Master Thesis

Robot Systems 2019

MORF

Modular Robot Framework

Mathias Thor
mthor13@student.sdu.dk
331855

Supervisors:
Poramate Manoonpong & Jørgen Christian Larsen

Project period: September 1st 2017 - February 14th 2019

Abstract

The advantage of walking robots when compared to robots on wheels is that they can interact with generic physical environments that are either designed for legged motion or complex terrain filled with obstacles. Current solutions to adaptive locomotion for legged robots are promising, but often ineffective and far from able to compete with the behaviors of real animals. This is presumably because the benefits of using legs most often are overshadowed by the high design complexity. Hence, there is a need for a platform which enables researchers to start working on the actual locomotion controller faster.

In this work, we present MORF, a MOdular Robot Framework. The framework is intended for a wide range of research studies and is aiming at being easy and convenient to use. The framework consists of a modular multi-legged robot and a software suite. The legged robot, also called MORF, is modular as it defines standards that can be used for reconfiguring, extending, and replacing parts (e.g., body shape). The software suite includes simulations of MORF and hardware interfacing software based on the Robot Operating System. The framework is developed and validated based on a thorough analysis of both existing methods and technical issues.

When compared to other modular robot frameworks, MORF is advantageous in areas like processing power, mobility, controllability, completeness (includes a software suite), sensory feedback, and expandability, but lacks an easy mechanism for connecting parts together (e.g., using magnets or threaded collars). This will have to be improved upon in future revisions of MORF to stress and improve the modularity of MORF even more.

Preface

In this thesis, I present a robotics project conducted at the University of Southern Denmark (SDU) under the Centre for BioRobotics at the Maersk McKinney Moller Institute. The thesis is conducted under the so-called 4+4 educational scheme offered by the Danish educational system. This scheme allows students to begin a Ph.D. project before finishing the Master's programme. This way it is possible for the student to do their Master project in synergy with the Ph.D. project. However, the thesis should still be seen as a stand-alone Master project reported as a 45-ECTS Master Thesis in Robotic Systems.

The MOdular Robot Framework (MORF), presented in this thesis, will subsequently be used in my Ph.D. project as a tool for testing and verifying my control algorithms. Besides the framework implemented in this thesis two copies of MORF has been made. One for Vidyasirimedhi Institute of Science and Technology (VISTEC) in Thailand and another for a Ph.D. project on spiking neural networks at SDU.

MORF has already drawn much attention. It was presented for the first time at the robot exhibition R-18 in Odense where industry found a vast amount of use-cases for MORF. It was afterward featured in a television segment from the exhibition in the local news. Later it was presented at Thinkers50, also in Odense, and the MORF copy at VISTEC in Thailand was displayed at an opening ceremony of their new institute. MORF has also made its presence in China, where I personally presented the robot. Finally, MORF was officially presented at the IYCBE2018 conference at SDU where it won the best student paper award and afterward featured in the conference proceedings.

Besides this written report, a zip package containing supplementary material is also handed in. The supplementary material includes high-resolution images from the report as well as working drawings, CAD models, and parts list of the hardware parts used to build the robot.

Finally, I would like to thank my supervisors for support and inspiration, my colleagues for competent feedback and help, and my family for their invaluable loving support.

Contents

Chapter 1

Introduction

Every mobile robot needs some mechanism to make it able to move. The most frequently used one is wheels, as their simplicity and low power consumption is appealing. However, wheeled robots are limited to paved or predominantly flat surfaces with few obstacles. This is a significant drawback, especially when considering that over 50% of the earth's surface is inaccessible to traditional vehicles [1]. Finally, wheeled robots are in need of additional mechanisms if manipulative tasks are to be performed as well. In other words, the world must often change to fit wheeled robots and not the other way around.

An alternative solution is legs. A legged robot needs legs with at least two degrees of freedom to move, one for lifting and one for swinging, but are usually equipped with legs that have three to allow additional maneuvering. This increases power consumption and requires a more complex controller due to the complexity of the body structure [2, 3]. The question thus remains; why use legs at all? Legged robots are first of all able to interact with generic physical environments that are either designed for legged locomotion (humans) or complex terrain filled with obstacles [3]. Moreover, most animals are equipped with legs they use for all sorts of fascinating tasks spanning from manipulating objects to chasing prey at high speed through harsh terrain. We thus know the limitations and possibilities of using legs, and from where to get inspired.

Over the last two decades, the research and development of legged robots have grown steadily [1]. A more recent development is that of reconfigurable robots [4] where either the user or the robot itself can change the robots morphology [5]. This trait makes the system more versatile and scalable to the task at hand [6, 7]. Besides a physical body, a brain/controller is also needed to make a legged robot walk[1]. Current solutions to adaptive locomotion control for legged robots are

[1] Note that this is not the case for passive dynamic walkers that are capable of walking down an incline without any actuation or control.

promising but often ineffective and far from able to compete with the behaviors of real animals. This is presumably because the benefits of using legs most often are overshadowed by their high design complexity. This can among other be seen from the fact that many types of research on legged locomotion only includes simulations and rarely contains real-world experiments [8, 9, 10, 11]. There is thus a need for a platform which enables researchers to get started with the actual locomotion controller faster.

For these reasons, we present MORF, a MOdular Robot Framework. The primary goal of MORF is for it to be applicable in a wide range of research studies while still being easy and convenient to use, such that researchers can focus on the controller of the robot and not the hardware. The framework consists of a modular multi-legged robot (as shown on the front page in different configurations) and a software suite. The design of the legged robot, also called MORF, makes use of state-of-the-art components for high performance as well as kinematics and embodied solutions inspired by nature. This enables some of the complexity to be moved from the controller to the mechanics of the system. MORF is modular as it defines standards that can be used for reconfiguring, extending, and replacing parts of the robot, e.g., body shape. When using its default components, it is possible to configure MORF as both an insect or a mammal. The software suite includes a full simulation of the physical robot in different configurations as well as hardware interfacing software based on the Robot Operating System (ROS). The simulations will allow researchers and students to work with MORF even when the physical system is not present and it is furthermore a great advantage when doing machine learning and bio-inspired control (e.g., evolutionary robotics). The fact that suite is based on ROS also makes it easy for the user to quickly test their code and to interface with the physical system using any language compatible with ROS. Additionally, the software suite includes a simple locomotion controller, which can be used as a template for other controllers.

Designing a legged-robot is far from trivial as a wide range of design possibilities exists. We will start by reviewing some existing modular legged robots, including their advantages and disadvantages (Chap. 2). A set of key features or high-level requirements will then be defined, which describes "*what*" features MORF should include (Chap. 3). These features will be used to design and analyze the required hardware and software of the framework (Chap. 4). The resulting designs will then be verified as to whether they meet the key features (end of Chap. 4) and later used for an elaborated requirement specification, specifying "*how*" to achieve the desired framework (Chap. 5). Finally, MORF will be implemented (Chap. 6) and validated against the requirements (Chap. 7).

Note that this thesis will mainly elaborate on the construction of the physical robot together with the software suite for controlling and interfacing with the hardware. It will not go deep into the control of the robot, as it would surpass the scope of a Master of Science thesis.

Chapter 2

Related works

In this chapter, recent contributions to the field of modular legged robots will be described. Each section will start with a general description followed by the advantages and disadvantages of the modular robot in focus. Note that the list is not complete, but covers most of the resent and more popular robots. For more general reviews of legged robots including their history we refer to [12], [13], and [14].

2.1 Snapbot

In [6] Kim et al. presented a modular legged robot called Snapbot as shown in Fig. 2.1. The body of Snapbot houses an OpenCM9.04 micro-controller and an 800mAh lithium-ion battery for untethered operation. Snapbot has three types of modular legs with different kinematics. All legs use two Dynamixel XL-320 servos and can be connected to the body using a magnetic interlocking system with multi-pin spring-loaded electrical connectors for power and data. When attaching or detaching legs the system/controller automatically identifies its configuration using only internal sensors in the servos.

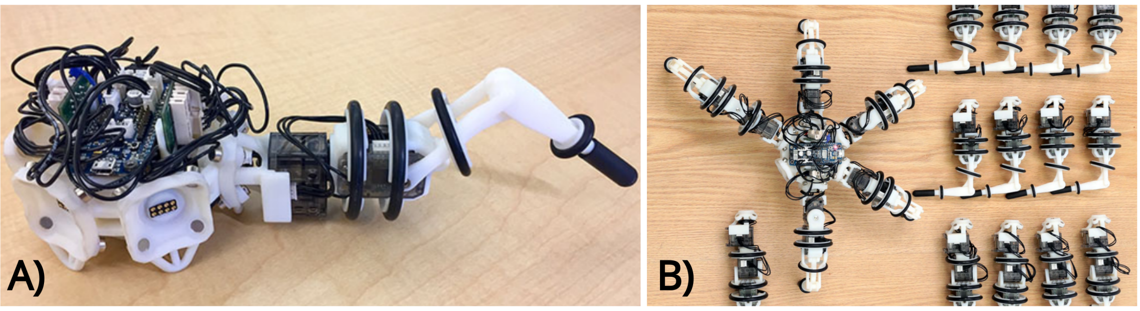


Figure 2.1 – The Snapbot robot by Kim et al. [6]. **A)** shows a side view of Snapbot with one leg attached. **B)** shows a top view of Snapbot as well as the three different leg types. Both images are from [6].

2.1.1 Advantages

Snapbot is, as described above, extremely easy to reconfigure and requires no external wires when operating. It is also fairly lightweight and compact. Finally, it comes with three different leg types resulting in a total of 700 different configurations.

2.1.2 Disadvantages

The onboard OpenCM9.04 micro-controller board is based on the 32bit ARM Cortex-M3. This is a light-weight controller which limits the complexity of the locomotion controller that may be used. It also requires a physical connection (i.e., cable) whenever the user wants communicate with the controller, as the micro-controller does not include any module for wireless communication. The result is a less accessible system with constrained movement and flexibility.

The Dynamixel XL-320s are fast (114 RPM), light-weight (16.7g), and cheap (21.90$). However, when compared to other Dynamixel servos of the same series (e.g., the XM or MX series) the XL-320s suffers from reduced precision (0.29°), controllability (only position controlled), range of movement (0°$\sim$ 300°), stall torque (0.39Nm), and robustness (made from plastic with LEGO-like connectors). The body of Snapbot is likewise built from less robust materials like plastic (3D print) and rubber which reduces durability and may increase maintenance.

Snapbot is only equipped with the sensors inside the servos (position, velocity, current/torque, and temperature). The authors state that they want to add additional sensors, but due to the limitations of the onboard controller this may prove difficult and even impossible to do without the use of connections (i.e., wires) to offboard devices.

Finally, the authors do not mention anything about a software suite for Snapbot (i.e., no simulation).

2.2 Modular platforms for advanced inspection, locomotion, and manipulation

In [7] Ansari et al. presented a series of physically robust hardware modules (made from aluminum) that can be configured into a variety of robots morphologies like a snake, robot arm, and legged robots (see Fig. 2.2). The main hardware module (also called the actuator module) consists of various sensors and a high-performance actuator (7Nm and 33 RPM at 48V) with a series elastic element to sense and control interaction forces. This actuator module was first presented in [15] by Rollinson et al. where it was used in a snake robot. Other kinds of modules are either dedicated sensors (e.g., vision) or passive interfaces that provide structure and allow external connections (e.g., the feet and body modules).

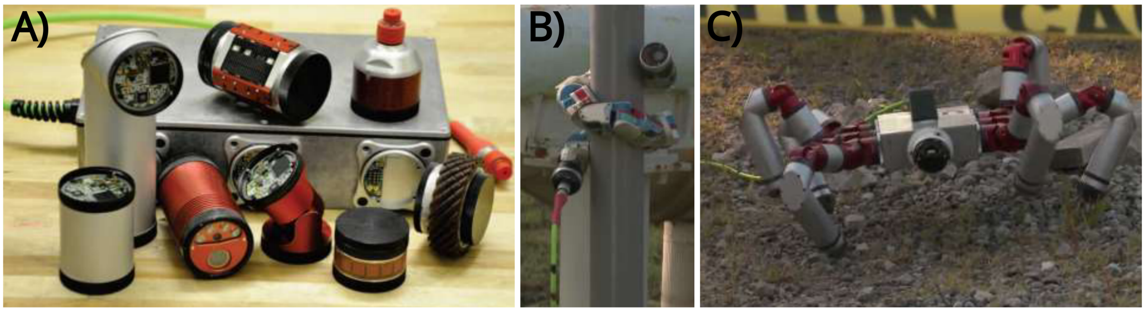


Figure 2.2 – The hardware modules by Ansari et al. [7]. **A)** shows the hardware modules when dissembled. **B)** shows when the modules are configured into a snake robot. **C)** shows when the modules are configured into a six-legged robot. All images are from [7].

2.2.1 Advantages

The different modules make it easy to create different robust and complex structures using their threaded connectors. Each actuator module contains not only sensors similar to other high-end servos (i.e., position, velocity, torque, and temperature) but also a 3-axis Accelerometer and a 3-

axis Gyro. Finally, the well-defined interface between modules adds to the simplicity of designing additional ones.

2.2.2 Disadvantages

While the design of additional modules may be trivial, the production and prototyping are not. This is because threads are not easy and often impossible to make using a 3D printer. New modules are therefore expensive to make as they have to be made from metals.

The actuator modules are relatively heavy and long with a weight of 205g and length of 6.4cm (longer modules may use more torque). The modules furthermore need 48V as input voltage resulting in the need of a 13 celled LiPo battery in order to make the robot truly mobile. This kind of battery is highly uncommon and also heavy. It is believed that this is the main reason why the platform uses a wire to an external energy source and controller. Such a wire is of course not desired as it limits the range of the robot and also risks getting stuck in the complex terrain that the robot is supposed to navigate. Finally, the actuators are relatively slow and have narrow ranges of motion (+-90 degree).

Similar to Snapbot, the authors do not mention anything about a software suite (i.e., no simulation).

2.3 Octavio

In [5] Twickel et al. presented Octavio as shown in Fig. 2.3. Octavio is a modular four-, six-, or eight-legged robot made from carbon fiber with fully autonomous legs with regard to control (micro-controller) and energy storage (LiPo battery). Each leg has three active and two passive joints of which each active one is equipped with a DC-motor-gear combination (2.0Nm and 39,5 RPM at 18.5V or 6.5Nm and 27,3 RPM at 18.5V), a spring coupling, an angle sensor, and a current sensor [16]. The legs also contain infrared sensors to measure the distance to the ground and obstacles as well as foot force sensors. The leg module can be mechanically and electronically attached to a body via screw-less flange adapters.

Octavio is meant as a testbed for modular neural control. Each micro-controller is therefore equipped with software that includes sensor calibration data, neural networks, and muscle models. Besides the hardware, the platform also comes with a simulation of the robot. It is implemented in the open source simulation YARS [16] which is based on the Open Dynamics Engine (ODE) [17]. The simulation is optimized for speed and not precision, but it has been calibrated to reduce the reality gap (gap between reality and simulation).

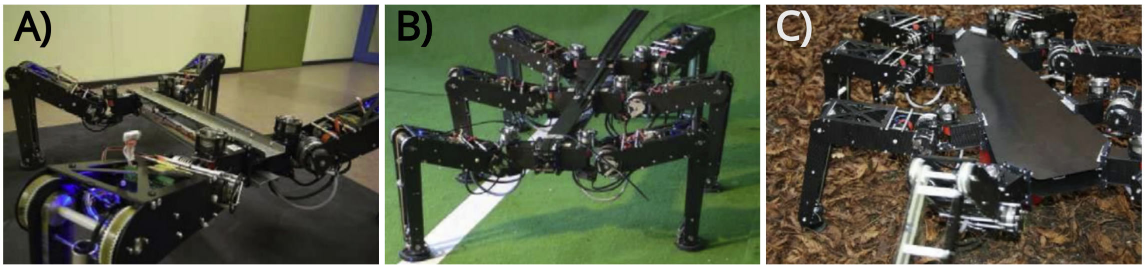


Figure 2.3 – The Octavio robot by Twickel et al. [5]. **A)** shows when Octavio is equipped with 4 legs on flat indoor terrain. **B)** when Octavio is equipped with 6 legs on flat indoor terrain. **C)** when Octavio is equipped with 8 legs on complex outdoor terrain. All images are from [5].

2.3.1 Advantages

The main advantage of Octavio is that each leg is fully autonomous. This makes it possible to experiment with a single leg and subsequently attach it to a body. Another advantage is the actuators which have good performance, includes compliance, and has many sensors. Finally, the platform comes with a calibrated simulation where it is possible to transfer the controller from simulation to the real-world robot. This is an advantage in many types of research as one may test and evolve the controller in a simulated environment.

2.3.2 Disadvantages

A disadvantage of Octavio is that it is complex and specialized in terms of hardware. For example, each leg has a controller and each of them has to be programmed individually. This setup also forces the locomotion controller to be somewhat decentralized.

Another disadvantage is the onboard micro-controllers which have limited processing power and does not support any kind of wireless communication. The micro-controllers can maximum handle a network with 140 neurons and are custom made, making it harder to scale the system with new sensors and actuators. The lack of wireless communication forces the user to communicate with the robot using a physical connection (i.e., cable), significantly constraining its movement as well as making the system less accessible.

Octavio is modular in the sense that it is possible to regulate the number of legs. However, other components like controller, battery, body, and leg configurations remain fixed. This limits the range of research that the robot can be applied to. Another limitation is the use of cheap sensors, which are imprecise and noisy [5].

Finally, the YARS simulation used for simulating Octavio is almost 11 years old and seemingly no longer receives updates [18]. The same is true for the ODE physics engine that has not been updated since 2014 and generally performs worse than other open source physics engines [19].

2.4 PhantomX hexapod mark III

The PhantomX AX Metal Hexapod MK-III, as shown in Fig. 2.4, is a six-legged robotic platform by Interbotix Labs [20]. PhantomX is different from the before-mentioned platforms as it is developed by a company to be sold and not as a part of a research project. Both the software and hardware are, however, open source. The platform is included in this review to display a state-of-the-art legged robot that can be bought on the market.

PhantomX AX Metal Hexapod MK-III is the 3rd major revision of the popular Hexapod robot kit. It comes in two versions; one using the Dynamixel AX-12A (1.5Nm and 59RPM at 12V), and another using the faster Dynamixel AX-18A (1.8Nm and 97RPM at 12V) smart servos. The entire frame of PhantomX is made from aluminum, and each leg has three degrees of freedom. PhantomX is equipped with both an Arduino-Compatible ArbotiX Robocontroller (ATMEGA644p) and a 4500mAh LiPo battery for untethered operation.

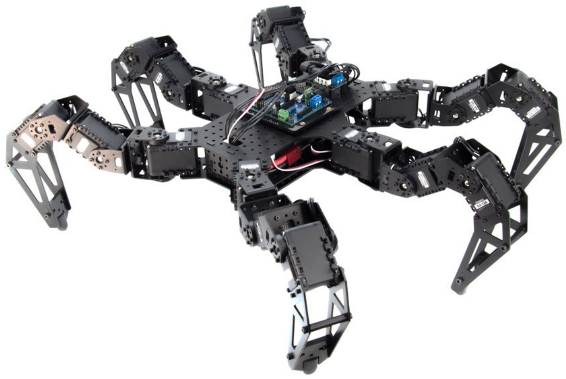


Figure 2.4 – The PhantomX Hexapod Mark III by Interbotix Labs. Image from [20].

2.4.1 Advantages

The main advantage of PhantomX is its large community that contributes to the improvement of both software and hardware for the platform. This can be noted from the fact that the platform ships with advanced Inverse Kinematics Driven Gait Engine containing 6 Different Walking Gaits developed in cooperation with the community. The company behind the platform has also made many detailed guides on how to set up the system, making it easy for a new user to get started with the robot.

Another advantage is the large onboard battery that allows the platform to run for long periods before having to be recharged. Coupled with the onboard controller, this stresses the mobility of the system.

2.4.2 Disadvantages

Like for Snapbot and Octavio, the onboard micro-controller limits the complexity of the locomotion controller that may be used. It likewise requires the user to communicate with the board using a physical connection (i.e., cable) as it does not include any modules for wireless communication. The result is again a less accessible system.

Another disadvantage is the platforms limited amount of sensors. It is only equipped with the ones from the servos (i.e., position, velocity, torque, and temperature). This may limit the amount of research that it can be applied to, especially when considering that the morphology of the robot is fixed. Furthermore, due to the limited servo torque, it may not be possible to add additional sensors or mechanics. This issue has also been reported by a user on their official forum who added additional weight to the robot and as a result also had to add 3D printed brackets with cooling fans to the second actuator of each leg.

Finally, the PhantomX AX Metal Hexapod MK-III does not include any simulation.

2.5 Discussion

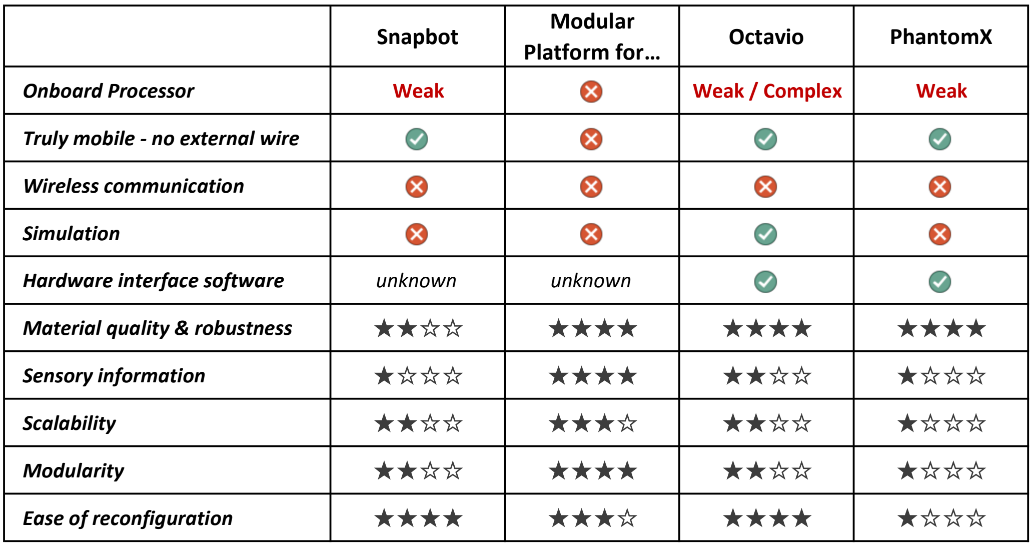
Figure 2.5 shows an overview of the advantages and disadvantages of the legged-robots discussed in the previous sections. From this table, it can be seen that the four presented robots share common shortcomings.

The first is the use of complex controllers that in many cases requires specialized knowledge and software libraries. Another problem with the controllers is their poor options for wireless communication. This limits the mobility of the robot as the user cannot program, control, or receive data from the robot using wireless communication (e.g., WiFi and Bluetooth).

Another shortcoming is the lack of processing power. This limits the complexity of the locomotion controller in use, as many neural networks and especially deep neural network based controllers require a lot of processing power to run at a reasonable speed. It may additionally limit the use of middleware such as the Robot Operating System (ROS) to distribute sensory information.

A third shortcoming is their lack of realistic simulations. This is especially important when doing artificial evolution of control architectures which typically involves repetitive testing of many (hundreds to thousands) trials as to their ability to behave in certain ways.

Finally, it can be said that most of the reviewed platforms are specialized for a set of tasks with few sensors and their modularity often boils down to the number of legs (except for [7]). This means that they may not scale well for different tasks. Following this logic, the research often has to fit the legged robot platform and not the other way around.



	Snapbot	Modular Platform for...	Octavio	PhantomX
Onboard Processor	Weak	✗	Weak / Complex	Weak
Truly mobile - no external wire	✓	✗	✓	✓
Wireless communication	✗	✗	✗	✗
Simulation	✗	✗	✓	✗
Hardware interface software	*unknown*	*unknown*	✓	✓
Material quality & robustness	★★☆☆	★★★★	★★★★	★★★★
Sensory information	★☆☆☆	★★★★	★★☆☆	★☆☆☆
Scalability	★★☆☆	★★★☆	★★☆☆	★☆☆☆
Modularity	★★☆☆	★★★★	★★☆☆	★☆☆☆
Ease of reconfiguration	★★★★	★★★☆	★★★★	★☆☆☆

Figure 2.5 – Overview of the advantages and disadvantages for the legged-robots discussed in this chapter. The platforms are rated based on a personal assessment of available information (e.g., scientific articles about the platforms).

Chapter 3

Key features of MORF

In this chapter, a set of key features or high-level requirements will be defined, which describes *"what"* features MORF should include. The main purpose of MORF is to be used as a research framework for testing new control techniques and demonstrating that a given locomotion controller will not only work in a controlled simulation environment but also in the real world. The framework thus needs to be applicable to a wide range of research studies, meaning that it has to support different kinematic configurations, sensory feedback, and control architectures.

3.1 Modularity and scalability

For MORF to apply to a wide range of research studies, it should be modular and scalable. A modular design implies that a system is subdivided into modules that can be rearranged in different ways. Obvious choices of modules for legged robots are legs, sensors, controller/computer[1], and body (see Fig. 3.1). It is preferable if the individual modules are easy to attach and detach meaning that a convenient fastening method must be found.

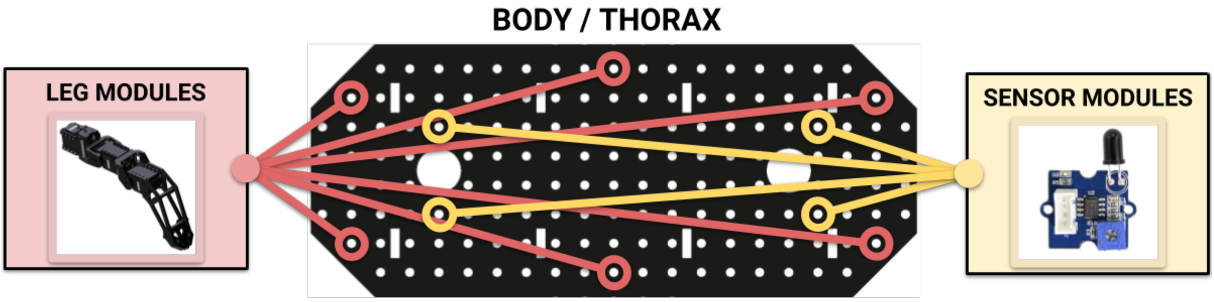


Figure 3.1 – Illustration of using leg and sensor modules. The red circles indicate that a leg module is placed at that location, where yellow is for sensor modules. In this case, the robot has six legs (a hexapod) and four sensor modules attached.

It is challenging to predict what sensors and how many legs the user wishes to attach, but it is fair to assume that they will need to either subscribe to data from the system, publish data to the system, or both. This makes the connections of the modules vital, as they must be scalable and independent of different data formats [21].

3.2 Control architecture

It is difficult to find a computer that meets the requirements of all thinkable studies. For some a Raspberry Pi will meet most requirements, others may need a larger processor with performance closer to a high-end desktop computer, while a third group need specialized systems like the Zybo board from Digilent. For this reason, the computer unit itself is seen as a replaceable module. MORF should, however, be *born* with one default computer, as explained in the following.

The default computer of MORF should, first of all, facilitate wireless communication, so that the platform can remain untethered and truly mobile. The user will in this way be able to access and program the system remotely and in real-time, as well as using an external and more powerful computer for heavy calculations that the onboard computer cannot handle. Besides handling wireless communication, the computer should also handle a locomotion controller. It can be hard

[1] To differentiate between the locomotion controller, written in software, and the physical system it runs on, we call the physical system a computer.

to specify the exact processor demands for this, but it should be able to handle most of today's locomotion controllers (ranging from controllers based on neural networks to controllers based on inverse kinematics). Finally, it should be able to handle communication to the actuators and sensors on the system. The desired default computer setup on MORF is illustrated in Fig. 3.2.

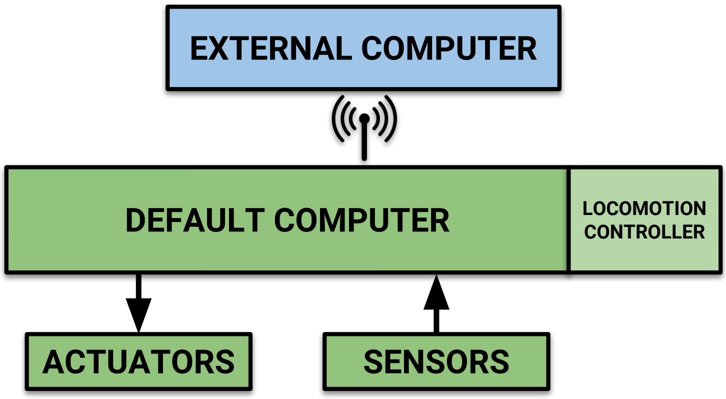


Figure 3.2 – Illustration of the proposed computer architecture for MORF. The blue module is offboard, while green are onboard. Arrow connections symbolizes wired connections, while the symbols between the blue and green modules symbolizes wireless connections. Note that a locomotion controller is running on the default onboard computer.

3.3 Energy source

MORF needs to be powered by an onboard energy source in order to keep it truly mobile. There is no hard requirement on the energy capacity, but a fair assumption is that the robot should be able to walk for about one and a half hours. Also, the energy source should have high specific energy to keep the mass of the robot down.

3.4 Performance

A current disadvantage of legged-robots is their low energy-efficiency [13, 22]. It is important that MORF is energy-efficient although this performance measurement is most often limited by the locomotion controller and state-of-the-art components in use. A list of related sources of energy dissipation in legged locomotion, not related to the choice of controller nor components, is described by D. Todd [3] as follows:

1. Loss of kinetic energy which must be applied to the legs to make them oscillate
2. Power wasted in supporting the body against gravity and other forces
3. Soil adhesion to feet, and other forms of motion resistance

Possible ways of addressing these sources of energy dissipation are:

1. Reduction of the kinetic energy of the legs by reducing their mass [22]
2. Reduction of the required power used on supporting the body against gravity by reducing the weight of the body and energy source
3. Reduction of energy lost to external forces by having an embodied and compliant foot

A second disadvantage of today's legged-robots is their relatively slow movement speed [13, 22]. The speed of the robot is determined by the stepping length, the speed of the actuator, and the locomotion controller in use (e.g., which gait the robot walks with). As a system designer, we are only able to decide the first two parameters with the choice of actuator being the most prominent one.

The above points will be used in the following sections when specifying key features for the legs, actuators, feet, and body. It may, however, be difficult to comply with all the points due to the state of today's technology. An example could be a state-of-the-art actuator that performs as required, but at the expense of a heavier design compared to other alternatives with poorer performance.

3.5 Leg design

The leg design is crucial for every legged-robotic system, as the legs are both in contact with the environment and also have to support the body against gravity and other forces. The following will describe the key features that the default leg of MORF should include.

3.5.1 Material

The material of the legs should be light and sustainable in order to keep the system energy-efficient as discussed in the previous section. The reason for requiring sustainable material is that the legs are the parts of the system that are exposed the most to the surrounding environment and are therefore also expected to be worn most.

3.5.2 Kinematic architecture

Literature shows that various leg configurations for legged-robots exist, all with their advantages and disadvantages [13]. The two main leg types, shown in Fig. 3.3, are bio-inspired (e.g. mammals [23], reptiles [24], insect [25], etc.) and non-zoomorphic legs (e.g. under-actuated [26], telescopic [27], hybrids [28], etc.). MORF should be equipped with bio-inspired three degrees of freedom (DOF) legs with configurations similar to those of insect and mammals (i.e., reconfigurable). The reason for this is that by using these leg configurations, it is possible to create solutions inspired by nature. The mammal and insect legs are furthermore well studied both in biology and robotics [29, 30, 31, 32]. Note that reptile legs can be regarded as a special case of insect legs as demonstrated in [33] and may therefore easily be configured in software without any hardware changes[2].

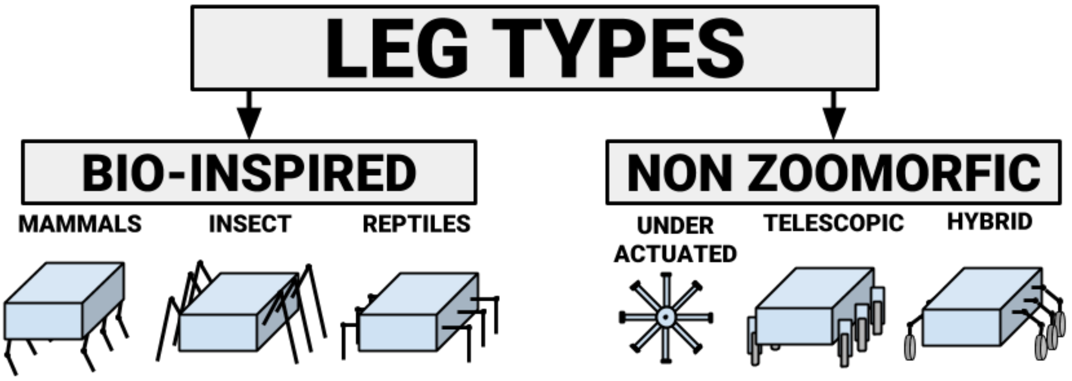


Figure 3.3 – Illustration of different leg types. Inspiration from [13].

