Integration of a Unitree Go1 quadruped robot into a university ecosystem

M a s t e r work

Hof University of Applied Sciences Faculty of Computer Science
Master's degree program in Computer Science

Submitted to
Prof. Dr. Christian Groth
Alfons-Goppel-Platz 1
95028 Hof

Submitted by
Noah Lehmann

Hof, September 15, 2023

Thanks to my friend and mentor Prof. Dr. Jürgen Heym, who encouraged me to study for a Master's degree and always supported me during my studies.

Table of contents

List of illustrations

List of abbreviations

.....APNAccess Point Name

.....BMSBattery Management System

.....CPUCentral Processing Unit

.....DARPADefense Advanced Research Projects Agency

.....DHCPDynamic Host Configuration Protocol

GB.....Gigabyte

.....Go1Unitree Robotics Go1 Edu

.....GPSGlobal Positioning System

.....GPUGraphics Processing Unit

.....HDMIHigh Definition Multimedia Interface

.....hostapdHost Access Point Daemon

HTML5 . . . . . Hypertext Markup Language V.5

HTTP .....Hypertext Transfer Protocol

.....IPInternet Protocol

AI .....Artificial intelligence

LED.....Light emitting diodes

.....LidarLight Detection and Ranging

.....LTELong Term Evolution

.....LXDELightweight X11 Desktop Environment

.....MCUMain Control Unit

.....MITMassachusetts Institute of Technology

.....MLMachine Learning

NM.....Newton/meter

.....PDFPortable Document Format

PIN.....Personal Identification Number

PIXEL.....Pi Improved Xwindows Environment, Lightweight

RCTA.....Robotics Collaborative Technology Alliance

.....ROIReturn on investment

.....ROSRobot Operation System

RTSP.....Real-Time Streaming Protocol

.....SDSecure Digital

.....SIMSubscriber Identity Module

.....SLAMSimultaneous Localization and Mapping

.....SoCState of Charge

.....SSDSolid State Drive

.....	SSHSecure Shell
.....	SSIDService Set Identifier
.....	URLUniform Resource Locator
.....	USUnited States
USA	United States of America
.....	USBUniversal Serial Bus
USD	US dollar
VDI	Association of German Engineers
VPN	Virtual Private Network
WebRTC	Web Real Time Communications
.....	WLANWireless Local Area Network
.....	WWANWireless Wide Area Network

Listings

List of tables

1 Introduction

The constant growth of global economic areas and the constant renewal of known mechanisms in development, production and sales require a constant increase in efficiency in all areas that influence the key figures for measuring the success of the economy. One common way to increase efficiency is to automate various processes, which often goes hand in hand with the elimination of human error potential during implementation. Approaches to this often include the use of artificial intelligence and highly specialized machines, and in many cases even a combination of both. The result is a fast and precise response to given influences and data.

This phenomenon has been observed for decades in the industrial sector, particularly in production, material handling and warehousing. Machines are configured in such a way that they can take over individual steps of a production chain efficiently and largely error-free, which reduces costs and simplifies quality assurance, especially compared to the use of human labor. From a certain degree of automation, such machines are called robots, which is particularly appropriate in the area of human support and partial replacement, as the word *robot* comes from the Czech *robota*, which means something like service, i.e. supporting humans.

With the spread of robots in the industrial sector and their use in various scenarios worldwide, the academic and consumer-oriented sectors have also become aware of the concept and have embraced it. Robots not only support industrial processes, but have also been used for many years in the consumer-oriented environment in the toy industry and as household helpers. For these consumer-oriented and also well-known industrial purposes, research is increasingly addressing the topic and developing new types and uses of robotics worldwide and evaluating them in real-life applications in areas such as efficiency, human-machine communication and the limits of what is possible. For example, institutions such as *Boston Dynamics*, *MIT (Massachusetts Institute of Technology)* and the Chinese robot manufacturer *Unitree Robotics* have recently worked on highly developed four-legged robots whose locomotion and appearance are very reminiscent of animals such as big cats or dogs. In addition to developing these so-called *quadruped robots*, academic institutions such as MIT are also researching possible applications and their usefulness. The possibilities and limits of such robots must also be constantly tested, especially when new models with new promises and functions come onto the market.

This thesis deals with this topic. As part of the acquisition of two quadruped robots of the *Go1* type *(Unitree Robotics Go1 Edu)*, the robots are initially examined in two phases, their functions tested and any difficulties documented. The first phase of this thesis will deal with the robots themselves and work out the strengths and weaknesses of the devices.[1] will deal with the actual use of the robot in a first possible environment.

1.1 Unitree Robotics Go1 Edu

In order to provide a basis for further work, the following chapter will briefly discuss the robot, which will be the subject of all non-fundamental work in this thesis.

Robot manufacturer *Unitree* was founded in 2016 by Wang Xingxing in the Chinese province of Zhejiang. After the first designs of its own quadruped robots, the Go1 robot was released in June 2021. Since then, the manufacturer has been promoting it as the first affordable and consumer-oriented quadruped robot and thus as a milestone in accessible robotics.[2] According to the manufacturer, the robot can be purchased for around 2700 USD (US dollars) in the basic version (Go1 Air), which, however, does not offer the option of expanding the functionalities or accessing the robot's internal components. In addition to the basic version, Unitree also offers the Go1 in the *Pro* and *Edu* versions, which have a number of sensor and computing power enhancements compared to the *Air*. The Go1 Edu is the most comprehensive and expensive version of the Go1 and has an extensive range of sensors, cameras and computing units, which are listed in more detail later in Chapter 3. This thesis deals exclusively with the Unitree Go1 Edu. Figure 1 shows the Go1 as advertised on the manufacturer's website.

The successor in the *GO* series - the Unitree GO2 - had already been released during the period in which this work was being written. With an even lower entry-level price of USD 1600 and, judging by the specifications, significantly improved and extended functionalities, it can be regarded as a consumer-friendly alternative to the Go1.

---

[1] Jonas Kemnitzer. "Development of an intelligent lidar-based 3D navigation system for the Unitree GO1". Master thesis. Hof University of Applied Sciences, 2023.

[2] Unitree Robotics. *About Unitree Robotics*. URL: https://www.unitree.com/en/about/ (visited on 28. 08. 2023).

[3] Unitree Robotics. *Unitree Go1*. URL: https://shop.unitree.com/cdn/shop/products/75_540x.jpg?v=1668073774 (visited on 28. 08. 2023)

Figure 1: Sales image of the Unitree Go1[3]

1.2 Procedure

The following work basically consists of four parts: the basics, the robot architecture, the analysis of the robot and the functional extension of the robot. In the first part, the basics, a general overview of the topic of robotics will be provided, which will serve in particular to classify the Go1 in question in the field in general. This will be followed by an overview of the current state of research and industry in the field of robotics. To this end, work and publications as well as products in the field of quadruped robotics and their use will be shown first. Then resources on Go1 that are particularly useful when working with the robot will be shown. This includes not only official resources, but also related publications and unverified resources from a non-academic environment, which should nevertheless be taken into account due to their scope. As a conclusion to the basics, some challenges in working with the Go1 are listed, some of which will be taken up again in the course of the work.

After establishing the basics of robotics in general and the more specific information on the Go1, it is examined in more detail. The structure of the robot is described in detail and illustrated using graphics and photographs. After an overview of the robot's structure, the installed mechanical components, the sensors and the computing units are documented. This is followed by a detailed analysis of the computing units installed, their individual functions and the communication between them

with each other and with the user. Finally, the technical limitations of the Go1 will be shown.

Based on the knowledge gained about the robot's structure, the robot's function is documented in the analysis. After describing the commissioning of the Go1, the majority of the functions that are possible ex works are documented and evaluated. Functions that could not be tested as part of the work are listed as additions.

Finally, some functions from the previous chapters are taken up and extended. Useful functions for the real use of the robot are developed and possibilities for eliminating or relativizing some limitations are shown. These will then be integrated into the objectives of the work, after which the work with the Go1 will be evaluated once again.

1.3 Objective

The following chapter is intended to narrow down the scope of this work and clearly define the objectives. An evaluation of what has been achieved will take place at the end of this work.

Goals

The objectives of this thesis are derived from the title of this thesis - *Integration of a Unitree Go1 Quadruped Robot into a University Ecosystem*. This *integration* of the robot can be achieved in various aspects of the university ecosystem. This includes the aspects relevant to this work, i.e. *research* on the robot, *teaching* using the robot as an example and the *use of* the robot in real-life scenarios. The use of the robot in the university ecosystem will only be presented as an example in this thesis. Elaborations that require a certain classification of the findings can exemplify possible uses. However, an actual implementation of a project using the robot for use on the university campus is not realized within the scope of this work.

The second relevant aspect of the ecosystem is research, which, in combination with teaching, can particularly benefit from the acquisition of a Go1 by the university. As a basis for further work on the robot, in particular testing its limits, exploring new possible applications or the interaction of the robot with humans, a large part of the functions are to be tested in detail and their limitations, if any, presented.

A similar principle applies to teaching: the presentation of the functions and documentation of the details of the robot should make it easier for teachers and students to carry out work and experiments on the robot. The documentation of the structure should also make it easier for newcomers to the field of quadruped robotics to gain an overview of the function of such a robot.

In summary, this work is a kind of *survey* of the Go1 and its functions. This involves inspecting it in detail, documenting all of its components, testing and documenting the most important functionalities and, through extensions, eliminating the most serious limitations of the robot and facilitating its simplest applications.

Delimitation

The robot contains various preparations on the subject of AI (artificial intelligence), which will be mentioned as much as possible in this paper. The implementation of these functions is also not documented. These functions include topics such as camera image optimization, environment recognition using ultrasound and lidar (light detection and ranging), object recognition using the various sensors and optimization of motion sequences. The mechanical and electrotechnical principles of the motor system and the movement apparatus are also not described further.

2 Basics

The following chapter describes robotics and the possible classification of Go1 within it. To this end, robotics is defined, categorized and the relevant sub-areas are examined in more detail. As this thesis deals with a Go1, it is classified and categorized into the areas of research, industry and further use. To further understand the aim of the thesis - the analysis and integration of the Go1 into an existing ecosystem - the current state of research is examined in more detail. Of particular relevance here are the insights already gained into the use of the same or similar robots, as well as the integration of other models into existing ecosystems. For this purpose, not only research alone will be considered, but also the work of private companies and developers. With the knowledge of the correct classification of Go1 and the state of research on related topics, the challenges of this work should be emphasized.

2.1 Robotics

Robotics deals with the field of knowledge surrounding *robots*. The term *robot* comes from the Czech word *robota*, which means something like *compulsory service*. As apt as this translation is for today's use of the word, this description is also vague. The recognized definitions of the word *robot* are similarly unclear. A common definition in German-speaking countries is that of the VDI (Association of German Engineers):

> [Robots] are universally applicable automatic motion machines with several axes whose movements are freely programmable (i.e. without mechanical intervention) in terms of movement sequence and paths or angles and may be sensor-guided[4].

Even though the VDI guideline has since been withdrawn, the definition is still frequently quoted due to its accurate definition of the term.

A precise and recognized definition of the word *robot* has hardly been found to date, which also makes it difficult to divide robotics into sub-areas. Since no elaborate categorization of robotics is necessary for this work and the classification of Go1 , a division of robots into two classes is used: industrial robots and service robots. Figure 2 shows an overview of the two classifications and their characteristics.

[4]VDI Society for Production Engineering. *VDI 2860/ Assembly and handling technology. Handling functions, tions, handling equipment; terms, definitions, symbols*. Beuthe Publishing House, 1990.

[5]Sophie Fischer. *Robotics - Market data analysis & forecasts*. Statista Technology Market Outlook. Statista, Aug. 2022. URL: https://de.statista.com/statistik/studie/id/116785/dokument/robotics-report/ (visited on 04. 07. 2023)

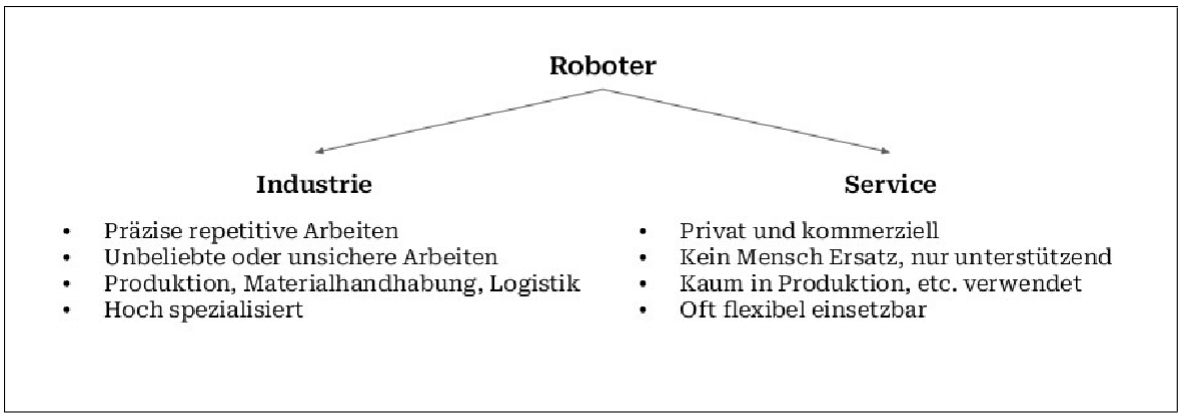

Figure 2: Comparison between industrial and service robots[5]

2.1.1 Industrial robots

Industrial robots are robots that are used in industrial environments and usually replace humans in unsafe, unprofitable or undesirable - repetitive - tasks. They are largely stationary due to their high degree of specialization in their intended use. The advantage of industrially used robots is the high efficiency and accuracy of the work and their adaptability to the intended use. The disadvantage is the high cost of implementation, as a highly specialized robot must be created anew for each application. Applications for industrial robots include welding, painting, assembly and material handling in production environments. The limitation of use and the strict classification of industrial applications leads to the conclusion that the Go1 cannot be classified as an industrial robot. Instead, it can be categorized in the simple subdivision used in this work to the service robots.

2.1.2 Service robots

Service robots essentially differ from industrial robots in the assumption that they do not replace humans in their intended use, but support them or are supported by humans. The term *service robot* is intentionally broad and, within the scope of this paper, only excludes industrial robots as described in the previous paragraph. In the case of service robots, a fundamental distinction can be made between commercially used and consumer-oriented robots[6].

Service robots are generally not stationary, as they are much more flexible in their possible applications than industrial robots. The advantage of flexibility can also be

[6]Fischer, see note 5.

the disadvantage of complexity. The wider field of application of service robots generally increases the development effort, but also increases the robot's yield.

If one wishes to further subdivide the class of service robots, the following subdivisions are conceivable:

- Toy robot
- Exploration robot
- Military / combat robots
- Assistant robot

It should be noted that the simple classification of robots into only two classes - industrial robots and service robots - was chosen so as not to limit the applications of the Go1 in this work. As already explained, the Go1 can be used in a variety of ways, but it is not suitable for completely replacing humans in its areas of application, nor is it strictly limited to industrial purposes. It is therefore not possible to classify it as an industrial robot. A more precise classification within the class of service robots is possible, but depends on the final purpose of the quadruped robot. Possible applications in the university environment are outlined in chapter 2.2.2.

2.1.3 Cobots

Another suitable term that does not contradict the classification of the Go1 as a service robot is *cobot*. The word *cobot* is made up of the English word *collaborate* and the word *robot*. Some characteristics are:[7]

- Collaborative and secure, as opposed to stationary and secured
- Interactive and adaptive to the environment
- Simple commissioning thanks to forward-looking development
- Flexible applications
- Faster ROI (return on investment)

[7]Fischer, see note 5.

Due to its flexible locomotion options and the high number of sensors and expansion options of the Go1 as well as the built-in loudspeaker for communication[8] the Go1 can also be described as *a cobot* in addition to its classification as a service robot.

2.1.4 Quadrupedal

The last classification of the Go1 is the term quadruped robotics. According to the *encyclopedia of biology, quadrupedal robotics* is a four-legged mode of locomotion in which all four extremities are in contact with the ground[9]. Quadruped robots are therefore strongly based on biology, which is confirmed by the rest of the Go1's external form. The term *quadruped robot* therefore only means that the robot in question uses four legs to move around.

2.2 State of research

Much research has already been carried out in the field of quadruped robots. The work usually focuses on controlling the motors for locomotion, autonomous navigation of the robots in unknown environments and testing the possible applications of the quadruped robots. This work, on the other hand, deals with the integration of the robot into a university ecosystem, which does not anticipate the possible applications. For this reason, only generally interesting research on quadruped robots is shown briefly below, after which general research on the integration and use of service robots is presented. Finally, work specifically on the Go1 model will be shown. These are not necessarily of academic origin and are intended to provide the reader with an overview of the available resources on the Go1 model.

2.2.1 Quadruped robot

The first implementation of a quadruped robot to attract public attention is the so-called *Big Dog* from *Boston Dynamics* in Boston, USA (United States of America). This was developed in 2008 out of military interest in an all-terrain alternative to conventional military vehicles and was therefore also funded by the American DARPA (Defense Advanced Research Projects Agency). *Big Dog* was developed with hydraulic extremities to enable it to carry heavy luggage in military operations. The flexibility of the four legs and the ability

[8] See chapter 3.1.3

[9] Spectrum. *Encyclopedia of Biology*. Spektrum Akademischer Verlag Heidelberg, 1999.

This ability to move relatively quickly through difficult terrain gave *Big Dog* a potential advantage over conventional means of locomotion based on tracks or wheels. Further developments on the *Big Dog* were later funded by the RCTA (Robotics Collaborative Technology Alliance) of the US (United States) Army Research Laboratory.[10,11]

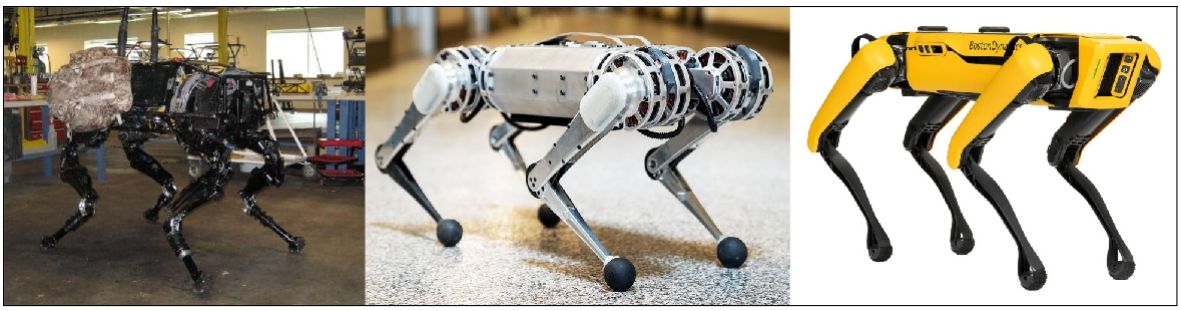

Figure 3: Big Dog (left), MIT Cheetah 3 (center) and Spot (right)[12]

In 2009, MIT's *Biometric Robotics Lab* announced a project called *Cheetah*. The aim of the project was to develop a quadruped robot that could compete with the real animal world in terms of speed and energy efficiency. This project was also funded by DARPA[13,14]. In contrast to the *Big Dog*, however, the *Cheetah* was never promoted commercially or militarily. Instead, the robot's focus is on research. As early as 2012, the *Biometric Robotics Lab* published the first videos of the robot in action, which implies that the robot was already completed at that time. Since the project announcement, several iterations of the robot have been developed. The current iteration of the robot is optimized in its design and therefore cost efficient to purchase and repair. Unlike the *Big Dog*, the MIT *Cheetah* is fully electric, which has the goal of simplifying development.[15] Many

[10]Marc Raibert et al. *BigDog, the Rough-Terrain Quaduped Robot*. Boston Dynamics, Apr. 8, 2008.

[11]Defense Advanced Research Projects Agency. *Big Dog*. URL: urhttps://www.darpa.mil/about-us/timeline/big-dog (visited on 07. 08. 2023).

[12]Boston Dynamics. *Photo of a BigDog*. URL: https://www.bostondynamics.com/wp-content/uplo ads/2023/07/bigdog-1-1.jpg (visited on 28. 08. 2023)

MIT. *Photo of a Cheetah 3*. URL: https://news.mit.edu/sites/default/files/styles/news_article image_gallery/public/images/201903/MIT- Mini- Cheetah- 01_0.jpg (visited on 28. 08. 2023)

Boston Dynamics. *Photo of a spot*. URL: https://bostondynamics.com/wp-content/uploads/2023/05/spot-explorer-web-2-2048x1152.jpg (visited on 28. 08. 2023)

[13]Devan Joseph. "MIT reveals how its military-funded Cheetah robot can now jump over obstacles on its o wn". In: *Business Insider* (June 3, 2015).

[14]Defense Advanced Research Projects Agency. *Maximum Mobility and Manipulation (M3)*. URL: https://www.darpa.mil/program/maximum-mobility-and-manipulation (visited on 07. 08. 2023).

[15]Jason Falconer. "MIT Cheetah Robot Runs Fast, and Efficiently. It's now the second fastest legged robot in the world". In: *IEEE Spectrum* (May 14, 2013).

