DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Closed Loop Walking Controller for a Biomimetic, Robotic Mouse

Adrian Schultz

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Closed Loop Walking Controller for a Biomimetic, Robotic Mouse

Entwurf eines Gangreglers für eine biomimetische, robotische Maus

Author:	Adrian Schultz
Supervisor:	Prof. Dr.-Ing. habil. Alois Christian Knoll
Advisors:	Dr. Ph.D. Alexander Lenz
	Peer Lucas, M.Sc.
Submission Date:	15. September 2020

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15. September 2020 Adrian Schultz

Abstract

Natural movements of animals and humans are undeniably worth copying. Since ancient times with the legend of Daedalus and Icarus trying to fly like a bird, researchers are trying to mimic nature as good as possible. For the last decades, nature-like locomotion is the main focus of modern research in the area of biomimetic robots. In this thesis, the neurorobotic mouse (NerMo) - a biomimetic image of a rattus norvegicus also known as the common rat - is used as the object of research. Improving NerMo's locomotion can be stated as the overall goal of this work. Therefore, sensors at the knees and feet of the robot are included in the mechanical design and in hardware and software to get feedback of the actual state of NerMo. Using the foot pressure and knee angle sensors, the current open loop control is analysed and deemed good for simple tasks as walking straight on even ground. Regarding the locomotion on more natural and difficult terrain e.g. walking the stairs or on gravel, the current setup fails completely. To improve the capabilities of NerMo, the height the feet are lifted, is increased and the initialisation of NerMo with its legs and tendons is simplified and improved with a closed loop control.

Contents

1. Introduction

Movements of animals in nature are indisputably worth imitating as they were developed and improved over thousands and thousands of years. Several experiments and projects during the past decades and even way before, which try to mimic or be inspired by nature as good as possible, were all based on this idea. But mostly, only some small aspects of the biology were integrated in projects like the structure of the wings of a bird in the first aviation attempts, which can be traced back to the Greek legend of Daedalus and Icarus. When not only one, but both aspects are considered - nature-like appearance and movement -, the field of research becomes smaller. The background for the endeavour to combine both can be summarized like that:

> "Replicating not only one part of a biological system, but trying to mimic complete animals, allows for more animal focused applications. The created robots can be introduced to environments inhabited by the biological counterpart and interact with them." [1]

Based upon this idea, combining the biomimetic appearance of a given rodent robot with versatile, controlled and nature-like locomotion is the main motivation of this thesis.

1.1. Aim of the Work

Improving the locomotion of an already existing biomimetic rodent robot can be seen as the primary goal of this work. Therefore, the neurorobotic mouse (NerMo) developed at the Chair for Robotics, Artificial Intelligence and Real-time Systems at Technical University of Munich (TUM) within the human brain project (HBP) will be used. As a first step the sensors have to be integrated into the hardware and software infrastructure and afterwards tested in detail, evaluated and improved if necessary. Therefore, the four magnetic knee angle sensors to get feedback about the bending of the elastic legs and the thin-film foot pressure sensors were included. Those are mounted at the bottom of each leg to allow precise balancing and actuation of NerMo. The reasoning behind this design was, that it should be comparable to a human or animal walking on an uneven surface with just the pressure feedback from the feet without actually looking at the ground, e.g. taking the stairs or walking along a forest path while just looking in front. As a second step the prior integrated sensors are used for a closed loop control to ease and refine the initialisation of NerMo's legs and tendons as well as analysing the current state of the open loop control locomotion. Additionally, suggestions for possible improvements regarding the overall control of NerMo are made.

1.2. Definitions

Learning from nature can be seen as a more and more popular area of research within the last few years. This can be observed in several scientific disciplines like biology, chemistry, physics, but also robotics. Reading the literature, one will note three terms being used frequently: biomimetics (biomimicry), bioinspiration and bionics. But as those terms were often used misleading or even mixed up, the actual definitions can be summarized as follows: (Reference for the following definitions [2] [3] [4] [5])

- **Bionics**
  Bionics can be defined as a fusion of biology and electronics. By reading literature one can notice that bionics is by far the most common used of the three discussed terms, although it is mostly only used in robotics. Bionics was first introduced by Jack Steele, who was working for the US Air Force, in 1960. It is defined by the Webster online dictionary as

  > "a science concerned with the application of data about the functioning of biological systems to the solution of engineering problems"

  So both of the following terms to be defined - bioinspriation and biomimetics - can be included in the research field of bionics, which itself can be seen as a subcategory of robotics using nature or biology as role model.

- **Bioinspiration**
  According to the Oxford Dictionary, the term bioinspiration can be described as follows:

  > "Inspired by or developed using living systems as a model or guide; biologically inspired."

  According to the same entry, the first use can be dated back to the 1990s in Science News, so the term is the most recent of the three here defined. Although, this term is seen less strict than biomimetics, as bioinspired systems or projects have only to be inspired by nature, but not harshly mimic it. A famous example for bioinspired robots is Boston Dynamics' quadruped Spot (see Figure 1.1).

Figure 1.1.: Boston Dynamics' Spot (Figure originally from: [6])

- **Biomimetics**
  Biomimetics is often misused for actually bioinspired systems in literature. Otto Schmitt, who was researching the electrical behavior of a nerve, first used the term biomimetics in the 1950s and defined biomimetics in short words as "the mimicry of life, or biology". So, to call a system biomimetic it has to truly try to mimic nature as good as possible in its appearance or behavior, which is the goal with NerMo.

1.3. Neurorobotic Mouse (NerMo v4.1)

In this thesis, the neurorobotic mouse in its version 4.1 (short NerMo v4.1) is used. This mouse was mainly developed by Peer Lucas at TUM as part of the Human Brain Project and its goal was to be biomimetic, but also low cost and simple to use.

The development started in 2017 with the master's thesis "Design, Construction and Validation of a Life-Size Robot Model for Biomimetic Locomotion in Small Mammals" [1] (see Figure 1.2), which was supported by the design of a biomimetic leg [7] and merged later with an actuated vertebrae [8]. The main objective for this first version was the biomimetic leg design and the appearance, which should be as close as possible to a common rat (rattus norvegicus). [9]

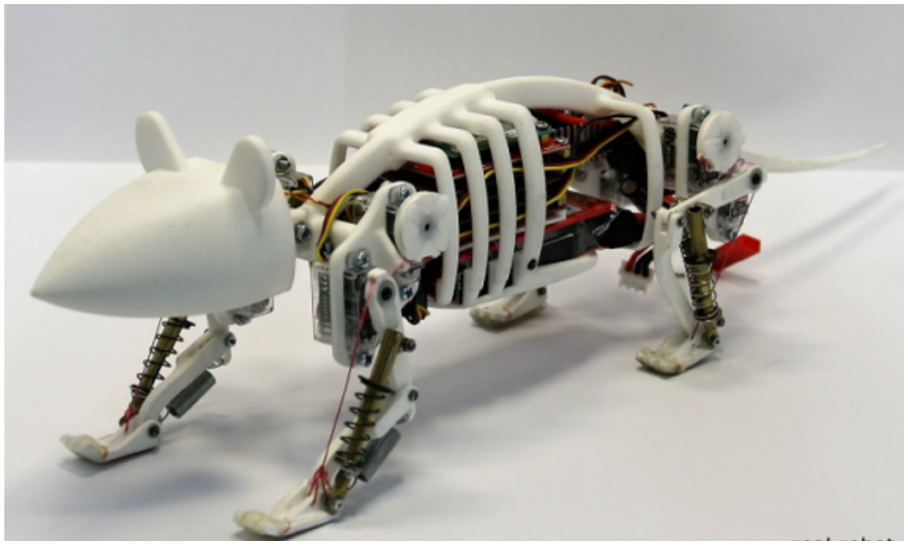

Figure 1.2.: NerMo Version 1 (Figure originally from: [1])

The proposed leg design can be described as quite complicated and consisted of 26 parts. The spine integrated to the robot mouse was fixed. These two deficits were removed in version 2.0, which included a flexible spine and tail and 2.1, which was using a simpler structure for the legs. The improved spine was realised by elastic joints, which are

working similarly to real vertebrae and were actuated by 2 tendons as seen in Figure 1.3. For the leg design a different 3D printing material was used. This material allowed an elastic joint design that reduced the parts needed to one plus a tendon to lift the foot by moving the knee. [10]

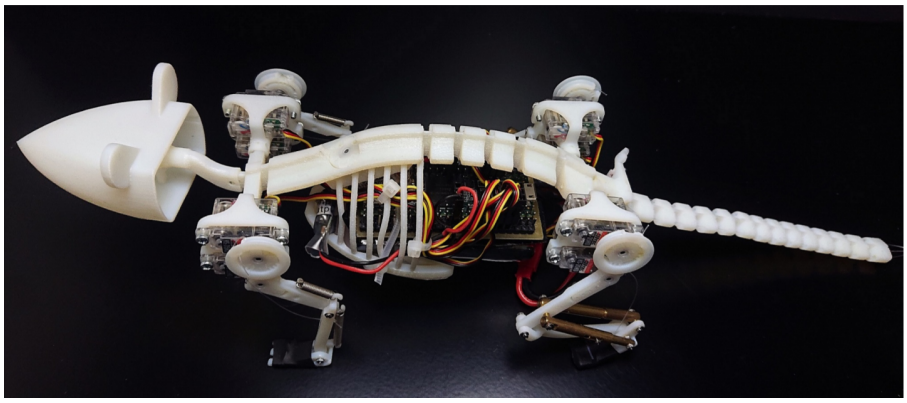

Figure 1.3.: NerMo Version 2 (Figure originally from: [10])

In version 3 the research was focused on the further improvement of the spine, but the changes were discarded due to undefined movements, which prevent controlled locomotion of the robotic mouse. [10]

Version 4 is a complete re-design including the change from Intel's Edison Board to a Raspberry Pi as compute unit as well as own board designs for the communication, the servos, and the leg sensors. Furthermore, the undefined spine movement of version 3 was eliminated by fixed limits of the vertebrae and now allows lumbar and lateral movement in a controllable way. Another major change of this version is the improved head (see Figure 1.4), that includes a large diversity of sensors like two cameras as eyes, a bumper and a touch sensor. Essential for this thesis, sensors for the knee angles, foot pressure, servo position and an IMU were integrated. [10]

The most recent iteration of NerMo is version 4.1. Since this version, a precise digital twin is imported to the Neurorobotics Platform (NRP) and some smaller changes regarding the 3D printed elements were integrated. In summary, the latest robot has 13 degrees of freedom (DOF) – 2 for each leg, 2 for the head, 2 for the spine and one for the tail – and 26 sensors – 13 hall servo position sensors, 4 thin film foot pressure sensors, 4 knee angle sensors, 2 cameras, 1 touch sensor, one button (nose bumper) and a 9-axis IMU. [11]

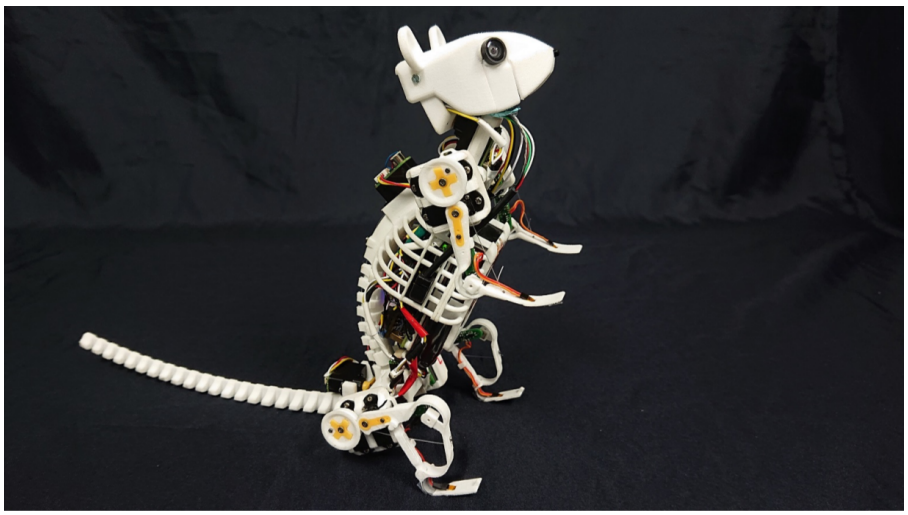

Figure 1.4.: NerMo Version 4 (Figure originally from: [10])

1.4. Biomimetic Robots in Research

Biomimetic robots can be developed with the focus on several topics. The here discussed topics education and human interaction, sensing and actuating and quadrupeds are chosen due to their widespread interest and connection to NerMo.

Education and Human Interaction

Education and natural interaction with humans - for example in case of robot assisted therapy - can be an aim for biomimetic robots. Therefore, the behavior is seen as the main focus for researchers in this certain area. Probably the best known robot of this kind is MiRo (see Figure 1.5). It is designed to be "biomimetic in aesthetics, morphology, behaviour, and control architecture." [12] As this robot was developed to be used by a wide range of researchers with version MiRo-B (Beta Developer Kit) the biomimetic brain-based control system was integrated to make the human robot interaction as natural as possible. [13]

Sensing and Actuating

Besides the above mentioned aims, biomimetics is used to develop new methods for nature-like sensors and actuators. Knowledge from robotics and biology, respectively physics, has to be combined to achieve progress here. One well known example is the active tactile whiskered robot developed by the University of Bristol. An array of 10 whiskers is used to localise touch input via deflections like the vibrissae of a rodent. [14] [15]

Figure 1.5.: Biomimetic edutainment robot MiRo (Figure originally from [12])

As an example for biomimetic actuation, mimicking the behavior of the gecko's feet is researched in another more physics-related project. The main perspective and goal of the research group can be summarized like that: "Gecko has excellent adhesive capability. This leads people to biomimicing the dry gecko adhesive, which would be used in wall-climb robots in the outer space." [16]

Quadrupeds

Although MiRo with its biomimetic behavior and interaction, the whisker touch input and the adhesive gecko-like foot are somehow related and could be used as inspiration for NerMo, biomimetic quadrupeds can be compared best to the rodent-like robot used for this work. For that specialization, the improvement of robot locomotion can be seen as the main focus, but some are also focusing on the closest possible replica of nature - behavior and appearance - to research interaction with real animals as a greater scope.

