4CBLA20 Course

Multiped Robot

Quartile 3 - 2024-2025

Full Name	Student ID
Arseny Bykov	2110768
Noah van der Horst	2164817
Dani Martínez Ricarte	2110326
Kasper Merkx	2156725
Francesco Salvatore	2118297
Ignas Sinkevičius	2110148
Koen van Weerden	2115964

Tutor: Tim Pansters

Eindhoven, September 18, 2025

Contents

List of Symbols

Table 1: List of symbols

Symbol	**Variable**	**Unit**	**Unit symbol**
m	mass	kilogram	kg
M	total mass	kilogram	kg
CoM	Center of mass	-	-
x	distance	meter	m
y	distance	meter	m
z	distance	meter	m

1 Introduction

For this course, the teams were tasked to design a multiped robot, which would have to complete an obstacle course as fast as possible following the project description [6]. Whilst the robot does not necessarily have to be multiped (have multiple legs), it is recommended to include them.

This report goes over the group's process of developing the robot, from the initially given requirements, to developing a chosen design and conclusions gathered from testing. Each of the sections represent the 7 stages of design: Framing the question; Conceptualization; Selection; Detailing; Assembly; Testing and Optimisation, and Evaluation. All of these phases are tied closely to this project.
The first section is "Concept choice and evaluation", describing the RPC list, the proposed designs and sketches, as well as the group's preferences for the designs. Following is the "Working mechanism and operation" section, which will describe the final design in detail.
Finally, 'Testing and evaluation' reflects on the test, and analyze what could be improved for future projects.

2 RPC List

2.1 The RPCs

The task is to design and create a robot that is able to traverse the racetrack and overcome obstacles. The entire robot needs to make it past the finish line and the given materials must be used, but extra material can be requested and simple materials from home are also allowed to be used. Preferably, the robot should stand out, to set it apart from the other robots. The robot should also be optimized for speed, to finish the course within one minute. Making the robot simple or making it easily modifiable would offer great ease in case modifications are needed to increase the performance. It would also be preferable if the robot would go underneath the bar near the end of the racetrack, as that would most likely save time. There are however some constraints set, as the robot must fit within the starting box of 30 cm x 30 cm x 15 cm and it also needs to finish the course within a time of 5 minutes, otherwise it will count as a DNF, unless the checkpoint is reached. Some other constraints are the fact that the robot must be powered by an external power supply and it must be controlled by an Arduino board and lastly, for 3D printed parts, the parts must not take more than 6 hours to print.

Besides the list for the RPC, the robot must also be designed taking the difficulties of the racetrack into account. There are a few main parts of the track, the first part is a simple straight flat area, followed by 2 steps, which are different in height, and lastly there is a 10 cm high bar which needs to be overcome by either going over it, or going underneath. While all the sections of the track need to be done as quickly as possible, the stability and the safety of the robot must also not be forgotten. As the robot must not fall off the stairs and flip over for example. The balance of safety and speed is of paramount importance.

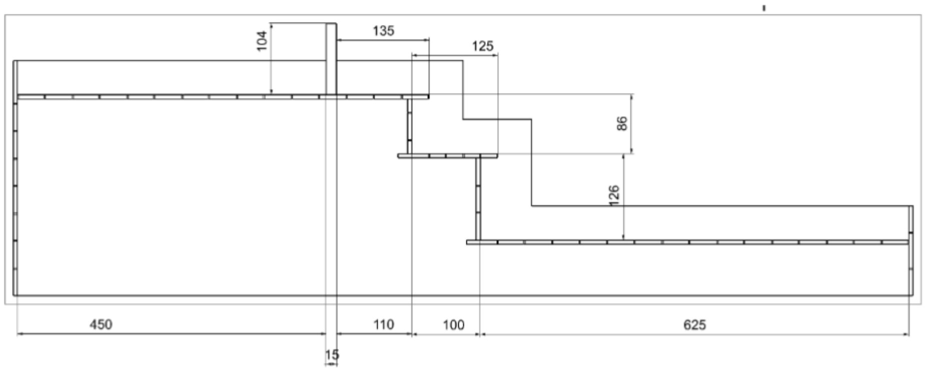


Figure 1: Racetrack dimensions with width 550 mm

3 Conceptual Design

3.1 Jumping Robot

One of the proposed concepts was a jumping robot designed to leap over the stair obstacle and proceed to the finish line. A schematic of the design is shown in Figure 2.

The jumping mechanism, consists of four beams connected by joints. In the resting position (left side of the figure), the robot remains static. To initiate a jump, a red gear rotates clockwise, winding a cable that pulls the lower section upward. Springs, horizontally attached between the beams, store elastic potential energy during this movement. Once the gear reaches its final position and releases tension, the springs contract, converting stored energy into an upward force that propels the robot off the ground. This design would clear the track very quickly and efficiently however very high precision is needed for the robot to land and not clip the stairs. Additionally the robot may not have any prestored potential energy therefore the motors would have to rotate to create the tension.

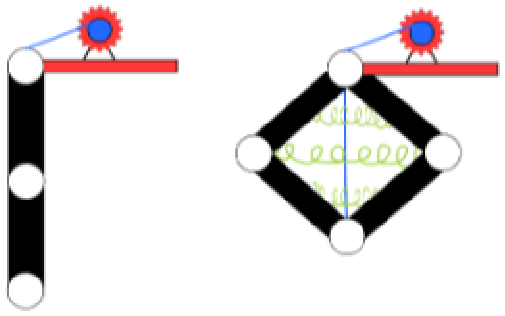


Figure 2: Jumping robot concept: mechanism at rest and fully tensioned. Red beam is the robot's main body, and the black beams are one of its legs

3.2 Cuboid Robot

Another conceptual design was a cuboid-shaped robot inspired by robotic vacuum cleaners [1]. As shown in Figure 3, this robot climbs stairs by rotating its articulated legs, represented by the green and blue sections, while the grey section indicates the central body.

