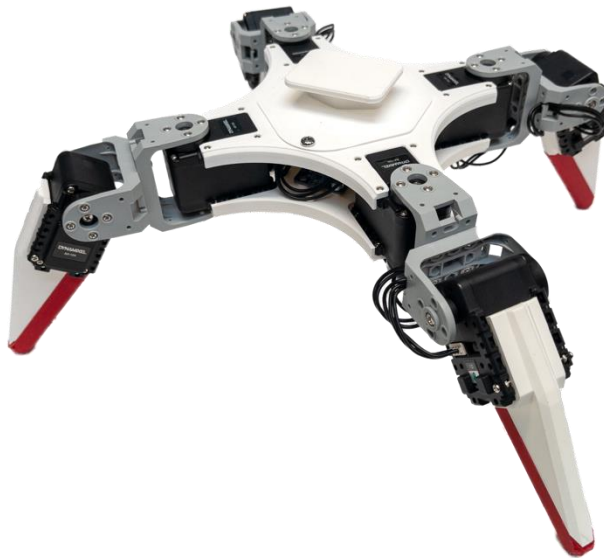


Final Report

Project #1-10

Building a Quadruped Robot for Reinforcement Learning Research



Date: 31.05.2022

Antti Sippola
Eric Hannus
Jed Muff
Jere Vepsä
Julius Mikala
Rituraj Kaushik

Information page

Students

Antti Sippola

Eric Hannus

Jed Muff

Jere Vepsä

Julius Mikala

Project manager

Jed Muff

Official Instructor

Rituraj Kaushik

Starting date

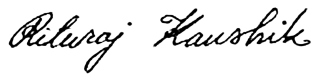
24.01.2022

Completion date

03.06.2022

Approval

The instructor has accepted the final version of this document



Date: 02/06/2022

Abstract

This document contains a full review of the "Building a Quadruped Robot for Reinforcement Learning Research" project conducted by Group 1-10 in conjunction with the "ELEC-E8004 Project Work" module. The project aimed to build and improve a quadruped robot that will be used for reinforcement learning research. The quadruped robot built is based on the open-source Real-Ant designed by Ote Robotics at Aalto University. With the previous design, issues with robustness and usability were identified. These are related to the screw fittings and the repairability of the design. Design improvements included soft attachments to dampen damage to the robot, proper board mounting for easy construction, access hatches for ease of use and repairability and a stand for a motion tracking marker (used in reinforcement learning).

Along with an improved robot design, some code was developed that utilized DYNAMIXEL AX12 actuators to produce walking gaits and gestures the quadruped robot could use for initial functionality testing.

This project involved documenting the final designs and the code developed. Overall the project was a success, and all objectives were achieved.

Table of Contents

Abstract	3
Table of Contents	4
1. Introduction.....	6
2. Objective.....	7
3. Project plan	7
3.1. Phases	7
3.2. Milestones.....	8
3.3. Roles Within the Project.....	9
3.3.1. Project Manager	9
3.3.2. Instructor.....	9
3.3.3. Work Package Leaders	10
4. Results	10
4.1. Top Level Design.....	10
4.2. New parts.....	11
4.2.1. Soft Parts	11
4.2.2. Body Assembly.....	15
4.3. Testing.....	18
4.3.1. Dynamixel servos	18
4.3.2. Python code	19
5. Reflection of the Project.....	19
5.1. Achieving Objectives.....	19
5.2. Timetable.....	20
5.3. Risk analysis	21
5.4. Project Meetings and Communication	21
5.4.1. Project Meetings	21
5.4.2. Communication plan.....	23
5.5. Quality	23
6. Discussion and Conclusions	24
6.1. Success of the Project	24
6.2. Group Member Personal Reflections.....	24
6.3. Conclusion and Future Work.....	26
7. List of Appendixes	26
7.1. Guide to Dynamixel AX-12A servos.....	26
7.2. How to use the U2D2 adapter with Dynamixel Wizard 2.0 and DynamixelSDK	29
7.3. Meetings and minute template.....	33
References	34

Table of Figures

Figure 1 Ant Benchmark Mujoco Simulation	6
Figure 2 Ote Robotics Real-Ant	6
Figure 3 Project Life Cycle Diagram.....	8
Figure 4 The Final Product.....	10
Figure 5: Wiring diagram.....	11
Figure 6:The soft edge leg with both soft and hard pieces visible.	12
Figure 7: Soft edge leg redesign	13
Figure 8: Soft pieces pictured in Cura.....	13
Figure 9: Tube and 3d printed pieces attached to the actuator connector.....	14
Figure 10: Unused leg designs.....	14
Figure 11. Original CAD design for the body plates.	16
Figure 12. CAD model of the redesigned body assembly.....	17
Figure 13. Redesigned lower (left) and upper (right) body plates.....	17
Figure 14. Lid (left) and stand for motion tracking markers (right).	18

1. Introduction

In this project, all work is based on the open-source Real-Ant quadruped robot designed by Ote Robotics. The Real-Ant is an open-source robot designed to represent the eight degrees-of-freedom robots in the Ant benchmark for reinforcement learning (RL). Figure 1 shows what the Ant looks like in the Mujoco Simulation.

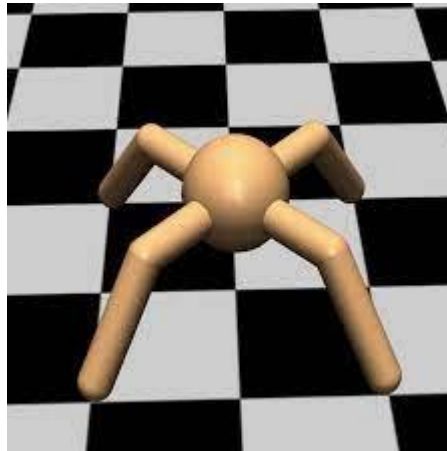


Figure 1 Ant Benchmark Mujoco Simulation

In RL, the goal is for the robot to learn desired behaviour, for example, manoeuvres such as standing, turning and moving in the case of the Real-Ant. Another goal in RL can be to learn to operate in untypical situations, e.g., moving with broken legs or joints.

The Real-Ant platform is open-source, and instructions and code are freely available online (<https://github.com/OteRobotics/realant>). A paper previously researched with the Real-Ant gives a detailed description of the robot ((Rinu Boney, 2020) <https://arxiv.org/abs/2011.03085>, <https://github.com/AaltoVision/realant-rl>). Figure 2 shows what the Real-Ant robot looks like.



Figure 2 Ote Robotics Real-Ant

Using a physical robot for RL results in a more realistic scenario than simulations. The physical robot has increased complex dynamics, noise/inaccuracies, and transmission delays, and the robot's durability must be considered. The robot consists of a 3D printed body with 3D printed legs, connected with servos at the eight joints. For pose estimation, a web camera is used to detect a

marker on the robot and a reference marker in its operating environment. The training system in which the robot is used is distributed across multiple devices. The microcontroller for joint actuation and the previously mentioned web-camera communicates with higher performance computers that coordinate the communication and run the RL algorithms. Real-Ant aims to be a low-cost platform for RL research, resulting in 3D printed parts and a simple web-camera.

Due to the exploratory nature of RL, applying RL algorithms to a physical robot will cause wear on parts due to violent manoeuvres. As this has proven to be an issue in previous research using the Real-Ant at Aalto University, areas of improvement are identified in this project, and new parts are designed to make the robot more durable and easier to repair. Thus, the main goal of this project work is to make the Real-Ant more robust and easier to maintain. This makes the robot more useful in future research projects, as less time and resources are used to maintain the robot. However, before improving on the robot, the project group had to manufacture a standard version and test it to verify that it works.

Boney et al. justified using more high-end servos (Dynamixel AX-12A) for the Real-Ant due to their durability, so improvements in the choice of servos will not be pursued. Instead, the aim is to improve the design of 3D printed parts and find additional ways to increase durability, for example, by adding cushioning.

The project group worked with the design, manufacturing, and assembly of parts and learned about the system in which Real-Ant is used during the testing of the robot. The project also included some software development related to testing the robot's functionality.

2. Objective

This project aims to produce a Real-Ant robot for RL research. The user for the finished robot is the Intelligent Robotics research group at Aalto University, but the robot can also be used by other groups focused on RL research.

The project can be divided into two objectives. The main objective is the production of a working Real-Ant robot. This includes the procurement of parts, 3D printing of parts, assembly of the robot, and verification that the robot works as intended.

The secondary objective is to improve the robot to make it better suited for its intended use in RL research. The improvements are focused on improving the robustness of the robot so that it can perform RL training without needing maintenance. The improvements were planned to happen in two stages. The first stage involved making soft/rubberized feet for the robot and happens concurrently with the building of the robot. Soft/rubberized feet make the robot more robust by reducing the shocks on the components. A second benefit is that rubberized feet might reduce slipping depending on the surface.

The second stage of improving robustness involves testing and then improving possible points of failure. This second stage was optional but was finished.

3. Project plan

3.1. *Phases*

The project itself ran very similarly to how it was planned. The project followed a waterfall type cycle followed by agile development, which will lead up to the final product. This was the planned cycle and is how the project went.

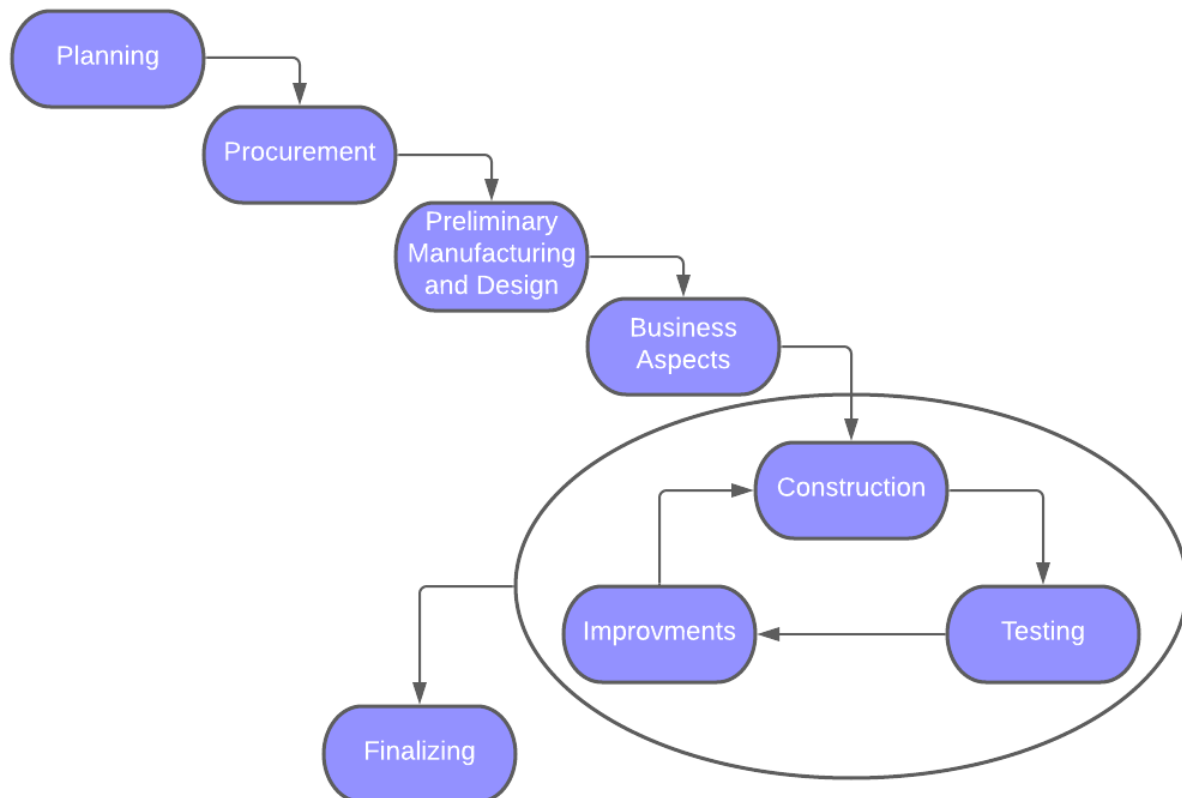


Figure 3 Project Life Cycle Diagram

Each phase is explained below:

- **Planning** - Finalization of the project plan was done and agreed.
- **Procurement** - Entailed ordering and 3D printing of the parts of the robot.
- **Preliminary Manufacturing and Design** – Tasks were done whilst parts were being delivered, and the Business aspects had not started yet. This task involved 3D printing and some design testing.
- **Business aspects** – Involved in producing a presentation and report about the business aspects of the project.
- **Construction** - Once the parts were ready, the robot was assembled.
- **Testing** - was done to confirm that assembled robot performed as intended. This phase also includes the testing of improvements.
- **Improvements** - included the soft feet mentioned previously and other improved parts.
 - **Research** was done to identify what and how parts are improved.
 - **Implementation** entailed designing and procuring the proposed implementations.
- **Finalizing** – documenting what happened and producing reports and presentations to present what has been achieved.
 - **Final Gala**
 - **Final report**

Figure 3 and the phases described above show an abstract representation of how the project was conducted. Tasks from each phase may overlap with other phases based on what could be done concurrently to maximize the time spent on the project.