Robotic legs are typically placed perpendicular to the body as seen in [6, 7, 29]. However, such simplifications might not benefit the performance of the system as described in [10]. Here, it is shown that by using non-identical legs together with a minimum of simplifications a dung beetle inspired robot was able to walk faster than its simplified twin. MORF should, therefore, support non-perpendicular legs like those of stick-insects shown in Fig. 3.4. This will also widen the range of scientific applications.

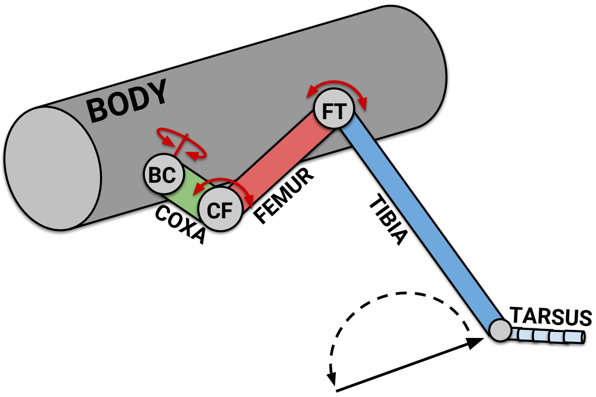


Figure 3.4 – Illustration of the leg configuration for the stick insect. The leg consists of three segments (coxa (green), femur (red), tibia (purple), and tarsus (blue)) and three joints (BC (body-coxa), CF (coxa-femur), and FT (femur-tibia) joint). Note that the leg segment called coxa is not placed perpendicular to the body. Inspiration from [34].

[2] Note that reptiles actively use their backbone joint for locomotion, which is not considered in this thesis.

3.5.3 Obstacle avoidance capability

The length of the legs directly specifies the systems' obstacle avoidance capabilities through its stepping height. MORF should at least be able to walk on stairs, as it is to be used in generic physical environments. Fulfilling this requirement will also enable MORF to avoid obstacles of equal heights. Note that this requirement is purely related to the morphology and it is up to the user to develop a controller that in practice can make MORF walk up stairs.

3.5.4 Foot design

Many of today's legged-robots with three or more legs uses one of the three foot design's shown in Fig. 3.5, with **B)** being the most common. It is, however, rare that this choice is made with any scientific reasoning. This can among other be seen from the fact that most research on foot design is from studies on bipeds [3]. Yet when legged-robots of all kinds are tried on different ground the importance of the foot cannot be neglected [3, 22]. This is because the stability, velocity, and energy-efficiency of legged-robots strongly depend on the established ground contact [35]. MORF should, for this reason, make use of a foot that is scientifically justified.

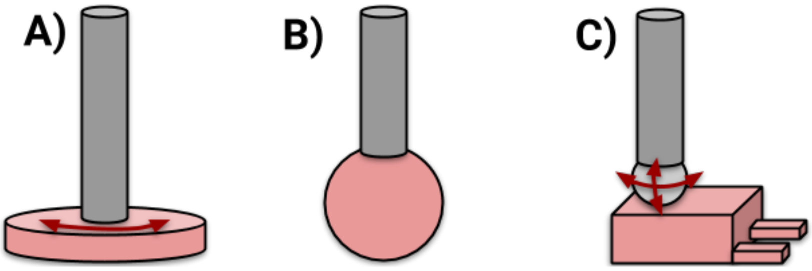


Figure 3.5 – Illustration of foot designs often found in literature. Gray indicates the tibia and red indicates the actual foot. **A)** rotating plate foot [33]. **B)** sphere shaped foot [36, 25, 37]. **C)** complex foot with toes/claws attached and a spherical joint which is either passive or active [38, 39].

The feet of MORF should also be equipped with force sensors for measurement of stepping force. Foot force sensors have been used in several locomotion controllers where acts as a vital component [30, 11, 40, 41, 42]. This is because stepping force enables good assessment of the type of terrain and in consequence adaptation to the changing characteristics of the foot-ground contact [39]. The sensor is unfortunately not trivial to implement as it both have to be placed in the right location while also providing some mechanism for calibration. This is often solved using a compliant system, as seen with the spring system on the AMOS II hexapod by P. Manoonpong et al. [29]. MORF will need a similar system which is further elaborated in Sect. 3.5.6. An alternative to using dedicated force sensors in each leg is to calculate the stepping force from the actuators load current. This approach is seen in [43], [44], and [45]. It would be preferable if the selected actuators, discussed in the following section, also could provide such feedback in order to have redundancy in the system.

3.5.5 Actuators

Many kinds of actuators can be used for operating legged-robots. It was, however, decided to equip MORF with electrical rotating actuators due to their relatively low price, simple control, and the existence of suitable technologies for storing the energy [13]. A newer type of electrical actuator to be considered are smart servos, also known as robot servos. The main difference between regular and smart servos is the way they are controlled. With regular servos the communication is unidirectional, but with smart servos, the user can get useful feedback through serial communication. Most notably feedback's are position and force (i.e., load current) which are valuable information in most projects on locomotion control, including the famous Walknet controller [30]. Many smart servos can also handle different input commands due to a built-in microcontroller and serial communication. This means that the servos will accept not only force commands but also velocity and position commands. Another advantage to the serial communication is that the smart servos can be daisy-chained. Finally, since smart servos are the latest technology, most of them benefit from good overall performances, such as high precision, large rotational range, large stall torque, etc. [46].

Another requirement for the actuators is their ability to carry MORF's own weight including a margin for additional payload. This is to support the weight of attachable sensors and mechanics that might be added by the user. The speed of the actuators are not of primary concern, but they should not be too slow. The trade-off between the actuator's speed and torque should thus reflect the need of both.

3.5.6 Compliance

In classical robotics, actuators are preferred to be stiff as they often have to follow a predefined path with little error. This is often not the case for bio-inspired robots where compliance or spring-like behaviors, like those found in biological systems (muscles), are preferred. Compliance generally offers several valuable advantages in applications such as; safe human-robot interaction, comfortable actuated prostheses, and in the design of legged-robots [47]. In the case of the legged-robots, a compliant design can prevent damage from occurring from imposed motions on the actuators/robot and give the robot a more natural feeling. It may also make the legged-robot more energy-efficient by enabling energy to be stored during touchdown of the feet and released during push-off [47].

It should for the above reasons be clear that compliance is a significant advantage for any legged-robot, which is also why it should be implemented on MORF.

3.6 Body design

It can be hard and nearly impossible to design a single body structure that can be used in all types of legged-robot studies. The body itself is, therefore, considered a module and thus replaceable. It will hereby be possible to change the body to one that supports any desired number of legs (e.g., hexapod, quadruped, and even bipedal) and task.

The default body module of MORF should be designed with the maximum of symmetry to keep the legged-robot balanced and stable while walking. The body module is what ties all other modules together, and its primary objective is to support all modules developed in this thesis while being flexible enough to support future ones. As mentioned earlier some convenient fasting method for connecting the leg and sensor modules to the body must be found. One idea could be to equip the body with a grid of mounting holes, as illustrated in figure 3.1. Sensor and leg modules could then be connected to the body with the help of screws or some kind of plugs. The body module also needs to incorporate cable management, as all of the attached modules include some cable that needs to be connected with the computer module.

For supporting the computer modules, a test concerning the size of the most popular computers has to be conducted. This is to make sure the mounting area for the computer is large enough, although if the area is too wide it will increase inertia and by extension the torque needed from the supporting legs [25].

Another large component that is to be mounted on the body is the energy source. It is essential that it is protected from the environment to avoid possible failures that could potentially damage surrounding modules.

3.7 Software suite

To make the framework complete a software suite is also needed. This suite should first of all include all the necessary software for controlling the hardware and reading the sensor values. It is important that communication is fast as well as scalable like MORF itself.

Besides the hardware-related software, a simulation of the MORF should also be developed. This will enable the user to quickly test their code and thereby avoid potential damages on the real robot. It will also make it possible to experiment with different morphologies of MORF and to create controllers based on machine learning (e.g., evolutionary robotics). This is because artificial evolution of control architectures typically involves constant and repetitive testing of hundreds upon thousands of agents with respect to their ability to perform a specific task or behave in a certain way [48]. Finally, a simulation will enable students to test their controllers for errors and inconsistencies before being approved by their supervisors to test it on the real robot. For these reasons, the simulation framework should be fast, precise, and easy to use.

3.8 Price and quality

Although the price is not a main focus, the cost should be held at a reasonable level. This means that no hard limit exists, but a rough estimate would be around 50.000 DKK. The reason for this rather high price is that parts that are exposed to the environment should be made from quality materials and that the different components for the robot should be state-of-the-art from well-established brands. This is to ensure better maintainability and that the various components will be available for future replacements.

3.9 Summary of key features

Table 3.1 is a summary of the key features (KF) discussed in the above sections. In the next chapter, these features will be used to design the hardware and software of MORF.

Table 3.1 – Summery of the Key Features

KF	Sect.	Description
		Structure
KF01	3.1	Legs, sensors, computer, and body should be separate modules
KF02	3.1	MORF should be scalable with new modules
		Leg module
KF03	3.5.2	The leg module should be reconfigurable and bio-inspired with configurations similar to those of insect and mammals
KF04	3.5.2	The leg module should be attachable in non-perpendicular ways
KF05	3.5.3	The leg should enable MORF to walk on stairs and similar sized obstacles
KF06	3.5.1	The leg module should be lightweight
KF07	3.5.1	The leg module should be made from sustainable materials
KF08	3.5.4	The leg module should make use of a foot design that is scientifically justified
KF09	3.5.4	The leg module should include a mechanism for measuring the stepping force, both in the feet and actuators
KF10	3.5.5	The leg module should be equipped with smart servos that are able to carry the robots own weight while having a margin for additional payload

Table 3.1 continued on next page

Table 3.1 continued from previous page

KF11	3.5.6	The leg module should include some kind of compliant mechanism to give the system a more natural feeling and prevent damage
Body module		
KF12	3.6	The body module should be designed for a symmetric hexapod
KF13	3.6	The body module should have a convenient fastening method and some cable management mechanism
KF14	3.6	The body module needs to support relevant and popular computers
KF15	3.6	The body module should include mechanics that holds the desired energy source while protecting it from the environment
Computer		
KF16	3.2	The computer module should be able to handle wireless communication, a locomotion controller, and hardware interface at the same time
Power source		
KF17	3.3	The system should be powered by an onboard energy source, that can last for around one and a half hours of use
Price and supplier		
KF18	3.8	The system should not cost much more than 50.000 DKK
KF19	3.8	The components of the system should come from well established brands
Software		
KF20	3.7	The system should include a complete software suit containing methods for reading sensor values and controlling the hardware in a fast and scalable way
KF21	3.7	The system should include a complete software suit containing complete and realistic simulations of MORF

Chapter 4

Design and analysis of MORF

In this chapter, it will be explained "*how*" to achieve the desired framework as defined by the key features (KF) presented in the previous chapter. It will consist of various analyses from which components and designs emerge. All of the designs are presented in the form of computer-aided designs (CADs) made using Autodesk's 3D CAD software called Inventor [49]. Technical drawings used for 3D printing and manufacturing can be found in the supplementary materials in the `Mechanical_parts` directory.

Note that this chapter cannot be written entirely sequentially, as many of the KFs depends on one another. An example is the required actuator torque that depends on the weight of the mechanical parts, which in turn depends on the weight and dimensions of the actuator itself. The mechanical design of MORF is for this reason presented before the choice of actuator.

Finally, every time a KF is addressed it will be indicated in the following way: (**KF##**), where `##` is the number of the KF.

4.1 Leg module

The leg module is the most complex part of MORF. It will be bio-inspired and consist of four leg parts (pre-coxa, coxa, femur, and tibia) connected by three actuators/joints (BC (body-coxa)[1], CF (coxa-femur), and FT (femur-tibia) joint) (**KF03**). The design of each leg part reflects the desire to both comply with the KFs specified in the previous chapter and to keep the leg parts as light and short as possible in order to limit the required torque in the three actuators. Note that the leg parts are designed to fit the Dynamixel XM430-W350 smart servo whose choice is discussed in Sect. 4.4.

4.1.1 Mechanical parts

Figure 4.1 shows the first leg part, also known as pre-coxa, consisting of the two connectors BC-C-1 and BC-C2 which together serves as a rigid connection between the body and the actuator acting as the BC joint (black rectangular box).

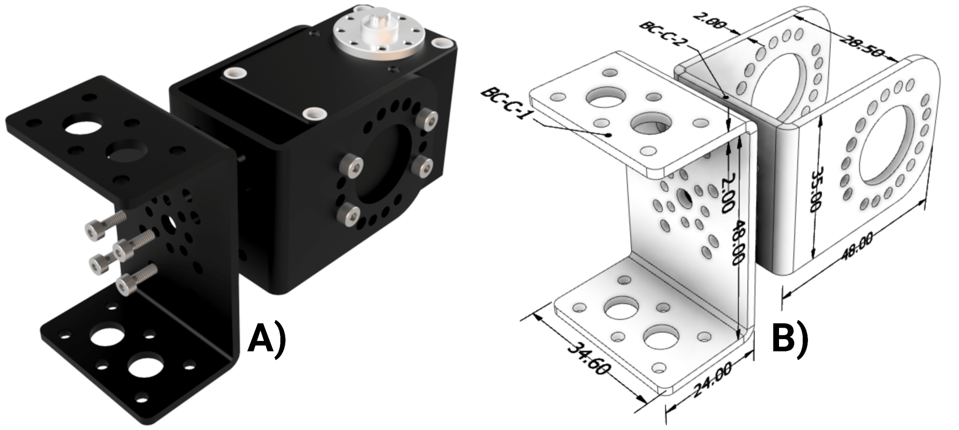


Figure 4.1 – The first leg part consisting of two connectors (BC-C-1 and BC-C-2) that can be bolted together in different orientations. **A)** shows a realistic rendering of the two connectors and the Dynamixel smart servo (acting as the BC joint). **B)** shows the two connectors and their dimensions. All units are in (mm).

Both connectors in the pre-coxa are made from 2mm bent aluminum (**KF07**), and both have a Circular Hole Pattern (CHP) that allows them to be attached in various orientations with respect to each other (see Fig. 4.2D-F). A close up of the CHP can be seen in Fig. 4.2A. Using the yellow holes, the remaining leg can be attached, so that it is rotated up to 360° divided into 12 steps of 30° each (**KF03**). Additionally, the blue and red holes make it possible to attach the connector directly to an actuator or the body module (see Fig. 4.2B and C). The connectors

[1] This joint is sometimes called the TC (thorax-coxa) joint.

can in this way be used as a base for a robotic arm like the one seen on SpotMini[2] from Boston Dynamics. It may also be used for a leg configuration comparable to that of mammals (**KF03**).

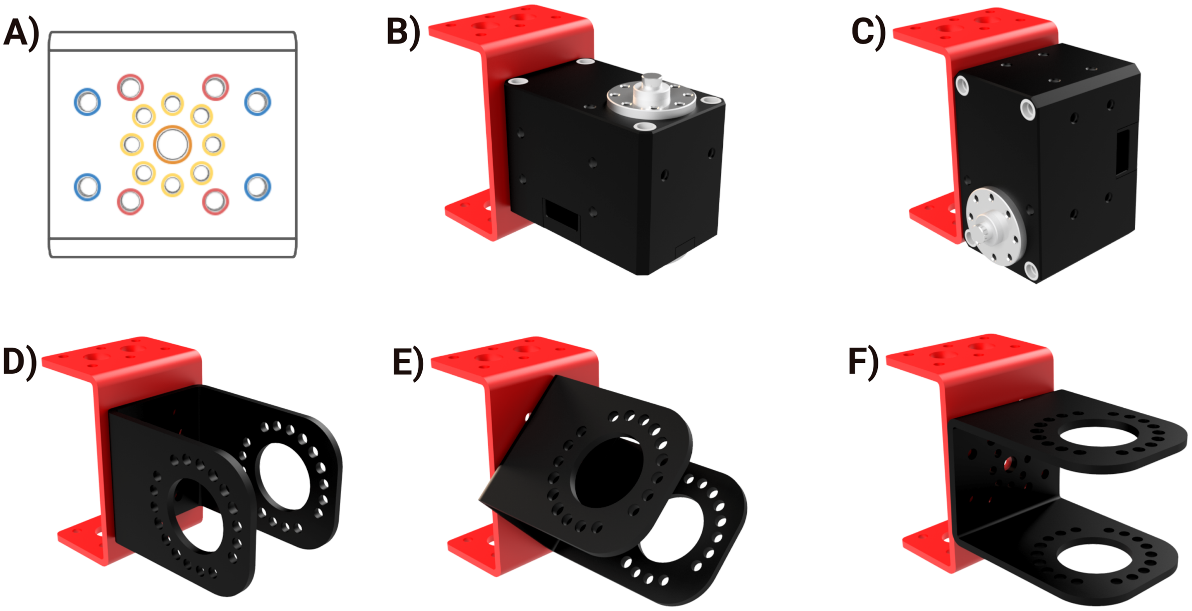


Figure 4.2 – **A)** top view of the Circular Hole Pattern (CHP). The yellow holes are for bolting two connectors together. Using these, they can rotate up to 360° in steps of 30° relative to each other. The orange hole in the middle is for weight reduction. The blue and red holes are for attaching the actuator or body plate directly to the connector. **B-C)** examples of connecting the actuator directly to the BC-C-1 connector (shown in red). **D-F)** examples of connecting the BC-C-1 (shown in red) with the BC-C-2 (shown in black) in different orientations.

The reason for using the BC-C-2 connector and not connecting the actuator directly to the BC-C-1 connector is that it can be used to rotate the actuator up or downwards ± 90° in steps of 18°. This is shown in Fig. 4.3, where the actuator is rotated 0°, 36°, and 90° respectively. It is hereby possible to attach the leg in a non-perpendicular way to the body (**KF04**).

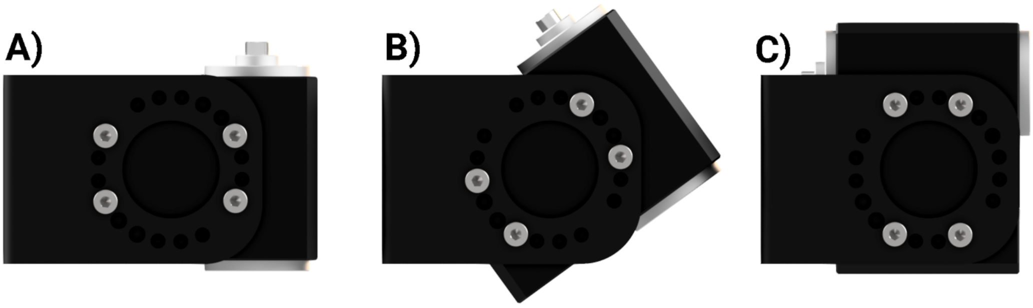


Figure 4.3 – Three different attachment orientations of the BC joint actuator. **A)** the BC joint is perpendicular to the body, **B)** 36° upward, and **C)** vertical upwards. Each step the actuator can rotate is equal to a 18° rotation.

[2] https://www.youtube.com/watch?v=aFuA50H9uek

The form of the BC-C-2 connector is derived using Autodesk Inventors topological optimization. This is a mathematical method that optimizes the design layout (i.e., weight) based on a set of constraints and loads similar to those expected during real-world operation. The connector before and after topological optimization can be seen in Fig. 4.4. The reason for not using this optimization method on the BC-C-1 connector is its small size and many constraints. It is, however, inspired by the results of the optimization done on the BC-C-2, e.g., the weight reduction holes on top of the BC-C-1 connector (**KF06**).

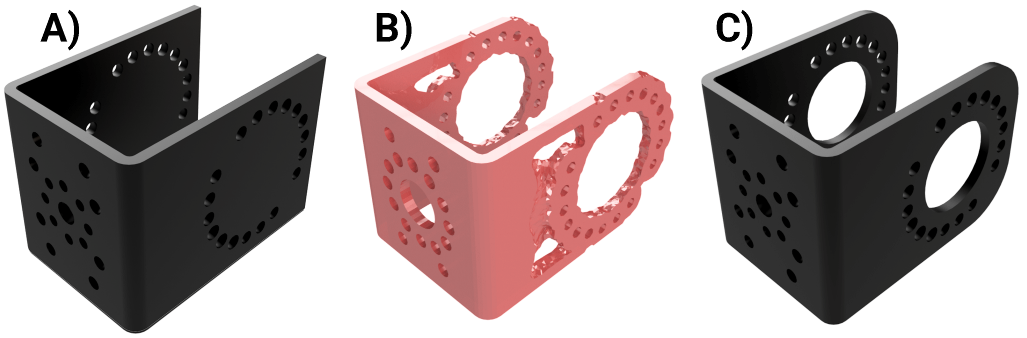


Figure 4.4 – Topological optimization of the BC-C-2 connector used in the pre-coxa leg part. **A)** is without any optimization, **B)** is the output from Inventor's topological optimization, and **C)** is the connector with the new optimized form.

Figure 4.5 shows the second leg part, also known as coxa. It consists of the HORN-C and SLIM-C connectors which together serves as a rigid connection between the two actuators acting as BC and CF joint. These connectors are also made from 2mm bent aluminum (**KF07**) and are likewise equipped with the CHP (**KF03**).

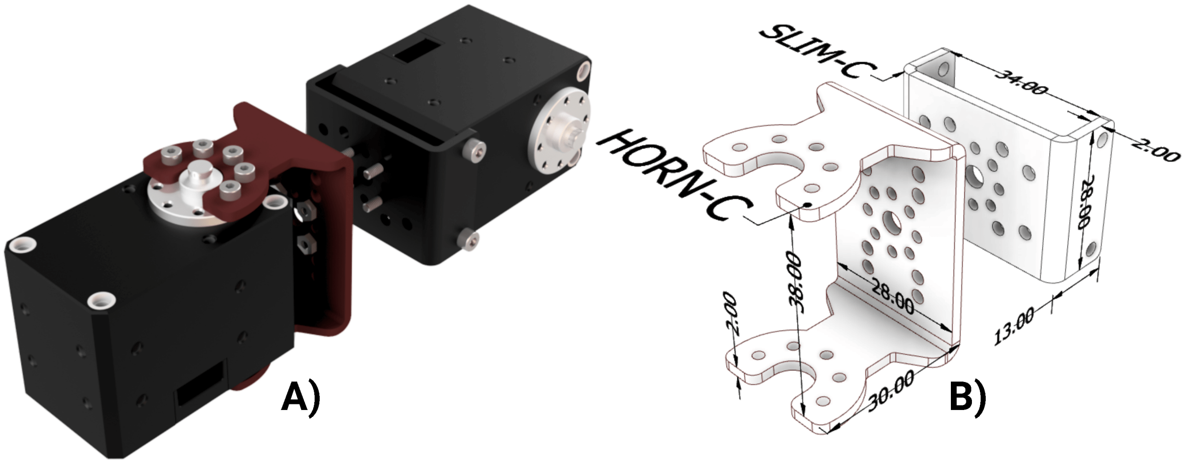


Figure 4.5 – The coxa leg part consisting of two connectors (HORN-C and SLIM-C) that can be bolted together in different orientations. **A)** shows a realistic rendering of the two connectors and the Dynamixel smart servos (acting as the BC and CF joint). **B)** shows the two connectors and their dimensions. All units are in (mm).

The HORN-C connector is also optimized using topological optimization, see Fig. 4.6 (**KF06**).

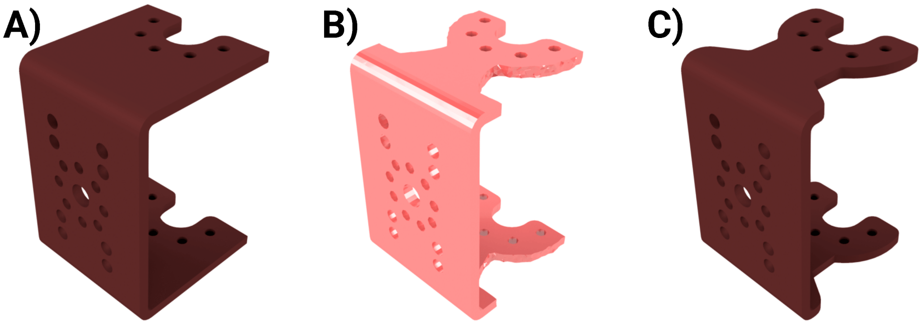


Figure 4.6 – Topological optimization of the HORN-C connector. **A)** without any optimization, **B)** the output from topological optimization, and **C)** the connector with the new optimized form.

An advantage of this optimization, besides the reduced weight, is an extended range of motion for the BC joint when placed in a non-perpendicular position (see Fig. 4.3B and C). This is because its concave shape allows the connector to *bend* around the BC-C-2 connector. Like for the BC-C-1 in the pre-coxa leg part, the SLIM-C connector is also considered too small and constrained for topological optimization.

Figure 4.7 shows the third leg part, also known as femur. It consists of two FEMUR-C connectors made from 2mm aluminum (**KF07**). The connector is angled 30° to better separate the two joints and to enable it to be placed vertically on the CF joint when walking resulting in a smaller arm and by this a reduction in torque needed support the body. Note that the length of FEMUR-C directly correlates to the maximum stepping height of the leg, which will be discussed at the end of this section.

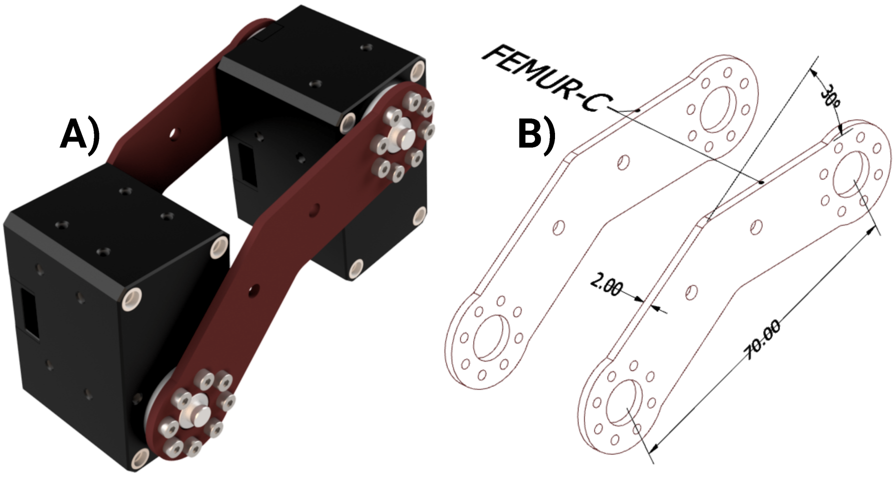


Figure 4.7 – The femur leg part consisting of two identical connectors (FEMUR-C) that are bolted to the horns of the Dynamixel servos. **A)** shows a realistic rendering of the two connectors and the Dynamixel smart servos (acting as the CF joint and FT joint). **B)** shows the two connectors and their dimensions. All units are in (mm).

Figure 4.8 shows the final leg part, also known as tibia. It consists of the FOOT and the SLIM-ANGLE-C connector that connects the FOOT to the FT joint. The connector is made from 2mm bended aluminum (**KF07**) and is also equipped with the CHP (**KF03**). It is furthermore angled 20° to give the FT joint additional range of motion underneath the robot. It also makes it possible to place the FOOT tip underneath the FT joint when walking resulting in a smaller arm and thus reduced torque.

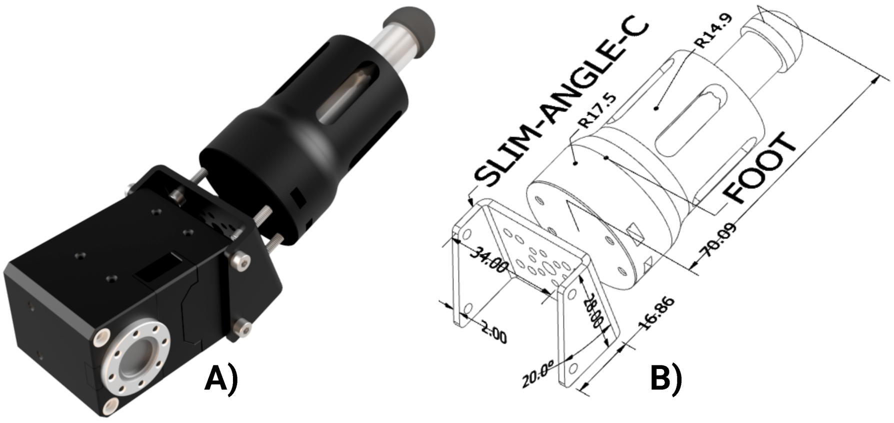


Figure 4.8 – The tibia leg part consisting of a FOOT and the the SLIM-ANGLE-C connector. **A)** shows a realistic rendering of the connector, the FOOT, and the Dynamixel smart servos (acting as the FT joint). **B)** shows the connector, the FOOT, and their dimensions. All units are in (mm).

To comply with **KF11**, the FOOT should include some kind of compliance. The following list states some ideas for how such a mechanism could be implemented in the framework.

- A passive compliance design to provide a degree of resilient suspension and compliance (e.g. using silicone or springs) [29, 36]
- Compliant actuator couplings like GS Compact from ROTEX® [50]
- Active compliance control [51, 52]

The two first ideas are passive compliance, while the last one is active compliance. Such compliance could be realized with active impedance control where compliance is achieved in software allowing real-time adjustment of stiffness and damping [51]. Since the leg should remain compliant at all times, active compliance will be left for the user to implement. The GS compact coupling from ROTEX is also discarded as it requires attachment to the horn of the actuator and would make the leg very wide and reduce the range of motion. A passive compliance mechanism in the form of a spring is therefore designed (**KF11**). This can be seen in Fig. 4.9 which shows **A)** an exploded view of the FOOT and **B)** the inside of the FOOT (i.e., when cut in half). This design is greatly inspired by that of AMOS II [29]. As can be seen, the FOOT consists of six parts: a force SENSOR-PEN, a SPRING, an aluminum PIPE, a cylindrical 3D printed FOOT-SHELL, and half sphere rubber FOOT-TIP. The FOOT-SHELL is designed to be robust and easy to 3D print, i.e., thick and requires minimal use of support structures. As seen in Fig. 4.9B the shell has a rippled chute for the PIPE to slide and rotate in. The fact that the PIPE can rotate minimizes foot slipping during contact phases [5]. The SPRING inside the PIPE is replaceable depending on the desired amount of compliance. When no force is applied to the foot, the leg will be fully extended like seen in Fig. 4.10A. When force is applied to the leg (Fig. 4.10B and C) the spring will act as passive compliance, and the SENSOR-PEN will be able to measure the applied force through the spring (**KF09**). This configuration will reduce the noise introduced to the force sensor. The presented foot design is not final and should be seen as a temporary solution. A future and improved design (shown in appendix A) will include a 3D force sensor used together with a silicone pad in replacement of the spring. The reason for not implementing this design yet is that the sensor is still under development and will subsequently undergo a patenting process.

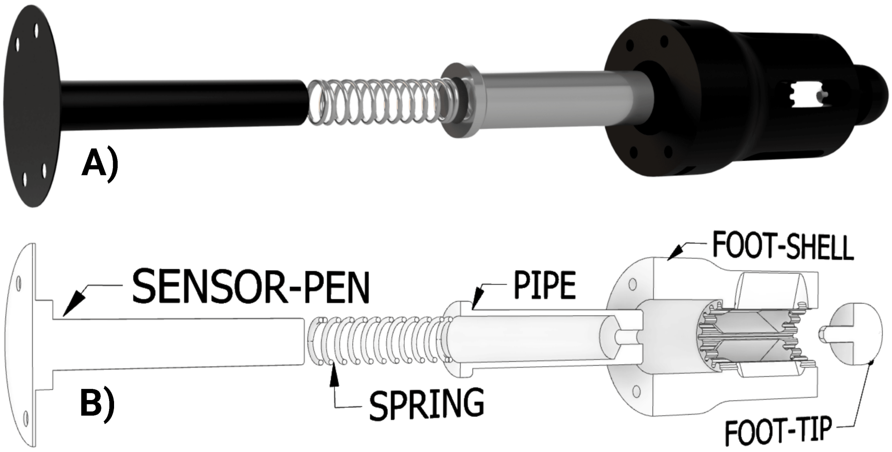


Figure 4.9 – Exploded view of the foot. **A)** shows a realistic rendering of the foot. **B)** shows the names of the different parts and how they look inside when cut in half.

