Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Gretchen - a humanoid robot for research and education

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Anastasia Prisacaru
geboren am: 18.10.1996
geboren in: Chisinau

Gutachter/innen: Prof. Dr. Verena V. Hafner
Prof. Dr. Hans-Dieter Burkhard

eingereicht am: ..... verteidigt am: .....

Content

1 Introduction

Robots that resemble the human body, also called humanoid robots, increased in popularity in the last few decades due to advances in mechanical engineering, artificial intelligence, and robotics. Humanoid robots are currently used in public relations, personal assistance, healthcare, education, and entertainment where the human-like appearance enables a more natural and effective interaction with humans [1]. The shared morphology with humans also enables countless research opportunities, such as the exploration of theories about human cognition, social interactions, behaviors, and embodied intelligence [2]. Furthermore, humanoid robots provide a versatile platform for education, due to many disciplines which can be learnt in their context, namely: mechanical engineering, electrical engineering and computer programming.

Humanoid robots can be categorized into robots which model particular parts of the body, such as upper or lower body; robots which have a torso, a head, two arms, and two legs, however still having a machine-like appearance; and robots which aesthetically resemble humans - also called androids. In this thesis, we mention humanoid robots with machine-like appearance used for research and education, which resemble either the lower part or the whole human body.

With the increasing popularity of research on humanoid robotics, many companies started developing humanoid robots, the leading manufacturers being SoftBank, ROBOTIS, Kawada Robotics and UBTECH Robotics. They provide robust and versatile robots, with full support and warranty. However, when used in research and education, all commercially available robots have certain disadvantages or limitations, which either depend on their size, actuators, price or software availability. For instance, the well-known adult-sized robots ASIMO [3], Hubo [4] and HRP4 [5] are not designed to survive falls and can only act in controlled environments [6]. Furthermore, these robots have extremely high acquisition and maintenance costs, which means that they are unreachable for most universities and thus unavailable for many research efforts [6]. The robots LOLA [7] and Atlas [8] have more complex locomotion capabilities, however, they are not commercially available and their software and hardware specifications are not publicly available. Similarly to the robots LOLA and Atlas, the majority of commercially available research humanoid robots, such as NAO [9], Pepper [10], Hubo [4], HRP4 [5] and Romeo [11], are not open-platform, meaning that the manufacturers don't allow direct access to the source code and the hardware of the robot, and the user can only program it via a high-level Application Programming Interface (API). These limitations slow down the progress in the humanoid robotics field, as open-platform and low-cost humanoid robots would allow more users to access the low-level functionality

of the robot, modify its hardware and software, and as a result - contribute to the development of the robot as well as to the open-source community.

There are multiple communities of researchers, students, and hobbyists engaged in building and programming open-platform and low-cost humanoid robots who are collaborating and sharing knowledge within initiatives, such as the Open Automation Project [12], RoboCup [13] and Robo One [14]. The Humanoid League of the RoboCup Competition, which was initiated in 2002, is a good example of a steadily growing community, where some of the best self-built and commercial autonomous humanoid robots in the world compete against each other [15]. Until 2012, the majority of teams in this league were competing with self-built robots, as there were no affordable, open-platform commercial robots available. Due to the introduction of open-platform robots DARwIn-OP [16], NimbRo-OP [6] and THOR-OP[1] [17], more teams were able to join this league, without the need of building their own robots and being able to focus on the development of the software. Moreover, some universities were inspired by the design of the DARwIn-OP robot and developed the open-platform robots Igus [18] and Hambot [19] which are bigger in size and the latter is cheaper than the original. These robots have had a great impact on the humanoid league, however they are not completely open-hardware, as they are powered by Dynamixel servo motors, which have certain restrictions of third party manufacturers [20]. Therefore, there is a rising interest within research and maker communities to develop open-hardware electronic components, as they aim to have full control over the software as well as the hardware of the robots.

The Gretchen robot was designed with the aim of creating an accessible, low-cost and open-platform robot for research and education. Gretchen resembles the lower part of the human body and is built from low-cost materials, namely wooden, 3D-printed, and commercially available electronic components. The robot is powered by 10 low-cost servo motors with custom-manufactured open-source boards, which complement the motors with additional features necessary for powering a humanoid robot. The complete source code, as well as all hardware specifications and sketches are open to the public [2]. The software is separated into two different repositories, which provide the code base and necessary documentation for getting started with programming the motors. The first repository, called Sensorimotor, contains the firmware for the custom boards and the hardware specifications. The second one, Libsensorimotor, contains the C++ libraries and the Python API for controlling single motors. Gretchen is a prototype in an early development stage and it still lacks detailed documentation, customized software, autonomy as well as robustness and reliability tests. The goal of this thesis is to analyze the hardware and the software of the Gretchen robot by performing and documenting an experimental assembly, comparing it with other open-platform humanoid robots, giving an overview of the software architecture, and evaluating its accessibility for research and education projects.

[1] THOR-OP is also known as THORMANG1

[2] The official GitHub page of the Gretchen robot `https://github.com/suprememachines`

1.1 Contributions of This Work

This section summarizes the main contributions of this work. A literature survey was conducted, where the current state of the art of the humanoid robotics research field was analyzed. An overview of relevant platforms was compiled, with the focus on their affordability, availability, and whether they have open-source software and hardware. Gretchen was discussed in comparison with five open-platform humanoid robots used for research and education, and thereby its prominent aspects, as well as limitations, were examined.

An experimental assembly of the Gretchen robot was performed, based on a prototype of the assembly kit presented in the documentation [21]. The hardware assembly process was scrutinized and documented together with gathered observations and missing details from the assembly documentation. Therefore, this work can be considered as an extension of the assembly documentation, making the robot more feasible and accessible. In addition, the software architecture was closely examined and described with the focus on its connection to the hardware. Basic experiments with the software were conducted, demonstrating the basic motion capabilities of the robot.

A discussion was conducted about the research domains where the robot can be used. Several of the current limitations of the robot were examined and corresponding solutions for future work were proposed, to encourage the further development of the Gretchen robot within the open-source and research communities.

1.2 Acknowledgements

This work was supported by the AI Brain Company [22], which provided the assembly kit containing all necessary hardware components mentioned in the assembly documentation [21]. Matthias Kubisch is the main contributor to the Gretchen project and he provided support during the experimental assembly. Heinrich Mellmann has performed the first steps of the experimental assembly and has offered his guidance through each stage of the process.

1.3 Outline

This work is structured as follows: In Chapter 2, we discuss the current challenges in humanoid robotics, introduce the Gretchen robot and compare it with other open-platform humanoid robots. In Chapter 3, we provide a detailed overview of all hardware components, together with our observations from the experimental assembly. Chapter 4 is focused on the software architecture, as well as the functionality and control of the motors. We discuss the limitations of the current configuration of the robot, propose solutions for future work, and conclude this thesis in Chapter 5.

2 Background

In this chapter, we analyze the current state of the art of humanoid robotics by giving an overview of several well-known humanoid robots and discussing the current challenges in this research field. Furthermore, we describe the Gretchen robot as well as five other open-platform humanoid robots deployed for research and education and put Gretchen into perspective with them. Finally, we discuss the dissemination of the Gretchen robot.

2.1 Current State of the Art of Humanoid Robotics

With the rising popularity of humanoid robotics research in the last two decades, many companies started to develop humanoid robots, the most prominent models being Honda's ASIMO [3], Atlas [23] developed by Boston Dynamics, HRP[2] [5] series developed by AIST[3] and produced by Kawada Industries, the Toyota Partner Robot [24], Hubo [4], developed at KAIST[4], iCub [25] built by IIT[5], DARwIn-OP [16] manufactured by ROBOTIS and NAO [9] currently manufactured by SoftBank Robotics. These and several other humanoid robots, 19 in total, are included in Table 2.1, showing which humanoid robots are currently commercially available and whether they are open-platform. Additionally, we included the prices of the robots where possible, as some companies and universities don't make this information public. For instance, we have only included the summed up price of the actuators of the robots CHARLI[6] [26], Thormang3 [27] and LOLA [7], as their price is not publicly available. Moreover, the robots in Table 2.1 are divided into two categories: those which are relatively affordable and widely used in research and education, and more complex and expensive robots, which are highly specialized for certain areas of research. For instance, the robots iCub [25], CHARLI [26] and RobotThespian [28] are mainly used in the research of cognitive robotics and real-time interaction with humans; the robots THORMANG [27], LOLA [7] and Atlas [8] are mainly intended for the research of locomotion and emergency response applications; ASIMO [3], Romeo [11], HRP-4 [5] and Reem-C [29] were deployed for researching human-robot interaction and assistance. All these robots fulfill distinct purposes and have certain advantages in comparison with the other ones,

[2]Humanoid Robotics Project - a project for the development of general domestic helper robots in Japan

[3]Japanese National Institute of Advanced Industrial Science and Technology

[4]Korea Advanced Institute of Science and Technology

[5]Italian Institute of Technology

[6]Cognitive Humanoid Autonomous Robot with Learning Intelligence

Robot	Manufacturer	Open source Software	Open source Hardware	Can be purchased	Price (k€)	Price Source
Affordable robots for research and education						
Gretchen	AI Brain	x	x		1.4	
Nao	SoftBank			x	7	[30]
Hambot	Uni Hamburg	x	x		7.5	[19]
Poppy	INRIA	x	x	x	7.5	[31]
DARwIn-OP	Robotis	x	x	x	12	[32]
Igus	Uni Bonn	x	x	x	16	[18]
NimbRo-OP	Uni Bonn	x	x		20	[33]
Highly specialized research robots						
CHARLI	RoMeLa				>5[1]	
LOLA	TU Munich	x			>20[2]	
Thormang3	Robotis	x	x	x	>64[1]	
RoboThespian	Engineered Arts			x	55	[28]
HOAP	Fujitsu				150[3]	[15]
Romeo	Aldebaran Robotics			x	250	[11]
HRP-4	Kawada Industries			x	300	[5]
iCub	IIT	x	x	x	300	[34]
Reem-C	PAL Robotics	x		x	350	[29]
DRC-HUBO	KAIST			x	500	[4]
Atlas	Boston Dynamics				>1.000	[23]
ASIMO	Honda				2.500[3]	[35]

Table 2.1: Overview of 19 well-known humanoid robots, focusing on which of them are currently commercially available, whether they are open-platform and how much they cost.

[1]Exact data is not available; only the price of the Dynamixel Servos was taken in account `http://www.robotis.us/dynamixel/`

[2]Exact data is not available; only the price of the servo motors was taken in account `http://www.parkermotion.com/products/Rotary_Servo_Motors____30_32_80_567_29.html`

[3]is not manufactured anymore

however, all of them have certain limitations which either depend on their size, price, actuators, or software availability. Please note that some humanoid robots were not included in the table due to the lack of publicly available information or technical data (e.g. Partner Robot [24], Bruno [36] and Yanshee [37]). In the following, paragraphs we analyze the current challenges and limitations, as well as the prominent aspects of the humanoid robots from Table 2.1.

Small robots such as NAO and DARwIn-OP, with a body height of approximately half a meter, are limited by their size. For instance, when conducting research on the interactions between these robots and their environments, miniature experimental setups are required [6] and the robots can be hardly deployed for real-world use cases. Nevertheless, small robots have the following advantages: they are cheaper and easier to repair than big robots, and they don't suffer any destructive damage when falling down during locomotion. Adult-sized robots, such as ASIMO, HRP and Hubo are not designed to survive falls or to move freely in unstructured environments [6]. However, the adult-sized robot Atlas [8] is extremely robust, being able to fall, roll and move on uneven terrains such as snow and woods. Unfortunately, Atlas is not commercially available and its price is estimated to exceed one million euro [23]. Therefore, more research is required for developing adult-sized robots, which would be robust, agile, and at the same time accessible for more research efforts.

As mentioned before, the robots listed in Table 2.1 have either partially or wholly publicly available data about their hardware specifications, software and price. Therefore, we have included all open-source robots we have found during our research, as they were easy to find due to open-source projects available on GitHub or on the pages of robot competitions such as RoboCup [15] and the DARPA Robotic Challenge[1]. We can observe that more than half of the robots included in the table are not open-source meaning that the manufacturers provide the users with a high-level API for programming and customizing the behavior of the robot, however, the user is granted only limited access to the software and cannot perform any hardware modifications. On one hand, limited access is beneficial for both the manufacturer and the user, as the robot cannot be easily damaged, in case an inexperienced user makes something wrong. Moreover, high-level APIs often make the software user-friendly and easy to comprehend, freeing the user from the necessity of working with the complex low-level functionality. On the other hand, once the robot is used for research and education, full control over the robot is an essential feature, as it allows researchers and students to perform low-level experiments and explore, test, and modify both the hardware and the software. Furthermore, an open-platform robot benefits from the efforts of the open-source and maker communities, which contribute to further development as well as to the popularity of the robot.

There are only a few commercially available open-platform humanoid robots which are commonly used for research and education, the most known ones being: DARwIn-OP [16], Poppy [38], Igus [18], THORMANG3 [27] and iCub [25]. The DARwIn-OP robot

[1]The official website of the DARPA Robotic Challenge: `https://www.darpa.mil/`

Servo	Robot	Operating voltage range	Stall torque range	Size range (mm)	Weight range (g)	Price range (€)
Hitec with Sensorimotor board	Gretchen	6V - 12V	2.5Nm-≈5Nm	66x30x60	200	80
Dynamixel AX Series	Poppy	9V - 12V	1.5Nm-1.8Nm	32x50x40	53-55	40-90
Dynamixel MX Series	DARwIn-OP Igus, Hambot, Poppy, Nimbro-OP	11.1V-14.8V	2.3Nm-10Nm	36x51x36	67-72	220-450
Dynamixel PH Series	Thormang3	24V	5.1Nm - 44.7Nm	42x84X42-54X126X54	340-855	1,390-2,590
Kollmorgen BLDC RBE Series	iCub	28V - 70V	11Nm-40Nm	Motor: diam.:15 len.: 26	369-936	>1400
Faulhaber BDC 1319 Series	iCub	6V - 24V	0,45Nm-0,65Nm	Motor: diam.: 13 len.: 19.2	12	≈110

Table 2.2: Comparison of the servo motors used for powering open-platform robots, with the focus on their main performance features, physical characteristics and price.

had a significant impact on the development of other open-platform robots which were inspired by its design. For instance, the Igus [18] and Hambot [19] robots were developed by the Universities of Bonn and Hamburg respectively, by using DARwIn-OP's design and actuators as a base. Besides iCub, all open-platform robots listed in Table 2.1 use Dynamixel servos as actuators, which are widely used in research and industry due to their compact size, high performance, versatility and robustness. However, Dynamixel servos have certain restrictions, which make them not entirely open-source due to their proprietary nature [20]. The actuators powering the iCub robot were assembled from separately ordered components, mainly from the Kollmorgen and the Faulhaber manufacturers. By this means, they could integrate the actuators more easily in the endoskeletal structure of the robot and keep its hardware open-platform. Table 2.2 gives an overview of the servo motors used for the open-platform robots from Table 2.1. As we can see in this table, the servos from the Dynamixel MX series are the most commonly used servos for open-platform robots, due to its optimal size and high performance. However, they have a substantial influence on the overall price of the robots, as one motor exceeds the price of a few hundreds of euro. Note that for

the iCub robot, we included only the dimensions and the weight of the motors and not the servos themselves, as this information is not publicly available. The price of the Kollmorgen servo motor series is a rough estimate because we compare it to another commercially available Kollmorgen motor with less torque[1]. The servos used for powering Gretchen have the major advantages of being open-platform, cheap, and fairly easy to assemble. Furthermore, they deliver more torque than the Dynamixel AX and some of the Dynamixel MX Series. The Dynamixel AX and the Faulhaber 1319 series are among the cheapest and the smallest servos, however they deliver less torque. The Dynamixel PH and Kollmorgen RBE series are the most powerful servos, having the highest price as well as weight. Based on Table 2.2, we can draw the conclusion that there is always a trade-off between the performance, size, and the price of the servo. Therefore, the servo motors of the Gretchen robot fulfill an optimal trade-off, due to the combination of a low-cost servo and an open-source board.

Besides the small variety of open-platform robots available on the market, these robots are expensive, with the price ranging between seven and hundreds of thousands of euro. Certainly, the price is a relative measurement and in the context of research projects in universities that receive funding, 12,000€ for a DARwIn-OP might not seem expensive. Nevertheless, when discussing academic research and education in universities worldwide or in the context of maker communities, a DARwIn-OP robot might be hardly affordable. The benefit of open-hardware robots is that they can be manufactured by 3D-printing the CAD files available online and using commercially available materials and electronic components, thereby drastically lowering the price of the robot. For instance, the price of a DARwIn-OP robot can be reduced by 50% when manufacturing it from scratch [39].

All the limitations of humanoid robots listed above can be minimized by involving more research efforts into these issues. Therefore, affordable and open-platform robots have the potential of engaging more communities, researchers, and students to collaborate and improve the software and the hardware of humanoid robots. Gretchen is a low-cost and open-platform robot at an early stage in prototype development, which is still hardly comparable with the robots mentioned above, however, its main purpose is to make education and research in humanoid robotics more accessible and affordable. In the following sections, we introduce the Gretchen robot and put it into perspective with five open-platform humanoid robots currently used for research and education.

