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 31, 2016

Keywords: impedance control, virtual model, 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 31, 2016

Abstract

Acknowledgements

Amir Patel Callen Fisher Craig Burden Gareth Callanan Roberto Aldera

Ben Bingham Luke Bell

Terms of Reference

Description

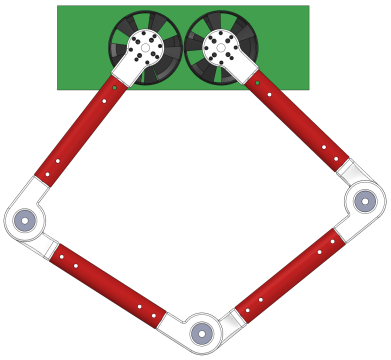


Figure 1: Version 1 of Baleka leg platform (Ben Bingham, 2016).

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

Declaration iii

Abstract iv

Acknowledgements v

Terms of Reference vi

List of Figures xiii

List of Tables xvi

List of Source Codes xvii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Objectives of the Study . . . 1
  - 1.2.1 Problems to be Investigated . . . 1
  - 1.2.2 Research Questions . . . 1
  - 1.2.3 Purpose of the Study . . . 2
- 1.3 Scope and Limitations . . . 2
- 1.4 Plan of Development . . . 3

2 Literature Review 4
- 2.1 Introduction . . . 4
- 2.2 State of the Art . . . 6
  - 2.2.1 Monoped Robots . . . 6
  - 2.2.2 Biped Robots . . . 6
  - 2.2.3 Quadruped Robots . . . 6
  - 2.2.4 Bio-inspired Legged Robotics . . . 6
  - 2.2.5 Humanoid Robots . . . 6
  - 2.2.6 Closed Kinematic Chain Leg . . . 6
- 2.3 Legged Locomotion in Nature . . . 10
- 2.4 Raibert Control . . . 10
- 2.5 Applications in Industry . . . 11
  - 2.5.1 Soft-robotics . . . 11

2.5.2 Bose Active Suspension 11

3 Project Plan and Methodology 13

4 Theory Development 14
4.1 General Co-ordinates 14

5 Geometry 15

6 Kinematics 17
6.1 The Jacobian 17
6.1.1 Velocity Mapping 18

7 Geometric Simulation 19

8 Hardware Design 24
8.1 Original Leg Design 24
8.1.1 Leg Hip 24
8.1.2 Leg Guide 24
8.1.3 Problems with Design 24
8.2 Mechanics and Construction 27
8.2.1 Alumnium Mounting Plate Design 27
8.2.2 Leg Linkage and Foot Design 27
8.2.3 Testing Platform 29
8.3 Mass Distribution 32
8.4 Electronics and Communication 34
8.4.1 Accelerometer and Gyroscope 34
8.4.2 Distance Sensor 34
8.4.3 Microcontroller 35
8.5 Motors and Drivers 36
8.5.1 Driver Selection 36
8.5.2 Motor Selection 36
8.5.3 Motor Model Calculations 37
8.5.4 Driver Configuration 40
8.5.5 Motor Encoders 43

9 Software Design 45
9.1 RTOS Communication Protocol 45
9.1.1 Heartbeat Task 45
9.1.2 PC TX Task 46

9.1.3 PC RX Task . . . 46
9.1.4 TX Motor Task . . . 46
9.1.5 RX Motor Task . . . 46
9.1.6 Controller Task . . . 46
9.1.7 FreeRTOS Timing . . . 46
9.2 Packet Transmission . . . 47
9.2.1 Structuring . . . 47
9.2.2 Integrity Checking . . . 47
9.2.3 Encoding and Decoding . . . 47
9.3 Peripheral Configuration . . . 48
9.3.1 Protocol . . . 48
9.3.2 Data Rates . . . 48
9.3.3 Direct Memory Access . . . 48
9.4 Graphic User Interface . . . 50
9.4.1 Serial Communication . . . 51
9.4.2 Logging . . . 51
9.4.3 Live Plotting . . . 51
9.4.4 Control Plug-in . . . 51

10 Dynamic Modelling 53
10.1 Robotic Leg Modelling . . . 53
10.1.1 Virtual Model . . . 53
10.1.2 Dynamic Model . . . 53
10.2 Spring-damper Mass Motion . . . 56
10.3 Leg Spring-damper Model . . . 59
10.3.1 Impact Energy . . . 59
10.3.2 Launch Energy . . . 60
10.4 Virtual Compliance Model . . . 60

11 Controller Development 61
11.1 Active Compliance . . . 61
11.2 Dynamic Stability . . . 61
11.3 Mechanical Impedance . . . 62
11.4 Control Loop Sampling Frequency . . . 62
11.5 Force Control . . . 63
11.5.1 Current Control . . . 63
11.6 Control Loop Design . . . 66
11.6.1 Current Control for Impulse Launch . . . 66

11.6.2 Torsional Spring-damper 66
11.6.3 Full-leg vs. Joint Active Compliance 66

12 Experimental Testing 68
12.1 Virtual Spring-damper Tests 68
12.2 Drop Tests 72
12.3 Launch Tests 73
12.4 Current Tracking 78

13 Design Validation 80

14 Conclusions 81

15 Recommendations and Future Work 82

A Code 83

List of Figures

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

2.1 Humanoid robots in popular culture. 5
2.2 Monoped robots. 7
2.3 3D Biped - MIT Leg Laboratory (1989-1995). 7
2.4 Quadruped robots. 8
2.5 MIT Cheetah V1 & V2. 8
2.6 Bio-inspired legged robots. 9
2.7 Atlas Humanoid Robot - Boston Dynamics (2013). 9
2.8 Closed Kinematic Chain Leg using Raibert's Scissor Algorithm (Duperret, Koditschek, 2016).[**?**] 10
2.9 Legged Robots That Balance cover page and exert.[**?**] 11
2.10 Compliant soft robotic handling (Forbes, 2016). 12
2.11 Bose Active Suspension (Bose Corporation, 1980s)[**?**]. 12

5.1 Geometric view of leg. 16

7.1 Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 5cm\ l_2 = 30cm$. 20
7.2 Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 15cm\ l_2 = 30cm$. 21
7.3 Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm\ l_2 = 30cm$. 22
7.4 Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm\ l_2 = 15cm$. 23

8.1 Original 'hip' design by Ben Bingham, 2016. 25
8.2 Final leg design mounted to platform and linear guide: front. 26
8.3 Final leg design mounted to platform and linear guide: back. 27
8.4 Leg mounting plate iterations. 28
8.5 Motor driver interface mounting plate. 28
8.6 Leg foot lateral slipping. 29
8.7 igus DryLin T - Low-profile linear guide. 30
8.8 Linear guide mounted leg model (CAD Solidworks assembly). 31
8.9 Mass distribution of leg assembly. 33

8.10 AMC Servo Drive and Mounting Card. 36
8.11 T-Motor U10 Plus Brushless DC Motor. 36
8.12 WYE connected BLDC motor windings. 38
8.13 Velocity vs. time plot for 1A equivalent DC command. (500 rpm; 500 ms/div) 39
8.14 Motor model open loop root-locus plot. 40
8.15 AMC DigiFlex Performance Servo Drive control loops (AMC, 2014). 41
8.16 Motor driver current loop tuning plots. 42
8.17 Motor driver position loop tuning - (-350:200) count 1Hz sinusoid command with 300 count offset. (100 ct; 100 ms/div) 43
8.18 3D printed PLA motor encoder shaft. 44

9.1 PC TX packet structure. 47
9.2 PC RX packet structure. 47
9.3 STM32F4 microcontroller peripheral configuration. 48
9.4 FreeRTOS communication protocol flow diagram. 49
9.5 Communication protocol packet timing with 5 ms sampling rate. 50
9.6 Control interface plug-in. 52

10.1 Leg spring-damper virtual model. 54
10.2 Joint spring-damper virtual model. 55
10.3 Spring-damper mass model. 58

11.1 Polar co-ordinate spring force mapping to motor torque. 64
11.2 Polar co-ordinate spring force mapping to motor torque. 65
11.3 Virtual model impedance control loop. 67
11.4 Rotational foot force comparison using angle and arc-length torsional spring virtual model. 67

12.1 Full-leg spring damper testing for radial offset. 69
12.2 Joint spring damper testing for radial offset. 69
12.3 Full-leg spring damper motor control. 70
12.4 Joint spring damper motor control. 71
12.5 Leg spring damper drop testing. 73
12.6 Jump foot radial position and velocity (launch and compliant landing). . . 74
12.7 Jump foot force output (launch and compliant landing). 75
12.8 Jump motor current feedback (launch and compliant landing). 76
12.9 Leg launch with compliant landing. 77
12.10Current control tracking. 79

List of Tables

8.1 Solidworks leg assembly mass distribution. 34

9.1 Motor driver command protocol. 50

List of Source Codes

1 FreeRTOS timing configuration. 83
2 PC RX "packed" packet structure. 83
3 Byte conversion union. 83
4 PC RX packet processing. 84
5 Motor packet compilation function. 84
6 Motor packet compilation current command example. 84

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

Baleka is one small step in a greater project - the UCT Mechatronics Lab Cheetah project. The leg linkage system was initially designed and built in the mechatronics lab. After completion of the undergraduate thesis project the robotic leg will be further developed by postgraduate researchers in the laboratory, using the same geometric leg configuration.

Baleka will be used to investigate modelling and control of rapid accelerations, with a preliminary controller developed to perform stable consecutive hopping with the robot.

1.2 Objectives of the Study

1.2.1 Problems to be Investigated

- High speed embedded system communication and packet processing.
- Virtual model control for accurate end effector force output.
- Effective high speed kinematic control.
- Effective use of motor drivers to achieve rapid accelerations with a direct drive BLDC motor.

1.2.2 Research Questions

- Without an accurate dynamic model of a robotic platform is effective force control using a virtual model achievable?

- Is high fidelity foot force control using a simple Jacobian kinematic force mapping achievable?
- What control sampling frequency is necessary to achieve high fidelity force control?
- Can a compliance model effectively absorb high speed impacts and effect rapid acceleration jumps?

1.2.3 Purpose of the Study

The purpose of this study is to develop a robotic leg platform and testing rig capable of rapid acceleration and high fidelity force control experimentation.

1.3 Scope and Limitations

The scope of this project covers:

- Development of a mechanical robotic leg platform using available resources.
- Development of a control system capable of high fidelity force control for rapid acceleration hopping.
- Development of a GUI for robot configuration, data logging and live plotting.
- Development of basic robotic models.
- Effective configuration of motor drivers based on motor model.

The limitations of this project include:

- Motor drivers with communication speeds that limit the control sampling frequency to $200\ Hz$.
- Motor drivers that are limited to $60A$ peak output current.
- Limited budget for extended mechanical development.

1.4 Plan of Development

This study will initially investigate the current state of the art based on pre-existing legged robotic platforms and control techniques applicable to this project.

The project plan and methodology followed by theory development will develop an outline of the structure of the investigation and scientific method.

System modelling and simulation will be performed on a higher level covering motor dynamics, current control and basic kinematics.

The mechanical and electrical construction and design of the leg will be covered extensively followed by the development of the communication protocol and graphic user interface.

The leg kinematics and dynamic model will be investigated before the controller development is considered.

Experimental testing will be performed in the following section which will include spring-damper tests, drop tests and hopping tests.

The final design and implementation of the robot will be critically validated before the study ends with conclusions and recommendation for future work based on the initial objectives of the study.

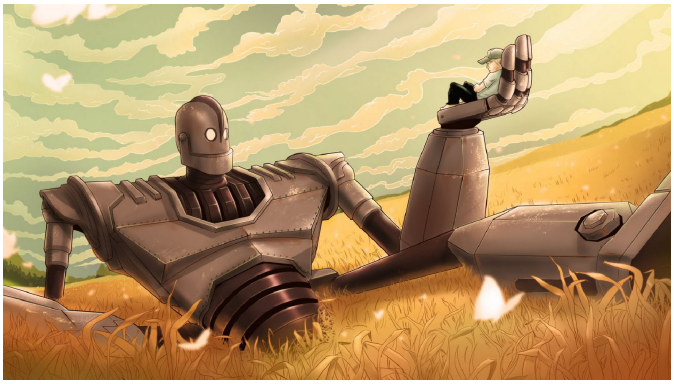
In the appendix code listings and other miscellaneous material will be included for extended study.

2 Literature Review

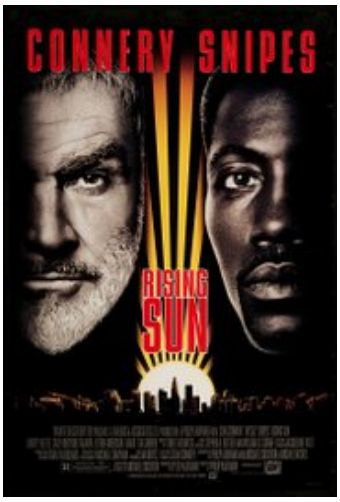
2.1 Introduction

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

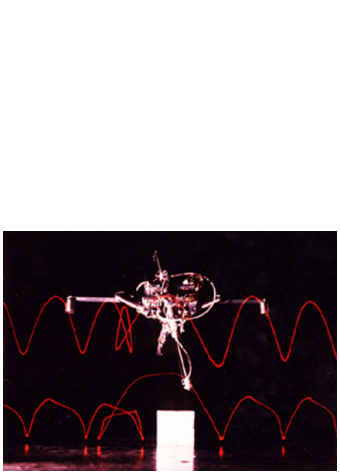
2.2.2 Biped Robots

2.2.3 Quadruped Robots

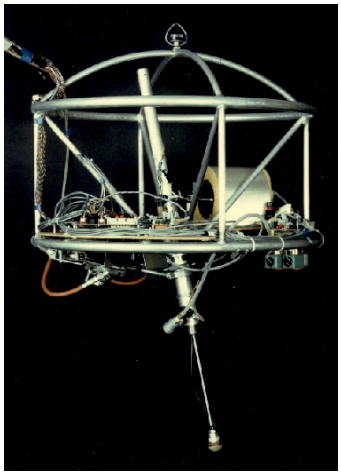
2.2.4 Bio-inspired Legged Robotics

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.

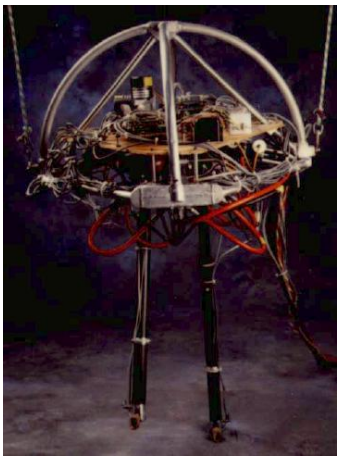
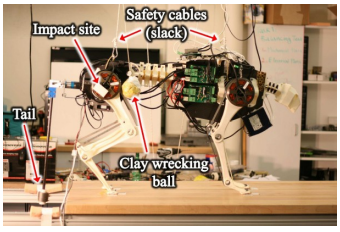


Figure 2.3: 3D Biped - MIT Leg Laboratory (1989-1995).

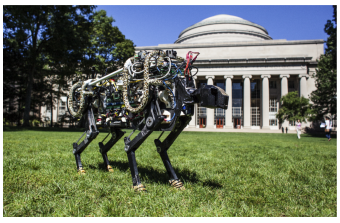
(a) Quadruped - MIT Leg Laboratory (1984-1987).

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

Figure 2.4: Quadruped robots.

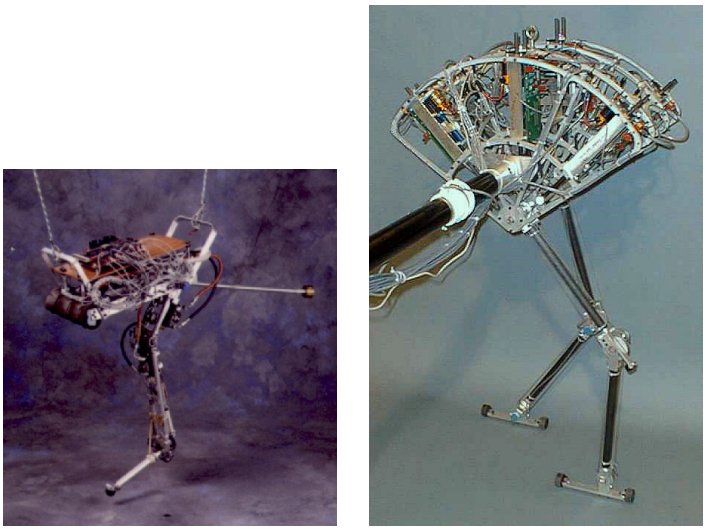


(a) Cheetah Robot - MIT Biomimetics (2012).



(b) Cheetah Robot V2 - MIT Biomimetics (2015).

Figure 2.5: MIT Cheetah V1 & V2.



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

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

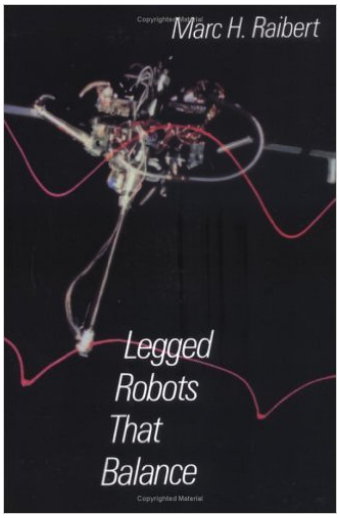
Figure 2.6: Bio-inspired legged robots.

Figure 2.7: Atlas Humanoid Robot - Boston Dynamics (2013).

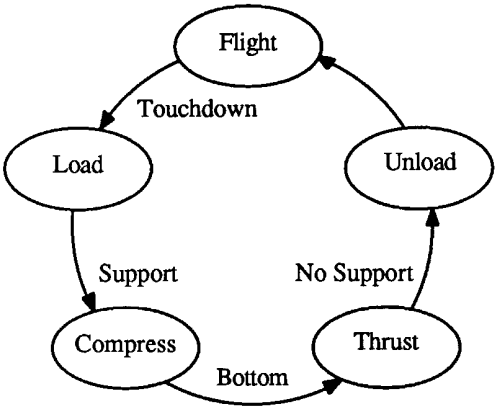
Figure 2.8: Closed Kinematic Chain Leg using Raibert's Scissor Algorithm (Duperret, Koditschek, 2016).[?]