Typical for the first point is for example the development of "a quadruped robot focusing on different ways to achieve connectivity in the pectoral and pelvic girdles. The experiments show that a flexible shoulder joint allows the robot to redirect the motion of the body from down to up smoothly upon landing on the ground from an elevated place." [17] As another example Cheetah (see Figure 1.6), developed by the MIT biomimetic robotics lab is designed to use biomimetics in its favour, but not completely

mimic nature. [2] "[This] robot incorporates a flexible spine — known to play a critical role in generating the stride of many mammals." [2] [18]

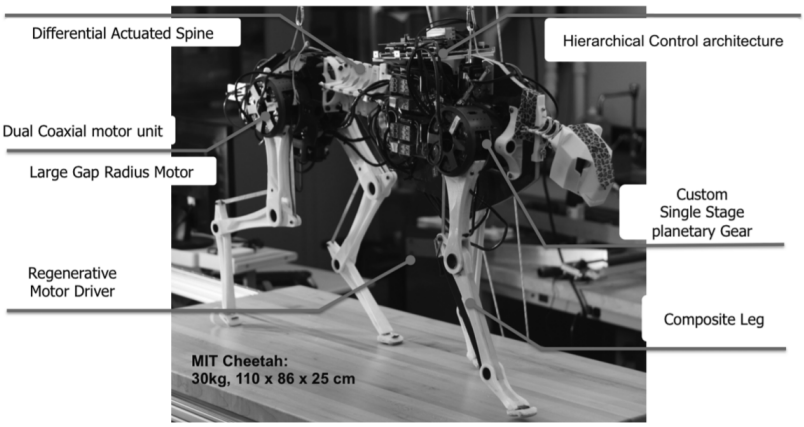

Figure 1.6.: MIT's Cheetah (Figure originally from: [19])

Unlike focusing on locomotion optimization with mostly larger robots, appearance and behavior has to be the main goal for interaction studies. Therefore, Waseda-Mouse (1998) and Waseda Rat No. 2 (2009, see Figure 1.7) were developed. [20] Even basic interactions like: "The movement of WM-2 is recognized by rats" and "Movement of WM-2 influences the behavior of rats, and acts as discriminative stimulations in learning." (both [21]) were successful in that early stage, but for further research and more complex interactions, the robot would have to "behave more like a rat" [21], which can be seen as the overall goal for NerMo.

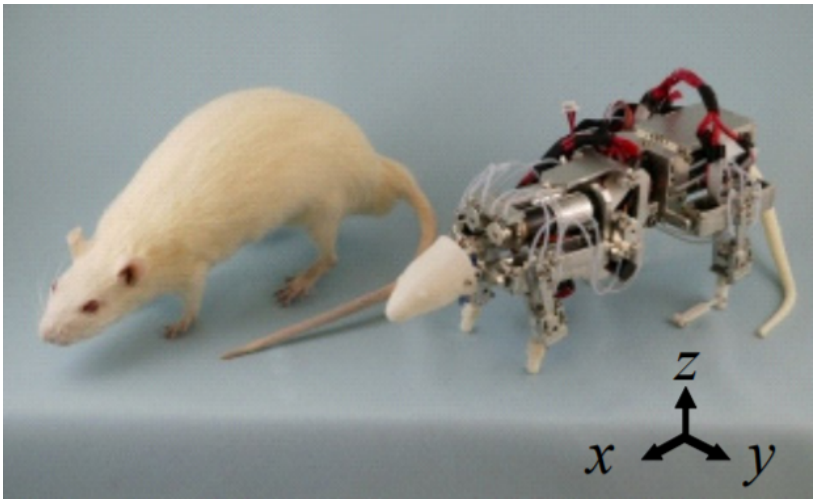

Figure 1.7.: Waseda Rat No.2 compared to a real rat (Figure originally from: [20])

Gait Control of Quadruped Robots

As examples for gait control of quadruped robots, MIT's Cheetah 2 and Mini Cheetah can be mentioned, although they are not strictly biomimetic unlike their predecessor. Both try to experiment with different sensor inputs and optimize gait control. The control of Cheetah 2 with only using the foot impact feedback as input seemed to be unstable, but showed good results with an additional gait pattern stabilizer. [22] The focus of Mini Cheetah is set even more on versatile and precise controlling. For that, a convex model predictive control is used, which "determines ground reaction forces for the stance feet by solving an MPC problem using a highly simplified model of the robot dynamics, and a prediction horizon spanning one gait cycle." [23]

1.5. Focus Shift from the Actual Goal

"A Closed Loop Walking Controller for a Biomimetic, Robotic Mouse" was set to be the initial goal, which could not be reached. This is caused by the small errors and bugs that often appear, when working with hardware. In this case NerMo had to be improved first by changing sensors, voltage regulators, resistors, I2C protocol, software on 2 different ARM micro-controller and in C++, design shoes, etc. Furthermore NerMo had to be repaired a lot as the durability of small biomimetic robots is not anywhere close to nature yet - e.g. a human's finger working a hole life without a lot of maintenance needed, but also having a small form factor. For the duration of this thesis approximately 500 soldering points, around 20 tendons, 16 electronic components, 6 circuit boards - shipped from Kungliga Tekniska Högskolan (KTH) in Stockholm - and even a complete set of feet had to be replaced. Those bugs were not known, when starting the work and therefore shifted the timetable by a lot as all these changes had to be done to have an as stable as possible foundation for later controlling the robot.

1.6. Structure of this Thesis

This master's thesis can be structured into two main parts. The first part will cover the sensor integration as a basis to enable controlled locomotion and the second the initial position control and open loop control analysis to establish a baseline for further research. A new shoe design for NerMo to enable valid foot pressure sensor feedback, I2C basics and timings to include the knee angle and foot pressure sensors to NerMo and a concept for inner-NerMo communication on single wire UART are presented within the sensor integration chapter. After that the currently used open loop control for locomotion is analysed in the next chapter. The focus is set on a detailed review including advantages and disadvantages as well as suggestions for improving the locomotion with a closed loop walking controller. Lastly, a closed loop control is described, which allows a simple, fast and precise initial setup of NerMo's legs with its joints and tendons to allow a better walking performance, although using open loop control for the actual locomotion.

2. Sensor Integration

The integration of sensors can be seen as the basis of a closed loop controller and therefore as the first of the main chapters in this thesis. This procedure was started by Peer Lucas in [10] in collaboration with Prof. Jörg Conradt at KTH in Stockholm, but yet had included problems in hardware and software. These deficiencies have now led to major changes regarding thin-film foot pressure sensors, the I2C implementation and setup as well as the inner-NerMo Communication itself.

2.1. Foot Pressure Sensors

The only point of contact to the ground is performed by the feet. Due to that, sensors at the feet can be used to monitor misbalancing of NerMo or detect uneven underground. Therefore, foot pressure sensors are integrated in the robot's feedback loop.

2.1.1. Current Foot Setup

The thin-film pressure sensor is connected to the sensor board and attached to the bottom of the feet. Problematic with this type of sensors is that the pressure has to transferred selective on a certain area, although a rubber has to be added on top to enable the needed grip for the feet of NerMo. The impact of non selective pressure can be seen in Table 2.1. In the current version a setup with a rubber glued on top of the sensor is used, which leads to a not usable behavior as the mouse is too light (by a factor of ~ 8) to trigger the sensor. In order to figure out the best possible solution for that specific problem, a series of tests was carried out.

Table 2.1.: Foot pressure sensor accuracy

Setup	Minimum Pressure
Mouse Weight (per Leg)	~ 60 g
Non Selective with Rubber	≥ 500 g
Non Selective without Rubber	≥ 200 g
Selective without Rubber	≥ 20 g

2.1.2. Development Process and Test Series

To achieve a fast and simple solution one could use glue to either attach a washer underneath the rubber - super glue or flexible glue were used and tested for that scenario - or a drop of silicone to allow selective pressure on the designated area. These approaches were tested regarding their accuracy and their durability. The results in the initialisation position - pressure nearly the same for all four legs ($\sim$ 60 g) - can be seen in Table 2.2, which are either accurate, but not durable - e.g. falling apart - or vice versa.

As a more reliable and accurate solution 3D printed shoes were created. The goal was to develop a design, which maintains the biomimetic appearance of NerMo and is nevertheless simple to assemble. To achieve those requirements, the initial structure of the legs is included as much as possible.

For the hind leg shoe (see Figure 2.1) the preparation to assemble 3D printed toes is used as bearing as seen in detail in Figure 2.3. To allow selective pressure on the sensor a small cylinder is integrated at the specific position, where the sensor is attached to the foot. The stopper in the back of the shoe can be seen as range limiter to ensure good accuracy with only a small input delay.

For the front leg shoe (see Figure 2.2) a small notch with 3,5 mm diameter is filed into the "toes" to attach the front part, which can be secured with a semicircle glued on top afterwards. The back and the cylinder are designed to be similar to the shoe for the hind leg. The front leg shoe can be seen in detail in Figure 2.4 and the results for both shoes in the initialisation position - pressure nearly the same for all four legs ($\sim$ 60 g) - in Table 2.2, which surpass the glue versions by far regarding both, durability and accuracy.

Table 2.2.: Test results foot pressure sensor

Setup	ADC Reading	Durability
Reading without Pressure	$\sim$ 3400	
Super Glue	$\sim$ 2700	Good
Silicone	$\sim$ 1500	Medium - Bad
Flexible Glue	$\sim$ 700	Bad
Front Leg Shoe	700 - 1100	Good
Hind Leg Shoe	500 - 700	Very Good

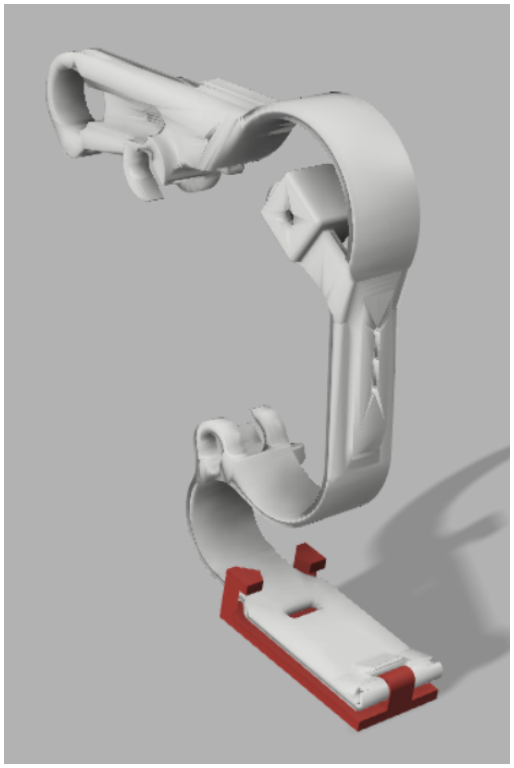

Figure 2.1.: Shoe for hind-legs

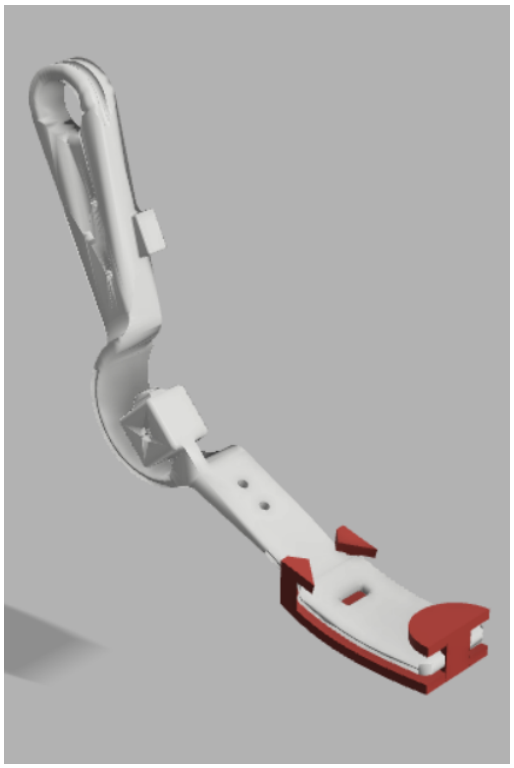

Figure 2.2.: Shoe for front-legs

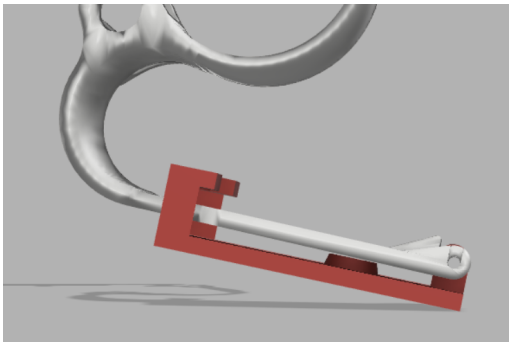

Figure 2.3.: Sideview hind-leg shoe

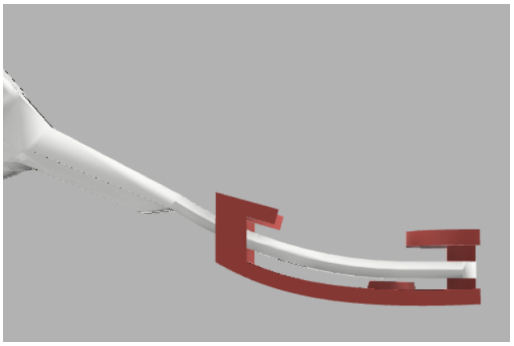

Figure 2.4.: Sideview front-leg shoe

2.1.3. Evaluation of the Shoe Design

For evaluating the shoe design the front leg shoes and hind leg shoes were tested separately. The same pressure was applied to both sides - right and left foot - and measured for both legs simultaneously. Afterwards the measured weight was divided by two to get the resulting pressure for each foot individually. To get a better and realistic distribution each pressure level was measured about 20 times with re-placing NerMo at least 5 times. The pressure was applied as good as possible in a middle foot position - so neither too much on the heel nor on the toes - and as perpendicular to the ground as possible.

Hind Leg Setup

Looking at the graph in Figure 2.5 one can see that the behavior of the hindleg setup matches the expectations from the sensor's datasheet [24] quite well as it looks like a similar hyperbolic function in the area of interest. Furthermore, clearly separated value ranges for the ADC readings using $\sim$ 10 g steps in the range of 0 g - 80 g of pressure on each hindleg can be observed, which is a good basis for ground contact detection and balancing NerMo in the initialisation phase.

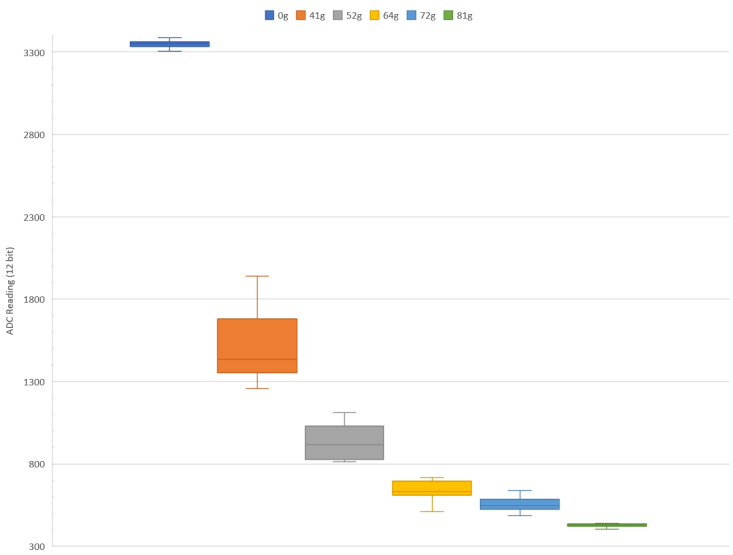

Figure 2.5.: ADC readings for the hind leg setup

Although the value ranges for low pressure are by far wider than the ones for higher pressure, this can be explained by two factors and is normal with this certain setup. The first reason is the resistor choice (27 kΩ) used as voltage divider, which corresponds to the resistance of the foot pressure sensor in that certain area (10 - 40 g) and therefore

leads to a very fine gradation, respectively higher range. Secondly, this area is also the range where the foot pressure sensor changes its resistance the most and hence deviates the most, which amplifies this effect.

