DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Design and Stability of a Quadruped Robot
Design av en balanserande fyrbent robot

ALGOT LINDESTAM

DAVID LORANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Design and Stability of a Quadruped Robot

ALGOT LINDESTAM, DAVID LORANG
algotl@kth.se, lorang@kth.se

Bachelor's Thesis at ITM
Supervisor: Nihad Subasic, subasic@kth.se
Examiner: Nihad Subasic, subasic@kth.se

TRITA-ITM-EX 2021:33

Abstract

We are currently in a revolution in robotics where more tasks are being handled by machines than ever before. For this reason, the goal of this project was to build a four-legged robot with an implementation of dynamic stability.

The developed robot is a *dog style* robot with reversed knee joints. The robot is controlled by an Arduino UNO microcontroller, that processes information from a gyroscope to drive it's servo motors. It is capable of maintaining it's balance while standing on a varying incline. The motion is based upon an inverse kinematic model of the leg geometry and assumes a planar ground surface.

Keywords: Mechatronics, Robotics, Balance, Stability, Quadruped.

Referat

Design av en balanserande fyrbent robot

I dagsläget ser vi en snabb expansion i användningen av robotar för att utföra alltmer avancerade uppgifter. På grund av detta var målet med detta projekt att utveckla en fyrbent robot med en enkel implementation av självbalansering.

Den framtagna prototypen är en robot av *hundstil* med bakåtgående knän. Styrenheten är en Arduino UNO mikrokontroller. Med information från ett gyroskop styr denna roboten med hjälp av dess servomorer. Prototypen är kapabel att hålla balansen då den står på lutande underlag. Rörelsen är baserad på en kinematisk modell av bengeometrin och förutsätter en plan markyta.

Acknowledgements

We would like to thank our fellow students Viktor Kårefjärd and David Stridfeldt for helping us construct the poster for this project, when our software was not cooperating.

In addition to this, we would to thank Boston Dynamics and their development of the ”Spot” as inspiration for this project.

Lastly, we would like to show our gratitude for the demonstrations, lessons and guidance from our supervisor and mentor, Nihad Subasic.

Contents

List of Abbreviations

1 Introduction **1**
1.1 Background 1
1.2 Purpose 1
1.3 Scope 1

2 Theory **3**
2.1 Robot Components 3
2.1.1 General Structure 3
2.1.2 Microcomputer 4
2.1.3 Motors 4
2.1.4 Sensors 5
2.1.5 Batteries 6
2.1.6 PWM 6
2.2 Robot Kinematics 6
2.2.1 Reference frame and coordinate system 6
2.2.2 Inverse Kinematics 7
2.3 Motion 8
2.3.1 Balance 8
2.3.2 Three dimensional body motion 9
2.4 Regulatory Controller 10

3 Demonstrator **12**
3.1 Problem formulation 12
3.2 Construction 13
3.3 Components 13
3.3.1 Micro controller 13
3.3.2 Motors 13
3.3.3 Sensors 13
3.3.4 Batteries 14
3.3.5 DC-converter 14
3.3.6 Voltage display 14
3.4 Software 14
3.5 Function 15
3.6 Calibration 16

4 Results **17**
4.1 Mechanical design 17
4.2 Performance 18

5 Discussion **20**

5.1 Improvement Opportunities 20
5.1.1 Grip 20
5.1.2 Center of Mass 20
5.1.3 Motor Strength 21
5.1.4 Jitter 21
5.1.5 Kinematic Modeling 21
5.2 Further Development 21
5.3 Conclusion 21

Bibliography **23**

Appendices **24**

A Leg Inverse Kinematics **25**

B Wiring Diagram **27**

C Arduino Software **28**

D Acumen Simulation **36**

List of Figures

2.1 Dog and spider style robots. 3
2.2 The microelectromechanical gyroscope is as part of the MPU 6050 where the pins *SCL* and *SDA* are used to specifically control the gyro 5
2.3 Pulse width [13] 6
2.4 Schematic view of the robot with the main reference frame marked. Drawn by authors in Google Drawings. 7
2.5 Schematic view of the leg geometry with target coordinates and help dimensions. Given values are in black, wanted angles are in green and help measurements are in red. Drawn by authors in Google Drawings. 8
2.6 Robot stability criterion, foot-base illustrated in green. Drawn by authors in Google Drawings. 8
2.7 Desired body translation relative ground (left), and corresponding foot translations relative the body (right). Drawn by authors in Google Drawings. 9
2.8 Demonstration of a body-centered rotational transform of the foot coordinates. Drawn by authors in Google Drawings. 10
2.9 Demonstration of a base-centered rotational transform of the foot coordinates. Drawn by authors in Google Drawings. 10

3.1 The finished demonstrator. Image taken by Algot Lindestam. 12
3.2 Arduino UNO [19] 13
3.3 Functional overview of the robot control loop. Drawn by authors in Google Drawings. 15

4.1 Schematic view of the robot with joint axis and construction dimensions marked. Drawn by authors in Google Drawings. 17
4.2 Side view of robot response. Images taken by Algot Lindestam. 18
4.3 Front view of robot response. Images taken by Algot Lindestam. 18

5.1 Electronics chassis CAD-model with dummy battery holder (grey) on the right. By authors via Fusion 360. 20

A.1 Schematic view of the leg geometry with target coordinates and help dimensions. Given values are in black, wanted angles are in green and help measurements are in red. 25

B.1 Overview of the wiring between components in the robot. Motors are combined into one node for simplicity. Drawn by authors in Google Drawings. 27

List of Tables

4.1 Robot parameters 18

List of Abbreviations

BLDC	Brushless direct current. 5
CAD	Computer-aided Design. 13
CPU	Central Processor unit. 4
DC	Direct current. 5, 13
DOF	Degrees of freedom. 3, 7
ESC	Electronic speed controller. 4
GND	Ground. 14
I2C	Inter-Integrated Circuit. 5
ICSP	In Circuit Serial Programming. 13
IMU	Inertial Measurement unit. 6
IO	Input Output. 5
MEMS	Microelectromechanical system. 5
PLA	Polylaktid. 13, 20
PWM	Pulse width modulation. 6, 16
RC	Remote Control. 4, 5
SCL	Serial Clock line. 5, 14
SDA	Serial Data line. 5, 14
VCC	Voltage common connector. 6, 14

Chapter 1

Introduction

1.1 Background

Robots have evolved drastically in the last century, but we are now entering a revolution in robotics [1]. Machines are now able to perform tasks that previously was not imaginable, such as handling sensitive objects, driving and functioning in traffic. However, there are multiple areas where robots have yet to evolve, which can help and substitute humans in different ways. To be able to develop a machine of such calibre, one must first understand the fundamentals of robotics and mechatronics, which can be taught by creating a robot with basic attributes. A foundational attribute can be transportation by feet rather than by wheels, since walking allows the machine to travel through rougher terrain in a more efficient manner. [2].

1.2 Purpose

The goal of this project was to build a four-legged robot with a simple implementation of dynamic stability. This was done with the underlying purpose to improve our understanding and integration of inverse kinematics, regulatory control and mechanical construction.

To control the robot, two things were required. Firstly, end-point control of all feet must be possible. Secondly, to implement dynamic feedback, some way to sense the environment and it's effect on the robot was necessary. Therefore, these primary questions were asked:

- How can the kinematics of the robot's legs and body be calculated in three dimensional space?
- How can falling, or being otherwise disturbed from a stable position, be detected?
- How are leg movements planned to counteract disturbances given sensory feedback?

1.3 Scope

The task of constructing any type of legged robot with dynamic stability was undoubtedly a difficult one. The processing power on a micro-controller also limited the amount and size of possible calculations. Considering this and our available time and knowledge, the problem had to be simplified to be able to achieve any kind of result. The target was therefore set to build a robot that keeps it's body level and maintains a standing position when the ground tilts. To motivate this, consider a robot walking over uneven terrain. For the robot to not fall it would have to react to, and compensate for, the ground tilting in different ways. In part very similar to the set goal. Solving the problem would require a working kinematic model, and sensory feedback to detect the environment changing. For further simplification, the following things were not considered in the project:

- Stability on non-planar (uneven) terrain.
- Force feedback (often implemented with pressure sensors in the feet).
- Robot inertia and moment of inertia in the kinematic modelling.

Chapter 2

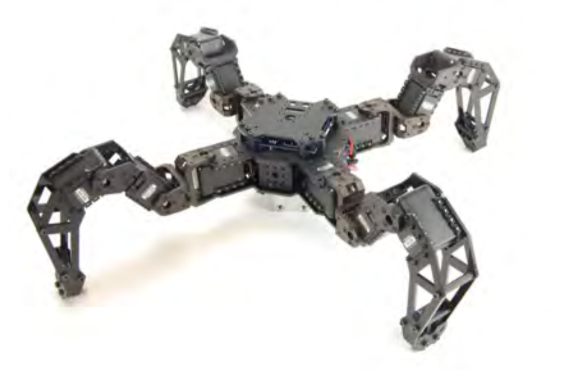
Theory

This chapter describes the underlying theory and function of components used in the project, as well as examining important concepts such as balance and stability.

2.1 Robot Components

2.1.1 General Structure

To examine dynamic stability the robot has to be able to move its body in three dimensional space. This means that the body has three axis of translation and three axis of rotation, totalling in six Degrees of freedom. By having three joints (three DOF) in each leg one could theoretically ensure control of the body with only two feet contacting the ground. There are then several possible configurations for these three leg joints. From this research, two main variations were found, named the *spider* and *dog* styles. The distinguishing difference is in the first joint between body and leg. The *spiders* rotate this joint around an axis normal to the ground, and the *dogs* rotate around one parallel to the ground. See Figure 2.1 below for clarification. A result of this is that the *spider* style of robot has an inherently wide foot-base and a low center of mass, compounding into a robot that is very stable from a static-mechanics point of view.


(a) Spider style robot, PhantomX MkII
Derelict Robot Industries

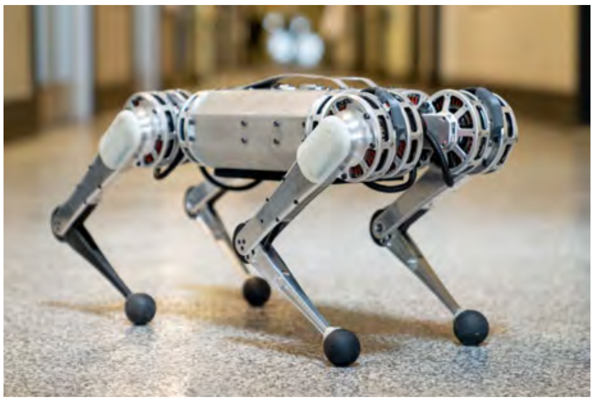

(b) Dog style robot, Mini Cheetah
MIT Biomimetic Robotics Laboratory

Figure 2.1: Dog and spider style robots.

2.1.2 Microcomputer

A microcomputer is a device that uses a single microprocessor as its CPU to perform all logic and arithmetic operations [3]. In other words, a microcomputer is used to communicate with all electronic devices in a system for the devices to operate.

2.1.3 Motors

Arguably the most vital component of the robot are the motors. They are what enable all useful robot function so having the right motors should be prioritized. To drive the leg joints correctly, the motors needed to satisfy two criterion.

1. Have angular feedback to position the leg properly.
2. Have enough torque to drive the joints under the robots own weight.

Considering criterion 1 there are two main options. You could use a motor with built in position feedback, commonly named servo motors. Or you could use an open loop motor and add external position feedback through use of an angular sensor (for example hall effect or potentiometer). Criterion 2 is harder to assess, but generally speaking the motors should be small and light with as much torque as possible. The project budget restricted the motor choice to ”hobby” and ”RC” class motors. On these price levels there are four common motor types. Following is a short summary of their respective properties with advantages and disadvantages (advantages marked with plus, disadvantages with minus).[4] Please note that price assessments were based upon findings made when researching motor choice among models available for purchase, and are therefore highly subjective.

1. DC motors
   - − No built in position feedback.
   - − Requires H-bridge or similar controller to be driven with a microcontroller.
   - + Relatively cheap.
2. BLDC motors
   - − No built in position feedback.
   - − Requires an ESC to be driven with a microcontroller.
   - − Relatively expensive.
3. Stepper motors
   - − No built in position feedback.
   - − Requires stepper driver to be driven with a microcontroller.
   - + High holding torque.
   - + Precise angular positioning.
   - − Relatively expensive.
4. RC-class servo motors
   - + Built in position feedback.
   - + Can be driven directly with a microcontroller.

+ High torque through use of a gearbox.
+ Relatively cheap.

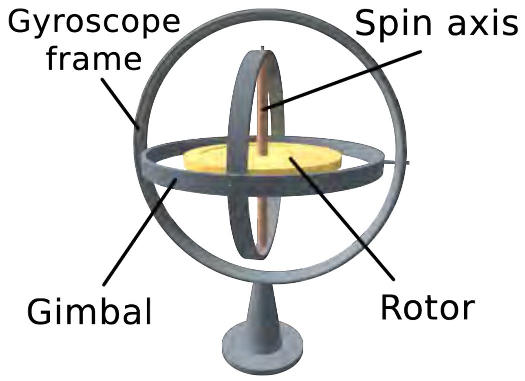
Servo motors were ultimately deemed the most fitting solution. The decision was based upon the fact that using anything else would drastically increase mechanical complexity, software complexity (increasing computational cost) and monetary cost. In addition to requiring external positional feedback, the alternative motor types (DC, BLDC or stepper) also require some form of controller external to the Arduino. Considering the fact that our robot were to use twelve motors, adding that many external controllers and angular sensors would require a minimum of 24 IO pins on the main controller. At the same time it would also significantly increase the design complexity, weight and size, that in turn would require more powerful motors to handle the increased load. In contrast, servo motors make for a simple design where the only required connection between the motor and microcontroller is a pulse-width modulated signal (see section 2.1.6 below), totalling in twelve IO pins used. The robot was therefore constructed using RC-class servo motors.

2.1.4 Sensors

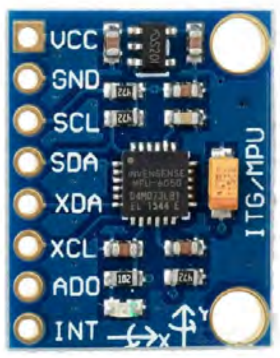
The robot must utilize sensors as an input to the micro computer, to adjust its hips and legs to regain stability when tilt is detected. However, since the pins of the micro computer were limited, it was necessary that the sensors use as few pins as possible. A method used to detect this sort of disturbance was to use a gyroscope.

A traditional gyroscope is a mechanical device that helps decide orientation with respect to earth's gravity [5]. The instrument's design includes a rotor which is placed on a spinning axis in the midpoint of larger spinning wheels (gimbals). The rotor remains still as the axis turns due to the gravitational pull and can thereby indicate which direction is towards earths mass center (See Figure 2.2a below).

MEMS gyroscopes on the other hand, detect rotation in different axis with small sensors that are sensitive to centrifugal force [6]. A change of rotation will adjust the position of the sensor with respect to the rotational axis, which gives the same result as a mechanical gyro (See figure 2.2b below). The electromechanical gyro is operated by I2C, a serial communication protocol, to transfer data bit by bit in a single line, Serial Data line (SDA). The output of the signal is then synchronized by a clock signal (SCL) [7].



(a) Mechanical Gyroscope [8]



(b) MPU 6050 [9]

Figure 2.2: The microelectromechanical gyroscope is as part of the MPU 6050 where the pins *SCL* and *SDA* are used to specifically control the gyro

2.1.5 Batteries

The batteries need to provide enough voltage to drive the electronics while simultaneously being able to provide enough current. With today's technology a readily available option is to use lithium ion batteries for their high energy and/or power density compared to to other battery types [10]. This type of battery has a nominal voltage of 3,6 volts and exist in many different form factors. With one of the most common being the "18650" cylindrical cell [11].

To adjust voltage the cells can be coupled in series, increasing voltage only, or in parallel, increasing capacity and current capability. Different configurations are herein written as xSyP, where x and y note the number of cells in series and parallel respectively. For example, 2S3P would mean having three blocks of two cells in series, coupled in parallel.

2.1.6 PWM

PWM is a method to use digital means to get analog results. A binary digital control is used to get square waves, which can simulate voltage in bewteen the VCC and ground. It does so by changing the time duration the signal spends on and off. The pulse width is equivalent to the time the signal stays on (see Figure 2.3 below). To change the analog value of the signal, the pulse width is to be increased or decreased. If the on-and-off pattern switches fast enough, a steady voltage between 0 and the maximum for the VCC will be held [12]. PWM has countless use cases but is here only used to control the servo motors.