2.3 Legged Locomotion in Nature

2.4 Raibert Control



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



(b) Raibert control state machine.

Figure 2.9: 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

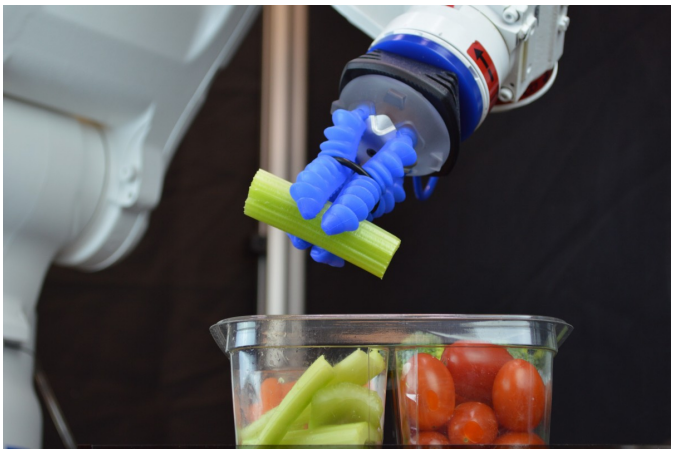


Figure 2.10: Compliant soft robotic handling (Forbes, 2016).

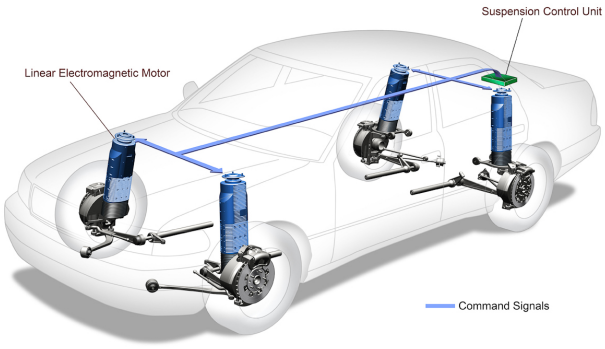


Figure 2.11: Bose Active Suspension (Bose Corporation, 1980s)[?].

3 Project Plan and Methodology

4 Theory Development

4.1 General Co-ordinates

5 Geometry

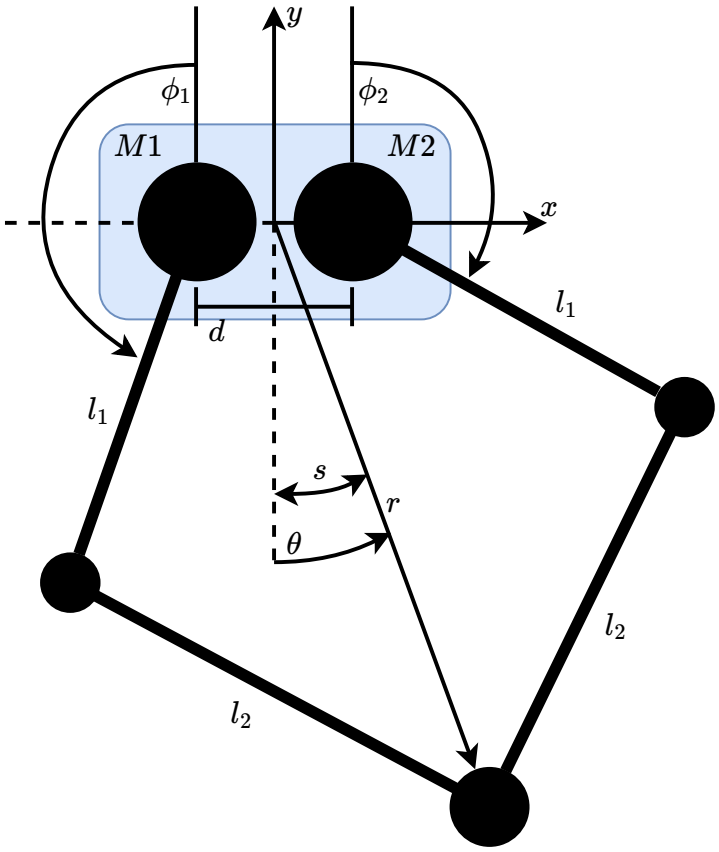


Figure 5.1: Geometric view of leg.

6 Kinematics

The 5-bar linkage design of the leg was first designed constructed in the study [**?**]. The geometry of the leg is fairly complex and the derivation of the kinematic equations equally so. J.M. Duperret and D.E. Koditschek derived the kinematic equations eq. (6.1) and **??** in the study [**?**].

In this study the assumption was made that the distance d, as seen in fig. 5.1, is zero. This simplifies the derivation of forward and reverse kinematic equations of the leg design by making the leg a 4-bar linkage. These kinematic equations are more easily calculated on board a microcontroller[**?**], leaving more processing power for other control tasks if needed.

The ease of calculation makes the loss in accuracy acceptable - in practise the simplified kinematic equations worked well with an insignificant calculation time made possible by the STM32F4's on-board floating point unit.

$$f(\phi_1, \phi_2) = \begin{pmatrix} \sqrt{{\mathrm{l_2}}^2 - {\mathrm{l_1}}^2 \sin\left(\frac{\phi_1}{2} + \frac{\phi_2}{2}\right)^2} - \mathrm{l_1}\, \cos\left(\frac{\phi_1}{2} + \frac{\phi_2}{2}\right) \\ \frac{\phi_1}{2} - \frac{\phi_2}{2} \end{pmatrix} \tag{6.1}$$

$$g(r, \theta) = \begin{pmatrix} \pi - acos(\frac{r^2 + l_1^2 - l_2^2}{2rl_1}) + \theta \\ \pi - acos(\frac{r^2 + l_1^2 - l_2^2}{2rl_1}) - \theta \end{pmatrix} \tag{6.2}$$

6.1 The Jacobian

The Jacobian is formed by taking partial derivatives of the forward kinematic equation eq. (6.1) as shown in eq. (6.3).

It is used as a mapping from the joint angles ϕ_1 and ϕ_1 to the end effector generalized coordinates r and θ. The Jacobian can be applied in robotic kinematic control to determine joint velocities and forces to achieve a desired force or velocity at the end effector, in this case the leg foot.

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:

$$J = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{X}}\right] \tag{6.3}$$

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

6.1.1 Velocity Mapping

Velocity mapping from motor rotational velocity to polar velocity.

$$v(\dot{r}, \dot{\theta}) = Jw(\dot{\phi_1}, \dot{\phi_2}) \tag{6.4}$$

7 Geometric Simulation

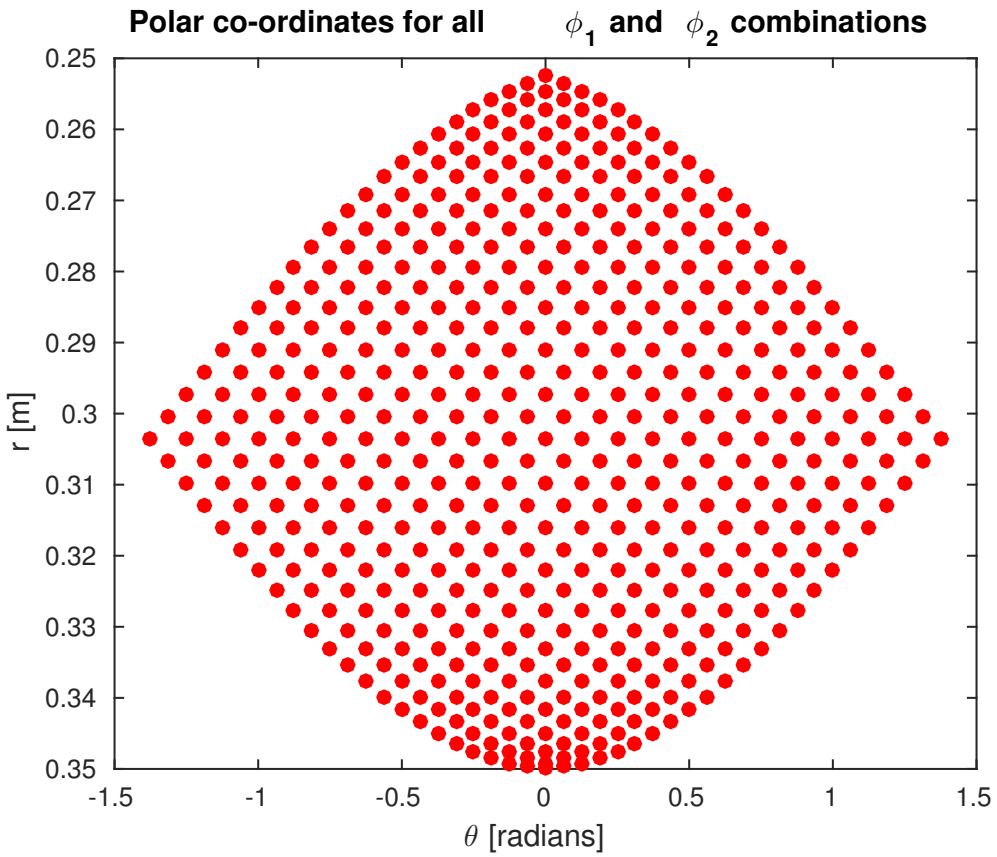


Figure 7.1: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 5cm$ $l_2 = 30cm$.

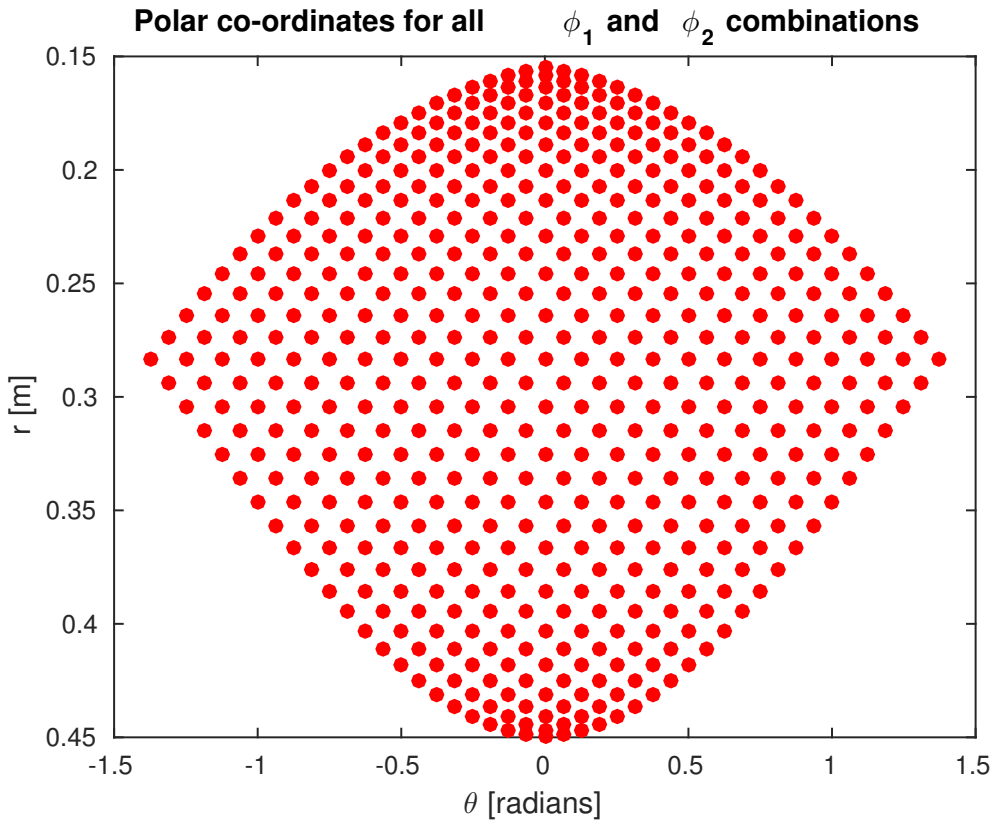


Figure 7.2: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 15cm$ $l_2 = 30cm$.

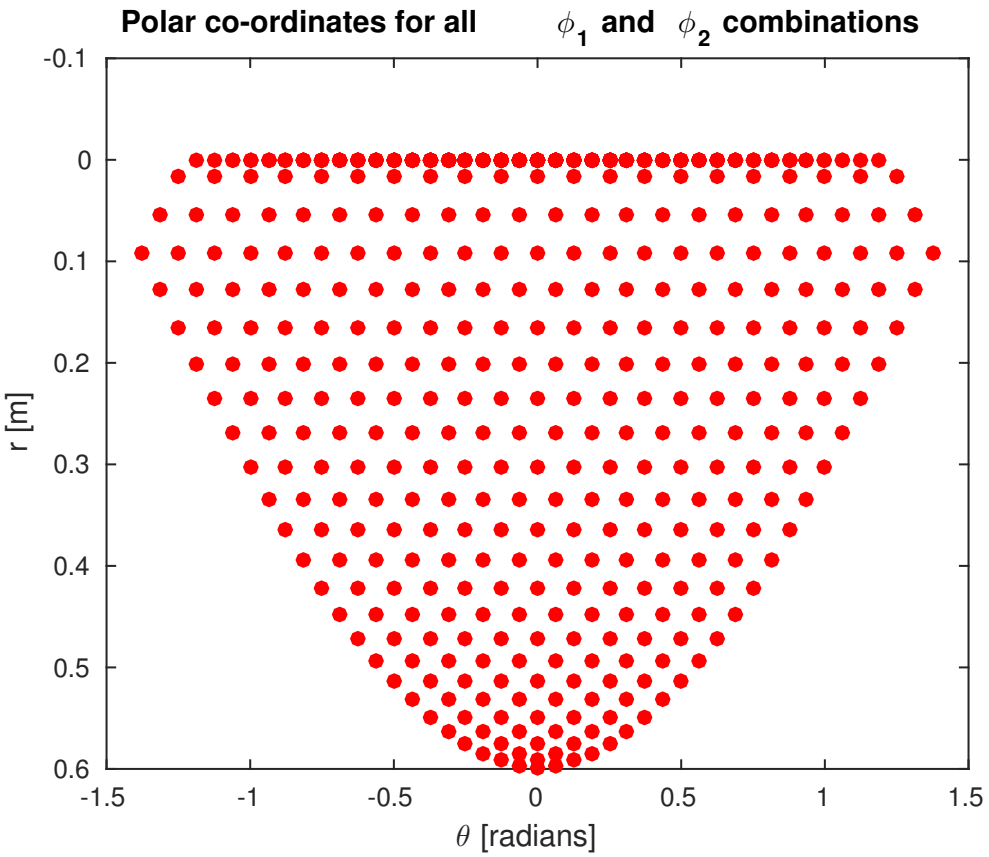


Figure 7.3: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm$ $l_2 = 30cm$.

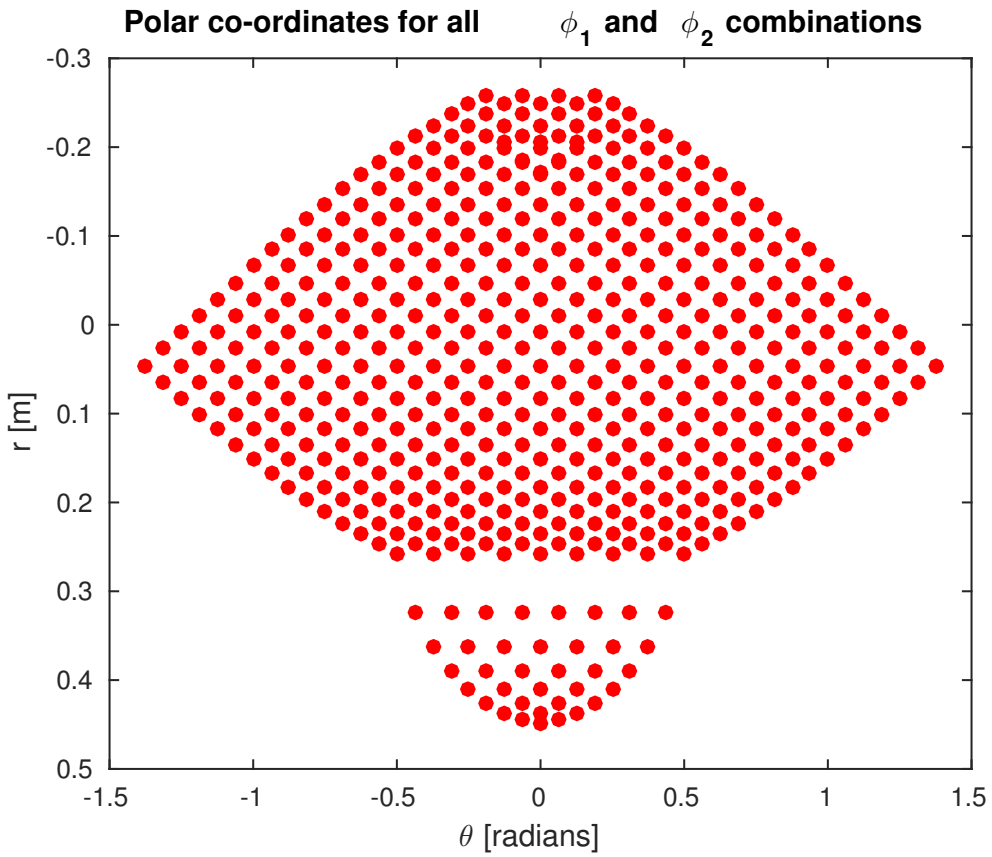


Figure 7.4: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm$ $l_2 = 15cm$.

8 Hardware Design

8.1 Original Leg Design

8.1.1 Leg Hip

The original leg 'hip' was designed by Ben Bingham in 2016 in completion of his undergraduate vacation work as seen in fig. 8.1.

