Master Thesis Project

Human-like Crawling for Humanoid Robots
- Gait Evaluation on the NAO robot

Author: Andreas Aspernäs
Supervisor: Johan Hagelbäck
Examiner: Welf Löwe
Semester: VT 2018
Course Code: 4DV50E
Subject: Computer Science
Abstract

Human-robot interaction (HRI) is the study of how we as humans interact and communicate with robots and one of its subfields is working on how we can improve the collaboration between humans and robots. We need robots that are more user friendly and easier to understand and a key aspect of this is human-like movements and behavior. This project targets a specific set of motions called locomotion and tests them on the humanoid NAO robot. A human-like crawling gait was developed for the NAO robot and compared to the built-in walking gait through three kinds of experiments. The first one to compare the speed of the two gaits, the second one to estimate their stability, and the third to examine how long they can operate by measuring the power consumption and temperatures in the joints. The results showed the robot was significantly slower when crawling compared to walking, and when still the robot was more stable while standing than on all-fours. The power consumption remained essentially the same, but the crawling gait ended up having a shorter operational time due to higher temperature increase in the joints. While the crawling gait has benefits of having a lower profile than the walking gait and could therefore more easily pass under low hanging obstacles, it does have major issues that needs to be addressed to become a viable solution. Therefore these are important factors to consider when developing gaits and designing robots, and motives further research to try and solve these problems.

Keywords: Human-robot interaction, HRI, Human-like crawling, NAO, Humanoid robots, Locomotion, Crawling gait, Walking gait
Preface

Robotics is something that have always interested me, but so far both academically and career wise I have been going down a different path. You never know what you will be working on in the future but to me this was a potential once in a lifetime opportunity. I would like to thank the Linnaeus University for giving me the opportunity to work on this project and allowing me access to a real robot to use in my experiments. I would also like to express my thanks to my supervisor Johan Hagelbäck for supervising me and offering his advice and support when I needed it.

Doing this project has been a true learning experience. Robotics in general and the NAO robot in particular was something completely new to me and there are unique problems you can encounter when writing something in software with the goal of achieving a desired effect in the physical world and not just within a computer.

It has been a long and hard journey with many obstacles, changes and revisions to the original idea, so again I would like to thank my supervisor, Johan Hagelbäck and also my course manager Narges Khakpour for believing in me and allowing me to continue working despite setbacks. I would like express my thanks to opponents Charilaos Skandylas, Maria Ulan and Ayoub Yaghmaelishah, and examiner Welf Löwe for providing me with valuable feedback, challenging my theories and help me improve my thesis. Lastly I would like to express my thanks to my friends and family for supporting me during this time and help me realize a dream.
## Contents

1 **Introduction** 1  
1.1 Background 1  
1.2 Related work 2  
1.2.1 Low-Profile Crawling for Humanoid Motion in Tight Spaces 2  
1.2.2 Generating Human-like Motion for Robots 3  
1.2.3 Humanoids that Crawl: Comparing Gait Performance of iCub and NAO Using a CPG Architecture 3  
1.3 Motivation 4  
1.4 Problem Statement 4  
1.5 Contributions 5  
1.6 Target group 5  
1.7 Outline 6  

2 **Technical Background** 7  
2.1 The NAO robot 7  
2.2 NAOqi 8  
2.3 Python 9  
2.4 Choregraphe 9  
2.5 Flow Diagram 9  

3 **Approach** 11  
3.1 Scientific approach 11  
3.2 Method description 11  
3.3 Reliability and Validity 11  
3.4 Summary 12  

4 **Implementation** 13  
4.1 Design of Flow Diagrams 13  
4.2 Crawling Gait 16  
4.3 Measurement Readings 17  

5 **Evaluation** 19  
5.1 Detailed Experiments 19  
5.2 Experiments Results 21  
5.3 Heat map 26  
5.4 Discussion 29  

6 **Conclusions and Future Work** 30  

References 32  

A **Appendix 1** A  
A.1 Static All-fours A  
A.2 Active-All-fours C  
A.3 Crawling Gait F  
A.4 Readings I  
A.5 Measurements K
1 Introduction

The field of robotics has come a long way; machines have gone from being merely tools being capable of autonomous actions and decision making when interacting with their environment. Robots can help us by performing a number of different tasks, but the future of robotics is not just to have robots working on their own or together with other robots, but also being able to collaborate with a human partner on a shared task. This is one of the goals of human-robot collaboration research and one of the key challenges is enabling good communication between human and robot. If a robot’s actions are difficult to understand then the teamwork is affected negatively. Humanoid robots gives us a familiar form to identify with, but human-like features is merely the first step, research like [1], [2], [3] have indicated that if the robot exhibits human-like movements then collaboration is improved.

The focus of this thesis is a particular set of motions called locomotion, which is the act of moving from one place to another. There are several kinds of locomotion, e.g. walking, crawling, running and jumping, and different styles can give advantages and disadvantages depending on the situation. This project will be focusing on analyzing crawling and walking for the humanoid NAO robot. The robot does not however come with a crawling behavior, so in order to test a human-like crawling gait a new behavior needs to be developed.

The first of the research questions was if a human-like crawling gait could be developed for the NAO robot capable of sustained forward momentum. This was achieved by developing a crawling gait behavior in Python and tested on a physical NAO robot. The second research question was about the performance of the crawling gait in terms of speed, stability, power consumption and joint temperature. The third and final questions was how the performance of the crawling gait would compare to the built-in walking gait. The results from the experiments showed that the walking gait was superior in all tests except in power consumption where both gaits was essentially equal. The temperature in the joints increased at a higher rate in the crawling gait, giving it a shorter operational time than the walking gait. These results does however open up possibilities of improvement in terms of code optimizations and design. Not all motors needs to be turned on all the time and if they are turned off while not needed then temperature and power consumption could possibly be decreased. Also, the speed could potentially be improved for the crawling gait if the robot was designed with knees with better friction against the floor so it would not slide as easily.

1.1 Background

Robots can be autonomous or semi-autonomous. Semi-autonomous robots require confirmation from a human operator before executing certain actions, while autonomous robots are capable of decision making and taking actions based on this decision. But robots are in the end machines, and machines are to some extent tools built with a specific purpose in mind. One such purpose can industrial assembly lines and their form of industrial robots often reflects that purpose in order to make the robot more effective at its given task. Humanoid robots are set apart from other robots because they are instead built to resemble the human body [4]. If familiarity gives us comfort and understanding then humanoid robots could have potential to improve human-robot collaborations.
To work in a human environment the robot needs to be able to move around in its surroundings, and humanoid robots can be beneficial compared to other robots because they can be used in different human environments without needing to make any changes [4]. The ability to move from one place to another is called locomotion [5]. For humans and humanoid robots the most common form of locomotion is walking, but when we need to enter a tight space or pass under low-hanging obstacles, other kinds of locomotion such as crawling can be more beneficial. The locomotion is created by us moving our limbs in a repeating pattern to create momentum, and this repeating pattern is called a gait. Whether we run, walk or crawl, all three kinds of locomotion use different kinds of gaits. However, even if two individuals use the same general pattern to move there might still be slight variations between them, and even if a humanoid is crawling on its hands and knees, is does not mean the gait is human-like.