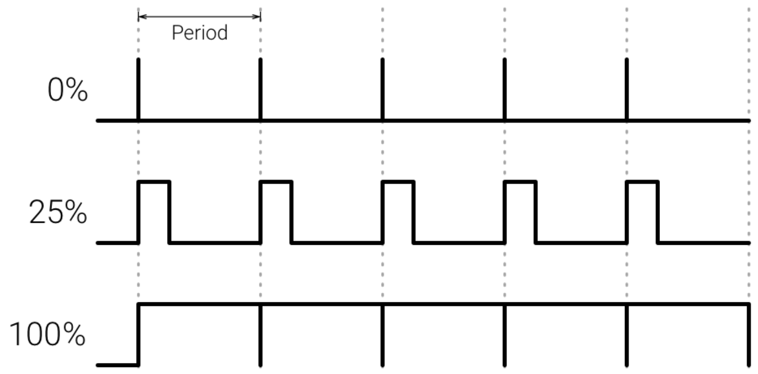
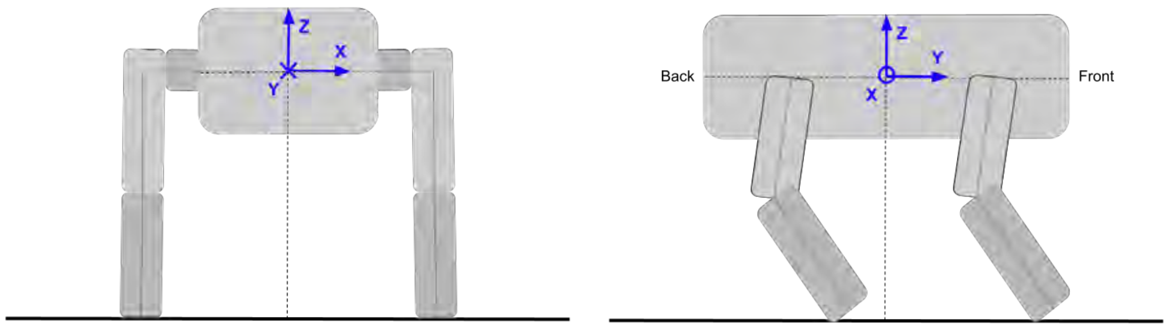


Figure 2.3: Pulse width [13]

2.2 Robot Kinematics

2.2.1 Reference frame and coordinate system

All calculations were done in a body-centered coordinate system where all foot placement is done relative to the robot body and not relative to the robot's surroundings. The main reference frame was thereby chosen as seen below in Figure 2.4. By calibrating the IMU with the body parallel to the ground, the robot also knows how this reference frame is rotated relative the ground, which is essential for all implementations of stability.



(a) Back view (b) Side view

Figure 2.4: Schematic view of the robot with the main reference frame marked. Drawn by authors in Google Drawings.

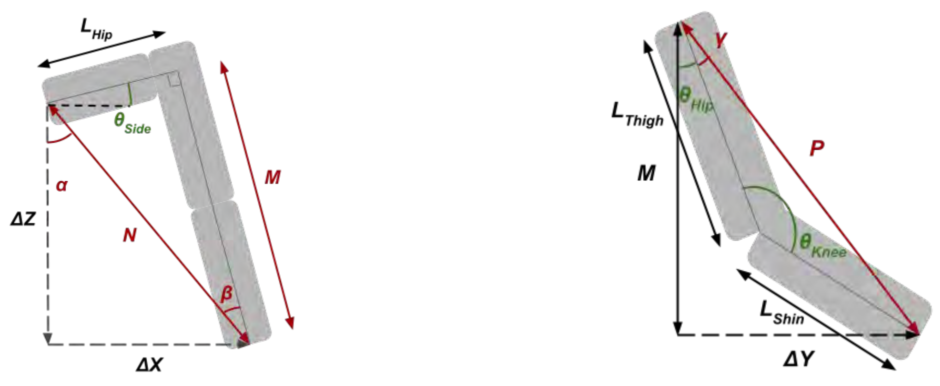
2.2.2 Inverse Kinematics

Inverse kinematics are the mathematics that calculate actuator states to achieve a given end point position [14]. The actuators are the joint angles and the end point is the foot on each leg (see figure 2.5 for angle definitions). Since each leg has three DOF (joints), and the position of a foot needs to be controlled in three dimensional space, there is a unique set of joint angles for each foot position in the task space [14]. Considering that the mechanical structure is relatively simple, we chose to calculate the inverse kinematics with an algebraic, geometric approach, as described in [14]. The target foot position were set in coordinate deltas (ΔX,ΔY,ΔZ) from the joint between the hip and body. Coordinates in the main reference frame can easily be transformed into this hip based frame by subtracting the hip joint coordinates (given by the construction dimensions L_{Side} and L_{Front}). For those interested, these calculations can be found in full in appendix A. The resulting expressions are as following.

$$\theta_{side} = tan^{-1}\left(\frac{\Delta X}{\Delta Z}\right) - tan^{-1}\left(\frac{L_{hip}}{\sqrt{\Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right) \tag{2.1}$$

$$\theta_{knee} = cos^{-1}\left(\frac{\Delta Y^2 + \Delta X^2 + \Delta Z^2 - L_{hip}^2 - L_{thigh}^2 - L_{shin}^2}{-2L_{thigh}L_{shin}}\right) \tag{2.2}$$

$$\theta_{hip} = tan^{-1}\left(\frac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right) - cos^{-1}\left(\frac{L_{shin}^2 - L_{thigh}^2 - \Delta Y^2 - \Delta X^2 - \Delta Z^2 + L_{hip}^2}{-2L_{thigh}\sqrt{\Delta Y^2 + \Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right) \tag{2.3}$$



(a) Back view (b) Side view

Figure 2.5: Schematic view of the leg geometry with target coordinates and help dimensions. Given values are in black, wanted angles are in green and help measurements are in red. Drawn by authors in Google Drawings.

2.3 Motion

2.3.1 Balance

The term *balance control* refers to how mankind, or other species, manage to keep posture for a given task such as running, walking or standing, and to maintain equilibrium while executing those tasks [15]. Since the robot bears strong resemblance to a dog and proper balance control is vital for the daily activities of an animal, balance control plays a great role within the overall control of the robot. However, an animal is, just like a human, able to predict instability before disturbing actions occur, which is something the robot is unable to do. Hence, the robot only reacts to external forces after they affect it.

The big question is then how do you keep the robot stable? We consider a simple model given from static-mechanics. More advanced stability criterion exist, but were outside the scope of this project [16]. Static-mechanics state the tipping criterion in simple terms. If the center of gravity of the robot is located outside the base area constructed by the planted feet, the robot will fall. This is illustrated in two dimensions in figure 2.6 below. The robot has to keep its center of mass over the base at all times.

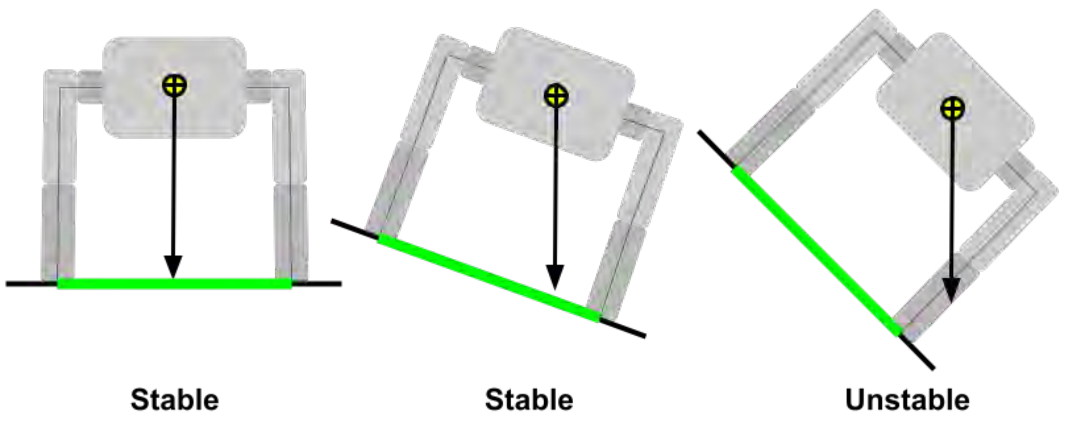


Figure 2.6: Robot stability criterion, foot-base illustrated in green. Drawn by authors in Google Drawings.

2.3.2 Three dimensional body motion

Assume that a foot in contact with the ground does not slip relative to the ground (such a foot is hereby designated as *planted*). Moving the robot relative to the ground is then equivalent to moving the body relative to the planted feet. Thereby, translation of the body along any axis in the reference frame can be achieved by translating all planted feet opposite to the desired body movement. Since this type of motion is linear, the same technique can be applied to translation along any vector in 3-D space.

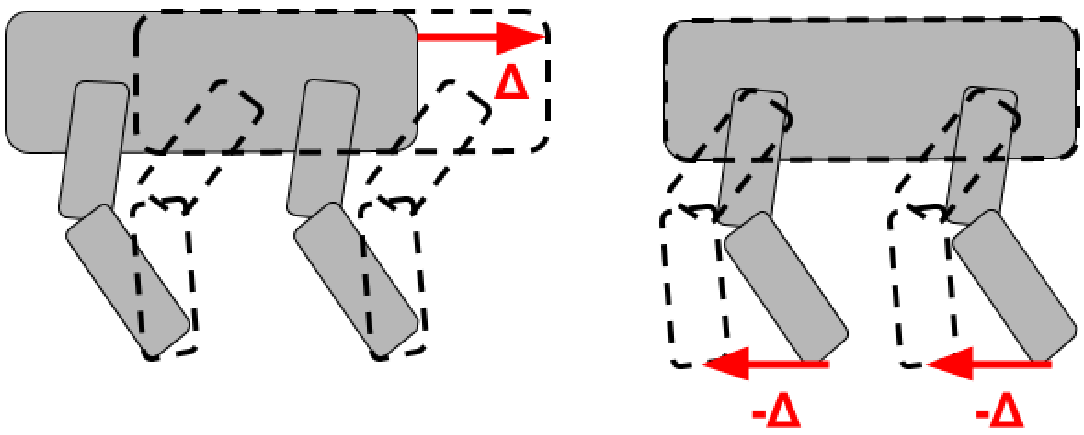


Figure 2.7: Desired body translation relative ground (left), and corresponding foot translations relative the body (right). Drawn by authors in Google Drawings.

Following the same logic, a rotation of the body around a point in space can be achieved by rotationally transforming the coordinates of all planted feet around this point, opposite to the desired body rotation. This fact is what the robot uses to correct against tilted ground.

For this correction to be effective the center point of the rotation has to be taken into consideration. By rotating around the body center, the robot will correct for ground tilt but its center of mass will drift to the side and not stay centered above the base (See Figure 2.8). Also, with this type of correction the motion mainly comes from a rotation of the hip, having to swing around the entire leg. Consider instead what animals tend to do. When standing on a slope your natural behaviour is to lower one foot and raise the other, keeping your body upright while maintaining stability. To emulate this exact behavior, let the feet positions instead rotate around the center of the support base itself (Figure 2.9), making the robot body stationary above the base and maintaining stability. This is done by simply transforming the foot coordinates by the current body height before applying the rotational transform and then transforming them back into the body centered frame to apply the kinematic model.

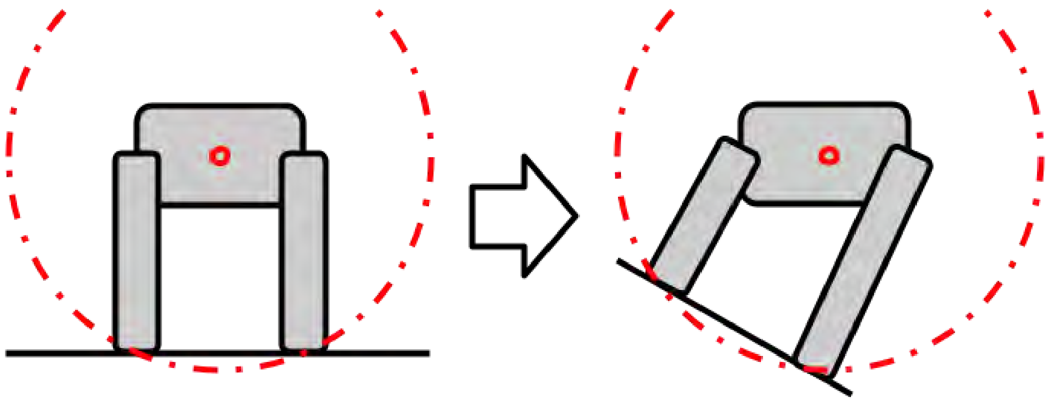


Figure 2.8: Demonstration of a body-centered rotational transform of the foot coordinates. Drawn by authors in Google Drawings.

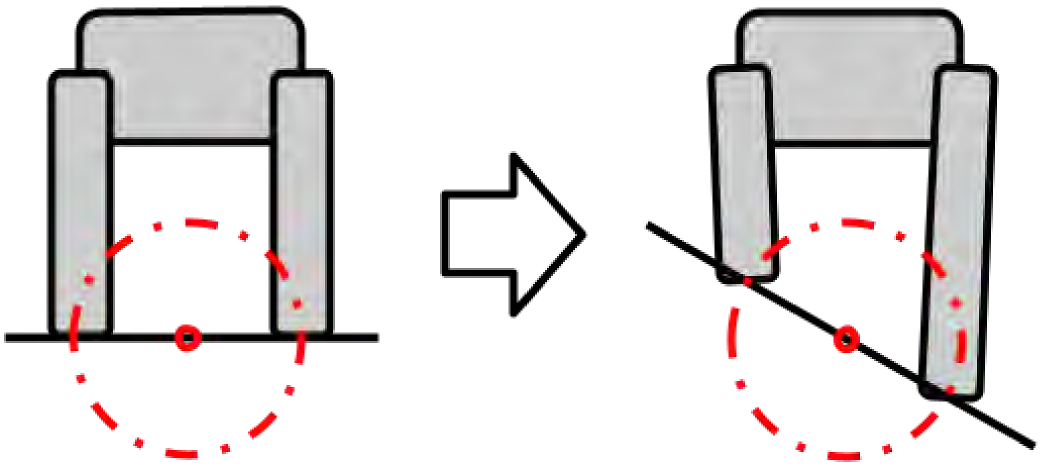


Figure 2.9: Demonstration of a base-centered rotational transform of the foot coordinates. Drawn by authors in Google Drawings.

2.4 Regulatory Controller

The problem formulation in this project is a so called ”regulatory problem”. A system output value y should be kept as close as possible to a setpoint value r, when the system is disturbed or changes. In this case the output is the body tilt, the disturbance is the tilting ground and the setpoint is the calibrated gyro value (e.g. having the body horizontal). A very common and flexible regulator is the so called PID-controller, where the abbreviation comes from the words *proportional*, *integral* and *derivative* [17]. Because of its flexibility and common use, the robot will use this kind of controller. In the regulator an error value e is calculated as:

$$e = r - y \tag{2.4}$$

The regulator output u is then calculated as the sum of the error, its time integral and its time derivative. All multiplied by scaling factors K_P, K_I and K_D. This value u is then the input to the system and gives the total system output. In this case the value u affects the amount of rotational compensation that is done each iteration of the control loop.

$$u = K_P e + K_I \int e + K_D \dot{e} \tag{2.5}$$

Chapter 3

Demonstrator

This chapter goes through the process of constructing the robot demonstrator. Specific steps are component choice, software functionality and calibration.

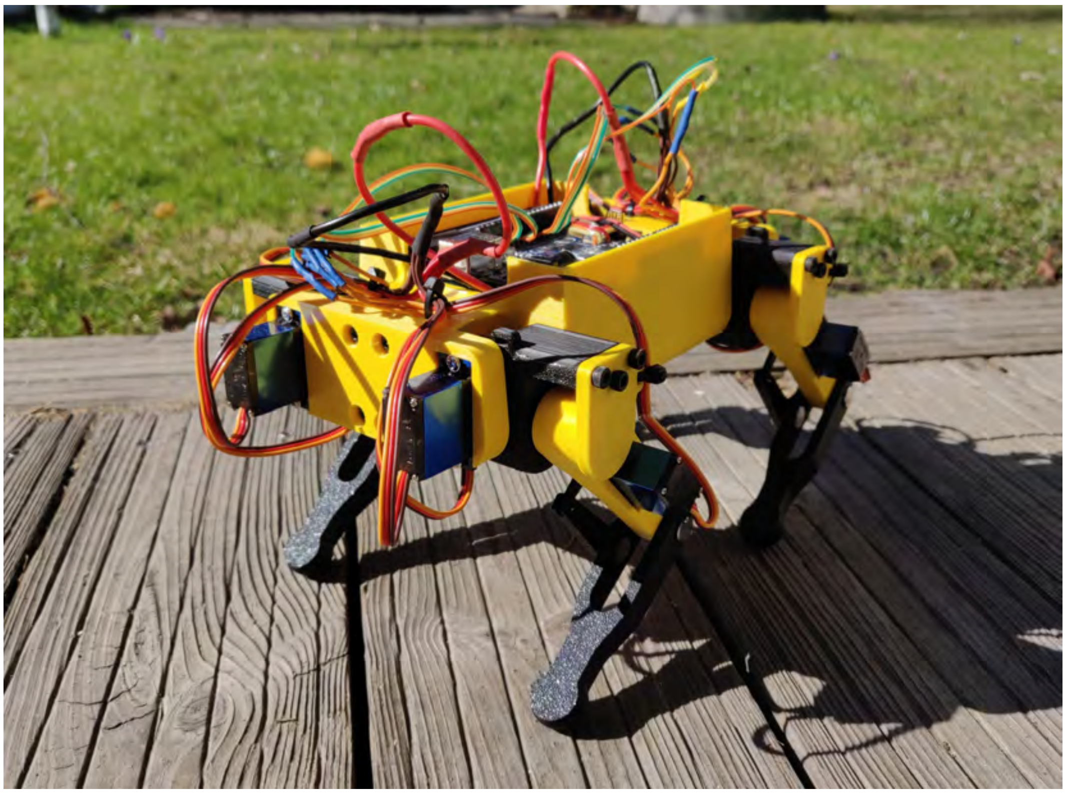


Figure 3.1: The finished demonstrator. Image taken by Algot Lindestam.

