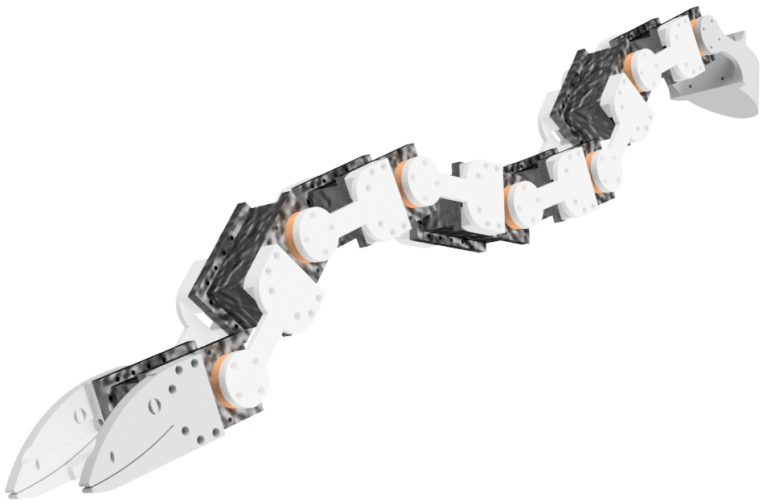
CHALMERS
UNIVERSITY OF TECHNOLOGY



Modeling, Design and Construction of a Snake-Like Robot

Bachelor thesis

VIKTOR HALLÉN
JOHAN IVARSSON
JOAKIM JANSSON
MARTIN JOHNSSON
ANDRÉ OLSSON
MAX REIDAL

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

Chalmers University of Technology
Gothenburg, Sweden 2017

Supervisor: Yiannis Karayiannidis, Department of Signals and Systems
Examiner: Jonas Fredriksson, Department of Signals and Systems

Abstract

In this project you will follow the process of modeling, designing, and constructing a segmented and modular snake-like robot that can successfully traverse a constructed obstacle course. The robot is adaptable with the help of a sensor and different movements. The forwarding locomotion consists of a pulse movement as well as a rolling movement for longer distances. It is also able to climb vertical obstacles up to a height of 24cm, with the robot being 60cm long, and climb slopes up to 27°. All the movements were constructed with a geometrical analysis.

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

In today's modern society, technology is advancing at a rapid speed, computers are made smaller, and the development of robots is increasing[1]. This has opened up many new avenues of possible applications for robots. Such as robots that can travel through harsh terrain in, for example, search-and-rescue missions. A harsh terrain poses many challenges to the locomotion of the robots[2]. A good source of inspiration while searching for solutions to this problem is nature. Among nature's more interesting forms of movements are the movements of serpents and snakes and more specifically, how they are efficiently able to access the most restricted areas. A more in-depth analysis of the biomechanical study of snakes can be found in the book by Harvey B. Lillywhite (2014)[3].

1.1.1 Snake-like robots

Snake-like robots are a subject that has been popular for quite some time with the first research on the area conducted in 1972 and has since then been continued upon, with the robots becoming more and more advanced[4][5]. Snake-like robots have many applications and can be categorized into two kinds of robots, segmented and continuum robots[5].

Segmented robots are made up of multiple, usually similar segments that with the help of motors generate movement in one or more axes. By re-positioning the segments with the support of the motors movement can be generated. Advanced versions of these robots are usually constructed such that each segment provides rotation in more than one axis, while less complex ones only provide rotation in one axis. This enables the more advanced robots to have the ability to use more complex forms of movement. Segments and their modulated property lead to the option of increasing or decreasing the number of segments that a robot includes. Some concept systems have automated the process of self-assembly and -disassembly to such a degree that the system can do it automatically[6].

A common way to generate movement with segmented snake-like robots is in the form of serpentine movement, i.e. shifting the body parallel to the ground to generate propulsion (see Fig. 1.1).

Continuum robots have, with the help of minuscule segments, a big amount of variability. This, in addition to an external source for propulsion, gives it a multitude of applications. In general, the movement is generated by using hydraulics or wires to produce contractions in the robot[5]. The big difference, between segmented and continuum robots, is that while segmented robots use their segments to achieve locomotion, continuum robots are usually used as manipulators. These kinds of manipulators are e.g. found among surgical robots[7].

Figure 1.1: Figure representing the serpentine movement often found in biological snakes. (Picture:Rushenb[8]).

1.1.2 Modular Robots

The meaning of the word modular differs in what context it is used. In the case of robotics, it is used to describe a robot with several generic modules that can change the connection and order of these modules for better adaptation and flexibility. This not only greatly cooperates with automation of robots, but also allow for greater effectiveness in one specialized area. In the context of snake-like robots, each segment in a segmented snake-like robot can be represented by a module[9].

1.2 Purpose

The purpose of this project was to model, design and construct a snake-like robot that can efficiently move over harsh terrain using a sensor to identify obstacles and adapt its movement accordingly.

The concept of snake-like robots is quite broad and allows many opportunities for this project to use unique implementations and solutions. Even so, the project's primary purpose was found in the members' practice and union of different fields, so as to familiarize themselves with the boundary between theoretical design and physical implementation.

1.3 Problem description

Expanding upon the purpose, the problems were narrowed to specifications. Other than setting demands of climbing obstacles and slopes, an objective was set such that the robot could move faster in an unhindered environment rather than in obstructed areas. It should also be able to base its movement pattern on the terrain in front of it and, therefore, become adaptable. Due to constraints in resources, this robot will be restricted to eight segments.

Specifications, to fulfill the purpose of the project, are as follows:

- The robot needs to be able to move forward in at least two different ways.
- The robot needs to achieve its propulsion through manipulation of separate segments.
- The robot needs to have an appearance that is similar to the body of a snake.
- The robot needs to be able to ascend an incline of at least 20°.
- The robot needs to successfully climb a vertical obstacle reaching a height of at least a fourth of its length.
- The robot should be modular and adaptable.

1.4 Methodology

We chose to divide the problem into smaller sub-problems. Our method for dealing with these problems was by splitting the project into three major areas: **Modeling**, **Design**, and **Implementation**. These areas, in turn, got divided into sub-areas. The intended workflow of the project was such that the **Modeling** works on the theoretical parts of the project, while **Implementation** and, to an extent, **Design** applied the theoretical basis.

The modeling and the relating simulations used geometric analysis to create mathematical expressions for the robot. This is extended upon by a basic mechanical analysis for critical points. If reconfiguration is needed, finding it at this stage costs fewer resources than discovering them during testing.

The **Design** area focused on the physical design of the robot and also motivated hardware choices. Several of these choices were motivated using the data produced in **Modeling**. This also included the sensor, which was the tool used for identifying the path ahead.

In this project, the **Implementation** involved programming of the algorithms developed through modeling, but it also includes the implementation of the distance sensor. Further, it contains the integration between the distance sensor and the implemented movements to create the adaptive part of the robot.

2
Modeling

In this project, a snake-like robot was designed that can traverse a designed obstacle course (further described in 3.4), where the obstructions consist of vertical objects as well as a slope. Since there was a focus on climbing, and the project had limited resources, the robot was constrained to only be able to move in the vertical plane and therefore unable to move sideways.

Symbol	Description
l	Length of a segment
k	Distance moved relative to a segments length
ψ_i	The angle for rotational joint number i
h	Height of an obstacle
m	Mass of a segment
g	Gravitational acceleration
ρ_s	Angle of a slope
μ	The friction coefficient

Table 2.1: Table of recurring variables in this chapter.

2.1 Movement

In this section, three different kinds of movements are sketched and evaluated to achieve the objectives of the project. These movements include two ways of forwarding locomotion and different variations of a climbing algorithm. Climbing slopes also take advantage from high friction which disables the robot from both pushing and dragging segments that are in contact with the ground. This implies that to achieve forward locomotion; the robot needs to lift the back-end of its body and then place it a distance ahead, followed by moving the front-end forward.

The three different movements were developed to be specialized for various tasks. The first movement introduced is the *main movement*, which is also one of the forward movements. This is used to traverse shorter distances accurately and was therefore designed as a pulse-movement. For longer distances, a second forward locomotion was made, referred to as the *fast movement*, which resembles the "rolling" movement of a crawler belt. To make the climbing as effective as possible, three similar sequences were made. This was designed such that the robot could efficiently climb different heights.

2.1.1 Main movement

This movement was first sketched on paper and afterward made into a simplified model that was simulated in Matlab. The simulation revealed that when positioning the segments based on the simplified model, it would result in the segments being unnecessarily pushed or dragged. This means that the simplified movement was not unaffected by friction. Therefore, another model was made such that all segments touching the ground would stay static. From the simplified model, it could be concluded that one can base the angles between the segments upon how far the pulse should move the robot forward. In the following, we derive the kinematic model that connects the angles of the joints to the distance covered by the robot. The distance is parameterized as $k \cdot l$, where k is how far the tail is moved relative to a segment's length, and l is the length of a segment.

Figure 2.1 shows the position of the robot after the tail is moved, where α, β, and γ are the angles in this position. Before moving to this position, the tail is lifted just enough not to get dragged on the ground. With the help of geometry and the law of cosines, the angles are evaluated by the following equations:

$$\alpha = \arccos\left(\frac{12 - 6k + k^2}{12 - 4k}\right) \tag{2.1}$$

$$\beta = \arccos\left(\frac{6k - 4 - k^2}{4}\right) \tag{2.2}$$

$$\gamma = \arccos\left(\frac{6 - 6k + k^2}{6 - 2k}\right). \tag{2.3}$$

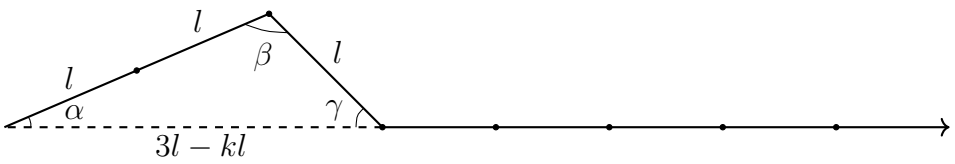


Figure 2.1: This is an illustration of the first position in the *main movement*. Here the segments are angled according to α, β, and γ. The dots in the figure represents rotational joints which are controlled by motors, and the arrow shows the direction of the head segment.

The next step is to place down the tail on the ground. To do so without pushing the tail backward, the angles always need to follow each other in a specific way (see Fig. 2.2). Using geometry and the law of cosines every angle was made dependent on angle v, this resulted in Equations (2.4) - (2.9).

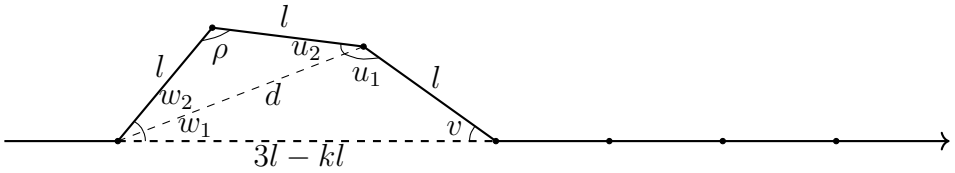


Figure 2.2: This figure shows an example of how the robot has to be positioned between the steps for the robot to not push any segments along the ground.

$$d = l\sqrt{10 - 6k + k^2 - (6 - 2k)\cos(v)} \tag{2.4}$$

$$\rho = \arccos\left(\frac{2l^2 - d^2}{2l^2}\right) \tag{2.5}$$

$$u_1 = \arccos\left(\frac{d^2 + l^2 - 9l^2 + 6kl^2 - k^2l^2}{2dl}\right) \tag{2.6}$$

$$u_2 = \arccos\left(\frac{d}{2l}\right) \tag{2.7}$$

$$w_1 = \arccos\left(\frac{d^2 + 8l^2 - 6kl^2 + k^2l^2}{6dl - 2dkl}\right) \tag{2.8}$$

$$w_2 = u_2 \tag{2.9}$$

where $u = u_1 + u_2$ and $w = w_1 + w_2$ for a convex quadrilateral and $u = u_1 - u_2$ and $w = w_1 - w_2$ for a concave quadrilateral. With the help of these equations, the angles can now be manipulated such that all the segments touching the ground stay static. The next step is to move from the first position shown in Fig. 2.1 to the second position shown in Fig. 2.3. This position is when the entirety of the tail is placed on the ground. The angles σ and δ are calculated by the use of geometry and the law of cosines as follows:

$$\delta = \arccos\left(\frac{4k - 2 - k^2}{2}\right) \tag{2.10}$$

$$\sigma = \frac{\pi - \delta}{2} \tag{2.11}$$

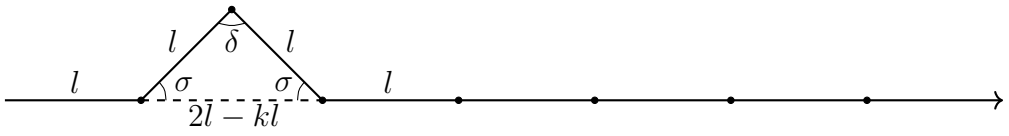


Figure 2.3: Illustration of the second position in the *main movement*.

The last position to mention is as shown in Fig. 2.4.

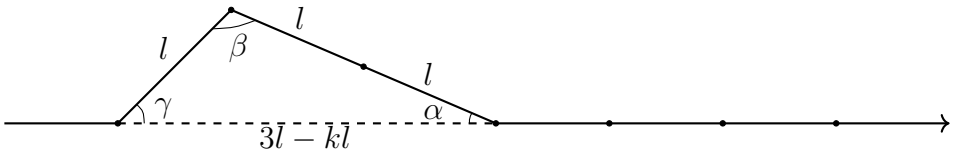


Figure 2.4: Illustration of the last position.

From this position, the robot only has to repeat the steps in the same order, with the only difference that the positions are moved one segment forward. When this is repeated through all the segments of the robot, it has moved $k \cdot l$ cm. By defining the angles α, β, γ, δ, and σ and expressing them with respect to k, we can easily generate a reference for the motors. Even though this movement is not efficient for traveling long distances, it is very useful for moving shorter and more exact distances.

2.1.2 Fast movement

The *fast movement* consists of three steps: Assembling, "rolling", and disassembling. For the assembly step, the robot folds the tail to the head as depicted in Fig. 2.5, while the disassembly step is simply the assembly done in reverse. The "rolling" motion is based on repeating one step, but to more easily understand how the step iterates it is split into two, as shown in Fig. 2.6. It is important to note that the robot will only pass through the first step when it is moving from the start position to the second step. After this movement has been repeated eight times, one for each segment, the robot has completed one cycle and has returned to the start position. Afterward, the robot can either disassemble or move another cycle. This movement is more efficient than the *main movement* because every iteration of the motors makes the robot move a whole segment. The only downside of the *fast movement* is the constraint of having to complete an entire cycle, making it unusable for distances shorter than the robot's length. An earlier example of this movement is found in ACM R7, presented in 2010 by Ohashi and Hirose[10].

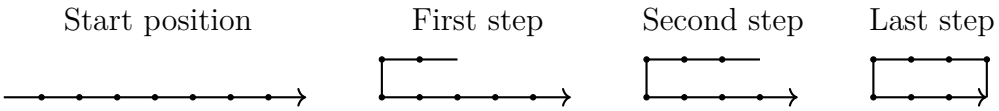


Figure 2.5: Illustration of the assemble for the *fast-movement*.

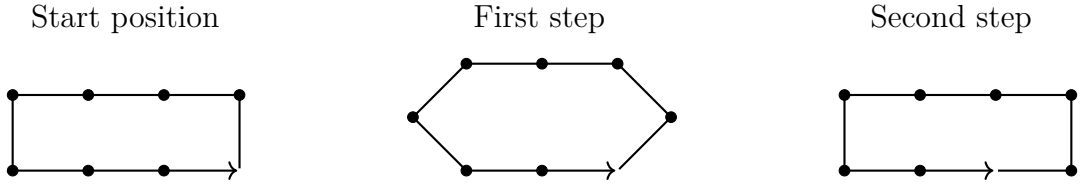


Figure 2.6: Illustration of the actual movement for the *fast movement*.

These two types of movement can now compensate each other such that the robot can choose movement pattern according to the situation. If the robot is going to move a distance shorter than its length, it will use the *main movement*, and if the distance is further than the length of the robot's body, it will use the *fast movement*. For the *fast movement*, the tail and head have to join without pushing each other. It is also important to note that when using the *fast movement*, the robot has to make an entire turn, i.e. it moves the whole length of its body. Therefore, the robot has to identify objects further away than its length.

2.1.3 Climbing movement

Climbing a vertical object can be done in different ways. The most important outcome is to end up with as many segments on the obstacle as possible, such that the robot can lift the remaining segments without falling backward. The climbing can be done by pushing the robot against the wall while lifting the first parts of the robot, one at a time. The challenge during this type of climbing is to move forward while using fewer parts, as more and more parts are in the air. Therefore, another way of climbing was developed where the robot balances on the head and lifts the tail as high as possible. By testing and doing a minor analysis of the forces and torque for getting to the last position, it was clear that balancing on one segment would be too hard and unreliable for the actual robot. However, balancing on two segments is more stable and was used in the development of the climbing algorithm.

Depending on the height of the object, three different variations of the climbing algorithm were designed. In this way, the robot can choose a faster and more reliable method every time an object appears. The robot starts by getting into the climbing position as shown in Fig. 2.7, where h is the height of the object, $\psi_5 = 110°$, and $\psi_3 = 45°$.

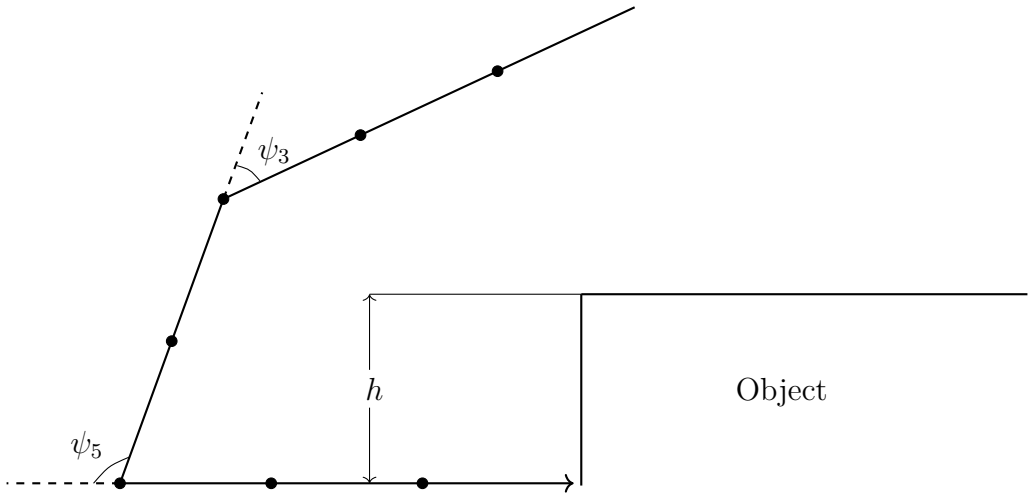


Figure 2.7: This figure illustrates the starting position for the climbing. Here the back-end of the robot is lifted and tilted over the object.

The first variation of the climbing algorithm was designed to make the robot able to

climb heights up to one and a half segment efficiently. To make the climbing as stable as possible, the robot leans on the object with the tail, as shown in Fig. 2.8, where S is a distance that can be adjusted for optimization, $\psi_6 = 90°$, $\psi_5 = \arccos\left(\frac{S+h-l}{l}\right)$, and $\psi_4 = \arccos\left(-\frac{S}{4l}\right) - \psi_5$.

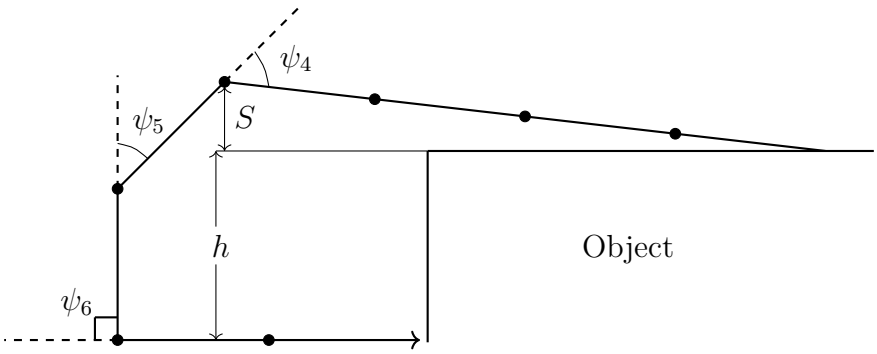


Figure 2.8: Illustration of the first position in the climbing of low objects.

In order to move closer to the object the robot lifts itself by setting $\psi_7 = 90°$ while at the same time changing the rest of the joints as shown in Fig. 2.9, where $\psi_6 = \arccos\left(\frac{S+h-l}{l}\right)$, and $\psi_5 = \arccos\left(-\frac{S}{5l}\right) - \psi_6$.

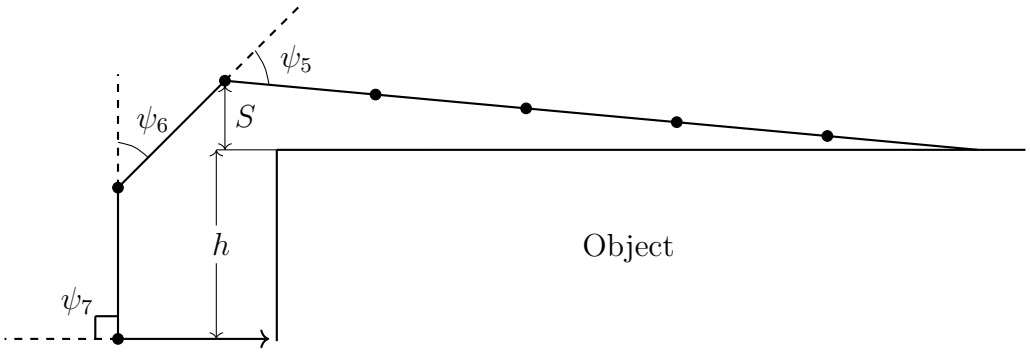


Figure 2.9: Illustration of the second position in the climbing of low objects.

In the last step, the robot gets as close to the object as possible through diminishing ψ_7, because this will make the robot balance on the head instead of falling backward while the center of mass is in front of the head. At the same time, the rest of the joints changes, making the robot end up in the position shown in Fig. 2.10, where $\psi_7 = \arccos\left(\frac{S+h-l}{l}\right)$ and $\psi_6 = \arccos\left(-\frac{S}{6l}\right) - \psi_7$.

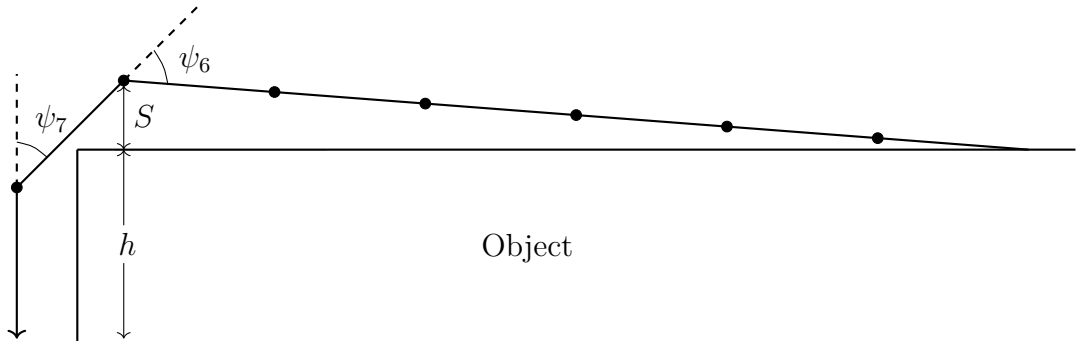


Figure 2.10: Illustration of the third position in the climbing of low objects.

The robot now has to flatten out to complete the climbing process. However, as a result, the robot is turned upside down with the tail in front. As a consequence, it has to fold back to the normal state with the head in front, which is explained later on in this section.

Climbing objects higher than one and a half segments but lower than two and a half segments starts with the same opening position as before (see Fig. 2.7). This climbing is, for the most part, the same except for some small modifications. As before, the tail is placed on the object to make the robot stable (see Fig. 2.11), where $\psi_5 = \arccos\left(\frac{S+h-l}{2l}\right)$, and $\psi_3 = \arccos\left(-\frac{S}{3l}\right) - \psi_5$.

