Multiped Robot

4GA40, Multiped Robot
Q3 (2022-2023)

Group 028

Full Name	**Student ID**
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
Tobias Wejbora	[redacted]

Tutor: Hodzelmans, Maarten
Project Coordinators: Dr. Ye Wang - Dr. Ir. Yoeri van de Burgt

Eindhoven, March 23, 2026

Contents

1 | List of symbols

Table 1.1: list of symbols

Symbol	Variable	Unit	Unit abbreviation
r	horizontal distance	meter	m
d	vertical distance	meter	m
α	angle between the rotation axis	degrees	$^\circ$
θ	variable angle	degrees	$^\circ$

2 | Introduction

The main goal of this CBL project is to design, program, and build a multiped robot. This robot should be able to go around parkour where it has to climb a slope, turn and go over some gaps as fast as possible. The parts that have been designed and 3D printed and laser-cut. All of the designed parts have been created in Siemens NX. More information about the materials and the rules mentioned later in this report.

There are three sections to this report. Conceptual design, working mechanism, and testing. In the first part, the brainstorming, RPC list, and conceptual designs have been described. Details of the design, how the robot behaves and construction have been explained in the working mechanism section. In the testing chapter the comparison between the robot movements and the Matlab model, testing of the robot, and further improvements have been discussed.

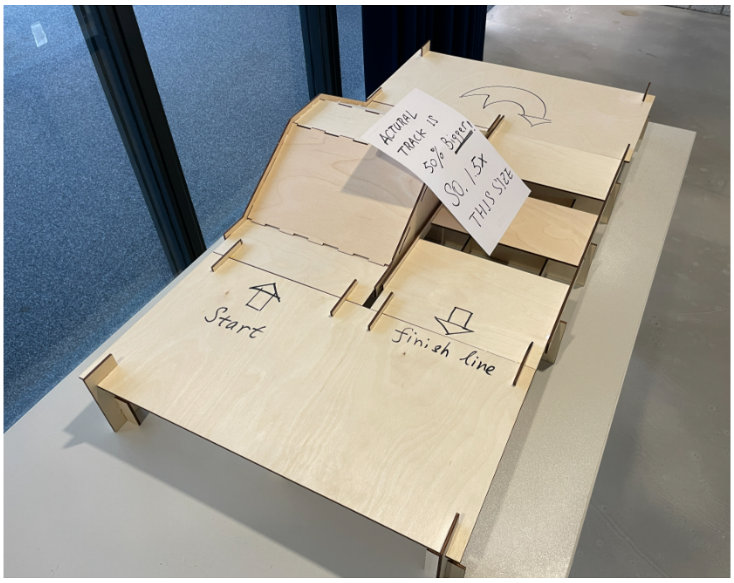


Figure 2.1: Scaled version of the parkour (real one is 50 percent bigger)

It is important to note that the standard parkour has a slope of 20 degrees, however, it is possible to get a higher grade if the robot can also climb a slope of 30 degrees.

At first, an RPC(References-Preferences-Constraints) list was created with the ideas of the group members. Ideas for the robot were made and discussed using this RPC list. Multiple concepts for the robot have been worked on to finalize the selection. With each of these designs, there was a different manner to go from the incline or pass from the gap. The pros and cons of each of the designs have been discussed and you will find them in the 'Conceptual designs' part. The final design ended up being a hybrid between two concepts that were made during the conceptualizing phase.

For this project, all members have to use and learn multiple skills that are required for this project. For example, all members have to learn to use Matlab for the movement (which will be explained in more detail), which is not only useful for this project but also for projects that will follow. Also, skills like using Siemens NX for designing parts of the robot are required.

3 | Concept Design and Evaluation

The RPC list can be seen as a roadmap for this project. The conceptual designs and the final design have been made based on this RPC list.

3.1 | The RPC List

Requirements:

- The robot should climb the slope
- The robot should make a turn
- The robot should cross gaps of 7.5 cm

Preferences:

- The robot should be as quick as possible
- The robot should move without slipping
- The robot should turn corners efficiently
- The robot should have a tidy wire management
- The robot should not be hard-coded
- The robot should be able to move in more than 2 directions
- The robot should have a low center of mass
- Most of the longitudinal weight should be just ahead of the center of the symmetry

Constraints:

- The robot must be coded using an Arduino UNO
- The MG996r Servo motors can only rotate 180 degrees
- The wheels must avoid a continuous rolling contact
- The internal structure of the servos must not be altered

3.2 | Conceptual Designs

After finishing the RPC list the concepting process began. In this brainstorming phase, multiple concepts were created and sketched. These were evaluated using the RPC list.
Since there is a lot of freedom in this project there are a lot of varying concepts. Each with different ways to move and with different pros and cons. These will be discussed in this section of the report.

Caterpillar/Centipede Design

This concept was inspired by looking at the movement of certain insects, like caterpillars and centipedes. By making a long and flexible body, the robot could potentially keep moving while turning. It could also be advantageous for crossing the gaps.

