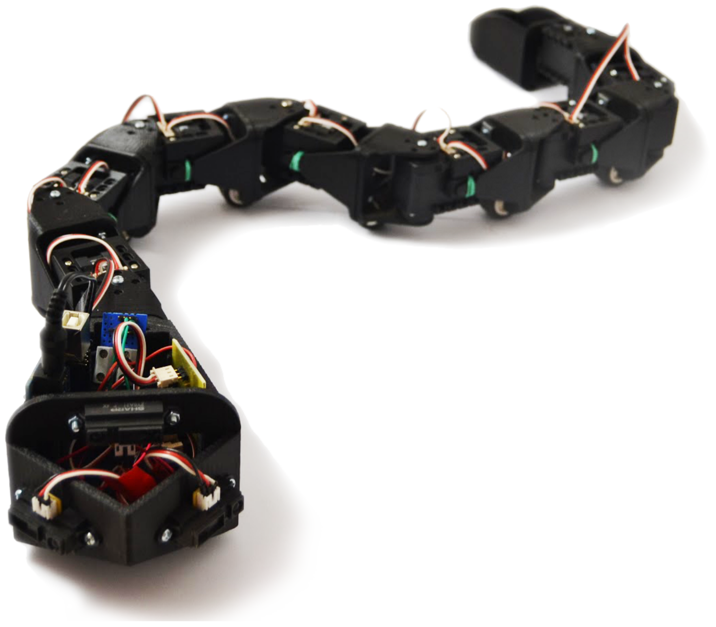
CHALMERS
UNIVERSITY OF TECHNOLOGY



Kuggorm: Design and Construction of a Snake-Like Robot

Report

Bachelor's thesis

FREDRIK BJERSING, DAVID EKSTRÖM, HERMAN HÖRNSTEIN, JAKOB BRAMSTÅNG, ANTON ÖQVIST, VIKTOR JOHANSSON

Department of Signals and systems (S2)
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Bachelor's thesis 2015:SSYX02-15-26

Report for the project Kuggorm: Design and Construction of a Snake-Like Robot

Fredrik Bjersing, David Ekström, Herman Hörnstein,
Jakob Bramstång, Anton Öqvist, Viktor Johansson

Department of Signals and Systems
Division of Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2015

Report for the project Kuggorm: Design and Construction of a Snake-Like Robot

Fredrik Bjersing, David Ekström, Herman Hörnstein, Jakob Bramstång, Anton Öqvist, Viktor Johansson

Supervisor: SeyedMehrdad Hosseini, Department of Signals and Systems
Examiner: Jonas Fredriksson, Department of Signals and Systems

Bachelor's Thesis 2015:SSYX02-15-26
Department of Signals and Systems (S2)
Division of Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Picture of the final product.

Typeset in LaTeX
Gothenburg, Sweden 2015

Report for the project Kuggorm: Design and Construction of a Snake-Like Robot

Fredrik Bjersing, David Ekström, Herman Hörnstein, Jakob Bramstång, Anton Öqvist, Viktor Johansson
Department of Signals and Systems
Chalmers University of Technology

Abstract

The project presented in this report aims to explore and mimic the movement of a snake in order to gain its advantages, both in theory and in practice. The purpose of this is to investigate how the motion of snakes can be used for robotic movement.

The project was conducted at the Department of Signals and Systems at Chalmers University of Technology during the spring of 2015. The report includes the mathematical model and simulations made on the most common movement pattern of the snakes, lateral undulation, together with the choice of hardware and software development based on this model. This work resulted in a robot that resembled both the appearance and characteristics of a real snake. The result provides an insight into the possibilities and significance of snake-like robotic movement. The movement pattern of the robot did not surpass already existing robots but proved to have great potential for further development. Due to the time limit of four months the movement of the robot is limited to only lateral undulation on flat surfaces and in slopes with an inclination between zero and ten degrees.

Keywords: lateral, undulation, lateral undulation, snake, robot, snake-like.

Preface

This report together with the produced robot are a part of a bachelor project conducted during the spring of 2015 at the Department of Signals and Systems at Chalmers University of Technology.

Special thanks goes out to the supervisor SeyedMehrdad Hosseini for his continuous support during the project. His feedback and suggestions was greatly appreciated.

Thanks to David Pantzar for helping the group with the editing of several pictures in the report.

Contents

1 Introduction

The project presented in this report revolves around the studying of the snake movement and the implementation of that very movement into a snake-like robot. This chapter of the report introduce some of the greatest difficulties concerning the implementation while also explaining why the snake's motion is interesting to examine.

1.1 Background

Ever since humans started building robots, the problem of robotic movement has been an issue. A robot must often be able to move a certain distance and sometimes through a suboptimal environment. The conventional method of transporting a robot from one location to the other often involves wheels as the tool for generating propulsion. These types of robots, despite their several benefits, struggle in narrow and debris filled environments. Therefore, investigating a different kind of movement could be valuable.

Finding a new method of moving might be difficult, whereas observing an already existing one could potentially be more rational and less time consuming. One could then replicate the movement and build a robot focusing on that very motion. This could cause the robot to gain the same benefits. One type of movement, which does not struggle with the ability to manoeuvre through different environments and narrow spaces is the locomotion of the snake and is therefore an interesting movement pattern to observe.

1.1.1 The snake

The snake is a limbless reptile that is commonly noticed by its particular way of moving, where it moves forward by performing different kinds of side-to-side motions. Where the most common one is called lateral undulation, [1, p. 8].

According to Encyclopedia Britannica, [2], the existence of the snake dates back several million years, where the oldest fossil ever found has been calculated to be around 167 million years old. The evolution of snakes proves that its behaviour and its way of moving is somehow advantageous, otherwise it would probably not exist today.

1.1.2 Lateral undulation

Lateral undulation, also known as serpentine crawling, is the most common movement amongst the snake species. As described in [1, p. 8], the muscles around the spine of the snakes make up a wavelike motion which propagates backward along the body of the snake. The sides of the snakes body pushes against irregularities on the ground and exerting a force perpendicular to its current position. This motion can be observed in the figure 1.1 where the arrows illustrate the force exerted to the sides that produces the resulting force forward.

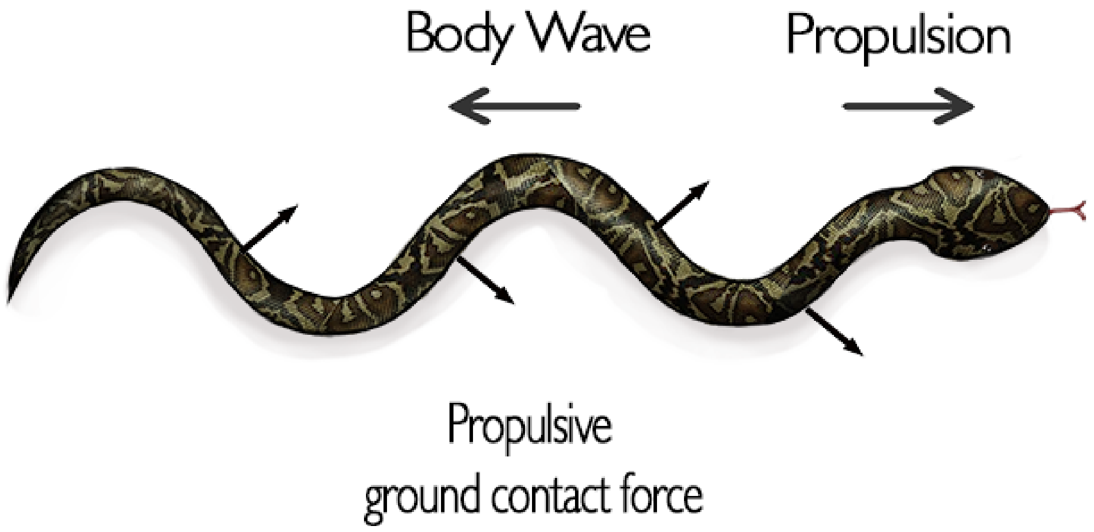


Figure 1.1: The most common snake movement called lateral undulation. The picture describes the propulsion generated by the ground contact forces.

1.1.3 Practical applications

Several earlier projects have been carried out to investigate the movement of the snake and implementing it into a robot, but few projects have resulted in a practical application. Finding a constructive function for a snake-like robot is however a task that does not come naturally, since today's infrastructure promotes wheel driven robots and/or flying robots to a major extent. Often robots operate in known closed environments, the robot can then be designed for that specific surrounding. However in some situations there might not be an optimal surface to move around on and thusly other methods of moving becomes more attractive. There have also been several research projects conducted on practical applications for snake-like robots, where the focus was not solely based on the movement. In the article SnakeFighter - Development of a Water Hydraulic Fire Fighting Snake Robot [3] the concept of having snake-like robots extinguishing fires is studied. The article discuss the potential of equipping a snake-like robot with a water hose and sending it into dangerous areas. The shape of the robot resembles one of a water hose, which can be utilised when putting out fires.

Another area where a snake-like robot could potentially serve, where other robots might not be as optimal, is in search and rescue missions in areas where earthquakes are frequent.

[1] describes these types of missions as optimal for snake-like robots, since the robot could use small narrow paths in the demolished buildings to find trapped humans. The conventional method for such rescue missions often involves removing debris and potentially harming trapped humans in the process. A snake-like robot could then perform the task with less risk of hurting humans.

1.2 Purpose

The purpose of this project is to build a snake-like robot and investigate how the movement of snakes can be used for robotic movement. The goal is to build a robot that mimics the movement and appearance of a real snake in order to gain its advantages.

1.3 Problem description

The definition of a snake-like robot is indistinct and depends on which properties of the snake that is prioritised to be replicated. In this project the movement pattern called lateral undulation is to be transferred into a robot, while the focus on other properties have lower priority. In this section the definition of snake-like in the sense of this project will be clarified.

1.3.1 Task

The task of this project is to build a robot which inherits the properties of a real snake. Which implies that the robot should be snake-like in the sense that its movement and appearance will represent a snake as much as possible. This means that it must be pin-jointed and move solely by a slithering motion produced by torque actuators between each segment in the robot. Moreover, it should move and make its own decisions without any human interference. More detailed demands of the robot's performance can be seen in the following section 1.3.2.

1.3.2 Demands

The robot must meet the following demands:

1. Have the visual representation of a snake.
2. Move forward using the same technique of propulsion as a snake, i.e. lateral undulation.
3. Produce its propelling force solely by twisting the segments.
4. Move through a predefined course by itself without any human control using wall detection and collision avoidance.
5. Manage to climb up and down slopes with 10° inclination.

6. Be possible to build within a time limit of 4 month period and a budget of 5000 SEK.

To verify that these demands are fulfilled the robot must pass a series of tests described in appendix A.1.

1.3.3 Problem formulation

In order to replicate the movement of a real snake a great deal of understanding in the lateral undulation is needed. The muscular body wave that a real snake produces must be imitated by a mechanical wave in the robot. This requires hardware and design that can manage the slithering motion but also a controller with an algorithm that manage to create a propagating wave through the segments of the robot.

Moreover the algorithm for the autonomous steering must be developed and tested while suitable visual input for obstacle detection must be found and implemented in the robot.

1.3.4 Subproblems

The problems above can be broken down into three major subproblems that explore the above problem formulation in more details. Each subproblem has its own chapter in the report.

The first subproblem is to create a mathematical model that can be used to simulate the lateral undulation of a robot in a simulation environment so the motion can be studied in detail. See chapter 2.

Another subproblem is to choose the hardware and components to use in the construction as well as creating a design so that the robot meets the demands on the slithering motion and visual representation. See chapter 3.

Lastly, the design of electrical circuits for the hardware communication and development of the software for control of both the lateral undulation and autonomous steering, is a subproblem on its own. See chapter 4.

1.3.5 Boundaries

The project should be conducted during a four month period where the resources are limited. The project has a budget consisting of 5000 SEK which is to be distributed towards components, material and production costs. The project also has access to materials which resided in the Prototype lab managed by Chalmers University of Technology. In this lab, materials can be acquired up to a sum of 2000 SEK.

Due to the time and budget limitations, the spectrum of the project is limited. The snake-like robot should only be designed to achieve the demands presented in 1.3.2. Therefor, any other type of snake-like movement than lateral undulation will not be in the spectrum of this project.

1.4 Method

The project has been conducted through an iterative method. The objective of this process is to successively develop the product and bring the desired result nearer with each iteration. The first step in this process is to understand the lateral undulation through simulations and mathematical modelling of the problem. The results of this is then used as inspiration and guidelines to the design prototypes and control algorithms that was continuously developed along with further simulations and testing during the project. This iteration process eventually result in a final product that can be built and tested.

2

Modelling and simulation

The general purpose of the mathematical modelling and simulation is to create deeper understanding of how the lateral undulation works and how the motion forward is created and controlled. By creating a kinematic and dynamic model of the robot, complete simulations of the movement are possible and changes in the movement and properties of the robot can thoroughly be tested before the implementation. The knowledge from this tests is useful throughout the project when constructing the robot, choosing the components and programming the controller.

Moreover, the simulations is used to run different algorithms for the autonomous steering.

2.1 Demands

The modelling and simulation should fulfil the following demands:

1. Allow simulations of the movement, so that different settings in the robot control structure and in the reference signal can be tested and evaluated before testings on the actual robot.
2. Answer questions about different design and hardware demands, like how strong the motors must be and how properties like weight is affecting the movement.
3. Simulating the autonomous steering so that different algorithms for the autonomous steering can be tested.

2.2 Construction of mathematical modelling and simulation

The mathematical model for the planar snake movement used in this project is the same as in [1], both when it comes to design and notation. The authors of [1] have likewise used earlier work from two different authors in order to develop their model, so the actual dynamic model dates back to 2001. The corrections of the model for movement in a slope is however solely the work of this project.

All the calculations for the simulation are made using *matlab* and the visual presentations of the simulations are made either by the plotting tool in *matlab* or in a separate display tool written in C++ by the project group.

2.2.1 Notation

A summary of the notations used in the model can be seen in table 2.1. Each segment (link) in the robot, is viewed as a rod of length $2l$ with a centre of mass located in the middle of the rod and one angle actuator located in the front. The head lacks the angle actuator in the front but is otherwise a replication of the other links. All links are oriented with an angle towards the global x-axis of the room called link angle $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_N)^T \in \mathbb{R}^N$ where N is the number of links in the robot and link one is the tail, therefore having the angle θ_1. The angle actuators give each link control of its angular orientation in relation to the link in the front of it, also called joint angle $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_{N-1})^T \in \mathbb{R}^{N-1}$. Note that since the head does not have a servo the head lack the angle ϕ_N and $\boldsymbol{\phi}$ therefore is N-1 elements long. Moreover the model uses two different coordinate systems as can be seen in figure 2.1, the global, and the link fixed system. The centre of mass for each segment is located at (x_i, y_i) in the global frame and the centre of mass for the whole robot is located in (p_x, p_y) which will also be expressed with the vectors $\boldsymbol{X} = (x_1, x_2, ..., x_N)^T \in \mathbb{R}^N$, $\boldsymbol{Y} = (y_1, y_2, ..., y_N)^T \in \mathbb{R}^N$ and $\boldsymbol{p} = (p_x, p_y)^T \in \mathbb{R}^2$.

Each link are subject to two forces, the friction force from the centre of mass of the link $f_{R,i}$ and the constraining force h_i and h_{i-1} that arises due to the fact that the link is connected to the leading and preceding link. Moreover each link is affected by the torques u_i and u_{i-1} from the leading and preceding links. The forces will also be expressed in vector form as $\boldsymbol{f}_{R,x} = (f_{R,x,1}, f_{R,x,2}, ..., f_{R,x,N})^T, \boldsymbol{f}_{R,y} = (f_{R,y,1}, f_{R,y,2}, ..., f_{R,y,N})^T \in \mathbb{R}^N$ and $\boldsymbol{h}_x = (h_{x,1}, h_{x,2}, ..., h_{x,N-1})^T, \boldsymbol{h}_y = (h_{y,1}, h_{y,2}, ..., h_{y,N-1})^T \in \mathbb{R}^{N-1}$, note that the head link lacks the constraint force from a leading link and the tail link lacks the constraint force of a preceding link. The vectors $\boldsymbol{h}$ is therefore $N-1$ long. The torque will be expressed as the vector $\boldsymbol{u} = (u_1, u_2, ..., u_{N-1})^T \in \mathbb{R}^{N-1}$. The forces and torque on each link will be explained in more detail in 2.2.4.

2.2.2 Kinematic model

The movement of the snake-like robot can be expressed by deriving the kinematic equations.

Since the snake moves in two directions and has N number of link angles, $\boldsymbol{\theta} \in \mathbb{R}^N$, it has $N+2$ degrees of freedom. Because of the constant twisting of the links it is hard to tell the actual direction of the robot. However, the orientation of the robot can be defined as

$$\theta_{heading} = \frac{1}{N}\sum_{i=1}^{N}\theta_i, \tag{2.1}$$

which is the average link angle.

The position for the centre of mass of the robot is dependent on all the link positions and their masses according to

$$\boldsymbol{p} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \frac{1}{N}\sum_{i=0}^{N}\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \frac{1}{N}\begin{pmatrix} \boldsymbol{e}^T\boldsymbol{X} \\ \boldsymbol{e}^T\boldsymbol{Y} \end{pmatrix} \tag{2.2}$$

where $\boldsymbol{e} = (1, 1, ..., 1)^T \in \mathbb{R}^N$ and therefore works as a summation operator.

Table 2.1: Parameters and variables that will be used to describe the model.