Front Leg Setup

Comparing the graph in Figure 2.6 for the front leg setup with the above for the hind legs and the sensor's datasheet, this setup can be used as well for basic tasks like ground contact detection, but even balancing NerMo in the initialisation phase could be challenging as the input is not always as stable and precise as the hind leg design. Therefore, it may cause problems when using it for a closed loop walking controller as sensor input. This unstable behavior is mostly caused by the sometimes stuck front "bearing", which can only be improved by a complete re-design of the front leg.

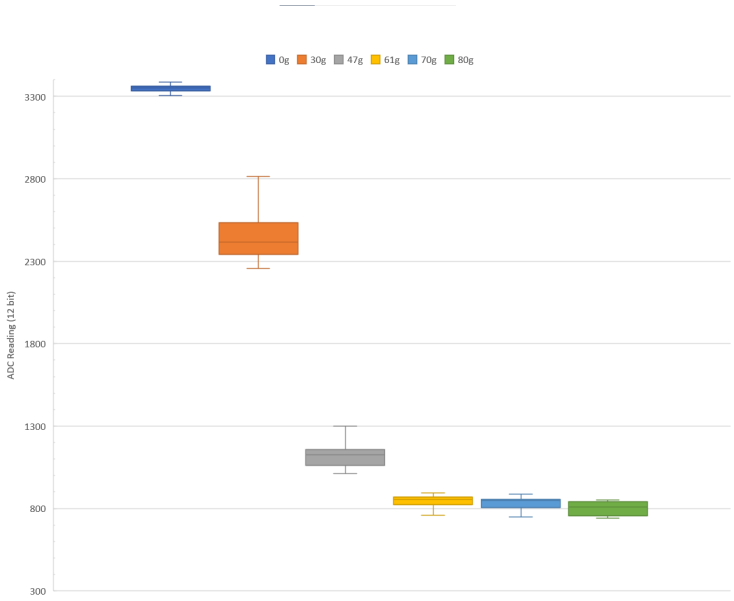

Figure 2.6.: ADC readings for the front leg setup

2.2. I2C

First, the I2C basics and most common patterns are covered in this section. Those are followed by the use of I2C in NerMo and completed with a discussion about correct pull-up resistor sizing for a certain setup to achieve the desired timings.

I2C Basics

Inter-Integrated Circuit (I2C) was developed by Philips Semiconductors in the early 1980s to allow simple communication. This communication protocol was designed to only need two wires - Serial Clock Line (SCL) and Serial Data Line (SDA) - connected besides VCC and GND for a bidirectional setup and is now widely used in embedded systems. The basic setup is consisting of at least one master and one slave as seen in Figure 2.7, but multi-master configurations as well as multi-slave configurations are possible up to a total bus capacitance of 400 pF.

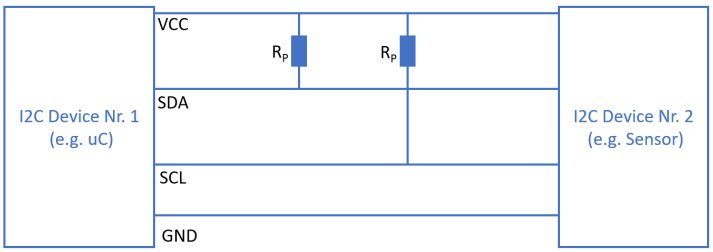

Figure 2.7.: Basic I2C example setup with a μC, a sensor and two pull-up resistors

These devices can be addressed with either 7-bit or 10-bit addressing, where the first 4 bits are uniquely set to identify the vendor and the last bits are device specific [25]. The clock signal to synchronize the data transfer is generated by the master and is dependent on the chosen mode (see Table 2.5), whereas SDA can be driven by any device connected to the I2C bus. [25]

Table 2.3.: I2C modes

Mode	Maximum f_{SCL}	Bidirectional
Standard Mode	100 kHz	yes
Fast Mode	400 kHz	yes
Fast Mode Plus	1 MHz	yes
High-Speed Mode	3.4 MHz	yes
Ultra Fast Mode	5 MHz	no

I2C Patterns

In I2C, there are several basic commands and patterns, which have to be implemented and understood in detail to enable an effective device to device communication. The most common commands to allow an error free - error handling can be researched in [25] - I2C communication are:
(Reference for following listing: [25])

- **Start Condition**:
  The start condition is used to start communication and is most often sent by the master to start a request. Therefor, SDA is set to have a falling edge, while SCL is HIGH for that period (see Figure 2.8).
- **Stop Condition**:
  The stop condition is used to stop communication and is most often sent by the master to end a exchange with a slave. So, no more data should be sent afterwards by a slave, although some registers might not have been read completely. Therefor, SDA is set to have a rising edge, while SCL is set to HIGH for that period (see Figure 2.8).
- **ACK**:
  The acknowledgement is sent by the master or slave to confirm the last 8bit received. This is dependent on whether a configuration gets written by the master (ACKs are only sent by the slave) or a sensor reading gets requested (first ACK is sent by the slave to acknowledge the request and the following ACKs by the master to acknowledge the transferred data). SDA is set to LOW by the receiver, while SCL is set to HIGH (see Figure 2.8).
- **Data Setup Time**:
  This delay is used to have stable SDA data during an upcoming rising clock edge. Therefor, SDA has to be set a bit prior to ensure valid data reading (see Figure 2.9). It is usually set to $\sim$100 ns in fast mode.
- **Data Hold Time**:
  The Data Hold Time is used to ensure that the clock signal is LOW, when the data line gets changed. This delay is usually set to 0 as the falling edges of the clock signal are in the range of a few nanoseconds (see Figure 2.9).
- **Rising edge of SCL and SDA**:
  This time has to be smaller than 1000 ns in Standard Mode, 300 ns in Fast Mode and 120 ns in Fast Mode Plus and can be adjusted with a correctly sized pull up resistor (for calculations see subsection 2.2.2 and for graphic representation see Figure 2.9).

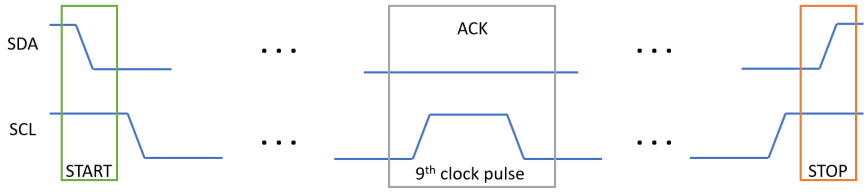

Figure 2.8.: START, ACK and STOP condition for I2C communication

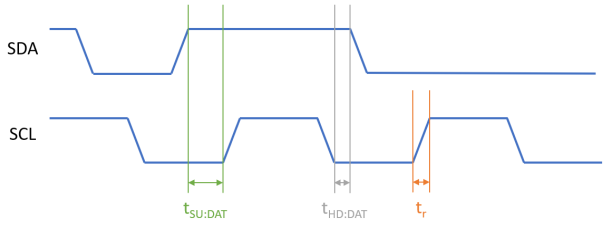

Figure 2.9.: Important timings for I2C communication

Communication Example

The I2C communication is usually started with the START condition sent by the master. After that, the 7-bit address of the slave (e.g. sensor, whose address is specified by the vendor and can be found in the data-sheet) has to be transmitted. These 7 bits have to be followed by the read (1) or write (0) bit. This byte then has to be acknowledged by the slave. Depending on the read or write bit set the slave or master has to start sending data byte by byte each acknowledged by the other. To end the communication a STOP condition has to be sent by the master. This communication example can be seen in Figure 2.10.

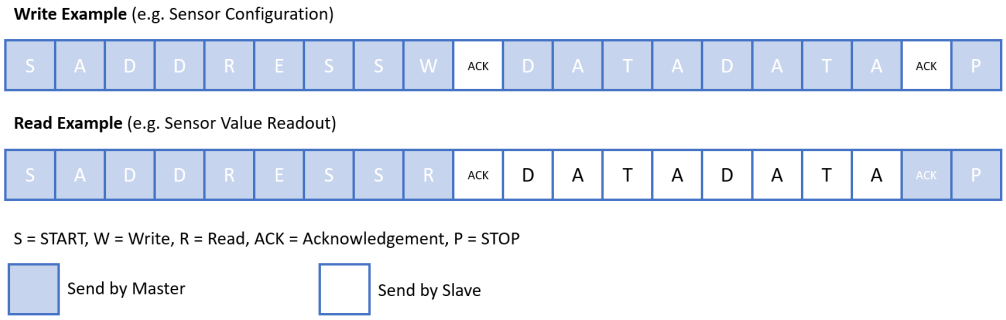

Figure 2.10.: Write and read example for I2C communication

2.2.1. I2C in NerMo v4.1

The choice to use I2C was made as the neurorobotic mouse has biomimetic legs and therefore limited space as seen in Figure 2.11, which is one of the advantages of I2C as there are only two wires - SDA and SCL - that need to be connected to each device. Furthermore, the protocol is standardized and therefore simple to use and has no need for additional electrical circuits to integrate the sensors.

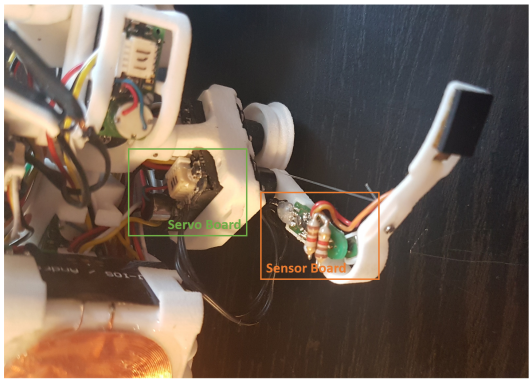

Figure 2.11.: I2C in Nermo v4.1

In NerMo v4.1 the micro-controller on the servo board is set to be the master and to generate the clock signal (SCL) with 3 V and 400 kHz, which is also known as fast mode. The three sensors - motor position, foot pressure and knee angle - are configured as slaves as seen in Figure 2.12. This setup is designed to be the same for all four legs to ease the implementation in software.

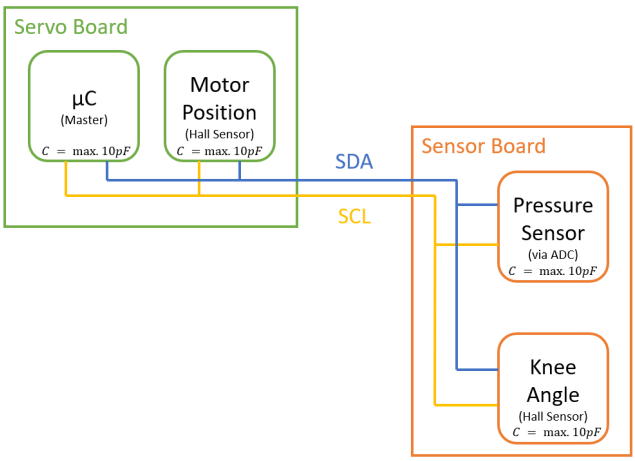

Figure 2.12.: I2C setup schematic

2.2.2. Timings and Resistors

In I2C a open collector (NPN - transistor) or open drain (MOSFET) is used on each device. Therefore, on each device a transistor pulling against LOW (GND) is built-in. The device is enabled to have a stable LOW signal by this concept (see Figure 2.13), but is unable to reach a stable HIGH. For that reason VCC needs to be connected to the input and to prevent shortcuts, a so called pull-up resistor is used as a connection between those. As every circuit has a certain capacitance (here: 32,8 pF) the whole circuit can be described as RC-circuit. So, the sizing of the pull-up resistor has a impact on the voltage as a function of time.

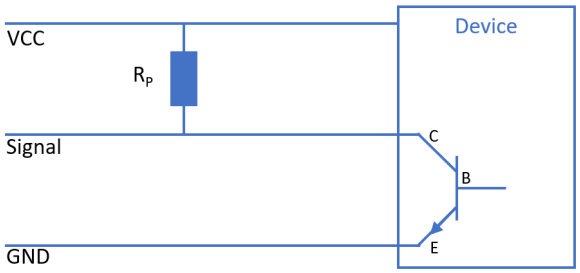

Figure 2.13.: RC-circuit with open collector with NPN - Transistor and pull-Up resistor

As the timings of a signal are relevant to achieve a well working communication infrastructure, pull-up resistor sizing is covered in this section. First, the theory is covered, followed by the approximation for the bus capacitance and finished by a comparison of the expected and measured results.

Calculation

To match the requirements of the I2C specification the pull up resistors (see Figure 2.13) have to be correctly sized as they have an impact on the rising edge of SDA and SCL. There are two bounds, which have to be considered for the resistor choice. The lower bound is set by the maximum current the devices tolerate and the lower bound by the maximum rise time allowed for the specific mode, e.g. 300 ns for fast mode (see [25] for other modes).

$$R_P(min) = \frac{V_{CC} - V_{OL}(max)}{I_{OL}} \tag{2.1}$$

with $V_{OL}(max)$ being the maximum LOW output voltage (here 0.4 V) and I_{OL} the maximum LOW output current for all devices (here 3 mA)[25]. These values lead to a lower bound of $R_P(min) \approx 867\Omega$ with $V_{CC} = 3V$ for the configuration used in NerMo v4.1.

The maximum rise time can be derived from the charging function of a RC-circuit as a function of time applied to the current voltage:

$$V(t) = V_{CC} \cdot \left(1 - exp(\frac{-t}{\tau})\right) = V_{CC} \cdot \left(1 - exp(\frac{-t}{R_P \cdot C_B})\right) \tag{2.2}$$

with R_P being the size of the pull up resistor and C_B the bus capacitance (for estimation see below). If now t_1 and t_2 are set as the times where 70 % and 30 % of V_{CC} are reached, the pull up resistors' upper bound can be calculated with:

$$R_P(max) = \frac{t_R}{\left(ln(1 - 30\%) - ln(1 - 70\%)\right) \cdot C_B} \tag{2.3}$$

where t_R is the desired rise time of the signal. For fast mode one can plug in 300 ns as upper bound for t_R and for NerMo v4.1 the bus capacitance can be approximated with 32.8 pF per I2c Bus (one per leg), which leads to $R_P(max) \approx 10.8k\Omega$.

If one is approaching this upper limit or exceeding it, the frequency will drop as seen in Figure 2.14, Figure 2.15 and Figure 2.16. Although being a slightly different setup with $V_{CC} = 1.8V$ and $t_{HIGH} = 156,25ns$ and $t_{LOW} = 2125ns$, the observations can be transferred to any I2C setup. To match the measured results with calculations (see Table 2.4) the time for a complete cycle can be calculated with

$$t_{cycle} = t_{HIGH} + t_{LOW} + t_R + t_F + t_{Filter} \tag{2.4}$$

with in case of NerMo v4.1 t_F approximated with ~ 0 ns and t_{Filter} approximated with ~ 500 ns [26].