1.2 Related work

This chapter will discuss previous robotic research that are relevant for this thesis. They were chosen because each touches upon a key aspect of this project and serves to position this work in relation to other research in the field.

1.2.1 Low-Profile Crawling for Humanoid Motion in Tight Spaces

In [6] the authors developed a low profile crawling gait. They have noticed some problems or limitations with an infant-like crawling gait related to the physical height of the robot. When moving under low hanging obstacles a low profile gait could prove to be advantageous. To control the robot, software was developed and tested on the NAO robot. They ran several experiments which demonstrated their proposed crawling pose. The robot would approach a low hanging obstacle and when a red dot appeared in the robot’s camera view, the robot would recognize the dot, duck under the obstacle and then crawl under it. After passing the obstacle the robot would then return to its standing pose. The developed gait was divided into three different motion segments. When the robot crawled these three segments are repeated over and over until the robot has passed the obstacle and stands up again. The authors also developed algorithms calculate the motions and letting the robot compute the different angles each joint needed to be set to in order to create the motion. The proposed solution was directly applicable to any robot with basic human-like joint structure, that of ankle, knee, hip, shoulder. Though not clearly stated, the robot approaches the red dot and only when getting within a certain distance the crawling gait was activated. They also never stated if they used the built in walking functionality in the NAO robot when it walked up to the obstacle, but by not mentioning it is possible to assume they did. Their proposed crawling gait did effectively lower the height of the robot, but it did however increase the width, which can a challenge when crawling through tight spaces.

[6] was a great inspiration for this work as they also studied and developed a crawling gait for the NAO robot. Their goal was however not a human-like gait, instead focusing on lowering the profile of the robot while crawling to pass under low-hanging obstacles. Their results showed that a crawling gait for the NAO was possible, even using an algorithm to calculate the motions, they did not consider the potential benefits of using human-like motions. The goal of this thesis is to show
the possibility of getting the benefits of a more human-like gait while still getting a lower profile than walking.

1.2.2 Generating Human-like Motion for Robots

In [7] the authors are discussing the benefits of humans working together with robots. When robots are working with humans, their movements and actions are hard to understand because they work differently from humans. If a robot is hard to understand then that can cause problems. This research field is referred to as human-robot interaction (HRI). The authors present an algorithm that “autonomously synthesizes human-like variants of an input motion” to create motion clarity and improve performance. The goal is to increase the understanding of the robot’s intention and in turn improve the collaboration between the robot and the human partner. To do this they created an algorithmic solution divided into three parts. The first step was to generate a human like motion based on some input, and optimizing it through spatio-temporal coordination. The second step was to create a variance in the motions. Our muscles are not as exact as mechanical ones can be, even if a human performs the same action multiple times, there is always a slight variation in the motions. By generating some variance we can create a more human-like movement pattern, giving the robot more alternatives to choose from randomly. The reasons for this, according to the authors, is that most techniques for generating motions for robots are not human-like. This is particularly important for social robots.

Humanoid robots that looks human is not enough, it needs to move like a human in order not to cause a disconnect. Even when the robot learns by watching a human move and try to replicate their motions through motion capture, information can be lost along the way, causing the robot not to move human-like. This algorithm does however require some kind of input, such as an example of a motion. The authors performed a series of experiments with both human and robot collaboration partners in order to validate the results. To test the algorithm they used a robot called Simon. It has an upper-torso humanoid robot and the robot was developed in the SIM Lab (Socially Intelligent Machines) in the College of Computing’s School of Interactive Computing [8]. The torso has two controllable DOFs (degrees of freedom), each arm has seven, four per hand, two per ear, three eye DOFs and four for the neck. They did eight experiments and the results showed improved recognition accuracy of the robots motions. This gave increased clarity, minimize distractions, misinterpreted motions and gave better collaboration fluidity. The algorithm also gave the robots the ability to synchronize their movements if they could observe their partner.

[7] was chosen because it demonstrates the value of human-like motions in robots and the possibility of further research in to this particular field of robotic science. They only tested their theory on the iCub robot and not on the NAO, as well as not looking in the performance issues of human-like locomotion, something that is also important to consider.

1.2.3 Humanoids that Crawl: Comparing Gait Performance of iCub and NAO Using a CPG Architecture

In [9] the authors developed used a central pattern generator (CPG) to create a rhythmic crawling pattern. The CPG is a neural network that in animals acts as an interface between higher-level brain centers and the spinal cord to reduce necessary
bandwidth between them. The generated crawling gait was tested on the NAO and iCub robots while measuring their performance.

While performance analysis of crawling gaits in robots is a key aspect in both their work and this one, both the approach on how to control the robot and how gaits are compared are different. [9] does a comparison on gait performance but they used a CPG architecture instead of a step-by-step manual coding approach of this thesis. The purpose of their work was also to compare two different robots using the same gait, while this thesis compares to same gait on two different kinds of robots.

1.3 Motivation

If a robot is going to operate and collaborate with humans in a human environment it needs to be able to perform any number of different tasks [10]. Tasks that are very simple for humans such as correctly recognizing objects or walking down stairs can however be very difficult for a robot. This opens up many possibilities for different kinds of research where complex tasks can be broken down into smaller components and solved separately. For example in [11] A. L. Edsinger discusses the problems for robots operating in human environments. Because our environments can be dynamic and difficult to predict, they provide special challenges that the robots need to overcome. Edsinger addresses these challenges and presents advances in robot manipulation by illustrating the design behind the humanoid robot Domo and its method for assisting a person in everyday tasks. Edsinger also discusses some general strategies for building robots with the purpose of operating along side humans in their homes and workplaces.

The reason for choosing the NAO robot was due to its lightweight and versatile design, making it excellent for experimental purposes [12]. Due to its small and otherwise friendly appearance the NAO robot could at first glance be thought of as a simple toy, but it would work the same way even if it was two meters tall. The smaller size makes it both cheaper and easier to handle. The NAO is also easy to control with either directly writing code, or building applications by connecting boxes containing different kinds of behaviors. It has been built to provide both quality and performance while still being affordable, making it a viable option for universities and also primary schools wishing to do research on robots and artificial intelligence, or simply learning the basics of robot interaction and control [12].

The problems facing robot developers are many and solutions are dependent on the robotic platform in question. The NAO robot is a popular platform for research and for many different possible applications, something that is demonstrated by the wide variety of published reports [13][14][15]. This thesis focuses on developing a human-like crawling gait for the NAO robot and evaluating its performance. The results of this work will help further the field of robotics and getting us one step closer, albeit a small one, to robots and humans working together in the future.

1.4 Problem Statement

There are benefits of using human-like crawling movements but we need to know more about the implications of doing so. The NAO itself might even prove unable to crawl like a human due to limitations in its design. Should a human-like crawling gait be possible questions will still remain in regards to its performance and how well it would compare to the standard built-in walking gait.
When evaluating locomotive gaits two important metrics are speed and for how long the robot can operate. Power consumption is vital for operational time if the robot uses on-board battery power to function, but for robots like the NAO that also uses electrical motors to move its joints and actuators, the major limiting factor could be joint temperature. The electric current in the motors makes the joints heat up and if the current is not lowered or turn off completely the motors would eventually get overheated and break.