Symbol	**Description**
N	Number of links in robot
l	Half the length of links
m	Mass of links
J	Moment of inertia for each link
$\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_N)^T \in \mathbb{R}^N$	Angle from each link to the global x-axis
$\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_{N-1})^T \in \mathbb{R}^{N-1}$	Angle of each link to the link in front of it
$\boldsymbol{X} = (x_1, x_2, ..., x_N)^T \in \mathbb{R}^N$	Position of each link in global x-direction
$\boldsymbol{Y} = (y_1, y_2, ..., y_N)^T \in \mathbb{R}^N$	Position of each link in global y-direction
$\boldsymbol{p} = (p_x, p_y)^T \in \mathbb{R}^2$	Position of centre of mass for the whole robot
$\boldsymbol{u} = (u_1, u_2, ..., u_{N-1})^T \in \mathbb{R}^{N-1}$	Torque on each link u_i exerted on link i from link $i+1$ and u_{i-1} is exerted on link i from link $i-1$
$\boldsymbol{f}_{R,x} = (f_{R,x,1}, f_{R,x,2}, ..., f_{R,x,N})^T \in \mathbb{R}^N$	Friction force in global x-direction on each link
$\boldsymbol{f}_{R,y} = (f_{R,y,1}, f_{R,y,2}, ..., f_{R,y,N})^T \in \mathbb{R}^N$	Friction force in global y-direction on each link
$\boldsymbol{h}_x = (h_{x,1}, h_{x,2}, ..., h_{x,N-1})^T \in \mathbb{R}^{N-1}$	Constraint force on each link in the global x-direction h_i exerted on link i from link $i+1$ and h_{i-1} is exerted on link i from link $i-1$
$\boldsymbol{h}_y = (h_{y,1}, h_{y,2}, ..., h_{y,N-1})^T \in \mathbb{R}^{N-1}$	Constraint force on each link in the global y-direction h_i exerted on link i from link $i+1$ and h_{i-1} is exerted on link i from link $i-1$
$\boldsymbol{f}_{drag,x} = f_{drag,x,1}, ..., f_{drag,x,N})^t \in \mathbb{R}^N$	Force exerted by gravity that drags each segment in x-direction while in a slope
$\boldsymbol{f}_{drag,y} = (f_{drag,y,1}, ..., f_{drag,y,N})^T \in \mathbb{R}^N$	Force exerted by gravity that drags each segment in y-direction while in a slope

Since the links are connected to each other the positions of the links must obey the following constraints

$$x_{i-1} - x_i = lcos\theta_i + lcos\theta_{i+1} \tag{2.3}$$
$$y_{i-1} - y_i = lsin\theta_i + lsin\theta_{i+1}. \tag{2.4}$$

This can be written in matrix-form as

$$\boldsymbol{DX} = -l\boldsymbol{A}cos\boldsymbol{\theta} \tag{2.5}$$
$$\boldsymbol{DY} = -l\boldsymbol{A}sin\boldsymbol{\theta} \tag{2.6}$$

where $sin\boldsymbol{\theta} = (sin\theta_1, sin\theta_2, ..., sin\theta_N)^T$ and $cos\boldsymbol{\theta} = (cos\theta_1, cos\theta_2, ..., cos\theta_N)^T$ and $\boldsymbol{D} = \begin{pmatrix} 1 & -1 & & \\ & \cdot & \cdot & \\ & & \cdot & \cdot \\ & & 1 & -1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times N}$ works as a differential operator that subtracts pairs

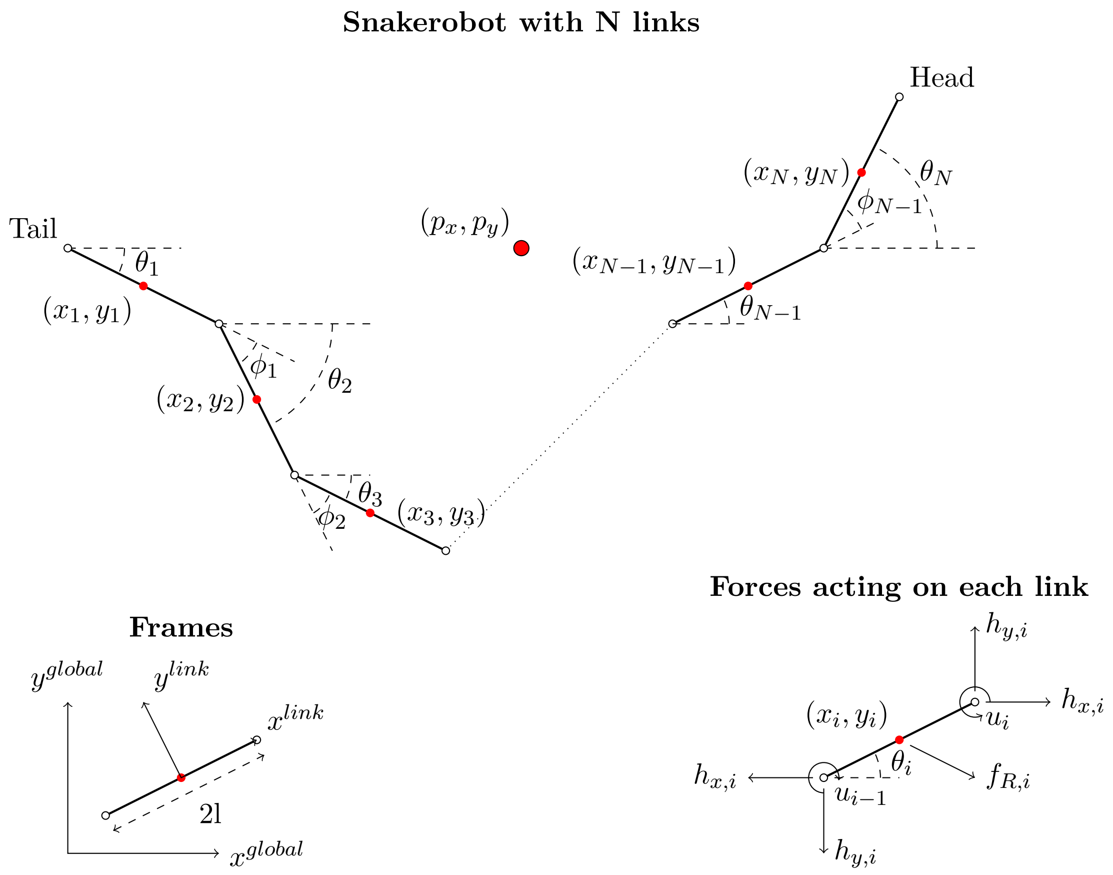


Figure 2.1: A schematic view of the model. The links is numbered by 1 to N where the 1:th is the tail, N is the number of links and the N:th link is the head link. The two different frames can be seen in the small picture in the bottom left corner and the forces on each link can be seen in the bottom right corner.

of neighbouring links and $\boldsymbol{A} = \begin{pmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ & & & 1 & 1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times N}$ as a summation operator that sums pairs of neighbouring links.

The position of the centre of mass can now be expressed as the function of the position of each link as

$$\begin{pmatrix} \boldsymbol{D} \\ \frac{1}{N}\boldsymbol{e}^T \end{pmatrix} \boldsymbol{X} = \begin{pmatrix} -l\boldsymbol{A}cos\boldsymbol{\theta} \\ p_x \end{pmatrix} \tag{2.7}$$

$$\begin{pmatrix} \boldsymbol{D} \\ \frac{1}{N}\boldsymbol{e}^T \end{pmatrix} \boldsymbol{Y} = \begin{pmatrix} -l\boldsymbol{A}sin\boldsymbol{\theta} \\ p_y \end{pmatrix}. \tag{2.8}$$

It can be shown that

$$\begin{pmatrix} \boldsymbol{D} \\ \frac{1}{N}\boldsymbol{e}^T \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{D}(\boldsymbol{D}\boldsymbol{D}^T)^{-1} & \boldsymbol{e} \end{pmatrix} \tag{2.9}$$

which gives us the very useful equation that describes the position of each link as a function of the centre of mass

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{D} \\ \frac{1}{N}\boldsymbol{e}^T \end{pmatrix}^{-1} \begin{pmatrix} -l\boldsymbol{A}cos\boldsymbol{\theta} \\ p_x \end{pmatrix} = -l\boldsymbol{K}^T cos\theta + \boldsymbol{e}p_x \tag{2.10}$$

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{D} \\ \frac{1}{N}\boldsymbol{e}^T \end{pmatrix}^{-1} \begin{pmatrix} -l\boldsymbol{A}sin\boldsymbol{\theta} \\ p_y \end{pmatrix} = -l\boldsymbol{K}^T sin\theta + \boldsymbol{e}p_y. \tag{2.11}$$

Here $\boldsymbol{K} = \boldsymbol{A}^T(\boldsymbol{D}\boldsymbol{D}^T)^{-1}\boldsymbol{D} \in \mathbb{R}^{N\times N}$. In the following section it will be shown that the friction force is dependent on the velocity of each segment. The velocity can be found by differentiating (2.10) and (2.11) with respect to time which is

$$\dot{\boldsymbol{X}} = l\boldsymbol{K}^T\boldsymbol{S}_\theta\dot{\boldsymbol{\theta}} + \boldsymbol{e}\dot{p}_x \tag{2.12}$$

$$\dot{\boldsymbol{Y}} = l\boldsymbol{K}^T\boldsymbol{C}_\theta\dot{\boldsymbol{\theta}} + \boldsymbol{e}\dot{p}_y \tag{2.13}$$

where $\boldsymbol{S}_\theta = diag(sin\boldsymbol{\theta}) \in \mathbb{R}^{N\times N}$ and $\boldsymbol{C}_\theta = diag(cos\boldsymbol{\theta}) \in \mathbb{R}^{N\times N}$ is square matrices of zeros with $sin\boldsymbol{\theta}$ and $cos\boldsymbol{\theta}$ on the diagonal.

2.2.3 Friction model

The snake's movement is fundamentally dependent on ground friction. To produce the forward force necessary to move the snake, the friction of each link needs to be anisotropic [1, s. 45], meaning that the friction constant in the snake's tangential direction needs to be small relative to the friction in the normal direction of the snake. In the model, we assume that the friction acts on the centre of mass of each link and is denoted by

$$\boldsymbol{f}_{R,i} = \boldsymbol{f}_{R,i}^{global} = \begin{pmatrix} f_{R,x,i} \\ f_{R,y,i} \end{pmatrix} \in \mathbb{R}^2, \tag{2.14}$$

which can then be written in matrix form as

$$\boldsymbol{f}_R = \begin{pmatrix} \boldsymbol{f}_{R,x} \\ \boldsymbol{f}_{R,y} \end{pmatrix} \in \mathbb{R}^{2N}. \tag{2.15}$$

Here $\boldsymbol{f}_{R,x} = (f_{R,x,1}, ..., f_{R,x,N})^T \in \mathbb{R}^N$ and $\boldsymbol{f}_{R,y} = (f_{R,y,1}, ..., f_{R,y,N})^T \in \mathbb{R}^N$ are column vectors containing the friction forces on each link in x and y directions. In the model the Coulomb's law of friction is used in order to obtain a realistic image of the actual snake's movement. Coulomb's law of friction takes into account both the velocity of the snake and the normal force acting on each link. The tangential and normal friction is here respectively denoted by μ_t and μ_n. The Coulomb friction force acting on link i can now be defined as

$$\boldsymbol{f}_{R,i}^{link,i} = -mg \begin{pmatrix} \mu_t & 0 \\ 0 & \mu_n \end{pmatrix} sgn(\boldsymbol{v}_i^{link,i}), \tag{2.16}$$

where $\boldsymbol{v}_i^{link,i} \in \mathbb{R}^2$ represent the link velocity expressed in the local frame and g represent the gravitational acceleration constant. To express the global frame Coulomb friction on link i in the form of (2.14) we create the rotation matrix $\boldsymbol{R}_{link,i}^{global}$ as

$$\boldsymbol{R}_{link,i}^{global} = \begin{pmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{pmatrix}. \tag{2.17}$$

This rotation matrix is used to translate the global frame to the frame of link i. We can now write the global frame Coulomb friction force on link i in the form of (2.14) as

$$\begin{aligned} \boldsymbol{f}_{R,i} = \boldsymbol{f}_{R,i}^{global} &= \boldsymbol{R}_{link,i}^{global} \boldsymbol{f}_{R,i}^{link,i} & (2.18) \\ &= -mg\boldsymbol{R}_{link,i}^{global} \begin{pmatrix} \mu_t & 0 \\ 0 & \mu_n \end{pmatrix} sgn(\boldsymbol{v}_i^{link,i}) & (2.19) \\ &= -mg\boldsymbol{R}_{link,i}^{global} \begin{pmatrix} \mu_t & 0 \\ 0 & \mu_n \end{pmatrix} sgn\left((\boldsymbol{R}_{link,i}^{global})^T \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix} \right). & (2.20) \end{aligned}$$

By carrying out these matrix multiplications and then writing the forces on all links in matrix form, we can write the global frame Coulomb friction forces on the links in the form of (2.15) as

$$\boldsymbol{f}_R = \begin{pmatrix} \boldsymbol{f}_{R,x} \\ \boldsymbol{f}_{R,y} \end{pmatrix} = -mg \begin{pmatrix} \mu_t \boldsymbol{C}_\theta & -\mu_n \boldsymbol{S}_\theta \\ \mu_t \boldsymbol{S}_\theta & \mu_n \boldsymbol{C}_\theta \end{pmatrix} sgn\left(\begin{pmatrix} \boldsymbol{C}_\theta & \boldsymbol{S}_\theta \\ -\boldsymbol{S}_\theta & \boldsymbol{C}_\theta \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{X}} \\ \dot{\boldsymbol{Y}} \end{pmatrix} \right) \in \mathbb{R}^{2N}. \tag{2.21}$$

2.2.4 Dynamic model

In order to understand how the acting forces affects the robot's movement, the dynamic equations must be added to the model. The forces will here be presented as a function of the acceleration of the angles, $\ddot{\boldsymbol{\theta}}$, and the robot's centre of mass, $\ddot{\boldsymbol{p}}$. As presented in figure 2.1 there are two sources of force acting on each link, the ground friction force, $\boldsymbol{f}_{R,i}$, and the joint constraint forces, $-h_{x,i-1}$, $-h_{y,i-1}$, $h_{x,i}$ and $h_{y,i}$. The force balance can now be presented as

$$\begin{aligned} m\ddot{x}_i &= f_{R,x,i} + h_{x,i} - h_{x,i-1} & (2.22) \\ m\ddot{y}_i &= f_{R,y,i} + h_{y,i} - h_{y,i-1}. & (2.23) \end{aligned}$$

Which can then be written in matrix form as

$$\begin{aligned} m\ddot{\boldsymbol{X}} &= \boldsymbol{f}_{R,x} + \boldsymbol{D}^T \boldsymbol{h}_x & (2.24) \\ m\ddot{\boldsymbol{Y}} &= \boldsymbol{f}_{R,y} + \boldsymbol{D}^T \boldsymbol{h}_y, & (2.25) \end{aligned}$$

where $\boldsymbol{h}_x = (h_{x,1}, .., h_{x,N-1})^T \in \mathbb{R}^{N-1}$ and similar for $\boldsymbol{h}_y$. Another equation for the link acceleration is found by differentiating (2.5) and (2.6) twice with respect to time, giving

$$\begin{aligned} \boldsymbol{D}\ddot{\boldsymbol{X}} &= l\boldsymbol{A}\left(\boldsymbol{C}_\theta \dot{\boldsymbol{\theta}}^2 + \boldsymbol{S}_\theta \ddot{\boldsymbol{\theta}} \right) & (2.26) \\ \boldsymbol{D}\ddot{\boldsymbol{Y}} &= l\boldsymbol{A}\left(\boldsymbol{S}_\theta \dot{\boldsymbol{\theta}}^2 + \boldsymbol{C}_\theta \ddot{\boldsymbol{\theta}} \right). & (2.27) \end{aligned}$$

To find the acceleration of the centre of mass we need to differentiate (2.2) twice with respect to time and insert the equations for the force balance, this now gives

$$\begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \end{pmatrix} = \frac{1}{N} \begin{pmatrix} \boldsymbol{e}^T \ddot{\boldsymbol{X}} \\ \boldsymbol{e}^T \ddot{\boldsymbol{Y}} \end{pmatrix} = \frac{1}{Nm} \begin{pmatrix} \boldsymbol{e}^T \boldsymbol{f}_{R,x} \\ \boldsymbol{e}^T \boldsymbol{f}_{R,u} \end{pmatrix} = \frac{1}{Nm} \boldsymbol{E}^T \boldsymbol{f}_R. \tag{2.28}$$

Since the robot's movement is based on the torque from the servos, we will need to set up a torque balance. The torque balance for link i is given by

$$J\ddot{\theta}_i = u_i - u_{i-1} - lsin\theta_i(h_{x,i} + h_{x,i-1}) + lcos\theta_i(h_{y,i} + h_{y,i-1}), \tag{2.29}$$

where u_i and u_{i-1} represent the torque acting on link i from the servos on link $i+1$ and on link $i-1$. These equations may now be written in matrix form as

$$J\ddot{\boldsymbol{\theta}} = \boldsymbol{D}^T \boldsymbol{u} - l\boldsymbol{S}_\theta \boldsymbol{A}^T \boldsymbol{h}_x + l\boldsymbol{C}_\theta \boldsymbol{A}^T \boldsymbol{h}_y. \tag{2.30}$$

In order to remove the joint constraint forces from this equation, we premultiply (2.24) and (2.25) with $\boldsymbol{D}$ and solve for $\boldsymbol{h}_x$ and $\boldsymbol{h}_y$, by now inserting (2.26) and (2.27) the following expression for the joint constraint forces are received

$$\boldsymbol{h}_x = \left(\boldsymbol{D}\boldsymbol{D}^T\right)^{-1} \left(ml\boldsymbol{A}\left(\boldsymbol{C}_\theta \dot{\boldsymbol{\theta}}^2 + \boldsymbol{S}_\theta \ddot{\boldsymbol{\theta}}\right) - \boldsymbol{D}\boldsymbol{f}_{R,x}\right) \tag{2.31}$$

$$\boldsymbol{h}_y = \left(\boldsymbol{D}\boldsymbol{D}^T\right)^{-1} \left(ml\boldsymbol{A}\left(\boldsymbol{S}_\theta \dot{\boldsymbol{\theta}}^2 - \boldsymbol{C}_\theta \ddot{\boldsymbol{\theta}}\right) - \boldsymbol{D}\boldsymbol{f}_{R,y}\right). \tag{2.32}$$

By using these equations, the joint constraint forces can be removed from (2.30) and finally the dynamics of the model can be expressed as

$$\boldsymbol{M}_\theta \ddot{\boldsymbol{\theta}} + \boldsymbol{W}\dot{\boldsymbol{\theta}}^2 - l\boldsymbol{S}_\theta \boldsymbol{K}\boldsymbol{f}_{R,x} + l\boldsymbol{C}_\theta \boldsymbol{K}\boldsymbol{f}_{R,y} = \boldsymbol{D}^T \boldsymbol{u} \tag{2.33}$$

$$Nm\ddot{\boldsymbol{p}} = Nm \begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \end{pmatrix} = \boldsymbol{E}^T \boldsymbol{f}_R, \tag{2.34}$$

where $\boldsymbol{M}_\theta$, $\boldsymbol{W}$, $\boldsymbol{V}$ and $\boldsymbol{K}$ are created to make the model more compact. These matrices are given by

$$\boldsymbol{V} = \boldsymbol{A}^T \left(\boldsymbol{D}\boldsymbol{D}^T\right)^{-1} \boldsymbol{A} \tag{2.35}$$

$$\boldsymbol{K} = \boldsymbol{A}^T \left(\boldsymbol{D}\boldsymbol{D}^T\right)^{-1} \boldsymbol{D} \tag{2.36}$$

$$\boldsymbol{W} = ml^2 \boldsymbol{S}_\theta \boldsymbol{V} \boldsymbol{C}_\theta - ml^2 \boldsymbol{C}_\theta \boldsymbol{V} \boldsymbol{S}_\theta \tag{2.37}$$

$$\boldsymbol{M}_\theta = J\boldsymbol{I}_N + ml^2 \boldsymbol{S}_\theta \boldsymbol{V} \boldsymbol{S}_\theta + ml^2 \boldsymbol{C}_\theta \boldsymbol{V} \boldsymbol{C}_\theta. \tag{2.38}$$

Note that the friction force $\boldsymbol{f}_R$ defined in (2.21) is dependent on the velocities of each segment, which is dependent on the velocity of the centre of mass for the whole robot, $\boldsymbol{p}$ and the the velocity of the link angles $\boldsymbol{\theta}$. This makes the equation (2.33) a second order differential equation and (2.34) a first order differential equation.