Findings from the MIT *Cheetah* project are still being used in new projects today. This also applies to the implementation of the Go1, although this will be discussed less in this paper. Probably the best-known current implementation of a quadruped robot is the *Spot* developed by *Boston Dynamics*, which was publicly advertised in its first form back in 2015. Unlike its predecessor robots at *Boston Dynamics*, it was the company's first robot to be 100% electrically powered. This makes it look very similar to MIT's *Cheetah*, but the *Spot* platform had a focus on commercial or military use from the outset. What is particularly noteworthy about the robot is its compactness compared to its predecessors. The current version of the *Spot* is already commercially available and was initially advertised for sale for around 75,000 US dollars.[16] for purchase.[17] *The* illustration *The Rise of the Robot Quadrupeds* provides an overview of the timeline of the most influential implementations of quadruped robots in recent history. Past[18].

2.2.2 Integration of robots

In the field of robotics, in addition to the focus on testing the possibilities, there is always the search for a meaningful use of the knowledge gained and the robots developed in real scenarios. The possibilities here are hardly limited and always depend on the form and maturity of the robot implementations. Some current approaches to the integration and use of quadruped robots in real scenarios are presented below.

The possible applications of quadruped robots listed below are merely a selection of the obvious potential and are only intended to provide an overview. The aim here is not to provide a complete list of integration approaches.

Control and monitoring

Due to their ability to move around in unstructured environments, quadruped robots are well suited to taking over the control function of humans in more dangerous areas. In addition, a sufficiently developed robot can take over tasks autonomously and perform them at regular intervals, which represents a major advantage over humans. One possible application for quadruped robots is

[16]Evan Ackerman. "Boston Dynamics' Spot Robot Dog Now Available for $74,500. For the price of a luxury car, you can now get a very smart, very capable, very yellow robotic dog". In: *IEEE Spectrum* (June 16, 2020). [17]Boston Dynamics. *Legacy Robots*. URL: `https : / / bostondynamics . com / legacy/` (visited on 08. 08. 2023).

[18]Jeremy Moses and Geoffrey Ford. *The Rise of the Robot Quadrupeds*. Dec. 11, 2020. URL: `https://map pinglaws.net/rise-robot-quadrupeds.html` (visited on 07. 08. 2023).

According to the article "Real-Time and Remote Construction Progress Monitoring with a Quadruped Robot Using Augmented Reality", monitoring and checking progress on construction sites. The authors describe the manual inspection of a construction site by a human inspector as time-inefficient, unstructured and unreliable. They also criticize the fact that human inspections are hardly repeatable due to the smallest deviations in the system and are therefore difficult to classify. Their approach of virtualizing the environment to be inspected and then comparing it with the results to be achieved using automated processes and scans by automated robots does present some hurdles, but promises a consistent result over the course of the construction site. In addition, the authors emphasize the time savings and thus the efficiency of the approach, as even with only partially automated implementation, distributed monitoring is possible, where only the robot with sensors and cameras needs to be on site.[19]

In the area of integrating such a quadruped robot in the university environment, this approach can be translated into automated modeling of the campus, for example. It is not necessary to create an exact model beforehand with this approach; it is simply a matter of using the possibility of modeling with sensors such as ultrasound, lidar and cameras.

Security

As the environment in urban environments is largely unstructured, the use of dedicated robots for operation on roads or in the air is often considerably restricted. In addition to the more flexible maneuverability of quadruped robots, their expandability is another advantage over simpler robots based on wheels or drones. According to the study "Robotics' Role in Public Safety", the addition of an arm with a gripper to a standard spot *robot* makes it possible to minimize danger for humans in applications such as checking suspicious objects and possibly removing them from a danger zone, detecting potentially dangerous substances in the environment (radiation, leaks, liquids) or checking an environment in advance to give emergency services a better overview in emergency situations without putting themselves in danger. In their study, *Boston Dynamics* particularly emphasize the high modularity of quadruped robots, which makes them flexible to use.[20]

[19]Srijeet Halder et al. "Real-Time and Remote Construction Progress Monitoring with a Quadruped Robot Using Augmented Reality". In: *Buildings* (Dec. 2022). URL: https://doi.org/10.3390/buildings12112 027.

[20]Boston Dynamics. "Robotics' Role in Public Safety. How robots like Boston Dynamics' Spot are keeping people safe." In: (2023).

For the Go1 in the university environment, it would be conceivable to use the robot to search for casualties in the event of a fire. It would also be conceivable to walk through potentially dangerous laboratories and check for leaks. During closing times, quadruped robots such as the Go1 could also be used to search the campus for unauthorized visitors.

Automation and efficiency

As explained in chapter 2.1, the origin of the word *robot* reveals one of the core objectives of robotics. The focus here is on automating and relieving humans of simple or less desirable tasks. Robots can take over simple and repetitive tasks and sometimes perform them with greater precision and endurance than the human they replace could have done. If this insight is translated to quadruped robots, new automation possibilities arise in unstructured environments, such as agriculture in particular. The paper *Towards Computer-Vision Based Vineyard Navigation for Quadruped Robots*, for example, describes the use of a quadruped robot to automatically control vines in uneven vineyards and, with the help of an extension, to shorten them so that the plant does not waste unnecessary resources. The same can also be applied to other areas of agriculture. For example, quadruped robots could partially replace or increase the frequency of inspections by hunters and foresters.

Here, too, a bridge can be built to integration into a university ecosystem. An automated robot could, for example, control and enforce closing times on several levels of a university building. Running and checking the locks and, if necessary, locking them is possible with little expansion effort.

Service

Quadruped robots are also helpful in the area of general human support, especially in unstructured environments. For example, the *Big Dog* described in chapter 2.2.1 was developed to carry heavy luggage in war zones and combat operations. This relieves the human soldier immensely and allows him to take on other tasks.[21] This load-carrying approach is also conceivable in non-military applications. Another approach is that of navigation assistance. By moving flexibly over uneven surfaces, such a quadruped robot can help a human to navigate to a destination.

[21]Raibert et al, see note 10.

This scenario is also conceivable in a university environment. For example, the robot could receive new guests and, on request, provide navigation instructions to the desired locations across the campus and also accompany the guests.

2.2.3 Unitree Go1 Resources

The following lists and evaluates some resources that are helpful for working on Go1. These are not exclusively academic in nature and must therefore be viewed with particular caution. Nevertheless, these resources are particularly helpful and can provide a good basis and supplement to the results in the course of the work.

Official documentation

The website https://www.docs.quadruped.de/projects/go1/html/index.html [22] provides a basic overview of the structure of the Go1 and the functions included with the robot. It contains all the necessary information that the developers of the robot have collected and published. The site offers a good introduction to using the instructions directly on the robot without having to understand them further.

All official instructions are also documented on the /operation.html subpage, which can be downloaded as a PDF (Portable Document Format). The website also contains some simple configuration instructions for the software extensions that are installed and activated ex works.

In addition to the general documentation of the robot, some *GitHub* repositories and documentation on individual functions of the Go1 are also provided. The most important ones are

- **Unitree-Legged-SDK**
  https://github.com/unitreerobotics/unitree_legged_sdk
  → A library to influence the control of the robot
- **Unitree-Camera-SDK**
  https://github.com/unitreerobotics/UnitreecameraSDK
  → A library to use the robot's cameras
- **Unitree-Actuator-SDK**
  https://github.com/unitreerobotics/unitree_actuator_sdk
  → A library that controls the installed motors individually

[22] Unitree Robotics. *GO1 Manuals - GO1 Tutorials 1.0.0 documentation*. URL: https://www.docs.quadruped.de/projects/go1/html/index.html (visited on 08. 08. 2023).

- **Unitree-ROS2-to-Real**
  https://github.com/unitreerobotics/unitree_ros2_to_real
  → A library that can replace the outdated ROS (Robot Operation System) version 1 on the robots

It should be noted that although the official documentation on the website and the GitHub repositories provide a good introduction to the direct use of the robot, it is mostly incomprehensibly written, incomplete and partly not translated, but published in the original *Chinese* language. It is therefore recommended in this paper to also use the alternative sources of knowledge listed later, albeit with a critical eye, as many have not been officially confirmed or checked.

MIT Cheetah Documentation

The mechanical structure and general appearance of the Go1 are similar to those of the MIT *Cheetah* 3 and even identical in parts. A visual comparison can be made using Figure 3. It can therefore be assumed that *Unitree Robotics* reused parts of the results of the research on the MIT *Cheetah* 3 for the development of the Go1. Although unclear shortly after publication, the license of the MIT software used in the Go1 is now recognized and referenced. For this reason, it is advisable to consult the excellently documented findings on the MIT Cheetah 3 if you want to know more about the structure and function of the Go1. The article "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot", which appeared parallel to the publication of the Cheetah 3, provides an introduction.

Since the Cheetah 3 was released, MIT has published further research articles about it. A look at the website of the *MIT Biometric Robotics Lab* - https://biomimetics.mit.edu/publications - where the Cheetah 3 and its predecessors were developed, is helpful here. For a more technical overview of the hardware and software controls of the MIT Cheetah 3 and thus also the Go1, the *GitHub repository* of the Biometrics Robotic Lab is helpful[23].

In contrast to the official documentation from *Unitree Robotics*, the findings from the MIT documentation are well-founded and comprehensible. Unfortunately, the overlaps between the two robots only relate to the general concept and parts of the specific implementation of the Go1; the exact functionality of the two robots is very different.

[23] https://github.com/mit-biomimetics/Cheetah-Software

Unofficial repositories

Due to the low financial barrier to purchasing a Go1, a comparatively large and active community quickly formed after the robot was released, which supports and exchanges information with each other when working with the Go1. Within the community, a few members have distinguished themselves by creating well-documented repositories with application examples and implementations of new functions. As the real names of these users do not emerge from the forums and repositories, their references are acknowledged and categorized here.

- https://github.com/MAVProxyUser/YushuTechUnitreeGo1
  A very detailed repository that takes a user-focused approach to the robot's documentation. The author not only describes the structure in simple terms, but also provides simple instructions for configuring and using individual functions. Especially for beginners, this is a good guide for getting closer to advanced use.

- https://github.com/maggusscheppi/Go1
  A less detailed repository, which nevertheless offers good approaches to solving common problems and also contains pragmatic implementations of new functions. Recommended for battery management and monitoring[24].

- https://github.com/Bin4ry/free-dog-sdk
  An elaborate repository that deals with the reverse engineering of the official Unitree libraries. The author's results can be used to equip the *Air* and *Pro* versions with some of the functions of the *Edu* or to better understand the libraries.

Unofficial forums

As already mentioned, the user base and community around the Go1 has grown considerably since the robot was released and went on sale. New users of the Go1 should therefore not be afraid to look for forums and similar resources to contact more experienced owners of the robot if they have any problems. Probably the most present and active forum is *the* Slack *channel The Dog Pound animal control for Stray robot dogs*[25]. New users are welcome to join, as is active participation. The link in

[24] See chapter 5

[25] https://join.slack.com/t/robotdogs/shared_invite/zt-1fvixx89u-7T79~VxmDYdFSIoTnSagFQ

of the footnote can be used to join the channel, unfortunately it is not possible to use the resource without joining.

2.3 Challenges

In conclusion to the basics of robotics and the state of research on integration, quadruped robotics and the Go1, some of the challenges that need to be considered when working with the robot and which will be partially mastered in the course of the work will be described.

Documentation

As can be seen from the previous chapter 2.2.3, the official documentation from *Unitree Robotics* is inadequate, which is why a look at related and unofficial resources is also recommended. Not only the general use of the robot in its delivery state, but also the supplied and extending libraries are hardly or insufficiently described. For example, many libraries for extending the basic function are hardly commented or documented. This challenge in dealing with the Go1 is to be eliminated in the course of the work for large parts of the basic functionalities and for the extensions in chapter 5. One of the aims of the work is to document the use of and work on Go1 in such a way that it can be used more efficiently in the university environment in the future - as an independent part of the ecosystem, but also as an extension of teaching.

Outdated resources and libraries

The software version of Go1, which was released in 2021, has become seriously outdated in recent years, including the development time of the robot by *Unitree Robotics*, which can only be estimated. The software part of the robot is examined in more detail in Chapter 3.2 in conjunction with the installed hardware components, which is why only two problems are presented here as examples.

1. **Operating systems**
   Only Debian-based software is installed on the Go1's internal computers, which is now outdated and hardly maintained. Parts of the installed libraries have therefore been taken out of maintenance or have not been updated for some time. This is particularly relevant when considering the security of the robot and can lead to unexpected problems. Due to the

complex configurations and the large number of subsequently installed libraries.

2. **Robotics software**
   Similar to the problem of operating systems, the software for controlling the robot - ROS - is installed in the first version. This has long since been replaced by the successor ROS 2 and is hardly compatible with modern software and third libraries and extensions. This severely restricts the use of the robot and even resulted in an adapter library being developed by the manufacturer itself just for this problem[26].

Similar problems can be found in many of the robot's components, making it difficult to work with the device. This work will only deal remotely with updating the inventory. Instead, the extensions - where relevant - will describe how to deal with the outdated software in Chapter 5.

Insufficient quality of existing functions

Parallel to the poor quality of the documentation is the inadequate quality of the existing functions. This refers to all functions that were available and functional when the Go1 was delivered from the factory. These include functions such as movement, camera image transmission, sensor technology and autonomy in processing and reacting to collected data. Individually, all functions fulfill the minimum requirements, but when some functions are combined, the quality of the results of each individual function drops sharply. The reasons for this are often the unbalanced allocation of resources for processes running in parallel, which leads to high utilization of the computing units as well as the fundamentally high utilization of all components as a whole. Especially in the area of combinations of locomotion and computing power for evaluating video data and sensor technology, so-called *throttling* often occurs. [27] of the computing units due to overheating of the components caused by the thermal emissions of the mechanical components. However, the poor quality of the Go1's built-in parts also limits the implementation of the functions on the software side, some of which are quite good enough.

Parts of this work deal with the efficient outsourcing of some functions in order to make sensible use of the robot's resources and to optimize them in possible sub-areas.

[26] See chapter 2.2.3

[27] Throttling refers to the skipping of clocks or the reduction of the clock frequency of processors or graphics units in order to reduce the temperature and thus protect the unit from damage.

protect. The installed parts of the robot and the software used - where relevant for the extensions and functions of the Go1 shown - are evaluated for their efficiency and suitability for use.

Energetic and mechanical complexity

The use of the robot's biologically inspired four legs with high flexibility in the rotation and range of movement of each joint offers a great advantage over simpler solutions in robotics, such as wheels or chains, in terms of the versatility of locomotion and the ability to react to difficult terrain. The precise control of the motors in mutual interaction, on the other hand, has an enormous disadvantage compared to other locomotion options due to its complexity. Not only is the motion sequence much more complex to implement, but the large number of motors is also much more energy inefficient than simpler motion sequences, such as rotation in an axis on wheels. Changes to standard locomotion sequences are therefore more difficult to implement. The endurance of the built-in battery system also suffers from the high energy consumption of the robot's motors and computing units. Enduring or fully autonomous processes are therefore only possible to a limited extent, also due to the need to change the battery manually once it is fully discharged. Due to the lack or at least inadequate protection of the robot when the battery used is low, this work deals with the protection and resilience of the battery system in particular with regard to this challenge.[28]. In terms of the sensible integration of the Go1 into a university ecosystem, the above-mentioned challenges are critically examined and, if useful in the course of the work, also evaluated. In the following chapter, the core of the work, the Go1, is precisely analyzed and described so that knowledge of the robot is sufficient for the further course of the work in order to understand the descriptions of the extensions.

[28]See chapter 5

3 Robot architecture and system components

The following chapter describes the physical structure of the Go1 and its factory configuration. In order to better understand the complexity of the system, the structure is explained in several steps. First, the external structure is shown, then the structure of the internal system elements and the sensors, and finally the simplified internal network communication of the computing units is shown.

3.1 Structure

The architecture of the Go1 is described in detail below. For this purpose, some perspectives of the robot are shown in order to explain and illustrate the component groups classified according to their intended use.

3.1.1 Overview

The names of the externally recognizable components can be derived from the quadrupedal locomotion and the close reference to biology. Figure 4 shows an overview of the external features.

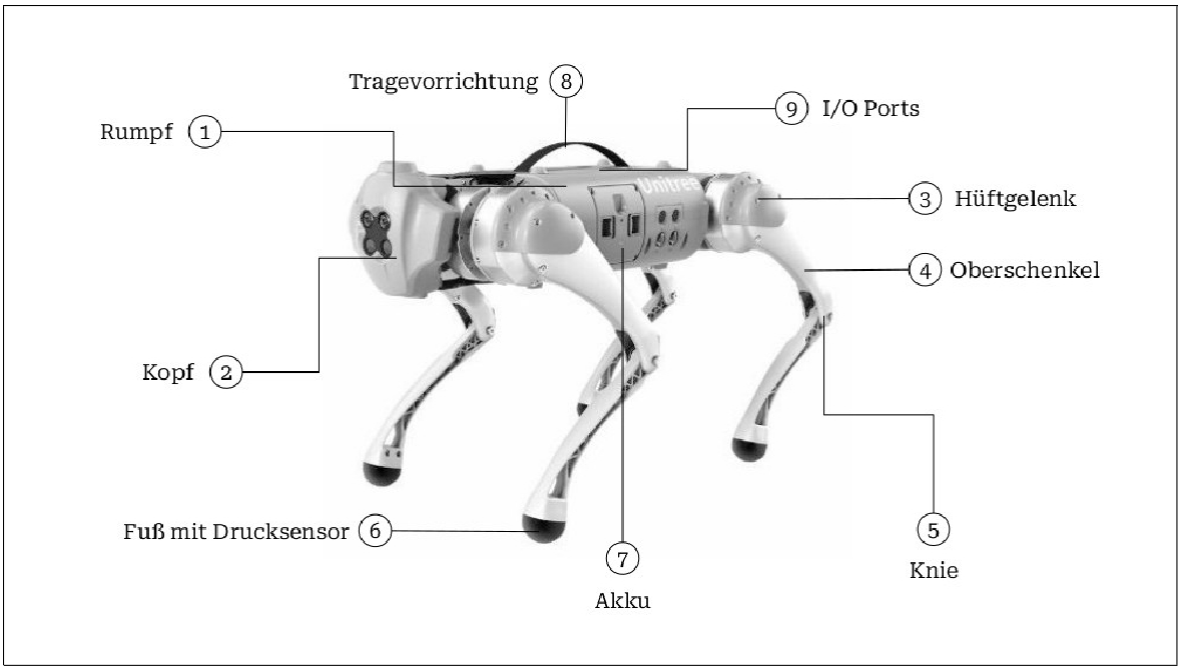

Figure 4: Overview of the Go1

The basis of the robot is the body - labeled 1 in Figure 4.
This contains most of the Go1's components, including the following:

- Internal hardware components
- Intelligent battery
- Parts of the sensors and cameras
- Hip joints and leg motors

A more detailed description of the individual parts can be found in the following sub-chapters. On the

The head (2) of the robot is installed at the front of the body. This contains, for example *an NVIDIA Jetson Nano*, a stereo camera and stereo ultrasonic sensors and other components such as loudspeakers. The robot's legs are built into the four outer corners of the body. Inside the body are the motors for controlling the hip joints (3) are integrated. Outside the hip joints on the upper side of the four thighs (4) are four additional motors for vertical control of the legs. Parallel to these motors, identical motors are integrated in the outer part of the upper thigh to control the knees (5), which bend and extend the joints using rigid axles and cable pulls can. Feet (6) with integrated pressure sensors are installed at the ends of each leg.

In addition to the outwardly conspicuous features, there is an intelligent rechargeable battery (7) is installed. On the upper side of the body below the carrying device are (8) Interfaces (9) for physical connection to the integrated hardware components installed.

3.1.2 Mechanical components

The proportion of mechanical elements in the Go1 shown in Figure 5 is limited. Only four moving parts are attached to the body itself - the four servomotors of the hip joints (1). Attached to the hip joints are the only other moving parts of the robot, which are two additional motors per leg for tilting the legs. (2) and the knee joints (3).

To extend the knee joint, a cable attached to the outer motor (4) is detached, while a used at the rigid connection (5) in the front of the knee joint is used to bend the lower leg.