Figure 2.14.: Clock signal SCL with $V_{CC} = 1.8V$ and $R_P = 1,8k\Omega$ resulting in $t_{cycle} = 2,81\mu s$

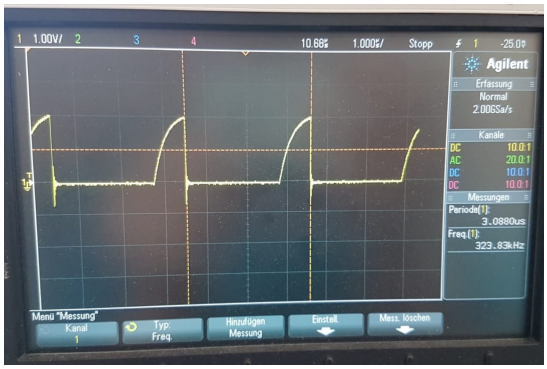

Figure 2.15.: Clock signal SCL with $V_{CC} = 1.8V$ and $R_P = 10k\Omega$ resulting in $t_{cycle} = 3,09\mu s$

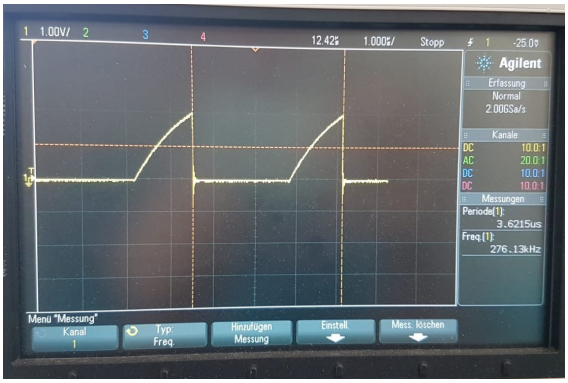

Figure 2.16.: Clock signal SCL with $V_{CC} = 1.8V$ and $R_P = 30k\Omega$ resulting in $t_{cycle} = 3,62\mu s$

Table 2.4.: Comparison of different pull-up resistors

Resistor	$t_R(calc)$	$t_{cycle}(calc)$	$t_{cycle}(meas)$
1k8	50 ns	$\sim$ 2830 ns	2810 ns
10k	278 ns	$\sim$ 3060 ns	3088 ns
30k	834 ns	$\sim$ 3615 ns	3622 ns

Bus Capacitance estimation

For a good approximation of the rising edge the bus capacitance has to be known. This can be done with the above used formulas.

1. Set R_P to a reasonable size between $R_P(min)$ (Equation 2.1) and an approximation for $R_P(max)$ (Equation 2.3)
2. Measure the rise time either for different upper and lower bounds, e.g. 20 % to 80 % , 20 % to 70 % , 30 % to 70 % or different pull-up resistors
3. The bus capacitance can now be approximated with

$$C_B = \frac{t_R}{\Big(ln(1 - LOW\%) - ln(1 - HIGH\%)\Big) \cdot R_P} \tag{2.5}$$

For NerMo v4.1 the total bus capacitance can be approximated with $C_B \approx 32,8pF$ as:

- $t_R(20\ \%\ \text{to}\ 80\ \%) = 50ns \rightarrow C_B = 32,8pF$
- $t_R(20\ \%\ \text{to}\ 70\ \%) = 36ns \rightarrow C_B = 33,4pF$
- $t_R(30\ \%\ \text{to}\ 70\ \%) = 30ns \rightarrow C_B = 32,2pF$

Resulting Timings

The longer HIGH and LOW period seen in Table 2.5 can be explained with the device specific times tsync1 and tsync2 (see reference manual of the micro-controller [26]), which are the sum of analog / digital filter, falling / rising edge and clock synchronization (just for multi-master configurations). In this scenario the times are approximated with $tsync1 =$ <260ns (analoge filter [26]) $+ 5ns$ (falling edge) $+ 0ns$ (only one master) and $tsync2 =$ <260ns (analoge filter [26]) $+ 30ns$ (rising edge) $+ 0ns$ (only one master). The time for the analog filter is known to be <260 ns, which, when it is subtracted from the measured times, leads to the in the implementation expected times.
The rise time, data setup time and data hold time results all were as expected by the calculations above.

Table 2.5.: Timing comparison

	Specification I2C	Implementation	Measured
Frequency	< 400 kHz	400 kHz	$\sim$ 385 kHz
Data Setup Time	> 100 ns	> 125 ns	>> 125 ns
Data Hold Time	$\geq$ 0 ns	0 ns	$\sim$ 0 ns
HIGH ($\geq$ 70 % VCC)	$\geq$ 1300 ns	1437,5 ns	1650 ns
LOW ($\leq$ 30 % VCC)	$\geq$ 600 ns	687,5 ns	905 ns
Rise Time (30 % to 70 % VCC)	20-300 ns	30 ns	30 ns

2.3. Inner-NerMo Communication

Besides I2C for the sensor integration, the main inner-NerMo communication is implemented on the basis of the serial communication protocol Universal Asynchronous Receiver Transmitter (UART [27]) and single wire UART (SWU). UART is used for the data exchange between Raspberry Pi, on which the control software is executed and the spine board, which has the purpose of communication management. To connect the servo boards used for sensor integration and servo control in a servo line to the spine board (see Figure 2.17) SWU is used, as UART itself only allows 'one to one' communication, but 'one to n' is needed for a more sufficient wiring setup and board design [28] in NerMo v4.1.

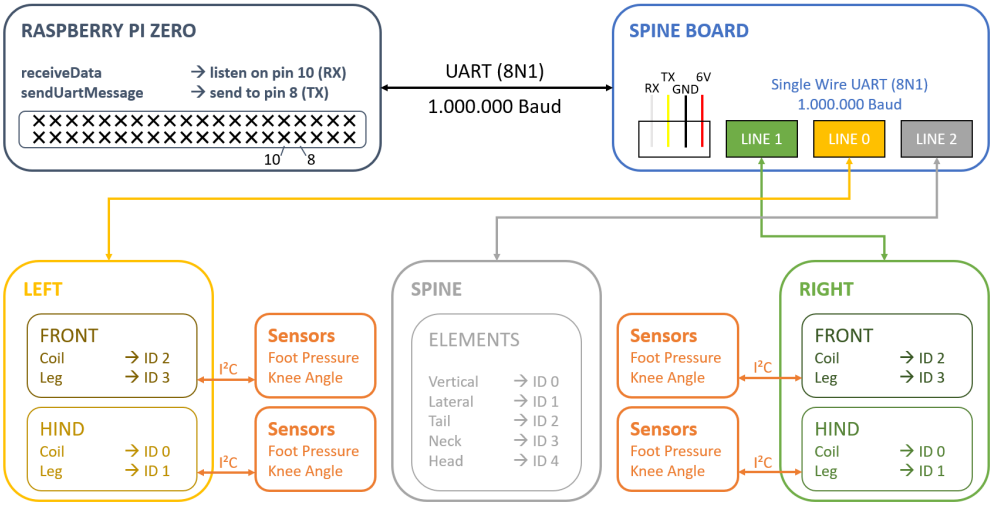

Figure 2.17.: Inner-NerMo communication overview with Raspberry Pi, spine board, servo boards (coil, leg) and sensor boards and their IDs for communication

2.3.1. Commands

For the inner-NerMo communication a basic, but functional command format for transmitting and receiving data via UART and SWU was implemented by Jörg Conradt and Peer Lucas and adapted and refined during this thesis.

General Format

To forward a command from the Raspberry Pi or spine board to the specific servo board the respective commands have to match the following format of the servo board ID followed by the actual command (*:<ID><command>*), where the ID assignment for all servo boards can be seen in Table A.1 and the command taken from the following listing. If a forwarded command has a return value the spine board will add *L:* (left), *R:* (right) and *S:* (spine/ middle) in front of the reply (e.g. *L:F-ID-XXXX*) depending on the servo line the servo board is connected to.

To run a command on the spine board the command has to be sent alone without the *:<ID>* prefix. Similarly, a command can be executed on the servo board directly (e.g. when it is connected as seen in Figure A.10 with a special cable to a PC) with the second digit of the servo ID followed by the command.

Requests

- **Help**
  Command: *??*
  Return Value: Overview of possible commands clearly formatted (debugging)
  Explanation: The returned prompt is just meant for debugging purposes and testing the reachability of the servo board. The sensor options are only shown if they were properly connected and initialized correctly, which allows further and easy debugging.
  Device: Servo Board and Spine Board

- **Servo ID**
  Command: *?ID*
  Return Value: Welcome message and servo ID clearly formatted (debugging)
  Explanation: The returned prompt is just meant for debugging purposes as well. The current software version running on the device, the date, the time as well as the second digit of the actual servo ID are included. The first digit is not known by the servo itself as the servo chain, which the servo board is connected to, is only known by the spine board.
  Device: Servo Board

- **Servo Position**
  Command: *?P*
  Return Value: *P-<ID>-<current>-<desired>*
  Explanation: The *ID* is set to have 2 hex digits. The first is set to 0 and the second digit to the second digit of the actual ID. *<current>* is replaced by 4 hex digits, which return the current servo position read by the internal hall sensor (range 0x0000 to 0x0FFF).

  $$position = (2048/\pi) \cdot angle + 2048$$

  Similarly, the desired position is returned with 4 hex digits in the same format.
  Device: Servo Board

- **Knee Angle**
  Command: *?K*
  Return Value: *K-<ID>-<BX>-<BY>*
  Explanation: The *ID* is set to have 2 hex digits. The first is set to 0 and the second digit to the second digit of the actual ID. *BX* and *BY* are returned by the magnetic flux sensor mounted at the knees of NerMo (range 0x0000 to 0x0FFF). These values are sent in the two's complement. So, if they are bigger than 2047, 4096 has to be subtracted for further calculations. After the subtraction the knee angle can be calculated as follows:
  $$angle = atan2(BY, BX)$$
  , which returns an angle between $-\pi$ and π.
  Device: Servo Board

- **Foot Pressure**
  Command: *?F*
  Return Value: *F-<ID>-<value>*
  Explanation: The *ID* is set to have 2 hex digits. The first is set to 0 and the second digit to the second digit of the actual ID. *<value>* is forwarded from the thin-film pressure sensor mounted at the feet of NerMo (range 0x0000 to 0x0FFF). The value is decreasing with increasing pressure and if properly mounted (punctual pressure on sensor) working from 20 g to 200 g of pressure on each foot.
  Device: Servo Board

- **PID terms**
  Command: *?C*
  Return Value: *C-<ID>-<P>-<I>-<D>*
  Explanation: The *ID* is set to have 2 hex digits. The first is set to 0 and the second digit to the second digit of the actual ID. P, I and D are initially set to hand tuned 8, 1 and 21 (corresponding hexadecimal 0x15) and are represented each with 4 digits (range 0x0000 to 0xFFFF). These parameters are used for the servo motor position control, which is working with the current and desired positions explained above.
  Device: Servo Board

Set Values

- **Power**
  Command: *!P<options>*
  Options: *- / RD / R+ / R- / S+ / S-*
  Explanation: - can be used for power off. To turn the power for the Raspberry Pi of with a delay of 30 seconds (e.g. see Figure A.4) *RD* has to be send, *R+* turns it back on and *R-* without delay off. Similar to that, *S+* and *S-* have the same effect, but for the servo boards' power supply.
  Device: Spine Board

- **LED**
  Command: *!L<options>* or *!L=<value>*
  Options: + / - / value in range of 0x0000 to 0xFFFF
  Explanation: + is used to turn the LED on, - to turn it off. If a value is send, it will be interpreted as a delay for blinking the LED (Servo Boards only)
  Device: Servo Board and Spine Board

- **Servo Position**
  Command: *!P=<value>*
  Options: *OFF / BLOCK* / value in range of 0x0000 to 0x0FFF
  Explanation: The motor position control is disabled with *OFF* and the speed is set to zero, the same is done with *BLOCK*, but in addition the two timers used are being set to the maximum values. A new desired position for the servo motor is set when a value is inserted.
  Device: Servo Board

- **PID terms**
  Command: *!C<param>=<value>*
  Options: param: *P / I / D* ; value in range of 0x0000 to 0x0FFF
  Explanation: The P, I or D gain for the servo motor control is set. $P = 8$, $I = 1$ and $D = 21$ (corresponding hexadecimal 0x15) are chosen to be the default, hand tuned values (done by Peer Lucas).
  Device: Servo Board

- **UART**
  Command: *!U<options>*
  Options: + / -
  Explanation: The UART output of the servo board is enabled, respectively disabled with this command, but is outdated since setting a value has no return value any longer.
  Device: Servo Board

Other Commands

- **Reset**
  Command: *RESET*
  Explanation: The selected device can be reset and restarted by *RESET* and feedback via UART communication in human readable form is given as well.
  Device: Servo Board and Spine Board

- **Bootloader**
  Command: *BOOTL*
  Explanation: To set the board software-wise to boot mode *BOOTL* has to be used, which gives human readable feedback. For an explanation of how to execute the command in detail, see section A.2 and subsection A.3.2
  Device: Servo Board and Spine Board

- **Programm Servo through Spine**
  Command: *PROG<C><S>*
  Explanation: Forwarding software through the spine board to program the servo boards without disassembling (see subsection A.3.1) with human readable feedback is allowed by this command.
  Device: Spine Board

2.3.2. Sensor Update Cycle

All eight sensor values - four magnetic sensor knee angle and four foot sensor pressure readings - are requested and stored step by step by the Raspberry Pi. First, the communication channel is blocked, then a request for the certain sensor is sent to the specific servo board. The servo boards reply is forwarded by the spine board and afterwards stored by the Raspberry Pi. After that procedure the communication channel is freed again, to allow the next request. This rather simple design was chosen to ensure save communication via SWU as collision avoidance has to be realized manually in software when using SWU. Multiple implementations have shown that this is most likely the best approach for the current communication setup.

Parallel solutions - e.g. sending data to the left, right and spine line at the same time - often lead to a large packet loss due to collisions (at least $\sim$ 20% for this specific setup) and as a result to slower update rates. So not until the first sensor value - e.g. *:01?K* as explained in detail in Figure 2.18 - is inquired and the updated value is returned and stored locally on the Raspberry Pi, the second one (e.g. *:03?K*) will be requested. Also, to prevent possible deadlocks in case a servo is not sending a reply, a conservative timeout was integrated in the step-wise execution.

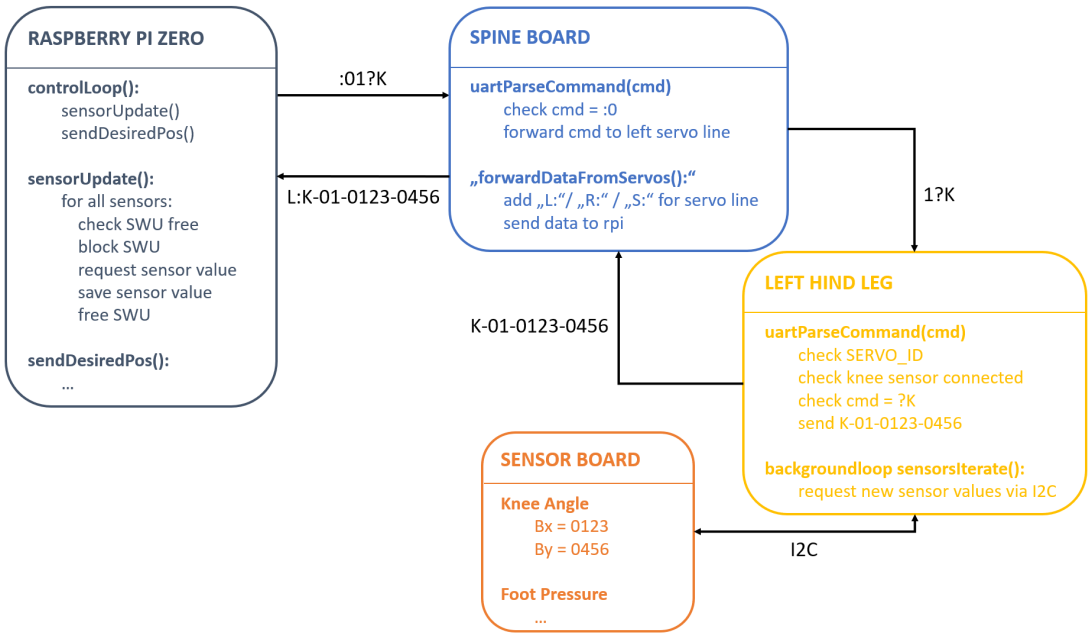

Figure 2.18.: Protocol example for inner-NerMo communication

Furthermore, the forwarding infrastructure provided by the spine board and already implemented by Jörg Conradt and Peer Lucas is used as much as possible to allow fast development on only the Raspberry Pi and the servo boards without changing the communication basis.