Each leg is equipped with an individual servo motor shown in dark blue, enabling independent rotation and precise control. By synchronizing leg motions, the robot can lift itself onto a stair step and reposition its body as needed. This design would be quite slow and risks tangling wires around its legs however it would be very reliable.

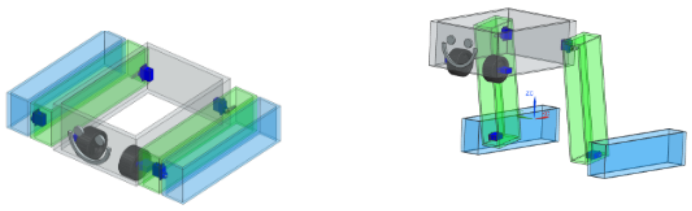


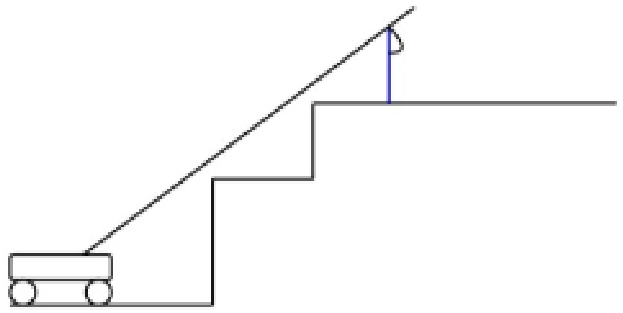
Figure 3: Cuboid robot design showing folded and extended configurations.

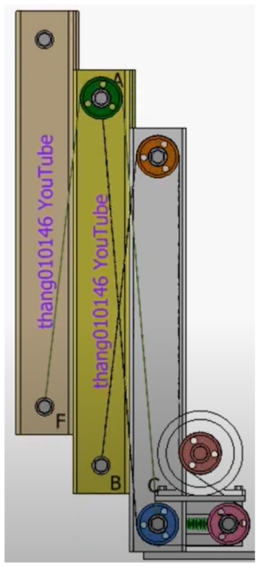
3.3 Telescopic Arm Robot

The third concept features a robot equipped with a telescopic arm and a hook mechanism. The idea is to anchor the hook to a horizontal bar at the top of the stairs and retract the arm to pull the robot upwards.

This design is shown in Figure 4a.

The actuation mechanism, inspired by [5], is illustrated in Figure 4b. It comprises three sliding parts connected via a pulley system. A red pulley rotates to wind two cables: one attached at point B, routed over the gray segment's top pulley, and another passing under a blue pulley, anchored at point A. This motion causes the middle segment to slide. A third cable, fixed at points C and F and routed over the green pulley, ensures the outer segment follows. The telescoping motion enables the robot to extend and contract the arm in a controlled manner. This design would be very reliable however it would have to go over the final bar which would cost the team valuable time.


(a) Telescopic arm robot conceptual design.



(b) Telescopic arm actuation mechanism [5].

Figure 4: Conceptual design and mechanism of the telescopic arm robot.

4 Final Design

4.1 Design Choice

The cuboid robot design was selected due to its balance between simplicity and effective stair-climbing functionality. Compared to the jumping and telescopic arm concepts, it provided greater reliability and control with fewer dependencies on external elements (such as a bar to hook onto) or precise actuation timing. Its motion was more predictable and more feasible to fabricate within the time and material constraints of the project.

A key advantage of this design was its maneuverability. Two servo-controlled wheels enabled forward, backward, and rotational movement, allowing the robot to reposition itself if misaligned. In addition, individually actuated legs gave the robot modular control during climbing, while the simple rectangular body geometry allowed for straightforward modeling and rapid prototyping.

4.2 First Model and Dimensions

Figure 5 shows the first model of the robot. The original body dimensions were 14 cm $\times$ 7 cm, with four total servos embedded into the base two for wheel control and two for rotating the legs. The wheel servos enabled differential steering, allowing the robot to rotate on the spot by moving one wheel while keeping the other stationary.

During development, potential wire tangling during leg rotation was identified as a risk. Slip rings were considered but ultimately dismissed due to the complexity of self-manufacturing. Further analysis showed that wire movement remained within acceptable limits, so additional components were unnecessary. Another issue was that the initial concept exceeded the maximum allowed width of 30 cm and had insufficient clearance to reach the first step, which was 12.5 cm tall. These findings led to significant design refinements.

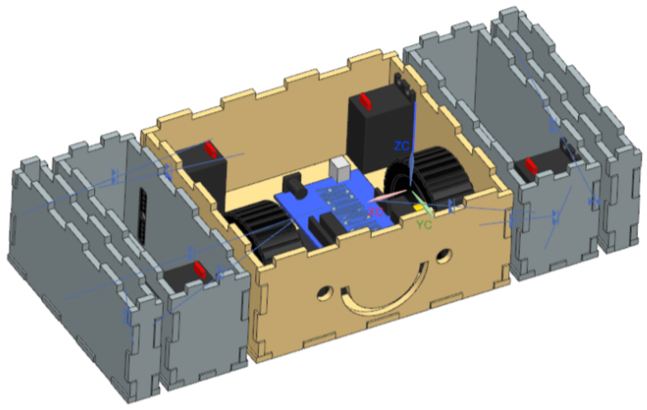


Figure 5: Initial 3D model of the robot, showing carcass and leg assemblies.