3.1 Problem formulation

Keeping balance requires great reflexes, which means that the constructed robot must check and adjust the position of its feet as often as possible. In addition, it is also essential that the motors are strong enough to lift the machine in all reachable positions. The solution to these problems can be divided into three major parts:

1. Construction
2. Components

3. Software

3.2 Construction

As discussed in section 2.1.1 quadrupedal robots of the *spider* style are inherently very stable. For this reason we chose to make our robot in the *dog* style, making stabilization more important and the goal to examine stability more relevant. We were also inspired by advanced robots such as Boston Dynamics's ”Spot”, Unitree ”A1” and the MIT ”Mini Cheetah” [18]. All being of the dog style with all knees backward.

The robot was designed in the CAD software Fusion 360 from Autodesk. Manufacture was primarily done in PLA plastic with a 3D printer for ease of quick prototyping and simplicity in making complex shapes. To reduce bulk, the knee servos were mounted inside the thigh and drive the knee joint through a 1:1 linkage. For dimensions see section 4.1.

3.3 Components

3.3.1 Micro controller

The robot is controlled by an Arduino UNO microcontroller (depicted in figure 3.2 below). The UNO has 14 digital input/output pins, six analog inputs, I2C communication capability, and an ICSP header [19]. ICSP is a technique where a programmable device is programmed after its placement on a circuit board [20]. This allows the robot to be adjusted during the time of the project and updated thereafter. In addition, the Arduino UNO can be powered by a DC source such as a battery, making the robot wireless.

Figure 3.2: Arduino UNO [19]

3.3.2 Motors

As motivated in section 2.1.3, the robot uses servo motors to drive all joints. The specific servo chosen was the *Tower Pro MG92B* [21]. For the form factor it was the strongest motor available with its 3,1 Kgcm stall torque (at 5 V) [21]. Moving to a bigger form factor and stronger motors was ruled out by budgetary restrictions. Because of this restriction, the robot was designed as small and light as possible. If it however turned out that the robot was to heavy for the motors, the plan was to move the heaviest components (batteries and DC-converter) outside the robot and supply power via an external cable. Significantly reducing robot weight. This was ultimately not needed.

3.3.3 Sensors

The only sensor used in this project is a MPU-6050, which consist of a MEMS gyroscope and accelerometer (See Figure 2.2b above). Even though the MPU-6050 consists of more inputs than just the SDA, SCL, GND and VCC, which are the only pins necessary for this project, the weight of the unused components nor the space of them are of any trouble.

3.3.4 Batteries

The first step in choosing batteries is to determine the needed voltage, given that the chosen electronics all should be driven with 5 V. A 2-series (2S) configuration of lithium cells give a nominal voltage of 7,4 V (3,6-3,7 volts per cell depending on the battery). In its discharged state, a lithium battery rests around 3 V, so even a fully discharged 2S configuration has enough voltage to supply the system. The battery voltage then needs to be stepped down to a stable 5V to avoid damage to the micro controller or other voltage sensitive electronics (see section DC-converter below).

Step two is to determine the current draw of the system. Compared to the motors, current draw by the microcontroller and gyro are negligible and were therefore not considered further. Regrettably, current draw for the MG92B was not listed by the manufacturer. By comparing with similar motors such as the SG90, the datasheet states roughly 700 mA at stall [22]. This gives a maximum current draw of 8,4 A, assuming stall loads on all 12 motors simultaneously. This is very unlikely to happen. Note however, that this is the current draw at 5 V. At the nominal battery voltage of 7,4 V the equivalent output power requires only 5,8 A at 100% transformation efficiency. Assuming a more realistic 80% efficient conversion gets the maximum current draw from the batteries to 7,3 A.

The batteries chosen were therefore the ”ICR-18650 22P” cells by Samsung. They can deliver 10 A sustained current while having a capacity of 2050 mAh with a nominal voltage of 3,6 V, enough to satisfy the voltage and current needs of the robot. As for the capacity and robot run time, simple worst-case calculations were deemed sufficient. Assuming constant maximum current draw of 7.4 A, the batteries can provide 95% relative capacity [23]. In theory, a 2S1P configuration of cells should therefore let the robot operate for a minimum of 16 minutes; sufficient for this project.

3.3.5 DC-converter

The dc converter used to step down battery voltage is a generic switching transformer capable of 8 A output. Enough to handle the calculated maximum current draw. The retailer did not list a model number and or name but a google search for ”XL4016 Step Down 8A” seems to yield the same part. For our specific retailer see [24].

3.3.6 Voltage display

Since the project is wireless and the batteries are chargeable, one could argue that it would be beneficial to connect a voltage display to the wiring diagram. This makes it possible to see the potential difference in the circuit, to estimate the power left in the batteries and avoid over-discharge.

3.4 Software

The Arduino Uno was programmed in through the Arduino IDE. Used libraries are ”Wire.h”, ”Servo.h” and ”MPU6050_light.h”, all avaliable through the IDE:s built in library manager. All code and the code of added functions can be found in Appendix C below.

The kinematics required complicated calculations for the mathematical relation between the feet positions and the angles of all joints (see Appendix A below). Therefore, to guarantee that the calculations were executed correctly, the robot was simulated in a program called *Acumen* [25]. The purpose of the simulation was to use joint angles calculated by and output from the Arduino to visualize robot movement. Doing this made it possible to double check that all calculations were working properly before running any motor-driving code on the physical robot, potentially risking self-inflicted damage to the construction (See Appendix D for simulated code).

3.5 Function

The robot acts according to the following schematic. Here we will go through each step in order and explain how the robot functions.

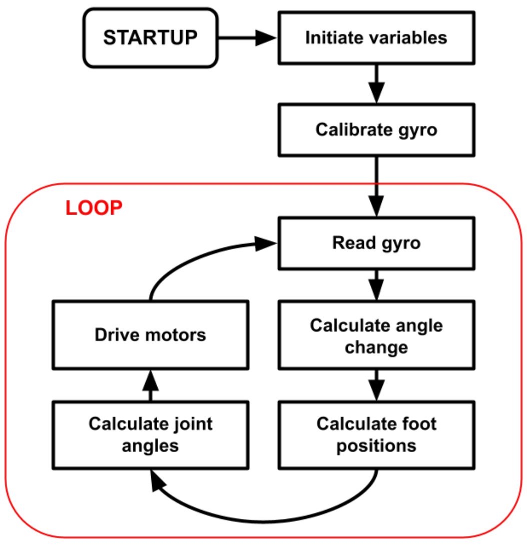


Figure 3.3: Functional overview of the robot control loop. Drawn by authors in Google Drawings.

On startup, the robot initializes all variables and assumes a base stance (in the code defined as a foot base width, length, and body height). This stance is the base for all following compensation moves. The no-slip condition set in section 2.3.2 means that the realative foot positions will not change from this base position. After taking the base position the robot calibrates its gyro, setting the reference value and the body orientation it will try to hold.

On each loop, the gyro data is read, giving the body tilt along the X, Y and Z axes respectivley. These errors are then passed through the PID- controller (section 2.4) that calculates a desired angle change of the body. The feet coordinates are thereafter rotationally transformed by this angle (section 2.3.2) and new joint angles are calculated by the inverse kinematic model (see Appendix A). The motors are then driven to these new angles and the robot corrects towards the reference value.

Pay attention to the fact that the PID-controller is reduced to a simpler P-controller in the robot code. This was done for several reasons. Firstly, the gyro data is very noisy so taking a meaningful derivative of it is challenging without much filtering. Therefore the D-term was not implemented. The I-term was not implemented either, since the results with P-control were deemed sufficient.

Note also how the inverse kinematics handle target points outside the range of motion. The equations in section 2.2.2 would in these cases give complex numbers, something that Arduino's default ”math.h” library does not support, thus returning not a number *nan*. To get around this restriction a simple check was implemented. For each leg, if any of the calculated angles are *nan*, or a joint limit would be exceeded on any joint. That leg does not update its joint angles or position.

3.6 Calibration

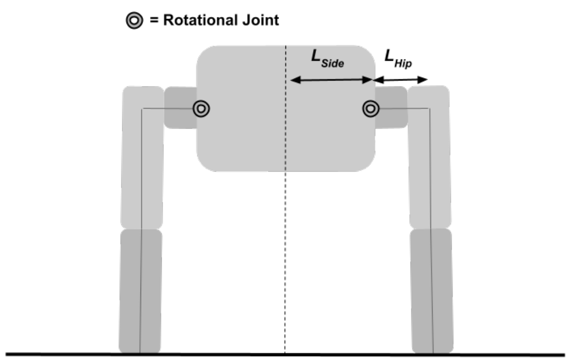
To ensure proper function all joints of the robot have to be calibrated in software. Since the servo motors vary in both mounting and rotational direction, the same PWM signal does not result in equal joint angles in all similar joints. For the robot to not damage itself, a calibration is thereby mandatory to ensure proper motion. The procedure is as following. Each joint is individually driven to its minimum and maximum angle and the corresponding PWM duty cycles are noted. These values are thereafter input into the code (see Appendix C, main). With this two-point calibration, it is possible to linearly interpolate to get the correct duty cycle value for any angle, ensuring functionality of all joints.

Chapter 4

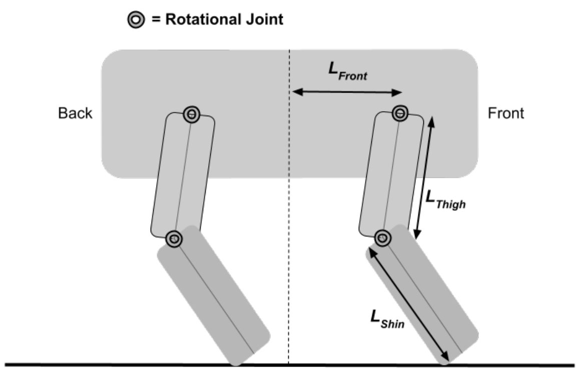
Results

4.1 Mechanical design

The construction consisted of four legs with twelve joints combined and a main body. As previously mentioned, each leg consist of one knee joint, one hip joint and one joint to make the thighs go from side to side. The specific measurements and weights previously referred to can be seen in Table 4.1 below. All limbs and main body is constructed in Polylastic acid plastic. Furthermore, each and every wire between electrical components and the micro controller is soldered on corresponding pin point on the micro computer.



(a) Back view



(b) Side view

Figure 4.1: Schematic view of the robot with joint axis and construction dimensions marked. Drawn by authors in Google Drawings.

Table 4.1: Robot parameters

Results		
Parameter	Value	Unit
L_{Hip}	25,25	[mm]
L_{Thigh}	65	[mm]
L_{Shin}	65	[mm]
L_{Shin}	65	[mm]
L_{Side}	32,5	[mm]
L_{Front}	63,5	[mm]
mass	733	[g]
$\theta_{side,min}$	-27,5	[°]
$\theta_{side,max}$	27,5	[°]
$\theta_{hip,min}$	-60	[°]
$\theta_{hip,max}$	60	[°]
$\theta_{knee,min}$	45	[°]
$\theta_{knee,max}$	135	[°]

4.2 Performance

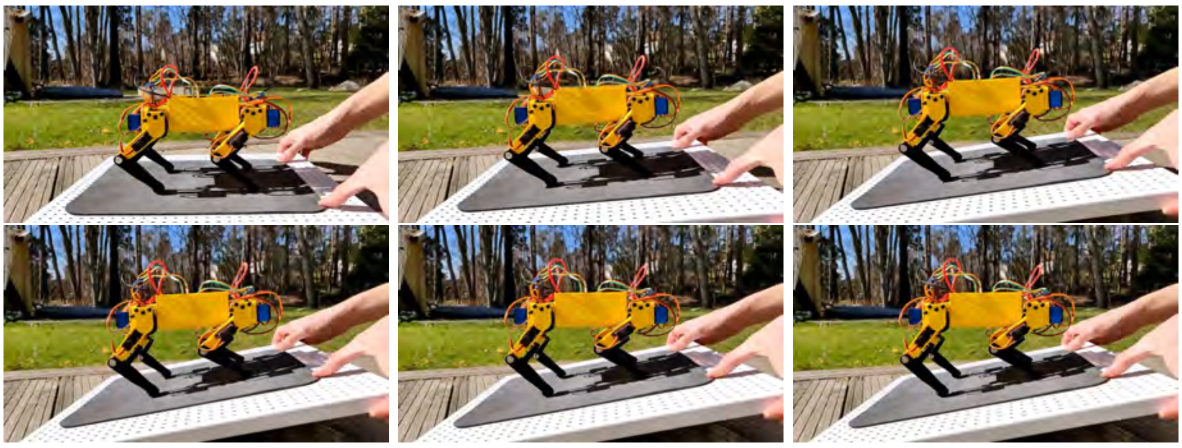


Figure 4.2: Side view of robot response. Images taken by Algot Lindestam.

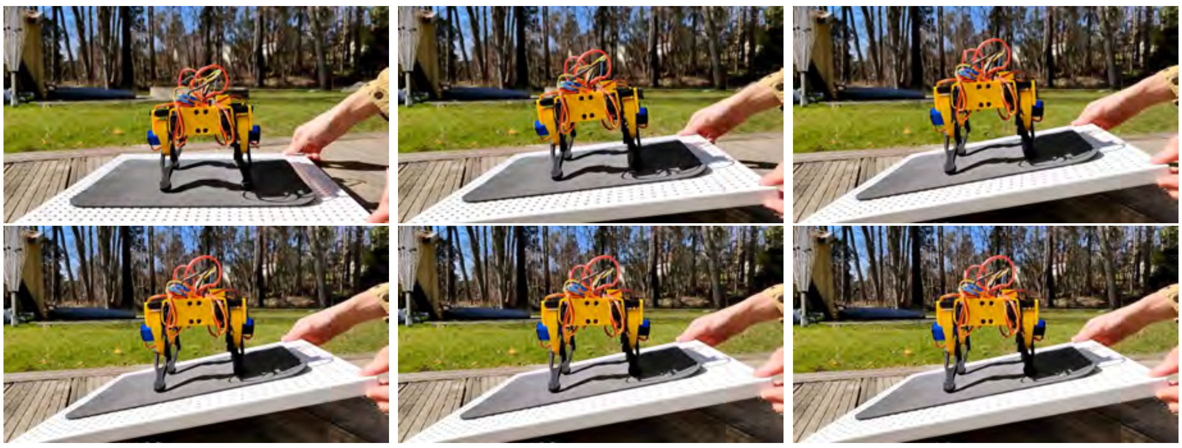


Figure 4.3: Front view of robot response. Images taken by Algot Lindestam.

As seen in Figure 4.2 and 4.3, the robot successfully corrects for a change in ground tilt. A video showcasing the performance and capabilities of the robot can be found with the same TRITA-number as this paper. The control system works for any viable base position and can correct for ground tilt with any reachable foot base (please see demonstration video). This means that the stance width, height and length can all be adjusted.

To increase the correction speed, the factor K_P was increased as much as possible without causing oscillations in the system. The used value was $0,05$.

The performance is however not optimal. Easily noticeable is the fact that the robot jitter slightly, even when not disturbed. Another undesirable trait is what happens when the robot does not have its feet planted. If held in midair and tilted, the robot will try to compensate by rotating the foot plane. But with the feet not planted this has no effect, rotating the foot plane until limits are reached.

Chapter 5

Discussion

5.1 Improvement Opportunities

5.1.1 Grip

The demonstrator has simple feet made from stiff PLA plastic, meaning that they have very little grip on hard surfaces such as wooden flooring. This results in the robot easily sliding down slopes if it is not placed on a high-grip surface. A simple solution would be to cover the feet in some sort of rubber or other high-grip material. An alternative and perhaps simpler solution is redesigning the feet to have some manner of rubber insert.

5.1.2 Center of Mass

A serious problem, is that of the center of mass. To fit all components the chassis was built in such a way that the batteries are located on the right side of the body. The weight of the batteries shifts the center of mass to the right, resulting in higher motor load on that side. The heavier loaded motors then act as a bottleneck compared to the left legs. This would not be as big a problem with stronger motors. Another effect of the off center mass is that of balance. Not having the center of mass in the middle means that it is closer to the edge of the foot-base, reducing the balance margin and robot stability. The solution would be to redesign the electronics-chassis to keep the center of mass in the middle, although this would be challenging with the size of currently used components.

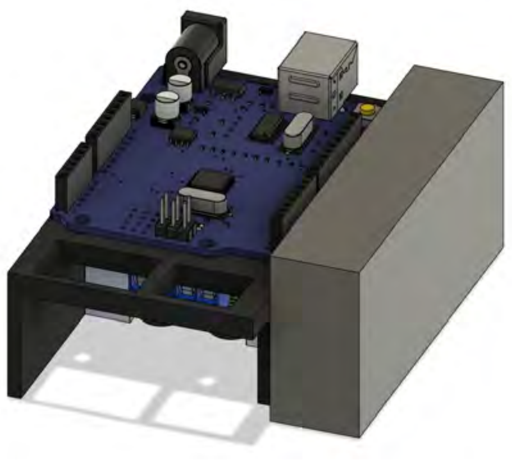


Figure 5.1: Electronics chassis CAD-model with dummy battery holder (grey) on the right. By authors via Fusion 360.

5.1.3 Motor Strength

As mentioned above, the motor strength is a limiting factor for performance. The motors are sufficiently strong for the current functionality but they would most likely hinder further development such as a implementation of walking. If possible, stronger motors are therefore desirable. This could simply be done by substituting the already existing ones, increasing the voltage in the system or adding a reduction gearing. To be able to fund better motors, or get components that handles a higher voltage, the budget for the project must be increased. Since the budget is already exceeded, this is not an improvement applicable at the moment. A negative impact of gearing would be decreased rotational speed and range of motion for each limb, even though it increases the torque in each joint.