The 'hip' was constructed of 6 mm perspex sheet in a box design with metal L connectors to join the sheets securely. The design of the 'hip' allowed the motor drivers as well as the microcontroller to be mounted on one leg, with space provided for an extra leg for future two-legged movement.

8.1.2 Leg Guide

The guiding system consisted of two parallel steel rods with ball bearings mounted on the 'hip'.

8.1.3 Problems with Design

The leg mounting plate went through three design iterations after the original 'hip' design before the final design was created, as seen in fig. 8.4c.

The original 'hip' design had the following mechanical design flaws:

1. The 6 mm perspex box construction with on-board microncontroller and motor drivers is too heavy for efficient jumping action when compared to similar designs like [**?**] (1.3 kg), [**?**] (2.5 kg), [**?**] (4.2 kg) where there is a high leg torque to mass ratio.
2. The ball bearings are particularly heavy.
3. The mounting of the leg guide places a significant torque in all three cartesian coordinates being off-center from the center of mass.

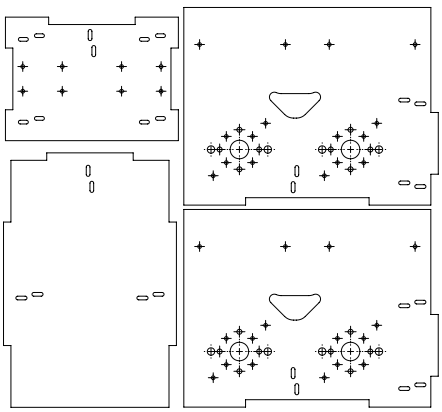


Figure 8.1: Original 'hip' design by Ben Bingham, 2016.

4. The design of the leg guide requires the two steel rods to be perfectly parallel to remove resistance to movement, which is difficult to achieve practically.

These problems were accounted for by replacing the original 'hip' with a rigid aluminium mounting plate with off-board microcontroller and motor drivers. The leg guide consisting of parallel steel rods and ball bearings was replaced with a linear guide as seen in fig. 8.7.

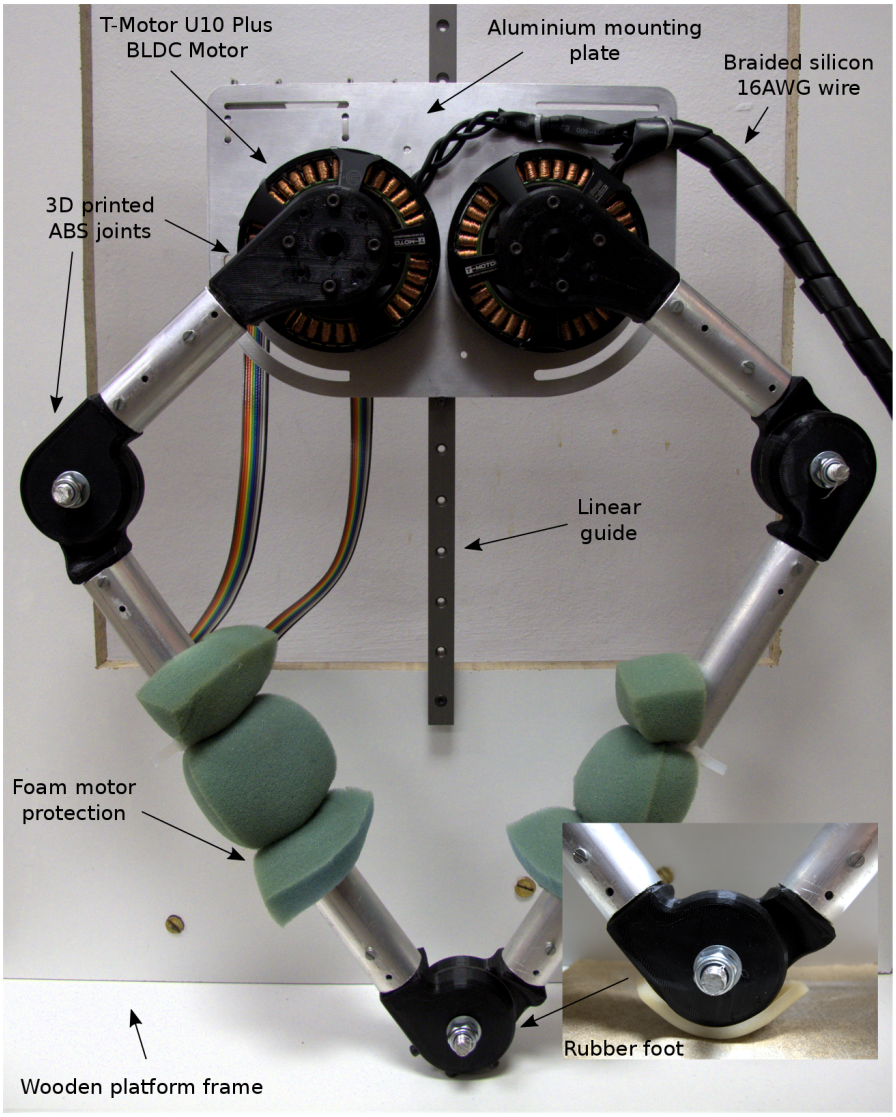


Figure 8.2: Final leg design mounted to platform and linear guide: front.

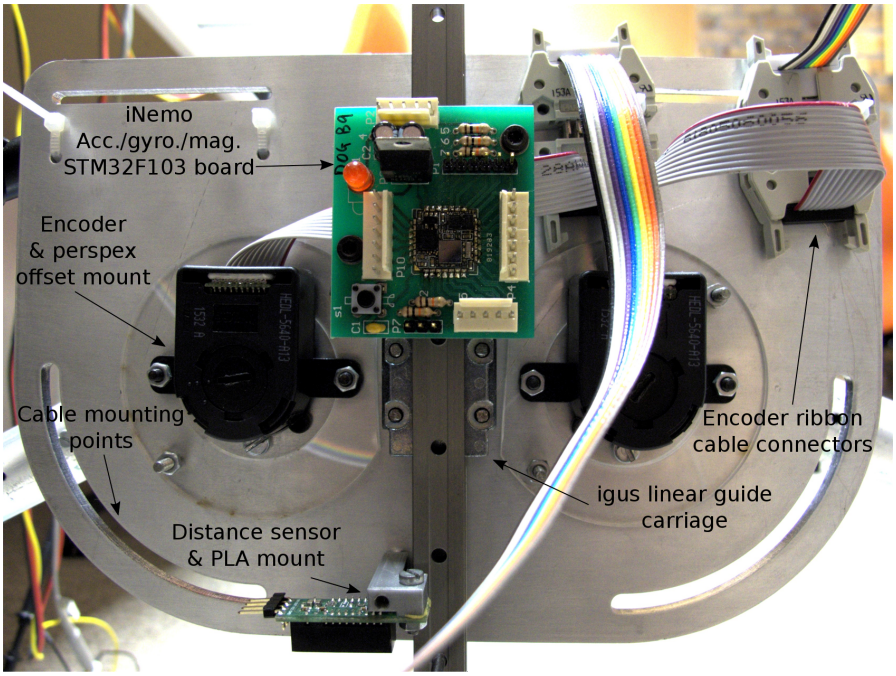


Figure 8.3: Final leg design mounted to platform and linear guide: back.

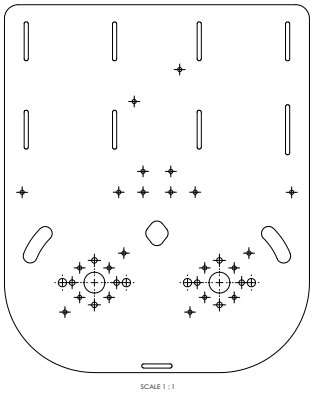
8.2 Mechanics and Construction

8.2.1 Alumnium Mounting Plate Design

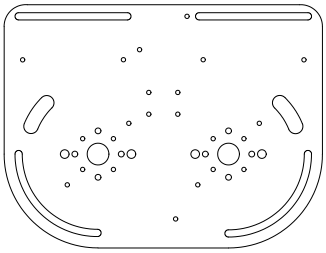
8.2.2 Leg Linkage and Foot Design

The 4-bar linkage design of the leg was originally constructed by Ben Bingham in 2016 for completion of his undergraduate engineering vacation work in the mechatronics lab. The leg was constructed as follows:

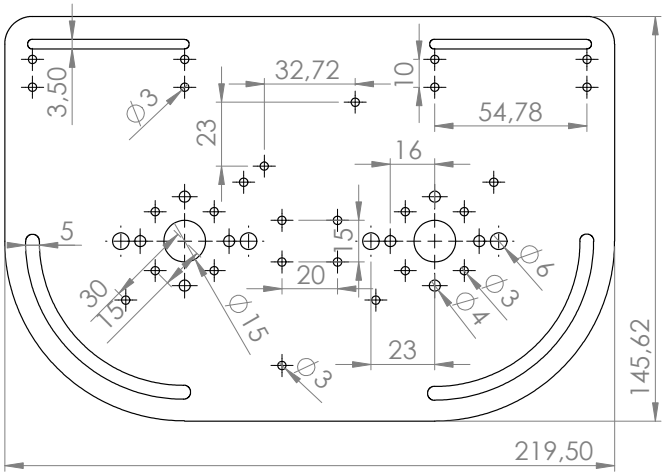
- Three sets of rotational joints make up the linkage system. The joints were 3D printed using ABS plastic and used 3 mm screws to connect to the aluminium leg sections.
- 8 mm loctite nut, bolt and washer combinations connected the joint components with perspex discs to reduce friction between the joints.
- The aluminium leg sections were constructed of 25 mm diameter tubing to form a leg of 0.15 m and 0.3 m sections including the joints.



(a) CAD mounting plate V1.



(b) CAD mounting plate V3.1.3.



(c) CAD mounting plate final design.

Figure 8.4: Leg mounting plate iterations.

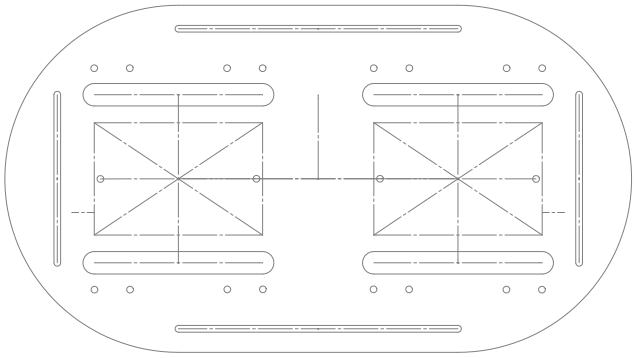


Figure 8.5: Motor driver interface mounting plate.

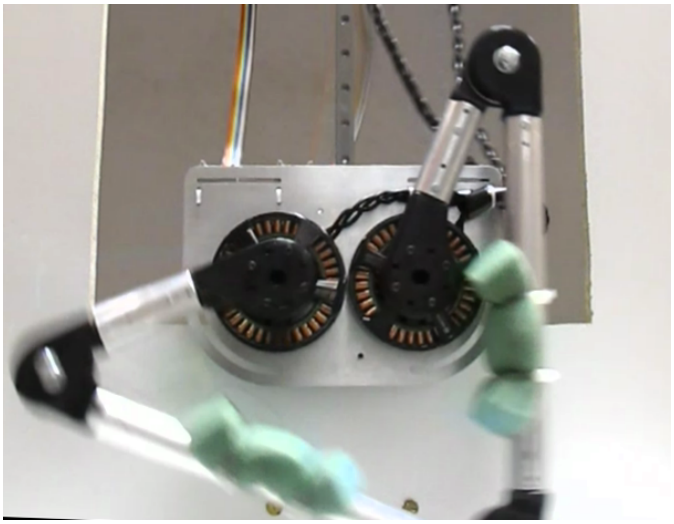


Figure 8.6: Leg foot lateral slipping.

8.2.3 Testing Platform

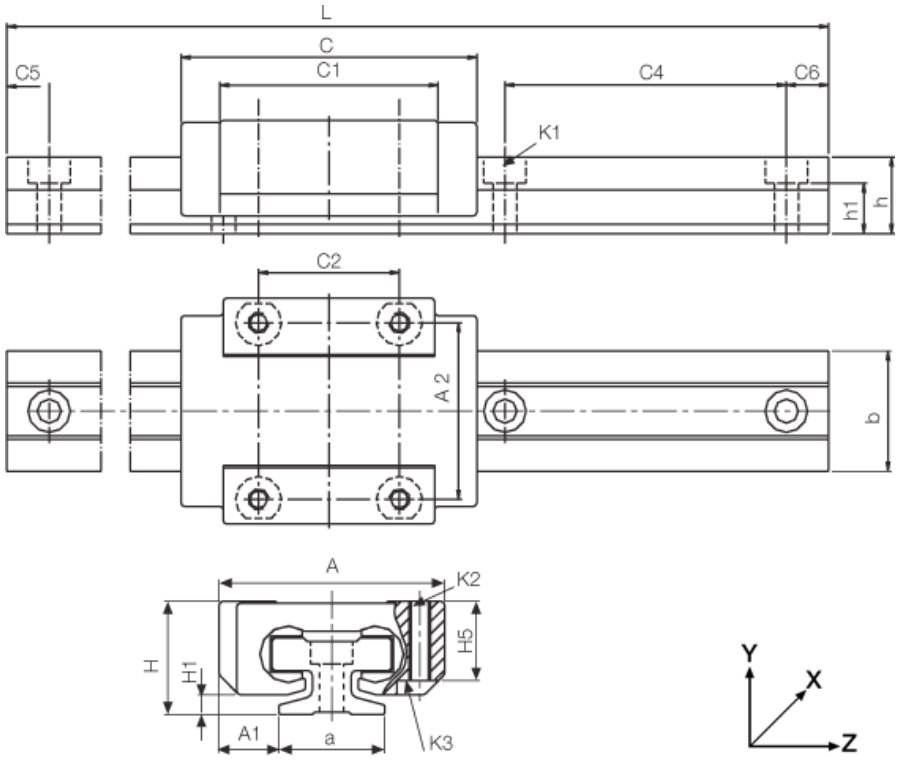


Figure 8.7: igus DryLin T - Low-profile linear guide.

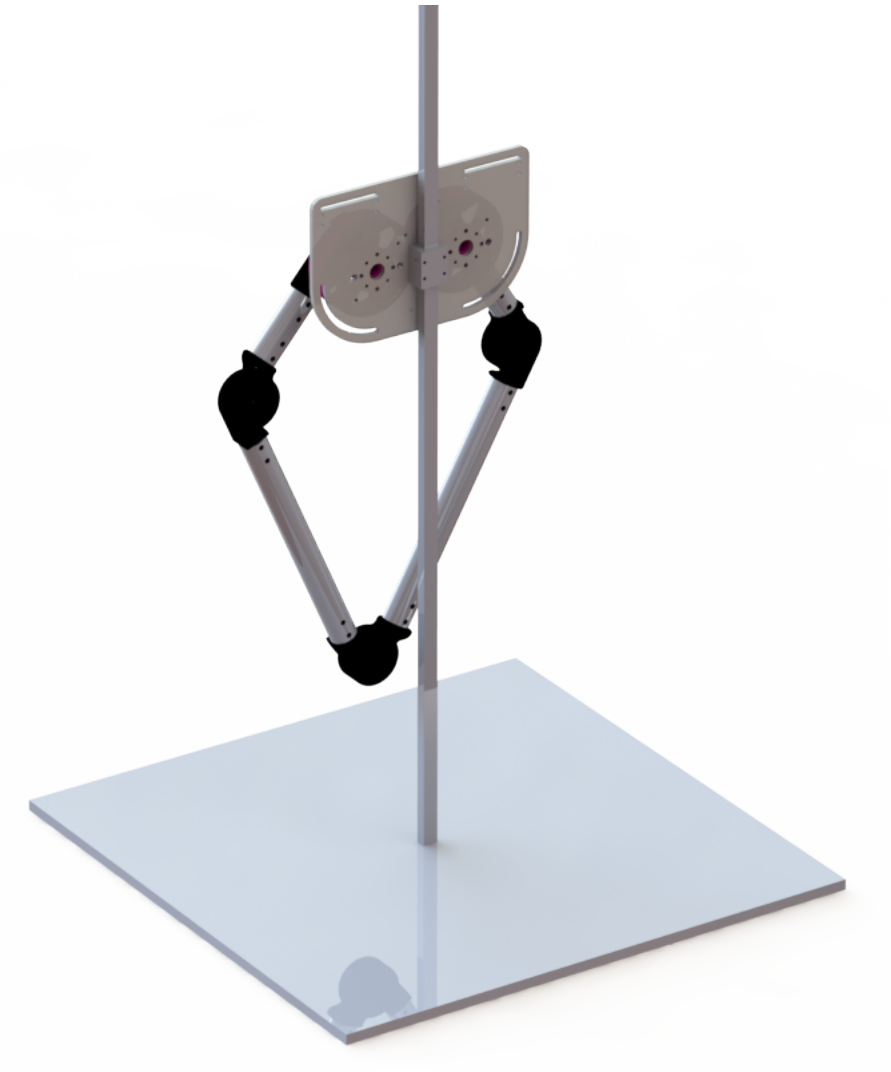


Figure 8.8: Linear guide mounted leg model (CAD Solidworks assembly).