In the next iteration, major dimensional changes were required. The original body height in the vertical climbing position measured only 9.5 cm from the ground to the base, which was insufficient to clear the 12.5 cm stair. The body was resized to 18 cm in length and 5 cm in height, while retaining the 5 mm puzzle-joint indents used for laser-cut plywood connections.

The reduction in width was more challenging. Due to the size of the servos and wheels, the robot remained too wide. To address this, a new connection method was designed: star-shaped cutouts in the side walls of the legs allowed servo arms to be mounted more compactly, reducing the robot's total width by 1.2 cm and maintaining a safe margin under the 30 cm limit.

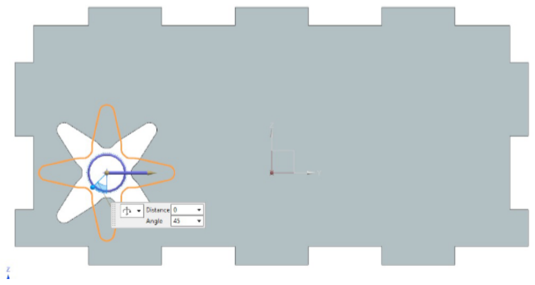
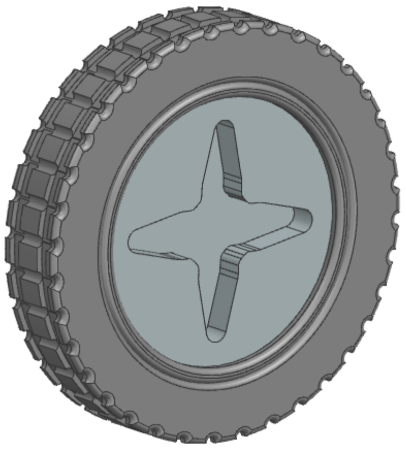
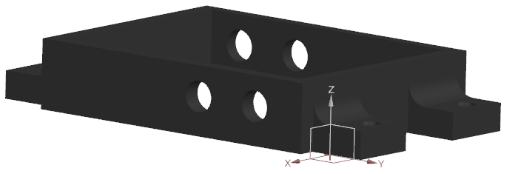


Figure 6: Star-cut geometry in the leg wall for compact servo arm connection.

Other modifications included rotating the servo orientation and shifting their mounting slots to lower the center of mass and reduce the overall height of the robot. Additionally we also laser cut extra legs to prevent the robot from falling back while on top of the steps, shown in figure 8, and further elaborated in chapter 4. While all walls were made from laser-cut plywood, a number of parts were 3D-printed for strength and flexibility, including the wheels and servo holders, shown in Figure 11.


(a) 3D printed wheel.



(b) Servo holder designed to support rotational wheel motion.

Figure 7: 3D printed mechanical components integrated into the final design.

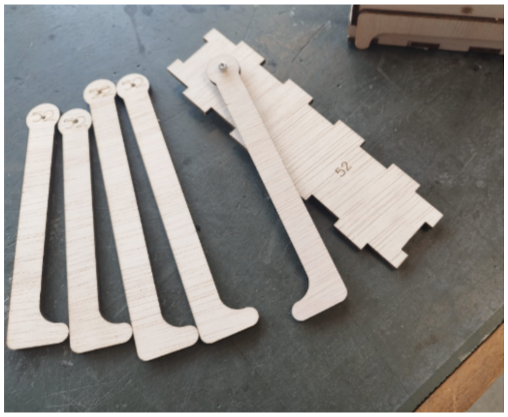


Figure 8: Multiple sizes of the laser cut back support legs.

5 Modelling

To predict and evaluate the robot's ability to clear the race track, two modelling approaches were used: kinematic simulation in Simcenter and analytical kinematic analysis in MATLAB. The goal was to determine whether the robot would succeed in climbing both steps and, if not, identify key reasons for failure and guide design improvements.

5.1 Simcenter Simulation

The full robot assembly was first imported into Simcenter, where its rotation and motion sequences were modelled to simulate step-climbing. The simulation focused on verifying whether the robot's legs could lift the body sufficiently, and whether the climbing motion maintained balance and stability.

