Faculty of Engineering and the Built Environment
Department of Electrical Engineering

Hopping Control of a Single Leg Robot

Prepared for Dr. Amir Patel.
Submitted to the Department of Electrical Engineering
at the University of Cape Town in partial fulfilment of the academic requirements
for a Bachelor of Science degree in Mechatronics.

Benjamin Scholtz

October 19, 2016

Keywords: impedance control, dynamic control, force control, mechatronics

To my dearest...

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This final year project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Name: Benjamin Scholtz
Signature: _____
Date: October 19, 2016

Abstract

Acknowledgements

Amir Patel Callen Fisher Craig Burden Gareth Callanan Roberto Aldera

Terms of Reference

Description

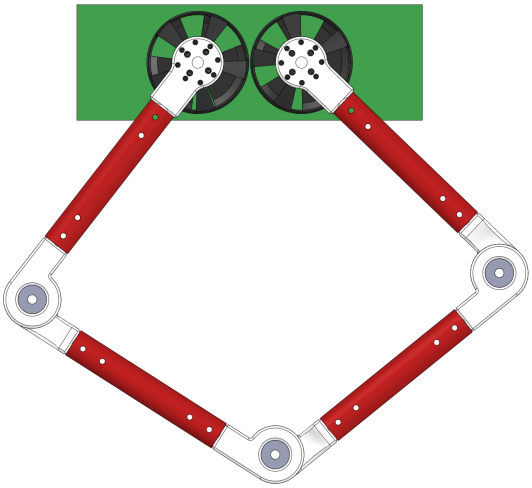


Figure 1: Version 1 of Baleka leg platform (Ben Bingham, 2016).

The Mechatronics Lab has recently developed a single leg, direct drive robot, Baleka, to investigate modelling and control of rapid accelerations. This project will involve the design of a control system to perform stable hopping with the robot. Various controller algorithms will be investigated and compared (eg. PID, MPC, etc.). The project will also involve developing a test rig for the robot.

Deliverables

- Mathematical model of the hopping robot must be developed in Simulink/Matlab
- Hopping controller design
- Mechanical design of the test rig
- Experimental testing of the robot

Skills/Requirements

- Mathematical Modelling
- Mechatronics Design
- Control Systems
- Embedded Systems
- Strong Practical and Mathematical skills required

ELO3: Engineering Design

Perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.

The student is expected to design:

- Robot feedback control system
- Rig for testing of hopping motion

Area of Research

- Bio-inspired robotics
- Control systems

Extra Information

`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5648972`

`http://kodlab.seas.upenn.edu/uploads/Avik/compositionTR_sc.pdf`

Contents

List of Figures

List of Tables

List of Source Codes

1 Introduction

> *"Begin at the beginning," the King said, gravely, "and go on till you come to an end; then stop."*
>
> — Lewis Carroll, *Alice in Wonderland*

With a hop, skip, and a jump – the journey begins!

1.1 Background

1.2 Objectives of the Study

1.2.1 Problems to be Investigated

1.2.2 Research Questions

1.2.3 Purpose of the Study

1.3 Scope and Limitations

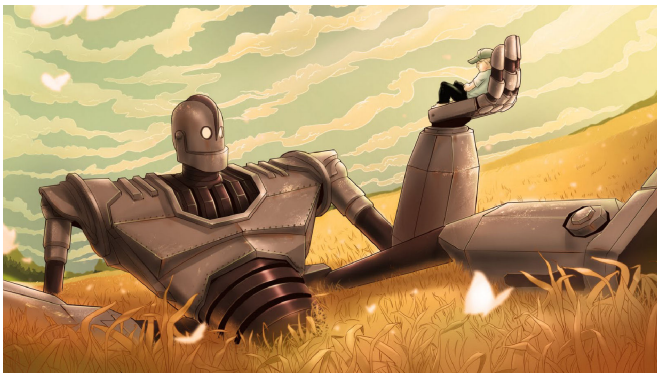
1.4 Plan of Development

2 Literature Review

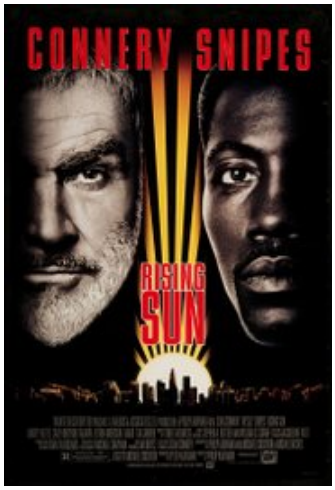
2.1 Introduction

Bio-inspired robots have fascinated humans since the Greek mathematician, Archytas of Tarentum, built the first true mechanical robot, where a robot is some device performing an automated mechanical task. His mechanical steam powered bird was just the start.[**?**]

Would be engineers take their inspiration from popular culture with The Iron Giant and B.E.N. fresh in mind. The Rising Sun included robots developed by Marc Raibert, founder of the CMU (now MIT) Leg Laboratory, who pioneered self-balancing dynamic control of hopping robots.



(a) The Iron Giant (1999).



(b) Rising Sun (1993).

(c) Treasure Planet (2002).

Figure 2.1: Humanoid robots in popular culture.

2.2 State of the Art

2.2.1 Monoped Robots

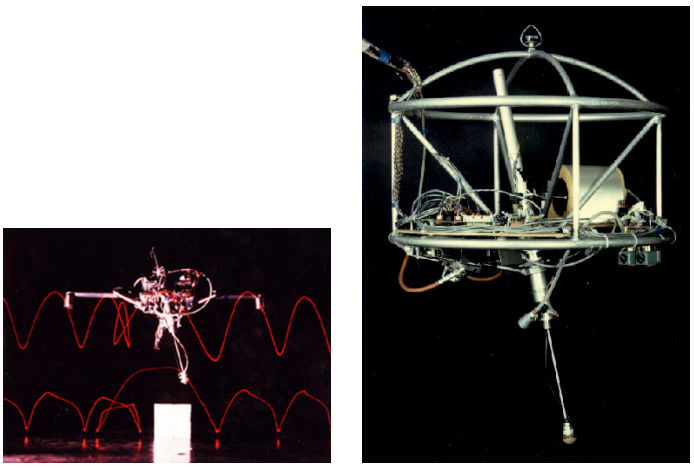
2.2.2 Biped Robots

2.2.3 Quadruped Robots

2.2.4 Bio-inspired Legged Robotics

2.2.5 Humanoid Robots

2.2.6 Closed Kinematic Chain Leg



(a) Planar One-Leg Hopper - MIT Leg Laboratory (1980-1982).

(b) 3D One-Leg Hopper - MIT Leg Laboratory (1983-1984).

Figure 2.2: Monoped robots.