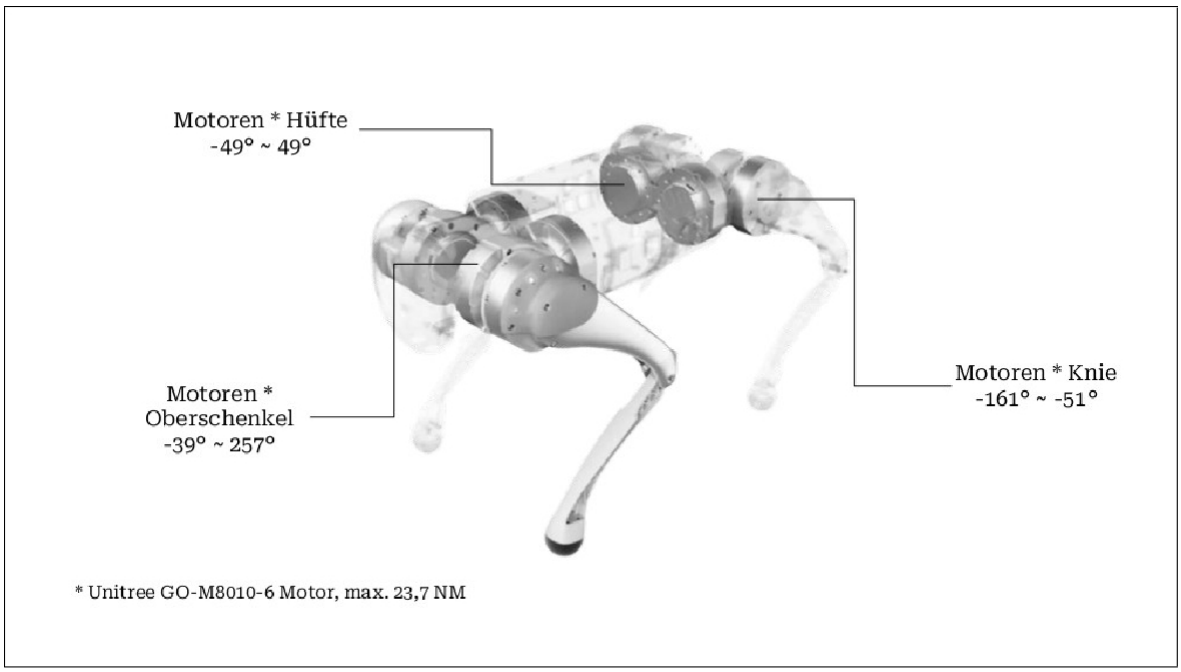

Figure 5: Mechanical components of the Go1

Features of the servomotors

The servomotors on the hip joint - Figure 5, component ①, the motors on the leg in- ② and the motors on the outside of the legs ③ are of the same model - *Unitree Robotics GO-M8010-6 motor*[29]. The servomotors have a maximum torque of 23.7NM (Newton/meter) and can be divided into 3 different configurations:

1. **Hip motors**
   Movement radius from -49° to 49°
2. **Thigh motors**
   Movement radius from -39° to 257°
3. **Knee motors**
   Movement radius from -161° to -51°

The motors are also equipped with sensors that can detect the current status of the component and send it to the MCU (Main Control Unit). This functionality is described in more detail in section 3.1.3.

[29]Unitree Robotics. *GO-M8010-6 Motor - Unitree Robotics*. URL: `https://shop.unitree.com/produc ts/go1-motor` (visited 07. 07. 2023).

3.1.3 Sensors

For intelligent use of the Go1, a number of sensors or intelligent hardware are installed at many points in the robot. In addition to the dedicated sensors for recognizing the environment, simpler sensors are also integrated in some components such as the mechanical components of the walking apparatus. Figure 6 shows the sensors and intelligent hardware components installed for this purpose.

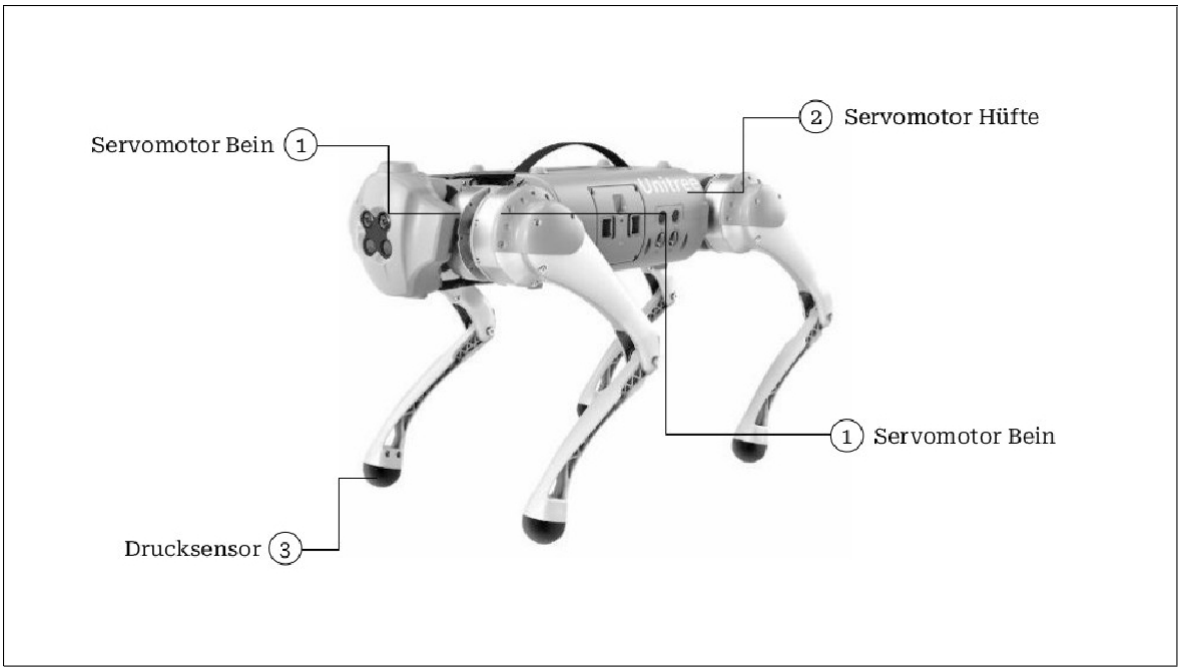

Figure 6: Sensors of the running apparatus

Similar to the mechanical properties of the twelve *GO-M8010-6* servomotors

- two per leg of the robot (1) and one motor each in the body of the robot (2) - are the

sensor functions of these are identical. Figure 12 in chapter 3.3.1 shows that the twelve motors of the Go1 are connected to the MCU via an *RS-485* interface. This evaluates the information and controls the individual motors via the same interface. For rotational movements, the motors measure the force required for execution and the difference to the expected force. In this way, the motors alone can provide a sufficient amount of data to control the movement apparatus.

As a supplement to the intelligent measurement of the motors, the ends of all four legs are equipped with

(3) pressure sensors are installed, which measure the force generated on the underside of the legs and

to the MCU. The combination of the rotation data and forces as well as the measured forces at the ends of the legs enable the Go1 to control the motors to compensate for the movements and thus enable the most efficient locomotion possible.

In addition to the integrated sensors for controlling the walking apparatus, cameras, ultrasonic sensors and audio technology have been installed in the Go1. Figure 7 shows the distribution of the cameras and ultrasonic sensors.

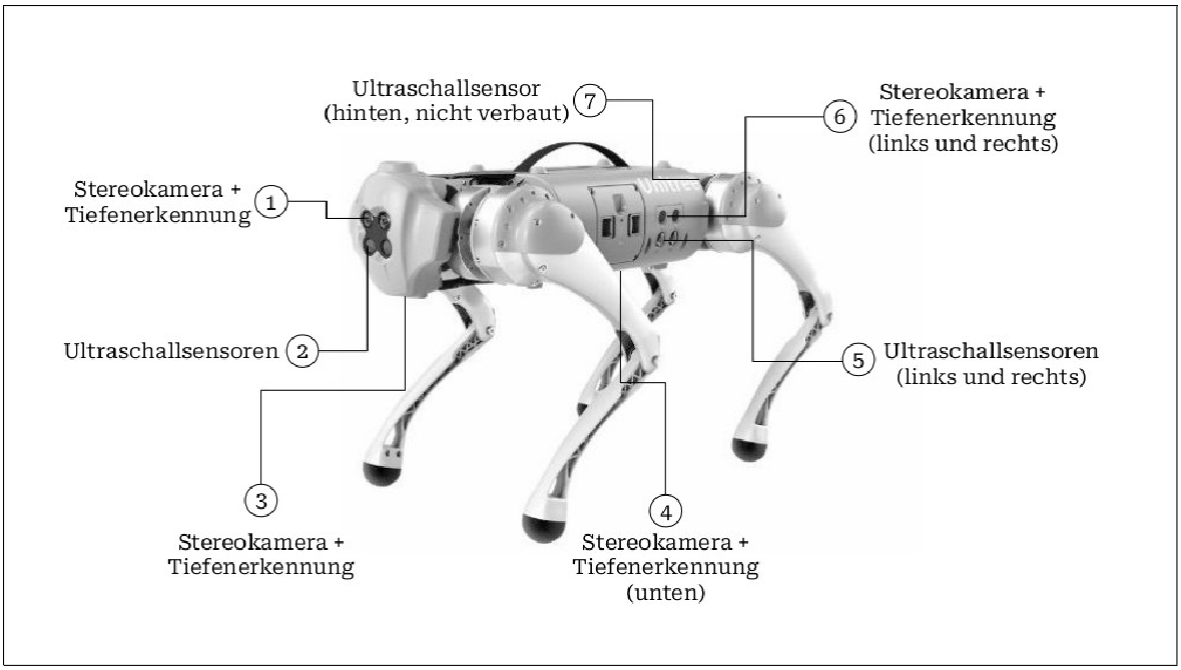

Figure 7: Illustration of the installed camera and sensors

A total of five stereo camera systems are installed in the Go1, which are also capable of depth detection thanks to their dual design. The cameras are distributed as follows:

- 1 Head unit facing forwards Head
- 3 unit facing downwards Torso left and right
- 4 Torso pointing downwards
- 6

The 1 and 3 cameras on the head of the Go1 are controlled by the Jetson Nano in the head. controlled, the two outward-facing cameras 4 by another Nano and the Camera on the fuselage pointing downwards 6 from the last of the three installed Jetsons, the Xavier NX.[30] More information on the control system and the functions of the cameras and computing units can be found in chapters 3.2 and 4.2.6. Ultrasonic sensors are installed under three of the five camera systems.

These are located on the head towards the front 2 and on the body towards the sides 5. According to

The documentation of the ultrasonic sensors requires an additional ultrasonic module on the

software side.

[30] *Development and use of Go1 binocular fisheye camera*. Unitree Robotics.

on the fuselage directed to the rear (7), but this was never realized on the hardware side.

was[31]. The ultrasonic unit in the head of the robot is controlled by the Jetson Nano in the head of the robot while the two side sensors are connected to the Raspberry Pi in the body of the Go1.

3.1.4 Computing units and interfaces

The *Edu* version of the Go1 has been equipped with some integrated computing power. Various small board computers are installed inside the torso and the head, for example to utilize the data from the sensors described above, to manually manipulate the movement apparatus and to add further functions to the robot. Figure 8 shows the internal composition of the Go1.

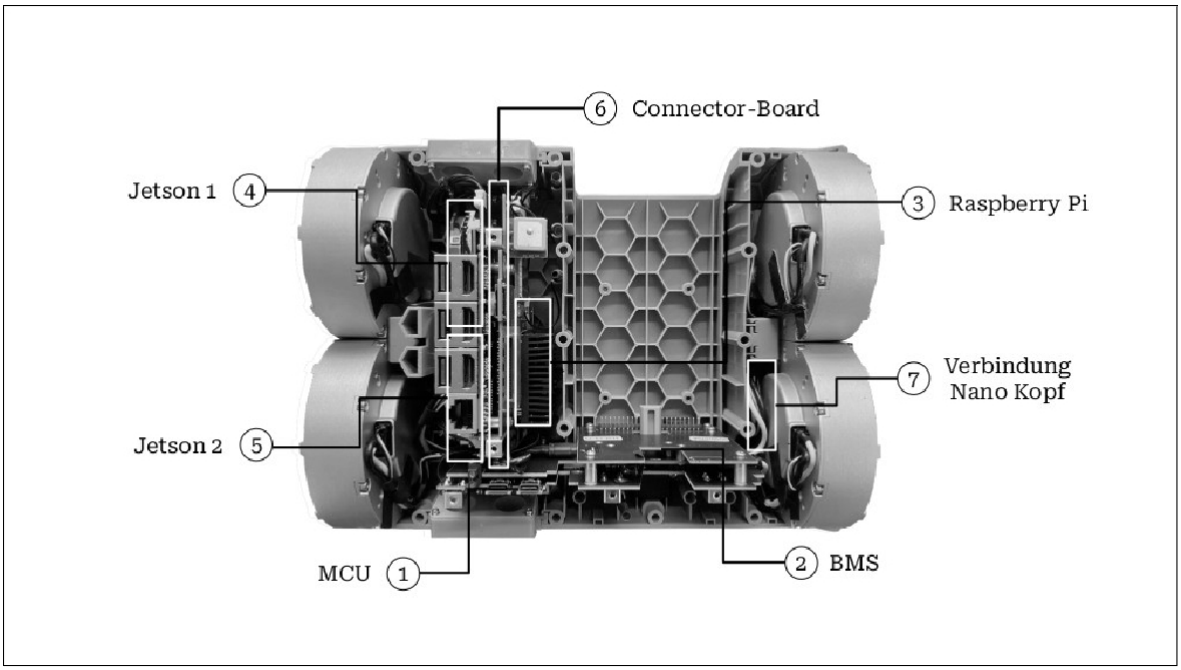

Figure 8: View of the internal components

The heart of the robot is the MCU (1), which is installed on the right-hand side of the fuselage. This controls the motors and accesses data from the BMS (battery management system).

system) (2) via the intelligent battery[32] off. In the center behind the built-in battery is the

The core interface of the Go1 with the users is the built-in *Raspberry Pi* (3). This is directly

via an expansion board (6) for interfaces and fixed connections with the two

NVIDIA Jetsons (4) +(5) installed in the fuselage. Along the area reserved for the battery

The wired connection (7) runs between the two components installed in the fuselage.

[31] *Development and use of Go1 ultrasonic module*. Unitree Robotics.

[32] Figure 1(0) (1)

components and those in the robot's head. Figure 9 shows the remaining components installed in the dog's head.

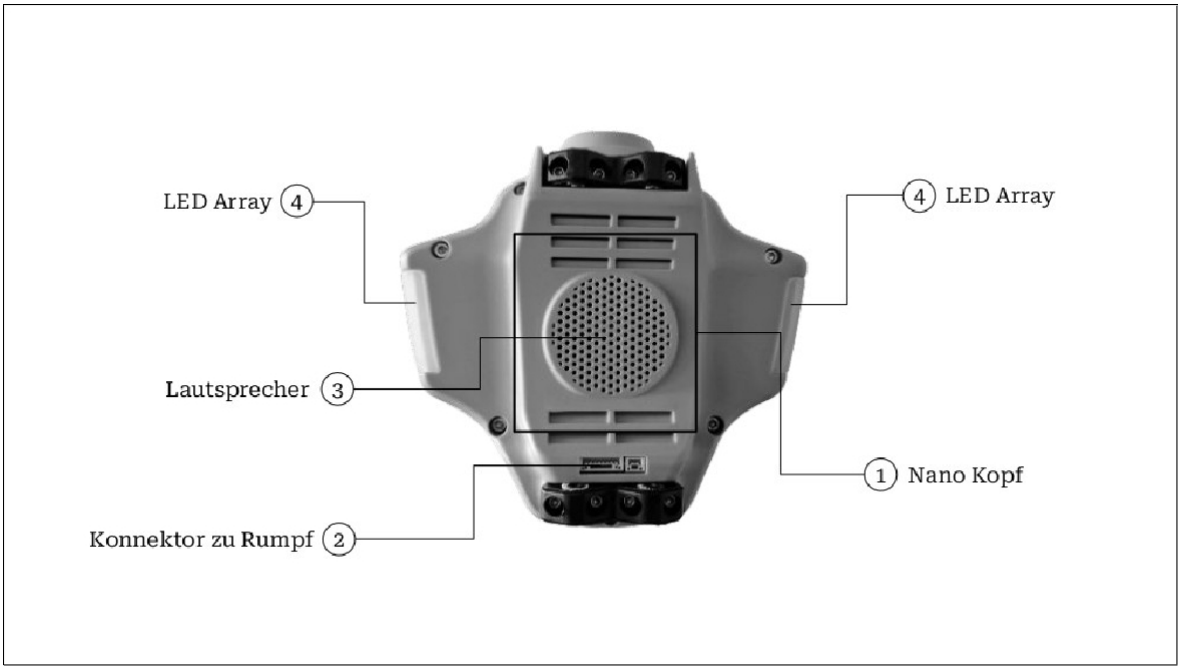

Figure 9: View of the head from behind

The central control unit for the components in the head of the robot is the built-in *NVIDIA Jetson Nano* 1, which is connected via wired connections 2 to the compo

nents is coupled. A loudspeaker 3 is installed directly on the Nano, which is free in the Edu *model*.

is usable[33]. Also connected directly to the Nano are two rows of LEDs (light-emitting diodes) on the left and right of the head, facing backwards and to the side. These are also freely programmable[34]. A more detailed description of the computing units and their functions follows in chapter 3.2.

A number of interfaces are installed on the top of the Go1 to facilitate work on the robot's internal components and computing units. Figure 10 shows the top side and the installed interfaces. As an alternative to the power supply via the built-in

Battery 1 on the left outer side of the robot is an XT-30 *connector* 2 on the right outer side of the robot.

installed on the top. This can either operate the Go1 instead of the battery or supply external extensions with power via the battery. For more details, see chapter 4.1. Directly above this connection are two USB (Universal Serial Bus) *type C* ports 3, which can be used as interfaces to the MCU. Extensions, such as GPS

[33]See chapter 4.2.4
[34]See chapter 4.2.5

(Global Positioning System) modules can be ④ via a serial interface. connected

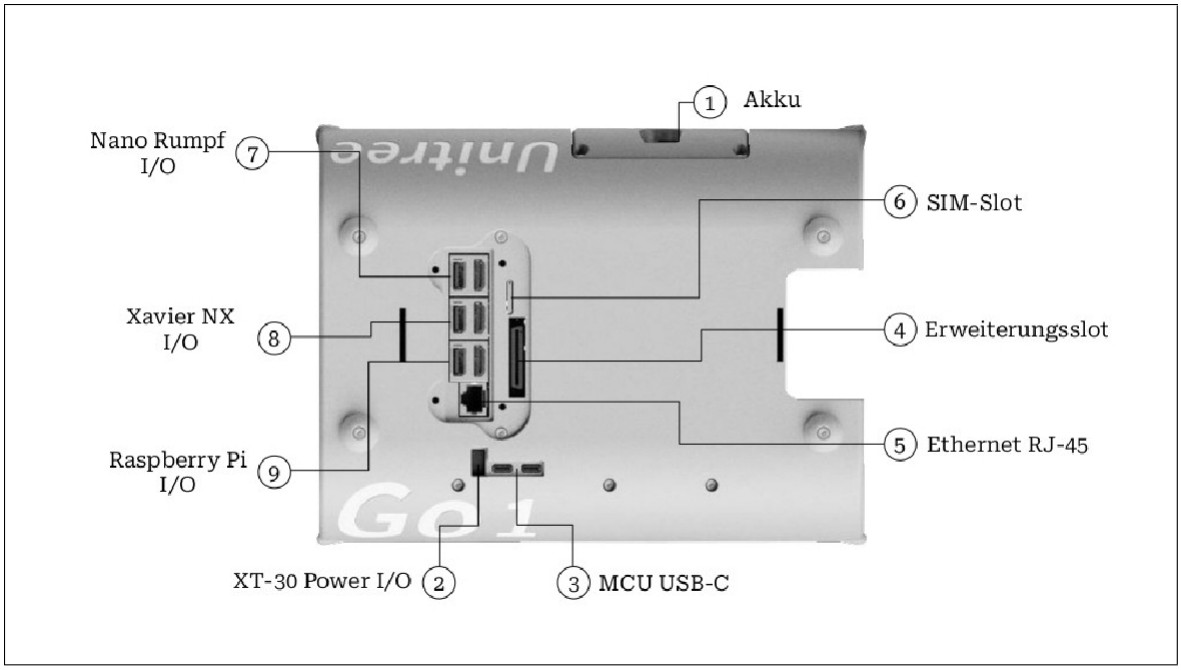

Figure 10: Bird's eye view with hardware

For the wired connection of an external computer to the internally installed switch and thus the computing units[35] an RJ-45 *Ethernet socket* ⑤ is installed. As an alternative to the wireless network can be used to connect to the Internet or a mobile network.
The built-in slot for *4G/ LTE (Long Term Evolution)* cards ⑥ can also be used. This is directly connected to the installed Raspberry Pi[36].

The remaining connections on the rear of the fuselage are three pairs, each with an HDMI (High Definition Multimedia Interface) and a USB-A port for direct connection to the small board computers by developers. The distribution is as follows:

- ⑦ NVIDIA Jetson Nano (hull) NVIDIA
- ⑧ Jetson Xavier NX Raspberry Pi
- ⑨

No direct connection is possible to the Nano in the head of the robot via the ports.

[35]See chapter 3.3
[36]See chapter 5.1

3.2 Hardware architecture

The following chapter describes the structure of the internal hardware system and the function of the individual components. For this purpose, a brief overview of the components and their interrelationships is first provided, which overlaps in part with the description in the previous chapter 3.1.4. With the overview created, the individual parts of the system are examined and documented in more detail. This should go well beyond the manufacturer's simple documentation. Finally, the internal communication of the components within themselves and with external components is described.

3.2.1 Overview

Figure 11 serves as an illustrative introduction to the overview. It shows the individual components and their connections to each other and to the extended systems such as the BMS or the sensors.