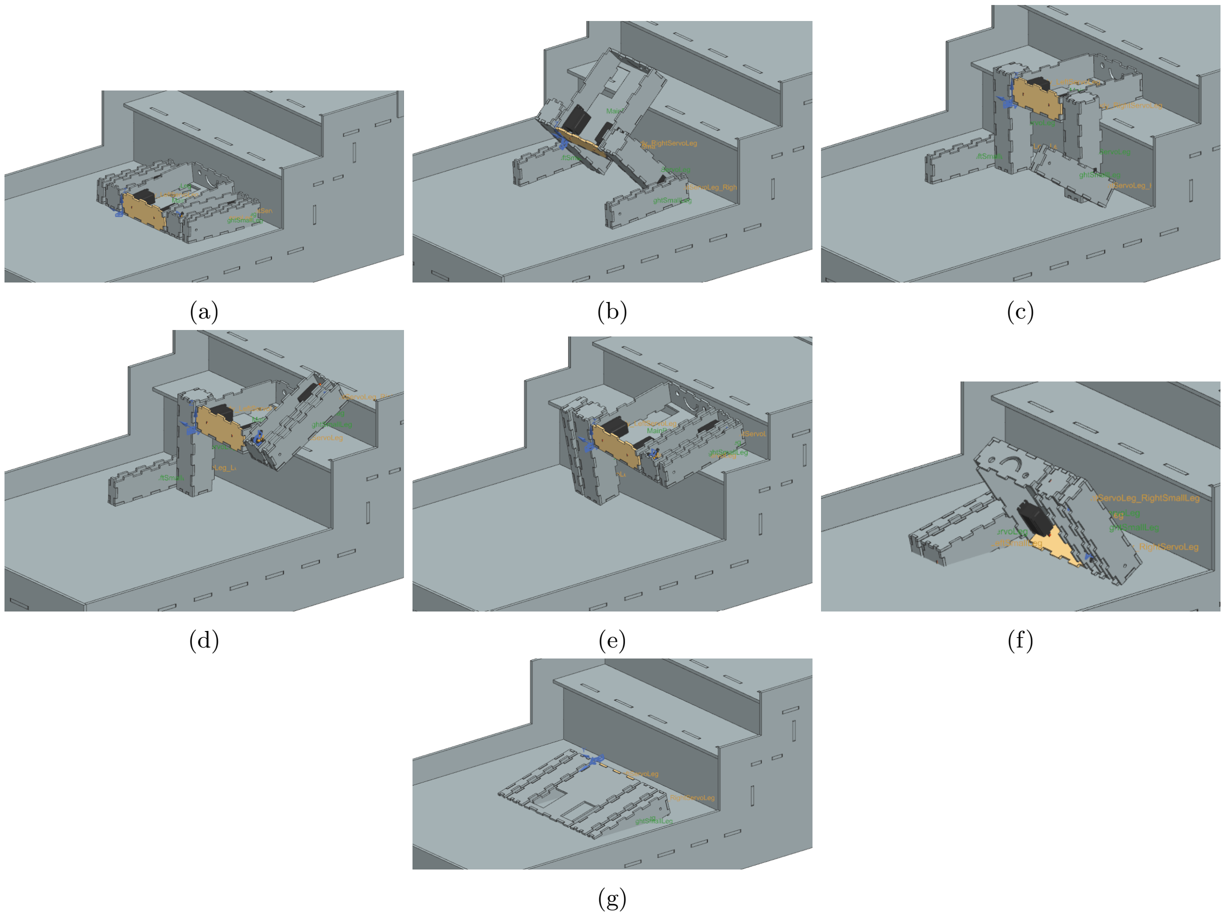


Figure 9: Rotation simulation of the robot in Simcenter.

The results showed that the robot managed to almost climb the first step but when retracting its left leg in sub figure 9.e it would fall down. This was likely due to the center of mass of the robot being towards the rear end. This prompted a deeper investigation into the robot's stability, weight distribution and center of mass during the climbing sequence.

5.2 Modeling of Center of Mass

To further analyze this issue the team had to track the robots center of mass. To do that the following [3]. formula was used:

$$\frac{\sum m_i x_i}{M_{total}} \tag{1}$$

a MATLAB model [2] was created to track the robot's center of mass (CoM) during various stages of its ascent. The hypothesis was that the cause of failure was an unstable CoM position during critical transitions.

The MATLAB model was structured in three parts:

- A function to calculate the center of mass based on input mass and position data.
- A catalog of all major component masses and coordinates (wheels, legs, servos, body).
- A couple of manually defined postures representing key climbing positions.

This allowed the team to generate a plot showing the CoM trajectory relative to the robot's contact points.