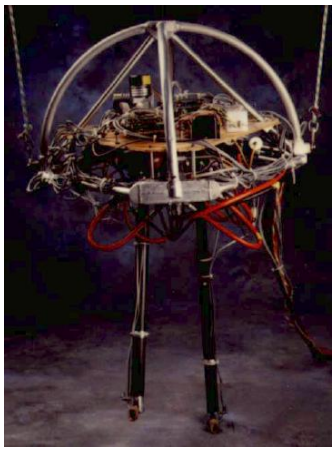
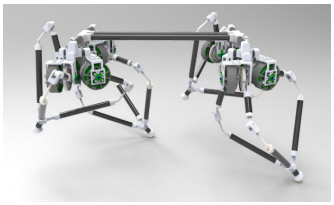


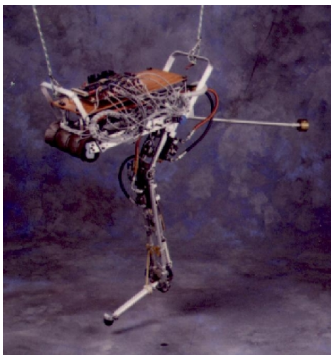
Figure 2.3: 3D Biped - MIT Leg Laboratory (1989-1995).

(a) Quadruped - MIT Leg Laboratory (1984-1987).



(b) GOAT 3-DOF Leg Topology - (Kalouche, 2016).

Figure 2.4: Quadruped robots.



(a) Uniroo - MIT Leg Laboratory (1991-1993).

(b) Spring Flamingo - MIT Leg Laboratory (1996-2000).

Figure 2.5: Bio-inspired legged robots.

Figure 2.6: Atlas Humanoid Robot - Boston Dynamics (2013).

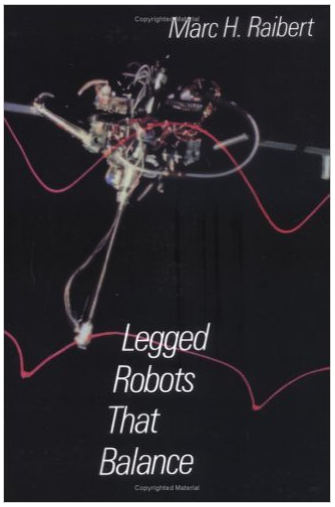
Figure 2.7: Closed Kinematic Chain Leg using Raibert's Scissor Algorithm (Duperret, Koditschek, 2016).[?]

2.3 Legged Locomotion in Nature

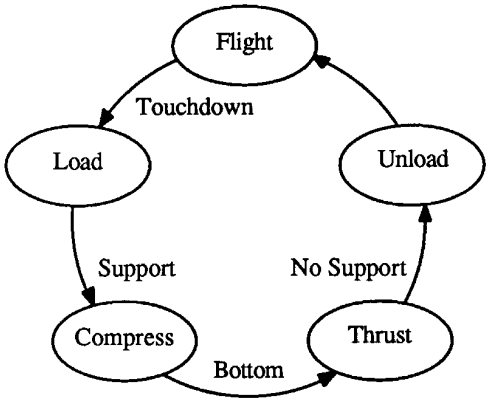
2.4 Raibert Control

2.4.1 Raibert's Scissor Algorithm

2.4.2 Phases of Motion



(a) Legged Robots That Balance - Marc H. Raibert (1986).



(b) Raibert control state machine.

Figure 2.8: Legged Robots That Balance cover page and exert.[?]

2.5 Applications in Industry

2.5.1 Soft-robotics

Factories safe human robot interaction Handling of compliant products (farming, manufacturing)

[?]

2.5.2 Bose Active Suspension

2.5.3 Dynamic Stability vs Static Stability

2.5.4 Phases of Motion

2.5.5 Leg Stance Control

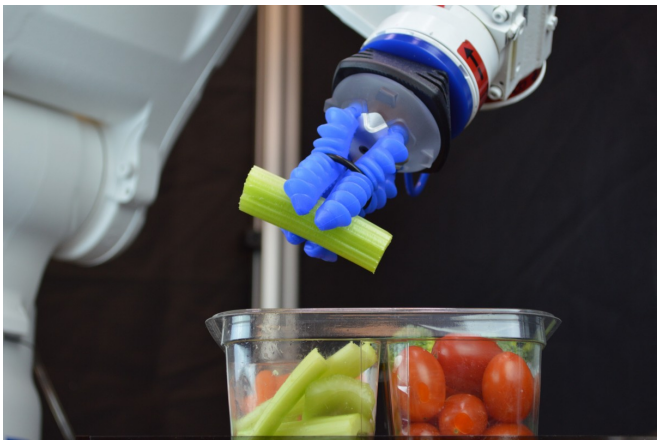


Figure 2.9: Compliant soft robotic handling (Forbes, 2016).

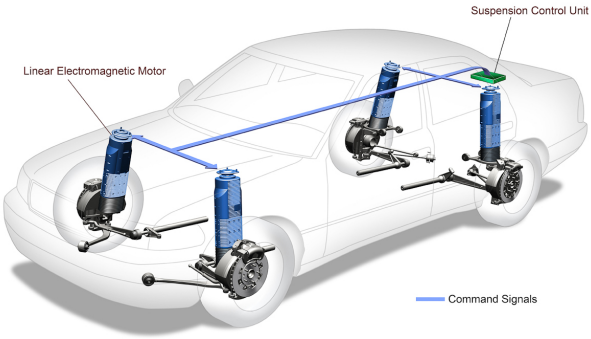


Figure 2.10: Bose Active Suspension (Bose Corporation, 1980s)[**?**].

2.6 Force Control

3 Project Plan and Methodology

4 Theory Development

4.1 General Co-ordinates

5 System Modelling and Simulation

6 Leg Design and Construction

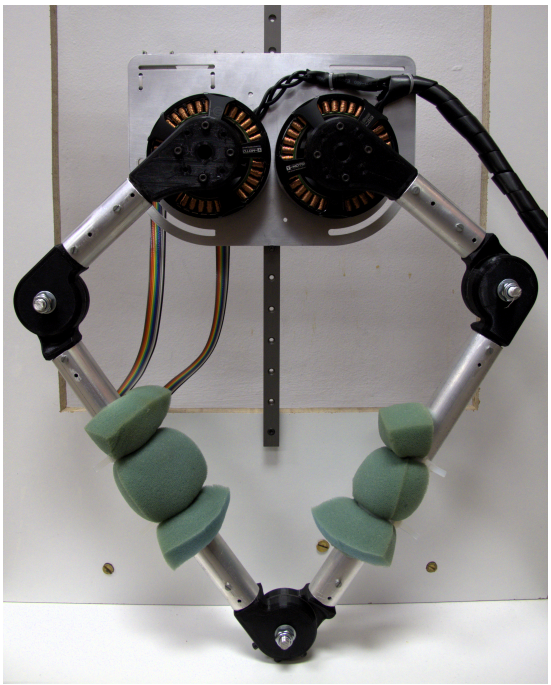


Figure 6.1: Final leg design mounted to platform and linear guide.