2.2 The Gretchen Robot

The Gretchen robot was designed and developed by Matthias Kubisch in collaboration with the AI Brain Company in 2018. One of the prototypes of the robot is currently used for academic research and education in the lab of the Adaptive Systems Group in

[1] A similar Kollmorgen servo used as a price reference `https://www.maas-automation.de/kollmorgen-akm44g-acc2r-00-servomotor-1-61-kw-unused`. Accessed: 2020-07-10

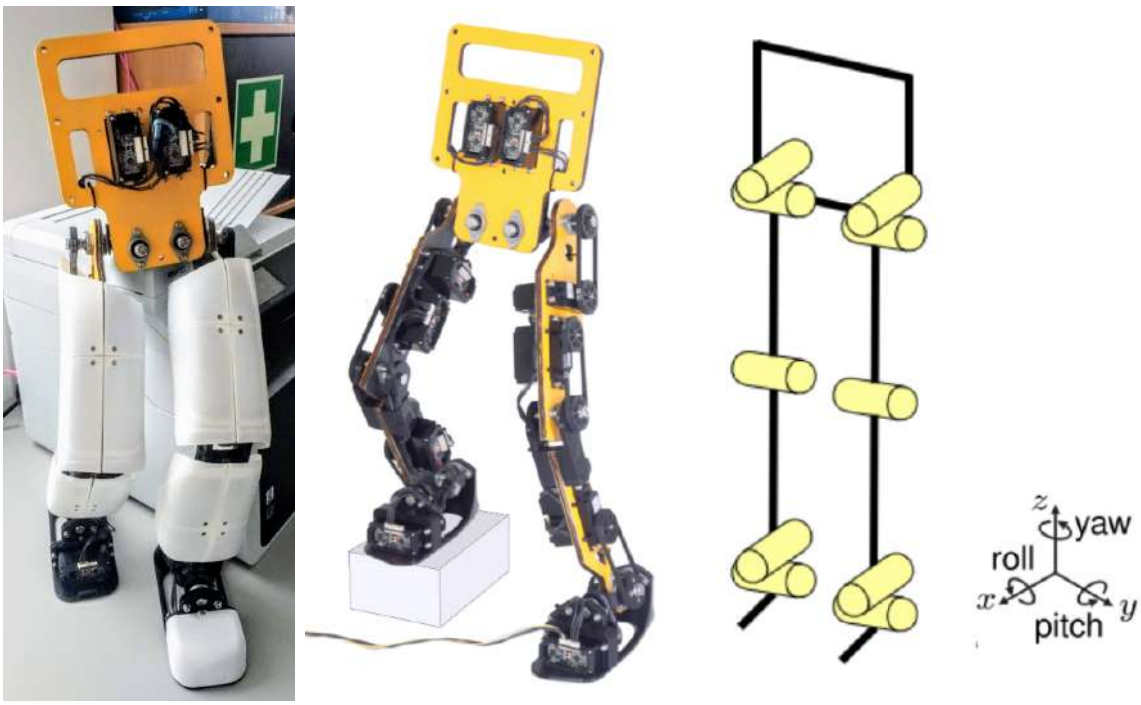

Figure 2.1: The first two pictures show the fully-assembled Gretchen robot, with and without protective 3D-printed shells. On the right side - the kinematic structure is illustrated with all 10 degrees of freedom. The photos of the Gretchen robot were taken from the assembly documentation [40].

the Computer Science department of the Humboldt University. The Gretchen robot is a versatile platform for education, as its modular design allows working and experimenting by its hardware design, actuators, 3D-printed components, firmware, power supply, communication bus, software libraries or the API.

Gretchen is 74 cm high, weighs approximately 5 kg, and has 10 degrees of freedom (DoFs), 5 of which are in each leg: 2 in the hip (roll and pitch), 1 in the knee (pitch), and 2 in the ankle (roll and pitch). The robot resembles the lower part of the human body, by incorporating feet, legs and lower torso as shown in Figure 2.1. The body parts are made from low-cost materials such as wood and 3D-printed components, the CAD files of which can be found on GitHub [21]. The robot is powered by 10 low-cost Hitec HS-805MG Mega servo motors, the distinguishing features of which are the custom Sensorimotor boards. These boards were designed by the creators of Gretchen, aiming to add more features, namely, more torque, sensory data, bus communication and customizable firmware to simple and low-cost servo motors, thereby drastically lowering the overall price of the robot. The servos are placed either above or below the robot's joints and are connected to them with toothed belts. These belts ensure an indirect transmission between the joints and the motors, thereby making the connection between them more elastic and protecting the motors from the direct damage of heavy loads. In the left picture of Figure 2.1 the 3D printed shells are mounted on the feet,

iCub[41] 0,9m | DARwIn-OP[42] 0,45m | Poppy[43] 0,84m | Igus[18] 0,92m | Hambot[19] 0,87m

Figure 2.2: Open-platform humanoid robots used for research and education, with which the Gretchen robot is compared. The names and the corresponding heights of the robots are listed under the pictures. The sources of the photos are included next to the names of the robots.

legs and hips, in order to protect the motors, cables and belt system from collisions. The actuators are communicating with each other via the RS485 serial bus and with the computer - via a USB-to-Serial adapter. The available C++ libraries can be compiled on a Linux machine and the robot can be further developed through a Python API. The motion of the robot can be programmed using both position and torque control. The Libsensorimotor library offers an implementation of a PID controller, which is a position control mechanism.

The robot cannot be purchased, however, it can be manufactured from 3D-printed as well as commercially available materials and electronic components. The Sensorimotor boards are the only components that require custom manufacturing and they can be ordered by using the provided schematic circuits. Gretchen's overall manufacturing costs lie at roughly 1,400€.

2.3 Open-platform robots

In this section we examine the following robots, shown in Figure 2.2: iCub [25], DARwIn-OP [16], Poppy [38], Igus [18] and Hambot [19]. These robots were chosen for the comparison with Gretchen, as they are open-platform and are mainly used by universities for research and education. The iCub robot is hardly comparable with the other open-platform robots, due to its cognitive abilities as well as its price. However, it is the only open-platform robot that doesn't use Dynamixel actuators and is therefore interesting for us to investigate. The source code, circuit diagrams, mechanical CAD

files, and documentation of all five robots are available to the public, meaning that they can be built from 3D-printed and commercially available materials.

In the following subsections, we examine the main aspects of each robot from Figure 2.2, such as physical description, areas of use, degrees of freedom, actuators and software architecture, and finally summarize them in Table 2.3. We included the robots LOLA [7] and Reem-C [29] to the bottom of Table 2.3, as they are open-source and the information about their hardware is partially available to the public. The robots NimbRo-OP [6] and Thormang3 [27] can be considered as the "big brothers" of the robots Igus and DARwIn-OP respectively, as they originate from the same manufacturers and their architectures are fairly similar. Therefore, they can be found in Table 2.3, however, we didn't describe them in a separate subsection.

2.3.1 iCub

The Robot iCub Project [25] was initiated in 2006 at the Italian Institute of Technology (IIT) with the goal of realizing an embodied robotic child with the physical and cognitive abilities of a 2.5-year-old human infant. According to the IIT official website[1], iCub is currently used in more than 30 laboratories worldwide for the research in embodied cognition and development of cognitive capabilities through interaction with the environment and humans. Some of the capabilities of the robot are: crawling, solving complex 3D-mazes, expressing emotions through facial expressions, grasping small objects, learning archery, and avoiding collisions within non-static environments.

The body structure corresponds to the real size and proportions of a 2.5-year-old child: it is 90cm high, weighs 23kg, and has 53 DoFs, 6 of which are in each leg: 3 - hips, 1 - knee and 2 - ankle. The body parts are mainly made of aluminium alloy and the joints are fabricated from stainless steel. The actuation of the iCub robot is a combination of harmonic drives, brushed DC motors from Faulhaber and brushless frameless motors from the Kollmorgen frameless RBE series. The major components of the actuators, namely, motor, rotor, stator, encoder and gearbox, were ordered separately and integrated within the endoskeletal structure without frames. The robot doesn't have an on-board battery, so it should be powered through a cable by an external power supply and it receives sensory data and motor commands via a gigabit Ethernet cable from a remote computer. The embedded PC-104 based CPU and several digital signal processors (DSPs) generate motor control signals and acquire sensory data. The middleware is based on the open-source architectural system called Yet Another Robot Platform (YARP) written in C++, and the communication between the servos and the computer is managed by a PC104 card via the CAN bus. The software can be compiled on Linux, Mac OS, Windows and Solaris operating systems and it can be downloaded together with the documentation, hardware specifications, and mechanical drawings from the official website.

[1] `http://www.icub.org/` – official website for humanoid robot iCub

2.3.2 DARWIN-OP

The Dynamic Anthropomorphic Robot with Intelligence–Open Platform (DARWIN-OP) [16] was manufactured by the Korean company Robotis in collaboration with Virginia Tech[1] in 2010 with the goal to provide an affordable and versatile humanoid robot for research, education and outreach activities. The robot is able to perform a wide range of activities such as riding a bicycle, roller skating, dancing, playing golf, playing ice hockey, and climbing a ladder. Furthermore, DARwIn-OP has participated and competed in the FIRA[2], IEEE ICRA[3] and RoboCup [15] competitions.

It is 0,45m high, weighs 2,8kg and has 20 DoFs, 6 of which are in each leg: 3 - hip, 1 - knee and 2 - ankle. Darwin-OP has 2 embedded controllers, the main one being an Intel Atom Z530 @1.6GHz CPU and as a sub-controller, the ARM 32-bit Cortex-M3 72MHz CPU is used. The sub-controller communicates with the actuators, controls the on-board LEDs and sensors, is connected to the power supply and communicates everything to the main controller via an USB connection. The Dynamixel MX-28T servo motors are powering the robot and communicate with each other and the sub-controller via the Dynamixel serial bus. The position and speed of the servos is controlled by the PID controller integrated in the MX-28 servos. The middleware is a C++ API which can be compiled on Linux and programmed in Java using the open-DARwIn Software Development Kit.

2.3.3 Poppy

The Poppy robot [31] was designed in the French Institute for Research in Computer Science and Automation (INRIA) in 2013, with the goal to realize a robust, easily reproducible and affordable humanoid robot for the research of biped locomotion and compliant physical interaction. Poppy has a bio-inspired morphology and it can easily bend, perform artistic moves, walk, roll and crawl.

Poppy is 0,84m high, weighs 3,5kg, and has 25 DoFs, 6 of which are in each leg: 3 - hip, 1 - knee and 2 - ankle. Poppy is powered by 25 Dynamixel motors 21 MX-28, 2 MX-64 and 2 AX-12. The motors can be controlled by the internal PID or by torque control. The INRIA team developed their own middleware PyPot[4] - a library written in Python which makes it easy to control robots based on Dynamixel motors. The servo motors are communicating between each other via the serial Dynamixel bus and are controlled from a remote computer via two USB2AX dongles, one for the upper and one for the lower body. The only embedded devices are two Arduino nano boards which compute the sensor data from the feet.

[1] Virginia Polytechnic Institute and State University
[2] The Federation of International Robot-soccer Association: `http://www.firaworldcup.org/`
[3] International Conference on Robotics and Automation: `https://www.ieee-ras.org/`
[4] Pypot repository on GitHub: `https://github.com/poppy-project/pypot`

2.3.4 Igus

The Igus robot [18] was designed at the University of Bonn in 2015, with the goal to realize a robust, affordable, versatile, and customisable open standard platform for humanoid robots in the child-sized range. According to the official website[1], the Igus robot has been successfully participating in the teen- and kid-sized classes of the Humanoid League of RoboCup since 2015 and has won several awards.

Igus is 0,9m high, weighs 6.6kg, and has 20 DoFs, 6 of which are in each leg: 3 - hip, 1 - knee and 2 - ankle. It's design was inspired by DARwIn-OP and it is powered by the same Dynamixel actuators: 12 MX-106 servos for the legs and 8 MX-64 servos for the head and arms. The software architecture is similar to the one of DARwIn-OP, which also has two embedded microcontrollers: the main one - Gigabyte Brix GB-BXi7-5500 and a secondary one - STM32F103RE Cortex M3 (CM730). The actuators are communicating between each other and with the CM730 microcontroller via the Dynamixel serial bus. The main controller, which is a Linux machine, communicates with CM730 via a USB connection. The Robot Operating System (ROS) is used as middleware.

2.3.5 Hambot

The Hambot robot [19] was designed at the University of Hamburg in 2014, with the goal to realize an affordable, easy to use, open source humanoid robot which will play soccer in the RoboCup Competition. The team of 5 Hambots has been playing in the kid- and teen-size classes of the Humanoid League of RoboCup since 2014.

Similarly to Igus, Hambot's design was inspired from DARwIn-OP, by using the same actuators and software architecture. The robot is 0,87m high, the weight is unknown (we assume it should be heavier than the DARWIN-OP robot), and has 24 DoFs, 7 of which are in each leg: 3 - hip, 1- knee, 2 - ankle and 1 - toe. The actuators are 25 Dynamixel servos, 20 of which can be reused from a DARwIn-OP with 9 additional MX-64 and MX-106 servos, thereby lowering the production costs. Hambot is completely 3D printable and can be built and assembled using the documentation on GitHub [44]. The servos are communicating via the RS485 serial bus with the Dynamixel Protocol.

2.4 Comparison between Gretchen and Other Robots

All the robots listed above have 6 degrees of freedom in each leg (not including the toes), which are necessary for proper locomotion in different directions. Gretchen lacks the yaw in the hips and as a result, it can move forward and backward, but it is not able to easily turn in other directions. The assembly of the Gretchen robot substantially differs from the other robots, which only require a 3D-printer and commercial servo

[1]The official website of the Nimro team: `http://www.nimbro.net/OP/`

motors and electronic components. Additionally, one needs to order the custom boards, use a laser cutting machine for cutting the wooden body parts and a soldering iron for soldering all electronic components to the custom boards. Gretchen is the only robot with toothed belts, which ensure an indirect transmission between joints and actuators.

Another common feature of the all above-listed robots besides Gretchen is the wireless control, which is only possible with an embedded computer and eventually an on-board source of power supply. The Gretchen robot is currently in a prototype stage and it can be programmed from an external computer via a cable. Gretchen's overall production costs lie under 1.500€, which is a big advantage in comparison to all the humanoid robots listed above. Even though all the robots we've described are open-platform, some of the hardware components have certain restrictions of third party manufacturers, such as the Dynamixel servos and the CM730 motor controller [19]. The actuators of the iCub robot are open-source, however, they consist of separately ordered components, which make the assembly process more complex and time-consuming. Moreover, the Kollmorgen motors that are used for powering the legs of the iCub robot, have the highest price out of all motors listed in Table 2.2. The servo motors with the custom Sensorimotor boards used for powering Gretchen deliver less torque than the Dynamixel motors used for the majority of the above-listed robots, however, they are fully customizable, low-cost and fairly easy to assemble. The boards are suitable for any brushed DC servo motors, which can be supplied with a voltage, ranging between 6V and 12V. Gretchen's electronic components are entirely open-hardware and open-source, which makes it a great platform for research experiments and education.

2.5 Dissemination of the Gretchen Robot

One of the future goals of the Gretchen robot is joining the Humanoid League of RoboCup and driving collaboration with the other teams who have extensive experience in designing and manufacturing humanoid robots. The Gretchen robot has already been demonstrated at two events organized by the RoboCup community.

The first public appearance of the Gretchen robot was at the Robotic Hamburg Open Workshop (RoHOW) [45], to which we traveled with the Berlin United Team from the Humboldt University. The RoHOW is an educational and networking event of the Hamburg University of Technology (TUHH) with students and scientists from all over the world that are active in research on humanoid robots [45]. The researchers and students from other teams showed great interest in the Gretchen robot, especially in its actuators powered by the Sensorimotor boards. In the left photo in Figure 2.3, the Gretchen robot is sitting next to the Hambot robot from the Bit-Bots team from the University of Hamburg [19], which was the only team from the Humanoid League of RoboCup at the RoHOW. We exchanged experience with each other and discussed collaboration ideas and the potential of the Gretchen robot for joining the Humanoid League of RoboCup. The photo on the right in Figure 2.3 shows the members of the Berlin United Team from the Humboldt University who attended the RoHOW, with

Figure 2.3: Left picture: The Gretchen and the Hambot [19] robots sitting side by side at the Robotic Hamburg Open Workshop (RoHOW). Right picture: The members of the Berlin United Team from the Humboldt University who attended the RoHOW, with the Nao and Gretchen robots sitting on the laps of two team members.

the Nao and Gretchen robots sitting on the laps of two team members.

We have also presented the Gretchen robot in a 60-minute talk at the Virtual RoboCup Humanoid Open Workshops (V-RoHOW) [46], together with Heinrich Mellmann and Matthias Kubisch. V-RoHOW is an online workspace for the RoboCup Humanoid community to get together, pitch ideas and socialize [46]. This online event was organized for the first time, due to the cancellation of RoboCup 2020. Our presentation was called 'Gretchen - a Humanoid Open Hardware Platform for Education and Research', where we presented the current state of the Gretchen project, some of the robot's most interesting hardware features, the software architecture, and the Sensorimotor boards. In the 15-minute discussion after our talk, questions about the Sensorimotor boards were asked and discussed, namely about the board microcontroller, serial bus and its cables, and the features of the boards. The talk was recorded and published on YouTube [47].