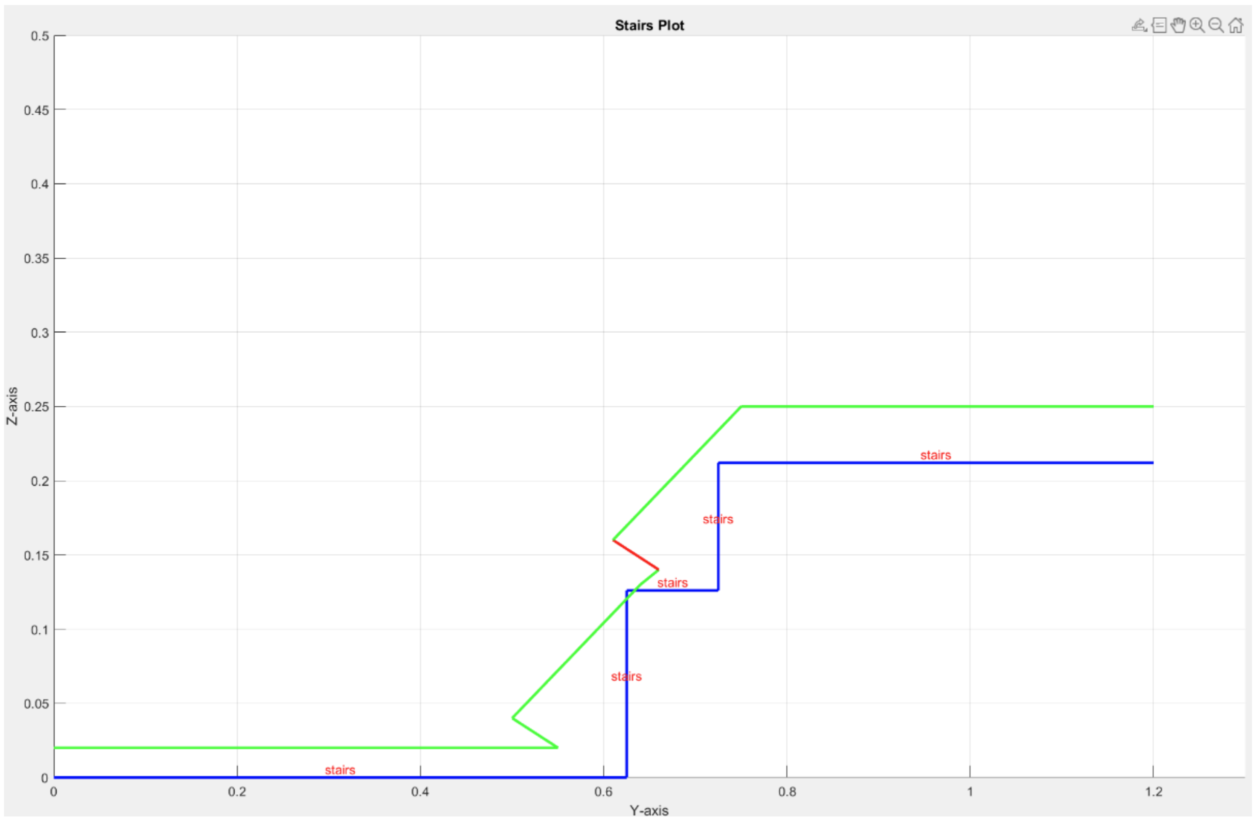


Figure 10: Center of mass trajectory across climbing sequence.

As shown in the chart, during the transition to the second step, the robot's center of mass briefly projected beyond the step. This imbalance caused the robot to tip backward during tests, matching the observations in Simcenter.

5.3 Design Revision Based on Model Results

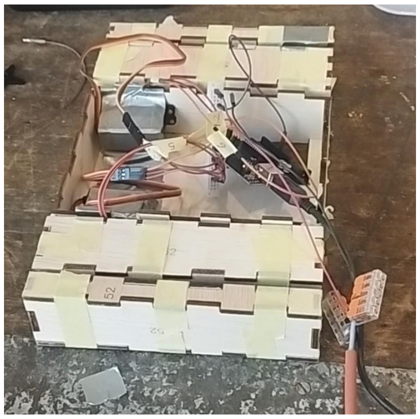
Based on these insights, the team introduced additional supporting legs, Figure 8. This increases ground contact and extend the area of contact during critical transitions. This change ensured that the center of mass remained within the robot's stable region throughout the climbing motion. As there is now a contact point at the rear end of the robot when climbing the step at the the critical point. The supporting legs were designed after this conclusion, laser cut out of plywood and integrated into the final robot assembly.

6 Testing

6.1 Assembling the robot

The robot has two primary functionalities that require testing: the ability to drive smoothly on flat surfaces while maintaining stability and control, and the execution of a predefined movement sequence to climb stairs and retract its legs.

Ensuring the robot's ability to climb stairs was considered the most critical aspect of the project, making it the first functionality to be implemented and tested. To achieve this, the necessary components were loosely assembled using tape and provisional methods, and the robot's stair-climbing capability was evaluated. The tests confirmed that the robot could successfully climb the first step and retract its legs onto it. However, it was unable to clear the second step consistently, as it would lose balance and fall backward, as predicted in the Simcenter and Matlab simulation shown in Chapter 5.


(a) Loosely built robot.

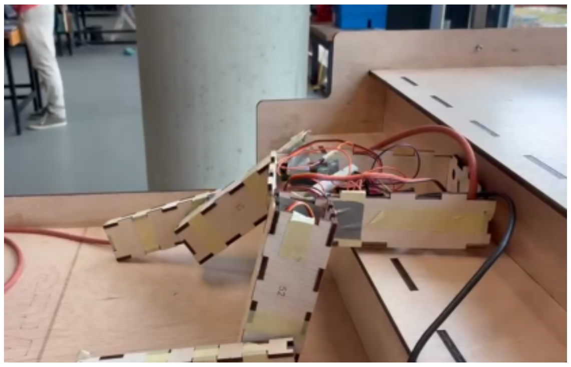

(b) Loosely built robot climbing first step.

Figure 11: Initial tests with robot.

After confirming that the robot could climb individual steps, it was assembled using more permanent methods, such as glue and screws although tape was still used. The wheels, servos for rotating the wheels, and back supports were then added.

6.2 Movement of the robot

The robot's movement on flat surfaces was tested and showed good performance. After adding a high-friction material to the wheels, the robot's control improved significantly, allowing for more effective navigation and better stability even on top of the steps.

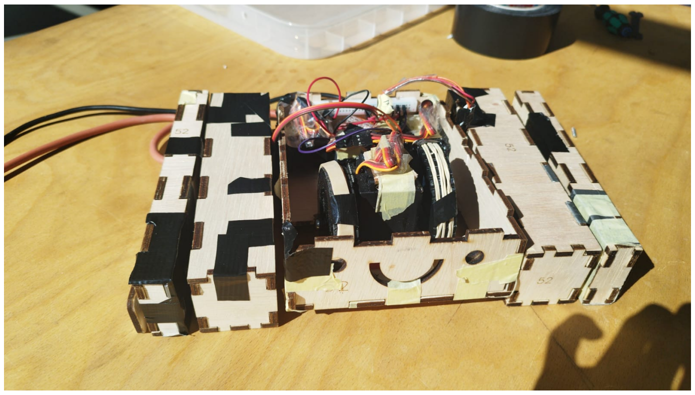


Figure 12: Fully built robot.