3.2. Milestones

This section outlines the deadlines for the events and deliverables related to this project. Some milestones are connected to course deadlines, so they cannot be delayed. Those deadlines have been

bolded. M6 is connected to the aim of the project and thus cannot be postponed further than M7, so it has two deadlines, the latter being a hard deadline.

M1 Project plan document finished and approved by instructor (10/2/2022)

M2 Procurement and printing normal parts (25/05/22)

M3 Design and printing of alternative parts (04/03/22)

M4 Business plan presentation slides (11/03/22)

M5 Business plan document and approved by instructor (18/03/2022)

M6 Assemble and confirm working robot (20/05/22) (24/05/22)

M7 Gala (24/05/22)

M8 Final Report and approved by instructor (03/06/2022)

The following deadlines are optional and will be finished if there is time.

M9 Test how the robot should be improved (15/04/2022)

M10 Development and printing of improvements (29/04/2022)

M11 Test improvements (23/05/2022)

These were the planned milestones, although these milestones were not adhered to in some ways as described in the 5.2 Timetable section.

3.3. Roles Within the Project

Within the project are three types of roles. Two are made for two individuals in the group, while the third is a more general one relating to the work package leaders. Although each role has its responsibilities, everyone has some general responsibilities. These responsibilities include:

1. Ensure their work is of good enough quality (details found in the 5.5 Quality).
2. They can do the set work within the time frame given.
3. They do the work within the time frame given.
4. It is human nature to make mistakes; therefore, if it is impossible to do the work in the given time frame, they need to let everyone know so the group can re-plan accordingly.

3.3.1. Project Manager

The project manager in this project has the following responsibilities:

- The project meetings are efficient, on topic and well documented. This involves:
 - Writing an agenda and sending the agenda to everyone at least an hour before the meeting. The purpose is to keep the meetings productive and with direction.
 - To keep meeting memos to document the decisions made and essential notes.
- Ensure the project remains on topic and relevant to the expected output. This involves referring back to the expected output after each project phase.
- The project manager needs to keep everyone informed and maintain a good level of communication to make it clear what people need to do before specific deadlines.
- The project manager also communicates with the course leads/departments about project logistics. This mainly refers to the Gala at the end of the project.
- The project manager is also responsible for submitting the reports to the appropriate place.

3.3.2. Instructor

The instructor acts as the technical expert within the project. Where all other members could lack experience, the instructor's responsibility is to provide advice on areas the members lack knowledge of. This does not mean doing the work but advising how they would do it. They also serve as the primary point of contact between the group and the Robotics Learning group, the primary customer

of this project. They also act as an assistant in using the campus workspaces because of their prior experience with the department workspaces and the COVID restrictions. The responsibilities include:

- They provide the group with advice about the project and answer their questions to their best.
- Acting as a communicator between the Intelligent Robotics Research group and the Team
- The handling of order forms.
- The organization of accessing and using laboratory equipment
- Storing project components

3.3.3. Work Package Leaders

The work package leaders serve to divide the micro-managing parts of the project. Each work package leader is given a specific part of the project. Their primary goal is to ensure that this part is done with good quality. Their responsibilities include:

- Organizing the work package. Whereas the project plan already outlines what needs to be done, this role requires the work package leader to understand each part of the plan, know precisely what needs to be done, and tell others what they should do next.
- Given resource constraints, they should plan efficiently and effectively how to do that work package to get the best result.
- Organize meetings should they be necessary.

These were the planned roles, although some were changed as the project continued. This is discussed later as the timetable changed (see 5.2 Timetable).

4. Results

4.1. Top Level Design

The project's main objective was to build a quadruped robot based on Ote Robotics' open-source Real-Ant robot, which will be used for real-world reinforcement learning research. Figure 4 shows what the final produced result looked like.

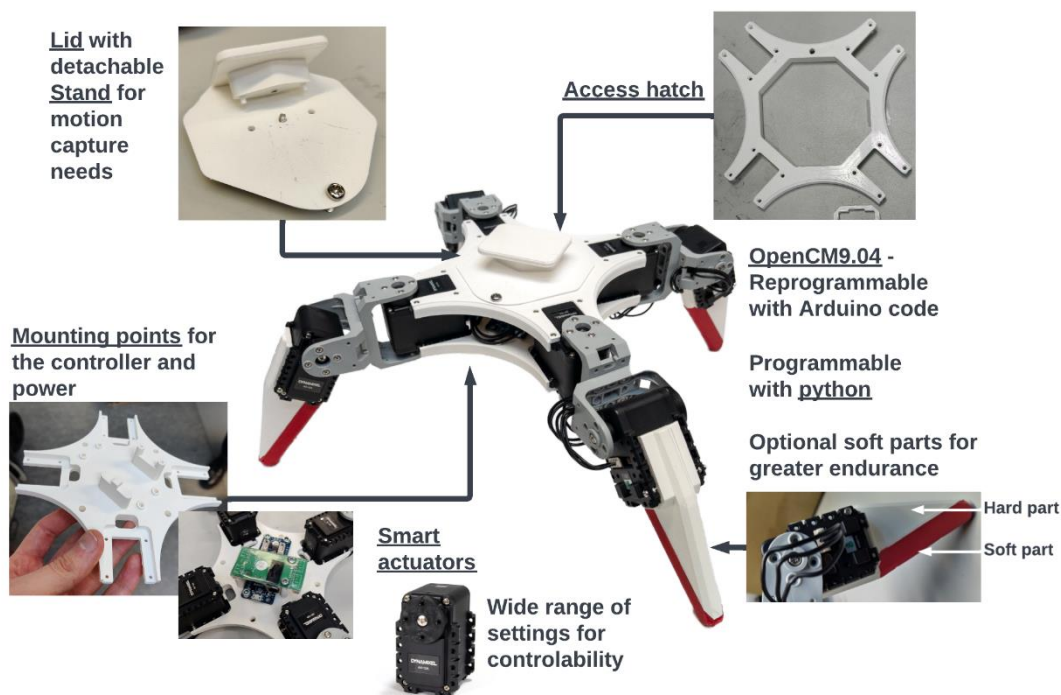


Figure 4 The Final Product

The robot is wired as shown in Figure 5. The Dynamixel bus allows the servos to be daisy chained together and power to be drawn from the SMPS, resulting in less wiring being needed overall. The order of wiring on the boards does not matter. The ID numbers for the actuators does matter depending on the code that is used. The diagram shows what was used, with the front right actuators being 1 at the top and 2 at the bottom. This ID orientation was used to allow for easy visualization of the actuators when testing.

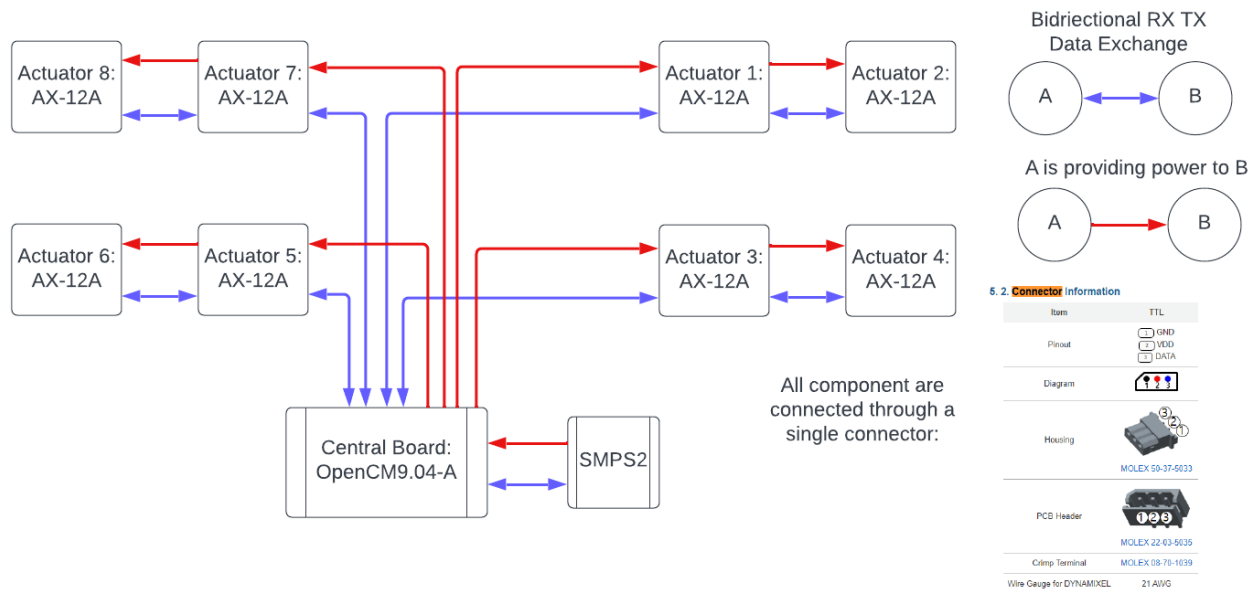


Figure 5: Wiring diagram

The original design of the robot consists of a cover and base plates, which also form the robot's frame that holds everything together and protects the power and control boards. A triangular-shaped part at the end of each leg makes contact with the surface. Both plates and leg parts are made with a 3D printer. The robot utilizes eight high-end Dynamixel AX-12A servos for movements, two in each leg, and has eight degrees of freedom. Movements are controlled by Arduino based OpenCM9.04 control board and powered by a SMPS2Dynamixel power supply connector. The robot is controlled by a computer with a serial connection via USB cable using python scripts. A visual orientation tracker can be added on top of the robot's cover plate.

After building the robot that followed the original design, some changes and improvements were made to the design. Both plates and leg parts were redesigned and printed to improve the durability and maintainability of the robot. The main improvements were newly redesigned legs and body plates, which improved the usability of the robot.

