 encoder PCB BOM list as of January, 2017.

Quantity	Value	Reference	Seller	Single Price	Price
1	AS5048B	AS5048B-HTSP-500	Mouser	6,08	6,08
1	MPU-6050	MPU-6050	Aliexpress	1,62	1,62
1	24LC16BT-I/OT	24LC16BT-I/OT	Mouser	0,302	0,302
1	10uF	GRM188R61C106MAALD	Mouser	0,321	0,321
1	1u	CC0402KRX5R7BB105	Mouser	0,094	0,094
4	100nF		Mouser	0,01	0,04
1	10nf		Mouser	0,01	0,01
1	2,2nF		Mouser	0,01	0,01
5	10k	AF0402JR-0710KL	Mouser	0,104	0,52
					8.997 €

References

[1] Stuart Mcfarlan (username oomlout). URL: https://www.flickr.com/photos/snazzyguy/3465236215.

[2] Trossenrobotics. Robotis ax-12a. [Online; accessed 14 January 2017]. URL: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx.

[3] D. Kaiser. Fundamentals of servo motion control. *Motion System Design*, 43(9):22+24+26+28, 2001. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-17944399301&partnerID=40&md5=7afec604117ae23eac145a7eb3a02429.

[4] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall PTR, Fourth edition, 2001.

[5] J.G. Ziegler and N.B Nichols. Optimum settings for automatic controllers. *Transactions of the American Society of Mechanical Engineers (ASME)*, 64:759–768, 1942.

[6] M.R. Elhami and D.J. Brookfield. Identification of coulomb and viscous friction in robot drives: An experimental comparison of methods. 210(6):529–540, 1996. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0030406730&partnerID=40&md5=0291f9e41f40b140dc2a7580c2df36a3.

[7] J. Gonçalves, J. Lima, and P.G. Costa. Dc motors modeling resorting to a simple setup and estimation procedure. *Lecture Notes in Electrical Engineering*, 321 LNEE:441–447, 2015. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84907312811&partnerID=40&md5=71134001d6fbe55c9409495c1b64dfd1, doi:10.1007/978-3-319-10380-8_42.

[8] Openservo. Openservo circuit. [Online; accessed 13 July 2016]. URL: http://openservo.com.

[9] 01mechatronics. Supermodified servomotor. [Online; accessed 13 July 2016]. URL: http://www.01mechatronics.com/product/supermodified-v30-rc-servos.

[10] Moti. Moti servomotor. [Online; accessed 13 July 2016]. URL: http://moti.ph.

[11] Robotis. Robotis ax-12a. [Online; accessed 14 January 2017]. URL: http://www.robotis.us/ax-12a/.

[12] AMS. As5048b datasheet. [Online; accessed 20 January 2017]. URL: http://ams.com/eng/content/download/438523/1341157/file/AS5048_DS000298_3-00.pdf.

[13] Microchip. 24lc16b datasheet. [Online; accessed 20 January 2017]. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/20002213B.pdf.

[14] InvenSense. Mpu6050 datasheet. [Online; accessed 20 January 2017]. URL: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[15] Micrel. Mic5219 datasheet. [Online; accessed 20 January 2017]. URL: http://ww1.microchip.com/downloads/en/devicedoc/mic5219.pdf.

[16] Fairchild. Bss138 datasheet. [Online; accessed 20 January 2017]. URL: https://www.fairchildsemi.com/datasheets/BS/BSS138.pdf.

[17] Maxim Integrated. Ds18b20 datasheet. [Online; accessed 20 January 2017]. URL: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[18] Texas Instruments. Ina219a datasheet. [Online; accessed 20 January 2017]. URL: http://www.ti.com/lit/ds/symlink/ina219.pdf.

[19] Texas Instruments. Sn74lvc2g241 datasheet. [Online; accessed 20 January 2017]. URL: http://www.ti.com/lit/ds/symlink/sn74lvc2g241.pdf.

[20] Texas Instruments. Drv8835 datasheet. [Online; accessed 20 January 2017]. URL: http://www.ti.com/lit/ds/symlink/drv8835.pdf.

[21] Diodes Incorporated. Dmp3099l datasheet. [Online; accessed 20 January 2017]. URL: http://www.diodes.com/_files/datasheets/DMP3099L.pdf.

[22] Maxim Integrated. Application note 636. [Online; accessed 14 January 2017]. URL: http://pdfserv.maximintegrated.com/en/an/AN636.pdf.

[23] Texas Instruments. Txs0101 datasheet. [Online; accessed 20 January 2017]. URL: http://www.ti.com/lit/ds/symlink/txs0101.pdf.

[24] Tolgay Kara and Ilyas Eker. Nonlinear modeling and identification of a dc motor for bidirectional operation with real time experiments. *Energy Conversion and Management*, 45(7-8):1087 – 1106, 2004. URL: www.sciencedirect.com/science/article/pii/S0196890403002139, doi:http://dx.doi.org/10.1016/j.enconman.2003.08.005.

[25] A. Shahzadeh, A. Khosravi, and S. Nahavandi. Path planning for cnc machines considering centripetal acceleration and jerk. pages 1759–1764, 2013. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893571387&doi=10.1109%2fSMC.2013.303&partnerID=40&md5=bdfd77193a76ca3ad2d0950c7219f2ee, doi:10.1109/SMC.2013.303.

[26] H. Li. A jerk-constrained asymmetric motion profile for high-speed motion stages to reduce residual vibration. 53(2):149–156, 2016. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84957089645&doi=10.1504%2fIJCAT.2016.074453&partnerID=40&md5=e37962fcc0a75f1e47966ce60c3610b7, doi:10.1504/IJCAT.2016.074453.