After measuring the performance of the human-like crawling gait, the same experiments will be conducted on the built-in walking gait of the NAO robot and by comparing them to discover possible issues, strengths or weaknesses, of using a human-like crawling movements. The problems described above can be broken down into the following three research questions:

<table>
<thead>
<tr>
<th>RQ1</th>
<th>Can a human-like crawling gait be developed for the NAO robot capable of sustained forward momentum?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ2</td>
<td>How would the crawling gait perform in terms of speed, stability, joint temperature and power consumption?</td>
</tr>
<tr>
<td>RQ3</td>
<td>How would the crawling gait compare to the standard walking gait of the NAO robot?</td>
</tr>
</tbody>
</table>

The expected result of this thesis is that a human-like crawling gait is possible for the NAO robot, and that we will have gained a greater understanding of potential problems of using human-like crawling gaits. Performance-wise it is expected that the power consumption to be approximately the same due to all motors in the joints being turned on in both gaits, regardless if the joint is actively being used or not. Speed is expected to be slightly slower for the crawling gait but temperatures to be overall the same. Profile dimensions such as height will be lower when crawling and because it will have lower center of gravity it should also be more stable than when standing up [16][17].

1.5 Contributions

The goal of this project is to develop a single human-like crawling gait and measure its performance. It would be interesting to develop a second human-like crawling gait, one based on walking on its elbows rather than its hands, but it is outside the scope of this project. This thesis will also not compare other crawling gaits developed in previous works such as [6]. This project will hopefully be a baseline test that others can build upon. There are also potential for optimization that could lower power consumption and temperature increase by lowering the stiffness in certain joints or completely turning them off while not needed.

1.6 Target group

The target group is primarily researchers of robotic movements, in particular those who study human-like motions in humanoid robots, but designers and software developers of robots could also gain some benefit.
1.7 Outline

The outline of the report is as follows. Chapter 2 is the technical background and will explain the different technologies involved in the project. Chapter 3 describes the chosen scientific approach and how the results are verified. Chapter 4 describes the implementation and will explain the developed crawling gait and how the robot was controlled. In chapter 5 the results of the experiments are presented and analyzed. Finally in chapter 6 reflections will be made on the project, possible conclusions drawn and end with a discussion on possible future work.
2 Technical Background

This chapter will describe the different technologies used in the project. It will serve as a guide to understand the implementation used to answer the previously stated research questions and lay the necessary fundamentals for evaluating and discussing the results of this thesis.

2.1 The NAO robot

The NAO robot is a humanoid robot developed by Aldebaran in 2006 [18]. The robot has gone through several iterations and the latest is version five. It stands 58 cm tall and approximately 10 000 units have been sold throughout the world. NAO is an interactive companion robot and is able to interact with humans. It has 25 degrees of freedom and is able to “lift its arms”. It is able to adapt to the world around it through its inertial unit, keeping its balance. It has several different types of sensors such as tactile sensors, bumpers, sonars, two cameras, four directional microphones and two load speakers. It can send and receive data through either an Ethernet cable or WiFi. [19]

![Figure 2.1: Overview of the NAO robot. Image taken from [20]](image)

Specifications are as follows:

<table>
<thead>
<tr>
<th>Motherboard</th>
<th>Connectivty</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM Z530 CPU</td>
<td>Ethernet</td>
</tr>
<tr>
<td>1.6 GHz</td>
<td>1 x RJ45 - 10/100/1000 base T</td>
</tr>
<tr>
<td>RAM 1 GB</td>
<td>WiFi</td>
</tr>
<tr>
<td>Flash Memory 2 GB</td>
<td>IEEE 802.11 b/g/n</td>
</tr>
<tr>
<td>Micro SDHC 8GB</td>
<td>USB 1 x USB</td>
</tr>
<tr>
<td>Battery</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Nominal voltage/capacity</strong></td>
<td></td>
</tr>
<tr>
<td>21.6 V / 2.25 Ah</td>
<td></td>
</tr>
<tr>
<td><strong>Max charge voltage</strong></td>
<td></td>
</tr>
<tr>
<td>25.2 V</td>
<td></td>
</tr>
<tr>
<td><strong>Recommended charge current</strong></td>
<td></td>
</tr>
<tr>
<td>1.8 A</td>
<td></td>
</tr>
<tr>
<td><strong>Max charge / discharge current</strong></td>
<td></td>
</tr>
<tr>
<td>2.3 A / 2.0 A</td>
<td></td>
</tr>
<tr>
<td><strong>Energy</strong></td>
<td></td>
</tr>
<tr>
<td>48.6 Wh</td>
<td></td>
</tr>
<tr>
<td><strong>Charging duration</strong></td>
<td></td>
</tr>
<tr>
<td>&lt; 3h00</td>
<td></td>
</tr>
<tr>
<td><strong>Autonomy</strong></td>
<td></td>
</tr>
<tr>
<td>60 min (active use) 90 min (Normal use)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.2: Overview of the NAO robot. Image taken from [21]

2.2 NAOqi

The NAOqi OS [22] is the operating system of the robot and is an embedded GNU/Linux distribution based on Gentoo Linux. It was developed by Aldebaran
to specifically fit the needs of the robot by providing and running programs and libraries that together brings the robot to life. The main software of the NAOqi OS is NAOqi which is responsible for running behaviors on the robot but can also be used to test the code on a simulated robot on your computer. The NAOqi Framework provides two extensive libraries of NAOqi APIs for both C++ and Python.

2.3 Python

Python [23] is a high level and object-oriented programming language made for general-purpose programming and was developed and published by G. van Rossum in 1991. It is portable as it runs on different operating systems including several Unix variants, macOS, and on Windows 2000 and later.

2.4 Choregraphe

Choregraphe [24] is a desktop application available on multiple platforms including Linux, Windows and Mac. It allows developers to create applications containing animations, behaviors and dialogues and test them out on either a physical or simulated robot, as Choregraphe also includes a virtual robot simulator.

2.5 Flow Diagram

The flow diagram is what makes up the behavior of the NAO robot [25]. It has a starting point where the behavior begins, and an endpoint where the behavior ends. It is possible to place different boxes into the flow diagram that can perform certain tasks like moving a hand, saying something etc. A behavior will start executing from the starting point and move into one or more boxes. It is therefore possible for multiple boxes to be executed and run at the same time. When the execution reaches the end the behavior is over. These boxes can also contain their own flow diagrams for even more complex behaviors. If there is no box for a particular desired behavior it is also possible to use Python boxes in which you can write Python code to express the required behavior when executed. Experimentation usually required as all syntactically incorrect behaviors was only discovered at run-time.

![Flow Diagram](image)

Figure 2.1: Crawling behavior.