5.1.4 Jitter

The jitter mentioned in results is likely caused by a noisy gyro signal and/or the motors not holding position exactly. While the motor performance is impossible for us to affect, the gyro signal can be improved. To reduce signal noise some manner of filtering should be implemented. The simplest implementation would be averaging multiple gyro readings, reducing the effect of random noise [26]. A filtered signal also brings back the possibility of including the D-term in the regulatory controller. This could theoretically decrease the settling time of the system [17].

5.1.5 Kinematic Modeling

Worth mentioning is also the ”locking” effect that occurs when a leg’s target point is set outside the robot’s range of motion (for example when standing on extreme inclines). The simple check described in 3.5 functions as intended, making the leg lock in place. The question is however if this is the optimal behaviour. Locking removes all capability of other corrections for the affected legs. A more sophisticated solution, where the ability to act is retained is thereby a prime subject for further investigation.

5.2 Further Development

Further development of the robot could be to include more functionality, the most obvious being the ability to walk, perhaps using some sort of autonomous navigation. This would involve further research into gait planning and implementation thereof.

A very useful addition to the robot would be pressure sensitive feet. This would provide the robot with information whether or not feet are planted and could be the basis for much improved functionality. For example, only compensating with planted feet would solve the problem mentioned in 4.2 where non-planted feet try to correct for body tilt. Furthermore it could enable functionality on non-planar terrain. By raising and lowering feet until all of them are planted, the robot could in theory function on any surface.

5.3 Conclusion

The kinematics of the robot’s legs and body could be calculated according to appendix A below. Since the position of all feet were given, all necessary angles for each limb could be calculated. These angles were essential when calculating the correlation between motor adjustment and feet adjustment.

Even though there were multiple ways to detect disturbances from the robots stable standpoint, this project’s way of using a gyroscope was a successful one. The MPU6050 is being calibrated every time the machine is being turned on and it sends out a signal of its angular coordinates with respect to earths gravitational force vector. These angles work as an input to the adjustment process.

The leg movements were calculated to keep the body at the same angle no matter the adjustment of its environment. The work was executed by adjusting the distance from the core to each foot in the three dimensional space, thus moving the body.

Bibliography

[1] D. Berreby, “The robot revolution has arrived”, *Natrional geographic*, 2020, Accessed: 27.01.2021. [Online]. Available: `https://www.nationalgeographic.com/magazine/2020/09/the-robot-revolution-has-arrived-feature/`.

[2] F. Hardarson, “Stability analysis and synthesis of statically balanced walking for quadruped robots”, Accessed: 27.01.2021, Ph.D. dissertation, KTH, 2002. [Online]. Available: `https://www.diva-portal.org/smash/get/diva2:9172/FULLTEXT01.pdf`.

[3] The Editors of Encyclopaedia Britannica, “Microcomputer”, *Britannica*, 2008, Accessed: 06.04.2021. [Online]. Available: `https://www.britannica.com/technology/microcomputer`.

[4] Helen, “Choosing the right motor for your project – dc vs stepper vs servo motors”, Accessed: 03.05.2021. [Online]. Available: `https://www.seeedstudio.com/blog/2019/04/01/choosing-the-right-motor-for-your-project-dc-vs-stepper-vs-servo-motors/`.

[5] R. Goodrich, “Accelerometer vs. gyroscope: What’s the difference?”, *LiveScience*, 2018, Accessed: 28.01.2021. [Online]. Available: `https://www.livescience.com/40103-accelerometer-vs-gyroscope.html`.

[6] J. Watson, “Mems gyroscope provides precision inertial sensing in harsh, high temperature environments”, *Analog Devices*, 2021, Accessed: 15.02.2021. [Online]. Available: `https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html`.

[7] S. Campbell, “Basics of the i2c communication protocol”, *Circuit Basics*, 2016, Accessed: 07.04.2021. [Online]. Available: `https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/`.

[8] RF Wireless World, “Advantages of gyroscope — disadvantages of gyroscope”, Accessed: 21.2.2021. [Online]. Available: `https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Gyroscope.html`.

[9] eLAB PEERS, “Mpu6050 gyroscope and accelerometer”, Accessed: 12.02.2021. [Online]. Available: `https://www.elabpeers.com/mpu6050.html`.

[10] M. Morris, “Comparison of rechargeable battery technologies”, *ASME Early Career Technical Journal*, vol. 11, pp. 148–155, Nov. 2012.

[11] Battery University, “Advantages and limitations of the different types of batteries”, Accessed: 26.04.2021. [Online]. Available: `https://batteryuniversity.com/learn/archive/whats_the_best_battery`.

[12] Arduino Store, “Pwm”, *Arduino Store*, 2021, Accessed: 24.02.2021. [Online]. Available: `https://www.arduino.cc/en/Tutorial/Foundations/PWM`.

[13] Developer Android, “Pwm”, Accessed: 11.05.2021. [Online]. Available: `https://developer.android.com/things/sdk/pio/pwm?hl=es`.

[14] P. González de Santos, "Quadrupedal locomotion an introduction to the control of four-legged robots", eng, 2006.

[15] N. Dragomir, K. Atsushi, and T. Teppei, "Humanoid robot", *ScienceDirect*, 2019, Accessed: 10.02.2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128045602000122.

[16] Y. Jia, X. Luo, B. Han, G. Liang, J. Zhao, and Y. Zhao, "Stability criterion for dynamic gaits of quadruped robot", *Applied Sciences*, vol. 8, no. 12, 2018, ISSN: 2076-3417. DOI: 10.3390/app8122381. [Online]. Available: https://www.mdpi.com/2076-3417/8/12/2381.

[17] G. Torkel and L. Lennart, *Reglerteknink, Grundläggande teori*, 4:14. Studentlitteratur AB, 2016, ISBN: ISBN=978-9144-02275-8.

[18] Boston Dynamics, "Spot®", Accessed: 27.04.2021. [Online]. Available: https://www.bostondynamics.com/spot.

[19] Arduino Store, "Arduino uno rev3", *Arduino Store*, 2021, Accessed: 24.02.2021. [Online]. Available: https://store.arduino.cc/arduino-uno-rev3.

[20] Microchip Technology Inc, "In-circuit serial programming™(icsp™) guide", *Microchip TechnologyInc*, 2003, Accessed: 12.02.2021. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/30277d.pdf.

[21] Tower Pro, "Mg92b specification", Accessed: 08.04.2021. [Online]. Available: https://www.towerpro.com.tw/product/mg92b/.

[22] Luxorparts, "Sg90 specification", Accessed: 28.04.2021. [Online]. Available: https://www.kjell.com/globalassets/mediaassets/701916_87897_datasheet_en.pdf?ref=4287817A7A.

[23] Samsung SDI Co., Ltd., "Specification of product for lithium-ion rechargeable cell model : Icr18650-22p", 2010.

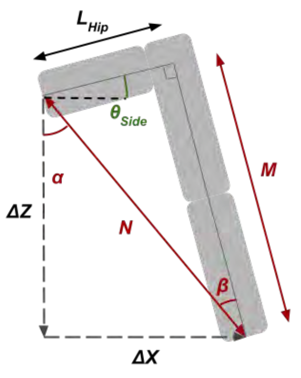
[24] Electrokit Sweden AB, "Dc-dc-omvandlare 1.2 – 35v 8a", Accessed: 10.04.2021. [Online]. Available: https://www.electrokit.com/produkt/dc-dc-omvandlare-1-2-35v-8a/.

[25] Effective Modeling Group (EMG), "Acumen", Accessed: 10.02.2021. [Online]. Available: http://www.acumen-language.org/.

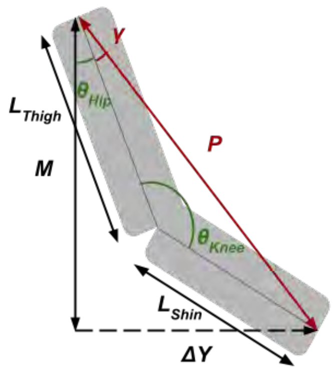
[26] B. Hans, A. Kjell, and C. Ulf, *Applied Signal Analysis.* Stockholm, Sweden: KTH Aeronautical and Vehicle Engineering, 2014.

Appendix A

Leg Inverse Kinematics



(a) Back view



(b) Side view

Figure A.1: Schematic view of the leg geometry with target coordinates and help dimensions. Given values are in black, wanted angles are in green and help measurements are in red.

Back View Calculations

We are given the target coordinates ΔX,ΔY and hip offset L_{hip}. We want to calculate the angle θ_{side}. Which is the sideways lift of the leg relative to the body. Illustrated in figure 2.5a. The Pythagorean theorem gives us an expression of the lengths N and M.

$$N = \sqrt{\Delta X^2 + \Delta Z^2} \tag{A.1}$$

$$M = \sqrt{N^2 - L_{hip}^2} = \sqrt{\Delta X^2 + \Delta Z^2 - L_{hip}^2} \tag{A.2}$$

With some geometric reasoning we can also conclude that:

$$\theta_{side} = \alpha - \beta. \tag{A.3}$$

Expressing these angles as an inverse tangent and inserting the expression for M gives us an expression for the hip angle θ_{side} using only the given variables.

$$\theta_{side} = tan^{-1}\left(\frac{\Delta X}{\Delta Z}\right) - tan^{-1}\left(\frac{L_{hip}}{\sqrt{\Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right). \tag{A.4}$$

Side View Calculations

Continuing from the back view we now continue with the side view, as seen in figure 2.5b. The given variables are the leg lengths L_{thigh}, L_{shin} and the target coordinate ΔY. Note that the Length M can also be considered given since it was calculated in the section above. The Pythagorean theorem therefore gives P.

$$P = \sqrt{\Delta Y^2 + M^2} \tag{A.5}$$

The law of cosines then give knee extension angle θ_{knee}. With expressions for P and M inserted the expression becomes:

$$\theta_{knee} = cos^{-1}\left(\frac{\Delta Y^2 + \Delta X^2 + \Delta Z^2 - L_{hip}^2 - L_{thigh}^2 - L_{shin}^2}{-2L_{thigh}L_{shin}}\right). \tag{A.6}$$

Using the law of cosines once again gives us γ.

$$\gamma = cos^{-1}\left(\frac{L_{shin}^2 - L_{thigh}^2 - P^2}{-2L_{thigh}P}\right) \tag{A.7}$$

And using the tangent of ΔY and M to calculate $\gamma + \theta_{hip}$ means that θ_{hip} can be expressed as:

$$\theta_{hip} = tan^{-1}\left(\frac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right) - cos^{-1}\left(\frac{L_{shin}^2 - L_{thigh}^2 - \Delta Y^2 - \Delta X^2 - \Delta Z^2 + L_{hip}^2}{-2L_{thigh}\sqrt{\Delta Y^2 + \Delta X^2 + \Delta Z^2 - L_{hip}^2}}\right). \tag{A.8}$$

Appendix text goes here.

Appendix B

Wiring Diagram

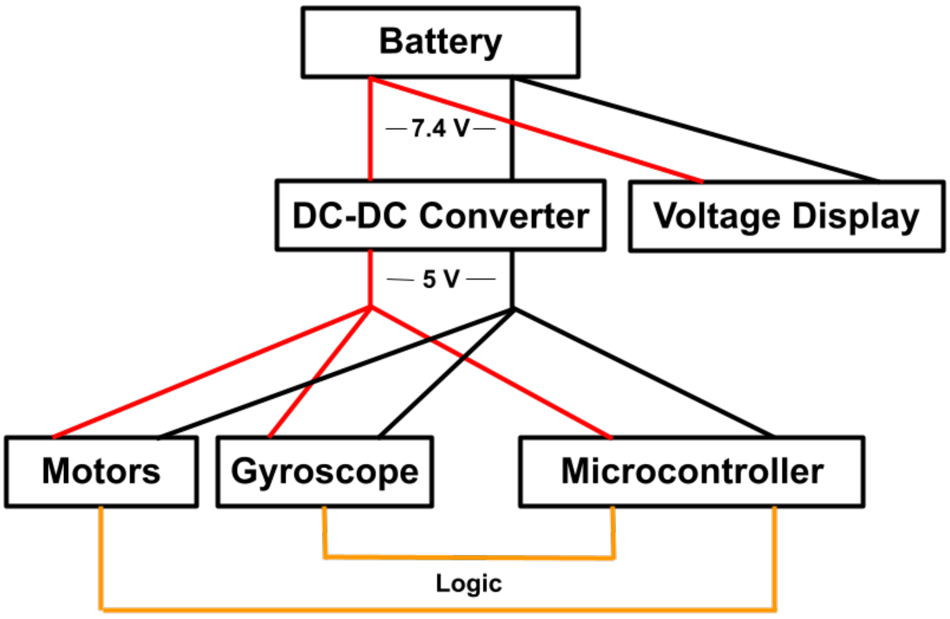


Figure B.1: Overview of the wiring between components in the robot. Motors are combined into one node for simplicity. Drawn by authors in Google Drawings.

Appendix C

Arduino Software

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// main
//
// DESCRIPTION:
// Main function for robot control using other constructed functions.
// Initiizes all required structs and variables.
// Runs the control loop of the robot.
//
// REQUIREMENTS
// Made for Arduino UNO, mpu6050 & digital RC-servos.
// Uses "Servo.h", "Wire.h", "MPU6050_light.h" libraries.


#include <Servo.h>
#include "structs.h"
#include "Wire.h"
#include <MPU6050_light.h>

// Declare all body parts
BODY body;
LEG leg1;
LEG leg2;
LEG leg3;
LEG leg4;
MPU6050 mpu(Wire);

// Declare foot target positions to ease variable handling and transfer
VEC3 pos1;
VEC3 pos2;
VEC3 pos3;
VEC3 pos4;

//Set base stance
float stanceWidth = 5.775;
float stanceLength = 6.35;
float stanceHeight = 9;

//Declare control loop parameters
float Kp = 0.05;
float xLim = 30;
float yLim = 30;
float zLim = 45;

void setup() {
  // Begin communication
  Serial.begin(57600);
  Wire.begin();

  // Initialize body structs with construction parameters and base values
  initBody( &body , 6.5,12.7 , 2.525,6.5,6.5 , -27.5,-60,45 , 30,60,135 );

  initLeg( &leg1 , &body , 1,1,1 , 1550,760,973 , 940,2020,1912 );
  initLeg( &leg2 , &body , -1,1,1 , 1320,2105,2060 , 1880,950,1220 );
  initLeg( &leg3 , &body , -1,-1,1 , 1625,1955+50,1985 , 1046,735+50,1062 );
  initLeg( &leg4 , &body , 1,-1,1 , 1295+10,865-50,950 , 1856+10,2056-50,1902 )
    ;
  // Testing gives 9.7666... us/degree of rotation. Maybe switch to single
    point calibraton?

  // Attatch servo motors to their respective pins
  leg1.servoSide.attach(11);
  leg1.servoHip.attach(12);
  leg1.servoKnee.attach(13);

  leg2.servoSide.attach(8);
  leg2.servoHip.attach(9);
  leg2.servoKnee.attach(10);

  leg3.servoSide.attach(2);
  leg3.servoHip.attach(3);
  leg3.servoKnee.attach(4);

  leg4.servoSide.attach(5);
  leg4.servoHip.attach(6);
  leg4.servoKnee.attach(7);

  //Adopt base position for all legs
  pos1.e1 = +stanceWidth;
  pos1.e2 = +stanceLength;
  pos1.e3 = -stanceHeight;

  pos2.e1 = -stanceWidth;
  pos2.e2 = +stanceLength;
  pos2.e3 = -stanceHeight;

  pos3.e1 = -stanceWidth;
  pos3.e2 = -stanceLength;
  pos3.e3 = -stanceHeight;

  pos4.e1 = +stanceWidth;
  pos4.e2 = -stanceLength;
  pos4.e3 = -stanceHeight;

  updateFootPosition( &leg1 , pos1 );
  updateFootPosition( &leg2 , pos2 );
  updateFootPosition( &leg3 , pos3 );
  updateFootPosition( &leg4 , pos4 );
  updateLegAngles( &leg1 , &body);
  updateLegAngles( &leg2 , &body);
  updateLegAngles( &leg3 , &body);
  updateLegAngles( &leg4 , &body);
  moveLeg( &leg1 , &body );
  moveLeg( &leg2 , &body );
  moveLeg( &leg3 , &body );
  moveLeg( &leg4 , &body );
  delay(1000);
  //Gyro startup and calibration. Taken from "MPU6050-light" library example
  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while(status!=0){ } // stop everything if could not connect to MPU6050

  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(1000);
  mpu.calcOffsets(); // gyro and accelero
  Serial.println("Done!\n");
}

void loop() {
  //Reads gyro data and calculates body tilt
  mpu.update();
  body.bodyTilt.e1 = -mpu.getAngleX()/180.0*PI;
  body.bodyTilt.e2 = -mpu.getAngleY()/180.0*PI;
  body.bodyTilt.e3 = mpu.getAngleZ()/180.0*PI;

  //Control loop for the foot positions
  body.footRotation.e1 = constrain( body.footRotation.e1+Kp*body.bodyTilt.e1 ,
    -xLim/180*PI , xLim/180*PI );
  body.footRotation.e2 = constrain( body.footRotation.e2+Kp*body.bodyTilt.e2 ,
    -yLim/180*PI , yLim/180*PI );
  body.footRotation.e3 = constrain( body.footRotation.e3+Kp*body.bodyTilt.e3 ,
    -zLim/180*PI , zLim/180*PI );