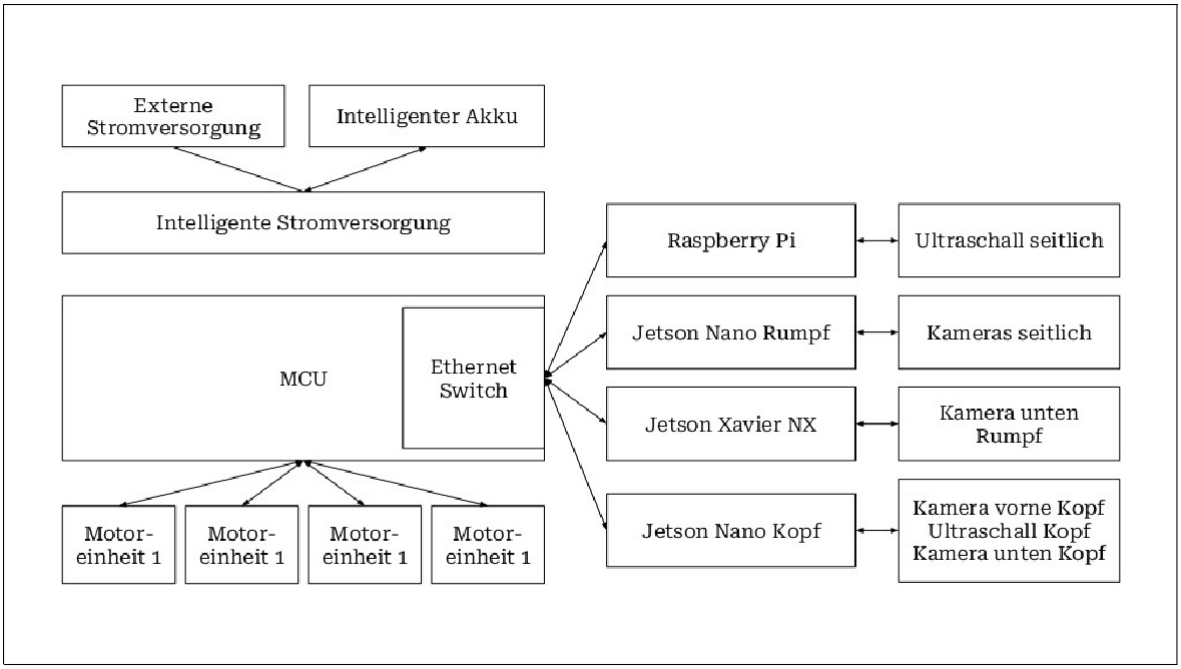

Figure 11: Overview of the Go1's internal architecture

The MCU and the BMS serve as the basis for both the internal hardware and the mechanical components. Both are not enabled for access by the manufacturer and can only be used indirectly. For example, the data of the motors and their control can currently only be read and manipulated via libraries. The BMS data is also read-only. A switch is located centrally to all components, which connects them via a network. All three are connected directly to it

NVIDIA Jetson units - two Nanos and one Xavier NX, the *Raspberry Pi*, the *MCU* and the *RJ-45 port* made available to the outside. The Raspberry Pi can be described as the central building block for all developers on the Go1. With the exception of dedicated evaluations or access to the camera modules, most processes are at least managed on the Pi. The NVIDIA units, on the other hand, process the camera modules assigned to them, with the exception of the nano in the robot's head, which also taps and makes available the sensor data from the forward-facing ultrasonic sensor. The NVIDIA Jetson Xavier NX is also the most powerful unit in terms of processor and graphics units and can therefore be used for ML (machine learning) and video analysis. The following overview shows the distribution of the accessible computing units to the modules to be managed.

Raspberry Pi	**Nano 1 (head)**	**Nano 2 (hull)**	**Xavier NX**
Wifi module Web hosting App connection Monitoring libraries Lateral ultrasound	LED control Audio output Ultrasonic frontal Video evaluation Head front/bottom	Video evaluation left + right	Video evaluation hull bottom ML processes

Based on the overview, the next chapter takes a closer look at the individual components of the Go1's internal hardware.

3.2.2 Core elements

The core elements of the Go1 are the built-in *Raspberry Pi*, the two built-in *NVIDIA Jetson Nanos* and the *NVIDIA Jetson Xavier NX*. In principle, the MCU can also be described as a core element, but it is not enabled for access by the developer and is therefore not considered further. For a more detailed inspection of the components, an external computer can be connected via *Ethernet* to the RJ-45 *port* described in section 3.1.4. This computer must then be assigned a static IP (Internet Protocol) address in the network 192.168.123.0/24. As the IP address must not yet be assigned, the address 192.168.123.51 was used for the analysis in this work.

Raspberry Pi

According to the manufacturer's documentation, a *Raspberry Pi* 4 is installed in the robot. The documentation only shows the IP address of the Pi - 192.168.123.161, but no information about its properties. To check the properties of the Pi, you can connect to the robot via Ethernet. According to the documentation, the standard user can connect to the robot via SSH (Secure Shell).

```
# user : pi , password : 123 , root - password
:123 ssh pi@ 192 . 168 . 123 . 161
```

The first step is to recognize the exact model of the Raspberry Pi:

```
pi@ raspberrypi :~ $ grep Model / proc / cpuinfo
Model            : Raspberry Pi Compute Module 4 Rev 1.0
```

You can check the installed variant of the Compute Model 4 using the following command:

```
pi@ raspberrypi :~ $ grep Mem Total / proc /
meminfo Mem Total :     1894664 kB
```

A quick check of the manufacturer's website shows us that a Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz is installed in the variation with 2 GB (gigabytes) of RAM. The boot partition and the initial hard disk memory are often implemented via an SD (Secure Digital) card on various small board computers. To check this on the Raspberry Pi, the assignment of the file system can be output. The temporary file systems are excluded here.

```
pi@ raspberrypi :~ $ df - HTx tmpfs - x devtmpfs
File system     Type   Size   Used  Avail Use % Mounted on
/ dev / root      ext4    32 G   18 G   13 G   59% /
/ dev / mmcblk 0  vfat   265 M   69 M  196 M   26% / boat
p 1
```

You can see that an SD card has actually been mounted as a boot partition under /dev/mmcblk0p1. You can also see the total size of the file system - 32 GB. The following command can be used to check the installed operating system and the Linux kernel:

```
pi@ raspberrypi :~ $ grep PRETTY_NAME / etc / os -
release PRETTY_NAME =" Debian GNU / Linux 10 ( buster
)" pi@ raspberrypi :~ $ uname - r
5.4.81 - rt45 - v8 +
```

The core data of the Pis can be summarized as shown in Table 1.

Model	Raspberry Pi Compute Module 4 Rev 1.0
SoC	Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM	1894664 kB (1.8 GB) Working memory
Memory	32 GB hard disk storage via an SD card
OS	Debian 10 (Buster)
Kernel	Linux Kernel 5.4.81-rt45-v8+

Table 1: Characteristics of the Raspberry Pi

If you take a look at the connected components in addition to those integrated in the Raspberry Pi, you can easily recognize the function of the Pi as an interface between the developer and the robot. Connected devices can largely be checked via the output of the devices connected via USB using the lsusb command.

```
pi@ raspberrypi :~ $ lsusb
Bus 001 Device 004: ID 0 bda : c812 Realtek Semiconductor Corp .
Bus 001 Device 003: ID 2 c7c : 0125 Quectel Wireless Solutions Co., Ltd .
   ↪  EC25 LTE modem
[...]
```

The *Realtek* USB WiFi adapter and the *Quectel* modem for inserting the 4G/ LTE SIM (Subscriber Identity Module) card are particularly worth mentioning in this issue. The exact function of the USB devices can often be checked by searching for the device identification to the left of the full name of the device on the manufacturer's website. Both devices allow the Pi to connect to the Internet or a local network in addition to the interfaces installed on the board.

A look at the official documentation for the ultrasonic sensors shows that the two sensors on the left and right of the robot's torso are connected via the serial port ttyAMA0.

```
pi@ raspberrypi :~ $ dmesg | grep tty AMA
0     1 . 251673 ] fe 201000 . serial : tty AMA 0 at MMIO 0 xfe 201000 (
[   ↪irq = 14 ,
        base_baud = 0) is a PL011 rev2
```

As the manufacturer's libraries for using the ultrasonic sensors are pre-compiled and not documented, no further information could be found about the connection type and use of the sensors without the library. This also applies to the sensors in the head of the Go1.

NVIDIA Jetson Nano head

If you follow the manufacturer's documentation, you can reach the computing unit in the head of the robot at the IP address 192.168.123.13. The computer can be connected via the user unitree and the password 123.

```
# user : unitree , password : 123 , root disabled
ssh unitree@ 192 . 168. 123. < nano - ip ( 13 |
14 | 15 ) >
```

According to the manufacturer, the Ubuntu operating system is installed on all three NVIDIA chips, which is based on Debian but has some useful functions beyond the Debian base. This includes the lshw command, which can be used to output a summary of the hardware used on the system.

```
unitree@ unitree - desktop :~$ sudo lshw -
short
[...] Class        Description
[...] ========================
[...] system       NVIDIA Jetson Nano Developer Kit
[...] memory       3962 MiB System memory
[...] bridge       NVIDIA Corporation
[...] multimedia   USB2    Camera RGB
[...] multimedia   .0      Camera RGB
[...] generic      USB2
                   UP2102 N USB to UART Bridge Controller
[...] multimedia   @
                   USB Audio Device
```

The shortened version shows that an *NVIDIA Jetson Nano* with 4 GB of RAM is installed in the head of the Go1. In addition, four external devices are connected via USB, a speaker in the back of the head, a bridge controller to control the two LED strips and two cameras. The front-facing camera is mounted under /dev/video1, while the downward-facing camera in the head is mounted under /dev/video0. The documentation of the ultrasonic sensors also shows that the ultrasonic sensor on the head of the robot is located on serial port ttyTHS1.

```
unitree@ unitree - desktop :~$ dmesg | grep tty
THS 1
[    1 . 099918 ] 70006040 . serial : tty THS 1 at MMIO 0 x 70006040 ( irq
↪ = 64 ,
        base_baud = 0) is a TEGRA_UART
```

These can be read under the mount points of the cameras and used as a source. In contrast to the same command on the Raspberry Pi, a look at the file systems only shows the root partition, but a further inspection then also shows the boot partition on the SD card.

```
unitree@ unitree - desktop   df - Hx tmpfs - x devtmpfs
:~$
Filesystem       Size   Used  Avail Use % Mount
/ dev / mmcblk 0 p  115    12    2.5    83% /
G                           G     G
unitree@ unitree - desktop   sudo lsblk
:~$
NAME           FSTYPE    SIZE  MOUNTPOINT
mmcblk 0                14.7
-> mmcblk  0 p   ext4     G      /
1   mmcblk   0             14 G
boot  0 mmcblk               4 M
0 boot 1                     4 M
```

The processor variant in /proc/cpuinfo shows ARMv8 Processor rev 1 (v8l). A look at the manufacturer's website shows that a quad-core ARM Cortex-A57 MPCore processor is installed with the processor designation[37]. As the NVIDIA Jetson series is optimized for use in robotics and ML applications, powerful graphics units are installed alongside the CPU (Central Processing Unit). According to the manufacturer, an NVIDIA Maxwell GPU with 128 cores is installed. The exact version of the operating system can be determined in the same way as in the Raspberry Pi:

```
unitree@ unitree - desktop :~$ grep PRETTY_NAME / etc / os - release
PRETTY_NAME =" Ubuntu 18 . 04 . 5 LTS "
unitree@ unitree - desktop :~$ uname - r
4.9.201 - tegra
```

The core data of the NVIDIA Jetson Nano in the head of the robot can be summarized as shown in Table 2:

Model	NVIDIA Jetson Nano Developer Kit
GPU	NVIDIA Maxwell GPU with 128 cores
Processor	Quad-core ARM Cortex-A57 MPCore
RAM	3962 MiB (4 GB) main memory
Memory	16 GB hard disk storage via an SD card
OS	Ubuntu 18.04.5 LTS
Kernel	4.9.201-tegra

Table 2: Characteristics of the NVIDIA Jetson Nano in the head of the robot

[37]NVIDIA Corporation. *Developer Kit and Modules for Embedded Systems*. 2023. URL: urhttps://www.nvidia.com/en/autonomous-machines/embedded-systems/.

NVIDIA Jetson Nano hull

The nano in the body of the robot is identical to the nano in the head of the robot except for the connected devices and its IP address 192.168.123.14. The output for the operating system in /etc/os-release, the kernel version from uname -r and the information on the SD card from df -Hx tmpfs -xdevtmpfs are the same for both devices. The differences in the connected hardware can be seen in the following output:

```
unitree@ unitree - desktop :~$ sudo lshw -
short
[...] Class        Description
[...] ========================
[...] system       NVIDIA Jetson Nano Developer Kit
[...] memory       3964 MiB System memory
[...] bridge       NVIDIA  Corporation
[...] multimedia   USB2    Camera RGB
[...] multimedia   .0      Camera RGB
                   USB2
                   .0
```

Similar to the Nano in the head of the Go1, two cameras are installed, one camera on the left in the fuselage viewed from behind under the mounting point /dev/video1 and one camera on the right in the fuselage under /dev/video0. Table 2 can be used as an overview of the core data of the Nano in the body of the robot.

NVIDIA Jetson Xavier NX

The central processing unit with the highest performance is the *NVIDIA Jetson Xavier NX* installed in the hull with the internal IP address 192.168.123.15. The Ubuntu operating system is installed on the NX, as on both Nanos. The kernel version is also identical to both Nanos.

```
unitree@ nx :~$ grep PRETTY_NAME / etc / os -
release PRETTY_NAME =" Ubuntu 18 . 04 . 5 LTS "
unitree@ nx :~$ uname - r
4.9.201 - tegra
```

Essentially, the NX differs from the two Nanos in terms of its platform, processor unit and built-in RAM. The following overview shows the hardware equipment:

```
unitree@ nx :~$ sudo lshw -
short
[ sudo ] password for unitree :
[...] Class        Description
[...] ========================
[...] system       NVIDIA Jetson Xavier NX Developer Kit
```

```
[...] memory       7773 MiB System memory
[...] bridge       NVIDIA Corporation
[...] multimedia   USB2 .0 Camera RGB
```

According to the manufacturer, the NX is equipped with a 6-core NVIDIA Carmel ARM v8.2 64-bit CPU[38]. It can be stated that the NX is significantly more capable than the two Nanos in the head and body. In addition to the better processor, the NX has 8 GB of RAM. The connected camera is the downward-facing camera in the fuselage. A look at the hard disk capacities of the NX also makes it clear why Unitree advertises this unit as the core of the Go1's computing power.

```
unitree@ nx :~$ df -   tmpfs  - x
Hx                             devtmpfs
File system             Used Avail  Use Mounted on
/ dev / nvme 0 n   Size    22    90     % /
1 p 1              118      G    G    20% / media / unitree / cd8bfc0a -0
/ dev / mmcblk 0   G    116833 84 08   14    1% f39 -4
p 1 f 45           15 G          G
```

The significantly higher storage capacity by connecting an SSD (Solid State Drive) in addition to the SD card used for the boot process enables the NX to analyze larger amounts of data than on the two Nanos with only 16 GB storage capacity on their SD cards. As on the Nanos, the graphics unit on the NX is much more important than the computing unit. According to the manufacturer, an NVIDIA Volta GPU with 384 cores and 48 tensor cores is installed. Here too, the NX is significantly better equipped than the two Nanos. Table 3 briefly summarizes the features of the NVIDIA Jetson Xavier NX.

Model	NVIDIA Jetson Xavier NX
GPU	NVIDIA Volta GPU with 384 cores and 48 tensor cores
Processor	6-core NVIDIA Carmel ARM v8.2 64-bit CPU
RAM	7773 MiB (8 GB) RAM
Memory	16 GB hard disk storage via an SD card 120 GB SSD memory
OS	Ubuntu 18.04.5 LTS
Kernel	4.9.201-tegra

Table 3: Characteristics of the NVIDIA Jetson Xavier NX

It should be noted that the manufacturer's documentation for the Xavier NX lists an option to start the NX in different performance modes. Depending on the input power and cooling capacity, the CPU and GPU (Graphics Processing Unit) have the option of increasing their clock frequency. However, this function has not been tested.

[38]Corporation, see note 37.

3.3 Network

Due to the many components, functions and possible extensions of the Go1, robust internal communication is necessary. The robot's internal communication structure is largely *based* on network standards such as *Ethernet* and *Wi-Fi*, but also relies on additional standards such as *Bluetooth* and *WWAN (Wireless Wide Area Network)*, particularly for connectivity with external components.

The following chapter explains the existing communication of the internal and external components of Go1 and analyzes their strengths and weaknesses. In addition, the methodology used to analyze the network and possible problems identified and rectified are also described.

3.3.1 Overview

Figure 12 serves as a reference for the following explanations. The central communication unit is the built-in Ethernet switch and the internally installed *Raspberry Pi*. As can be seen in Figure 12, all five computing units - the Raspberry Pi, the MCU, the two NVIDIA Jetson Nanos and the Jetson Xavier NX - are connected to the switch via *Ethernet.* The externally accessible Ethernet port on the back of the robot is also connected to the switch.

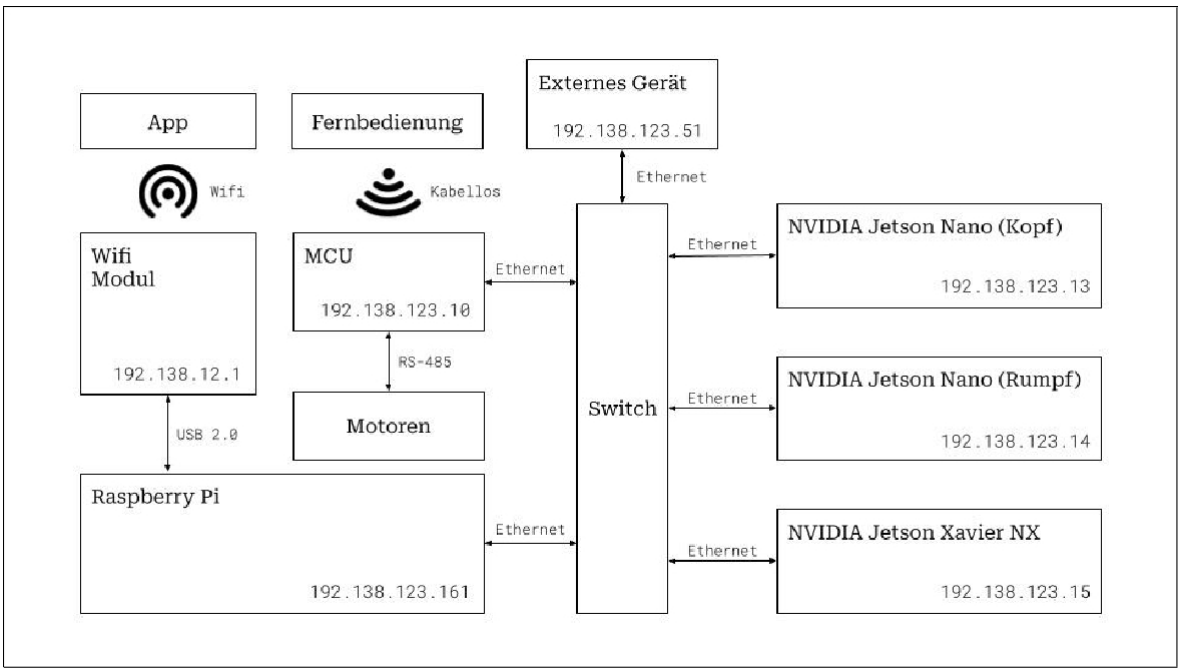

Figure 12: Overview of network configuration

All switched components of the network are registered in the 192.168.123.0/24 network. The distribution of IP addresses is preconfigured as follows:

- **MCU:** 192.168.123.10
- **Raspberry Pi:** 192.168.123.161
- **NVIDIA Jetson Nanos:**

  1st header: 192.168.123.13

  2nd pages: 192.168.123.14
- **NVIDIA Jetson Xavier NX:** 192.168.123.15

The end device that can be connected to the external Ethernet port on the top of the robot must be assigned a static IP address in the range 192.168.123.0/24 that is not already being used by one of the above-mentioned devices.

In addition to its physical connection to the switch and the 192.168.123.161 IP address, the Raspberry Pi also has a WLAN (Wireless Local Area Network) module installed, with which it publishes the network 192.168.12.0/24. This network is required ex works to connect the app to the system. Furthermore, this network can be used to establish a wireless connection with the robot's overall system.

3.3.2 Wired connection

As already mentioned in Chapter 3.1.4, the Go1 has an RJ-45 interface on the top of the body for physical connection to the computing units. Once a wired connection has been established, a static IP address must be configured on the connected unit that is not already occupied by one of the installed components. In this work, the IP address 192.168.123.51 is always specified. Only the addresses already listed in the overview are assigned. If the static address is configured, all systems on which SSH is activated can now be reached. It should only be noted that the network is only a switched network, not a routed network. The internal communication must therefore be configured via routes on the individual computing units. An example of this is shown in the following paragraph on the subject of *wireless connections*.

3.3.3 Wireless connection

Unlike a wired connection, no static IP address configuration is required for a wireless connection to the Go1 network. It only needs to register with the

WLAN access point, which the Raspberry Pi sets up after system start. The exact configuration of this is documented in chapter 4.2.3.

After starting the robot, an access point is set up with the serial number of the Go1 used as the SSID (Service Set Identifier). The default password for the connection is 00000000. The IP configuration of the host is then controlled via DHCP (Dynamic Host Configuration Protocol). The access point has the address 192.168.12.1, which is why the host is also assigned an address in the network 192.168.12.0/24.