Each box as one or two “entries” on its left side where the execution can flow into. A single entry (usually in the form of a empty white square) signify a single kind of behavior, while two signifies one start-behavior (play-icon) and one stop-behavior (x-icon). Each box can also have zero or more exits where the execution can flow out of the box when the behavior inside the box is completed. The “Motors On”-node shown in figure 2.1 has both an empty entry and exit, this signifies that
the execution will simply enter the box, execute a single behavior and then exit the node. The “Static-All-fours”-node on the other hand one start-behavior and one stop-behavior. Notice also that its exit point is a combined red X and a black play-icon; this means that no matter if the execution passed in through the start-entry or the stop-entry it is exit through this exit point if another node is connected to it. In contrast the “Tactile Head”-node has both the combined play-x-icon and the empty exit points. Same as before the execution will exit the node through the play-x point if another node is connected to it. If not not then the execution in that particular node will be halted until a condition is fulfilled, in this case one of the three buttons on the robots head is pressed, one for each empty exit point.

One important thing to consider is that although the overall flow diagram starts with a single line of execution it can split into multiple. So from one box the execution can be passed to one or more boxes, making it possible to execute multiple kinds of behaviors simultaneously. Execution can even be passed back to previous executed nodes and both entry and exit points of a node can be connected multiple nodes. It is even possible to connect two points on the same node to create the desired overall behavior of the robot, making the flow diagram a very flexible tool.
3 Approach

This chapter will explain the approach used to conduct the research of this thesis. It will present the methods used to answer the research questions as well as discuss what efforts have been made to achieve high reliability. It will also clarify how the results are to be validated.

3.1 Scientific approach

In order to answer the previously stated research questions the scientific approach was to develop a human-like crawling gait and evaluate it. This will primarily be done using an empirical method and collecting quantitative data through experiments. The research also contains qualitative aspects as there is a need to determine what qualifies as a human-like crawling gait.

3.2 Method description

The goal is to develop and evaluate the performance of a human-like crawling gait by measuring speed, how long it can operate, and stability. When considering how long the robot can operate both heat and power can be limiting factors, as overheating of motors or loss of power will halt the robots movements. Different movement patterns can effect both the energy consumption as well as the heat output of the motors. By examining this we can get a better understanding when developing gaits for humanoid robots.

In order to answer the research questions, the crawling gait needed to be developed and evaluated based on three aspects. Firstly it needed to be able to produce a continued forward crawling motion without falling over. Secondly the gait needed to have similar movement pattern to that of a crawling human in order for it to be considered human-like. Thirdly the performance of the gait was compared to the standard walking gait of the NAO robot based on speed, stability, joint temperature and power consumption.

3.3 Reliability and Validity

Reliability [26] deals with how well others could replicate the work and still get the same result. In this project there are two major challenges of reliability, the first of which is the problem of the surface area the robot will be working on as different kinds of surfaces can have different attributes which could affect the robot’s ability to get a grip on the surface. Smoother surfaces could give too little friction and the arms and legs would simply slide over and not push the robot forward.

The other challenge of reliability is battery degradation. As lithium-ion batteries are charged and discharged the capacity diminishes. Tests made by the Battery University [27] on lithium-ion batteries for mobile devices showed a drop in capacity from 88–94% to 73–84% after 250 full discharge cycles. A drop in battery capacity would have a direct impact on the operational time of a battery powered robot such as the NAO. Other factors such as temperature can also have a detrimental effect on the rate of degradation as a battery stored at 0°C and at full charge showed a loss of 6% capacity after one year, while a battery stored at 25°C showed a loss of 20%.

Next is the problem of validating the results. A construct validity [28] issue is the problem of defining what is human-like. It could be considered a subjective
question but in order to evaluate the developed gait the term “human-like” needs to be broken down into some basic requirements that the gait could be validated against.

A problem of internal validity [28] is when measuring the speed of a gait. When the robot is crawling, some of the parts that gets in contact with the surface, like the knees, are made of smooth plastic that can slide easily on other smooth surfaces, making it more difficult for the robot to push itself forward. In the worst case scenario the robot will not be able to move forward at all. The walking gait receives and advantage over the crawling gait because the underside of the feet are made of a rubbery material, and should give it better purchase against the surface.

Another internal validity problem is testing the stability of the robot. The friction between the robot and the surface it is on is not enough to keep the robot from sliding off before it would otherwise fall over. This could however make it impossible to test the stability of the robot while it is actually moving, making it harder to draw conclusions about the stability of the crawling and walking gaits.

Finally there is a problem of external validity [28]. Although the scope has been limited to only focus on the NAO, it can still be questionable how much general knowledge can be acquired and used on other kinds of robots. It could also be considered a problem that a crawling gait is only compared to a walking gait, instead of another crawling gait. To make a better argument the developed human-like crawling gait should be compared to other crawling gaits in order to achieve more conclusive results.

### 3.4 Summary

In this chapter we discussed the approach of how the research questions will be answered. The goal is to develop and evaluate a crawling gait for the NAO robot through empirical experiments. The three aspects that will be measured is the speed, operational time and stability of the robot. There are two challenges to the reliability of the results and that is the particular surface the robot will be traveling on and its properties, and battery degradation and performance. There are also three problems to the validity of the results. The first is that different some parts of the robot in contact with the surface can get a better grip then others. The second that it could be impossible to accurately measure stability while the robot is moving. And the third is the crawling gait is compared to a walking gait and not an alternative crawling gait.
4 Implementation

To test the human-like crawling behavior a crawling gait was developed. The crawling gait is a pattern of repeatable motions which the robot goes through. There is no algorithm in this work that calculates each move, instead each instruction of how each joint should move is specifically coded into the behavior and tested through trial and error. Previous work like [7] and [9] have generated motion through the use of algorithms but this is something that would have been too difficult for me to implement while also not necessary to answer the research questions; therefore left out of the scope of this thesis. Choregraphe these behaviors are built using different types of boxes and the crawling gait used Python boxes containing Python code to express how each joint should be moved, what angles they should be set to and how fast it should move. The software and motors behave asynchronously, meaning that when a command to move a joint is executed the joint starts to move immediately but the execution is not paused while the joint is completing the command. Because of this, when some actions are required to be completed before doing the next action, a timed pause was added after next action to give the first action time to finish before executing the next instruction.

4.1 Design of Flow Diagrams

The Figure 4.1 shows the flow diagram for the crawling behavior. It is comprised of five nodes and execution moves from the left to the right side. The first node turns on all motor joints in the robot and execution is passed to the “Static All-fours”-node which sets all joints to their passive resting pose. Execution is then passed to the node “Tactile Head” which pauses execution until one of the three buttons on the head is pressed. The front head button calls the “Active All-fours”-node which moves the robot from its passive pose to its active pose, which is a requirement before the robot can start crawling. The second button (middle) activates the crawling gait which is a set of pattern moves that are continuously executed in a loop until halted by the user. The final button stops the execution by exiting the crawling gait loop and the program then terminates.

Figure 4.1: Crawling behavior.
The second Figure 4.2 shows a similar design for the walking behavior. The major difference is that the nodes for “Static Stand”, “Active Stand” and “Walk Gait” are built-in functionality and provided directly in Choregraphe. It should be noted however that unlike the crawling behavior the walking behavior does not require the “Active Stand” to be run before walking. That is because the “Walk Gait”-node is smart enough to figure the steps it needs to take before it can start to walk. The moves are the same as when executing it as a separate step and the reason why we separated them is to more clearly distinguish between them, and that the design for the crawling behavior and walking behavior would match up.