Climbing the all the stairs in one go, however, required more extensive testing. The goal was to replicate the motion seen in the model, but unfortunately replicating that motion proved to be challenging. Getting the correct speed and time that every servo motor needs to be activated for every movement needed to be found by trial and error. There were some discrepancies between the theoretical values needed to rotate the servo motors and the values that worked in practice due to drag and gravity making down movements faster than upward ones. These discrepancies lead to the robot struggling to maintain stability while climbing or even falling over. Which in turn lead to mechanical problems, as pieces of the robot broke or got disconnected.

To resolve these issues, the robot underwent several repairs, with some parts being replaced to enhance its overall performance, even one leg had to be replaced after it broke during testing. The robot's motion was then iteratively tested by adjusting servo motor speeds and timing sequences. To do so the code for the robot, see Appendix A, was adjusted after every testing trial. The motion for the first step was programmed using a hard-coded motions of the rotation of the whole body onto the step and retraction of the legs one by one, see Figure 13.

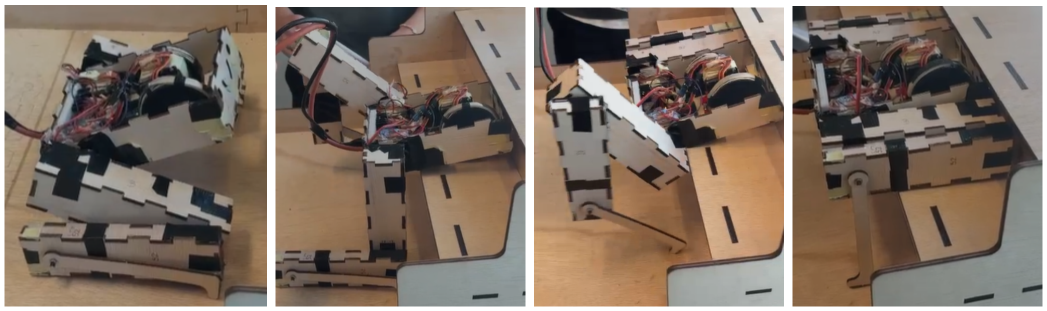


Figure 13: Motion of the robot on to the first step.

However, the hard-coded approach failed to control the robot's movement on the second step due to which the testing process was adjusted. The movement was broken down into smaller motions, each mapped to a specific keyboard key. Additionally, individual controls were introduced for each servo, allowing movement in both directions. This enhanced precision enabled the correction of any misalignment or imperfect positioning of the robot on the track.

6.3 Design ismprovements

Throughout the development process, several key improvements were made to enhance the robot's performance and reliability. The original wheels were found to be too bulky for the constrained frame dimensions, so custom 3D-printed wheels were designed to provide a better fit and smoother motion. To further improve climbing performance, the team laser-cut additional stability legs, which increased the robot's ground contact during stair ascent and prevented tipping. Another enhancement was rearranging the wiring at the carcass of the robot, which decreased the tangling of the wires near the servos and allowed for easier wires connection in case any changes had to be initiated.
Motor stability was also a critical focus. Initially, the servos exhibited excessive play, compromising control precision. To resolve this, the servos were securely fastened to the frame using bolts, significantly reducing mechanical play and improving the consistency of joint movements. Ultimately, the adjustments in the assembly as well as the code allowed the robot to reach the finish line of the track.
For future iterations, several enhancements could be considered. Integrating sensors such as ultrasonic distance sensors would allow for more accurate environmental awareness and precise positioning during stair-climbing sequences. Reducing the robot's overall size and weight could eliminate the need for auxiliary stability legs, allowing it to fit entirely on the first step and improving overall agility. Additionally, curving the edges of the legs could improve retraction by reducing interference with the overhang. Unlike square legs, the curved design would minimize contact, allowing for smoother movement.

These refinements would further optimize the robot's performance while maintaining its simplicity and reliability.

7 Conclusion and discussion

7.1 Conclusion

This project successfully demonstrated the design, development, and optimization of a multiped robot capable of navigating the race track that included stair climbing and time constraints. Through a structured design process and iterative prototyping, the final robot achieved a reliable balance between stability, and mechanical simplicity. Simulation and modeling, particularly through Simcenter and MATLAB, provided valuable insights into the robot's limitations, most notably in stability during transitions, which directly informed key design modifications such as the addition of supporting legs and structural reinforcements. Ultimately however, it was able to complete a whole track proving that a cuboid robot was a robust solution, offering predictable motion, compact construction, and effective stair-climbing capabilities. Despite some remaining limitations, such as reliance on passive elements for balance, the project met the set requirements of finishing the course, preference of being a complex design that is easily adjustable and highlighted the importance of iterative optimization, testing, and simulation-driven design.

7.2 Evaluation

Although the robot successfully completed the entire course, several design problems affected its consistency. Beyond the previously mentioned improvements, the most significant issue was the connection of the legs. Excessive mechanical play weakened the robot's overall stability and performance. While multiple solutions were considered, such as screwing the servo arm to the leg or 3D printing particular connection parts, the constrained width of the design prevented their implementation.

Another key improvement would be the addition of a dedicated controller, as the current setup is controlled using over 15 key binds, making movement less efficient. Currently, while the robot is capable of finishing the course, frequent manual adjustments make the completion time much longer. Enhancing the robot in these ways, along with the improvements discussed in the testing section, could significantly reduce the time required to complete the course, which would fulfill more the preferences of the design.

References

[1] MIGO$_r$*obotics*. *MIGO ascender*. https://migorobotics.com/. May 2024.

[2] Cleve Moler. *MATLAB*. https://www.mathworks.com/company/aboutus/founders/clevemoler.html. 1984.