2.2.5 Compensation for movement in slope

The following compensation for movement in a slope is valid if the whole robot is located in the slope and not while it is entering the slope or leaving it.

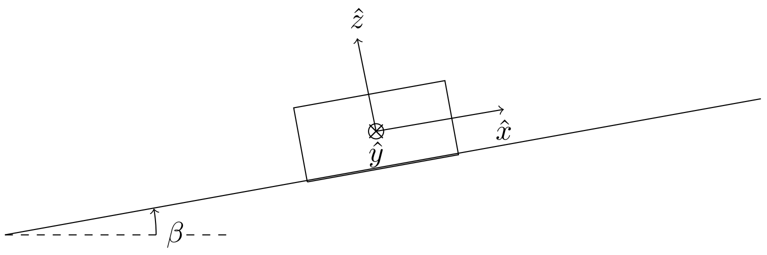


Figure 2.2: The global coordinate system is tilted with angle β so that the global x-axis points straight up the slope and the z-axis is normal to the slope.

If the global coordinate system is tilted with the slope so that the z-axis and x-axis points in normal direction to the slope and up the slope respectively, see figure 2.2 for clarification.

In the tilted global xy-plane the kinematics is the same as earlier but the dynamics change since the normal force exerts less force in the z-direction which results in less friction and a new force that drags the links down the slope in negative x-direction. See figure 2.2.

The force exerted by the slope on the links in the normal z-direction is

$$f_{normal,z} = mgcos\beta, \tag{2.39}$$

where β is the angle of the slope.

This results in a scaling of the friction force by a factor $cos\beta$

$$\boldsymbol{f}_R = \begin{pmatrix} \boldsymbol{f}_{R,x} \\ \boldsymbol{f}_{R,y} \end{pmatrix} = -mgcos\beta \begin{pmatrix} \mu_t \boldsymbol{C}_\theta & -\mu_n \boldsymbol{S}_\theta \\ \mu_t \boldsymbol{S}_\theta & \mu_n \boldsymbol{C}_\theta \end{pmatrix} sgn\left(\begin{pmatrix} \boldsymbol{C}_\theta & \boldsymbol{S}_\theta \\ -\boldsymbol{S}_\theta & \boldsymbol{C}_\theta \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{X}} \\ \dot{\boldsymbol{Y}} \end{pmatrix} \right) \in \mathbb{R}^{2N}. \tag{2.40}$$

The new dragging force down the slope in link system is

$$\boldsymbol{f}_{drag}^{link,i} = mgsin\beta \begin{pmatrix} -cos\theta_i \\ sin\theta_i \end{pmatrix} \tag{2.41}$$

which in global tilted coordinates is

$$\boldsymbol{f}_{drag} = \boldsymbol{R}_{link,i}^{global} \boldsymbol{f}_{drag}^{link,i} = mgsin\beta \begin{pmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{pmatrix} \begin{pmatrix} -cos\theta_i \\ sin\theta_i \end{pmatrix} \tag{2.42}$$

$$= mgsin\beta \begin{pmatrix} -cos^2\beta - sin^2\beta \\ 0 \end{pmatrix} = -mgsin\beta \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{2.43}$$

The force balance is now

$$m\ddot{\boldsymbol{X}} = \boldsymbol{f}_{R,x} + \boldsymbol{D}^T \boldsymbol{h}_x + \boldsymbol{f}_{drag,x} \tag{2.44}$$

$$m\ddot{\boldsymbol{Y}} = \boldsymbol{f}_{R,y} + \boldsymbol{D}^T \boldsymbol{h}_y + \boldsymbol{f}_{drag,y} = \boldsymbol{f}_{R,y} + \boldsymbol{D}^T \boldsymbol{h}_y, \tag{2.45}$$

since $\boldsymbol{f}_{drag,y} = 0$. This gives the following acceleration of the centre of mass

$$\begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \end{pmatrix} = \frac{1}{N} \begin{pmatrix} \boldsymbol{e}^T \ddot{\boldsymbol{X}} \\ \boldsymbol{e}^T \ddot{\boldsymbol{Y}} \end{pmatrix} = \frac{1}{Nm} \begin{pmatrix} \boldsymbol{e}^T \boldsymbol{f}_{R,x} + \boldsymbol{e}^T \boldsymbol{f}_{drag,x} \\ \boldsymbol{e}^T \boldsymbol{f}_{R,u} \end{pmatrix} = \frac{1}{Nm} \boldsymbol{E}^T (\boldsymbol{f}_R + \boldsymbol{f}_{drag}) \tag{2.46}$$

because $\boldsymbol{e}^T \boldsymbol{D}^T = 0$.

The torque balance is still

$$J\ddot{\boldsymbol{\theta}} = \boldsymbol{D}^T \boldsymbol{u} - l\boldsymbol{S}_\theta \boldsymbol{A}^T \boldsymbol{h}_x + l\boldsymbol{C}_\theta \boldsymbol{A}^T \boldsymbol{h}_y \tag{2.47}$$

but the constraint forces $\boldsymbol{h}$ is affected by the change in friction and the new dragging force. If the constraint forces is substituted from (2.44) and (2.45) and inserted into (2.47) the following dynamic model is withheld

$$\boldsymbol{M}_\theta \ddot{\boldsymbol{\theta}} + \boldsymbol{W}\dot{\boldsymbol{\theta}}^2 - l\boldsymbol{S}_\theta \boldsymbol{K}(\boldsymbol{f}_{R,x} + \boldsymbol{f}_{drag,x}) + l\boldsymbol{C}_\theta \boldsymbol{K}\boldsymbol{f}_{R,y} = \boldsymbol{D}^T \boldsymbol{u} \tag{2.48}$$

$$Nm\ddot{\boldsymbol{p}} = Nm \begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \end{pmatrix} = \boldsymbol{E}^T (\boldsymbol{f}_R + \boldsymbol{f}_{drag}). \tag{2.49}$$

The constant matrices is the same as before in (2.35)-(2.38). Note that if the slope angle $\beta = 0$ the drag force $f_{drag} = -mgsin\beta = 0$ and the scaling of the friction is $cos\beta = 1$. This results in the exact same model as before in a horizontal plane.

We can now define the model on state space form with the state vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)^T = (\boldsymbol{\theta}, \boldsymbol{p}, \dot{\boldsymbol{\theta}}, \dot{\boldsymbol{p}})^T$ as

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \\ \dot{\boldsymbol{x}}_3 \\ \dot{\boldsymbol{x}}_4 \end{pmatrix} = \boldsymbol{F}(\boldsymbol{u}, \boldsymbol{x}) \tag{2.50}$$

where $\boldsymbol{f}(\boldsymbol{u}, \boldsymbol{x})$ is the nonlinear equation system of (2.48) and (2.49) and the $\dot{\boldsymbol{x}}$ can be found by solving the equations for each element. The control signal $\boldsymbol{u}$ is the torque exerted by each link actuated by the angular actuators.

2.2.6 Separation of actuated and unactuated degrees of freedom

To better understand how to control the robot the actuated and unactuated degrees of freedom can be separated. The actuated degrees of freedom is only the $N-1$ joint angles $\boldsymbol{\phi}$ since this is all that can be controlled with the angular actuators between the links. The heads link angle θ_N and the position of the centre of mass $\boldsymbol{p}$ is unactuated since this is states that can not directly be controlled by any control signal.

As can be seen in figure 2.1, θ can be added up from the link angle of the head with the joint angles of the links behind $\theta_i = \theta_N + \phi_{N-1} + \phi_{N-2} + ... + \phi_i$. If a new angle vector $\bar{\boldsymbol{\phi}} = (\phi_1, ..., \phi_{N-1}, \theta_N)^T$ that builds up by $\boldsymbol{\phi}$ and link angle of the head link θ_N is defined it is possible to transform $\bar{\boldsymbol{\phi}}$ to $\boldsymbol{\theta}$ and back with the following invertible $N \times N$ matrix $\boldsymbol{H}$

$$\boldsymbol{\theta} = \boldsymbol{H}\bar{\boldsymbol{\phi}} = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ 0 & 1 & 1 & ... & 1 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & ... & 1 \end{pmatrix} \bar{\boldsymbol{\phi}}. \tag{2.51}$$

The dynamic model can now be rewritten with $\boldsymbol{\theta} = \boldsymbol{H}\bar{\boldsymbol{\phi}}$, $\dot{\boldsymbol{\theta}}^2 = diag(\dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} = diag(\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}})\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}}$ which results in

$$\boldsymbol{M}_\theta \boldsymbol{H}\ddot{\bar{\boldsymbol{\phi}}} + \boldsymbol{W} diag(\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}})\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}} - l\boldsymbol{S}_\theta \boldsymbol{K}(\boldsymbol{f}_{R,x} + \boldsymbol{f}_{drag,x}) + l\boldsymbol{C}_\theta \boldsymbol{K}\boldsymbol{f}_{R,y} = \boldsymbol{D}^T\boldsymbol{u} \tag{2.52}$$

$$Nm\ddot{\boldsymbol{p}} = Nm\begin{pmatrix} \ddot{p}_x \\ \ddot{p}_y \end{pmatrix} = \boldsymbol{E}^T(\boldsymbol{f}_R + \boldsymbol{f}_{drag}). \tag{2.53}$$

The system (2.52) is N rows since $\bar{\phi}$ is N long but only the first N-1 lines depends on the actuated degree of freedom $\begin{pmatrix} \phi_1 \\ \phi_2 \\ ... \\ \phi_{N-1} \end{pmatrix}$ and the last line together with the two lines of (2.53) depends on the unactuated degree of freedom $\begin{pmatrix} \theta_N \\ p_x \\ p_y \end{pmatrix}$.

If (2.52) is premultiplied with $\boldsymbol{H}^T$ and the variables are changed to the actuated and unactuated variables $\boldsymbol{q_a} = \boldsymbol{\phi}$ and $\boldsymbol{q_u} = \begin{pmatrix} \theta_N \\ \boldsymbol{p} \end{pmatrix}$ the following system is withheld

$$\bar{\boldsymbol{M}}_{11}\ddot{\boldsymbol{q}}_a + \bar{\boldsymbol{M}}_{12}\ddot{\boldsymbol{q}}_u + \bar{\boldsymbol{W}}_1 + \bar{\boldsymbol{G}}_1\boldsymbol{f}_R = \boldsymbol{u} \tag{2.54}$$

$$\bar{\boldsymbol{M}}_{21}\ddot{\boldsymbol{q}}_a + \bar{\boldsymbol{M}}_{22}\ddot{\boldsymbol{q}}_u + \bar{\boldsymbol{W}}_2 + \bar{\boldsymbol{G}}_2\boldsymbol{f}_R = \boldsymbol{0}_{3x1}. \tag{2.55}$$

Here $\bar{\boldsymbol{M}}_{11} \in \mathbb{R}^{(N-1)\times(N-1)}$ is the first N-1 rows columns of $\bar{\boldsymbol{M}}$, $\bar{\boldsymbol{M}}_{12} \in \mathbb{R}^{(N-1)\times 3}$ is the first N-1 rows and the last 3 columns, $\bar{\boldsymbol{M}}_{21} \in \mathbb{R}^{3\times(N-1)}$ is the last 3 rows and first N-1 columns and lastly $\bar{\boldsymbol{M}}_{22} \in \mathbb{R}^{3\times 3}$ is the last 3 rows and columns. $\bar{\boldsymbol{M}}$ is built up by

$$\bar{\boldsymbol{M}} = \begin{pmatrix} \boldsymbol{H}^T\boldsymbol{M}\boldsymbol{H} & \boldsymbol{0}_{N\times 2} \\ \boldsymbol{0}_{2\times N} & Nm\boldsymbol{I}_2 \end{pmatrix}. \tag{2.56}$$

$\bar{\boldsymbol{W}}_m$, $\bar{\boldsymbol{G}}_m$ are on the same way parts of the matrices $\bar{\boldsymbol{W}}$ and $\bar{\boldsymbol{G}}$ where $\bar{\boldsymbol{W}}_1 \in \mathbb{R}^{(N-1)}$ is the first N-1 rows, $\bar{\boldsymbol{W}}_2 \in \mathbb{R}^3$ is the last 3 rows, $\bar{\boldsymbol{G}}_1 \in \mathbb{R}^{(N-1)\times 2N}$ is the first N-1 rows and 2N columns and $\bar{\boldsymbol{G}}_2 \in \mathbb{R}^{3\times 2N}$ is the last 3 rows and 2N columns. $\bar{\boldsymbol{W}}$ and $\bar{\boldsymbol{G}}$ are

$$\bar{\boldsymbol{W}} = \begin{pmatrix} \boldsymbol{H}^T \boldsymbol{W} diag(\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}})\boldsymbol{H}\dot{\bar{\boldsymbol{\phi}}} \\ \boldsymbol{0}_{2\times 1} \end{pmatrix} \tag{2.57}$$

$$\bar{\boldsymbol{G}} = \begin{pmatrix} -l\boldsymbol{H}^T \boldsymbol{S}_{\boldsymbol{H}\bar{\boldsymbol{\phi}}}\boldsymbol{K} & l\boldsymbol{H}^T \boldsymbol{C}_{\boldsymbol{H}\bar{\boldsymbol{\phi}}}\boldsymbol{K} \\ -\boldsymbol{e}^T & \boldsymbol{0}_{1\times N} \\ \boldsymbol{0}_{1\times N} & -\boldsymbol{e}^T \end{pmatrix} \tag{2.58}$$

where $\boldsymbol{S}_{\boldsymbol{H}\bar{\boldsymbol{\phi}}} = \boldsymbol{S}_{\phi}$ and $\boldsymbol{C}_{\boldsymbol{H}\bar{\boldsymbol{\phi}}} = \boldsymbol{C}_{\phi}$.

We can now instead introduce the state vector $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)^T = (\boldsymbol{q}_a, \boldsymbol{q}_u, \dot{\boldsymbol{q}}_a, \dot{\boldsymbol{q}}_u)^T \in \mathbb{R}^{2N+4}$ and the state space form is now

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{q}}_a \\ \dot{\boldsymbol{q}}_u \\ \ddot{\boldsymbol{q}}_a \\ \ddot{\boldsymbol{q}}_u \end{pmatrix} = \boldsymbol{F}(\boldsymbol{u}, \boldsymbol{x}) \tag{2.59}$$

where $\boldsymbol{F}(\boldsymbol{u}, \boldsymbol{x})$ is the system of nonlinear differential equations in (2.54) and (2.55).

2.2.7 Controller design and reference signal

In order to test how the model works the following simple PD-controller have been used to change the control signal

$$\boldsymbol{u} = k_p(\boldsymbol{\phi_{ref}} - \boldsymbol{\phi}) + k_d\dot{\boldsymbol{\phi}}. \tag{2.60}$$

Where $\boldsymbol{\phi}_{ref}$ is a joint angle vector that corresponds to a proper lateral undulation and $\boldsymbol{\phi}$ and $\dot{\boldsymbol{\phi}}$ is the actual joint angle and joint angle velocity.

As described in 1.1.2 the snake produce its propelling motion by a muscular wave that propagates backwards along its body. This means that the reference joint angle will be time varying and all links must have a angular phase shift with regards to each other. According to [1, s. 81] the reference $\boldsymbol{\phi}_{ref}$ can be shown to be

$$\phi_{ref,i} = \alpha sin(\omega t + (i-1)\delta) + \Phi_0. \tag{2.61}$$

Here α is the amplitude of the angle, ω the frequency of angle oscillation and δ the phase shift between neighbouring links i. Φ_0 is a constant offset for all links that can be used to turn the direction of travel for the whole robot.

2.2.8 Simulating the visual input

The autonomous steering of the robot needs some kind of visual input in order to take decision of the steering. This is implemented in the real robot with sensors on the head that can measure distance to objects, which can be seen later in section 3.2.2. The sensors has a maximum and a minimum distance between where they can measure distance properly. In the simulation these sensors are modelled by line segments that starts at a point located at the minimum distance from the head and ends at a point located at the maximum distance from the head. See figure 2.3.

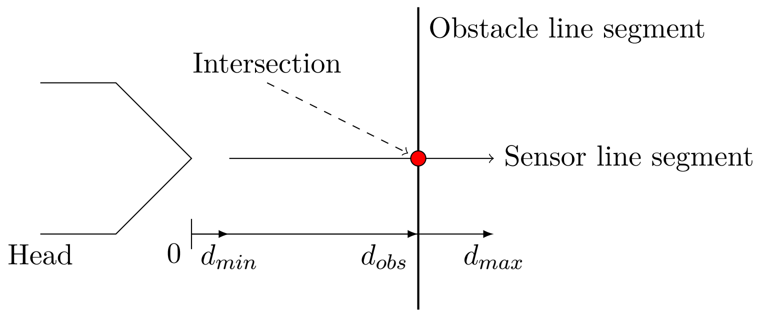


Figure 2.3: The sensor line segment starts at its minimum range d_{min} from the head and ends at its maximum range d_{max}. The measured distance d_{obs} is simply the distance from the head to the intersection point.

Obstacles that the robot must avoid can be modelled with polygons located in the room. If the sensor beams intersects with one of the line segments in the polygon it means that the sensor can see the object and the distance from the head to the point of intersection can be calculated.

2.2.9 Simulation environment

The simulations is made by solving the equation (2.50) with the ode solver *ode15s* [4]. Each time step in *ode15s* the control signal u is fetched from a controller function that holds the equation (2.60), the control function also fetch the reference from a reference function that holds the equation (2.61). In the reference function the Φ_0 can be changed during different time intervals in order to make the robot turn. This is done by a steering function that read the values from the sensors on the head as described in 2.2.8 and with this input take decisions about changing Φ_0 to steer accordingly. By changing this function, different algorithms for the autonomous steering can therefore be tested in the simulation environment with different obstacles.

When *ode15s* is done with its calculations it returns a list of all state variables and torques (control signals) during each time step. This can then be used to plot the movement and torques either in matlab or in the display tool developed by the project group. A flowchart of the simulation can be seen in figure 2.4.

The display tool was developed in C++ as a supplement to matlab. In order to show how all variables in the simulation interrelate it displays the robot in 2D, all its segments, the forces generated and the applied torque. The variables are saved in a file from the matlab program and then loaded by the display tool. An example figure can be seen in appendix B.4.

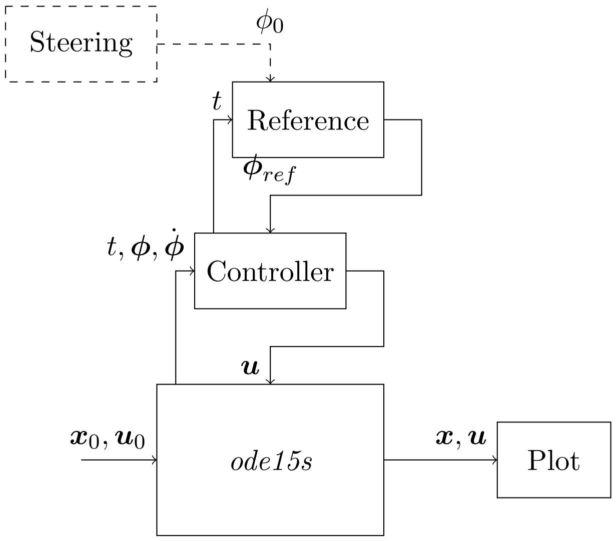

Steering
ϕ_0
t
Reference
$\boldsymbol{\phi}_{ref}$
$t, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}$
Controller
$\boldsymbol{u}$
$\boldsymbol{x}_0, \boldsymbol{u}_0$
ode15s
$\boldsymbol{x}, \boldsymbol{u}$
Plot

Figure 2.4: A flowchart of how the simulation is made. *ode15s* holds the equation (2.50) or (2.59), the controller is the controller in (2.60) and the reference is the reference signal in (2.61). The autonomous steering is done by the steering function that simulates the visual input to the robot and changes the reference signal in order to turn the robot.