4.2. New parts

4.2.1. Soft Parts

The regular legs of the Real-Ant robot are reasonably fit for their intended use, as can be observed in the paper introducing the robot (Boney, R. et al., 2020. RealAnt: An Open-Source Low-Cost Quadruped for Research in Real-World Reinforcement Learning). One issue with the legs stems from the robot's primary use: reinforcement learning. This use case makes the Real-Ant likely to make more erratic motions and even slam itself to the ground, which would likely not be the case with an inverse kinematics approach for control. By observing the finished Real-Ant, it was speculated that the motions might cause some durability problems for the robot. One such durability problem was that the shocks caused by the movement meant the screws holding the robot together

would come loose. This is a problem since it might require repairing, causing frustration and wasted time.

A solution to the screws coming loose is using thread locking fluid found on the Real-Ant Github (OteRobotics, 2020). The thread locking fluid only solves the issue by fastening the screws, but it does not remove the shocks caused by the robot moving. It is, however, unclear whether the shocks are a real problem. The solution was to change the design of the robot to incorporate soft parts.

Designs

It was observed from the videos of the robot operating that the soft parts needed to be situated at the bottom of the leg and the bottom plate. The soft parts at the bottom plate are required because the robot can strike the ground with its body first. As the designs focused mainly on the legs, the complex parts of the legs were also changed: Making the hard part fit the soft part and the leg spacer part a bit shorter to get a better fit.

For the prints, Ultimaker TPU 95 Red was used. The settings were 10% infill density, gyroids as the infill shape, and wall thickness of 1 mm. It was found that settings produced suitably soft parts, as experimenting with a higher infill density was too rigid for dampening. Lastly, each person in the group used their preference of CAD software, including OpenSCAD and SOLIDWORKS. The slicing software that made the instructions for the 3D printers was Ultimaker Cura. 3D printers used were either the Ultimaker S3 or S5.

Soft edge leg

The idea behind this design is to create a small and easily printable soft leg part that held on to the larger and more complex rigid piece with two taps. The taps are slightly more prominent at the top, and the soft part has holes that match the taps. The idea is that the taps can be pushed through the smaller diameter portion of the holes of the soft part, making them fit securely in place. The design can be seen in Figure 6. The modified leg also features a 0.5 mm smaller leg spacer. The leg spacer modification was done because the normal leg spacer leg combination was challenging to fit into the motors.

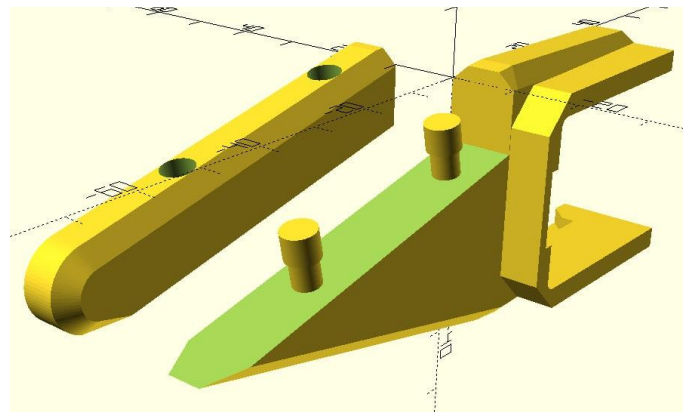


Figure 6: The soft edge leg with both soft and hard pieces visible.

The next iteration for this design was removing the pegs and adding a hole in the soft part for better softness. The pegs were removed to make the part more uniformly soft. Problems with the initial design were that the hole made the side of the printed part different widths, which required removing the edge rounding. The edge rounding can be removed as it is likely just a visual addition. This redesign can be seen in Figure 7. This iteration proved too soft overall but was rigid on the rounded tip. Furthermore, the now completely straight sides showed issues with layer adhesion on

the walls. The adhesion problem can be solved by increasing the printing temperature, and the max temperature for TPU 95 of 235 C will be used in further iterations.

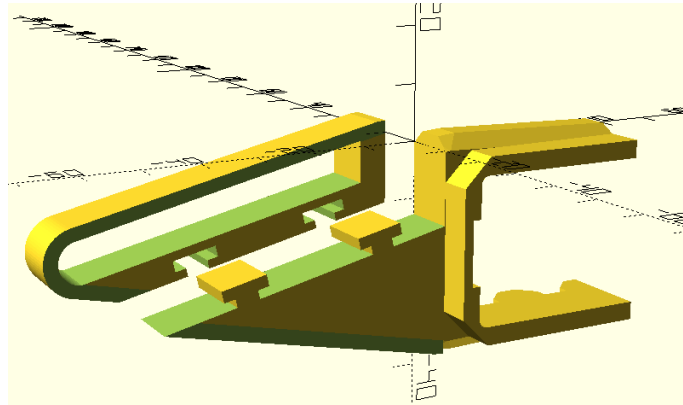


Figure 7: Soft edge leg redesign

The next iteration of the soft leg only keeps the empty circle at the tip of the leg. This is done to create a softer tip while preserving the adequately soft long edges of the leg. The leg is pictured in Figure 8.

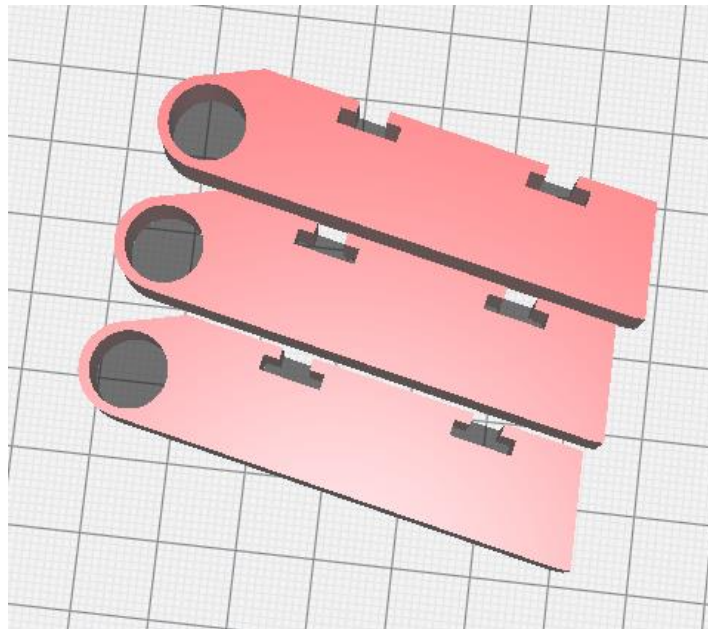


Figure 8: Soft pieces pictured in Cura.

Further iterations also enlarged the t-shapes for easier fastening and added back the chamfers, likely producing a better leg shape.

Body soft pieces

It was determined that the robot needs a soft piece for the body as the body and the parts connecting the actuators touch the ground when the robot moves around. The parts can be added to the body plate, meaning those pieces must extend past the actuator connectors. The soft parts may also be added to the actuator connectors.

Firstly, designs were tried with the soft parts connecting the body plate. One of these is shown in Figure 10, but it was determined that these would be too impractical. The design added holes for the body plate where the soft pieces could be connected.

Finally, a design was settled upon that connects to the actuator connectors. Those parts can be as simple as a soft tube attached to the actuator connector with metal wire. Pieces were also designed that can be 3d printed and attached with screws. Both designs are shown in Figure 9. After printing

the design attached with screws, it turned out that the small size of the piece, in combination with the multiple edges close to the holes, made the design relatively inflexible, so its dampening effect would not be significant. Additionally, this design requires enlargement of the holes in the plastic frame to 2mm by drilling, so this design was also deemed suboptimal as it required modifications to the non-3d-printed components. Thus, the simple tube solution was used instead.

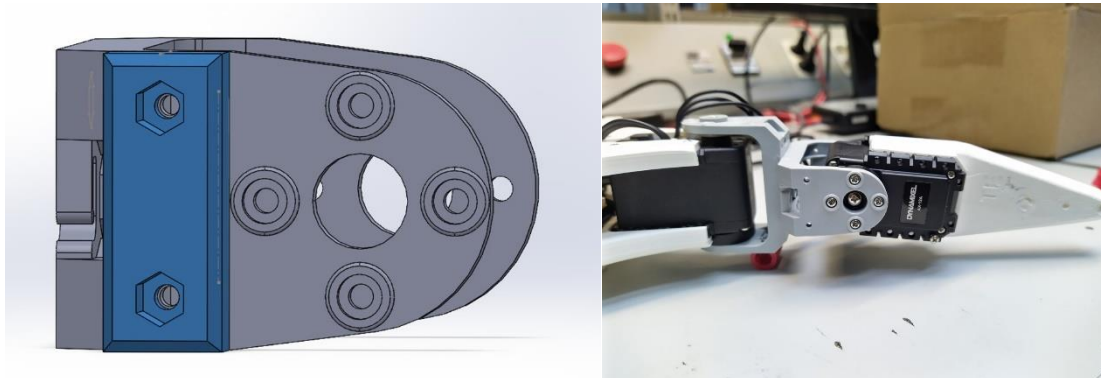


Figure 9: Tube and 3d printed pieces attached to the actuator connector.

Other Designs

Other designs have been made but were never printed as these part may be challenging to print. These parts may be challenging to print due to the support needed for the 3D printing to be successful.