This idea was scrapped early on since it would be hard to make parts of the body flexible and move correctly. Also, due to its length, the robot would require a lot of space in the track and have a wide turning radius. Hence, there wouldn't be much room for mobility.

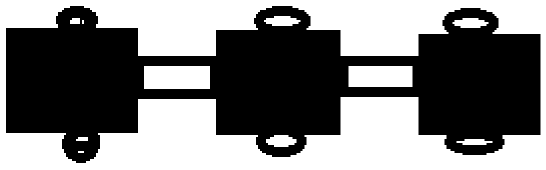


Figure 3.1: Conceptual Design: Caterpillar top view

Table 3.1: Advantages, Disadvantages

Advantages	Disadvantages
Can move while turning	Large turning radius
Can cross the gaps	Takes up a lot of space
Very difficult to make it flexible (and strong)	

Spider Design

The spider design consists of a platform with multiple legs attached to it in the shape of a spider. Multiple versions of this concept were made. In the beginning, 6 legs were considered, however, this would require a large number of servomotors (at least 12). This could be a disadvantage in terms of programming, weight, and weight distribution.

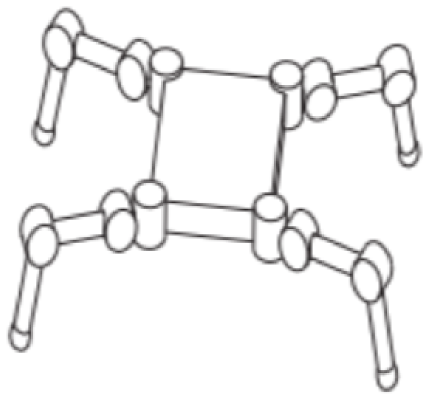


Figure 3.2: Conceputal Design: 4 Leg Spider Design

The spider robot will move as follows. First, one of the legs will raise up using a leg servo. Then the leg will rotate forward and move down. Then another leg will follow this procedure. This design can move very quickly and rotate easily. There are a few downsides to it. It is quite complex to code since there are four legs each with three servos and it is also hard to step over the gaps in the parkour.

Table 3.2: Advantages, Disadvantages, and Influences for the final design

Advantages	Disadvantages	Influence
Can turn easily	Can be harder to program	4 legs were used
Can climb up the slope	Can be harder to cross the gaps	The platform is very similar

Table Design

This concept was made in order to easily climb the slope and easily cross the gaps. As can be seen in the figure, it consists of a platform with 4 static legs like a table. Also, in the middle, there are legs that can move the table forward.

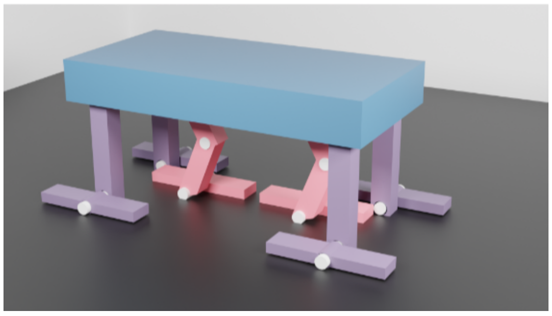


Figure 3.3: Conceptual Design: Table design, purple legs are stationary and pink legs are mobile

The stable legs have a free axis at the bottom so they can position themselves by the slope. The moving legs go back and forth dragging the robot forward. The first problem with this concept is its speed. The robot will not be nearly as quick as we want it to be because of the fact that there are only 2 legs moving the robot.

The second problem is turning. Because with this movement 'style' it is impossible to make a turn. So if this concept would be chosen, some big changes would be necessary.

Table 3.3: Advantages, Disadvantages, and Influences for the final design

Advantages	Disadvantages	Influence
Can climb up the slope	Hard to turn	Nothing
Can cross the gaps		

Wheelbarrow Design

This concept was made after having a good look at the rules of this project concerning wheels. The idea behind it was that the problems of turning and crossing the gaps could be solved by adding a big wheel (or multiple small wheels) under the robot that could be controlled.

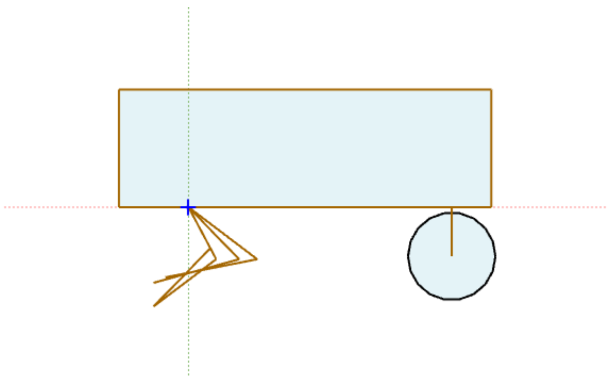


Figure 3.4: Wheelbarrow

The wheelbarrow design has two dynamic rear legs and one guiding wheel at the front to act as a sort of wheelbarrow. The use of a wheel allows for fast turn-around corners and good stability. However, a major concern regarding this design is how well the leg movements can be manufactured. Since there

are only two legs, the robot can be vulnerable to off-centering its center of gravity. Including more 2 more legs were considered which improved the situation, but this doesn't stop the legs from getting stuck in the gaps. Giving the legs a wide base was also taken into consideration, but this doesn't change the fact that the front wheel would still get stuck.
The wheel itself would not be powered since that would be against the rules. However, by adding legs behind or on the sides of the robot that could push the robot, would not be necessary.