  //Update foot position variables with rotational transform
  updateFootPosition( &leg1 , rotatePoint(pos1,body.footRotation,stanceHeight)
    );
  updateFootPosition( &leg2 , rotatePoint(pos2,body.footRotation,stanceHeight)
    );
  updateFootPosition( &leg3 , rotatePoint(pos3,body.footRotation,stanceHeight)
    );
  updateFootPosition( &leg4 , rotatePoint(pos4,body.footRotation,stanceHeight)
    );

  //Update leg angles to reach the new positions
  updateLegAngles( &leg1 , &body);
  updateLegAngles( &leg2 , &body);
  updateLegAngles( &leg3 , &body);
  updateLegAngles( &leg4 , &body);

  //Move all legs
  moveLeg( &leg1 , &body );
  moveLeg( &leg2 , &body );
  moveLeg( &leg3 , &body );
  moveLeg( &leg4 , &body );
}
```

Listing C.1: main code

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// initBody()
//
// DESCRIPTION:
// Initializes a BODY struct at given memory adress with robot construction
    parameters.
//
// IN:
```

```
// BODY struct pointer
// float body dimensions (length and width)
// float leg dimensions (hip-, thigh-, shin-length)
// float minimum allowed angles in degrees (side, hip, knee)
// float maximum allowed angles in degrees (side, hip, knee)
//
// OUT:
// none

void initBody( BODY * body  ,  float bodyWidth,float bodyLength , float
    hipLength,float thighLength,float shinLength , float sideAngleMin,float
    hipAngleMin,float kneeAngleMin , float sideAngleMax,float hipAngleMax,float
    kneeAngleMax){
  body->bodyWidth=bodyWidth;
  body->bodyLength=bodyLength;

  body->hipLength=hipLength;
  body->thighLength=thighLength;
  body->shinLength=shinLength;

  body->sideAngleMin=sideAngleMin*PI/180;
  body->hipAngleMin=hipAngleMin*PI/180;
  body->kneeAngleMin=kneeAngleMin*PI/180;

  body->sideAngleMax=sideAngleMax*PI/180;
  body->hipAngleMax=hipAngleMax*PI/180;
  body->kneeAngleMax=kneeAngleMax*PI/180;

  body->bodyTilt.e1=0;
  body->bodyTilt.e2=0;
  body->bodyTilt.e3=0;

  body->footRotation.e1=0;
  body->footRotation.e2=0;
  body->footRotation.e3=0;
}
```

Listing C.2: initBody()

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// initLeg()
//
// DESCRIPTION:
// Initializes a LEG struct at given memory adress with robot construction
    parameters, which are partly given from a BODY struct.
//
// IN:
// LEG struct pointer
// BODY struct pointer
// inverter variables which marks legs positioned in different quadrants (x, y,
     z)
// microsecond values for the servos corresponding to minimum allowed angle (
    side, hip, knee)
// microsecond values for the servos corresponding to maximum allowed angle (
    side, hip, knee)
//
// OUT:
// none

void initLeg( LEG * leg  ,  BODY * body  ,  int xInverter,int yInverter,int
    zInverter  ,  float usSideMin,float usHipMin,float usKneeMin  ,  float
    usSideMax,float usHipMax,float usKneeMax ){
  leg->coordInverter.e1 = xInverter;
  leg->coordInverter.e2 = yInverter;
  leg->coordInverter.e3 = zInverter;

  leg->posOrigin.e1 = body->bodyWidth/2*xInverter;
  leg->posOrigin.e2 = body->bodyLength/2*yInverter;
  leg->posOrigin.e3 = 0;

  leg->posFoot.e1 = (body->bodyWidth/2+body->hipLength)*xInverter;
  leg->posFoot.e2 = (body->bodyLength/2)*yInverter;
  leg->posFoot.e3 = -1/sqrt(2)*(body->thighLength+body->shinLength);

  //Initiate angles to "safe" values to not leave the variables empty.
  leg->angles.e1 = 0;
  leg->angles.e2 = -PI/4;
  leg->angles.e3 = PI/2;

  leg->microsecondMin.e1 = usSideMin;
  leg->microsecondMin.e2 = usHipMin;
  leg->microsecondMin.e3 = usKneeMin;

  leg->microsecondMax.e1 = usSideMax;
  leg->microsecondMax.e2 = usHipMax;
  leg->microsecondMax.e3 = usKneeMax;

  //printLeg(leg);
}
```

Listing C.3: initLeg()

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// moveLeg()
//
// DESCRIPTION:
// Writes microsecond signals to the joint servos to move them to the angles
   stored in the LEG struct.
// This is done with simple linear interpolation (note that this assumes that
   the target angles are inbetween minimum and maximum).
//
// IN:
// LEG struct pointer
// BODY struct pointer
//
// OUT:
// none

void moveLeg(LEG * leg,BODY * body){

  int microsecondsSide = round( leg->microsecondMin.e1 + (leg->angles.e1 - body
   ->sideAngleMin) * (leg->microsecondMax.e1 - leg->microsecondMin.e1) / (body
   ->sideAngleMax - body->sideAngleMin) );
  int microsecondsHip = round( leg->microsecondMin.e2 + (leg->angles.e2 - body
   ->hipAngleMin) * (leg->microsecondMax.e2 - leg->microsecondMin.e2) / (body->
   hipAngleMax - body->hipAngleMin) );
  int microsecondsKnee = round( leg->microsecondMin.e3 + (leg->angles.e3 - body
   ->kneeAngleMin) * (leg->microsecondMax.e3 - leg->microsecondMin.e3) / (body
   ->kneeAngleMax - body->kneeAngleMin) );

  leg->servoSide.writeMicroseconds(microsecondsSide);
  leg->servoHip.writeMicroseconds(microsecondsHip);
  leg->servoKnee.writeMicroseconds(microsecondsKnee);

}
```

Listing C.4: moveLeg()

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// rotatePoint()
//
// DESCRIPTION:
// Applies a rotational transform on a foot point to rotate it around the
    center of the suport base.
//
// IN:
// VEC3 struct with point to rotate
// VEC3 struct with rotational angles in radians
// float with the base height of the robot stance
//
// OUT:
// VEC3 struct with rotated point

VEC3 rotatePoint( VEC3 point, VEC3 rotation,float stanceHeight){
  float x=rotation.e1;
  float y=rotation.e2;
  float z=rotation.e3;

  VEC3 out;

  //Rotate around x>y>z using a rotational "matrix" transform
  out.e1 = point.e1*(cos(z)*cos(y)) + point.e2*(cos(z)*sin(y)*sin(x)-sin(z)*cos
    (x)) + (point.e3+stanceHeight)*(cos(z)*sin(y)*cos(x)+sin(z)*sin(x));
  out.e2 = point.e1*(sin(z)*cos(y)) + point.e2*(sin(z)*sin(y)*sin(x)+cos(z)*cos
    (x)) + (point.e3+stanceHeight)*(sin(z)*sin(y)*cos(x)-cos(z)*sin(x));
  out.e3 = point.e1*(-sin(y))       + point.e2*(cos(y)*sin(x))
        + (point.e3+stanceHeight)*(cos(y)*cos(x)) - stanceHeight;

  return(out);
}
}
```

Listing C.5: rotatePoint()

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// structs.h
//
// DESCRIPTION:
// Header file holding defenitions for structs used in the project.
//

// Simple vector-3 struct to cleanly handle variable transfer and storage
struct VEC3 {
  float e1;
  float e2;
  float e3;
};
```

```

// Struct holding information relevant to the entire robot
struct BODY {
  float bodyWidth;
  float bodyLength;

  float hipLength;
  float thighLength;
  float shinLength;

  float sideAngleMin;
  float hipAngleMin;
  float kneeAngleMin;

  float sideAngleMax;
  float hipAngleMax;
  float kneeAngleMax;

  VEC3 bodyTilt;
  VEC3 footRotation;
};

// Struct holding information relevant to a specific leg of the robot
struct LEG {
  Servo servoSide;
  Servo servoHip;
  Servo servoKnee;
  VEC3 posOrigin;
  VEC3 posFoot;
  VEC3 coordInverter;
  VEC3 angles;
  VEC3 microsecondMin;
  VEC3 microsecondMax;
};
```

Listing C.6: structs.h

```
// Algot Lindestam, David Lorang
// 03/05-2021
//
// NAME:
// updateFootPosition()
//
// DESCRIPTION:
// Updates the target position variables in a given leg struct with new
   positions.
//
// IN:
// LEG struct pointer
// VEC3 struct with body-centered target position (x,y,z)
//
// OUT:
// none

void updateFootPosition(LEG * leg,VEC3 pos){
  leg->posFoot.e1 = pos.e1;
  leg->posFoot.e2 = pos.e2;
  leg->posFoot.e3 = pos.e3;
}
```

Listing C.7: updateFootPosition()

```
// Algot Lindestam, David Lorang
```

```c
// 03/05-2021
//
// NAME:
// updateLegAngles()
//
// DESCRIPTION:
// Calculates leg angles to place the foot at the target position and stores
    these in the LEG struct.
// Does not change existing leg angles if the target position is out of reach
    or would require exceeding joint limits.
//
// IN:
// LEG struct pointer
// BODY struct pointer
//
// OUT:
// none

void updateLegAngles(LEG * leg , BODY * body){
  //Calculations to get leg angles
  float dx = ( leg->posFoot.e1 - leg->posOrigin.e1 ) * leg->coordInverter.e1;
  float dy = leg->posFoot.e2 - leg->posOrigin.e2;
  float dz = - (leg->posFoot.e3 - leg->posOrigin.e3);

  float nS = pow(dx,2) + pow(dz,2);
  float mS = nS - pow(body->hipLength,2);
  float m = pow(mS,0.5);
  float pS = mS + pow(dy,2);
  float p = pow(pS,0.5);
  float thighLengthS = pow(body->thighLength,2);
  float shinLengthS = pow(body->shinLength,2);

  float side = -(atan(dx/dz) - atan(body->hipLength/m));
  float hip = atan(dy/m) - acos( (shinLengthS-thighLengthS-pS)/(-2*body->
    thighLength*p) );
  float knee = acos( (pS-thighLengthS-shinLengthS)/(-2*body->thighLength*body->
    shinLength) );

  //Check to see that calculated values are allowed
  if( isnan(side) == false &&  isnan(hip) == false && isnan(knee) == false ){
    if( side>=body->sideAngleMin && side<=body->sideAngleMax ){
      if( hip>=body->hipAngleMin && hip<=body->hipAngleMax ){
        if( knee>=body->kneeAngleMin && knee<=body->kneeAngleMax ){
          leg->angles.e1 = side;
          leg->angles.e2 = hip;
          leg->angles.e3 = knee;
        }
      }
    }
  }
}
```

Listing C.8: updateLegAngles()