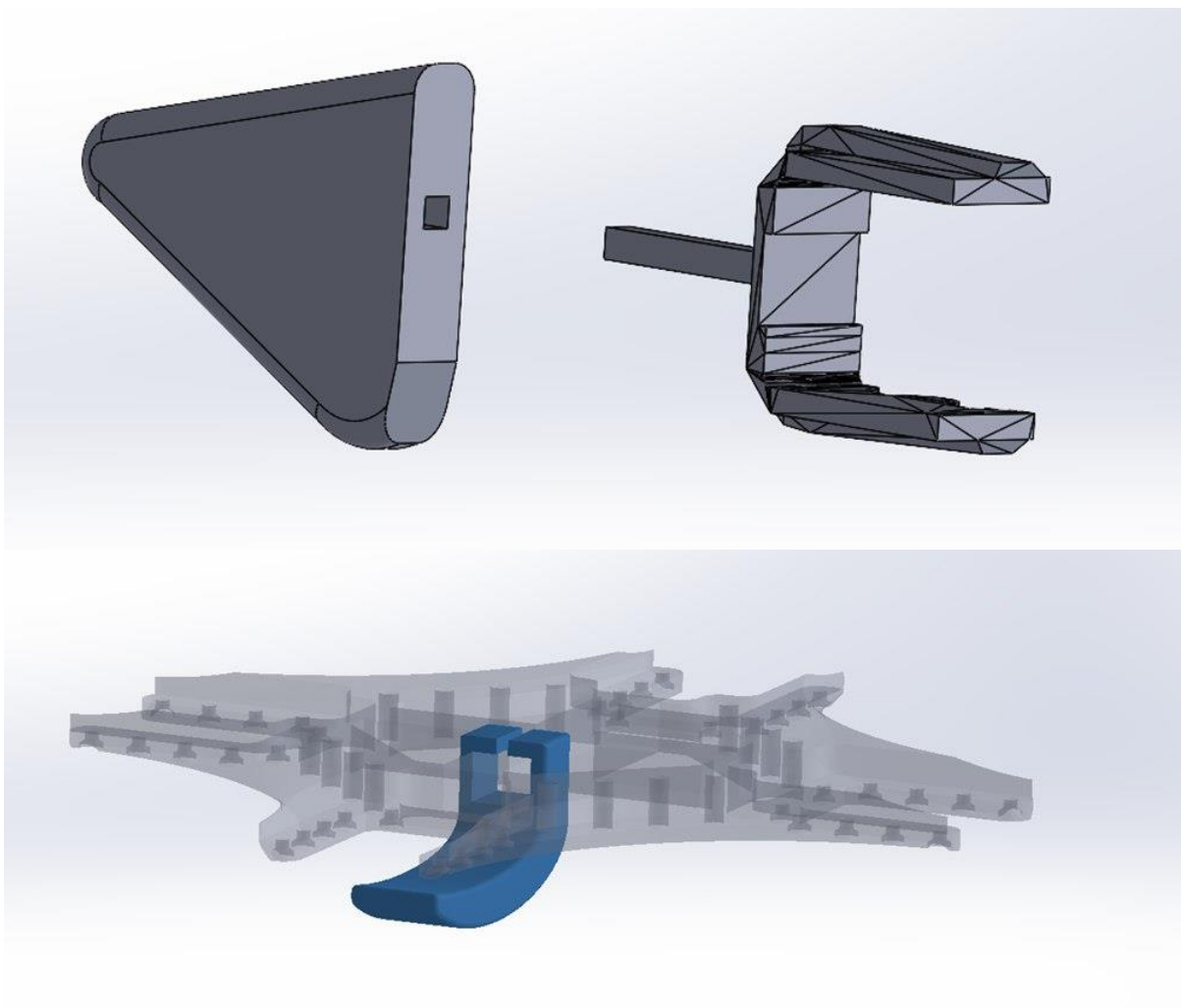


Figure 10: Unused leg designs

Soft Pieces Results

The settings used for the soft edge leg pieces produce sufficiently soft parts for the impact dampening use case. By examining the part, it was observed that the rounded front end, particularly the holes, is less flexible and may reduce the dampening gained from the soft piece. This reduced dampening is likely not an issue for this use case, but it can be considered in later designs.

Secondly, it was noted that when printing several soft pieces together, the printing speed should be reduced from the default setting to reduce printing defects.

The first iteration proved to be the best as it provides good ruggedness and an easy printing process. The first iteration does have the drawback of not being uniformly soft along the long edge due to the fastening mechanism. However, that will likely not be an issue as the long edge rarely contacts the ground. Instead, the rounded tip of the soft part will contact the ground. When examining the rounded tip, it was found that the shape increases the rigidity, which is not ideal as this will reduce the dampening effect. Although the dampening effect may be reduced, it is believed to be enough for the robot's purpose.

The final iteration of the soft leg piece has a softer rounded tip because of the hole that has been added. It also has a redesigned fastening mechanism, making the long edge softer. Both changes make the part more difficult to print and fasten to the hard piece because the new part needs support to be added for the print to succeed. The supports increase the number of printing defects and parts that need to be removed from print. This contributes to the most considerable drawback of the part, concerns regarding the durability. From visual inspections, it can be speculated that the final iteration of the soft leg part will likely break more easily around the rounded tip due to printing defects and the hole that was added.

In conclusion, the first iteration of the soft leg piece is likely the best overall, but the final iteration has better results in rigidity.

4.2.2. Body Assembly

The body assembly of the robot serves two purposes. Firstly, it provides a structural element connecting the four leg assemblies. Secondly, it houses the electronics that power and control the actuators. In addition, since the body is the only non-actuated part of the robot, its motion gives the best representation of the whole robot's motion. Thus, it is the best-suited part of the robot to be tracked by a motion tracking system.

The original design for the robot's body consisted of two identical 3D printed plates (see Figure 11). The plates were designed to be mounted on top of each other such that the inner actuators of each leg assembly were attached in between the plates. This construction left enough room for the power and control electronics to be placed between the plates in the centre of the robot. Next to the cutouts for each actuator, the plates had holes for routing the cables from each leg assembly to the inside of the robot's body. The outer edges of the plates were curved to enable each leg's inner joint to rotate 180° without hitting the plates. Additionally, there were four cutouts in the middle of the plate (presumably for reduced print time and material costs) and 16 mounting holes.

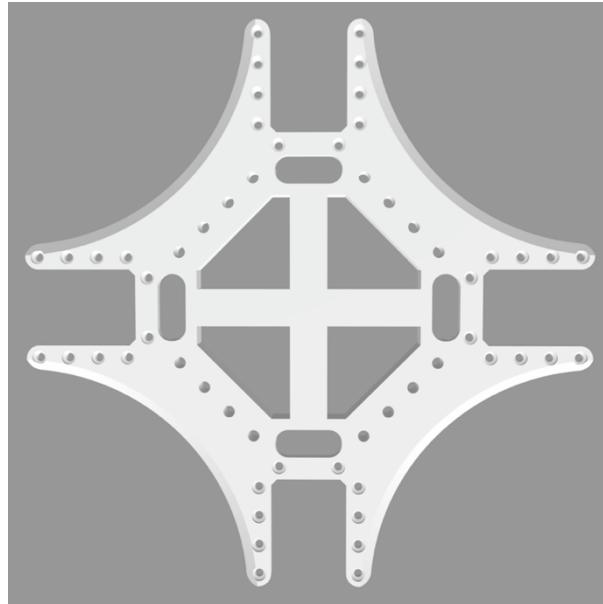
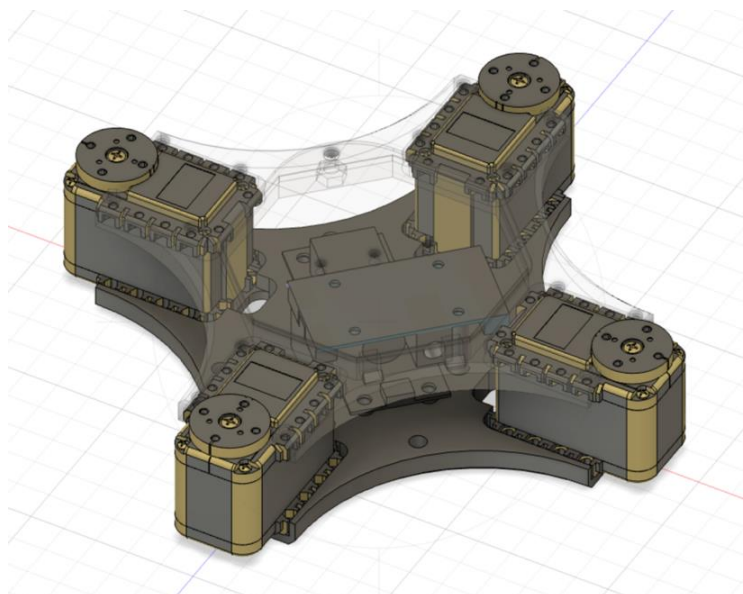


Figure 11. Original CAD design for the body plates.

After the initial construction and testing, the original body plates were proven functional. Once the body structure was assembled, it was sufficiently rigid, and there was enough room for the power and control boards. However, two issues were encountered that needed to be fixed. Firstly, the mounting holes in the original body plates did not align with any of the holes used in the power or control PCBs. Therefore, those boards could not be properly mounted with screws. It was found that the best way to mount the components to the original plates was to use zip-ties wrapped around the cross-shaped structure at the centre of each body plate. However, this method meant that the boards were located at the centre of the body, which made plugging in the power and USB cables difficult once both body plates were screwed down.

The second problem encountered was accessing the components inside the body. With the original design, the power and control boards and the wiring of the actuators could only be accessed by removing one of the body plates. Removing a body plate was unnecessarily time-consuming since each body plate was held down with 16 screws.

The original design issues were fixed by designing body assembly from scratch using CAD models of the control board, power board, and actuators (see Figure 12). A CAD model for the actuators



could be downloaded from the manufacturer's website, while the power and control boards had to be model by us since no existing models could be found online. By using CAD models of the components, different component layouts could be easily experimented with without manufacturing test parts for each iteration. Thus, this approach's major advantages were a high problem space exploration rate and the reduced need for physical prototypes.

The general idea of the original body structure was kept with the new design. With the new design, the plates have similar attachments for the actuators, similar curves on the outer edges, and the actuator cables run through similar holes. However, unlike the original plates, the new design uses different parts for the top and bottom plates (see Figure 13). This was required due to the addition of an access hatch and component mounting structures. The new design also adds parts for covering the hatch and attaching optional motion tracking markers.

The power and control boards were given mounting structures that allow them to be secured to the lower body plate with screws. Since the power PCB is positioned on top of the control board, it is mounted on stand-offs that ensure clearance between the two boards. The layout of the boards is carefully considered so that the DC power and USB ports are accessible from the same side of the robot. Additionally, the distance from the ports to the edge of the plate is minimized to make reaching the DC power and USB ports as convenient as possible.

The upper body plate got a more radical redesign compared to the lower plate. All the component mounting structures and cable run holes were replaced with an octagonal cutout for accessing the components inside the body. Additionally, a screw hole, a cutout for an M4 nut, and small chamfers

Figure 12. CAD model of the redesigned body assembly.

were added to attach a lid covering the cutout. The lid has small feet that go under the chamfers in the top plate which along with the screw on the opposite side prevent the lid from coming off. This mounting mechanism was chosen as it enabled the lid to be securely attached and removed with a single screw.

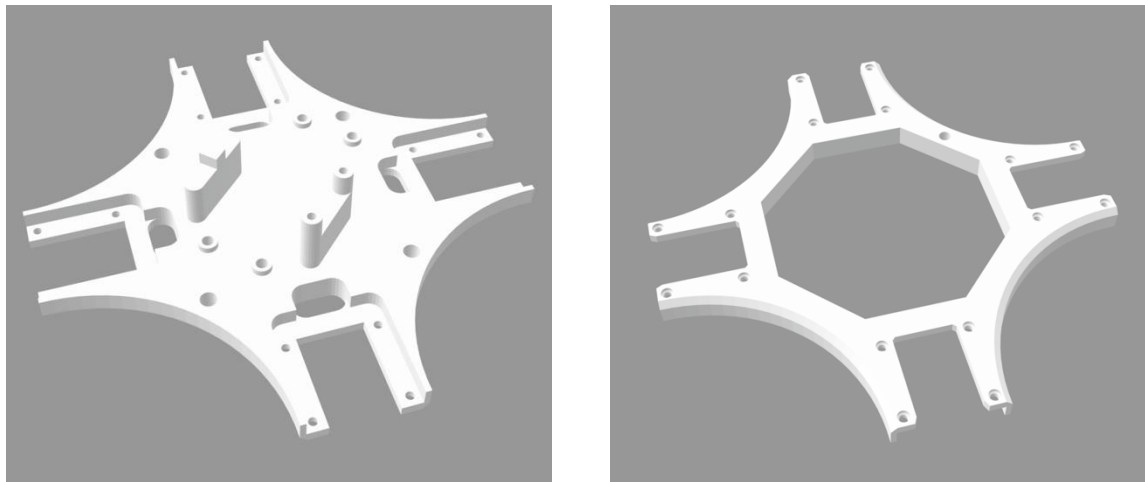


Figure 13. Redesigned lower (left) and upper (right) body plates.

The last addition made to the design of the body was a stand for motion tracking markers (see Figure 14). The stand is attached to the lid and its purpose is to elevate markers just enough that they will not interfere with the moving legs. For the type of markers the robot is used with (OptiTrack Rigid Body Marker) a 15mm elevation was sufficient. The stand is designed to be replaceable, so if another user would like to use different motion tracking system, they can easily replace the stand with another design that better suits their needs.