The Figures 4.3 and 4.4 show the behavior for the experiment of measuring the time to go from a static pose to an active pose. Normally the execution would pause for user interaction through the “Tactical Head”-node, but that would have introduced uncertainty when the active pose step was completed. To solve this, another node was used so make the robot say “Done” immediately after the active step was finished, so that the operator would know when to stop the timer.
The Figures 4.5 and 4.6 show the behaviors when measuring the speed of the two gaits. Same as before, all joints will turn on and the robot will enter its initial pose as implemented in the static-pose node. The user then manually tells the robot to go into the active pose by pressing the button on the forefront of its head. After the robot has completed its active pose the user can initialize the crawl or walk gait by pressing the middle of the buttons on the robot’s head. Once the experiment is completed the user can stop the robot by pressing the button on the back of the robot’s head.

The Figures 4.7 and 4.8 show the behaviors when measuring the temperature, status and stiffness of all joints. When the user presses on the middle button on its head the execution will be split into two, one going to the gait node that will make the robot crawl or walk. The other execution will sent to the “Timer” node will immediately activate the “Readings” node that will take measurements of battery level, as well as the stiffness, status and temperature of all the joints and log the results to the console inside of Choregraphe. After the initial readings further measurements will be logged every 30 seconds.
4.2 Crawling Gait

The crawling gait is contained in a Python box in Choregraphe. When execution is passed to the node it will execute a motion pattern of different joints to move the robot forwards. At the nodes core is the following code:

\[
\text{motionProxy} = ALProxy("ALMotion")
\]

\[
\text{while self.crawl:}
    \text{self.crawlGait(motionProxy)}
\]

First, a proxy connection is created to the ALMotion module that gives us access to all the methods of that module that is used to control the robot. Next, a loop is entered that will continue until the global variable “crawl” becomes false. This is what happens when the operator presses the designated stop button on the back of the robots head. A secondary execution command will be passed to the robot that will set the variable to false and the loop will exit. Inside the loop there is a single function call to crawlGait which contains all code for moving the robot forward two steps.

\[
\text{# Lift left arm}
\]

\[
\text{names} = \left\{ \"LShoulderPitch\", \"LElbowRoll\" \right\}
\]

\[
\text{angles} = [1.5, -1.5446]
\]

\[
\text{fractionMaxSpeed} = 0.2
\]

\[
\text{motionProxy.setAngles(names, angles, fractionMaxSpeed)}
\]

\[
\text{time.sleep(1)}
\]

\[
\text{# Start moving left arm}
\]

\[
\text{names} = \left\{ \"LShoulderPitch\", \"LElbowRoll\" \right\}
\]

\[
\text{angles} = [0.2, -0.0349]
\]

\[
\text{fractionMaxSpeed} = 0.2
\]

\[
\text{motionProxy.setAngles(names, angles, fractionMaxSpeed)}
\]

\[
\text{time.sleep(0.5)}
\]

The crawlGait function is too large to include here but will is included in the appendix. It is a list of steps that when executed causes the robot to take two steps forward. Shown above is the first two steps that first lifts the left arm and then moves it forward. The key part is the method call .setAngles which takes three arguments, an array of strings corresponding to joints to be moved, an array of double values for the target joint angle, and double value which is the fraction of the maximum speed. A value of 0.2 means that the joint will move at 20% of its maximum speed. All the
values such as the angles, speed and sleep time were not calculated in anyway but instead tested through trial and error.

### 4.3 Measurement Readings

While running the experiments there is a need to somehow log different measurements such as battery and temperature. To do this, another Python node was created that would be able to run in parallel to the code related to locomotion. The following code is a short example how it has been implemented. First, two proxies are created; from the batteryProxy information about how much battery power remains and from the memoryProxy it is possible to retrieve information from the joints about their temperature, stiffness and status.

```python
batteryProxy = ALProxy("ALBattery")
memoryProxy = ALProxy("ALMemory")

devicesHead = ["HeadPitch", "HeadYaw"]

self.logger.info(str(self.count) + "Reading———")
self.count = self.count + 1

battery = batteryProxy.getBatteryCharge()
self.logger.info("Battery:" + str(battery) + ",")

self.printDevicesTemperatureAndStatus(
    memoryProxy, devicesHead)

Due to the number of joints (a total of 25) they were divided into five arrays, one for each limb and one for the head. In the above example the `devicesHead`-array and proxy is then passed to the function `printDevicesTemperatureAndStatus`.

```python
for device in devices:
    deviceString = "Device/SubDeviceList/" + device + "/Temperature/Sensor/Value"
    temperature = memoryProxy.getData(deviceString)
    self.logger.info(device + "Temperature:" + str(temperature) + "C")

deviceString = "Device/SubDeviceList/" + device + "/Temperature/Sensor/Status"
    status = memoryProxy.getData(deviceString)
    self.logger.info(device + "Status:" + str(status))