Table 3.4: Advantages, Disadvantages, and Influences for the final design

Advantages	Disadvantages	Influence
It can turn easily	Climbing the slope	Wheels were used
It can cross the gaps		

3.3 | First Prototype

The final concept is a spider design with the influence of the wheelbarrow design. The design will have four legs which all have two servos. These will be used to move forward. Two servos are attached on the top which controls the two arms with wheels. The purpose of these arms with wheels is to support the robot and create more stable and fluid movements while crossing the gaps.

This concept was chosen since it incorporates the advantages of the spider and wheelbarrow design while removing their limitations. Going over the gaps was a big problem with the spider design. With these wheels present to stabilize, it will be simpler to go over them. With the wheelbarrow design, it was considered that it could be difficult to go up the hill. Now, since the wheels are retractable and the spider is very quick to move uphill it will not be an issue. The robot will be 3d printed except for the frame/baseplate and the two-wheel arms. As seen in Figure(3.5), the robot moves with 4 legs, each leg having two servos. The servo attached to the body/frame of the robot controls the side-to-side movement while the servos in the middle of the leg control the height. The heigh servo may be nenoted as the thigh of the leg. Furthermore, in the center of the robot, a mount has been made for the arduino.

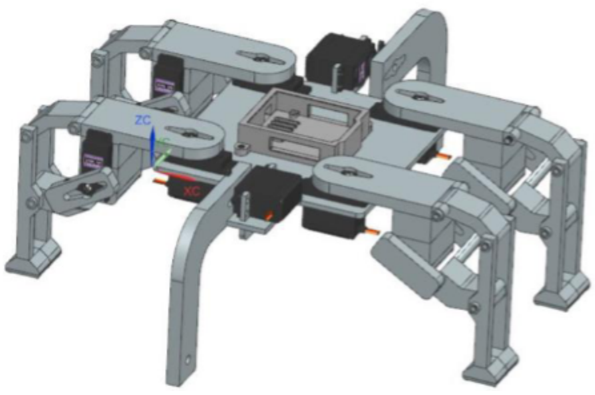


Figure 3.5: First Prototype (wheels on the arms are missing)

4 | Working Mechanism and Evaluation

In this part of the report, all individual parts of the final design will be discussed. Also, the mechanisms and the movement of the robot will be explained in further detail.

4.1 | Detailing of the Design

After testing the first prototype, major changes were proposed. Most notably, these changes were introduced in order to improve the stability and functionality of the robot.

Detailing the leg

By analyzing the design of the leg in Figure (4.1), a weak point was found where the leg attaches to the frame of the body. This would cause issues like bending as well as cause a higher chance of inaccuracies while moving. Thus, the best way to combat this issue is to either connect another point where the leg and the frame connect or to create a bracket that connects the legs together. It was determined that connecting the legs to each other is more efficient due to the position of the servo in the frame. Figure(4.2) shows the updated thigh of the leg. It features a lower connection point on the lower right point of Figure(4.2) where a bracket is connected to. This bracket will be used to connect all the legs together. This connection point works by allowing an M3 nut to slot in the hexagonal hole. Then the bracket can be screwed to this M3 nut. Note that the design of the bracket will be discussed in section(4.1). Lastly, a slit was made on the bottom of the part to allow for the servo wires to slide in.

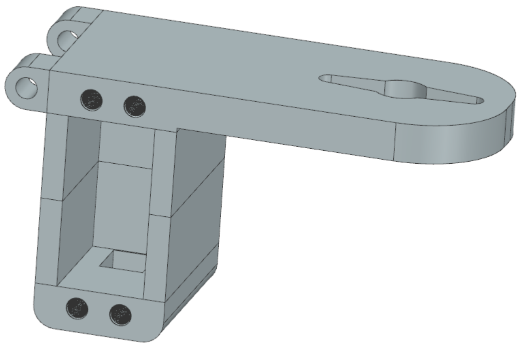


Figure 4.1: Leg Prototype

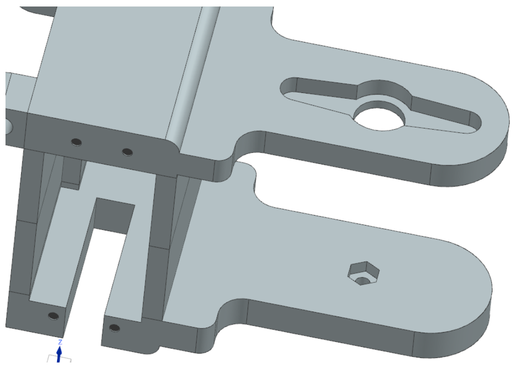
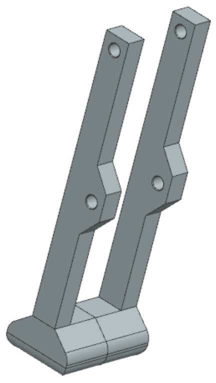