Robot	Creators	Year of creation	Weight	Size	DOFs	Actuators	Middleware	Bus	Embedded computer
iCub	IIT, Italy	2006	23kg	0,9m	53	Kollmorgen RBE BLDC	YARP	PC104-CAN	PC104 card
DARwIn-OP	Robotis, Korea	2010	2,9kg	0,45m	20	Dynamixel MX-28T	C++ API	Dynamixel Serial Bus	Intel Atom Z530
Poppy	INRIA, France	2013	3,5kg	0,84m	25	Dynamixel MX-28 & AX-12	PyPot	Dynamixel Serial Bus	Arduino
Igus	Uni Bonn, Germany	2015	6,6kg	0,92m	20	Dynamixel MX64 & MX-106	ROS	Dynamixel Serial Bus	GB-BXi7-5500, CM-730
Hambot	UHH, Germany	2014	N/A	0,87m	24	Dynamixel MX-46 & MX-28T	ROS	RS485 Serial Bus	Odroid XU3 lite
Gretchen	AI Brain, Germany	2018	5kg	0,74m	10	Hitec HS-805MG, Sensorimotor Boards	C++ and Python API	RS485 Serial Bus	no
NimbRo-OP	Uni Bonn, Germany	2012	6,6kg	0,95m	20	Dynamixel MX64 & MX-106	ROS	Dynamixel Serial Bus	AMD E-450, CM-730
Thormang3	Robotis, Korea	2016	42kg	1,37m	31	Dynamixel PH42 & PH54	ROS	Dynamixel Serial Bus	Intel Core i7
LOLA	TUM, Germany	2004	55Kg	1,8m	25	PMSM Parker Bayside BLAC	C++ API	SERCOS III	Intel Core 2 T7600
REEM-C	PAL Robotics, Spain	2013	80Kg	1,65m	68	22 BLDC & 14 BDC	ROS	CAN bus	Intel Core i7

Table 2.3: An overview of the most important characteristics of open-platform robots (iCub, DARwIn-OP, Poppy, Igus, Hambot, Gretchen, NimbRo-OP, Thormang3) and robots with open-source software only (LOLA, REEM-C).

3 Gretchen - Hardware

This chapter elaborates on the manufacturing and the assembly process of the Gretchen robot. The goal of this section is to describe the manufacturing stages, explain the importance of certain hardware components, evaluate the accessibility of the robot, and examine the comprehensiveness of the provided assembly documentation.

The manufacturing process of the Gretchen robot can be divided into 3 stages which require different levels of expertise. The first stage includes purchasing all commercially available electronic, wooden, and drive components and ordering the custom boards. The second stage involves manufacturing the body parts by using special equipment such as a laser cutting or milling machine and a 3D-printer. The final stage includes gluing and coloring the wooden parts, soldering the electronic components to the custom boards, assembling the motors, manufacturing the cables, and putting everything together.

In this chapter, we focus on the evaluation of the third manufacturing stage, as we had at our disposal a prototype of the assembly kit containing all required hardware components. During the assembly process, we used the provided assembly documentation [40], which is a step-by-step guide for building the Gretchen robot. As the robot is in its early prototype stage, so is its documentation, which partially lacks details or extensive explanations. Therefore, we communicated with Matthias Kubisch during the assembly process and discussed our progress as well as our challenges. In the following section, we describe each hardware component in detail and complement it with our observations from the experimental assembly and with the information provided by Matthias Kubisch. At the end of this chapter, we summarize our experience of the experimental assembly and assess the level of difficulty of this process from the standpoint of a Bachelor student of Computer Science, with no extensive background in electrical engineering and robotics. Each subsection corresponds to one of the major structural components: wooden parts, 3D-printed parts, actuators and electronic components, cables and wires and the power supply.

3.1 Wooden Parts

The robot's body incorporates feet, shanks, thighs and lower torso, which are shown on the right side of Figure 3.1. Gretchen's legs and lower torso are made of plywood, which is a versatile and low-cost building material made from thin layers of wood veneer glued together [48]. The way plywood is made ensures consistent strength across the entire length of the wood, which makes it resistant to warping, cracking and twisting [48].

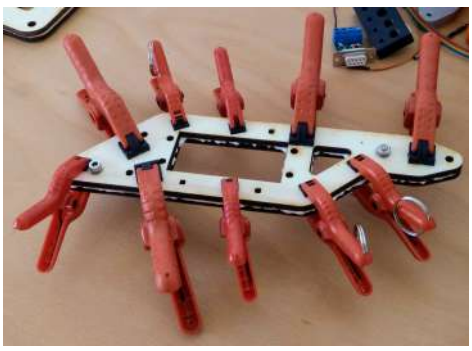


Figure 3.1: Left picture: The process of gluing 2 plywood plates together and using small clamps to apply appropriate pressure distributed on the full surface. Right picture: The wooden body parts drying on paper towels, after being painted with yellow color and covered with clear coating.

Plywood is the optimal choice of material for Gretchen's limbs because 3D-printing parts of this size is time-consuming and they wouldn't fit into a conventional 3D-printer.

The assembly documentation states that there are 10 wooden pieces, which need to be glued together pairwise, in order to achieve the needed thickness of 1cm. The reason for this is that the laser cutting machine used for carving the wooden parts, was able to cut through 5mm thick wood surfaces. One can use 1cm wide plywood plates when having access to a more powerful laser cutting or milling machine. The left photo in Figure 3.1 illustrates two wooden plates glued together using small clamps to apply uniform pressure on the full surface.

Each side of the wooden body parts is painted with a different color and covered with a few layers of clear coating for protection against humidity and fat. Having the back and the front of the robot painted in different colors is not a compulsory step, but it may ease the assembling process and make the robot more appealing. All wooden parts have either one or two rectangular holes for the servomotors, a few round holes for the cable ties and the bearings. The lower torso has 3 additional rectangular holes, which are designed for grabbing and lifting the robot.

3.2 Drive Parts

Drive parts include all moving components which ensure that the body parts and the joints of the robot are able to move or rotate. In this section, we describe the main drive components used for Gretchen: bearings and toothed belts. The assembly documentation only provides information about how to install them. Below we offer a more detailed overview of each drive component.

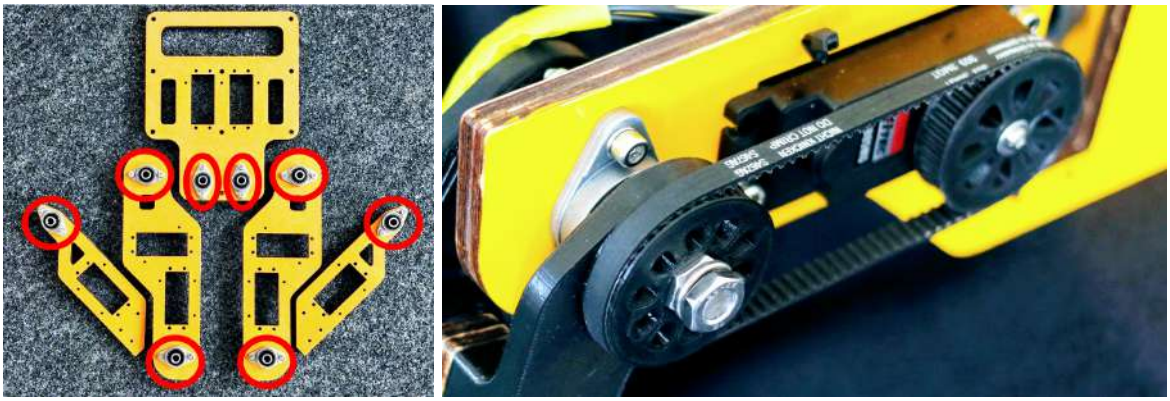

Figure 3.2: On the left side: Bearings mounted on the wooden parts, marked with the red circles. On the right side: A toothed belt connecting a joint and an actuator. Both pictures were taken from the assembly documentation [40].

3.2.1 Bearings

The bearings ensure a smooth connection between the joints and the body plates, due to tiny steel balls inside, which allow the parts to roll around each other with reduced friction. The assembly kit contains 20 KFL08 bearings, with the inside diameter of 8mm and two mounting holes.

The bearings should be connected in pairs on each side of the wooden plates, as well as on the 3D-printed feet frames, as shown in Figure 3.1. It's important to place spring washers between the wood and the steel components, in order to protect the surface from damage, distribute the pressure, and prevent the bolts and nuts from moving or corroding.

3.2.2 Toothed Belts

Toothed belts are flexible bands with teeth carved onto their inner surface, designed to run over matching toothed pulleys. These belts ensure an indirect transmission between the joints and the motors, thereby making the connection between them more elastic and protecting the motors from the direct damage of heavy loads. However, this mechanism has a few disadvantages, such as lack of precision due to the mechanical freedom at the connecting points between the joints and the motors; as well as power loss, due to the motion devices between the motor and the joint which need to be moved [49]. The indirect drive mechanism is rarely used in humanoid robotics and therefore represents an interesting research direction, as its elasticity allows more naturally looking movements of the robot.

The 3D-printed pulleys of the motors and the joints are connected pairwise via 10 GT3 6mm wide toothed belts, eight of which have the pitch length of 303mm and the other two - 174mm. The position of the timing belts is very important because they define

Figure 3.3: The nuts should be glued carefully into the pulleys, so that the bolts are strictly perpendicular to their surface. In this picture the bolt is slightly flexed to the left.

the movement ratio of the servos. Therefore, Matthias Kubisch recorded a helpful video where the installation of the toothed belts is demonstrated, which can be found linked in the assembly documentation [40].

3.3 3D-printed Parts

In this section, we briefly introduce the 3D-printing technique of manufacturing objects and describe the 3D-printed components, namely: pulleys, joints, feet, motor covers and shells.

3D-printing is a low-cost manufacturing process of objects, by applying thin layers of a material, also called filament, onto one another [50]. All open hardware robots come with CAD files containing the 3D models of different components, which can be either 3D-printed or reproduced through other manufacturing techniques. The 3D-printed parts for Gretchen were manufactured in an Ultimaker printer by using ABS plastic and tough PLA filament.

3.3.1 Pulleys

The joints of the robot have pulleys on the outer sides, which are coupled with the servo pulleys with toothed belts. Pulleys are wheels, designed to support the movement and change the direction of the belts along their circumference. The assembly kit contains pulleys that either differ in size or in the shape of the hole in the center. The D-spline motor pulleys from the assembly documentation, 10 in total, refer to the ones which

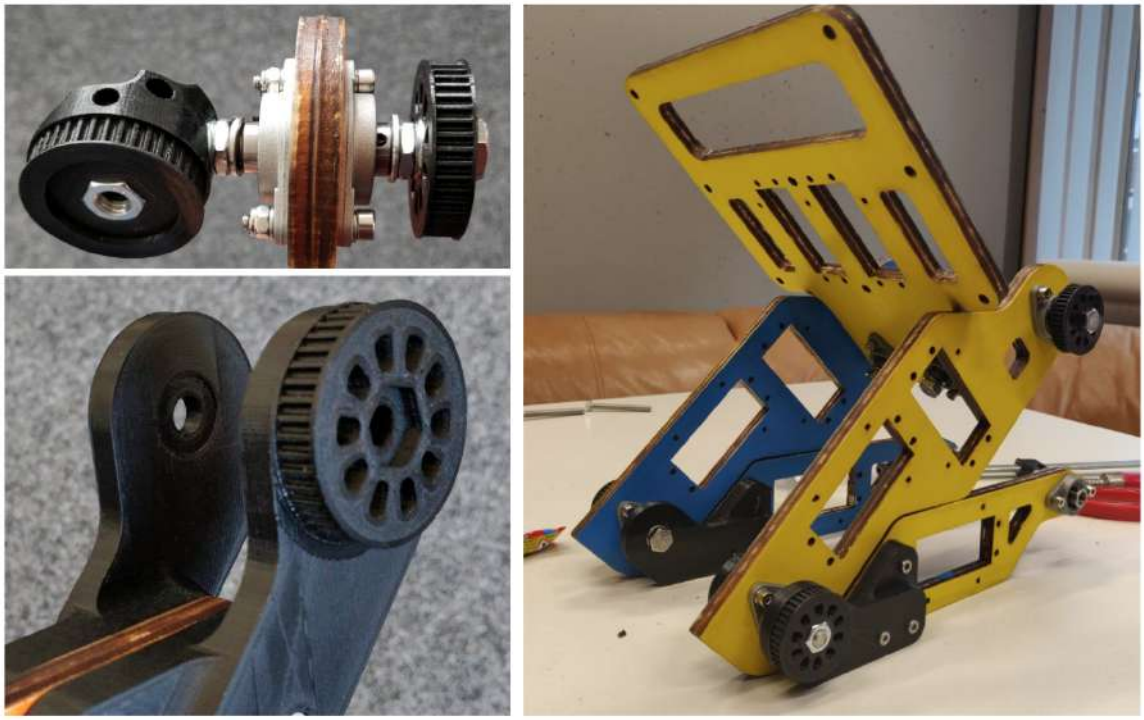

Figure 3.4: Top left: The hip joint. Bottom left: The knee joint. Both photos where taken from the assembly documentation [40]. Right picture: Wooden body parts after connecting the wooden plates with the joints.

have to be installed on the servo shaft instead of the horns (they are a default part of the servo set). The remaining four pulleys are used for the gimbal joint, which will be discussed in the following section. The terms 42T and 33T following the name of the pulleys, represent the number of teeth, carved into the surface of the pulleys.

3.3.2 Joints

Each knee and hip joint consists of two 3D-printed parts, namely the inner and the outer one, which are shown in Figure 3.4. The knee components are symmetrical, the only difference being the integrated pulley on the outer side of the knee. Both parts should be connected through the shank bearing. It's important to use a moderate amount of power when screwing the nuts, in order to prevent the screw from damaging the plastic. The knee of the robot has only one degree of freedom - the pitch.

The hip mechanism consists of a flat pulley on the outer side and an X/Y integrated pulley on the inner side. Before mounting the hips, one should check whether the hex nuts fit inside the pulleys. Depending on the tolerance of the 3D printer, the pocket of the pulley might be a bit loose, which can be solved by using superglue for fixing the nut. In case the pocket is a bit tight, one should use a clamp for adding a little force

Figure 3.5: The first picture illustrates the ankle joint. This photo was taken from the assembly documentation [40]. In the left picture: the foot with the gimbal joint attached.

when inserting the nut inside it. Once the flat hex nut pulleys are done, the bolt must be inserted from the nut side first, in order to ensure continuous threading between the nut and the plastic surface. Before inserting the bolt through the other side, the remaining plastic or sharp edges should be removed, for instance, by using a paper cutter. A 15mm deep hole should be made in the pulley on the inner side of the hip, prior to connecting both pulleys through the hip bearings. This step should not be skipped, as, without it, the bolt would get stuck halfway through the assembly process of the hip joint. When connecting the hip components, a normal washer must be used at the outer side and the spring washer at the inner side.

The ankle joint incorporates three 3D-printed components, namely, a 33 teeth pulley, a 42 teeth pulley, and a cardan gimbal frame. Both pulleys should be attached to the cardan joint, as shown on the left side of Figure 3.5. The small pulley ensures the roll movement of the joint and the bigger one - the pitch movement.

3.3.3 Feet

The feet of the robot are two perfectly identical 3D-printed parts. Before mounting the ankle joint on the foot frame, the toothed belt with the pitch length of 174mm should be inserted around the ankle. Note that in the second picture of Figure 3.5, the toothed belt is missing, meaning that we had to unmount the ankle joint, put the belt around it and mount it again. Gretchen's feet are different from the majority of the other humanoid robots, as they carry one servo motor each. These servos enable the roll movement of each foot, however, they add extra weight to the feet and may, therefore, make walking more challenging and less energy-efficient.

3.3.4 Motor Covers

Since the new boards occupy more space than the old ones, due to the additional electronic components, the old motor cover cannot be used for closing the motor box.

Figure 3.6: Top left picture: The Sensorimotor board situated halfway inside the 3D-printed motor cover. The side hole is slightly shifted to the right and the board doesn't fit inside the motor cover. Bottom left picture: A permanently attached adapter to the wooden part enables a quick assembly and removal of the shells. Right picture: The shells are protecting the thighs, shanks and feet of the robot. The last two pictures were taken from the assembly documentation [40].

The assembly kit contains ten 3D-printed motor covers, which should hold the custom boards and attach them to the motor housing. As shown on the top left part of Figure 3.6, the side opening, designed for the bus connectors, may be slightly shifted (in this example to the right). This shift can result from the error of the 3D-printer. Therefore, in order to insert the board inside the cover, the side opening should be enlarged. We used a drill to make these holes a few mm wider.

3.3.5 Shells

Our assembly kit didn't include the shells, however, they represent a useful hardware component and are therefore worth mentioning. The 3D-printed shells were designed for protecting the electronic components and the belt system from collisions and damage. In the right picture of Figure 3.6 the Gretchen robot is shown, with the shells protecting its thighs, shanks and feet. The shells are mounted on 3D-printed adapters, which are

attached to the wooden parts, as shown in the lower-left picture of Figure 3.6. The adapters allow a quick assembly and removal of the shells. Although the limbs of the robot weren't designed with attachment points for the shells, they have holes for cable management zip-ties, which can also be used for attaching the shell adapters. The shells for the thighs and the shanks consist of multiple 3D-printed pieces, and therefore can be printed by using a conventional 3D-printer.

3.4 Servos and Electronic Components

The assembly kit provides twelve servo motors, ten of which will be powering the robot and the other two are supplementary. Before mounting the servos on the robot, it is necessary to exchange the boards of the servos in order to improve their functionality. In the following subsections, we give a detailed overview of the most important electronic components.