2.3 Simulation results

Table 2.2: List of parameters used in the simulation.

Parameter	**Value**
N	8
l	55 cm
m	100 g
J	$\frac{1}{3}ml^2$
μ_n	0.7
μ_t	0.1
k_p	50
k_d	10
α	40°
ω	$80°\,\mathrm{s}^{-1}$
δ	$\frac{360°}{N} = 45°$

The following simulations are made with the parameters in table 2.2. These parameters all have a unique effect on the motion of the robot. The amplitude α denotes how wide the robot slithers which is directly linked to the force that drives the robot forward. An increasing amplitude will create higher force forward but also decrease the speed of the forward motion. This means a greater amplitude will make the robot able to climb greater slopes but in a slower velocity. In order to increase the speed of the forward motion, the joint angle frequency ω needs to be increased since the robot then moves its segments faster. To achieve a desired speed and forward force, these parameters must then be selected with each other in mind. The phase shift δ is directly linked to the shape of the robot since this denotes how much each joint angle differs from the one in front and back. If $\delta = \frac{360°}{N}$ the robot's segments together make up one sine wave period and if $\delta = \frac{720°}{N}$

they make two periods and so on. The effect of δ is not influenced by the amplitude or the joint angular frequency.

The simulation later runs for 30 seconds and during the time interval $11 < t < 14$ the robot turns by shifting the Φ_0 in the reference signal by ten degrees ($\Phi_0 = 10°$). This simulation produce the movement that can be seen in figure 2.5 which clearly is a snake-like lateral undulation. As it can be seen in the simulation, a positive joint angular offset causes the robot to turn right while a negative offset causes it to turn left. A change in this offset should be considered as a rotational movement rather than a turning movement according to [1, s. 96], as the entire robot rotates instead of continuously turning one segment at a time. The magnitude of the offset is not a measurement of how many degrees the robot will rotate, it is instead a measurement of how fast the robot will rotate. As visualised in 2.5 a 90° turn can be achieved by setting the joint angular offset to $\phi_0 = 15°$ for three seconds in this particular case. This however, depends on what position the robot and its segments has when the rotation begins and a specific $\phi_0 = 15°$ for a set amount of time does therefore not always result in the same turn. Moreover the $\phi_0 = 15°$ has different effect on the turning dependent on the other settings in the movement like α, ω and δ.

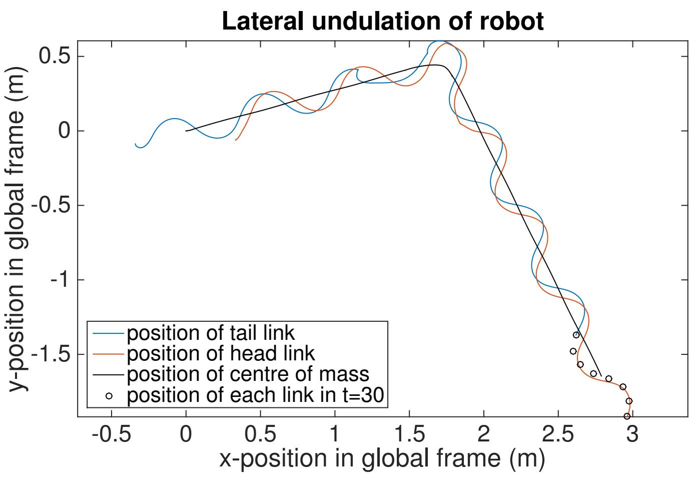


Figure 2.5: A simulation of the lateral undulation during 30 seconds. Between $11 < t < 14$ the robot turns by shifting its offset angle Φ_0 by ten degrees.

During the simulation, each angular actuator produces the torques that can be seen in figure 2.6. It is interesting to see that the turning of the robot makes the actuator produce more torque than the straight line motion.

To measure the robot's ability to deal with various slopes, the simulation seen in 2.7 have been produced, where the plane is tilted 10° after 10 seconds in to the simulation. This simulation does not take the phase where the robot is moving onto the slope into consideration, instead the plane on which the robot moves on is tilted 10° after 10 seconds in to the simulation.

It can be observed that the torque requirements on each angular actuator is significantly increased when the plane is tilted. Through further simulation it can also be observed that the robot's ability to handle even greater slopes is limited by the ground friction rather than by the torque, since the torques never get much higher than in figure 2.7 even if the slope is steeper.

The values obtained from these simulations can now be used to support the choice of angular actuators.

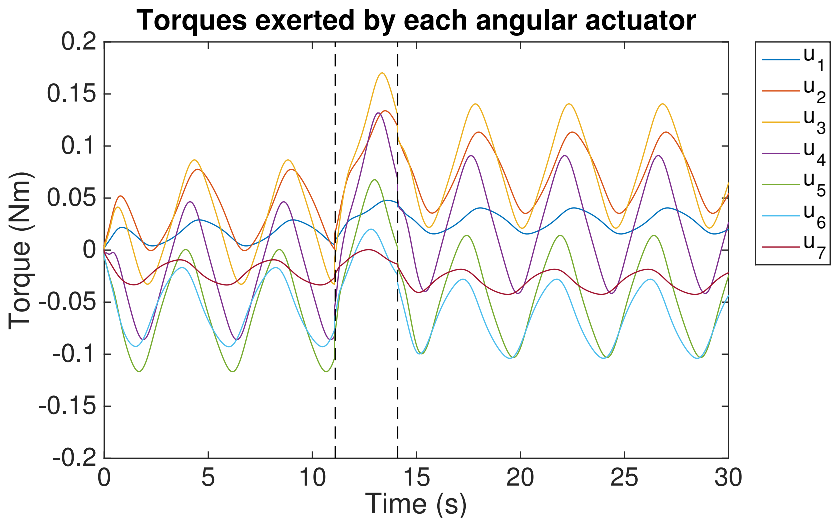


Figure 2.6: The variations of the torques by the angular actuators during the simulation. During the time interval $11 < t < 14$ (between the dashed lines) the torques increase due to the turning of the robot.

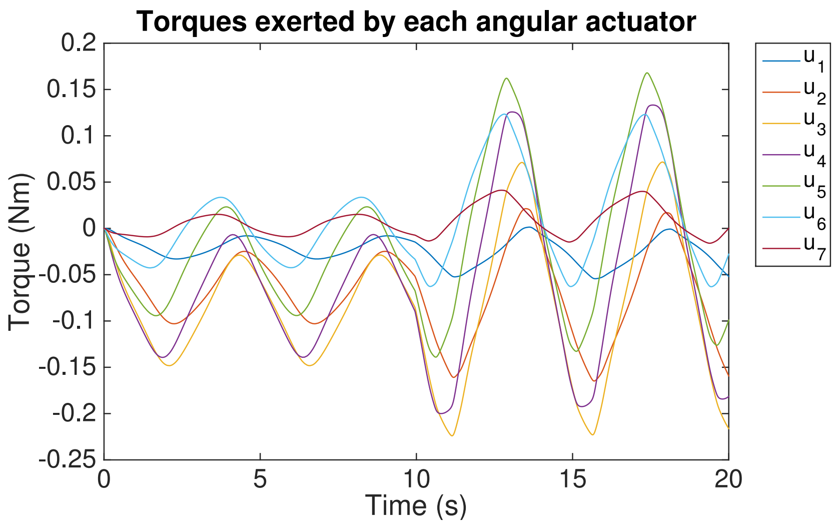


Figure 2.7: Variations of the torques when moving in 10° slope compared to horizontal plane. After 10 seconds the horizontal plane is tilted to a 10° slope.

3 Design of the robot

Designing the snake-like robot requires a transfer of the muscular movement of the snake into a robot. As explained in 1.1.2 the movement of the snake is made up of muscular contractions around joints in the spine, causing a wave-like motion. The design of the robot is represented in the same fashion. Segments make up the space in between the spinal joints while angular actuators makes up the muscular contractions around these joints. This representation can be directly transferred from the modelling and simulations made in section 2.3.

This chapter presents the process of designing the snake-like robot based on this segment approach and these simulations. The chapter is divided into several parts explaining both hardware choices as well as design choices which will be considered individually.

3.1 Demands

Demands for designing the robot are derived from the global demands and expanded to include specific demands for the design as well as requirements set by the simulation.

1. The robot must be autonomous, which implies that some kind of visual input is necessary to detect walls and obstacles.
2. The friction between the robot and the ground must be anisotropic enough for the lateral undulation to be effective.
3. The source of propulsion must counter the weight of the robot and torque demands withheld in the simulations in section 2.3.
4. The angular actuators must have the ability to perform a periodic and precise movement.
5. The energy source has to deliver enough energy to drive the angular actuators.
6. Microcontroller must be able to handle the control algorithm and communicate with the hardware.
7. The design should be flexible in three dimensions and be able to move in different angles, just like a snake.
8. The structure and material of the robot should be strong and stable enough to endure the stress that will occur during the movement of the robot.

9. The robot should look like a real snake.

3.2 Hardware choices

In order to choose a proper set of hardware, each demand is put into a Morphological matrix [5]. The matrix represents all of the different demands on each row together with all of the possible solutions in each column. From this matrix several solutions is obtained, together with a elimination matrix and a decision matrix, the best solution is then derived based on the requirements of the robot, the complexity of the solution and the price of that very solution. The process of choosing the best solutions revolves around examining each demand separately and finding the best solution for that very demand. The created morphological matrix can be found in A.1 and the result of each demand is presented below.

3.2.1 Angular actuators

As presented in chapter 2 the robot moves by changing each joint angle continuously over time. The angular actuator makes sure that the desired joint angle is achieved. It is implied that each angular actuator, according to (2.61) should retain a specific angle at each time step, mimicking the motion of a sine wave. If the desired position is not accomplished with an adequate precision it can result in a suboptimal robot movement. Therefore the ability of retaining the desired angle with high precision is an important aspect when choosing the device.

The chosen angular actuator, the DC servo *Dynamixel AX12+* [6] is equipped with a control circuit which regulates the position based on user input and a PID regulator inside of the servo. The servo contains a position sensor which continuously sends feedback to the integrated control circuit, regulating the position. Due to this, the servo has an angular resolution of 0.29°, which is relevant to the precision control of the robot. The integrated control circuit also has the functionality of providing valuable information to the user, such as the current angular position of the servo, the current load applied to the servo and the current angular speed. More information regarding the properties of the servo can be found in Appendix A.2.

The Dynamixel AX12+ servo utilises a packet based protocol for communication through the UART protocol [7, p. 553]. Each servo has an unique identifier included in the package, making it possible to connect several servos onto a buss for individual communication. The standard communication rate for the Dynamixel AX12+ servos is at 1 Mbit/s, which remains unchanged during the project.

As it can be seen in the data sheet for the Dynamixel servo A.2 the chosen servo has a maximum stall torque of 1.5 Nm together with a stable maximum motion torque of 0.3 Nm. The demand presented in 3.1 states that the angular actuator must be able to counter the weight of the robot. The simulation made in 2.7 presents the torques produced by each segment while the robot is moving in a 10° incline. The weight of each of the simulated segments is 100 g and as can be observed in the simulation, the maximum torque achieved is 0.23 Nm, meaning that weight of a segment could be increased to 100 g without having problems generating the motion.

3.2.2 Visual input

In order to make the robot autonomous, it must be aware of its surroundings. This can be achieved through multiple choices of visual inputs, such as global positioning systems and integrated sensors. In this project the robot uses range detecting IR-sensors.

The robot is equipped with three Sharp GP2Y0A21YK0F [8] IR-sensors for detecting objects close to the current position of the robot's head. These IR-sensors are optimised for detecting objects within the range of 10-80 cm and delivers an analog output signal. Each sensor has three pins where the output pin delivers a voltage proportional to the length of a detectable object. A graph showing the range-to-voltage relation can be viewed in [8, p. 5]. The sensors are relatively cheap while still maintaining a precise enough measurement for this project.

Due to the sensors measuring distance between 10-80 cm, the sensors deliver unstable output-voltage between 0-10 cm. Looking at the graph in [8, p. 5], the output in the interval 0-10 cm can be interpreted as the sensor detecting objects further away.

3.2.3 Microcontroller

To handle the movement and the sensor input a Arduino Uno[9] microcontroller is used. The Arduino card is equipped with a Atmega328 processor running on 16 MHZ clock frequency. It has 5 analog inputs, 13 digital I/O pins. Three of the analog inputs are used for monitoring the sensors. The board is able to communicate at speeds up to 2 Mbit/s, which works well with the chosen Dynamixel servo.

3.2.4 Energy source

A common energy source for the Arduino, IR-sensors and the servos has been used. Both the microcontroller and IR-sensors needs 5 V to function while the servos use a voltage around 9-12 V. The Arduino has an on board voltage regulator that provides 5 V output when supplied with a 7-12 V input. It is used for driving both the microcontroller and the IR-sensors. A 11,1 V LiPo battery capable of 800 mAh [10] is used for its high performance to weight ratio.

3.2.5 Ground friction

The ground friction is provided by a set of passive LEGO-wheels placed between the segments. The anisotropic friction is achieved by having some of the wheels slide to the sides with high friction while others roll forward with ease as the segments rotate. The wheels used can be seen in figure 3.1.

Figure 3.1: Picture of one of the wheels.

3.3 Design

The design is made with respect to the selected hardware in section 3.2. It is also designed to fulfil the demands in 3.1, such as to be movable in all directions and to match the physical requirements that is critical for the robot's movement. It is strong and stable enough to endure the stress that occurs during the robot movement, however the weight was to be kept low according to section 2.3.

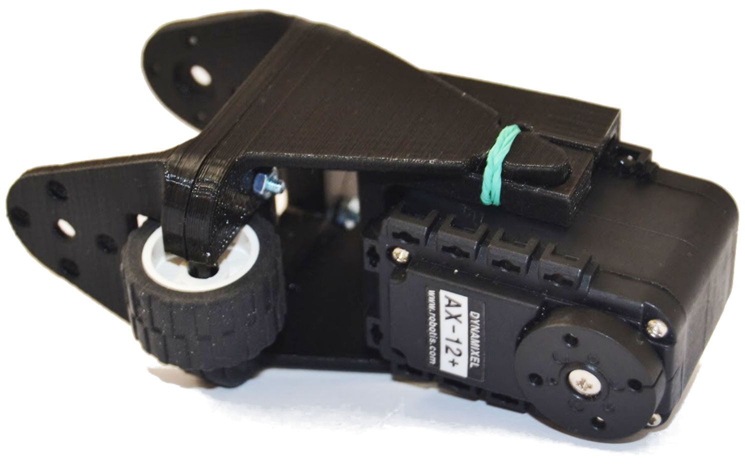


Figure 3.2: Assembled view of one section of the robot that is controllable in the horizontal plane and is flexible in the vertical direction.

The design procedure is based on module segments according to chapter 3, which means that identical sections are connected to each other to create the whole robot. All segments except for the head and the tail are identical and contains a servo, a wheel and space for cables. The microcontroller and the sensors are located in the head, while the battery is placed in the tail in order to compensate for its low weight, since the simulation in section 2.3 is based on equal weight for each segment.

The segments are not only controllable in the horizontal plane but also flexible in the third dimension, this enables it to go up and down a hill.

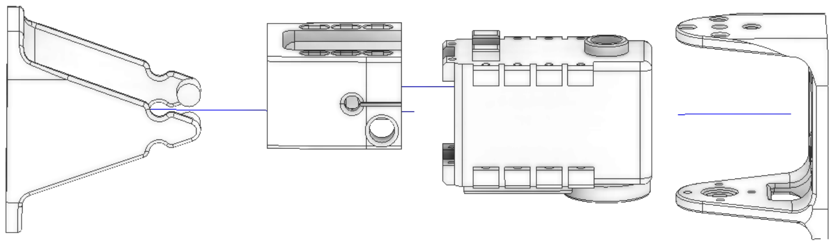


Figure 3.3: Exploded view of one module segment, the parts are from the left, the flexible arm, the base, the servo and the propulsion arm.

3.3.1 The module segments

The task for each segment is to generate a forward force so that the segments together can drive the robot, this propulsion comes from seven DC servos 3.2.1, one in each segment. The segment consist of three designed parts and one servo, which also can be seen in figure 3.3.

- Base
- Flexible arm
- Propulsion arm

The base, the second part from the left in figure 3.3, works as a centre for each segment and is the holder for the servo. It is designed to embrace the servo and its holes match the servo's screw holes for easy mounting. On both sides there is a 6 mm hole and an open track between the hole and the front end. The hole matches the flexible arm and the track is made so that a rubber band can be fastened between the track and the flexible arm. This solves the flexibility in the vertical plane since this setup act as a spring.

The flexible arm, the first part from the left in figure 3.3, is the key to the robot's manoeuvrability in the vertical plane. The arm embrace the base and matches the 6 mm holes on the side of the base. It is long enough to rotate freely around the servo, but is held in place by the rubber band. This also ensures that the cables will fit.

The flat side seen in figure 3.4 of the flexible arm has a big hole for cables in the middle, it also has four holes for screws to make it easy to build together with any other part. In the bottom under the central cable hole there is room for a wheel and its axis. This flat side that is described here is of the same kind in all parts, except for the base.

The function for the propulsion arm, the first part to the right in figure 3.3, is to transmit the movement created by the servo. It is a strong arm with the same flat side on the end

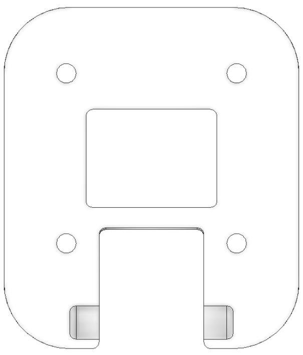


Figure 3.4: The flat side that constitute the connection between each designed part, except the base. With four screw holes, cable hole and a hole for a wheel in the bottom.

as the flexible arm for connections with other segments, seen in figure 3.4.

The result of this design can be seen in figure 3.2 where the produced parts are put together, embracing the chosen servo. The segment has a ± 90° of freedom in the horizontal plane. Several segments are then supposed to be connected together, creating the body of the robot.