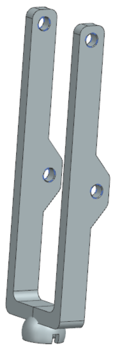
Figure 4.2: Updated Leg

Furthermore, the bottom of the feet in Figure(4.3a) have been modified to have a ball and socket joint with a bigger surface area on the foot. This design change has been implemented to help the feet continuously have full surface contact while traversing the hill. The ball and socket joint is made up of two parts, the ball seen in Figure(4.3c) and the socket(Figure(4.3b).

Detailing the frame

The state of the frame in Figure(4.4) has a lot of unnecessary space which ultimately increases the size of the robot. Creating a smaller frame would make the robot more agile and allow for more leeway on the course and put more of the robot's weight in the center. The biggest issue encountered while decreasing the size was the placement of the wheel arm servos. These had to be relocated into the center of the robot. In order to do so, the Arduino bracket was removed and a new mount was for the servos. This mount will also be featured in Section(4.1). By making the frame smaller, the robot decreased four centimeters in length and width.


(a) Leg Prototype


(b) Ball and Socket Joint Leg

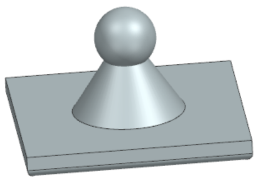

(c) Ball and Socket Foot

Figure 4.3: Leg parts

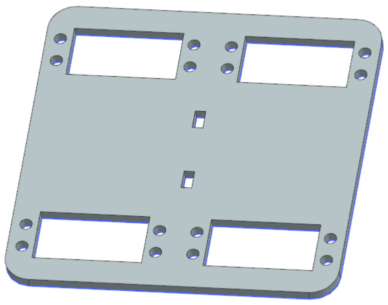


Figure 4.4: Frame Prototype

Detailing the wheel arms

As discussed in Section(4.1), decreasing the size of the frame means that the wheel arms have to be moved to the center of the robot. Therefore, the wheel arms themselves have to be made longer to account for this. The updated wheel arms are found in Figure(4.5) and have been made 10cm longer in both the x and y directions.

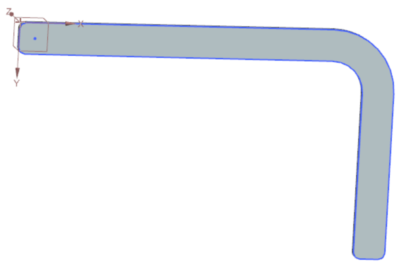


Figure 4.5: Wheel Arm

Detailing newly designed parts

Leg Connector Bracket: The Leg connector(Figure(4.6)) bracket connects all the legs together so that they don't buckle. The bracket does not include any holes for the reason that the location of these holes is not known. The real-life robot will have inaccuracies during assembly and thus it is more reliable to use a drill to create custom holes in the right position.

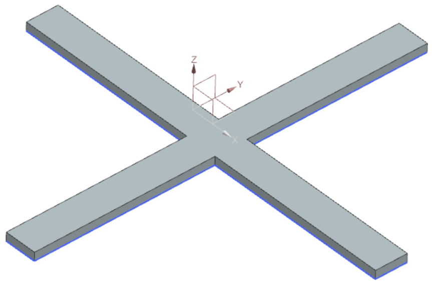
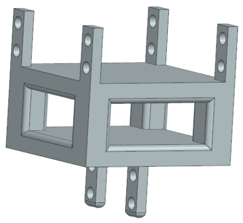
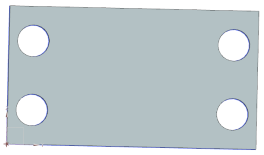


Figure 4.6: Leg Connector

Wheel Arm Servo Mount: A bracket for the wheel arm servos is required due to the fact that the servos are now in the middle of the frame. Furthermore, the servos have to be elevated above the height of the other servos in the frame. In order to meet this requirement, a mount was made. The mount, as seen in Figure(4.7a) has multiple features that lock the servos in place and the actual mount into the frame. Firstly. the servos will be attached at the top of the mount as seen in Figure(**??**) and the mount will be inserted into the frame with the bottom two prongs. The amount will be secured into the frame with a lock(Figure(4.7b)). The wheel arm mount can be inserted into the frame and the lock will be attached to the wheel arm mount on the underside of the frame with screws, as seen in Figure(**??**).