Appendix D

Acumen Simulation

```
/*
 * Algot Lindestam, David Lorang
 * 23/03-2021
 *
 * Program for simulating guadruped robot movement based on a vector of joint angles and construction parameters.
 * The angles are ment to be exported from our arduino program.
 *
 */

model Main(simulator) =
initially
  _3DView = ((400,400,100), (0, 0, -50)),
  _3D=(),
  time=0,
  time'=0,

  // Matrix of joint angles, each row represents one arduino loop (step) and contains angles for all legs in radians.
  M1 = ((0.00,-1.05,1.68,0.00,-0.79,1.57,0.00,-1.05,1.68,0.00,-1.05,1.68)
    ,(0.00,-1.05,1.68,0.03,-0.79,1.56,0.03,-1.06,1.66,0.00,-1.01,1.73)
    ,(0.00,-1.04,1.67,0.06,-0.80,1.54,0.06,-1.06,1.65,0.00,-0.97,1.79)
    ,(0.00,-1.04,1.67,0.09,-0.80,1.53,0.09,-1.07,1.63,0.00,-0.93,1.85)
    ,(0.00,-1.04,1.66,0.11,-0.81,1.52,0.11,-1.07,1.62,0.00,-0.89,1.91)
    ,(0.00,-1.03,1.66,0.14,-0.81,1.51,0.14,-1.07,1.60,0.00,-0.84,1.96)
    ,(0.00,-1.02,1.65,0.17,-0.82,1.51,0.17,-1.06,1.59,0.00,-0.80,2.02)
    ,(0.00,-1.01,1.64,0.19,-0.82,1.50,0.19,-1.05,1.58,0.00,-0.76,2.07)
    ,(0.00,-1.00,1.63,0.21,-0.82,1.50,0.21,-1.04,1.56,0.00,-0.71,2.12)
    ,(0.00,-0.99,1.62,0.23,-0.82,1.50,0.23,-1.03,1.55,0.00,-0.67,2.17)
    ,(0.00,-0.97,1.61,0.25,-0.82,1.50,0.25,-1.01,1.54,0.00,-0.63,2.22)
    ,(0.00,-0.95,1.61,0.27,-0.82,1.50,0.27,-1.00,1.53,0.00,-0.58,2.27)
    ,(0.00,-0.93,1.60,0.29,-0.82,1.50,0.29,-0.98,1.52,0.00,-0.54,2.31)
    ,(0.00,-0.91,1.59,0.30,-0.82,1.50,0.30,-0.95,1.52,0.00,-0.51,2.34)
    ,(0.00,-0.89,1.58,0.31,-0.82,1.50,0.31,-0.93,1.51,0.00,-0.47,2.37)
    ,(0.00,-0.87,1.58,0.32,-0.82,1.50,0.32,-0.91,1.51,0.00,-0.44,2.40)
    ,(0.00,-0.84,1.57,0.32,-0.82,1.50,0.32,-0.88,1.50,0.00,-0.40,2.42)
    ,(0.00,-0.81,1.57,0.33,-0.82,1.50,0.33,-0.85,1.50,0.00,-0.38,2.43)
    ,(0.00,-0.79,1.57,0.33,-0.82,1.50,0.33,-0.82,1.50,0.00,-0.35,2.43)
    ,(0.00,-0.76,1.57,0.33,-0.82,1.50,0.33,-0.79,1.50,0.00,-0.33,2.43)
    ,(0.00,-0.73,1.57,0.32,-0.82,1.50,0.32,-0.76,1.50,0.00,-0.32,2.42)
    ,(0.00,-0.70,1.58,0.32,-0.82,1.50,0.32,-0.73,1.51,0.00,-0.31,2.40)
    ,(0.00,-0.67,1.58,0.31,-0.82,1.50,0.31,-0.70,1.51,0.00,-0.30,2.37)
    ,(0.00,-0.64,1.59,0.30,-0.82,1.50,0.30,-0.67,1.52,0.00,-0.29,2.34)
    ,(0.00,-0.61,1.60,0.29,-0.82,1.50,0.29,-0.64,1.52,0.00,-0.29,2.31)
    ,(0.00,-0.58,1.61,0.27,-0.82,1.50,0.27,-0.61,1.53,0.00,-0.29,2.27)
    ,(0.00,-0.56,1.61,0.25,-0.82,1.50,0.25,-0.59,1.54,0.00,-0.29,2.22)
    ,(0.00,-0.53,1.62,0.23,-0.82,1.50,0.23,-0.56,1.55,0.00,-0.30,2.17)
```

```
,(0.00,-0.51,1.63,0.21,-0.82,1.50,0.21,-0.54,1.56,0.00,-0.30,2.12)
,(0.00,-0.49,1.64,0.19,-0.82,1.50,0.19,-0.51,1.58,0.00,-0.31,2.07)
,(0.00,-0.47,1.65,0.17,-0.82,1.51,0.17,-0.49,1.59,0.00,-0.32,2.02)
,(0.00,-0.45,1.66,0.14,-0.81,1.51,0.14,-0.47,1.60,0.00,-0.33,1.96)
,(0.00,-0.44,1.66,0.11,-0.81,1.52,0.11,-0.46,1.62,0.00,-0.35,1.91)
,(0.00,-0.43,1.67,0.09,-0.80,1.53,0.09,-0.44,1.63,0.00,-0.36,1.85)
,(0.00,-0.42,1.67,0.06,-0.80,1.54,0.06,-0.43,1.65,0.00,-0.38,1.79)
,(0.00,-0.42,1.68,0.03,-0.79,1.56,0.03,-0.42,1.66,0.00,-0.40,1.73)
,(0.00,-0.42,1.68,-0.00,-0.79,1.57,-0.00,-0.42,1.68,0.00,-0.42,1.68)
,(0.00,-0.42,1.68,-0.03,-0.78,1.59,-0.03,-0.41,1.69,0.00,-0.44,1.62)
,(0.00,-0.42,1.67,-0.06,-0.77,1.60,-0.06,-0.41,1.71,0.00,-0.46,1.56)
,(0.00,-0.43,1.67,-0.08,-0.76,1.62,-0.08,-0.41,1.72,0.00,-0.49,1.51)
,(0.00,-0.44,1.66,-0.11,-0.75,1.64,-0.11,-0.41,1.74,0.00,-0.51,1.45)
,(0.00,-0.45,1.66,-0.13,-0.74,1.66,-0.13,-0.42,1.75,0.00,-0.54,1.40)
,(0.00,-0.47,1.65,-0.16,-0.73,1.69,-0.16,-0.43,1.77,0.00,-0.57,1.35)
,(0.00,-0.49,1.64,-0.18,-0.72,1.71,-0.18,-0.43,1.78,0.00,-0.60,1.30)
,(0.00,-0.51,1.63,-0.20,-0.71,1.73,-0.20,-0.45,1.79,0.00,-0.64,1.25)
,(0.00,-0.53,1.62,-0.22,-0.70,1.75,-0.22,-0.46,1.80,0.00,-0.68,1.21)
,(0.00,-0.56,1.61,-0.24,-0.69,1.77,-0.24,-0.47,1.82,0.00,-0.71,1.17)
,(0.00,-0.58,1.61,-0.25,-0.68,1.79,-0.25,-0.49,1.83,0.00,-0.76,1.13)
,(0.00,-0.61,1.60,-0.27,-0.67,1.81,-0.27,-0.51,1.83,-0.00,-0.80,1.09)
,(0.00,-0.64,1.59,-0.28,-0.66,1.82,-0.28,-0.53,1.84,0.00,-0.85,1.06)
,(0.00,-0.67,1.58,-0.29,-0.65,1.83,-0.29,-0.55,1.85,0.00,-0.89,1.04)
,(0.00,-0.70,1.58,-0.29,-0.65,1.85,-0.29,-0.57,1.85,0.00,-0.94,1.02)
,(0.00,-0.73,1.57,-0.30,-0.64,1.85,-0.30,-0.59,1.86,0.00,-0.99,1.00)
,(0.00,-0.76,1.57,-0.30,-0.64,1.86,-0.30,-0.62,1.86,0.00,-1.03,1.00)
,(0.00,-0.79,1.57,-0.30,-0.64,1.86,-0.30,-0.64,1.86,0.00,-1.07,0.99)
,(0.00,-0.81,1.57,-0.30,-0.64,1.86,-0.30,-0.67,1.86,0.00,-1.11,1.00)
,(0.00,-0.84,1.57,-0.30,-0.64,1.85,-0.30,-0.69,1.86,0.00,-1.15,1.00)
,(0.00,-0.87,1.58,-0.29,-0.65,1.85,-0.29,-0.72,1.85,0.00,-1.18,1.02)
,(0.00,-0.89,1.58,-0.29,-0.65,1.83,-0.29,-0.75,1.85,0.00,-1.21,1.04)
,(0.00,-0.91,1.59,-0.28,-0.66,1.82,-0.28,-0.77,1.84,0.00,-1.23,1.06)
,(0.00,-0.93,1.60,-0.27,-0.67,1.81,-0.27,-0.80,1.83,-0.00,-1.25,1.09)
,(0.00,-0.95,1.61,-0.25,-0.68,1.79,-0.25,-0.83,1.83,0.00,-1.26,1.13)
,(0.00,-0.97,1.61,-0.24,-0.69,1.77,-0.24,-0.85,1.82,0.00,-1.26,1.17)
,(0.00,-0.99,1.62,-0.22,-0.70,1.75,-0.22,-0.88,1.80,0.00,-1.26,1.21)
,(0.00,-1.00,1.63,-0.20,-0.71,1.73,-0.20,-0.90,1.79,0.00,-1.25,1.25)
,(0.00,-1.01,1.64,-0.18,-0.72,1.71,-0.18,-0.93,1.78,0.00,-1.24,1.30)
,(0.00,-1.02,1.65,-0.16,-0.73,1.69,-0.16,-0.95,1.77,0.00,-1.22,1.35)
,(0.00,-1.03,1.66,-0.13,-0.74,1.66,-0.13,-0.97,1.75,0.00,-1.20,1.40)
,(0.00,-1.04,1.66,-0.11,-0.75,1.64,-0.11,-0.99,1.74,0.00,-1.18,1.45)
,(0.00,-1.04,1.67,-0.08,-0.76,1.62,-0.08,-1.01,1.72,0.00,-1.15,1.51)
,(0.00,-1.04,1.67,-0.06,-0.77,1.60,-0.06,-1.02,1.71,0.00,-1.12,1.56)
,(0.00,-1.05,1.68,-0.03,-0.78,1.59,-0.03,-1.04,1.69,0.00,-1.08,1.62)
,(0.00,-1.05,1.68,0.00,-0.79,1.57,0.00,-1.05,1.68,0.00,-1.05,1.68)
,(0.00,-1.05,1.68,0.03,-0.79,1.56,0.03,-1.06,1.66,0.00,-1.01,1.73)
,(0.00,-1.04,1.67,0.06,-0.80,1.54,0.06,-1.06,1.65,0.00,-0.97,1.79)
,(0.00,-1.04,1.67,0.09,-0.80,1.53,0.09,-1.07,1.63,0.00,-0.93,1.85)
,(0.00,-1.04,1.66,0.11,-0.81,1.52,0.11,-1.07,1.62,0.00,-0.89,1.91)
,(0.00,-1.03,1.66,0.14,-0.81,1.51,0.14,-1.07,1.60,0.00,-0.84,1.96)
,(0.00,-1.02,1.65,0.17,-0.82,1.51,0.17,-1.06,1.59,0.00,-0.80,2.02)
,(0.00,-1.01,1.64,0.19,-0.82,1.50,0.19,-1.05,1.58,0.00,-0.76,2.07)
,(0.00,-1.00,1.63,0.21,-0.82,1.50,0.21,-1.04,1.56,0.00,-0.71,2.12)
,(0.00,-0.99,1.62,0.23,-0.82,1.50,0.23,-1.03,1.55,0.00,-0.67,2.17)
,(0.00,-0.97,1.61,0.25,-0.82,1.50,0.25,-1.01,1.54,0.00,-0.63,2.22)
,(0.00,-0.95,1.61,0.27,-0.82,1.50,0.27,-1.00,1.53,0.00,-0.58,2.27)
,(0.00,-0.93,1.60,0.29,-0.82,1.50,0.29,-0.98,1.52,0.00,-0.54,2.31)
,(0.00,-0.91,1.59,0.30,-0.82,1.50,0.30,-0.95,1.52,0.00,-0.51,2.34)
,(0.00,-0.89,1.58,0.31,-0.82,1.50,0.31,-0.93,1.51,0.00,-0.47,2.37)
,(0.00,-0.87,1.58,0.32,-0.82,1.50,0.32,-0.91,1.51,0.00,-0.44,2.40)
,(0.00,-0.84,1.57,0.32,-0.82,1.50,0.32,-0.88,1.50,0.00,-0.40,2.42)
,(0.00,-0.81,1.57,0.33,-0.82,1.50,0.33,-0.85,1.50,0.00,-0.38,2.43)
,(0.00,-0.79,1.57,0.33,-0.82,1.50,0.33,-0.82,1.50,0.00,-0.35,2.43)
```

```
,(0.00,-0.76,1.57,0.33,-0.82,1.50,0.33,-0.79,1.50,0.00,-0.33,2.43)
,(0.00,-0.73,1.57,0.32,-0.82,1.50,0.32,-0.76,1.50,0.00,-0.32,2.42)
,(0.00,-0.70,1.58,0.32,-0.82,1.50,0.32,-0.73,1.51,0.00,-0.31,2.40)
,(0.00,-0.67,1.58,0.31,-0.82,1.50,0.31,-0.70,1.51,0.00,-0.30,2.37)
,(0.00,-0.64,1.59,0.30,-0.82,1.50,0.30,-0.67,1.52,0.00,-0.29,2.34)
,(0.00,-0.61,1.60,0.29,-0.82,1.50,0.29,-0.64,1.52,0.00,-0.29,2.31)
,(0.00,-0.58,1.61,0.27,-0.82,1.50,0.27,-0.61,1.53,0.00,-0.29,2.27)
,(0.00,-0.56,1.61,0.25,-0.82,1.50,0.25,-0.59,1.54,0.00,-0.29,2.22)
,(0.00,-0.53,1.62,0.23,-0.82,1.50,0.23,-0.56,1.55,0.00,-0.30,2.17)
,(0.00,-0.51,1.63,0.21,-0.82,1.50,0.21,-0.54,1.56,0.00,-0.30,2.12)
,(0.00,-0.49,1.64,0.19,-0.82,1.50,0.19,-0.51,1.58,0.00,-0.31,2.07)
,(0.00,-0.47,1.65,0.17,-0.82,1.51,0.17,-0.49,1.59,0.00,-0.32,2.02)
,(0.00,-0.45,1.66,0.14,-0.81,1.51,0.14,-0.47,1.60,0.00,-0.33,1.96)
,(0.00,-0.44,1.66,0.11,-0.81,1.52,0.11,-0.46,1.62,0.00,-0.35,1.91)
,(0.00,-0.43,1.67,0.09,-0.80,1.53,0.09,-0.44,1.63,0.00,-0.36,1.85)
,(0.00,-0.42,1.67,0.06,-0.80,1.54,0.06,-0.43,1.65,0.00,-0.38,1.79)
,(0.00,-0.42,1.68,0.03,-0.79,1.56,0.03,-0.42,1.66,0.00,-0.40,1.73)
,(0.00,-0.42,1.68,0.00,-0.79,1.57,0.00,-0.42,1.68,0.00,-0.42,1.68)
,(0.00,-0.42,1.68,-0.03,-0.78,1.59,-0.03,-0.41,1.69,0.00,-0.44,1.62)
,(0.00,-0.42,1.67,-0.06,-0.77,1.60,-0.06,-0.41,1.71,0.00,-0.46,1.56)
,(0.00,-0.43,1.67,-0.08,-0.76,1.62,-0.08,-0.41,1.72,0.00,-0.49,1.51)
,(0.00,-0.44,1.66,-0.11,-0.75,1.64,-0.11,-0.41,1.74,0.00,-0.51,1.45)
,(0.00,-0.45,1.66,-0.13,-0.74,1.66,-0.13,-0.42,1.75,0.00,-0.54,1.40)
,(0.00,-0.47,1.65,-0.16,-0.73,1.69,-0.16,-0.43,1.77,0.00,-0.57,1.35)
,(0.00,-0.49,1.64,-0.18,-0.72,1.71,-0.18,-0.43,1.78,0.00,-0.60,1.30)
,(0.00,-0.51,1.63,-0.20,-0.71,1.73,-0.20,-0.45,1.79,0.00,-0.64,1.25)
,(0.00,-0.53,1.62,-0.22,-0.70,1.75,-0.22,-0.46,1.80,0.00,-0.68,1.21)
,(0.00,-0.56,1.61,-0.24,-0.69,1.77,-0.24,-0.47,1.82,0.00,-0.71,1.17)
,(0.00,-0.58,1.61,-0.25,-0.68,1.79,-0.25,-0.49,1.83,-0.00,-0.76,1.13)
,(0.00,-0.61,1.60,-0.27,-0.67,1.81,-0.27,-0.51,1.83,0.00,-0.80,1.09)
,(0.00,-0.64,1.59,-0.28,-0.66,1.82,-0.28,-0.53,1.84,0.00,-0.85,1.06)
,(0.00,-0.67,1.58,-0.29,-0.65,1.83,-0.29,-0.55,1.85,0.00,-0.89,1.04)
,(0.00,-0.70,1.58,-0.29,-0.65,1.85,-0.29,-0.57,1.85,0.00,-0.94,1.02)
,(0.00,-0.73,1.57,-0.30,-0.64,1.85,-0.30,-0.59,1.86,0.00,-0.99,1.00)
,(0.00,-0.76,1.57,-0.30,-0.64,1.86,-0.30,-0.62,1.86,0.00,-1.03,1.00)
,(0.00,-0.79,1.57,-0.30,-0.64,1.86,-0.30,-0.64,1.86,0.00,-1.07,0.99)
,(0.00,-0.81,1.57,-0.30,-0.64,1.86,-0.30,-0.67,1.86,0.00,-1.11,1.00)
,(0.00,-0.84,1.57,-0.30,-0.64,1.85,-0.30,-0.69,1.86,0.00,-1.15,1.00)
,(0.00,-0.87,1.58,-0.29,-0.65,1.85,-0.29,-0.72,1.85,0.00,-1.18,1.02)
,(0.00,-0.89,1.58,-0.29,-0.65,1.83,-0.29,-0.75,1.85,0.00,-1.21,1.04)
,(0.00,-0.91,1.59,-0.28,-0.66,1.82,-0.28,-0.77,1.84,0.00,-1.23,1.06)
,(0.00,-0.93,1.60,-0.27,-0.67,1.81,-0.27,-0.80,1.83,0.00,-1.25,1.09)
,(0.00,-0.95,1.61,-0.25,-0.68,1.79,-0.25,-0.83,1.83,0.00,-1.26,1.13)
,(0.00,-0.97,1.61,-0.24,-0.69,1.77,-0.24,-0.85,1.82,0.00,-1.26,1.17)
,(0.00,-0.99,1.62,-0.22,-0.70,1.75,-0.22,-0.88,1.80,0.00,-1.26,1.21)
,(0.00,-1.00,1.63,-0.20,-0.71,1.73,-0.20,-0.90,1.79,0.00,-1.25,1.25)
,(0.00,-1.01,1.64,-0.18,-0.72,1.71,-0.18,-0.93,1.78,0.00,-1.24,1.30)
,(0.00,-1.02,1.65,-0.16,-0.73,1.69,-0.16,-0.95,1.77,0.00,-1.22,1.35)
,(0.00,-1.03,1.66,-0.13,-0.74,1.66,-0.13,-0.97,1.75,0.00,-1.20,1.40)
,(0.00,-1.04,1.66,-0.11,-0.75,1.64,-0.11,-0.99,1.74,0.00,-1.18,1.45)
,(0.00,-1.04,1.67,-0.08,-0.76,1.62,-0.08,-1.01,1.72,0.00,-1.15,1.51)
,(0.00,-1.04,1.67,-0.06,-0.77,1.60,-0.06,-1.02,1.71,0.00,-1.12,1.56)
,(0.00,-1.05,1.68,-0.03,-0.78,1.59,-0.03,-1.04,1.69,0.00,-1.08,1.62)
,(0.00,-1.05,1.68,0.00,-0.79,1.57,0.00,-1.05,1.68,0.00,-1.05,1.68)
,(0.00,-1.05,1.68,0.03,-0.79,1.56,0.03,-1.06,1.66,0.00,-1.01,1.73)
,(0.00,-1.04,1.67,0.06,-0.80,1.54,0.06,-1.06,1.65,0.00,-0.97,1.79)