3.3.2 Head

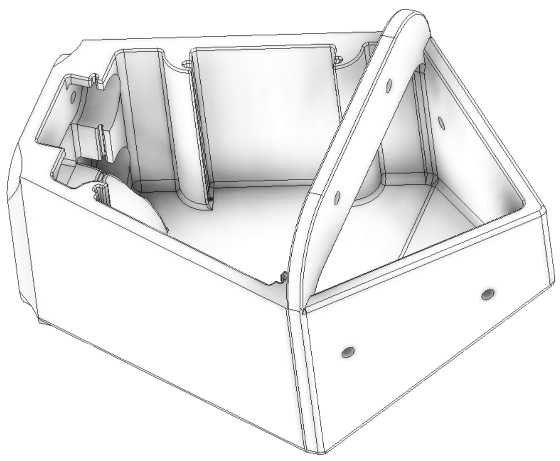


Figure 3.5: The head of the robot, with room for the micro-controller, sensors, and all the electronics

The head is the larges part of the robot and can be seen in figure 3.5. All the electronics, except the battery, are placed in the head which have an open design that enables easy modifying of the electronics. It has three tracks for each circuit board to fit and has a 90° front. The three eyes, IR-sensors, of the robot is positioned on each side of the 90° front and one on the top. Underneath the head there are two bumps to lift up the head and keep it in a horizontal position. The back of the head also has the flat side which makes it easy to attach and detach the head as it matches all other segments.

3.3.3 Tail

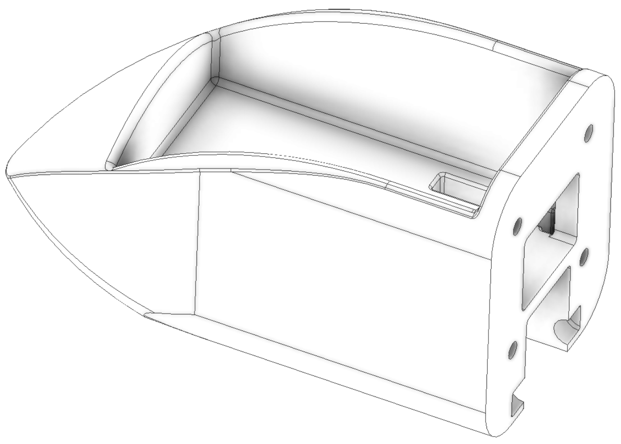


Figure 3.6: The tail of the robot, works as a battery holder

The primary function of the tail is to host the batteries and to give a proper weight for the last wheel, so that it can produce enough friction. The tail also serves a aesthetic function for the visual impression of the robot. Like all other parts it has the standard flat side for easy connection with the snake and has room for the LiPo battery selected in 3.2.4. The designed can be seen in figure 3.6.

3.3.4 Assembly of the whole robot

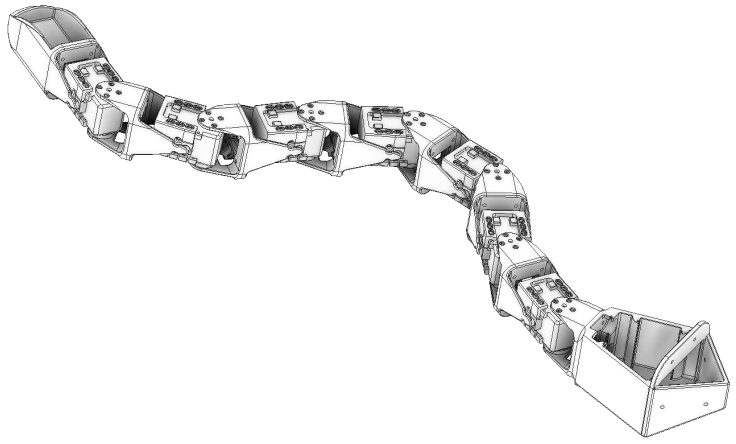


Figure 3.7: The whole robot, with seven segment a tail and a head.

With all seven segments and parts assembled, the complete design of the robot is obtained. The design can be observed in figure 3.7 where all of design pieces are connected together. In order to make a more snake-like appearance of the robot the designed parts are all printed in black ABS plastics. Further information about the design development and all the exact drawings on the designed and printed parts are located in appendix B.1 and appendix B.2.

4

Implementation and testing

In order to finalise the implementation of lateral undulation and to make the robot autonomous, the hardware has to be connected and implemented together with the microcontroller. The microcontroller has to be able to communicate with the servo while also periodically sample values through the sensors. In this part of the report, the implementation of the robot is described together with the final testing of the complete robot.

4.1 Communication between microcontroller and servos

As mentioned in section 3.2.1, the communication between the ATmega328 microcontroller (Arduino) and the Dynamixel 12+ servo is done with the UART protocol. The servos are connected in a daisy-chain [7, p. 134] with one wire handling both reading and transmitting. This can not be done simultaneously, either a message is sent from a servo to the Arduino or vice versa, this is called half-duplex communication [7, p. 239]. To allow communication both ways a converter is placed before the servos.

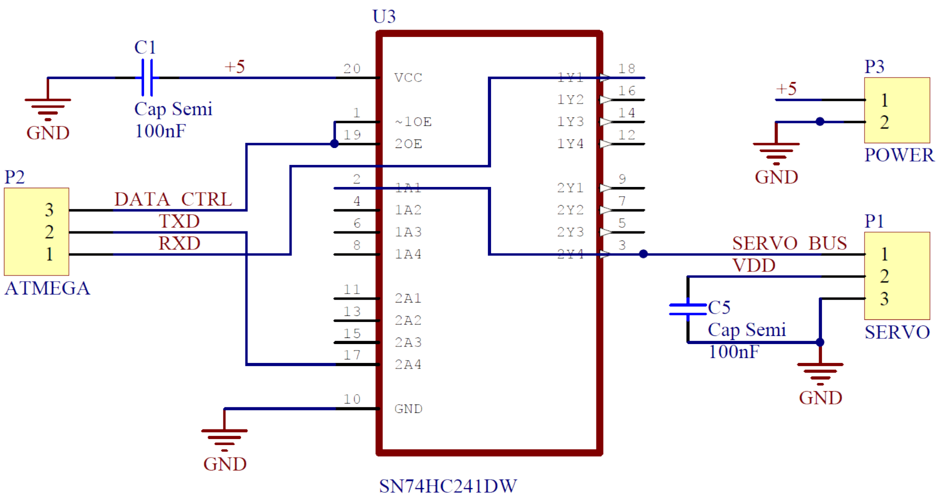


Figure 4.1: Converter from full-duplex to half-duplex to enable communication between Arduino and Servos

Using the SN74HC241DW [11] tri-state buffer and line driver integrated circuit configured as seen in figure 4.1, allows control over whether to send or receive data. When the level on DATA_CTRL pin is high the signal on TXD pin is transmitted to the bus, if the level is low any signals on the bus is transmitted to the Arduino through the RXD pin.

4.2 Communication with sensors and sensor filtering

The chosen IR-sensors gives an approximate answer of how far away an object is, see section 3.2.2. The analogue output of the sensor is converted into a digital signal using an A/D converter inside of the microcontroller, making it possible to sample and interpret the sensor signal. The sensor values can then be used to control the robot. In order to make sure the sensors give the appropriate value, they were tested.

The tests were conducted using an Arduino board, the Sharp GP2Y0A21YK0F sensor [8] and a blank sheet of white paper. The sensor was placed in clear view and sampled through the Arduino with a 10 ms delay in between samples. The blank sheet of paper was inserted at 32 cm away from the sensor shortly after the sampling started. The unfiltered result of the test is shown in the left most graph in figure 4.2. As it can be seen the sampled data is widely spread, sampling points varying from 0.7 to 1.16 V after the insertion of the blank sheet of the paper. By looking at the graph in datasheet [8, p. 5] the distance of 32 cm should be interpreted as roughly 0.83 V, meaning that there is a variation of -0.13 to $+0.33$ in the worst case.

Having this kind of spread in the sensor input could potentially cause the robot to make decisions based on false information. To avoid the spread of sampled inputs, the sampled values are put into a moving average filter. The moving average filter is defined by the following equation:

$$y(k) = \frac{1}{n}\left(x(k) + x(k-1) + ... + x(k-(n-1))\right) \tag{4.1}$$

where k denotes a discrete sample, $y(k)$ is the filtered sample value, $x(k)$ is the sampled value and n is the number of samples. The second graph to the right in 4.2 shows the same test performed together with an moving average filter working with $n = 20$ samples. As can be seen in that very graph, the resulting values are closer to the actual values with only ± 0.01 V variation. The payoff however is that the rise time is higher, taking approximately 800 ms to achieve the appropriate value.

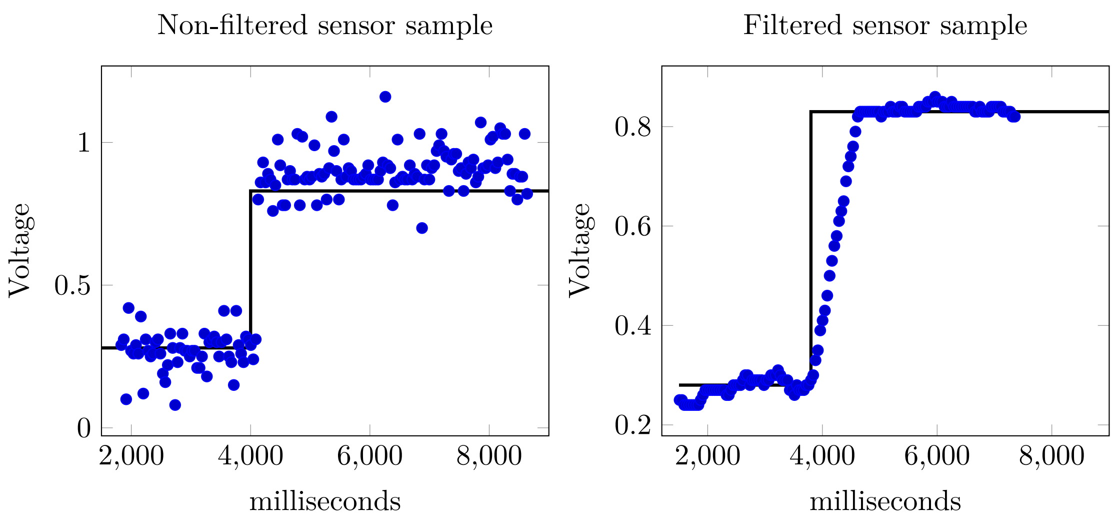


Figure 4.2: The graphs show the plotting of sampled sensor values during the test described in 4.2. The test was conducted with a sample interval of 10 ms where a blank sheet of paper was inserted after a set amount of time. The insertion of the paper can be seen as the black step function. The left graph shows the sampled unfiltered values. The right graph shows filtered values using a moving average moving filter with $n = 20$ samples.

4.3 Software

To move and steer the robot autonomously, a given software must run via the microcontroller. This software must set the servos to the desired positions for the robot to gain propulsion. The current implementation makes use of the filtered sensor input to check what kind of situation it has to deal with. Depending on the situation, decisions about the steering is made. This process is described by the flowchart of figure 4.3.

4.3.1 Steering states

The software implementation makes use of two states, forward mode or obstacle avoidance mode. If q is the state of the robot and S is the set of states then,

$$S = \{q_0, q_1\}, q \in S \tag{4.2}$$

where q_0 is the state when the robot should go straight and q_1 is the state when the robot should change its course to avoid observed obstacles.

At appropriate times, steering should be applied so that an object can be avoided. The condition for when this should happen is a function of the input. The function can be called $C(d)$, where d is the angle-to-distance mapping which is derived from the IR-sensor input. The implementation of this function will be described in more detail later on, see algorithm 1.

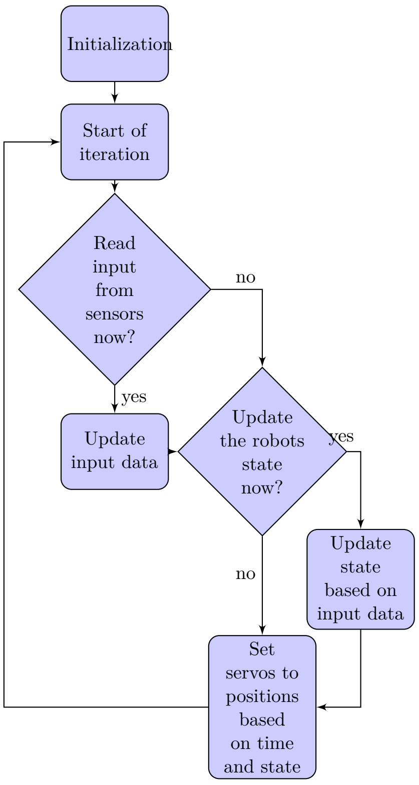

Initialization
Start of iteration
Read input from sensors now?
no
yes
Update input data
Update the robots state now?
yes
no
Update state based on input data
Set servos to positions based on time and state

Figure 4.3: The flowchart of the controller of the snake-like robot. Note that the input and state does not need to be updated in the same iteration. An iteration can pass without either being updated at all.

4.3.2 Forward mode

When in state q_0, that is, when the robot is supposed to steer forward, the positions of the servos only consists of a part that is the reference angular position. The theory behind how this results in forward motion was discussed in section 2.2.7. Using the formula (2.61) with Φ_0 set to 0 yields,

$$\phi_{ref,i} = \alpha sin(\omega t + (i-1)\delta), \tag{4.3}$$

where i is the number for each servo in order.

4.3.3 The input data

Since the robot only have three sensors, one in the front of the head and the two others directed to the left and right by 45° each, see figure 4.4. It is only possible to have knowledge of the surroundings consisting of three discrete points around the robot at any instant. This is not enough. For example, it is easy to imagine an obstacle that is within a radius of the robot that is considered too close, but is not intersected by any of the sensors.

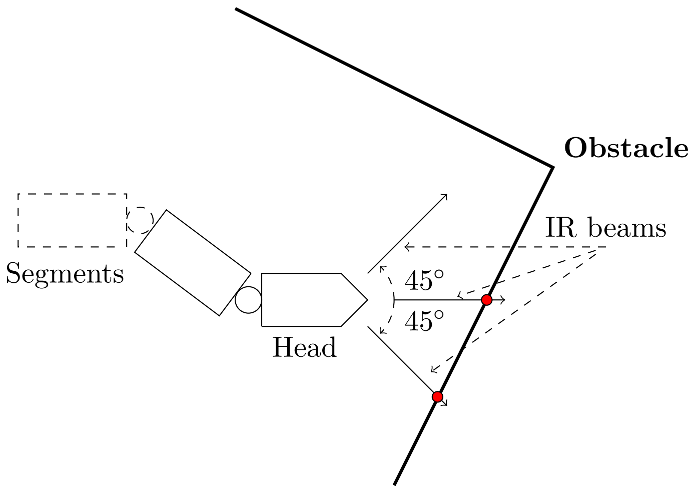


Figure 4.4: The sensors are mounted so that one points straight forward from the head and two points 45° to the left and right.

Therefore the robot scans its surroundings and saves the three data points in an angles-to-distance mapping d. The angles are relative to the robot's direction, which means that as the head moves around, it sweeps an angular interval and saves the distances in the map. Each half period the distances of the whole interval has been refreshed. This means that the quality of the distance at some angles are worse than the updated ones.

4.3.4 The state transition function

The condition $C(d)$ is currently implemented as seen in algorithm 1.

If $C(d) = 1$, the robot will have to turn, and if $C(d) = 0$ the robot can continue straight ahead. These transitions can be seen in the state chart of figure 4.5.

```
**Input**: $d$
**Output**: $b \in \{0, 1\}$
$n \leftarrow mapToIndex(-90°)$;
$b \leftarrow 0$;
**while** $n \leq mapToIndex(90°)$ **do**
    **if** $getDistance(d, n) < tooCloseLimit$ **then**
        $b \leftarrow 1$;
        stop;
    **end**
    $n \leftarrow n + 1$;
**end**
```

Algorithm 1: *mapToIndex*(φ) is the method of getting the index of the underlying distance-storage array where the distance at angle φ can be found. *tooCloseLimit* is the distance limit for obstacles. To summarize, if $b = 0$, the robot will not have to turn. If there exists any distance in the angle-to-distance mapping between $-90°$ and $90°$, then $b = 1$, which means that the robot should turn.

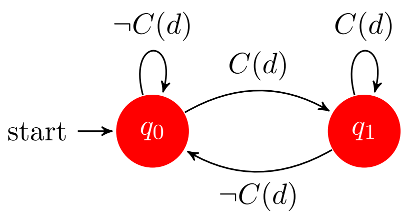


Figure 4.5: q_0 is the state when the robot should go straight and q_1 is the state when the robot must turn. The transitions from state q_0 to q_1 happens when the collision detection condition $C(d)$ evaluates to 1 and transitions the other way when it evaluates to 0.

4.3.5 Object avoidance mode

When in the steering state, $q = q_1$, a new suitable course for the robot is needed in order to avoid observed obstacles. The current implementation searches for a course offset, $\hat{\theta}$, which is the smallest suitable offset in relation to the current course of the robot. If $\theta_{heading}$ in (2.1), is the general direction of the robot, and θ is the general suitable course, then,

$$\hat{\theta} = \theta - \theta_{heading}. \tag{4.4}$$

The condition for a suitable course is that the distance at the angle can be observed to be larger than a certain limit, d_0. It is still possible that such an angle can not be found. In that case, the most suitable angle offset is the angle with the largest distance found.

This function, $\hat{\theta}(d)$, takes as input the known angle-to-distance mapping d, and returns the best course offset $\hat{\theta}$.

The course offset function is found and can be combined with the robot state q to form the general steering function,

$$\Phi_0(\hat{\theta}, q) = \begin{cases} 0, & q = q_0 \\ F(\hat{\theta}), & q = q_1 \end{cases}, \tag{4.5}$$

where $F(\hat{\theta})$ is currently implemented as a control function of the following form,

$$F(\hat{\theta}) = K_p\hat{\theta}. \tag{4.6}$$

4.3.6 Controlling the robot

The last step in each iteration is to always send the most recently calculated angular position to each servo. This angular position for servo i, is the combination of (4.3) and (4.5), which results in,

$$\phi_{ref,i}(t, \hat{\theta}, q) = \alpha sin(\omega t + (i-1)\delta) + \Phi_0(\hat{\theta}, q). \tag{4.7}$$

When a servo receives the new command from the controller, it will independently control the angular position with its own microcontroller. With all the servos working simultaneously, snake-like propulsion and steering of the robot is achieved.

4.4 Final product

In order to finalise the product, the design and hardware is combined with the control circuit to achieve the ability to move. The filtered sensors is connected with the microcontroller in order to gain the needed visual input. All of this is then combined with the software demonstrated in section 4.3 which completes the final product.

The final product consists of seven segments connected with each other. The head is mounted in the front of the robot and contains the sensors, the microcontroller and the control circuit. The robot is also equipped with a tail, holding the battery. All of the specifications of the robot can be viewed in the table 4.1.

Table 4.1: The specifications of the final robot.

Weight :	928g
Length :	85 cm
Nr. of segments :	7 + head + tail
Segment weight :	102g
Segment length :	
Tail weight :	125g
Tail length :	
Head weight :	89g
Head length :	
Microcontroller :	Arduino™ Uno
Angular actuators :	7 Dynamixel AX12+ DC servos
Power source :	11.1V LiPo battery
Ground friction :	LEGO wheels
Sensors :	3 Sharp GP2Y0A21YK0F IR-sensors

4.5 Verification of final product

To verify if the robot meet the demands in section 1.3.2 and that the hardware and software were correctly implemented, three verification tests that each examine a specific demand were conducted on the robot. A complete description of each test can be seen in A.1.