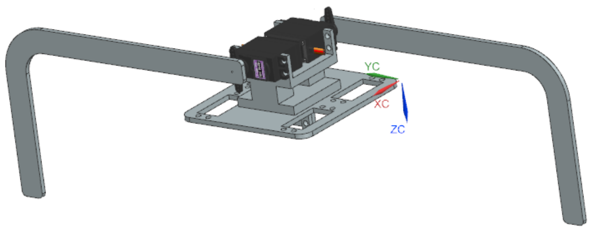


(a) Wheel Arm Mount



(b) Lock for Wheel Arm mount

Figure 4.7: Wheel Arm Servo Mount



(a) Wheel Arm Assembly

Wheel Arm, Wheels and Axle: The wheels which have a diameter of 25mm can be found in Figure(4.9a). This diameter combines the benefits of size and stability. The axle(Figure(4.9b)) for the wheels is 10mm in diameter which gives a large enough margin of error regarding strength.

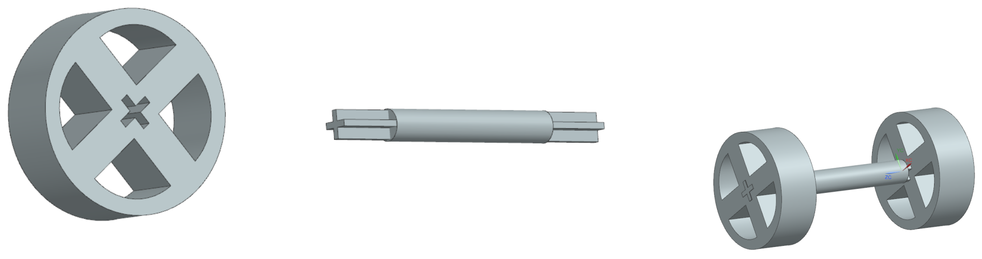

(a) Wheel (b) Axle (c) Wheel Assembly

Figure 4.9: Wheel Arm Wheels and Axle

Wheel Arm Axle Support: Due to the fact that the wheel arms are made out of plywood, they are only 4mm thick. Thus the wheel and axle will be flimsy and unstable. To counteract this, wheel arm supports can be made, as seen in Figure(4.10). This part can simply be glued onto the wheel arm.

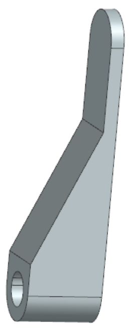


Figure 4.10: Wheel Arm Axle Support

Final Design

Through the detailing process, a fully functional robot has been designed. Combining the various changes, the final design is displayed in Figure(4.11).

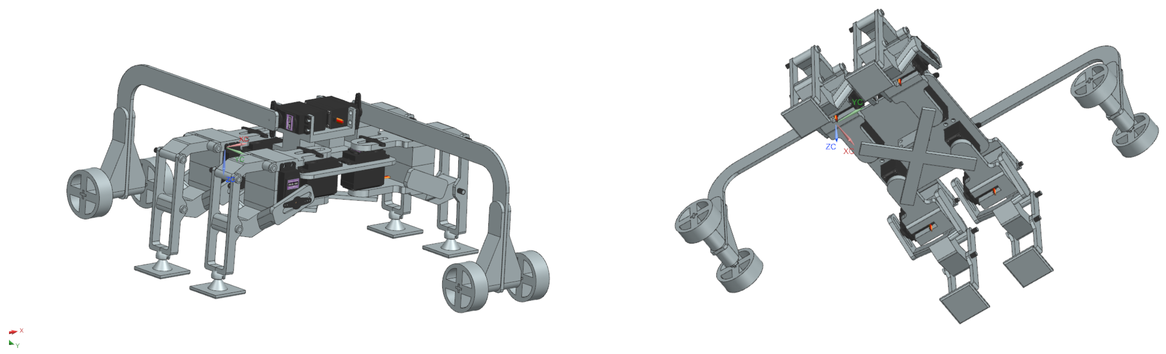

(a) Final Assembly Isometric (b) Final Assembly Underside

Figure 4.11: Final Assembly

4.2 | The Robot Behaviour

In this subsection behavior of the robot will be explained. So not only how it moves but it will also be explained how it was designed and programmed for this movement.

Anatomy

Correct naming of variables is an important part of the programming, it doesn't only keep the code cleaner/neater but also helps in debugging. For the naming, camel case is used and the naming of the variable for the servos is as follows:

- The servos on the front/back have the prefix front/back
- The servos on the right/left have the suffix right/left
- The servos which contribute to lifting the legs have another suffix Jnt
- The arms are simply front arm and rear Arm

Movement

Since the chosen design is a quadruplet, this gives the freedom to choose from many different kinds of movements, it could be a stroke, trot-gait, creep-gait, etc. Each movement has its pros and cons depending on the situation, the movement which is a perfect balance of speed, stability, freedom of movement, and simplicity according to the needs is the creep-gait.