8.3 Mass Distribution

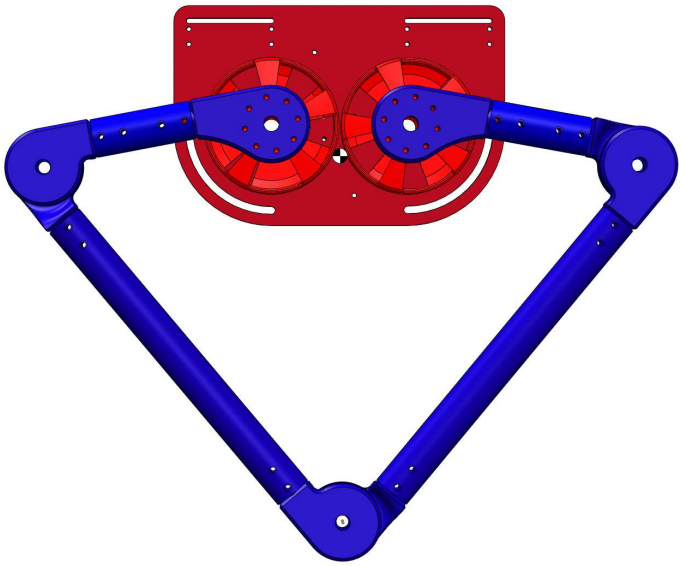
Servo drive mounting card = 50.7g Servo drive = 123.9g T-Motor U10 Plus = 500g

Savings = 349.2g

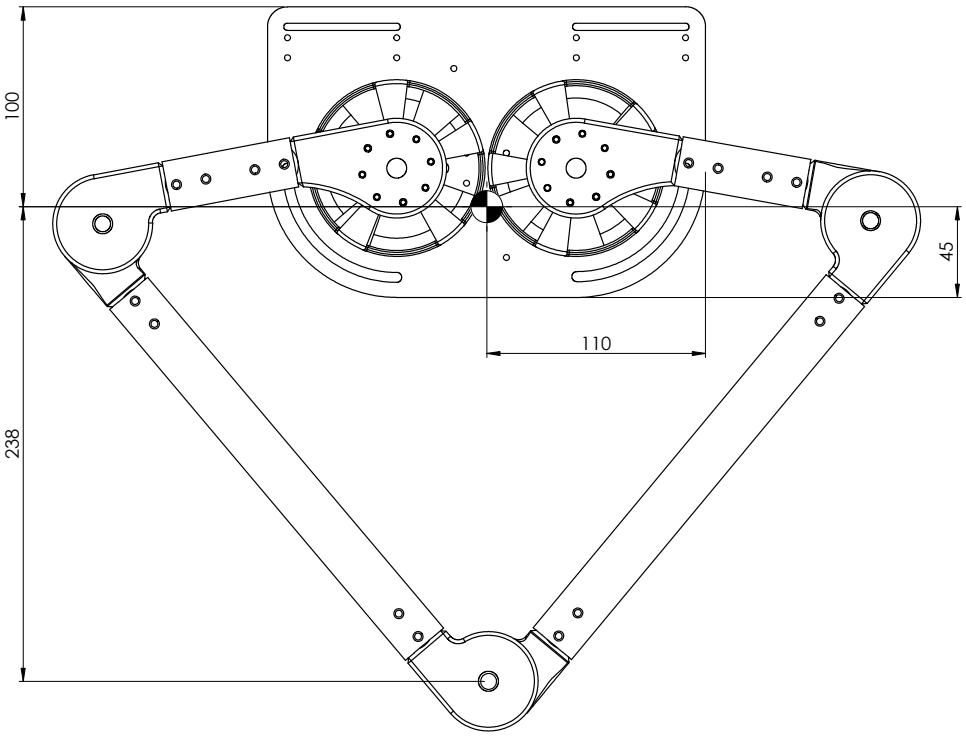
Complete leg = 2.2kg Leg = 0.5kg Plate = 0.7kg Motors = 1kg

The center of mass (COM) moves negligibly with the foot position. To calculate a COM for placement of the linear guide mount and iNemo, a intermediate radial foot position of $0.25m$ was chosen - this position set-point was used for compliant landing and launch sequences. ...

...



(a) Mass distribution graphic.



(b) Center of mass of leg assembly.

Figure 8.9: Mass distribution of leg assembly.

Colour legend	Component	No.	Mass (g)
	T-Motor U10 Plus	2	424.56
	Mounting Plate	1	378.86
	Linear Guide Carriage	1	100.37
	ABS Motor Joint	2	50.55
	Long Linkage	2	46.40
	Joint	5	39.83
	Foot Joint	1	39.63
	Short Linkage	2	12.81
	Washer Bearing	3	2.41
Total:			**1793.88**

Table 8.1: Solidworks leg assembly mass distribution.

8.4 Electronics and Communication

8.4.1 Accelerometer and Gyroscope

8.4.2 Distance Sensor

A distance sensor was mounted to the base of the mounting plate, as seen in fig. 8.3. This provided feedback of the height of the leg's center of mass above the ground.

The leg height was used for height control as well as flight phase determination.

An infra-red distance sensor was chosen with a narrow beam width - this ensures there is minimal reflection off surrounding objects that could interfere with distance readings. The beam reflects off the surface of the ground and a time-of-flight calculation is used to determine distance. Infra-red is open to possible interference from surrounding fluorescent light sources, and this can be accounted for by using it in an area out of direct line of site of light sources. Infra-red was chosen because it is cheaper than an equivalent laser distance sensor.

The Pololu Carrier with Sharp GP2Y0A60SZLF Analog Distance Sensor was used. It requires a $3\ V$ voltage source which can be supplied directly from the micrcontroller which runs off the same voltage.

The distance sensor outputs an analog signal which is related to the height. This analog

signal is fed into the ADC of the microcontroller and using a linear relationship is mapped to the distance.

The Pololu distance sensor has a range of $10 - 150\ cm$, which is more than enough for the intended hopping height of $20 - 50\ cm$ due to the testing rig limits.

The height sensor was calibrated by setting it at $0.1\ m$ from a surface and using a scaling factor to adjust the linear relation between distance and voltage until an accurate value was achieved.

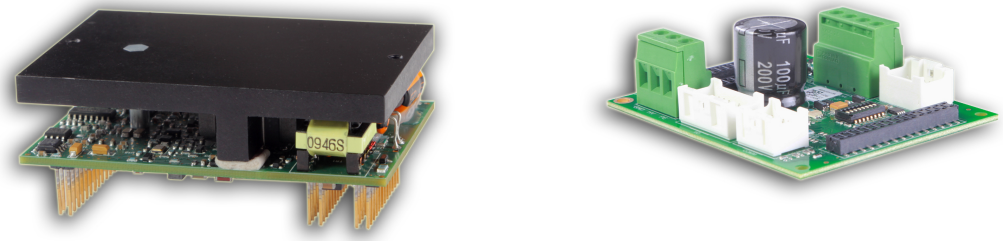
A 3D printed carrier was designed for the distance sensor which offset the sensor from the mounting plate and placed it in a central location above the linear guide.

8.4.3 Microcontroller

The microcontroller had to meet the following specifications:

- 4 x UART Ports
- 1 x ADC
- 1 x Floating point unit
- DMA Capabilities
- USB Debugging
- 5V tolerant UART ports

Being familiar with the STM32 series of microcontrollers, a STM32F4 board was chosen and met the above specifications with two additional USART ports for future peripheral needs.



(a) AMC DigiFlex Performance Servo Drive.

(b) AMC DigiFlex Performance Servo Drive mounting card.

Figure 8.10: AMC Servo Drive and Mounting Card.

Figure 8.11: T-Motor U10 Plus Brushless DC Motor.

8.5 Motors and Drivers

8.5.1 Driver Selection

8.5.2 Motor Selection

In the study by [**?**] various COTS (commercial off the shelf) motors were compared using the thermal specific torque as a performance measure. The T-Motor U10 Plus was found to have the highest thermal specific torque at $0.42\frac{Nm}{kgC^o}$ at $r_{gap} = 40mm$ [**?**]. When compared to the custom made MIT Cheetah motors at $0.71\frac{Nm}{kgC^o}$ at $r_{gap} = 49mm$ found in [**?**] they perform favourably.

8.5.3 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. 10.1.

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 $K_t = 0.08\ [Nm/A]$ 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. (8.1).

$$
\begin{aligned}
&V_t = K_e\omega_m + IR_m \\
&\tau_m = K_t I \\
&P_{elec.} = V_t I = K_e\omega_m I + I^2 R \\
&P_{mech.} = \tau_m\omega_m = K_t I\omega_m \\
&P_{loss.} = I^2 R_m \\
&P_{elec.} = P_{mech.} + P_{loss.} \\
&\therefore K_e\ [V/rad/s] = K_t\ [Nm/A] = 0.08
\end{aligned}
\tag{8.1}
$$

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.

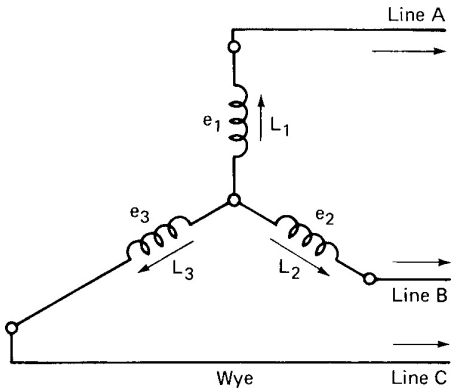


Figure 8.12: WYE connected BLDC motor windings.

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. (8.2) can be used to calculate J_m. The velocity vs. time plot used can be seen in fig. 8.13.

$$\begin{aligned} J_m &= \frac{T_{acc.}[N/m]}{a[m/s^2]} \\ &= \frac{IK_t}{a} \\ &= \frac{IK_t}{\frac{\Delta v}{\Delta t}}\ [kg/m^2] \end{aligned} \tag{8.2}$$

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.

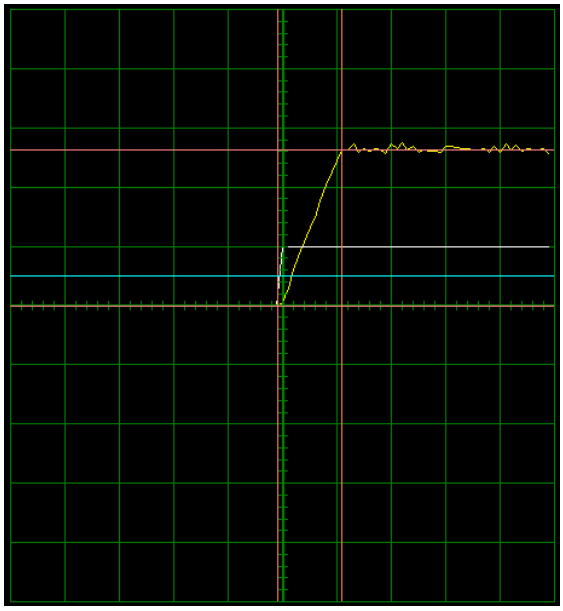


Figure 8.13: Velocity vs. time plot for 1A equivalent DC command.
(500 rpm; 500 ms/div)

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. (8.3).

$$\begin{aligned} K_m &= \frac{1}{B_m} \\ \tau_m &= \frac{J_m}{B_m} \\ K_e &= \frac{1}{R_m} \\ \tau_e &= \frac{L_m}{R_m} \end{aligned} \tag{8.3}$$

From eq. (8.3) 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 the motor viscous friction B_m is insignificant which is usually the case in mechanically well made BLDC motors.

The resulting motor model open loop root-locus plot can be seen in fig. 8.14. As expected the system has only negative real roots and will be stable in open loop.

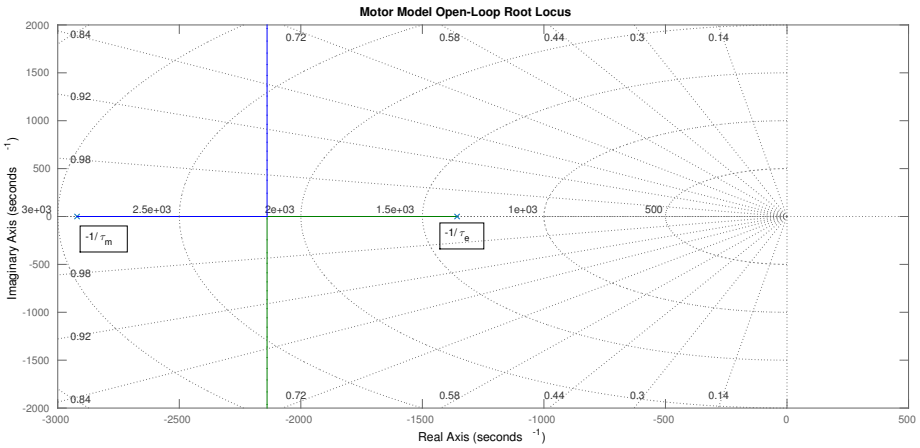


Figure 8.14: Motor model open loop root-locus plot.

8.5.4 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. 8.15.

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. 8.13, 8.16 and 8.17.

Current Control Loop

By using a 1 A 120Hz square wave current command the current PI control loop was tuned, as seen in fig. 8.16. 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.

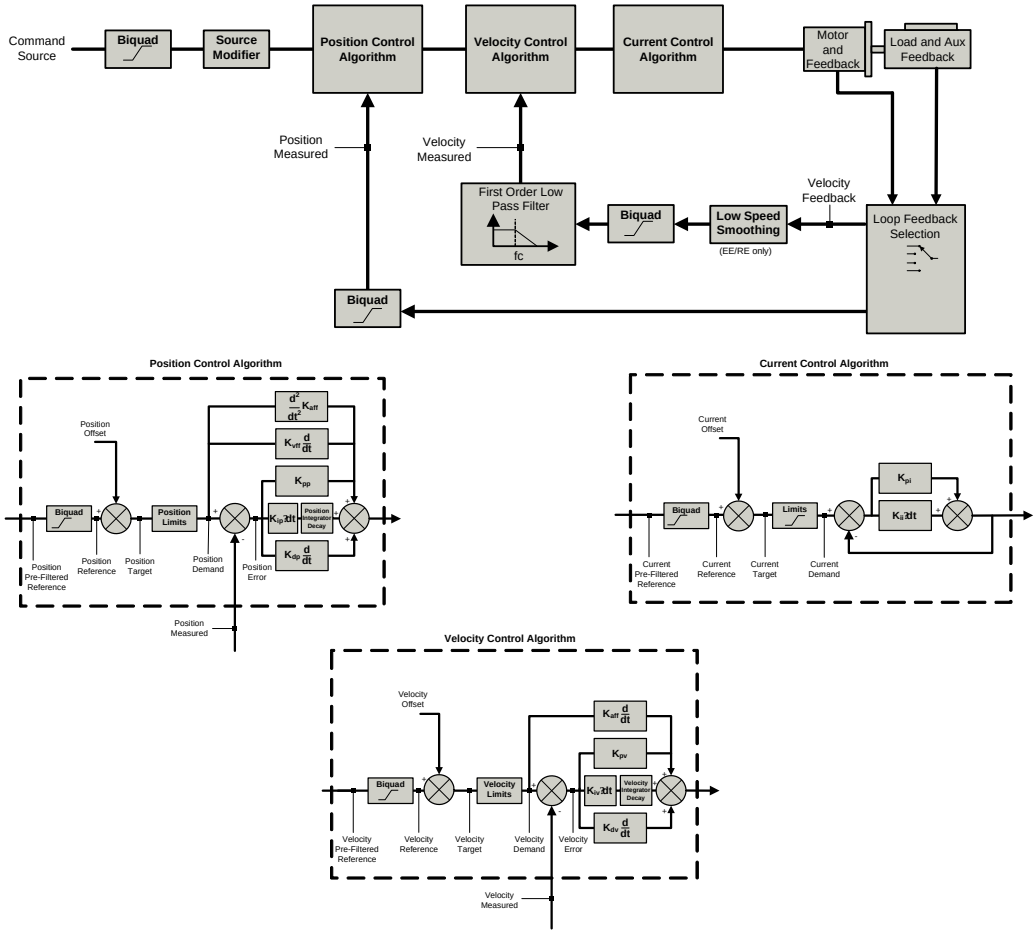
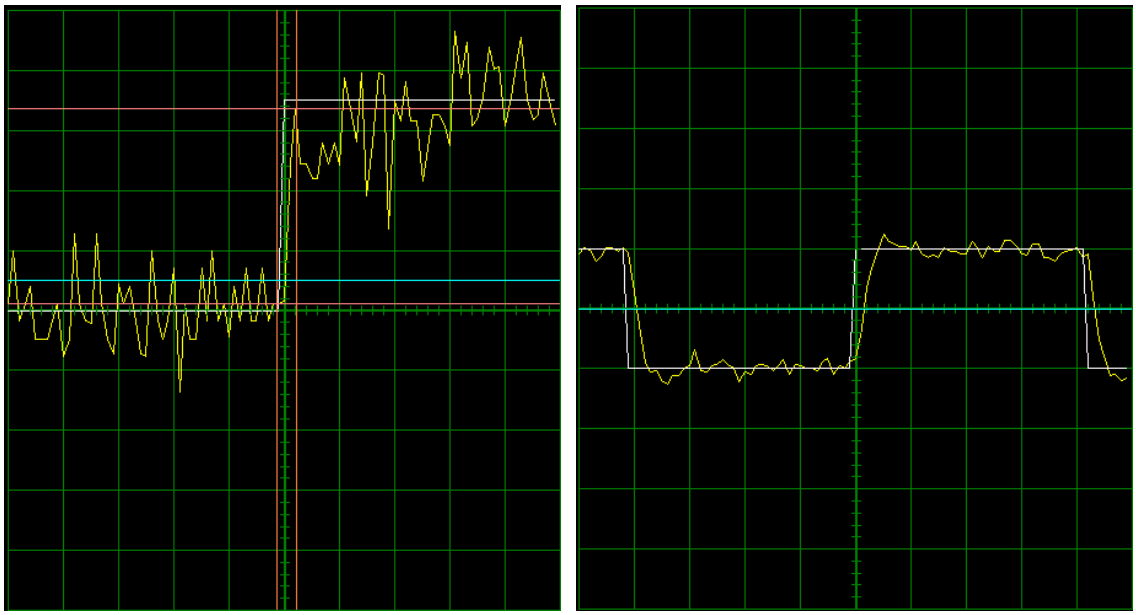


Figure 8.15: AMC DigiFlex Performance Servo Drive control loops (AMC, 2014).



(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 8.16: 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 8.5.5.

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. 8.17. 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.