deviceString = "Device/SubDeviceList/" + device + "/Hardness/Actuator/Value"
    stiffness = memoryProxy.getData(deviceString)
    self.logger.info(device + "Stiffness:" + str(stiffness) + "%")
```

The function `printDevicesTemperatureAndStatus` contains a single for-loop
which loops through each joint, creates a new string to send to the proxy and retrieve the data. Finally the retrieved data is logged to the screen through Log Viewer inside of Choregraphe.
5 Evaluation

In this section we will first briefly some of more important details of the experiments and then present and analyze the results from the experiments.

5.1 Detailed Experiments

The first step before the start of an experiment the robot was placed in its initial passive state. The passive state is a static pose that is not part of the crawling or walking gait patterns. While kneeling on all fours the passive state is also a static pose that the robot can remain in without falling over even if the stiffness in the joints is turned off. The second step was for the robot to go from its passive pose to its active pose; this is necessary because the passive pose was not part of the crawling or walking gaits and so needed to enter a pose that is part of it before it can start to crawl or walk. The time to complete this step is different for the crawling and walking behavior, therefore this step is completed before the measuring of the robot's speed begins. The time it takes for the robot to go from the passive pose to its active pose is also included for comparison. When executing the behaviors the robot will automatically go into its initial state, but each of the following steps was activated by the operator by pressing one of the three buttons on the robot’s head.

In order to test the stability of the robot it was first set into its initial passive state and placed on a flat surface 5.9. The surface was then tilted until the robot fell over while measuring the angle of the surface. The robot needed to be affixed to the surface because otherwise it would slide off the surface, and because of this the stability could not be tested while the robot was moving. While kneeling the robot’s left hand and leg is affixed to a flat surface and the left foot is affixed while standing.

Figure 5.9: Stability experiment.
The stability experiment starts from a horizontal starting point of zero degrees, and incline of the surface is measured until the robot falls over. The highest incline the robot can reach without falling over is considered to be its measure of stability. Each experiment was performed five times to get an average result. Same as with the speed experiments, after five times the results was precise enough to make a valid argument about the stability in the context of this work.

The walking and crawling experiments will have to be conducted on the same kind of surface, however the rubberized feet could potentially give the walking gait an advantage due to better surface grip. The speed of the robot is determined by measuring the distance the robot can travel under 60 seconds. 60 seconds was chosen because during testing the robot could travel at least one meter. One second would not be enough for the robot to take a single step and after six minutes the temperatures in the joints when crawling. To get an average of the result each speed experiment was performed five times. Fives was chosen because after performing five experiments the difference between each result was stable enough that performing more experiments would not improve the accuracy of the average value.

To understand how long the robot can operate, the robot crawled and walked on the floor for ten minutes each. At first the goal was for the experiments was to run for one hour each, but it was changed to ten minutes because after six minutes the robot was unable to crawl properly due to the joint temperature being too high. One hour would also be too unpractical because the need of constant monitoring so it does not gets stuck against a wall or object. ten minutes was also enough to record a change in battery level of the robot. The measuring of battery level, as well as temperature, stiffness and status of the joints of the robot was recorded every 30 seconds because the experiments only was ten minutes long. If it was recorded every minute a graph would only get ten data points and some measurements could spike and recede in that interval and therefore be invisible. 30 seconds was instead chosen so those potential spikes could be recorded. The temperature of the robot is visualized using a heat map. A heat map is a image divided into color-coded sections with different colors corresponding to different temperature ranges. This is to help the reader to understand the results without being overloaded with information.

The robot rested for 12 hours between experiments so that the battery had time to recharge and the joints to cool down. 12 hours was a practical number because the temperature and battery was measured after 12 hours after an experiment and the robot was fully charged and the temperatures back to normal. This should not to be considered the minimum time to wait, it was simply practical in the context of this work. The battery of the NAO used in these experiments is not a new one and is unable to charge to up 100%. If other researchers tries to replicate these results there is a possibility that the NAO they are using does not have the exact same battery capacity as the one used in these experiments, therefore there is risk of slightly varying results.
5.2 Experiments Results

Here we go through the results from each of experiments and analyze the data.

![Graph showing battery charge of robot](image1)

**Figure 5.1:** Battery charge of robot starting from zero to ten minutes.

The results shown in graph 5.1 start from zero minutes which is after reaching the active pose and just as the robot takes the first step. The batteries were charged a minimum of 12 hours yet the charge was unable to reach 100%. The results from this experiment show that the walking gait drew marginally more power than the crawling gait.

![Graph showing degree angle](image2)

**Figure 5.2:** The degree angle at which the robot could lean without tipping over.

The results shown in Figure 5.2 varied for each test run. The robot started from static pose on a zero degrees flat surface. On average the static walking pose could
Figure 5.3: The distance the robot traveled under 60 seconds.

The initial state is the active pose and the timer starts when the robot takes the first step. Figure 5.3 displays the robot’s traveled distance under 60 seconds. The initial state is the active pose and the timer starts when the robot takes the first step.

Figure 5.4: The time from static pose to active pose.

Figure 5.4 demonstrates the time the different behaviors needs to prepare before starting to crawl or walk. The walking behavior needs much less time to go from the static pose to the active pose.
Figure 5.5: Temperature and status for LKneePitch

Figure 5.5 shows the temperature and status in the LKneePitch joint. The lines represent temperature on the left Y-axis and the bars represent stiffness status on the right Y-axis.

Figure 5.6: Temperature and status for RKneePitch.

Figure 5.6 shows the temperature and status in the RKneePitch joint. The lines represent temperature on the left Y-axis and the bars represent stiffness status on the right Y-axis.
Figure 5.7: Temperature and status for LShoulderPitch.

Figure 5.7 shows the temperature and status in the LShoulderPitch joint. The lines represent temperature on the left Y-axis and the bars represent stiffness status on the right Y-axis.

Figure 5.8: Temperature and status for RShoulderPitch.

Figure 5.8 shows the temperature and status in the RShoulderPitch joint. The lines represent temperature on the left Y-axis and the bars represent stiffness status on the right Y-axis.
Figure 5.9: Stiffness and status for LKneePitch.

Figure 5.9 shows the stiffness and status in the LKneePitch joint. The lines represent stiffness on the left Y-axis and the bars represent stiffness status on the right Y-axis.

Figure 5.10: Stiffness and status for RShoulderPitch.

Figure 5.10 shows the temperature and status in the RShoulderPitch joint. The lines represent stiffness on the left Y-axis and the bars represent stiffness status on the right Y-axis.
5.3 Heat map

The heat maps 5.11 and 5.12 shows the difference in temperature between the crawling behavior and the walking behavior. At the initial time of zero minutes into the experiment both behaviors exhibit similar temperatures for all devices.
The heat maps 5.13 and 5.14 show the difference in temperature between the crawling behavior and the walking behavior. Five minutes into the experiments more prominent differences are starting to appear as joints in arms have heated up in the crawling experiment while remaining stable during the walking experiment. The walking behavior shows an increase in temperature in the leg joints while arm joints are stable.