4.5.1 Straight line lateral undulation

In the test for straight line lateral undulation, A.1.1, the robot simply had to move at least twice its length in a straight line without stops using only lateral undulation.

This test was successful 10 out of 10 times during the official testings. The angular frequency, amplitude and phase shift could easily be changed in the control code to change the properties of the movement according to the simulations in section 2.3. Moreover the robot looked very much like a snake when it moved. One smaller problem were that the wheels seamed to slide a bit in the movement, and this raised a concern that the weight of the segments might be to low in order to get enough friction for the hill climbing.

4.5.2 Turning and autonomous steering

To test the turning and autonomous steering the robot had to pass a narrow corridor with a 90° turn without hitting the walls, A.1.2.

During this test the following parameters was used $\alpha = 40^\circ$, $\omega = 22.9^\circ\,\mathrm{s}^{-1}$ and $\delta = \frac{360^\circ}{N} = 51.4^\circ$.

The robot managed to navigate through the course 6 out of 10 times during the official testing in a left turn course, see table 4.2. The tests were successful if the robot cleared the course without hitting the wall. One exception was made when only the tip of the tail hit the wall and the rest of the robot was clear. The robot seemed to make fairly correct decisions about where and how much to turn and the most common reason why the test failed were that the robot came too close to the walls and then touched the walls either with its head or tail. Often it seamed that the robot came closer than the minimum range of the sensor and was then either unable to increase the distance or turned in the wrong direction into the wall. One problem is also that the turns were rough and not particularly smooth which resulted in a twitching motion through the course.

When the tests were conducted in a right turn the results were however really bad. The robot seamed to always turn in the wrong direction and the tests where never successful.

Table 4.2: Results from the official tests in a left turn. Distance between walls were 70 cm wide.

Test nr.	Result	Comment
1	approved	clean run without wall hits
2	rejected	managed the course but hit the left wall after the bend
3	approved	clean run without wall hits
4	approved	the tip of the tail hit the inner corner when passing
5	rejected	came close to left wall, made wrong turn into it
6	rejected	came close to right wall, turned to slow in the bend
7	approved	clean run without hits
8	approved	clean run but very close to inner corner
9	rejected	came too close to left wall and hit it
10	approved	clean run without hits

4.5.3 Hill climbing

The hill climbing was tested with a 10° slope that the robot had to climb up and down, see A.1.3 for more detail.

During this test the α was increased to 50° and ω was increased to $34.4° \, \mathrm{s}^{-1}$ in order to generate more propelling force and speed. The δ was unchanged.

Here the low friction became a problem despite that the slope was covered in a plaid rug that offer high friction against rubber. The robot slided a lot when climbing which not only resulted in a slow ascend, but also in an involuntary turning since the head slided downwards when it moved out to the side in the sine wave. To increase the friction a rubber rug was used in the ascend and the same plaid rug on the other parts. The test was then successful 9 out of 10 times, see table 4.3.

When external weights was mounted over the wheels in the segments to increase the weight and hopefully the friction force, the test was successful 9 out of 10 times without the rubber rug, see table 4.3.

The reason why the test did not always work were that the involuntary steering made the robot hit the walls on the sides of the hill and could not leave that position by itself. During the test the autonomous steering was shut off because it did not work properly in the descend. The slope was 70 cm wide.

Table 4.3: Results from official testings of the hill climbing. The robot where tested both without extra weights and with 10 g weights in each segment. Different rugs where used to generate enough friction.

	No weights, rubber rug	
Test nr.	**Result**	**Comment**
1	approved	hit the wall a few times to the left
2	approved	hit the left wall on the plateau
3	rejected	stuck alongside the left wall while climbing
4	approved	hit the left wall in the plateau once
5	barely approved	hit the wall a lot but managed by its own
6	approved	clean run without hitting the wall
7	approved	hit the right wall on the way up
8	approved	hit the right wall a bit on the way up
9	barely approved	hit the left wall hard
10	approved	lightly touched the right wall on the way down
	10 g weights, plaid rug	
Test nr.	**Result**	**Comment**
1	approved	hit the left wall a lot
2	approved	clean run without hitting the wall
3	approved	hit the left wall a little on the way down
4	barely approved	moved alongside the left wall the whole run
5	approved	hit the left wall a little
6	rejected	stuck alongside the left wall
7	barely approved	hit the left wall a lot
8	approved	hit the left wall a bit
9	approved	only touched the left wall once
10	approved	clean run without hits

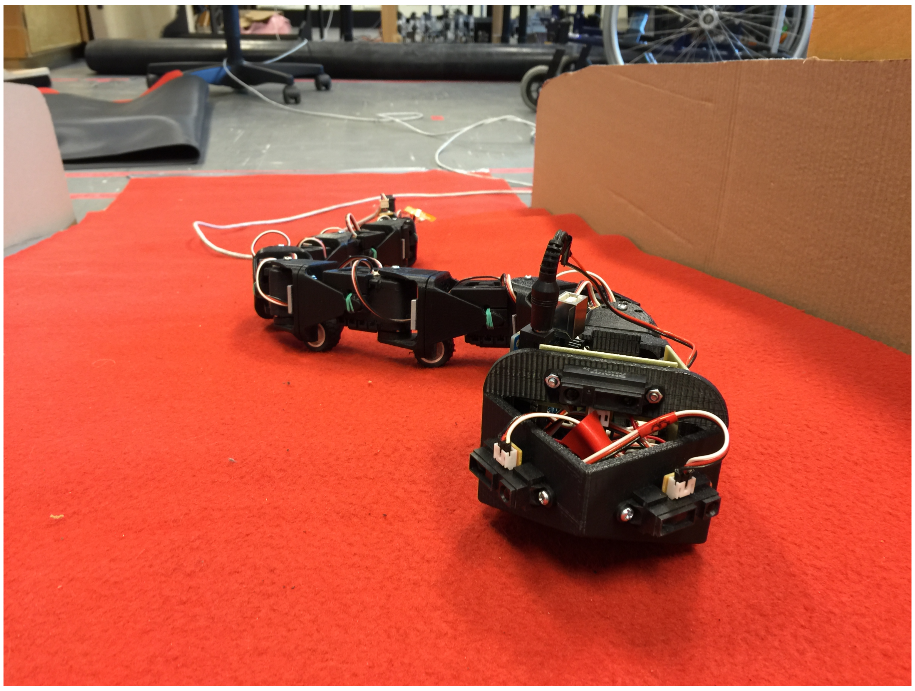


Figure 4.6: The robot climbing up the 10° inclination in the verification test.

5
Discussion

The following chapter brings up discussions regarding the outcome of the project and how well this outcome fulfilled the set demands. Choices made during the project are questioned, evaluated and motivated. The chapter ends with a discussion of how to further develop the robot and finally a conclusion.

5.1 Evaluation of final product

The verification tests of the final robot shows that most, but not all, of the demands are fulfilled.

Most vital is that the robot moves like a snake using lateral undulation. The first test shows that this is very well achieved since it moves forward in a controlled manner and all properties of the movement can be changed with the same effect as in the simulation. The low friction is however a bit frustrating. While it does not affect the motion in the horizontal plane it becomes a problem in the slope. The fact that the surface had to be changed to a rubber rug in order for the robot to successfully climb the slope is bad. But since the external weights solved this issue and the test thereby is successful even without the rubber rug, the demand for the hill climbing should be considered fulfilled by the final robot.

The demand for autonomous steering is clearly not fulfilled since the robot only passes the test in a left turn. The fact that it can pass the test in a left turn is however proof that the robot can both turn in the horizontal plane and detect walls, it only fails in the collision avoidance. This is considered a software problem rather than a design and hardware problem and the project group judge that the robot would absolutely manage the test without reconstruction, if the software problems were corrected. Unfortunately the time limitation of the project was not enough.

The budget for the project was also held as can be seen in appendix A.3, but the time schedule was unsuccessful since the software for the autonomous steering was not finished in time.

5.2 Evaluation of decisions and solutions

During the project a lot of the decisions and solutions regarding specific parts of the project have been made. The line of thought and motivation for each decision is further explained and discussed below.

5.2.1 Model

The demands for the model were that it should be possible to simulate the movement and see how different settings in the reference signal affects the motion, as well as answer the question of how much torque the servos must exert. Moreover the algorithm for the autonomous steering should be able to be tested in the simulation.

The simulation in section 2.3 shows that the first two demands are fulfilled since the real robot's movements is affected the same way as in the simulations when the reference signal is changed. The torque plot in figure 2.6 and 2.7 can not really be confirmed to be absolutely true but the choice of servo that was based on these results worked very well. The last demand was also fulfilled and the algorithm for the autonomous steering was possible to test in the simulation.

As mentioned earlier, the model for planar movement has been used before in similar projects and can not really be developed further. The only missing element is that all segments must have the same mass and length. Since the head and tail of the actual robot has a different shape and weight than the other segments, see table 4.1 for more detail, the simulations may therefore differ a bit from the real result. This is because the friction, dragging and constraint forces as well as the torque will be different for these segments and the one closest to them. During the designing of the robot this has however been taken into consideration and the electronics has been distributed so that the weight of the head and tail should match the other segments as much as possible.

The compensation for movement in a slope is not a complete model of snake movement in three dimensions but it works within the limits of this project. The problem is that the model can not describe the critical part when the robot enters or leaves the slope when some but not all the segment are tilted. During this time the segments can jam a bit which can create high torques and the simulated torques in figure 2.7 is not really accurate in that matter. The simulation is true when all segments is in the slope and the maximum torques in figure 2.7 should not differ all to much from a complete model. The maximum torques was the most interesting information behind the choice of servos and the final choice was highly over-dimensioned, therefore the slightly increased torques while entering the slope was never considered a problem. A complete model for 3D-motion would probably be necessary if the robot were to climb for example steps and would need to lift itself upwards or make similar complex moves. But since the design of the robot only involves a flexible vertical movement and such complicated movements never are performed, a complete 3D-model would not contribute with anything of value to this project.

5.2.2 3D-movement

A big choice that the project group had to take in the planning phase were whether to have a product that could actually move itself in the vertical direction or just be flexible. If the robot where to climb more complex obstacles like stairs it would need the ability to bend itself vertically in order to lift itself up. The robot could also lift parts of its body to increase the friction in the critical anchor points that produce constructive propelling force, and reduce the friction where the resulting friction is slowing down the motion. This method is used by many biological snakes to create a more efficient and fast motion, for example when they sprint or crawl on smooth or slippery surfaces [12].

The plan for achieving a controlled vertical movement was to use two servos in each segment and flip one of them 90° so that it could bend the connection in the vertical plane. This however created a great deal of problems for the designing of the segments since twice the amount of servos means that each segment must be almost twice as big. If the segments were longer, the simulations show that the movement look less smooth and authentic and the performance is generally worse than for short segments. If on the other hand the segments were wider or higher the robot would be short and fat and look less like a snake. The biggest problem though was how the robot should manage slopes and irregularities on the ground. If the servos are given a command to take a specific angle it holds it until you give it another. This means that the robot must detect all irregularities and move the vertical servos according to the ground so that the segments could follow the ground smoothly, otherwise they would be stiff and some would loose contact with the ground. In order to solve this problems the robot would need more sensors to read its surrounding and a lot of effort would be put into making the movement smooth. Moreover the cost of the robot would be significantly higher since the servos is without competition the most expensive part of the construction and twice as many would be used.

Since the goal for the 3D-motion was simply to manage a 10° slope the full movement in the vertical plane was not really necessary since the robot does not need to lift itself upwards at all, it can simply push itself up using the exact same motion as on a horizontal plane. So to save the money and trouble, the simpler solution with the flexible vertical motion were chosen. As expected it worked well and probably better than it would have with the more complex solution.

5.2.3 Source of friction

One of the greatest discussions during the project revolved around solving the problem regarding the source of friction with the task of providing the robot with the necessary anisotropic friction. This problem was solved by simply placing one passive wheel under each segment, forcing the segments to roll forward or slide sideways. By using passive wheels, the area which the friction acts on is relatively small, causing the robot to have difficulties when moving across uneven terrain as the wheels can easily get stuck or lose contact with the ground. A wheel losing contact with the ground becomes problematic as it can cause the robot to involuntarily slow down, turn or even stop as the force balance alters.

This problem can be reduced or eliminated with a different source of friction, such as rails or scales, which was two solutions that was discussed during the project. The area of which the friction acts on is larger for these solutions providing the robot with a more stable movement. However, these solutions entails difficulties regarding the anisotropic friction as the coefficient of friction is very similar in both the tangential and normal direction. Biological snakes solve this by lifting certain parts of their body to use their friction in a more efficient way. Experimenting with this seemed to be more time consuming than it would be necessary and these two solutions was therefore rejected as solutions.

5.2.4 Sensors

As presented in 3.2.2 the task of making sure the robot received visual input about its surroundings was done by equipping the robot with IR-sensors. Even though the verification test for the autonomous steering was not successful, the fact that the algorithm worked in a left oriented course and the robot tried to move away from the walls shows that the sensors work and provide this visual input.

Due to the sensors having an unstable output in the interval 0 cm to 10 cm, they might limit the robot's ability to move in narrow places. One reason why the robot sometimes hits the wall seams to be that it came to close to the wall and the sensors provided the microcontroller with wrong values. Two of the sensors used were placed 45° to the sides of the forward sensor on the head of the robot. The fact that the placement of these sensors were on the outer brim of the head give these sensors a high risk of picking up false samples when the robot moves in narrow places. This problem could be solved or at least reduced by simply replacing the existing sensors with ones capable of handling a smaller interval, or locating the sensors further inside the head.

5.2.5 Servo motors

The Dynamixel AX12+ servo chosen for this project has proven very useful in the final solution. The demands on the servos where that they should deliver an angular position based on the control signals sent from the Arduino, while managing to maintain the stress torque from the motion during operation. In the simulations the torque demand was set to 0.23 Nm while the servos has a stable motion torque of 0.3 Nm. This proved to be enough during the tests since the servos never seemed to fail at delivering the right angular position. Moreover the servos was easily programmable and the documentation around them was easy to understand which was very useful during the construction of the robot.

A limiting factor concerning the servos, is the fact that the regulation is in a closed-loop system, which cannot be changed unless physically modified. This puts a restriction on the regulation, and the user can only control the desired position through the communication interface. For this project however, this simplified regulation was a blessing because no work had to be put on the regulation of the angular actuator more than sending correct angles.

While a DC servo solution worked well for this project, it was not the only choice considered for angular actuators. The two most discussed choices in this project was having DC servos or stepper motors, which both presented a solution for the actuator demands while still managing to remain within the budget price range. However, the servo solution was chosen over the stepper motor mainly because it often contains a gearbox and a controller, whereas the stepper motors generally are sold without them. Because of the internal gearbox inside of the servo, it can reach higher torque values for a lower price and since the project required seven of the same motor, the cheaper alternative was preferable. Therefore, the DC servo solution was chosen.

5.2.6 Steering algorithm

The purpose of the steering algorithm was to fulfil the demand set in section 1.3.2; moving through a predefined course by itself without any human interference using wall detection and collision avoidance. The current implementation did not entirely fulfil this demand. The problem was that the wall detection and path decision making did not work as expected. It was hard to identify the cause of this and more time would have been needed to test and troubleshoot the autonomous steering. One thing that could have been done differently, was to simulate the steering algorithm, which would probably have given more detailed feedback on this matter. It was decided not to develop a version of the steering algorithm for the already existing simulation environment due to the time constraint. The simulation environment was previously used with success while developing the first steering algorithm. The conclusion in hindsight is that the development would probably have benefited from the simulation even considering the time constraint.

The robot steering algorithm could have made use of more states in order for the robot to be able to deal with more situations. It was initially designed with more states than only going straight or avoiding an object. But for troubleshooting purposes these states were disabled and there were not enough time to enable and test them later on. An additional state that would have improved the robot, would be to identify the situation when the robot could not find any suitable course to take to avoid an obstacle. The robot could then assume a state to perform a U-turn. While in this state it would take a sharp turn and always search for a suitable course to take. If a course was found, it would then transition back into the normal steering state. To make the path more predictable, it should have assumed a wall following state. The robot could identify this state when it steered away from a wall, but still being close enough to steer back to and follow the wall. This would make the robot alternate between this state and the steering state until the wall eventually is lost.

The algorithm also had limits defined by the sensors and microcontroller. There was no position feedback apart from the IR-sensors. This makes it harder to map the surrounding accurately, since it’s harder to really know how the saved observed data relates to new positions after moving forward and turning. The microcontroller put limits on software size. This was not really a problem in our implementation, but if more detailed information about the surrounding was needed, the 2kB read-write memory of the chosen microcontroller could have been too low.

5.3 Future work

Besides working on the demands the robot failed to meet, the robot could easily be developed further with future work. While designing and constructing the robot there has always been a thought of keeping the construction simple so that everything is easy to change and improve during future work with the robot. Each segment is therefore independent of the others and are easily connected mechanically with a few standard bolts and electrically with a contact. If one segment breaks down it could just be removed and the robot works without problem. The head is also easy to replace with a new one if for example more sensors is to be used for development of more complex manoeuvres. Besides improving the steering code and make configurations for the already existing abilities the following features could be implemented in the robot during future work without reconstruction of the whole robot.

More segments could be installed in so that the segments could make up more than one period of a wave or have smoother shape since the angular offset δ can be smaller. This could result in a more snake-like motion as well as a stronger robot because more propelling force could be generated.

Due to restrictions set early in the project, no other movement than lateral undulation have been examined. This could be a potential area of expansion for future project. Movements like side winding [1, p. 8], where the snake moves sideways could be implemented. The robot is absolutely capable of performing this manoeuvre but due to the time limit of the project, there was not enough time to develop a mathematical model and proper controlling code for this kind of movement. If the robot were to be developed further this could be a great place to start.

5.4 Conclusion

To summarise, a snake-like robot was made. The robot had the ability of moving forward using the movement lateral undulation and succeeded to meet almost all of the demands. Even though one of the demands were not fulfilled, the project in its entirety can be considered as a success regarding the time limit and the budget. The final product emulated the movement and appearance of a biological snake in a satisfactory way. The robot could however not really inherit enough properties from the snake to be better than wheel- or caterpillar band-driven robots but it was never really expected by the project. In order to make the snakes movement lateral undulation meaningful and suitable, a different source of friction other than wheels would be necessary to make the robot handle uneven terrain in a more efficient way. The resulting robot is however a great step in the right direction and has lot of room for further development.

Bibliography

[1] Pål Liljebäck. *Snake Robots*. Springer Verlag, DE, 2013.

[2] James A. Peters and Van Wallach. Encyclopedia britannica - snake, 2015.

[3] P. Liljeback, P. Liljeback, O. Stavdahl, and A. Beitnes. Snakefighter - development of a water hydraulic fire fighting snake robot. pages 1–6. IEEE, 2006.

[4] Inc. The MathWorks. ode15s, 1994. `http://se.mathworks.com/help/matlab/ref/ode15s.html`.

[5] T. Ritchey. Modeling alternative futures with general morphological analysis. *World Future Review*, 3(1):83–94, 2011.

[6] ROBOTIS. Ax-12/ ax-12+/ ax-12a, 2010. `http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm`.