Evaluation of Update Cycle

For the evaluation of the implemented sensor update cycle, the time starting from the first sensor value update to the last sensor update is measured on the Raspberry Pi. Although this procedure is no proper benchmark, a reasonable statement regarding the expected performance can be made. The resulting measurements can be seen in Table 2.6 including the fastest and slowest measured time as well as the average over the observation period and their corresponding possible update frequencies.

Table 2.6.: Sensor value updates

	t_{CYCLE}	Update Rate
MIN	2,10 ms	$\sim$ 475 Hz
AVG	2,35 ms	$\sim$ 425 Hz
MAX	2,70 ms	$\sim$ 370 Hz

The communication time (transmission time) can be approximated for NerMo v4.1 as follows:

$$t_{COMM} = \frac{\text{number of characters} \cdot 10bit \text{ (8N1)}}{\text{average measured baudrate}} = \frac{264 \cdot 10bit}{1,15 \cdot 10^6 bit/sec} \approx 2,3ms \tag{2.6}$$

with 264 characters being the total number of characters transmitted during an update cycle. The faster than expected ($t_{expected} = 2640bit/10^6 Bd = 2,64ms$) t_{COMM} is the cause of a slightly faster than programmed baudrate of $\sim 1,15 \cdot 10^6 Bd$ instead of $1 \cdot 10^6 Bd$. The remaining time can be explained with the execution time, the different interacting devices - Raspberry Pi, Spine Board, 4 servo boards - need for data processing, forwarding and storing.

$$t_{EXEC} = t_{CYCLE} - t_{COMM} \approx 2,35ms - 2,3ms = 50\mu s \tag{2.7}$$

To further improve the communication, one could decrease the number of characters transmitted to at least 224 characters with removing the two, respectively three zeros sent in each reply message (e.g. *L:K-01-0234-0567* to *L:K-1-234-567* or *L:F-01-0234* to *L:F-1-234*). Therefore, a decreased update time of around 14,9 % can be achieved.

$$t_{improved} = \frac{224 \cdot 10bit}{1,15 \cdot 10^6 bit/sec} + t_{EXEC} \approx 2ms$$

To implement the proposed changes, the complete infrastructure would have to be re-implemented as it is based on sending values with either two (8 bit) or four (16 bit) digits in hexadecimal format.

3. Analysis of Open Loop Walking Control

The current state of the open loop control implemented by Peer Lucas [29] is analysed in detail in this chapter. Therefor, the in the previous chapter integrated and tested knee angle and foot pressure sensors are used. To create a baseline for later improvements e.g. with a closed loop walking control, NerMo's current locomotion patterns are tested extensively and documented. This is done with several different experiments.

First, the implemented walking pattern is analysed without ground contact to get an order of magnitude, which trajectory was planed and how it differs from walking with ground contact. Walking with ground contact is researched in the next section. Therefor, the knee angles and foot pressure readings are plotted and compared to the first setup. Additionally, the repeatability of a certain trajectory like walking straight for ten steps is reviewed. In a last section the main focus is set on the natural habitat of a common rat like gravel, stairs etc., so the movement in more difficult terrain.

3.1. Walking without Ground Contact

To create a comparison baseline of the current open loop control, the foot pressure sensor and knee angle sensor readings are plotted over a period of approximately ten steps. This is done without ground contact to later see the impact of the environment and the bending of the legs, which can differ quite a lot between the feet and over a long period of time.

As a result, the foot pressure reading without ground contact in Figure 3.1 for the hind legs and in Figure 3.2 for the front legs can be compared to the results from Table 2.2. When looking at the knee angle sensor readings, one can see two major anomalies. First, a strange behavior for the first two steps for the left hind leg (see Figure 3.1) and the front right leg (see Figure 3.2), which can be eliminated by transmitting the command *w* for walking straight forward to the main control program of NerMo a second time. This nominal behavior can be seen for steps three to ten in the same figures. Secondly, a slight deviation in symmetry can be observed. This can be explained by a small variance regarding the tendons used for lifting the rat's feet and their initial tension.

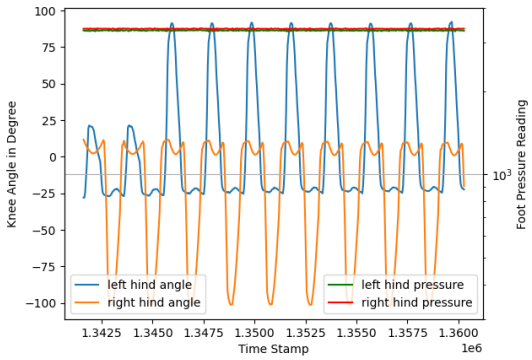

Figure 3.1.: Hind leg foot pressure and knee angle reading without ground contact

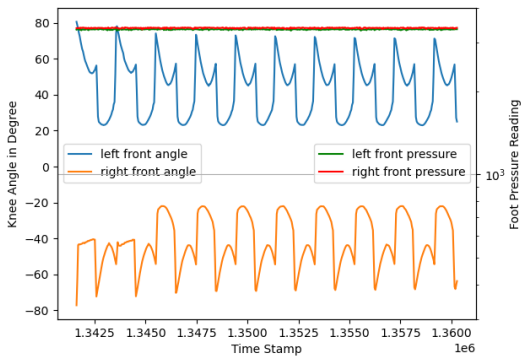

Figure 3.2.: Front leg foot pressure and knee angle reading without ground contact

3.2. Walking on Even Ground

The analysis of NerMo v4.1 walking on even ground is split into three parts. In the beginning the foot pressure reading is plotted and evaluated. Afterwards the same analysis is done for the knee angle and further compared to the plots without ground contact. Lastly, the repeatability of a certain trajectory - here walking straight forward for ten steps - for the robot is tested.

3.2.1. Foot Pressure

The foot pressure reading works as expected just for a basic analysis. Three major aspects can be observed (see Figure 3.3 and Figure 3.4) nevertheless:

- **Ground Contact Detection**
  The sensors can be used as presumed beforehand for ground contact detection, which is especially interesting for a future closed loop control designed to work well on several different surfaces.

- **Major Misbalancing**
  In this certain graph (Figure 3.4) one can see that the centre of gravity is mainly on the right front foot. This misbalancing leads to an undefined and unwanted slight right turn.

- **Sensor Bearing Issues**
  When looking at the green functions in Figure 3.3 and in Figure 3.4, one can observe that the bearings are a little bit stuck. As a result, the unloaded foot pressure reading level (ADC reading $\sim$ 3400) is not achieved within every step cycle. This issue can be minimized by finer filing the front leg bearing and widening the holes for the hind leg bearing a little.

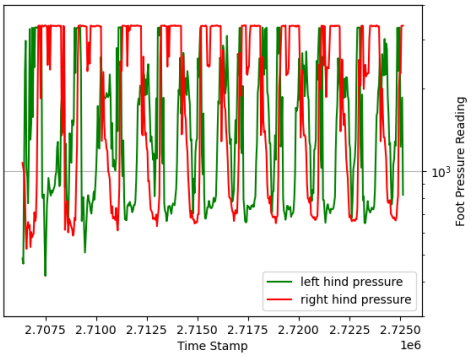

Figure 3.3.: Hind leg foot pressure reading with ground contact on even surface

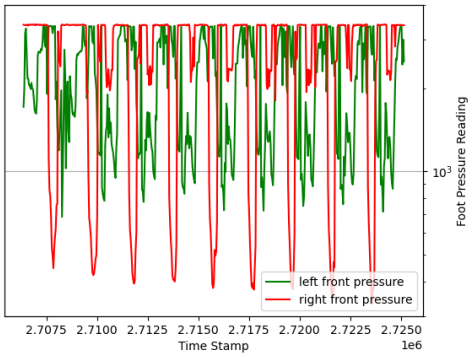

Figure 3.4.: Front leg pressure reading with ground contact on even surface

3.2.2. Knee Angle

The walking performance regarding the knee angles on even ground can be summarized as okay to good, but some issues occurred during the testing period. Looking at the knee angle sensor readings plotted in Figure 3.5 and in Figure 3.6, one can see that during the ground contact the local minima and maxima introduced for a more rodent-like locomotion are shifted (compare Figure 3.6 and Figure 3.2) or even lost (compare Figure 3.5 and Figure 3.1). Furthermore, it can be observed that the symmetry, respectively balance is worse than without ground contact, which leads to a non-straight trajectory. Also the starting behavior as seen for the first two steps is not only affecting the right front and left hind leg as without ground contact, but all four legs.

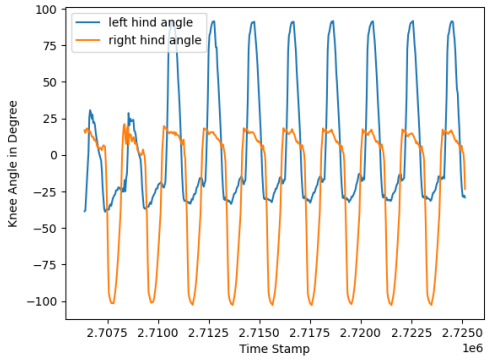

Figure 3.5.: Hind leg knee angle reading with ground contact on even surface

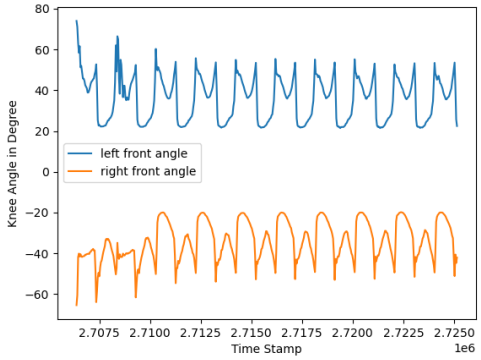

Figure 3.6.: Front leg knee angle reading with ground contact on even surface

3.2.3. Repeatability of a Trajectory

To measure the repeatability of a trajectory, a simple trajectory on even ground was chosen to ease the comparability of the different runs. Therefor, walking straight for ten steps is set to be the task. A step in this experiment is defined as a complete cycle of the right front leg - starting with the first ground contact and ending at the next - to allow simple manual step counting. To start the experiment, NerMo is set to the initialisation position before each run and aligned to a centered position (see Figure 3.7).

Figure 3.7.: Experiment begin with NerMo in the initial position

After the initialisation, the rodent robot is programmed to walk ten steps straight by pressing *w* for two times in the main control program (see Figure 3.8). This experiment is repeated ten times and for each run the end position - here defined as the middle between the right and left front foot - is marked by a cross on the grid.

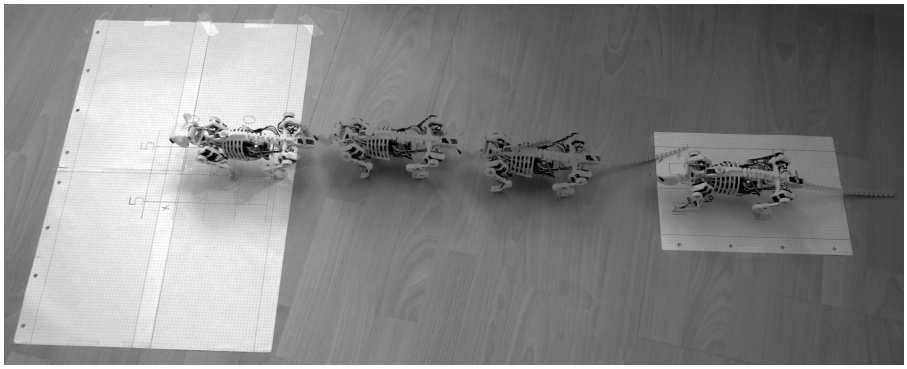

Figure 3.8.: Example experiment run with NerMo walking for ten steps

The result can be summarized as good because the deviation of end positions after 10 steps of walking is quite small (see Figure 3.9). The two end position furthest apart in x-direction are distanced by 11 cm with the left most being 6 cm and the right most 5 cm away from the middle. The maximum deviation from the exact straight line (orange line in Figure 3.9) is 6 cm, where the average deviation is 3,1 cm. Looking at the distance covered, the deviation is slightly bigger. The two end position furthest apart in y-direction are distanced by 14,5 cm with the shortest being 67 cm and the furthest 81,5 cm. The average covered distance after ten steps is 72,8 cm (green cross in Figure 3.9). Taking the average covered distance as expected distance, the average deviation is 3,2 cm.

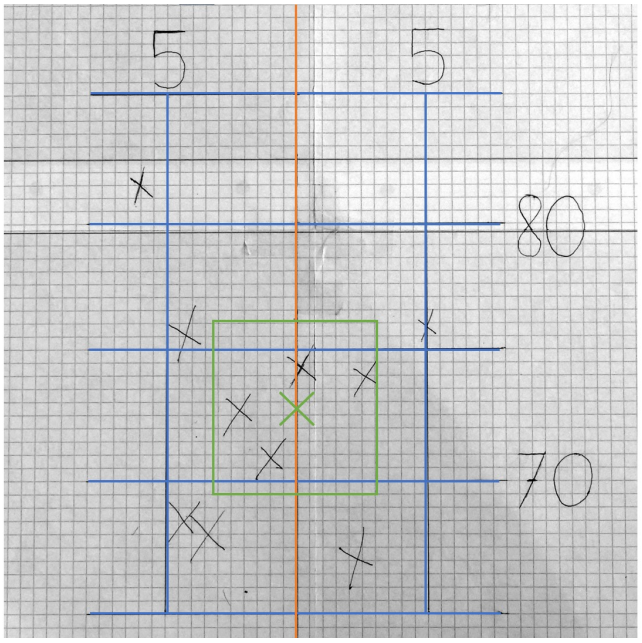

Figure 3.9.: Grid with the 10 experiment run's end positions crossed in black. The orange line marks the exact straight line and the blue lines mark 5 cm by 5 cm cells labeled with the distances - in y-direction from the initial position of the robot and in x-direction as distance from the straight line - both in cm. The green cross is marking the average distance performed during the 10 runs and the green rectangle is displaying the measured deviation from that average.