Creep-gait: Creep-gait is the type of movement that has been derived from nature. The main principle of creep gait is that at all times, at least 3 legs should be on the ground, and the robot's center of mass should fall under the area made by the three legs. This gives the robot the stability no other movement gives, however, there is a big trade-off in speed.

Moving in a Particular direction: Although the same method can be used for any direction, for the purpose of explaining moving in the front will be focused. First, the robot has to get into the initial position which is, the legs on the right side being perpendicular to the side of the robot and the legs on the left being on 45 degrees and 135 degrees respectively. After getting into the initial position, the front right leg would be raised and rotated to 135 degrees, while it is in the air over 135 degrees, the front left and back right legs rotate to 90 degrees and 45 degrees, which pushes the robot in the front. This is then followed by the back left leg positioning perpendicular to the side of the robot. The position of the robot now is just a reflection of the initial position, and the movement is just reflected.

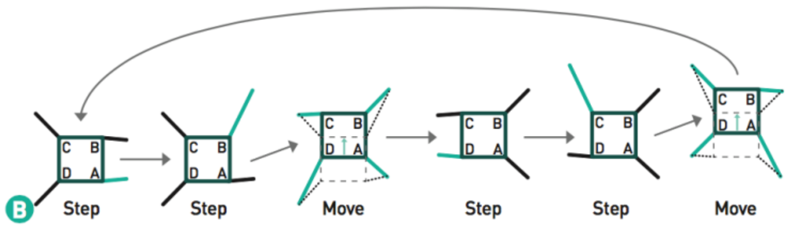


Figure 4.12: Representation of a creep-gait
[1]

Rotation: The rotation for the robot was not really required because after getting to a turn the robot can simply get into the position of going left and move left like a spider. The rotation was rather needed to correct the path of the robot if it dealigns. The way it is done is by making all the legs lift up one by one and turning 45 degrees clockwise/anti-clockwise and after they touch down, the legs turn back to 45 degrees/135 degrees making the whole body rotate.

Programmming

The code for the robot is written in C++ derived Arduino programming language. The purpose of writing a program is to make it useable in as many situations as possible or can be said to not hardcode. The difference between hardcode and softcode is that hardcoding only works in a specific situation for which the code is written, and if it is uncertain in the situation then the code just collapses. Softcoding is when the program adapts itself to the external condition by using sensors, processors, or user input. The whole program is softcoded which makes it much more practical and can be adapted to work in real-life conditions.

- positionIn() : all the legs go to 45/135 degrees
- rehearse() : all the joints are tested
- frontPos(), rightPos(), backPos(), leftPos() : the robot gets into corresponding position
- moveFront(), moveRight, moveBack(), moveLeft() : corresponding movements for the direction
- frontUp(), frontDown(), rearUp(), rearDown(), frontDownMore(), rearDownMore() : these functions are used to control the arms of the robot, functions ending with More just makes the arms go down by a larger angle
- rotateClock(), rotateAntiClock(): the robot rotates clockwise or counterclockwise at a single point

The robot is controlled using serial communication, and for that, a Python script is used. The Python script reads the keystrokes using the keyboard module and then communicates the pressed key using a serial bus, which is then interpreted in the Arduino code. After the keystroke is read in the Arduino code, the functions are called in an if()else if()else statement, initially a switch statement was used but due to some reason it did not work as expected. The only difference between the switch case and the if-else case in this situation was that the switch case made the code look neater.

4.3 | Construction and Troubleshooting

In this subsection, the problems encountered during the building sessions will be discussed.

Hardware

For the prototype, it was decided beforehand to use pins that would be printed out. However because some of the holes in the prototype were too small, it was decided to partly use pins and partly use screws.

The bottom of the prototype was also not connected anywhere, so the entire robot was connected by just one baseplate. The result of this was that the baseplate would bend a little bit which made the movement a lot less efficient.

Another problem was that the robot did not have enough grip to climb the slope, the first solution was attaching sandpaper to the feet. However, after trying multiple ideas the best solution was decided to be adding rubber bands to the feet.

Software

While the robot was going on the flat surface, its leg would hit the side walls and small things coming out of the surface and the robot would be dealigned. To solve that, the robot's resting height would be increased so it is less wide, this increased the center of mass of the robot and the robot would topple back. To counter this, the best bet was to write the code in such a way that the robot's resting height would decrease when it tries to go uphill, and then the height would increase again when the robot is going on flat surfaces.

5 | Testing and Evaluation

5.1 | Tests and Analysis

Testing began at the first prototype stage. The CAD model discussed in Figure(3.5) was printed and the changes made in the detailing part of the report emerged from this design. In simple terms, after the first tests, the robot did not have enough structural integrity to take accurate steps and it was not able to maintain full surface contact with the course. Therefore, the robot could not go up the ramp consecutively or cross the gaps. The robot was also unnecessarily large and the frame could be made smaller. After these changes, the robot gained better traction going up the ramp, however, crossing the gaps was still an issue. The use of the wheel arms did not work as well as it was theoretically intended. The reason behind this can be narrowed down to the limitations of the servos, the robot dimensions, and the unforeseen complex programming required.