To better understand the internal communication of the Pi with the rest of the network, the forwarding rules and the IP routes of the system can be output.

```
pi@ raspberrypi :~ $ cat / proc / sys / net / ipv4 / ip_forward
1 # IP - Forwarding activated
pi@ raspberrypi :~ $ cat / etc / sysctl . conf | grep ip_forward
net . ipv4 . ip_forward =1 # Permanent activation of forwarding
pi@ raspberrypi :~ $ sudo iptables - L
[...]
target     prot  opt  source               destination
ACCEPT     all   --   anywhere             anywhere
ACCEPT     all   --   anywhere             anywhere
[...]
```

It can be seen that the system allows all forwarding attempts. Furthermore, the routes on the Pi must be checked for communication between the 192.168.12.0/24 network and the hosts in the 192.168.123.0/24 network.

```
pi@ raspberrypi :~ $ ip route | grep  default
[...]
default via 192 . 168 . 123 . 1 dev eth0 src 192 . 168 . 123 . 161 metric 202
default via 192 . 168 . 12 . 1 dev wlan1 [...] src 192 . 168 . 12 . 1 metric 305
```

It can be seen that the standard route of all requests to the gateway in network ..12.0/24 is routed via the interface wlan1, via which they are then sent to the connected host. Conversely, all messages from the host to the network ..123.0/24 are routed via the Ethernet interface eth0, which acts as a gateway and forwards the packets directly to the recipients, as these can be reached via the switch without further gateways.

However, only one standard route is defined on the connected NVIDIA Jetsons.

```
unitree@ unitree - desktop :~$ ip route | grep default
```

```
default via 192 . 168 . 123 . 161 dev eth0
default via 192 . 168 . 123 . 1 dev eth0
 onlink
```

Here, the Ethernet interface eth0 acts as a gateway and forwards all unknown addresses, for example those of the network ..12.0/24, via this interface to the address 192.168.123.161, which is the address of the Ethernet interface of the Raspberry Pi.

3.3.4 Further network functions

In addition to the eth0 and wlan1 interfaces, other network cards are installed inside the Raspberry Pi. These can be used, for example, to connect the Pi to a WLAN access point or to dial it into a mobile network. These functions are explained in more detail in chapters 5.1 and 5.1.

4 Analysis of the robot

This and the following chapters only deal with the infrastructure around the robot and the hardware and functions that are already built into the robot or can be added. The functions relating to ML, advanced robotics and lidar are not covered. This chapter deals with the approach to analyzing the robot. It shows how the existing functions can be tested and used, how to recognize which functions are already activated and which parts of the software or hardware are not activated. Finally, the existing functions are shown and explained in the scope in which they were supplied ex works. Some of the functions will also be expanded or modified later in the work. The documentation for this can be found in chapter 5. Affected functions are explicitly highlighted here in the chapter.

Chapter 3 is referenced as the basis for the information assumed in this chapter. This should be read before continuing if you do not yet have extensive basic knowledge about the structure of the Go1. It should also be noted that the robot analyzed in the following chapters 4 and 5 is the *Go1 Edu* version, which comes with significantly more extensive features and software options than the less equipped versions of the Go1.

4.1 Commissioning

This chapter describes the initial commissioning of the robot. The contents and extensions included in the scope of delivery for a Go1 Edu are also listed. Operation using the battery or a power supply unit is then described.

4.1.1 Scope of delivery

Figure 13 shows the polystyrene transport box included in the scope of delivery and its contents, which is also the entire scope of delivery of the Go1 in the *Edu* version.
The following elements are included in the scope of delivery:

1. Go1 including battery
2. Compact remote control with *Follow* Me *function*
3. Power supply unit for the battery charger

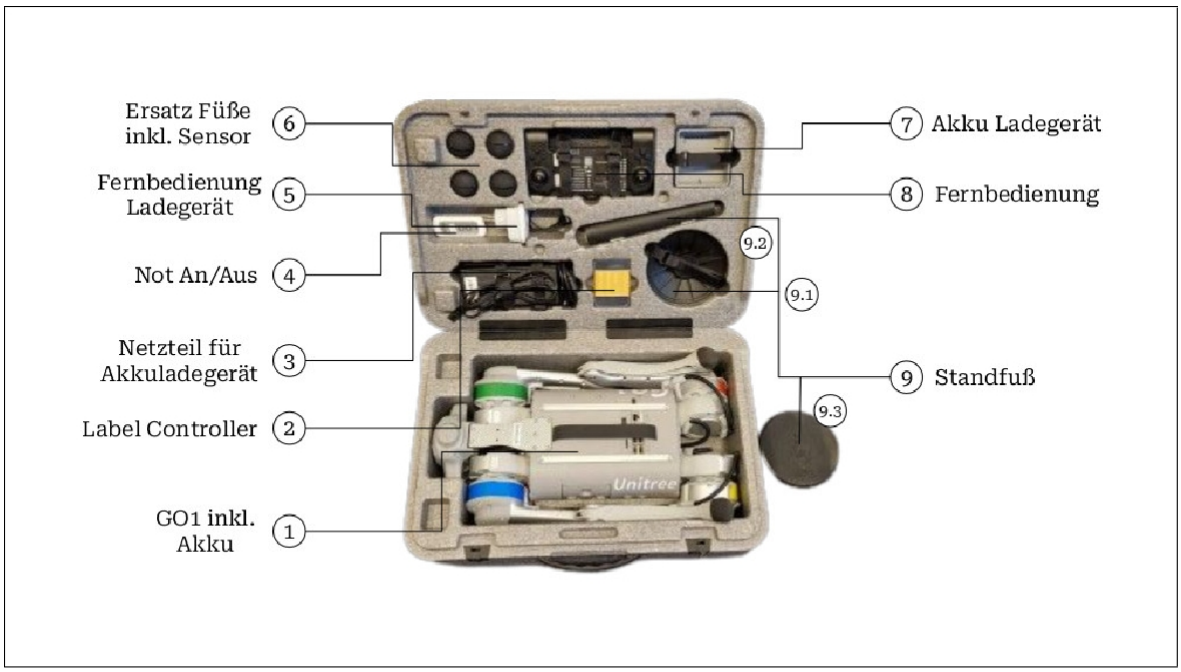

Figure 13: View of the transport box and scope of delivery

4. Remote control for on/off function
5. Charger for the remote controls including USB-A to USB-C cable
6. Four replacement feet including integrated pressure sensors
7. Battery charger with USB-C connection for battery inspection
8. Remote control
9. Stand consisting of three parts for storing the Go1
    a) Base
    b) Extension
    c) Platform including notches for balancing the Go1 on top

If no other accessories are configured, rubber ends are fitted to the mounting points on the top of the robot to protect the body during rotation. Depending on the accessories ordered - such as a professional lidar sensor or a robot arm on the Go1 - rails are mounted on the top of the body. In this case, the rubber ends and a protective cover for the developer ports, which is fitted as standard, are supplied in the transport box.

4.1.2 Mobile commissioning

Mobile commissioning here refers to the commissioning of the robot with a power supply from the battery. The following parts are required for commissioning in this way:

Prerequisites
Go1, battery, battery charger including mains adapter, remote control including charger

To prepare, both the battery and the remote control must be charged. Once they are sufficiently charged, the battery can be plugged in and the Go1 can be brought into the starting position. To do this, the four legs must be rotated so that both feet and knees touch the ground. Figure 14 shows the robot's starting position.

Figure 14: Starting position of the robot

By simply pressing the button above the battery charge indicator, the charge status is indicated by the four LEDs. Pressing and holding the button again immediately starts the robot. This is signaled by a brief serial flashing of all four LEDs. The fans can also be heard clearly. The same procedure - pressing followed by a second press and hold of the button next to the charging indicator on the underside of the remote control also switches it on. A single acoustic signal sounds when it is switched on. The remote control automatically connects to the robot. In the factory setting, the robot stands up after about 70 - 80 seconds after being switched on. The robot can now be controlled using the remote control.

To switch off the Go1, it should be placed in a horizontal position (*prone state*) and then in *the* damping *state.* This is done using the key combinations

L2+A and L2+B are reached. The battery can then be switched off as when switching on by pressing and holding again. The LEDs signal successful switch-off and the fans switch off. The remote control can also be switched off using this procedure. Here, three short acoustic signals indicate successful switch-off.

4.1.3 Stationary commissioning

In contrast to mobile commissioning, this refers to commissioning the robot using a continuous power supply from a mains adapter. Some preparations must be made for this, as the scope of delivery does not include a power supply unit for operating the Go1.

Prerequisites

Go1, power supply unit for battery charger

Additional: M 5.5/2.1 mm hollow plug to screw terminal, XT30-M plug with sufficient (> 10 cm) cabling. Alternatively: XT30-M to hollow plug M 5.5/2.1 mm adapter.

If you compare the output voltage and maximum current of the installed battery with the same values of the power supply unit for the battery charger, you will see that the charger has the right output voltage and power to charge the Go1 over the range described in chapter
3.1.4 Figure 10 described *XT-30* port. However, as the power supply unit only offers a hollow plug for connection, an adapter to XT-30 must be produced by the user. This requires the components described in the resources. Although the screw connection is optional and can be replaced by a soldered joint or a prefabricated part, it makes it easier to procure and assemble the adapter. The only thing to note here is the correct sheathing of the cables for the screw terminal and the orientation of the cables in the terminal. Figure 15 shows an example of the described implementation and the correct orientation of the XT-30 connector.

For safety reasons, the so-called *sports mode* of the Go1 should be deactivated, as otherwise it will stand up after starting the robot, which could loosen the wiring. In addition, the maximum output power of the power supply unit is not high enough to operate the robot continuously, including the motors. Operation is therefore called stationary. To deactivate the sport mode, you must first connect to the Raspberry Pi. The folder Unitree/autostart/triggerSport with the file triggerSport.sh is located in the home folder of the user *pi*. This simply needs to be renamed, for example to triggerSport.disabled.sh.

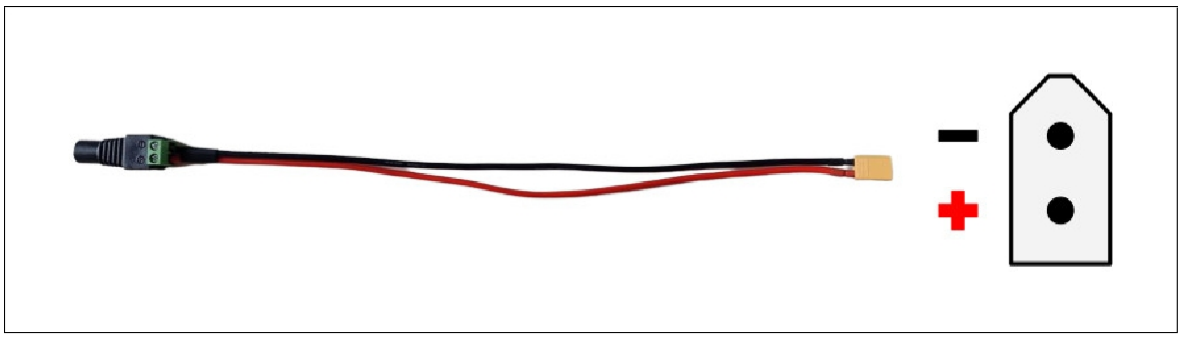

Figure 15: XT-30 to barrel connector cabling and XT-30 orientation

```
pi@ raspberrypi :~ $ cd Unitree / autostart / trigger
Sport /
pi@ raspberrypi :~/ Unitree / autostart / trigger
Sport $ ls build                    log   trigger Sport
pi@ raspberrypi :~/ Unitree / autostart / trigger
sport                                       $ mv trigger Sport .
sh version . txt
   ’→  trigger Sport . disabled . sh pi@
                                            $ ls
raspberrypi :~/ Unitree / autostart / trigger
build   log   trigger Sport . disabled . sh version .
Sport
txt
```

When the robot is restarted, it will not automatically switch to sport mode and the motors will not move. Further information on the Go1's *autostart function* can be found in chapter 4.2.1. Finally, the power supply unit including the adapter can be plugged into the XT-30 port on the back of the robot. A successful start of the robot can be checked by hearing the fans. To switch off, simply remove the plug and disconnect the robot from the power supply. To do this, it is only necessary to ensure that all manual operations on the computing units have been completed so that they cannot be switched off. be corrupted.

4.1.4 Graphic applications

After commissioning the Go1, the user has two graphical applications at their disposal with which they can perform robot functions such as monitoring, control and simulations. This chapter briefly documents the initial setup and opening of the applications.

Web interface

The Raspberry Pi of the Go1 automatically starts a web server with a website that makes various functions accessible at system startup. To access the website, you must be in the same network as the Go1. Further information on this is described in chapters 3.3 and 4.2.3. On the HTTP (Hypertext Transfer Protocol) port

80 of the Pis[39] the page can now be accessed. In browsers, all you need to do is enter the IP address; the port does not need to be defined. Figure 16 shows a screenshot of the web interface.

Figure 16: Screenshot of the web interface

Mobile app

Unitree Robotics has also published a mobile application for the *iOS* and *Android* platforms to complement the web interface. This can be downloaded ready-built from the developer website and must be installed on the respective platforms as a finished app from an unknown source[40]. The developer options usually need to be activated for this. As the installation processes for the platforms and end devices may differ, they will not be discussed in this paper. [41] Figure 17 shows two screenshots of the mobile application.

To connect to the robot, you only need to connect to the robot's network. The IP address of the robot must then be stored in the settings. The information from the Go1 can then be viewed via the application.

[39]Accessible at `192.168.123.161` or `192.168.12.1` in your own WLAN hotspot (see 4.2.3)

[40]Unitree Robotics. *Go1 APP Download*. URL: `uhttps : / / www . unitree . com / app/` (visited on 17. 08. 2023).

[41]Further information on installing beta apps for iOS at `https://testflight.apple.com/join/KraKgqam` and for Android at `https://www.heise.de/tipps-tricks/Externe-Apps-APK-Dateien-bei-Android-installieren-so-klappt-s-3714330.html`

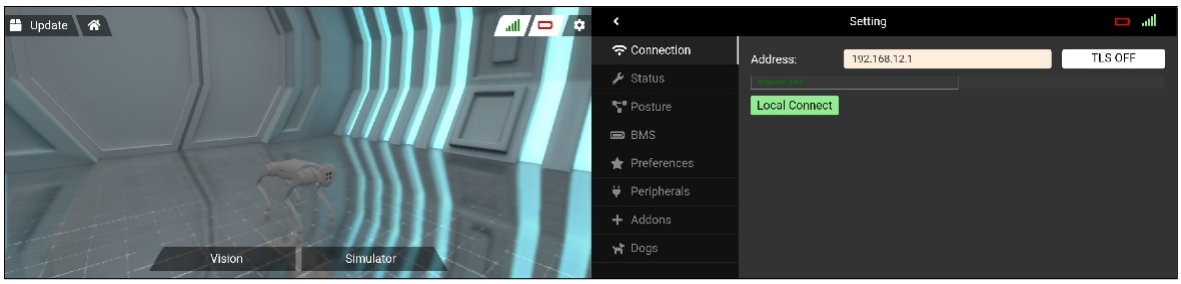

Figure 17: Screenshot of the mobile application

4.2 Functions

The following chapter explains and demonstrates some of the Go1's factory default functions. Some of the functions are already enabled and require little or no configuration. Others are not enabled or are barely documented, which is why they are described in more detail. Some of the functions have not been included in the scope of this work. All known functions that have not been included are listed at the end of the chapter.

4.2.1 Software Autostart

Unitree Robotics has used a simple, user-accessible autostart function of the Raspberry Pi and NVIDIA Jetson operating systems to implement the startup sequences of its software components. On the Ubuntu systems, the package gnome
-startup-applications is used. The installation of the package can be checked using the command apt list -installed | grep gnome-startup-applications. It should be noted that the functionality is provided by the *Gnome* desktop environment, not by the operating system of the computing units. The desktop environment PIXEL (Pi Improved Xwindows Environment, Lightweight) based on LXDE (Lightweight X11 Desktop Environment), which is installed on the Raspberry Pi, provides a different option for the startup configuration. Nevertheless, the setup is identical to the gnome-startup- applications library, which is not installed on the system. An official documentation for the autostart function of the Pi is therefore not available, it is only apparent that the function corresponds to the standard *Autostart Of Applications During Startup* described on the open source project *FreeDesktop*[42]. The function is demonstrated in this chapter using the Raspberry Pi as an example.

[42]FreeDesktop.org. *Autostart Of Applications During Startup*. URL: https://specifications.freedesktop.org/autostart-spec/0.5/ar01s02.html (visited on 17. 08. 2023).

Implementation

To check the implementation, you must first connect to the Raspberry Pi. The first step is to check which user is automatically logged in to the system after a boot so that the desktop environment starts the appropriate programs. The user *pi* is expected here, which is confirmed during the check.

```
pi@ raspberrypi :~ $ cat / etc / lightdm / lightdm .  conf| grep autologin
-
   '→user | grep - v \#
autologin - user =pi
```

The /home/pi/.config/autostart/ folder defined in the FreeDesktop standard only contains a file of the type .desktop. Files with this extension are used by the desktop environment to start programs and to display additional information such as images and descriptions in the user interface.

```
pi@ raspberrypi :~ $ ls / home / pi /. config / autostart /
unitree . desktop
pi@ raspberrypi :~ $ cat / home / pi /. config / autostart / unitree
. desktop [ Desktop Entry ]
Name = unitree Comment =
unitree autostart
Exec = bash / home / pi/ Unitree Upgrade / start .
sh Terminal = false
Type = Application Categories =
System ; Utility ; Archiving ; Startup
Notify = false
No Display = true
```

The value of the Exec field is interpreted and executed as a command line command after the boot process and the user *pi* logs in. A look at lines 6 and 7 of the executed script merely refers to another script.

```
6  cd / home / pi/ Unitree / autostart
7  ./ update . sh &
```

The script /home/pi/Unitree/autostart/update.sh creates a new log file and then initiates the autostart process.

```
13  for dir in ' cat . startlist . sh '
14  do
15    if [[ $dir = \#* ]] ; then
16      echo $dir ': skipped ' >>/ home / unitree / unitree / autostart /.
           '→startlog
```

```
17    else
18      cd $dir
19      echo $dir ':'' sed - n '1p' version . txt ' >> ${ script Path }.
            '→ detailed version
20      ./ $dir . sh
21      sleep 3
22    fi
23    cd $script Path
24  done
```

The content of the .startlist.sh file is a line-break-separated list of all folders that are searched for executable scripts for autostart. Line 13 iterates over all these folders. Line 15 checks whether the current folder in the list has been commented out and skips it if necessary. Otherwise, line 18 navigates to the folder and line 20 executes the script in the folder that has the same name as the folder itself, including the .sh file extension.

To extend the autostart function or to add your own processes after system startup, all you have to do is create a folder in the path /home/pi/Unitree/autostart/ in which an executable file with the name of the folder including the file extension
.sh is located. Further information on this procedure and an example of the extension are shown in section 5.2.

4.2.2 Remote control

The Go1 can be controlled remotely in four different ways:

- Remote control
- App
- Web interface
- Follow-up function

This chapter briefly describes all four options and shows various limitations of the individual implementations. The robot must be switched on and sport mode activated for all four control options. If the sport mode is not activated and a connection to the Raspberry Pi is not possible or desired, it can be activated via the paired remote control and the key combination L2+START. To do this, the Go1 must be in the starting position as described in chapter 4.1.2.

Remote control

The scope of delivery of the Go1 includes two physical remote controls with which the robot can be controlled. The main remote control has two so-called joysticks, which can manipulate the position and movement of the robot in different axes. The second remote control, called *Label Controller* by Unitree, has only one joystick, which can control the movement forwards, backwards, left and right. It also serves as a transmitter for the *sequence function*. Figure 18 shows the main remote control on the left and the label controller on the right.

Figure 18: Main remote control (left) and label controller (right)

Simple operation of the main remote control is already possible after switching on the robot as described in chapter 4.1.2. The remote control automatically pairs with the robot. Documentation on the type of connection cannot be found, but it is assumed that this is not done via Bluetooth, but via a different protocol and that the remote control connects to the MCU instead of the Raspberry Pi, which would also have Bluetooth. However, it is possible to pair the remote control with a smartphone via Bluetooth. To do this, pin 1234 must be used to verify the pairing. The pairing process differs between manufacturers and is therefore not documented here. After successful pairing, the remote control can be connected via the mobile app[43] to connect the remote control. Via the settings and the menu item Peripherals > Bluetooth Gamepad, the remote control with the matching serial number can be selected in the Gamepad List. This is labeled on the remote controls ex works. Figure 19 shows the screenshots of the relevant menu items in the app.

It is now possible to control the Go1 remotely via the remote control, regardless of which network it is in. It is only necessary for the robot and the cell phone to be in the

[43]See chapter 4.1.4

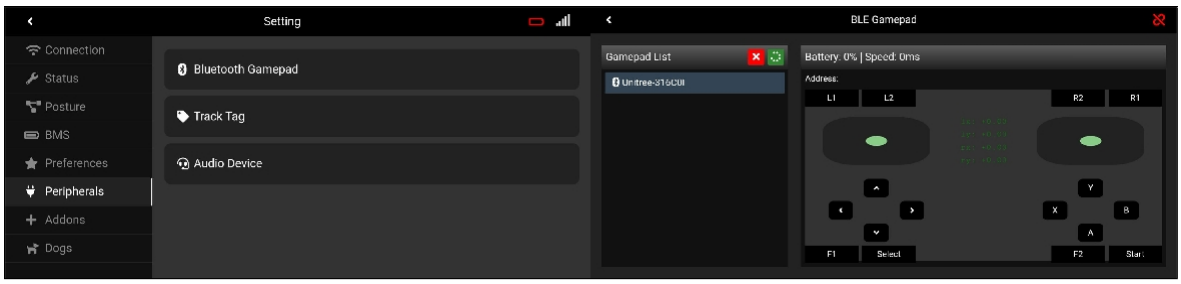

Figure 19: App menu items Peripherals > Bluetooth Gamepad and Gamepad List

are in the same network. Further information on network expansion is documented in chapter 5.