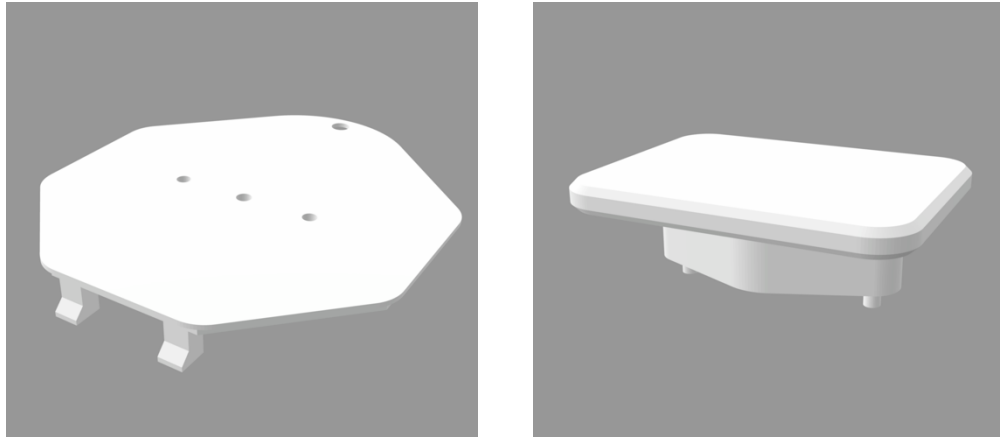


Figure 14. Lid (left) and stand for motion tracking markers (right).

4.3. Testing

4.3.1. Dynamixel servos

The robot components that have not been 3D printed are the DYNAMIXEL AX-12A servos with accompanying control board OpenCM9.04 and power supply connector SMPS2Dynamixel. After assembling the robot in the default Real-Ant configuration, the objective was to test the codes provided in the Real-Ant Project GitHub. The software setup of the robot is described in more detail in the next chapter. It was expected that testing the codes would be a quick task, as the default Real-Ant was assembled according to the existing instructions. However when running the *ant_send_cmd.py* file found in the Real-Ant GitHub, a simple script for moving each leg between two setpoints, unexpected behaviour was observed. The serial communication between the laptop and the robot would break, usually within tens of seconds, as the *ant_server.py* script would crash. Originally it was thought this could be a hardware issue, so the servos, TTL cables and control and power board were changed. The robot would still disconnect after these changes, so a software issue was suspected. Additionally, it was observed that only one of the legs moved according to the setpoints in *ant_send_cmd.py*, while the others moved between other positions.

These problems led to re-examining previous assumption that the AX-12A servos could be used as a plug-and-play device. It turned out that the AX-12A has internal non-volatile EEPROM for storing settings, such as the DYNAMIXEL ID. This ID is a number identifying the device (servo or control board) in the communication over the DYNAMIXEL bus, and it needs to be unique. It turned out that the robot had been disconnecting as all servos had been assigned the same ID by default, which caused problems with the communications. To fix this, new IDs needed to be set. The solution to servos not following the given setpoints was also found in the EEPROM control table: angle limits for the servo had been set which limited the movement of the servos, so by removing the limits, the expected motions were achieved.

Accessing and configuring the settings in the AX-12A control table was not straightforward but eventually it was done through functions in the Dynamixel2Arduino library, written in the Arduino IDE and uploaded to the OpenCM9.04 control board. The "Guide to Dynamixel AX-12A servos" explains how this is done (see appendix chapter 7.1). Whilst figuring out how to configure the servos, the U2D2 adapter was ordered for connecting to the servos directly via USB without having to use the OpenCM9.04 control board and the Arduino IDE. Using the adapter, the DYNAMIXEL Wizard 2.0 software was tested, which provides an extensive graphical user interface for accessing

and modifying servo parameters. It was concluded that the easiest way to use DYNAMIXEL servos is with this adapter and software, but they can also be configured using only the control board. Moreover, the U2D2 adapter lets Python code be run directly from a computer on a servo using the DynamixelSDK library, again without having to use the OpenCM9.04 control board. Instructions were documented for using the U2D2 adapter in the appendix chapter 7.2 How to use the U2D2 adapter with Dynamixel Wizard 2.0 and DynamixelSDK.

4.3.2. Python code

The default Real-Ant, from which the improved robot was developed from, comes with some code for executing commands on the control board and for creating a connection between the robot and a computer. The *ant11_cmd_dxl.ino* file is uploaded to the OpenCM9.04 control board and writes goal positions for each AX-12A servo based on the setpoints transmitted as a byte string from the computer. No modifications were done to this file. On the computer three python scripts are ran: *ant_server.py*, *ant_send_cmd.py*, and *antproxy.py*. The *ant_server.py* script is used to print messages sent back from the robot, *ant_send_cmd.py* is used to write commands to the robot, and *antproxy.py* is used to combine the sending and receiving of data in a single network proxy, i.e., connecting publishing and subscribing sockets. These scripts use the ZeroMQ library to set up sockets for network communication.

Out of these scripts only modifications to *ant_send_cmd.py* was done. This was initially a very simple script that only assigned setpoints for moving each leg between two positions, but from these basics two new scripts were developed that were used in testing: *tester.py* and *commander.py*. The *tester.py* script is used to test individual servos, selected by their DYNAMIXEL ID. The code contains functions for writing a position to a servo, testing the full motion range of a servo, and resetting the servos to their default positions. The script prints messages to the terminal to guide the user. The *commander.py* script uses the "keyboard" Python library to read user inputs for robot movement. It contains functions for pre-programmed linear and rotational motion. The script also calls other pre-programmed actions stored in the *move_set.py* file, e.g., for making the robot wave with a leg. The *move_set.py* file can easily be extended to contain new motions without having to change the main *commander.py* script. The linear and rotational movement was programmed based on the gait presented in Atique, Sarker & Ahad (2018), in which a similar robot was studied.

5. Reflection of the Project

5.1. Achieving Objectives

The objectives were to build a functioning Real-Ant robot and to be able to verify that the robot is in working condition. This part of the goal was achieved.

Additionally, the secondary objective of improving the robot was to make it better suited for RL, which mainly included soft parts. This also included testing that would be used to iteratively design better improvements. This part of the project was optional and was made flexible so that the extent of what was done could be readjusted depending on how well the project progressed. Because of this, the secondary objective was only met in part. The robot was improved with soft parts but did not test the designs as well as was planned. This was because the servos took longer than expected to get in working condition than expected (this is further explained in chapter 5.2 Timetable). Testing of the robot was also made harder as the RL testing environment was not set up. Both mentioned reasons resulted in the iterating of the soft parts being done with visual inspections. Improvements of the design (some explicitly mentioned) was also achieved. A major aspect of this was improving the body to make it easier to use. This included: a hatch for easier access of

components, a body plate made to fit the boards in use, and stand for the motion trackers. Secondly, improvements in documentation, found difficulties with building the robot and some problems in getting the servos to work. The documentation should make further use of the robot easier.

5.2. *Timetable*

The group managed to follow the timetable presented in the original project plan quite closely until the business aspects part of the course had been completed (work packages WP1 – WP4), but after this point some adjustments were made. In the original timetable it was estimated that the ordered robot parts (e.g. actuators and control board) for the standard Real-Ant would come by the 22nd of March. As the power board and switched-mode power supply arrived later than the actuators and control board, the assembly of the full robot got delayed by approximately a week.

As mentioned in chapter "4.3.1 Dynamixel servos", the DYNAMIXEL AX-12A servos turned out to be more complex than was initially expected, so they required additional configuring that was not scheduled for initially. This resulted in a phase of debugging and researching the servos, as well as writing documentation for how to use them for future reference, as described in chapter "4.3.1 Dynamixel servos", and in the appendix "7.1 Guide to Dynamixel AX-12A servos", which was not envisioned initially. In conjunction with this phase the U2D2 adapter was also ordered, tested, and documented to figure out the best way to work with the smart servos in the future, as described in the "7.2 How to use the U2D2 adapter with Dynamixel Wizard 2.0 and DynamixelSDK" appendix. Not all problems with the servos configurations were fully solved until the final week of April, so the goal of verifying working code on an assembled default Real-Ant by the 1st of April was heavily delayed.

Whilst debugging the problems with the servos and testing the code from the original Real-Ant it was noticed that there was a need for more advanced Python scripts for testing individual servos when building the robot. It was also identified a need for a Python script that let the robot move using pre-programmed actions. This was needed so that the robot could be showcased also outside of the lab that houses machine vision cameras for running reinforcement learning tests on the robot, e.g., on a stand in the final Gala. Development of this code was not something that had planned for initially, but it was prioritized over some of the superfluous iterative design and testing work packages that were initially scheduled. The coding started in conjunction with debugging the servos, as the need for easier testing of individual servos appeared (ca. 31.03) and continued with iterative improvements until preparations for the final Gala started (ca. 13.05).

The group worked flexibly as to re-prioritize and make time for debugging and creating new code. The work packages for iteratively testing and creating new designs (WP6 – WP8) for soft parts, and associated milestones, were scrapped, and these iterations were instead carried out as an extension of the initial WP for creating alternative parts (WP3). While working with the robot the group also identified improvement needs in the hard 3D-printed structure, not for increased robustness but rather for usability of the design. This resulted in the design and manufacture of new hard parts, as described in chapter "4.2.2 Body Assembly " of the results section, which is also work that had not been planned for initially.

In conclusion, the timetable from the initial project plan was only followed in the beginning, and the group flexibly adapted the plan and reprioritized work as new challenges and needs appeared. Creating a fully working version of the standard robot took more time than expected, and time was made for tasks like coding and redesigning the body, that was not initially planned for. On the other hand, more time was scheduled for designing and testing new robust parts initially than was needed, so less time was spent on this work when unexpected challenges appeared. During all work the

main goal given by the project instructor was kept in mind (building a working robot) and the group members were able to take responsibility for new tasks that appeared and work simultaneously on parallel tasks, due to the good communication within the group and with the instructor as explained in chapter "5.4 Project Meetings and Communication".

5.3. Risk analysis

The risk analysis proved to be quite accurate and comprehensive. The likelihood of multiple risks was even estimated to be a bit higher than what they actually were. The group were still well prepared for the risks in the planning phase, and many of them, if realized, would not have significantly affected the success of the project.

The risks that realized during the project were sick leave and the availability of 3D printers. The group coped with absences through flexible scheduling and good communication. The poorer-than-expected availability of 3D printers caused small schedule changes and rescheduling of tasks, which were not relevant to the project on a large scale.

5.4. Project Meetings and Communication

5.4.1. Project Meetings

A total of 17 scheduled meetings were held. The first was held on the 24th of January and the last 20th of May. Of course, this does not include one meeting after this document is made.

Agendas were made before each of these meetings, stressing what needed to be discussed to ensure the meetings were productive. The project manager was responsible for releasing the agenda before the meeting. All of them were released 24 hours before the meeting and no later than an hour before the meeting. The agenda was an editable word document to allow group members to add topics they would like to discuss before the meeting. This allowed everyone to add what they want without relying on the project manager.

The agenda contained the following general topics:

- Progress report on last week's action log.
- General progress report of the project and relating it to the project plan.
- Summary of what needs to be done next.

Each of these 17 meetings were documented in memos that summarized the contents of the meeting. The project manager (Jed Muff) documented all these meetings. The project manager used these memos to ensure the project was kept on track and everyone knew what they needed to do next.

The memos contained the following information:

- When: Date and time
- Place
- Attendance
- Agenda
- Outcomes of the meeting (What was discussed, what is left to discuss, what was decided, other notes)
- Action log

A template can be found in the appendix section 7.3 Meetings and minute template. The memos were always published within 24 hours after the meeting to ensure the information is the most accurate and detailed. Once complete, they were announced in the Teams chat to allow other group members to challenge, change or add information. All memos were uploaded to the shared Teams file manager (Meetings and Minutes).