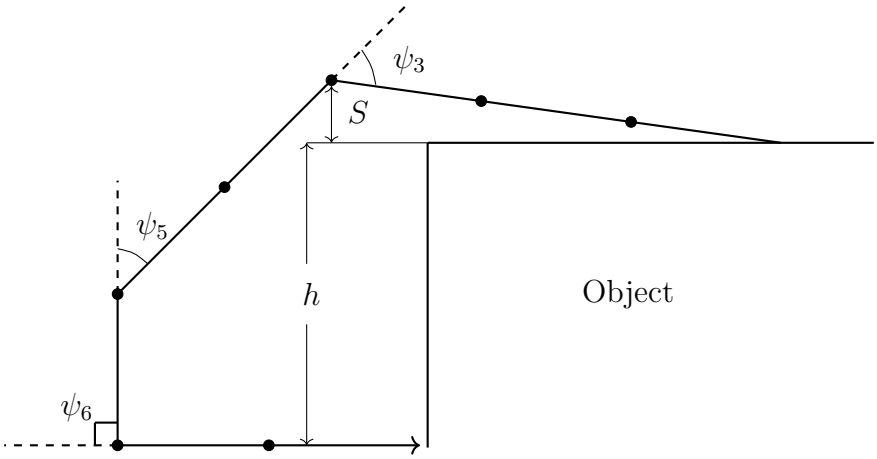


Figure 2.11: Illustration of the first position during the climb of objects of medium height.

The tail is then pushed to the position shown in Fig. 2.12, where $\psi_5 = \arccos\left(\frac{S+h-2l}{l}\right)$, and $\psi_4 = \arccos\left(-\frac{S}{4l}\right) - \psi_5$.

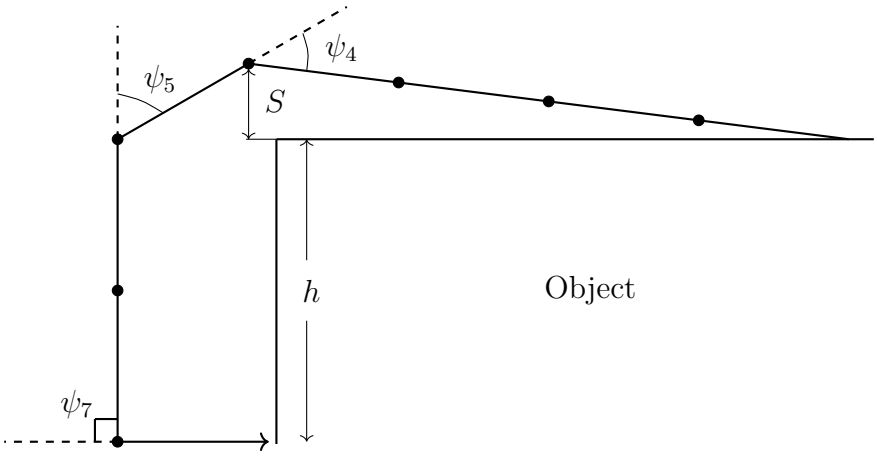


Figure 2.12: Illustration of the second position during the climb of objects of medium height.

Lastly, it achieves the position shown in Fig. 2.13, where $\psi_6 = \arccos\left(\frac{S+h-2l}{l}\right)$, and $\psi_5 = \arccos\left(-\frac{S}{5l}\right) - \psi_6$. From this position, the robot executes the same movements as for lower objects.

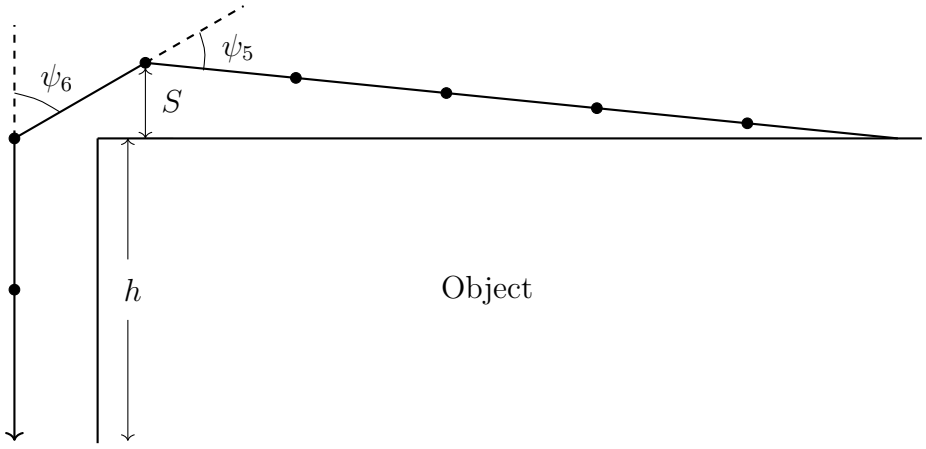


Figure 2.13: Illustration of the third position during the climb of objects of medium height.

To climb objects higher than two and a half segments, a final variation was made, such that the robot could still end up with more than four segments on the object to avoid falling backward. This means that the theoretical model can climb heights less than four segments. In the same way as the previous climbing variations, the start position is first achieved as shown in Fig. 2.7. Afterward, the robots start the climbing algorithm by reaching the position illustrated in Fig. 2.14, where $\psi_5 = \arccos\left(\frac{S+h-l}{3l}\right)$, and $\psi_2 = \arccos\left(-\frac{S}{2l}\right) - \psi_5$.

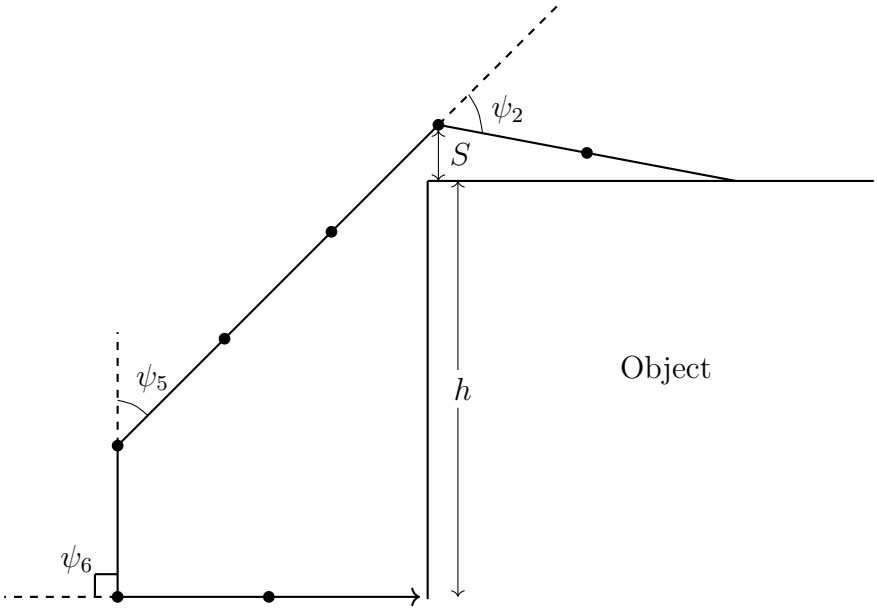


Figure 2.14: Illustration of the first position during the climb of objects of large heights.

After the robot has achieved the first position, it moves into the next position shown in Fig. 2.15, where $\psi_5 = \arccos\left(\frac{S+h-2l}{2l}\right)$, and $\psi_3 = \arccos\left(-\frac{S}{3l}\right) - \psi_5$.

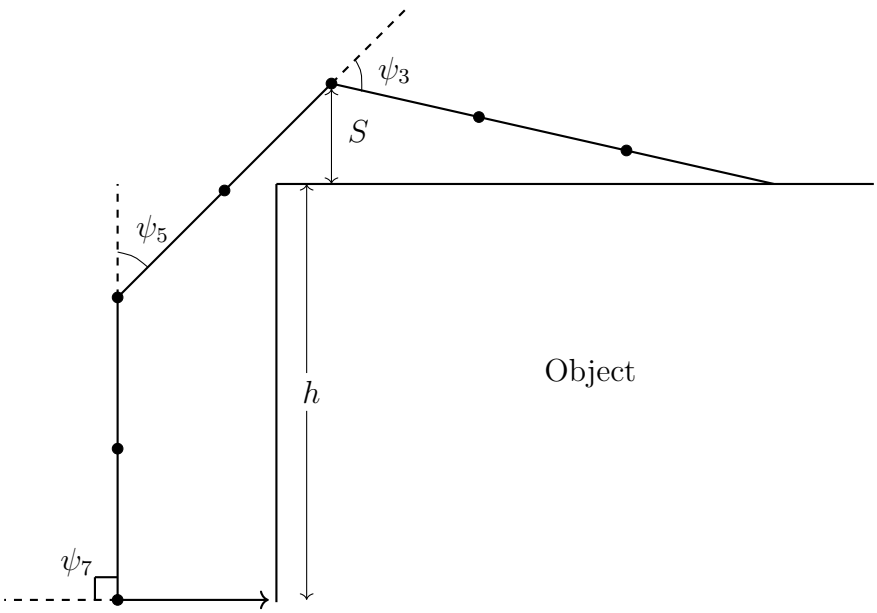


Figure 2.15: Illustration of the second position during the climb of objects of large heights.

The last position is represented in Fig. 2.16, where $\psi_6 = \arccos\left(\frac{S+h-2l}{2l}\right)$, and $\psi_4 = \arccos\left(-\frac{S}{4l}\right) - \psi_6$.

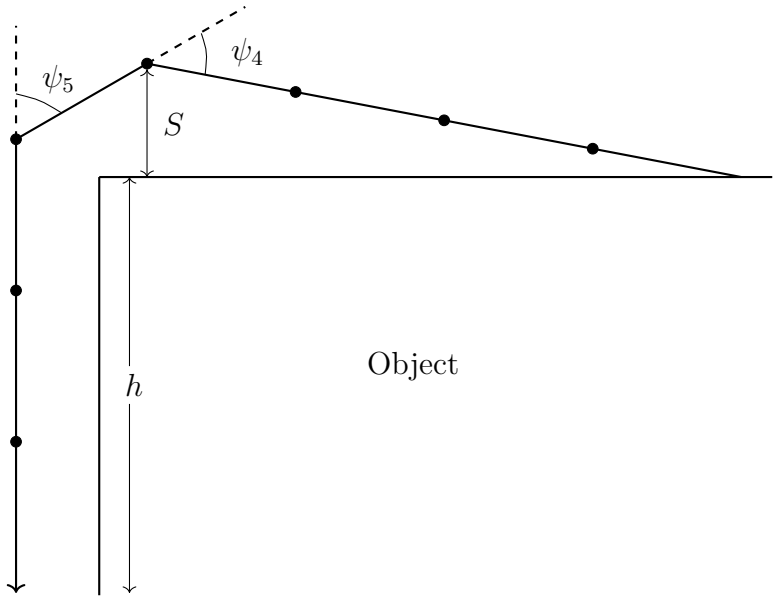


Figure 2.16: Illustration of the third position during the climb of objects of large heights.

2.1.4 Flip

As mentioned in the section above, when the robot has climbed a vertical object it is turned upside down. Therefore, a sequence of movements was made that will flip the robot back to its original orientation. This movement is based upon the back having low friction and, therefore, being able to slide on the ground when needed.

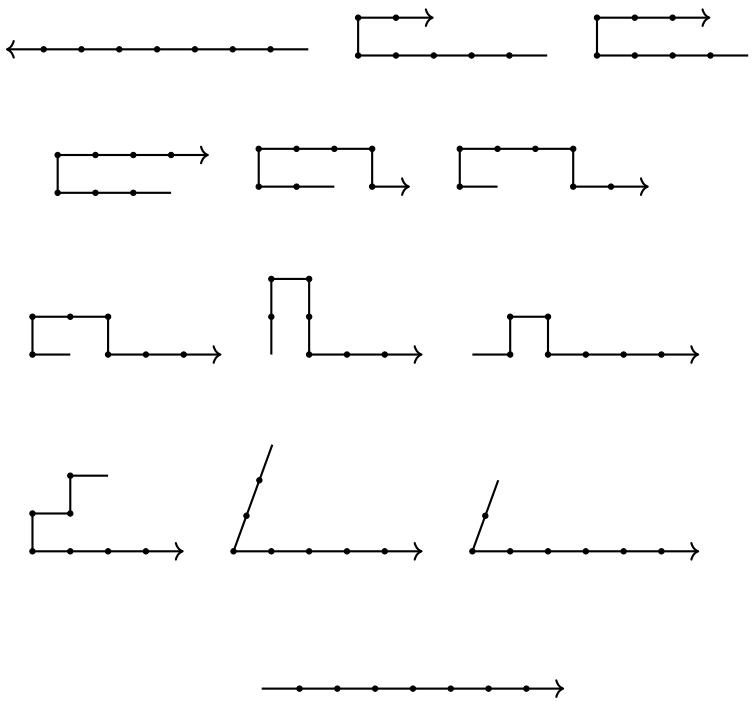


Figure 2.17: Illustration of the different steps to flip around.

2.2 Critical positions

To make the robot able to accomplish the lifting and moving that is explained in this chapter, specifications on motors, friction material, and links between the motors are needed. Therefore, important critical positions were evaluated such that the robot will be capable of accomplishing the needs from the problem description and the different movements. The first critical position examined is when a motor would have to lift four segments as this is the maximum amount of torque that a motor will have to exert. The following calculations assume that the mass of all segments is equal and, therefore, that their center of mass is located in the middle. The torque in this situation is $M = F \cdot 2l$ (see Fig. 2.18) and $F = 4 \cdot mg$ because the center of mass of the four left segments is located in the middle of them. With a segment's length of $l = 0.1$m and a mass of $m = 0.1$kg, the maximum torque is $M_m = 0.7846$Nm. Concluding that a motor which can produce a torque of $M_m > 0.7846$Nm is enough for the robot.

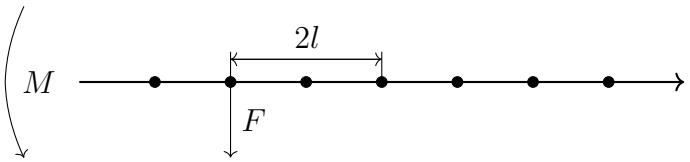


Figure 2.18: Illustration of the torque produced by the left half of the robot.

The critical position when moving up a slope is found when the center of mass is located too far back on the robot, causing it to fall backward. Therefore, the most uncertain position in the *fast movement* was evaluated, which is just after an iteration has begun (see Fig. 2.19). This is because the rear (point 1) is now in the air. If the center of mass ended up on the left side of point 2 in this situation, the robot would fall backward because point 2 is the last part touching the ground. Theoretically, without considering the torque on the joints, the biggest angle is $\rho_s = 90° - \rho_r$. Where $\rho_r = 45°$ is easily found from Fig. 2.19. This means that the absolute biggest angle the robot could climb using the *fast movement* is $\rho_s = 45°$.

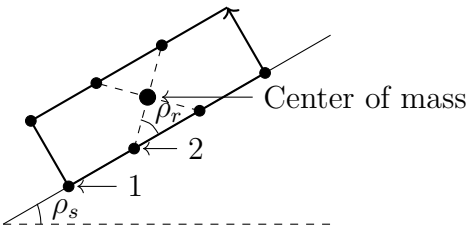


Figure 2.19: This figure illustrates the critical position when climbing a slope while using the *fast movement*.

When moving up or down a slope, the friction is also a variable that needs to be considered in order to ensure stability. In Section 1.3 it is specified that a slope of $\rho_s = 20°$ needs to be manageable. The coefficient of friction has to be $\mu > \tan(\rho_s) = [\rho_s = 20°] = 0,364$. Furthermore, with the steepest slope manageable being $\rho_s = 45°$, the coefficient of friction should be $\mu < \tan(45°) = 1$.

With the modeling being based on free rotations and simple lines instead of actual segments, the design of the robot has to follow some parameters. The motors and the design must enable a rotation between each segment of at least $\pm 110°$.

3
Design of the robot

In this project, the snake-like robot needed to have a practical design that also looks like a real snake. Since the robot moves vertically with a pulse-movement and is also able to "roll" forward, as is described in Section 2.1.2, the design had to be different from a robot that is supposed to move with a serpentine movement. To make a robot with a vertical movement, some boundaries to the physical appearance had to be made. This lead to the design of a more functional robot, rather than perfectly resembling a biological snake.

3.1 Specifications

The physical design needs to fulfill the movement requirements described in Section 1.3, and Chapter 2.

- The design must allow each segment of the robot to rotate more than $\pm 110°$.
- The robot's motors need to have a torque of at least 0.8Nm, as is calculated in Section 2.2.
- Cables running along the robot must not risk getting pinched.
- The robot needs to adapt according to the next obstacle, which implies that the robot needs to have space for a visual sensor.
- A friction coefficient of at least 0.364, as is calculated in Section 2.2, between the robot and the ground needs to be implemented.
- The energy source needs to fulfill the electronics energy demands.
- The design of the robot should be as light as possible without impairing the durability.
- All the electronics should be easily accessible.
- The robot needs to have an appearance resembling that of a real snake.

3.2 Hardware choices

This section addresses the hardware choices made in order to fulfill the demands in the previous section, namely the motors, microcontroller, and sensor.

3.2.1 Servo motors

To fulfill the climbing demands of the snake, strong and relatively quick servo motors were necessary. The selection of a digital servo motor with its own control circuit allowed for simpler programming, which meant that less work was needed to

implement the control of the servo motor. An added benefit was to be able to select the servo motors maximum and current torque, as well as monitoring its voltages and currents. For this reason, the XYZRobot A1-16 servo motor was selected. It is a servo motor built specifically for robots, with simple control and easy integration. Table 3.1 summarizes the chosen servomotors specifications taken from the servo data sheet, see Appendix A.1.

Size:	50x32x40.5 mm
Weight:	60 g
Voltage:	12 V
Resolution:	0.323°
Max speed:	70 rpm
Stall torque:	2.5 Nm
± max. angle	155°

Table 3.1: Servo motor specifications.

3.2.2 Microcontroller

While not a requirement of the project, a desire throughout the work was to make the robot completely wireless. A Wemos D1 mini module, featuring an EspressIf ESP8266 microcontroller and micro USB connector for programming was chosen for the central control of the robot. It contains WIFI capabilities, as well as being small and, therefore, easy to place inside the robot. Furthermore, it is simple to program using the Arduino IDE.

3.2.3 Sensor

To enable the robot to sense its surroundings, a distance sensor was required. To enable fast and simple reading of distances, an analog sensor is preferred. The Sharp GP2Y0A21YK0F provides a narrow reading between 10-80cm, and since this is longer than the robot, it is enough for this project (the robot's length is ∼60cm, this is further explained in Section 3.3).

3.3 Physical design of the robot

As seen in Section 3.1, different aspects need to be considered when designing the robot. The design of all parts of the project was made in the CAD program Autodesk Inventor Professional 2017. Since no CAD-file for the used motors was found, an approximate copy of its design had to be made to get an accurate impression of how the robot would look when finished, as well as to make sure that all parts would fit the motors. All parts were then printed and adjusted to match with the hardware choices made in Section 3.2.

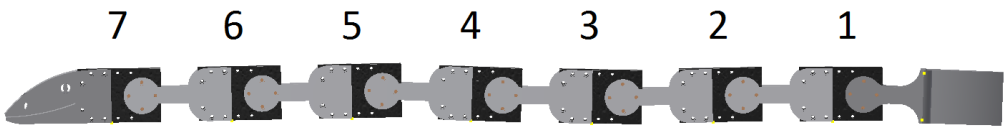


Figure 3.1: Illustrations of the robot with associated motor numbers.

The design of the snake can be divided into three different parts; the head, the midsection, and the tail. Each part has to take some of the specifications listed in Section 3.1 into account. As seen in Fig. 3.1, each part of the midsection received its own identification number, which is later used in the programming and implementation of the robot.

3.3.1 Midsection

The main part of the robot is its midsection, which consists of seven motors that are all connected to each other by links as seen in Fig. 3.2. Two slightly different variations of the links were designed to match with each side of the motor. This was necessary since a motor have small differences between either side of it. These links were designed to be easily mounted as well as being robust enough. The length and shape was chosen such that each segment of the robot would have at least $\pm 110°$ of freedom in the vertical plane. This design also allows integration of battery cells on the end of each segment, if the interest of a wireless version of the robot would arise.

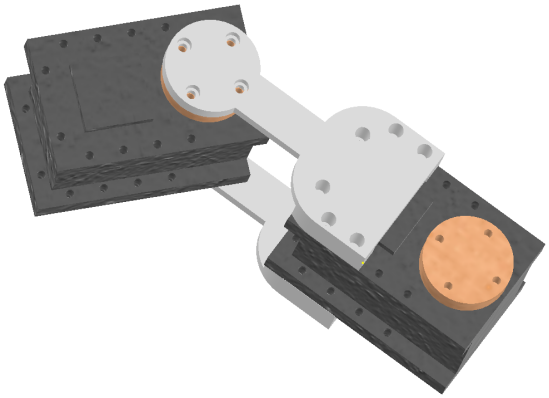


Figure 3.2: Illustration of the links connecting two motors.

3.3.2 Head

One of the main focuses while designing the head components of the snake, seen in Fig. A.5, was to make sure that the sensor would be easy to mount and dismount. This was acquired by making one slit on each side, where the sensor could be attached to the head. In addition to the practical characteristics of the head, it was also designed to have an appearance of a snake. This is the reason why the head received modifications resembling a mouth, a nose, and an eye, which does not have any practical use. Another demand for the head is that it has to be designed with an opening for the tail seen in Fig 3.4, which makes it possible to do the "rolling" movement. This implies that the space between the head components needs to be larger than the width of the tail.

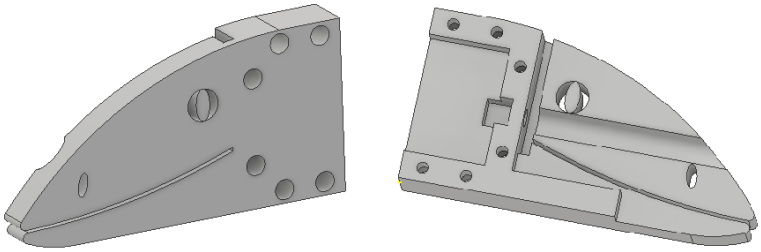


Figure 3.3: Illustration of the head.

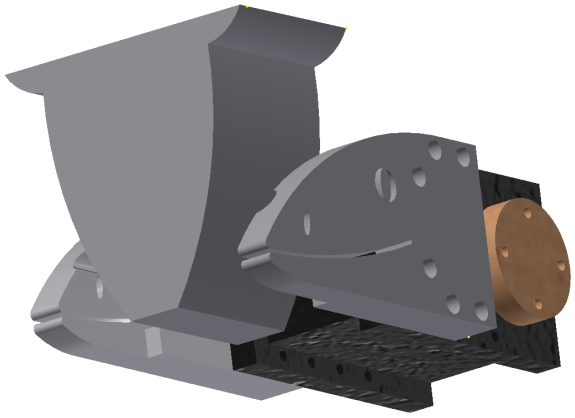


Figure 3.4: Illustration of the tail and head while assembled.

3.3.3 Tail

The parts seen in Fig. 3.5 is the robot's tail which is the only part of the robot that does not include a motor. The complete part is built with three smaller components. The main component is a small box that holds the electronics, as well as the cables for the electronics. Therefore, both a hole from the side and one towards the front is included. The connection between motor one and the tail is made in a similar way to the links to get the same properties as the rest of the segments.

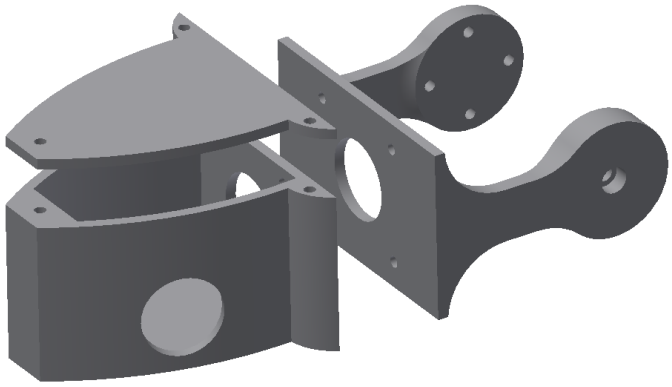


Figure 3.5: Illustration of the separated parts of the tail.