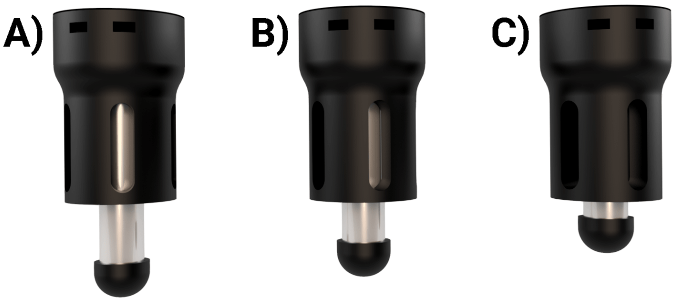


Figure 4.10 – Illustration of the compliance of the foot. **A)** is without any force applied to the foot, **B)** is with medium force applied, and **C)** is with maximum force applied.

At the end of the PIPE, a rubber FOOT-TIP is placed. The reason for choosing this design is due to the work of A. Roennau et al. [35] who are one of the few that has made research on foot designs for multi-legged robots (**KF08**). In their work, they test several foot designs in various environments and compare the walking velocity and energy consumption from these tests. They also state four basic requirements that a good foot design needs to include:

- **Compliance:** To reduce peak stress induced by collisions with the ground (e.g., using deformable materials like rubber, springs, etc.).
- **Adaptability:** By adapting to the terrain structure and increasing the contact area it is possible to improve the load capacity and have a positive influence on the slippage.
- **Friction:** A high friction coefficient between the two involved materials (foot tip and ground surface) can reduce the slippage significantly.
- **Ground Foot Alignment:** The orientation of the foot tip towards the ground has a big influence on the contact area. A solution is the usage of a ball like foot tip design. In this case, the contact area is nearly the same during the stance phase, but the area is also smaller than a flat foot design.

Results from their tests showed that different foot designs are necessary for different environments just like humans wear special shoes on the beach or in the mountains [35]. The spherical rubber foot with moderate stiffness did, however, prove to be the most versatile solution as it had an overall good performance in all of the environments. This is presumably because the spherical design provides a good ground foot alignment while the rubber provides a good grip on most surfaces and a bit of compliance (**KF11**). MORF is for this reason using a similar design.

4.1.2 Complete leg design

The complete leg design in different configurations can be seen in Fig. 4.11 (**KF03**). Figure 4.11A shows a leg that is using all the presented connectors in an insect-like configuration (called long insect leg). Figure 4.11B also shows an insect-like leg configuration, however, this leg uses a minimum of connectors in order to stay compact and thereby reduce the torque requirements (called short insect leg). Figure 4.11C shows a leg in a mammal-like configuration where the FEMUR-C connector is a straight version of the one seen in 4.7 specifically made for this configuration (called mammal leg). Figure 4.12 shows the technical drawings for the long insect leg, i.e. when using all the presented connectors. From this is can be seen that the leg has large range of movements and that the CF and FT joint are placed on a straight line for easy inverse kinematics. The technical drawings for the mammal and short insect leg can be found in appendix B.

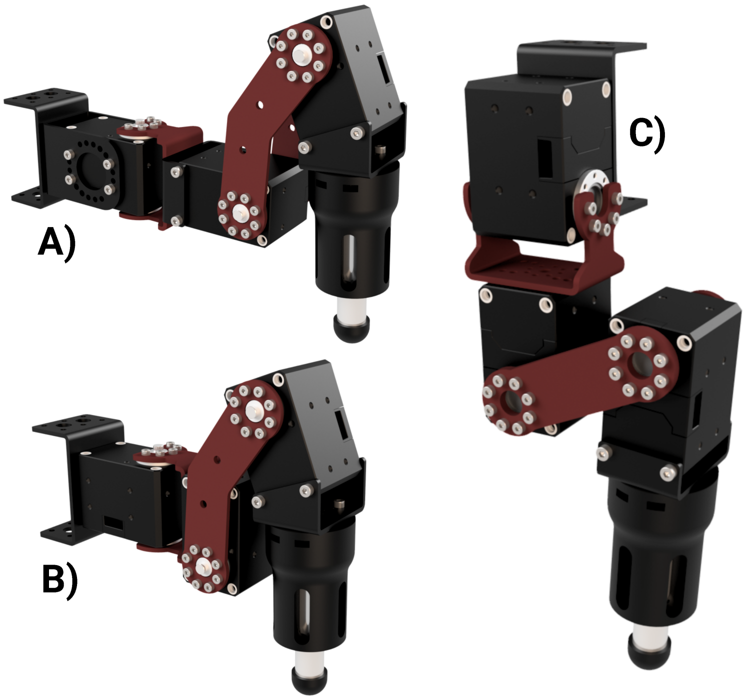


Figure 4.11 – Three different leg configurations. **A)** long insect leg with the option to rotate the connections, **B)** short insect leg with no rotating connections, and **C)** mammal leg.

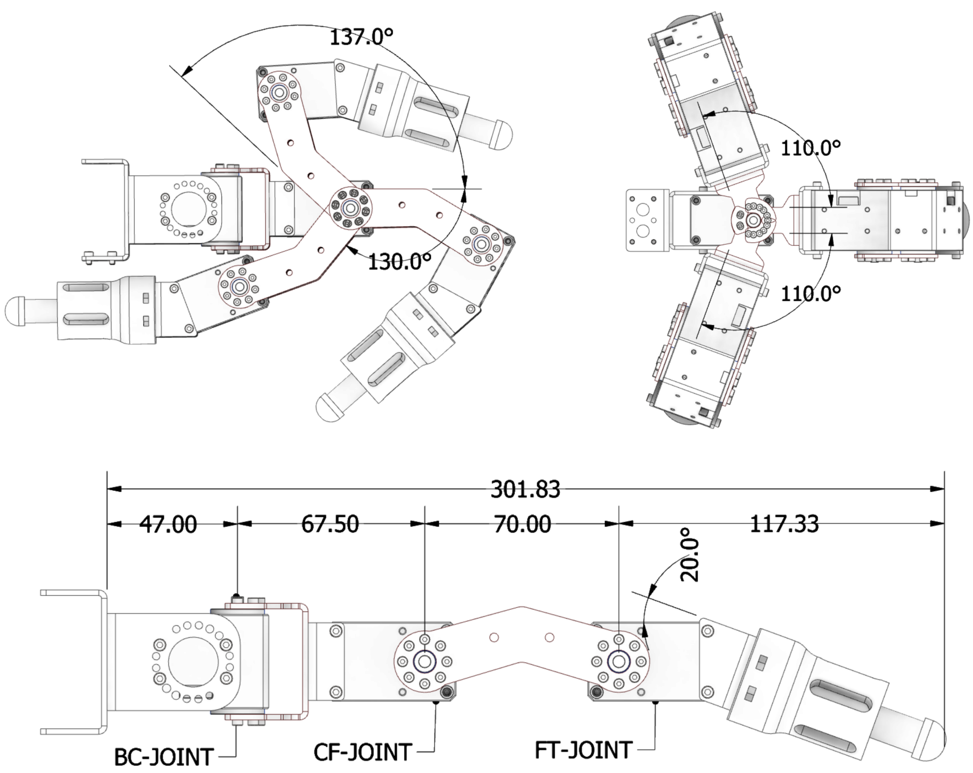


Figure 4.12 – Technical drawings for the long insect leg configuration shown in Fig. 4.11A. **A)** shows the vertical range of movement (side view), **B)** shows the horizontal range of movement (top view), and **C)** shows the leg dimensions. All units are in (mm).

As mentioned earlier the stepping height of the leg can be controlled by the length of the FEMUR-C connector. This is shown in Fig. 4.13 where the length of the FEMUR-C is plotted against the maximum stepping height when using the stepping motion shown in Fig. 4.14. Note that the minimum length of FEMUR-C capable of separating the two actuators is 34mm.

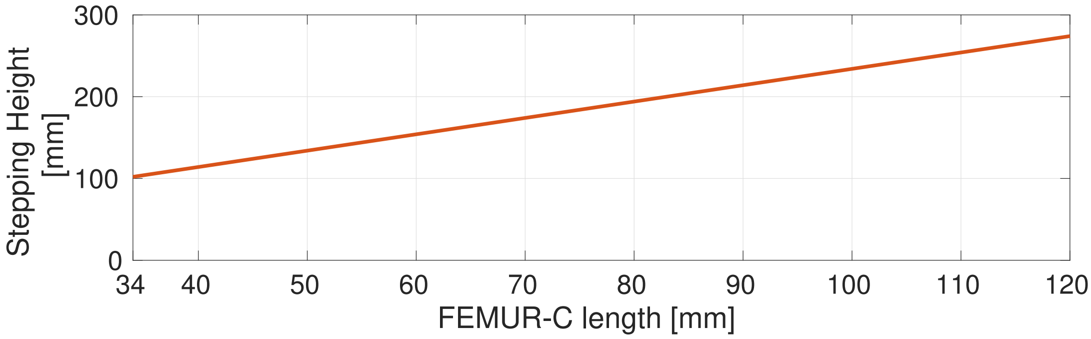


Figure 4.13 – Length of the FEMUR-C connector versus the maximum stepping height of the leg. Note that the minimum length of FEMUR-C is 34mm as the actuators would otherwise not be fully separated.

According to [53] the general stepping height requirements for stairs are a minimum of 150mm and a maximum of 220mm. If MORF should be able to climb all stairs, FEMUR-C would have to be 93mm Long (see Fig. 4.13). However, to get a more compact design, it was set to 70mm resulting in a maximum stepping height of 174mm. MORF will in this way be able to climb most stairs, and due to the simplicity of FEMUR-C, it is easy to design and produce a longer version (**KF05**).

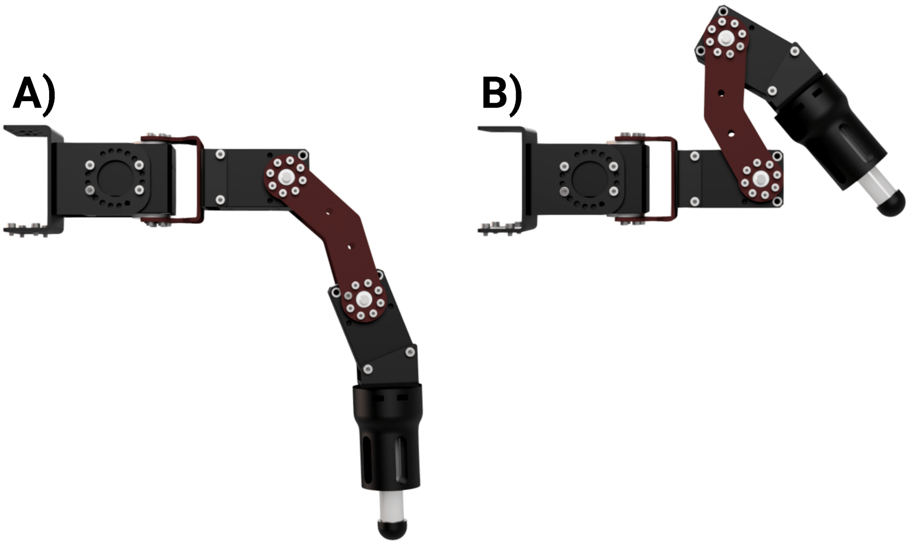


Figure 4.14 – Stepping motion of MORF. **A)** shows the configuration of a leg before a step and **B)** shows the configuration of a leg after a step.

For the remaining part of this thesis the long insect leg configuration with the BC-C-1 and SLIM-C connector, as shown in Fig. 4.11B, will be used. The reason is that this configuration will have the worst case configuration due to the longer legs and higher torque demand. Rotation of the connectors are seen as special cases, and the effect of this will not be investigated. This is likewise the case for the mammal leg shown in Fig. 4.11C. It is, however, the belief that these changes will have close to no impacts on the actuator requirements.

4.2 Body module

The body module is the core mechanical part of the legged robot as it connects all other modules in order to create a unified system. The body will consist of two symmetric plates (**KF12**) perforated with M2.5 sized holes placed in a fixed pattern with 12 mm in-between each hole. Other modules can hereby be connected, via bolts, to the body using connectors with the same pattern (**KF01** and **KF02**). On the top plate, three larger holes are placed in order to allow for cable management between the different layers of the body design (**KF13**).

To determine the overall width of the body an investigation of the dimensions for the most common computers that might be used in the system is needed (**KF14**). The computers investigated was selected through a mixture of feedback from researchers at the Mærsk Mc-Kinney Møller Institute at the University of Southern Denmark and personal experience. Table 4.1 shows a list with the selected computers and their dimensions. From this list, it can be seen that the width of the body module should be at least 104.1mm but was set to 124.0mm, to allow space for a shell.

Table 4.1 – Common computer dimensions.

Computer name	**Width x Length [mm]**
OpenCM 9.04 + Ext. Board	66.5 x 68
Intel NUC i7	106.8 x 104.1
Raspberry Pi 3	56 x 85
Tinker Board	56 x 85
Zybo Board	83.8 x 121.9
Arduiono Uno	53.4 x 68.6
Arduiono Mega	53.3 x 101.5
ArbotiX-M Robocontroller	71.1 x 71.1
NVIDIA Jetson TX2	50 x 87

The length of the body is chosen such that each BC joints will have a worst-case range of motion of 55° (±27.5°) when using six legs (**KF12**). 55° is chosen as a qualitative goal with the main goal of a relatively large stance and swing phase. A large stance and swing phase results in more energy-efficient locomotion as the actuators do not have to change movement direction as often. The required distance between the legs is calculated based on the length of the legs, from which a collision point can be derived. This derivation is done by considering the legs full 180° movement around the BC joints, which result in half-circles as shown in Fig. 4.15.

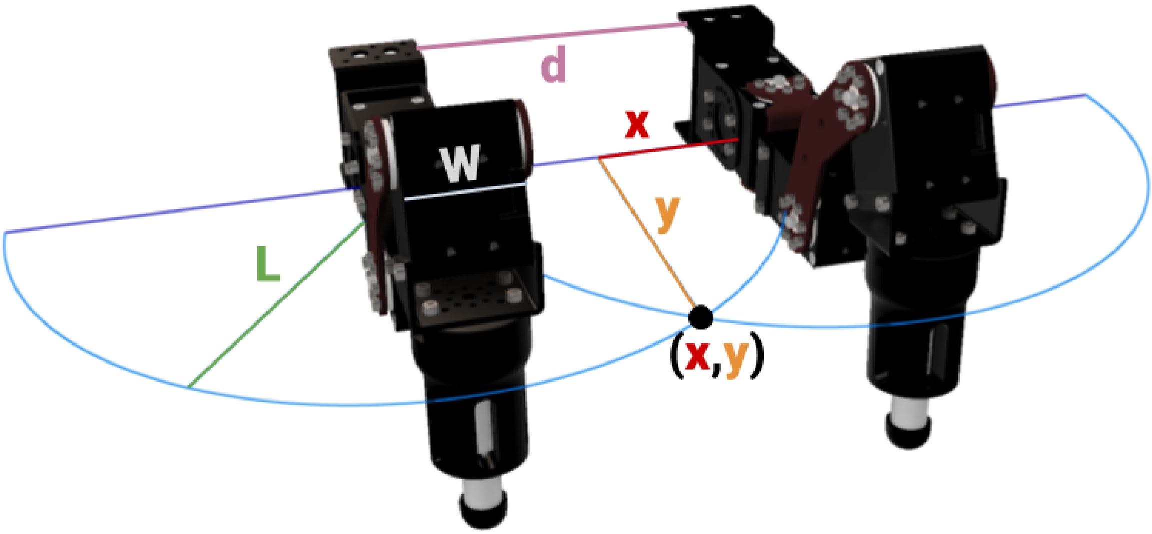


Figure 4.15 – Two legs with width w (shown in white) placed with a distance of d (shown in pink) from each other. The legs swing in a half-circle (shown in blue) with a radius of L (shown in green) and collide in the point (x, y).

The collision point between the two legs is the intersection for the two half circles. This collision point, (x, y), is defined by the coordinates in Eq. 4.1, where d is the distance between the origin of the two half-circles, and L is the horizontal length of the leg. Note that Pythagoras theorem is used to find y.

$$\begin{aligned} x &= \frac{d}{2} \\ y &= \sqrt{L^2 - \left(\frac{d}{2}\right)^2} \end{aligned} \tag{4.1}$$

The angle between the legs when placed perpendicular to the body (the vector in Eq. 4.2) and when at the collision point (the vector in Eq. 4.3) can then be found using Eq. 4.4. Note that the vector for the perpendicular position is offset by the width of the actuator, w, to account for the fact that the leg is not a thin line.

$$\overline{A} = \{w; L\} \tag{4.2}$$

$$\overline{B} = \left\{ \frac{d}{2}; \sqrt{L^2 - \left(\frac{d}{2}\right)^2} \right\} \tag{4.3}$$

$$\theta = cos^{-1}\left(\frac{\overline{A} \bullet \overline{B}}{\left|\overline{A}\right| \cdot \left|\overline{B}\right|}\right) = cos^{-1}\left(\frac{w \cdot \frac{d}{2} + L \cdot \sqrt{L^2 - \left(\frac{d}{2}\right)^2}}{\sqrt{w^2 + L^2} \cdot \sqrt{\left(\frac{d}{2}\right)^2 + \sqrt{L^2 - \left(\frac{d}{2}\right)^2}^2}}\right) \tag{4.4}$$

By isolating d in Eq. 4.4 it is possible to get an expression for the required distance between the legs in order to obtain a desired range of motion, $2 \cdot \theta$, for the BC joints. The resulting equation is shown in Eq. 4.5.

$$d = 2 \cdot \left(\frac{\sqrt{L^2} \cdot w \cdot \cos(\theta)}{\sqrt{L^2 + w^2}} + \frac{\sqrt{L^4 - L^4 \cos^2(\theta)}}{\sqrt{L^2 + w^2}}\right) \tag{4.5}$$

When using a desired range of motion of 55° for the BC joints, an actuator width, w, of 35 mm, and a horizontal leg length[3], L, of 117.75 mm, the required distance between legs, d, is found to be 163.8 mm. This means that the length of the body needs to be at least $2 \cdot d \approx 325$ mm long, as it has to support up to three legs on both sides of the body (**KF12**). Note that the 55° of motion mainly holds for the middle legs, as the front and hind legs are able to either move further forward or backward.

[3] The length from the BC joint axis to the end of the leg when configured like in Fig. 4.11A.

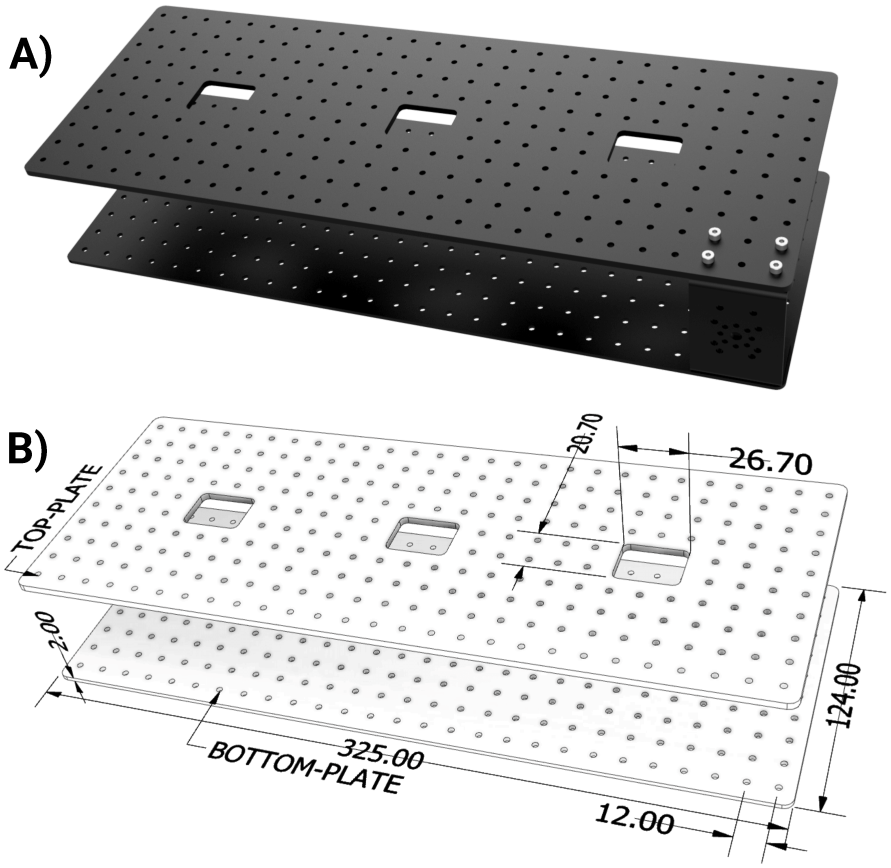


Figure 4.16 – The top and bottom body plates. **A)** shows a realistic rendering of the plates with the BC-C-1 bolted in between them. **B)** shows the two plates and their dimensions. Units are in (mm).

The final body design can be seen in Fig. 4.16. The two symmetric plates are made from 2mm aluminum (**KF12**). A future improvement would be to replace the aluminum with carbon-fiber as this material is known to have a remarkably high strength-to-weight ratio when compared to metals. This is also the case for the FEMUR-C connectors, as they are also flat without bends and therefore easily can be replaced with carbon-fiber.

4.2.1 Body shell

To cover the electronics attached on the TOP-PLATE and to give MORF an appealing visual look a 3D printed shell is designed. The shell is split into two parts. Figure 4.17A shows the head, where various electronics may be mounted (e.g., antennas, LEDs, and buttons). Figure 4.17B shows the body, where nothing should be mounted as it is intended to be taken on and off frequently when the user wants to replace the battery or access the underlying hardware. Figure 4.17C and D shows a side and bottom view of the shell respectively.

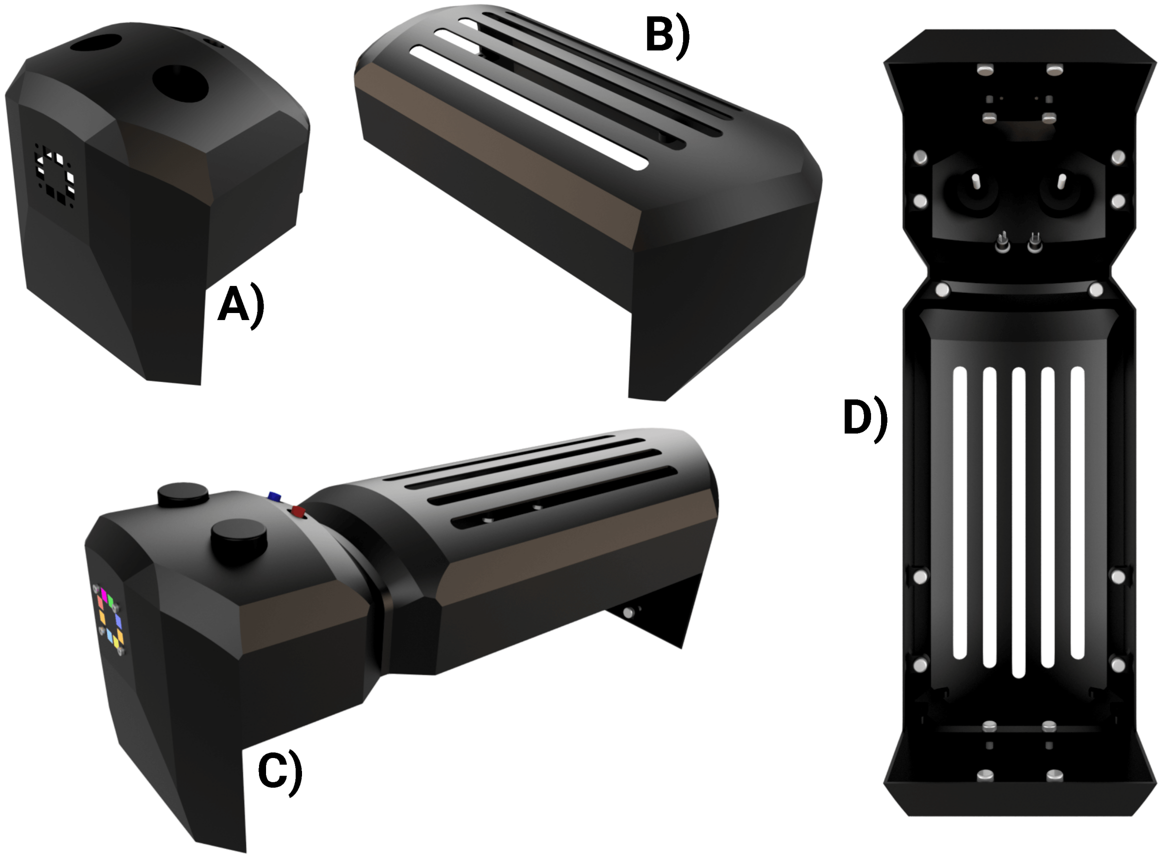


Figure 4.17 – The body shell that encapsulates the two body plates. **A)** shows the head part, where antennas, LEDs, and buttons may be mounted. **B)** shows the body part that is easy to take on and off. **C)** shows a side view of the shell when put together and with the antennas, LEDs and buttons attached. **D)** shows a bottom view of the shell with screws used in a magnetic attachment mechanism.

As can be seen from the bottom view steel screws may be attached, both horizontally and vertically, on the inside of the shell. These are used together with two magnet holders seen in Fig. 4.18 and 4.19 to fix the shell to the plates. The user will in this way be able to quickly and easily gain access to the parts underneath the shell without having to use a screwdriver.

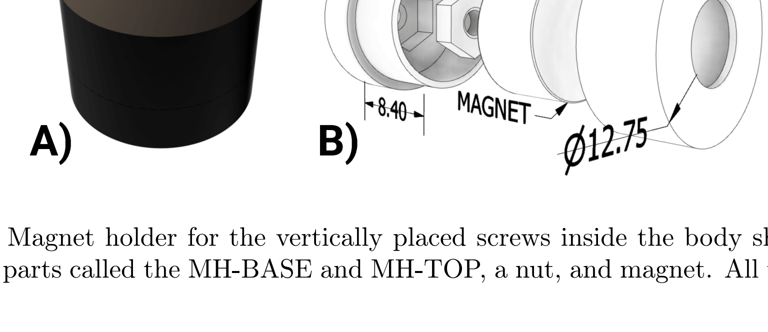


Figure 4.18 – Magnet holder for the vertically placed screws inside the body shell. It consists of two 3D printed parts called the MH-BASE and MH-TOP, a nut, and magnet. All units are in (mm).

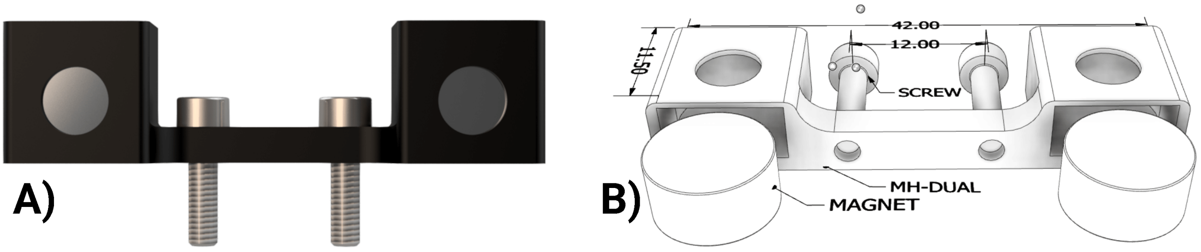


Figure 4.19 – Magnet holder for the horizontally placed screws inside the body shell. It consists of a 3D printed part called the MH-DUAL and two magnets. The screws holes are placed accordingly to the holes in the body plates. All units are in (mm).

4.2.2 Battery mount

Figure 4.20 shows the mechanism for mounting the battery. It consists of two 3D printed parts called BH-END and BH-STICK. Both parts fit in between the two body plates and the battery will in this way be protected from the environment (**KF15**). The BH-END and BH-STICK, furthermore, function as stiffeners between the two body plates thus making the structure even more rigid. The battery will by default be fixed in the middle of the plates to keep the center of mass centered in the robot. The user is, however, free to move it around and it could for example be fixed in the back of the robot to enhance the stability during operations using the two front legs. The 3D printed parts are made accordingly to the dimensions of the battery described in Sect. 4.6, but may be changed to fit other types.

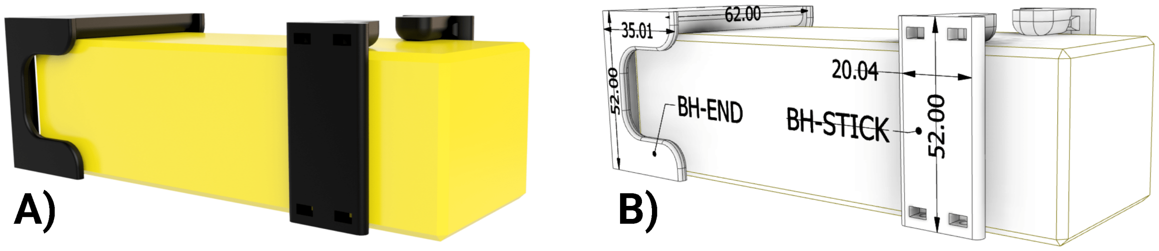


Figure 4.20 – Battery holder consisting of two 3D printed parts called BH-END and BH-STICK. All units are in (mm).

4.3 Complete mechanical design

Figure 4.21 shows CAD models of MORF when fully assembled using the three leg types and the presented mechanics.

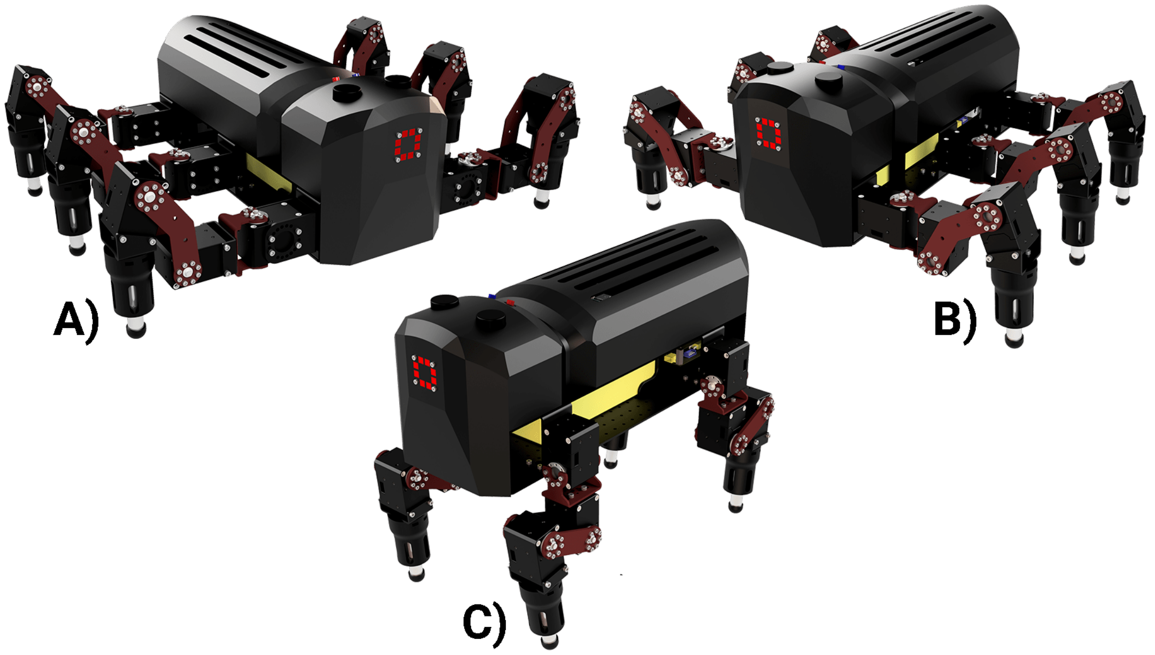


Figure 4.21 – Three different MORF configurations. **A)** shows MORF with long insect legs with the option to rotate the connections. **B)** shows MORF with short insect legs with no rotating connections. **C)** shows MORF with mammal legs.