[3] R Nave. *Center of mass*. http://hyperphysics.phy-astr.gsu.edu/hbase/cm.html. N/A.

[4] OpenAI. *ChatGPT*. https://chatgpt.com/. Nov 2022.

[5] thang010146. *Cable telescopic sliders 2*. https://www.youtube.com/watch?v=7rgVmOvIMuY. Oct. 2023.

[6] J.Bakers Ye Wang Y.B van de burgt. $Project_Information_Document_24-25$. https://canvas.tue.nl/courses/27553/files/6112996?module_item_id=645341. Feb 2025.

Use of AI

AI was used throughout the project in various ways.
The first application was for debugging. This was frequently used for the Sim Center when receiving error messages. These messages are often very unclear and hard to define where the error lies. To address this, we used ChatGPT [4] to analyze these messages and suggest potential solutions. By asking questions like "What is going wrong in the simulation?" or "Where can I find possible solutions to this error message", we were able to improve the Sim Center progress significantly.
Another way AI was used was in the coding process. This was mainly done for debugging and making sure we understood the code better. We made sure to not use ChatGPT to actually code, but used it to aid us in the coding process.
To ensure that the answers we were given were trustworthy, we used multiple techniques. One of which was to ask ChatGPT to also give the sources it used when finding these answers. We could then ourselves judge if these sources were of high enough quality and in validity/trustworthiness. The second way made sure the responses were of sufficient level, was to ask the same question in different prompts. This was done to see if the AI gave consistent responses. We made sure in this process to open new chats, as otherwise ChatGPT was likely to repeat its previous answer.

A Code for the movement of the robot

```
#include <Servo.h>

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;

const int stopAngle = 90;
const int forwardLeft = 180;
const int forwardRight = 0;
const int backwardLeft = 0;
const int backwardRight = 180;

char lastCommand;
bool runFirstPart = false;
bool runSecondPart = false;
bool runThirdPart = false;
bool runFourthPart = false;
bool runFifthPart = false;

void setup() {
  servo1.attach(9);
  servo2.attach(10);
  servo3.attach(11);
  servo4.attach(12);
  servo5.attach(8);
  servo6.attach(7);
  Serial.begin(9600);
  stopRobot();
  Serial.println("Robot ready. Use WASD to control, X to stop, P to run first sequence, L to run second
}

void loop() {
  if (Serial.available()) {
    lastCommand = Serial.read();
    if (lastCommand == 'p') {
      runFirstPart = true;
    } else if (lastCommand == 'l') {
      runSecondPart = true;
    } else if (lastCommand == 'o') {
      runThirdPart = true;
    } else if (lastCommand == 'i') {
      runFourthPart = true;
    } else if (lastCommand == 'k') {
      runFifthPart = true;
    }
  }

  if (runFirstPart) {
    runServoSequence();
    runFirstPart = false;
  } else if (runSecondPart) {
    runServoSequence2();
    runSecondPart = false;
  } else if (runThirdPart) {
    runServoSequence3();
    runThirdPart = false;
  } else if (runFourthPart) {
    runServoSequence4();
    runFourthPart = false;
  } else if (runFifthPart) {
    runServoSequence5();
    runFifthPart = false;
  }else {
    executeCommand(lastCommand);
  }
  delay(100);
}

void executeCommand(char command) {
  switch (command) {
    case 'w':
      moveRobot(forwardLeft, forwardRight);
      break;
    case 's':
      moveRobot(50, 120);
      break;
    case 'a':
      moveRobot(90, forwardRight);
      break;
    case 'd':
      moveRobot(forwardLeft, 90);
      break;
    case '1':
      moveArm(100,90);
      break;
    case '2':
      moveArm(80,90);
      break;
    case '3':
      moveArm(90,100);
      break;
    case 't':
      moveArm(90,140);
      break;
    case 'y':
      moveArm(140,90);
      break;
    case '4':
      moveArm(90,80);
      break;
    case '5':
      moveMini(100,90);
      break;
    case '6':
      moveMini(80,90);
      break;
    case '7':
      moveMini(90,100);
      break;
    case '8':
      moveMini(90,80);
      break;
    case 'x':
      stopRobot();
      break;
    default:
      return;
  }
}

void moveRobot(int leftAngle, int rightAngle) {
  servo1.write(leftAngle);
  servo2.write(rightAngle);
}
void moveArm(int angleLeft, int angleRight) {
  servo3.write(angleRight);
  servo4.write(angleLeft);
}
void moveMini(int lefty, int righty) {
  servo5.write(righty);
  servo6.write(lefty);
}
void stopRobot() {
  moveRobot(stopAngle, stopAngle);
  moveArm(90, 90);
  moveMini(90,90);
}

void runServoSequence() {
  servo5.write(106);
  servo6.write(75);
  servo3.write(105);
  servo4.write(75);
  delay(2000);

  servo5.write(90);
  servo6.write(90);
  servo3.write(90);
  servo4.write(90);
  delay(3000);
}

void runServoSequence2() {
  servo3.write(110);
  delay(1500);
  servo6.write(70);
  delay(1500);
  servo6.write(95);
  delay(500);
  servo3.write(90);
  delay(3000);
}

void runServoSequence3() {
  servo4.write(70);
  delay(500);
  servo5.write(110);
  delay(1600);
  servo5.write(95);
  delay(600);
  servo4.write(90);
  delay(3000);
}

void runServoSequence4() {
  servo3.write(104);
  servo4.write(76);
  delay(400);

  servo5.write(112);
  servo6.write(68);
  servo3.write(102);
  servo4.write(78);
  delay(1500);

  servo5.write(90);

  servo6.write(90);
  servo3.write(90);
  servo4.write(90);
  delay(3000);
}

void runServoSequence5() {

  servo3.write(80);
  servo4.write(110);
  delay(150);

  servo3.write(90);
  servo4.write(90);
  delay(500);
}
```