3.4 Obstacle course

To verify that the robot fulfills the set specifications in Section 1.3, an obstacle course was constructed (see Fig. 3.6). This part of the project was also made in a slight modular way, to change or swap different parts easily. Firstly, a small plateau to test the robot's climbing abilities for small obstacles. After this plateau, a ramp with an angle of 20° to verify that the robot meets the set requirements for climbing slopes. This ramp ends with a flat part where the robot can adjust for the main climbing challenge. This is a vertical obstacle that reaches 20cm in height followed by a similar obstacle of 10cm. This part of the course is to resemble stairs, where the robot has a limited distance (40cm) to adjust before climbing again. The usage of a vertical obstacle with a height of 20cm was implemented to challenge the climbing algorithm.

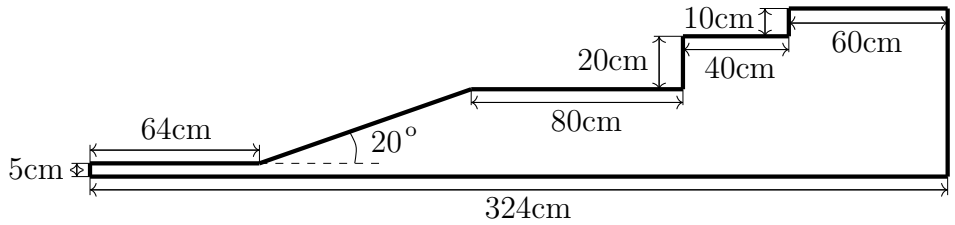


Figure 3.6: Schematic of the built obstacle course.

4

Implementation

The project's implementation was divided into three different parts. The first part, **Motor Control**, focuses on the control of the motors and implementation of the algorithms. The second part, **Distance sensor**, around the implementation of the robot's sensor by scanning and identifying obstacles. The third and last part, **Autonomous Driving**, on the integration of the distance sensor into the different movements to create an adaptive robot.

4.1 Programming language

Programming of the snake-like robot was early on decided to be done on the C-based Arduino platform with its associated IDE, as it provides a simple starting ground for coding microcontrollers as well as containing preexisting code and libraries for various tasks. Although at first, the idea was to create a PC application for controlling the robot, it was replaced with a served web page with accessory control through the Arduino IDE Serial Console.

4.2 Motor Control

The project's motor control module started with control of a single servo motor, and later the control of all servo motors individually. XYZRobots A1-16 smart servo motors can be named for individual control while having them in a parallel connection; this meant controlling the servo motors from a single control pin instead of needing to have a separate control pin for each servo motor (see Fig. 4.1).

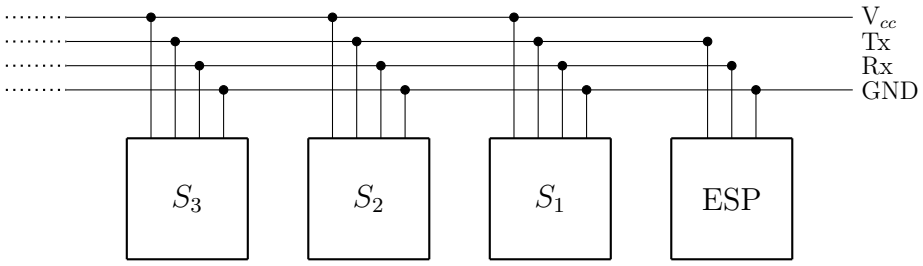


Figure 4.1: Connection diagram for digital servo motors, e.g. the ones used.

Instead of implementing a communication's protocol based upon the servo motor's data sheet, it was decided to use already existing code. This code was provided

by the manufacturer of the servo motors The important part to note about this communication protocol is that each servo motor is sent a target position, a value between 0 (+155 degrees) and 1023 (-155 degrees), and a playtime, the time in which a servo motor attains the given position.

Before the implementations of the different movements, three milestones were set; *Control of a single servo motor*, *Renaming a servo motor for serializable communication* and *Control of seven servo motors in series*. After testing each segments response, functions that could convert angles between degrees, radians and the servo motors angle were implemented (see fig. 4.2).

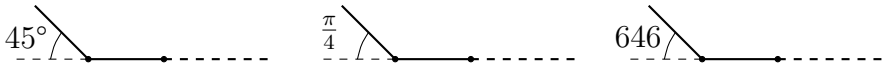


Figure 4.2: Conversion between degrees, radians, and servo motor position.

Each algorithm was implemented by converting their calculated equations into separate arrays of length seven, with each array's element corresponding to a different servo motors angle. This means that element one in a given array represents the angle that servo motor one attains. As seen in Section 5.1.1, the *main movement* was converted into an array of length eight, with the first seven elements corresponding to a servo motors angle, and the eighth elements being there for iterating purposes. The conversion between equations and arrays was done by calculating what angle the servo motor of a given joint needed to attain. For example, to achieve an angle for the second joint, it needs to be converted to the servo motors actual angle (see fig 4.3)

Figure 4.3: Conversion between a joint angle into a servo position,

4.3 Distance sensor

The implementation of the distance sensor started with making a conversion between the output (voltage) from the sensor into a distance in centimeters. The accuracy of the calculated distance is important as the robot is going to base every action on different measures of distance.

After the conversion between voltage and distance had been implemented, two different measuring algorithms were designed. One to accurately measure the height of an obstacle, and one to accurately measure the angle of an obstacle. This meant that the robot could first measure the angle, and afterward measure the height,

given that it was a vertical obstacle.

To measure the angle of an obstacle the robot makes two different distance measurements. One measure at an angle of 0° as to get the distance to the obstacle (d_s), and afterward a measurement at a different angle (in our case 10°), to get the hypotenuse (d_{hyp}). To do a reading at a given angle means to tilt the head upwards (v_t) (see Fig. 4.4). Afterward, the robot calculates the height (d_h) and length (d_t) of the inner triangle (see Eq. (4.1)-(4.2)). These are then used to calculate the slope's angle (v_s) in Eq. (4.3).

$$d_l = \cos(v_l) \cdot d_{hyp} - d_s \tag{4.1}$$

$$d_h = \sin(v_t) \cdot d_{hyp} \tag{4.2}$$

$$v_s = \arctan(\frac{d_h}{d_l}) \tag{4.3}$$

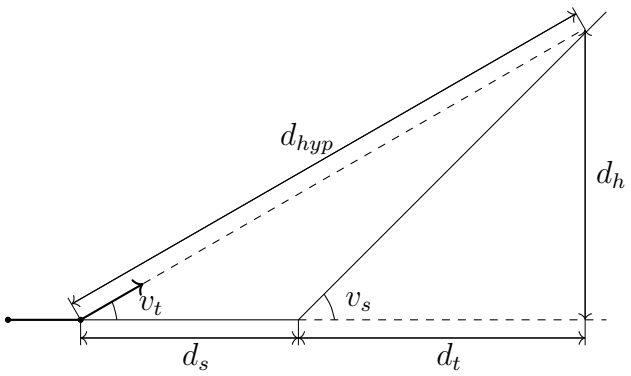


Figure 4.4: Measuring the angle of a slope.

To get the height of an obstacle, the robot starts by measuring the length to the obstacle. Afterward, the robot makes a sweep of readings upwards until it no longer sees the obstacle. This means that the distance measured before it no longer sees the obstacle is the hypotenuse (d_{hyp}), as is seen in Fig. 4.5. To make the sweep of measurements, the robot subsequently takes a measurement and increments the angle in which the head is tilted (v_t) by 1°. After obtaining the hypotenuse the height (d_h) of the obstacle can be obtained (see Eq. (4.4)).

$$d_h = d_{gyp} \cdot \sin(v_t) \tag{4.4}$$

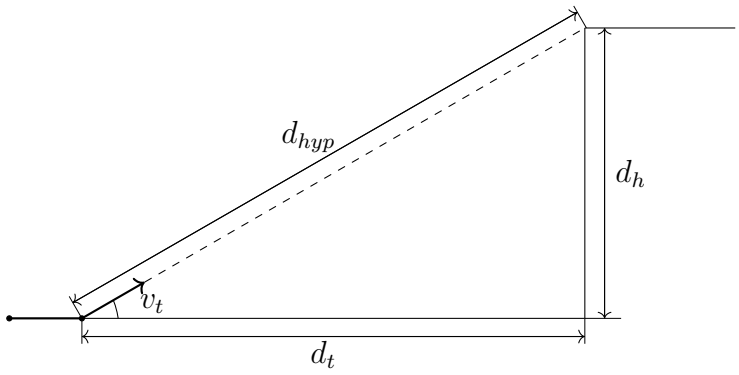


Figure 4.5: Measuring the height of a vertical obstacle.

During the implementation, an unexpected problem arose. To get the hypotenuse of the triangle, seen in Fig. 4.5, the assumption was made such that the robot only had to measure until the obstacle was no longer detected. Instead of the measurements going from a regular length to the maximum detectable, only a slight increase of the length was seen. Thus the problem occurred of having to implement a detection of a gradual change in the distance readings, signifying the edge of the obstacle.

4.3.1 Autonomous driving

Implementing the autonomous driving meant integrating the distance sensor together with the movement algorithms. This meant that the robot makes a sequence of choices based upon different measurements. The sequence of choices can be seen in a simplified manner in Fig. 4.6. Figure 4.7 shows a flowchart depicting the full sequence of choices.

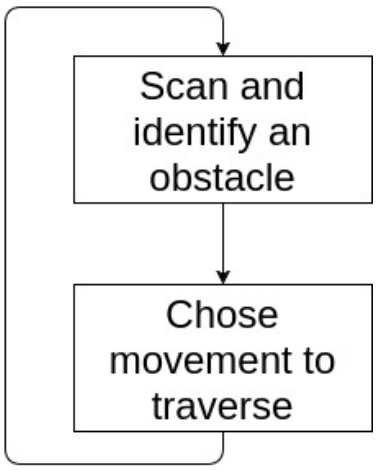


Figure 4.6: Simplified diagram describing the robot's thought process.

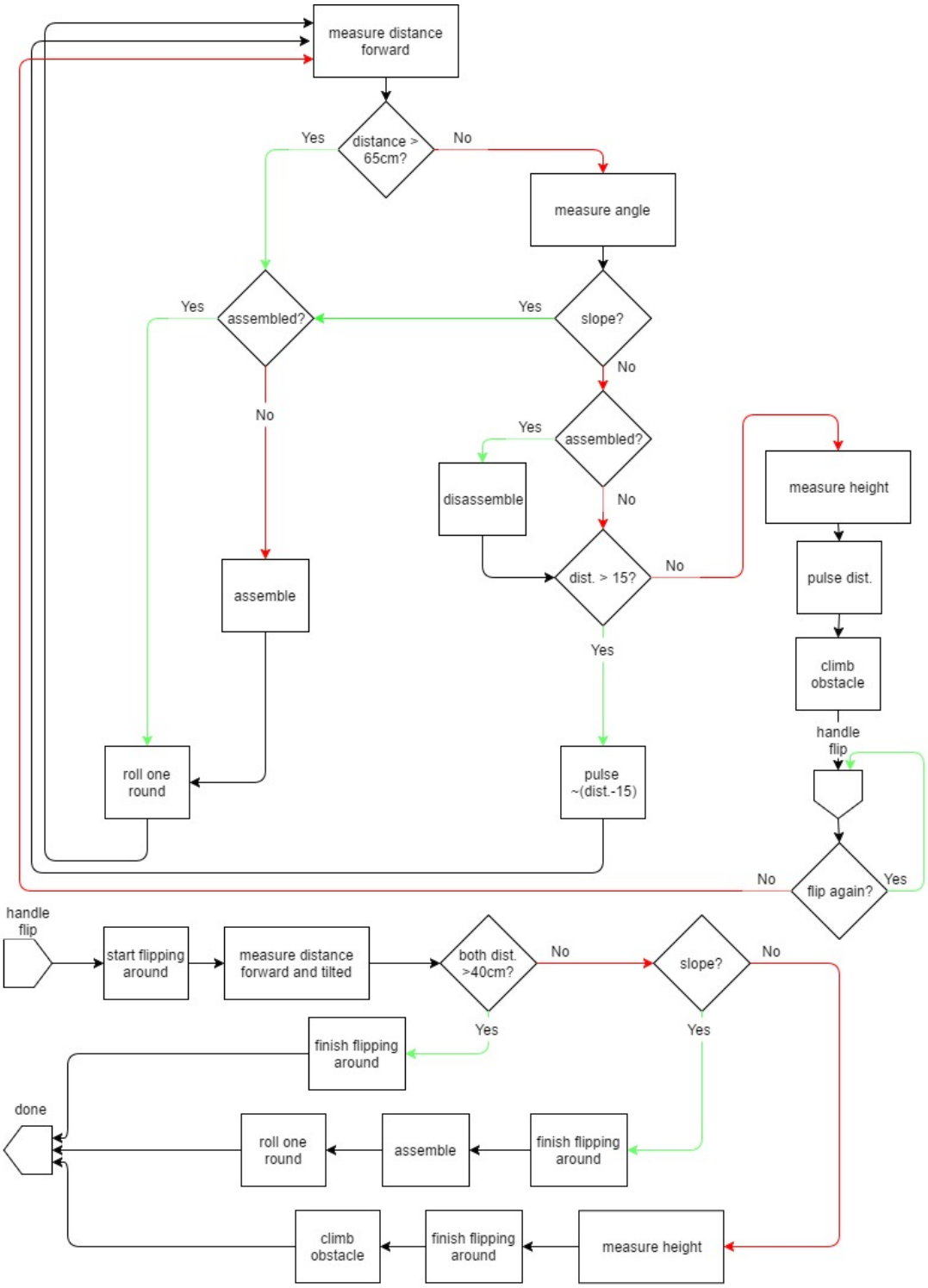


Figure 4.7: In-depth diagram describing the robot's thought process.

As seen in the diagrams mentioned above, the robot has to take measurements while it is assembled and while it flips around. To make a measurement while assembled, the robot tilts the back-end, such that it does not obstruct the head (see fig. 4.8) and then makes the measurement. While flipping, the robot measures in a similar fashion. It first stops at a given position (see fig. 4.9), makes a distance measurement

in front and at a tilted angle. From these measurements, it can decide whether it is an obstacle close-by or if it is free to flip around. If there is an obstacle in its way, the robot identifies it. Figure 4.10 shows the end result while measuring the height to get the length to the top of the obstacle (d_t) which it afterward uses to calculate the height (see Eq. (4.5)). If the robot has identified another close-by obstruction while flipping, it chooses to skip the last four steps in the flip motion, as those steps are unnecessary.

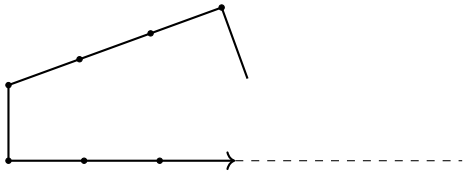


Figure 4.8: Depicting how the tail is lifted to give the sensor an unobstructed view.

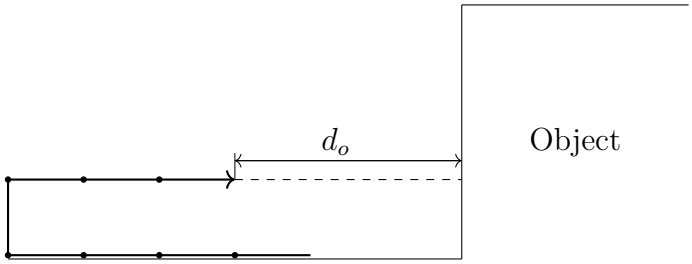


Figure 4.9: Stopping to make a measurement while unfolding.

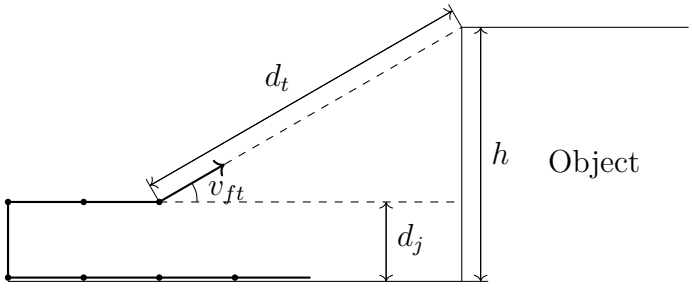


Figure 4.10: Final position after making the sweep.

$$h = d_j + d_t \cdot \sin(v_{ft}) \tag{4.5}$$

As seen in Appendix A.2, the accuracy of the sensor has a significant decline after about 40cm. To solve this problem, the robot only measures heights within 40cm of the obstacles. For distances measured between 40 and 65cm the robot only needs to identify whether the object is a slope or a vertical obstacle, choosing movement accordingly. To identify whether an obstacle is a slope or not, the robot looks for a big enough angle. This means that if the angle is above 60° it is identified as a vertical obstacle.

5
Results

5.1 Modeling

Following is the result of the movement algorithms implemented on the robot. This includes figures and tables containing the steps of the movements as implemented on the robot.

5.1.1 Main movement

Figures 5.1-5.4 and Table 5.1 contains the resulting matrix and movement by converting Equations (2.1)-(2.11) from Section 2.1.1.

$$k = \frac{\textit{distance to move}}{\textit{length of a segment}},\ \alpha = \arccos\frac{12 - 6k + k^2}{12 - 4k},\ \beta = \arccos\frac{6k - 4 - k^2}{4}$$

$$\gamma = \arccos\frac{6 - 6k + k^2}{6 - 2k},\ \delta = \arccos\frac{4k - 2 - k^2}{2},\ \sigma = \frac{\pi - \delta}{2}$$

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7	Iteration
It.1 Step 1	0	$\beta - \pi$	γ	0	0	0	0	α
It.1 Step 2	σ	$\delta - \pi$	σ	0	0	0	0	0
It.1 Step 3	γ	$\beta - \pi$	0	α	0	0	0	0

Table 5.1: The resulting matrix for the *main movement*.

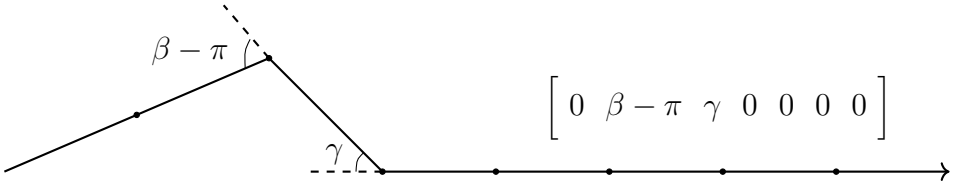


Figure 5.1: Iteration 1 Step 1 of the *main movement*.

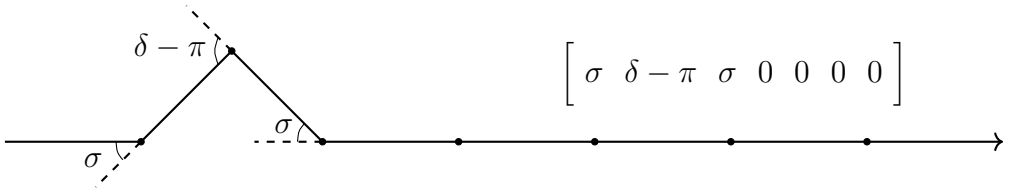


Figure 5.2: Iteration 1 Step 2 of the *main movement*.

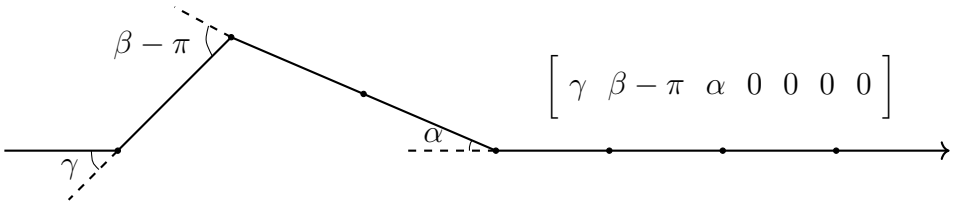


Figure 5.3: Iteration 1 Step 3 of the *main movement*.

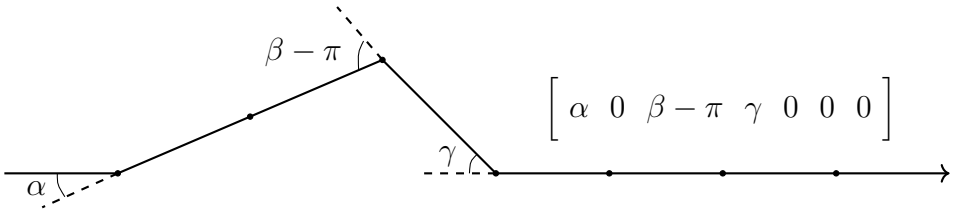


Figure 5.4: Iteration 2 Step 1 of the *main movement*.

5.1.2 Fast movement

Table 5.2 and Figure 5.5 contain the assembling sequence and movement made by converting the steps in Figure 2.5 into corresponding angles for the servo motors.

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	0	0	0	0	0	0	0
Step 2	0	90	90	0	0	0	0
Step 3	0	0	90	90	0	0	0
Step 4	90	0	0	90	90	0	0

Table 5.2: The resulting matrix for assemble.

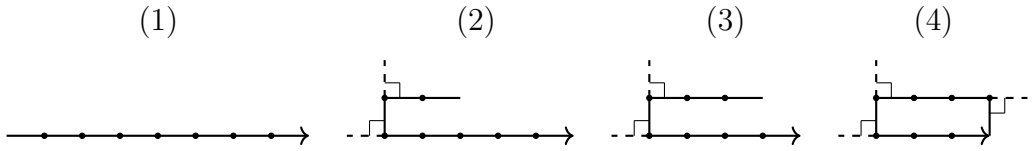


Figure 5.5: The assembly- and disassembly steps.

Table 5.3 contains the disassembling, which is the assembling sequence backwards (see Fig. 5.5).

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	90	0	0	90	90	0	0
Step 2	0	0	90	90	0	0	0
Step 3	0	90	90	0	0	0	0
Step 4	0	0	0	0	0	0	0

Table 5.3: The resulting matrix for disassemble.

The forward locomotion is the result of converting the steps of Fig. 2.6 into an array of angles (see Eq. (5.1)). These angles represent the start position. For continued movement, the array iterates through the snake by shifting each array element to the right (see Fig. 5.6 and Table 5.4).

$$[90, 0, 0, 90, 90, 0, 0, 90] \tag{5.1}$$

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7	Iteration
It.1	90	0	0	90	90	0	0	90
It.2	90	90	0	0	90	90	0	0
It.3	0	90	90	0	0	90	90	0
It.4	0	0	90	90	0	0	90	90
It.5	90	0	0	90	90	0	0	90
etc.	.	.	.	.	.	.	.	.

Table 5.4: The resulting matrix for the *fast movement*.

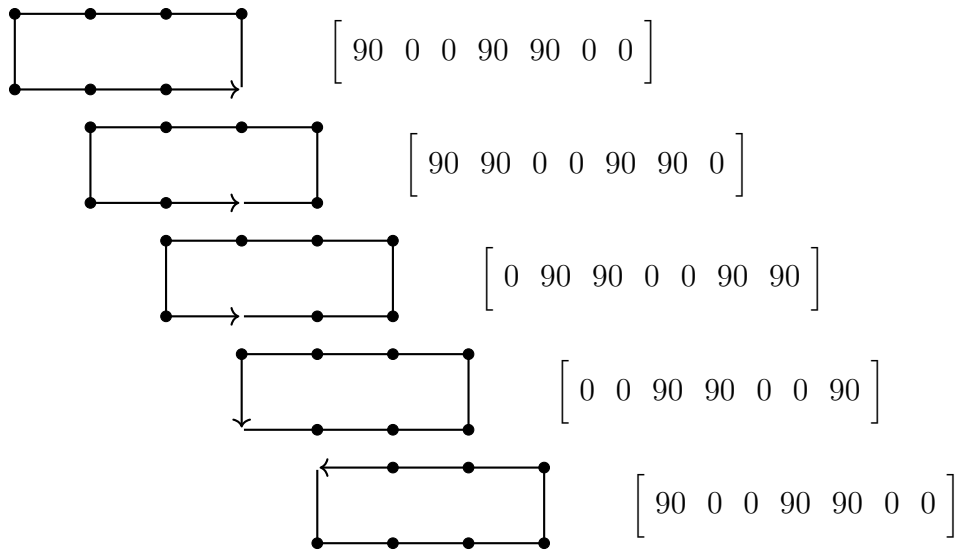


Figure 5.6: Illustration of the actual movement for the *fast movement*.