[7] F Hargrave. *Hargrave's communications dictionary*. Wiley-IEEE Press, 2001.

[8] SHARP GP2Y0A21YK0F. Distance measuring sensor unit measuring distance: 10 to 80 cm analog output type, 2006. `http://www.sharpsma.com/webfm_send/1489`.

[9] Arduino™Uno. `http://www.arduino.cc/en/Main/ArduinoBoardUno`.

[10] RFI. Ack li-po. http://www.hobbex.se/sv/artiklar/ack-li-po-111v-800mah-25c.html.

[11] Texas Instruments SN74HC241DW. Octal buffers and line drivers with 3-state output, 2003. `http://www.ti.com/lit/ds/symlink/sn74hc241.pdf`.

[12] Brad Moon. Snake locomotion, 2001. http://www.ucs.louisiana.edu/ brm2286/locomotn.htm.

A

Appendix

A.1 Verification tests

To verify if the demands have been fulfilled the robot will have to pass a series of tests that are all designed to test a specific feature of the robot.

A.1.1 Lateral undulation on flat plane

First of all the robot must be able to move on a flat horizontal plane using the lateral undulation. This will be tested by simply letting the snake move forward so that the head has moved twice the length of the robot. In order for the robot to pass the test it have to move the hole distance in a smooth motion without any stops. Properties of the motion like speed and amplitude should also be changeable During this test the deviation from a straight line could also be measured in order to verify if the robot is really moving in a straight line or not.

A.1.2 Steering on flat plane

Another important demand of the robot is the ability to turn on a horizontal plane and avoid obstacles. This will be tested with a course that the robot should navigate through. The course is simply a 90° bend with walls on either side and the robot will have to pass the course without crashing into the walls. The width of the course will be decreased to test the capacity of the robot in narrow spaces. A more detailed view of the course can be seen in figure A.1.

A.1.3 Hill climbing

The robot should also be able to climb up and down smaller hills. This will be tested with another course that consist of two tilted planes that together with another horizontal make up a plateau. The slope of the plateau should be at least 10° and the robot need to pass over it in a smooth motion in order for it to pass the test. If this demand is fulfilled a steeper slope can be used to specify the robots limits. A more detailed view of the course can be seen in figure A.2

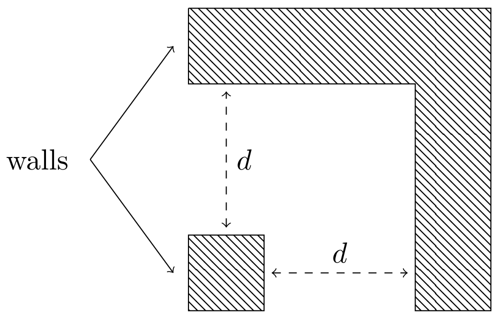


Figure A.1: The course of verification for turning on the flat plane. The robot has to detect the walls and manage to turn in the 90° bend without crashing into the walls on either side. The width, d, of the corridor should be as low as possible.

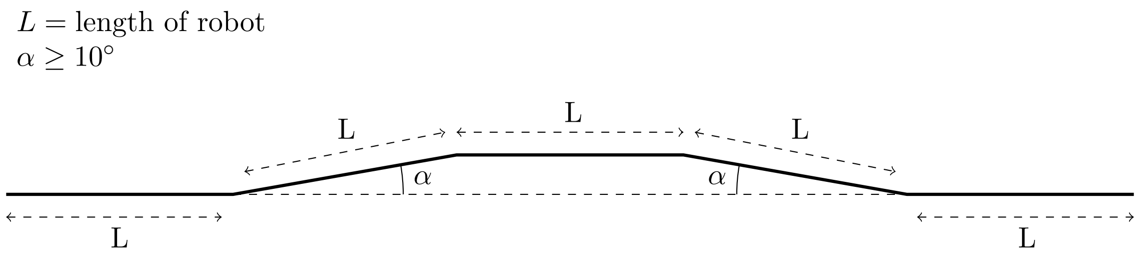


Figure A.2: The verification course for the hill climbing. In order to pass the test the robot has to climb up the hill, over the plateau and down the hill on the other side. The angle of the slope should be at least 10°.

A.2 Morphological matrix

Table A.1: The morphological matrix containing the possible solutions to the hardware specific demands. The highlighted green cells represents the choices made concerning the robot.

Demand	Option 1	Option 2	Option 3	Option 4	Option 5	Option 6
Sensor	Gyroscope	Ultrasonic	Touch	Accelerometer	Camera	IR
Ground friction	Wheels	Scales	Caterpillar band	Rails		
Source of Propulsion	DC servo	AC servo	Stepper motor	Pneumatics	Hydraulics	
Energy source	Electric battery	Electricity by wire	Accumulator			
Material	Metal	Plastic	Wood	Carbon fiber	Glass fiber	LEGO
Processor	Atmega328	Raspberry pi				

A.3 Dynamixel AX12+ Datasheet

Table A.2: Datasheet of the Dynamixel AX12+ sensor taken from [6].

Weight :	53.5g AX-12+
Dimension :	32mm * 50mm * 40mm
Resolution :	0.29°
Gear Reduction Ratio :	254 : 1
Stall Torque :	1.5N.m (at 12.0V, 1.5A)
Stable Motion Torque:	0.3N.m
No load speed :	59rpm (at 12V)
Running Degree	0° - 300°/Endless turn
Running Temperature :	-5℃ -+70℃
Voltage :	9 - 12V (Recommended Voltage 11.1V)
Command Signal :	Digital Packet
Protocol Type :	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical) :	TTL Level Multi Drop (daisy chain type Connector)
ID :	254 ID (0-253)
Communication Speed :	7343bps - 1 Mbps
Feedback :	Position, Temperature, Load, Input Voltage, etc.
Material :	Engineering Plastic

A.4 Budget

The budget presented is cost of making the robot. The budget does not include the actual cost of the entire project.

Table A.3: The budget for the hardware of the robot.

Component	**Amount**	**Cost(SEK)**
Dynamixel AX-12+	7	2455
LiPo Battery 11,1V 800mAh	1	179
Arduino™Uno	1	279
Sharp GP2Y0A21YK0F	3	240
SN74HC241DW	1	7
Molex 50-37-5033	30	23
Molex 08-70-1040	100	139
LEGO Wheels	8	80
Total cost		**3402**

B

Appendix

B.1 The design development

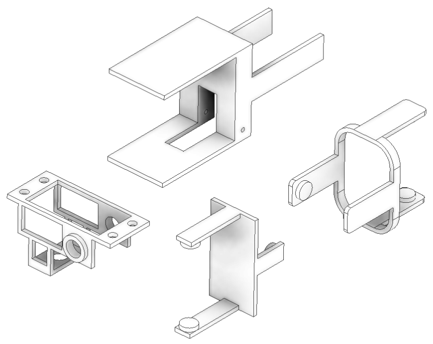


Figure B.1: The very first ideas for the structure

It all started with trying to merge the basic ideas for the fundamental function of the design. As can be seen in figure B.1 the first sketches are quite similar to the final design. Each part went through a development process that made it change with aspects on new demands and requirements but also some wishes.

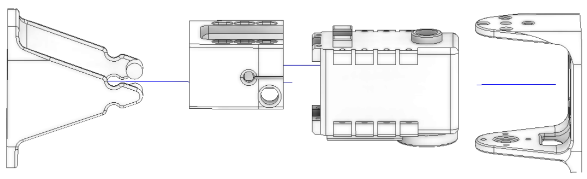


Figure B.2: Exploded view of one segment, the parts are from the left, the flexible arm for the 3D motion, the base, the servo and the propulsion arm.

The principle of the design has not changed much during the time of the development. As seen in the figure B.2 the design of the section consist of three designed parts and a servo. At the beginning of the project, the aim was to have two servos in each segment for complete 3D manoeuvrability. But since that idea was discarded, the principle for the design started to take form and have not been changed much since then.

One of the biggest difference is the strength of the parts. The final parts are much stronger and thicker than the first ones. The first printed parts worked as experiment to see what the printed plastic parts could endure.

The most challenging design problems was to solve the 3D flexibility demanded according to 3.1. Since the manufacturing method was to use a 3D-printer, the part seen in figure B.1 had to be designed in separate parts.

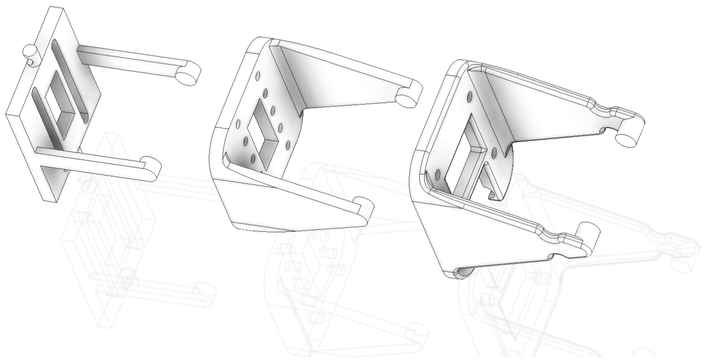


Figure B.3: The development of the flexible arm

The first and easiest solution was to let the joint between the flexible arm and the base be free, but this resulted in that every module segment tilted and made the robot's movement harder to control. Something that makes the joints more stable was now needed. After some thought the solution was to use rubber bands to add a rotational resistance on each joint, rubber bands were chosen because they are easy to apply to the design. In the figure B.3 a track can be seen on the right parts arm. It is created for the band to be stable, and not move around.

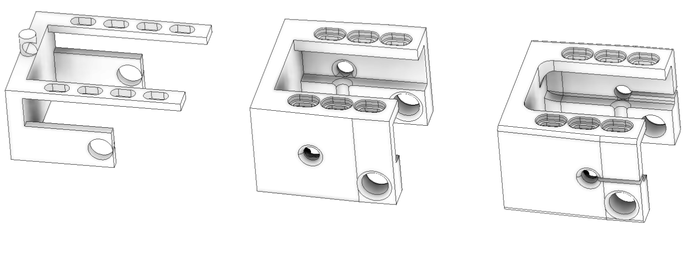
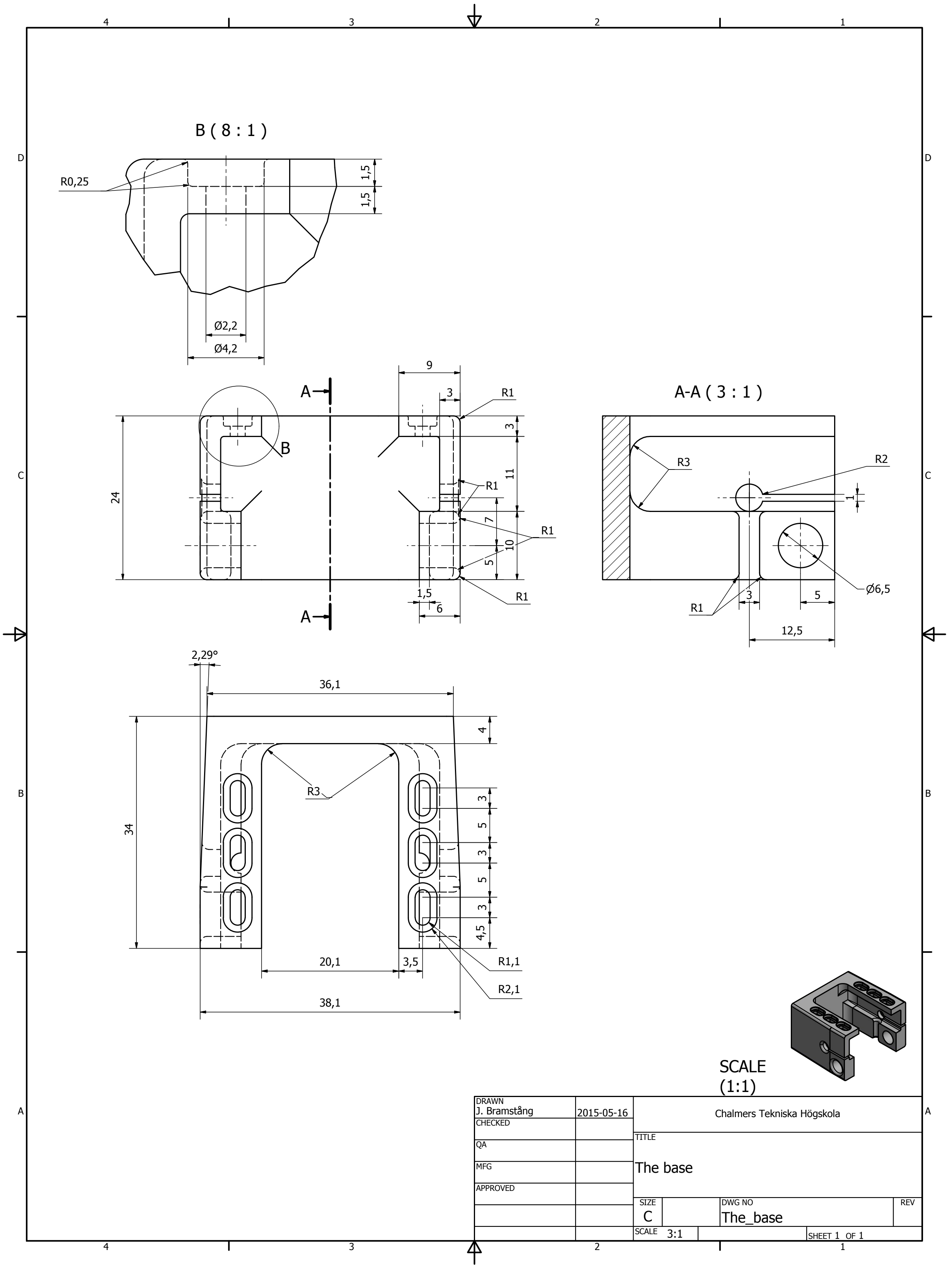


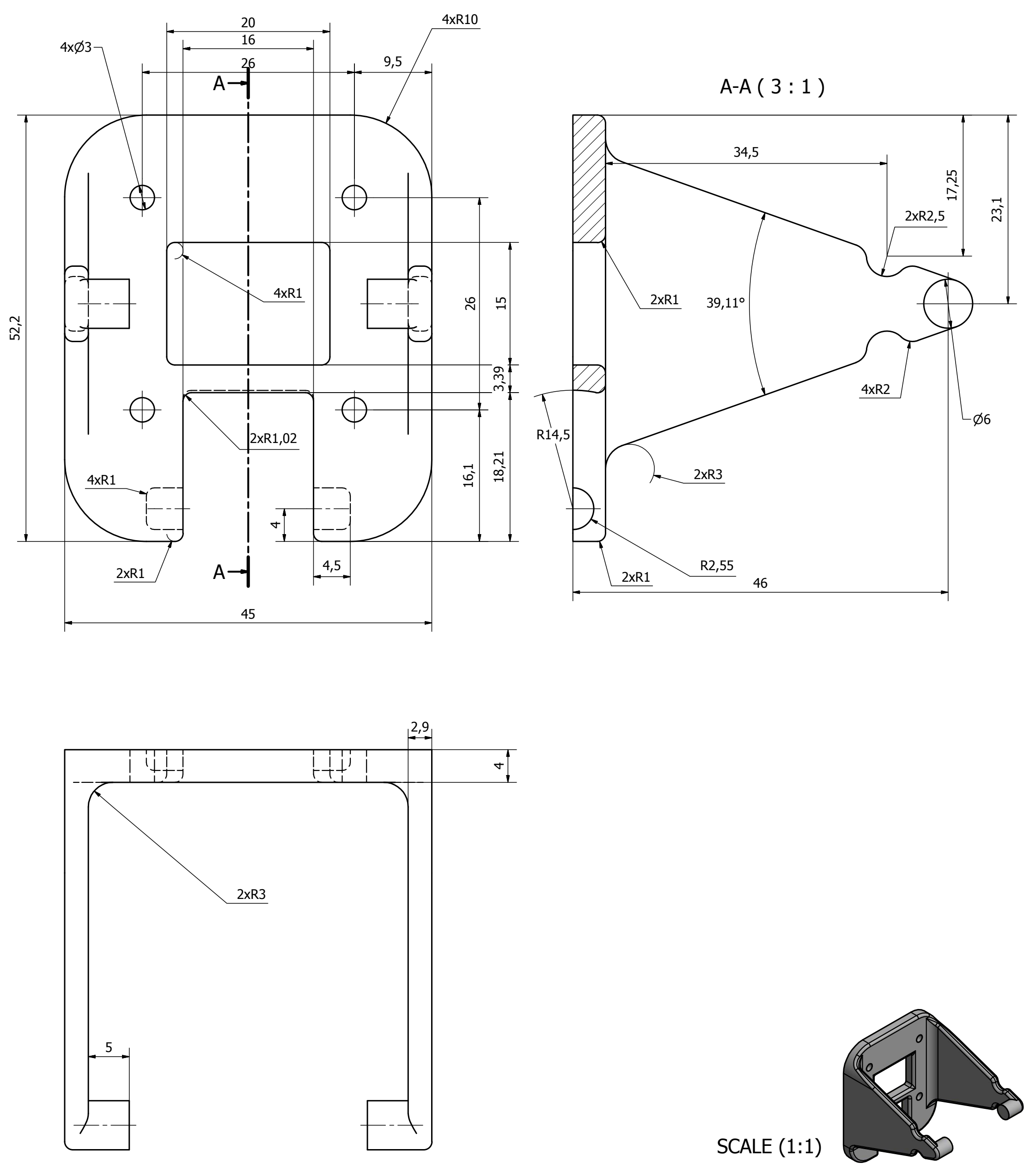
Figure B.4: The developmet of the section base

The solution to the flexibility also made an impact on the section base. Figure B.4 shows the development of the base and the biggest difference between the two latest ones is related to the flexibility solution.