3.4.1 Servos

A servo is a small device used for controlling mechanisms such as robotic arms, legs, or rudders, due to its limited rotation radius. It incorporates a two-wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft and has a three-wire system known as power, ground and control [51]. A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. This is done by a variable resistor called potentiometer, which is used for reading the position of the output shaft and comparing it then with the desired position of the arm, received via the data wire. If these values are not equal, a series of signals will be sent to the motor from a microcontroller, which will move the gears in a given direction, until the desired position of the potentiometer is achieved. There are two categories of DC servos: analog and digital. Both servo types have the same hardware components, except for the microprocessor of the digital one, which analyses the signals from the data wire and the potentiometer and controls the motor.

For powering Gretchen the HS-805MG Mega Giant Scale, Metal Gear analog Servo [52] is used. It belongs to the largest category of Hitec servos, with a heavy-duty metal gear train, a strong 10mm, 15 tooth output shaft which generates 2,4Nm of torque and a 3 pole brushed motor. This servo cannot be used for powering a robot just as it is, because of a few disadvantages. First, it cannot be easily connected to other servos in order to create a communication bus. Second, its on-board Hitec RCD ht7004 microcontroller is custom made and cannot be easily programmed, because there's no datasheet available online. These challenges can be undoubtedly solved with another servo motor, with publicly available microcontroller information, a more complex integrated circuit and bus connectors, but for a much higher price. The creators of Gretchen overcame these challenges by designing a custom board with an easily

Servo	Hitec HS-805MG Mega	Hitec HS-805MG Mega with custom Sensorimotor Board
Microcontroller	Hitec RCD ht7004	ATmega328P, 16MHz
Open-source firmware	no	yes
Max speed	0.14sec / 60° at 6V	0.14sec / 60° at 6V (to be measured for 12V)
Voltage range	4.8V - 6V	6V - 12V
Stall torque	1,9 Nm at 4,8V 2,4 Nm at 6V	2.47Nm at 6V/6A Approx. 5Nm at 12V/6A
In-built sensors	position	position, current, voltage, temperature, external measurements via I2C
Communication bus	no	RS485
Control loop frequency	N/A	100Hz

Table 3.1: Comparison of the specifications of the servo with the default board and with the custom Sensorimotor board

programmable on-board microcontroller, bus connectors and many more useful features that are displayed in Table 3.1 and will be discussed in the next subsection.

3.4.2 Sensorimotor Boards

As mentioned in the previous paragraph, the analog servo from the assembly kit has a few disadvantages and the custom boards and the electronic components from the assembly kit are a low-cost solution for overcoming them. The new boards have an ATmega328P 16MHz microcontroller, an ISP connector for programming it, two RS485 Bus connectors, an I2C bus connector, and a temperature sensor. The integrated circuit of the new board can handle a voltage up to 12V, which is twice as much as on the old board, thereby doubling the torque of the motor to 5Nm. This board can be used with other servos of similar size and features, with the maximum voltage limit of 12V.

Before inserting the new board inside the servo, the old one has to be carefully removed, by desoldering the ground and the motor pins with a soldering iron and a desoldering pump. We used a 50W soldering station which can heat up to 450°C. We used a 1mm thick lead-free soldering wire with 99,3% tin and 0,7% copper. The grounding pin is sinking much heat and even after setting the maximum temperature, it was still very

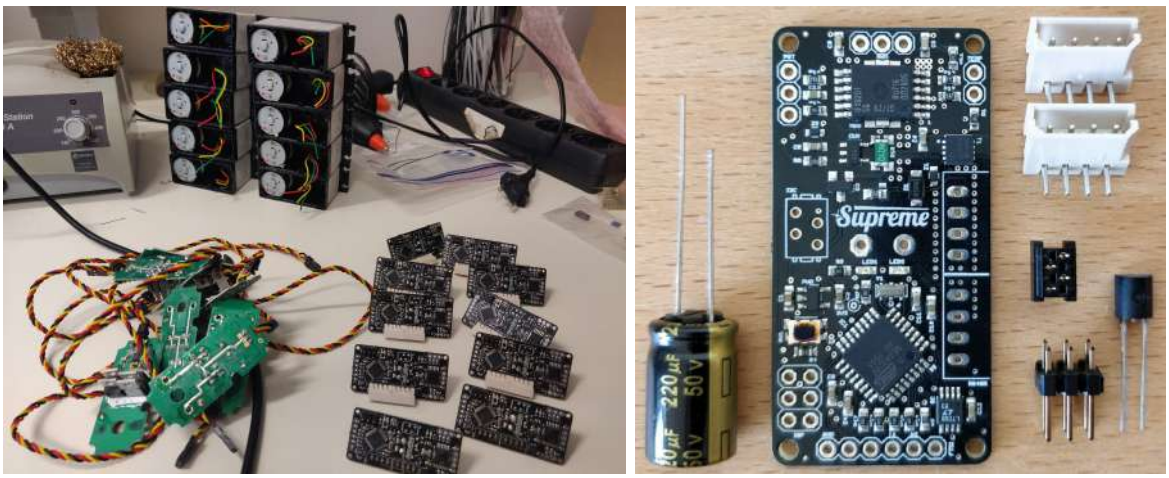

Figure 3.7: To the left: The process of exchanging the boards of the servos with the custom ones (old boards are on the left pile, new boards - on the right one). The Sensorimotor board and the electronic components which should be soldered to it (from left to right): electrolytic capacitor, 2x bus sockets, I2C connector, 6-pin ISP header, temperature sensor. This picture was taken from the assembly documentation [40].

hard to desolder it, so we lifted most of the boards together with the ground pin and the glue with which it was fixed to the motor. In this case, the process of soldering a cable to the motor was more challenging, than when the grounding pin remained on the motor. We'll elaborate on solving this problem in the Cables and Wires Section.

After removing the old board, the following components have to be soldered to the new one: a temperature sensor and a capacitor on the backside and an I2C connector, an ISP 6-pin header, and two rectangular 4-pin motorcord sockets on the front side of the board. The new board and the mentioned electronic components can be seen in the left picture of Figure 3.7.

The **temperature sensor** is placed on the backside of the board behind the microprocessor, in order to measure its temperature and eventually, prevent it from overheating. The KTY 81-110 silicon temperature sensor has a positive temperature coefficient (PTC), meaning that its electrical resistance increases with the temperature [53]. This sensor is very stable and can measure the temperature in the range between -55°C and +150°C with a negligible drift over 50 years [53].The KTY 81-110 temperature sensor has 2 pins: GND (Ground) and ADC (Analog to Digital Converter). The ADC pin transforms the analog voltage input into the digital one, where a voltage change of 7.59 mV corresponds to a change of 1°C [53]. The body of the sensor should be placed below the H-bridge with the flat side faced to the board surface. The position of the temperature sensor is very important, because when closing the motor box, it will be tightly packed between the motor's 'hill' and one of the motor pins. The right picture in Figure 3.10 illustrates the damaged insulation of the motor wire positioned under the temperature sensor, which resulted after closing the servo cover. This damage can potentially generate a short circuit and break the motor. In order to prevent this from

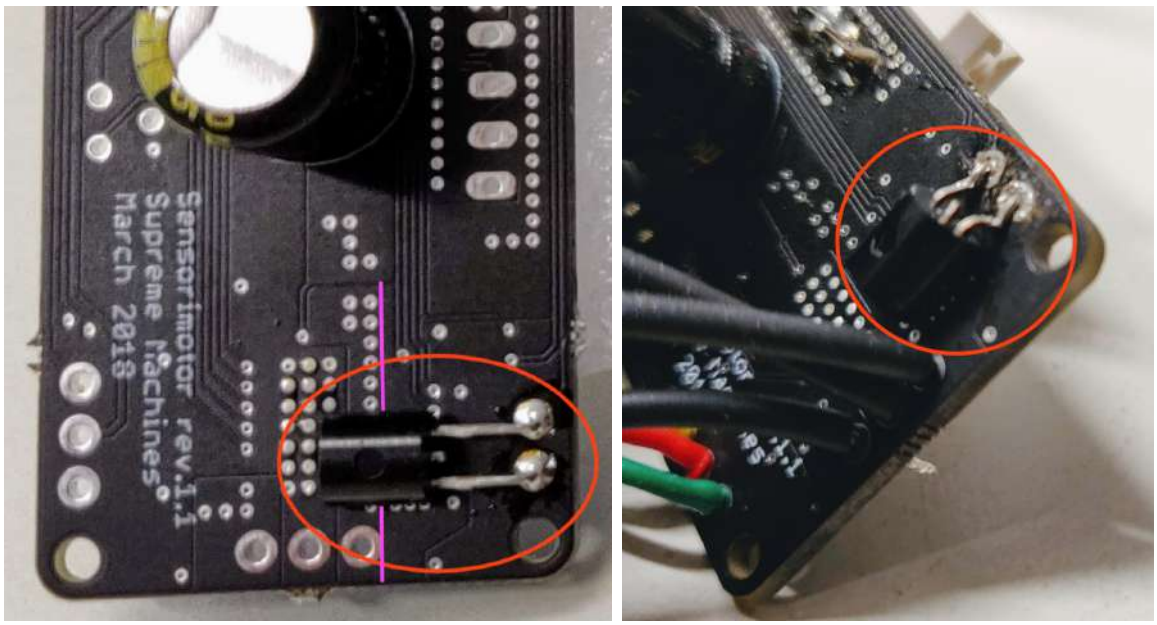

Figure 3.8: In the left picture, the position of the temperature sensor is wrong, as it is above the H-bridge line, which is marked with purple. The right picture illustrates the optimal position of the sensor with its feet bent in a staircase shape.

happening, the feet of the temperature sensor should be bent 3 times by 90 degrees in a staircase shape, as shown in the second picture of Figure 3.8.

The 6-pin **ISP header** is used for flashing the firmware on the microcontroller, this process being called In-System-Programming (ISP) [54]. The 6 pins of the ISP header correspond to the following lines: Master In – Slave Out (MISO), Serial Clock (SCK), Reset, Power, Master Out – Slave In (MOSI) and Ground [54]. We elaborate more on the ISP process in the Software Section. While soldering the ISP header, the pins shouldn't be heated for too long, as they heat up very fast and can change their position. We fixed the ISP header with tape, as shown in the third picture in Figure 3.8. The tape stabilized the pins and prevented them from changing their position during the soldering process.

The 220μF **electrolytic capacitor** is a polarized component that stores electricity and then discharges it into the circuit in case of an electricity drop. It has two pins of different lengths: the anode (the longer one) which receives voltage and the cathode (the shorter one) which sinks it to the ground. When soldering the capacitor, the cathode (ground lead) will sink a lot of heat, because it is directly connected to a large ground plane of the Sensorimotor board. We solved this issue by applying soldering fat around this pin, setting the maximum temperature on the soldering station and heating the area around the pin for a longer time. Due to the longer heating process, the ground pin of the capacitor has more flux than the other pin, as shown in the right picture of Figure 3.9.

The rectangular 4-pin **RS485 bus sockets** are used for connecting the servos in a

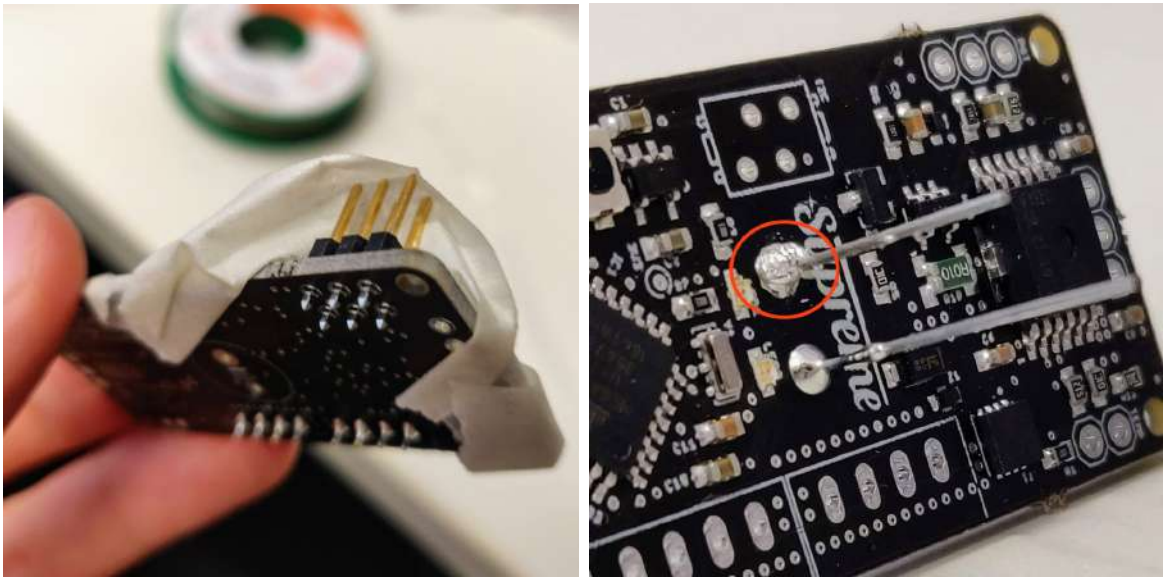

Figure 3.9: To the left: the 6-pin ISP header fixed with tape to the board, in order to prevent its pins from slipping down during the soldering process. To the right: the ground pin of the capacitor after heating it up for a longer time.

chain with the serial cables, which are described in Subsection 3.5.2. Before soldering the bus sockets, it is important to push them tightly into the board surface, in order to ensure a stable connection after putting cables in and out multiple times. Each RS485 socket consists of 4 pins: GND, VCC, A and B. We will elaborate on the functionality and its features in the Software Section 4.2.3.

The **I2C connector** can be used for connecting external sensors to the robot, such as the MPU6050 [55] accelerometer and gyroscope sensor. The I2C (pronounced I-squared-C) serial protocol allows low-speed devices like microcontrollers and similar embedded systems to communicate with each other via a two-wire interface: SCL (clock line) and SDA (data line) [56]. It allows both multiple masters and slaves, and in our case, the AVR microcontroller acts as the master, meaning that it generates the clock and initiates the communication with the slave (e.g. MPU6050), which receives the clock and sends a response back. The master supplies the slave with voltage through the 3V3 pin and sinks it with the GND pin to the ground plane.

3.5 Cables and Wires

Self-made cables are another important component that keeps the overall costs of the robot lower. The assembly kit provides two black and red silicon wires, as well as a sleeving and cable ties. In this section, we will first elaborate on the wires connecting the servo motor to the Sensorimotor board and then the cables connecting the servos between each other.

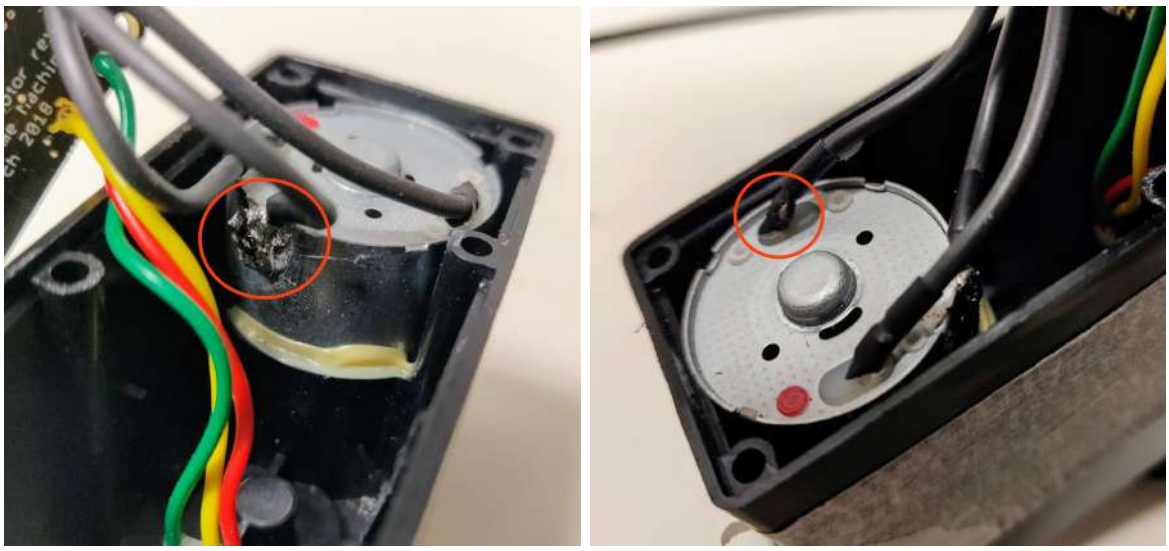

Figure 3.10: On the left side: Soldering the motor wire to the motor with the missing ground pin. On the right side: Damaged motor wire after closing the servo cover.

3.5.1 Motor and Potentiometer Wires

Each servo should have three 5cm wires connecting the ground and the motor pins to the board. A stripping tool is required for removing around 4mm of insulation from both ends of all wires. The wires' ends should be twisted and pre-soldered, in order to hold the wires together and ensure a good electrical connection.

If the ground pin was removed when desoldering the old servo board, then the remaining glue on the motor should be first removed. We used sandpaper to make the motor's surface smooth. Then we applied zinc chloride on the area where the ground pin would be soldered. Zinc chloride is a corrosive substance used for the pre-fluxing of metals prior to soldering.