5.1.3 Climbing

Table 5.5 and Figure 5.7 shows the movement before the robot starts the climbing algorithm. This movement's purpose is to safely enter each variation of the climbing algorithm.

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	0	0	0	0	0	0	0
Step 2	0	0	100	0	0	0	0
Step 3	0	0	0	110	0	0	0
Step 4	0	0	45	0	110	0	0

Table 5.5: The resulting matrix for the rising movement.

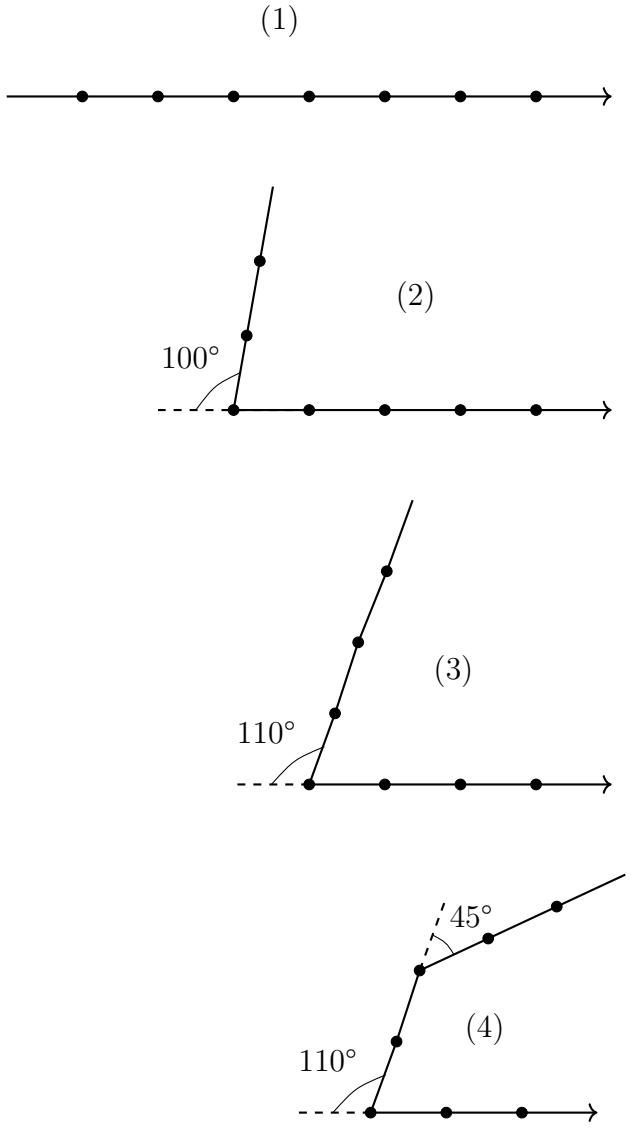


Figure 5.7: Illustration of the rising movement before climbing.

Figure 2.8-2.10 and Table 5.6 shows the steps and movements of the first climbing algorithm. This algorithm is used to climb obstacles up to 10.95cm (length of 1.5 segments).

$$h = height,\ l = lenght\ of\ a\ segment,\ s = extra\ space$$

$$\theta_1 = \arccos\frac{s+h-l}{l},\ \theta_2 = \arccos\frac{-s}{4l}$$

$$\theta_3 = \arccos\frac{-s}{5l},\ \theta_4 = \arccos\frac{-s}{6l}$$

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	0	0	0	$\theta_2 - \theta_1$	θ_1	90	0
Step 2	0	0	0	0	$\theta_3 - \theta_1$	θ_1	90
Step 3	0	0	0	0	0	θ_4-θ_1	θ_1
Step 4	0	0	0	0	0	0	0

Table 5.6: The resulting matrix for the first climbing algorithm.

Figure 2.11-2.13 and Table 5.7 shows the steps and movements of the second variation of the climbing algorithm. This variation is used to climb obstacles between 10.95cm and 18.25cm (length of 2.5 segments).

$$\theta_1 = \arccos\frac{s+h-l}{2l},\ \theta_2 = \arccos\frac{-s}{3l}$$

$$\theta_3 = \arccos\frac{-s}{4l},\ \theta_4 = \arccos\frac{s+h-2l}{l}$$

$$\theta_5 \arccos\frac{\frac{-s}{5}}{l}$$

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	0	0	$\theta_2 - \theta_1$	0	θ_1	90	0
Step 2	0	0	0	$\theta_3 - \theta_4$	θ_4	0	90
Step 3	0	0	0	0	$\theta_5 - \theta_4$	θ_4	0
Step 4	0	0	0	0	0	0	0

Table 5.7: The resulting matrix for the climbing.

Figure 2.14-2.16 and table 5.8 shows the steps and movements of the third variation of the climbing algorithm. This variation is used to climb obstacles higher than 18.25cm.

$$\theta_1 = \arccos\frac{s+h-l}{3l},\ \theta_2 = \arccos\frac{-s}{2l}$$

$$\theta_3 = \arccos\frac{-s}{3l},\ \theta_4 = \arccos\frac{s+h-2l}{2l}$$

$$\theta_5 \arccos \frac{\frac{-s}{4}}{l},\ \theta_6 = \arccos \frac{s+h-3l}{l}$$

	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5	Servo 6	Servo 7
Step 1	0	0	$\theta_2 - \theta_1$	0	θ_1	90	0
Step 2	0	0	0	$\theta_3 - \theta_4$	θ_4	0	90
Step 3	0	0	0	$\theta_5 - \theta_6$	θ_6	0	0
Step 4	0	0	0	0	0	0	0

Table 5.8: The resulting matrix for the climbing.

Figure 2.17 contains the movement after the climbing algorithm. This is done to flip the robot to resume its original orientation. Before finishing the flip, the robot scans for nearby obstacles (see figure 4.9).

5.2 Sensor

5.2.1 Distance measurements

Figure 5.8 contains the sampled values from the sensor at different distances. This was used to create the Eq. (5.2) using Matlab, where v=voltage and d=distance.

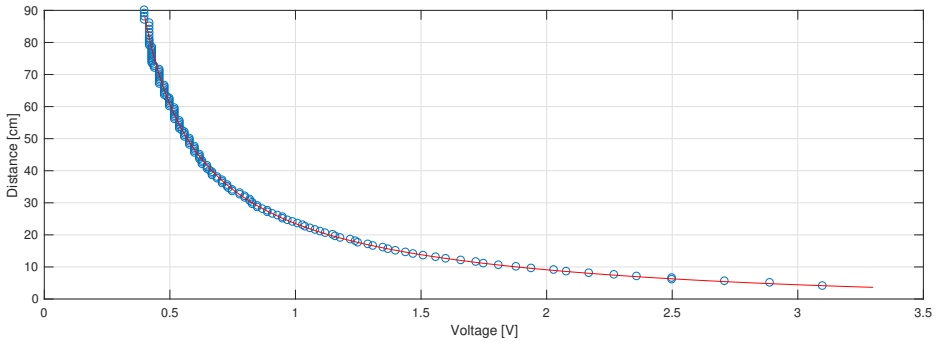


Figure 5.8: Graph that shows both the sampled voltages as well as the extracted function.

$$d = \frac{0.02652 * v^5 - 0.2823 * v^4 + 1.52 * v^3 - 5.158 * v^2 + 5.615 * v + 17.46}{v - 0.1844} \tag{5.2}$$

Figure 5.9 contains the result of the robots measured (see Fig. 5.10) distance in comparison to the actual distance. This was tested for distances between 5-80cm in 5 cm increments.

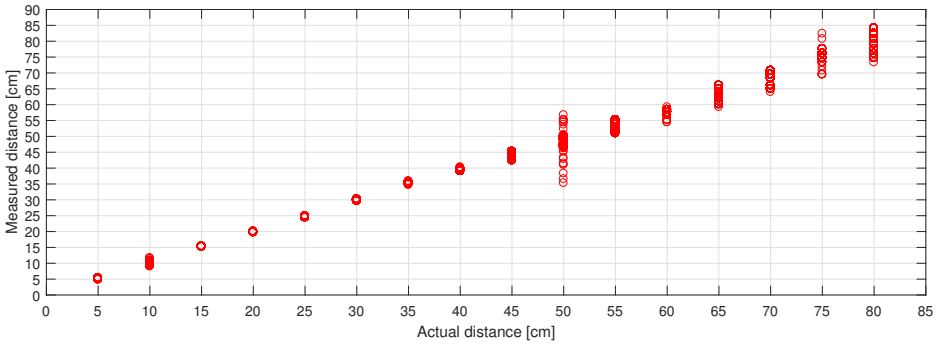


Figure 5.9: The spread of result for measuring different lengths between 10-80cm.

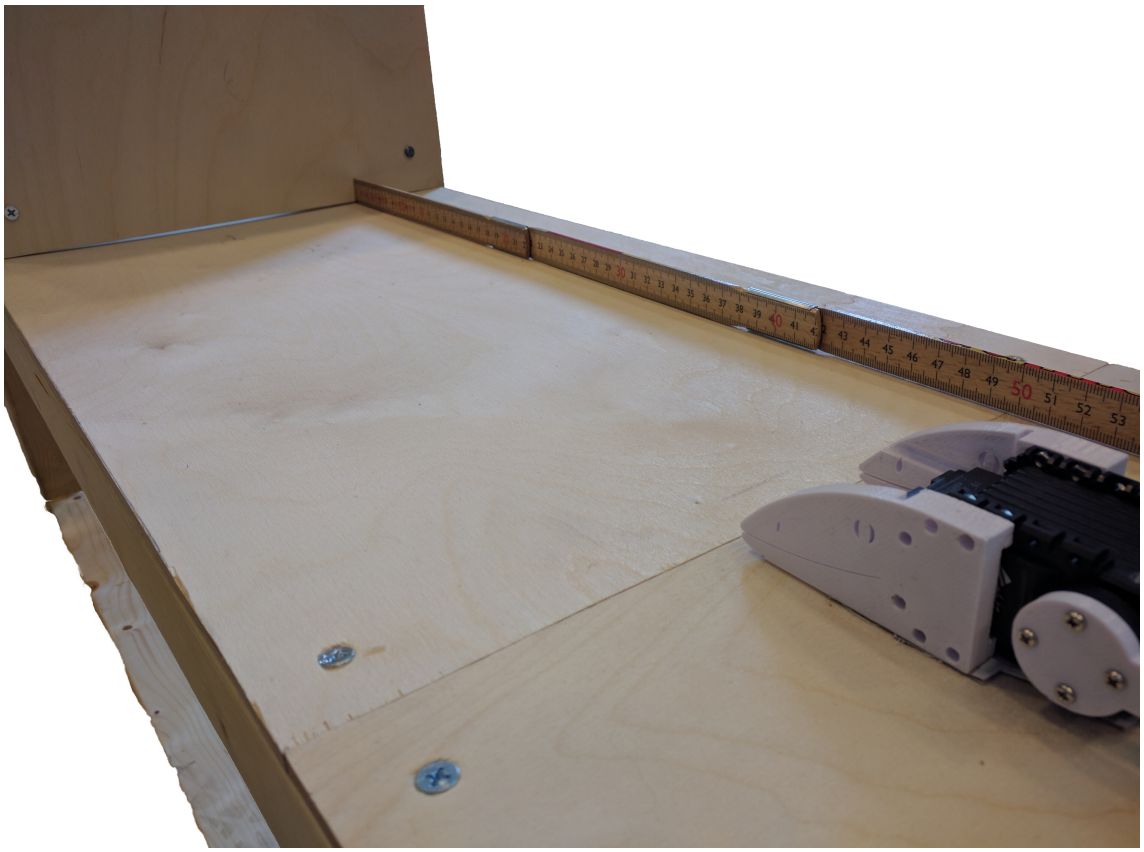


Figure 5.10: The picture shows how the distance measurements were made.

5.2.2 Height measurements

Figure 5.12-5.16 contains the result of different height measurements (see fig.5.11). These measurements were made at different distances away from the obstacle so as to give an idea of the spread. Table 5.9 contains the average value as well as the standard deviation of the different measurements.

Figure 5.11: The picture shows how the height measurements were made.

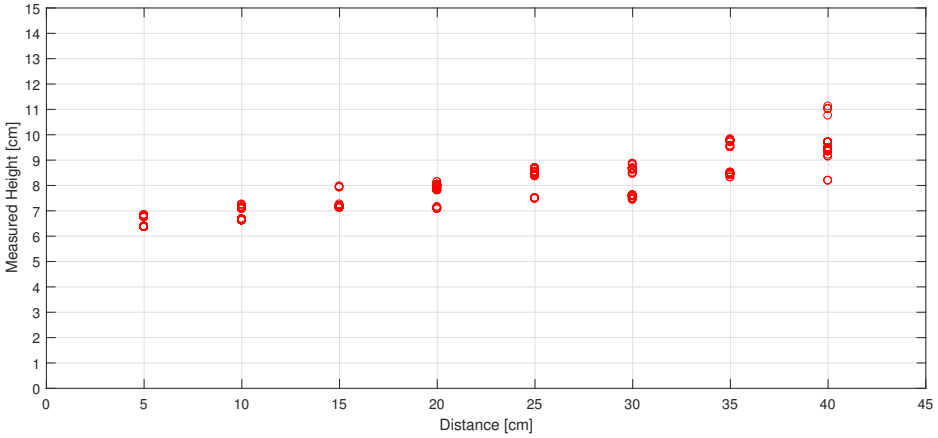


Figure 5.12: Measuring the height of 5 cm at different distances.

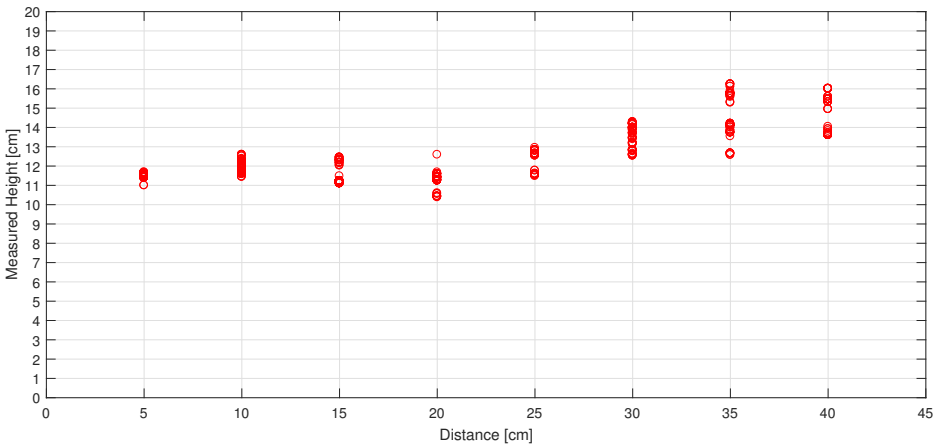


Figure 5.13: Measuring the height of 10 cm at different distances.

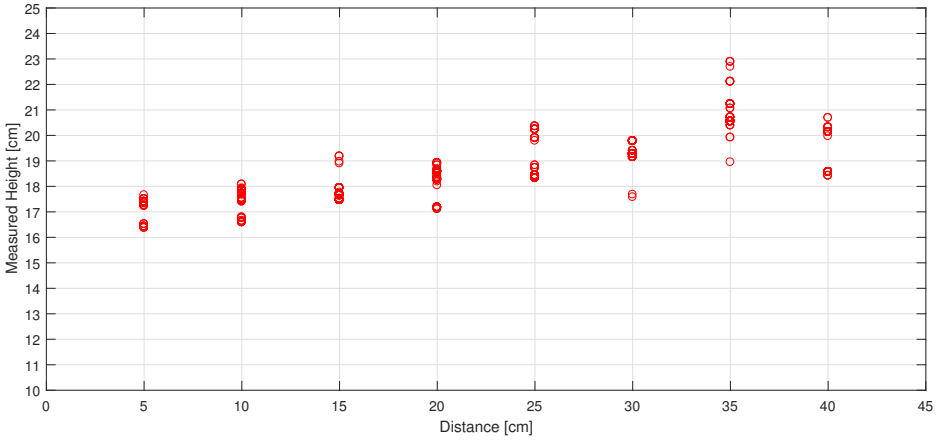


Figure 5.14: Measuring the height of 15 cm at different distances.

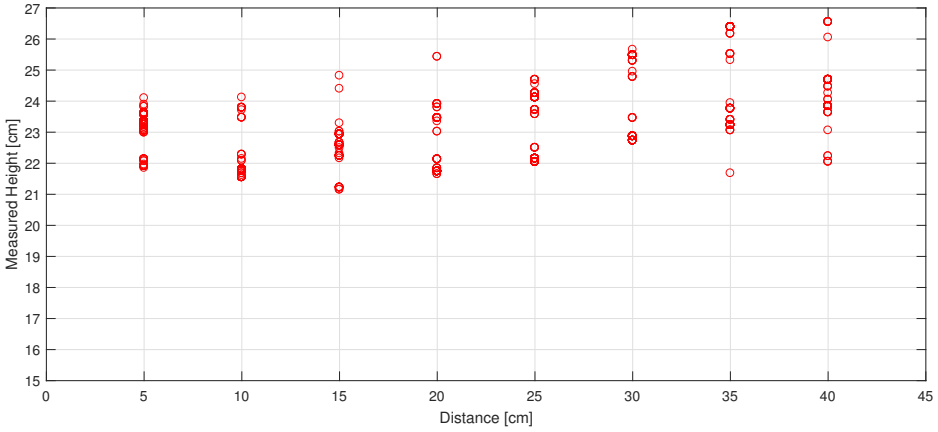


Figure 5.15: Measuring the height of 20 cm at different distances.

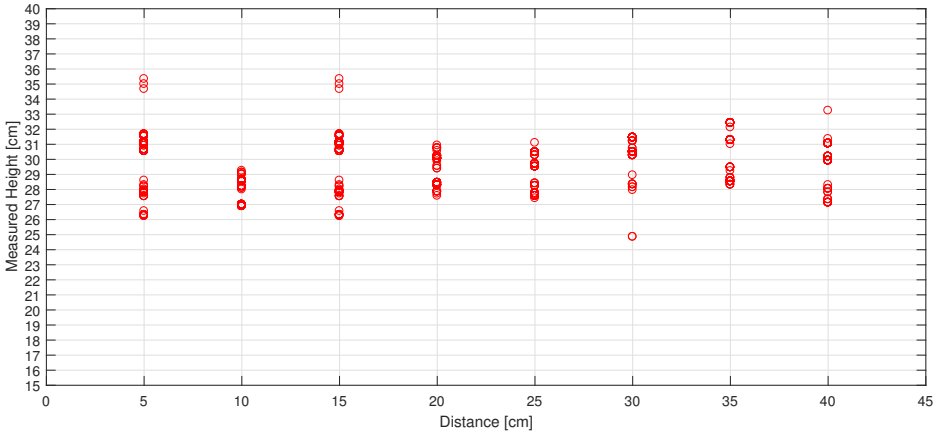


Figure 5.16: Measuring the height of 25 cm at different distances.

Distance [cm]	Height [cm]	Average value [cm]	Standard Deviation [cm]
5	5	6.58	0.20
10	5	6.74	0.19
15	5	7.19	0.15
20	5	7.73	0.37
25	5	8.40	0.33
30	5	7.97	0.52
35	5	8.93	0.62
40	5	9.5	0.42
5	10	11.49	0.10
10	10	11.95	0.28
15	10	11.44	0.48
20	10	11.29	0.37
25	10	12.12	0.55
30	10	13.51	0.60
35	10	14.71	1.09
40	10	15.03	0.83
5	15	16.73	0.47
10	15	17.45	0.38
15	15	17.71	0.35
20	15	18.23	0.58
25	15	18.77	0.73
30	15	19.36	0.34
35	15	20.80	0.61
40	15	19.57	0.78
5	20	23.02	0.61
10	20	22.04	0.69
15	20	22.53	0.59
20	20	22.28	0.89
25	20	23.08	1.02
30	20	24.00	1.27
35	20	24.82	1.49
40	20	24.24	0.92
5	25	29.91	2.01
10	25	27.75	0.82
15	25	28.19	0.42
20	25	29.27	0.98
25	25	29.10	1.15
30	25	30.53	1.96
35	25	29.75	1.56
40	25	29.98	1.40

Table 5.9: Average value and standard deviation for the different height measurements at the different distances away.

5.2.3 Angle measurements

Table 5.10 contains the result while measuring the type of obstruction, where Wall is a vertical obstacle, and Slope means an incline of 10-20°.

Distance [cm]	Height [cm]	Type	Result
5	30	Slope	Slope
10	30	Slope	Slope
20	30	Slope	Slope
30	30	Slope	Slope
50	30	Slope	Slope
55+	30	Slope	Uncertain
5	5	Wall	Wall
10	5	Wall	Wall
20	5	Wall	Wall
25-30	5	Wall	Uncertain
30	5	Wall	Slope
5	10	Wall	Wall
10	10	Wall	Wall
20	10	Wall	Wall
30	10	Wall	Wall
50	10	Wall	Wall
70	10	Wall	Wall
5	20	Wall	Wall
10	20	Wall	Wall
20	20	Wall	Wall
30	20	Wall	Wall
50	20	Wall	Wall
70	20	Wall	Wall

Table 5.10: Results while identifying obstacles at different distances.

5.3 Designing

5.3.1 Robot

Size:	605x35.5x50 mm
Weight:	658 g
Tail weight:	79 g
Friction coefficient μ	0.73

Table 5.11: Specifications for the physical robot and friction coefficient between the robot and the obstacle course.

Figure 5.17-5.18 shows the final appearance of the snake-like robot; one picture was taken from the left side (5.17), and one from the right (5.18).

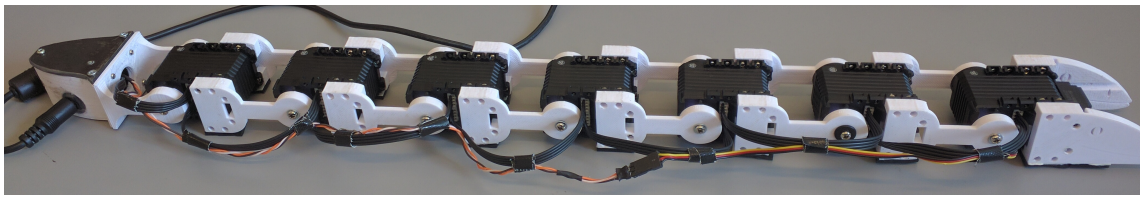


Figure 5.17: The final appearance of the robot from its left side.

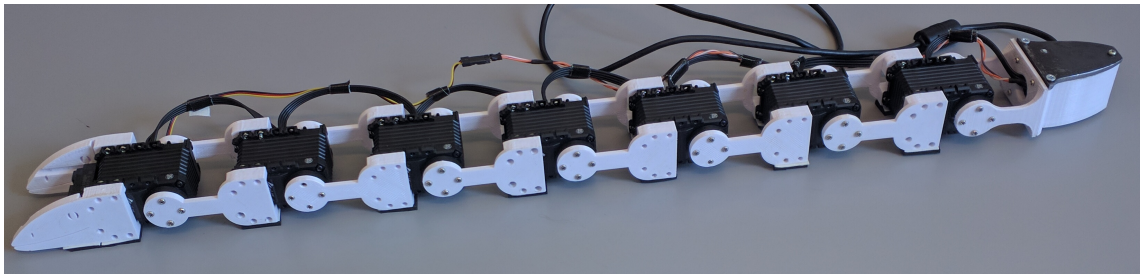


Figure 5.18: The final appearance of the robot from its right side.

Figure 5.19-5.20 shows the final appearance of the head; one picture was taken from above, showing how the distance sensor is placed inside (5.19), as well as one close-up image of the head segment. (5.20).