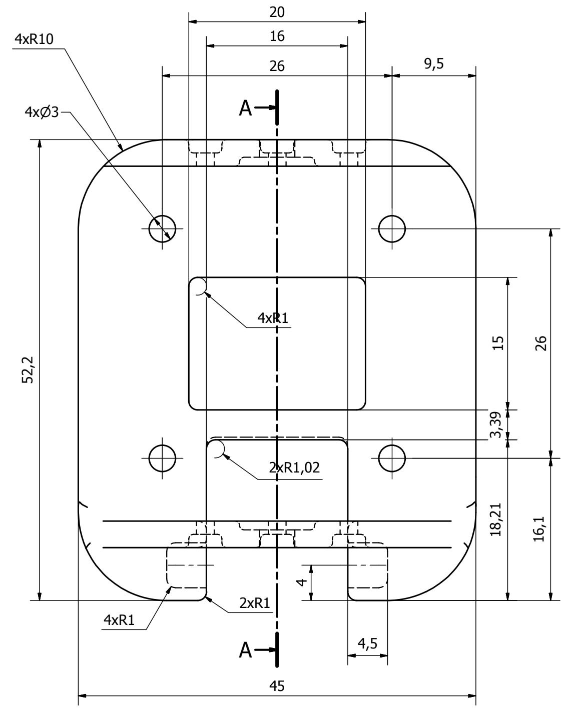
B.2 Drawings

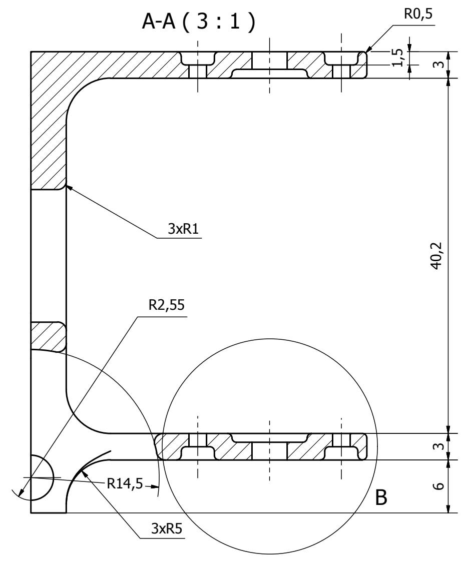


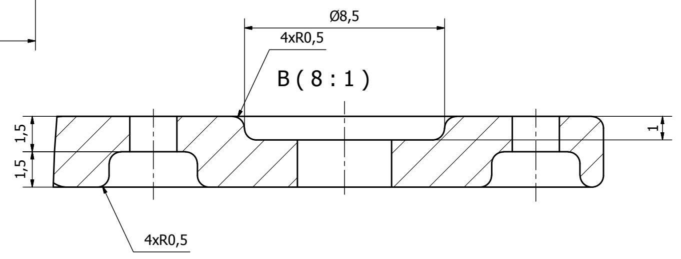
DRAWN J. Bramstång	2015-05-16	Chalmers Tekniska Högskola
CHECKED		
QA		TITLE
MFG		The base
APPROVED		
		SIZE C — DWG NO The_base — REV
		SCALE 3:1 — SHEET 1 OF 1

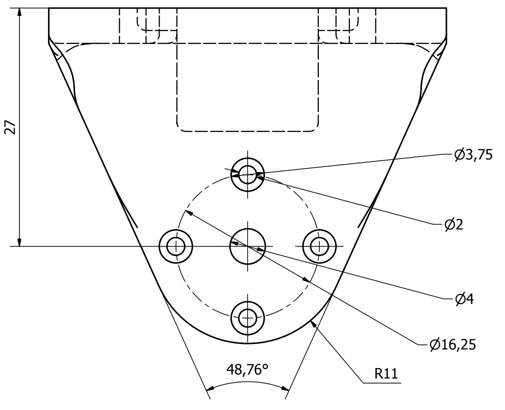


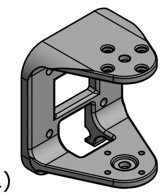
DRAWN J. Bramstång	2015-05-16	Chalmers Tekniska Högskola
CHECKED		TITLE The flexible arm
QA		
MFG		
APPROVED		SIZE C · DWG NO The_flexible_arm · REV
		SCALE 3:1 · SHEET 1 OF 1





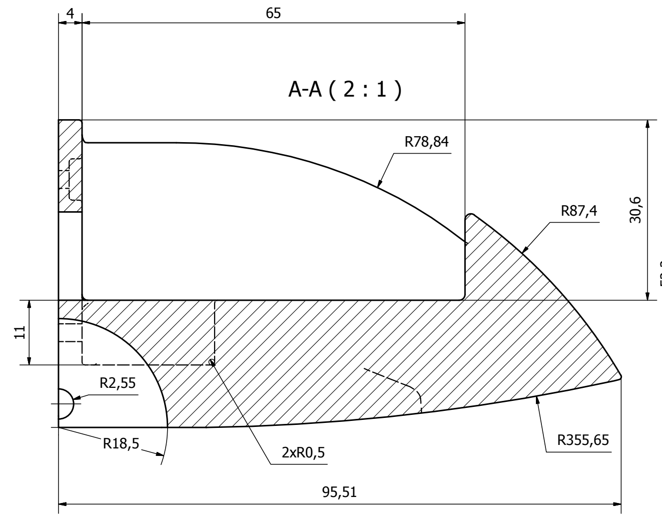


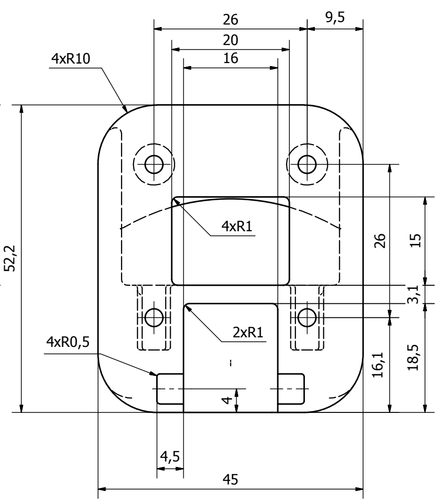


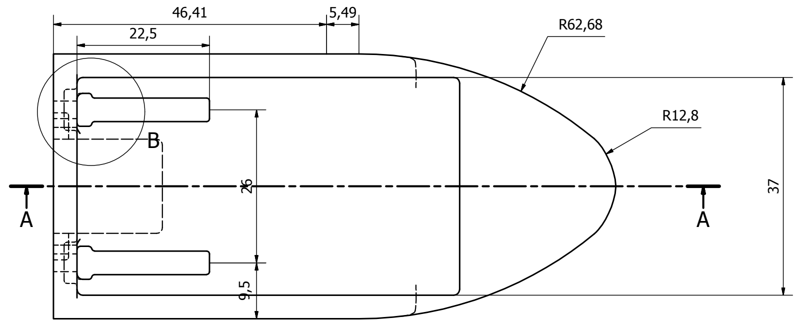

SCALE (1:1)

DRAWN J. Bramstång	2015-05-16	Chalmers Tekniska Högskola
CHECKED		TITLE
QA		The propulsion arm
MFG		
APPROVED		SIZE C / DWG NO The_propulsion_arm / REV
		SCALE 3:1 / SHEET 1 OF 1

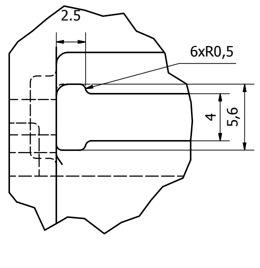
A-A (2 : 1)



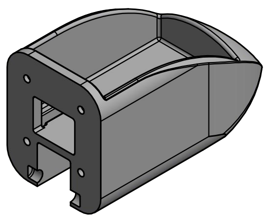




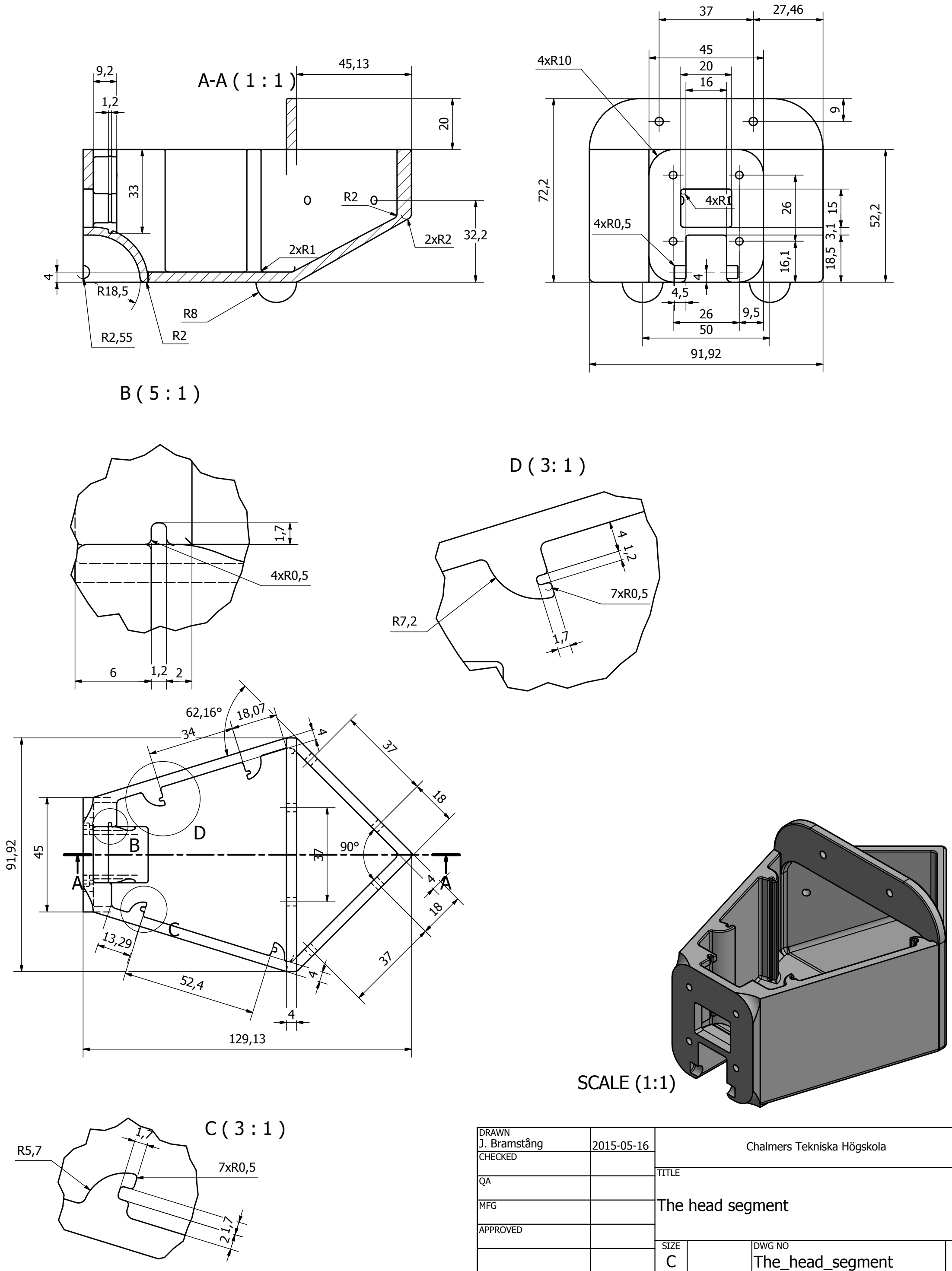
B (4 : 1)



SCALE (1:1)



DRAWN J. Bramstång	2015-05-16	Chalmers Tekniska Högskola			
CHECKED		TITLE			
QA		The tail segment			
MFG					
APPROVED					
		SIZE C	DWG NO The tail segment		REV
		SCALE 2:1		SHEET 1 OF 1	



DRAWN J. Bramstång	2015-05-16	Chalmers Tekniska Högskola	
CHECKED		TITLE	
QA		The head segment	
MFG			
APPROVED		SIZE C	DWG NO The_head_segment	REV
		SCALE 1:1	SHEET 1 OF 1

B.3 The first iteration of the autonomous steering algorithm

The current implementation is the second algorithm that was tried. The first one, which is presented below, did not include enough functionalities. For example, it steers by only taking decisions of when the front sensor is directed forwards, which only happens two times per period. This can result in too slow decision making. Another problem was that when it tried to avoid an object, it applied steering for a certain period of time without any feedback. This would cause both under- and over-steering.

The autonomous steering is basically designed to pass the second verification test, namely to move in a narrow corridor that turns 90°. This is made with three IR-sensors located on the head as in figure B.5. One is facing straight forward in the same direction as the head and the other two is pointing forward but angled 45° from the direction of the head, one to the left and one to the right. The general idea is that the forward pointing sensor can detect distance to walls in front of the robot and the side pointing sensors can detect the distance to the walls on the sides.

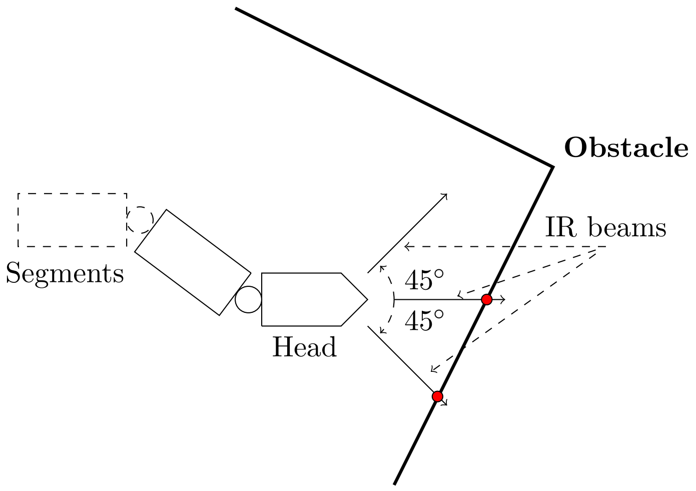


Figure B.5: The sensors are mounted so that one points straight forward from the head and two points 45° to the left and right.

In order to beat the second verification test, the robot must be able to correct its course according to the corridor so that it moves straight without hitting the walls, but also detect when the 90° turn comes and make make corrections to its course accordingly. The microcontroller does this by splitting all possible readings from the sensors into three different cases in which three different actions are to be done. The cases are

1. The forward sensor and some or both of the side sensors all detects walls - this means that the robot should make a 90° turn since this is most likely the corner of the corridor. The direction of the turn is chosen so that it moves away from the closest of tho walls to the sides.

2. Only the forward sensor detects a wall - this still means that the robot must do a 90° turn since but the direction is unknown and should be randomised.

3. Some or both of the side sensors detects walls but not the forward sensor - this means

that the robot is heading towards one of the sidewalls and need to make a little turn away from it. The direction of the turn is away from the wall that is closest.

The problem now is that the head is constantly turning from side to side together with the rest of the segments so that in a small corridor the forward sensor will all to often detect walls that are really to the sides and not in front of the robot. This is solved by always reading the sensors when the joint angle of the head ϕ_{N-1} is the same as the general direction of the robot $\theta_{heading}$ which happens two times during each period of the joint angles (2.61). Since only the joint angles $\boldsymbol{\phi}$ is known to the robot and not the link angles $\boldsymbol{\theta}$ the microcontroller must calculate if the following condition is fulfilled for it to know that the head is facing forward

$$\begin{aligned}
\theta_N = \theta_{heading} &= \frac{1}{N}\sum_i \theta_i = \frac{1}{N}\sum_i (\boldsymbol{H}\bar{\boldsymbol{\phi}})_i \\
&= \frac{1}{N}\left((\phi_1 + \phi_2 + ... + \phi_{N-1} + \theta_N) + (\phi_2 + \phi_3 + ... + \phi_{N-1} + \theta_N) + ... + (\theta_N)\right) \\
&= \frac{1}{N}\phi_1 + \frac{2}{N}\phi_2 + ... + \frac{N-1}{N}\phi_{N-1} + \theta_N \\
\Rightarrow \phi_{N-1} &= \frac{1}{1-N}\phi_1 + \frac{2}{1-N}\phi_2 + ... + \frac{N-2}{1-N}\phi_{N-2}.
\end{aligned} \tag{B.1}$$

If the above condition is fulfilled with a certain tolerance the head is facing forward and the microcontroller can read the sensors. The joint angle offset Φ_0 is thereafter changed according to

$$\Phi_0 = \Phi_{big/small} sign(d_{right} - d_{left}) \tag{B.2}$$

where $d_{right/left}$ is the distance to the left and right wall, $\Phi_{big} = 15°$ and $\Phi_{small} = 2°$ is the changes in Φ_0 that corresponds to a big and small turn. The *sign* function determines the direction of the turn since a positive Φ_0 means that the robot turns to the right.

B.4 Simulation display tool

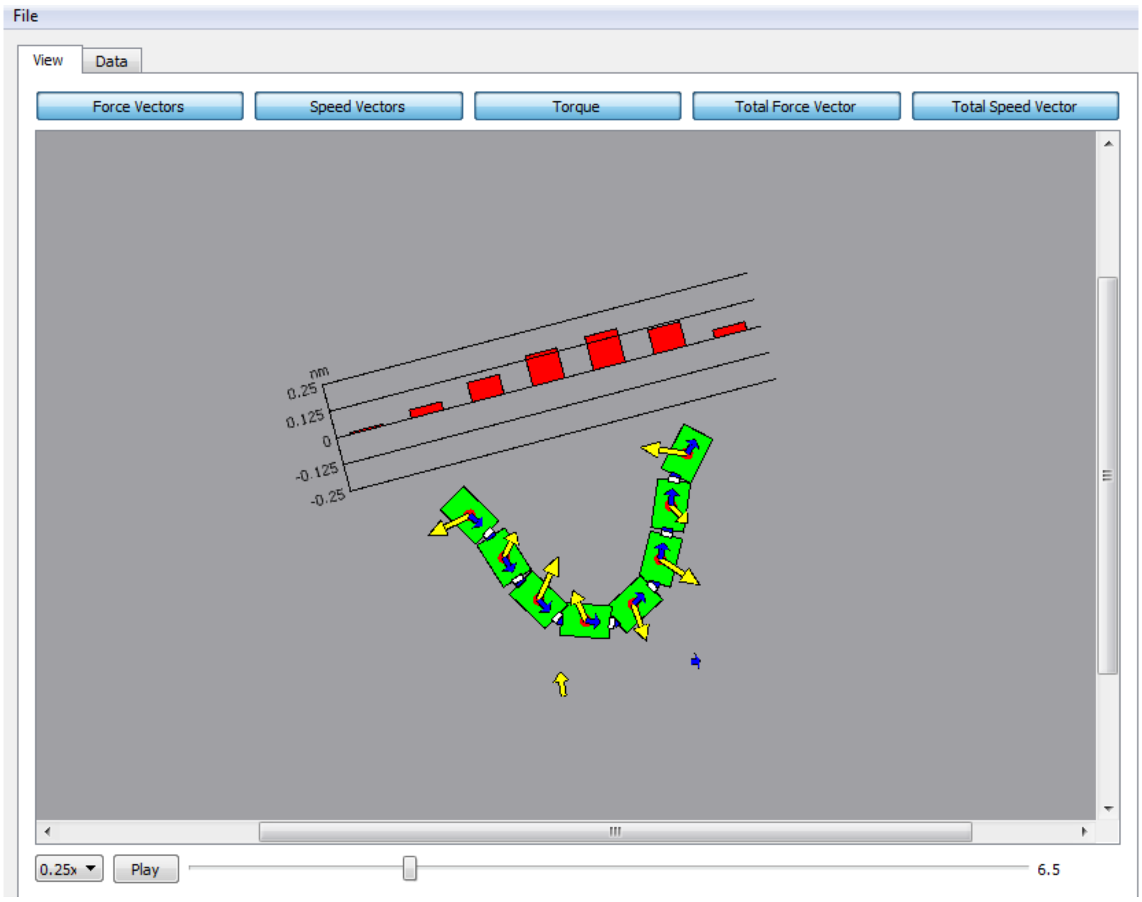


Figure B.6: A view of the robot simulation, 1.5 seconds after it has initiated a turn to its left relative to the direction of its velocity , displayed by a tool written in C++. The tool displays forces (yellow arrows), velocities (blue arrows) and torques (red bars). The blue and yellow arrows outside of the robot shows the velocity and the resulting force acting on the robot's centre of mass, respectively. By default the simulation will be presented in real time, but the option exists to view the run in slower or higher speeds. It's easy to view any part of the simulation by just dragging the bar to any desired point in time.