Table 1 shows the summary of all meetings that took place before this document. Ticks represent the person attended. Crosses represent that the person was planning to come but couldn't for some reason. Blank spaces represent that a meeting took place, but that person was not intending on coming and it was agreed they didn't need to.

Attendance initials are defined as follows: Jed Muff (JeM), Eric Hannus (EH), Julius Mikala (JuM), Antti Sippola (AS), Jere Vepsä (JV), Rituraj Kaushik (RK)

Table 1 Summary of project meetings

#	Date	Time	Duration (mins)	Place	Attendance						Main topic
					JeM	EH	JuM	AS	JV	RK	
1	24/01/22	1300	20	Zoom	✓	✓	✓	✓	✓	✓	First meeting
2	25/01/22	1330	30	Zoom	✓	✓	✓	✓	✓	×	Discuss plan
3	27/01/22	1130	90	Zoom	✓	✓	✓	✓	✓	✓	Refine Scope, order form
4	03/02/22	1130	45	Zoom	✓	✓	✓	✓	✓	✓	Project plan, Refine scope
5	10/02/22	1500	50	TUAS	✓	✓	✓	✓	✓	✓	Establish facilities
6	17/02/22	1130	15	Zoom	✓	✓	✓	✓	✓	✓	Preliminary research update
7	24/02/22	1130	20	Zoom	✓	✓	✓	✓	×	×	3D printed parts
8	01/03/22	1730	15	Teams	✓	✓	✓	✓	✓		Business aspects
9	03/03/22	1100	35	Zoom	✓	✓	✓	✓	✓	✓	Business aspects
10	10/03/22	Day	-	TUAS	✓	✓	✓	✓	✓	×	Business aspects, ordered parts
11	17/03/22	1030	30	TUAS	✓	✓	✓	✓	✓	✓	Construction
12	24/03/22	1100	30	TUAS	✓	✓	✓	✓	✓	✓	Midterm evaluation, problems with actuators
13	31/03/22	1100	20	Zoom	✓	✓	✓	✓	×	✓	Construction and actuator problems
14	07/04/22	Day	-	TUAS	✓	✓	✓	✓	✓	✓	Actuator problems
15	21/04/22	1100	20	TUAS	✓	✓		✓	✓	✓	Divergence of plan
16	05/05/22	1400	30	Teams	✓	✓	✓	✓	✓	✓	Progress report, finishing last parts
17	20/05/22	1000	30	TUAS	✓	✓		✓		✓	Gala preparation, finalizing the project
		Total	480		100%	100%	100%	100%	88%	81%	

From Table 1 a total of 8 hours (480 minutes) was spent in meetings over a total of 17 weeks (1 meeting a week average). That gives an average of approximately 30 minutes per week. It was stated at the start in the project plan that "Meetings were crucial for communication within the group for adaptability and efficiency. However, having meetings too long and full of irrelevant detail can get in the way of productivity". A total of 8 hours over the course of the project justifies that this quote for meetings was adhered to.

Additionally, 10 of the 17 meetings were online (59%). The hybrid flexibility of meetings being online and in person made these meetings time effective for everyone.

Attendance was good all round, most of the times people were missing were either due to illness or just a mistake. However, this effected the project little as memos made catching up with what was missed easy.

Overall, this format for meeting management worked very well. Although a little more work for the project manager each week it made sure that the project was kept on track, and everyone knew what they needed to do. This also made the division of workload fairer. Everyone was happy with the outcomes of the meeting and no time was wasted.

5.4.2. Communication plan

The main communication channel was the Team's chat. This is where most communication with both the instructor and other team members took place. Within Teams, only two chats were used: 'General' and 'Meetings and Minutes'. 'General' serves as a place to communicate to everyone about general topics. 'Meetings and Minutes' serves primarily to keep memos of the meetings and document the times they are released, this saved clutter in the general chat.

A telegram group chat was set up as a less formal way of communication. This provided us with a quick and easy backup way for communicating within the group, and just in case people do not have Teams notifications, it served as a way of communicating urgent messages. The last way of communicating was via email. Email was primarily used for communicating externally to the group. If not everyone in the email chain was cc'd for any reason, it was the group member's responsibility to forward the essential information to the Teams chat.

5.5. Quality

Quality management was needed to deliver a project output that meets its requirements. The only way to ensure quality is through the customer, as the customer defines the specifications and what is acceptable. In the case of this project, the customer is the Intelligent Robotics Research group, and the point of contact for the research group is the instructor. Therefore, the instructor's role was to help ensure the project is heading to a product with good quality as defined by the Intelligent Robotics Research group. This was done through project meetings with progress reports each week. In the end, quality of the work produced was acceptable after some feedback.

Internally (without customer consultation), there were a couple of techniques used to ensure quality. These were primarily the responsibility of the project manager. However, everyone was responsible for the quality of the work they produced.

- If any group member observed problems in quality, they brought it up as soon as possible to the group to address the problem and act/change the plan accordingly.
- Because this was a small project, everyone in the group read through each critical document to ensure it is clear and everyone understood it. This allowed for clear understanding in the group and keeps an excellent quality of work. People should be free to critically review work and tell people what can be done to improve it.

One of the issues with working in a group is ensuring all work is up to the same standard. To ensure the maximum amount of work is done, all members were utilized to their fullest abilities. This meant everyone was given an equal amount of work. However, because people's skills are different,

this leads to a difference in the quality in certain tasks. This means not all work is of the best quality that the group can produce. To counter this, the group critically reviewed each other's work and provided feedback that worked towards a solution. This did not mean providing only criticisms but ways to improve the outcome. Everyone in the group acted well to feedback, keeping an open mind and working together to find a better solution.

6. Discussion and Conclusions

6.1. Success of the Project

The essential criteria for the success of the project (defined at the start of the project) are:

Essential Criteria	Status	Comments
Completing the project within the scheduled time.	✓	All objectives were fulfilled before the final deadline
Planned training is being conducted successfully with the appropriate teams.	✓	The main planned training was learning some CAD skills and 3D printing. Everyone learnt this.
Achieving the customer/client satisfaction target.	✓	The instructor Rituraj was pleased with the final outcome of the project.
Project handover is well documented and completed in the required manner.	✓	Everything is well documented in the github, although a proper handover has not taken place. In future the instructor may contact us with questions.
The project delivers all deliverables within the agreed scope.	✓	All objectives were fulfilled.
The Learning Goals of the project are achieved.	✓	See chapter 6.2 Group Member Personal Reflections

A working Real-Ant robot has been produced. The robot can be verified to be working correctly as it can run the codes available on the Real-Ant Git page and perform like other examples of the robot found on the internet (Real-Ant Introduction). Verification of the improvements happened through testing where points of failure are identified, and the changes are confirmed to improve the robustness.

6.2. Group Member Personal Reflections

In this section each group member reflects upon how the project went for them. In *italics* show what they stated at the start of the project and in regular font shows the reflection.

Antti Sippola: *I view this project as an opportunity to focus on the hardware side of robotics. This project will require skills in parts design, 3D printing, electronics, and microcontrollers. My goal is to develop my abilities in those areas as well as learn about project management and teamwork.*

Most of my efforts were focused on parts design and 3D printing. I gained a lot more confidence working with 3D printers. Especially, I learned to detect and fix common issues that lead to poor print quality and interrupted prints. On the parts design side, I learned about designing part assemblies and got familiar with Fusion 360 CAD program. I also learned about the limitations of 3D printing and how to factor in those limitations in the parts design process.

Eric Hannus: *My goals are to get practical experience of 3D printing, learn problem solving by identifying and fixing weaknesses in the Real-Ant design, get a glimpse of what kind of research is being done in fields that interest me like robotics (even though we will not be creating new RL code), and to learn more about working as part of a team and develop related skills.*

In this project I achieved the goals I had set in the beginning of the course: I got to design some CAD models in SOLIDWORKS, even though we did not end up using most of them ultimately, I learned how to use 3D printers, and I got a positive teamwork experience as the coordination and communication within the team was good. Moreover, I got valuable experience I had not anticipated at the beginning of the project from debugging the problems we had with the servos, and from getting familiar with the ROBOTIS servos, controllers and associated libraries and configuration tools. It was also nice to get to brush up my Python skills and work on the commander and servo tester scripts for the robot, and also to study the Python and Arduino code that was already given to us to try to understand how the various software parts fit together. Even though we did not do any reinforcement learning ourselves, it was still exciting to see the robot perform learned movements in the lab and see the motion tracking cameras that were even better than the grid board solution mentioned in the original RealAnt documentation.

Jed Muff: *My current skills relate primarily to programming and theoretical understanding, so I hope to expand my practical skills. These practical skills relate to some mechanical sides: 3D printing, CAD and Manufacturing, some practical robotics and artificial intelligence: seeing how robots are programmed and how reinforcement learning is introduced, and finally growing some project management skills and even some business sense. I am also looking forward to learning more about the intelligent robotics research group as they may heavily influence my future career.*

Looking back on my personal goals, I believe I have succeeded in most of them. The main practical skill I learned was CAD skills. As before I had some idea of what to do I am now a lot more confident with my CAD skills, although they could still be improved. Additionally, 3D printing is another skill I am now more confident with. Hard coding the robot was a good exercise and gave me a good experience in some very basic robotics. Not a lot of reinforcement learning was done but reflecting on this project will be useful for learning it in future. After this project, I am much more confident with my project management skills.

I look forward to working with the Intelligent Research robotics group in future.

Jere Vepsä: *I have some experience in 3D printing, CAD design and robotics. I will try to take advantage of this project to develop these skills. My main learning goal is to develop competencies related to carrying out this type of hardware assemble project and possibly learn something new about ROS also.*

I feel like I reached some of my goals better than others. During this project I learned about capabilities of 3D printing and using flexible materials such as TPU 95 and improved my skills with assembly, debugging and trouble shooting skills. I also learned about how to use python and python libraries in projects like this and what kind of programming skills are required to achieve a working robot.

Julius Mikala: *My goals include further learning how to use CAD if that is a possibility, as well as learning how to assemble and configure hardware such as this. I also believe that learning how to work in a group project of this size will prove instructive. Lastly, I'm excited to see RL techniques at work.*

I believe I achieved most of my personal goals. The main part of this was learning how to use cad programs as well as getting a better understanding in how 3D printing is done. I also believe that the main part of my responsibilities, soft part design and printing, gave me a look into product

development as my design process had several iterations. The group work was a refreshing change to how group work has previously been done. Having a clearer work distribution in combination with a group manager made the group work proceed much more smoothly than it would have otherwise. I believe that what I learned about teamwork can be useful further on in my studies or career. This project did not include much RL so that part of my goals was not met, but that isn't a big since it would only have been a nice addition to the content learned during this course. Lastly, I feel like I learned a lot about the importance of documentation, how to read documentation and datasheets, as well as trouble shooting skills.

6.3. Conclusion and Future Work

The robot was successfully built according to the original Real-Ant design by Ote Robotics. Then a new design with improved robustness and usability was developed. This design included soft parts from TPU 95A for absorbing shocks and vibrations, a hatch for easier access to the control and power board and a stand for attaching, e.g., reflectors used in machine vision for reinforcement learning. Testing proved that the built robot works in the RL setup it was designed for. Additionally, pre-programmed movement and actions were designed so that the robot can perform simple motion without needing to set up the more extensive RL setup.