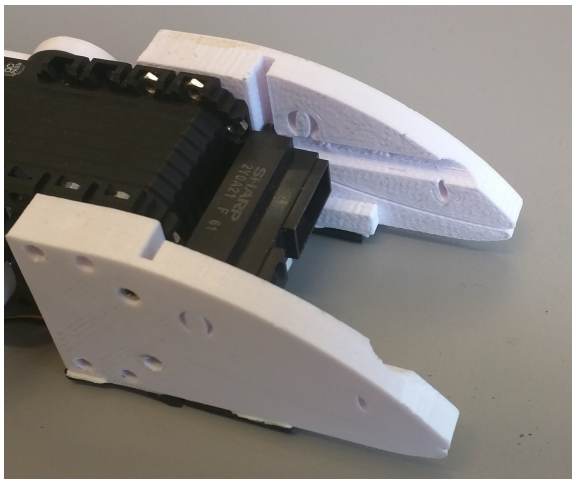


Figure 5.19: A picture taken from above that show the entirety of the head segment.

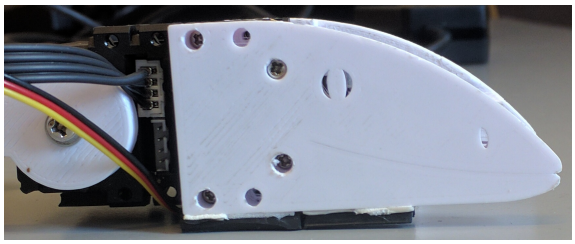


Figure 5.20: A close-up of the side of the head segment.

Figure 5.21 shows the final appearance of a segment. This picture was taken when the robot was turned upside down to better show the underneath of the segment.

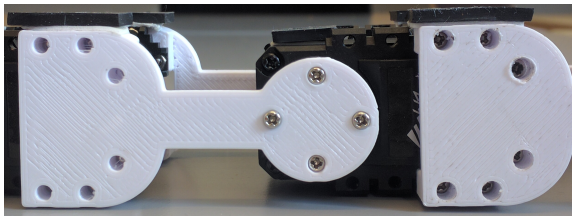


Figure 5.21: Image that shows a segment turned upside down.

Figure 5.22-5.23 shows the final appearance of the tail, taken from above. The first shows the tail closed (Fig. 5.22), and the second shows it open (Fig. 5.23). In figure 5.23 one can see the different components that make up the inside electronics of the tail; 1 is the voltage regulator, 2 is the barrel connector for power, 3 is the gyroscope (not implemented), and 4 is the Wemos D1 Mini.

Figure 5.22: A picture taken from above that show the whole tail.

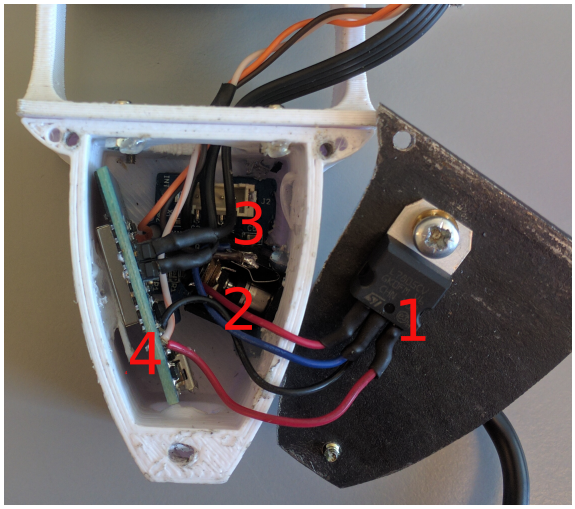


Figure 5.23: A picture taken from above that shows the inside of the tail.

5.3.2 Obstacle course

Figure 5.24 shows the outcome of the constructed obstacle course, based on the design in Fig. 3.6.

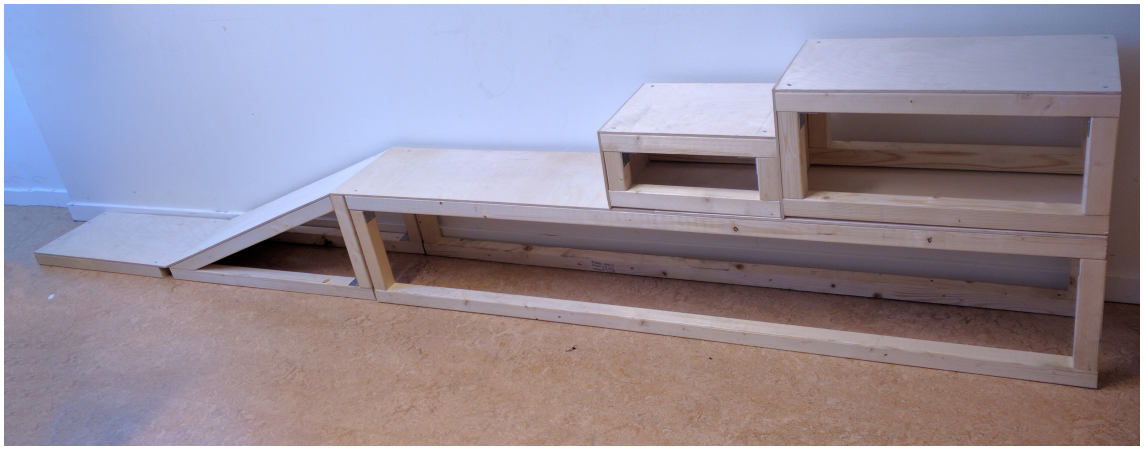


Figure 5.24: Picture of the completed obstacle course.

5.4 Outcome

This section contains; the comparison between theoretical analysis and the physical result, the speed and time in which the robot executes different movements, and the result of traversing the obstacle course.

5.4.1 Comparison of theoretical analysis and physical result

Table 5.12 contains the comparison between the theoretical maximum and the measured maximum (approx.) of different movements.

Movement	Theoretical max.	Actual max.
Climbing a vertical obstacle	Approx. 29.2cm (4 segments)	24cm
Rolling up a slope	45°	27°

Table 5.12: Difference between maximum values and actual values.

Table 5.13 contains the comparison between the theoretical distance a movement should achieve and the actual distance the movement achieved, measured as shown in Fig. 5.25 and Fig. 5.26, as well as how much the robot deviates from the path after ten pulses or one whole rotation. When measuring the deviation, deviation to the left of the robots path is considered to be positive and to the right negative.

Movement	Theoretical [cm]	Actual (approx.) [cm]	Offset sideways [cm]
10x Pulse k=0.5	36.5 (0.5 seg.)	32	0
10x Pulse k=0.5	36.5 (0.5 seg.)	32	0
10x Pulse k=0.5	36.5 (0.5 seg.)	32	0
10x Pulse k=0.5	36.5 (0.5 seg.)	32	0
10x Pulse k=0.5	36.5 (0.5 seg.)	32	0
10x Pulse k=1	73 (1 seg.)	51.5	4.5
10x Pulse k=1	73 (1 seg.)	52.5	5
10x Pulse k=1	73 (1 seg.)	52.5	5
10x Pulse k=1	73 (1 seg.)	53	5
10x Pulse k=1	73 (1 seg.)	53.5	5
Roll 8 seg.	58.4	60.0	0.2
Roll 8 seg.	58.4	60.5	1
Roll 8 seg.	58.4	60.5	-0.5
Roll 8 seg.	58.4	60.5	-0.5
Roll 8 seg.	58.4	60.5	0

Table 5.13: Comparison between the actual and theoretical distance.

Figure 5.25: Picture of the setup used to measure distance before the test.

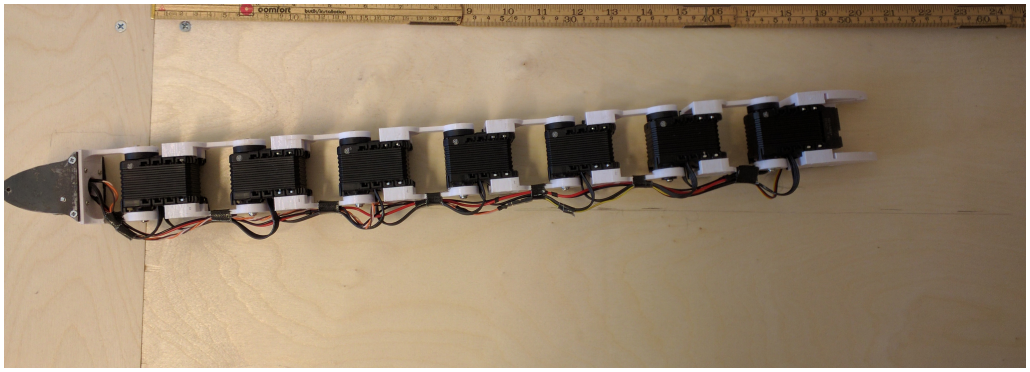


Figure 5.26: Picture of the setup used to measure distance after the test. Notice the offset from the straight line.

5.4.2 Speed

Table 5.14 shows an approximation of the time it takes to complete three different cases of forwarding locomotion, as well as their speed. The first and second rows show two different cases of the pulse; one where the robot tries to move a half segment forward with each pulse, and one where it tries to move a whole segment forward. The third row shows how fast the robot can roll a single round. In addition to being three different cases of forwarding locomotion, the time for the movements is measured when the robot runs at normal speed. Normal speed refers to the speed used when running the obstacle course with stability.

Movement	Time [m:s]	Speed [cm/s]
10x Pulse k=0.5	0:43	0.74
10x Pulse k=1	0:43	1.22
Roll one round	0:16	3.78

Table 5.14: Approximate results while running with normal speed.

Table 5.15 contains the approximate total time it takes to climb three different heights. These height are the ones contained in the obstacle course.

Height [cm]	Time [s]
5	14
10	14
20	14

Table 5.15: Approximate results while running with normal speed.

Table 5.16 shows the result of running the four different auxiliary movements at normal speed.

Movement	Time [s]
Assemble	6
Disassemble	6
Rise	6.3
Flip	24

Table 5.16: Approximate results while running at normal speed.

5.5 Obstacle course

Table 5.17 contains the result while running the obstacle course. This includes the time it took, how many minor adjustments were made (i.e. straightening it out when it deviates too much from the path), as well as how many major corrections that had to be made and why. It also contains additional comments about each run. Important to note is that the robot start 15cm away from the first plateau.

Time (m:s)	**Minor**	**Major**	**What?**	**Why?**	**Comment**
4:06	2	1	Fell while traversing the slope.	Instability in segments design, mostly the tail.	Made one false distance measurement between the slope and stairs, which caused a delay.
4:15	3	1	Fell while traversing the slope.	Instability in the segments design, mostly the tail.	Made two erroneous distance measurements between the slope and stairs, which caused a delay.
4:20	4	1	Identified the slope as a vertical obstacle.	Ended up too close to the slope after climbing.	Overall a good run.
3:39	1	2	Fell two times on the slope.	Instability in the design, mostly the tail.	Made quick corrections, overall a good run.
3:56	0	1	Fell at the end of the slope.	Instability in the design.	Overall good run.
3:47	0	0	-	-	Perfect run.
4:13	0	2	Fell in slope. Fell while flipping.	Instability in design. Dragged power cable.	Otherwise perfect run.
3:38	1	0	-	-	Perfect run.
3:57	0	3	Fell thrice at the slope.	Ended up to close to the slope. Tried to climb. Failed because of slope. Got corrected. Tried to assemble. Failed because of slope. Fell at the end of the slope.	Otherwise a good run.
3:42	3	0	-	-	Perfect run.

Table 5.17: Results while running the obstacle course.

6
Discussion

6.1 Implementation of algorithms

After implementing the different movements, the results (see Section 5.1) have, for the most part, followed our expectations with the only exception being the pulse (see Table 5.13). When we modeled the pulse we had a continuous update of angles for a smoother movement, but when implementing it on the robot we cut it down into three different steps (see Section 5.1.1). This is because when controlling the motors we could give them a playtime, in which the motor tries to attain a target position. This means that instead of needing continuous updates, we could send the servos a single target position to achieve a necessary step, this turned out to be much easier to implement. Although sending target positions to the motors is sub par for the pulse movement, every other movement benefits from this. This is because when we made the geometrical analysis, we based it on following a pattern of steps which is easily replicated by sending positions to the motors.

When trying to optimize the speed of each movement, we, in the end, decided that the use of a slower speed was better for the robot's performance. This is partly because we lacked time to find a more efficient speed but also because the final design of the robot left it more unstable than planned (further explained later). This was not a problem when traversing flat ground, but it became an issue when, for example, the robot was moving up a slope or climbing an obstacle. An alternative to this would be to use a faster speed when moving on flat ground and then to use a slower speed elsewhere, but we decided to keep the speed so we could focus on more stressing matters.

The implementation of the *main movement* could have been made marginally faster if it had been made in such a way that more than one pulse at a time could be sent, instead of only having a few joints executing the movement at a time. The streamlining of other implementations would also affect the overall performance, i.e. to skip the last step of assembling. Another example would be to rearrange the steps of the *fast movement* such that lifting the back-end before measuring would no longer be necessary.

The different variations of the algorithms perform admirably. However, they are defined for exactly eight segments and, therefore, not as modular as they could have been. Although the movements aren't very modular, it is still relatively easy to

change them while implementing, as each matrix and array can be expanded.

6.2 Environment Sensing

We have been able to measure distances between 5-40cm accurately, and when it comes to distances between 40-80cm the result is sufficient. When measuring heights we get a result above the actual height, but this is not a problem when it comes to distances between 5-20cm. Because when the robot executes a climbing algorithm, it uses the tail as support which removes the problem of height difference. The inaccuracy becomes a problem at heights around 25cm because if the robot tries to climb too high, i.e. approx. 28.5cm, it generates too much backward momentum and falls. Although the robot falls when it tries to climb a height of 28.5cm, our goal was to climb a height of 15.1cm (fourth of the length) and the obstruction used in the obstacle course is 20cm, meaning that we met the set objective.

When it comes to measuring the angle of different obstructions in the obstacle course, the robot can accurately decide whether an obstruction is a slope or a vertical object. The only case where this does not succeed is when the robot is too far away from a small obstacle (see Table 5.10). This is because when the robot makes the second measurement at 10°, it misses the obstacle.

The worst case scenario is, therefore, when the robot judges a vertical object within 60cm of the robot as a slope or if it makes an incorrect measurement, judging the actual distance further away than it is. This would mean that the robot starts rolling and collides with the obstruction.

Overall, we have integrated a single distance sensor which means we can only identify obstacles in front of the robot. To distinguish obstacles in a real life environment, more sensors need to be implemented. However, integrating more sensors requires more time resources that are not available in this project.

6.3 Physical design

The physical design was made as easy as possible using CAD. The resulted snake was built using the purchased servos and printed links. Even though the design is very simple, it is also quite practical due to easy mounting between servo and link. The disadvantage with the links is that the robot did not reach the stability we wanted to achieve. This could have probably been fixed by making a design that would have encased the servos instead of just connecting them. Of course, that would also come with a downside of a limited rotation for each segment. The most apparent instability in the design is seen in Fig. 5.21, where we show a picture of an upside down segment. In the picture it is possible to see the uneven friction that leads to instability. This is an example of what could have been fixed if we had something encase the servos such that anything touching the ground would do it

uniformly.

The head of the snake makes a positive detail to the whole robot since it has a snake-like appearance which the other parts does not have, except when one looks at the combined shape of the robot. The design of the head is satisfactory, but it could have been improved the same way as the links to increase the robustness and stability of the robot.

The robot's tail consists of printed parts which hold all the electronics. Slots inside the tail to hold the electronics were avoided since it would both take much time as well as being unnecessary since the tail contains a lid. However, it would have been a desired feature of the tail if more electronics were included, as it would be easier to keep track of all the cables.

An additional design modification that would have improved the robot would be to have a heavier tail. If the tail had a higher weight, we would get closer to the theoretical maximum height. This is due to instead of having at least four segments on the obstacle during climbing, we could achieve the aspired center of mass by having the majority of the weight on said obstacle (see Section 2.1.3).

As mentioned in Section 3.3.1, the design of the segments allows for battery integration, but due to time constraints it was never implemented. This means that the usefulness of the WIFI module on the EspressIf ESP8266 has been reduced. Another part that could have been implemented is the gyroscope which could be used, for instance, to allow a faster movement while on flat ground (see Section 5.23).

6.4 Autonomous driving

The overall result while traversing the obstacle course is good, the only thing we would have preferably improved upon would be the stability in the design. This would have removed every major correction as they only occurred due to the instability. Although this problem was not accounted for, when we designed the segments we considered encasing each segment but decided against it as we preferred to design small and easy-to-use connections.

In the case of some runs, we made a false identification of the slope. This is because of the design of the obstacle course. When the robot climbs small heights, it pushes itself forward. This distance is almost as far as the length of the robot, which means that it will either end on the slope or extremely close to the slope. Furthermore, as mentioned in Section 6.2, the sensor readings only accurately works between 5-40cm. A solution to this problem would be to change the climbing for small heights, such that it does not push itself as much forward.

6.5 Mechanical Forces

When we created our movement algorithms, we used a geometrical analysis instead of a mechanical analysis. We opted for a basic mechanical analysis for some chosen critical points, as shown in Section 2.2. The basic analysis was made not only to calculate if any of the movements needed some alterations to achieve our goals but also to find out what kind of demands we had to put on the design. Besides, we were also interested in finding out what kind of limits the theoretical model had so it could be possible to compare it to our measured result (see Table 5.12).

To get a better idea of the mechanical forces, one needs to remember that the forces on the segments are quite complex since the system has many degrees of freedom. This means that there are many variables that depend on each other in convoluted ways. This leads to a complex system where not only a single segment needed to be taken into account, but the entirety of the system. The analysis gave us an understanding of how the system behaves, we, however, lacked the time to further give it a higher complexity.

If we had focused more on the analysis of mechanical forces, we could have made a more accurate mathematical model and a closed-loop control system. This would have resulted in a different forward locomotion in addition to a more stable system. Furthermore, this could have improved the climbing algorithm, because the main problem to climb higher vertical obstacles is balancing, which a closed-loop control system could handle.

More accurate mathematical models have already been done in many different projects. For example, the Ver-vite project done in 2003, which resulted in a model that shows how the mechanical forces act upon a segmented system with four segments[11].

6.6 Mobility

The mobility of a segmented snake-like robot will always depend on many moving parts. In this project, a limited number of moving parts was considered due to budget limitations. An early decision was to make the robot move in a vertical plane because it would make it possible to move over small vertical obstacles, e.g., a stair. However, it would be preferable to make the robot able to move in all directions, but in this project it had to be excluded due to limited numbers of motors and connectors. In particular, it would require twice the amount of servos if the robot was going to have as much movement sideways as it does vertically. This would also make the segments close to twice as long due to difficulties in efficient designing. An example of a solution to this came later in the project, that it may be possible to add wheels or servos in the front and back to make it possible for the

robot to rotate its body while standing still. This solution would make the robot keep its flexibility in the vertical plane in addition to added movement sideways. However, for this example, depending on where you locate the e.g. servos, it would demand the robot to disassemble each time it has to turn since it would need to use both the front and the back to turn.

In our case, besides being restricted to the vertical plane, our robot's overall movement has for the most part been satisfactory, with the only exceptions being our simplification of the pulse and missing encasing for the segments.

Besides using more sensors to get a better understanding of the surroundings, we could also have used a sensor to implement a closed-loop system. To make an accurate closed-loop system, it would have required a bigger feedback system, especially if we wanted to use a closed-loop system to balance the robot.

7
Conclusions

7.1 Final Result

We have designed and implemented a snake-like robot with eight connected segments. All movements are achieved by rotating the seven different motors. The robot's forward locomotion is implemented in two different ways, a pulse movement (*main movement*) at an average speed of 0.74($k = 0.5$) cm/s and 1.22cm/s($k = 1$), and a "rolling" movement (*fast movement*) at an average speed of 3.78cm/s. Additionally, the robot can accurately climb vertical obstacles up to 24cm.

In addition to implementing the algorithms mentioned above, four auxiliary movements have been modeled and implemented. The assembling and disassembling having a speed of 6s, the flip movement having a speed of 24s, and the rising movement having a speed of 6.3s.

Measuring distances can accurately be done for lengths between 5-40cm, and is sufficiently accurate for lengths between 40-80cm. It is also able to measure heights accurately between 5-20cm, as well as the type of the obstruction.

7.1.1 Main conclusion

The *fast movement* performs well, which greatly reduces the time at which the robot traverses the obstacle course. The remaining movement for forwarding locomotion, the *main movement*, greatly assists in traveling specific distances.

The climbing algorithm executes with high stability at heights up to 24cm; this is 8.9cm over our desired height. The robot can also climb inclines of 27°, this being 7° more than what we set out to achieve. This concludes that the robot succeeds in challenges more difficult than required (see Section 1.3).

The robot can also adapt to different obstacles in front of it to not only be able to traverse difficult terrain but also to traverse the terrain in the most efficient way it can achieve. It can run, with some minor and major corrections, a designed obstacle course that contains four obstacles. The biggest problem being the instability in design which makes the robot deviate from the path as well as making it hard for the robot to traverse a slope.

7.1.2 Additional conclusions

If we had focused on mechanical forces, we could have made a closed-loop system which could be used to develop a different balancing method and, thereby, possibly a better climbing algorithm. It could also be used to get a better understanding of what limitations a segmented snake-like robot has, and how to optimize the movements at critical points, e.g. rolling while in 45°.

The usage of additional sensors would have resulted in more accurate sensing of the environment. This would also have been needed to make a robust closed-loop system.

Adding an encasing for each servo to be the bulk of each segment would have increased the robot's stability and, therefore, enabled a higher movement speed. This speed would not only have improved the *fast movement* but also the climbing algorithm, as well as the auxiliary movements.

Another design modification that would greatly have helped the robot would be to add a heavier weight to the tail of the snake, so that the climbing algorithms could be used for higher obstacles, instead of being limited to a height of 24cm. If this modification had been made, we would have needed to change the rising movement, so as to not fall backward.

Bibliography

[1] A. E. Berman and J. Dorrier, *Technology Feels Like It's Accelerating — Because It Actually Is*, 2016. [Online]. Available: https://singularityhub.com/2016/03/22/technology-feels-like-its-accelerating-because-it-actually-is/.

[2] P. Mathur, "Terrain Classification for Traversability Analysis for Autonomous Robot Navigation in Unknown Natural Terrain", *International Journal of Engineering Science and Technology*, vol. 4, no. 01, pp. 38–49, 2012.

[3] H. B. Lillywhite, *How snakes work : structure, function and behavior of the world's snakes*. Oxford: Oxford University Press, 2014, p. 241, ISBN: 0199701571.

[4] S. Hirose and M. Mori, "Biologically Inspired Snake-like Robots", *2004 IEEE International Conference on Robotics and Biomimetics*, pp. 1–7, 2004. DOI: 10.1109/ROBIO.2004.1521742.

[5] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, *Snake Robots*. 2013, pp. 287–291, ISBN: 978-1-4471-2995-0. DOI: 10.1007/978-1-4471-2996-7. [Online]. Available: http://www.springerlink.com/index/10.1007/978-1-4471-2996-7.

[6] M. Yim, D. G. Duff, and K. D. Roufas, "Walk on the wild side", *IEEE Robotics & Automation Magazine*, vol. 9, no. 4, pp. 49–53, 2002, ISSN: 1070-9932. DOI: 10.1109/MRA.2002.1160071. [Online]. Available: http://ieeexplore.ieee.org/document/1160071/.

[7] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum Robots for Medical Applications: A Survey", *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015, ISSN: 15523098. DOI: 10.1109/TRO.2015.2489500.

[8] Wikimedia, *Hypiscopus_plumbea,_Rice_paddy_snake_2.jpg (6022×4019)*, 2017. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/8/8b/Hypiscopus_plumbea%2C_Rice_paddy_snake_2.jpg.

[9] C. Wright, A. Johnson, and A. Peck, "Design of a modular snake robot", *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007.*, pp. 2609–2614, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4399617.

[10] T. Ohashi, H. Yamada, and S. Hirose, "Loop forming snake-like robot ACM-R7 and its serpenoid oval control", *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 413–418, 2010, ISSN: 2153-0858. DOI: 10.1109/IROS.2010.5651467.

[11] D. Rincon and J. Sotelo, "Ver-vite: dynamic and experimental analysis for inchwormlike biomimetic robots", *IEEE Robotics & Automation Magazine*,

vol. 10, no. 4, pp. 3–7, 2003, ISSN: 1070-9932. DOI: 10.1109/MRA.2003.1256298.

A

Appendix 1

The appendix includes the datasheet for the Servo aswell as the Matlabcode used in chapter–.