We soldered the ground cable to the motor by using a gas torch, which can heat its tip up to 550°C. After soldering the cables to the motor, we cut diagonally 5mm long pieces of sleeving, in order to cover the ground and the motor pins. We used the open flame option of the gas torch for shrinking the sleeving tubes and isolating the motor pins. This is an important step because the cables will be packed tightly near each other after closing the motor box and open contacts may therefore generate a short circuit.

The copper potentiometer cables are not as heat resistant as the silicone ones, because copper and silicon wire insulation break down if the temperature exceeds 100°C and 200-250°C respectively. Therefore, while pre-soldering them, the insulation might start melting. In order to avoid this, we dipped the twisted wires' ends into soldering fat, therefore significantly speeding up the pre-soldering process.

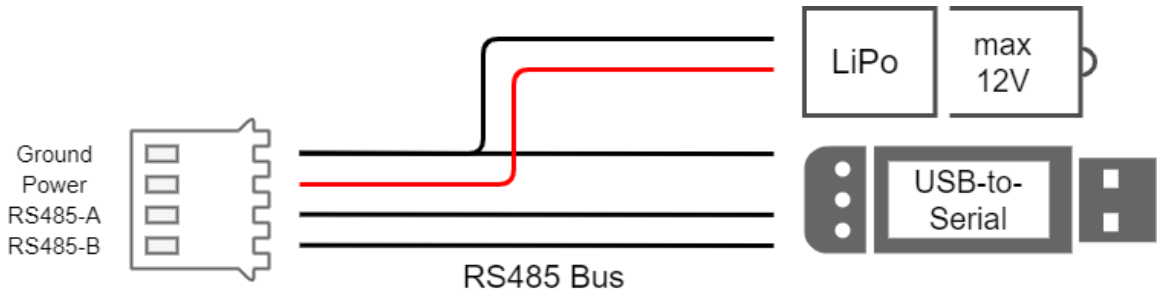

Figure 3.11: Schematic representation of the Y-junction between the USB-to-Serial connector, external power supply and the RS485 bus socket

3.5.2 RS485 Bus Cables

The connection between the servos is ensured by the RS485 bus, which consists of cables with 4 wires each: ground, power, data lines A and B. The distance between the motors is different, therefore cables of the following lengths are required: 42cm (4 cables), 38cm (2 cables) and 16cm (3 cables). Custom made cables have two major advantages: low manufacturing costs and the exact necessary length. Nevertheless, self-manufacturing may be time-consuming if a crimping tool is not available and if not handled with care, the connectivity of the cable might be reduced due to insulation damage. Using cables of different colors is not compulsory, but it may ease the manufacturing process as well as denote a personalized style. Therefore the black silicone cables are used for denoting the ground and the data lines and the red cable is used for the voltage carrying line, as shown in Figure 3.11. The cables should be twisted pairwise 4 to 6 times, in order to reduce the electromagnetic noise. Connecting the actuators with the RS485 cables represents the last step of the hardware assembly. The left photo in Figure 3.12 shows the fully assembled Gretchen robot, however the bus cables are still loose and can be easily damaged if the robot starts moving. Therefore, we tied the cables close to the wooden parts using zip-ties. The fully assembled Gretchen robot is shown in the picture to the right in Figure 3.12.

3.6 Power Supply

After finishing soldering the boards and connecting them with the motor and potentiometer wires to the servo motors, it can be beneficial to check if nothing is broken at this point. We checked the boards for short circuits with a multimeter before closing the lid and plugging a power supply into the servo for the first time. As mentioned in the previous paragraphs, the cables are packed very tightly in the motor's housing and there's a probability that the wire's insulation gets damaged when pressing the board inside the motor, which might result in a short circuit and a broken motor. Therefore, we checked the boards for short circuits both before and after closing the servo lid, in order to make debugging easier in case a problem would have been detected.

When connecting to the RS485 bus, it is necessary to use an external DC source of power, which can produce a constant output voltage. A single servo can draw up to 6000mA and the input voltage range is from 5V up to 12.6V. The HS-805MG servos can properly function with an input voltage of 6V, but due to the new electronic components, it can be powered with doubled power and therefore deliver twice as much torque. Therefore, for the best performance of the servo motors, it is recommended to either use a 12V power adapter plugged to a power socket or a lithium polymer (LiPo) battery with 3 cells. Each cell of a LiPo battery has a nominal voltage of 3.7V so a battery with 3 cells will deliver around 11,1V. We manufactured a Y-shaped adapter for connecting the serial bus to a computer via a USB-to-Serial device and a power supply, as shown in Figure 3.11.

3.7 Hardware Assembly Conclusion

In this section, we summarize our experience from the experimental assembly, all the challenges and the solutions we came up with, and discuss the accessibility of the robot and the level of difficulty of the hardware assembly.

The provided documentation is a helpful guide for the hardware assembly, however, it partially lacks detail. For instance, the importance of the position of the temperature sensor is not specified, and we had to solder it twice on all boards as they didn't fit in the motor housing when closing the lid. Moreover, besides the correct position of the temperature sensor, the feet of the sensor needed bending in a staircase shape, in order to avoid contact with the motor cables when closing the lid. We repaired the damaged cables by putting an additional layer of insulation. Furthermore, desoldering the old servo board resulted in the removal of the ground motor pin, which made the soldering of the motor wires much harder. We solved this issue by using a gas torch, which was able to deliver more heat than our soldering station. Another challenge that results from the error of the 3D-printer, is the shifted position of the side hole of the servo cover. This slight shift makes it impossible to insert the custom boards inside the 3D-printed lids. Therefore, we enlarged the holes using a drill.

The assembly of the Gretchen robot is a process that requires certain skills, such as soldering, using cable crimping tools for confectioning the cables, using a multimeter, working with superglue, using a laser cutting machine, a gas torch, a drill, and various other tools. From the standpoint of a Bachelor student of Computer Science, the assembly process is a challenging process, due to the lack of such practical skills. However, these skills can be acquired relatively fast when working alongside an experienced supervisor. Previous knowledge in electronics is certainly beneficial, however, currently available sources of information in form of videos, articles, and online forums, make the process much easier even without any prior knowledge or skills. Therefore, the Gretchen robot can serve as an accessible and low-cost platform for education and research of humanoid robotics.

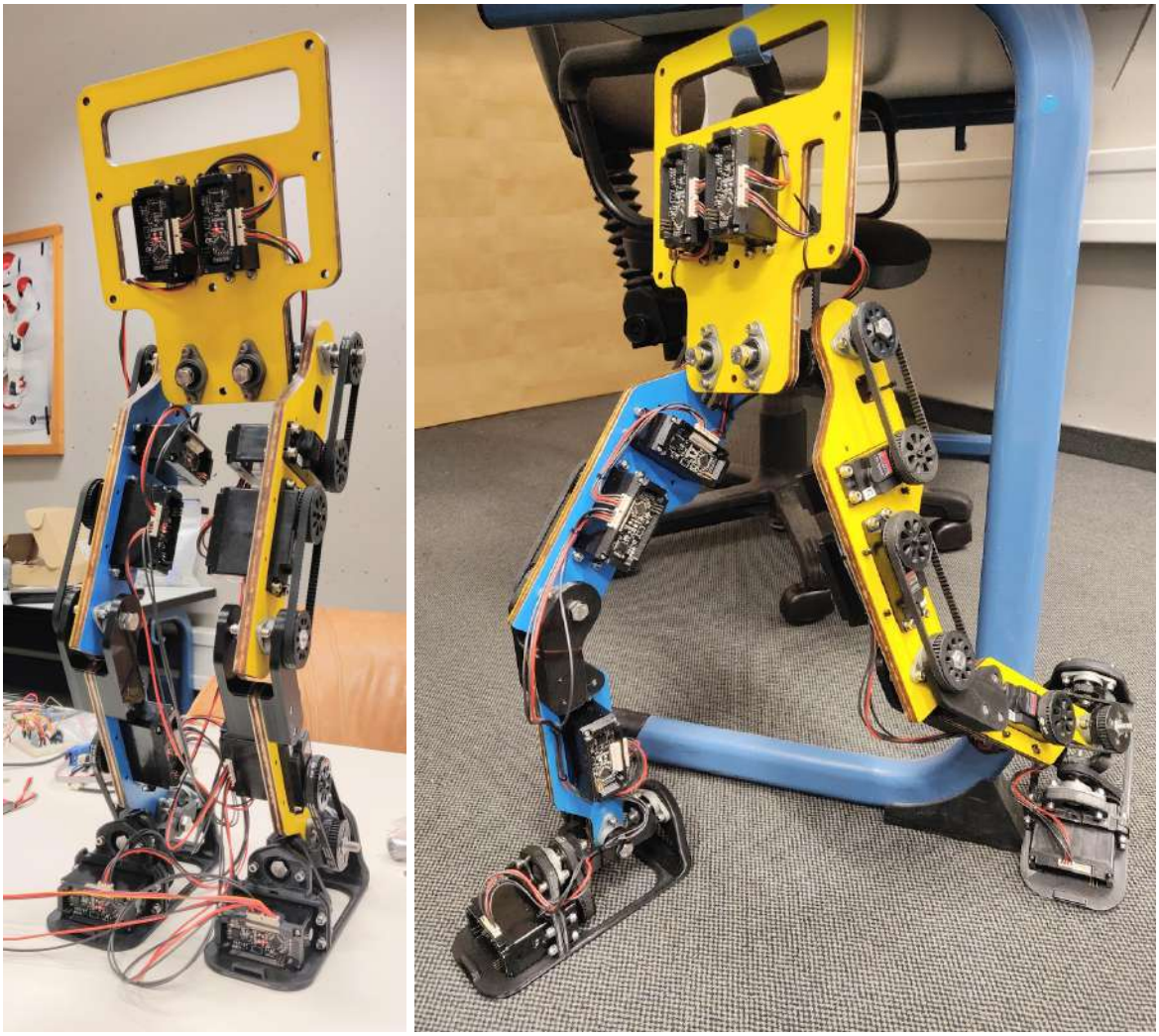

Figure 3.12: Left picture: The Gretchen robot standing up by itself, with the bus cables not bound to the wooden parts. Right picture:: the side view of the fully assembled Gretchen robot.

4 Gretchen - Software

This chapter elaborates on the software architecture, the most important software components in relation to the hardware, and the functionality and control of the servo motors. The currently available software libraries on the main GitHub page of the Gretchen robot [21] provide a minimal and clean interface to control the robot's actuators, which can be used for further developing complex motions and conducting motion experiments.

As previously mentioned in the Background Chapter, the Gretchen robot doesn't have an embedded computer, and its software can be therefore divided into two major modules: one module - runs on the embedded microcontroller on the board of each servo motor; and the second module - runs on an external computer and communicates with the robot via a cable. A high-level structure of these modules is illustrated in Figure 4.1 and in the next few paragraphs we give a general overview of each of them.

In the middle of Figure 4.1, the Gretchen robot is illustrated with 10 servo motors, which communicate with each other through a serial bus - marked with the red arrows. Each servo motor has an inbuilt Sensorimotor board (previously discussed in Subsection 3.4.2), which has an embedded microcontroller, which is illustrated on the left side of Figure 4.1. The main software module that runs on the embedded microcontroller is called firmware. The source code for the firmware can be found in the Sensorimotor repository [21], together with several test programs, tools, documentation and board circuits. The firmware contains the necessary programs that manage the communication between the actuators and the computer via the serial bus, control the servo itself and process the acquired motor and sensor data. The firmware can be edited or changed entirely, due to the open-source nature of the robot. The desired firmware is uploaded on the microcontroller of the board via the ISP interface and the motors are communicating with the computer via a USB-to-Serial adapter. Both of these communication interfaces will be discussed in the following section.

The module illustrated on the right side of Figure 4.1 represents the high-level architecture of the software libraries, which run on an external computer and are used to control the motions of the robot. These libraries are available in the Libsensorimotor repository [21] and can be divided into two main parts, written in different programming languages, namely: C++ and Python. The C++ libraries manage the low-level communication of the computer with the serial bus and control the actuators using different control mechanisms. The Python API provides a high-level and simplified interface to control the motion of the robot, without requiring extensive knowledge about the complex communication and control mechanisms of the C++ libraries.

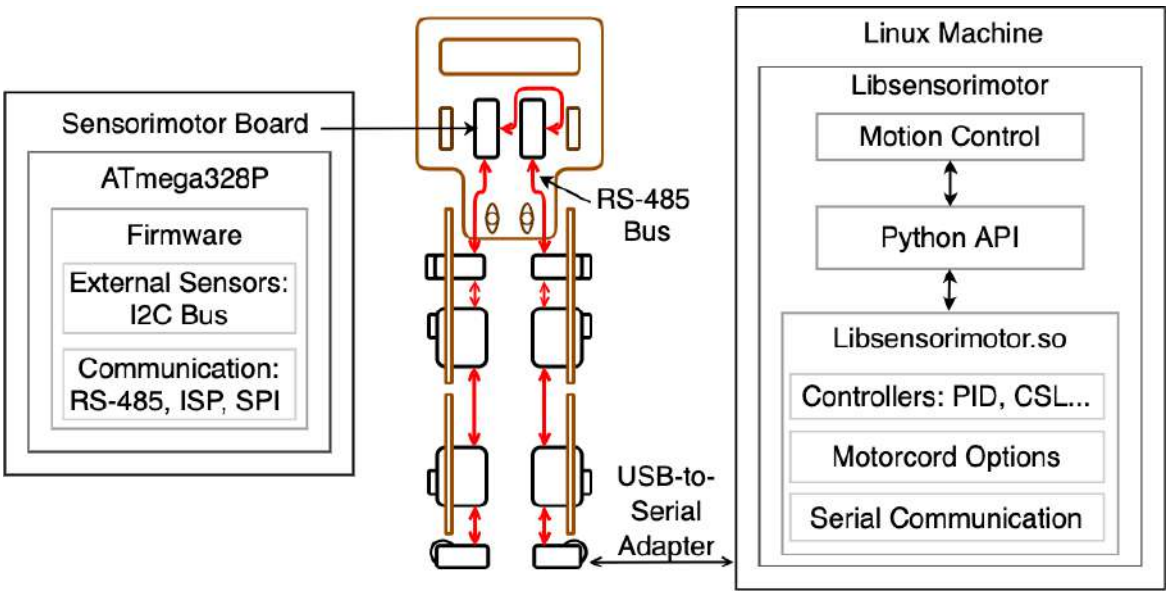

Figure 4.1: Overview over the most important software components running on the Sensorimotor boards and the external computer, and data flow of Gretchen. Red arrows illustrate the RS-485 bus connecting the motor boards.

The experimental assembly and software setup processes have shown the importance of being familiar with several basic notions from computer science and electrical engineering, which are necessary for understanding the structure and the underlying concepts of the control software. Therefore, we've included the definitions and explanations of relevant notions at the beginning of the following sections, before elaborating on their functionality and purpose in the context of the Gretchen robot. In the Firmware and Sensorimotor Boards Section, the connection between the hardware and the software is closely examined. We scrutinize the embedded microcontroller, namely its features, firmware and communication capabilities. Furthermore, several motor control approaches are discussed and compared in the Motor Control Section. Finally, in the Libsensorimotor - Python API Section, the implementation of one of the control approaches is described and a few basic motion experiments are conducted. Please note that for clarity reasons, in the following sections a motor refers to a motor equipped with a Sensorimotor board, which was previously described in Subsection 3.4.2.

4.1 Firmware and Sensorimotor Boards

The goal of this section is to describe the connection between the software and the hardware of the Gretchen robot. The embedded microcontroller on the Sensorimotor board is an important and complex hardware component, which is worth discussing in detail before moving on to the firmware installation and the communication between the boards and the computer.

4.1.1 ATmega238P

Each servo motor powering the Gretchen robot contains a Sensorimotor board with an embedded ATmega238P [57], which is a high-performance and open-source single-chip microcontroller created by Atmel and commonly used for open-source projects e.g. Arduino UNO [58]. Its maximum operating frequency amounts 20 MHz and it operates on a voltage ranging from 3.3V to 5.5V. The ATmega238P has several important features, which are relevant to examine in the context of the software of the Gretchen robot, namely: fuses, In-System Programmable Flash, Pulse Width Modulation channels, Serial Communication and Serial-Peripheral-Interface. We elaborate on each of these features in the following subsections.

The fuses are 3 bytes, called low, high and extended byte, which store the configuration of the ATmega238P, each bit corresponding to a setting or flag [59]. For all fuses, a value of 1 means not set and a value of 0(zero) means set. The low byte fuse manages the clock settings and speed, as well as the delay after the startup of the microcontroller. The high fuse determines the settings of serial programming and preserving or erasing EEPROM and the bootloader size. The extended fuse ensures that the microcontroller receives enough voltage in order to work properly and it is not set by default. The default fuse settings of the low, high and extended fuses of the ATmega238P are 0x62 (B01100010), 0xD9 (B11011001) and 0xFF (B11111111) respectively.

4.1.2 Firmware

The firmware is the essential software program or set of instructions running on a microcontroller, which ensures communication with other devices and determines its actions [60]. It is commonly located either in the Electrically Erasable Programmable Read-Only Memory (EEPROM) or in the Flash Memory. Both of them belong to the non-volatile type of memory, meaning that the stored data remains unchanged even after disconnecting the power supply. The main difference between EEPROM and flash memory is that flash memory is block-accessible, meaning that an entire block must be erased before writing, while EEPROM erases one byte at a time [61]. Flash memory is therefore faster and cheaper, however it wears out faster than EEPROM due to a smaller number of write cycles. The EEPROM memory of the ATmega328P microcontroller is 1.024KB and can be rewritten up to 100,000 times, while its flash memory is 32KB and can be rewritten up to 10,000 times. Due to the fact that the firmware is commonly uploaded on the flash memory, this process is often called "flashing the firmware".