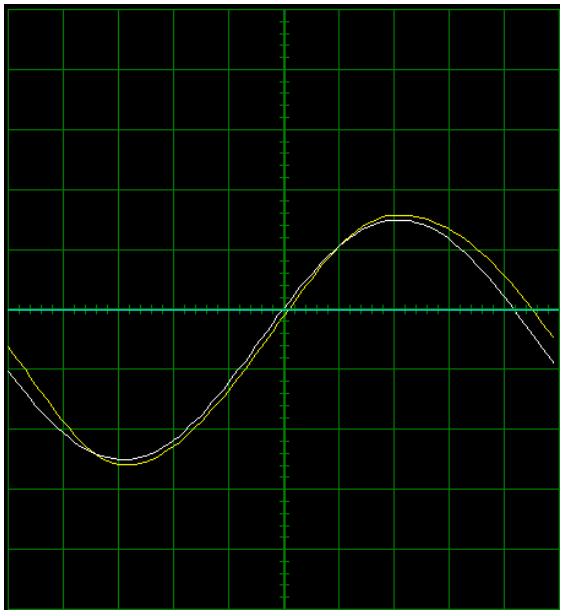


Figure 8.17: Motor driver position loop tuning - (-350:200) count 1Hz sinusoid command with 300 count offset.
(100 ct; 100 ms/div)

8.5.5 Motor Encoders

The Avago Technologies HEDL-5640-A13 rotary encoder was used for feedback of encoder position to the motor drivers which then calculate the motor position and velocity relative to a starting encoder count position.

The encoder has the following specifications:

- Optical sensing.
- Incremental counting.
- 500 counts per revolution.

The Avago encoder was chosen for its light, easily mountable frame along with the relatively high resolution position feedback of $0.72 deg./count$.

A shaft was designed to mount to the rear of the BLDC motor and interface to the encoder. The shaft was designed in OpenSCAD using programmatic CAD and can be seen in fig. 8.18. It was 3D printed by Justin Pead in White Lab using PLA plastic.

The shaft was designed to be mounting using three hex screws to the provided mounting point on the rear of the motors. A star configuration was used for the interface between

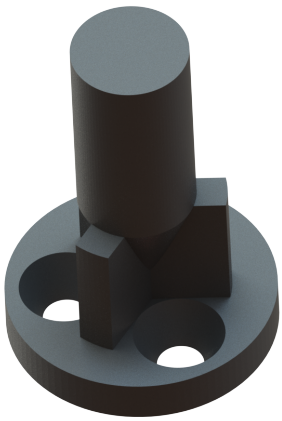


Figure 8.18: 3D printed PLA motor encoder shaft.

the shaft mounting point and the cylindrical encoder shaft so that these hex screws could be accessed easily.

Ideally the shaft should be milled using metal for better heat dissipation. Initially, before the aluminium mounting plate was used, the encoder shafts warped slightly while in operation and had to be bent back into shape. 3D printing was used for rapid prototyping and due to cost considerations.

9 Software Design

9.1 RTOS Communication Protocol

Ideally FreeRTOS tasks should operate with as much isolation as possible, with each task performing a specific function with different priority levels. In the case of a communication protocol, the overall task is inherently sequential so this leads to FreeRTOS being used in a sequential way with all tasks operating at real-time priority level.

FreeRTOS was chosen because of the following benefits:

1. Segmentation of code into specific tasks provides readable code for future users.
2. FreeRTOS task configuration provides a framework for future embedded robotic control systems as well as easily reconfigurable code.
3. Semaphores and queues lend themselves to sequential reception and processing of packets at predefined intervals very well.
4. Isolation of tasks provides easy debugging of communication issues.

9.1.1 Heartbeat Task

The *Heartbeat* task's primary function is to synchronise the communication protocol. It is the only task running with a 5 ms non-blocking delay. Every 5 ms it completes the following functions:

1. Compiles a read command packet for current, position and velocity.
2. Appends these three packets to the two motor driver transmit queues.
3. Gives a binary semaphore to the motor 1 and motor 2 transmit tasks.
4. Starts a 5 ms non-blocking delay to allow the two motor transmit tasks to complete their transmission and for the motor driver replies to be received and decoded.
5. Gives a binary semaphore to the PC transmit task to compile and send the newly received motor data for logging.

9.1.2 PC TX Task

The *TXPC* task is used purely for logging of data over serial on the Baleka C++ application, as seen in section 9.4.

Data is received in a queue from both the motor *RXMotor1* and *RXMotor2* tasks as well as the *Controller* task which is compiled into a packet along with status bits indicating various events and conditions.

9.1.3 PC RX Task

The *PCRX* task is the only task that is required to do non-synchronous reception of packets. The DMA reception is initialised and then the task waits until the full packet has been received before the RX complete interrupt handler gives the *PCRX* task a semaphore to continue decoding of the data.

9.1.4 TX Motor Task

9.1.5 RX Motor Task

9.1.6 Controller Task

9.1.7 FreeRTOS Timing

The default 1000 Hz tick rate was overridden with a tick rate of 5000 Hz to enable more fine tuned packet timing and delays. The timing configuration can be seen in listing 1.

In order to achieve a control loop rate of 200 Hz, a sampling time of 5 ms or 25 ticks was used.

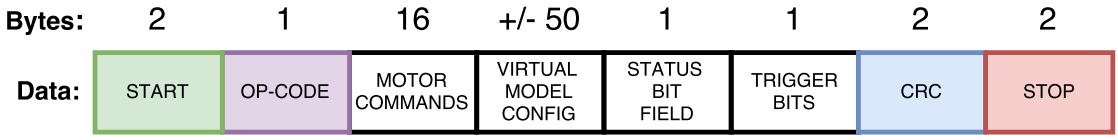


Figure 9.1: PC TX packet structure.

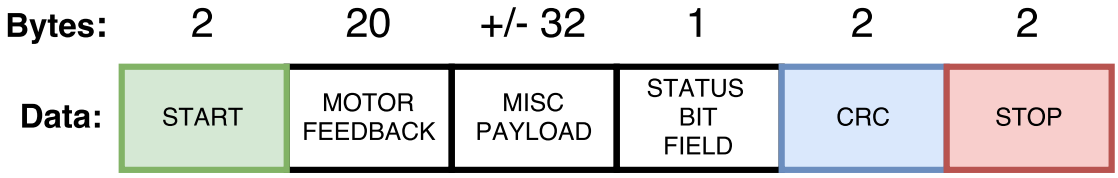


Figure 9.2: PC RX packet structure.

9.2 Packet Transmission

9.2.1 Structuring

packed struct...

9.2.2 Integrity Checking

CRC ...

9.2.3 Encoding and Decoding

PC Packet

Motor Driver Packet

SeqBits bits 2-5 switch case

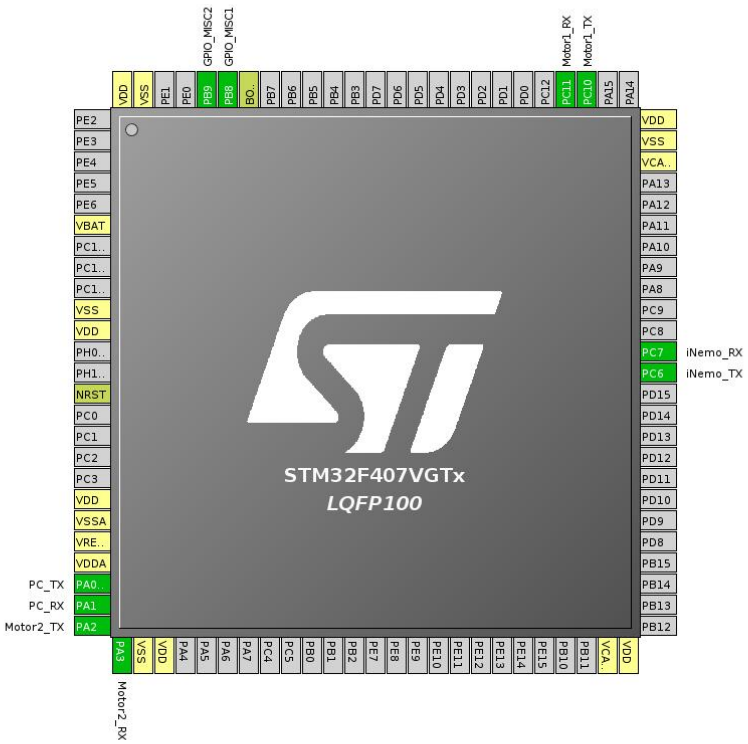


Figure 9.3: STM32F4 microcontroller peripheral configuration.

9.3 Peripheral Configuration

9.3.1 Protocol

9.3.2 Data Rates

9.3.3 Direct Memory Access

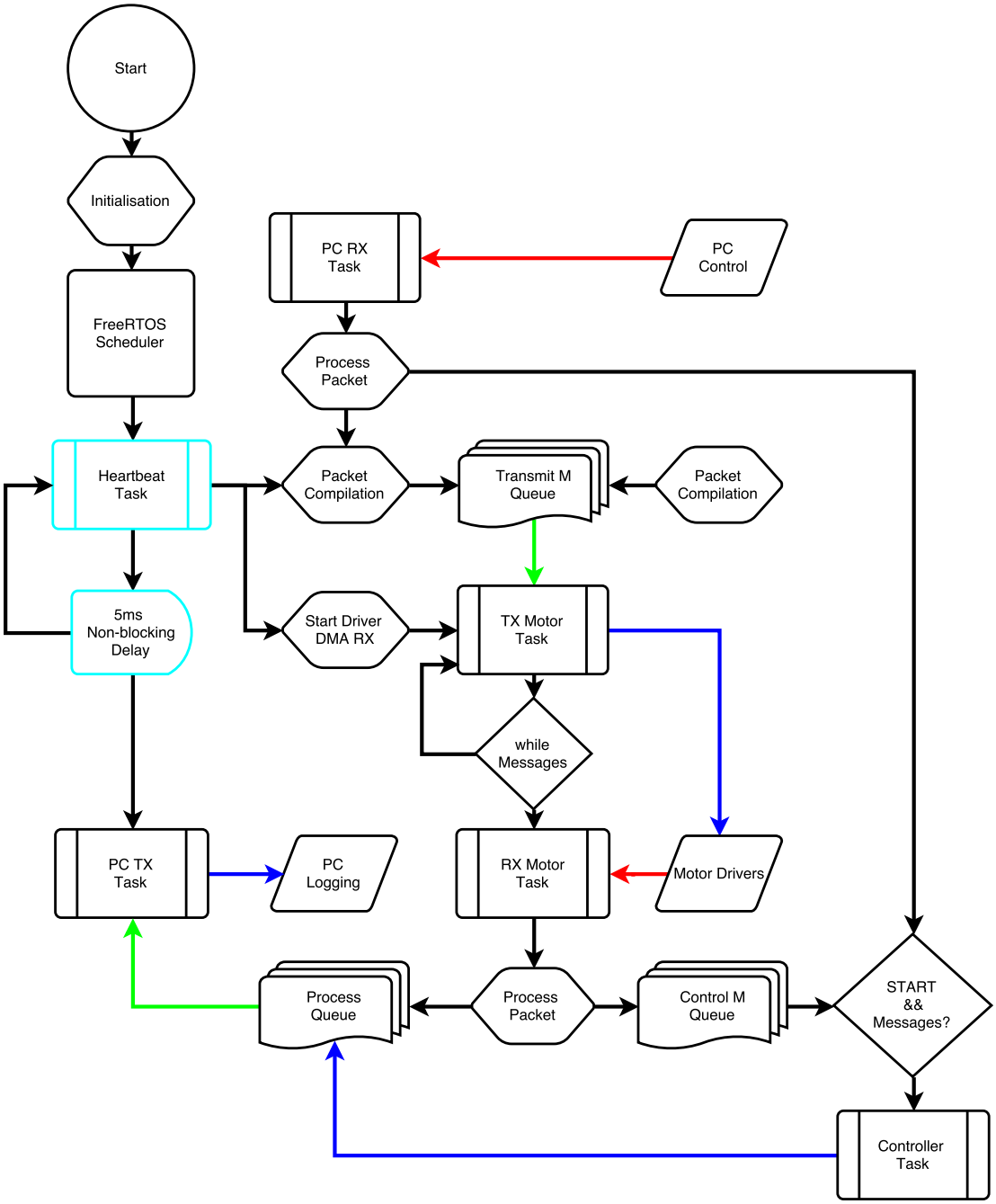


Figure 9.4: FreeRTOS communication protocol flow diagram.

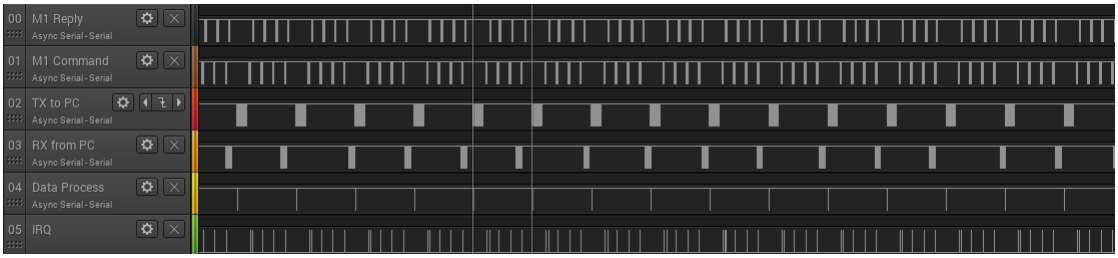


Figure 9.5: Communication protocol packet timing with 5 ms sampling rate.

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 9.1: Motor driver command protocol.

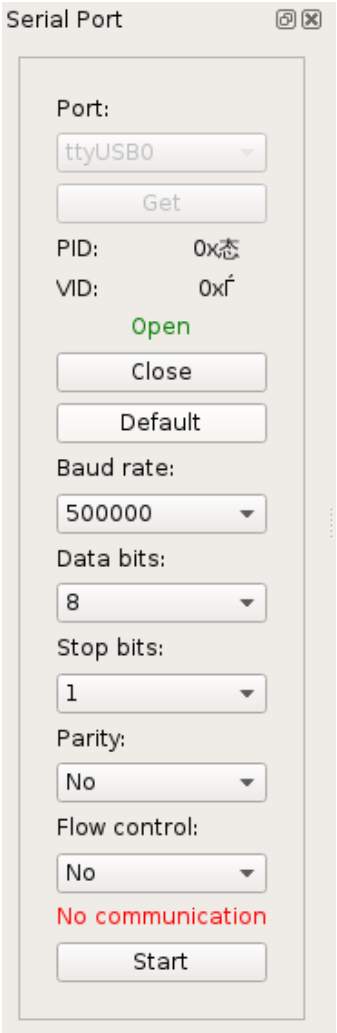
9.4 Graphic User Interface

To enable rapid prototyping, a basic pre-existing serial logging platform was used and plug-ins were developed to adapt the software for controlling, configuring, logging and live plotting of the Baleka robotic leg platform.

qSerialTerm, described as "A Qt based Serial Port terminal emulator", was built upon. It is distributed under the GNU General Public License (GPL) v3, which allows distribution, customisation and even sale of software published under the license, so long as basic conditions are met. It is copyright 2012 by Jorge Aparicio who's software repository can be found on GitHub: `https://github.com/JorgeAparicio/qSerialTerm`.

Qt Creator CPP was used for the software development and can be compiled to run on both Linux and Windows platforms - for reference all development was completed on a Linux platform.

Three .cpp files were used for the majority of the added functionality, namely: CRC.c, framewidget.cpp and serialportwidget.cpp along with their respective header files and Qt forms.

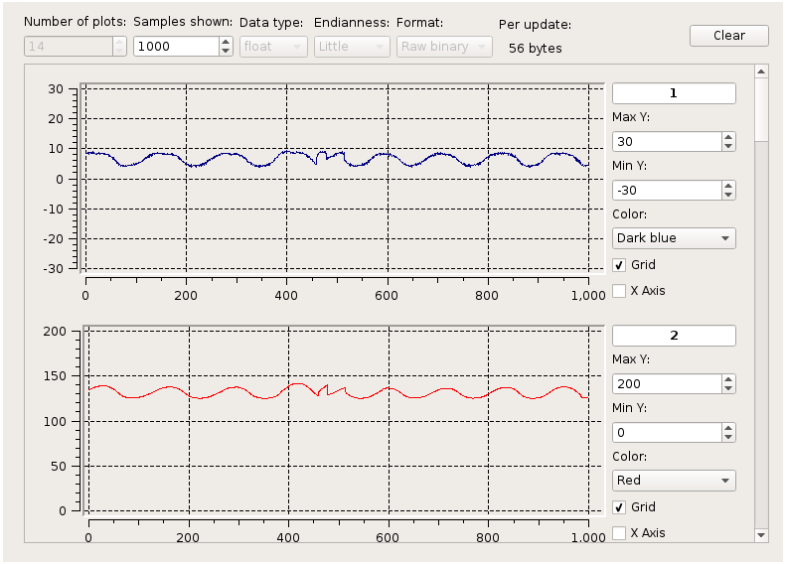


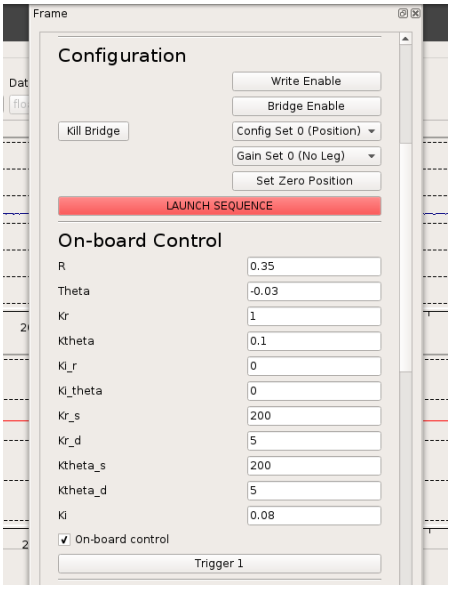
9.4.1 Serial Communication

9.4.2 Logging

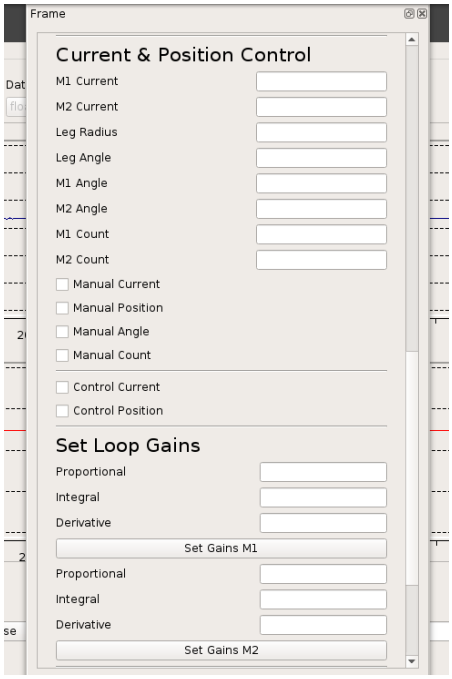
9.4.3 Live Plotting

9.4.4 Control Plug-in





(a) Configuration & On-board Control



(b) Current/Position Control & Control Loop Gains

Figure 9.6: Control interface plug-in.