6.1 Geometry

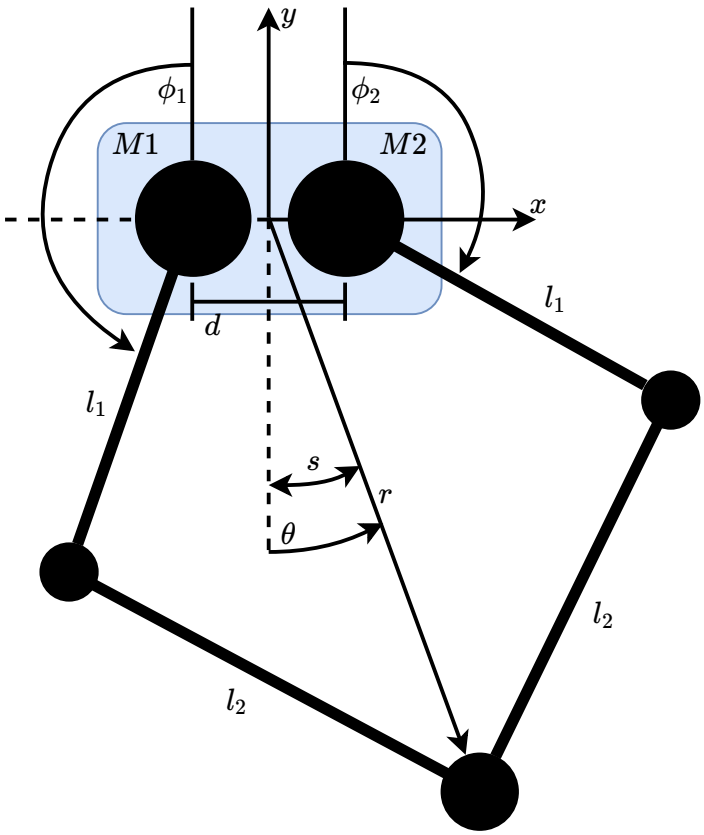


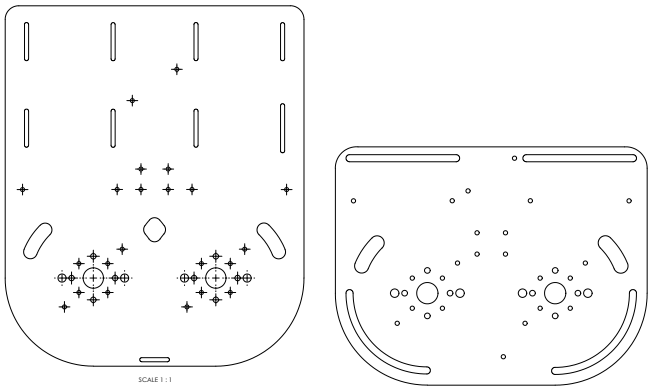
Figure 6.2: Geometric view of leg.

6.2 Mechanics and Construction

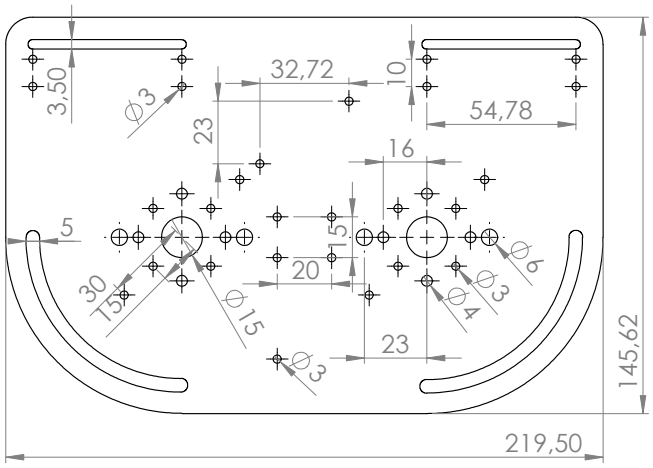
6.2.1 Alumnium Mounting Plate Design

6.2.2 Linear Guide

6.2.3 CAD Robotic Leg Assembly



(a) CAD mounting plate V1. (b) CAD mounting plate V3.1.3.



(c) CAD mounting plate final design.

Figure 6.3: Leg mounting plate iterations.

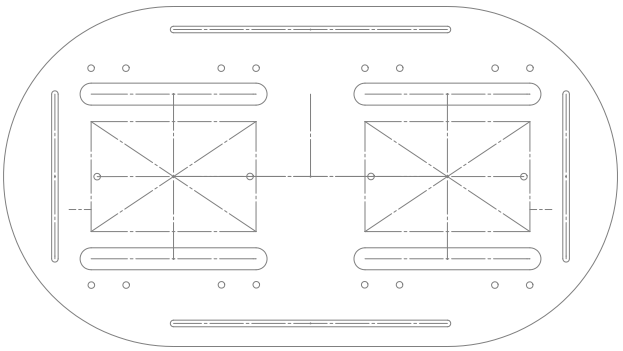


Figure 6.4: Motor driver interface mounting plate.

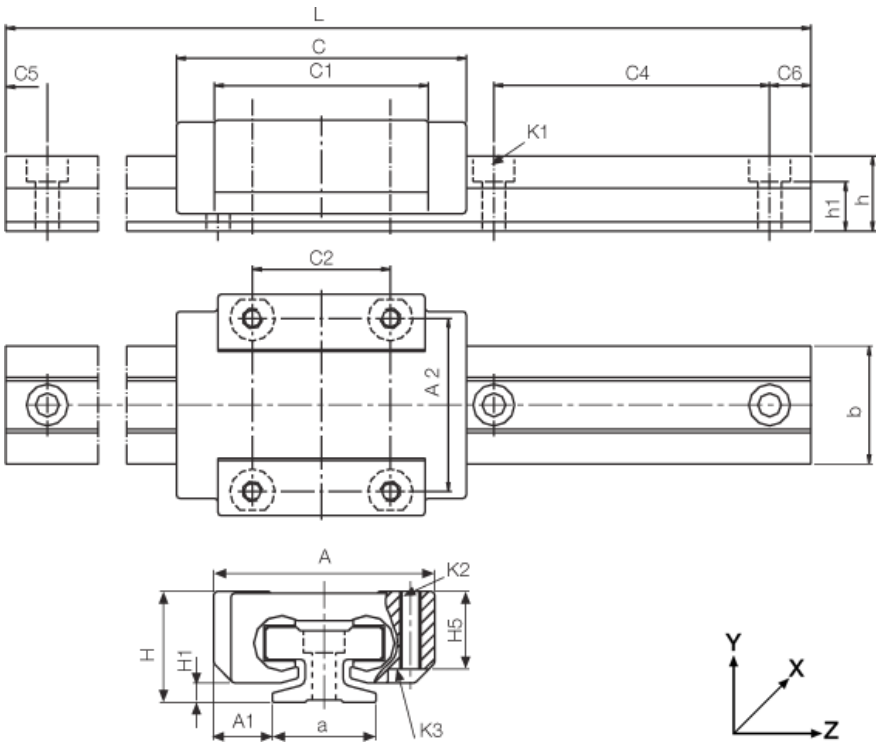


Figure 6.5: igus DryLin T - Low-profile linear guide.