3.3. Walking on Uneven Ground

To further test and check the limits of the open loop walking control, the rodent robot should be tested on a more natural surface such as gravel or small stairs. But as the

ground clearance, respectively the height, which the feet can be lifted of the current setup is very limited, these tests failed. The robot was stuck before nearly all obstacles higher than 2-3 mm and walking up a slight incline is possible, but very limited due to the small ground clearance and the grip of the feet of NerMo. So regarding this more difficult terrain, the walking capability of NerMo can be highly improved.

4. Closed Loop Walking Control

The focus shift caused by the unpredictable circumstance explained in section 1.5 led to the fact that this chapter is less covering a newly designed closed loop walking control per se, but more basic improvements via an initial position control and suggestions how later research could enhance NerMo's walking capabilities.

4.1. Initial Position Control

A well working initial position closed loop control would not only ease and accelerate NerMo's setup process, but even improve NerMo's capabilities as a optimized setup also enables more versatile locomotion. For example by this optimized setup, walking straight would be improved by a more symmetrical feet alignment. Furthermore it would be possible to lift the feet higher, which is a important step closer to walking on more difficult terrain.

Sensor Feedback

As the magnetic knee angle sensors provide a stable and well documented behavior (as shown in chapter 2), those sensors were used in the feedback loop of the closed loop controller for the initial position of NerMo. The foot pressure sensors were tested to be used additionally, but finally found to be not stable enough yet.

P-Control Design

To send a position to a coil servo motor to tense or relax the connected tendon, this position has to be an integer value between 0 and 4095. Therefore, the later calculated angles have to be transferred via the formula in Equation 4.1 to that format.

$$y[t] = \left\lfloor \left| 4095 \cdot \frac{y_{angle}[t]}{360^\circ} \right| \right\rfloor \tag{4.1}$$

To calculate a new, better angle for the coil (see algorithm 1), the initialisation position $y_{angle}[0]$ - the initial default value is here set to 180°, which showed quite good results - is corrected by a certain factor in each iteration (see Equation 4.2) dependent upon the current knee angle error $e[t]$ (see Equation 4.3) and the proportional multiplying factor $K_P[t]$ of the P-controller (see Equation 4.4). If the angle error is smaller than 1.0° (achievable accuracy), $e[t]$ is set to zero and the control loop is completed.

$K_P[t]$ is set to be adaptive in case a lot of tendon has to be collected by the coil first, before the knee angle is changed, but to allow the needed precision to reach the accuracy of 1.0° when it is changed.

$$y_{angle}[t] = y_{angle}[0] + \sum_{i=1}^{t} e[i] \cdot K_P[i] \tag{4.2}$$

$$e[t] = \begin{cases} \alpha_{knee}(goal) - \alpha_{knee}[t] & \text{if } \left|\alpha_{knee}(goal) - \alpha_{knee}[t]\right| \geq 1,0° \\ 0 & \text{else} \end{cases} \tag{4.3}$$

$$K_P[t] = \begin{cases} 0,1 & \text{if } \left|\alpha_{knee}[t] - \alpha_{knee}[t-1]\right| \geq 0,2° \\ 0,15 & \text{else} \end{cases} \tag{4.4}$$

If the knee angle $\alpha_{knee}[t]$ is not changed for 30 iterations - "check motor, tendon and leg because angle is not changed or re-init" - or the goal angle $\alpha_{knee}(goal)$ is not reached after a whole turn of the coil - "check leg and goal angle because goal angle could not be reached or re-init" - an error message is prompted (see Figure 4.1 for an example error message caused by a broken servo motor).

Evaluation

In Figure 4.1 a typical prompt of the initial position controller can be seen, which includes a statement if the goal angles could be reached or an error message if not. Furthermore, the calculated initial positions (see Figure 4.1 bottom) are stored for later improving NerMo's walking capabilities. If the coils and tendons were mounted not too far from the optimum position, the results by the P-controller can be summarized as good because the goal angle of all four coils are reached quite fast and therefore allow a simple and fast initialisation of NerMo's tendons in comparison to a time consuming manual optimization. Otherwise, the P-control is sometimes faulty as a too far off pre-mounting of the tendons leads to the "check leg and goal angle because goal angle could not be reached or re-init" error as described above because although using an adaptive $K_P[t]$ the tendon is not tightened fast enough. This is done as an intentional safety measure as a bigger $K_P[t]$ could unintentionally rip tendons.

```
init successful!
k = 0: left hind; k = 1: left front, k = 2: right hind, k = 3: right front
0: ERROR; check motor, tendon and leg because angle is not changed or re-init
1: init angle reached
2: init angle reached
3: init angle reached
-----------------------------------
332.899 62.9469 162.477 328.839
Positions stored
```

Figure 4.1.: Example output of closed loop initial position control

Result: optimized initialisation position

```
/* initialise variables                                          */
```
set $\alpha_{knee}(goal)$ for all legs
initialise $y_{angle}[0]$ with default value

```
/* main control loop                                             */
```
while *for all legs e[t] > 1.0* **do**
  forall *legs* **do**
    update $\alpha_{knee}[t]$ for certain leg
    calculate e[t] for that leg
    if α_{knee} *changed* **then**
      $K_P = 0.1$
    else
      $K_P = 0.15$
    end
    $correction = e[t] \cdot K_P$
    $y_{angle}[t] = y_{angle}[t-1] + correction$
  end
end

```
/* save positions and print messages                             */
```
forall *legs* **do**
  if $\alpha_{knee}(goal)$ *reached* **then**
    save $y_{angle}[t]$
    print "init angle reached"
  else
    if α_{knee} *changed* **then**
      print "check leg and goal angle because goal could not be reached"
    else
      print "check motor, tendon and leg because angle is not changed"
    end
  end
end

Algorithm 1: Pseudo-code for Initial Position Control

When comparing the knee angles from the closed loop to the open loop initial position control, the knee angles - except for ID 01 (goal angle of $+10.0°$) in Figure 4.2 and in Figure 4.3 as the servo motor is broken - are as expected more symmetrical and therefore would lead to a more precise behavior of NerMo, when for example walking straight. This could unfortunately not be tested as a servo motor broke without replacement in stock.

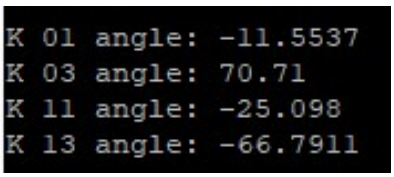

Figure 4.2.: Open loop controlled initial knee angles in degree - servo with ID 01 broken

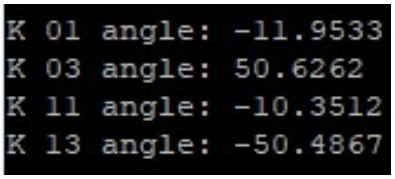

Figure 4.3.: Closed loop controlled initial knee angles in degree - servo with ID 01 broken

4.2. Further Possible Improvements

With the newly implemented initial position control the first steps towards a closed loop walking controller were done and the required software interfaces implemented. In this section possible further improvements regarding NerMo's locomotion are proposed. Starting with the possibilities enabled by a better foot pressure setup and the mounted IMU integrated as well, recommendations for a walking controller are summarized.

Improving the Foot Pressure Sensor Setup

Improving the current foot pressure sensor setup to be more stable and precise would allow for a better and more accurate initial position control. Not just the tendons could be adjusted then, but also the legs could be aligned properly, which would reduce misbalancing of NerMo. Furthermore, a simple walking controller could be implemented using the foot pressure sensor input as interrupt. So, the planned trajectory is followed until the interrupt and afterwards the next sequence is started, e.g. trajectory for the foot in the air, interrupt of the foot pressure sensor sensing ground contact, trajectory for the foot on the ground, interrupt of the foot pressure sensor sensing release etc. This algorithm would enable NerMo's locomotion to be more defined on uneven surfaces like gravel.

IMU Integration

By the proper integration of the pre-mounted IMU in hardware and software with soldering correctly sized pull-up resistors to the spine board and implementing the I2C communication between the micro-controller and the IMU, a large set of improvements could be implemented. A 3-axis gyroscope, a 3-axis accelerometer, a 3-axis compass is included within the IMU. Better position regulation of NerMo would be enabled by the gyroscope, a more defined locomotion with acceleration target values included to the closed loop control by the accelerometer and a improved control to move NerMo in certain directions by the compass. Overall, combining the three input sources - foot pressure, knee angle and IMU - could allow vision free control of NerMo comparable to MIT's Mini Cheetah [23].

Walking Controller

Most of the work can only be started after solving the problem of the not high enough lifted feet (see section 3.3). This development can be seen as the largest possible single improvement besides enhancing the durability of NerMo. Afterwards, using the multi sensor input for the closed loop control as described in [30], [31] and [32] can be researched in greater detail. Thereby, the aim should always be to allow and improve the varied and nature-like movements of NerMo for different tasks on several surfaces.

5. Conclusion and Outlook

As working not only with software, but also with hardware, can be quite challenging sometimes, a section with the lessons learned, while working on a biomimetic robot is included to this conclusion and outlook chapter, which gives insights into more general aspects to be considered, when (re-)designing a biomimetic robot.

5.1. Lessons Learned for Biomimetic Robots

With biomimetic robots, there are a few things to pay special attention to. In this work the tendons, circuit boards, 3D printing and cabling were causing problems. Therefore, the lessons learned are summed up in this section.

Tendons

For tendons, the simple solution of nylon cords screwed to the required parts has led to two major problems - screws tearing the cord and the cord ripping while bending. Therefore, experiments with other materials or braided cords could be interesting. Regarding the mounting, glue could be seen as an alternative replacing the screws, but was not used here because the cords yet ripped too often.

Circuit Boards

As the circuit boards for biomimetic robots can be very small as it is the case for NerMo, the design can be quite challenging. Nevertheless, connecting sensors or other components in general via a transistor is highly recommended as some parts need to have certain start-up times or to be re-started sometime. Also having a closer look at the size of internal pull-up resistors and in case the resistors are too big adding external ones, can prevent unwanted workarounds. Additionally, the components should all match the same protocol requirements as some e.g. define fast mode in I2C communication with < 400 kHz and some > 800 kHz.

3D Printing

Although 3D printing is becoming more and more popular, this technology has to be understood in detail to achieve good results as accuracy and longevity is often a problem. Furthermore, printing flexible parts can be especially challenging. The parts printed for NerMo by Peer Lucas were using the Selective Laser Sintering (SLS) process. "[The]

chosen SLS process made it possible to produce Nylon parts that can undergo more substantial deformations in the elastic domain repeatedly without breaking." [11] This material though wears through very fast and therefore the legs have to be replaced quite often - in a period of two months the measured knee angles were reduced by 10 - 20 degree just for that reason.

Cabling and Connections

Cabling can be very challenging, when having limited space on a biomimetic robot and therefore has to be well thought out. Plug-in connectors will probably be the best choice, if the circuit board is sufficiently large as they allow a strong, but simple to change connection. More problematic are the often used soldering pads with the wire soldered to it, when space is very limited. For those, the size and mechanical load have to be taken into consideration as the soldering connection may be sturdy, but most often small soldering pads are torn off the boards, which is sometimes irreparable and therefore has to be prevented.

5.2. Conclusion

Within this thesis, the magnetic knee angle and foot pressure sensors were integrated in software and hardware and tested in detail. Therefore, the I2C implementation - also in hardware and software - had to be changed, the knee angle sensor was changed to another model and a shoe for making reasonable use of the foot pressure sensor was designed. Afterwards, the sensors were integrated to the software consisting of software running on the servo and spine boards as well as high level code running on a Raspberry Pi for the main control. Therefore, a protocol was implemented, which allows bidirectional communication and efficient data exchange on single-wire UART. This protocol uses a basic master-slave architecture and step by step execution with the Raspberry Pi being the master and sending the requests and the custom boards as slave only sending replies. Thereafter, these sensors were used to analyse the walking performance of NerMo with the current open loop walking control. The analysis showed that simple tasks like walking without ground contact or straight for a few steps etc. are working quite well, but even walking across a ruler or a small piece of LEGO is too challenging, so that NerMo fails. At the end, a closed loop initial position controller was designed to ease the setup of NerMo's legs and tendons to later allow a more precise walking performance.

5.3. Outlook

To use NerMo as a well working robot for research, a few aspects have to be changed. Starting with the mechanical design, the legs' durability has to be improved for an extensive use. Also, the new leg design should consider the foot pressure sensors to

find a replacement or improvement of the shoes, which are only a provisional solution by now. In addition to that, the used tendons should be reconsidered and included to the new leg design as mentioned in the lessons learned section above. Apart from the legs, the motor mounts should be improved as they are designed to fit only a motor, but not a motor with the wires to connect the sensor board soldered to it. This leads to the possible improvements regarding the electronic circuits and cabling. The cabling could be improved with larger solder pads or plug-in connectors for better durability. Furthermore, the required pull-up resistors for the servo board to sensors and the spine board to IMU, which is by now not integrated in software and never tested, I2C communication and transistors to restart sensors should be included with the next circuit board design. Summing up, the overall durability of the rodent robot has to be further improved to be truly biomimetic not only by appearance and behavior, but also the nature-like longevity.

After that revision, interesting research questions like a closed loop walking controller for simple and difficult terrain using the provided multi sensor setup, more advanced robot-animal interaction as it was started in [21], learning behavior from actual animals, navigation using SLAM etc. can be discussed. Other interesting open topics would be the integration of biomimetic sensors like the wide-angle cameras as eyes or the whiskered robot vibrissae [14][15] to the rodent robot NerMo.

A. Manuals

A.1. Connect to Nermo v4.1

1. **Setup WiFi AP**
   A WiFi access point has to be configured with the SSID *LOSTLITTLEROBOT* and the password *********. The Mouse will automatically connect to that WiFi.

2. **Connect via SSH**
   Now a SSH connection to *pi@<IP>* can be established. For this log in procedure the same password as above can be used. Those - log in and password - can be saved as reusable credentials as in Figure A.1.

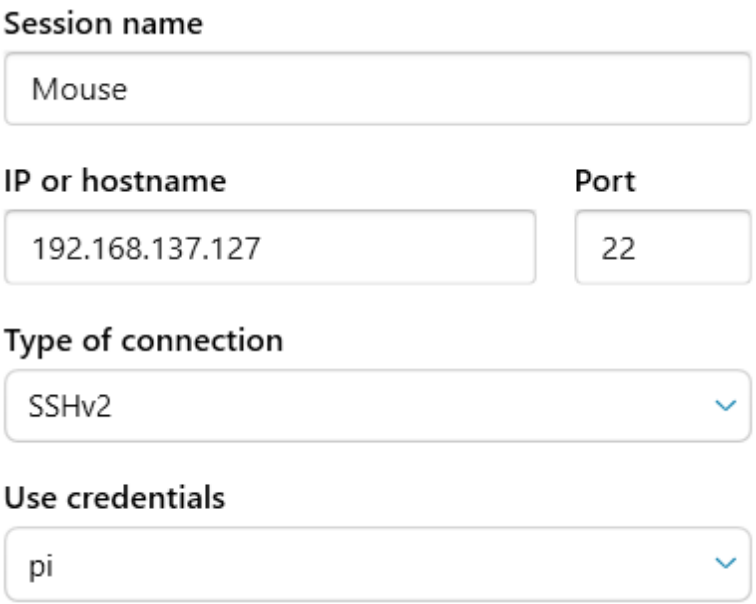

Figure A.1.: SSH connection

A.2. Program Spine Board

1. **Hardware Setup**
   To program the spine board three wires (GND, TX and RX) have to be soldered to the spine board (Figure A.2 left). After that step it can be connected via a USB to UART adapter to a computer (Figure A.2 right).