4.4 Actuator selection

The three leading manufactures of smart servos are; ROBOTIS (Dynamixel series) [54], Dongbu Robots (HerkuleX series) [55], and XYZrobot (XYZrobot series) [56] (**KF10**). To comply with **KF19**, the manufactures has to be a well-established brand from where it is known that future replacements will be available. This is the case for ROBOTIS and Dongbu Robots that both were founded in the year of 1999 in South Korea. XYZrobot is, on the other hand, a rather new company that currently only provides one kind of smart servo (the A1-16), with a rather low stall torque. This is in contrary to **KF10** and **KF19**, and the XYZrobot smart servo is therefore not considered as a potential actuator for MORF.

Both the Dynamixel and HerculeX series can be controlled through a USB to serial interface. Dynamixel has an elaborated software development kit SDK that supports four different operatidng systems, the Robot Operating System (ROS), and eight different programming languages. This is not the case for the HerculeX that only has simple libraries for Arduino and C/C++ with limited documentation.

Even though the above discussion already indicates that the Dynamixel servos are favorable a detailed analysis of their performances is needed in order to make the final choice. This analysis is presented in the following section.

4.4.1 Actuator comparison

It has been decided to compare Dynamixel's XM430-W350 with herkuleX's DRS-0401 as they are almost equally priced and are one of the top actuators from each company (i.e., state-of-the-art smart servos). The comparison of the two actuators is shown in table 4.2. Factors where one actuator is better than the other is highlighted in **bold** and in situations where the difference is negligible, neither of them is highlighted.

Table 4.2 – Comparison between Dynamixel's XM430-W350 with herkuleX's DRS-0401

Specification	**Dynamixel XM430-W350**	**HerkuleX DRS-0401**
Price	\$239.90	\$229.00
Dimension in mm	**46.5 × 28.5 × 33**	56 × 35 × 38
Weight	**82g**	123g
Stall Torque @ 14.8V	4.8 Nm	5.1 Nm
Max Speed @ 14.8V	57 RPM	58 RPM
Operating Angle	**360°**	320°
Resolution	**4,096 step**	2048 step
Standby Current	40 mA	30 mA
Feedback	Position Speed Temperature Load Input voltage Real time tick Overload	Position Speed Temperature Load Input voltage Real time tick Overload
Material	**Case: Metal & Plastic** **Gear: Full Metal**	Case: Full Plastic Gear: Full Metal
Control Algorithm	Current Current based Position Velocity Position Extended Position PWM	More than 50 set-up parameters
Communication	**RS-485[4]** or TTL	TTL

The comparison shows that both actuators provide useful feedback and control properties, but also that the Dynamixel XM430-W350 has a size and weight advantage over HerkuleX DRS-0401 with only a negligible difference in speed and torque. It also shows that Dynamixel XM430-W350 has a larger operating angle of 360° and a finer resolution. For these reasons together with its elaborated SDK, the Dynamixel XM430-W350 is chosen as the actuators for MORF. However, before it can be safely implemented into the framework, the maximum required torque of the actuators must be calculated to see if the Dynamixel XM430-W350 will be able to lift and move the legged-robot system (**KF10**). This analysis is presented in the following section.

[4] RS-485 uses differential signal which is more immune to electrical noise

4.4.2 Calculation of maximum required joint torque

The literature on building walking robots generally shows that four different methods for calculating the required actuator torque exist.

In the "**first**" method the maximum required torque of each joint is calculated independently through simple static force analysis using free body diagrams, point masses, and a safety factor to account for inertial forces [57, 58].

The "**second**" method is less straightforward and requires a Jacobian matrix that relates a change in foot position to a change in joint angle. The idea is then to use this Jacobian to also relate a given force vector exerted by the leg to support and move the body to a vector of joint load torques and use this as the requirements [22, 33, 59].

The "**third**" method aims at finding a dynamic model of the robot leg via Euler-Lagrange. This dynamic model will then, using some software, be simulated to follow a desired trajectory from where the torque in the different joints can be logged and used as requirements [60].

The "**fourth**" and last method is similar to the third but is more dependent on the computer-aided simulation software and physics engine. In this approach, the entire robot, including its walking behavior, is simulated from were the torque values of the joint is logged and used as requirements [61].

In this thesis, method one and four will be applied. The reason for using method four is that a detailed simulation of the robot may also be used for verifying the mechanical design. The reason for using method one is that it is a popular quick and dirty method from which the results can be used to hold up against those of the simulation as a sanity check.

Simple static force analysis - method one

The following calculations assume that the system is supported by three legs that are all fully stretched horizontally (i.e., in their worst case configuration) [57]. Figure 4.22 shows the forces involved in the torque calculations. Note that the tip of the foot is resting on a rigid platform in order to lift the remaining part of the body. $L1$ and $L2$ are measured to be 0.117m and 0.070m respectively (see Fig. 4.12). The expected masses of the different body parts are shown in table 4.3. These are found based on the physical shape and material of the respective part using Autodesk's Inventor.

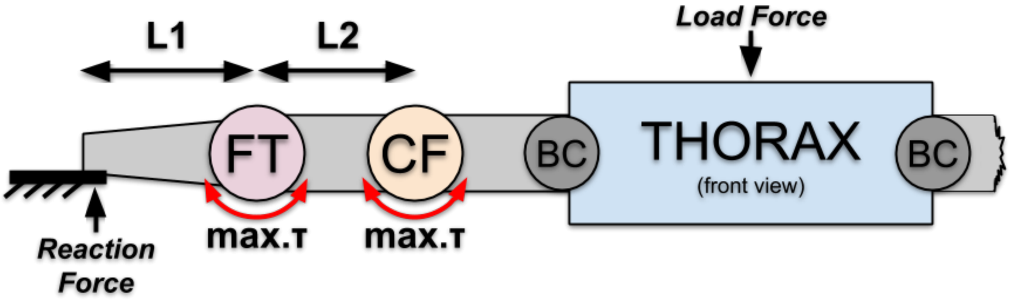


Figure 4.22 – Torque calculations for the CF and FT joints. Three legs are placed on the ground and are thus supporting the remaining three legs and body.

Table 4.3 – Masses of the different body and leg parts when using the MORF configuration shown in 4.21A. All masses are calculated based on the physical shape and material of the respective part using Autodesk's Inventor.

Name	Amount	Weight [kg]	Total Weight [kg]
TOP-PLATE	1	0.201	0.201
BOTTOM-PLATE	1	0.209	0.209
BC-C-1	6	0.016	0.096
BC-C-2	6	0.018	0.108
HORN-C	6	0.011	0.066
SLIM-C	6	0.008	0.048
FEMUR-C	12	0.009	0.108
SLIM-ANGLE-C	6	0.009	0.054
FOOT	6	0.043	0.258
SHELL	1	0.214	0.214
COMPUTER	1	0.300	0.300
DYNAMIXEL SERVO	18	0.082	1.476
BATTERY	1	0.745	0.745
SCREWS est.	-	-	0.100
WIRES est.	-	-	0.050
OTHER ELECTRONICS est.	-	-	0.150
Total	**-**	**-**	**4.183**

By considering the above weights, the reaction force can be calculated as:

$$\begin{aligned}
\mathrm{RF}_{CF} &= \frac{\mathrm{LF}_{CF}}{\text{supporting legs}} = \frac{m_{CF}\cdot g}{\text{supporting legs}} = \frac{3.517kg\cdot 9.82\frac{m}{s^2}}{3} = 11.512N \\
\mathrm{RF}_{FT} &= \frac{\mathrm{LF}_{FT}}{\text{supporting legs}} = \frac{m_{FT}\cdot g}{\text{supporting legs}} = \frac{3.871kg\cdot 9.82\frac{m}{s^2}}{3} = 12.645N
\end{aligned} \tag{4.6}$$

where RF is the reaction force, LF is the load force, m_{CF} is the weight carried by the CF joint (i.e., not including the FT joint, foot, and femur), m_{FT} is the weight carried by the FT joint (i.e., not including the foot), and supporting legs are the number of legs on the ground at the same time, which is assumed to be three as the robot will most likely walk with a tripod gait.

With these two reaction forces it is possible to calculate the maximum required torque in the CF and FT joint as

$$\begin{aligned} \max\_\tau_{CF} &= \mathrm{RF}_{CF} \cdot (L1 + L2) = 11.512N \cdot (0.117m + 0.070m) = 2.153Nm \\ \max\_\tau_{FT} &= \mathrm{RF}_{FT} \cdot L1 = 12.645N \cdot 0.117m = 1.479Nm \end{aligned} \tag{4.7}$$

This means that the CF and FT joint, in the worst case, should be able to produce a torque of 2,153 Nm and 1,479 Nm in order to keep the robot statically stable when resting on three fully stretched legs. However, to also account for inertias and other unexpected forces during dynamical use a safety factor of 20% is added to the torques (as proposed in [57]). This results in a maximum required torque of 2,584 Nm and 1,774 Nm for the CF and FT joint respectively.

Robot simulation - method four

In this section, a simulation of MORF in V-REP is used. The choice and implantation of this will be explained in Sect. 4.7.2 and Chap. 6 respectively. To find the required actuator torque two experiments are created. The setup in the first experiment is similar to that used in method one, as MORF will be statically hanging in three fully stretched legs, see Fig. 4.23.

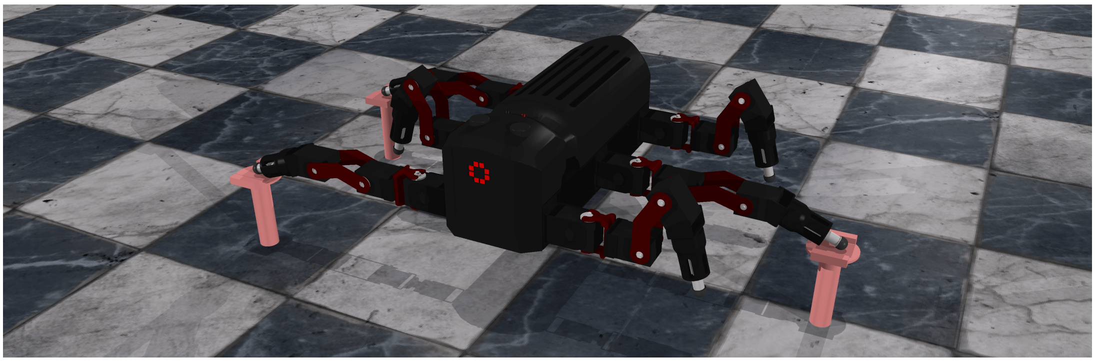


Figure 4.23 – Three fully stretched legs are placed on three light red platforms and are supporting the weight of the remaining three legs and body.

In the second experiment, MORF will be walking using a simple locomotion controller that makes it move using a tripod gait. The main component of the controller is a Central Pattern Generator (CPG), which is a group of interconnected neurons located at the spinal cord of vertebrates and in the thoracic ganglia of invertebrates. A CPG can be activated to generate a motor pattern without the requirement of sensory feedback, and it plays a central role for elucidating locomotion mechanisms and other rhythmic movements like berating [62, 63]. In the domain of robot control, most of the research has employed abstract CPG models using coupled oscillators to generate basic periodic patterns of movement [64]. In the following, the abstract CPG model called the

SO(2) or two-neuron oscillator will be explained and afterward applied on MORF [65].

The pure SO(2)-network consists of two ($N = 2$) fully connected neurons, H_0 and H_1, as shown in Fig. 4.24.

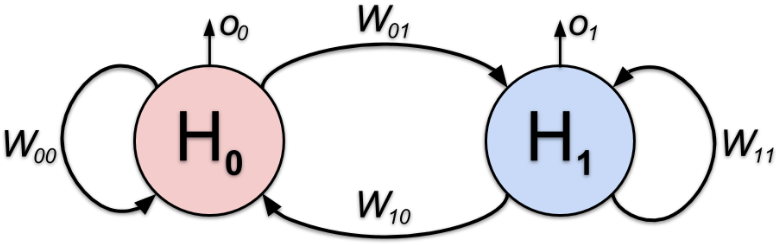


Figure 4.24 – The SO(2) CPG with two fully connected neurons (H_0 and H_1). $W_{\mathtt{ij}}$ denotes the synaptic weight from neuron i to j.

Both neurons use a sigmoid transfer function, and their outputs are given by

$$o_i(t+1) = tanh\left(\sum_{N-1}^{j=0} w_{ij}(t)o_j(t)\right), \quad i = 0, ..., N-1. \tag{4.8}$$

where o_i is the output from neuron i, N is the number of neurons, and w_{ij} is the synaptic weight from neuron i to j. The synaptic weight matrix is chosen according to

$$\begin{pmatrix} w_{00}(t) & w_{01}(t) \\ w_{10}(t) & w_{11}(t) \end{pmatrix} = \alpha \cdot \begin{pmatrix} \cos\ \phi(t) & \sin\phi(t) \\ -\sin\ \phi(t) & \cos\phi(t) \end{pmatrix} \tag{4.9}$$

with $0 < \phi(t) < \pi$ as the frequency determining parameter. The parameter α determines the amplitude and the nonlinearity of the oscillations. For this controller $\alpha = 1.01$ is used to obtain a very harmonic oscillation and an approximately linear relationship between ϕ and the intrinsic frequency of the oscillator [63].

The raw CPG output signals are visualized in Fig. 4.25A. This shows that the two outputs are phase shifted by $\frac{\pi}{2}$ with an amplitude of 0.2. By connecting H_0 to the BC joint and H_1 to the CF joint, the legs will be lifted while moving forward and grounded while moving backward. Figure 4.25B shows the post-processed output signals given as position commands to the actuators. Both outputs are scaled in amplitude such that each leg is moved in a way similar to insects while not colliding with other legs. The output of H_1 is furthermore thresholded to ensure stable movement when the legs are in contact with the ground. Note that in both plots ϕ is regulated to show different frequencies.

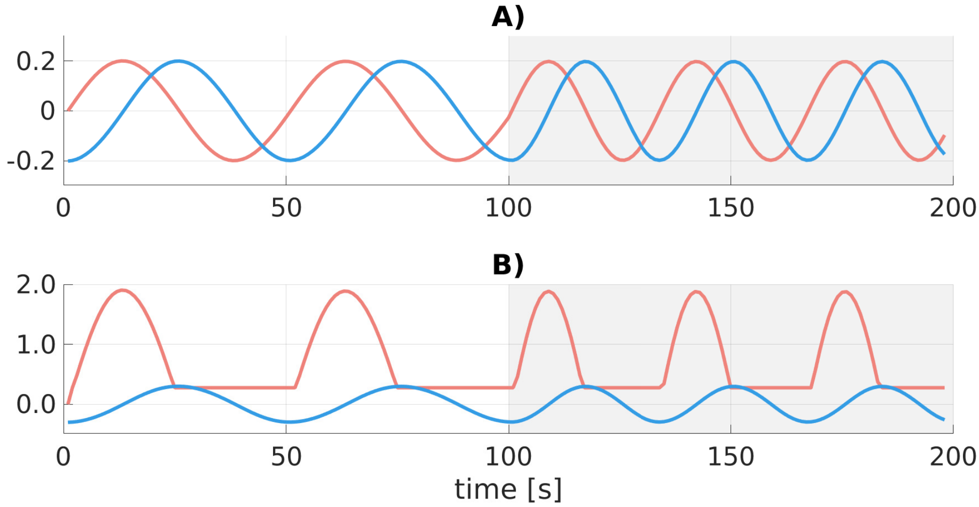


Figure 4.25 – The output signals of the SO(2) oscillator. From 0-100s, ϕ is set to 0.13 (white area) and from 100-200s it is set to 0.19 (gray area). **A)** shows the raw output signals from the oscillator and **B)** shows the post-processed signals given as position commands to the actuators.

The outputs are, as explained, connected to all BC and CF joint while the FT joints are set to a static position for simplicity. These connections are weighted, as shown in Fig. 4.26, such that it will generate a tripod gait where the left front (LF) and hind leg (LH) will move together with the right middle leg (RM), and the right front (RF) and hind leg (RH) will move together with the left middle leg (LM).

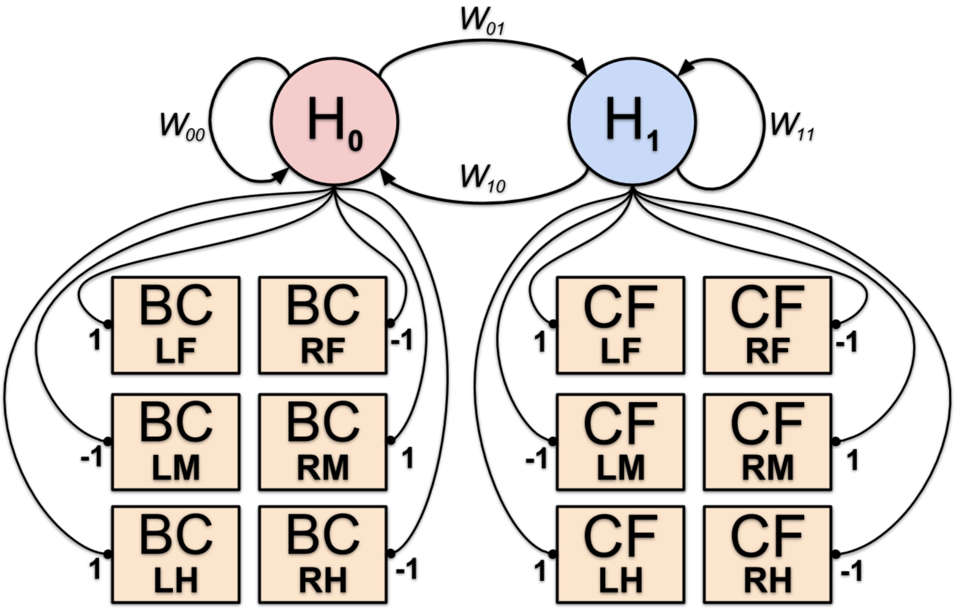


Figure 4.26 – Connection of the CPG to the leg joints (orange blocks). LF is left front leg, RF is right front leg, LM is left middle leg, RM is right middle leg, LH is left hind leg, and RH is right hind leg. The post-processing shown in Fig. 4.25 is made after the weighted connections. The FT joints are set to a static position.

The simulated version of MORF when using the network in Fig. 4.26 and the post-processing in Fig. 4.25B can be seen in Fig. 4.27 and in a small video using the following link: `https://youtu.be/tq25qsIGvNM`.

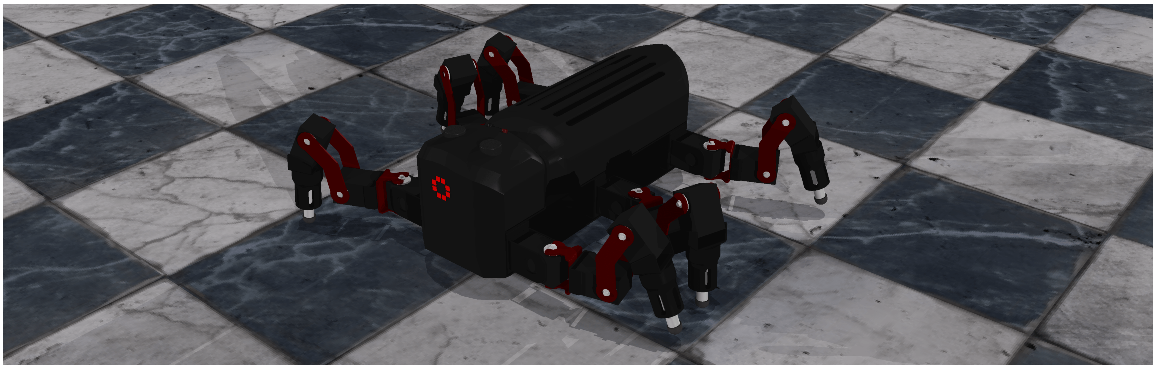


Figure 4.27 – The simulated version of MORF using the network in Fig. 4.26 and the post-processing in Fig. 4.25B to walk with a tripod gait.

Results

Figure 4.28 shows the torques in the three actuators of the right front leg (RF) and left middle leg (LM) during the static stand seen in Fig. 4.23. Note that the LM leg is the only leg on the left side that supports the weight of the robot, while there are two supportive legs on the right side.

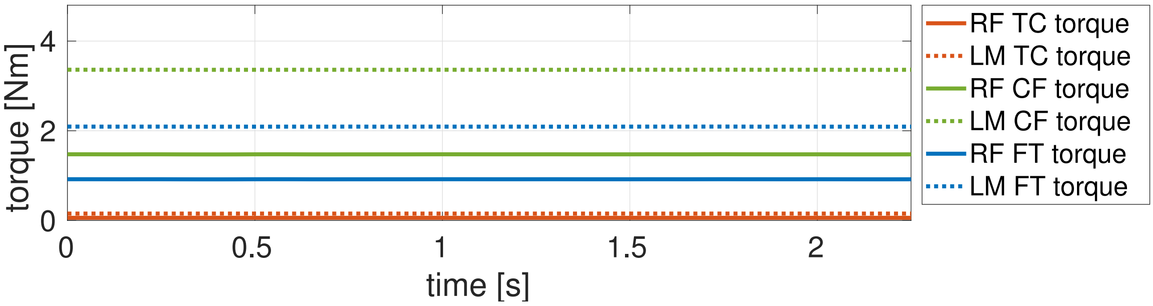


Figure 4.28 – The torques in the three actuators of the right front leg (RF) and left middle leg (LM) during the static stand seen in Fig. 4.23. Note that TC (thorax-coxa) is the same as BC.

Figure 4.29 shows the torques in the three actuators of the right front leg (RF) when the robot is walking with a tripod gait using the controller described in the previous section. Note again that there will always be a front and hind leg in ground contact on either the right or left side of the robot.

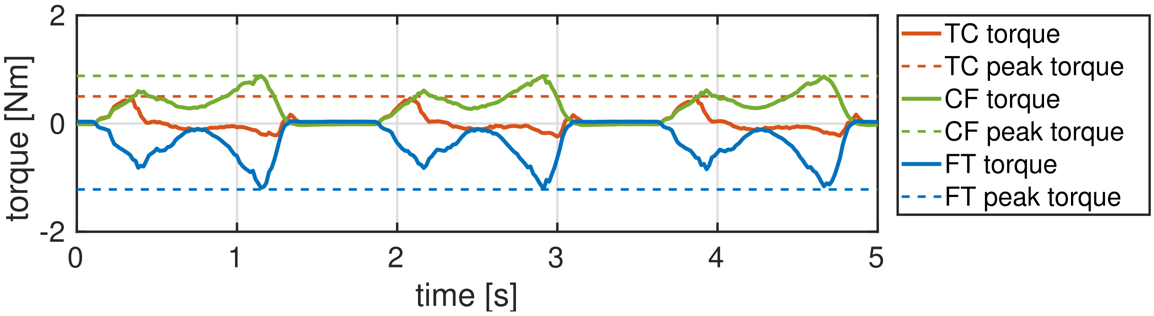


Figure 4.29 – The torques in the three actuators of the right front leg (RF) during a tripod gait. Note that TC (thorax-coxa) is the same as BC.

Figure 4.30 shows the torques in the three actuators of the left middle leg (LM) when the robot is walking with a tripod gait using the controller described in the previous section. Note that the middle leg always will be the only leg in ground contact on either the right or left side of the robot.

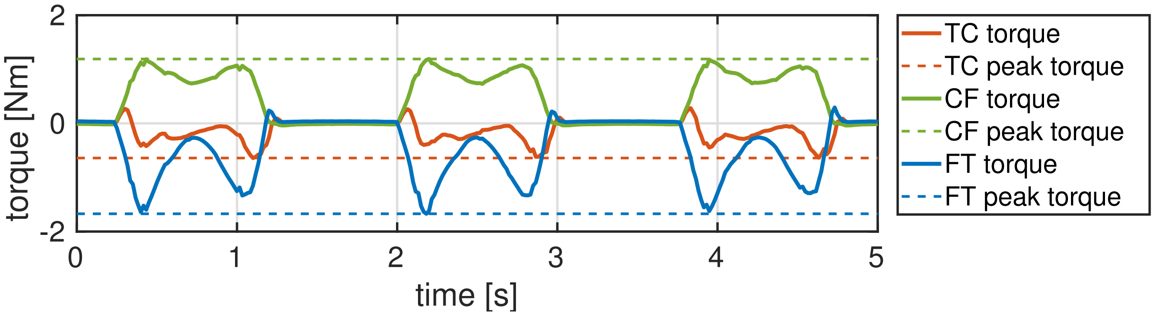


Figure 4.30 – The torques in the three actuators of the left middle leg (LM) during a tripod gait. Note that TC (thorax-coxa) is the same as BC.

Discussion

When comparing the static torques for the CF and FT joint found using simple static force analysis (method one) and robot simulation (method four) it can be seen that they are quite different, see table 4.4. Using method four, two different torques were found. These torques depend on whether the leg is alone on one side of the robot (single leg) or shares the side with another supporting leg (dual leg) as shown in Fig. 4.23. This is also why the average torque between the single and dual leg yield a result similar to that of method one (only 7.3% and 1.8% difference for the CF and FT joint respectively). It can thus be said that method one does not account for the geometry of the robot and assumes that the load is equally distributed among the supporting legs. For future projects that requires calculation of torque demands for a specific robot, it may, therefore, be an advantage to use a robot simulation like V-REP as it accounts for the geometry of the robot, which simple static force analysis does not.

Table 4.4 – Joint torques from the static tests found using method one and four. Single leg is for the joints in the leg that alone on one side of the robot and dual leg is for joints in the leg that shares the side with another supporting leg.

	CF joint torque [Nm]	**FT joint torque [Nm]**
Method One	2.153	1.479
Method Four - *Dual leg*	1.353	0.919
Method Four - *Single leg*	3.284	2.093
Method Four - *Average*	2.318	1.506

Another advantage of using a simulation (method four) is the possibility of measuring the joint torques while the robot is moving with its expected behaviors (i.e., a dynamic torque analysis). Results from the experiments where MORF walked with a tripod gait also showed that there is a difference in torque between the single and dual leg. The test furthermore shows that the BC joint is not that torque demanding, which could not be determined using method one alone.

Based on the results from the static and dynamic torque analysis it can be concluded that the Dynamixel XM430-W340's can carry MORF's own weight while having a margin for additional payload. The servos are furthermore relatively fast (57 RPM at 14.8V). However, if MORF needs to be faster, the BC joints could be replaced with the Dynamixel XM430-W210 that has the same dimensions as the XM430-W340 but are faster (95 RPM at 14.8V) with reduced stall torque (3.7Nm at 14.8V).

4.5 Computer module

To comply with **KF16**, the computer module should be able to handle a process-heavy locomotion controller, hardware communication, and wireless communication at the same time.

The computational demand of a locomotion controller varies a lot based on its complexity. Some of the most requiring controllers are the once using neural networks and especially deep neural networks. Such controllers often need a high-end CPU and possibly also a GPU to run at reasonably speeds. However, since it will be challenging to fit a large and power consuming GPU onto MORF, only the CPU will be taken into account. Instead, the GPU can be used on an external computer that communicates with the onboard computer via the wireless communication.

An ideal computer choice is a mini PC as they are usually small and equipped with a strong CPU, a WiFi module, a Bluetooth module, USB connectors, and upgradeable storage. In the following, four mini PCs are listed and compared based on CPU, weight, and dimensions. The initial requirements for the four PCs were that they are not much heavier than 1.0kg and that they are equipped with a high-end processor (e.g., Intel i7). The four mini PCs has the following specifications:

Asus VivoMini:

- CPU: Intel i7-7500U 3.5GHz
- Weight: 0.7kg
- Dimensions: 13.1 x 13.1 x 5.2cm

HP Elite Slice:

- CPU: Intel i7-6700T 3.6 GHz
- Weight: 1.04kg
- Dimensions: 16.5 x 3.5 x 16.5cm

MSi i7 Cubi-028BUS:

- CPU: Intel i7-5500U 3.0GHz
- Weight: 0.7 kg
- Dimensions: 11.5 x 11.1 x 3.5cm

Intel NUC board NUC7I7DNBE:

- CPU: Intel i7-8650U 4.2 GHz
- Weight: 0.3kg
- Dimensions: 10.1 x 10.1 x 2.7cm

From the above information, it is clear that the Intel NUC is the better choice. It is equipped with the best CPU version, it has the most compact form, the lowest weight, and is the only PC that ships without a case. The Intel NUC is, therefore, to be implemented on MORF.

4.6 Battery selection

To comply with **KF17**, the system should be powered by an onboard battery that can last for around 1.5 hours of use. Like almost all other mobile robotic systems MORF will be equipped with a Lithium Polymer (LiPo) battery. When compared to conventional Nickel-Cadmium and Nickel-Metal Hydride batteries, LiPos have the following pros and cons [66].

Pros

- Much lighter and can be made in almost any size or shape.
- Much higher capacities, allowing them to hold much more power.
- Much higher discharge rates, meaning they pack more punch.

Cons

- Much shorter lifespan; LiPos average only 150–250 cycles.
- The sensitive chemistry can lead to a fire if the battery gets punctured.
- Require special care for charging, discharging, and storage.

The performance of a LiPo battery is defined by three main parameters; Discharge (C) rating, Capacity, and Cell Count/voltage. Discharge rate is how fast a battery can be discharged safely. Capacity indicates how much power the battery pack can hold, typically in indicated in milliampere-hours (mAh). Cell Count indicates how many cells the LiPo contains. A LiPo cell has a nominal voltage of 3.7V, so a two-cell (2S) pack is 7.4V, a three-cell (3S) pack is 11.1V, and so on [66]. For MORF a 6S battery (22.2V) is used together with a DC-DC converter for transforming the 22.2V to 14.8V. The reason for doing this is that the servos should have a constant voltage below their maximum input voltage limit of 16V. This is not possible when connecting a LiPo battery directly to the servos as a single LiPo-Cell discharge from around 4.2V to around 3.3V during each cycle. A 4 celled LiPo battery would, in this case, cause an input voltage error in the servos as it is charged to $4 \cdot 4.2 = 16.8V$ and would neither be able to provide a constant voltage to the servos. The consequence of this would be a robot that slowed down as a result of discharging the battery, which is not desired when using the platform for scientific experiments.

Figure 4.31 shows a power board designed at SDU by the electronics technicians in the Maersk Mc-Kinney Moller Institute. The board houses the DC-DC converter, mentioned above, which can deliver a maximum of 20A. Since the stall torque of each servo is 2.7A at 14.8V a total of three boards are needed, i.e., one board for two legs. This also makes enough room for connecting the Intel NUC to one of the power boards power jack socket.

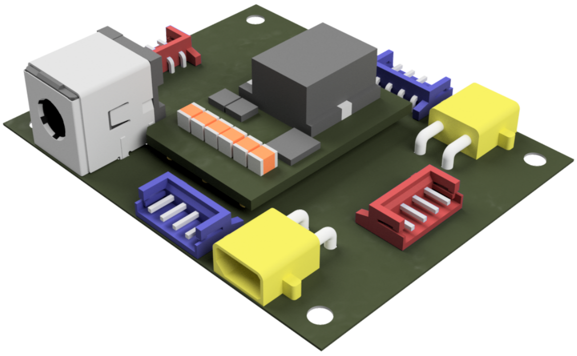


Figure 4.31 – The power board. The gray brick in the middle of the board is a DC-DC converter that transforms 22.2V from the battery to 14.8V for the servos and the Intel NUC. The blue connectors contain power and data, while the red ones only contain data. The yellow XT30 connectors are for the LiPo battery, and the gray power jack socket is for the Intel NUC.

Besides working as a DC-DC converter, the power board also acts as a hub for the Dynamixel servos. The 4pin connectors shown in blue are connected to the legs, providing both data and power. The 4pin connectors shown in red provides only data and are used to connect the power boards, computer, and thus all of the servos to each other. Finally, the yellow sockets (XT30 connectors) are for the LiPo battery.

In the following, the required battery capacity will be found using the requirement that MORF should be able to walk continuously for around 1.5 hours. Under normal usage (i.e., when walking with a tripod gait) the simulation results presented in Sect. 4.4.2 shows that all the servos on average will use a torque of $4.5Nm$ which is equal to $2.7A$ (for this conversion the linear torque-current relationship from the servos datasheet is used). This together with the $\sim 1A$ used by the Intel NUC gives a total current usage of $3.7A$. Using the following equation, the required capacity is calculated as

$$C = T \cdot L \tag{4.10}$$

where C is the capacity in ampere-hours, T is the desired run time in hours, and L is the current usage in ampere. When using $T = 1.5h$ and $L = 3.7A$ the capacity requirement is given as

$$\mathrm{C} = 1.5\mathrm{h} \cdot 3.7\mathrm{A} = 5.55\mathrm{Ah} = 5550\mathrm{mAh} \tag{4.11}$$

Based on the above calculations the ZIPPY Compact 5800mAh 6S 25C LiPo Pack with a weight of 745g is to be used on MORF. The discharge rating of 25C enables the battery to deliver $5800\mathrm{mAh} \cdot 25\mathrm{C} = 145.000\mathrm{mA} = 145\mathrm{A}$ which is more than enough. For longer trips up to 2 hours and 15 minutes the Multistar High Capacity 8000mAh 6S 12C LiPo Pack with a weight of 1110g may be used[5]. One does, however, need to 3D print a new battery holder that fits this slightly larger battery. Both the ZIPPY and Multistar battery was chosen over other brands due to their high specific energy.