10 Dynamic Modelling

10.1 Robotic Leg Modelling

10.1.1 Virtual Model

Simplicity

10.1.2 Dynamic Model

Complexity

Lagrangian Dynamic Model

In the study [**?**] a Lagrangian model and control system was developed for a hybrid machine system (constant speed motor with servo motor) with a five bar linkage end effector.

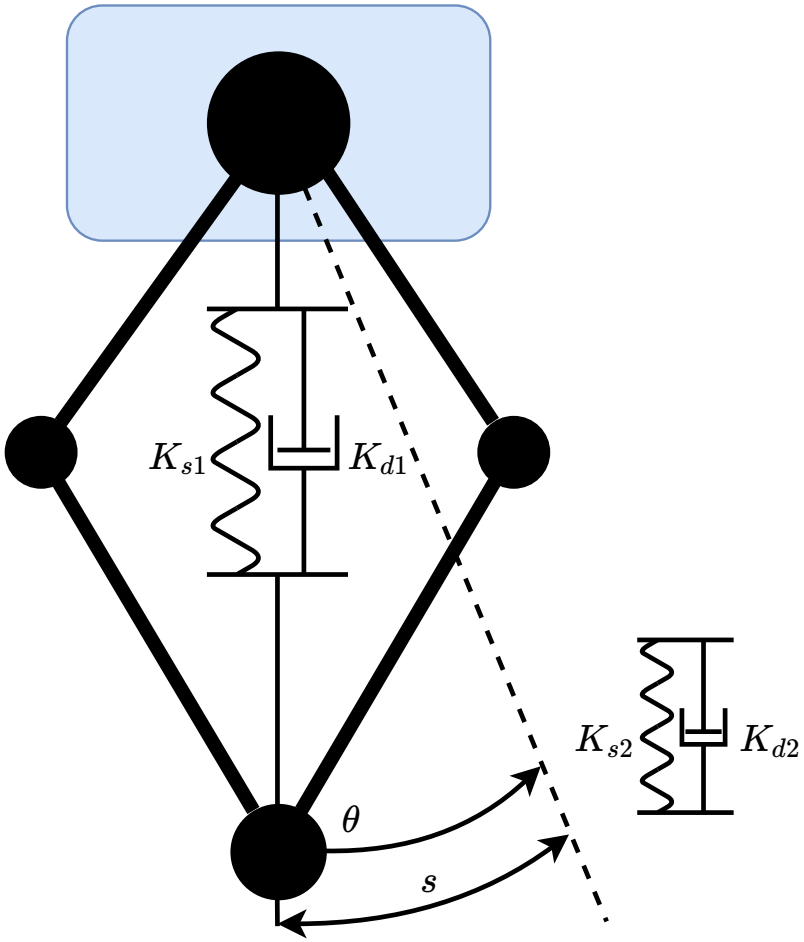


Figure 10.1: Leg spring-damper virtual model.

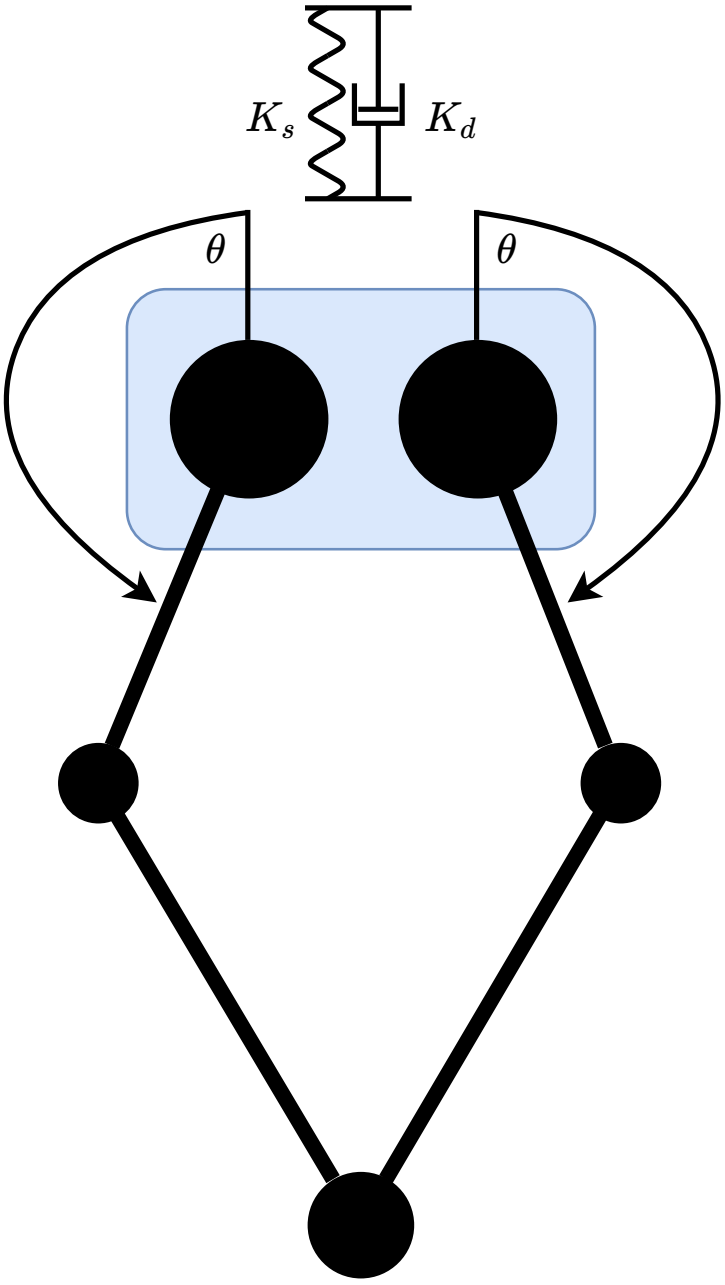


Figure 10.2: Joint spring-damper virtual model.

10.2 Spring-damper Mass Motion

Using Newton's second law, the downward force of a mass under acceleration is:

$$F_m = ma = m\ddot{x} \tag{10.1}$$

A spring with spring constant k and a damper with damping constant c have restoring forces as follows:

$$\begin{aligned} F_s &= kx \\ F_d &= c\dot{x} \end{aligned} \tag{10.2}$$

Given the spring-damper mass system in fig. 10.3, free to oscillate, the equation of motion below is derived:

$$\begin{aligned} -m\ddot{x} &= kx + c\dot{x} \\ m\ddot{x} + kx + c\dot{x} &= 0 \end{aligned} \tag{10.3}$$

By setting $x(t) = Ae^{\lambda t}$, the roots of the ODE above are:

$$\lambda_{1,2} = \frac{-c \pm \sqrt{c^2 - 4mk}}{2m} \tag{10.4}$$

Critical Damping

$$\begin{aligned} c^2 - 4mk &= 0 \\ c_{critical} &= 2\sqrt{mk} \end{aligned} \tag{10.5}$$

Damping Ratio

$$\zeta = \frac{c}{c_{critical}} \tag{10.6}$$

The equation eq. (10.4) can be restated in terms of the damping ratio as follows:

$$\lambda_{1,2} = (-\zeta \pm \sqrt{\zeta^2 - 1})\omega_0 \tag{10.7}$$

Under, Over and Critical Damping

For the spring-damper system to be under, over or critically damped, the following conditions must be met:

- Under: $\zeta < 1$ with imaginary roots

- Over: $\zeta > 1$
- Critical: $\zeta = 1$

For the robotic leg, ideally we want an under damped system when landing - this ensures some of the shock of landing is dissipated by the spring damper oscillations.

Experimental Damping

Experimentally the damping ratio ζ can be determined using the logarithmic decrement calculation. The logarithmic decrement is found by taking the natural logarithm of the ratio of amplitudes of two waveform peaks - this is then equated as follows:

$$\begin{aligned} \delta_n &= ln(\frac{x_a}{x_b}) \\ \zeta &= \frac{\delta_n}{\sqrt{(2\pi n)^2 + \delta_n^2}} \end{aligned} \tag{10.8}$$

where $n = b - a$ determines the number of peaks between measurements.

Natural Frequency

The natural frequency ω_0 is the frequency the undamped system will oscillate at if given an initial x offset and left to freely oscillate:

$$\omega_0 = \sqrt{\frac{k}{m}}\ [rad/s] \tag{10.9}$$

The damped natural frequency is the same as above, but with the damping constant c not equal to zero:

$$\omega_d = \sqrt{1 - \zeta^2}\omega_0\ [rad/s] \tag{10.10}$$

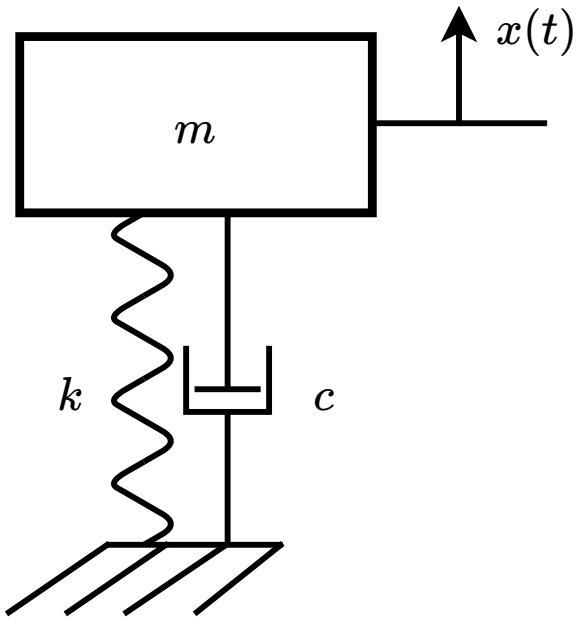


Figure 10.3: Spring-damper mass model.

10.3 Leg Spring-damper Model

The leg, when viewed as a spring-damper mass system in free space, has a mass of approximately $0.5\ kg$ which is the mass of the leg linkages.

On impact and launching the leg spring-damper mass system is in contact with the ground and therefore the mass of the plate and motors acts on the entire system with a mass of approximately $2.2\ kg$.

10.3.1 Impact Energy

When the leg is dropped on the linear guide from its maximum height of $0.5\ m$ the potential energy that needs to be absorbed by the spring-damper is $10.791\ J$ as calculated in eq. (10.11).

$$\begin{aligned} E_p &= mgh \\ E_p &= 2.2 \times 9.81 \times 0.5 = 10.791\ J \end{aligned} \tag{10.11}$$

The spring-damper system should absorb all this energy with the leg depressed to a radial set-point of at most $0.3\ m$, to insure the body does not impact the ground. The majority of the impact energy will be absorbed as spring potential energy with the dynamics being changed slightly by the damper kinetic energy as seen in the spring-damper drop tests fig. 12.1. The spring potential energy and damper kinetic energy are shown in eq. (10.12).

$$\begin{aligned} E_{ps} &= \frac{1}{2}kx^2 \\ E_{kd} &= \frac{1}{2}c\dot{x}^2 \end{aligned} \tag{10.12}$$

The velocity of the leg in free fall is approximately $2\ m/s$ found by performing drop tests and determining the number of video frames that it took for the leg to drop $0.4\ m$, which was approximately 5. Using eq. (10.13) the velocity was found.

$$v_{final} = \frac{height}{frames \times \frac{1}{fps}} = \frac{0.4}{5 \times 0.04} = 2\ m/s \tag{10.13}$$

A theoretical value for the spring constant k can be found by using conservation of energy as seen in eq. (10.14). A damping constant of $5\ N/(m/s)$ was assumed as determined

through experimentation in fig. 12.1.

$$\begin{aligned} E_p &= E_{ps} + E_{kd} \\ mgh &= \frac{1}{2}kx^2 + \frac{1}{2}c\dot{x}^2 \\ k &= \frac{2(mgh - \frac{1}{2}c\dot{x}^2)}{x^2} = \frac{2(10.791 - \frac{1}{2} \times 5 \times 2^2)}{(0.35 - 0.3)^2} = 632.8\ N/m \end{aligned} \tag{10.14}$$

10.3.2 Launch Energy

In order to launch the leg a set height of $0.4\ m$, the spring potential energy needs to be efficiently transferred into kinetic energy. This kinetic energy will launch the leg to a height with a corresponding potential energy. To calculate the spring constant needed to complete this jump assuming $80\ \%$ mechanical efficiency, equation eq. (10.15) is used. This is based on the principal of conservation of energy.

From eq. (10.12), $E_{ps} = \frac{1}{2}k\Delta x^2$. Using an initial radial set-point of $0.3\ m$ and decompressing the spring to a radial set-point of $0.4\ m$ we get a δx value of $-0.1\ m$.

$$\begin{aligned} E_p &= E_{ps} \\ mgh &= \frac{1}{2}k\Delta x^2 \\ k &= \frac{2mgh}{\Delta x^2} = \frac{2 \times 2.2 \times 9.81 \times 0.4}{0.1^2} = 1726.6\ N/m \end{aligned} \tag{10.15}$$

10.4 Virtual Compliance Model

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

$$F = [f_r\ f_\theta]^T \tag{10.16}$$

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

$$F = [f_r\ f_s]^T \tag{10.17}$$

$$f_a = k_s(a_{fbk} - a_{cmd}) + k_d(\dot{a}_{fbk} - \dot{a}_{cmd}) \tag{10.18}$$

11 Controller Development

11.1 Active Compliance

Active vs. Passive Compliance Passive compliance consists of using mechanical spring-damper components to adjust the dynamics of a robotics end effector. These come in the form of torsional springs, linear springs, hydraulic and pneumatic dampers to name a few.

In the study by G.A. Pratt and M.M. Williamson passive compliance in the form of a Series Elastic Actuator (SEA) was said to provide shock tolerance, lower reflected inertia, more accurate and stable force control, less damage to the environment, and the capacity for energy storage.[**?**]

Active compliance (AC) in robotic platforms is a relatively new area of research. Active compliance was chosen instead of passive compliance (PC) for the following reasons:

1. AC is mechanically simpler.
2. AC is mechanically lighter.
3. Good PC is expensive to implement due to lack of application specific off-the-shelf parts.

Most importantly, AC can be configured on the fly to adapt to the environment of the robot. This allows a robot to, much like their biological equivalents, to comply to the environment based on various sensory inputs, or to perform various acrobatic tasks making use of spring-damper compression and decompression.

Various combinations of AC spring-damper configurations were designed, implemented and tested in order to determine the best topology.

11.2 Dynamic Stability

Stand still, look around, turn around - these apparently simple tasks are composed of a network of biologically advanced sensors and muscular motor movements that seem

intuitive. To do the same relatively static and stable movement with a physically free robot is complex.

The alternative to static stability is dynamic stability. Much like a spinning top remains stable due to gyroscopic effects when a stationary top topples over; a dynamically hopping leg remains upright with a relatively simple control system compared to a leg trying to remain upright and walk forward.

In the book by M.H. Raibert et al., *Legged Robots That Balance*, an easily adaptable simple control algorithm for dynamic legged hopping was published. ...

11.3 Mechanical Impedance

The leg design has inherent mechanical impedance in the form of friction, slack and inertial mass. Without mechanical impedance the leg would continue acting like an ideal system when actuated. This can be seen in section 12.1 where the leg is modelled as a spring-damper - the amplitude of the oscillation of the virtual spring model with spring constant $K_s = 100$ decays with time.

This mechanical impedance is not easily accounted for in the dynamic model of the leg as it is highly non-linear in the case of a complex linkage system as used in the Baleka leg - this makes it difficult to control. The mechanical impedance was treated as a disturbance and in the case of dynamic movement of the leg it was ignored. Energy will be lost in the hopping motion and during leg movement, but this is insignificant and only noticeable when doing spring-damper tests as seen in fig. 12.1.

11.4 Control Loop Sampling Frequency

For high bandwidth proprioceptive force control S. Kalouche showed that the control loop sampling frequency should ideally be in the kHz scale.[**?**] This allows the foot force to ideally match the expected foot force when performing high frequency dynamic movements.

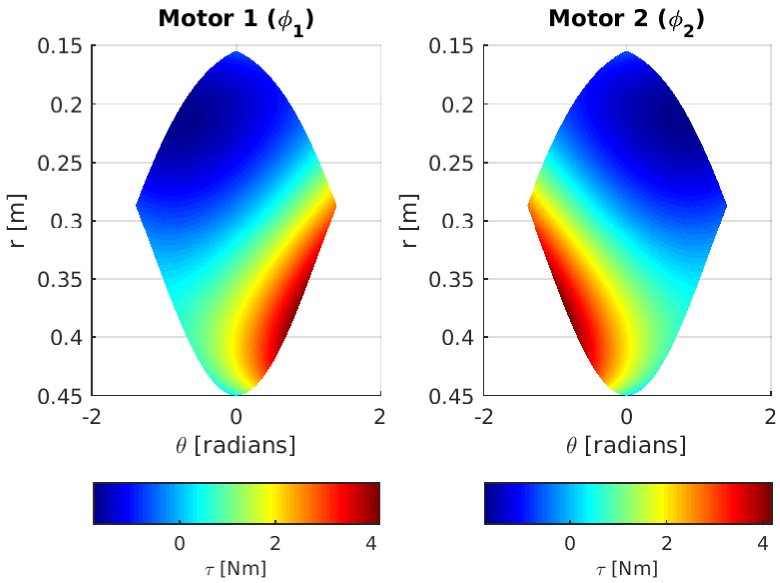
The control loop sampling frequency was practically limited by the motor driver response time as seen in fig. 9.5. For every packet sent to the motor driver a response is required before the next packet can be sent otherwise the driver system becomes unstable. A

maximum stable sampling frequency of $200Hz$ was achieved with room for packet decoding, processing and controller action.