The Figures 5.13 and 5.14 show the difference in temperature between the crawling behavior and the walking behavior. After ten minutes both experiments are completed and there is an even further increase in temperature in the arm joints for the crawling behavior and in the leg joints for the walking behavior. The highest temperature recorded is in the LShoulderPitch during the crawling experiment. The temperatures in the arm joints also increased at a higher rate than the leg joints in the walking behavior.
5.4 Discussion

To reiterate, the purpose of this project was to answer the following questions: 1) “Can a human-like crawling gait be developed for the NAO robot capable of sustained forward momentum?”, 2) “How would the crawling gait perform in terms of speed, stability, profile dimensions, joint temperature and power consumption?”, and 3) “How would the crawling gait compare to the standard walking gait of the NAO robot?” The first of these questions are the most challenging one and for two reasons. The first reason is that the NAO robot is designed with walking locomotion in mind, so there are technical difficulties to make it able to crawl. Even though it is humanoid in shape, it does not have all the same joints humans have, nor the same flexibility and strength. The outer shell is made of hard plastic and does not substitute well to flesh when it comes to properties like friction. The experiments did show that the robot was able to crawl, which is also supported by other projects. The second challenge is the human-like aspect. The human-like crawling pattern was defined as, starting from a kneeling position on all fours, put the left hand forward, then move the left leg back while moving the right leg forward and pushing back with the right arm. The initial step of moving the left arm forward was required in order for the robot to keep its balance.

The second and third research question was about the performance of the crawling gait compared to the in-built walking gait. In terms of speed and temperature the walking gait outperformed the crawling gait. The walking gait was close to twice as fast as the crawling gait, and the time to reach the active pose was only a third that of the crawling behavior. The walking gait also had a longer operational time than the crawling gait. The power consumption was close to same, but the temperature increase was higher when crawling than when walking. This was shown in the experiments as the robot was unable to crawl more than six minutes before temperatures had reached a point where the stiffness in the joints was decreased and the robot was unable to support its movements. The walking experiments was able to run the full course of ten minutes.

To protect the joints from overheating they will reduce the stiffness in joints exceeding certain temperature levels. These levels are the temperature status of each joint and ranges from zero to three. Zero is regular temperature; one means max limit has been reach and stiffness gets reduced; at two the joint is considered very hot and stiffness gets reduced over 30%; and three means the joint is critically hot and stiffness is set to zero. During the ten minute crawl and walk experiments temperature, temperature status and stiffness was recorded. The theory was the it would be possible to see at what rate the stiffness decreased in the joints as the temperature rose. But as seen in Figure 5.9 and 5.10 is quite the opposite. Even though the temperature reached higher levels the stiffness kept moving up and down quite irregularly. Still, the fact was that the robot was unable to crawl passed six minutes due to reduced stiffness but this seems to contradict the measured stiffness.
6 Conclusions and Future Work

The goal of this study was to determine if a human-like crawling gait could be developed for the NAO robot and measure its performance through experiments. In order to complement the results and draw more relevant conclusions the gait was compared to the built-in walking gait. The number of experiments was based on repeating an experiment and comparing the results of each run. After five times the results was stable enough that performing more experiments would not improve the accuracy of the results and would be enough to answer the research questions. It could be argued however that the number of experiments was still not enough so to further improve this method we would suggest increasing the number of experiments. Future work could try to disprove these results by lets say double the times each experiment was run and test if these results still hold up.

Because “human-like” is partly a subjective term, one of the challenges of the project was to define what a human-like crawling gait would be. This was necessary in order to evaluate if the developed gait qualified. The definition a human-like crawling gait was, starting from a kneeling position on all fours, put the left hand forward, then move the left leg back while moving the right leg forward and pushing back with the right arm. With those requirements the gait passed. However one aspect not included was the lack of variations. When a human performs the same action multiple times there are slight variations every time; it is hard if not impossible for us to do the exact same motion again. In contrast this is something that machines can be much better at. Because the focus of this project was to analyze the performance of a human-like crawling gait however, variations in motions was not necessarily a requirement. Varied motions could potentially have an affect the results, but because the variations would be small the effects on performance would likely be small as well. During prolonged or repeated experiments these small differences would eventually be averaged out. During the development of the crawling gait, the physical characteristics of the robot also prohibited alternative motions without the robot falling over. In particular the first step, when going from a passive state on all-fours to a active state, the robot needed to reallocate its center of mass in order to keep its balance before it could take its first step. Humans are more flexible allowing us greater variations in our movements.

The question of how well the crawling gait compared to the built-in walking was answered through several experiments and the results show that the performance of the crawling gait was significantly lower in terms of speed and temperature then the walking gait. One factor inhibiting the speed was how fast the joints moved from one position to the next. When the joints was set to move faster then the limbs did not get enough friction from the floor and the robot was unable to push itself forward. If the surface on the limbs that was in contact with the floor was given a better grip like the rubbery material under the feet, then the speed of the crawling gait could potentially have been increased.

The power consumption remained similar for both the crawling and the walking gait, making temperature the determining factor, and high enough temperature leads to lowered stiffness in the joints. The biggest increase in temperature was in arm joints when the robot was crawling, and in the leg joints when the robot was walking. The robot could however walk for a longer duration because the rate of increasing temperature was higher in the crawling gait. After only six minutes of crawling the stiffness was low enough to prevent it from push itself forward.
**Future Work**  The contribution this project hopes to achieve is better insight into human-like motions for humanoid robots and possible implications of such motions. It can help identify problems and challenges that would need further research. It would have been interesting to test an alternative crawling gait and compare the results. In this project the crawling gait used the robots hands and knees to move forward, but possibly a gait could be developed that used the elbows instead of its hands. If the robot crawled on its elbows instead then the working height would lowered further which would enable to pass under even lower hanging obstacles. It could also be possible to use an alternative walking gait, instead of using the default walking gait developed by Aldebaran.

The actual implementation is very specific to the NAO robot, to improve it researchers could try and convert the specific instructions into algorithms that could calculate how each joint should move and when. Future research could also try and make the gait more power efficient. Joints that are not needed at specific moments can be turned off in order to save power and give the joint time to cool down. This could possibly enable the robot to move for a longer period of time. Researchers could also look at how the robot could get from a standing position to the passive crawling pose and back again.
References


**A Appendix 1**

In the appendix you can put details that does not fit into the main report. Examples are source code, long tables with raw data and questionnaires.

**A.1 Static All-fours**