Future work with this robot could include making it wireless using Bluetooth communications and a battery as a power source, which was a possibility mentioned in Boney et al., 2020. As the documentation for the OpenCM9.04 control board was studied, and descriptions of how to connect a battery, there seems to be useful documentation available already to start working from if one were to implement this improvement.

7. List of Appendixes

7.1. Guide to Dynamixel AX-12A servos

The AX-12A Control Table

The AX-12A servos contain onboard memory, allowing the user to read the status of the motor or write configuration settings to it. This data is presented in the Control Table, which is divided in two fields: one for data stored in the non-volatile EEPROM and one for data stored in the volatile RAM. The control table, which is presented in the online e-Manual, also contains information about the default settings, as well as whether the address is read-write or read-only. Next some especially important control table addresses in the EEPROM are presented, which need to be considered when configuring the AX-12A servos, e.g., in the Real-Ant robot. Below is a subset of the table presented in the e-Manual:

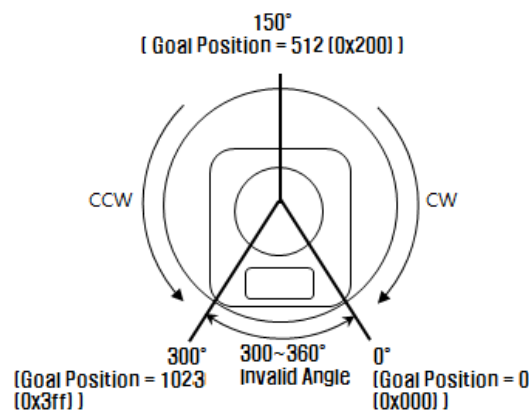
Address	Size (Byte)	Data Name	Description	Access	Initial value
3	1	ID	DYNAMIXEL ID	RW	1
4	1	Baud Rate	Communication Speed	RW	1
6	2	CW Angle Limit	Clockwise Angle Limit	RW	0

8	2	CCW Angle Limit	Counter-Clockwise Angle Limit	RW	1023
---	---	-----------------	-------------------------------	----	------

The DYNAMIXEL ID is used to identify the device in the network on the Dynamixel bus, and must thus be a **unique value** for each device. Thus, the default ID of 1 must be changed if multiple AX-12 servos are used. **The DYNAMIXEL ID 200 is the default ID for the OpenCM9.04 controller and should thus not be assigned to any other device.**

The Baud Rate determines the communication speed on the Dynamixel network (servos and a control board), and the initial value of 1 corresponds to 1 000 000 bps. For the baud rate of other values see the e-Manual.

The clockwise and counter-clockwise angle limits are used to limit the rotation range. The initial values of 0 and 1023 should not limit the joint motion, but if the servos for some reason have restricted motion when they are taken in use, this parameter should be checked and changed if necessary. See picture from e-Manual below for angles corresponding to range 0 – 1023:



Accessing the Control Table via the OpenCM9.04 controller

The OpenCM9.04 is a microcontroller used in conjunction with the Dynamixel servos (other options are also possible via e.g. standard Arduino controllers and expansion boards). If a OpenCM9.04 "A Type" board is acquired, it will have no buses connected, so the "OpenCM9.04 - Connectors and Accessory Set" needs to be acquired and the Dynamixel bus connectors need to be soldered manually. The AX-12A servos also need **external power** (while the OpenCM9.04 itself can be powered via USB), so this needs to be arranged for the controller e.g., via the SMPS2Dynamixel adapter. See the OpenCM9.04 e-Manual for instructions for how to solder the connectors.

The OpenCM9.04 is programmed via the Arduino IDE, and functions for the Dynamixel servos are found in the Dynamixel2Arduino Library. The OpenCM9.04 e-Manual contains detailed instructions how to configure the Arduino IDE and install the library, and the library contains useful example code under File->Examples->Dynamixel2Arduino in the IDE, but below some functions of the library needed to configure the Dynamixels are highlighted.

The example codes start off with code that needs to be altered depending on the hardware used. Pay attention to the comments; if the OpenCM904 board is used without any expansion boards the examples will have to be edited so that serial is Serial1 (not Serial3 as by default) and DXL_DIR_PIN is 28 (not 22 as by default). The correct ID also needs to be selected, if it has been changed to some other ID from the default value of 1. Finally, the correct Dynamixel protocol version must be selected (for the Real-Ant project protocol version 1.0 was used).

```
//Board settings
#define DXL_SERIAL Serial1 //OpenCM9.04 EXP Board's DXL port Serial. (Serial1 for the DXL port on the OpenCM 9.04 board)
#define DEBUG_SERIAL Serial
const uint8_t DXL_DIR_PIN = 28; //OpenCM9.04 EXP Board's DIR PIN. (28 for the DXL port on the OpenCM 9.04 board)

const uint8_t DXL_ID = 6;
const float DXL_PROTOCOL_VERSION = 1.0;
```

When the communication is started by calling `dxl.begin()`, this function takes the baudrate as its input. In the examples the baudrate is 57600 bps, but if the Dynamixel are used with their default settings the baudrate will be 1 000 000 bps, so it needs to be changed when calling the function as pictured below:

```
dxl.begin(1000000);
```

The `dxl.setID()` can be used to set a new ID (New_ID in the picture below) for the Dynamixel servo, provided the current ID (DXL_ID in the picture below) is known. AX-12A servos should have ID = 1 when coming from the factory, but if previously used servos are used, look at the label in case the previous user has written down a new ID.

```
//Setting new ID
dxl.setID(DXL_ID, New_ID);
```

Using the `dxl.setOperatingMode()` function we set the mode to "OP_POSITION", i.e. position mode, which also reset the CW and CCW Angle limits to 0 and 1023 respectively.

```
// Turn off torque when configuring items in EEPROM area
dxl.torqueOff(DXL_ID);
dxl.setOperatingMode(DXL_ID, OP_POSITION);
dxl.torqueOn(DXL_ID);
```

The current value of any Control Table address can be read and printed to the Arduino Serial monitor (remember to use correct baudrate in the monitor!) this way:

```
//Memory read test
//ID, CW & CCW limits
DEBUG_SERIAL.print("Current (new) ID:");
DEBUG_SERIAL.print(dxl.readControlTableItem(ID, New_ID));
DEBUG_SERIAL.print(" CW:");
DEBUG_SERIAL.print(dxl.readControlTableItem(CW_ANGLE_LIMIT, New_ID));
DEBUG_SERIAL.print(" CCW:");
DEBUG_SERIAL.print(dxl.readControlTableItem(CCW_ANGLE_LIMIT, New_ID));
DEBUG_SERIAL.print("\n");
```

Other address parameter names than ID, CW_ANGLE_LIMIT and CCW_ANGLE_LIMIT presented above can be found in the OpenCM9.04 e-Manual under 8.4.1.1 Dynamixel2Arduino Class > ReadControlTableItem(), or in the following link:

https://emanual.robotis.com/docs/en/popup/arduino_api/readControlTableItem/.

If writing directly to the control table is needed, e.g. to set some CW and CCW limits if full range of motion is not desired, a similar function called `writeControlTableItem()` can be used. This function can also be found in the OpenCM9.04 e-Manual under 8.4.1.1 Dynamixel2ArduinoClass > WriteControlTableItem(), or in the following link:

https://emanual.robotis.com/docs/en/popup/arduino_api/writeControlTableItem/

Sources:

AX-12A e-Manual: <https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/>

OpenCM9.04 e-Manual: <https://emanual.robotis.com/docs/en/parts/controller/opencm904/>

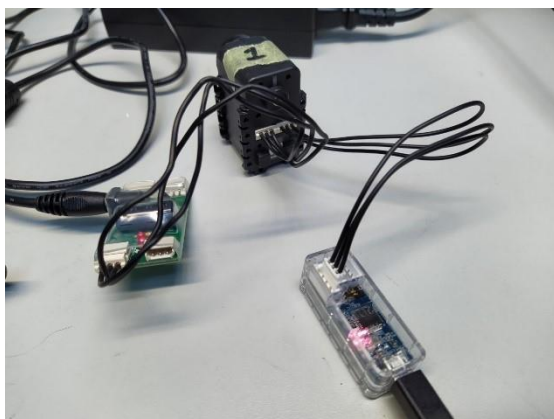
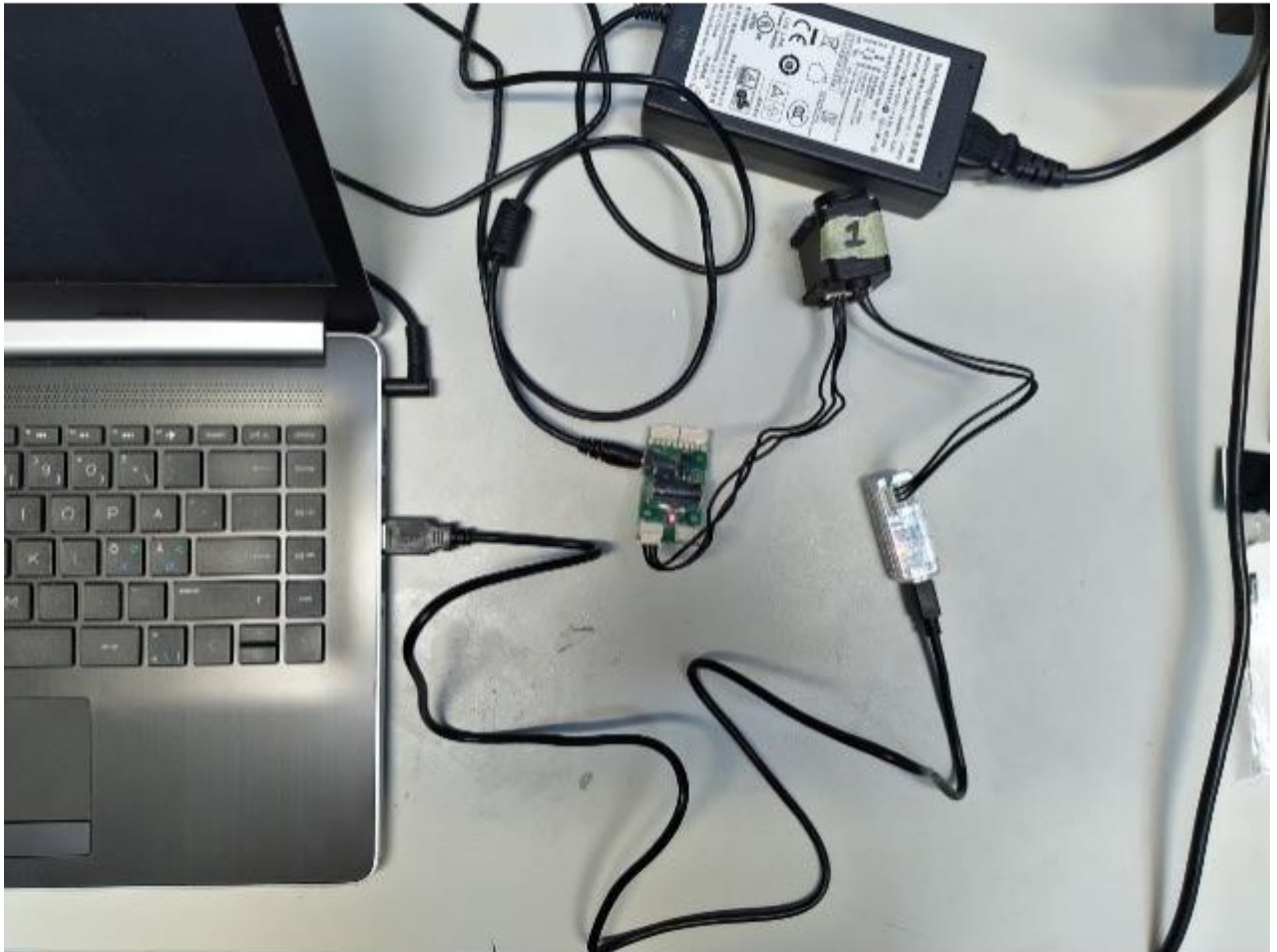
OpenCM9.04 Accessory Set: <https://www.robotis.us/opencm9-04-connectors-and-accessory-set/>