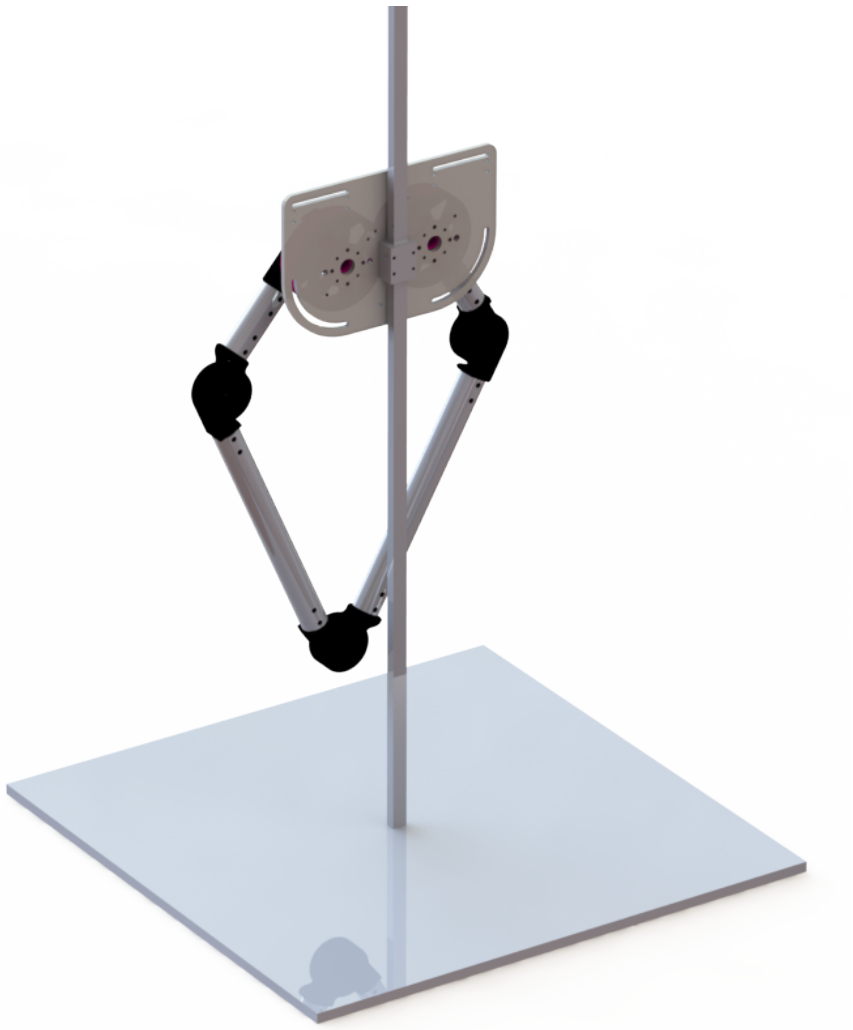


Figure 6.6: Linear guide mounted leg model (CAD Solidworks assembly).

6.3 Electronics and Communication

6.3.1 Accelerometer and Gyroscope

6.3.2 Distance Sensor

6.3.3 Microcontroller

6.4 Communication Interfaces

6.4.1 Shielding

6.5 Motors and Drivers

6.5.1 Driver Selection

6.5.2 Motor Selection

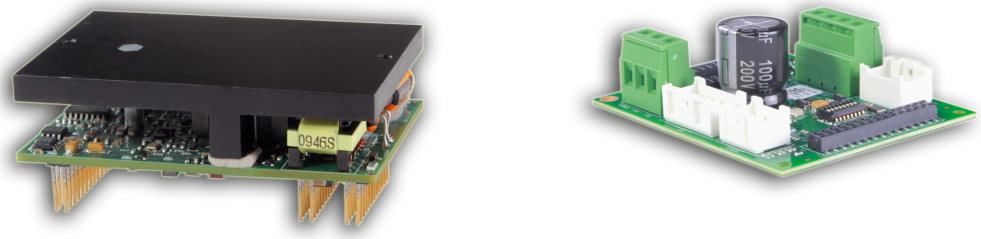
6.5.3 Motor Model Calculations

Experimental Calculation of K_t and K_e

The motor torque constant, K_t, was calculated using the torque current relation $\tau = K_t I$. The leg was modelled as a virtual spring-damper system, as seen in fig. 9.1.

The spring constant, K_{s1}, was set to $200\ [N/m]$, and the damping and torsional spring-damping constants were set to zero. K_t was tuned until the theoretical foot force matched the practical foot force measured via a scale. The leg was fixed at a set height imposing a radial offset on the virtual spring-damper system.

For a spring constant of $200\ [N/m]$ and a radial offset of $0.15\ m$ a theoretical foot force of $K_{s1}\Delta r = 30\ N$ was expected. A mass of approximately $3\ kg$ was measured with



(a) AMC DigiFlex Performance Servo Drive.

(b) AMC DigiFlex Performance Servo Drive mounting card.

Figure 6.7: AMC Servo Drive and Mounting Card.

Figure 6.8: T-Motor U10 Plus Brushless DC Motor.

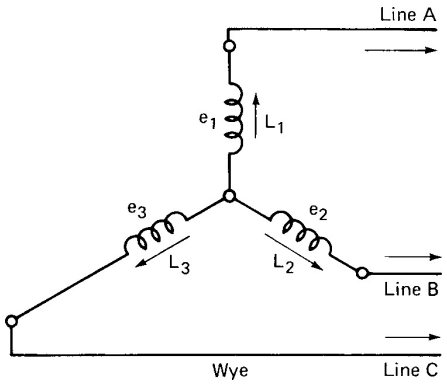


Figure 6.9: WYE connected BLDC motor windings.

$K_t = 0.08\ [Nm/A]$ set in the virtual leg model controller, resulting in a foot force of $3\ kg \times 9.81\ m/s^2 = 29.43\ [N]$.

The study in [**?**], using the same T-Motor U10 Plus motors, calculated a torque constant of $K_t = 0.072\ [Nm/A]$. This confirms the experimental results obtained above.

For an ideal motor at a constant operating point, K_e will equal K_t, as shown in eq. (6.1).

$$
\begin{aligned}
&V_t = K_e\omega_m + IR_m \\
&\tau_m = K_tI \\
&P_{elec.} = V_tI = K_e\omega_mI + I^2R \\
&P_{mech.} = \tau_m\omega_m = K_tI\omega_m \\
&P_{loss.} = I^2R_m \\
&P_{elec.} = P_{mech.} + P_{loss.} \\
&\therefore K_e\ [V/rad/s] = K_t\ [Nm/A] = 0.08
\end{aligned}
\tag{6.1}
$$

Calculation of R_m and L_m

The resistance and inductance of the 3 phase windings of the motor were calculated using a lab multimeter to be $R_m = 47.5\ m\Omega$ and $L_m = 35\ \mu H$ respectively.

Brushless DC motor windings are usually connected in WYE formation, as seen in **??**. This means the measured values for resistance and inductance were line-to-line values and had to be divided by two to get the per phase values above.