,(0.00,-1.04,1.67,0.09,-0.80,1.53,0.09,-1.07,1.63,0.00,-0.93,1.85)
,(0.00,-1.04,1.66,0.11,-0.81,1.52,0.11,-1.07,1.62,0.00,-0.89,1.91)
,(0.00,-1.03,1.66,0.14,-0.81,1.51,0.14,-1.07,1.60,0.00,-0.84,1.96)
,(0.00,-1.02,1.65,0.17,-0.82,1.51,0.17,-1.06,1.59,0.00,-0.80,2.02)
,(0.00,-1.01,1.64,0.19,-0.82,1.50,0.19,-1.05,1.58,0.00,-0.76,2.07)
,(0.00,-1.00,1.63,0.21,-0.82,1.50,0.21,-1.04,1.56,0.00,-0.71,2.12)
,(0.00,-0.99,1.62,0.23,-0.82,1.50,0.23,-1.03,1.55,0.00,-0.67,2.17)
```

```
,(0.00,-0.97,1.61,0.25,-0.82,1.50,0.25,-1.01,1.54,-0.00,-0.63,2.22)
,(0.00,-0.95,1.61,0.27,-0.82,1.50,0.27,-1.00,1.53,0.00,-0.58,2.27)
,(0.00,-0.93,1.60,0.29,-0.82,1.50,0.29,-0.98,1.52,0.00,-0.54,2.31)
,(0.00,-0.91,1.59,0.30,-0.82,1.50,0.30,-0.95,1.52,0.00,-0.51,2.34)
,(0.00,-0.89,1.58,0.31,-0.82,1.50,0.31,-0.93,1.51,0.00,-0.47,2.37)
,(0.00,-0.87,1.58,0.32,-0.82,1.50,0.32,-0.91,1.51,0.00,-0.44,2.40)
,(0.00,-0.84,1.57,0.32,-0.82,1.50,0.32,-0.88,1.50,0.00,-0.40,2.42)
,(0.00,-0.81,1.57,0.33,-0.82,1.50,0.33,-0.85,1.50,0.00,-0.38,2.43)
,(0.00,-0.79,1.57,0.33,-0.82,1.50,0.33,-0.82,1.50,0.00,-0.35,2.43)
,(0.00,-0.76,1.57,0.33,-0.82,1.50,0.33,-0.79,1.50,0.00,-0.33,2.43)
,(0.00,-0.73,1.57,0.32,-0.82,1.50,0.32,-0.76,1.50,0.00,-0.32,2.42)
,(0.00,-0.70,1.58,0.32,-0.82,1.50,0.32,-0.73,1.51,0.00,-0.31,2.40)
,(0.00,-0.67,1.58,0.31,-0.82,1.50,0.31,-0.70,1.51,0.00,-0.30,2.37)
,(0.00,-0.64,1.59,0.30,-0.82,1.50,0.30,-0.67,1.52,0.00,-0.29,2.34)
,(0.00,-0.61,1.60,0.29,-0.82,1.50,0.29,-0.64,1.52,0.00,-0.29,2.31)
,(0.00,-0.58,1.61,0.27,-0.82,1.50,0.27,-0.61,1.53,0.00,-0.29,2.27)
,(0.00,-0.56,1.61,0.25,-0.82,1.50,0.25,-0.59,1.54,0.00,-0.29,2.22)
,(0.00,-0.53,1.62,0.23,-0.82,1.50,0.23,-0.56,1.55,0.00,-0.30,2.17)
,(0.00,-0.51,1.63,0.21,-0.82,1.50,0.21,-0.54,1.56,0.00,-0.30,2.12)
,(0.00,-0.49,1.64,0.19,-0.82,1.50,0.19,-0.51,1.58,0.00,-0.31,2.07)
,(0.00,-0.47,1.65,0.17,-0.82,1.51,0.17,-0.49,1.59,0.00,-0.32,2.02)
,(0.00,-0.45,1.66,0.14,-0.81,1.51,0.14,-0.47,1.60,0.00,-0.33,1.96)
,(0.00,-0.44,1.66,0.11,-0.81,1.52,0.11,-0.46,1.62,0.00,-0.35,1.91)
,(0.00,-0.43,1.67,0.09,-0.80,1.53,0.09,-0.44,1.63,0.00,-0.36,1.85)
,(0.00,-0.42,1.67,0.06,-0.80,1.54,0.06,-0.43,1.65,0.00,-0.38,1.79)
,(0.00,-0.42,1.68,0.03,-0.79,1.56,0.03,-0.42,1.66,0.00,-0.40,1.73)
,(0.00,-0.42,1.68,-0.00,-0.79,1.57,-0.00,-0.42,1.68,0.00,-0.42,1.68)
,(0.00,-0.42,1.68,-0.03,-0.78,1.59,-0.03,-0.41,1.69,0.00,-0.44,1.62)
,(0.00,-0.42,1.67,-0.06,-0.77,1.60,-0.06,-0.41,1.71,0.00,-0.46,1.56)
,(0.00,-0.43,1.67,-0.08,-0.76,1.62,-0.08,-0.41,1.72,0.00,-0.49,1.51)
,(0.00,-0.44,1.66,-0.11,-0.75,1.64,-0.11,-0.41,1.74,0.00,-0.51,1.45)
,(0.00,-0.45,1.66,-0.13,-0.74,1.66,-0.13,-0.42,1.75,0.00,-0.54,1.40)
,(0.00,-0.47,1.65,-0.16,-0.73,1.69,-0.16,-0.43,1.77,0.00,-0.57,1.35)
,(0.00,-0.49,1.64,-0.18,-0.72,1.71,-0.18,-0.43,1.78,0.00,-0.60,1.30)
,(0.00,-0.51,1.63,-0.20,-0.71,1.73,-0.20,-0.45,1.79,0.00,-0.64,1.25)
,(0.00,-0.53,1.62,-0.22,-0.70,1.75,-0.22,-0.46,1.80,0.00,-0.68,1.21)
,(0.00,-0.56,1.61,-0.24,-0.69,1.77,-0.24,-0.47,1.82,0.00,-0.71,1.17)
,(0.00,-0.58,1.61,-0.25,-0.68,1.79,-0.25,-0.49,1.83,0.00,-0.76,1.13)
,(0.00,-0.61,1.60,-0.27,-0.67,1.81,-0.27,-0.51,1.83,-0.00,-0.80,1.09)
,(0.00,-0.64,1.59,-0.28,-0.66,1.82,-0.28,-0.53,1.84,0.00,-0.85,1.06)
,(0.00,-0.67,1.58,-0.29,-0.65,1.83,-0.29,-0.55,1.85,0.00,-0.89,1.04)
,(0.00,-0.70,1.58,-0.29,-0.65,1.85,-0.29,-0.57,1.85,0.00,-0.94,1.02)
,(0.00,-0.73,1.57,-0.30,-0.64,1.85,-0.30,-0.59,1.86,0.00,-0.99,1.00)
,(0.00,-0.76,1.57,-0.30,-0.64,1.86,-0.30,-0.62,1.86,0.00,-1.03,1.00)
,(0.00,-0.79,1.57,-0.30,-0.64,1.86,-0.30,-0.64,1.86,0.00,-1.07,0.99)
,(0.00,-0.81,1.57,-0.30,-0.64,1.86,-0.30,-0.67,1.86,0.00,-1.11,1.00)
,(0.00,-0.84,1.57,-0.30,-0.64,1.85,-0.30,-0.69,1.86,0.00,-1.15,1.00)
,(0.00,-0.87,1.58,-0.29,-0.65,1.85,-0.29,-0.72,1.85,0.00,-1.18,1.02)
,(0.00,-0.89,1.58,-0.29,-0.65,1.83,-0.29,-0.75,1.85,0.00,-1.21,1.04)
,(0.00,-0.91,1.59,-0.28,-0.66,1.82,-0.28,-0.77,1.84,0.00,-1.23,1.06)
,(0.00,-0.93,1.60,-0.27,-0.67,1.81,-0.27,-0.80,1.83,0.00,-1.25,1.09)
,(0.00,-0.95,1.61,-0.25,-0.68,1.79,-0.25,-0.83,1.83,0.00,-1.26,1.13)
,(0.00,-0.97,1.61,-0.24,-0.69,1.77,-0.24,-0.85,1.82,0.00,-1.26,1.17)
,(0.00,-0.99,1.62,-0.22,-0.70,1.75,-0.22,-0.88,1.80,0.00,-1.26,1.21)
,(0.00,-1.00,1.63,-0.20,-0.71,1.73,-0.20,-0.90,1.79,0.00,-1.25,1.25)
,(0.00,-1.01,1.64,-0.18,-0.72,1.71,-0.18,-0.93,1.78,0.00,-1.24,1.30)
,(0.00,-1.02,1.65,-0.16,-0.73,1.69,-0.16,-0.95,1.77,0.00,-1.22,1.35)
,(0.00,-1.03,1.66,-0.13,-0.74,1.66,-0.13,-0.97,1.75,0.00,-1.20,1.40)
,(0.00,-1.04,1.66,-0.11,-0.75,1.64,-0.11,-0.99,1.74,0.00,-1.18,1.45)
,(0.00,-1.04,1.67,-0.08,-0.76,1.62,-0.08,-1.01,1.72,0.00,-1.15,1.51)
,(0.00,-1.04,1.67,-0.06,-0.77,1.60,-0.06,-1.02,1.71,0.00,-1.12,1.56)
,(0.00,-1.05,1.68,-0.03,-0.78,1.59,-0.03,-1.04,1.69,0.00,-1.08,1.62)
,(0.00,-1.05,1.68,0.00,-0.79,1.57,0.00,-1.05,1.68,0.00,-1.05,1.68)
```

```
,(0.00,-1.05,1.68,0.03,-0.79,1.56,0.03,-1.06,1.66,0.00,-1.01,1.73)
,(0.00,-1.04,1.67,0.06,-0.80,1.54,0.06,-1.06,1.65,0.00,-0.97,1.79)
,(0.00,-1.04,1.67,0.09,-0.80,1.53,0.09,-1.07,1.63,0.00,-0.93,1.85)
,(0.00,-1.04,1.66,0.11,-0.81,1.52,0.11,-1.07,1.62,0.00,-0.89,1.91)
,(0.00,-1.03,1.66,0.14,-0.81,1.51,0.14,-1.07,1.60,0.00,-0.84,1.96)
,(0.00,-1.02,1.65,0.17,-0.82,1.51,0.17,-1.06,1.59,0.00,-0.80,2.02)
,(0.00,-1.01,1.64,0.19,-0.82,1.50,0.19,-1.05,1.58,0.00,-0.76,2.07)
,(0.00,-1.00,1.63,0.21,-0.82,1.50,0.21,-1.04,1.56,0.00,-0.71,2.12)
,(0.00,-0.99,1.62,0.23,-0.82,1.50,0.23,-1.03,1.55,0.00,-0.67,2.17)
,(0.00,-0.97,1.61,0.25,-0.82,1.50,0.25,-1.01,1.54,0.00,-0.63,2.22)
,(0.00,-0.95,1.61,0.27,-0.82,1.50,0.27,-1.00,1.53,0.00,-0.58,2.27)
,(0.00,-0.93,1.60,0.29,-0.82,1.50,0.29,-0.98,1.52,0.00,-0.54,2.31)
,(0.00,-0.91,1.59,0.30,-0.82,1.50,0.30,-0.95,1.52,0.00,-0.51,2.34)
,(0.00,-0.89,1.58,0.31,-0.82,1.50,0.31,-0.93,1.51,0.00,-0.47,2.37)
,(0.00,-0.87,1.58,0.32,-0.82,1.50,0.32,-0.91,1.51,0.00,-0.44,2.40)
,(0.00,-0.84,1.57,0.32,-0.82,1.50,0.32,-0.88,1.50,0.00,-0.40,2.42)
,(0.00,-0.81,1.57,0.33,-0.82,1.50,0.33,-0.85,1.50,0.00,-0.38,2.43)
,(0.00,-0.79,1.57,0.33,-0.82,1.50,0.33,-0.82,1.50,0.00,-0.35,2.43)
,(0.00,-0.76,1.57,0.33,-0.82,1.50,0.33,-0.79,1.50,0.00,-0.33,2.43)
,(0.00,-0.73,1.57,0.32,-0.82,1.50,0.32,-0.76,1.50,0.00,-0.32,2.42)
,(0.00,-0.70,1.58,0.32,-0.82,1.50,0.32,-0.73,1.51,0.00,-0.31,2.40)
,(0.00,-0.67,1.58,0.31,-0.82,1.50,0.31,-0.70,1.51,0.00,-0.30,2.37)
,(0.00,-0.64,1.59,0.30,-0.82,1.50,0.30,-0.67,1.52,0.00,-0.29,2.34)
,(0.00,-0.61,1.60,0.29,-0.82,1.50,0.29,-0.64,1.52,0.00,-0.29,2.31)
,(0.00,-0.58,1.61,0.27,-0.82,1.50,0.27,-0.61,1.53,0.00,-0.29,2.27)
,(0.00,-0.56,1.61,0.25,-0.82,1.50,0.25,-0.59,1.54,0.00,-0.29,2.22)
,(0.00,-0.53,1.62,0.23,-0.82,1.50,0.23,-0.56,1.55,0.00,-0.30,2.17)
,(0.00,-0.51,1.63,0.21,-0.82,1.50,0.21,-0.54,1.56,0.00,-0.30,2.12)
,(0.00,-0.49,1.64,0.19,-0.82,1.50,0.19,-0.51,1.58,0.00,-0.31,2.07)
,(0.00,-0.47,1.65,0.17,-0.82,1.51,0.17,-0.49,1.59,0.00,-0.32,2.02)
,(0.00,-0.45,1.66,0.14,-0.81,1.51,0.14,-0.47,1.60,0.00,-0.33,1.96)
,(0.00,-0.44,1.66,0.11,-0.81,1.52,0.11,-0.46,1.62,0.00,-0.35,1.91)
,(0.00,-0.43,1.67,0.09,-0.80,1.53,0.09,-0.44,1.63,0.00,-0.36,1.85)
,(0.00,-0.42,1.67,0.06,-0.80,1.54,0.06,-0.43,1.65,0.00,-0.38,1.79)
,(0.00,-0.42,1.68,0.03,-0.79,1.56,0.03,-0.42,1.66,0.00,-0.40,1.73)
,(0.00,-0.42,1.68,-0.00,-0.79,1.57,-0.00,-0.42,1.68,0.00,-0.42,1.68)
,(0.00,-0.42,1.68,-0.03,-0.78,1.59,-0.03,-0.41,1.69,0.00,-0.44,1.62)
,(0.00,-0.42,1.67,-0.06,-0.77,1.60,-0.06,-0.41,1.71,0.00,-0.46,1.56)
,(0.00,-0.43,1.67,-0.08,-0.76,1.62,-0.08,-0.41,1.72,0.00,-0.49,1.51)
,(0.00,-0.44,1.66,-0.11,-0.75,1.64,-0.11,-0.41,1.74,0.00,-0.51,1.45)
,(0.00,-0.45,1.66,-0.13,-0.74,1.66,-0.13,-0.42,1.75,0.00,-0.54,1.40)
,(0.00,-0.47,1.65,-0.16,-0.73,1.69,-0.16,-0.43,1.77,0.00,-0.57,1.35)
,(0.00,-0.49,1.64,-0.18,-0.72,1.71,-0.18,-0.43,1.78,0.00,-0.60,1.30)
,(0.00,-0.51,1.63,-0.20,-0.71,1.73,-0.20,-0.45,1.79,0.00,-0.64,1.25)
,(0.00,-0.53,1.62,-0.22,-0.70,1.75,-0.22,-0.46,1.80,0.00,-0.68,1.21)
,(0.00,-0.56,1.61,-0.24,-0.69,1.77,-0.24,-0.47,1.82,0.00,-0.71,1.17)
,(0.00,-0.58,1.61,-0.25,-0.68,1.79,-0.25,-0.49,1.83,-0.00,-0.76,1.13)
,(0.00,-0.61,1.60,-0.27,-0.67,1.81,-0.27,-0.51,1.83,0.00,-0.80,1.09)
,(0.00,-0.64,1.59,-0.28,-0.66,1.82,-0.28,-0.53,1.84,0.00,-0.85,1.06)
,(0.00,-0.67,1.58,-0.29,-0.65,1.83,-0.29,-0.55,1.85,0.00,-0.89,1.04)
,(0.00,-0.70,1.58,-0.29,-0.65,1.85,-0.29,-0.57,1.85,0.00,-0.94,1.02)
,(0.00,-0.73,1.57,-0.30,-0.64,1.85,-0.30,-0.59,1.86,0.00,-0.99,1.00)
,(0.00,-0.76,1.57,-0.30,-0.64,1.86,-0.30,-0.62,1.86,0.00,-1.03,1.00)
,(0.00,-0.79,1.57,-0.30,-0.64,1.86,-0.30,-0.64,1.86,0.00,-1.07,0.99)
,(0.00,-0.81,1.57,-0.30,-0.64,1.86,-0.30,-0.67,1.86,0.00,-1.11,1.00)
,(0.00,-0.84,1.57,-0.30,-0.64,1.85,-0.30,-0.69,1.86,0.00,-1.15,1.00)
,(0.00,-0.87,1.58,-0.29,-0.65,1.85,-0.29,-0.72,1.85,0.00,-1.18,1.02)
,(0.00,-0.89,1.58,-0.29,-0.65,1.83,-0.29,-0.75,1.85,0.00,-1.21,1.04)
,(0.00,-0.91,1.59,-0.28,-0.66,1.82,-0.28,-0.77,1.84,0.00,-1.23,1.06)
,(0.00,-0.93,1.60,-0.27,-0.67,1.81,-0.27,-0.80,1.83,0.00,-1.25,1.09)
,(0.00,-0.95,1.61,-0.25,-0.68,1.79,-0.25,-0.83,1.83,0.00,-1.26,1.13)
,(0.00,-0.97,1.61,-0.24,-0.69,1.77,-0.24,-0.85,1.82,0.00,-1.26,1.17)
,(0.00,-0.99,1.62,-0.22,-0.70,1.75,-0.22,-0.88,1.80,0.00,-1.26,1.21)
```

```
,(0.00,-1.00,1.63,-0.20,-0.71,1.73,-0.20,-0.90,1.79,0.00,-1.25,1.25)
,(0.00,-1.01,1.64,-0.18,-0.72,1.71,-0.18,-0.93,1.78,0.00,-1.24,1.30)
,(0.00,-1.02,1.65,-0.16,-0.73,1.69,-0.16,-0.95,1.77,0.00,-1.22,1.35)
,(0.00,-1.03,1.66,-0.13,-0.74,1.66,-0.13,-0.97,1.75,0.00,-1.20,1.40)
,(0.00,-1.04,1.66,-0.11,-0.75,1.64,-0.11,-0.99,1.74,0.00,-1.18,1.45)
,(0.00,-1.04,1.67,-0.08,-0.76,1.62,-0.08,-1.01,1.72,0.00,-1.15,1.51)
,(0.00,-1.04,1.67,-0.06,-0.77,1.60,-0.06,-1.02,1.71,-0.00,-1.12,1.56)
,(0.00,-1.05,1.68,-0.03,-0.78,1.59,-0.03,-1.04,1.69,0.00,-1.08,1.62)
,(0.00,-1.05,1.68,0.00,-0.79,1.57,0.00,-1.05,1.68,0.00,-1.05,1.68)
,(0.00,-1.05,1.68,0.03,-0.79,1.56,0.03,-1.06,1.66,0.00,-1.01,1.73)
,(0.00,-1.04,1.67,0.06,-0.80,1.54,0.06,-1.06,1.65,0.00,-0.97,1.79)
,(0.00,-1.04,1.67,0.09,-0.80,1.53,0.09,-1.07,1.63,0.00,-0.93,1.85)
,(0.00,-1.04,1.66,0.11,-0.81,1.52,0.11,-1.07,1.62,0.00,-0.89,1.91)
,(0.00,-1.03,1.66,0.14,-0.81,1.51,0.14,-1.07,1.60,0.00,-0.84,1.96)
,(0.00,-1.02,1.65,0.17,-0.82,1.51,0.17,-1.06,1.59,0.00,-0.80,2.02)
,(0.00,-1.01,1.64,0.19,-0.82,1.50,0.19,-1.05,1.58,0.00,-0.76,2.07)
,(0.00,-1.00,1.63,0.21,-0.82,1.50,0.21,-1.04,1.56,0.00,-0.71,2.12)
,(0.00,-0.99,1.62,0.23,-0.82,1.50,0.23,-1.03,1.55,0.00,-0.67,2.17)
,(0.00,-0.97,1.61,0.25,-0.82,1.50,0.25,-1.01,1.54,0.00,-0.63,2.22)
,(0.00,-0.95,1.61,0.27,-0.82,1.50,0.27,-1.00,1.53,0.00,-0.58,2.27)
,(0.00,-0.93,1.60,0.29,-0.82,1.50,0.29,-0.98,1.52,0.00,-0.54,2.31)
,(0.00,-0.91,1.59,0.30,-0.82,1.50,0.30,-0.95,1.52,0.00,-0.51,2.34)
,(0.00,-0.89,1.58,0.31,-0.82,1.50,0.31,-0.93,1.51,0.00,-0.47,2.37)
,(0.00,-0.87,1.58,0.32,-0.82,1.50,0.32,-0.91,1.51,0.00,-0.44,2.40)