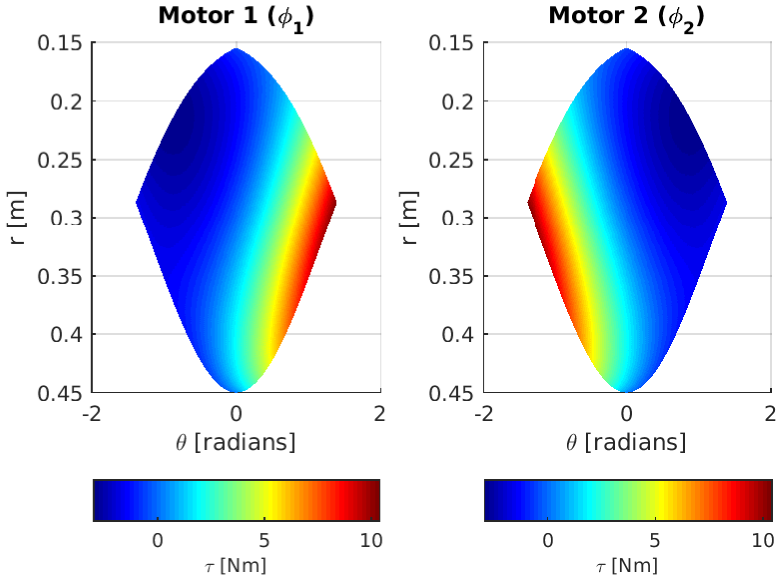
11.5 Force Control

$$\tau = J^T F \tag{11.1}$$

11.5.1 Current Control

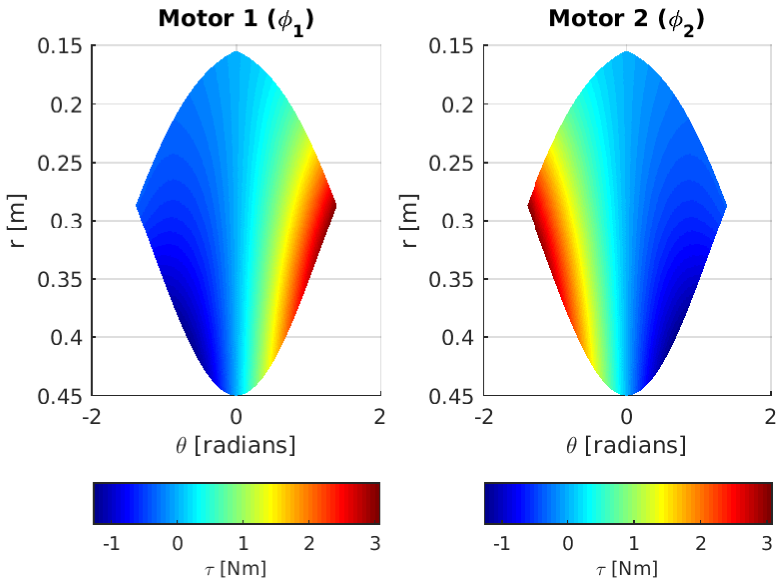


(a) Caption

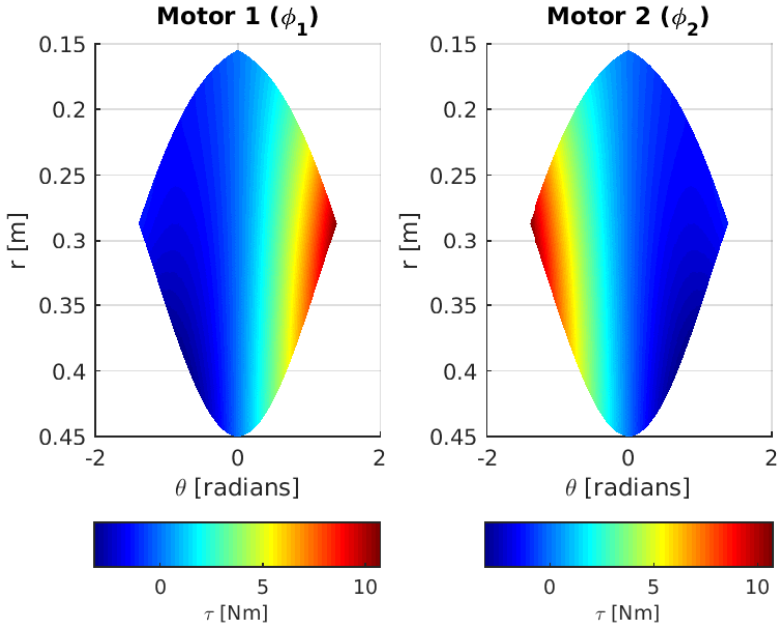


(b) Caption

Figure 11.1: Polar co-ordinate spring force mapping to motor torque.



(a) Caption



(b) Caption

Figure 11.2: Polar co-ordinate spring force mapping to motor torque.

11.6 Control Loop Design

Figure 11.3 is an adaptation of the control loop design found in [**?**]. Continuous position loop gain scheduling vs virtual spring-damper gain scheduling.

11.6.1 Current Control for Impulse Launch

Current saturation at 60A.

11.6.2 Torsional Spring-damper

$s = r\theta$

Normalises force - same unit of measurement, couples foot force to ground contact torque.

11.6.3 Full-leg vs. Joint Active Compliance

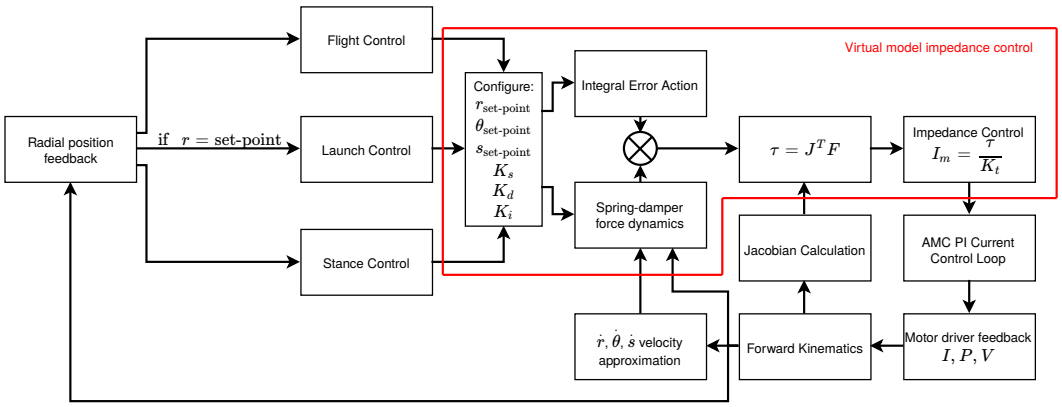


Figure 11.3: Virtual model impedance control loop.

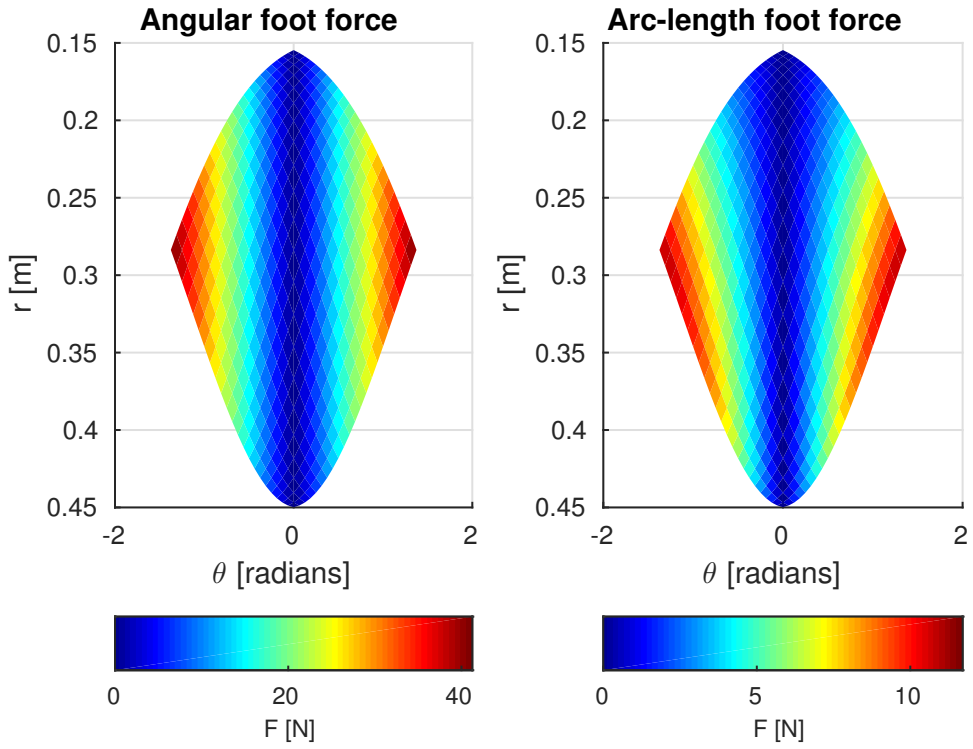


Figure 11.4: Rotational foot force comparison using angle and arc-length torsional spring virtual model.

12 Experimental Testing

"Jump!"

— Van Halen, *1984*

12.1 Virtual Spring-damper Tests

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

Scaling factor of 0.01...

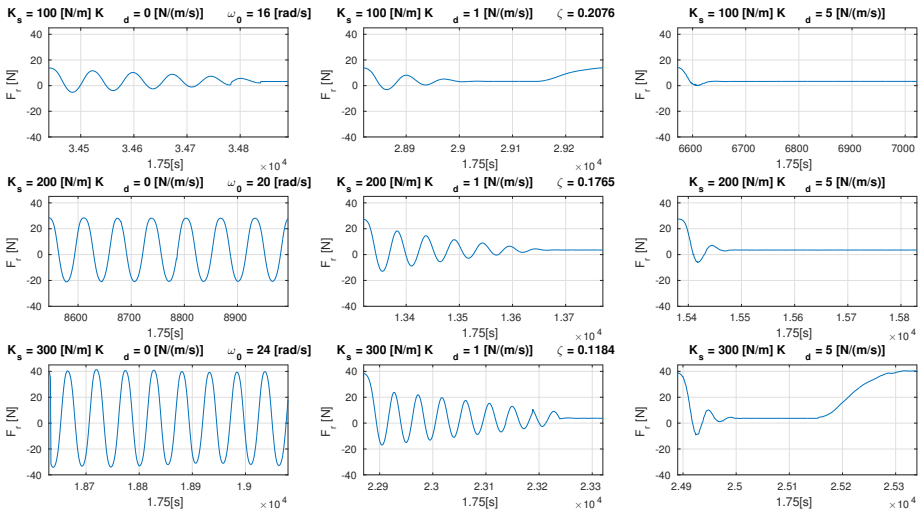


Figure 12.1: Full-leg spring damper testing for radial offset.

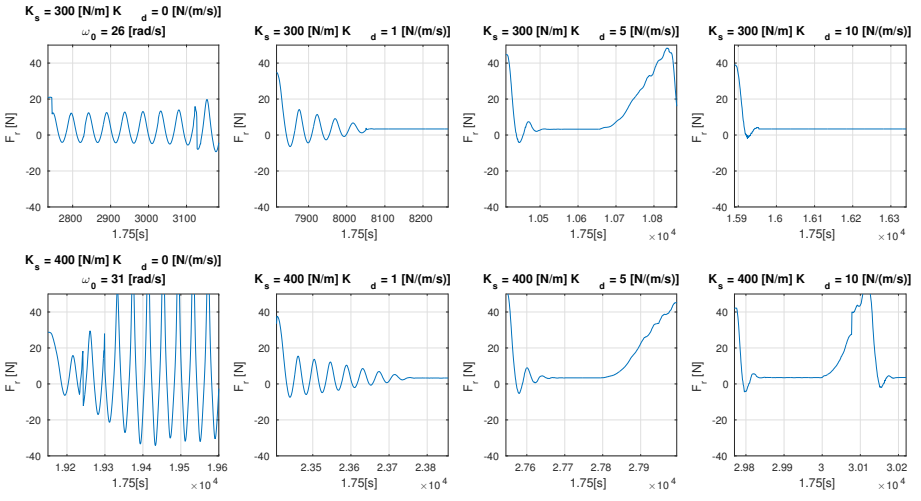


Figure 12.2: Joint spring damper testing for radial offset.

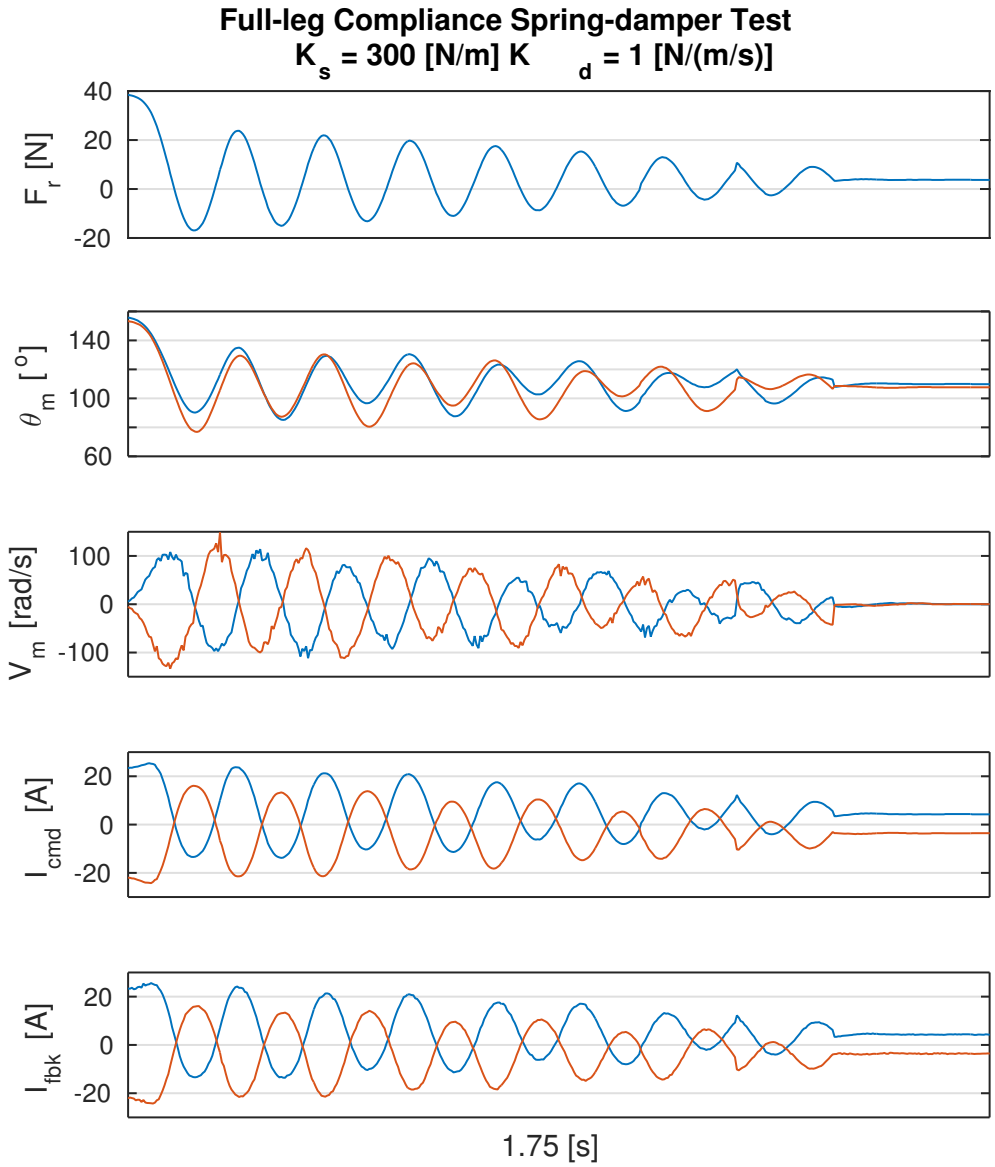


Figure 12.3: Full-leg spring damper motor control.