4.1.3 In-System Programming

In-System Programming (ISP) allows programming microcontrollers or other embedded devices, without removing them from the board [54]. The Serial-Peripheral-Interface (SPI) provides the communication between the microcontroller and the computer and

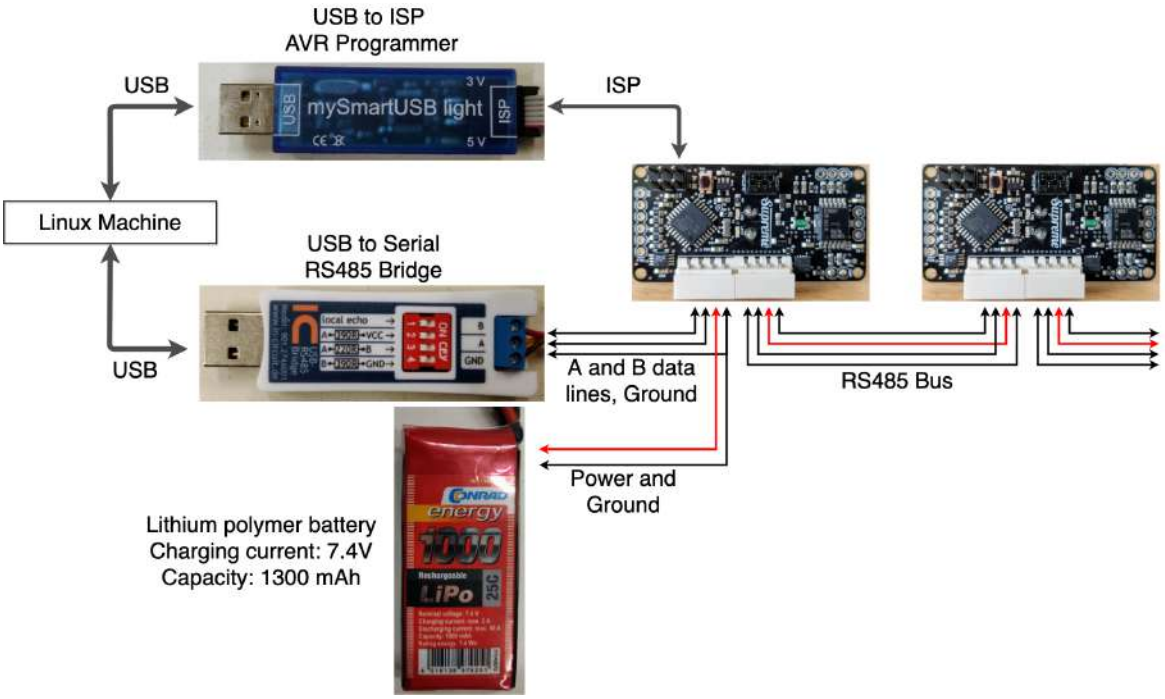

Figure 4.2: The communication channels between the servos with embedded Sensorimotor boards and the computer: On the top part - the USB to ISP Programmer is shown, which is used for the process of flashing the firmware. The bottom part illustrates the communication between the computer and the robot's serial bus via a USB to Serial Bridge and a power supply attached to it.

it consists of 3 wires: Serial Clock (SCK), Master In – Slave Out (MISO), and Master Out – Slave In (MOSI). The clock line synchronizes the communication between the computer (master) and the target system (slave), so that at each pulse of the clock line, the microcontroller receives one bit through the MOSI line and simultaneously sends one bit to the computer through the MISO line.

As previously discussed in Subsection 3.4.2 about the Sensorimotor Boards, the 6-pin ISP connector soldered on the board for flashing the firmware on the ATmega238P. The ISP header has 3 additional pins besides the ones of the SPI, namely: Reset, Power (5V) and Ground (GND). The Reset line is controlled by the master and ensures that once the communication has started it's not interrupted. For transferring the desired firmware to the microcontroller, we used an AVR-ISP MySmartUSB Light Programmer [62], which is illustrated in the upper part of Figure 4.2. Due to the 5V and the GND pins an external power supply is not necessary, as the programmer draws enough power from the computer for supplying the microcontroller with sufficient voltage to ensure its proper functionality.

4.1.4 Firmware Tests

As previously discussed in the Hardware Chapter, the assembly process of the actuators involves several steps which require precision and very careful actions, in order to avoid accidental damage of the boards. For instance, the soldering process of the capacitor involved heating the board for a longer time, as the ground pin was sinking a lot of heat into the board. Moreover, the board LEDs are located very close to the capacitor pins and can be easily damaged by the soldering iron if it slips a few millimeters away. Furthermore, the 3D-printed motor lid can damage the insulation of the motor cables, if the temperature sensor is slightly shifted, potentially causing unexpected behaviors of the board or short circuits. Thus, it is essential to test the boards' functionality before uploading the firmware and connecting it to the serial bus.

The blinky test script, which can be found in the Sensorimotor repository [21], provides the possibility to check if the board electronics are intact, if the on-board microcontroller can be connected to the computer and respond to basic requests, if its LEDs are intact and if the serial bus and ISP work. The advantage of using a test script as blinky is that no other tools are required to test if the hardware works properly. Additionally, blinky changes the default values of the fuses. The default value of the low byte fuse of the Atmega328p is changed from 0x62 to 0xCF, meaning that the clock speed will be bigger than 8MHz (instead of dividing it by 8), there will be no clock output and the chip will startup with a small delay. The high byte remains unchanged at 0xD9, meaning that serial programming is allowed, the EEPROM memory is preserved and the bootloader has a maximum size of 4KB. The extended fuse is changed to 0xFD, which signifies that the minimal voltage limit is 2,7V.

We uploaded the blinky test script to the ATmega238P via the AVR Programmer. The programmer supplies the board with 5V, which is enough for successfully transferring the code to the chip, but not enough for powering the on-board LEDs. In order to check if the test ran successfully and the LEDs were blinking, we had to plug in one RS485 connector with the battery, as shown on the bottom part of Figure 4.2. After this step, the firmware can be safely flashed on the microcontroller.

4.1.5 RS-485 Bus

The motors are connected with each other in a daisy-chained manner and communicate with each other and the external computer via the RS-485 serial bus. RS-485 is a data transmission standard for serial communication, which allows for robust transmission of moderate data rates over long distances [63]. It consists of 4 lines: power (VCC), ground (GND), data lines A and B. The RS485 network uses differential communication of two data lines, meaning that the voltage of the data line A is the inverse of the one on the line B. In other words, the same signal is passed through both lines, however, on one line it is inverted, thereby making the bus immune against noise. The output of the receiver is 1 if the voltage of data line A is 200mV bigger than the one of B, and 0

- otherwise. The RS-485 bus lets up to 32 devices to communicate via a single pair of wires, allowing only one device to communicate with the other ones via the bus at a time, with a data rate of 100Hz (10ms) per cycle.

There are multiple options of connecting the external computer to the serial bus of the robot, by using a USB-to-Serial/RS485 device, such as FTDI USB-RS485-PCB, Digitus DA-70157 or In-Circuit USB-RS485-Bridge [40]. We used the In-Circuit USB-RS485-Bridge by connecting it to the robot's foot, as illustrated in Figures 4.1 and 4.2 . As opposed to the other USB-to-Serial devices mentioned above, the USB-RS485-Bridge has only the 3 IO pins which are needed to connect to the robot's bus: RS485-A, RS485-B and Ground. We have soldered the ground cables of the USB-RS485-Bridge and of a battery, and created a USB-to-Serial connector which is illustrated on the lower part of Figure 4.2.

4.1.6 Pulse Width Modulation

The actuators of the Gretchen robot are controlled by Pulse Width Modulation (PWM), which is a commonly used control approach of powering brushed DC motors of legged robots and robotic arms. PWM is a technique of digitally encoding analog signals by chopping an electrical signal in discrete parts [64]. The voltage and the current are controlled by turning the power on and off at a fast rate [64]. The period of time in which the power is turned on - is called a duty cycle. For instance, if the power is on 20% and off - 80% of the time, the PWM output is at a 20% duty cycle [64]. The advantage of the PWM is that it eliminates the need for a digital-to-analog conversion. Due to the fact that the signal remains digital all the way from the processor to the controlled system, the noise effects are minimized, as the noise has to be strong enough to change a logical 1 to 0, or the other way around [64]. The maximally allowed PWM duty cycle is used for setting the voltage limit with which the actuators of the Gretchen robot are powered. This process is discussed in more detail in Subsection 4.2.5.

4.2 Motor Control

The actuator of the Gretchen robot can be controlled by the software libraries which run on an external computer with Linux OS, as shown on the right side of Figure 4.1. This introduces a significant delay in the control loop because at each time step the motors send commands with their current state to the computer and wait for it to recalculate the data and send the new commands back. In order to minimize the feedback delay between the computer and the servos, it's important to set the low-latency mode on the computer. However, the low latency mode doesn't eliminate the feedback delay completely. The movements of the robot are generated by changing the speed and the rotation of the actuators, which are determined by the current and its direction respectively. Conventional microcontrollers can't pass enough current or

voltage for powering a motor, therefore specialized circuits have been developed to supply motors with power and protect the microcontrollers from additional electrical problems [65], one of the being the H-bridge.

4.2.1 H-Bridge

An H-bridge is a widely used circuit for powering DC motors, which uses 4 switches for managing the direction and the speed of a motor [66]. The 'H' signifies the shape of the circuit, as the four switches are placed pairwise in series and in parallel, with the motor placed in the middle. The Sensorimotor boards have an integrated H-Bridge which receives as input a PWM and a direction signal and can handle up to 6A.

4.2.2 Controllers

Calculating the input values for the H-Bridge is not a trivial task and there are several approaches for controlling the motors, the most common ones being position, velocity and torque control. The position control mechanism is easy for humans to comprehend because the position is a physical variable more commonly used in the real world than torque or velocity. Torque control is a method of moving the actuators by working with voltage and the current, instead of the position. Velocity control, as the name suggests, calculates the speed of the motor by comparing the desired velocity with the measured velocity. Due to the open-source nature of the Gretchen robot, it allows a big room for experiments and for trying out alternative controllers. For instance, the Libsensorimotor repository contains the implementation of a Cognitive Sensorimotor Loop (CSL) controller, which is a novel, computationally simplistic, and energy-efficient approach of making a humanoid robot to stand up from various postures [67]. It does not require any additional sensors - the motor itself serves both as actuator and sensor [67]. In this work, we focus on the position control approach for manipulating the motions of the Gretchen robot.

4.2.3 PID Controller

The Libsensorimotor repository contains the C++ implementation of a position control mechanism, namely, the proportional-integral-derivative (PID) controller. The motors are sending their position to the computer, where the PID controller calculates the next position and sends it back to each of the motors at a rate of 100Hz until the desired positions are achieved.

A PID controller is the most commonly used control loop mechanism due to its simple implementation, clear functionality, and a wide range of applications [68]. Its transfer

function is shown below:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt} \tag{4.1}$$

where e(t) is the error, K_p, K_i and K_d are the proportional, integral and derivative parameters respectively. The error value indicates the difference between the desired position, also called setpoint, and the current position of the motor. The goal of the controller is to minimize the error by adjusting the control variable $u(t)$. The proportional term, as the name suggests, changes the controller output in proportion to the error, so the bigger the error - the greater the control action. If a controller is using only the proportional term, meaning that the other ones are set to zero, the end value will oscillate around the required position, but depending on the parameter there might always be an error. This problem is well known as the steady-state error and it can be minimized by the integral term. As long as the error is positive, the integral term will keep incrementing the rate of the control output, and otherwise - will be decrementing it, until the error is zero. The derivative term reacts to the rate of change of the error, meaning that once the error is being minimized with fast speed, the derivative term will gradually decrease the speed of the motor and prevent it from passing over the desired position. The three parameters of the PID controller require tuning, which is an important process, as the speed and the electric draw of the motors depend on them.

Another term that can be used for increasing the service life of the actuators and enhancing the controller's efficacy is the deadband. The deadband is a constant which is subtracted from the error value in order to limit the lower end of the voltage output [40]. In other words, if the error value is smaller than the deadband, the error will be set to 0 and the controller will stop the motors at that position. This approach prevents the actuators from infinitely trying to reach a setpoint to which they are very close, but cannot reach for some reason.

4.3 Libsensorimotor - Python API

As a final step of this work, we performed some basic experiments to demonstrate the functionality and motion capabilities of the robot. In this section, we describe the setup and the structure of the experiments on position control. Our first experiments show that the robot can be used for experimenting with various control approaches and that the motors are strong enough to make the robot stand up and perform basic motions.

4.3.1 Motorcord

The Python API of the Libsensorimotor library provides a possibility for high-level control of a series of motors equipped with the Sensorimotor boards and connected in a chain. We refer to the chain of motors connected via the RS485 bus as Motorcord.

Figure 4.3: All 10 servo motors displayed, before being connected in a chain, labeled from 0 to 9.

The IDs of the motors in the chain have to be set in sequence beginning with 0 (zero). We labeled the motors with numbers from 0 to 9 (see Figure 4.3) and have set the corresponding ID for each motor by using the `set_id.py` script available in the Sensorimotor repository [21]. In the following paragraphs, we elaborate on the process of creating the motorcord and controlling the motors based on the position control approach.

The following line in Python instantiates the connection to the robot's chain of motors:

```
motors = Sensorimotor(
    number_of_motors = 10,
    update_rate_Hz = 100,
    verbose = True)
```

The parameters `update_rate_Hz = 100` and `verbose = True` are default values and do not need to be set explicitly.

In order to verify that all motors in the chain respond correctly we can use the function `ping`:

```
n = motors.ping()
```

The returned number `n` indicates the number of motors that did respond. In the case of Gretchen, the expected number would be `n = 10`.

The class `Sensorimotor` provides an internal loop which realizes the communication with the motors. In each frame of the loop, new motor commands are sent to the motors and new sensor data is retrieved. The internal loop runs as an independent thread and can be started and stopped with the following commands:

```
motors.start()
motors.stop()
```

The function `get_position` reads out the positions of the motors, as measured by the position sensors:

```
s = motors.get_position()
```

The returned value `s` is a list containing the positions of the motors in radiant as floating numbers. The length of the list corresponds to the number of motors in the chain and the position in the list corresponds to the ID of the corresponding motor, i.e., in the above example `s[3]` is the angular position of the third motor.

In our experiments, we use the position control as described in Section 4.2.3. The target positions for the motors can be set with the following function:

```
motors.set_position(p)
```

Thereby `p` is a list of target angular positions for each of the motors in radiant as a floating number. The following simple example demonstrates setting the positions of all motors of the Gretchen robot to `0`:

```
p = [0.f,0.f,0.f,0.f,0.f,0.f,0.f,0.f,0.f,0.f]
motors.set_position(p)
```

A complete basic control loop has the following form:

```
motors = Sensorimotor(number_of_motors=10)
motors.start()

while( True ):
    s = motors.get_position()

    #do something with the positions ‘s‘ and
    #calculate new target values for the motors ‘p‘

    motors.set_position(p)
    sleep(0.01)

motors.stop()
```

The above loop is executed independently of the internal communication loop. The functions `get_position` and `set_position` are non-blocking, which makes it necessary to artificially pause the control loop with the `sleep(0.001)` function, in order to wait for the internal loop to finish execution. This makes the control-loop robust since the internal communication is not interrupted in case the calculations in the main loop take longer than permitted by the frame rate (in our case 10ms). In the future, the API should be extended with functionality which would also allow synchronizing the main loop with the internal communication loop, in order to ensure precise control of the motors' positions.

4.3.2 Voltage Configuration

We have conducted a few experiments with different input voltage values and observed the behaviors of the robot, in order to find an optimal voltage value. The voltage limit should be set according to the supply voltage and desired maximum motor speed with the following function:

```
motors.set_voltage_limit(limits)
```

The *limits* parameter is a list of N elements, where N is the number of attached motors and the N values in the list correspond to the supply voltage. The voltage limit is setting the maximally allowed PWM duty cycle, which is a value between 0 and 1. For instance, if the power supply delivers 12 Volts and the voltage limit is set to 0.25, then each motor is powered by maximally 3 Volts on average. In this case, the function call is written as follows:

```
motors.set_voltage_limit([0.25]*10)
```

It is recommended to set the voltage limit in accordance with the motors' specifications or below. The voltage limit specified in the datasheet of the Hitec servo motors used for powering the Gretchen robot is 6V. However, with the custom board, the voltage limit can be set up to 12V for getting higher performance. Setting a very low voltage limit can cause the robot to tremble or to get stuck in certain positions. For instance, with the motors receiving 3V on average as in the function call above, the robot will tremble when attempting to stand up and the motors located in the knees would not have enough power to lift the body up. When setting this parameter to a value bigger than 0.8, meaning that the actuators were powered with more than 9V on average, the movements of the robot were smooth and it could easily perform the standing up and sitting down motions. It is yet to be tested if the servos wear out faster when being constantly powered by a higher voltage than the one specified by the manufacturer.