,(0.00,-0.84,1.57,0.32,-0.82,1.50,0.32,-0.88,1.50,0.00,-0.40,2.42)
,(0.00,-0.81,1.57,0.33,-0.82,1.50,0.33,-0.85,1.50,0.00,-0.38,2.43)
,(0.00,-0.79,1.57,0.33,-0.82,1.50,0.33,-0.82,1.50,0.00,-0.35,2.43)
,(0.00,-0.76,1.57,0.33,-0.82,1.50,0.33,-0.79,1.50,0.00,-0.33,2.43)
,(0.00,-0.73,1.57,0.32,-0.82,1.50,0.32,-0.76,1.50,0.00,-0.32,2.42)
,(0.00,-0.70,1.58,0.32,-0.82,1.50,0.32,-0.73,1.51,0.00,-0.31,2.40)
,(0.00,-0.67,1.58,0.31,-0.82,1.50,0.31,-0.70,1.51,0.00,-0.30,2.37)
,(0.00,-0.64,1.59,0.30,-0.82,1.50,0.30,-0.67,1.52,-0.00,-0.29,2.34)
,(0.00,-0.61,1.60,0.29,-0.82,1.50,0.29,-0.64,1.52,0.00,-0.29,2.31)
,(0.00,-0.58,1.61,0.27,-0.82,1.50,0.27,-0.61,1.53,0.00,-0.29,2.27)
,(0.00,-0.56,1.61,0.25,-0.82,1.50,0.25,-0.59,1.54,0.00,-0.29,2.22)
,(0.00,-0.53,1.62,0.23,-0.82,1.50,0.23,-0.56,1.55,0.00,-0.30,2.17)
,(0.00,-0.51,1.63,0.21,-0.82,1.50,0.21,-0.54,1.56,0.00,-0.30,2.12)
,(0.00,-0.49,1.64,0.19,-0.82,1.50,0.19,-0.51,1.58,0.00,-0.31,2.07)
,(0.00,-0.47,1.65,0.17,-0.82,1.51,0.17,-0.49,1.59,0.00,-0.32,2.02)
,(0.00,-0.45,1.66,0.14,-0.81,1.51,0.14,-0.47,1.60,0.00,-0.33,1.96)
,(0.00,-0.44,1.66,0.11,-0.81,1.52,0.11,-0.46,1.62,0.00,-0.35,1.91)
,(0.00,-0.43,1.67,0.09,-0.80,1.53,0.09,-0.44,1.63,0.00,-0.36,1.85)
,(0.00,-0.42,1.67,0.06,-0.80,1.54,0.06,-0.43,1.65,0.00,-0.38,1.79)
,(0.00,-0.42,1.68,0.03,-0.79,1.56,0.03,-0.42,1.66,0.00,-0.40,1.73)
,(0.00,-0.42,1.68,0.00,-0.79,1.57,0.00,-0.42,1.68,0.00,-0.42,1.68)
,(0.00,-0.42,1.68,-0.03,-0.78,1.59,-0.03,-0.41,1.69,0.00,-0.44,1.62)
,(0.00,-0.42,1.67,-0.06,-0.77,1.60,-0.06,-0.41,1.71,0.00,-0.46,1.56)
,(0.00,-0.43,1.67,-0.08,-0.76,1.62,-0.08,-0.41,1.72,0.00,-0.49,1.51)
,(0.00,-0.44,1.66,-0.11,-0.75,1.64,-0.11,-0.41,1.74,0.00,-0.51,1.45)
,(0.00,-0.45,1.66,-0.13,-0.74,1.66,-0.13,-0.42,1.75,0.00,-0.54,1.40)
,(0.00,-0.47,1.65,-0.16,-0.73,1.69,-0.16,-0.43,1.77,0.00,-0.57,1.35)
,(0.00,-0.49,1.64,-0.18,-0.72,1.71,-0.18,-0.43,1.78,0.00,-0.60,1.30)
,(0.00,-0.51,1.63,-0.20,-0.71,1.73,-0.20,-0.45,1.79,0.00,-0.64,1.25)
,(0.00,-0.53,1.62,-0.22,-0.70,1.75,-0.22,-0.46,1.80,0.00,-0.68,1.21)
,(0.00,-0.56,1.61,-0.24,-0.69,1.77,-0.24,-0.47,1.82,0.00,-0.71,1.17)
,(0.00,-0.58,1.61,-0.25,-0.68,1.79,-0.25,-0.49,1.83,0.00,-0.76,1.13)
,(0.00,-0.61,1.60,-0.27,-0.67,1.81,-0.27,-0.51,1.83,0.00,-0.80,1.09)
,(0.00,-0.64,1.59,-0.28,-0.66,1.82,-0.28,-0.53,1.84,0.00,-0.85,1.06)
,(0.00,-0.67,1.58,-0.29,-0.65,1.83,-0.29,-0.55,1.85,0.00,-0.89,1.04)
,(0.00,-0.70,1.58,-0.29,-0.65,1.85,-0.29,-0.57,1.85,0.00,-0.94,1.02)
,(0.00,-0.73,1.57,-0.30,-0.64,1.85,-0.30,-0.59,1.86,0.00,-0.99,1.00)
,(0.00,-0.76,1.57,-0.30,-0.64,1.86,-0.30,-0.62,1.86,0.00,-1.03,1.00)
,(0.00,-0.79,1.57,-0.30,-0.64,1.86,-0.30,-0.64,1.86,0.00,-1.07,0.99)
```

```
  ,(0.00,-0.81,1.57,-0.30,-0.64,1.86,-0.30,-0.67,1.86,0.00,-1.11,1.00)
  ,(0.00,-0.84,1.57,-0.30,-0.64,1.85,-0.30,-0.69,1.86,0.00,-1.15,1.00)
  ,(0.00,-0.87,1.58,-0.29,-0.65,1.85,-0.29,-0.72,1.85,0.00,-1.18,1.02)
  ,(0.00,-0.89,1.58,-0.29,-0.65,1.83,-0.29,-0.75,1.85,0.00,-1.21,1.04)
  ,(0.00,-0.91,1.59,-0.28,-0.66,1.82,-0.28,-0.77,1.84,0.00,-1.23,1.06)
  ,(0.00,-0.93,1.60,-0.27,-0.67,1.81,-0.27,-0.80,1.83,0.00,-1.25,1.09)
  ,(0.00,-0.95,1.61,-0.25,-0.68,1.79,-0.25,-0.83,1.83,0.00,-1.26,1.13)
  ,(0.00,-0.97,1.61,-0.24,-0.69,1.77,-0.24,-0.85,1.82,0.00,-1.26,1.17)
  ,(0.00,-0.99,1.62,-0.22,-0.70,1.75,-0.22,-0.88,1.80,0.00,-1.26,1.21)
  ,(0.00,-1.00,1.63,-0.20,-0.71,1.73,-0.20,-0.90,1.79,0.00,-1.25,1.25)
  ,(0.00,-1.01,1.64,-0.18,-0.72,1.71,-0.18,-0.93,1.78,0.00,-1.24,1.30)
  ,(0.00,-1.02,1.65,-0.16,-0.73,1.69,-0.16,-0.95,1.77,0.00,-1.22,1.35)
  ,(0.00,-1.03,1.66,-0.13,-0.74,1.66,-0.13,-0.97,1.75,0.00,-1.20,1.40)
  ,(0.00,-1.04,1.66,-0.11,-0.75,1.64,-0.11,-0.99,1.74,0.00,-1.18,1.45)
  ,(0.00,-1.04,1.67,-0.08,-0.76,1.62,-0.08,-1.01,1.72,0.00,-1.15,1.51)
  ,(0.00,-1.04,1.67,-0.06,-0.77,1.60,-0.06,-1.02,1.71,0.00,-1.12,1.56)
  ,(0.00,-1.05,1.68,-0.03,-0.78,1.59,-0.03,-1.04,1.69,0.00,-1.08,1.62)
  ,(0,0,0,0,0,0,0,0,0,0,0,0)),

// Number of joint steps to take per second
movesPerSecond = 35,

// Construction parameters (should match the ones in the arduino code for
  correct behavior)
bodyLength = 60,
bodyWidth = 40,
bodyHeight = 40,
hipLength = 25,
thighLength = 65,
shinLength = 65,

// Indexing variable
targetIndex = 1,
temp = 0,

// Joint angles
sideA1 = 0,
hipA1 = -1.05,
kneeA1 = 1.68,
sideA1' = 0,
hipA1' = 0,
kneeA1' = 0,

sideA2 = 0,
hipA2 = -1.05,
kneeA2 = 1.68,
sideA2' = 0,
hipA2' = 0,
kneeA2' = 0,

sideA3 = 0,
hipA3 = -1.05,
kneeA3 = 1.68,
sideA3' = 0,
hipA3' = 0,
kneeA3' = 0,

sideA4 = 0,
hipA4 = -1.05,
kneeA4 = 1.68,
sideA4' = 0,
hipA4' = 0,
kneeA4' = 0
```

```


always
  time' = movesPerSecond,

  /*
   * If statment that checks linearly interpolates between joint steps.
   * If one step is reached, the next is set as target and the angular
    velocities updated.
   */
  if (time>=targetIndex) then
    targetIndex = temp,
    sideA1' = (M1(targetIndex,0)-M1(targetIndex-1,0))*movesPerSecond,
    hipA1' = (M1(targetIndex,1)-M1(targetIndex-1,1))*movesPerSecond,
    kneeA1' = (M1(targetIndex,2)-M1(targetIndex-1,2))*movesPerSecond,
    sideA2' = (M1(targetIndex,3)-M1(targetIndex-1,3))*movesPerSecond,
    hipA2' = (M1(targetIndex,4)-M1(targetIndex-1,4))*movesPerSecond,
    kneeA2' = (M1(targetIndex,5)-M1(targetIndex-1,5))*movesPerSecond,
    sideA3' = (M1(targetIndex,6)-M1(targetIndex-1,6))*movesPerSecond,
    hipA3' = (M1(targetIndex,7)-M1(targetIndex-1,7))*movesPerSecond,
    kneeA3' = (M1(targetIndex,8)-M1(targetIndex-1,8))*movesPerSecond,
    sideA4' = (M1(targetIndex,9)-M1(targetIndex-1,9))*movesPerSecond,
    hipA4' = (M1(targetIndex,10)-M1(targetIndex-1,10))*movesPerSecond,
    kneeA4' = (M1(targetIndex,11)-M1(targetIndex-1,11))*movesPerSecond
  else
    (
      sideA1' = (M1(targetIndex,0)-M1(targetIndex-1,0))*movesPerSecond,
      hipA1' = (M1(targetIndex,1)-M1(targetIndex-1,1))*movesPerSecond,
      kneeA1' = (M1(targetIndex,2)-M1(targetIndex-1,2))*movesPerSecond,
      sideA2' = (M1(targetIndex,3)-M1(targetIndex-1,3))*movesPerSecond,
      hipA2' = (M1(targetIndex,4)-M1(targetIndex-1,4))*movesPerSecond,
      kneeA2' = (M1(targetIndex,5)-M1(targetIndex-1,5))*movesPerSecond,
      sideA3' = (M1(targetIndex,6)-M1(targetIndex-1,6))*movesPerSecond,
      hipA3' = (M1(targetIndex,7)-M1(targetIndex-1,7))*movesPerSecond,
      kneeA3' = (M1(targetIndex,8)-M1(targetIndex-1,8))*movesPerSecond,
      sideA4' = (M1(targetIndex,9)-M1(targetIndex-1,9))*movesPerSecond,
      hipA4' = (M1(targetIndex,10)-M1(targetIndex-1,10))*movesPerSecond,
      kneeA4' = (M1(targetIndex,11)-M1(targetIndex-1,11))*movesPerSecond,
      temp = targetIndex+1
    ),

  _3D=(
    // BODY geometry
    Box
      center = (0,0,0)
      size = (bodyWidth*2,bodyLength*2+bodyHeight,bodyHeight)
      color = (255,0,0)
      rotation = (0,0,0)


    // LEG1 geometry
    Box
      center = (cos(sideA1)*hipLength/2+bodyWidth,bodyLength,-sin(sideA1)*
    hipLength/2)
      size = (hipLength,20,20)
      color = (0,255,0)
      rotation = (0,sideA1,0)
    Box
      center = ( cos(sideA1)*hipLength+bodyWidth-sin(sideA1)*cos(hipA1)*
    thighLength/2 , bodyLength+sin(hipA1)*thighLength/2 , -sin(sideA1)*hipLength
    -cos(sideA1)*cos(hipA1)*thighLength/2 )
      size = (20,20,thighLength)
      color = (0,0,255)
      rotation = (hipA1,sideA1,0)
    Box
      center = ( cos(sideA1)*hipLength+bodyWidth-sin(sideA1)*cos(hipA1)*
    thighLength-sin(sideA1)*sin(kneeA1-pi/2-hipA1)*shinLength/2 , bodyLength+sin
    (hipA1)*thighLength+cos(kneeA1-pi/2-hipA1)*shinLength/2 , -sin(sideA1)*
    hipLength-cos(sideA1)*cos(hipA1)*thighLength-cos(sideA1)*sin(kneeA1-pi/2-
    hipA1)*shinLength/2 )
      size = (20,20,shinLength)
      color = (0,255,255)
      rotation = (hipA1-kneeA1,sideA1,0)

    // LEG2 geometry
    Box
      center = (-cos(-sideA2)*hipLength/2-bodyWidth,bodyLength,sin(-sideA2)*
    hipLength/2)
      size = (hipLength,20,20)
      color = (0,255,0)
      rotation = (0,-sideA2,0)
    Box
      center = ( -cos(-sideA2)*hipLength-bodyWidth-sin(-sideA2)*cos(hipA2)*
    thighLength/2 , bodyLength+sin(hipA2)*thighLength/2 , sin(-sideA2)*hipLength
    -cos(-sideA2)*cos(hipA2)*thighLength/2 )
      size = (20,20,thighLength)
      color = (0,0,255)
      rotation = (hipA2,-sideA2,0)
    Box
      center = ( -cos(-sideA2)*hipLength-bodyWidth-sin(-sideA2)*cos(hipA2)*
    thighLength-sin(-sideA2)*sin(kneeA2-pi/2-hipA2)*shinLength/2 , bodyLength+
    sin(hipA2)*thighLength+cos(kneeA2-pi/2-hipA2)*shinLength/2 , sin(-sideA2)*
    hipLength-cos(-sideA2)*cos(hipA2)*thighLength-cos(-sideA2)*sin(kneeA2-pi/2-
    hipA2)*shinLength/2 )
      size = (20,20,shinLength)
      color = (0,255,255)
      rotation = (hipA2-kneeA2,-sideA2,0)

    // LEG 3 geometry
    Box
      center = (-cos(-sideA3)*hipLength/2-bodyWidth,-bodyLength,sin(-sideA3)*
    hipLength/2)
      size = (hipLength,20,20)
      color = (0,255,0)
      rotation = (0,-sideA3,0)
    Box
      center = ( -cos(-sideA3)*hipLength-bodyWidth-sin(-sideA3)*cos(hipA3)*
    thighLength/2 , -bodyLength+sin(hipA3)*thighLength/2 , sin(-sideA3)*
    hipLength-cos(-sideA3)*cos(hipA3)*thighLength/2 )
      size = (20,20,thighLength)
      color = (0,0,255)
      rotation = (hipA3,-sideA3,0)
    Box
      center = ( -cos(-sideA3)*hipLength-bodyWidth-sin(-sideA3)*cos(hipA3)*
    thighLength-sin(-sideA3)*sin(kneeA3-pi/2-hipA3)*shinLength/2 , -bodyLength+
    sin(hipA3)*thighLength+cos(kneeA3-pi/2-hipA3)*shinLength/2 , sin(-sideA3)*
    hipLength-cos(-sideA3)*cos(hipA3)*thighLength-cos(-sideA3)*sin(kneeA3-pi/2-
    hipA3)*shinLength/2 )
      size = (20,20,shinLength)
      color = (0,255,255)
      rotation = (hipA3-kneeA3,-sideA3,0)

    // LEG4 geometry
    Box
```

```
      center = (cos(sideA4)*hipLength/2+bodyWidth,-bodyLength,-sin(sideA4)*
    hipLength/2)
      size = (hipLength,20,20)
      color = (0,255,0)
      rotation = (0,sideA4,0)
    Box
      center = ( cos(sideA4)*hipLength+bodyWidth-sin(sideA4)*cos(hipA4)*
    thighLength/2 , -bodyLength+sin(hipA4)*thighLength/2 , -sin(sideA4)*
    hipLength-cos(sideA4)*cos(hipA4)*thighLength/2 )
      size = (20,20,thighLength)
      color = (0,0,255)
      rotation = (hipA4,sideA4,0)
    Box
      center = ( cos(sideA4)*hipLength+bodyWidth-sin(sideA4)*cos(hipA4)*
    thighLength-sin(sideA4)*sin(kneeA4-pi/2-hipA4)*shinLength/2 , -bodyLength+
    sin(hipA4)*thighLength+cos(kneeA4-pi/2-hipA4)*shinLength/2 , -sin(sideA4)*
    hipLength-cos(sideA4)*cos(hipA4)*thighLength-cos(sideA4)*sin(kneeA4-pi/2-
    hipA4)*shinLength/2 )
      size = (20,20,shinLength)
      color = (0,255,255)
      rotation = (hipA4-kneeA4,sideA4,0)
 )
```
Listing D.1: Acumen Simulation

TRITA ITM-EX 2021:33