Overview and Characteristics of Servo A1-16

A1-16 is a modular actuator, which combines a gear reducer, a DC motor and an embedded control board in one small package. A1-16 provides the necessary torque for building a small robot. Also, A1-16 could give much information of internal condition such as the internal temperature, supply voltage and so on. A1-16 is much easier to use for beginners and advance users than a traditional servo motor

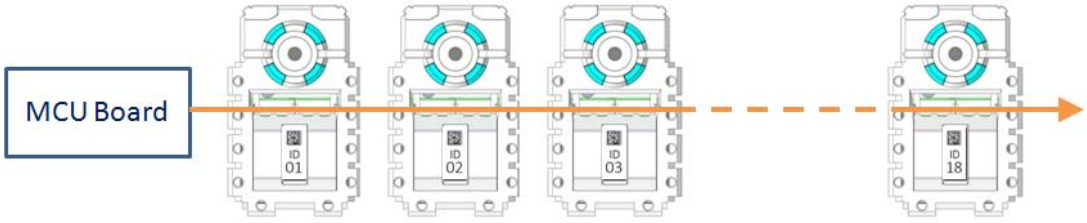
General Servo Motor Specifications

1. Operation voltage : 8 ~ 12 V(default)
2. Maximum speed : 70 ± 10 rpm
3. Stall torque : 25.0 kg-cm max
4. Rotary position feedback with 360° continuous rotation angle and maximum 330° effective position control range
5. Protocol type : Duplex UART 5V TTL serial communication(8, N, 1)
6. Communication Speed : 9600, 19200, 57600, 115200(default)
7. Feedback Information : Position, Temperature, Current, Voltage, etc

Dimensions of Servo Motor

1. Size : 50 x 32 x 40.5 mm
2. Weight : 60 ± 2 grams
3. Material : POM casing with metal gear

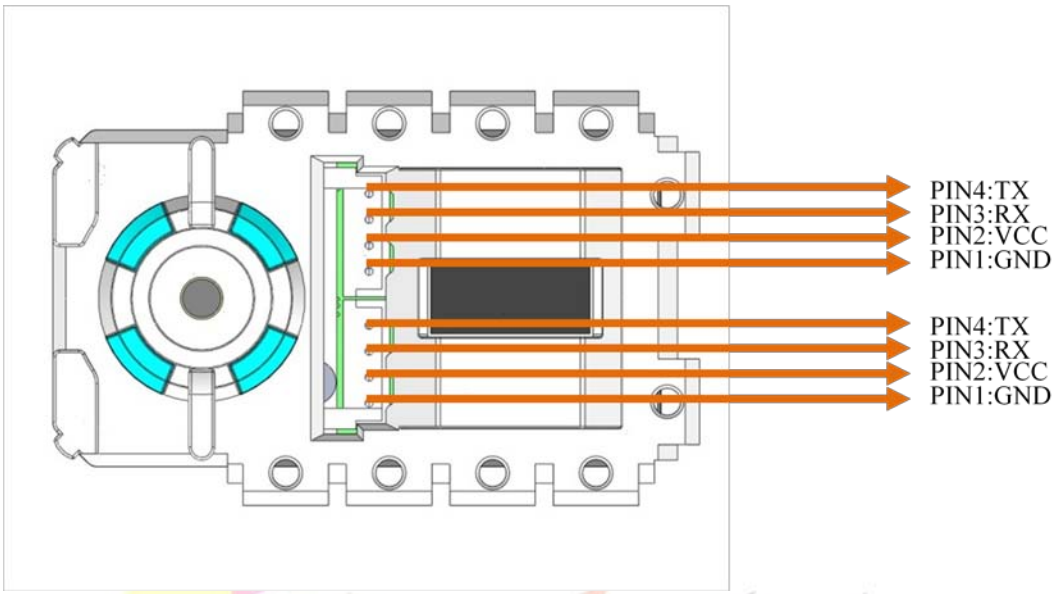
Wiring Connection



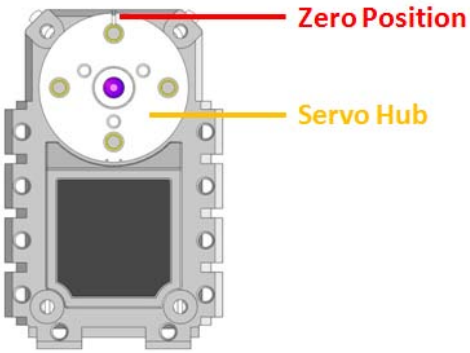
The A1-16 servos communicate with the main controller by daisy chain connection. Many A1-16 servos could be controlled by one single bus as shown above. Main controller provides power and sends control signal to A1-16 and receives respective data through the same bus. Every A1-16 servo has its unique ID value and communicates with the main controller by it, so user should be sure with the right ID

before assembly. When power is successfully applied to A1-16, the status LED blinks in sequence with red, white, blue and green LED twice.

The pin assignment of A1-16 is described as below. Each pin of two connector is internal connected. So A1-16 could function with any connector attached.

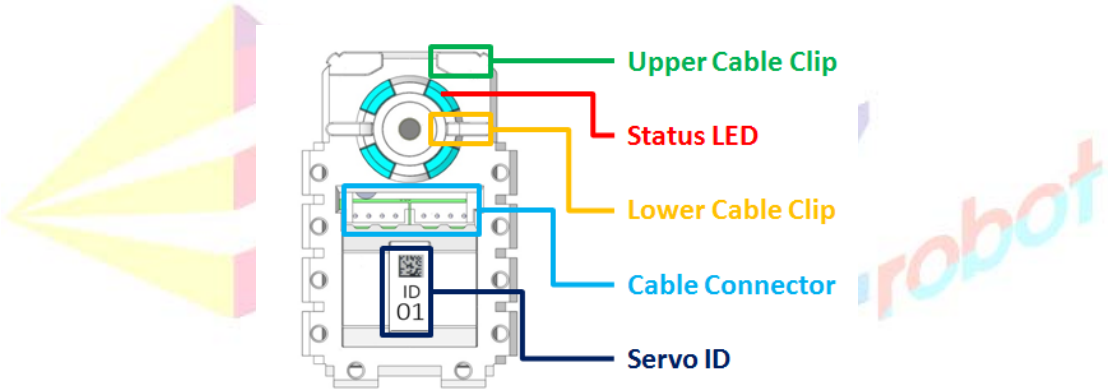


Front View of A1-16



1. Servo Hub: The servo hub is the rotation output part of A1-16.
2. Zero Position: The zero position shows the central position of A1-16 servo hub.

Back View of A1-16



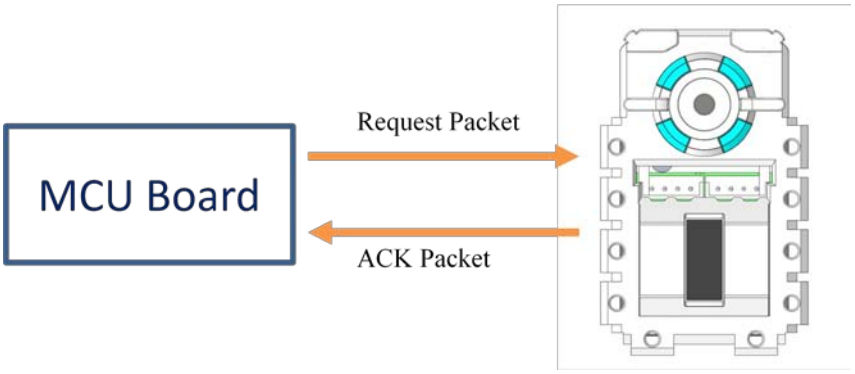
1. Cable Clip: The cable clip provide a route for cable.
2. Status LED: The status LED could indicate different error status to the users. The detail error information shows below.

Status Error	Error LED on/off
Normal Operation	White LED on
Exceed Potentiometer Range Error	Blue LED on
Over Voltage/Temperature/Current Limits Error	Red LED on/ White LED off
Requested Packet Error	Green LED on

3. Cable Connector: The cable connector provides power and communication signal for A1-16.
4. Servo ID: The servo ID shows default ID of A1-16.

Requested and ACK Packets

Main controller communicates with the servos in the UART network by sending a requested packet and receiving ACK packet back from the servo. Regardless of the number of servos in the network, only the servo with correct ID will acknowledge request packet and send the ACK packet to the main controller.



There are 9 UART command packets, as listed below, can be send from the master to servo controllers:

(1) EEP_WRITE	EEPROM parameters write
(2) EEP_READ	EEPROM parameters read
(3) RAM_WRITE	RAM parameters write
(4) RAM_READ	RAM parameters read
(5) I_JOG	independent control move
(6) S_JOG	synchronous control move
(7) STAT	read servo status
(8) ROLLBACK	reset all parameters to default values
(9) REBOOT	reset servo.

The servo controller may report ACK packets accordingly. The detail description of Requested and ACK packets are explained in Table1 through Table 9.

Table 1: Requested and ACK packets data string

bytes	1	2	3	4	5	6	7	8~107
description	header	header	packet size N	packet ID	CMD	check_ sum_1	check_ sum_2	data[i]
Requested packet	0xFF	0xFF	7~107	1~20, 254 (#)	0x01~ 0x09	(*)	(**)	...
ACK packet	0xFF	0xFF	7~107	1~20	0x40~ 0x49	(*)	(**)	...

Note: (#) When packet ID=254, broadcast ID, none of any servo will send ACK packet

(*) check_sum_1 = (N^ID^CMD^data[0]^data[1]^...^data[N-8]) & 0xFE

(**) check_sum_2 = (~check_sum_1) & 0xFE

Table 2: Requested and ACK packets CMDs

Requested packet CMD								
EEP_ WRITE	EEP_ READ	RAM_ WRITE	RAM_ READ	I_JOG	S_JOG	STAT	ROLLBACK	REBOOT
0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09
ACK packet CMD								
EEP_ WRITE	EEP_ READ	RAM_ WRITE	RAM_ READ	I_JOG	S_JOG	STAT	ROLLBACK	REBOOT
0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49

Table 3: Requested and ACK packets for EEP_WRITE(0x01) and RAM_WRITE(0x03)

(1) Requested packet for EEP_WRITE and RAM_WRITE CMD

1	2	3	4	5	6	7	8	9	+L
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	start addr.	length L	data[i]
0xFF	0xFF	9+L	1~20, 254	0x01, 0x03	0xXX	0xXX	0xXX	0xXX	...

Note: EEP_WRITE: 4 <= start addr. <= 53, , 5 <= start adds. + length <= 54

RAM_WRITE: 0 <= start addr. <= 47, 1 <= start adds. + length <= 48

(2) ACK packet for EEP_WRITE and RAM_WRITE CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status error	Status detail
0xFF	0xFF	9	1~20	0x41, 0x43	0xXX	0xXX	0xXX	0xXX

Note: The status_error and status_detail are listed in RAM parameters section.

Table 4: Requested and ACK packets for EEP_READ(0x02) and RAM_READ(0x04) CMD

(1) Requested packet for EEP_READ and RAM_READ CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	start addr.	length L
0xFF	0xFF	9	1~20	0x02, 0x04	0xXX	0xXX	0xXX	0xXX

Note: EEP_READ: 0 <= start addr. <= 53, , 1 <= start adds. + length <= 54

RAM_READ: 0 <= start addr. <= 79, 1 <= start adds. + length <= 80

(2) ACK packet for EEP_READ and RAM_READ CMD

1	2	3	4	5	6	7	8	9	+L
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	start addr.	length L	data[i]
0xFF	0xFF	9+L	1~20	0x42, 0x44	0xXX	0xXX	0xXX	0xXX	...

Table 5: Requested and ACK packets for I-JOG (0x05) CMD

(1) Requested packet for I-JOG CMD

1	2	3	4	5	6	7	+5	+(n-1)*5
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	I-JOG data	...
0xFF	0xFF	7+5*n	1~20, 254	0x05	0xXX	0xXX	(5-1)	...

Note: n=number of motor IDs send.

(5-1) I-JOG data

1	2	3	4	5
goal.lsb	goal.msb	set	ID	playtime (unit:10 msec)
0xXX	0xXX	0xXX	1~20	0xXX

Note: (1) goal=0~1023; (2) play time may be modified for a long movement;

(3) set = 0 (position control) / 1 (speed control) / 2 (torque off) /3 (position control servo on)

(2) ACK packet for I-JOG CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status error	status detail
0xFF	0xFF	9	1~20	0x45	0xXX	0xXX	0xXX	0xXX

Table 6: Requested and ACK packets for S-SOG (0x06) CMD

(1) Requested packet for S-JOG CMD

1	2	3	4	5	6	7	8	+4	+(n-1)*4
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	play time	S-JOG data	...
0xFF	0xFF	8+4*n	1~20, 254	0x06	0xXX	0xXX	0xXX	(6-1)	...

Note: n=number of motor IDs send.

(6-1) S-JOG data

1	2	3	4
goal.lsb	goal.msb	set	ID
0xXX	0xXX	0xXX	1~20

Note: (1) goal=0~1023; (2) goal position may not be reached for a short play time;
(3) set = 0 (position control) / 1 (speed control) / 2 (torque off) / 3 (position control servo on)

(2) ACK packet for S-JOG CMD

1	2	3	4	5	6	7	8	9
Header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status error	Status detail
0xFF	0xFF	9	1~20	0x46	0xXX	0xXX	0xXX	0xXX

Table 7: Requested and ACK packets for STAT(0x07) CMD

(1) Requested packet for STAT CMD

1	2	3	4	5	6	7
header	header	size	ID	CMD	check_ sum_1	check_ sum_2
0xFF	0xFF	7	1~20	0x07	0xXX	0xXX

(2) ACK packet for STAT CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status_ error	status_ detail
0xFF	0xFF	17	1~20	0x47	0xXX	0xXX	0xXX	0xXX

10	11	12	13	14	15	16	17
PWM. lsb	PWM. msb	pos_ref. lsb	pos_ref. msb	position. lsb	position. msb	Ibus. lsb	Ibus. msb
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

Table 8: Requested and ACK packets for ROLLBACK(0x08) CMD

(1) Requested packet for ROLLBACK CMD

1	2	3	4	5	6	7
header	header	size	ID	CMD	check_ sum_1	check_ sum_2
0xFF	0xFF	7	1~20, 254	0x08	0xXX	0xXX

(2) ACK packet for ROLLBACK CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status_ error	status_ detail
0xFF	0xFF	9	1~20	0x48	0xXX	0xXX	0xXX	0xXX

Table 9: Requested and ACK packets for REBOOT(0x09) CMD

(1) Requested packet for REBOOT CMD

1	2	3	4	5	6	7
header	header	size	ID	CMD	check_ sum_1	check_ sum_2
0xFF	0xFF	7	1~20, 254	0x09	0xXX	0xXX

(2) ACK packet for REBOOT CMD

1	2	3	4	5	6	7	8	9
header	header	size	ID	CMD	check_ sum_1	check_ sum_2	status_ error	status_ detail
0xFF	0xFF	9	1~20	0x49	0xXX	0xXX	0xXX	0xXX

3. EEPROM & RAM Parameters

The system parameters saved in EEOROM and RAM are shown in Table 10. There are 54 bytes parameter data in EEPROM and 80 bytes parameter data in RAM, in which the first 48 bytes of RAM data are same as the data in RAM from address of 6 to 54. The EEPROM data can be read and written to; some of the RAM data are read only.

Table 10: EEPROM & RAM Parameters

EEPROM Addr.	RAM Addr.	Parameter	Bytes	R/W /RO	Default Value
0		Model_No	1	RO	0x01
1		Year	1	RO	0x0F
2		Version/Month	1	RO	0x3A
3		Day	1	RO	0x01
4		Reserved	1	RO	0x01
5		Baud_Rate	1	R/W	0x0C
6	0	sID	1	R/W	0x01
7	1	ACK_Policy	1	R/W	0x02
8	2	Alarm_LED_Policy	1	R/W	0x00
9	3	Torque_Policy	1	R/W	0x01
10	4	SPDctrl_Policy	1	R/W	0x01
11	5	Max_Temperature	1	R/W	0x4B
12	6	Min_Voltage	1	R/W	0x77
13	7	Max_Voltage	1	R/W	0xE8
14	8	Acceleration_Ratio	1	R/W	0x00
15	9	Reserved	1	R/W	0xFF
16	10	Reserved	1	R/W	0x00
17	11	Reserved	1	R/W	0x00
18	12	Max_Wheel_Ref_Position	2	R/W	0x042E
20	14	Reserved	1	R/W	0x00
21	15	Reserved	1	R/W	0x00
22	16	Max_PWM	2	R/W	0x03FF
24	18	Overload_Threshold	2	R/W	0x00CC
26	20	Min_Position	2	R/W	0x00
28	22	Max_Position	2	R/W	0x03FF
30	24	Position_Kp	2	R/W	0x0F00
32	26	Position_Kd	2	R/W	0x0800

34	28	Position_Ki	2	R/W	0x0000
36	30	Close_to_Open_Ref_Position	2	R/W	0x03FF
38	32	Open_to_Close_Ref_Position	2	R/W	0x00
40	34	Reserved	2	R/W	0x03FF
42	36	Ramp_Speed	2	R/W	0x03FF
44	38	LED_Blink_Period	1	R/W	0x00
45	39	Reserved	1	R/W	0x00
46	40	Packet_Timeout_Detection_Period	1	R/W	0x0A
47	41	Reserved	1	R/W	0x00
48	42	Overload_Detection_Period	1	R/W	0x19
49	43	Reserved	1	R/W	0x00
50	44	Inposition_Margin	1	R/W	0x01
51	45	Over_Voltage_Detection_Period	1	R/W	0xFF
52	46	Over_Temperature_Detection_Period	1	R/W	0x0A
53	47	Calibration_Difference	1	R/W	0xXX
	48	Status_Error	1	R/W	0x00
	49	Status_Detail	1	R/W	0x40
	50	Reserved	1	R/W	0x00
	51	Reserved	1	R/W	0x00
	52	Reserved	1	R/W	0x01
	53	LED_Control	1	R/W	0x00
	54	Voltage	1	RO	0xXX
	55	Temperature	1	RO	0xXX
	56	Current_Control_Mode	1	RO	0x02
	57	Tick	1	RO	0x00
	58	Reserved	2	RO	0xXXXX
	60	Joint_Position	2	RO	0xXXXX
	62	Reserved	2	RO	0x0000
	64	PWM_Output_Duty	2	RO	0x0000
	66	Bus_Current	2	RO	0x0000
	68	Position_Goal	2	RO	0xXXXX
	70	Position_Ref	2	RO	0xXXXX
	72	Omega_Goal	2	RO	0x0000
	74	Omega_Ref	2	RO	0x0000
	76	Requested_Counts	2	RO	0x0000
	78	ACK_Counts	2	RO	0x0000

The description of EEPROM and RAM parameters above are summarized below.

(E0) Model_No：Servo model name

(E1) Year：Year

(E2) Version/Month：bit0~3：month, bit4~8：version of servo firmware

(E5) Baud_Rate：

0x01：9600

0x02：19200

0x06：57600

0x0C：115200

(E6,R0) sID：Servo ID, 1, 2, ..., 19, 20 ... , 253

(E7,R1) ACK_Policy：

only STAT command reply：0

only EEPROM/RAM RAED and STAT commands reply：1

all commands reply：2

(E8,R2) Alarm_LED_Policy：bit i = 0 (System Alarm LED), 1 (User LED)

Bit 0：White LED

Bit 1：Blue LED

Bit 2：Green LED

Bit 3：Red LED

(E9,R3) Torque_Policy：Shut down Motor when Voltage/Load/Temperature

Torque Free Control：0

Torque Limited：1

(E10,R4) SPDctrl_Policy：Speed open/close loop control

Open Loop Control：0

Close Loop Control：1

(E11,R5) Max_Temperature：The limit of A1-16 servo operating temperature. The value is in Degrees Celsius.

(E12,R6) Min_Voltage：The min value of A1-16 servo operating voltage. The value is 16 times the actual voltage.

(E13,R7) Max_Voltage：The max value of A1-16 servo operating voltage. The value is 16 times the actual voltage.

(E14,R8) Acceleration_Ratio = 0, 1, 2, ..., 50

Play_time	Acceleration_Ratio	Referenced position trajectory
0		Ramp-to-step position command, see (36)
1~255	0	Constant speed profile
1~255	1~50	T-curve speed profile

Note: acceleration_time = deceleration_time = play_time * Acceleration_Ratio/100

(E18,R12) Max_Wheel_Ref_Position：Start virtual position for speed close loop control.

(E22,R16) Max_PWM：The max value of A1-16 servo output torque.

(E24,R18) Overload_Threshold：The max value of A1-16 servo output torque.

(E26,R20) Min_Position：Min operational angle

(E28,R22) Max_Position：Max operational angle

(E30,R24) Position_Kp：msb is the integer number and lsb is the decimal number. The P control law is implemented below with a sampling time of 10 msec

(E32,R26) Position_Kd：msb is the integer number and lsb is the decimal number. The PD control law is implemented below with a sampling time of 10 msec

(E34,R28) Position_Ki：msb is the integer number and lsb is the decimal number. The PID control law is implemented below with a sampling time of 10 msec,

(E36,R30) Close_to_Open_Ref_Position：close loop continuous rotate mode close to open position.

(E38,R32) Open_to_Close_Ref_Position：close loop continuous rotate mode open to close position.

(E42,R36) Ramp_Speed = 0 (step position command), 1~1023 (slope of ramp-to-step)

(E44,R38) LED_Blink_Period：Blinking Period of LED with a sampling time of 10 msec.

(E46,R40) Packet_Timeout_Detection_Period：Packet Timeout Detection Period of LED with a sampling time of 10 msec. 1 = 10ms

(E48,R42) Overload_Detection_Period：Overload Detection Period of servo with a sampling time of 10 msec. 1 = 10ms

(E51,R45) Over_Voltage_Detection_Period：Over Voltage Detection Period of servo with a sampling time of 10 msec. 1 = 10ms

(E52,R46) Over_ Temperature _Detection_Period：Over Temperature Detection Period of servo with a sampling time of 10 msec. 1 = 10ms

(E53,R47) Calibration_Difference：The difference between newtral point and position raw data.

(R48) status_error

bit	Mask	Default	Status Error	Error LED on/off
1	0x01	0	Exceed Potentiometer Range Error	Blue LED on
2	0x02	0	Over Voltage Limits Error	Red LED on/ White LED off
3	0x04	0	Over Temperature Error	Red LED on/ White LED off
4	0x08	0	Overload/Over-current Error	Red LED on/ White LED off
5	0x10	0	Reserved	None

6	0x20	0	Requested Packet Checksum Error	Green LED on
7	0x40	0	Requested Packet Data Error	Green LED on
8	0x80	0	Requested Packet RX FIFO Error	Green LED on

(R49) status_detail

bit	Mask	Default	Status Detail
1	0x01	0	Reserved
2	0x02	0	Reserved
3	0x04	0	Reserved
4	0x08	0	Reserved
5	0x10	0	Motor Moving
6	0x20	0	Motor In-Position (Position control mode only)
7	0x40	0	1: Torque on (Position/Speed control), 0: Torque off
8	0x80	0	Motor Braked

(R53) LED_Control：bit i = 0 (LEDi off), 1 (LEDi on); (see Alarm_LED_Policy)

Bit 0：White LED

Bit 1：Blue LED

Bit 2：Green LED

Bit 3：Red LED

(R54) Voltage：The voltage currently applied to servo. The Value is 16 times the actual voltage.

(R55) Temperature：The internal temperature of motor in Degrees Celsius.

(R56) Current_Control_Mode：0 (position control), 1 (speed control), 2 (torque off)

(R57) Tick：Time servo operation. 1 = 10ms

(R60) Joint Postion：Servo Position

(R64) PWM_Output_Duty：The torque applied to motor

(R66) Bus_Current：The Current applied to motor. The Value is 200 times the actual current.

(R68) Position_Goal：Servo goal of position control mode

(R70) Position_Ref：Ref point for position control

(R72) Omega_Goal：Goal speed of speed close-loop control

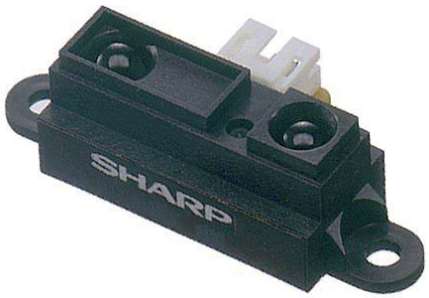
(R74) Omega_Ref：Ref speed of speed close-loop control

(R76) Requested_Counts：Total # of requested packets received since power on.