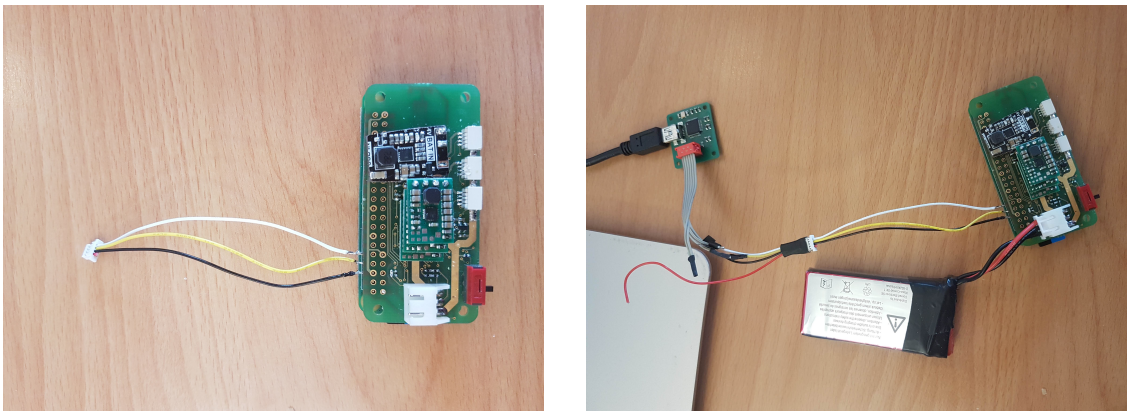

Figure A.2.: Wire and complete setup to program servo board

2. **SSH Connection**
   Now that the Hardware part is completed, the ssh connection (Figure A.3) to the Raspberry Pi has to be established as explained in section A.1.

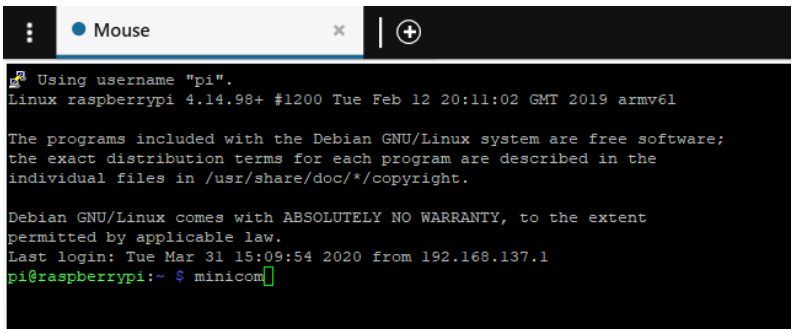

Figure A.3.: Connection to Nermo v4.1

3. **Minicom**
   With minicom started the available options can be printed with the command *??* (Figure A.4).

```
??

HBP Mouse Spine V0.8: Dec 13 2019, 15:37:45  DEBUG
Available Commands:
  :ixxx       forward xxx to servo chain i [0..3]

  !PS[+,-]    set power to servos
  !PR[+,-][D] set power to RPIZW (D:delayed by 30sec)

  !L[-,+,.]   set LED off/on/blinking
  ?S          send all sensor values (debug)

  RESET       restart spine
  BOOTL       enter boot-loader for reprogramming
  PROG<C><S>  reprogram servo <S>[0..4] on chain <C>[0..2,L,R,S]
```

Figure A.4.: Minicom options

4. **Disconnect Raspberry Pi**
   To disable the power-supply of the Raspberry Pi the command *!PRD* (Delayed to allow a proper shutdown of Raspberry Pi within the next 30sec) has to be executed. After that minicom can be closed with *Ctrl + A* and then *X*. Now the Rasperry Pi can be shut down with *sudo shutdown now*.

5. **Serial Connection**
   A serial terminal has to be configured to connect to the port, where the USB to UART adapter is pluged in with the setting 1.000.000 Baud, parity none, 8 data bits and 1 stop bit (Figure A.5).

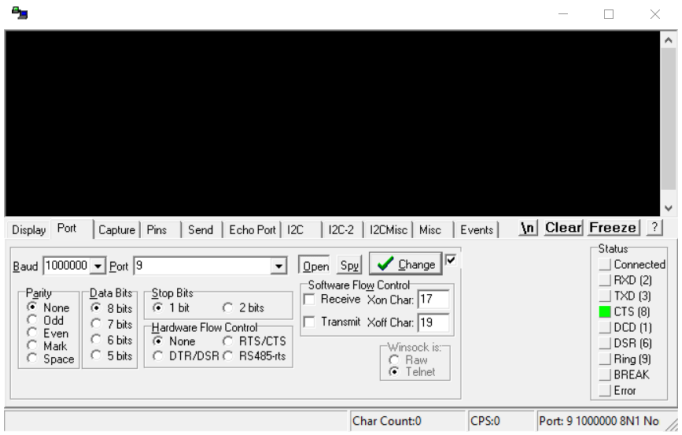

Figure A.5.: RealTerm setup

6. **Check Setup**
   To check the setup, the command *??* can be sent as above (Set end of line to *CR* and *LF*). The output should be the same as in minicom before (Figure A.6).

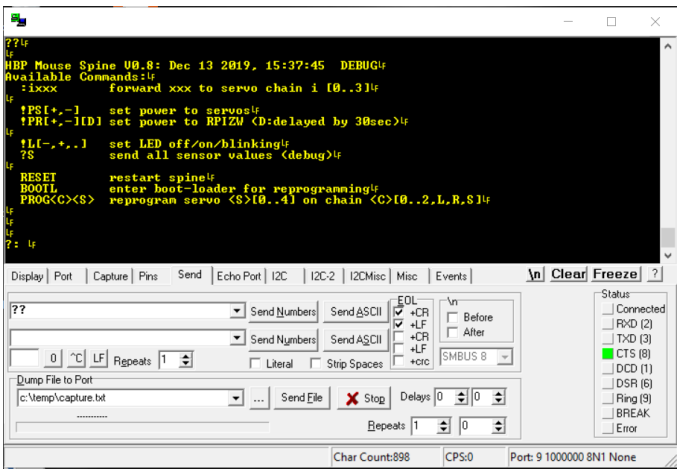

Figure A.6.: Send options in RealTerm

7. **Enter Boot Mode**
   Now as everything should be configured correctly, the command *BOOTL* allows to enter the boot mode on the spine board. If that was successful, the LED should stop blinking and the prompt as shown in Figure A.7 should appear.

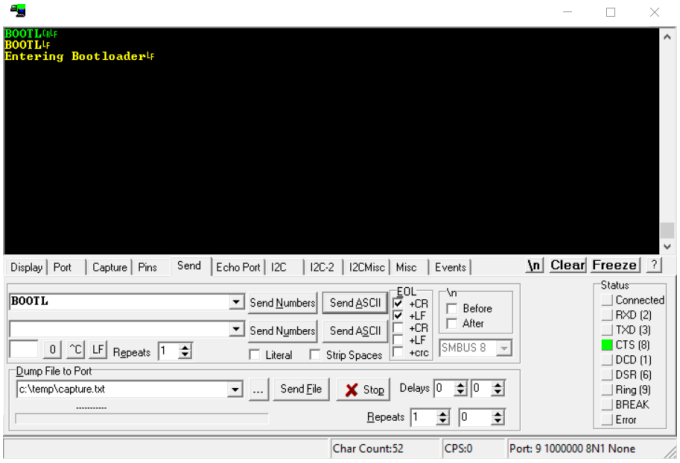

Figure A.7.: Enter boot mode

8. **Close Port**
   After entering the boot mode the port in the serial terminal has to be closed again to allow the connection to the STM32CubeProgrammer (Figure A.8).

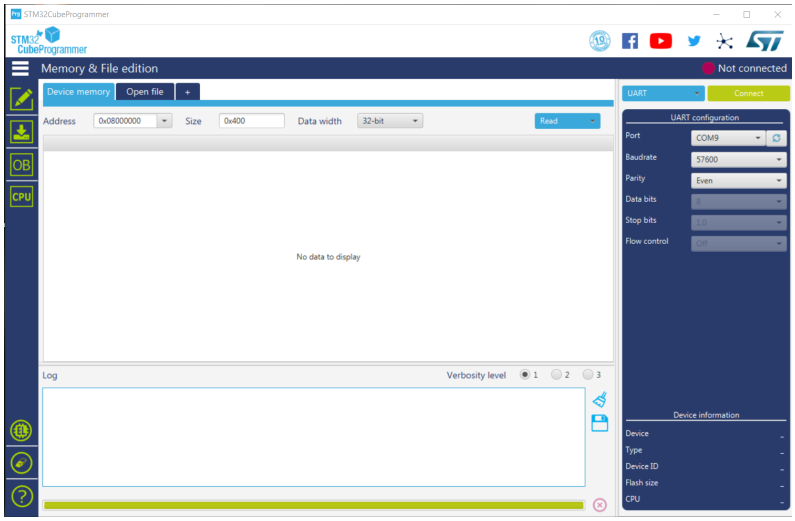

Figure A.8.: STM32CubeProgrammer

9. **STM32CubeProgrammer**
   Now the connection to the STM32CubeProgrammer can be established (UART / 57600 Baud / parity even / 1 stop bit) as in Figure A.9.

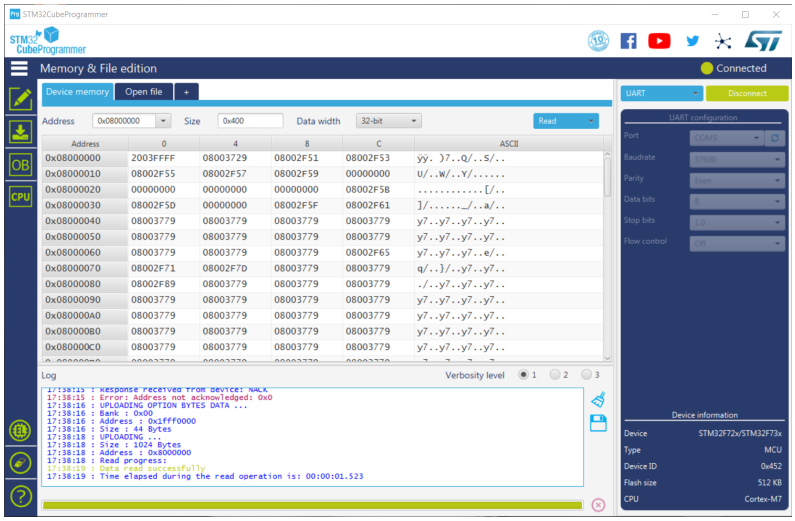

Figure A.9.: Connected to STM32CubeProgrammer

10. **Programming**
    To program the spine board the compiled code can be uploaded to the spine board as *.elf* file via the menu *Erasing & Programming*.

A.3. Program Servo Board

The servo boards can be programmed in two ways. The first one is the preferred and easier one through the spine board via code forwarding. The second one is just needed if the servo is not connected to the spine board. If the servo board is completely new or crashed, the boot mode will have to be entered manually.

A.3.1. Through the Spine Board

The procedure to program the servo board through the spine board can be compared to programming the spine board itself. The only difference is that instead of sending *BOOTL* to the spine board, *PROG<ID>* is sent.

A.3.2. Without the Spine Board

To program the servo board without the spine board a special cable (Figure A.10) is needed to connect the servo board to the USB to UART adapter.

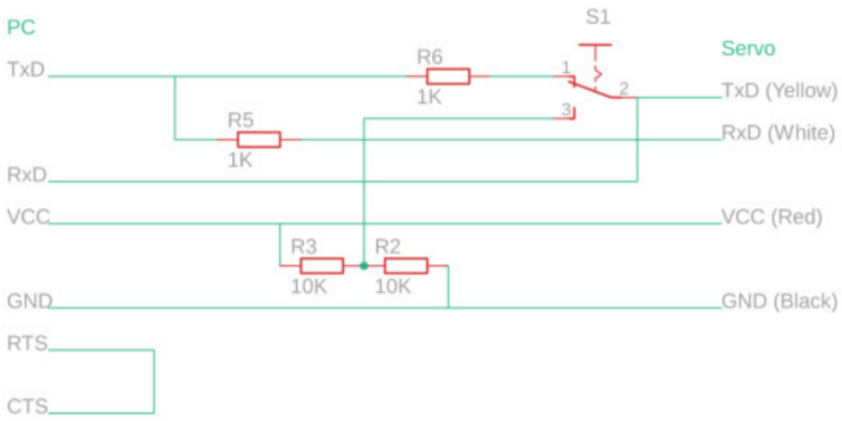

Figure A.10.: Cable to connect servo board (Figure originally from: [28])

This cable has to be connected to the USB to UART adapter and the servo board. After this is done, the serial terminal (1.000.000 Baud, parity none, 8 data bits and 1 stop bit) can be connected to the port again. In this scenario the command to enter the boot mode is *<ID>BOOTL*, where *<ID>* is just the second digit of the servos ID (Set end of line to *CR* and *LF*). The next steps are the same as for programming the spine board.

A.3.3. Set Servo Board Manually to Boot Mode

In case the servo board is not programmed yet or crashed, the boot mode will have to be entered manually. Therefore the servo has to be disassembled because the pin *BOOT0* has to be set to VCC (3.3V) during the power up. This pin can be found on the backside of the servo board. The easiest procedure (recommended by Jörg Conradt) is probably to connect the two marked pins (Figure A.11) with tweezers and a second person reprogramming the board as usual.

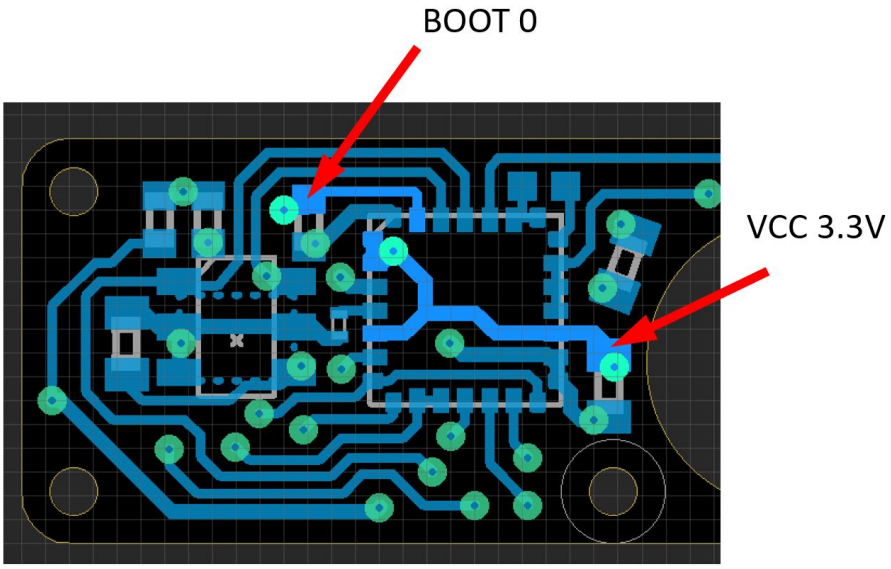

Figure A.11.: Position of the pins (Figure originally from Jörg Conradt)

A.4. Set Servo Board ID

After soldering all components according to the schematics found in [28], the ID for each servo board has to be set according to the position it will be used for. The ID, which has to be set in hardware is the last digit of the servo board ID in Table A.1. Therefor, the two pads marked with *ID encoding* in Figure A.12 have to be connected as follows:

- ID 0: open (not connected)
- ID 1: solder bridge (0Ω)
- ID 2: resistor with 1kΩ
- ID 3: resistor with 10kΩ
- ID 4: resistor with 100kΩ

The suitable size for those resistors is 0402 (1mm * 0.5mm).