4.3.3 Position Control Experiments

We have tested and tuned the proportional, integral and derivative parameters, as there is no default setting which will work for all types of motors. The parameters of the PID controller can be set in the following function call:

```
motors.set_pos_ctrl_params(motor_id = 0, Kp = 0.0, Ki =
0.0, Kd = 0.0, deadband = 0.0, pulse_threshold = 0.0)
```

The `motor_id` parameter, as the name suggests, corresponds to the ID of the motor whose position is going to be calculated by the PID controller. We can set this function call inside a **`for`** loop which iterates through all 10 servo motors of the robot and sets the `motor_id = i` where `i` iterates from 0 to 9.

As previously mentioned in Subsection 4.2.3, the `Kp, Ki` and `Kd` parameters of the PID controller require tuning. We started the tuning process with all parameters set to 0 and a low voltage limit. Then we began to gradually increase the proportional term (`Kp`) and the voltage limit until the desired reactivity and speed of the motors were achieved. Next, the derivative term (`Kd`) was gradually increased, in order to prevent the motors from accelerating too fast, if the target position is far away from the current position of the motor. At this stage, we noticed that once the motors are close to the target position, the voltage output is very low and the motors are not reaching the desired target positions completely. Therefore, we carefully increased the integral term (`Ki`), in order to bring the motors closer to the desired setpoints. Finally, the position control parameters achieved the following optimal values:

```
  motors.set_pos_ctrl_params(i, Kp = 10, Ki = 10, Kd =
0.1, deadband = 0.1, pulse_threshold = 0.00)
```

Note that we didn't set a value for the `pulse_threshold`, which is not a common parameter for a PID controller. It was included as an experimental setting, in order to switch the controller to a low-frequency pulse mode if the output voltage is below the chosen value of the `pulse_threshold`. The functionality and effects of this parameter are yet to be tested. In our experiments, the PID controller worked properly with the above-shown parameter values. As mentioned earlier in this subsection, the parameter values can vary from motor to motor and if certain values work for one robot, they might not work for another robot with the same configuration and actuators.

4.3.4 Basic Motion Experiments

In order to test how good the PID controller works, we programmed the robot to stand up from its default position and sit back down. As mentioned earlier in Subsection 4.2.3, the PID controller requires as input the predefined target position values of all 10 motors. Therefore, we read out the position values of all the motors by manually holding the robot in the sitting, squatting and standing postures, which are shown in Figure 4.4. We saved the positions of each motor in a list of 10 values, as previously described in Subsection 4.3.1. The values of the 3 positions are shown below:

```
  sit = [-0.2890625, -0.853515625,  0.826171875,
0.82421875, -0.56640625, 0.509765625,
-0.8125,-0.892578125, 0.78515625, 0.439453125]
```

```
  squat = [-0.318359375, -0.85546875, 0.283203125,
0.771484375, -0.53125, 0.45703125, -0.763671875,
-0.345703125, 0.787109375, 0.44140625]
```

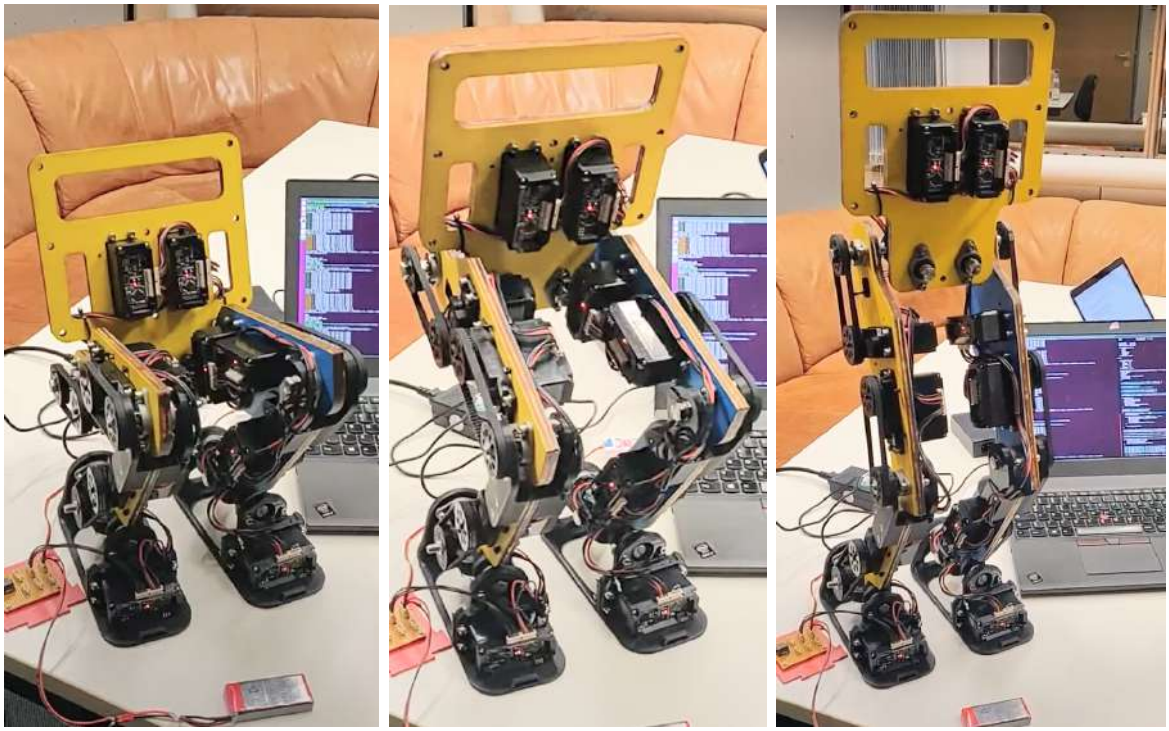

Figure 4.4: The standing up motion of the Gretchen robot, by interpolating through the sitting, squatting and standing positions

```
    stand = [-0.310546875, -0.78125, -0.0234375,
  0.23828125, -0.55078125,  0.46484375, -0.2265625,
  -0.05078125,  0.720703125, 0.421875]
```

We interpolated the values from the sitting position to squatting for 5s and from squatting to standing for 5s and let the robot stand 10s long. Then we interpolated these values in the reversed order all the way back to sitting position, for 5s in total, thereby generating a standing up and sitting down motion of the robot. With this motion experiment, we demonstrated that the robot's motors are strong enough to lift the robot up in the standing position and that the robot is capable of performing basic movements by using a PID controller. Additionally, the robot is able to walk with small steps, as shown in the following video by Matthias Kubisch [69].

4.4 Software Conclusion

In this chapter, the software architecture of the Gretchen robot and the communication between the most important software modules were discussed. In the current configuration of the robot, the motor control software runs on an external Linux machine,

which introduces a feedback delay in the control loop and also prevents users with other operating systems from working with the robot. Furthermore, we elaborated on the most important features of the embedded ATmega238 microcontroller in the Sensorimotor boards, namely the fuses, firmware, ISP, serial communication and PWM. Next, the functionality of the servo motors and various motor control approaches were described. Finally, we explained how to control the motorcord via the Python API with a PID controller and performed a few basic experiments which showed that the robot is capable of performing basic motions. The Python API proved to be a simple interface, which can provide a good basis for teaching purposes allowing a low entry threshold for beginner students. Moreover, the open-source nature of the Sensorimotor boards allows low-level experiments with different motor control approaches and can, therefore, drive the research on the motion of humanoid robots.

5 Discussion and Future Work

In this chapter, we summarize the Gretchen robot's features which stand out in comparison with other humanoid robots. Furthermore, we discuss several limitations of the current software configuration and hardware design and propose solutions for future work. We examine the robot's potential for being used in educational and research projects and discuss the accessibility of the robot from the standpoint of a Bachelor student of Computer Science.

The Gretchen robot stands out from the humanoid robots currently used in research and education, due to its open-source nature, modular design, open-platform and low-cost hardware. However, as a prototype at an early development stage it still lacks many important features, such as more degrees of freedom, an upper body, embedded computer, sensors and cameras, as well as a more complex and efficient software. In the following sections, we discuss these limitations in more detail and propose solutions for future work.

5.1 Hardware

In the current hardware configuration of the robot, the hip joint lacks one degree of freedom - the yaw, meaning that the robot can only walk forward and backward and cannot go around its rotational axis. The hip joint design can be inspired from morphologically similar robots, such as Hambot [44].

Another limitation of the current design of the feet is the unfavorable sitting position of the robot: the knees are too high and the robot leans back. A photo of the robot in the sitting position can be found in the previous chapters - in the Figures 2.3 and 4.4. Therefore, it gives the impression that a lot of power should be applied to the knee joints, in order to generate the standing up motion. In the future, this challenge could be solved in 2 possible ways. First, designing and manufacturing the upper body for the robot could compensate this position by supporting the standing up motion by leaning forward and by moving the hands forward. Another solution would be changing the design of the feet, so that the knees are positioned lower while sitting, ideally on the same level as the hips, similarly to the human body when squatting.

Integrating a foot pressure sensor can be useful for the stabilization of the robot while walking. The BitBots team from the University of Hamburg created a low-cost foot pressure sensor for Hambot called Bit Foot [20], which could be integrated under Gretchen's feet without major hardware modifications. They use an ADS1262 breakout

board, which is an analog-to-digital converter able to sample four analog signals with a resolution of 32 bit at a rate of up to 38400 Hz [20]. The resulting walking motions look more natural and smooth, as the robot changes the center of gravity similar to the way humans do while walking.

In order to ease the soldering process of the temperature sensor and avoid resoldering it in case it doesn't fit in the motor box, it would be beneficial to mark its position on the board, e.g. with a square. Another solution would be ordering a custom board with an integrated temperature sensor of a smaller size. This way, the risk of cutting through the cable insulation when closing the motor cover would be minimized. Another way of protecting the motor wires from damage when closing the motor cover, would be increasing the height of the motor covers by 2mm before the 3D-printing process.

5.2 Software

One of the limitations of the current state of the robot is that the controllers are implemented in the Libsensorimotor module and are therefore executed on an external computer. This introduces a significant delay in the control loop because at each time step the motors send commands with their current state to the computer and wait for it to recalculate the data and send the new commands back. One possible solution that doesn't involve any additional hardware is running the PID controller directly on the ATmega238P microcontroller. The firmware of the robot is completely customizable and the embedded microcontroller of each motor delivers enough performance to perform more complex calculations. Therefore, the RS-485 bus control could be handled directly by the firmware, if the controller software runs directly on them. On the left side of Figure 5.1 the modified firmware of the ATmega238P is shown, with the controllers running directly on it.

Wireless control is another useful feature of a humanoid robot that Gretchen is still lacking. In order to control the robot wirelessly, an on-board computer is required. A possible solution would be attaching a small embedded computer to Gretchen's lower torso. Based on Figure 5.1, we can observe a Raspberry Pi connected to the robot's serial bus via a T-shaped connector with 3 RS-485 input bus sockets. A Raspberry Pi is a small, low-cost, and open-source single-board computer, which is widely used in research and educational projects in the computer science field [70]. Raspberry Pi runs a variety of Linux distributions and is therefore a perfect fit for the software requirements of the Gretchen robot. Compiling the Libsensorimotor module on the Raspberry Pi would also allow Gretchen to be more accessible and easier to use. The Raspberry Pi can be easily controlled from any computer by using the Secure Shell (ssh) protocol. Therefore, the robot could be wirelessly controllable from an external computer with any operating system. This way, the robot would become more accessible for educational projects and would allow almost immediate entry into the programming process of the robot's motions.

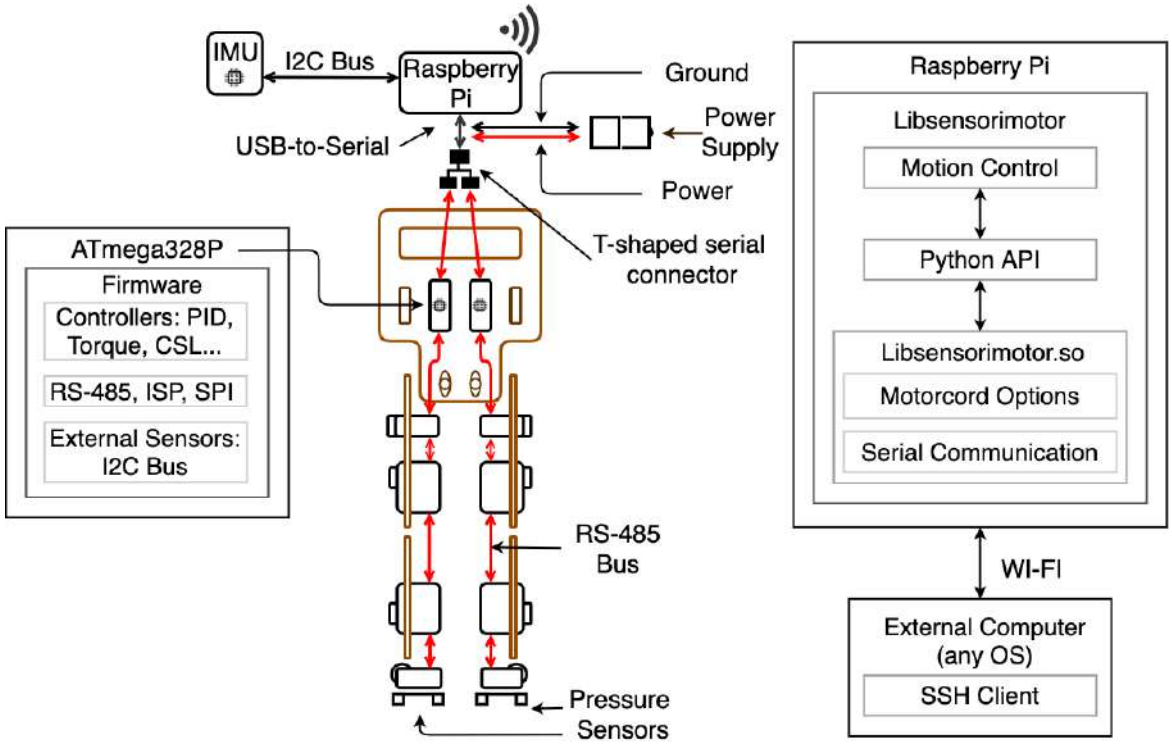

Figure 5.1: Proposed extension of the software architecture for optimizing the motion capabilities of the Gretchen robot. On the left side, the extended firmware of the ATmega328P microcontroller is shown, where the PID and other motion controllers run. The Raspberry Pi represents the on-board computer which manages the serial communication with the motorcord and runs the motion commands. An Inertial Measurement Unit is connected to the Raspberry Pi for additional sensory measurements. The robot can be stabilized during the motion by the pressure sensors located under its feet. The actions of the robot can be controlled wirelessly, from an external computer with any operating system with an SSH Client.

Another essential feature of a humanoid robot represent its cognitive abilities, for which a camera and a visual processing module are essential. In Figure 5.2 we illustrate a possible solution, by attaching 4 new hardware components: a T-shaped serial connector, an ESP32, a Raspberry Pi and a camera. ESP32 is a low-cost and low-power system on a chip microcontroller [71]. In the proposed architecture (see Figure 5.2), the ESP32 would take on the motion control, while the Raspberry Pi would be connected to the camera and perform the visual processing. Thereby, it would be possible to attach the chip or an embedded computer to the lower torso of the robot and connect it to the pair of servos located there. The Raspberry Pi Camera Module can be connected to the robot via a flex cable as shown in Figure 5.2.

In order to measure the robot's orientation, velocity and gravitational forces during the motion process, an Inertial Measurement Unit (IMU) can be connected to the robot. For instance, an MPU-6050 [55], which is a 6-axis motion device with an accelerometer

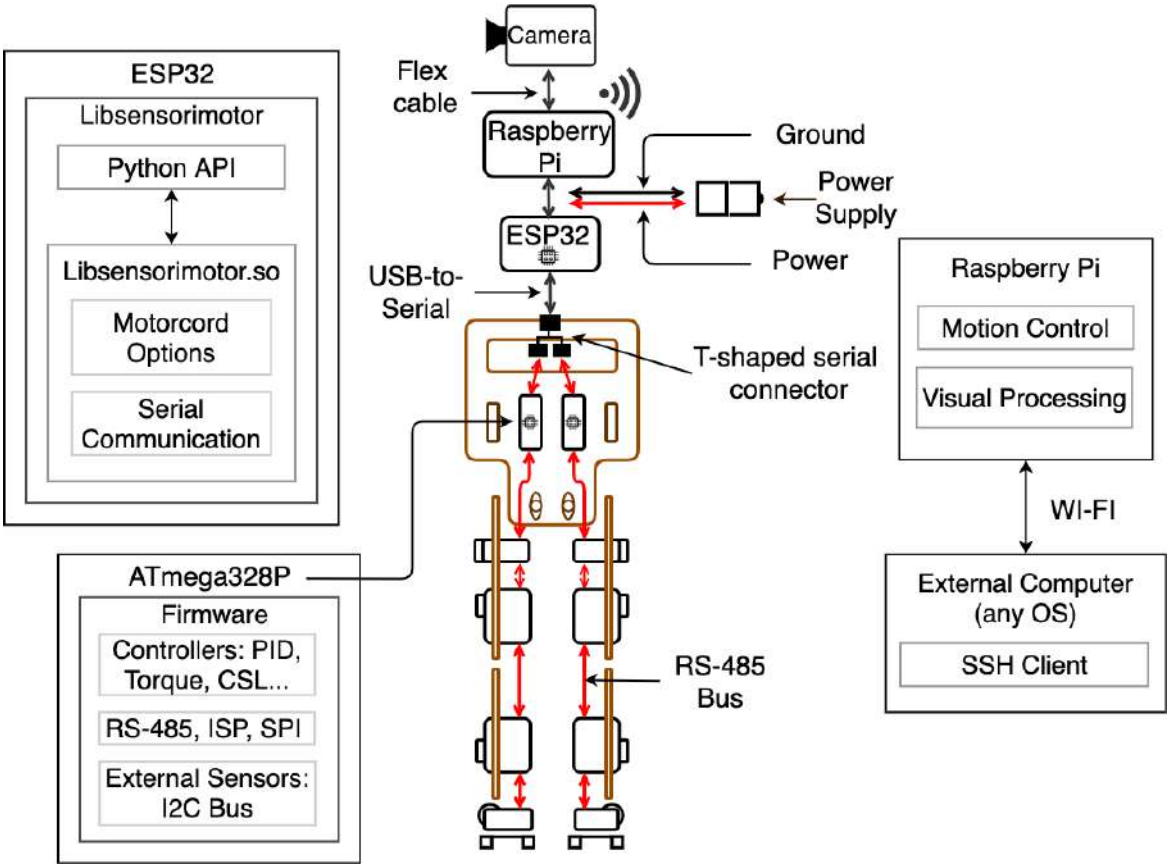

Figure 5.2: Proposed extension of the computational architecture for Gretchen robot with additional cognitive capabilities. On the top-left side, the ESP32 chip is handling the serial communication with the motorcord, while the Raspberry Pi (on the top-right side) runs the motion commands and the cognitive processes of the robot, such as visual processing. The features of the microcontroller and the external computer are the same as in Figure 5.1.

and gyroscope, could be connected to the Raspberry Pi via the I2C bus, as shown in the top part of Figure 5.1.