B Matlab code

```
%data
%1 main body: 390 grams - 16 cm w - 14 cm l - 3.25 cm h
%2 wheels:  63 g  - 3.15 cm r - 2.65 cm w
%3 servo: 55g - 5.36 cm h - 2 cm w - 4.76 cm l
%4 Arduino uno 25 g - 4.84 cm w - 6.34 cm l - 1 cm h
%5 inner legs 100g - 14 cm l - 3 cm h - 3 cm w
%6 outer legs 100g - 14 cm l - 3 cm h - 1 cm w

function CoMcoordinates = coordinates(m, x, y, z)

    x_cm = sum(m .* x) / sum(m);
    y_cm = sum(m .* y) / sum(m);
    z_cm = sum(m .* z) / sum(m);

    CoMcoordinates = [x_cm, y_cm, z_cm];

    fprintf('Center of Mass (X, Y, Z): (%.2f, %.2f, %.2f)\n', x_cm, y_cm, z_cm);

end



% masses and coordinates test values
% main body, wheels, servos, arduino, legs
m = [0.39, 0.063, 0.063, 0.055, 0.055, 0.055, 0.055, 0.055, 0.055, 0.1, 0.1, 0.1, 0.1];
% main body, wheel1, wheel2,
% servo1,servo2,servo3,servo4,servo5,servo6, leg1,leg2,leg3,leg4
x = [0.07, 0.025, 0.135, 0.08, 0.08, 0.15 , 0.02, -0.03, 0.17, 0.19 , 0.17 , -0.02, -0.04 ];  % X-coordi
y = [0.08, 0.095, 0.095, 0.1, 0.1, 0.03 , 0.03 ,0.1,0.1, 0.07, 0.07 , 0.07 , 0.07];  % Y-coordinates in
z = [0, 0, 0 , 0 , 0, 0,0, 0, 0,0,0,0,0];  % Z-coordinates test values


%position 2 90 degrees at stair
y2 = [0.03, 0.03, 0.03, 0.03, 0.03, 0.03 , 0.03 ,0.1, 0.1, 0.07, 0.07 , 0.07 , 0.07];  % Y-coordinates i
z2 = [0.07, 0.095, 0.095 , 0.095 , 0.095,0,0, 0,0, 0, 0,0,0];  % Z-coordinates test values

%position 3 2nd 90 degrees at stair 1
y3 = [0.17, 0.17, 0.17, 0.17, 0.17, 0.17 , 0.17 ,0.1, 0.1, 0.07, 0.13 , 0.13 , 0.07];  % Y-coordinates i
z3 = [0.21, 0.235, 0.235 , 0.235 , 0.235, 0.13, 0.13, 0,0, 0, 0.07,0.07,0];  % Z-coordinates test value


%position 4 rotating legs up
y4 = [0.2, 0.21, 0.21, 0.21, 0.21, 0.14 , 0.14 , 0.21, 0, 0.21, 0.21 , 0 , 0];  % Y-coordinates in m
z4 = [0.13, 0.16, 0.16 , 0.14 , 0.14, 0.14, 0.14, 0.14,0.14, 0.14, 0.13,0.13,0.13];  % Z-coordinates te


%same positions are repeated therefore no more calculations have to be performed


CoMcoordinates = coordinates(m, x, y, z);
CoMcoordinates = coordinates(m, x, y2, z2);
CoMcoordinates = coordinates(m, x, y3, z3);
CoMcoordinates = coordinates(m, x, y4, z4);







figure;
hold on;
grid on;
axis([0 1.3 0 0.5]);

stairs_x = [0, 0.625, 0.625, 0.725, 0.725, 1.2];
stairs_y = [0, 0, 0.126, 0.126, 0.212, 0.212];
CoMy = [0,0.55, 0.55, 0.5, 0.5, 0.64, 0.64 ,0.66, 0.66 ,0.61, 0.61, 0.75, 1.2  ];
CoMz = [0.02,0.02, 0.02, 0.04, 0.04, 0.13, 0.13, 0.14, 0.14 ,0.16, 0.16, 0.25, 0.25 ];

for i = 1:length(stairs_x)-1
    line([stairs_x(i) stairs_x(i+1)], [stairs_y(i) stairs_y(i+1)], ...
        'Color', 'b', 'LineWidth', 2);


    text_x = (stairs_x(i) + stairs_x(i+1)) / 2;
    text_y = (stairs_y(i) + stairs_y(i+1)) / 2;
    text(text_x, text_y, 'stairs', 'FontSize', 10, 'Color', 'r', ...
        'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom');
end

for i = 1:(length(CoMy) - 1)
    if CoMy(i) == 0.66 && CoMy(i+1) == 0.61 && ...
       CoMz(i) == 0.14 && CoMz(i+1) == 0.16
        lineColor = 'r';  % Red for the specific segment
    else
        lineColor = 'g';  % Green for all others
    end
    line([CoMy(i), CoMy(i+1)], [CoMz(i), CoMz(i+1)], 'Color', lineColor, 'LineWidth', 2);
end

xlabel('Y-axis');
ylabel('Z-axis');
title('Stairs Plot');
hold off;
```

C Sim Center Video

Please use the following link to find the video from which the sub figures of figure 9 were taken.
`https://youtu.be/iERrGqQHZnU`