App

The Go1 can also be controlled via the mobile application without having previously connected it to the main remote control. To do this, the main menu item Vision must be selected. Control must then be activated in the top right-hand corner of the camera view. As shown in Figure 20, the mode can then be selected in the bottom right-hand corner of the screen. If you select one of the walking modes here, the Go1 can be moved using the two control units shown. The left-hand control unit of the main remote control is shown on the left and the right-hand unit on the right.

Figure 20: Screenshot of the app controller

Web interface

The website hosted on the Raspberry Pi offers the option of activating the robot controls in the Vision menu in the top right-hand corner of the screen. After activation, the user is presented with two control elements, as shown in Figure 21. These can be controlled on the left using the W-A-S-D buttons and on the right using the ↑-←-↓-→ buttons. The left-hand control unit corresponds to the left-hand side of the main remote control, the right-hand one to the right-hand side.

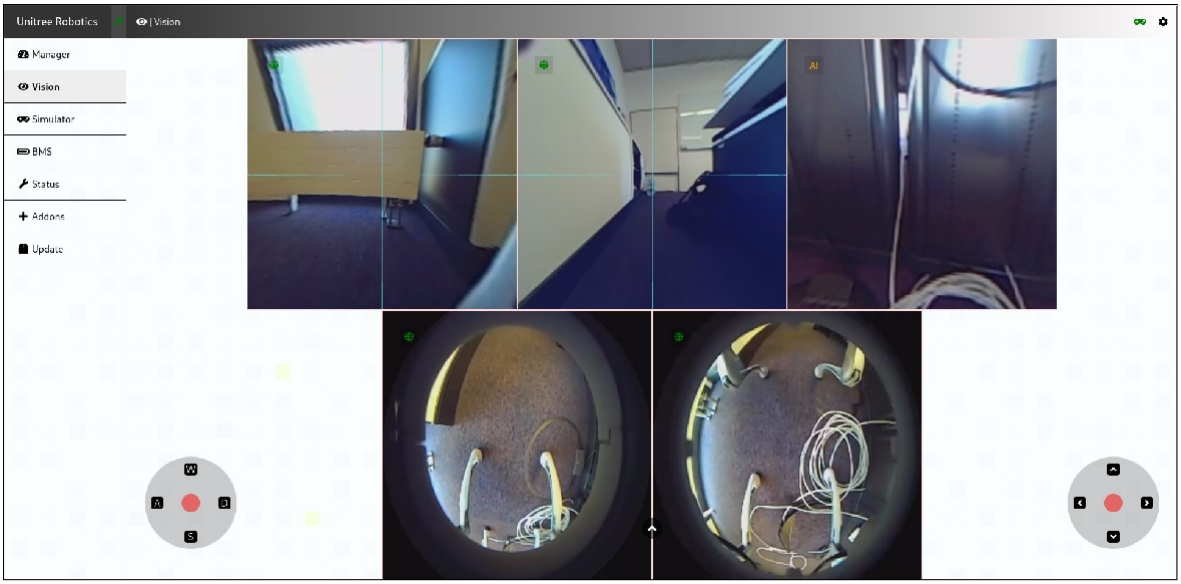

Figure 21: Screenshot of the web controller

Follow-up function

The so-called label controller can be switched on in the same way as the main remote control by pressing and holding the POW button again. The joystick makes it easier to control the robot. However, the actual function of the label controller is the *Follow Me* function advertised by Unitree Robotics. The controller constantly communicates with the robot and exchanges information on the approximate position of the Go1 relative to the label controller. This can be monitored in the mobile app. To do this, navigate to the Peripherals > Track Tag path via the settings. Figure 22 shows the two screen shots.

To make the robot follow, the value of the Follow option must be tapped. This should now change from OFF to ON. The dog will then always move relative to the label controller. The behavior can be minimally adjusted using the buttons on the label controller. Table 4 summarizes the button functions.

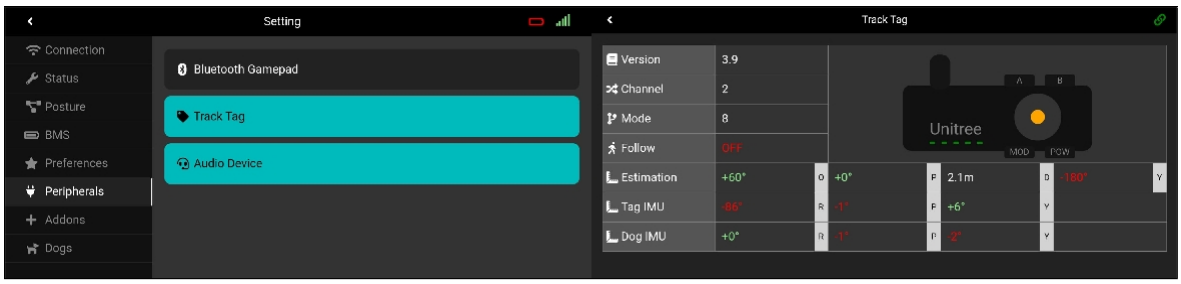

Figure 22: App menu items Peripherals > Track Tag and the tracking overview

button	Function
POW	Press for 1 second → Raise after fall Press briefly 2 times → Change standing modes Standing → Laying → Deactivate motors → Standing
MOD	Short press → Deactivate follow Press briefly 2 times → Change modes Slow following (1.5 m/s) → Fast following (3 m/s)
A	Change to alternative mode (not functional after test)
B	Short press → Rotate counterclockwise by approx. 6° Press 2 times briefly → Reset rotation to default value

Table 4: Summary of the button functions of the Label Controller

It should be noted that the label controller cannot be connected to the cell phone via Bluetooth.

4.2.3 Local network

The Raspberry Pi of the Go1 uses one of its network interfaces to program a WLAN. A look at the autostart module configNetwork shows that the interface wlan1 is used for this purpose.

```
35 # configure WIFI
36 sudo ifconfig wlan1 192 . 168 . 12 . 1 /
37 sudo 24
38 sudo ifconfig wlan1 up
       hostapd / etc / hostapd / hostapd .
       conf &
```

To find further information on the configuration of the WLAN, the hostapd (Host Access Point Daemon) configuration in /etc/hostapd/hostapd.conf must be viewed. The hostapd service is a service available to users of a system for various access points and authentication servers[44].

[44]Jouni Malinen. *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. Jan. 12. 2013. URL: https://w1.fi/hostapd/ (visited on 18. 08. 2023).

```
wpa =2
wpa_key_mgmt =WPA - PSK
rsn_pairwise = CCMP
ssid = Unitree_Go 501075 A
wpa_passphrase = 00000000
```

Here you can see that the SSID of the WLAN access point corresponds to the serial number of the respective Go1. This is marked on various parts of the robot. The default password is 00000000 and can also be changed in the configuration file. After restarting the interface wlan1, this new password comes into effect.

The function of a hidden SSID can also be helpful, as the Go1's access point would otherwise be visible to everyone in the immediate vicinity of the robot, which represents a potential security risk during regular operation. All you need to do is add the line ignore_broadcast_ssid=1 at the end of the file. This setting means that the WLAN publication does not disclose an SSID and connection requests without the full SSID are ignored.

4.2.4 Audio Interfaces

As described in section 3.2.2, a loudspeaker is installed in the head of the Go1. A look at the hierarchy of connected USB devices shows that the device with ID Dev 6 has the Audio class.

```
unitree@ unitree - desktop :~$ lsusb - t |
grep
    S|__ Port 4: Dev 6 , If 0 , Class = Audio
    ,
    S|__ Port 4: Dev 6 , If 1 , Class = Audio
    ,
    S|__ Port 4: Dev 6 , If 2 , Class = Audio
    ,
.
```

```
Class = Audio
Driver =snd - usb - audio , 12 M
Driver =snd - usb - audio , 12 M
Driver =snd - usb - audio ,
12 M
```

```
unitree@ unitree - desktop :~$ lsusb | grep " Device 006"
Bus 001 Device 006: ID 0 d8c :'0012 C- Media Electronics , Inc
```

The mount point /dev/snd/controlC2 of the device can be found via the sym link in the folder Read out /dev/snd/by-id/.

```
unitree@ unitree - desktop :~$  - l / dev / snd / by
ls                              - id/
total 0
                                28 23: 58 usb - C-
lrwxrwxrwx 1 root root 12 1 x   Media_Electronics_Inc .
    '→ _USB_Audio_Device -00 ->  .../ control C 2
```

Two options for using the robot's loudspeaker without further configuration are shown below.

Voice transmission via app

The mobile app for the robot makes it possible to send microphone recordings from the device on which the app is installed to the Go1 and then play them back via the speaker on the back of the head. To do this, the application must first be connected to the robot. The interface for configuring the transmission can then be accessed via the menu item Peripherals > Audio Device. Figure 23 shows the overview of the audio transmission.

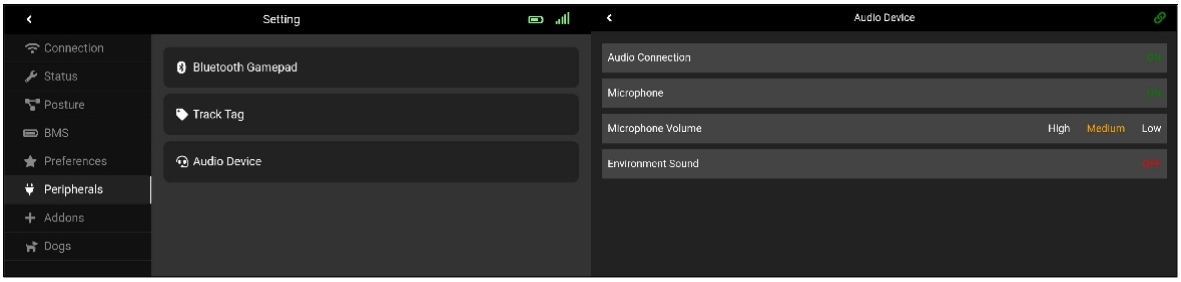

Figure 23: Overview of network configuration

The icon in the top right-hand corner of the screen shows whether the audio interface in the dog's head has been successfully connected. For this, the wsaudio autostart process on the nano must still be running. This can be checked using the following command.

```
unitree@ unitree - desktop :~$ ps - aux | grep
wsaudio
unitree    8205 95.0   0.4 466312 18360 ?          Rl    00:      0:23 ./
   →build /
   wsaudio
```

If this process is not active, the autostart script can be executed manually.

```
unitree@ unitree - desktop :~$ cd / home / unitree / Unitree / autostart
/ wsaudio
   '→/
unitree@ unitree - desktop :~/ Unitree / autostart / wsaudio$ ./ wsaudio
. sh &
[1] 9179
[ wsaudio ] starting ...
unitree@ unitree - desktop :~/ Unitree / autostart / wsaudio$ ps - ax |
grep
   '→wsaudio
 9179 pts /0    Rl      3:15 ./ build / wsaudio
```

When executing autostart scripts manually, it is important that they are started from their folder, as the scripts themselves often work with relative paths.

In the mobile application, the Audio Connection value can now be selected to switch it from OFF to ON. As soon as this has also been done for the Microphone value, audio can be heard via the built-in microphone of the cell phone on which the application is running.

can be recorded. The recorded material is then played back on the loudspeaker with network-related latency.

Playing audio files

To play audio files, they must first be copied to the nano in the robot's head. The scp function installed on most Linux-based operating systems can be used for this. If the computer with the file is in the Go1 network, the following command can be used to copy the file:

```
sshpass - p 123 scp example . wav unitree@ 192 . 168 . 123 . 13 : ~/ Music
```

To be able to use the loudspeaker, all processes that block it must first be terminated. By default, this is only the wsaudio process described in the previous paragraph, which can be terminated with the command pkill -f wsaudio. The file can then be played using the aplay command. This requires the device name, which can be read out with the command aplay -L, the output of which searches for the device with the identifier 2. The identifier was recorded at the beginning of the chapter via the mount point. The output of the file /proc/asound/cards shows the USB speaker under the index 2.

```
unitree@ unitree - desktop :~$ cat / proc / asound /
cards [...]
 2 [ Device          ]: USB - Audio - USB Audio Device
                        C- Media Electronics Inc . USB Audio Device at
                           '→usb - 70090000 . xusb -3.4 , full speed
```

The output of the file /proc/asound/card2/pcm0p/info shows that the card with index 2 has only one channel that can be used for playback [45].

```
unitree@ unitree - desktop :~$ cat / proc / asound / card2 / pcm0p
/ info card : 2
device : 0
subdevice : 0
stream :
PLAYBACK id: USB
Audio name : USB
Audio
subname : subdevice #0
class : 0
subclass : 0
```

[45] `card2` stands for index 2 of the `/proc/asound/cards list`, `pcm0p` for the first `PLAYBACK` device for output

```
subdevices_count : 1
subdevices_avail : 1
```

This results in the direct hardware name of the loudspeaker plughw:2,0 [46]which is required for the following command:

```
unitree@ unitree - desktop :~$ aplay - D plughw :2,0  Music / example .
wav Playing WAVE '/ home / unitree / Music / example . wav ' : Signed 16
bit
   '→Little Endian , Rate 44100 Hz , Stereo
```

For easier handling from an external device, the following commands can also be used if the device is connected to the Go1 network:

```
sshpass - p 123 ssh unitree@ 192 . 168 . 123 . 13 ' pkill - f wsaudio
' sshpass - p 123 ssh unitree@ 192 . 168 . 123 . 13 ' aplay - D plughw
:2 ,0 ~/
   '→Music / example . wav '
```

To adjust the volume of the speaker, the library, which also contains aplay, also provides the amixer function. With the command amixer -c 2 set Speaker <0-100>%. The -c 2 option again refers to the device ID.[47]

4.2.5 Head lighting

The LED rows on the two outer sides of the head are connected to the head's Jetson Nano, as described in chapter 3. If you check the functions there that are controlled via the autostart, you will notice two folders - faceLightServer/ and faceLightMqtt/. Unfortunately, the functions of both scripts are stored as binary files and therefore cannot be read out easily. The manufacturer's documentation also contains no information about the LED series. However, the name of the second folder indicates a possible functionality of the control via MQTT. To confirm this, an MQTT explorer can be used. This registers as a client with the MQTT broker and records all messages and published topics. You can find out more about MQTT on the official documentation of the standard[48], examples of using the standard on the robot in chapter 5.

To use the MQTT Explorer, the IP address of the broker and the port on which it is published are required. Only the IPs 192.168.123.13 are available for the address, ...14, ...15 and ...161 come into question. The *Nmap* library can be used to check whether

[46]`plug` stands for plugged hardware `hw` (e.g. USB), 2 for the index and 0 for the channel (subdevice)

[47]Alsa-Project.org. *Advanced Linux Sound Architecture (ALSA) project homepage*. Oct. 20, 2020. URL: `https://www.alsa-project.org/wiki/Main_Page` (visited on 18. 08. 2023).

[48]https://mqtt.org/

the MQTT standard port 1883 on one of the registered IPs in the network 192.168.123.0/24 is open.

```
nmap - sS - O - p1883 192 . 168 . 123 . 0 / 24
```

The output shows that the two IPs 192.168.123.15 and 192.168.123.161 have the MQTT port open. The NVIDIA Jetson Xavier NX is first tested as a broker via the MQTT Explorer. The connection works, but neither available topics nor messages are output. A test on the Raspberry Pi with the IP 192.168.123.161 shows the output as shown in Figure 24.

Figure 24: Output of an MQTT Explorer in conjunction with the Raspberry Pi as a broker

No information about the LEDs can currently be seen in the displayed topics. This may be due to the fact that no messages have yet been sent for the topic, which is a plausible conclusion as the Go1's headlights are switched off by default. If you now change this using the function provided in the mobile application under the Preferences menu item, you will also see the face_light/color topic in the MQTT Explorer, as shown in Figure 25.

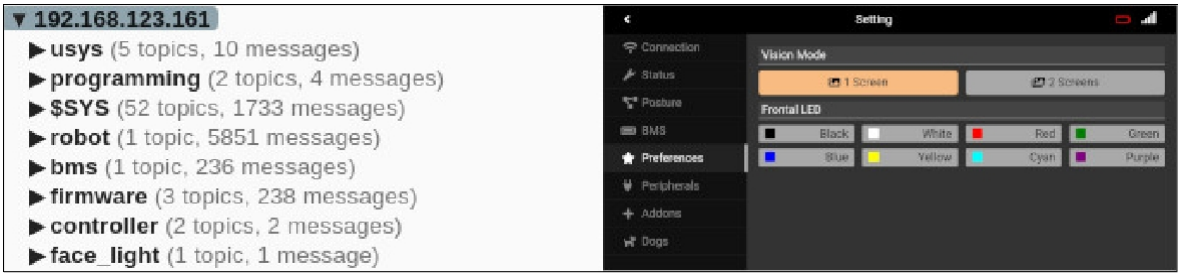

Figure 25: MQTT Explorer with topic face_light/color (left) and app function (right)

The message payload can be output via the *Mosquitto client* command line tool, for example. The following command must be used to output the binary payload in readable form. The computer used for this must be in the Go1 network.

```
mosquitto_sub - h 192 . 168 . 123 . 161 - t face_light / color
- F % x ff 0000
00 ff00
0000 ff
```

The three values in the last lines of the output are formatted message payloads of the topic face_light/color and were output when the three colors *red*, *green* and *blue* were set in this order in the mobile application. It can therefore be deduced that the robot's LEDs can be configured via the RGB *color space*. The mixture of red, green and blue can be set here for each color with a value of 0 - 255.

The mosquitto_pub command can now also be used to change the color of the robot. To do this, you only need to send the message payload in binary form. The following command changes the head light of the Go1 to red.

```
echo - ne "\ xFF \ x00 \ x00 \ x00 " | mosquitto_pub - h 192 . 168 . 123 .
161 - t
   '→face_light / color - s
```

It is also conceivable to control the robot's LEDs directly via the CP210x UART bridge, which was mentioned in chapter 3.2.2. However, this was not implemented as part of this work and may be documented in the future.

4.2.6 Video streaming

With its five cameras, the Go1 offers the possibility of transmitting images of its surroundings and thus enabling users to control the robot from a distance. The positioning, distribution and mounting points of the cameras within the robot, the computing units and the operating systems have already been described in chapter 3.2. To summarize briefly, two cameras are positioned in the head of the robot, facing forwards and downwards. Both are connected to the Jetson Nano inside the head. The two outer sides of the torso are equipped with two cameras that are connected to the Jetson Nano in the torso of the Go1. The last camera on the underside of the fuselage is connected to the Jetson Xavier NX. Using the forward-facing camera in the head of the Go1 as an example, this chapter will briefly explain how the cameras can be accessed and how to access the cameras from a connected computer outside the robot.

images can be accessed. The instructions shown are identical for all other cameras except for any mounting points and IP addresses.

Access to camera images

To be able to use the Go1 cameras, all processes that block the devices themselves must first be stopped. This can be checked using the command fuser -vm /dev/video1, here you must search for the lines that have the value m for memory mapped files as the ACCESS flag after the output. The processes can then be terminated with their names.

```
pkill - f point_cloud_nod
pkill - f example_point
```

To access the camera data via the mount point /dev/video1, the ffmpeg package can be used, which is pre-installed on all Ubuntu systems of the Go1. The following command including an explanation of the options can be used to start a video stream via RTSP (Real-Time Streaming Protocol) to a streaming server using ffmpeg.

```
ffmpeg - nostdin \                    # No interaction
    - f video 4 linux 2 \             # Input Format
    - i / dev / video 1 \ \           # Input - URL
    - vcodec libx 264 \               # Video encoding ( h264 )
    - preset : v ultrafast            # Encoding speed # No H264
    \                                 B-frames
    - tune zerolatency \              # Frame rate
    - framerate 5 \                   # Output format
    - f rtsp \                        # Output File / URL
    rtsp://< ip : 8554 | port >/ <
    stream >
```

Further details on execution and streaming via a server are covered in Chapter 5. The camera image is not processed in any way when streaming via ffmpeg and looks as shown in Figure 26.

The *OpenCV* library has already been implemented in the robot to process and connect the two *fisheyes* of the camera images. The *Unitree Camera SDK* mentioned in chapter 2.2.3 also uses OpenCV in its implementation. However, the extended functions provided by the use of OpenCV are not discussed in this thesis. For this purpose, the work "Development of an intelligent lidar-based 3D navigation system for the Unitree GO1"[49] can be consulted.

[49]Kemnitzer, see note 1.