4.7 Software

4.7.1 Hardware interface

To comply with **KF20** the system need to include a complete software suite that contains methods for reading sensor values and controlling the hardware in a fast and scalable way. To do this an operating system or middle-ware which can read, write, and distribute hardware information in a scalable way is needed. Many of such systems are available for robots, and each has its strengths and weaknesses [67]. For MORF the Robot Operating System (ROS) [68] will be used as it has been very successful especially in research. ROS is a highly modular and scalable system in which code is distributed into individual nodes. It follows the publish-subscribe structure where nodes can either publish and/or subscribe to a topic independent of the data formats (**KF02**). ROS is furthermore open source which limits the required maintenance and also makes it easy to scale the system (**KF02**). An example would be if the user were to add a Kinect camera on top of MORF. In this case, the user would only have to connect the camera to the onboard PC and install the already developed *kinect_camera* node for ROS[6].

[5] In this calculation the additional weight of the larger battery is neglected.

[6] Link to the Kinect camera ROS node: `http://wiki.ros.org/kinect_camera`

The software on MORF should be divided into two types of nodes; hardware interfacing nodes and locomotion controller nodes. The hardware interfacing nodes will be responsible for reading and publishing sensory information from the sensors and servos as well as subscribing to servo commands. It should do this with a rate higher than 60Hz as it with this is possible to get fairly smooth feedback (**KF20**). The locomotion controller nodes will subscribe to all the sensor values from the hardware interfacing nodes and publish position commands. It will in this way be possible for the user to implement the locomotion controller node in any programming language supported by ROS. For this project, the locomotion controller explained in Chap. 4, Sect. 4.4.2 is implemented in C++ and may be used as a template for future controllers.

4.7.2 Simulation

To comply with **KF21** the software suit should also include a complete and realistic simulation of MORF. When choosing a simulation platform, a variety of requirements has to be fulfilled. As stated in [69], the most important criteria, from most important to least important, based on a online survey is

- Realistic simulation
- Open source
- Light and fast
- Same code for real and simulated robot
- Customization
- Easy to learn/use
- Real-time based simulation

Framework comparisons

With respect to the above criteria, three popular simulation frameworks are investigated. The first is lpzrobots developed by Leipzig University [70]. This framework fulfills most of the criteria, but the simulation is complicated to use (no GUI) and not that realistic (e.g., it is unitless). There is thus no guarantee that the implemented controller will work on the real-world robot. By failing these criteria, lpzrobots will not be considered.

The second framework investigated is Gazebo [71]. Gazebo supports ROS by default unlike lpzrobots, and the simulated software may, therefore, be easily transferred directly to real hardware. The framework is closer to reality and it also offers multiple physics engines (Open Dynamic Engine [17], Bullet [72], Simbody [73], and DART [74]). Gazebo, however, falls short in the customization of the simulated world and ease of use. Objects are difficult to construct, as they have to be imported using third-party tools and then convert it to Gazebo's own format [75].

The third and last simulation framework investigated is V-REP (Virtual Robot Experimentation Platform) developed by Coppelia Robotics [76]. V-REP is similar to Gazebo as it supports ROS by default, offers very realistic simulations, and includes multiple physics engines (Bullet [72], Newton [77], Open Dynamic Engine [17], and Vortex [78]). Especially the vortex physics engine is a great advantage of V-REP as it offers many real-world parameters (i.e., corresponding to physical units) for a large number of physical properties, making this engine both realistic and precise [79]. V-REP is superior to Gazebo in customization since it is very intuitive to use with custom toolboxes and easily adjustable settings for every shape, object, and sensor. There are also pre-made objects that can be dragged and dropped directly into the simulated world, enhancing the user experience by being very simple and easy to use. Furthermore, there is no need for converting 3D models to a specific format, as they can be imported directly into the simulator and then customized as needed [80]. Although it uses significantly more processing power than LpzRobots, V-REP is still faster to run than Gazebo [75].

Form the above analysis it should be clear that V-REP with the Vortex physics engine is the better simulation framework, as also indicated in [81, 75, 69]. V-REP is, therefore, to be used for implementing the simulated versions of MORF (**KF21**).

Reality Gap

Even though V-REP contains realistic parameters, it is not given that a controller evolved or developed in the simulation will transfer successfully into reality, i.e., crossing the reality gap. In [48] N. Jakobi discussed the reality gap and outlined new ways of thinking about and building simulations. One of his points is that "It is impossible to build a simulation that is a perfect copy of the real world" [48] and even if it is possible such a simulation would be slow and take a long time to validate empirically. N. Jakobi mentions two reasons why a simulation will differ from a perfect copy; it models only a finite set of real-world features, and those features will be modeled inaccurately.

The main contribution of N. Jakobi in [48] is the introduction of the radical envelope-of-noise hypothesis, a general methodology for how to construct simulations. The methodology combines two general approaches to reducing the reality gap. The first is to improve the simulation accuracy either analytically or in a data-driven way. The second it to accept the imperfections of the simulation and instead aim to make the controller robust to variations. Such robustness can be achieved by randomizing various aspects of the simulation: adding noise to sensor values, randomizing the dynamics, and perturbing the system with random disturbances [82]. N. Jakobi claims that by using the methodology, controllers may be evolved and transferred into reality no matter how inaccurate or incomplete the simulation is.

The methodology can be summed up in three steps:

1. A set of desired robot-environment interactions (that are sufficient to underlie the behavior to be evolved) must be identified and made explicit. The simulation should include accurate models of these interactions and will in this way contain desired interactions aspects, which have a basis in reality, as well as implementation aspect, which do not.
2. Every implementation aspect of the simulation must be randomly varied from trial to trial so that evolved controllers that depend on them are unreliable. In particular, enough variation must be applied to ensure that the evolved controller actively ignore each implementation aspect entirely.
3. Every desired interaction aspect of the simulation must also be randomly varied from trial to trial so that evolved controllers are forced to be robust. This is to ensure that the controllers can cope with the inevitable differences between the simulation and reality. The amount of variation should be less than for the implementation aspects such that controllers do not fail to evolve at all.

The implementation of the simulated version of MORF should take into account the above points and methodology. It is difficult to predict what kind of desired robot-environment interactions the user will focus on, but it is expected that the joint position, joint velocity, joint torque, and to some extent the body orientation will often play key roles. These interactions should, therefore, be implemented such that their sensory feedback is reasonably close to reality. If the user would like to evolve and design a controller that depends on other interactions or more accurate models, he or she will have to calibrate and validate these them self.

4.8 Parts list

Table 4.5 show a list of all the parts needed to build MORF. It also lists the expected prices which comply with **KF18** as the system is not far from 50.000 DKK in total.

Table 4.5 – MORF parts list and estimated prices

Part Name	Price estimate
Aluminum parts - leg connectors and body pates	3000 DKK
Dynamixel XM430-W350 (18 pcs.)	27000 DKK
Intel NUC i7	3739 DKK
SSD 120GB	243 DKK
RAM 8GB	647 DKK
3D Printed Shell	250 DKK
Blinkstick Square	150 DKK

Table 4.5 continued on next page

Table 4.5 continued from previous page

Antennas (2 pcs.)	300 DKK
Foot shell (6 pcs.)	150 DKK
Aluminum pipe (6 pcs.)	2000 DKK
Rubber tip (6 pcs.)	150 DKK
3D Force sensor (6 pcs.)	9000 DKK
Adafruit 9-DOF Absolute Orientation IMU	250 DKK
ZIPPY Battery (2 pcs.)	1000 DKK
Power distribution board (3 pcs.)	1000 DKK
Other - magnets, wires, 3D prints, screws, etc.	2000 DKK
Total:	50879 DKK

Note that all of the parts are from well-established brands or can be produced in house and are thus easy to reorder in case of replacements or repairs(**KF19**).

4.9 Design verification

In this chapter a design that complies with almost all key features from Chap. 3 was described. The only KF that is not fully met is **KF13** - "*a convenient fastening method*". Currently, modules are attached using screws and nuts with the only exception being the shell that is connected using magnets. It may be tedious and time-consuming to reconfigure the robot as one would have to unscrew all screws. However, for now, this seems to be the most robust solution, and maybe the only one as the Dynamixel smart servos also requires the use of screws. One improvement could be to look into some *click* mechanisms for the body-leg connections such that the number of legs could easily be changed. By finding a better fastening method, it might be easier to stress the modularity of the robot.

Chapter 5

Requirement specification for MORF

Table 5.1 on the next page summarizes the requirements extracted from the design analysis (Chap. 4) and supports the implementation presented in the next chapter (Chap. 6). The requirement specification will also be used to validate the final and implemented system (Chap. 7). The measurement and type of validation performed is noted in the table. All of the specifications with *design* as type of validation will be addressed in the implementation chapter (Chap. 6), while specifications that requires a *test* are addressed in the validation chapter (Chap. 7).

Recall that the deliverable is a framework that includes a modular legged robot as well as a software suite with hardware interfaces and simulations.

Table 5.1 – Requirement Specification.

Requirement	**Insp.**	**Description**
Hardware		
R01: Mechanical design	Design	All parts are based on the design from Chap. 4
R02: Controller	Design	An Intel NUC w. WiFi is used as controller
R03: Cable management	Design	Cables are easy to follow and do not hang loose
R04: Joint type	Design	Dynamixel XM430-W350 is used as joints
R05: Modular	Design	Legs can be reconfigured
R06: Foot compliance	Design	The foot is compliant
R07: Step height	Test	Legs have a minimum step height of 174mm
R08: Joint torque	Test	The joints can carry additional payload
R09: Run time	Test	MORF can walk for 1.5 hours w/o recharge
R10: Stable power source	Test	Voltage to the servos and controller is constant
Software - Hardware interface		
R11: System interface	Design	Can be controlled and programmed over WiFi
R12: Hardware Interface	Design	Hardware interfaces are placed in ROS nodes
R13: Locomotion controller	Design	Locomotion controller is placed in a ROS node
R14: Fast feedback	Test/Design	The feedback rate from sensors is above 60Hz
R15: Computer	Test	The Intel NUC can handle wireless communication, locomotion controller, and hardware interface at the same time
Software - Simulation		
R16: MORF simulation	Design	An accurate V-REP simulation of MORF
R17: Portable controller	Design	Controller works in simulation and reality
R18: Realistic simulation	Test/Design	Small reality gap between simulation and reality
Other Specifications		
R19: Reasonable cost	Test/Cost	Can be implemented for around 50.000 DKK
R20: Scalability	Test	Is easily scaled using new modules and sensors
R21: Usability	Test/Survey	Is applicable in research

Note that the force sensor is not apart of the requirement specification and will not be implemented before the submission of this thesis. The reason is, as mentioned earlier, that the 3D force sensor used in the new foot design (appendix A) is still under development and will subsequently undergo a patenting process. In the mean time the torque in the CF and FT joint may be used as approximations for the foot contact force.

Chapter 6

Implementation of MORF

In this chapter, the process of implementing MORF is described. The chapter will consist a hardware and a software section. In the hardware section, a prototype of MORF and its assembly using the intended parts are presented. Additionally, new components, not discussed in the previous chapters, are introduced. In the software section, the software suite and its structure is introduced. This includes; setup of the onboard Intel NUC PC, implementation of the hardware interfaces, and implementation of the V-REP simulation.

Every time a requirement from the previous chapter is addressed it will be indicated in the following way: (**R##**), where ## is the number of the requirement.

6.1 Hardware

6.1.1 Prototype

Before ordering the mechanical parts of MORF, a prototype, as shown in Fig. 6.1, was made. The main purpose of the prototype was to verify the design and to look for possible improvements. All parts of the prototype are 3D printed except the body plates and FEMUR-C connector which are made from laser cut wood. Besides working as a design validation, the prototype may also be used to display different leg configurations.

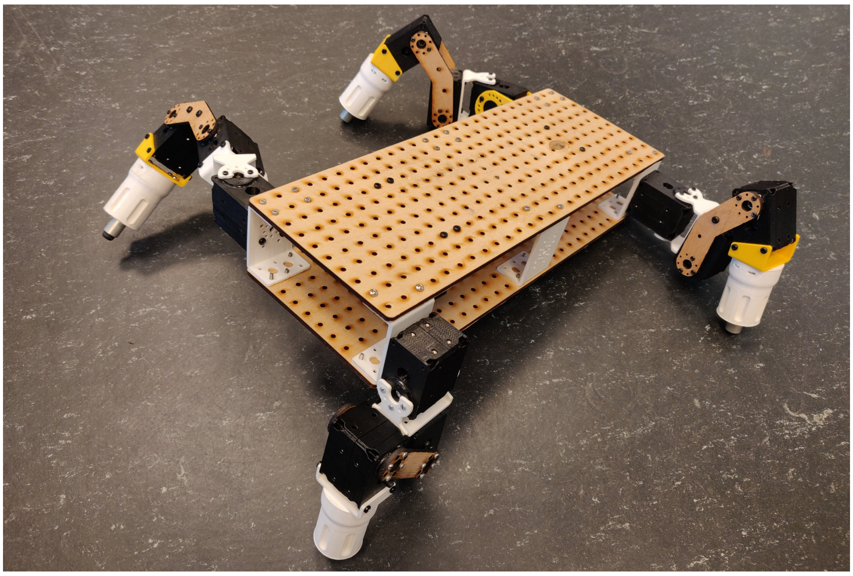


Figure 6.1 – The prototype of MORF made from 3D print and laser cut wood.

The prototype showed that the design worked as intended, but the following changes were made to improve it even more[1]:

- Holes for wires in the top plate
- Additional space around the holes of the HORN-C connector
- Additional space between the two body plates to make space for larger LiPo batteries
- Less weight reduction holes in the FOOT-SHELL as they made it too brittle
- Additional space for the horn hole in the FEMUR-C connector so that wires can pass through

6.1.2 Mechainical assembly

To assemble MORF all of the parts specified in table 4.5 was ordered and produced (**R01**). Every 3D printed part was produced in-house while all the aluminum parts were ordered from an

[1] Note that these changes were already introduced/implemented in Chap. 4.

external manufacturer. The aluminum parts were, furthermore, powder coated to give MORF a more visually appealing and smooth look. Powder coating also has the added benefit of protecting against short circuits as it makes the aluminum non-conductive. Both the 3D files for printing and the work drawings for the manufacturer can be found in the supplementary materials in the `Mechanical_parts` directory.

The actual assembly is carried out as specified in Chap. 4. Most parts, including the Dynamixel XM430-W350 servos, are connected using screws and nuts (**R04**). For these connections, it was necessary to also include washers as the vibrations generated when walking made the screws loose.

In order to attach the 3D printed body shell to the two body plates, the magnetic connections are used. The magnet holders are fastened to the body plates such that the screws fastened to the body shell are aligned with the magnets (see placement in Fig. 6.2). This mechanism proved to be extremely useful when changing the battery or accessing the underlying electronics. A video illustrating how it works can be seen using the following link: `https://youtu.be/DxmenC5Pdzo`.

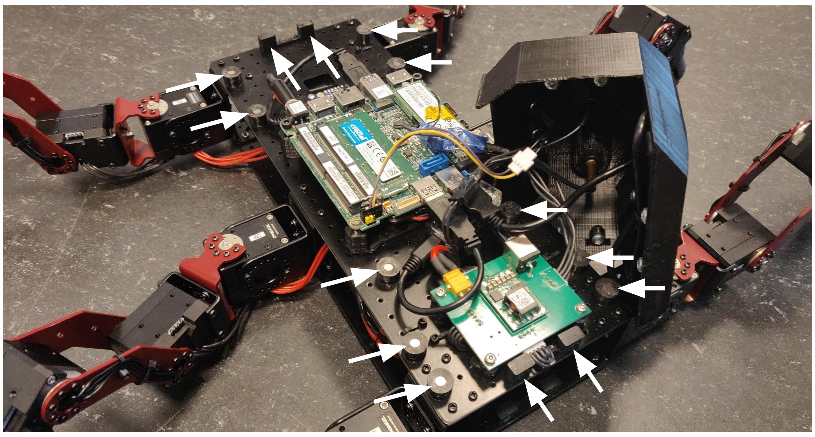


Figure 6.2 – The magnets used for attaching the body shell to the body plates. The magnet placements are indicated with white arrows.

Finally, the springs in the feet of MORF are chosen to have a spring constant of 1.0 N/mm as that makes the legs compliant enough to deal with impact and external forces without making the robot too shaky when walking[2] (**R06**).

6.1.3 Electronics

In addition to the assembly of the mechanical parts, various sensors and electronics are also attached on MORF. For the Intel NUC and power boards custom 3D printed stands were used

[2] The reason for not determining the spring constant experimentally is that the future revision of the foot will replace it with silicone.

to attach them directly to the body plates (**R02**), see Fig. 6.3.

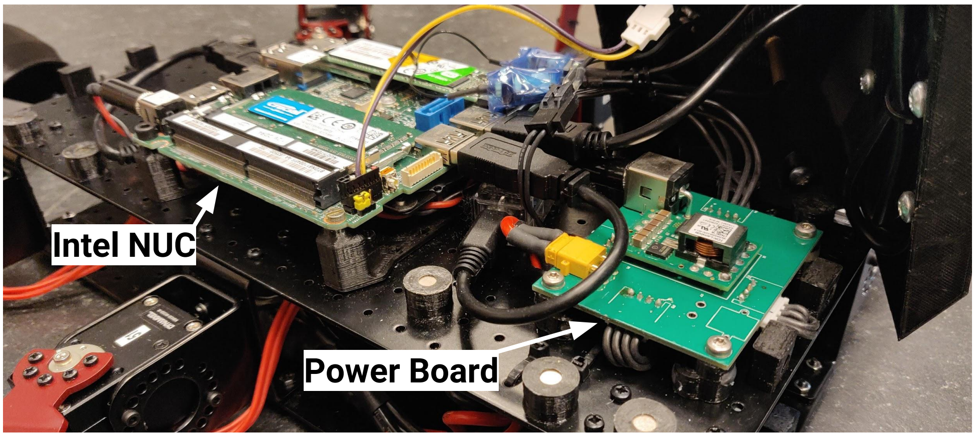


Figure 6.3 – The Intel NUC and power board when attached to the top plate of MORF using custom 3D printed black stands.

In addition to the parts in table 4.5 an Intel Dual Band Wireless-AC 8265 board for enabling WiFi on the Intel NUC (**R02**), a U2D2 for interfacing the PC with the Dynamixel servos, an BNO055 Absolute Orientation Sensor from Adafruit (9-DOF IMU) [83] for additional sensory information, and a LED array with 8 RGB LED's from BlinkStick [84] for debugging was ordered. The IMU is useful in many controllers and for quantifying the stability of the generated locomotion. It is connected to the NUC using a USB to UART board (Breakout board with an FT232H chip), as shown in Fig. 6.4. Here, it can also be seen that the IMU is physically attached using a custom 3D printed stand in the center line of the top body plate.

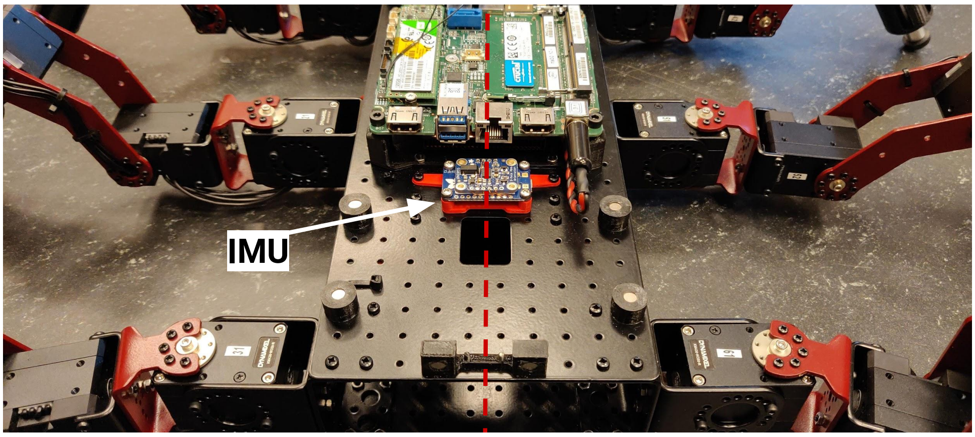


Figure 6.4 – The IMU when attached to the top plate of MORF using custom 3D printed red stand. The dashed red line indicated the center line of MORF.

The LED array is useful for visuals and debugging. It contains an integrated micro USB socket and microcontroller with a simple interface. The LED array is attached to the head of the body shell together with two omnidirectional antennas connected to the WiFi board on the Intel NUC and two buttons for turning on/off the NUC and battery, see Fig. 6.5. The reason for not attaching any electronics to the back of the body shell is that it should be easy to remove.

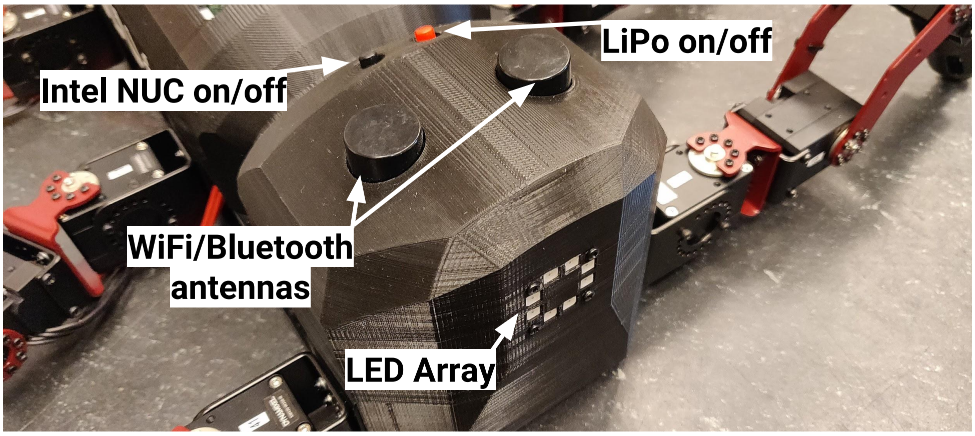


Figure 6.5 – The electronic components attached to the head shell.

6.1.4 Wiring

The wiring of the electronics on MORF is illustrated in Fig. 6.6. For each leg, a single 4pin cable from the power boards runs through all three servos which are made possible by the daisy chain property of the servos. One power board is used to power two legs and is additionally connected to the other power boards, the LiPo battery, the U2D2, and by extension the Intel NUC which also receives power from one of the boards. The wires from the power boards to the battery are fitted with 20A fuses in case of short circuits.

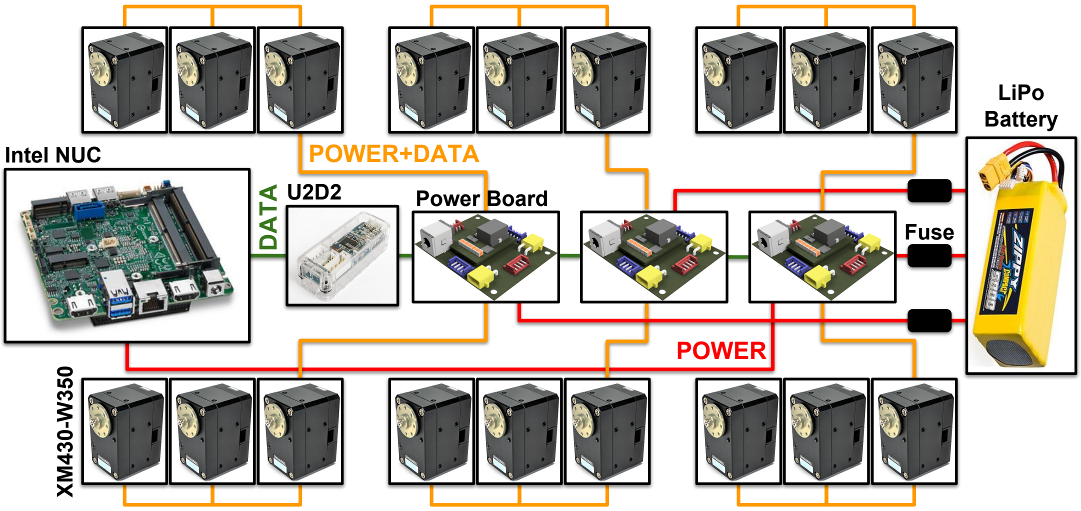


Figure 6.6 – The wiring of MORF. The orange wires transfer both power and data, the red wires transfer only power, and the green wires only data.

6.1.5 MORF assembled

Figure 6.7 shows MORF when fully assembled using the three different leg configurations (**R05**). The robot is fairly simple to reconfigure but does require many screws to be unscrewed and screwed back in. It takes around 1.5 hours to change all the legs from one configuration to another. Note that each wire on MORF is zip-tied to the body plates such that no wire hangs loose (**R03**).

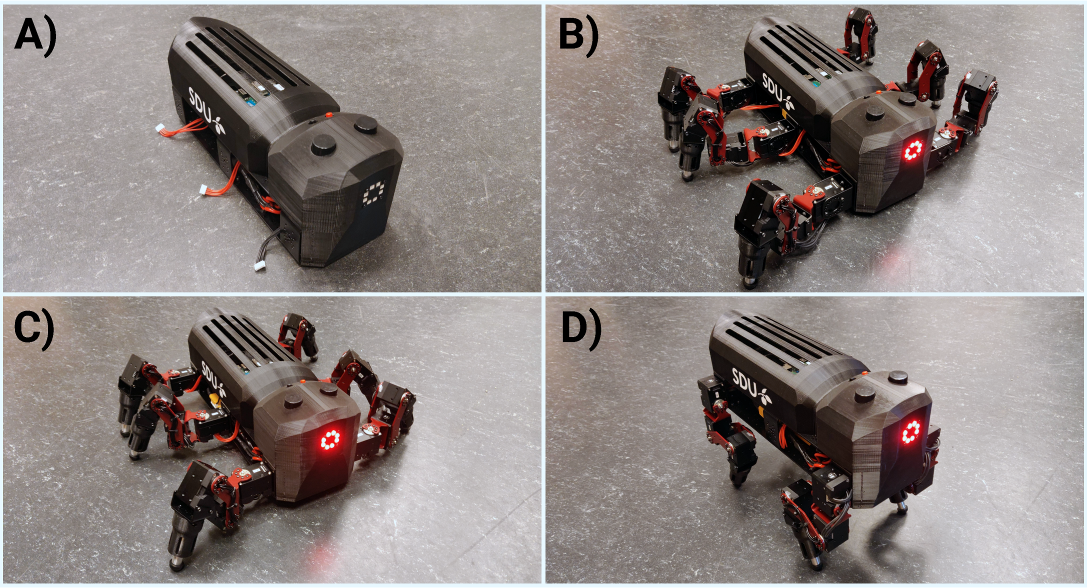


Figure 6.7 – Real versions of MORF using: **A)** no legs, **B)** the long insect legs, **C)** the short insect legs, and **D)** the mammal legs.

6.2 Software

6.2.1 Onboard computer setup

To keep the interface of the NUC as simple and familiar to researchers as possible a full version of Ubuntu 18.04 LTS is installed on it. This enables the user to test large parts of their code on a personal computer, and it furthermore makes it easy to set up a WiFi hotspot that starts at system boot.

When connecting to the WiFi hotspot using an external computer, it should be possible to gain full access to the NUC using ssh. This is achieved by installing OpenSSH-server and OpenSSH-client on both machines while making sure that port 22 on the NUC is open (**R11**).

It may, however, be tedious and time consuming to establish an ssh connection every time the user wants to start, stop, or update MORF. Thus, methods based on Ansible for starting and stopping services on MORF are developed. Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, synchronization, application deployment, intra-service orchestration, and many other IT needs [85]. Since Ansible is working with services, the hardware interfaces and locomotion controller, explained in the following, are placed in services using daemontools [86]. For MORF a so-called Ansible-Playbook with six different methods has been designed and implemented (**R11**). The methods provide the following control of MORF:

- Start hardware interface
- Stop hardware interface
- Update hardware interface
- Start locomotion controller
- Stop locomotion controller
- Update locomotion controller

An elaborated explanation of the six methods can be found in the MORF software manual at `https://github.com/MathiasThor/MORF/wiki/MORF-Software-Manual`.

6.2.2 Hardware interfaces

Hardware Interface to Dynamixel (R12)

The interface to the Dynamixel servos is based on the Dynamixel Workbench [87]. The workbench is written in C++ and contains various packages for interfacing with the Dynamixel series. One of these packages is the Controller package which defines methods for controlling the servos as well as reading their sensor values. The package uses the Dynamixel Workbench library called toolbox which is based on the Dynamixel SDK; a software development kit that provides Dynamixel control functions using packet communication. The Controller package consists of three ROS nodes; a position controller, a velocity controller, and a torque controller. In the following, we will be working with the position controller node. However, the modifications made to this node can easily be copied to the other nodes. When connected to the servos the position controller node creates the following publisher and subscriber:

- `joint_command` - Subscribes to position commands for a single servo
- `Dynamixel_state` - Publishes the states/sensor values of all servos
  - States: `model_name`, `id`, `torque_enable`, `goal_current`, `goal_position`, `present_current`, `present_velocity`, `present_position`, `moving`.

When used without any modifications the controller node is able to publish 9 states/sensor readings from the 18 servos on MORF at a rate of 1.2Hz. This is well below what is required.

A thorough investigation of the setup and the position controllers source code was thus carried out from where the following four improvements were developed and implemented:

1) The Controller node reads one control table entity (i.e., sensor value) from each servo at the time. As shown in Fig. 6.8A, this means that a total of $9 \cdot 2$ packages (request and response) is sent sequentially back and forth every time the sensory information is read. This creates a lot of overhead and slows down the rate of communication. It also means that for every servo added to the system or every additional sensor that should be read the feedback rate will decrease even more. To address this, the source code has been modified to read the entire control table from a servo. Each control table is then subsequently split up locally from where the desired sensory information can be subtracted and published, see Fig. 6.8B.

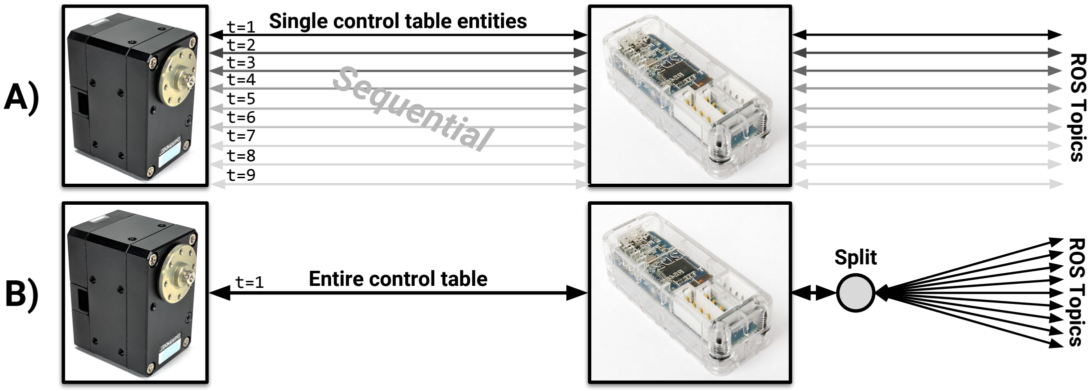


Figure 6.8 – Illustration of **A)** the old and **B)** the new way of reading the sensors values/control table of each Dynamixel servo.

2) The latency timer for the FTDI serial to USB converter chip used in the U2D2 was reduced from 16ms to 1ms.

3) The baud rate was increased from 56700 to 4.000.000 in both the source code and on the physical servos. According to the datasheet of the Dynamixel servos, this increase does not result in an increased communication error.

4) In order to synchronize the communication (i.e. read/write) to the servos the `GroupSyncRead` and `GroupSyncWrite` classes from the Dynamixel SDK are used. These classes make it possible to communicate with all connected servos simultaneously. This is an advantage for systems that contain many servos, like MORF, because a delay may be introduced between communicating with the first and last servo in the system. Such a delay is not desired as it may result in asynchronous sensor values and position commands. Furthermore, by synchronizing the communication the feedback rate becomes partly[3] independent of the number of servos in the system.

[3] The rate still reduce a little when adding additional servos due to the overhead of splitting the control tables.