SMPS2Dynamixel adapter: <https://www.robotis.us/smps2dynamixel/>

7.2. How to use the U2D2 adapter with Dynamixel Wizard 2.0 and DynamixelSDK

Connections:

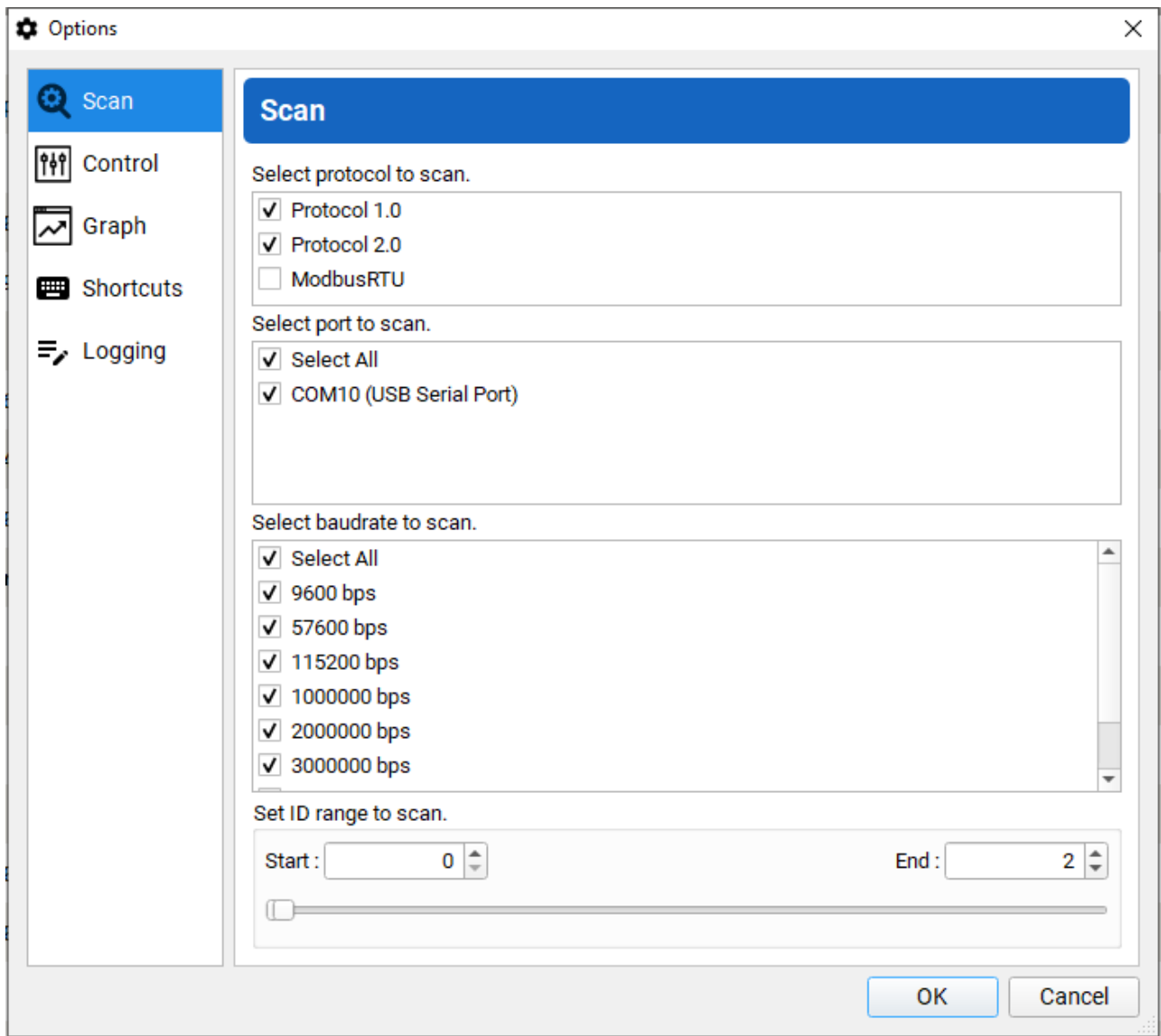
1. Connect the U2D2 to the computer via the provided USB-MicroUSB cable.
2. Connect the U2D2 to the Dynamixel AX-12A servo using the 3-pin TTL cable.
3. Power the servo(s) with a switched-mode power supply (SMPS), via the SMPS2Dynamixel board, using another 3-pin cable.



Configuring the servo in Dynamixel Wizard 2.0

Install the DynamixelWizard 2.0 program following the instructions for your operating system in https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/.

In the wizard click options and select for which settings you want to scan for servos:



For example, the range of baud rates or IDs can be reduced for faster if you know what you are looking for. Scan for Dynamixel. When the servo(s) you have connected are found you will see a list of Dynamixel to the left and the settings of the chosen servo in the middle of the screen:

The screenshot shows the DYNAMIXEL Wizard 2.0 interface. The left sidebar shows the device tree with 'COM10' and 'AX-12A' selected. The main window displays a list of parameters for the AX-12A servo. The 'ID' parameter is highlighted in blue. The right sidebar shows the 'AX-12A' section with a dial for position and various status indicators.

Address	Item	Decimal	Hex	Actual
0	Model Number	12	0x000C	AX-12A
2	Firmware Version	24	0x10	
3	ID	1	0x01	ID 1
4	Baud Rate (Bus)	1	0x01	1M bps = 2M / (1 + 1)
5	Return Delay Time	250	0xFA	500 [µsec]
6	CW Angle Limit	0	0x0000	0.00 [°]
8	CCW Angle Limit	1023	0x3FF	299.71 [°]
11	Temperature Limit	70	0x46	70 [°C]
12	Min Voltage Limit	60	0x3C	6.00 [V]
13	Max Voltage Limit	140	0x8C	14.00 [V]
14	Max Torque	1023	0x3FF	100.00 [%]
16	Status Return Level	2	0x02	Return for all
17	Alarm LED	36	0x24	
18	Shutdown	36	0x24	
24	Torque Enable	0	0x00	OFF
25	LED	0	0x00	OFF
26	CW Compliance Margin	1	0x01	0.29 [°]
27	CCW Compliance Margin	1	0x01	0.29 [°]
28	CW Compliance Slope	32	0x20	
29	CCW Compliance Slope	32	0x20	
30	Goal Position	2	0x0002	0.59 [°]
32	Moving Speed	0	0x0000	0.00 [rev/min]
34	Torque Limit	1023	0x3FF	100.00 [%]
36	Present Position	2	0x0002	0.59 [°]

To change a parameter, select the parameter from the list, select a value from the list opening in the bottom right corner, and press "Save". The following picture shows changing the ID to 2. Other parameter settings can be changed in a similar way.

The screenshot shows the DYNAMIXEL Wizard 2.0 interface with the 'ID' parameter selected. The 'ID' dropdown menu is open, showing a list of IDs from 0 to 22. The 'ID' parameter is highlighted in blue.

Address	Item	Decimal	Hex	Actual
0	Model Number	12	0x000C	AX-12A
2	Firmware Version	24	0x10	
3	ID	1	0x01	ID 1
4	Baud Rate (Bus)	1	0x01	1M bps = 2M / (1 + 1)
5	Return Delay Time	250	0xFA	500 [µsec]
6	CW Angle Limit	0	0x0000	0.00 [°]
8	CCW Angle Limit	1023	0x3FF	299.71 [°]
11	Temperature Limit	70	0x46	70 [°C]
12	Min Voltage Limit	60	0x3C	6.00 [V]
13	Max Voltage Limit	140	0x8C	14.00 [V]
14	Max Torque	1023	0x3FF	100.00 [%]
16	Status Return Level	2	0x02	Return for all
17	Alarm LED	36	0x24	
18	Shutdown	36	0x24	
24	Torque Enable	0	0x00	OFF
25	LED	0	0x00	OFF
26	CW Compliance Margin	1	0x01	0.29 [°]
27	CCW Compliance Margin	1	0x01	0.29 [°]
28	CW Compliance Slope	32	0x20	
29	CCW Compliance Slope	32	0x20	
30	Goal Position	2	0x0002	0.59 [°]
32	Moving Speed	0	0x0000	0.00 [rev/min]
34	Torque Limit	1023	0x3FF	100.00 [%]
36	Present Position	2	0x0002	0.59 [°]

In the **top right corner**, you can turn the indicator led on the servo on and off, and you can **rotate the servo to a chosen position by clicking on the black/grey/red dial**.

Using the DynamixelSDK library

Using the U2D2 you can directly run Python code for the Dynamixel servos. Example code can be found in the DynamixelSDK GitHub <https://github.com/ROBOTIS-GIT/DynamixelSDK>. For example, the **DynamixelSDK > python > tests > protocol1_0 > read_write.py** example can be used to get the servo

rotating between the minimum and maximum position set in the code. However, first the correct parameters for the AX-12A model servo must be set by modifying the code:

ADDR_MX_TORQUE_ENABLE = 24

ADDR_MX_GOAL_POSITION = 30

ADDR_MX_PRESENT_POSITION = 36

DXL_ID = 1 #or something else if it has been changed

BAUDRATE = 1000000 #or e.g. 57600 if it has been changed

DXL_MAXIMUM_POSITION_VALUE = 1023 #changed from 4000, as 1023 is the maximum for AX-12A

DEVICENAME = 'COM10' #or something else depending on the port and OS used in the computer

To use the DynamixelSDK examples you must install the library, for example by running "**pip install dynamixel_sdk**" on Windows.

For more information see:

U2D2 e-Manual: <https://emanual.robotis.com/docs/en/parts/interface/u2d2/>

SMPS2Dynamixel: <https://www.robotis.us/smps2dynamixel/>

Dynamixel Wizard 2.0 e-Manual:

https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

DynamixelSDK: <https://github.com/ROBOTIS-GIT/DynamixelSDK>

7.3. Meetings and minute template

Date:	Time:	Place:
-------	-------	--------

Attendance

Attended: Jed Muff (JeM), Eric Hannus (EH), Julius Mikala (JuM), Antti Sippola (AS), Jere Vepsä (JV), Rituraj Kaushik (RK)

Apologies:

Missing:

Agenda

1. Approval of previous minutes.
2. Session aims
3. Progress log:
 - a. Last week:
 - i.
 - b. This week:
 - i.
 - c. Next week:
 - i.

Outcomes:

Action Log

Action to be taken	Who is responsible	Deadline

--	--	--

References

Boney, R., Sainio, J., Kaivola, M., Solin, A. & Kannala, J., 2020. RealAnt: An Open-Source Low-Cost Quadruped for Research in Real-World Reinforcement Learning. arXiv. DOI: <https://doi.org/10.48550/ARXIV.2011.03085>

Atique, M.U., Sarker, R.I. & Ahad, A.R., 2018. Development of an 8DOF quadruped robot and implementation of Inverse Kinematics using Denavit-Hartenberg convention. Heliyon, Vol. 4(12). DOI: <https://doi.org/10.1016/j.heliyon.2018.e01053>