```python
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        # awarenessProxy = ALProxy('ALBasicAwareness')
        # awarenessProxy.stopAwareness()
        # put initialization code here
        pass

    def onUnload(self):
        # put clean-up code here
        pass

    def onInput_onStart(self):
        motionProxy = ALProxy("ALMotion")

        # Position head
        names = ["HeadYaw", "HeadPitch"]
        angles = [0.0, -0.6720]
        fractionMaxSpeed = 0.2
        motionProxy.setAngles(
            names, angles, fractionMaxSpeed)

        motionProxy.closeHand("LHand")
        motionProxy.closeHand("RHand")

        names = ["LWristYaw", "RWristYaw"]
        angles = [0.0, 0.0]
        fractionMaxSpeed = 0.2
        motionProxy.setAngles(
            names, angles, fractionMaxSpeed)

        names = ["LElbowYaw", "RElbowYaw"]
        angles = [-1.6, 1.6]
        fractionMaxSpeed = 0.2
        motionProxy.setAngles(
            names, angles, fractionMaxSpeed)

        names = ["LElbowRoll", "RElbowRoll"]
```

A
# angles = [−0.0349, 0.0349]
angles = [−0.05, 0.05]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LShoulderPitch", "RShoulderPitch"]
# angles = [0.5, 0.5]
# angles = [0.4, 0.4]
# angles = [0.2, 0.2]
angles = [0.6, 0.6]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LShoulderRoll", "RShoulderRoll"]
angles = [−0.06, 0.06]
# angles = [−0.2, 0.2]
# angles = [−0.3, 0.3]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipYawPitch", "RHipYawPitch"]
angles = [0.05, 0.05]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipRoll", "RHipRoll"]
angles = [0.0, 0.0]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipPitch", "RHipPitch"]
# angles = [−1.26, −1.26]
angles = [−0.7, −0.7]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LKneePitch", "RKneePitch"]
angles = [1.15, 1.15]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LAnklePitch", "RAnklePitch"]
angles = [0.932057, 0.932057]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

self.onStopped()

def onInput_onStop(self):
    self.onUnload()
    self.onStopped()

A.2 Active-All-fours

import time

class MyClass(GeneratedClass):
def __init__(self):
    GeneratedClass.__init__(self)

def onLoad(self):
    # put initialization code here
    pass

def onUnload(self):
    # put clean-up code here
    pass

def onInput_onStart(self):
    motionProxy = ALProxy("ALMotion")
    # time.sleep(3)

    self.crawlPose(motionProxy)
    self.onStopped()

def onInput_onStop(self):
    self.onUnload()
    self.onStopped()

def crawlPose(self, motionProxy):
    # Shift balance to left arm
    names = ["LShoulderRoll", "RShoulderRoll"]
    angles = [-0.3142, -0.26]
    fractionMaxSpeed = 0.1
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)

    names = ["LShoulderRoll", "RShoulderRoll"]
    angles = [-0.06, 0.06]
    fractionMaxSpeed = 0.1
# Extend knees
names = ["LHipYawPitch"]
angles = [−0.5]
fractionMaxSpeed = 0.1
motionProxy.changeAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipPitch", "RHipPitch"]
angles = [−0.7, −0.7]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LKneePitch", "RKneePitch"]
angles = [1.55, 1.55]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

# Lift right arm
names = ["RShoulderPitch", "RElbowRoll", 
    "RShoulderRoll"]
angles = [1.5, 1.5446, 0.06]
fractionMaxSpeed = 0.1
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(1)

names = ["RShoulderPitch"]
angles = [0.2]
fractionMaxSpeed = 0.1
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(1.5)

names = ["RElbowRoll"]
angles = [0.0349]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(1)

# Move left arm back by lifting it
names = ["LShoulderPitch", "LElbowRoll"]
angles = [1.0, −1.0]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

names = ["LShoulderRoll", "LShoulderPitch"]
angles = [−0.06, 1.5]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

names = ["LElbowRoll"]
angles = [−0.0349]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

names = ["LShoulderPitch"]
angles = [0.9]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

# Reset hips
names = ["LHipYawPitch", "RHipYawPitch"]
angles = [0.05, 0.05]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipRoll", "RHipRoll"]
angles = [0.0, 0.0]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LHipPitch", "RHipPitch"]
angles = [−0.7, −0.7]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LKneePitch", "RKneePitch"]
angles = [1.15, 1.15]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ['LAnklePitch', 'RAnklePitch']
angles = [0.932057, 0.932057]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

# Forward left leg
names = ['LKneePitch', 'LHipPitch']
angles = [1.92, -1.47]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

# Back right leg
names = ['RHipPitch', 'RKneePitch']
angles = [-0.39, 0.85]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

A.3 Crawling Gait

import time
class MyClass(GeneratedClass):

crawl = True

def __init__(self):
    GeneratedClass.__init__(self)

def onLoad(self):
    # put initialization code here
    pass

def onUnload(self):
    # put clean-up code here
    pass

def onInput.onStart(self):
    motionProxy = ALProxy("ALMotion")

    while self.crawl:
        self.crawlGait(motionProxy)
def onInput_onStop(self):
    self.crawl = False
    self.onUnload()
    self.onStopped()

def crawlGait(self, motionProxy):
    ### Left side
    # Lift left arm
    names = ["LShoulderPitch", "LElbowRoll"]
    angles = [1.5, -1.5446]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)
    time.sleep(1)

    # Start moving left arm
    names = ["LShoulderPitch", "LElbowRoll"]
    angles = [0.2, -0.0349]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)
    time.sleep(0.5)

    # Back right arm and forward left arm
    names = ["RShoulderPitch"]
    angles = [0.9]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)

    names = ["RElbowRoll"]
    angles = [0.5]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)

    # Back left leg and forward right leg
    names = ["LHipPitch", "RHipPitch"]
    angles = [-0.39, -1.47]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)

    names = ["LKneePitch", "RKneePitch"]
    angles = [0.85, 1.92]
    fractionMaxSpeed = 0.2
    motionProxy.setAngles(
        names, angles, fractionMaxSpeed)
    time.sleep(1)
names = ["RElbowRoll", "LElbowRoll"]
angles = [0.0349, -0.0349]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

### Right side
# Lift left arm
names = ["RShoulderPitch", "RElbowRoll"]
angles = [1.5, 1.5446]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(1)

# Start moving left arm
names = ["RShoulderPitch", "RElbowRoll"]
angles = [0.2, 0.0349]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)
time.sleep(0.5)

# Back right arm and forward left arm
names = ["LShoulderPitch"]
angles = [0.9]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["LElbowRoll"]
angles = [-0.5]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

# Back left leg and forward right leg
names = ["RHipPitch", "LHipPitch"]
angles = [-0.39, -1.47]
fractionMaxSpeed = 0.2
motionProxy.setAngles(
    names, angles, fractionMaxSpeed)

names = ["RKneePitch", "LKneePitch"]
angles = [0.85, 1.92]
fractionMaxSpeed = 0.2
A.4 Readings

class MyClass(GeneratedClass):

count = 1

def __init__(self):
    GeneratedClass.__init__(self)

def onLoad(self):
    # put initialization code here
    pass

def onUnload(self):
    # put clean-up code here
    pass

def onInput_onStart(self):
    batteryProxy = ALProxy("ALBattery")
    memoryProxy = ALProxy("ALMemory")

    devicesHead = ["HeadPitch", "HeadYaw"]
    devicesRArm = ["RShoulderRoll", "RShoulderPitch", "RElbowYaw", "RElbowRoll", "RWriskYaw", "RHand"]
    devicesRLeg = ["RHipPitch", "RHipRoll", "RKneePitch", "RAnklePitch", "RAnkleRoll"]

    self.logger.info(str(self.count) + " Reading--------")
    self.count = self.count + 1

    battery = batteryProxy.getBatteryCharge()
self.logger.info("Battery: " + str(battery) + "/")

self.printDevicesTemperatureAndStatus(memoryProxy, devicesHead)
sel.printDevicesTemperatureAndStatus(memoryProxy, devicesRArm)
sel.printDevicesTemperatureAndStatus(memoryProxy, devicesLArm)
sel.printDevicesTemperatureAndStatus(memoryProxy, devicesRLeg)
sel.printDevicesTemperatureAndStatus(memoryProxy, devicesLLeg)

self.logger.info("")
#self.onStopped()

def onInput_onStop(self):
    self.onUnload()
    self.onStopped()

def printDevicesTemperatureAndStatus(self, memoryProxy, devices):
    for device in devices:
        deviceString = "Device/SubDeviceList/" + device + "/Temperature/Sensor/Value"
        temperature = memoryProxy.getData(deviceString)
        self.logger.info(device + " Temperature: " + str(temperature) + "C")
        deviceString = "Device/SubDeviceList/" + device + "/Temperature/Sensor/Status"
        status = memoryProxy.getData(deviceString)
        self.logger.info(device + " Status: " + str(status))
        deviceString = "Device/SubDeviceList/" + device + "/Hardness/Actuator/Value"
        stiffness = memoryProxy.getData(deviceString)
        self.logger.info(device + " Stiffness: " + str(stiffness) + "/")
A.5 Measurements

Figure 1.17: Temperature and status for HeadPitch.
Figure 1.18: Temperature and status for HeadYaw.

Figure 1.19: Temperature and status for LAnklePitch.
Figure 1.20: Temperature and status for LAnkleRoll.

Figure 1.21: Temperature and status for LElbowYaw.
Figure 1.22: Temperature and status for LHand.

Figure 1.23: Temperature and status for LElbowRoll.
Figure 1.24: Temperature and status for LHipPitch.

Figure 1.25: Temperature and status for LHipRoll.
Figure 1.26: Temperature and status for LHipYawPitch.

Figure 1.27: Temperature and status for LShoulderRoll.
Figure 1.28: Temperature and status for LWristYaw.

Figure 1.29: Temperature and status for RAnklePitch.
Figure 1.30: Temperature and status for RAnklePitch.

Figure 1.31: Temperature and status for RELbowRoll.
Figure 1.32: Temperature and status for RElbowYaw.

Figure 1.33: Temperature and status for RHand.
Figure 1.34: Temperature and status for RHipPitch.

Figure 1.35: Temperature and status for RHipRoll.
Figure 1.36: Temperature and status for RShoulderRoll.

Figure 1.37: Temperature and status for RWristYaw.
Figure 1.38: Stiffness and status for HeadPitch.