The four modifications increased the feedback rate of the 18 servos used on MORF from 1.2Hz to 62.5Hz (**R14**).

Besides the source code modifications, the name and amount of ROS topics were also changed to make it more intuitive. The new publishers and subscribers are as follows:

- `joint_Positions` - Publishes the positions of all servos in [rad]
- `joint_Velocities` - Published the angular velocity of all servos in [rad/s]
- `joint_Torques` - Published the torque of all servos in [Nm]
- `joint_InputVoltage` - Published the input voltage to all servos in [V]
- `joint_ErrorStates` - Published the Error states of all servos
- `multi_joint_commands` - Subscribes to desired positions for all the servos in [rad]

ROS Interface to the IMU (R12)

The interface to the IMU is written in Python using the official Python module/API from Adafruit. From the IMU five sensory values are read and published using the following three ROS publishers:

- `imu` - Publishes the orientation [quaternion], angular velocity [rad/s], and linear acceleration [m/s^2] as an IMU ROS message
- `temperature` - Publishes the temperature [degree Celsius]
- `euler` - Publishes the Euler orientation [x,y,z]

The advantage of the BNO055 IMU from Adafruit is that it contains a sensor fusion algorithm that blends the onboard accelerometer, magnetometer, and gyroscope data into stable three-axis orientation output. However, in order to get good orientation data the sensors needs to be calibrated. The BNO055 includes an internal algorithm that constantly calibrates itself, but to get a good starting point at boot a small Python program that lets the user calibrate the sensor and save it to a `.json` file was made. The generated file will be loaded by the node every time it is started.

ROS Interface to LED array (R12)

The interface to the LED array is written in Python using the official Python module/API from BlinkStick. Two methods are made; one for setting the color of all LEDs simultaneously and another for setting the color for each LED individually. This resulted in the following two ROS subscribers:

- `set_all_led` - Subscribes to a color RGBA message where **R**,**G**, and **B** sets the color of all 8 LEDs (**A** is ignored).

- `set_single_led` - Subscribes to a color RGBA message where **R**,**G**, and **B** sets the color of LED number **A**.

Figure 6.9 shows examples of different settings for the LED array.

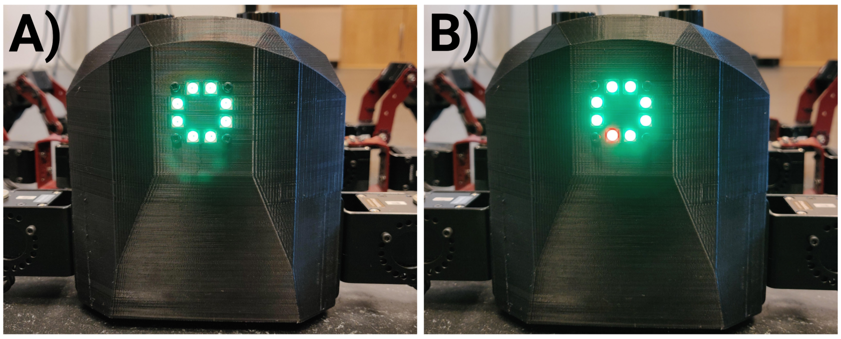


Figure 6.9 – LED array interface example. **A)** All of the LEDs are set to green (R=0, G=40, B=0) using `set_all_led` topic. **B)** LED 7 is set to red (R=40, G=0, B=0, A=7) using the `set_single_led` topic

ROS structure on MORF

The ROS structure of MORF is shown in Fig. 6.10. All three hardware interfacing ROS nodes (black ovals) are used by a locomotion controller (red oval). Subscriber topics are illustrated as green rectangles, and publisher topics are illustrated as blue rectangles. Note that all nodes related to hardware interfacing are placed under the namespace `/morf_hw`.

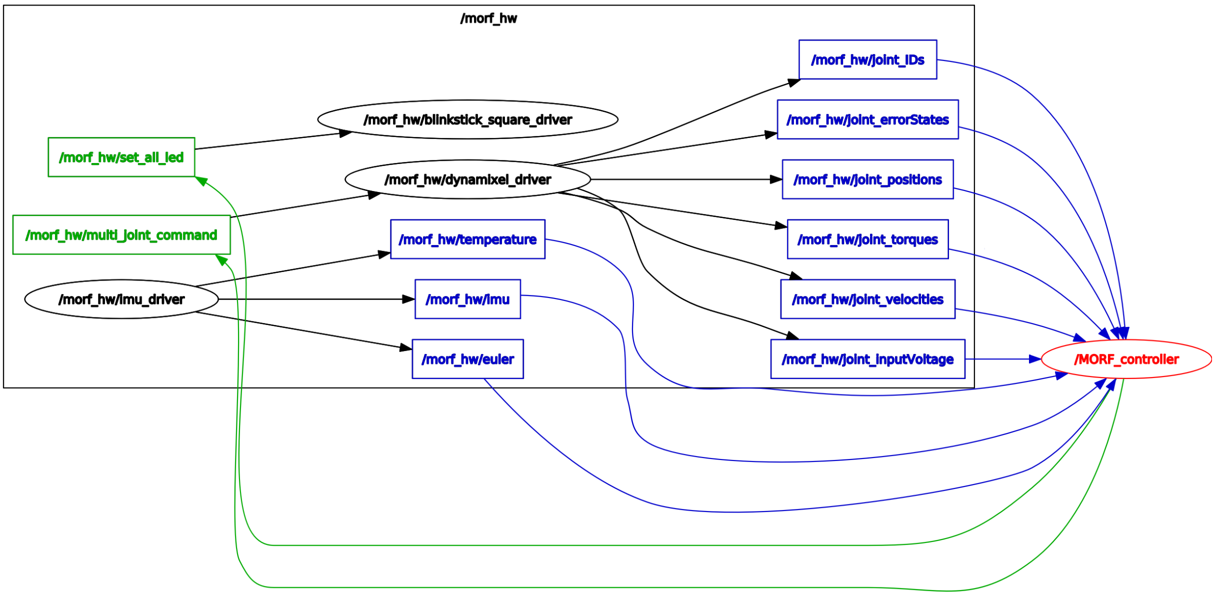


Figure 6.10 – The ROS structure of MORF when running a locomotion controller (called `/MORF_controller`). Rectangular blue entities are publisher topics and rectangular green entities are subscriber topics. Oval entities are ROS nodes.

6.2.3 Simulation

The simulated version of MORF is, as specified by the requirement specification, implemented in V-REP [76] (**R16**). To reduce the workload on the physics engine, MORF is separated into two models; A visual, and a dynamic model. The visual model is constructed from the CAD models. To also reduce the workload of the renderer the CAD models are simplified in Autodesk Inventor by removing small holes and flanges, see Fig. 6.11A. The dynamic model is an even more simplified version of the CAD models as it consists of only primitives, see Fig. 6.11B. The primitives are derived using a tool in V-REP that lets one select triangles in the visual model from which cuboids, cylinders, spheres, and planes can be automatically generated. Note that the body shape is a bit smaller in height than the real model to account for the weight distribution (i.e., the mostly empty shell). For each part of the dynamic model, the inertia's are calculated based on the primitives shape and the masses accordingly to the real-world parts.

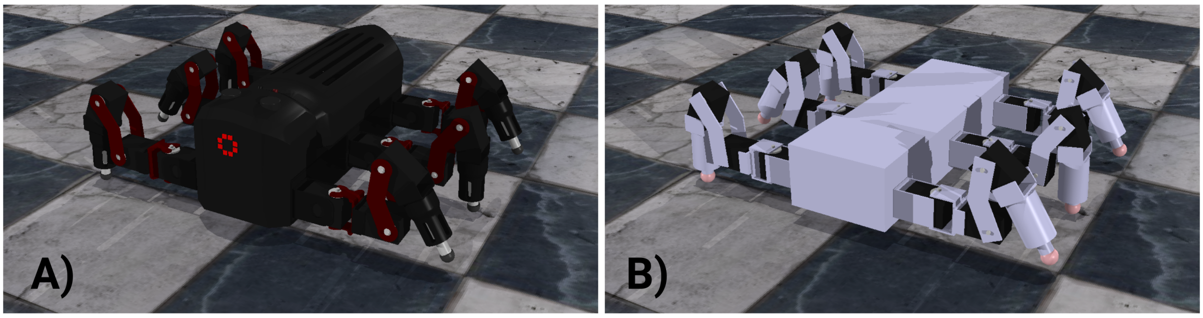


Figure 6.11 – Illustrations of the simulated version of MORF in V-REP. **A)** shows the visual model and **B)** shows the dynamic model.

During simulation, the CAD models will be visible and will at all time follow the position of their invisible dynamic counterparts. The physics engine will base all calculations (i.e., collision detection, etc.) on the dynamic model, resulting in a faster and smoother simulation. As physics engine, the Vortex engine is used for the reasons discussed in Chap. 4.

The primitives are either fixed or linked to other primitives using joints that are placed accordingly to the servos on the real-world version of MORF. MORF consists of 18 active joints (the servos) and 12 passive joints/DOFs, as shown in Fig. 6.12. In Vortex, the joint parameters for the active joints (BC joint is shown in red, CF joint shown in green, and FT joint shown in blue) are set accordingly to those of the Dynamixel XM430-W350. Six of the passive joints are the rotation pipes in the feet (shown in yellow). These are modeled as passive revolute joints. The remaining six passive joints are the springs (shown in purple). The normal way of modeling these would be to use a prismatic joint set to a spring-damper mode in V-REP. However, this method turned out to be unstable and inaccurate. The springs are instead modeled as a revolute joint with the z-axis perpendicular to the spring axis and the x-axis in the same direction. In the Vortex engine, joint relaxation along the x-axis is then enabled, and the appropriate parameter for stiffness is set. By directly using Vortex's implementation of a spring a much more realistic, reliable, and stable model of the spring was achieved.

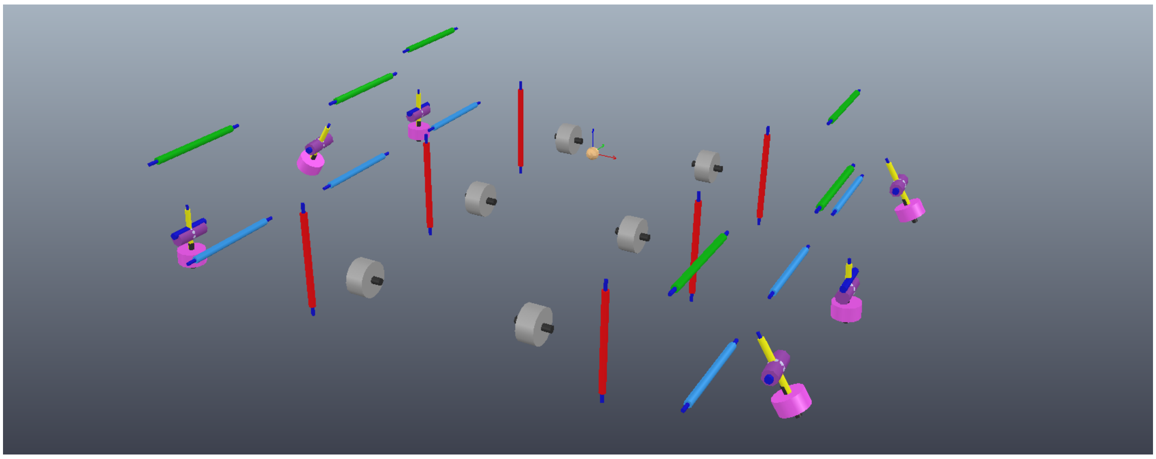


Figure 6.12 – The joint configuration of the simulated version of MORF. The grey cylinders are rigid connections that connect a leg to the body. The red, green, and blue joints are revolute joints that model the TC, CF, and FT joints respectively. The yellow joints are passive revolute joints that model the rotation of the pipes in the feet. The purple joints are passive revolute joints that model the springs placed inside the pipes in the feet. Finally, the pink cylinders are 3D force sensors that are to be used in the future version of the feet.

In addition to the servos and mechanical parts, most of the sensors of MORF are also replicated in the simulation. This includes sensors in the servos as well as the IMU. The interfaces to the sensors and the active joints is placed in a so-called child script attached to the MORF model in V-REP. The script is written in Lua and consist of a collection of routines that are executed in every time step. The primary purpose of the child script is to give the simulation similar interfaces as close to the real-world version of MORF as possible. The child script is thus also implemented in a ROS node, as described in the following section (**R17**).

Note that besides the simulation of MORF with long insect legs, simulations of MORF using short insect legs and mammal legs have also been made, see Fig. 6.13.

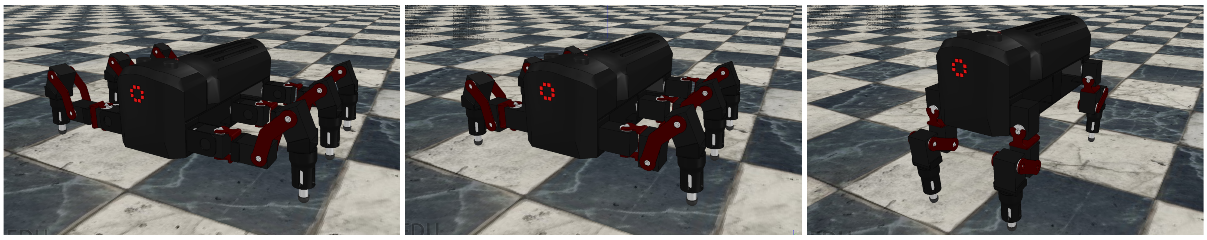


Figure 6.13 – The simulated versions of MORF with the legs in **A)** the long insect like configurations, **B)** the short insect like configurations, and **C)** the mammal configurations.

ROS structure in simulation

The ROS structure of the simulated version of MORF is shown in Fig. 6.14.

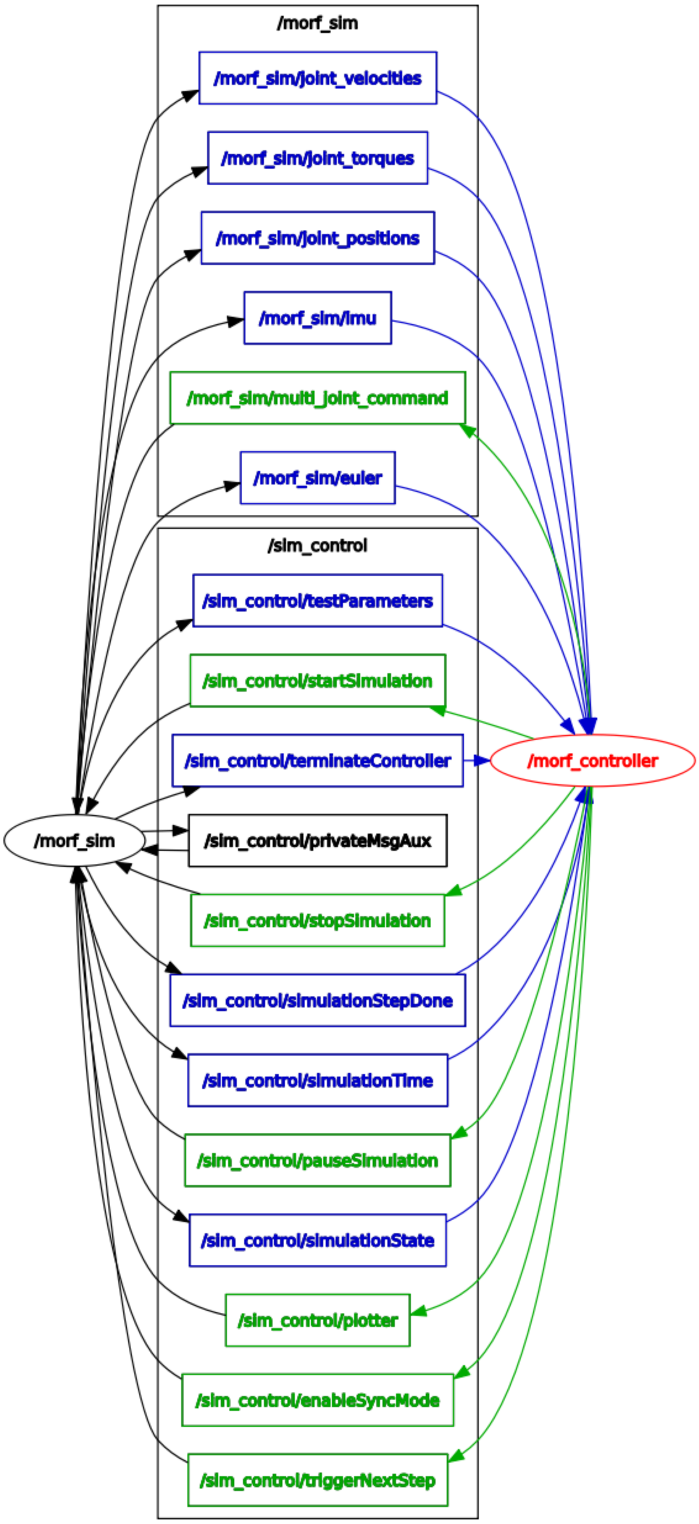

/morf_sim
/morf_sim/joint_velocities
/morf_sim/joint_torques
/morf_sim/joint_positions
/morf_sim/imu
/morf_sim/multi_joint_command
/morf_sim/euler
/sim_control
/sim_control/testParameters
/sim_control/startSimulation
/sim_control/terminateController
/morf_controller
/morf_sim
/sim_control/privateMsgAux
/sim_control/stopSimulation
/sim_control/simulationStepDone
/sim_control/simulationTime
/sim_control/pauseSimulation
/sim_control/simulationState
/sim_control/plotter
/sim_control/enableSyncMode
/sim_control/triggerNextStep

Figure 6.14 – The ROS structure of the simulated MORF when running a locomotion controller. Rectangular blue entities are publisher topics and rectangular green entities are subscriber topics. Oval entities are ROS nodes with the red being the locomotion controller (called `/MORF_controller`).

The simulated MORF (`/morf_sim` node) uses many of the same ROS topics as the real-world MORF, but now under the `/morf_sim` namespace. However, for obvious reasons, the topics for LED control, temperatures, input voltage, IDs, and error states are not included. To control the simulation and synchronize it with the locomotion controller various subscribers and publishers, under the `/sim_control` namespace, are created. The topics are defined as follows

- `testParameters` - Publishes test parameters [float array]
- `startSimulation` - Subscribes to a topic over which the simulation can be started [bool]
- `terminateController` - Publishes to a termination signal to the controller [bool]
- `stopSimulation` - Subscribes to topic over which the simulation can be stopped [bool]
- `simulationsStepDone` - Publishes when the simulation has finished one time step [bool]
- `simulationTime` - Publishes the simulation time [float]
- `pauseSimulation` - Subscribes to topic over which the simulation can be paused [bool]
- `simulationState` - Publishes the simulation state (paused, stopped, running) [int]
- `plotter` - Subscribes to topic over which it is possible to plot parameters in a graph window in V-REP [float array]
- `enableSyncMode` - Subscribes to topic over which it is possible to set the simulation in synchronization mode [bool]
- `triggerNextStep` - Subscribes to topic over which it is possible to trigger a new simulation step (requires synchronization mode to be activated) [bool]

6.2.4 Locomotion controller

The locomotion controller explained in Chap. 4, Sect. 4.4.2 is also implemented in C++ as a ROS node (**R13**) and consists of two main classes; a ROS handler class, and a controller class. The ROS handler class is responsible for all ROS related activities and serves the controller class by providing an interface to the sensory values and servo commands. The ROS handler class used for the simulated and real-world version of MORF are not identical but very similar. They are implemented in a way such that they provide similar interfaces for the controller class. It is as a result of this possible to have a single controller class as it does not *know* if it is subscribing and publishing to the simulated and real-world version of MORF (**R17**). A class diagram of the locomotion controller can be found in appendix C, which may be used, together with the source code, as a template for future controllers.

6.2.5 Source code

The source code for the improved dynamixel interface, the ROS interface to the IMU, and the ROS interface to the LED array can be found at `https://github.com/MathiasThor/dynamixel-`

workbench, https://github.com/MathiasThor/ros_bno055_driver, and https://github.com/MathiasThor/ros_blinkstick_square_driver respectively.

The source code for the locomotion controller, the simulations, and the ansible playbooks can all be found at https://gitlab.com/ens_sdu/gorobots under the project name morf. The repository is private as it contains work still to be published. However, access may be granted upon request.

Chapter 7

Validation of MORF

In this section, the implemented framework is validated to see if it complies with the requirement specifications from Chap. 5. Only the requirements with *test* as an inspection method will be addressed, as the once with *design* was addressed in the previous chapter. Note that only the MORF configuration with long insect legs is validated, as also mentioned in Chap. 4.

Every time a requirement is validated it will be indicated in the following way: (**R##**), where ## is the number of the requirement.

7.1 Hardware

7.1.1 Step height

To see if MORF has a stepping height of 174mm a small experiment where MORF has to put its foot tip on a 174mm tall wooden block is made.

Results and discussion

The results from the test are shown in Fig. 7.1, where it can be seen that MORF can lift the leg as required (**R07**). The distance from the floor to the bottom of MORF, when placed in the configuration seen in Fig. 7.1A, is 170mm. This means that MORF, in theory, is able to move to a platform of maximum 170mm in height if the body should be kept parallel to the ground at all times.

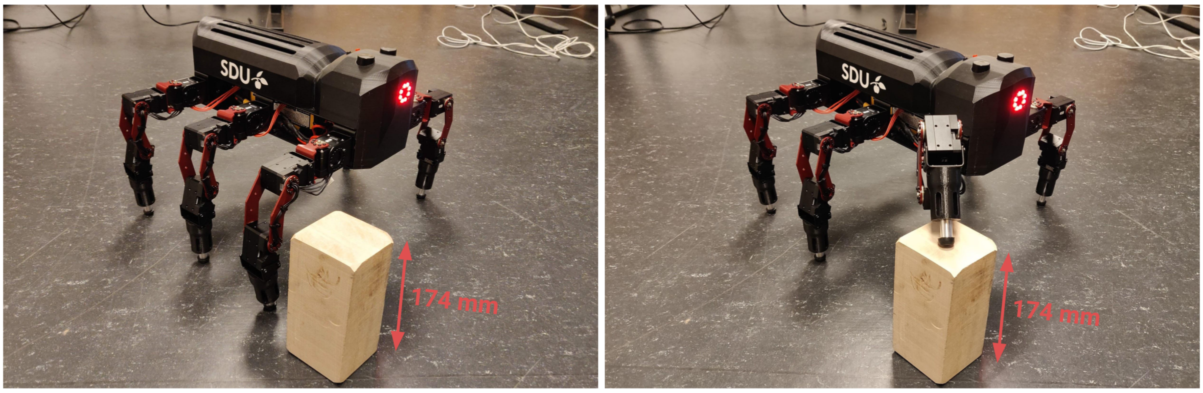


Figure 7.1 – **A)** MORF before stepping. **B)** MORF after stepping on the wooden block.

7.1.2 Payload

According to **R08**, MORF needs to be able to carry the weight of additional sensors and mechanics. To validate this the joint forces of MORF are measured during regular walking using the locomotion controller presented in the previous chapter.

Results and discussion

Figure 7.2 and 7.3 shows the torque feedback from the actuators in a front and middle leg when MORF is walking straight on flat ground. The actuator with the highest torque is the CF joint in the middle leg which experiences a max torque of 1.44Nm. This means that there is more than enough room for additional payload, as the actuator has a stall torque of 4.8Nm (**R08**). Using the simulated version of MORF is was furthermore found that MORF is able to carry approximately 5kg or 120% of additional payload on the body before the actuators, in this case the CF joint, would reach its stall torque[1].

The results in Fig. 7.2 and 7.3 also show that the BC/TC joint torque is low. It is thus expected that the BC joints will be able to move with a velocity close to 57 RPM resulting in a theoretical max walking velocity of: $57RPM \cdot 0.10472 \cdot r = 0.703m/s$, where $r = 117.75mm$ is the horizontal leg length[2].

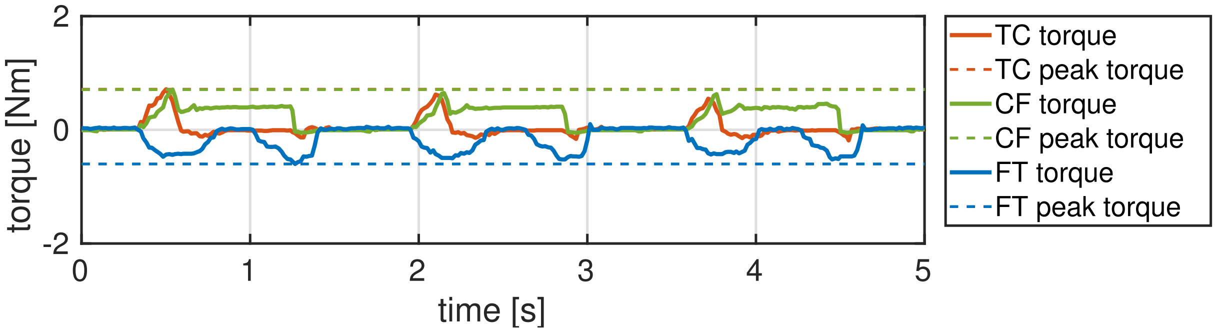


Figure 7.2 – The torques in the three actuators of the right front leg during a tripod gait. There will always be a front and hind leg in ground contact on either the right or left side of the robot. Note that TC (thorax-coxa) is the same as BC.

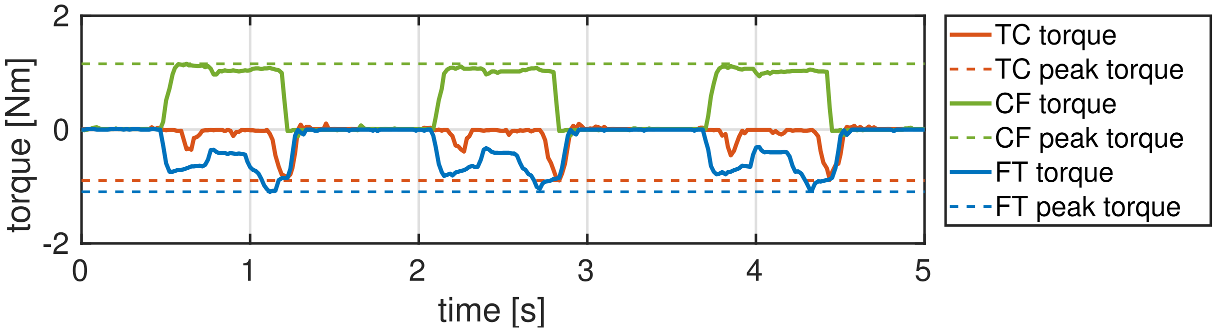


Figure 7.3 – The torques in the three actuators of the left middle leg during a tripod gait. The middle leg always will be the only leg in ground contact on either the right or left side of the robot. Note that TC (thorax-coxa) is the same as BC.

7.1.3 Run-time and energy source

To validate the run-time of MORF and the stability of the energy source an experiment is made. In this experiment, MORF will walk back and forth in the main corridor of the Maersk Mc-Kinney Moller Institute at SDU as shown in Fig. 7.4, using the locomotion controller introduced in the previous chapter. The experiment will start with MORF having a fully charged Zippy 5800 mAh LiPo battery and end when one of the LiPo cells drops below 3.4V.

[1] Note that this experiment is still to be replicated on the real world version.

[2] The length from the BC joint axis to the end of the leg when configured like in Fig. 4.11A.

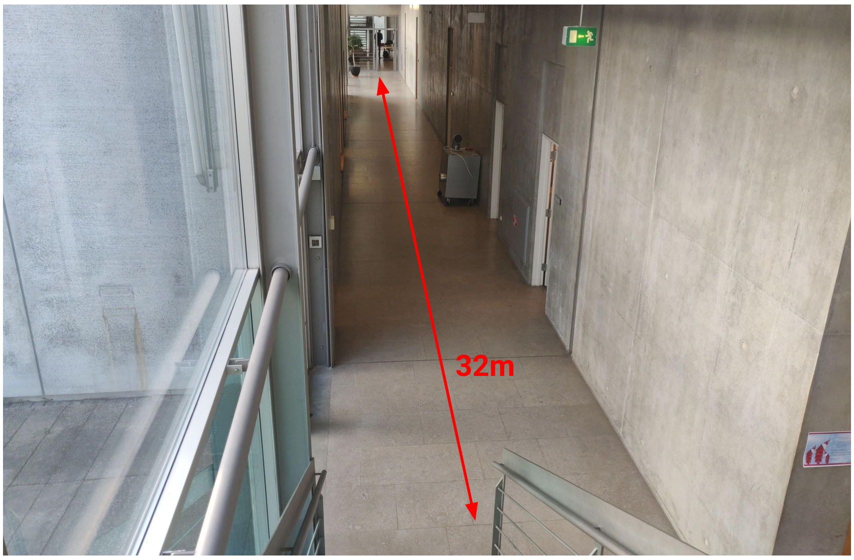


Figure 7.4 – The main corridor of the Maersk Mc-Kinney Moller Institute. The corridor is 32m long and the ground is made of flat concrete.

Results and discussion

Results from the experiment show that MORF can walk for 1 hour and 38 minutes in one battery discharge cycle (**R08**). During this period, MORF covered a distance of 1.26km with an average velocity of 0.213 m/s. The results also show that MORF can operate for a large period without any hardware or software related errors. In other words, the framework may be considered robust and reliable. A video from the experiment can be seen using the following link: `https://youtu.be/Q0SqIUr74o0`.

Figure 7.5 shows the input voltage to all the servos during the experiment. The pooled standard deviation of the 18 input voltages (one for each servo) is found to be 0.0835. The pooled standard deviation is used under the assumption that the variance of the input voltage to each servo is the same. Since the standard deviation is small the input voltage to the servos may be considered constant (**R09**).

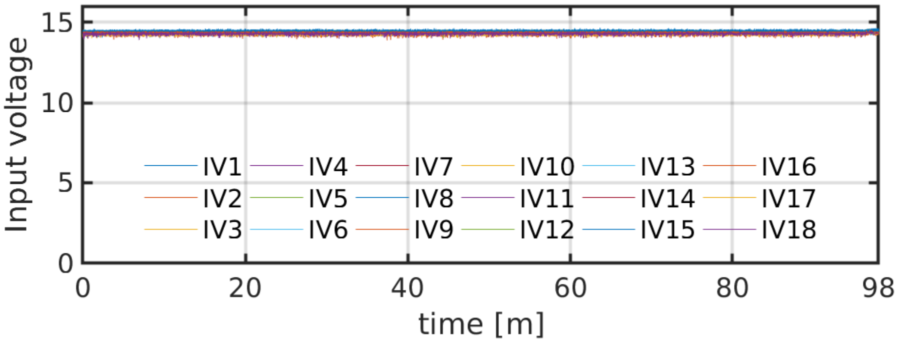


Figure 7.5 – Input voltage (IV) sensed by the 18 servos of MORF. All units in volts [V].

7.2 Software

7.2.1 Hardware interface

To test the feedback rates of the hardware interfaces a ROS graph with statistics information turned on is generated using the rosgraph command-line tool.

Results and discussion

Figure 7.6 shows the generated graph. Here it can be seen that the feedback rates from the sensors (blue rectangular entities) to the locomotion controller (red oval called `MORF_controller`) are all above 60Hz, as required (**R14**).

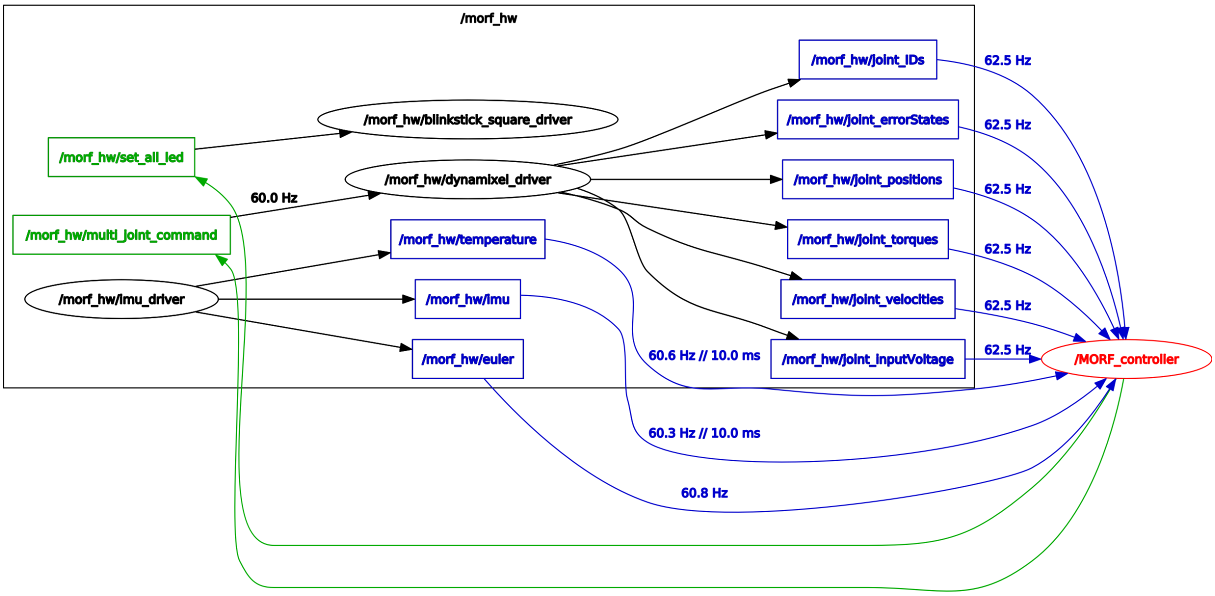


Figure 7.6 – ROS graph with statistics information. Rectangular blue entities are publisher topics and rectangular green entities are subscriber topics. Oval entities are ROS nodes.