Calculation of J_m

In order to calculate the moment of inertia of the motor, J_m, the ratio of acceleration torque to acceleration to steady state needs to be found. By commanding a DC equivalent current input of $1\ A$ and measuring the time taken to reach a steady state velocity, eq. (6.2) can be used to calculate J_m. The velocity vs. time plot used can be seen in fig. 6.10.

$$\begin{aligned} J_m &= \frac{T_{acc.}[N/m]}{a[m/s^2]} \\ &= \frac{IK_t}{a} \\ &= \frac{IK_t}{\frac{\Delta v}{\Delta t}}\ [kg/m^2] \end{aligned} \tag{6.2}$$

where $I = 1\ A$, $K_t = 0.08\ Nm/A$, $\Delta v = 1313.906 \times \frac{2\pi}{60}\ rad/s$ and $\Delta t = 588.889 \times 10^{-3}\ s$.

This results in a motor moment of inertia of $J_m = 3.424 \times 10^{-4}\ [kg/m^2]$.

Calculation of B_m

The motor damping or viscous friction, B_m, was assumed to be negligible. Brushless DC motors have near zero damping and will have little effect on the simulated motor model.

Calculation of τ_e and τ_m

The electrical and mechanical time constants of the motor, τ_e and τ_m respectively, can be used to plot a root-locus plot with poles at $-\tau_e$ and $-\tau_m$ as can be seen in **??**. This is useful when designing a current controller for the system. τ_e and τ_m can be calculated using eq. (6.3).

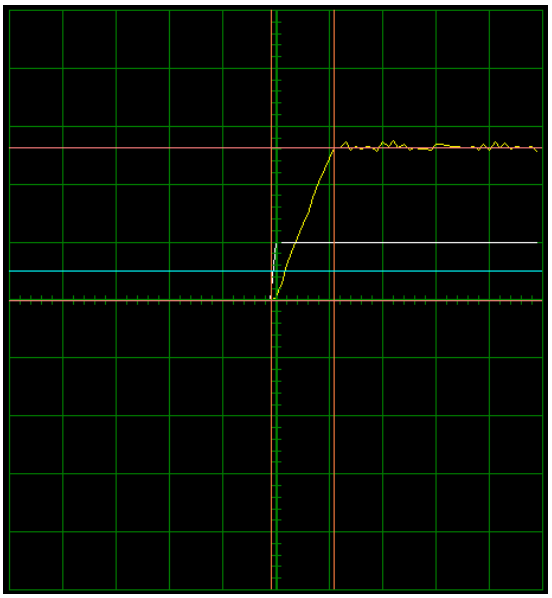


Figure 6.10: Velocity vs. time plot for 1A equivalent DC command.
(500 rpm; 500 ms/div)

$$
\begin{aligned}
K_m &= \frac{1}{B_m} \\
\tau_m &= \frac{J_m}{B_m} \\
K_e &= \frac{1}{R_m} \\
\tau_e &= \frac{L_m}{R_m}
\end{aligned}
\tag{6.3}
$$

From eq. (6.3) and using the previously calculated motor constants, $\tau_e = 7.368 \times 10^{-4}$ and $\tau_m = 3.424 \times 10^{-4}$. This is assuming a motor viscous friction of $B_m = 0$.

The resulting motor model open loop root-locus plot can be seen in fig. 6.11. As expected the system has only negative real roots and will be stable in open loop.

6.5.4 Driver Configuration

The AMC drivers allow extensive customisation. After the motor, encoder, and general communication control parameters are configured, the PID control loops of the drivers can be configured, as seen in fig. 6.12.

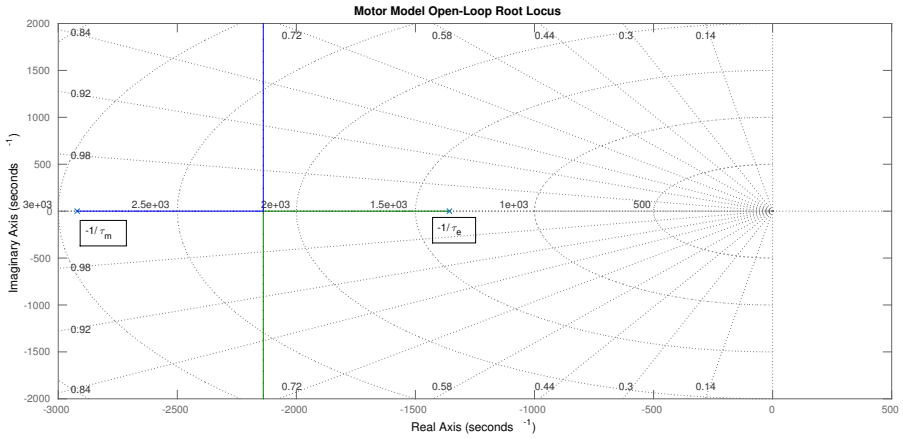


Figure 6.11: Motor model open loop root-locus plot.

The motor drivers were initially configured with both on-board PID current and position control loops enabled. This allowed initial modelling of the motors, configuring of the motor encoders, and determining of the position limits (in counts). For control of the leg, the position control loop was finally implemented on the STM32F4 microcontroller, while using the existing current control loop of the motor drivers.

The AMC drivers were configured using the AMC Driveware configuration software, which provided an oscilloscope to measure the relevant motor responses as seen in figs. 6.10, 6.13 and 6.14.

Current Control Loop

By using a 1 A 120Hz square wave current command the current PI control loop was tuned, as seen in fig. 6.13. Initially both the proportional gain, K_p, and the integral gain, K_i, were set to zero. K_p was slowly increased until the final amplitude of the current output just started to overshoot. K_i was then set to minimize the steady state error.

Values of $K_p = 0.277$ and $K_i = 0.262$ were obtained. Both motors were found to operate optimally with the same PI gain values.