Table A.1.: Servo board IDs

Position	ID	Position	ID	Position	ID
Left Hind Coil	00	Right Hind Coil	10	Spine Vertical	20
Left Hind Leg	01	Right Hind Leg	11	Spine Lateral	21
Left Front Coil	02	Right Front Coil	12	Spine Tail	22
Left Front Leg	03	Right Front Leg	13	Spine Neck	23
				Spine Head	24

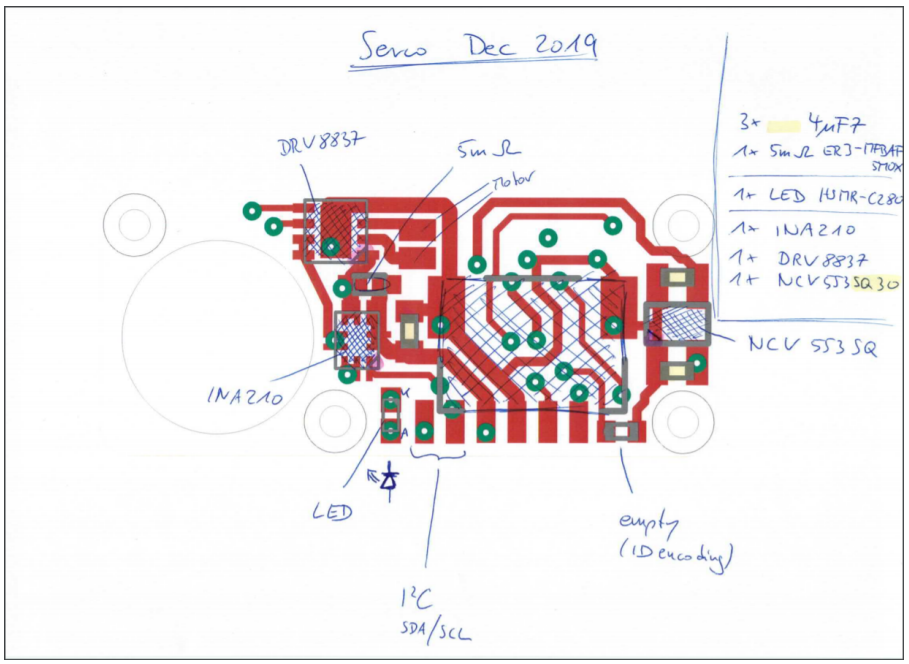

Figure A.12.: Set servo board ID (Figure originally from Jörg Conradt)

A.5. Wiring the Feet Setup

To wire the feet setup (see Figure A.13 for an overview) one should start with the connection of servo board and sensor board via four wires - SDA, SCL, V+, GND. After that procedure, the servo board can be attached to the servo via the blue and red cable - blue to *Motor 2* and red to *Motor 1* - and then fixed with screws. As a next step, the foot pressure sensor has to be connected to the sensor board via *VCC* respectively *VOUT* and *GND* (see [24] for more details). Therefor, the sensor has to be threaded through the designated slot in the foot first. Now, the sensor board can be screwed to the leg to finish the assembly.

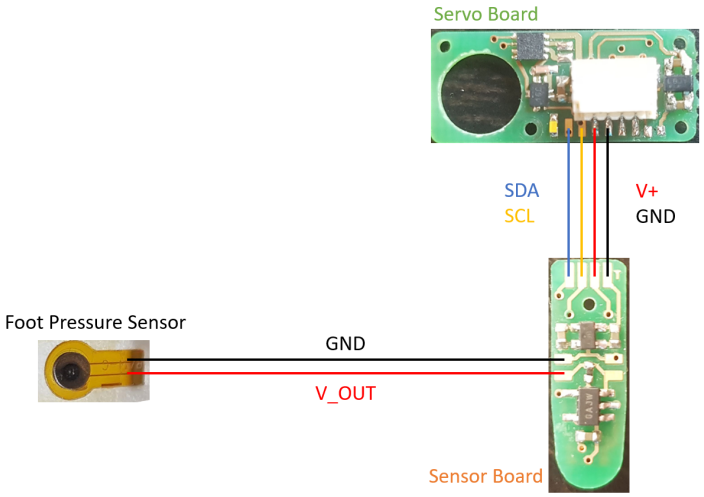

Figure A.13.: Soldering tracks

B. Masters Thesis during COVID-19 (Coronavirus SARS-CoV-2)

Working from Home with Hardware

Starting with my Master's Thesis in March 2020, the official registration was shortly submitted before the first consequences caused by the new corona virus coming to Europe. Afterwards, working from home with not only software, but also hardware involved during a period of several weeks of lockdown is quite challenging as university was closed, no meetings in-person were allowed and most online shops needed a lot more time to ship orders. In contrast to the usual procedure, the first weeks were nearly only used to gather all needed tools together to build a small lab at home and to get to know the robot in detail without hands on tips from the supervisors. Oscilloscope, soldering iron, replacement boards / sensors / electrical parts as well as simple things like wires and resistors had all to be somehow organized and were either borrowed from the university or companies, bought online or mailed from KTH by Jörg Conradt. In addition to that, the advisors had to deal with a new situation as well and helped a lot with creative ideas and lots of remote bug fixing sessions, which required deep understanding of general electronics (Alex) and detailed knowledge of the mouse (Peer).

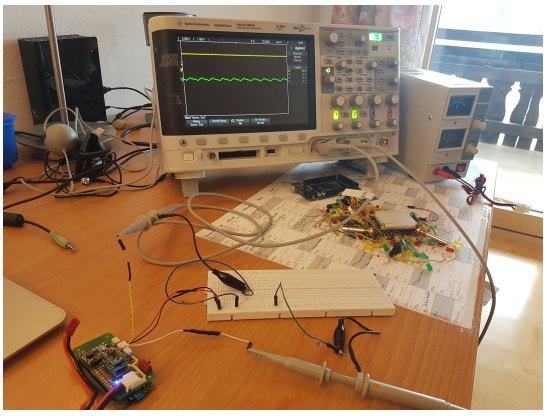

Figure B.1.: First desk setup with oscilloscope

Figure B.2.: Soldering setup in the basement

Figure B.3.: Second desk setup

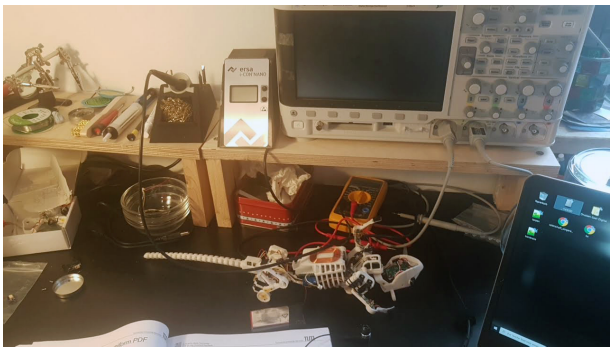

Figure B.4.: Most recent desk setup

Weekly Meetings

Not only the independent work by myself, but also the routine for an informative and productive weekly meeting had to be developed. After testing several options for remote meetings like Skype, BBB, dfn, Zoom, Jitsi Meet and structuring the weekly progress, my questions and the workload for the upcoming week, a well working solution was found, which could apart from helping with strange hardware problems nearly replace in-person meetings.

Figure B.5.: Improvised webcam from mouse spares

C. Data Storage Medium

C.1. Thesis

C.2. 3D Shoe Files

C.3. Source Code

C.3.1. Servo Boards

C.3.2. Spine Board

C.3.3. Mouse Control Software

List of Figures

List of Tables

Bibliography

[1] P. Lucas. "Design, Construction and Validation of a Life-Size Robot Model for Biomimetic Locomotion in Small Mammals". MA thesis. TUM, 2017.

[2] T. Prescott, N. Lepora, P. Verschure, and M. Szollosy. *Living machines: A handbook of research in biomimetic and biohybrid systems*. Jan. 2018. DOI: 10.1093/oso/9780199674923.001.0001.

[3] J. F. Vincent, O. A. Bogatyreva, N. R. Bogatyrev, A. Bowyer, and A.-K. Pahl. "Biomimetics: its practice and theory". In: *Journal of The Royal Society Interface* 3.9 (2006), pp. 471–482. DOI: 10.1098/rsif.2006.0127. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2006.0127. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2006.0127.

[4] J. Vincent. "Biomimetics–a review". In: *Proceedings of the Institution of Mechanical Engineers. Part H, Journal of engineering in medicine* 223 (Nov. 2009), pp. 919–39. DOI: 10.1243/09544119JEIM561.

[5] Y. Bar-Cohen. "Biomimetics: mimicking and inspired-by biology". In: *SPIE Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring*. 2005.

[6] B. Dynamics. *Spot*. 11.08.2020. URL: https://www.bostondynamics.com/spot.

[7] E. M. Siehmann. "Design, Construction and Validation of a Life-Size Robot Model for Biomimetic Locomotion in Small Mammals". MA thesis. TUM, 2017.

[8] T. Krieger. "Design of an Actuated Mechanical Vertebral Column Model for a Biomimetic Mouse Robot". MA thesis. TUM, 2017.

[9] P. Lucas, F. Walter, and A. Knoll. *Design of a Biomimetic Rodent Robot*. Tech. rep. TUM-I1869. TUM Chair of Robotics, Artificial Intelligence and Real-Time Systems, 2018.

[10] P. Lucas, S. Oota, J. Conradt, and A. Knoll. "Development of the Neurorobotic Mouse". In: *Proceedings IEEE International Conference on Cyborgs and Bionic Systems*. Munich, Germany, 2019.

[11] P. Lucas, F. O. Morin, J. Conradt, and A. Knoll. *Neurorobotic Mouse (NeRmo) V4.1*. Tech. rep. TUM-I2081. 2020.

[12] E. C. Collins, T. J. Prescott, B. Mitchinson, and S. Conran. "MIRO: A Versatile Biomimetic Edutainment Robot". In: *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*. ACE '15. Iskandar, Malaysia: Association for Computing Machinery, 2015. ISBN: 9781450338523. DOI: 10.1145/2832932.2832978. URL: https://doi-org.eaccess.ub.tum.de/10.1145/2832932.2832978.

[13] T. J. Prescott, B. Mitchinson, and S. Conran. "MiRo: An Animal-like Companion Robot with a Biomimetic Brain-Based Control System". In: *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '17. Vienna, Austria: Association for Computing Machinery, 2017, pp. 50–51. ISBN: 9781450348850. DOI: 10.1145/3029798.3036660. URL: https://doi-org.eaccess.ub.tum.de/10.1145/3029798.3036660.

[14] N. Lepora, M. Pearson, and L. Cramphorn. "Active Touch with a Biomimetic 3D-printed Whiskered Robot". English. In: *Conference on Biomimetic and Biohybrid Systems*. Lecture Notes in Computer Science. Germany: Springer Verlag, July 2018, pp. 263–275. ISBN: 9783319959719. DOI: 10.1007/978-3-319-95972-6_28.

[15] M. Evans, M. Pearson, N. Lepora, T. Prescott, and C. Fox. "Whiskered texture classification with uncertain contact pose geometry". English. In: *IEEE International Conference on Intelligent Robots and Systems*. 25th IEEE/RSJ International Conference on Robotics and Intelligent Systems, IROS 2012 ; Conference date: 07-10-2012 Through 12-10-2012. 2012, pp. 7–13. ISBN: 9781467317375. DOI: 10.1109/IROS.2012.6385634.

[16] J. Shan, T. Mei, L. Ni, S. Chen, and J. Chu. "Fabrication and Adhesive Force Analysis of Biomimetic Gecko Foot-Hair Array". In: *2006 1st IEEE International Conference on Nano/Micro Engineered and Molecular Systems*. 2006, pp. 1546–1549.

[17] A. Fukuhara, Y. Masuda, M. Gunji, K. Tadakuma, and A. Ishiguro. "Development of Quadruped Robot That Can Exploit Shoulder Hammock Structure". In: *2020 IEEE/SICE International Symposium on System Integration (SII)*. 2020, pp. 1139–1143.

[18] Y. Li, B. Li, J. Ruan, and X. Rong. "Research of mammal bionic quadruped robots: A review". In: *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*. 2011, pp. 166–171.

[19] H.-W. Park and S. Kim. "The MIT Cheetah, an Electrically-Powered Quadrupedal Robot for High-speed Running". In: *Journal of the Robotics Society of Japan* 32.4 (2014), pp. 323–328. DOI: 10.7210/jrsj.32.323.

[20] H. Ishii, Y. Masuda, S. Miyagishima, S. Fumino, A. Takanishi, C. Laschi, B. Mazzolai, V. Mattoli, and P. Dario. "Design and development of biomimetic quadruped robot for behavior studies of rats and mice". In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2009, pp. 7192–7195.

[21] A. Takanishi, T. Aoki, M. Ito, Y. Ohkawa, and J. Yamaguchi. "Interaction between creature and robot: development of an experiment system for rat and rat robot interaction". In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. Vol. 3. 1998, 1975–1980 vol.3.

[22] H.-W. Park, P. Wensing, and S. Kim. "High-speed bounding with the MIT Cheetah 2: Control design and experiments". In: *The International Journal of Robotics Research* 36 (Mar. 2017), p. 027836491769424. DOI: 10.1177/0278364917694244.

[23] B. J. Katz, J. D. Carlo, and S. Kim. "Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 6295–6301.

[24] LEANSTAR. *Datasheet: Flexible Thin Film Pressure Sensors SI4-G*. Rev. 2.2 - August 2019.

[25] N. Semiconductors. *UM10204 I2C-bus specification and user manual*. Rev. 6 — 4 April 2014. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[26] STM. *RM0377 Reference Manual STM32L0x1*. Rev. 8 — December 2017. URL: https://www.st.com/resource/en/reference_manual/dm00108282-ultralowpower-stm32l0x1-advanced-armbased-32bit-mcus-stmicroelectronics.pdf.

[27] T. Instruments. *Universal Asynchronous Receiver/Transmitter (UART) User Guide*. Novemeber 2010. URL: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf.

[28] P. Lucas. *nermo_robot GitHub Repository*. 08.04.2020. URL: https://github.com/Luchta/nermo_robot.

[29] P. Lucas. *nermo_code GitHub Repository*. 08.04.2020. URL: https://github.com/Luchta/nermo_code.

[30] J. Lunze. *Regelungstechnik 1*. Jan. 2016. DOI: 10.1007/978-3-662-52678-1.

[31] J. Lunze. *Regelungstechnik 2*. Jan. 2016. DOI: 10.1007/978-3-662-52676-7.

[32] B. Sciliano and O. Khatib. "Handbook of Robotics". In: Aug. 2020.