(R78) ACK_Counts：Total # of ACK packets send since power on.

GP2Y0A21YK0F

Distance Measuring Sensor Unit
Measuring distance: 10 to 80 cm
Analog output type



■Description

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

■Features

1. Distance measuring range : 10 to 80 cm
2. Analog output type
3. Package size : 29.5×13×13.5 mm
4. Consumption current : Typ. 30 mA
5. Supply voltage : 4.5 to 5.5 V

■Agency approvals/Compliance

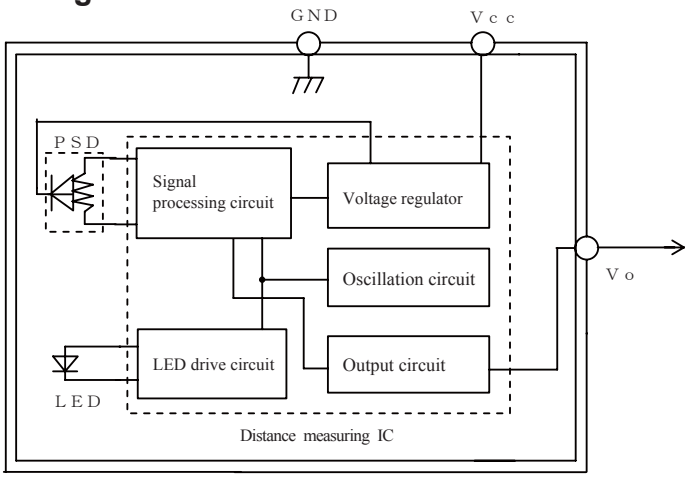
1. Compliant with RoHS directive (2002/95/EC)

■Applications

1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Robot cleaner
3. Sensor for energy saving
   (ATM, Copier, Vending machine)
4. Amusement equipment
   (Robot, Arcade game machine)

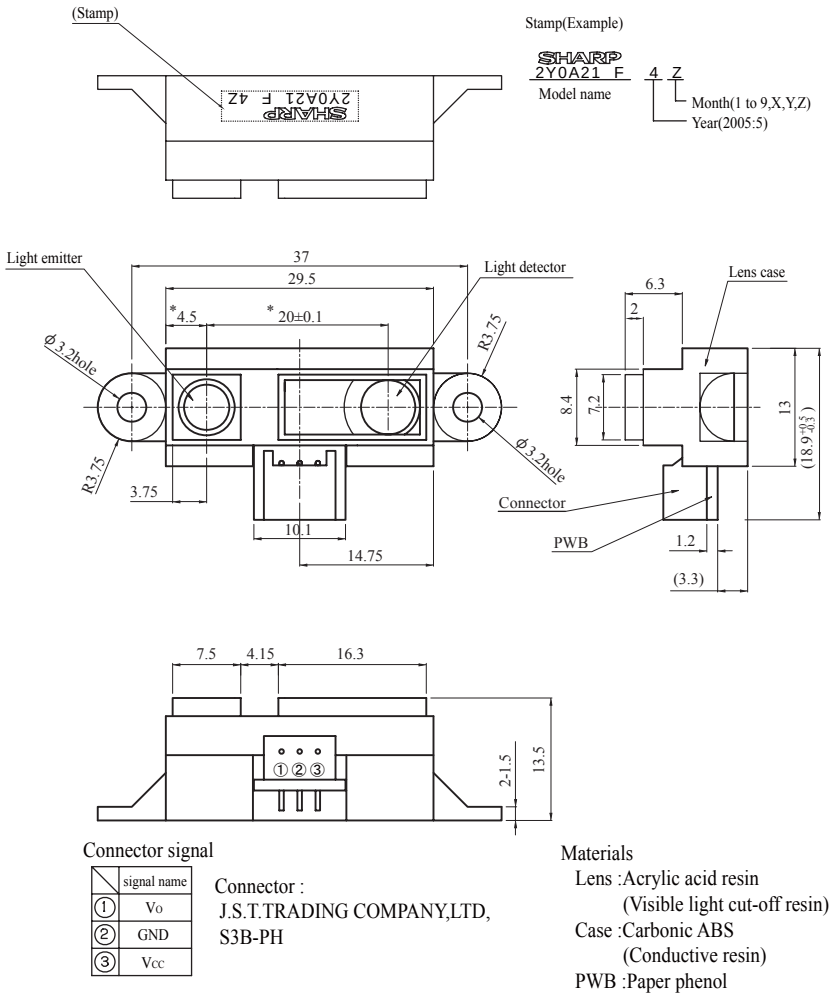
Sheet No.: E4-A00201EN
Date Dec.01.2006

■Block diagram



■Outline Dimensions

(Unit : mm)



Connector signal

	signal name
①	Vo
②	GND
③	Vcc

Connector :
J.S.T.TRADING COMPANY,LTD,
S3B-PH

Materials
Lens :Acrylic acid resin
(Visible light cut-off resin)
Case :Carbonic ABS
(Conductive resin)
PWB :Paper phenol

Note 1. The dimensions marked * are described the dimensions of lens center position.
Note 2. Unspecified tolerances shall be ± 0.3 mm.
Note 3. The dimensions in parenthesis are shown for reference.

Product mass : Approx. 3.6g

■Absolute Maximum Ratings

(T_a=25℃,V_{CC}=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	-0.3 to +7	V
Output terminal voltage	V_O	-0.3 to V_{CC}+0.3	V
Operating temperature	T_{opr}	-10 to +60	℃
Storage temperature	T_{stg}	-40 to +70	℃

■Electro-optical Characteristics

(T_a=25℃,V_{CC}=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Average supply current	I_{CC}	L=80cm (Note 1)	—	30	40	mA
Distance measuring	ΔL	(Note 1)	10	—	80	cm
Output voltage	V_O	L=80cm (Note 1)	0.25	0.4	0.55	V
Output voltage differential	ΔV_O	Output voltage differece between L=10cm and L=80cm (Note 1)	1.65	1.9	2.15	V

* L : Distance to reflective object

Note 1 : Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27・white face, reflectance; 90%)

■Recommended operating conditions

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	4.5 to 5.5	V

Fig. 1 Timing chart

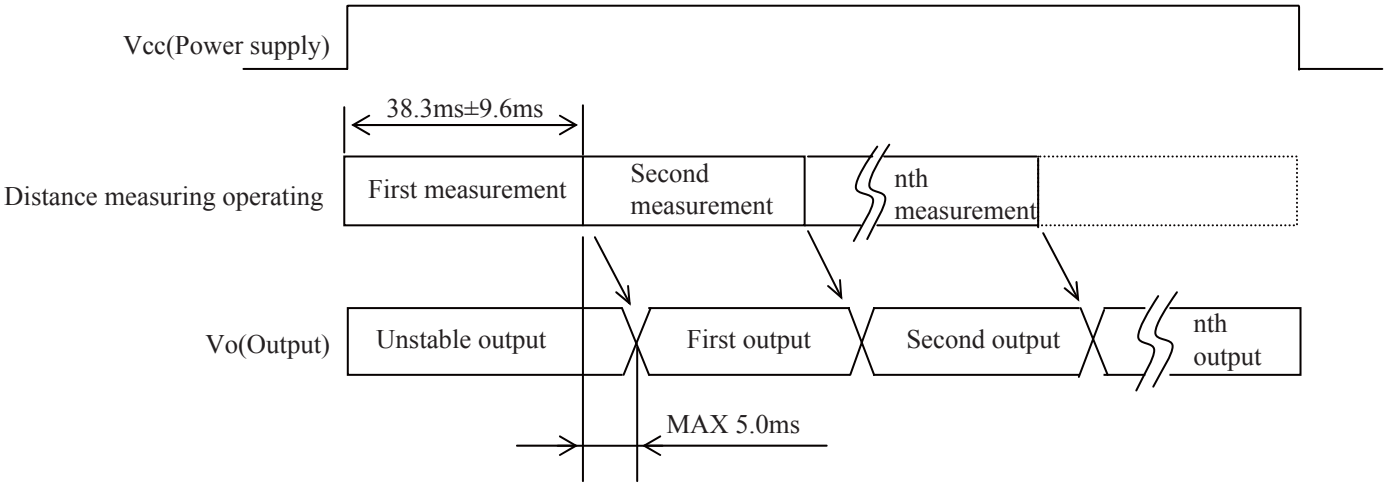
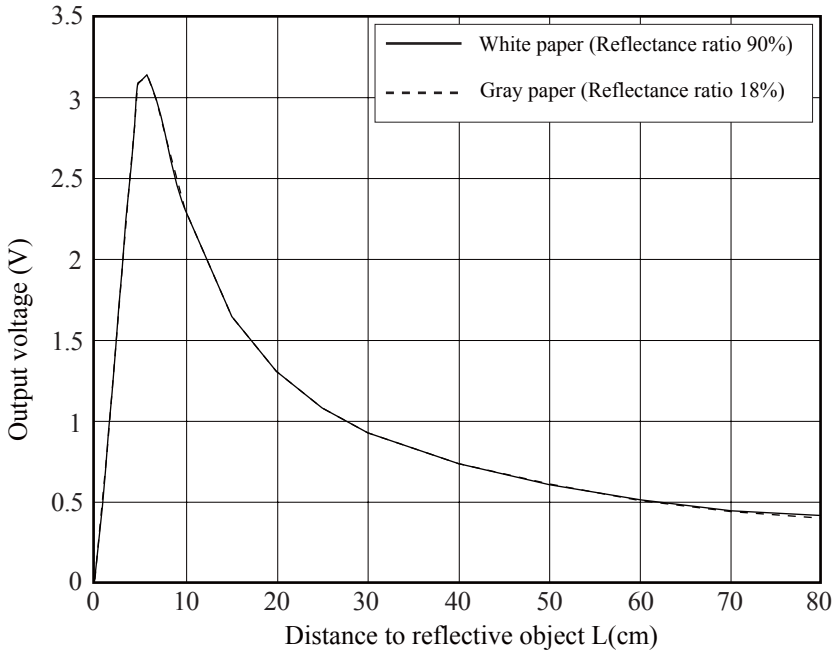
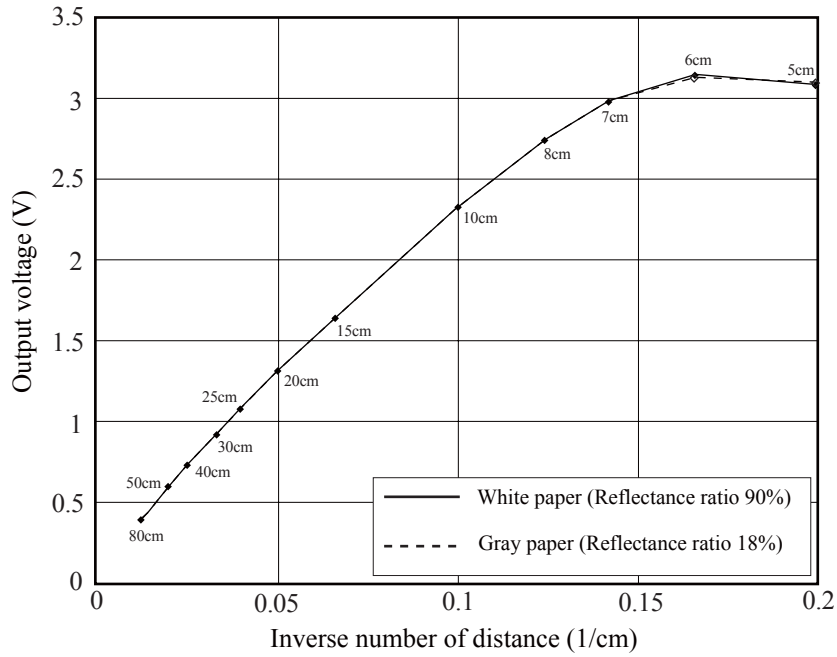


Fig. 2 Example of distance measuring characteristics(output)





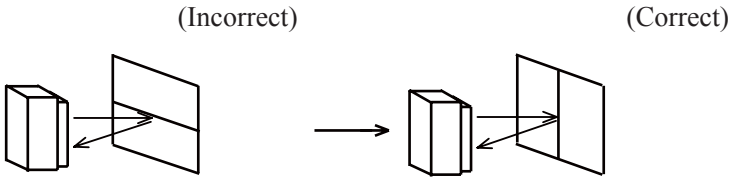
■Notes

●Advice for the optics

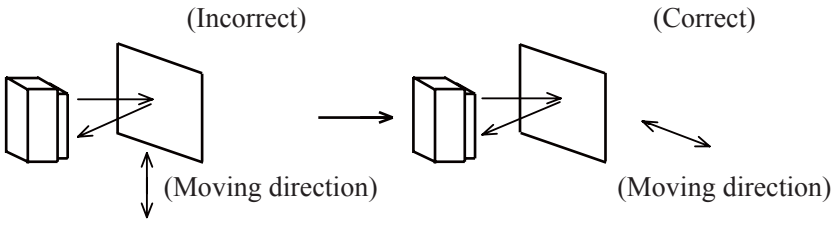
• The lens of this device needs to be kept clean. There are cases that dust, water or oil and so on deteriorate the characteristics of this device. Please consider in actual application.
• Please don't do washing. Washing may deteriorate the characteristics of optical system and so on. Please confirm resistance to chemicals under the actual usage since this product has not been designed against washing.

●Advice for the characteristics

• In case that an optical filter is set in front of the emitter and detector portion, the optical filter which has the most efficient transmittance at the emitting wavelength range of LED for this product ($\lambda = 870 \pm 70$nm), shall be recommended to use. Both faces of the filter should be mirror polishing. Also, as there are cases that the characteristics may not be satisfied according to the distance between the protection cover and this product or the thickness of the protection cover, please use this product after confirming the operation sufficiently in actual application.
• In case that there is an object near to emitter side of the sensor between sensor and a detecting object, please use this device after confirming sufficiently that the characteristics of this sensor do not change by the object.
• When the detector is exposed to the direct light from the sun, tungsten lamp and so on, there are cases that it can not measure the distance exactly. Please consider the design that the detector is not exposed to the direct light from such light source.
• Distance to a mirror reflector can not be sometimes measured exactly. In case of changing the mounting angle of this product, it may measure the distance exactly.
• In case that reflective object has boundary line which material or color etc. are excessively different, in order to decrease deviation of measuring distance, it shall be recommended to set the sensor that the direction of boundary line and the line between emitter center and detector center are in parallel.



• In order to decrease deviation of measuring distance by moving direction of the reflective object, it shall be recommended to set the sensor that the moving direction of the object and the line between emitter center and detector center are vertical.



●Advice for the power supply

• In order to stabilize power supply line, we recommend to insert a by-pass capacitor of 10μF or more between Vcc and GND near this product.

●Notes on handling

• There are some possibilities that the internal components in the sensor may be exposed to the excessive mechanical stress. Please be careful not to cause any excessive pressure on the sensor package and also on the PCB while assembling this product.

● Presence of ODC etc.

This product shall not contain the following materials.
And they are not used in the production process for this product.
Regulation substances : CFCs, Halon, Carbon tetrachloride, 1.1.1-Trichloroethane (Methylchloroform)

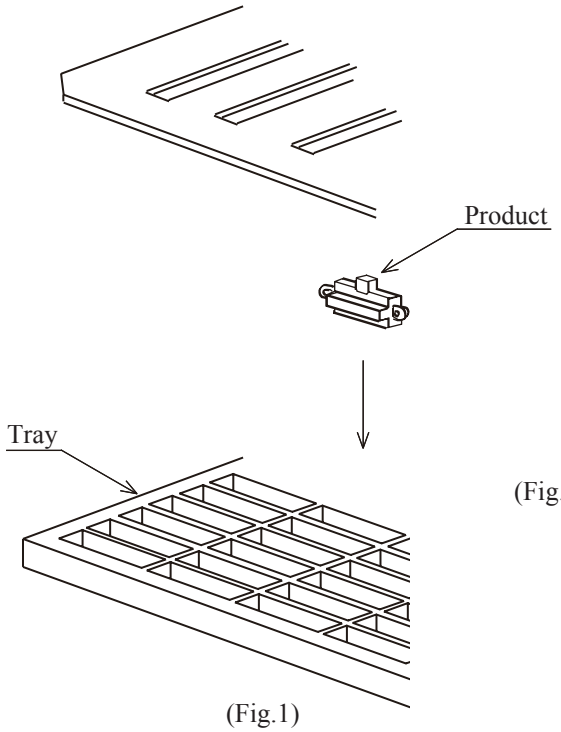
Specific brominated flame retardants such as the PBB and PBDE are not used in this product at all.

This product shall not contain the following materials banned in the RoHS Directive (2002/95/EC).

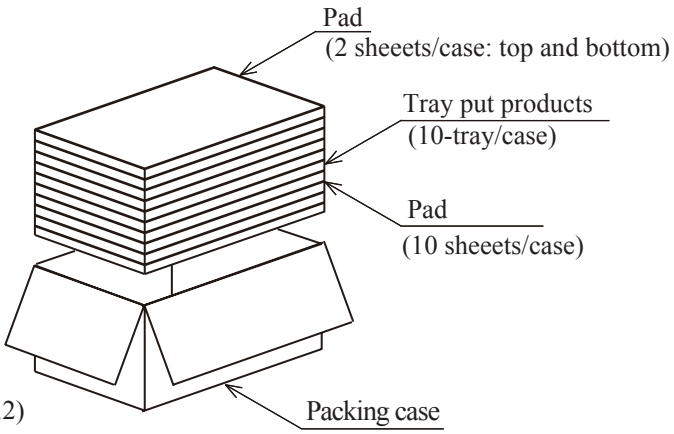
- Lead, Mercury, Cadmium, Hexavalent chromium, Polybrominated biphenyls (PBB), Polybrominated diphenyl ethers (PBDE).

■Package specification

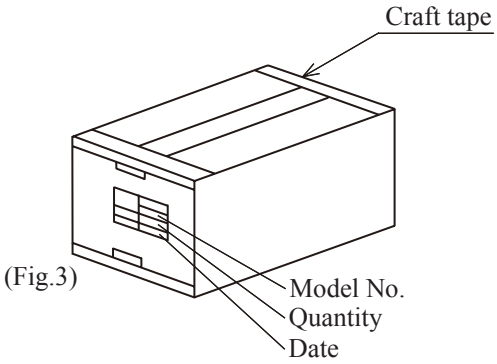
Package composition



(Fig.1)



(Fig.2)



(Fig.3)

Packaging method

1.Put products of 100pcs. in tray. packing method is showed in the above fig.(Fig.1)
2.Put them(10-tray) in the packing box. Put pads on their top and bottom.
And put pads on each trays(Total 10 sheets) (Fig.2).
3.Seal the packing box with craft tape.
Print the model No.,quantity,inspection date (1000 pcs./a packing box)(Fig.3).

■Important Notices

· The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.

· Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.

· Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

(i) The devices in this publication are designed for use in general electronic equipment designs such as:
- --- Personal computers
- --- Office automation equipment
- --- Telecommunication equipment [terminal]
- --- Test and measurement equipment
- --- Industrial control
- --- Audio visual equipment
- --- Consumer electronics

(ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
- --- Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
- --- Traffic signals
- --- Gas leakage sensor breakers
- --- Alarm equipment
- --- Various safety devices, etc.

(iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
- --- Space applications
- --- Telecommunication equipment [trunk lines]
- --- Nuclear power control equipment
- --- Medical and other life support equipment (e.g., scuba).

· If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Law of Japan, it is necessary to obtain approval to export such SHARP devices.

· This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.

· Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

```
clc
cla
l=7; %l?ngden p? segment
speed=1000000; %hastighet p? simuleringen
pulser=5; %antal pulser som ska utf?ras


X=[0, l, 2*l, 3*l, 4*l, 5*l, 6*l, 7*l, 8*l];
Y=[0, 0, 0, 0, 0, 0, 0, 0, 0];
Psi=[0, 0, 0, 0, 0, 0, 0];
Psiof7=0; Psiof6=0; Psiof5=0; Psiof4=0; Psiof3=0; Psiof2=0; Psiof1=0;


for varv=1:pulser


k=1; %l?ngden per pulse

%Anv?ndbara vinklar!
alpha=acos((12-6*k+k^2)/(12-4*k));
beta=acos((6*k-4-k^2)/(4));
gamma=acos((6-6*k+k^2)/(6-2*k));
delta=acos((4*k-2-k^2)/2);
sigma=(pi-delta)/2;
%steg1
n=0;
while n<=1
    n=n+0.01;
    if n<=1/1.1
    Psiof3=gamma*n*1.1;
    end
    Psiof2=(beta-pi)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
```

```
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof3=gamma*(1-n)+sigma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof3));
    Psiof2=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))-acos(d/(2*l)))-pi;
    Psiof1=pi-(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof3))/2)));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
    Psiof4=alpha*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof4));
    Psiof3=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof2=(acos(1-(d^2)/(2*l^2)))-pi;
    Psiof1=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
```

```
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof4=alpha*(1-n)+gamma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof4));
    Psiof3=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof2=acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof4))/2))-pi;
    Psiof1=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

    Psiof1=real(Psiof1);
    Psiof2=real(Psiof2);
    Psiof3=real(Psiof3);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;
```

```
plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof4=gamma*(1-n)+sigma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof4));
    Psiof3=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))-acos(d/(2*l)))-pi;
    Psiof2=pi-(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof4))/2)));
    Psiof1=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))-acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
    Psiof5=alpha*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof5));
    Psiof4=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof3=(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof5))/2)))-pi;
    Psiof2=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;
```

```
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof5=alpha*(1-n)+gamma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof5));
    Psiof4=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof3=acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof5))/2))-pi;
    Psiof2=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

    Psiof2=real(Psiof2);
    Psiof3=real(Psiof3);
    Psiof4=real(Psiof4);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));
```

```
Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof5=gamma*(1-n)+sigma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof5));
    Psiof4=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))-acos(d/(2*l)))-pi;
    Psiof3=pi-(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof5))/2)));
    Psiof2=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))-acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
    Psiof6=alpha*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof6));
    Psiof5=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof4=(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof6))/2)))-pi;
    Psiof3=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
```

```
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof6=alpha*(1-n)+gamma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof6));
    Psiof5=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof4=acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof6))/2))-pi;
    Psiof3=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

    Psiof3=real(Psiof3);
    Psiof4=real(Psiof4);
    Psiof5=real(Psiof5);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
```

```
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof6=gamma*(1-n)+sigma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof6));
    Psiof5=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))-acos(d/(2*l)))-pi;
    Psiof4=pi-(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof6))/2)));
    Psiof3=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))-acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
    Psiof7=alpha*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof7));
    Psiof6=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof5=(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof7))/2)))-pi;
    Psiof4=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

Psi(1,7)=Psiof7;
```

```
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof7=alpha*(1-n)+gamma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof7));
    Psiof6=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))+acos(d/(2*l)))-pi;
    Psiof5=acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof7))/2))-pi;
    Psiof4=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))+acos(d/(2*l));

    Psiof4=real(Psiof4);
    Psiof5=real(Psiof5);
    Psiof6=real(Psiof6);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
```

```
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof7=gamma*(1-n)+sigma*n;
    d=l*sqrt(10-6*k+k^2-(6-2*k)*cos(Psiof7));
    Psiof6=(acos((d^2-8*l^2+6*k*l^2-k^2*l^2)/(2*d*l))-acos(d/(2*l)))-pi;
    Psiof5=pi-(acos(-((8-6*k+k^2-(6-2*k)*cos(Psiof7))/2)));
    Psiof4=acos((d^2+8*l^2-6*k*l^2+k^2*l^2)/(6*d*l-2*d*k*l))-acos(d/(2*l));

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
%avslut
n=0;
while n<=1
    n=n+0.01;
    Psiof5=sigma*(1-n);
    Psiof7=sigma*(1-(n));
    Psiof6=(delta-pi)*(1-(n));
```

```
Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,2)=X(1,1)+l*cos(-Psi(1,1));
X(1,3)=X(1,2)+l*cos(-Psi(1,2));
X(1,4)=X(1,3)+l*cos(-Psi(1,3));
X(1,5)=X(1,4)+l*cos(-Psi(1,4));
X(1,6)=X(1,5)+l*cos(-Psi(1,5));
X(1,7)=X(1,6)+l*cos(-Psi(1,6));
X(1,8)=X(1,7)+l*cos(-Psi(1,7));
X(1,9)=X(1,8)+l*cos(pi-(Psiof5+pi+Psiof6+Psiof7));%-------------fixa-------------%

Y(1,2)=Y(1,1)+l*sin(-Psi(1,1));
Y(1,3)=Y(1,2)+l*sin(-Psi(1,2));
Y(1,4)=Y(1,3)+l*sin(-Psi(1,3));
Y(1,5)=Y(1,4)+l*sin(-Psi(1,4));
Y(1,6)=Y(1,5)+l*sin(-Psi(1,5));
Y(1,7)=Y(1,6)+l*sin(-Psi(1,6));
Y(1,8)=Y(1,7)+l*sin(-Psi(1,7));
Y(1,9)=Y(1,8)+l*sin(pi-(Psiof5+pi+Psiof6+Psiof7));%-------------fixa-------------%

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 9*l+pulser*l*k -1 9*l+pulser*l*k])
pause(1/speed)
end
end
%%
clc
cla
l=7; %l?ngden p? segment
speed=10000000; %hastighet p? simuleringen
puls=3; %antal pulser som ska utf?ras
X=[0, l, 2*l, 3*l, 4*l, 5*l, 6*l, 7*l, 8*l];
Y=[0, 0, 0, 0, 0, 0, 0, 0, 0];
Psi=[0, 0, 0, 0, 0, 0, 0];
Psiof7=0; Psiof6=0; Psiof5=0; Psiof4=0; Psiof3=0; Psiof2=0; Psiof1=0;
%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*n;
    Psiof3=(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
```

```
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*(1-n);
    Psiof4=(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
```

```
    Psiof1=(pi/2)*n;
    Psiof3=(pi/2)*(1-n);
    Psiof5=(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%f?rdig
for spin=1:puls
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*n;
    Psiof4=(pi/2)*(1-n);
    Psiof6=(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
```

```
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
n=0;
while n<=1
    n=n+0.01;
    Psiof1=(pi/2)*(1-n);
    Psiof3=(pi/2)*n;
    Psiof5=(pi/2)*(1-n);
    Psiof7=(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*(1-n);
    Psiof4=(pi/2)*n;
    Psiof6=(pi/2)*(1-n);
    %Psiof7=(pi/2)*n

Psi(1,7)=Psiof7+(pi/2)*n;
Psi(1,6)=Psiof7+Psiof6+(pi/2)*n;
```

```
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*n;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*n;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*n;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*n;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*n;

X(1,8)=X(1,9)-l*cos((pi/2)*n);
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,8)=Y(1,9)+l*sin((pi/2)*n);
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
Xf=X;
Yf=Y;
Xvu=[];
Yvu=[];
n=0;
while n<=1
    n=n+0.01;
    Psiof3=(pi/2)*(1-n);
    Psiof5=(pi/2)*n;
    Psiof7=(pi/2)*(1-n);
    Psiof1=(pi/2)*n;

Psi(1,7)=Psiof7+(pi/2)*(n+1);
Psi(1,6)=Psiof7+Psiof6+(pi/2)*(n+1);
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*(n+1);
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*(n+1);
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*(n+1);
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*(n+1);
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*(n+1);

X(1,9)=Xf(1,9)+l*(1-cos((pi/2)*n));
X(1,8)=X(1,9)+l*cos((pi/2)*(1-n));
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,9)=Yf(1,9)+l*sin((pi/2)*n);
```

```
Y(1,8)=Y(1,9)+l*sin((pi/2)*(1-n));
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

Xvu=[Xvu X(1,8)];
Yvu=[Yvu Y(1,8)];

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
Xf=X;
Yf=Y;
Xmu=[];
Ymu=[];
n=0;
%?ppning i ?vre v?nstra h?rnet
while n<=1
    n=n+0.01;
    Psiof4=(pi/2)*(1-n);
    Psiof6=(pi/2)*n;
    %Psiof8=(pi/2)*(1-n);
    Psiof2=(pi/2)*n;

Psi(1,7)=Psiof7+(pi/2)*(2);
Psi(1,6)=Psiof7+Psiof6+(pi/2)*(2);
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*(2);
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*(2);
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*(2);
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*(2);
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*(2);

X(1,9)=Xvu(1,round((n*100)))+l;
X(1,8)=X(1,9)+l;
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,9)=Yvu(1,round((n*100)));
Y(1,8)=Y(1,9);
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;
```

```
Xmu=[Xmu X(1,8)];
Ymu=[Ymu Y(1,8)];

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
Xmhu=[];
Ymhu=[];
Xlat=[];
Ylat=[];
Xf=X;
Yf=Y;
n=0;
%mitten-bak uppe
while n<=1
    n=n+0.01;
    Psiof5=(pi/2)*(1-n);
    Psiof7=(pi/2)*n;
    Psiof1=(pi/2)*(1-n);
    Psiof3=(pi/2)*n;

Psi(1,7)=Psiof7+(pi/2)*(2);
Psi(1,6)=Psiof7+Psiof6+(pi/2)*(2);
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*(2);
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*(2);
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*(2);
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*(2);
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*(2);

X(1,9)=Xmu(1,round((n*100)))+l;
X(1,8)=X(1,9)+l;
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,9)=Ymu(1,round((n*100)));
Y(1,8)=Y(1,9);
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

Xmhu=[Xmhu X(1,8)];
Ymhu=[Ymhu Y(1,8)];
Xlat=[Xlat X(1,7)];
Ylat=[Ylat Y(1,7)];

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
```

```
end


Xf=X;
Yf=Y;
n=0;
%mitten-fram uppe
while n<=1
    n=n+0.01;
    Psiof6=(pi/2)*(1-n);
    %Psiof8=(pi/2)*n;
    Psiof2=(pi/2)*(1-n);
    Psiof4=(pi/2)*n;

Psi(1,7)=Psiof7+(pi/2)*(2+n);
Psi(1,6)=Psiof7+Psiof6+(pi/2)*(2+n);
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*(2+n);
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*(2+n);
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*(2+n);
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*(2+n);
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*(2+n);

X(1,9)=Xmhu(1,round((n*100)))+l;
X(1,8)=Xlat(1,round((n*100)))+l;
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,9)=Ymhu(1,round((n*100)));
Y(1,8)=Ylat(1,round((n*100)));
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end

Xf=X;
Yf=Y;
n=0;
%?ppning h?gra h?rnet uppe
while n<=1
    n=n+0.01;
    Psiof7=(pi/2)*(1-n);
    Psiof1=(pi/2)*n;
    Psiof3=(pi/2)*(1-n);
    Psiof5=(pi/2)*n;

Psi(1,7)=Psiof7+(pi/2)*(3+n);
```

```
Psi(1,6)=Psiof7+Psiof6+(pi/2)*(3+n);
Psi(1,5)=Psiof6+Psiof7+Psiof5+(pi/2)*(3+n);
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4+(pi/2)*(3+n);
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3+(pi/2)*(3+n);
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2+(pi/2)*(3+n);
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1+(pi/2)*(3+n);

X(1,9)=X(1,8)+l*cos((pi/2)*(1-n));
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,9)=Y(1,8)+l*sin((pi/2)*(1-n));
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',X(1,9),Y(1,9),'o',Xmedel,Ymedel,'b*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
end
%steg3 tillbaka
n=0;
while n<=1
    n=n+0.01;
    Psiof1=(pi/2)*(1-n);
    Psiof3=(pi/2)*n;
    Psiof5=(pi/2)*(1-n);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
```

```
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg2 tillbaka
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*n;
    Psiof4=(pi/2)*(1-n);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg1 tillbaka
n=0;
while n<=1
    n=n+0.01;
    Psiof2=(pi/2)*(1-n);
    Psiof3=(pi/2)*(1-n);

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;
```

```
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%%
clc
cla
l=7; %l?ngden p? segment
speed=10000000; %hastighet p? simuleringen
puls=3; %antal pulser som ska utf?ras
X=[0, l, 2*l, 3*l, 4*l, 5*l, 6*l, 7*l, 8*l];
Y=[0, 0, 0, 0, 0, 0, 0, 0, 0];
Psi=[0, 0, 0, 0, 0, 0, 0];
Psiof7=0; Psiof6=0; Psiof5=0; Psiof4=0; Psiof3=0; Psiof2=0; Psiof1=0;

%steg1
n=0;
while n<=1
    n=n+0.01;
    Psiof3=((2*pi)/3)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
```

```
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg2
n=0;
while n<=1
    n=n+0.01;
    Psiof3=((2*pi)/3)*(1-n);
    Psiof4=((2*pi)/3)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg3
n=0;
while n<=1
    n=n+0.01;
    Psiof4=((2*pi)/3)*(1-n);
    Psiof5=((2*pi)/3)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;
```

```
X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg4
n=0;
while n<=1
    n=n+0.01;
    Psiof5=((2*pi)/3)*(1-n);
    Psiof6=((2*pi)/3)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
```

```
%steg5
n=0;
while n<=1
    n=n+0.01;
    Psiof6=((2*pi)/3)*(1-n);
    Psiof7=((7*pi)/12)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));

Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
%steg6
n=0;
while n<=1
    n=n+0.01;
    Psiof4=((pi)/2)*n;
    Psiof7=((7*pi)/12)*(1-n)+(pi/2)*n;

Psi(1,7)=Psiof7;
Psi(1,6)=Psiof7+Psiof6;
Psi(1,5)=Psiof6+Psiof7+Psiof5;
Psi(1,4)=Psiof5+Psiof6+Psiof7+Psiof4;
Psi(1,3)=Psiof4+Psiof5+Psiof6+Psiof7+Psiof3;
Psi(1,2)=Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof2;
Psi(1,1)=Psiof2+Psiof3+Psiof4+Psiof5+Psiof6+Psiof7+Psiof1;

X(1,7)=X(1,8)-l*cos(Psi(1,7));
X(1,6)=X(1,7)-l*cos(Psi(1,6));
X(1,5)=X(1,6)-l*cos(Psi(1,5));
X(1,4)=X(1,5)-l*cos(Psi(1,4));
X(1,3)=X(1,4)-l*cos(Psi(1,3));
X(1,2)=X(1,3)-l*cos(Psi(1,2));
X(1,1)=X(1,2)-l*cos(Psi(1,1));
```

```
Y(1,7)=Y(1,8)+l*sin(Psi(1,7));
Y(1,6)=Y(1,7)+l*sin(Psi(1,6));
Y(1,5)=Y(1,6)+l*sin(Psi(1,5));
Y(1,4)=Y(1,5)+l*sin(Psi(1,4));
Y(1,3)=Y(1,4)+l*sin(Psi(1,3));
Y(1,2)=Y(1,3)+l*sin(Psi(1,2));
Y(1,1)=Y(1,2)+l*sin(Psi(1,1));

Xmedel=(X(1,1)/2+X(1,2)+X(1,3)+X(1,4)+X(1,5)+X(1,6)+X(1,7)+X(1,8)+X(1,9)/2)/8;
Ymedel=(Y(1,1)/2+Y(1,2)+Y(1,3)+Y(1,4)+Y(1,5)+Y(1,6)+Y(1,7)+Y(1,8)+Y(1,9)/2)/8;

plot(X,Y,'r.',X,Y,'k',Xmedel,Ymedel,'*')
axis([-1 10*l+8*l*puls -1 10*l+8*l*puls])
pause(1/speed)
end
```

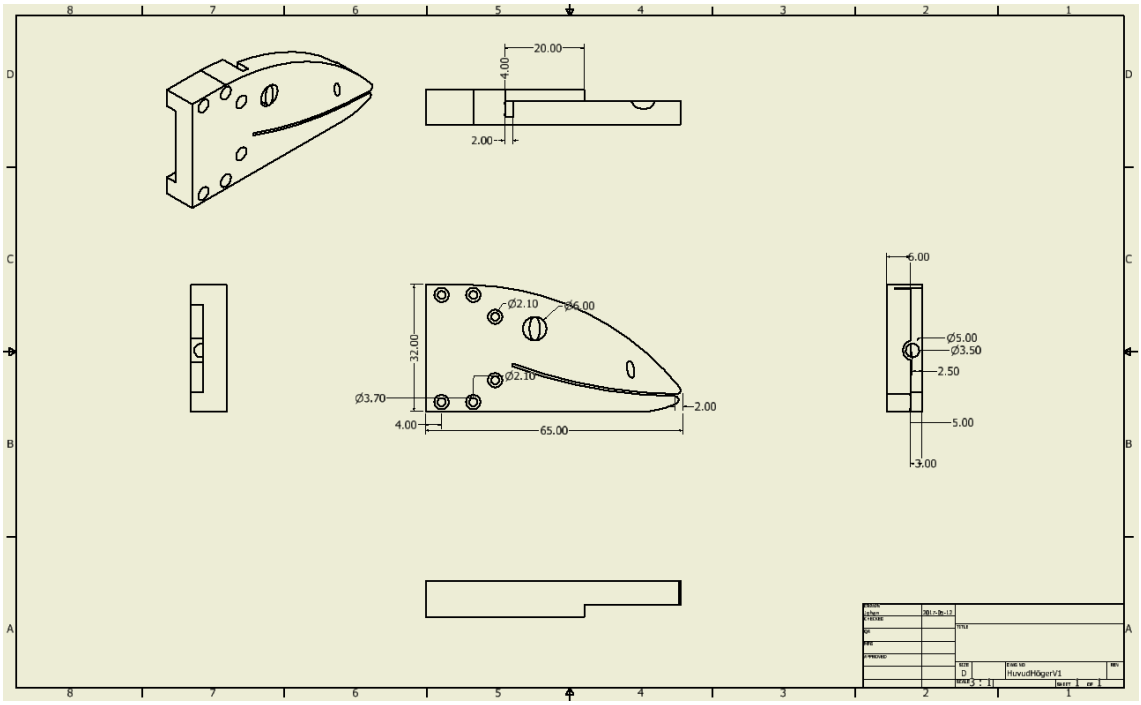


Figure A.1: Illustration of the head.

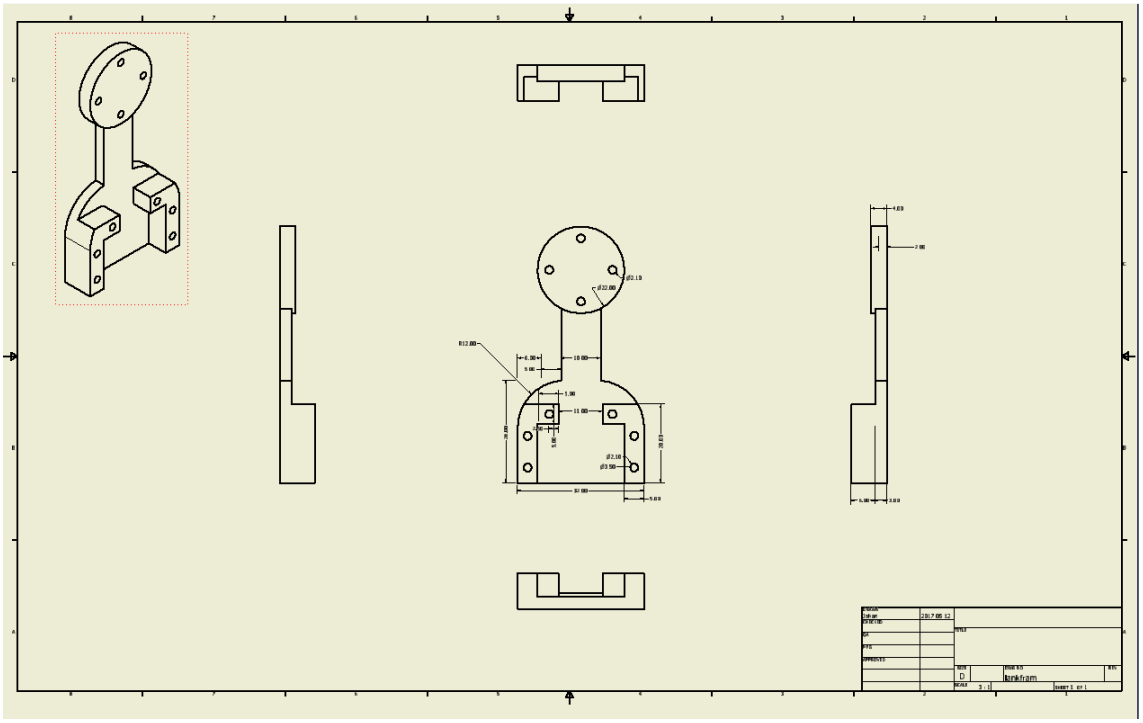


Figure A.2: Illustration of the head.

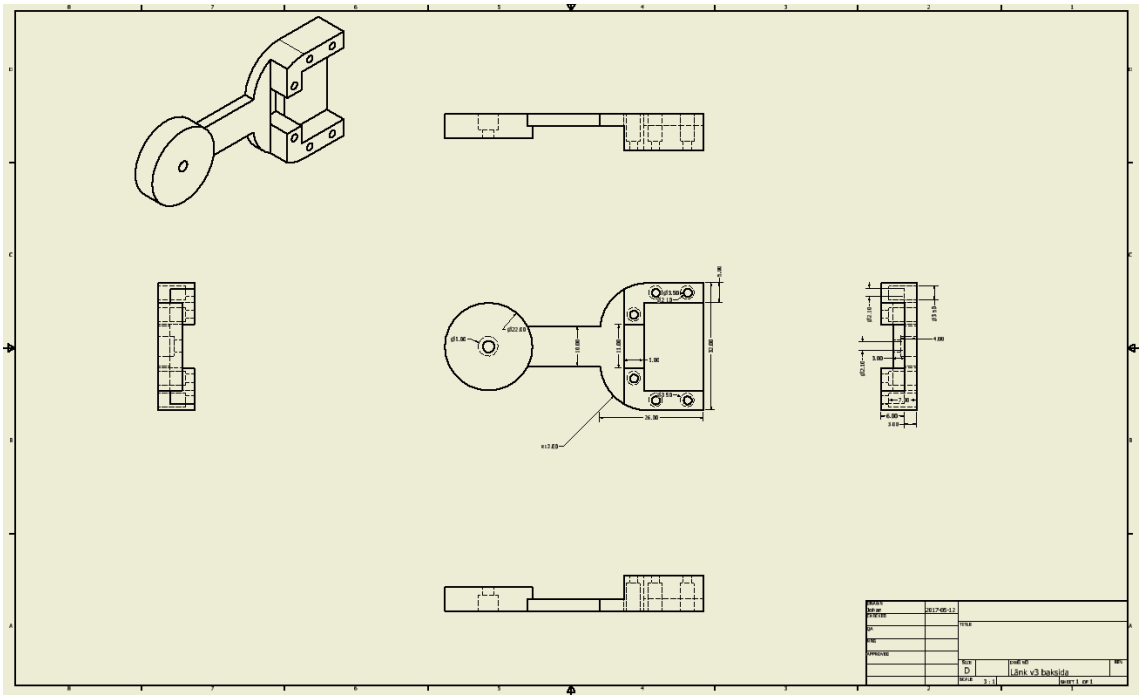


Figure A.3: Illustration of the head.

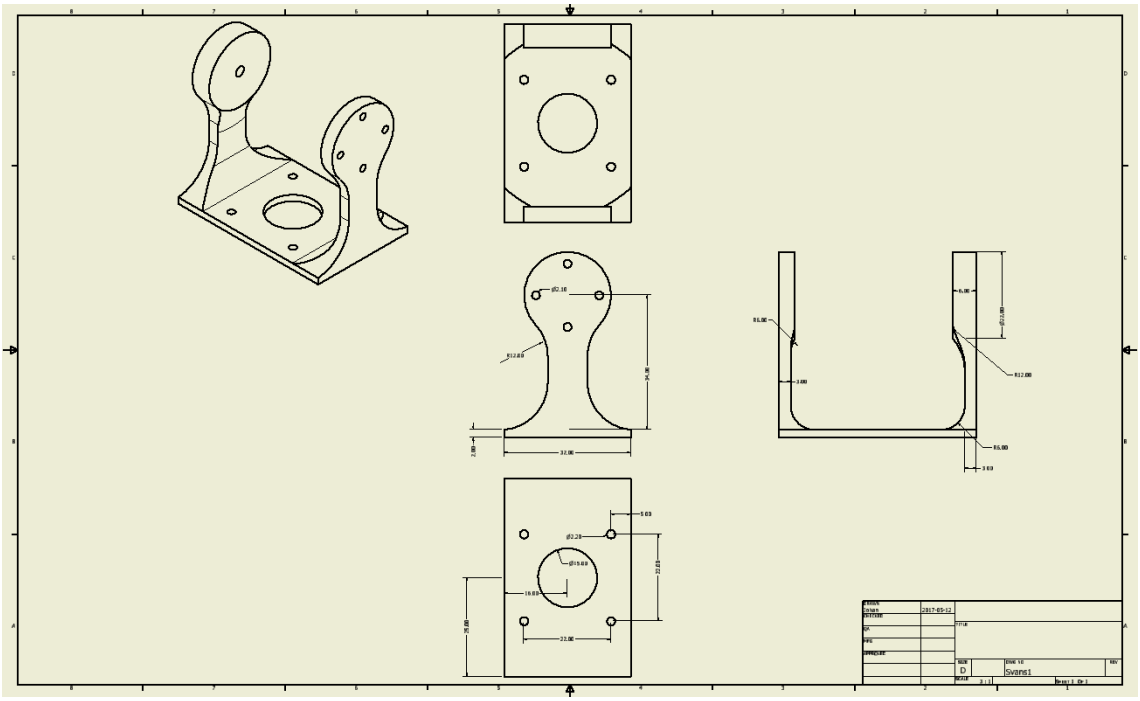


Figure A.4: Illustration of the head.

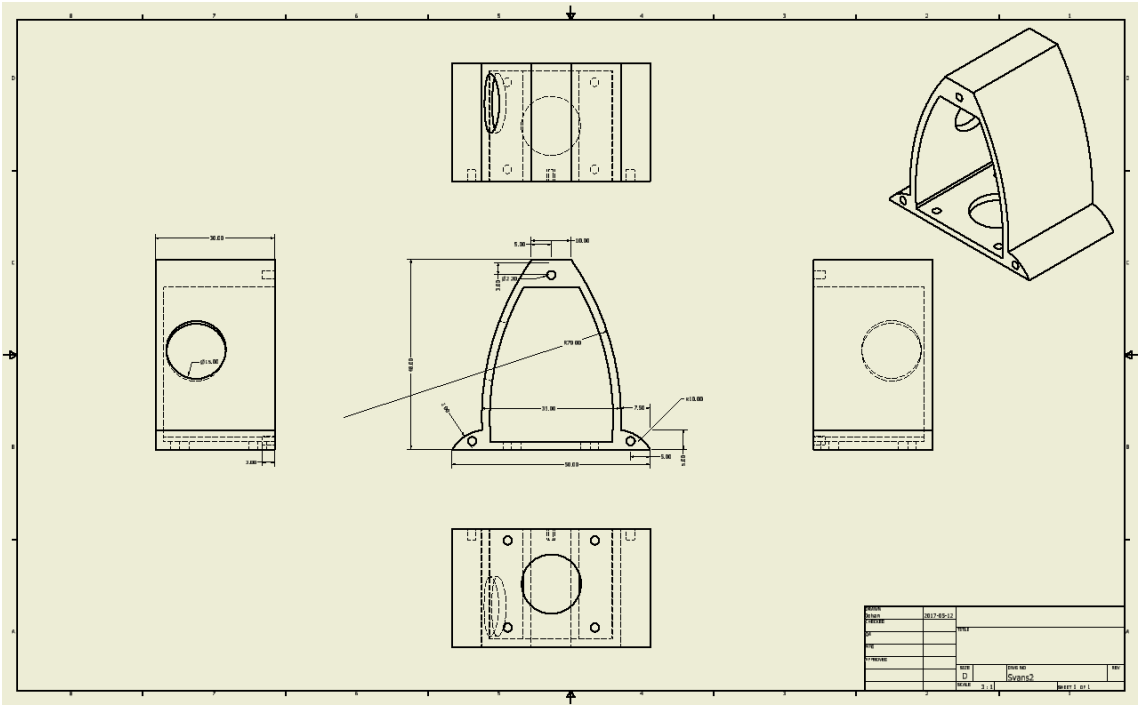


Figure A.5: Illustration of the head.