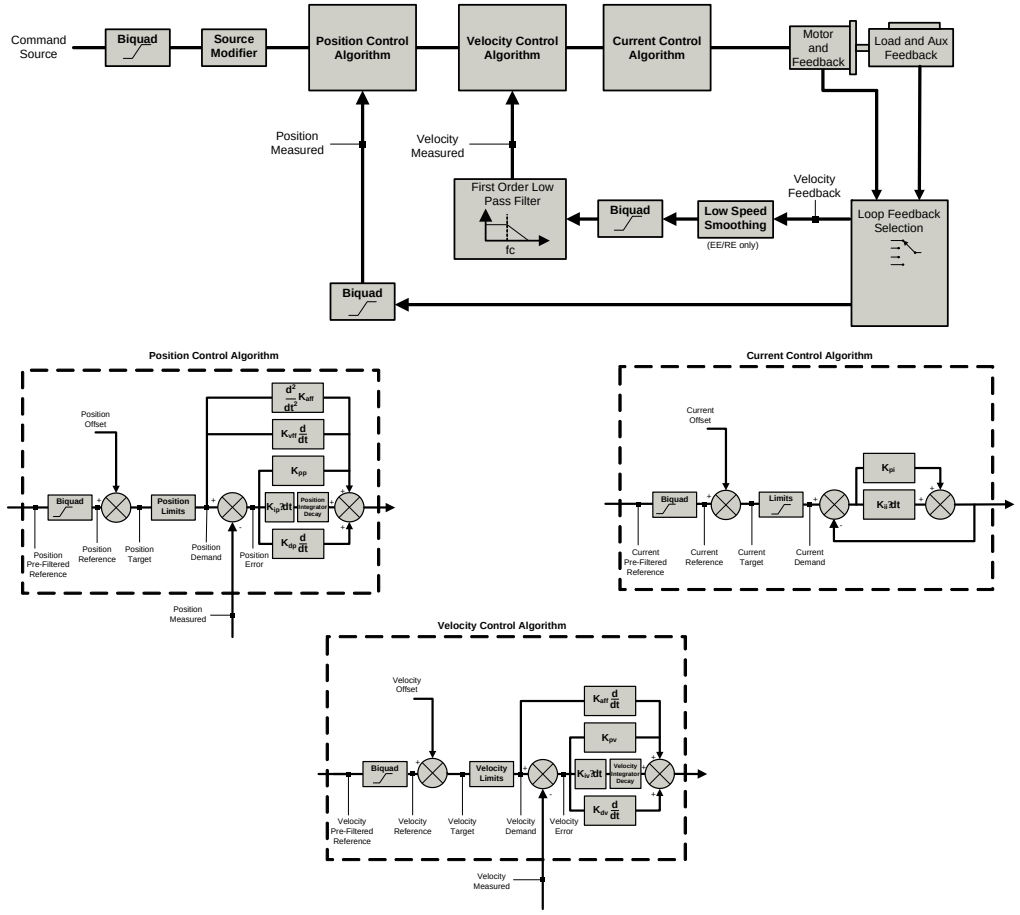
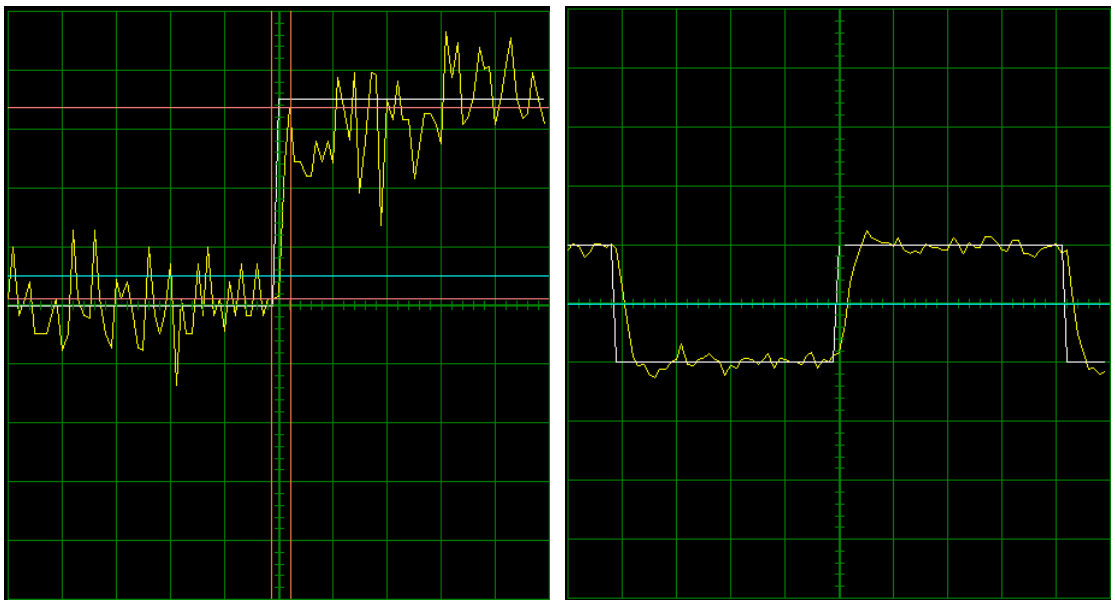


Figure 6.12: AMC DigiFlex Performance Servo Drive control loops (AMC, 2014).



(a) Motor driver current loop pre-tuning. (200 mA; 5 ms/div)

(b) Motor driver current loop tuning - 1A 120Hz square wave command. (1 A; 1 ms/div)

Figure 6.13: Motor driver current loop tuning plots.

Position Control Loop

The on-board AMC motor driver control loop was set up to test the encoder configuration. The encoder and relevant position limits can be seen in subsection 6.5.5.

A 1Hz sinusoid was used to tune the PID control loop gains. Values of $K_p = 0.0005793$, $K_i = 0.0006052$ and $K_d = 2.769e - 9$ were found to achieve optimal set-point tracking as seen in fig. 6.14. The sinusoidal set-point can be seen in white and the position feedback in yellow. A 10-30 ms lag time can be seen due to the inertial load. This lag time causes a dead-band which should be considered when implementing a controller.

These tests were performed with the leg attached - the inertial load provided by the leg made PID control loop tuning possible, whereas without any inertial load the BLDC motors overshot their set-point.

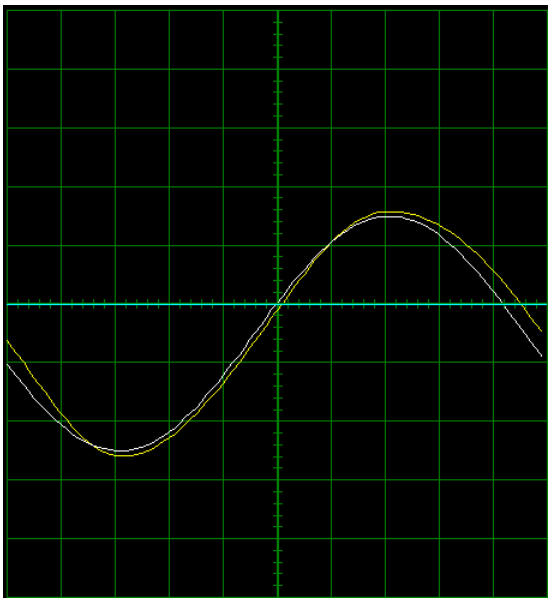


Figure 6.14: Motor driver position loop tuning - (-350:200) count 1Hz sinusoid command with 300 count offset.
(100 ct; 100 ms/div)

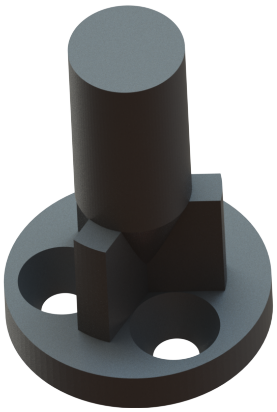


Figure 6.15: 3D printed PLA motor encoder shaft.