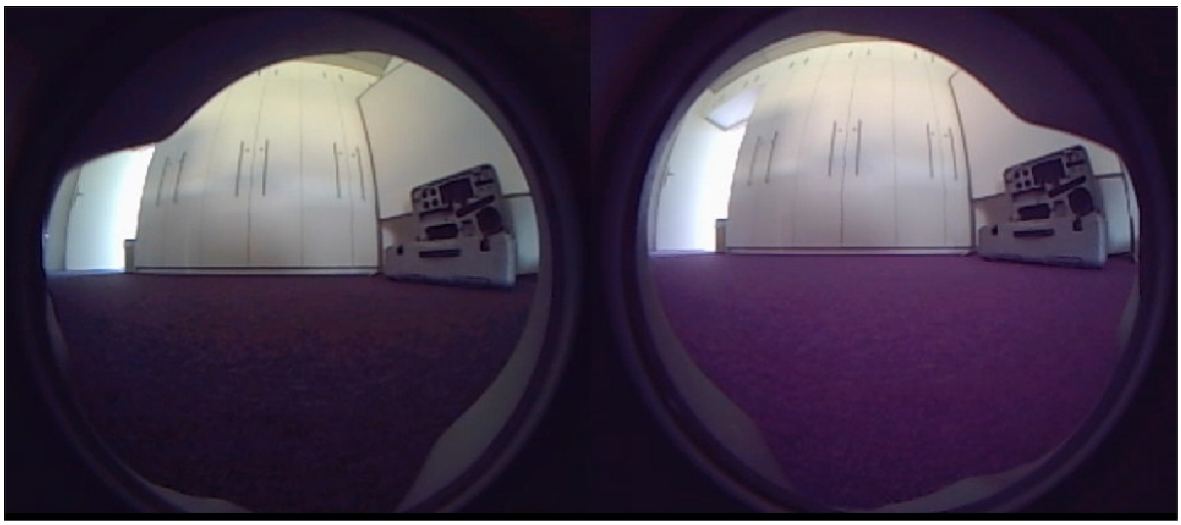

Figure 26: Camera image of the Go1

4.2.7 Battery management

The manufacturer's documentation and advertising indicate in some places that an intelligent BMS is installed in the Go1. This enables users to access information on the battery status in real time and react to the information if necessary. The manufacturer's documentation for the robot does not document how the data can be recorded or interpreted; reference is only made to the two overviews in the mobile application and the website, which are shown in Figure 27.

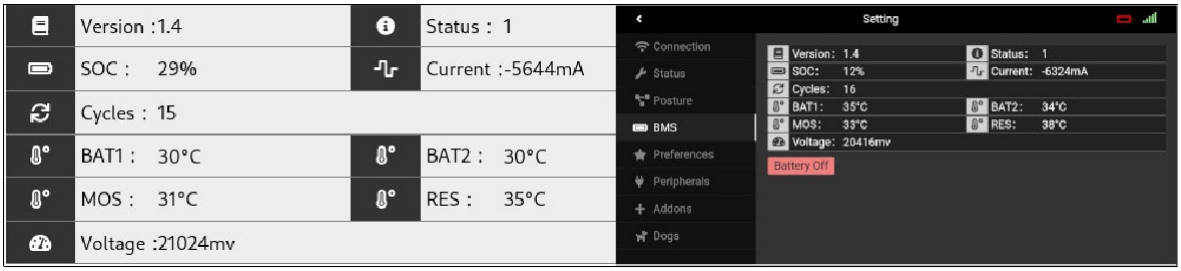

Figure 27: Battery information in the website (left) and app (right)

The output of the MQTT explorer from section 4.2.5 contains a topic called bms/state. Checking the website via the developer tools within modern browsers also indicates that the BMS data is provided via MQTT. The file path src/plugins/ mqtt/receivers/bmsReceivers.ts and the configured MQTT topic bms/state confirm this. If you now output the message payloads of the topic, you will receive the following output format.

```
mosquitto_sub - h 192 . 168 . 123 . 161 - t bms / state - F % x 0104011
bf 5 e 8 ffff 0 f 001 e 1 e 1 f 23800 da 00 d 20002000 a 00 da 00 da 00 d
 20002000800 d
```

The structure of the data can be traced via the developer tools. Line 14 of the file bmsReceivers.ts shows the conversion of the message payload from a byte buffer into a Uint8Array. According to the JavaScript documentation, a Uint8Array is an array of 8-bit unsigned little endian integers[50]. This means that two digits of the hexadecimal output of the BMS can be interpreted as one value of the array. Lines 14 to 17 and line 20 in Listing 4 show the conversions of the 8-bit integers.

```
14 const uint 8 s = new Uint 8 Array ( message );
15 data . bms . version = uint 8 s [0] + "." + uint 8
16 s [1]; data . bms . status = uint 8 s [2];
17 data . bms . soc = uint 8 s [3];
18 data . bms .        = data View . get Int 32 (4 , true );
19 current             data View . get Uint 16 (8 , true ); [ uint 8 s
20 data . bms .        [10] , uint 8 s [11] , uint 8 s [12] , uint 8 s [
   cycle = data        13] ;
21 for ( let i = 0; i < 10;i ++) {
   bms . temps =
22   data . bms . cell Voltages [ i] = data View . get Uint 16 (14+
23   i*2 , true );
24 }
   data . bms . voltage = data . bms . cell Voltages . reduce (( a,
   c) => a+c);
```

Listing 4: Contents of the webpack file bmsReceivers.ts

The documentation of the DataView class shows that the functions getInt32() and get Uint16() have the following syntax[51].

```
get Int 32 ( byte Offset , little
Endian ) get Uint 16 ( byte Offset ,
little Endian )
```

Consequently, in line 18 a 4-byte integer is read from the fifth byte of the message payload, in line 19 a 2-byte integer is read from the ninth byte and ten more from the fifteenth byte of the payload. The last ten 2-byte integers are added together to form a total number. The following overview of the payload format results from the evaluations of the website. The example used is the output of the mosquitto_sub command shown above.

Chapter 5 shows how the information from the BMS can be read out and utilized in a meaningful way.

[50] MDN contributors. *Uint8Array*. Aug. 14, 2013. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Uint8Array (visited Aug. 22, 2023).

[51] MDN contributors. *DataView*. Aug. 21, 2013. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/DataView (visited on 22. 08. 2023).

Byte	Format	Contents	Example	Conversion	
0-1	2 x uint8	Version	01, 04	v1.4	
2	uint8	Status	01	1	
3	uint8	State of Batch	1b	27 %	
4-7	int32	Electricity	f5e8ffff	-5899 mA	
8-9	uint16	Cycle	0f00	15	
10-13	4 x uint8	Temperatures	1e, 1e, 1f, 23	30 °C, 30 °C 31 °C, 35 °C	
14-33	10 x uint16	Cell voltage	800d, a00d, 2000, 2000, a00d, a00d, a00d, 2000, 2000, 800d	3456 mV, 3488 mV, 32 mV, 32 mV, 3488 mV, 3488 mV, 3488 mV, 32 mV, 32 mV, 3456 mV	**Sum:** 20,992 V

Table 5: Overview of the BMS message payload

4.3 Further functions

In addition to the functions documented in this document, the Go1 has other functions which are briefly listed and explained here.

- **ROS:** ROS is installed on some of the robot's computing units, which can be used for extensive functional enhancements in the field of robotics.
- **SLAM (Simultaneous Localization and Mapping):** Via the mobile application and some pre-installed libraries, the robot has the function of mapping an image of the environment and localizing itself in it using the built-in sensors.
- **OpenCV camera evaluation:** The supplied *Unitree-Camera-SDK* library uses OpenCV to evaluate the camera images. OpenCV can be used to combine the camera images into one image, use the camera's depth measurement and evaluate the images.
- **Ultrasonic sensors:** The ultrasonic sensors can also be read out via an official library in order to react to movements and obstacles in the robot's environment.
- **Motion control:** The motors of the Go1 can be controlled programmatically in two modes, the high and low level modes.

The functions ROS, SLAM, OpenCV and the use of ultrasonic sensors are addressed and documented in the work "Development of an intelligent lidar-based 3D navigation system for the Unitree GO1"[52].

[52]Kemnitzer, see note 1.

5 Functional enhancements

The following chapter deals with various extensions that are particularly useful for long-term use of the robot. These include points such as the constant connection to the Internet, a possible connection over long distances including remote control of the robot and the transmission of camera images for various purposes, such as controlling the robot from a distance.

5.1 Connectivity

Chapter 3.3 has already explained how the internal network of the Go1 and all its computing components is structured. Chapter 3.2 showed how to connect to the individual computing components. In the following chapter, further options will be developed to make the Go1 accessible via various networks. All of the following connectivity extensions have been tested and documented accordingly.

Wifi

The Raspberry Pi of the Go1 has three WLAN interfaces, only one of which is used to connect to its own access point. The free interface wlan2 can be used to connect the robot wirelessly to the Internet within an existing network. To do this, the configNetwork autostart function on the Raspberry Pi must be adapted. As documented in chapter 4.2.1, this can be found in the path /home/ pi/Unitree/autostart/configNetwork. In the configNetwork.sh script, all interfaces are switched off at the beginning, after which only the eth0 and wlan1 interfaces are reactivated and configured. The interface wlan2 must therefore first be reactivated.

```
pi@ raspberrypi :~ $ sudo ifconfig wlan2 up
```

Connection through the wpa_supplicant

The installed operating system of the Pi, Debian (Raspbian) 10, uses the wpa_supplicant package to configure the wireless network connections. In order not to corrupt the sensitive network configurations, a new configuration file is created for connecting to an access point. This is done in the directory
/etc/wpa_supplicant/ to create a new directory for your own connections.

```
pi@ raspberrypi :~ $ mkdir / etc / wpa_supplicant / config . d
```

The necessary configurations are then written to a file called wlan2.conf. It is important to set the correct country abbreviation in order to make it clear to the operating system which legal principles must be observed for the wireless connection via WLAN.

```
pi@ raspberrypi :~ $ echo "\ ctrl_interface = DIR =/ var /
run / wpa_supplicant GROUP = netdev update_config =1
country =DE \ n" \
> wlan2 . conf
```

This country reduction must also be stored globally in the system.

```
iw region set DE
```

The access data for the access point can then be sent in encrypted form to the wlan2.conf file must be attached.

```
pi@ raspberrypi :~ $ wpa_passphrase Example - SSID password >> / etc /
   '→ wpa_supplicant / conf . d/ wlan2 . conf
```

The file should now look like this:

```
1  ctrl_interface = DIR =/ var / run / wpa_supplicant GROUP = netdev
2  update_config =1
3  country =DE
4  network ={
5    ssid =" Example - SSID "
6    # psk =" password "
7    psk =6 d 2 c 8604 ecb 1 f 4825 d 410 b 859 ed 0 f
       '→ a 19621 bea 7 ffa 0 b 1 c 9 b 8 bdda 995 c 7135 c 20
8  }
```

Line 6 - psk - should be removed to keep the password secret. The interface with the created configuration can now be connected to the desired access point.

```
pi@ raspberrypi :~ $ wpa_supplicant - B - i wlan2 - c / etc /
wpa_supplicant /
   '→ conf . d/ wlan2 . conf
```

If there is already a running configuration for the interface, this must first be removed with the following command.

```
pi@ raspberrypi :~ $ sudo rm / var / run / wpa_supplicant / wlan2
```

The robot's Raspberry Pi is now connected to the access point. If this is also connected to the Internet, the Raspberry Pi will have a functioning network connection as long as it is within radio range of the access point.

Automation of the connection

The configNetwork auto-start function can be adapted to automatically connect the Pi to the Internet after system startup. An additional script containing the new logic should be created for this purpose. This can look as follows:

```
1  #!/ bin / bash
2  # / home / pi/ Unitree / autostart / config Network / connect
   Wlan 2 . sh
3  sleep 1
4  echo "[ connect Wlan 2 ] Start initializing wlan2
   interface "
   '-> $to start log
5  sudo ifconfig wlan2 up
6  sudo rm / var / run / wpa_supplicant / wlan2
7  echo "[ connect Wlan 2 ] Connecting wlan2 interface " $to Startlog
8  sudo wpa_supplicant - B - i wlan2 - c / etc / wpa_supplicant /
   config . d/
      '-> wpa_wlan 2 . conf
9  echo "[ connect Wlan 2 ] Done connecting wlan2 interface via
      '-> wpa_supplicant " $to Startlog
10 sleep 3
```

The script can now be called up at the end of the existing configNetwork.sh script.

```
42 sudo ./ connect Wifi 2 . sh&
```

The Raspberry Pi now connects to the configured network during system startup.

Mobile radio

According to the manufacturer's advertising, the Go1 *MAX* and *EDU* models have a built-in 4G/LTE modem, which can be used to connect the robot permanently to the internet or to control it remotely. However, the manufacturer's documentation does not provide any information

to configure or use the modem. A look at the connected USB devices provides an indication of the exact modem.

```
pi@ raspberrypi :~ $ lsusb | grep Quectel
Bus 001 Device 003: ID 2 c7c : 0125 Quectel Wireless Solutions Co., Ltd .
   '→  EC25 LTE modem
```

A *Quectel EC25 LTE modem* is installed, which is frequently used in Linux circles and can therefore also be used without having to install new drivers.

Preparation

A SIM card is required to connect the modem to the mobile network. This can be inserted into the slot provided on the top of the fuselage. More detailed information on the positioning of the slot can be found in chapter 3.1.4. The slot is suitable for micro SIM cards measuring 12 mm by 15 mm.

The majority of common SIM cards require a PIN (Personal Identification Number) for unlocking. This should be deactivated for simplified use in the robot. As this function can vary for all card providers, the provider's documentation should be consulted. The PIN function can often also be deactivated by end devices such as smartphones. Here too, however, the procedures vary greatly, which is why the deactivation of the PIN is not discussed further. After deactivation, the SIM card can be inserted into the slot and the robot switched on.

Analysis of the modem

LTE modems can usually be operated in different modes. The drivers used for the device also differ depending on the mode. To continue with the configuration of the modem, the mode in which the device is operated on the Pi must first be determined. The lsusb device number can be used for this, which can be determined as follows.

```
pi@ raspberrypi :~ $ lsusb | grep Quectel
Bus 001 Device 003: ID 2 c7c : 0125 Quectel Wireless Solutions Co., Ltd .
   '→  EC25 LTE modem
pi@ raspberrypi :~ $ lsusb - t  grep " Dev 3"
              |                 0 , Class = Vendor Specific Class ,
        |__ Port 3: Dev 3 , If Driver =
           '→option , 480 M
                                1 , Class = Vendor Specific Class ,
        |__ Port 3: Dev 3 , If Driver =
           '→option , 480 M
```

```
        |__ Port 3: Dev 3 , If  2 ,Class = Vendor Specific  Class , Driver =
           '→option , 480 M
        |__ Port 3: Dev 3 , If 3 ,Class = Vendor Specific  Class , Driver =
           '→option , 480 M
        |__ Port 3: Dev 3 , If 4 ,Class = Vendor Specific  Class , Driver =
           '→qmi_wwan , 480 M
        |__ Port 3: Dev 3 , If  5 ,Class = Audio , Driver    d - usb udio ,
                                                 =sn          - a
           '→480 M
        |__ Port 3: Dev 3 , If 6 , Class = Audio , Driver =snd - usb -
        audio ,
           '→480 M
        |__ Port 3: Dev 3 , If 7 , Class = Audio , Driver =snd - usb -
        audio ,
           '→480 M
```

You can see that the device is operated with the qmi_wwan driver. In this mode, the modem is listed as a wwan interface.

```
pi@ raspberrypi :~ $ ifconfig - a | grep wwan wwan0
: flags =4098 < BROADCAST , MULTICAST >mtu 1500
```

For simplified use of the modem in qmi mode, the package libqmi-utils is installed. To do this, the Raspberry Pi must be connected to the Internet[53].

```
pi@ raspberrypi :~ $ sudo apt update && sudo apt install libqmi - utils
```

To use the qmicli tool of the libqmi-utils package, the device path in the file system must first be determined. The kernel messages can be consulted for this.

```
pi@ raspberrypi :~ $ dmesg | grep qmi
[    7 . 606540 ] qmi_wwan 1 - 1. 3: 1. 4: cdc - wdm0 : USB WDM device
[    7 . 682555 ] qmi_wwan 1 - 1. 3: 1. 4 wwan0 : register ' qmi_wwan ' at
usb -
   '→fe 980000 . usb -1.3 , WWAN / QMI device , 8 e: c2 : 07 : 55 : 9 c: c3
[    7 . 686460 ] usbcore : registered new interface driver qmi_wwan
```

The device name is specified in the first line of the output as cdc-wdm0. This can be confirmed via an output of the possible devices in the /dev/ directory.

```
pi@ raspberrypi :~ $ ls / dev / cdc -*
/ dev / cdc - wdm0
```

[53] See chapter 5.1

Preparing the modem

Some preparations must now be made to connect the modem to the mobile network. First check whether the modem is *online*.

```
pi@ raspberrypi :~ $ sudo qmicli - d / dev / cdc - wdm0 -- dms - get -
operating -
   '→ mode
error : couldn ' t create client for the ' dms ' service : CID allocation
   '→ failed in the CTL client : Transaction timed out
```

If the error message occurs as here, the device is blocked by another process or service. A common service that accesses the modem is often the ModemManager.service. The following command checks whether this is active.

```
pi@ raspberrypi :~ $ systemctl | grep Modem
Modem Manager . service  loaded active running    Modem Manager
```

The following command stops this. The modem can then be queried again via qmicli.

```
pi@ raspberrypi :~ $ sudo systemctl stop Modem Manager
pi@ raspberrypi :~ $ sudo qmicli - d / dev / cdc - wdm0 -- dms - get -
operating -
   '→ mode
[/ dev / cdc - wdm0 ] Operating mode
        retrieved : Mode : ' online '
        HW restricted : ' no '
```

If the qmicli command still fails, you must check whether another process of the same package is blocking the modem. This can be done using the query ps aux | grep qmi. A possible blocking process must then be terminated via its process ID and the pkill command. If the displayed mode of the qmicli command is not online, this value must be set manually.

```
pi@ raspberrypi :~ $ sudo qmicli - d / dev / cdc - wdm0 -- dms - set -
operating -
   '→ mode =' online '
[/ dev / cdc - wdm0 ] Operating mode set successfully
```

Connection with mobile network

Next, the modem can be dialed into the mobile network. The following command can be used for this.

```
pi@ raspberrypi :~ $ sudo qmicli \
    - p \ # Request to use the ' qmi - proxy ' proxy
```

```
    - d / dev / cdc - wdm0 \ # Device Path
    -- device - open - net =' net - raw - ip|net - no -    \ # Open device
    qos - header '
       '→ with specific link protocol and QoS flags      # IPv4 and APN
    -- wds - start - network =" apn =' internet ', ip -   when exiting
    type =4" \
    -- client - no - release - cid # Do not release the
CID [/ dev / cdc - wdm0 ] Network started
Packet data handle : ' 2269312560 ' [/ dev
/ cdc - wdm0 ] Client ID not released :
Service : ' wds '
CID : '17 '
```

The value for the APN (Access Point Name) must be obtained directly from the provider, as this can vary depending on the type of card and network. In the test, the specification of the APN was not relevant, but this can be a possible source of error. Another possible error could be that the modem was not set for the raw-ip protocol. In this case, the network interface must be stopped, the modem configured and then reactivated. The operating mode can then be checked again.

```
pi@ raspberrypi   $ sudo ip link set wwan0 down
:~                $ echo 'Y' | sudo tee / sys / class / net / wwan0 /
pi@ raspberrypi     qmi /
:~
   '→ raw_ip
Y                 $ sudo  ip link set wwan0 up
                  $ sudo  qmicli - d / dev / cdc - wdm0 -- wda - get - data
pi@ raspberrypi     format
[/ dev / cdc - wdm0 ] Successfully got data
:~ pi@
format
raspberrypi :~       QoS flow header : no
                Link layer protocol : ' raw -
                ip '
    [...]
```

The output also shows that no QoS header should be set. The command to connect to the network can now be executed again.

The last step is to assign an IP address to the modem in the mobile network. As soon as the modem is dialed into the mobile network, it can be assigned an IP address via DHCP. To do this, another package is installed that searches for a configuration and, if none is found, makes a request in the network to assign an IP.

```
pi@ raspberrypi :~ $ sudo apt install udhcpc
pi@ raspberrypi :~ $ sudo udhcpc - q - f - i
wwan0 udhcpc : started , v1 . 30. 1
No resolv . conf for interface wwan0 . udhcpc
udhcpc : sending discover
```

```
udhcpc : sending select for 10 . 114 . 75 . 205
udhcpc : lease of 10 . 114 . 75 . 205 obtained , lease time 7200
```

The modem is now connected to the mobile network and therefore to the Internet.

Automatic connection

Once the modem has been configured, the Raspberry Pi is connected to the Internet until the next system start. However, if you restart the Pi, you will have to repeat the procedure from the beginning of the chapter. The autostart function described in chapter 4.2.1 can be used to simplify the automated connection to the mobile network. More precisely, the configNetwork function can be extended here, as already documented in chapter 5.1. To do this, simply call another script at the end of the configNetwork.sh file.

```
43 sudo ./ connect Wwan 0 . sh&
```

The content of the file reflects the connection of the modem to the network and the assignment of the IP via DHCP and looks as follows.

```
1  #!/ bin / bash
2  # / home / pi/ Unitree / autostart / config Network / connect
   Wwan 0 . sh
3  sleep 1
4  echo "[ connect Wwan 0 ] Preparing environment for wwan0 interface "
5     '-> $to start log
   sudo systemctl stop Modem Manager
6  echo "[ connect Wwan 0 ] Start initializing wwan0 interface "
7     '-> $to start log
8  sudo ifconfig wwan2 up
9  sudo qmicli - p - d / dev / cdc - wdm0 -- device - open - net =' net -
   raw - ip|net
      '-> - no - qos - header ' -- wds - start - network =" apn =' internet
      ' ip - type
10 echo "[ connect Wwan 0 ] Getting IP for wwan0 interface " $to
11 sudo udhcpc - q - f - i wwan0 -> $to Startlog
12 echo "[ connect Wwan 0 ] Connected wwan0 to internet " $to Startlog
13 sleep 3
```

The robot now connects to the provider's mobile network via modem when the system is started, provided there is a valid and unlocked SIM card in the slot on the top of the body.