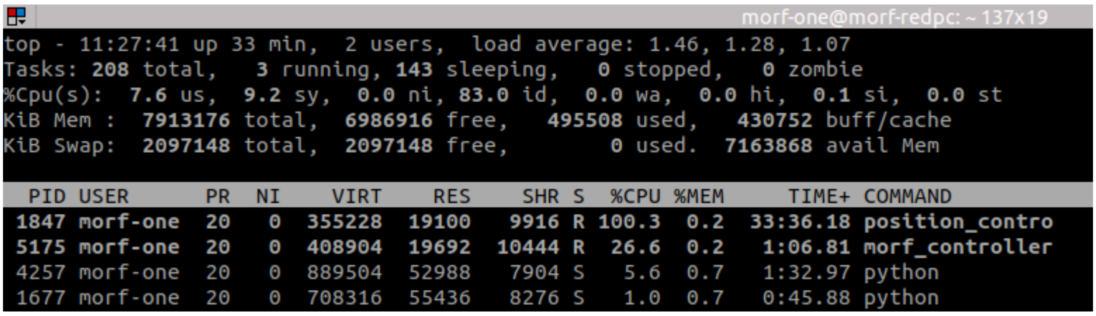
7.2.2 Onboard computer

To see if the Intel NUC can handle wireless communication, a locomotion controller, and the hardware interfaces at the same time, a small experiment is made. In the experiment the computer will be responsible for the following tasks over a time-span of 5 minutes:

- Running the locomotion controller.
- Facilitating a Bluetooth connection to an XBOX joystick.
- Facilitating a WiFi hotspot with an external PC connected using SSH.
- Running hardware interface and the respective ROS nodes.

Results and discussion

Figure 7.7 shows the load of the NUC using `top`, a command that allows users to monitor processes and system resource usage on Linux. Here it can be seen that the NUC has an average load of 1.28 over a time span of 5 minutes, which on a 4 core system is equal to 32% of the processors capacity [88]. Since this is relatively low with room for additional processes, it can be concluded that requirement **R15** is fulfilled.



```
top - 11:27:41 up 33 min,  2 users,  load average: 1.46, 1.28, 1.07
Tasks: 208 total,   3 running, 143 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.6 us,  9.2 sy,  0.0 ni, 83.0 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem :  7913176 total,  6986916 free,   495508 used,   430752 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used.  7163868 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1847 morf-one  20   0  355228  19100   9916 R 100.3  0.2  33:36.18 position_contro
 5175 morf-one  20   0  408904  19692  10444 R  26.6  0.2   1:06.81 morf_controller
 4257 morf-one  20   0  889504  52988   7904 S   5.6  0.7   1:32.97 python
 1677 morf-one  20   0  708316  55436   8276 S   1.0  0.7   0:45.88 python
```

Figure 7.7 – The output from the `top` command. Load average is the load (explained in [88]) for the last 1, 5, and 15 minute periods. Note that the `position_control` also known as the Dynamixel hardware interface node is running at 100% on its core, meaning that it is delivering feedback from the servos as fast as possible.

7.2.3 Simulation

To compare the difference between the simulated and real-world version of MORF (i.e., the reality gap) two experiments are created. Both versions will in both experiments use the locomotion controller presented in the previous chapter and save all sensory information in a `rosbag` [89]. In the first experiment, both versions will walk straight on flat ground for 5 seconds. In the second experiment, both versions will also turn 90 degrees left and right while walking by the use of an XBOX joystick that manipulates the amplitude of the CPG signals to the BC joints. First, the simulated version will be controlled and then by using the `rosbag` recorded from this experiment the topic for the joystick command will be replayed for the real-world version. It is hereby possible to ensure that both versions receive the same joystick input.

Note that the phase differences between the sensory signals from the simulation and the real-world are not investigated in this thesis. They will instead be manually adjusted such that the signals overlap so they can be compared. In the future, the phase shifts should be empirically determined and accounted for in the simulation.

Results

Figure 7.8 to 7.10 shows the sensory information from the three actuators in a front leg when MORF is walking straight on flat ground for 5 seconds. Figure 7.11 shows the Euler angles from the IMU where the z-axis of the IMU is pointing upwards and the x-axis is pointing in the direction of the head. The blue signals are from the sensors implemented in the simulation and the red signals are from the sensors on the real-world MORF.

All four figures include the absolute error between sensory signals from the simulated and real-world version. The absolute error is shown as a light red area starting on the x-axis of the plots. The mean absolute error (MAE) for all sensory comparisons is shown in table 7.1. The reason for using MAE and not root mean square error (RMSE) is that from an interpretation standpoint MAE is easier to understand as RMSE does not describe average error alone and has other implications that are more difficult to tease out. Sensory information from the actuators of a middle leg can be found in appendix D. Note that all the figures can also be found in the supplementary materials in the `Report_figures` directory for a better view.

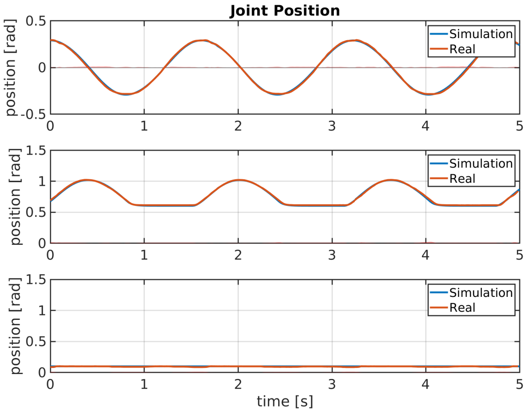


Figure 7.8 – Position feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

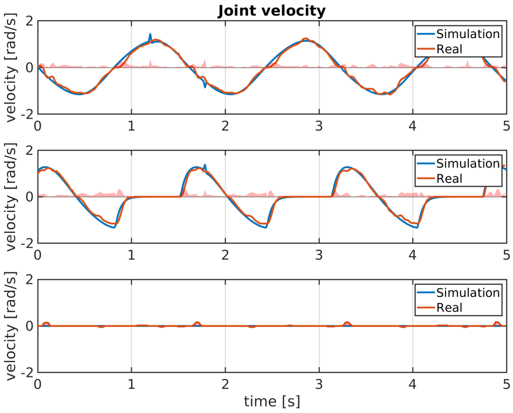


Figure 7.9 – Velocity feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

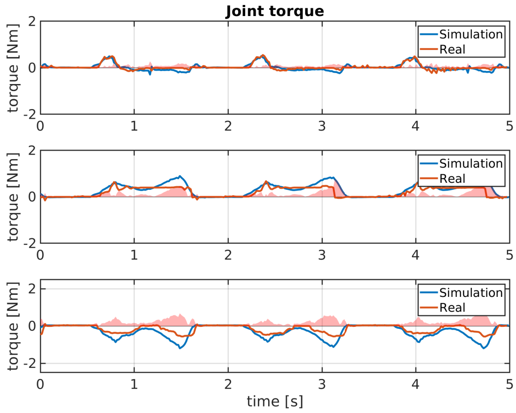


Figure 7.10 – Torque feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

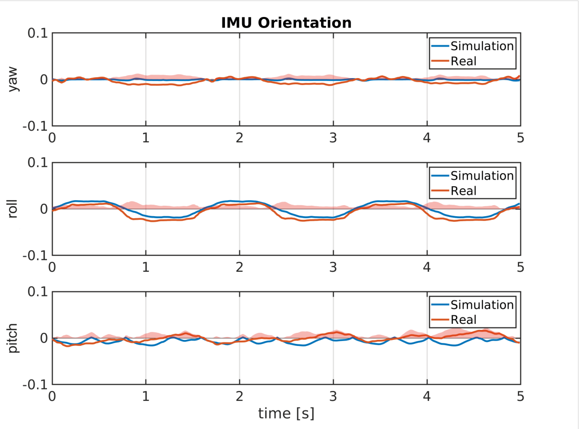


Figure 7.11 – Orientation feedback from IMU with Yaw/heading (top plot), roll (middle plot) and pitch (bottom plot). All units in [rad].

Table 7.1 – The mean absolute error (MAE) between the sensory information from simulation and the real-world MORF when MORF is walking straight.

Sensor Name	**MAE**
BC joint position	0.0057 rad
CF joint position	0.0072 rad
BC joint position	0.0042 rad
BC joint velocity	0.0717 rad/s
CF joint velocity	0.0710 rad/s
FT joint velocity	0.0127 rad/s
BC joint torque	0.0624 Nm
CF joint torque	0.1160 Nm
FT joint torque	0.1760 Nm
IMU x-axis orientation	0.0054 rad
IMU y-axis orientation	0.0070 rad
IMU z-axis orientation	0.0079 rad

Figure 7.12 to 7.14 shows the sensory information from the three actuators in a front leg when MORF is walking and turning to the left and right on flat ground. Figure 7.15 shows the Euler angles from the IMU where the z-axis of the IMU is pointing upwards and the x-axis is pointing in the direction of the head. Note that the rotation of the yaw/heading (rotation around the z-axis) can be used to see when MORF is turning. All four figures also include the absolute error between sensor values from the simulated and real-world. The mean absolute error (MAE) for all sensory comparisons is shown in table 7.2. Sensory information from the actuators of a middle leg can be found in appendix D. Note that all the figures can also be found in the supplementary materials in the `Report_figures` directory for a better view.

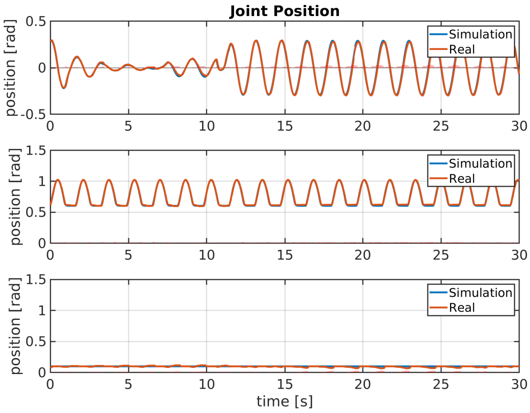


Figure 7.12 – Position feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

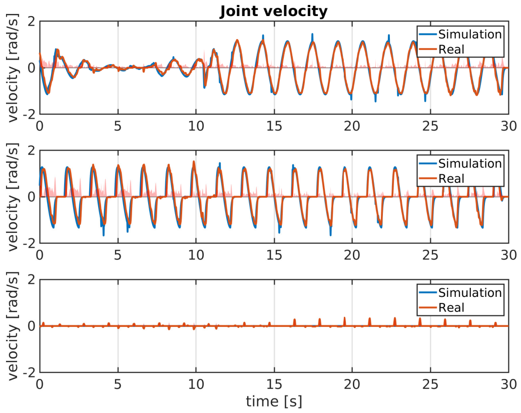


Figure 7.13 – Velocity feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

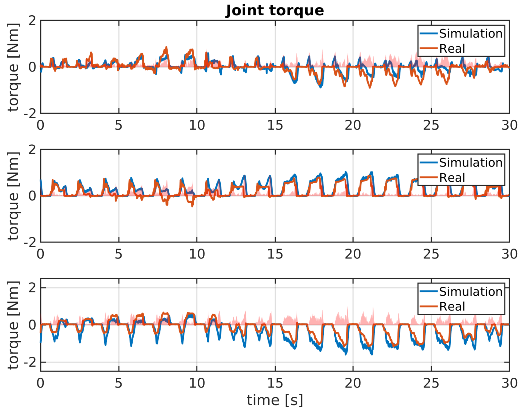


Figure 7.14 – Torque feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

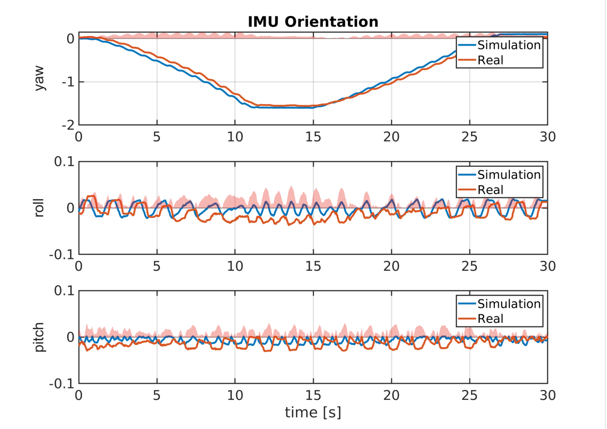


Figure 7.15 – Orientation feedback from IMU with Yaw/heading (top plot), roll (middle plot) and pitch (bottom plot). All units in [rad].

Table 7.2 – The mean absolute error (MAE) between the sensory information from simulation and the real-world MORF when MORF is turning left and right.

Sensor Name	MAE
BC joint position	0.0096 rad
CF joint position	0.0096 rad
BC joint position	0.0071 rad
BC joint velocity	0.0844 rad/s
CF joint velocity	0.1313 rad/s
FT joint velocity	0.0134 rad/s
BC joint torque	0.1397 Nm
CF joint torque	0.1310 Nm
FT joint torque	0.1867 Nm
IMU x-axis orientation	0.0786 rad
IMU y-axis orientation	0.0152 rad
IMU z-axis orientation	0.0106 rad

Discussion

From the above results, it can be concluded that the sensory feedback from the simulation is similar in shape and amplitude to that of the real-world (**R18**). The results also show that the feedback quality, from both versions, is nice and clean (i.e., low noise). The largest MAE is found in the sensory feedback from the torque sensors. This was expected as the torque depends on many parameters such as friction coefficients, mass distribution, etc.

It is, however, clear that the simulation is not "a perfect copy of the real world" [48]. This was especially apparent during the process of calibrating the simulation. Here it was found that the simulation is very sensitive to both friction and non-rigid parts like the rubber foot tips and springs. These can be removed/replaced by the user if a more accurate simulation is needed. The reason for not removing the non-rigid parts by default is that like nature the robot should remain dynamic. One could also argue that a proper controller would be able to cope with the dynamics of the system and exploit them (embodied robotics).

An interesting project for reducing the reality gap even further is to use reinforcement learning for tuning the parameters of the simulation (e.g., friction coefficient, spring constants, etc.). The feedback/fitness would, in this case, be the similarity between the sensory values of the simulated and real-world version of MORF. It would in this way be possible to evolve the simulation to mimic the real-world as accurate as possible.

7.3 Price

Table 7.3 lists the parts used to build MORF and their price, where new parts not included in the estimate are shown with *italic* fonts. As can be seen, the estimated price was 5322.13DKK higher than the actual price (**R19**). The main reason for this is that Dynamixel was so kind to provide a discount on their servos. It is important to note that this cost does not include salary for a technician to assemble the parts.

Table 7.3 – MORF parts list and prices. New parts not included in the estimate are shown with *italic* fonts.

Part name	**Price**
Aluminum parts - leg connectors and body pates	1738.6 DKK
Dynamixel XM-430-350 (18 pcs.)	19616.4 DKK
Dynamixel U2D2	479.46 DKK
Dynamixel Horn set (18 pcs.)	1288.44 DKK
Intel NUC i7	3649 DKK
SSD 120GB	241 DKK
RAM 8GB	647 DKK
WiFi module	140 DKK
3D Printed Shell	0 DKK
Blinkstick Square	151.6 DKK
Antennas (2 pcs.)	256.4 DKK
SMA to MHF4 (2 pcs.)	75.9 DKK
Foot shell (6 pcs.)	0 DKK
Aluminum pipe (6 pcs.)	3487.26 DKK

Table 7.3 continued on next page

Table 7.3 continued from previous page

Rubber tip (6 pcs.)	51.84 DKK
3D Force sensor (6 pcs.)	9000 DKK
Adafruit 9-DOF Absolute Orientation IMU	250 DKK
General Purpose USB to GPIO+SPI+I2C	97.02 DKK
ZIPPY Battery (2 pcs.)	960.74 DKK
Power distribution board (3 pcs.)	900 DKK
Powder coating	1000 DKK
Other - magnets, wires, 3D prints, screws, etc.	1211 DKK
XBOX controller	338.4 DKK
Total:	45556.87 DKK
Difference from estimated price	-5322.13 DKK

7.4 Scalability and usability in research

ROS makes it extremely easy to scale the system and to add new component/modules. Such a module could be an XBOX joystick for controlling the direction and speed of MORF (like in the previous experiment). Normally, one would have to write a driver for the joystick as it is not intended for a PC. This is not the case for MORF, where one can download the `joy_node` that is developed by the ROS community thanks to its open-source nature (**R20**). Figure 7.16 shows a ROS graph for the platform with the ROS `/joy_node` in use.

A second example of the scalability of MORF is MORF blue. MORF blue is a copy of the robot presented in this thesis and will be used in a Ph.D. project at SDU. MORF blue has the same mechanisms as presented in this thesis, but instead of the Intel NUC it is equipped with a custom-made controller with a Zynq processor for asynchronous processing. In the near future, MORF blue will additionally be equipped with a SpiNNaker board for research on spiking neural networks for locomotion control. This project really underlines the scalability and modularity of MORF (**R20** and **R21**).

Finally, MORF is also being used in a project by two master students at SDU. The students are developing a pipe clamping controller. This will enable MORF to move on pipes and inspect them using sensors such as thermal cameras (**R21**). Figure 7.17 shows MORF when walking on a pipe in simulation. The students are planning to test their control mechanism on the real version of MORF in the near future.

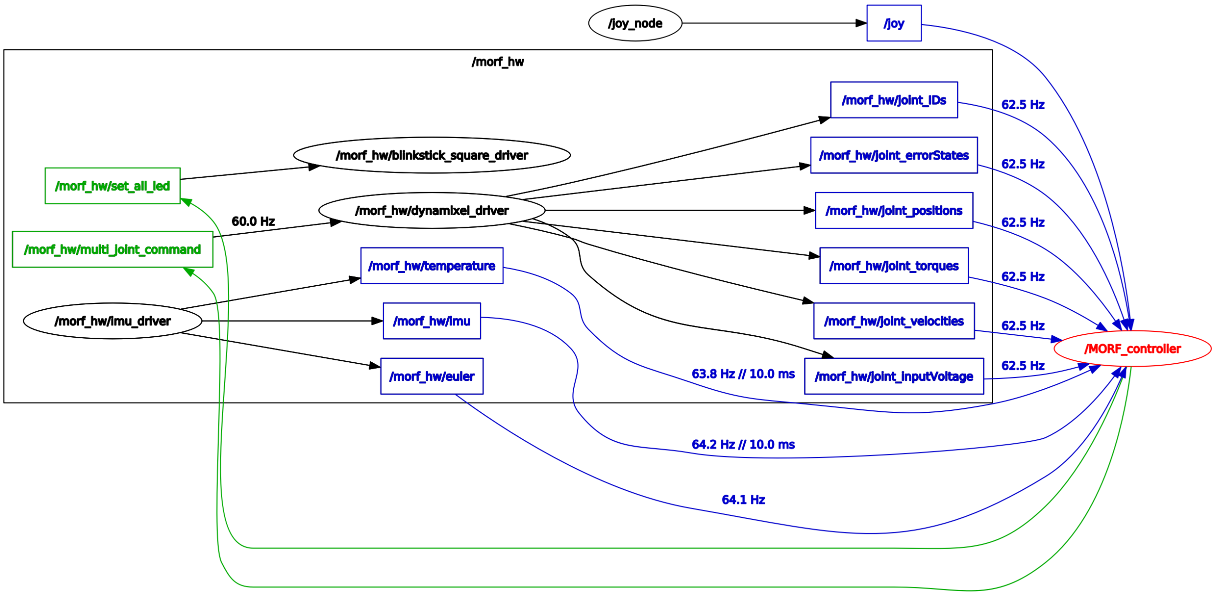


Figure 7.16 – ROS graph with the ROS `/joy_node`. Rectangular blue entities are publisher topics and rectangular green entities are subscriber topics. Oval entities are ROS nodes.

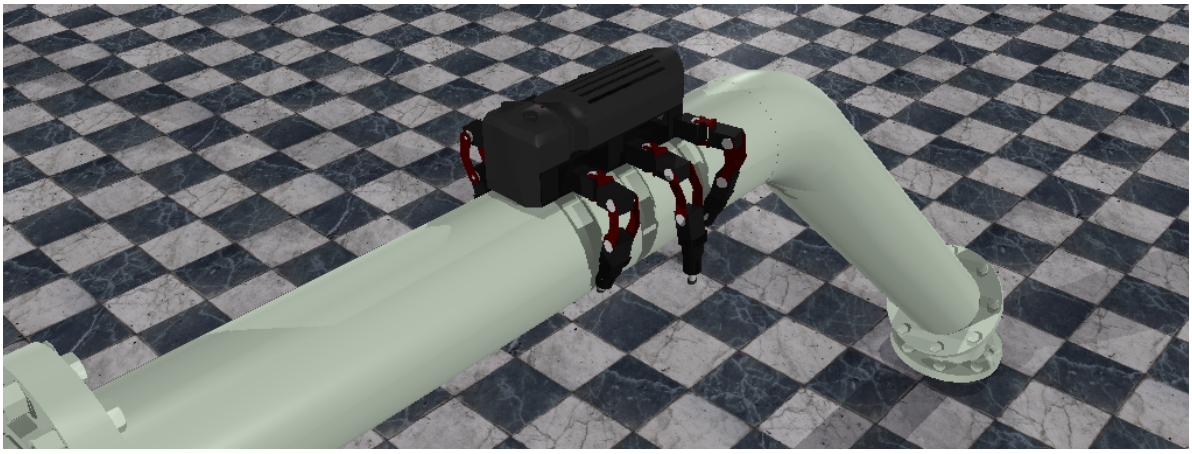
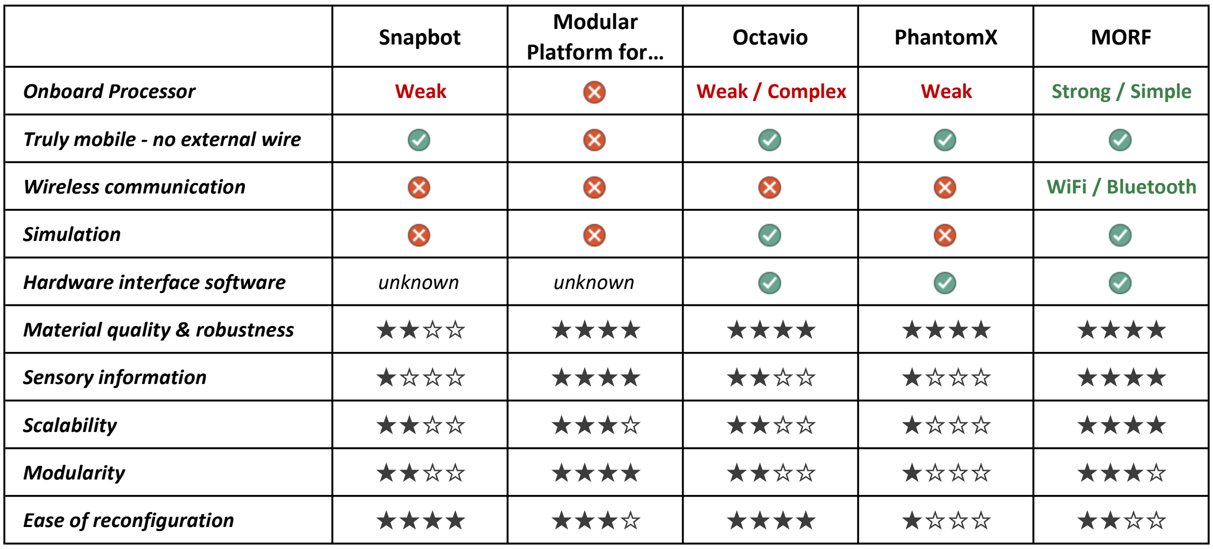


Figure 7.17 – MORF walking on pipes in simulation.

7.5 Comparison to other multi-legged platforms

In Chap. 2 several robots were reviewed, and their shortcomings were discussed. With MORF we have tried to address these shortcomings and to improve the current state of modular legged robots, see Fig. 7.18.



	Snapbot	Modular Platform for…	Octavio	PhantomX	MORF
Onboard Processor	Weak	✗	Weak / Complex	Weak	Strong / Simple
Truly mobile - no external wire	✓	✗	✓	✓	✓
Wireless communication	✗	✗	✗	✗	WiFi / Bluetooth
Simulation	✗	✗	✓	✗	✓
Hardware interface software	*unknown*	*unknown*	✓	✓	✓
Material quality & robustness	★★☆☆	★★★★	★★★★	★★★★	★★★★
Sensory information	★☆☆☆	★★★★	★★☆☆	★☆☆☆	★★★★
Scalability	★★☆☆	★★★☆	★★☆☆	★☆☆☆	★★★★
Modularity	★★☆☆	★★★★	★★☆☆	★☆☆☆	★★★☆
Ease of reconfiguration	★★★★	★★★☆	★★★★	★☆☆☆	★★☆☆

Figure 7.18 – Overview of the advantages and disadvantages for the legged-robots discussed in this Chap. 2 and MORF. The platforms are rated based on a personal assessment of available information (e.g., scientific articles).

The first thing that MORF improves is the onboard processor as it uses the Intel NUC computer. An Intel NUC requires no specialized knowledge and code can be directly ported from the user's laptop to MORF. Furthermore, the fact that the NUC is equipped with both WiFi and Bluetooth makes it easy to program, control, and receive data without the use of a cable.

Another problem improved by MORF is the ease of use. This is mainly accomplished by the hardware interfacing methods and the use of ROS which lets the user program and control MORF without any knowledge about the onboard hardware, software, and setup.

What was lacking for all the reviewed robot platforms was a realistic simulation that lets the user develop and evolve controller without having to use the physical robot. MORF includes realistic V-REP simulations with the Vortex physics engine of the three main configurations. The simulations are furthermore calibrated to reduce the reality gap such that there is a higher chance that a controller developed in simulation will also transfer completely to the real world.

Finally, MORF is designed to be very general such that it fits many types of research projects. From a software point of view, this is achieved by the Intel NUC and use of ROS as explained earlier. From a hardware point of view, this is achieved by the highly reconfigurable mechanics, the many sensors (160 in total when the 3D force sensors have been implemented), and the use of standards that makes it easy for the user to add additional sensors or mechanics. It is the goal that MORF in this way fits the research project in most cases and not the other way around.

Note that while MORF is advantageous in the above describes areas it still lacks some mechanism for attaching and detaching modules in an easy way. All mechanical parts are currently attached to each other using screws and bolts, making it time-consuming to reconfigure MORF. One improvement could be to use the screwless flange adapter design presented in [5] together with the spring pin connector presented in [6]. A new and improved attachment mechanism will stress the modularity of MORF as one could argue that in its current state it is closer to a walking robot building kit.

Chapter 8

Conclusion

A MOdular Robot Framework called MORF for research on locomotion control was developed based on a thorough analysis of existing methods and technical issues. It was designed for a wide range of research studies using state-of-the-art components for high performance while still being easy and convenient to use. MORF is modular as it defines standards that can be used for reconfiguring, extending, and replacing parts of the robot. MORF also includes a software suite with hardware interfacing software based on the Robot Operating System (ROS) as well as realistic simulations of MORF. The hardware interfacing software makes it easy for the user to use MORF even when their knowledge about hardware is limited. It also enables them to use any programming language compatible with ROS. The simulations of MORF are calibrated so that the reality gap is small, meaning that a controller developed in simulation has a high chance of also working on the real world system. The simulations also enable the framework to be used for education as the students/users can work on a controller even when the physical robot is not present.

All the requirements for the framework were tested, and the implementation passed nearly all of them. A more interesting aspect of the framework is, however, its shortcomings and how to improve them. Although a design has already been proposed the first part of MORF that could be improved is the foot design. This is because the foot is still to include a foot force sensor which will be implemented as soon as the sensor has been developed and patented. Compared to other modular robots like [5, 7, 6] and [20], MORF is advantageous in areas like processing power, mobility (no external wires), controllability, completeness (includes a software suite), accessibility (wireless connection), sensory feedback, and expandability, but lacks their ease of attaching parts together (e.g., using magnets, threaded collars, or screwless flange adapters). This will also have to be improved upon in future revisions of MORF to stress and improve the modularity of MORF more.

The real test of MORF is however if researchers are willing to use the framework for scientific projects, which only time will tell. There are, however, clear indications that this is the case as by now two copies of MORF has already been made; one for a Ph.D. project at the University of Southern Denmark and another Vidyasirimedhi Institute of Science and Technology in Thailand.

A final concluding remark is that MORF is still under developments and always will be. This is due to its modularity and scalability that lets users design new extensions. For that reason, we plan on releasing the technical drawings for MORF, so that they can be used as a basis for new designs. We, furthermore, plan on developing a website where users of MORF can share their results, ideas, and experiences with other users. The result will hopefully be an ever growing and improving framework. In the near future, I will personally be using MORF as the primary tool in my Ph.D. thesis about neurorobitc technology for advanced robot motor control. I will furthermore work with and supervise students that want to do projects with MORF. It is in this way possible to further explore the use cases of MORF and legged robots in general.

Chapter 9

Bibliography

[1] M. F. Silva and J. A. MacHado, "A literature review on the optimization of legged robots," *JVC/Journal of Vibration and Control*, vol. 18, no. 12, pp. 1753–1767, 2012.

[2] S. Kajita and B. Espiau, "Legged robots," in *Springer Handbook of Robotics*, 2008, pp. 361–389.

[3] D. J. Todd, *Walking Machines.* Springer US, 1985.

[4] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 43–52, March 2007.

[5] A. Von Twickel, M. Hild, T. Siedel, V. Patel, and F. Pasemann, "Neural control of a modular multi-legged walking machine: Simulation and hardware," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 227–241, 2012.

[6] J. Kim, A. Alspach, and K. Yamane, "Snapbot: A reconfigurable legged robot," in *Proc. of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 5861–5867.

[7] A. R. Ansari, J. Whitman, B. Saund, and H. Choset, "Modular Platforms for Advanced Inspection, Locomotion, and Manipulation," in *Proc. of 43rd Annual Waste Management Conference (WM2017)*, vol. 1, 2017, pp. 1017–1027.

[8] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, "Evolution of repertoire-based control for robots with complex locomotor systems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 314–328, April 2018.

[9] B. Leing, M. Thor, and P. Manoonpong, “Modular Neural Control for Bio-Inspired Walking and Ball Rolling of a Dung Beetle-Like Robot,” *ALIFE*, pp. 2–5, 2017.

[10] M. Thor, T. Strøm-hansen, L. B. Larsen, A. Kovalev, and S. N. Gorb, “Advantages of using a biologically plausible embodied kinematic model for enhancement of speed and multifunctionality of a walking robot,” *SWARM conference in Kyoto, Japan*, 2017.

[11] T. Sun, D. Shao, Z. Dai, and P. Manoonpong, “Adaptive neural control for self-organized locomotion and obstacle negotiation of quadruped robots,” in *Proc. of 27th IEEE International Conference on Robot and Human Interactive Communication*, 2018, pp. 1081–1086.

[12] J. a. T. Machado and M. F. Silva, “An Overview of Legged Robots,” *in Proc. of the MME 2006 International Symposium on Mathematical Methods in Engineering*, no. December, 2006.

[13] F. Tedeschi and G. Carbone, “Design Issues for Hexapod Walking Robots,” *Robotics*, vol. 3, no. 2, pp. 181–206, 2014.

[14] X. Zhou and S. Bi, “A survey of bio-inspired compliant legged robot designs,” *Bioinspiration and Biomimetics*, vol. 7, no. 4, 2012.