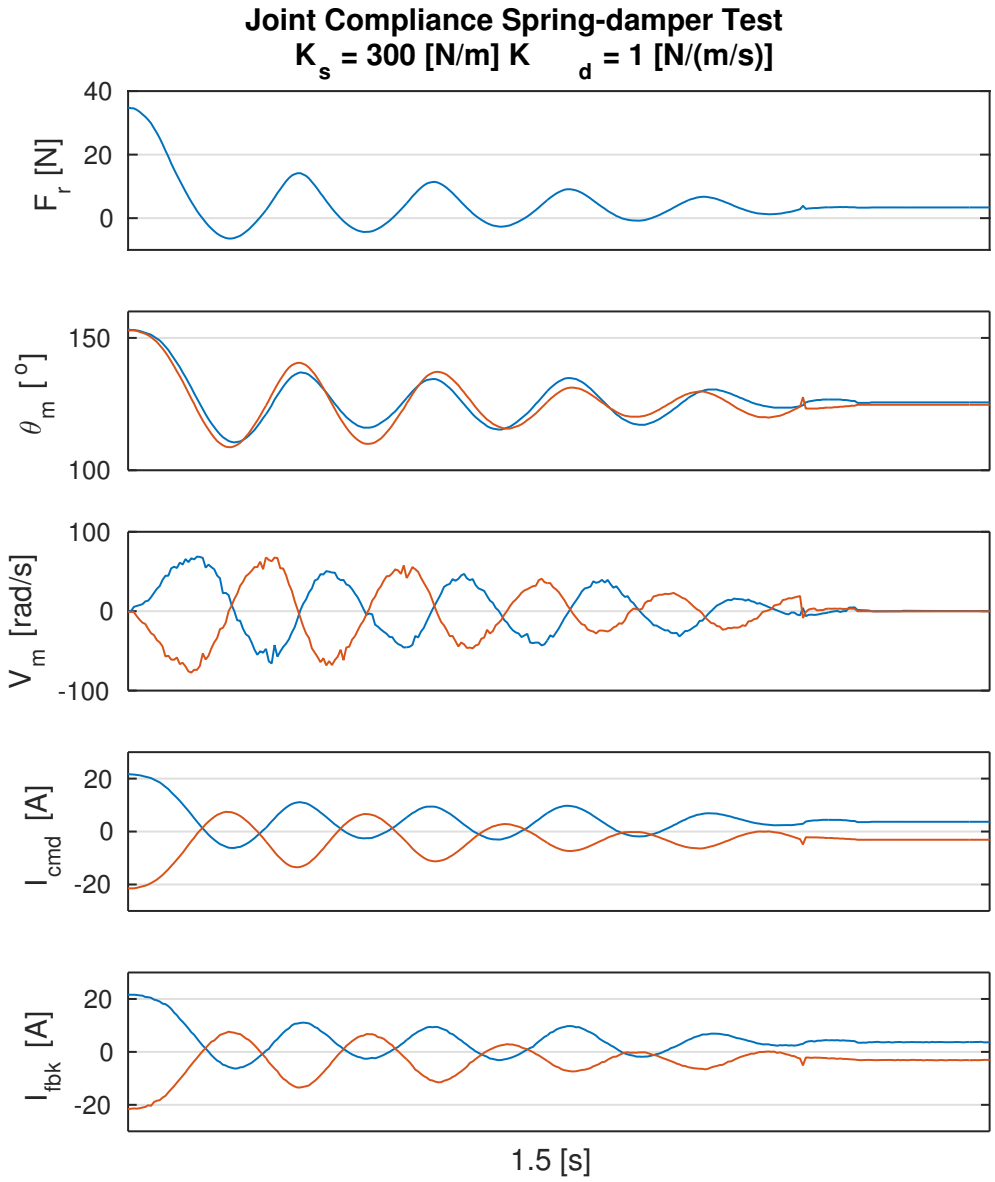


Figure 12.4: Joint spring damper motor control.

12.2 Drop Tests

A theoretical value for the spring constant K_s with $K_d = 5N/(m/s)$ for radial spring-damping upon impact were determined in section 10.3 using conservation of energy and an ideal dynamic response. The calculated spring constant of $632.8\ N/m$ was used as a starting point for testing and the damping was varied from $0\ N/(m/s)$ to $5\ N/(m/s)$ to confirm the values derived.

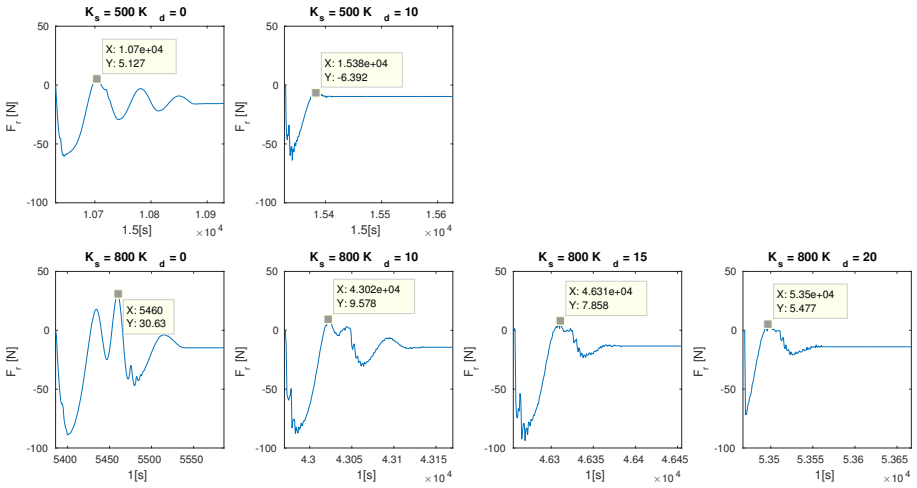


Figure 12.5: Leg spring damper drop testing.

12.3 Launch Tests

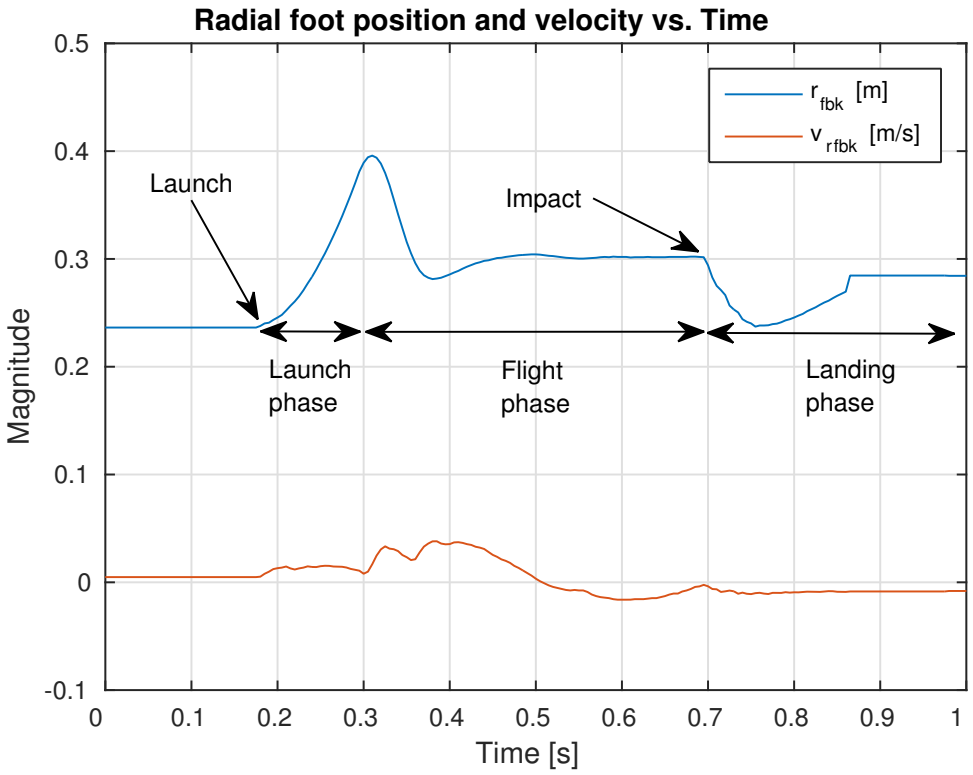


Figure 12.6: Jump foot radial position and velocity (launch and compliant landing).

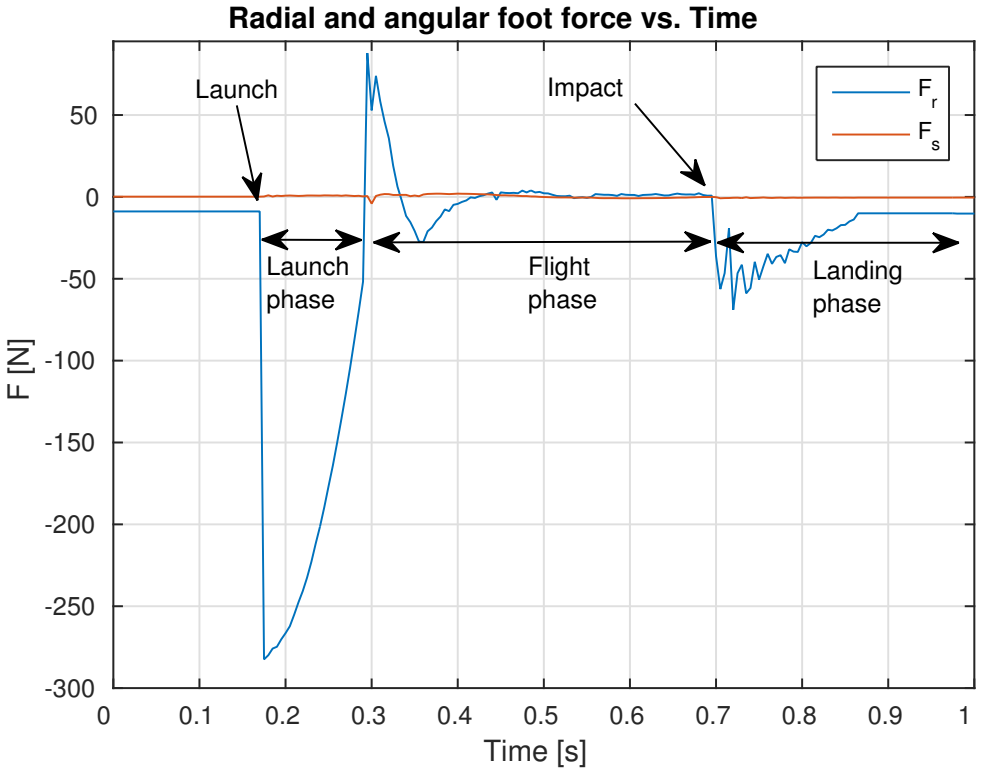


Figure 12.7: Jump foot force output (launch and compliant landing).

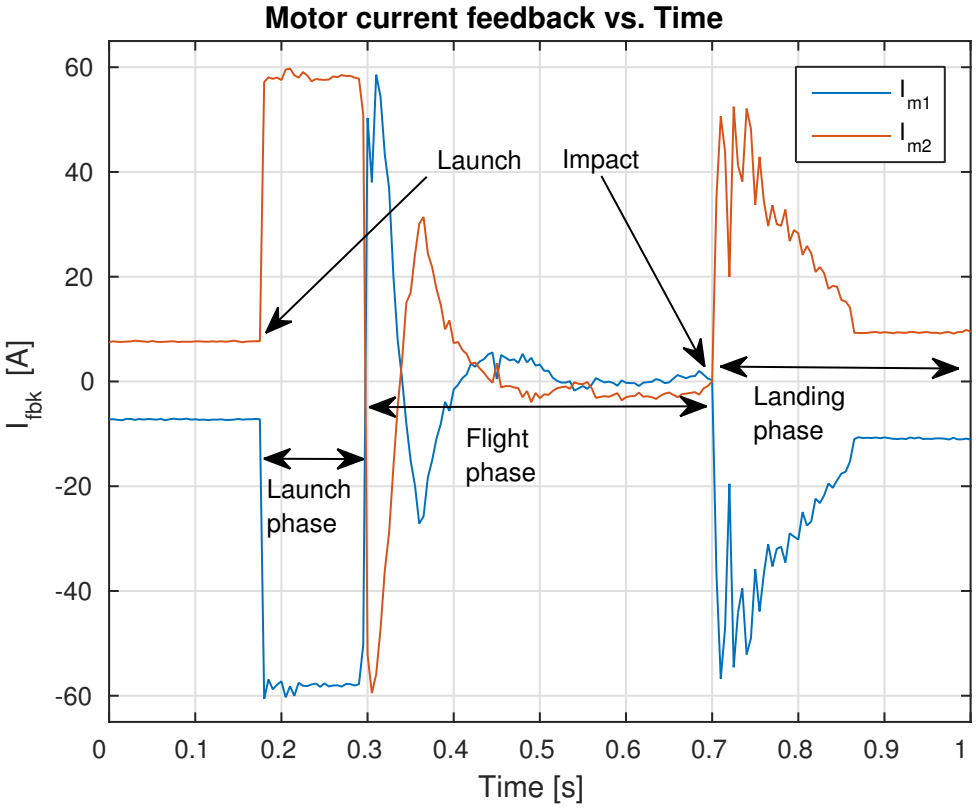
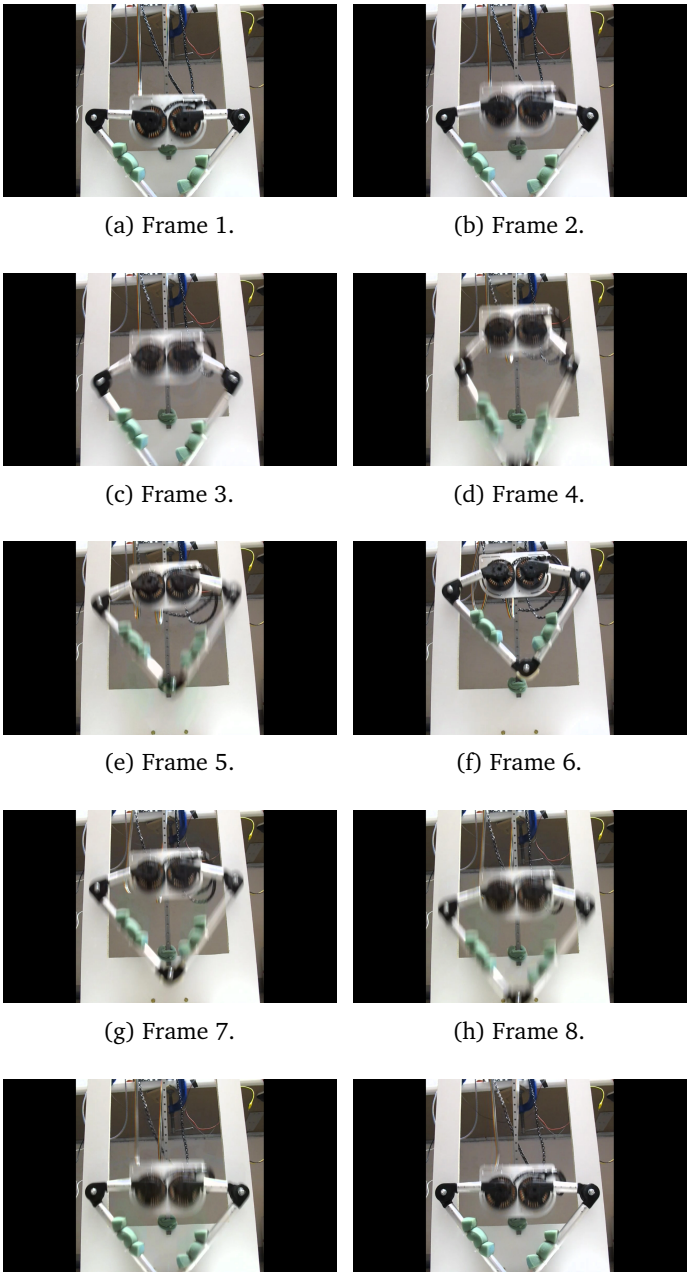


Figure 12.8: Jump motor current feedback (launch and compliant landing).



(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

(e) Frame 5. (f) Frame 6.

(g) Frame 7. (h) Frame 8.

(i) Frame 9. (j) Frame 10.

Figure 12.9: Leg launch with compliant landing.

12.4 Current Tracking

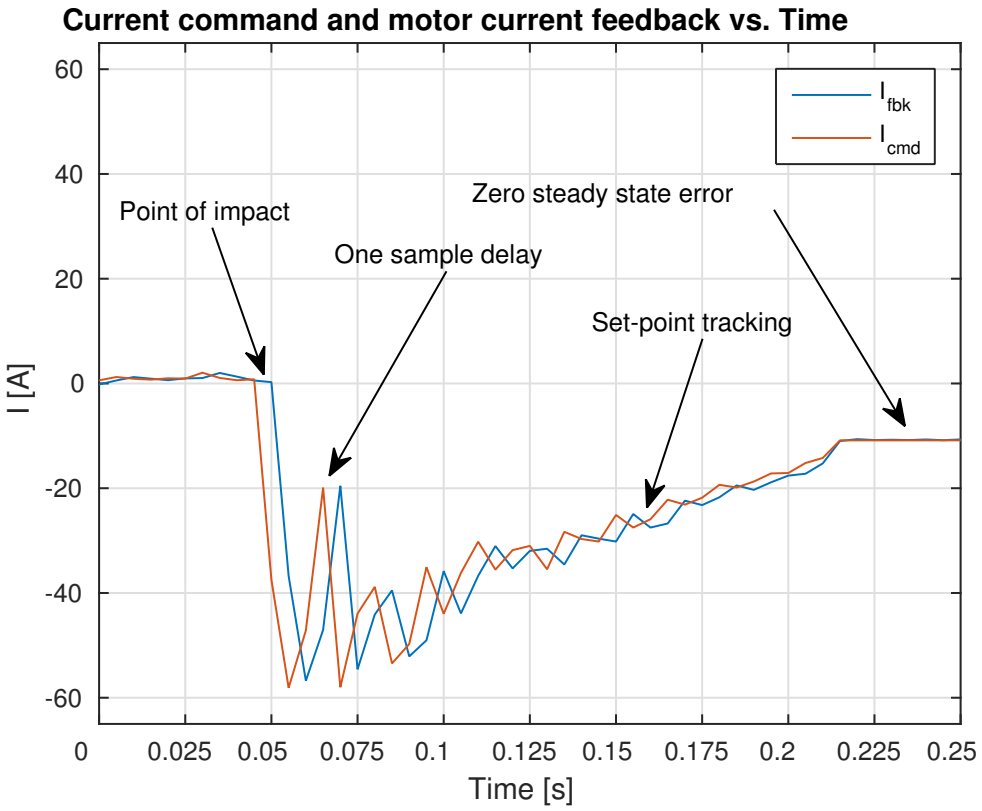


Figure 12.10: Current control tracking.

13 Design Validation

14 Conclusions

15 Recommendations and Future Work

Implement PID or MPC flight control with virtual model ground force control.

A Code

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

```
//Communication Timing
uint8_t Ts = 25; //Sampling time in 1/_X_ ms
uint8_t Td = 3;
//NB: #define configTICK_RATE_HZ ((TickType_t)_X_000) in FreeRTOSConfig.h
```

Listing 1: FreeRTOS timing configuration.

```
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 2: PC RX "packed" packet structure.

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

Listing 3: Byte conversion union.

```
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 4: 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 5: Motor packet compilation function.

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

Listing 6: Motor packet compilation current command example.