Any humanoid robot can benefit from having a corresponding 3D-model simulation, which can be used for testing new software and experimenting with various behaviors and motions, with no hardware exploitation. Once a camera is attached to the lower torso, the robot can be programmed to detect objects, localize itself and move to certain locations or even play football. The simulation software would also enable the use of the robot in educational projects for bigger groups of students, where there's no possibility of providing each person with a robot.

We conducted only basic motion experiments, which showed the basic motion capabilities of the robot. In future work, more complex experiments are required, in order to empirically analyze the capabilities of the robot, perform durability tests, analyze walking behaviors, measure the efficiency of the actuators in different positions, and evaluate the efficacy of the actuators in different postures.

5.3 Dissemination

The open-source nature, modular design, and low manufacturing costs of the Gretchen robot make it accessible for open-source projects within hobby communities as well as for research and educational projects within universities. Since our interest lies in allowing more university students to work with humanoid robots, we therefore present a couple of teaching directions for which the Gretchen robot could be used. The installation and development of the software can enable students to apply the acquired theoretical knowledge in practice and expand it with concepts from other disciplines such as electronics, physics and engineering. Hardware oriented teaching projects could focus on designing and 3D-modelling new body parts for the robot. Moreover, a 3D-model of the robot could be successfully used in a simulation software, where the robot's behaviors and various motion controllers can be programmed and tested. Furthermore, one could focus on the functionality of the actuators and the boards, by working with the firmware and making each actuator work more efficiently. These are just a few examples of educational and research directions based on single parts or modules of the robot, which could be adopted by universities and schools.

The Gretchen robot has already been demonstrated at two events organized by the RoboCup community, namely RoHOW and V-RoHOW. The members from both the Standard Platform (SPL) and Humanoid Leagues met Gretchen with a great deal of interest and shared their experience and ideas regarding its future development. Therefore, Gretchen could benefit from joining the Humanoid League of RoboCup and drive collaboration with the other teams who have extensive experience in designing and manufacturing humanoid robots.

5.4 Conclusion

In this thesis the Gretchen Robot Development Platform was evaluated by performing an experimental assembly, comparing it with other humanoid robots, and giving a detailed overview of its hardware components, software architecture, and motion control. The Gretchen robot is currently in a prototype stage and resembles the lower part of the human body with 10 degrees of freedom. Only widely accessible materials and electronic components, and manufacturing methods are used. Joints and drive parts are 3D-printed, limbs and the torso are laser cut from plywood. The joints are actuated indirectly through toothed belts which relieve stress on the motors. The robot is powered by 10 low-cost servo-motors controlled by custom developed control boards, the so-called Sensorimotor boards. The boards are completely open-source and can be used to drive a wide field of brushed DC motors and bring smart-servo features such as direct PWM control, various sensory feedback (position, current, voltage, temperature), RS485 bus communication and customizable firmware. The robot is capable of performing basic movements such as standing up, sitting down and walking with small steps. The modular design of the robot can enable its use for educational and research projects, by

working and experimenting with its design, actuators, 3D-printed components, firmware, power supply, communication bus, software libraries or the API. The Gretchen robot still lacks many important components, such as the upper body, more DOFs, wireless control, and visual processing, however, it has a substantial potential to become a widely used educational robot, due to its modular design, low manufacturing costs, and open-source hardware and software.

Bibliography

[1] O. Celiktutan, E. Sariyanidi, and H. Gunes, "Computational analysis of affect, personality, and engagement in human–robot interactions," in *Computer Vision for Assistive Healthcare*, pp. 283–318, Elsevier, 2018.

[2] R. A. Brooks, C. Breazeal, M. Marjanović, B. Scassellati, and M. M. Williamson, "The cog project: Building a humanoid robot," in *International Workshop on Computation for Metaphors, Analogy, and Agents*, pp. 52–87, Springer, 1998.

[3] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3, pp. 2478–2483, IEEE, 2002.

[4] E. Guizzo and E. Ackerman, "How south korea's drc-hubo robot won the darpa robotics challenge." `https://spectrum.ieee.org/automaton/robotics/humanoids/how-kaist-drc-hubo-won-darpa-robotics-challenge`, 2015. Accessed: 2020-07-02.

[5] S. Toto, "Hrp-4: Meet japan's new and awesome humanoid robot." `https://techcrunch.com/2010/09/16/hrp-4-meet-japans-new-and-awesome-humanoid-robot-video/`, 2010. Accessed: 2020-07-02.

[6] M. Schwarz, M. Schreiber, S. Schueller, M. Missura, and S. Behnke, "Nimbro-op humanoid teensize open platform," in *In Proceedings of 7th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Osaka*, Citeseer, 2012.

[7] S. Lohmeier, T. Buschmann, and H. Ulbrich, "Humanoid robot lola," in *2009 IEEE International Conference on Robotics and Automation*, pp. 775–780, IEEE, 2009.

[8] Boston Dynamics, "Atlas." `https://www.bostondynamics.com/atlas`. Accessed: 2020-07-02.

[9] K. Seo and ALDEBARAN Robotics, "Using nao: introduction to interactive humanoid robots," *AldeBaran Robotics*, 2013.

[10] A. K. Pandey and R. Gelin, "A mass-produced sociable humanoid robot: pepper: the first machine of its kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018.

[11] E. Guizzo, "France developing advanced humanoid robot romeo."

https://spectrum.ieee.org/automaton/robotics/humanoids/france-developing-advanced-humanoid-robot-romeo, 2010. Accessed: 2020-07-02.

[12] "Open automation project." http://oap.sourceforge.net/. Accessed: 2020-07-02.

[13] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *Proceedings of the first international conference on Autonomous agents*, pp. 340–347, 1997.

[14] "Robo one." https://www.robo-one.com/en/. Accessed: 2020-07-02.

[15] "Robocup humanoid league." https://humanoid.robocup.org/league/history/. Accessed: 2020-07-02.

[16] I. Ha, Y. Tamura, H. Asama, J. Han, and D. W. Hong, "Development of open humanoid platform darwin-op," in *SICE annual conference 2011*, pp. 2178–2181, IEEE, 2011.

[17] S. Yi, S. McGill, L. Vadakedathu, Q. He, I. Ha, J. Han, H. Song, M. Rouleau, D. Hong, and D. D. Lee, "Thor-op humanoid robot for darpa robotics challenge trials 2013," in *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 359–363, 2014.

[18] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke, "Child-sized 3d printed igus humanoid open platform," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 33–40, IEEE, 2015.

[19] M. Bestmann, B. Reichardt, and F. Wasserfall, "Hambot: an open source robot for robocup soccer," in *Robot Soccer World Cup*, pp. 339–346, Springer, 2015.

[20] M. Bestmann, J. Güldenstein, and J. Zhang, "High-frequency multi bus servo and sensor communication using the dynamixel protocol," in *RoboCup 2019: Robot World Cup XXIII*, Springer, 2019.

[21] M. Kubisch, "Supreme machines." https://github.com/suprememachines, 2018. Accessed: 2020-07-12.

[22] "Ai brain inc.." https://aibrain.com. Accessed: 2020-07-02.

[23] T. Simonite, "Cheaper joints and digits bring the robot revolution closer." https://www.technologyreview.com/s/525796/cheaper-joints-and-digits-bring-the-robot-revolution-closer/, 2014. Accessed: 2020-07-02.

[24] S. Takagi, "Toyota partner robots," *Journal-Robotics Society of Japan*, vol. 24, no. 2, pp. 10–11, 2007.

[25] N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, *et al.*, "icub: the design and realization of an open humanoid platform for cognitive and neuroscience research," *Advanced Robotics*, vol. 21, no. 10, pp. 1151–1175, 2007.

[26] Robotics and Mechanisms Laboratory (RoMeLa), "Charli: Cognitive humanoid autonomous robot with learning intelligence." http://www.romela.org/charli-cognitive-humanoid-autonomous-robot-with-learning-intelligence/. Accessed: 2020-07-02.

[27] "Thormang3." http://www.robotis.us/thormang3/. Accessed: 2020-07-02.

[28] E. Arts, "Robothespian brochure." https://www.engineeredarts.co.uk/robothespian/. Accessed: 2020-07-02.

[29] E. Guizzo and E. Ackerman, "Reem-c." https://robots.ieee.org/robots/reemc/, 2013. Accessed: 2020-07-02.

[30] Softbank Robotics, "Our offers: Pepper and nao." https://www.softbankrobotics.com/emea/en/our-offers. Accessed: 2020-07-02.

[31] M. Lapeyre, P. Rouanet, J. Grizou, S. Nguyen, F. Depraetre, A. Le Falher, and P.-Y. Oudeyer, "Poppy project: open-source fabrication of 3d printed humanoid robot for science, education and art," 2014.

[32] E. Guizzo, "Darwin-op humanoid robot demo." https://spectrum.ieee.org/automaton/robotics/humanoids/darwin-op-humanoid-robot-demo/, 2010. Accessed: 2020-07-02.

[33] "Nimbro-op." https://www.robotcenter.co.uk/products/nimbro-op. Accessed: 2020-07-02.

[34] IEEE Robots, "icub." https://robots.ieee.org/robots/icub/. Accessed: 2020-07-02.

[35] Z. Haider, "10 most expensive robots in the world." https://abouticles.com/expensive-robots/5/, 2019. Accessed: 2020-07-02.

[36] IEEE Robots, "Bruno." https://robots.ieee.org/robots/dribbler/. Accessed: 2020-07-02.

[37] UBTECH, "Yanshee." http://www.ubtechedu.com/global/pro_view-4.html. Accessed: 2020-07-02.

[38] M. Cortial and Génération Robots, "Poppy humanoid robot assembly guide." https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md, 2016. Accessed: 2020-07-02.

[39] "How to make a darwin-op robot clone on a diy 3d printer." https://www.3ders.org/articles/20130209-how-to-make-a-darwin-op-robot-clone-on-a-diy-3d-printer.html, 2013. Accessed: 2020-07-02.

[40] M. Kubisch, "Gretchen assembly documentation." https://github.com/aibrainag/Gretchen/blob/master/documentation/documentation.adoc, 2018. Accessed: 2020-07-02.

[41] F. Nori, "The design of the icub humanoid robot." https://www.researchgate.net/publication/

260506628_The_Design_of_the_iCub_Humanoid_Robot/figures?lo=1/. Accessed: 2020-07-02.

[42] IEEE Robots, "Darwin-op." https://robots.ieee.org/robots/darwin/. Accessed: 2020-07-02.

[43] Generation Robots, "Poppy humanoid robot raspberry pi version." https://www.generationrobots.com/en/403348-robot-poppy-humanoid-version-raspberry-sans-impressions-3d.html. Accessed: 2020-07-02.

[44] Bit-Bots, "Hambot documentation version 0.1." https://github.com/bit-bots/hambot/blob/master/doc/hambot_documentation.pdf, 2015. Accessed: 2020-07-02.

[45] "Rohow: Robotic hamburg open workshop." https://rohow.de/2019/en/. Accessed: 2020-07-02.

[46] "Program of the virtual robocup humanoid open workshops." https://humanoid.robocup.org/virtual-rohow-2020/program/. Accessed: 2020-07-02.

[47] "[robocup][v-rohow] "gretchen" – a humanoid open hardware platform for education and research." https://www.youtube.com/watch?v=tsjLt30-Pxw&t=2495s, 2020. Accessed: 2020-07-02.

[48] J. Stewart, "Plywood advantages and disadvantages." https://www.doityourself.com/stry/how-to-stain-particle-board, 2019. Accessed: 2020-07-02.

[49] M. Shoham, *A Textbook of Robotics 1: Basic Concepts*. Springer Science & Business Media, 2012.

[50] C. Schubert, M. C. Van Langeveld, and L. A. Donoso, "Innovations in 3d printing: a 3d overview from optics to organs," *British Journal of Ophthalmology*, vol. 98, no. 2, pp. 159–161, 2014.

[51] O. Higgins, "Virtual virtual reality pan / tilt control with a vr-style experience." https://diyodemag.com/projects/virtual_virtual_reality, 2018. Accessed: 2020-07-02.

[52] Hitec, "Announced specification of hs-805mg metal gear mega scale servo." https://asset.conrad.com/media10/add/160267/c1/-/en/000209902DS01/datasheet-209902-hitec-custom-servo-hs-805mg-analogue-servo-gear-box-material-metal-connector-system-jr.pdf, 2008. Accessed: 2020-07-02.

[53] Philips, "General silicon sensors for temperature measurement." https://www.mikrocontroller.net/attachment/243481/KTY-Philips.pdf, 2000. Accessed: 2020-07-01.

[54] Atmel, "Avr910: In-system programming application note." http://ww1.microchip.com/downloads/en/appnotes/atmel-0943-in-system-programming_applicationnote_avr910.pdf, 2016. Accessed: 2020-07-01.

[55] "Mpu-6050 six-axis (gyro + accelerometer) mems motiontracking devices." https:

//invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/. Accessed: 2020-07-02.

[56] O. Higgins, "I2c info – i2c bus, interface and protocol." https://i2c.info/#:~:text=I2C, 2020. Accessed: 2020-07-20.

[57] Atmel, "Atmel atmega328/p [datasheet]." https://datasheetspdf.com/pdf-file/1057332/ATMEL/ATmega328P/1, 2016. Accessed: 2020-07-20.

[58] Arduino, "Arduino uno rev3." https://store.arduino.cc/arduino-uno-rev3. Accessed: 2020-07-20.

[59] M. Currey, "Arduino / atmega 328p fuse settings." http://www.martyncurrey.com/arduino-atmega-328p-fuse-settings/, 2014. Accessed: 2020-07-20.

[60] P. Christensson, "Firmware definition." https://techterms.com/definition/firmware, 2006. Accessed: 2020-07-02.

[61] "Definition of:eeprom." https://web.archive.org/web/20130520014604/http://www.pcmag.com/encyclopedia/term/42403/eeprom. Accessed: 2020-07-02.

[62] myAVR, "mysmartusb light - avr isp programmer." http://shop.myavr.com/index.php?sp=article.sp.php&artID=200006, 2016. Accessed: 2020-07-20.

[63] B. Stiller, "Rs-485 basics: Introduction." https://e2e.ti.com/blogs_/b/industrial_strength/archive/2015/04/28/rs-485-basics-introduction?tisearch=e2e-sitesearch&keymatch=RS-485, 2015. Accessed: 2020-07-10.

[64] M. Barr, "Pulse width modulation," *Embedded Systems Programming*, vol. 14, no. 10, pp. 103–104, 2001.

[65] D. Cook, "Dc motor-driver h-bridge circuit." http://robotroom.com/HBridge.html. Accessed: 2020-07-02.

[66] "H-bridge." https://www.uni-weimar.de/kunst-und-gestaltung/wiki/H-Bridge. Accessed: 2020-07-02.

[67] M. Hild, "Defying gravity-a minimal cognitive sensorimotor loop which makes robots with arbitrary morphologies stand up," in *11th International Conference on Accomplishments in Electrical and Mechanical Engineering and Information Technology (DEMI)*, pp. 23–34, 2013.

[68] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 559–576, 2005.

[69] M. Kubisch, "First steps of gretchen, an open-source bipedal robot.." https://www.youtube.com/watch?v=ubMeLkMhT9Y&feature=youtu.be, 2020. Accessed: 2020-07-02.

[70] "What is a raspberry pi?." https://opensource.com/resources/raspberry-pi.

Accessed: 2020-07-02.

[71] "Esp32 series." https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Accessed: 2020-07-02.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 31. Juli 2020 .....