[15] D. Rollinson, Y. Bilgen, B. Brown, F. Enner, S. Ford, C. Layton, J. Rembisz, M. Schwerin, A. Willig, P. Velagapudi, and H. Choset, “Design and architecture of a series elastic snake robot,” *in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014*, no. Iros, pp. 4630–4636, 2014.

[16] K. Zahedi, A. Von Twickel, and F. Pasemann, “YARS: A physical 3D simulator for evolving controllers for real robots,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5325 LNAI, pp. 75–86, 2008.

[17] R. Smith, “Open Dynamics Engine,” accessed: 13-03-2018. [Online]. Available: http://www.ode.org/

[18] YARS - yet another robot simulator, “YARS - Revision history of "Main Page",” 2008, accessed: 22-01-2019. [Online]. Available: http://yars.sourceforge.net/w/index.php?title=Main_Page&action=history

[19] A. Boeing and T. Bräunl, “Evaluation of real-time physics simulation systems,” *Ecological Entomology*, pp. 281–288, 2007.

[20] Interbotix, "Professional grade robotic platforms for research, education and pro hobbyists." 2018, accessed: 19-12-2018. [Online]. Available: https://www.interbotix.com/

[21] L. B. Larsen, "System for Artificial Tutoring of Songbirds," Master's thesis, University of Southern Denmark, Campusvej 55 5230 Odense M, 2016.

[22] T. Zielinska, "Autonomous walking machines - discussion of the prototyping problems," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 58, no. 3, pp. 443–451, 2010.

[23] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ -A hydraulically and electrically actuated quadruped robot," *in Proc. of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[24] K. Karakasiliotis, R. Thandiackal, K. Melo, T. Horvat, N. K. Mahabadi, S. Tsitkov, J. M. Cabelguen, and A. J. Ijspeert, "From cineradiography to biorobots: an approach for designing robots to emulate and study animal locomotion," *Journal of The Royal Society Interface*, vol. 13, no. 119, jun 2016.

[25] P. Manoonpong, *Neural Preprocessing and Control of Reactive Walking Machines.* Springer-Verlag Berlin Heidelberg, 2007.

[26] T. L. Brown and J. P. Schmiedeler, "Energetic effects of reaction wheel actuation on underactuated biped robot walking," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2576–2581, 2014.

[27] F. Asano, "High-speed biped gait generation based on asymmetrization of impact posture using telescopic legs," *in Proc. of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4477–4482, Oct 2010.

[28] G. A. Vargas, D. J. Gomez, O. Mur, and R. A. Castillo, "Simulation of a wheel-leg hybrid robot in Webots," *in Proc. of the 2016 IEEE Colombian Conference on Robotics and Automation, CCRA 2016 - Conference Proceedings*, pp. 5–9, 2017.

[29] P. Manoonpong, U. Parlitz, and F. Wörgötter, "Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines." *Frontiers in neural circuits*, vol. 7, no. February, p. 12, 2013.

[30] H. Cruse, T. Kindermann, M. Schumm, J. Dean, and J. Schmitz, "Walknet—a biologically inspired network to control six-legged walking," *Neural Networks*, vol. 11, no. 7-8, pp. 1435–1447, oct 1998.

[31] A. Schneider, J. Paskarbeit, M. Schilling, and J. Schmitz, "Hector, a bio-inspired and compliant hexapod robot," in *Living Machines*, 2014.

[32] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 38–44.

[33] T. Zielinska and J. Heng, "Mechanical design of multifunctional quadruped," *Mechanism and Machine Theory*, vol. 38, no. 5, pp. 463–478, 2003.

[34] H. Cruse, V. Durr, and J. Schmitz, "Insect walking is based on a decentralized architecture revealing a simple and robust controller," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 221–250, 2007.

[35] M. M. Ankarali, E. Sayginer, Y. Yazicioglu, A. Saranli, and U. Saranli, "A dynamic model of running with a half-circular compliant LEG," *in Proc. of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, CLAWAR 2012*, vol. 1, no. September, pp. 425–432, 2012.

[36] A. Roennau, G. Heppner, M. Nowicki, and R. Dillmann, "LAURON V : A Versatile Six - Legged Walking Robot with Advanced Maneuverability," *in Proc. of the 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2014.

[37] I. Poulakakis, J. A. Smith, and M. Buehler, *On the Dynamics of Bounding and Extensions: Towards the Half-Bound and Gallop Gaits.* Tokyo: Springer Tokyo, 2006, pp. 79–88.

[38] D. Spenneberg, A. Strack, J. Hilljegerdes, H. Zschenker, M. Albrecht, T. Backhaus, and F.Kirchner, "Aramies: A four-legged climbing and walking robot," in *Proc. of 8th International Symposium iSAIRAS*, 2005.

[39] K. Walas, "Foot design for a hexapod walking robot," *Pomiary, Automatyka, Robotyka*, vol. 17, no. 193, pp. 283–287, 2013.

[40] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro, "Simple robot suggests physical interlimb communication is essential for quadruped walking," *Journal of The Royal Society Interface*, vol. 10, no. 78, 2013.

[41] X. Xiong, F. Wörgötter, and P. Manoonpong, "Adaptive and energy efficient walking in a hexapod robot under neuromechanical control and sensorimotor learning," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 1–14, 2015.

[42] W. H. Chen, G. J. Ren, J. H. Wang, and D. Liu, “An adaptive locomotion controller for a hexapod robot: CPG, kinematics and force feedback,” *Science China Information Sciences*, vol. 57, no. 11, pp. 1–18, 2014.

[43] G. Kenneally, A. De, and D. E. Koditschek, “Design Principles for a Family of Direct-Drive Legged Robots,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, 2016.

[44] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, “Proprioceptive actuator design in the MIT cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots,” *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, 2017.

[45] S. Seok, A. Wang, D. Otten, and S. Kim, “Actuator design for high force proprioceptive control in fast legged locomotion,” in *Proc. of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1970–1975.

[46] S. Latour, “Smart Servo: The Difference Between Smart And Regular Servos,” 2016, accessed: 27-09-2017. [Online]. Available: http://www.robotshop.com/blog/en/smart-servo-motors-part-1-the-difference-between-smart-and-regular-servos-18166

[47] R. Ham, T. Sugar, B. Vanderborght, K. Hollander, and D. Lefeber, “Compliant actuator designs,” *IEEE Robotics & Automation Magazine*, vol. 16, no. 3, pp. 81–94, 2009.

[48] N. Jakobi, “Evolutionary robotics and the radical envelope-of-noise hypothesis,” *Adaptive Behavior*, vol. 6, no. 2, pp. 325–368, 1997.

[49] Autodesk, “Inventor - 3D CAD software for product development,” accessed: 29-03-2018. [Online]. Available: www.autodesk.eu/products/inventor

[50] KTR Systems, “ROTEX GS Compact,” accessed: 16-03-2018. [Online]. Available: https://www.ktr.com/en/products/power-transmission-technology/couplings/backlash-free-servo-couplings/rotex-gs-backlash-free-servo-couplings/rotex-gs-compact/

[51] C. Semini, V. Barasuol, T. Boaventura, M. Frigerio, and J. Buchli, “Is active impedance the key to a breakthrough for legged robots?” *Springer Tracts in Advanced Robotics*, vol. 114, pp. 3–19, 2016.

[52] X. Xiong, F. Wörgötter, and P. Manoonpong, “An Adaptive Neuromechanical Model for Muscle Impedance Modulations of Legged Robots,” *in Proc. of Dynamic Walking 2012 (DWC2012), May 21-25, 2012 Florida, United States*, no. c, pp. 2–4, 2012.

[53] Designing Buildings, “Stair design,” 2017, accessed: 27-09-2017. [Online]. Available: https://www.designingbuildings.co.uk/wiki/Stair_design

[54] Robotis, “Dynamixel,” accessed: 21-01-2018. [Online]. Available: http://www.robotis.us/dynamixel/

[55] D. Robot, “HerculeX Servo Series,” accessed: 21-01-2018. [Online]. Available: www.dongburobot.com/jsp/cms/view.jsp?code=100782

[56] S. Latour, “Smart Servo: A1-16,” accessed: 21-01-2018. [Online]. Available: https://www.xyzrobot.com/us_en/product/edutainment-robot/robot-kits/smart-servo

[57] J. Y. Kim and B. H. Jun, “Design of six-legged walking robot, Little Crabster for underwater walking and operation,” *Advanced Robotics*, vol. 28, no. 2, pp. 77–89, 2014.

[58] C. Benson, “Robot Leg Torque Tutorial,” accessed: 20-02-2018. [Online]. Available: https://www.robotshop.com/blog/en/robot-leg-torque-tutorial-3587

[59] L. Allen, “Finding force on an AX-12 leg, using joint torques,” accessed: 20-02-2018. [Online]. Available: http://www.lukeallen.com/AX12robotlegforce.html

[60] M. Nitulescu, M. Ivanescu, V. D. H. Nguyen, and Manoiu-Olaru, “Designing the Legs of a Hexapod Robot,” *in Proc. of the IEEE International Conference on Robotics and Automation*, vol. 3, no. December, p. 2015, 2010.

[61] A. Mahapatra, D. K. Pratihar, and S. S. Roy, “Modeling and Simulation of Wave Gait of a Hexapod Walking Robot: A CAD/CAE Approach,” *IAES International Journal of Robotics and Automation (IJRA)*, vol. 2, no. 3, pp. 104–111, 2013.

[62] S. Aoi, P. Manoonpong, Y. Ambe, F. Matsuno, and F. Wörgötter, “Adaptive control strategies for interlimb coordination in legged robots: A review,” *Frontiers in Neurorobotics*, vol. 11, no. AUG, pp. 1–21, 2017.

[63] T. Nachstedt, C. Tetzlaff, and P. Manoonpong, “Fast dynamical coupling enhances frequency adaptation of oscillators for robotic locomotion control,” *Frontiers in Neurorobotics*, vol. 11, no. MAR, pp. 1–14, 2017.

[64] M. Thor and P. Manoonpong, “Error-based learning mechanism for fast online adaptation in robot motor control,” *IEEE Transactions on Neural Networks and Learning Systems (Under Review)*, 2018.

[65] F. Pasemann, M. Hild, and K. Zahedi, “SO(2)-Networks as Neural Oscillators,” *Computational methods in Neural Modeling*, vol. 2686, pp. 144–151, 2003.

[66] B. Schneider, “A Guide to Understanding LiPo Batteries,” accessed: 17-09-2018. [Online]. Available: https://rogershobbycenter.com/lipoguide/

[67] K. Jensen, M. Larsen, S. H. Nielsen, L. B. Larsen, K. S. Olsen, and R. N. Jørgensen, "Towards an open software platform for field robots in precision agriculture," *Robotics*, vol. 3, no. 2, pp. 207–234, 2014.

[68] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[69] S. Ivaldi, J. Peters, V. Padois, and F. Nori, "Tools for simulating humanoid robot dynamics: A survey based on user feedback," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov 2014, pp. 842–849.

[70] R. Der and G. Martius, *The LpzRobots Simulator.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 293–308.

[71] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[72] Bullet, "Bullet Real-Time Physics Simulation," accessed: 13-03-2018. [Online]. Available: https://pybullet.org/wordpress/

[73] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: multibody dynamics for biomedical research," *in Proc. of IUTAM*, vol. 2, pp. 241 – 261, 2011.

[74] J. Lee, M. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. Srinivasa, M. Stilman, and C. Karen Liu, "Dart: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, 02 2018.

[75] L. S.-C. Nogueira, "Comparative analysis between gazebo and v-rep robotic simulators," *in Proc. of Seminario Interno de Cognicao Artificial - SICA 2014*, 2014.

[76] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[77] J. Julio and S. Alain, "Newton Dynamics," accessed: 13-03-2018. [Online]. Available: newtondynamics.com/

[78] cmlabs, "Vortex," accessed: 13-03-2018. [Online]. Available: https://www.cm-labs.com/

[79] Coppelia Robotics, "Virtual Robot Experimentation Platform USER MANUAL," accessed: 13-03-2018. [Online]. Available: http://www.coppeliarobotics.com/helpFiles/

[80] J. Langaa and J. T. Nielsen, "Modeling, control, and simulation of a pipe inspection robot," Bachelor's Thesis, University of Southern Denmark, Campusvej 55 5230 Odense M, 2018.

[81] L. Pitonakova, "V-REP, Gazebo or ARGoS? A robot simulators comparison," accessed: 13-03-2018. [Online]. Available: http://lenkaspace.net/blog/show/120

[82] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: http://robotics.sciencemag.org/content/4/26/eaau5872

[83] Adafruit, "Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055," 2015, accessed: 22-01-2019. [Online]. Available: https://www.adafruit.com/product/2472

[84] BlinkStick, "BlinkStick Square," 2019, accessed: 22-01-2019. [Online]. Available: https://www.blinkstick.com/products/blinkstick-square

[85] Ansible, "How Ansible works," accessed: 22-11-2018. [Online]. Available: https://www.ansible.com/overview/how-ansible-works

[86] D. J. Bernstein, "Daemontools," accessed: 22-11-2018. [Online]. Available: https://cr.yp.to/daemontools.html

[87] ROBOTIS, "Dynamixel Workbench," accessed: 22-11-2018. [Online]. Available: http://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_workbench/

[88] Andre, "Understanding Linux CPU Load - when should you be worried?" 2009, accessed: 03-12-2018. [Online]. Available: http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages

[89] Jochen Sprickerhof, "Rosbag," 2015, accessed: 14-02-2019. [Online]. Available: http://wiki.ros.org/rosbag

Appendix A

New foot design with 3D force sensor

The new foot design with a 3D force sensor and a silicone pad as a compliant element. The design is inspired by a shoe that is equipped with a rubber sole and a silicone/gel insole.

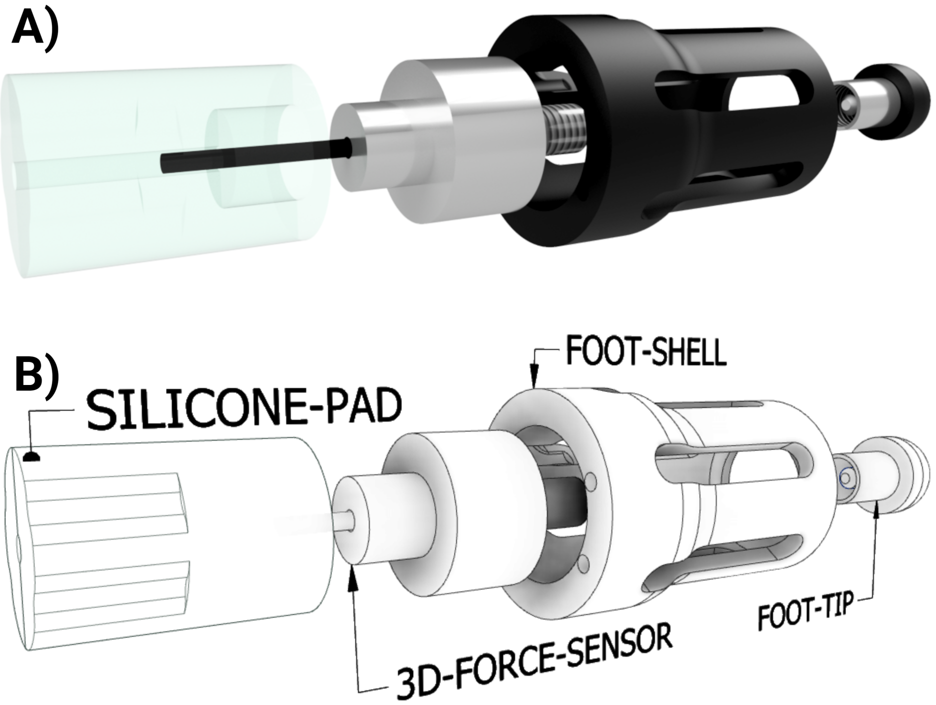


Figure A.1 – Exploded view of the new foot. **A)** shows a realistic render of the new foot. **B)** shows the names of the different parts.

Appendix B

Technical drawings for the legs

Figure B.1 shows the technical drawings for the short insect leg.

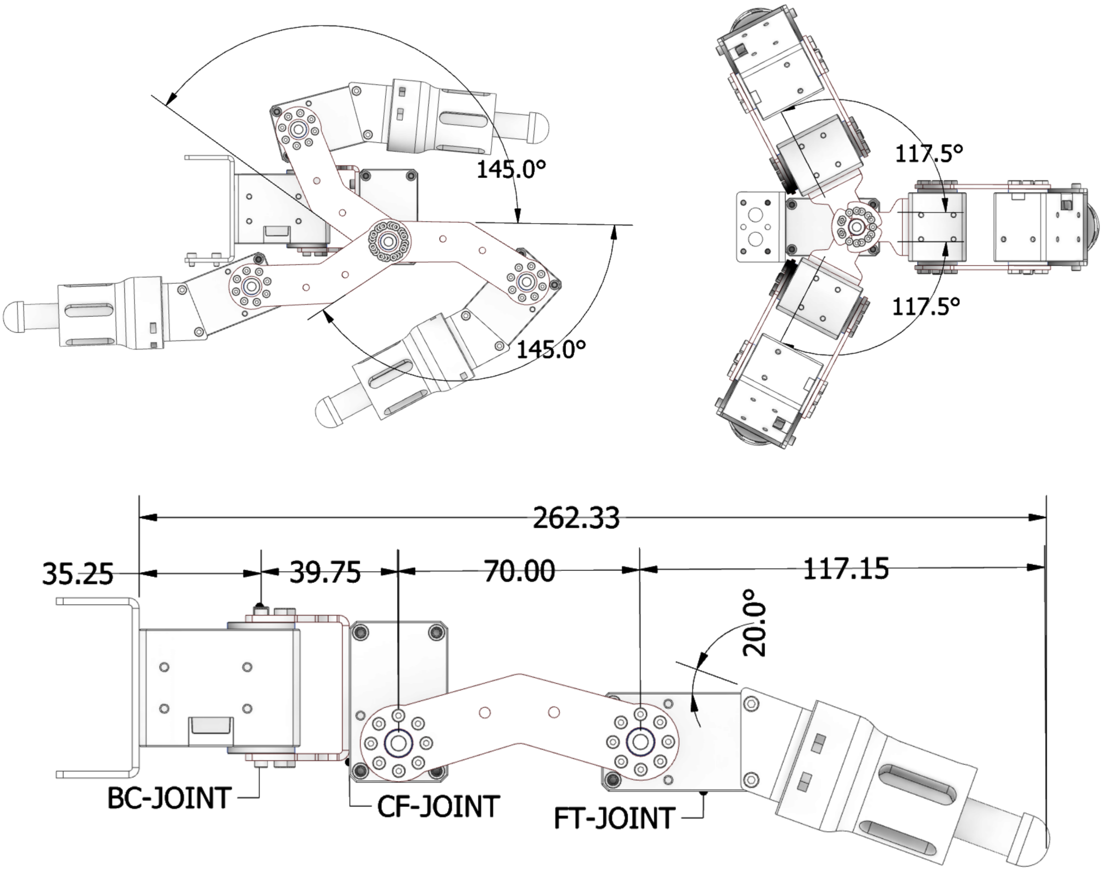


Figure B.1 – Technical drawings for the short insect like leg configuration shown in Fig. 4.11B. **A)** shows the vertical range of movement (side view), **B)** shows the horizontal range of movement (top view), and **C)** shows the leg dimensions. All units in (mm).

Figure B.2 shows the technical drawings for the mammal leg.

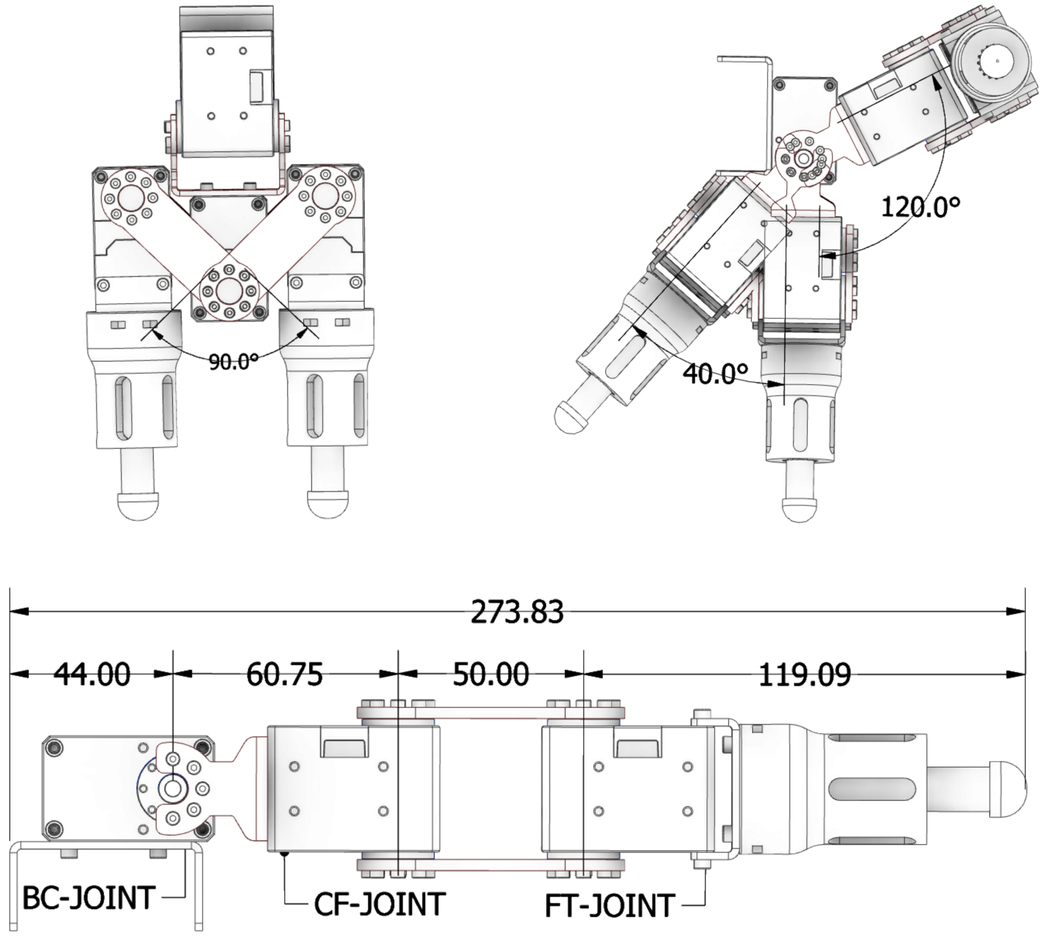


Figure B.2 – Technical drawings for the mammal like leg configuration shown in Fig. 4.11C. **A)** shows the range of movement for the CF-joint (front view), **B)** shows the range of movement for the BC-joint (side view), and **C)** shows the leg dimensions. All units in (mm).

Appendix C

Locomotion controller class diagram

Figure C.1 shows a UML class diagram of the locomotion controller used on MORF. The controller class is responsible for creating the CPG and ROShandler classes. It uses the methods of the CPG class to generate new joint positions for MORF. The ROS handler class is associated with the software on the physical robot (i.e., the hardware interfacing software). The controller uses this class to send the generated positions to the real-world robot.

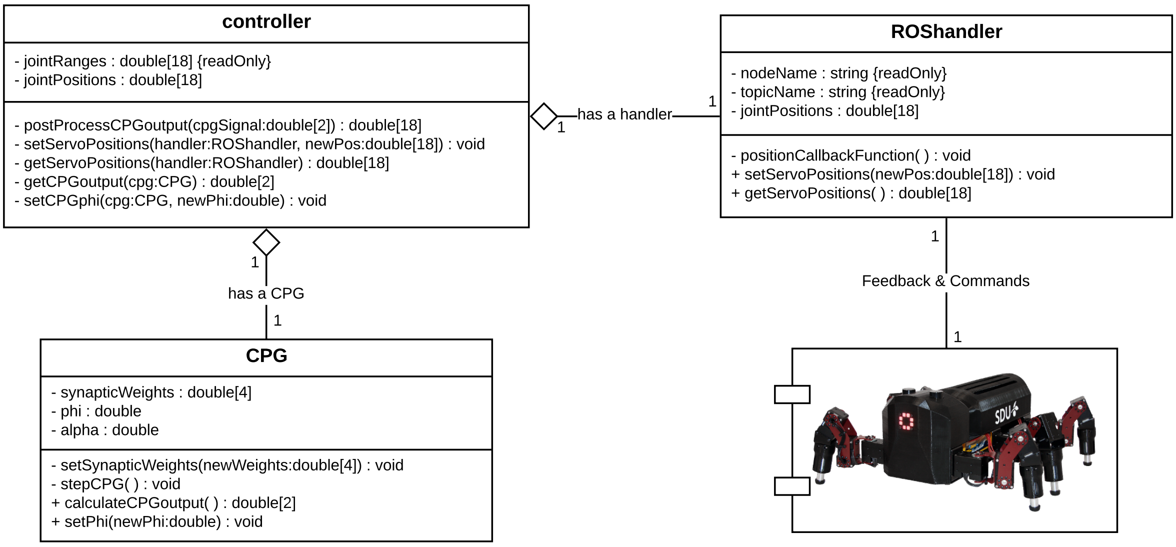

controller
- jointRanges : double[18] {readOnly}
- jointPositions : double[18]
- postProcessCPGoutput(cpgSignal:double[2]) : double[18]
- setServoPositions(handler:ROShandler, newPos:double[18]) : void
- getServoPositions(handler:ROShandler) : double[18]
- getCPGoutput(cpg:CPG) : double[2]
- setCPGphi(cpg:CPG, newPhi:double) : void
1
has a handler
1
ROShandler
- nodeName : string {readOnly}
- topicName : string {readOnly}
- jointPositions : double[18]
- positionCallbackFunction( ) : void
+ setServoPositions(newPos:double[18]) : void
+ getServoPositions( ) : double[18]
1
has a CPG
1
CPG
- synapticWeights : double[4]
- phi : double
- alpha : double
- setSynapticWeights(newWeights:double[4]) : void
- stepCPG( ) : void
+ calculateCPGoutput( ) : double[2]
+ setPhi(newPhi:double) : void
1
Feedback & Commands
1
SDU

Figure C.1 – UML class diagram of the locomotion controller used on MORF. The component node with the image of MORF is the physical robot on which the dynamixel and sensor ROS nodes run.

Appendix D

Sensory comparisons for a middle leg

Figure D.1 to D.3 shows the sensory information from the three actuators in a middle leg when MORF is walking straight on flat ground for 5 seconds. The blue signals are from the sensors implemented in the simulation and the red signals are from the sensors on the real-world MORF. All four plots also include the absolute error between sensor values from the simulated and real-world. The mean absolute error (MAE) for all sensory comparisons is shown in table D.1. Note that all the figures can also be found in the supplementary materials in the `Report_figures` directory for a better view.

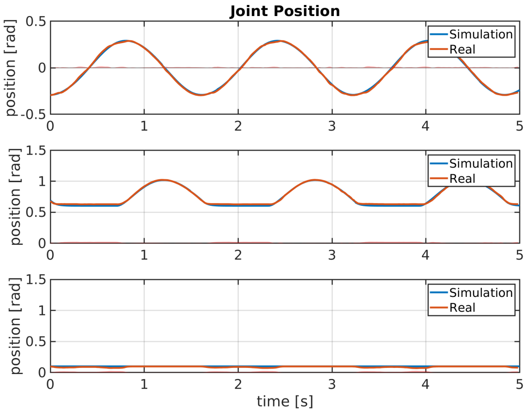


Figure D.1 – Position feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

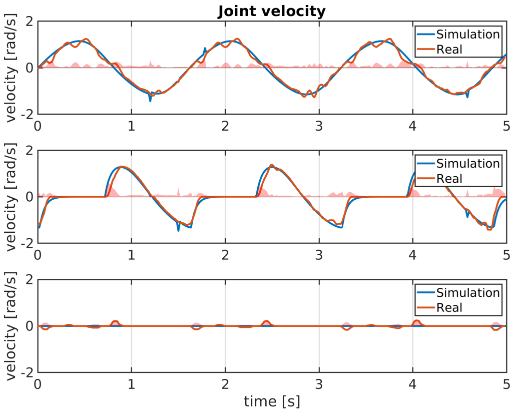


Figure D.2 – Velocity feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

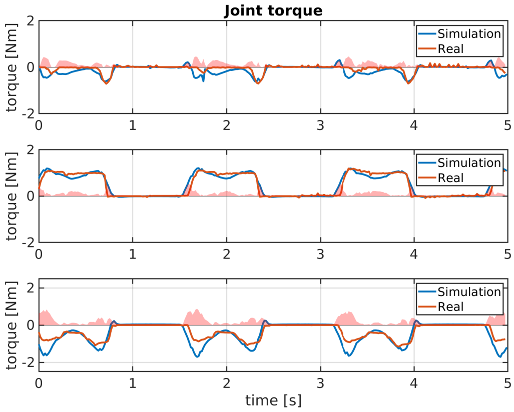


Figure D.3 – Torque feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

Table D.1 – The mean absolute error (MAE) between the sensory information from simulation and the real-world MORF when MORF is walking straight.

Sensor Name	**MAE**
BC joint position	0.0062 rad
CF joint position	0.0130 rad
BC joint position	0.0094 rad
BC joint velocity	0.0887 rad/s
CF joint velocity	0.0751 rad/s
FT joint velocity	0.0277 rad/s
BC joint torque	0.1037 Nm
CF joint torque	0.0865 Nm
FT joint torque	0.1753 Nm

Figure D.1 to D.3 shows the sensory information from the actuators in a middle leg when the robots are walking and turning to the left and right on flat ground. Figure 7.15 shows orientation information from the IMU. The blue signals are from the sensors implemented in the simulation and the red signals are from the sensors on the real-world MORF. All four plots also include the absolute error between sensor values from the simulated and real-world. The mean absolute error (MAE) for all sensory comparisons is shown in table D.2. Note that all the figures can also be found in the supplementary materials in the `Report_figures` directory for a better view.

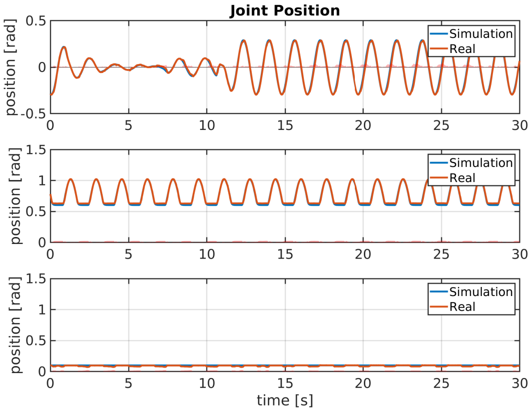


Figure D.4 – Position feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

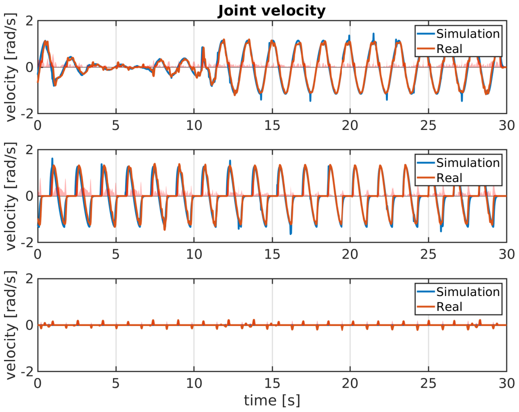


Figure D.5 – Velocity feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

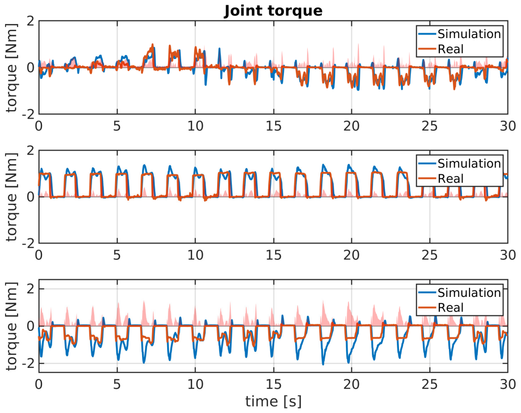


Figure D.6 – Torque feedback from the BC joint (top plot), CF joint (middle plot) and FT joint (bottom plot).

Table D.2 – The mean absolute error (MAE) between the sensory information from simulation and the real-world MORF when MORF is walking straight.

Sensor Name	MAE
BC joint position	0.0097 rad
CF joint position	0.0152 rad
BC joint position	0.0086 rad
BC joint velocity	0.0902 rad/s
CF joint velocity	0.1275 rad/s
FT joint velocity	0.1327 rad/s
BC joint torque	0.1327 Nm
CF joint torque	0.1370 Nm
FT joint torque	0.2493 Nm