6.5.5 Motor Encoders

6.5.6 Tuning and Optimisation

7 Communication Protocol

Included pieces of code that may not be obvious to users or code that was particularly important to the operation of the protocol ...

```
struct __attribute__((__packed__)) RXPacketStruct {
        uint8_t START[2];
        ...
        uint8_t StatBIT_1 : 1; //Bit field
        uint8_t StatBIT_2 : 1;
        uint8_t StatBIT_3 : 1;
        ...
        uint8_t CRCCheck[2]; //CRC-CCITT
        uint8_t STOP[2];
};
```

Listing 1: PC RX "packed" packet structure.

```
union {
        uint32_t WORD;
        uint16_t HALFWORD;
        uint8_t BYTE[4];
} WORDtoBYTE;
```

Listing 2: Byte conversion union.

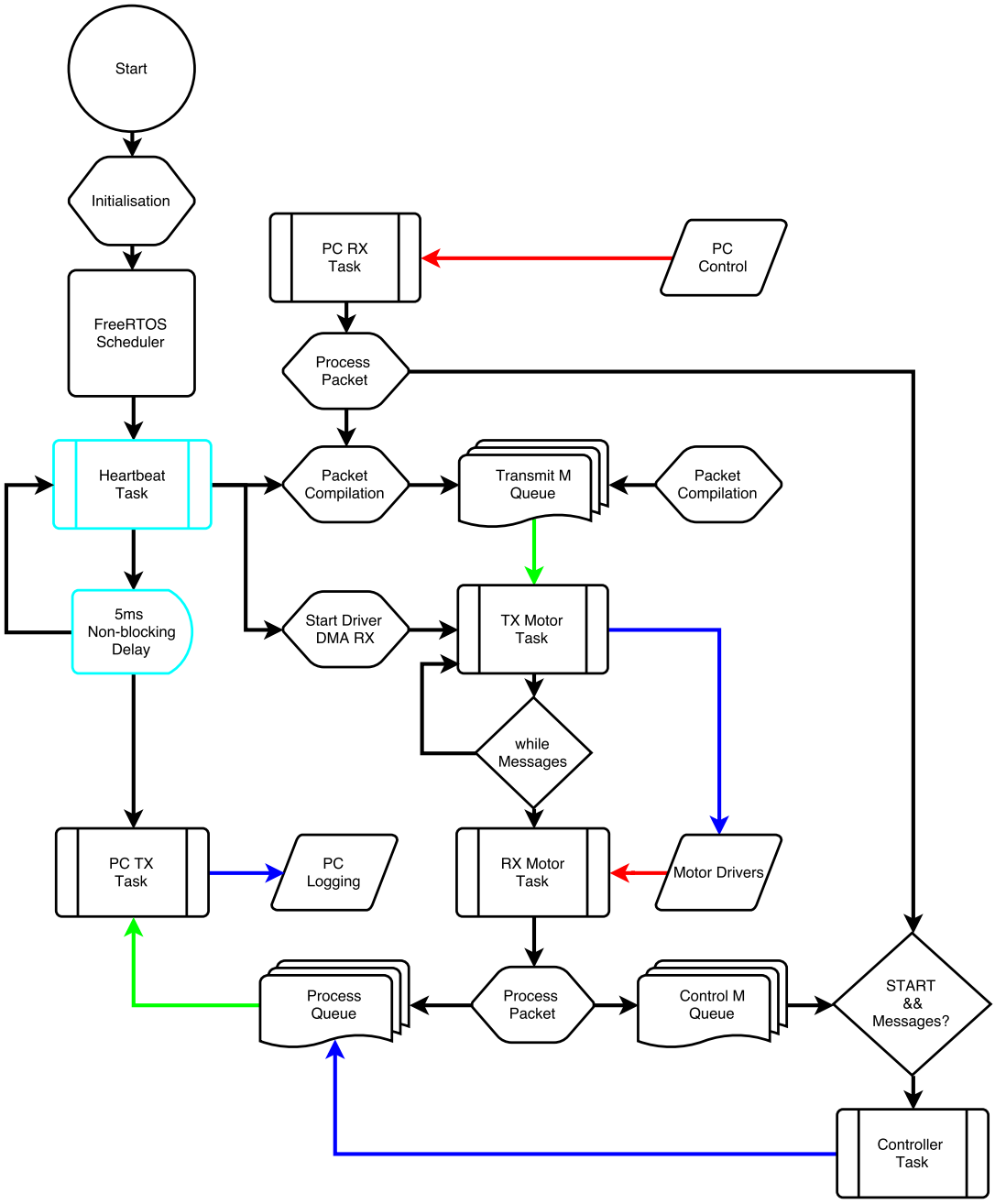


Figure 7.1: FreeRTOS communication protocol flow diagram.

```
HAL_UART_Receive_DMA(&PC_UART, RXBufPC, sizeof(RXPacket));
if(xSemaphoreTake( PCRXHandle, portMAX_DELAY ) == pdTRUE) {
        rcvdCount = sizeof(RXPacket);
        START_INDEX = findBytes(RXBufPC, rcvdCount,
        RXPacket.START, 2, 1);
        if(START_INDEX>=0) {
                memcpy(RXPacketPTR, &RXBufPC[START_INDEX],
                sizeof(RXPacket));
                RX_DATA_VALID = 0;

                WORDtoBYTE.BYTE[1] = RXPacket.CRCCheck[0];
                WORDtoBYTE.BYTE[0] = RXPacket.CRCCheck[1];
                CALC_CRC = crcCalc(&RXPacket.OPCODE, 0, PAYLOAD_RX, 0);

                //A useful tool when calculating and
                //confirming CRC values of various types:
                //https://www.lammertbies.nl/comm/info/crc-
                //calculation.html

                if(WORDtoBYTE.HALFWORD==CALC_CRC) {
                        RX_DATA_VALID = 1;
                        ... //Packet processing
```

Listing 3: PC RX packet processing.

```
void BaseCommandCompile(uint8_t n, uint8_t SeqBits, uint8_t ComBits,
uint8_t INDOFF1, uint8_t INDOFF2, uint8_t *DATA,
uint8_t LEN, uint8_t SNIP);
```

Listing 4: Motor packet compilation function.

```
BaseCommandPTR = &BaseCommand[RXPacket.OPCODE];
BaseCommandCompile(RXPacket.OPCODE, 0b0011, 0x02, 0x45, 0x02,
RXPacket.M1C, 2, 0);
xQueueOverwrite(ICommandM1QHandle, &BaseCommandPTR);
```

Listing 5: Motor packet compilation current command example.

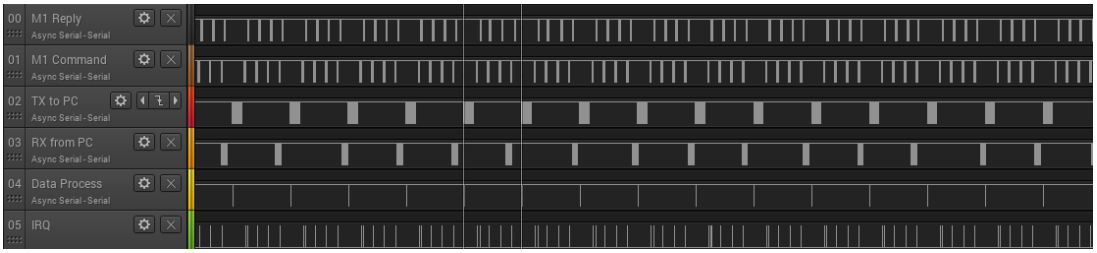


Figure 7.2: Communication protocol packet timing with 5 ms sampling rate.