5.2 | Further Improvements

Further improvements can always be made to a design. In this case, the biggest improvement is making the grip stronger. Finding a material that better suits the plywood course will improve the accuracy of the leg movements and thus improve mobility and speed. Since the robot was not able to move over the gaps, incorporating smaller steps could help improve the balance and accuracy of the robot.

6 | Matlab Modeling

Matlab is used to compare the model and it is also used to predict the speed of the robot. A model was created with two segments of the leg. This represents our spider leg minus the third segment, which is a translation of the second segment. We will assume that this does not make a difference in the model's predicted angles or in its accuracy in following the desired path. It also assumes that no external forces work on the model.

Steps

1. The first step was to create a DH table. The purpose of the DH table is to represent the joint locations and orientations in a standardized way. There are four parameters that make up a DH table. They are r, θ, α and d. r and d are horizontal and vertical distances between the rotation axes, respectively. α is the angle between the z-axis orientations of two axes compared to the x-axis. Finally, θ is the angle between the x-axis orientations of two axes compared to the z-axis. This coincides with the angle of the servos, so this is what will be modeled.

Origin	r	d	θ_1	α
1	$L_1 * sin(45)$	$L_1 * cos(45)$	θ_1	$1.5 * \pi$
2	L_2	0	θ_2	0

2. The desired path is defined. This path consists of six points the leg must pass to make an efficient forward movement. The model calculates all the leg's reachable points and with this information approximates the trajectory which is shown in figure 6.1. As you can see in figure 6.2, where the orange path is the desired path and the purple points represent the path the leg takes to approximate this.

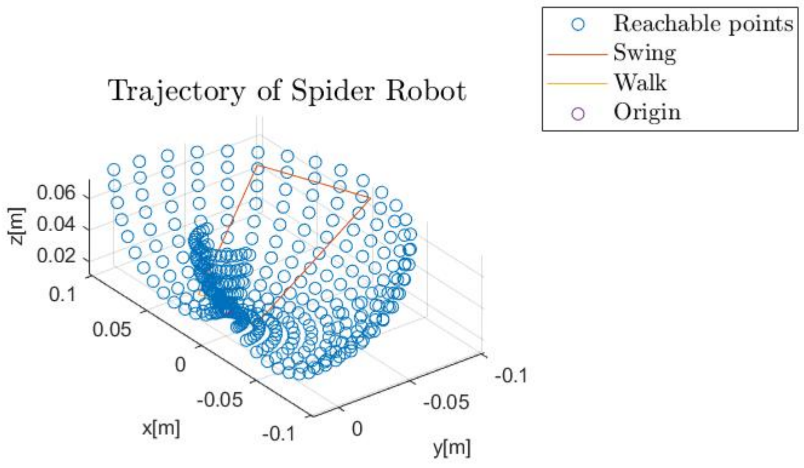


Figure 6.1: Reachable points

3. The last step is to recreate the entire robot. Ours be a square base with four legs. One in each corner. As described in 4.2 the robot will follow a creep gait. Meaning two legs will be in the air while the others move the robot forward. They will have a phase difference of 180. In figure 6.3 a picture of the model can be seen. With the help of this model the speed will be computed in section 3.