Command	Index	Op-Code	TX CB	TX CRC1	RX CB
Kill Bridge	1	0001	0x06	0xCBB6	0x04
Write Enable	2	0010	0x0A	0x3624	0x08
Bridge Enable	3	0100	0x12	0x1AE0	0x10
Set Current	4	0011	0x0E	0xBF7B	0x0C
Read Current	5	1100	0x31	0x9772	0x32
Read Position	6	1111	0x3D	0xD310	0x3E
Read Velocity	7	0101	0x15	0x5EAF	0x16
Set Position	8	1010	0x2A	0x42C4	0x28

Table 7.1: Motor driver command protocol.

8 Kinematics

Originally derived in [**?**]...

$$f(\phi_1, \phi_2) = \begin{pmatrix} \sqrt{\frac{9}{100} - \frac{9\sin\left(\frac{\phi_1}{2}+\frac{\phi_2}{2}\right)^2}{400}} - \frac{3\cos\left(\frac{\phi_1}{2}+\frac{\phi_2}{2}\right)}{20} & \frac{\phi_1}{2} - \frac{\phi_2}{2} \end{pmatrix} \tag{8.1}$$

$$g(r, \theta) = \begin{pmatrix} \pi - acos(\frac{r^2+l_1^2-l_2^2}{2rl_1}) + \theta \\ \pi - acos(\frac{r^2+l_1^2-l_2^2}{2rl_1}) - \theta \end{pmatrix} \tag{8.2}$$

Taking the Jacobian of the kinematic mapping $f(\phi_1, \phi_2)$ the foot force vector, F, can be transformed to the motor torque commands, τ:

$$J = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{X}}\right] \tag{8.3}$$

where $\mathbf{X} = [\mathrm{r}\ \theta]$.

The following force vector provides a constant angular force, f_{theta}:

$$F = [f_r\ f_\theta]^T \tag{8.4}$$

by using f_s, a force related to the arc-length of a polar system, the relation $s = r\theta$ exists:

$$F = [f_r\ f_s]^T \tag{8.5}$$

$$f_a = k_s(a_{fbk} - a_{cmd}) + k_d(\dot{a}_{fbk} - \dot{a}_{cmd}) \tag{8.6}$$

$$\tau = J^T F \tag{8.7}$$

9 Dynamic Modelling

9.1 System Modelling

9.1.1 SLIP Model

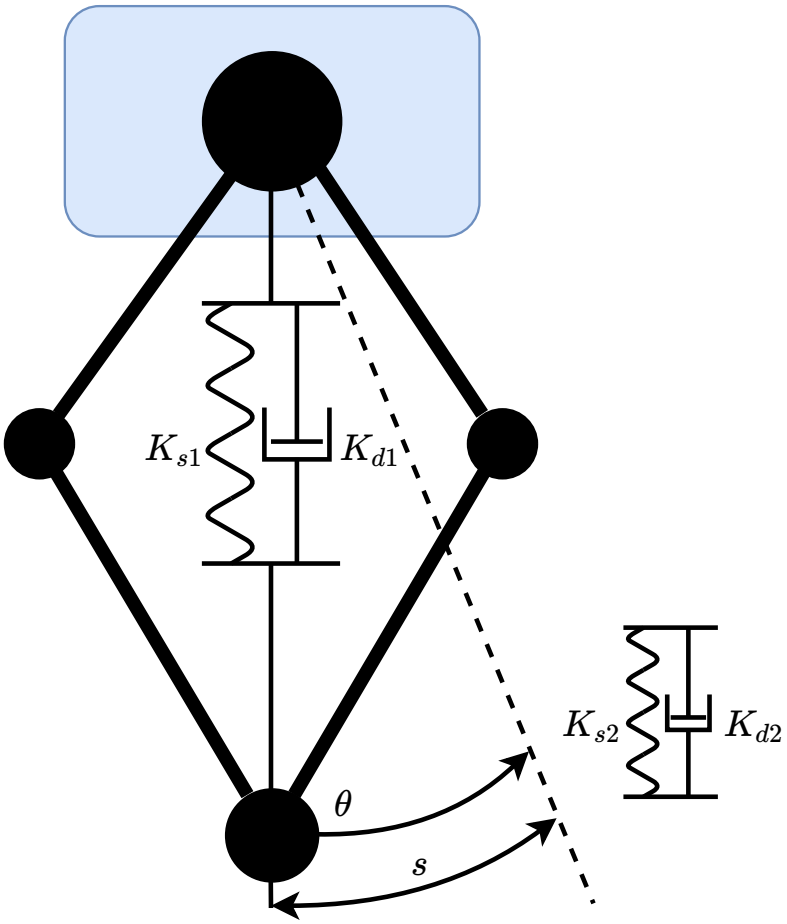


Figure 9.1: Leg spring-damper virtual model.

9.2 Virtual Compliance Model

10 Controller Development

10.1 Dynamic Actuation

11 Experimental Testing

"Jump!"

— Van Halen, *1984*

$r_0 = 0.3\ m$
$r_{offset} = r - r_0 = 0.13\ m$

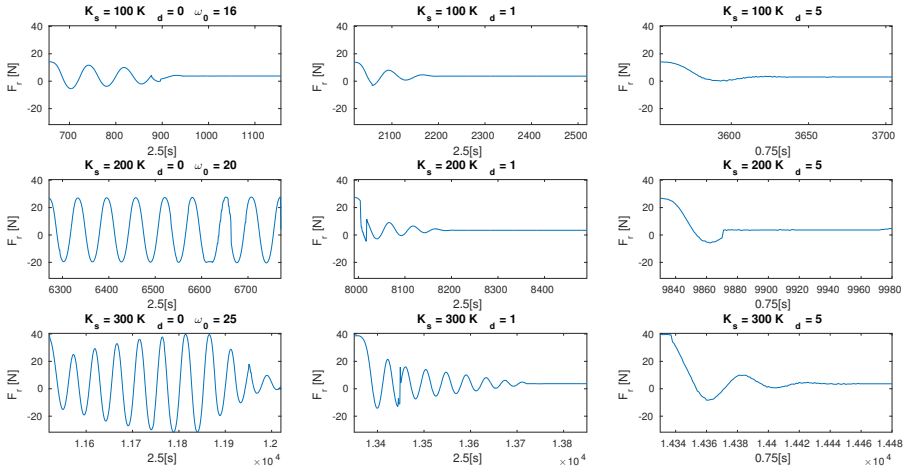


Figure 11.1: Leg spring damper testing for radial offset.

12 Design Validation

13 Conclusions

14 Recommendations and Future Work

A Code