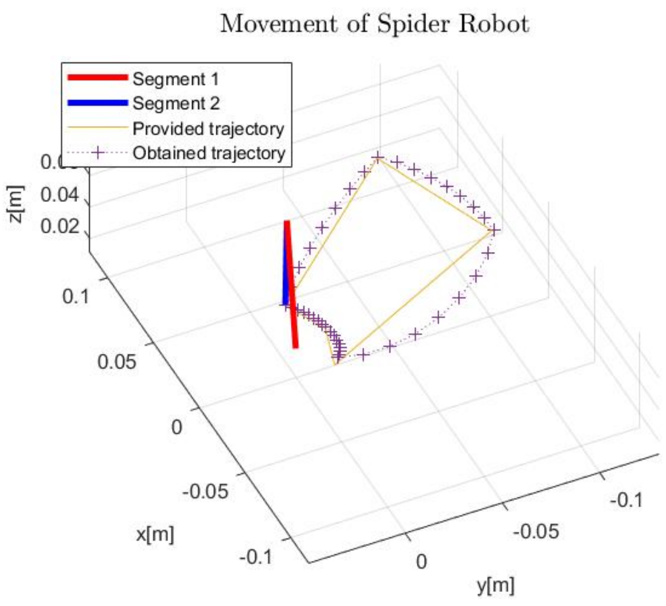


Figure 6.2: Desired trajectory and models approximation

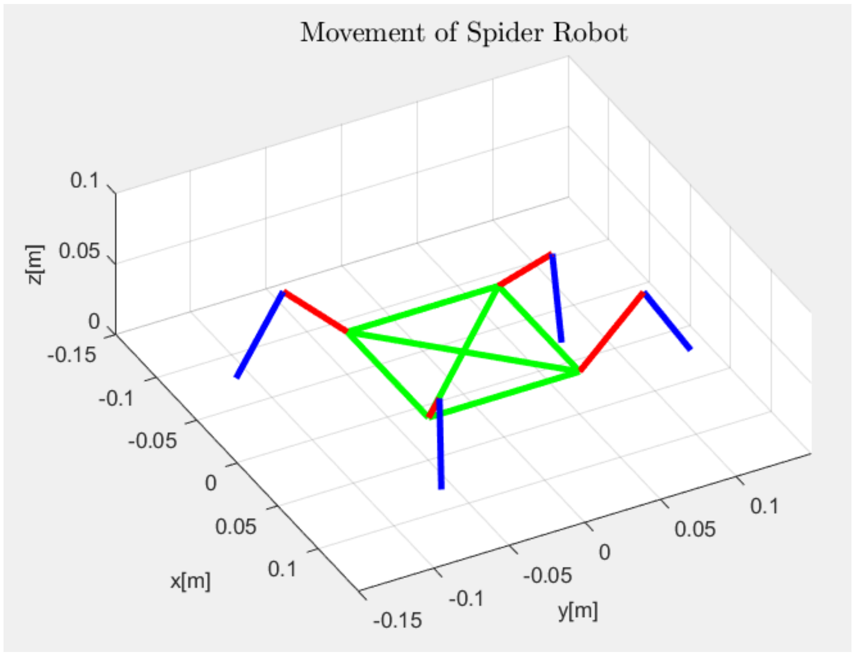


Figure 6.3: Final Matlab model

Calculating the Speed of the Robot

The speed of the MATLAB model can be compared to the real-life model to determine its efficiency and accuracy. Every step a leg made in the model would move the robot 0.055 meters forward and would take 0.5 seconds. Since the legs move in a 180 degrees phase shift it would have a speed of 0.24 meters per second. The speed of the real-life model can be determined by traversing the robot over a specific distance over the time it took. This method provided a speed of 0.2 meters per second. Thus, the model is faster. However, as stated earlier, the model does not account for any external forces. This means that the real-life model would be faster in practice.

7 | Conclusion

To summarise, in the beginning, several conceptual designs were made. An RPC list has been created to narrow down the number of concepts. During the designing phase, several criteria for the design had to be considered, such as: how to go as fast as possible, how to make a robot that can go more than 1 direction, and how to design a robot with a low center of mass. Regarding these criteria, 4 concept designs were made. Finally, it was decided to combine two concept designs and make a preliminary design that combined the Spider and support wheels.

Luckily, with the help of one member, it was possible to 3D print and test most of the parts by themselves. After the design, one critical problem had been found: the design was too wide for the track. This problem is solved by printing a smaller frame however this caused other problems for the wheel arm. To solve this problem a higher setup for the wheel arm was designed.

With the tests done and improvements made during the building sessions, a final design has been created. The design consists of 3 main components: The frame, the legs, and the wheel arms. The leg consists of 3 parts. These are the foot with ball and socket joints, the basket for the servo motors, and the joints. The frame has four spaces for servo motors and two holes for the wheel arm. The wheel arms are responsible for stabilizing the whole robot on the incline, also help while passing through the gaps. The wheel arms are connected to a servo motor, which helps them go down and up.

After the theoretical parts and the designing phase, a Matlab model of the design was created to estimate the robot's speed. With the help of Matlab, the design was improved for the final time. After modeling the movement in Matlab, the model got approximately (the speed of the model) m/s but in real life, the speed was (0.2 m/s) in a straight line. This difference is mainly caused by air drag and friction in the robot.

Discussion

In this part of the report, it will be discussed what possible improvements could have been made.

For this project, it was important to learn new skills such as making parts of the design with laser cutting and using Matlab for example. Even though the robot has performed to the expectations, there are still some things that would have been done differently if the project would have to be done again.

One of the first things that would have been done differently is making the holes in some of the parts bigger so that pins could have been used. This could have saved some valuable time in the building process.

The next thing that would have been done differently is that more parts (especially at the beginning of the project) should have been laser-cut instead of 3D printed. After making some small changes to the prototype however this problem became less important and by laser cutting more parts, there was less of a time loss on the production side. Also if there was more time then it would have been possible to test multiple versions of the wheel arms. If the arms could have been even longer then the crossing of the gaps could have gone even more efficiently, and this could have saved a lot of time in the final trial of the robot.

Overall the robot has performed well. Thanks to the combination of the spider design and the support wheels, it was possible both climb and cross over the gaps. The robot fits in the RPC list that was made and has reached the goals that were set.

8 | References

[1] Elijah. J. How to program a quadruped robot with arduino, https://makezine.com/article/technology/robotics/robot-quadruped-arduino-program/.