Outlook

In future, the connection can be used to control the robot from a distance. A so-called VPN (Virtual Private Network) can be used for this purpose. The robot's Raspberry Pi would have to be registered as a *client* in the VPN. If you now connect to the VPN with another client, both devices are in the same network, regardless of which network they are connected to the Internet via.

5.2 BMS

If you use the Go1 in battery mode, you should always make sure that the battery charge does not fall below 3%, as the motors are switched off at this charge level for safety reasons. However, the development team at Unitree has invested little work in this safety function. The only permanently visible battery charge indicator is attached to the outside of the battery and can only be seen from one side. In addition, the display can only show the charge in eight steps and is therefore not accurate enough. The abrupt shutdown of the motors is also potentially harmful to the robot, as it is not first brought into a lying position, which can result in it collapsing in any conceivable position. The individual limbs of the four legs and the torso then fall unprotected and hit the ground.

To counter this shortcoming and make the robot more resilient to avoidable damage, a battery monitoring system with active warning and cautious motor shutdown is implemented in this chapter. The aim is to display a warning via the LEDs on the head when the battery charge is low at 5% and then gently lower the Go1 to the ground before the built-in mechanisms take effect at 3% charge and switch off the motors. The already installed MQTT and Autostart functions are used for this purpose. Documentation for this can be found in chapter 4.2.

Draft

Figure 28 shows the sequence diagram for displaying the battery monitoring.
The robot components involved are the autostart function, the BMS sniffer, LED control and movement control software components, the MQTT broker, the head LEDs, the MCU and the twelve motors. Only the autostart function and the MQTT broker are actually interacted with. The three software components still have to be developed.

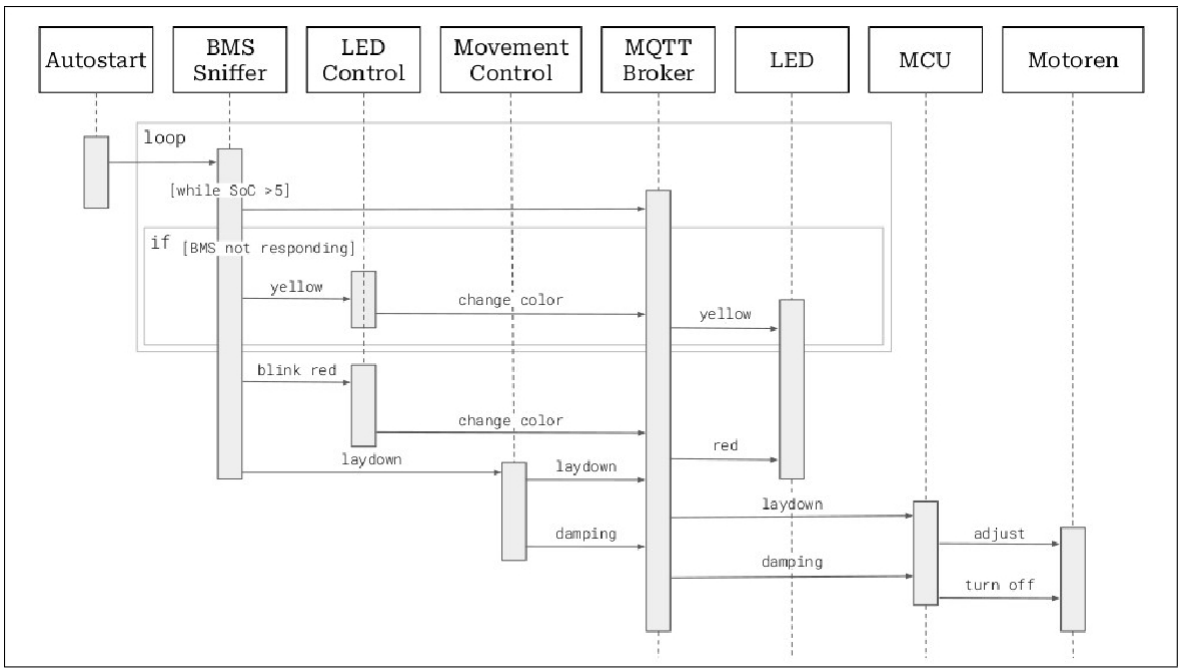

Figure 28: Sequence diagram of the battery monitoring components

The autostart module on the Raspberry Pi is to be expanded to include the new bmsMonitoring functionality. A folder with a script is created for this purpose, which triggers all other software components. The BMS-Sniffer module should permanently query the BMS data via the MQTT broker. If the BMS cannot be reached, for example because the robot was started in mains operation without a battery, the LED Control component is called up, which causes the robot's LEDs to light up yellow continuously. If the BMS can be reached and the SoC (State of Charge) falls to 5% or less, the LED control module is called first to make the Go1's LEDs flash red, after which the movement control module is called to lay the robot down and then relax the motors (damping state). Both the LED control module and the movement control module use the MQTT broker as an interface to the hardware components to be controlled.

Implementation

To implement the BMS monitor, the autostart function must first be added. To do this, the folder /home/pi/Unitree/autostart/bmsMonitor must be created. The script bmsMonitor.sh is created in this folder, which simply starts another script called run.sh as a background process. This step is necessary to decouple the monitoring function from the autostart process. The run.sh script now starts all logging activities and the necessary software components. The last step is to add the new function to the startup list /home/pi/Unitree/autostart/.startlist.sh.

To simplify the development and adaptation of the autostart function, the source code can be kept up to date in a folder elsewhere via the *Git* version management system. The individual files and folders can be integrated into the autostart function via an installation script. The following listing shows the relevant parts of the installation script. The complete files can be found in the appendix.

```
33 # Delete folder
34 rm - r " $DIR / bms Monitor " & > / dev / null
35 # Create symbolic links to the repository
36 cp - rs " $SCRIPT_DIR / bms Monitor " " $DIR "
37 # Copy script to get paths
38 cp - f " $SCRIPT_DIR / bms Monitor / bms Monitor . template . sh" "
   $DIR /
      '→ bms Monitor / bms Monitor . sh"
39 # Make scripts executable
40 chmod +x " $DIR / bms Monitor / bms Monitor . sh" " $SCRIPT_DIR / bms
   Monitor /
      '→ run . sh"
```

The following overview shows the structure of the source code folder for classifying the installation script and the further explanations.

```
go1-bmsMonitor
├── bmsMonitor
│   ├── bmsMonitor.sh
│   ├── constants.sh
│   ├── led_control.py
│   ├── requirements.txt
│   ├── run.sh
│   └── sniff_bms.py
└── install.sh
```

The actual start of the bmsMonitor is the file run.sh, which is started as a background process by the autostart. This first installs all the required Python

libraries that are stored in the requirements.txt file. The script sniff_bms.py, which contains the BMS monitor.

```
8  python 3 - m pip install - r / home / pi/ unitree / autostart / bms
   monitor /
      '→ requirements . txt
9  python 3 / home / pi/ unitree / autostart / bms monitor / snif_bms . py
```

Lines 13 to 23 of the sniff_bms.py script show the decoding of the binary message payload for new MQTT messages and the further evaluation of the SoC.

```
13  def on_message ( c, userdata , msg ):
14      print (" message ")
15      [ ver0 , ver1 , status , soc , current , cycle , temp0 , temp1 ,
16       temp2 , temp3 , cell_v0 , cell_v1 , cell_v2 , cell_v3 ,
17       cell_v4 , cell_v5 , cell_v6 , cell_v7 , cell_v8 , cell_v 9 ] \
18          = struct . unpack (' BBBBi HBBBBBBHHHHHHHHHHHH ', msg .
payload )
19      if not bms_responding ( msg . payload ):
20          const_led ( c, 10 , 10 , 0)
21      elif 5 >= soc :
22          alert_led ( c, 255 , 0 , 0 )
23          lay_down ()
```

The struct.unpack(...) method shows the decoding of the message payload of the topic bms/state developed in section 4.2.7. Line 19 checks whether the BMS is active. If the BMS is inactive, the message payload consists only of binary zeros. In this case, the function const_led(c,r,g,b) from the LED-Control module in the led_control.py file is called.

```
14  def const_led ( client , r, g, b):
15      client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB ', r,
        g, b),
           '→  qos = QOS )
16      time . sleep (1)
```

The transfer parameters r, g and b are the RGB values that the LEDs on the head of the robot should assume. The parameter c is the MQTT client instance that is used to communicate with the broker. The format of the MQTT topic face_light/color is described in chapter 4.2.5. The RGB values r=10, g=10 and b=0 set the LEDs to a strongly dimmed yellow.

Line 21 of the sniff_bms.py file checks the value of the battery's SoC; if this is greater than 5, the function ends and the system waits for the next MQTT message. Otherwise, the LEDs in the alert_led() function of the led_control.py file are configured to flash red.

```
7  def alert_led ( client , r, g, b):
8      client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB ', r,
       g, b),
          '→  qos = QOS )
9      time . sleep (1)
10     client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB ', 0 ,
       0 , 0) ,
          '→  qos = QOS )
11     time . sleep (1)
```

The transfer parameters here are the same as those of the const_led(c,r,g,b) function. Setting only the value r=255 causes the LEDs to light up red at maximum brightness. The lay_down() function of the Movement-Control module is formulated in the sniff_bms.py file for the sake of simplicity.

```
31 def lay_down ():
32     client . publish ( CONTROLLER_ACTION_TOPIC , payload =" stand
       Down ")
33     client . publish ( CONTROLLER_ACTION_TOPIC , payload =" damping
       ")
```

The function only sends two commands to the MQTT topic controller/action. The standDown action causes the robot to bend all four legs and thus move the torso close to the ground. The damping action causes the robot to release the tension from the joints and deactivate the motors. As a result, the robot slumps slightly, which is why it should be brought into a lying position beforehand. The entire action of the LED configuration and lying down is short enough that the SoC of the battery does not drop to 3%. As soon as this happens, the motors are switched off and can no longer be used.

The constants of the entire Python environment are stored in the constants.py file.

```
1  HOSTNAME = " 192 . 168 . 12 . 1 "
2  PORT = 1883
3  TIMEOUT = 60
4  FACE_LIGHT_TOPIC = " face_light / color "
5  BMS_STATE_TOPIC = " bms / state "
6  CONTROLLER_ACTION_TOPIC = " controller /
7  action " QOS = 2 " # exactly once
```

Proceed as follows for the final installation of the extension.

```
pi@ raspberrypi :~ $ mkdir ~/ Extensions / && cd ~/
Extensions pi@ raspberrypi :~/ Extensions $ git clone <
Repository Link > pi@ raspberrypi :~/ Extensions $ cd go1 -
bms Monitor /
```

```
pi@ raspberrypi :~/ Extensions / go1 - bms Monitor $ sh install . sh
```

The installation script creates a link in the autostart folder to the directory of the cloned Git repository and configures the scripts so that they are executable. When the Raspberry Pi is restarted, the function is started at the end of the autostart process. The entire source code can be examined more closely in the Git repository created for version management[54].

5.3 Remote video streaming

As shown in Chapter 4.2.6, the Go1 can stream the images from the five built-in cameras via a network. This chapter builds on the findings of chapter 4.2.6 and extends the functions shown there to include the streaming server and a display of the streamed camera image.

Preparation

To prepare for remote video streaming, a streaming server must first be prepared that is accessible via one of the networks connected to the robot. In this chapter, the *Docker* virtualization environment is *used* to simply host the server. A basic knowledge of this is assumed, but the application is kept very simple. Docker has the advantage here that the example shown can be reproduced on most computers. The actual server used is therefore not relevant, as long as Docker can be installed on it.

The open source library collection *bluenvironment* contains an implementation of a media server based on the *GO* programming language with support for various video protocols such as RTSP and WebRTC (Web Real Time Communications). These are also used in the following example. The implementation of the media server is available for use as a Docker image, which can be used with the following command.

```
docker run -- rm - it \
    - e MTX_PROTOCOLS = tcp \
    - p 8554 : 8554 \
    - p 8889 : 8889 \
```

If this command is executed on the external server to be used, the image starts a container with an integrated media server, which is used for streaming via the

[54] Noah Lehmann. *go1-bmsMonitor*. July 1, 2023. URL: https://github.com/noahlehmann/go1-bmsMonitor (visited on August 24, 2023).

RTSP opens port 8554 and port 8889 for the WebRTC protocol and makes it available on the host. In this case, RTSP is used to receive the camera streams from the Go1, while WebRTC is used to send a stream. As soon as the container is started, the media server waits for connections.

Streaming

Streaming the camera images from the robot to the server is similar to the procedure in chapter 4.2.6.

```
ffmpeg - nostdin \                    # No interaction
    - f video 4 linux 2 \             # Input Format
    - i / dev / video 1 \ \           # Input - URL
    - vcodec libx 264 \               # Video encoding ( h264 )
    - preset : v ultrafast            # Encoding speed # No H264
    \                                 B-frames
    - tune zerolatency \              # Frame rate
    - framerate 5 \                   # Output format
    - f rtsp \                        # Output File / URL
    rtsp :// < ip : 8554 | port >/ <
    stream >
```

Now enter the server IP from the previous paragraph in the last line of the command, followed by port 8554 for the RTSP. A term of your choice can be used as the stream name. If several cameras are streamed simultaneously, it is advisable to name them according to the orientation of the camera. For example, the following server URL (Uniform Resource Locator) is suitable for the stream of the Go1's front-facing cameras:

```
rtsp :// 192 . 168 . 123 . 70 : 8554 / head
```

After a successful connection, the following message is displayed on the server.

```
2023 / 08 / 25 10 : 22 : 02 INF [  [ conn 192 . 168 . 123 . 13 : 35428 ]
RTSP ]                             opened
2023 / 08 / 25 10 : 22 : 02 INF [  [ session f 35 fae 28 ] created by
RTSP ]
   ↪ 192 . 168 . 123 . 13 :        [session f 35 fae 28 ] is publishing
35428 2023 / 08 / 25 10 : 22 : 02  to
INF [ RTSP ]                       track ( H264 )
   ↪ path ' head ', with TCP , 1
```

The following message is displayed on the Nano in the head of the robot.

```
Output #0 , rtsp , to ' rtsp :// 192 . 168 . 123 . 51 :
  8554 / head ': Metadata :
    encoder          : Lavf 57 . 83 . 100
```

```
    Stream # 0: 0: Video : h264 ( libx 264 ), yuvj 422 p ( pc), 928 x400 ,
    q
       '→=-1 --1 , 100 fps , 90 k tbn , 100
    tbc Metadata :
      encoder          : Lavc 57 . 107 . 100 libx
    264 Side data :
      cpb : bitrate max / min / avg : 0/ 0/ 0 buffer size : 0 vbv_delay :
-1 frame = 15476 fps =101 q =24. 0 size =N/ A time = 00 : 02 : 34 . 75
bitrate =N/ A dup
   '→= 11652 drop =0 speed =1. 01 x
```

The camera images of the head are successfully transmitted. This procedure can be repeated for all other cameras, only a new stream must be sent to the server under a different name.

Displaying the stream

The WebRTC protocol has the advantage that it is already supported in most modern browsers and does not require any configuration. To display the camera images, simply enter the server URL including the port and the stream name in the browser's address bar. The camera image in the browser is displayed as shown in Figure 29.

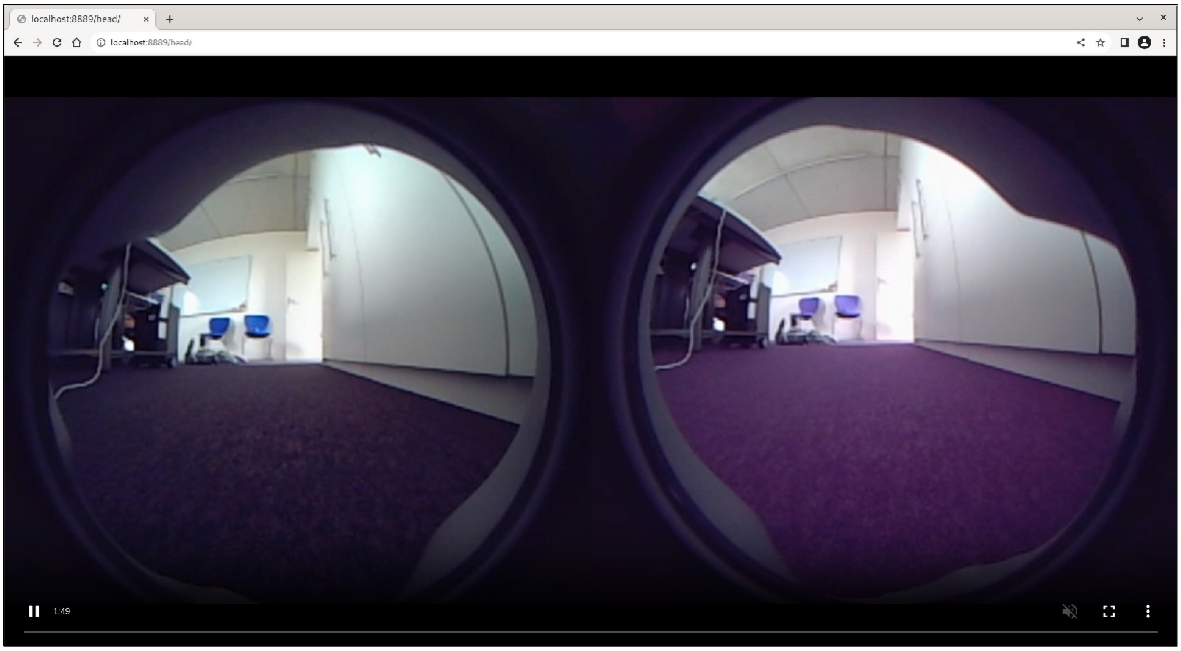

Figure 29: The camera image of the head in the browser view

This example was tested on the *Chromium* browser in version 116.0.5845.96 (Official Build) built on Debian 12.1, running on Debian 12.1 (64-bit).

5.4 Remote control

The final step to fully controlling the Go1 remotely is to remotely control the robot's movements. To do this, it is necessary to check the extent to which the remote control commands from Chapter 5.4 can be replicated via a network. The results from chapters 4.2.5 and 4.2.7 suggest that the movement can be controlled via MQTT.

Analysis of the website

An inspection of the source code of the website shows the following excerpt of the webpack file

:///src/views/Vision.vue.

```
172 const handle Stick Data = ( evt :
173 {    lx: number
174      ly: ;
175      rx: number
176      ry: ; number
177 })  => {  ;
178      const floats = new Float 32 Array (4)
179      ;
180      floats     = evt .
181      [0]        = lx;
182      floats [1] = evt . rx;
183      floats [2] = evt . ry;
184      floats     = evt .
         [3]        ly;
         mqtt . publish (" controller / stick ", floats . buffer , { qos
         : 0 });
};
```

Listing 15: Section of the Vision.vue file

The output of lines 173 to 176 suggests that the two joysticks of the remote control are emulated by specifying *x* and *y* values in order to calculate the position of the stick relative to the zero point - the center. By moving the left stick of the controls displayed on the website to an upper maximum, the following value of the four coordinates can be output by attaching a debugger.

```
evt = { lx: 0 , ly: 1 , rx: 0 , ry: 0}
```

The possible float values of the coordinates can therefore be between -1 and 1. The documentation of the implementation of the JavaScript class Float32Array shows that the array floats in line 178 stores four floats with a length of 32 bits. The floats.buffer function in line 183 writes the four values in binary representation to the message payload of the MQTT topic controller/stick. In order to now manually store the values lx=0, ly=1, rx=0 and ry=0 in the

to the MQTT broker of the Go1, the values must be converted into their binary form and put in the correct order. Lines 179 to 182 show that the correct sequence of the binary coordinates of the controller is lx →rx →ry →ly.

To test the results, a command can be sent to the broker using the mosquitto_pub package. The hexadecimal representation of the values is helpful for this. The hexadecimal representation of the binary value of a float32 with the value 1 is as follows.

```
        Binaer=
00111111100000000000000000000000  Hex= 3 F
800000
```

The command is as follows.

```
echo - ne "\ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00 \ x00
\ x00 \ x00 \ x00 \ x3f \ x80 \
   '→ x00 \ x00 " | mosquitto_pub - h 192 . 168 . 123 . 161 - t controller
   / stick
   '→  - s
```

As the position of the two sticks is permanently configured, the values should be reset to zero as quickly as possible for testing, otherwise the robot in the example above will run forwards permanently.

Structure of the remote control

As the complexity of an entire application for this example is beyond the scope of this paper, only a brief diagram of the components required for a minimal application for remote control of the robot is shown here.

An app server serves as the basis for the application, which provides both a web socket with an MQTT connection and hosts the website, which contains the controller and an optional video stream of the robot's camera images. The website sends the user input to the server via web socket, which converts the input into MQTT messages and sends them to the broker in the Raspberry Pi of the Go1.

Integration of the video stream

HTML5 (Hypertext Markup Language V.5) natively supports the transmission of WebRTC streams in most modern browsers. This has already been shown in section 5.3. The native HTML5 tag <video/> can also embed these streams. All you need to do is set the src attribute with the URL of the stream on the streaming server.

As shown in Figure 30, the web page with the embedded stream can now access the streaming server directly by using the native HTML5 tag <video/> and thus display not only the controls but also the camera images. For the remote control function and integrated video transmission to work, all of the elements shown in Figure

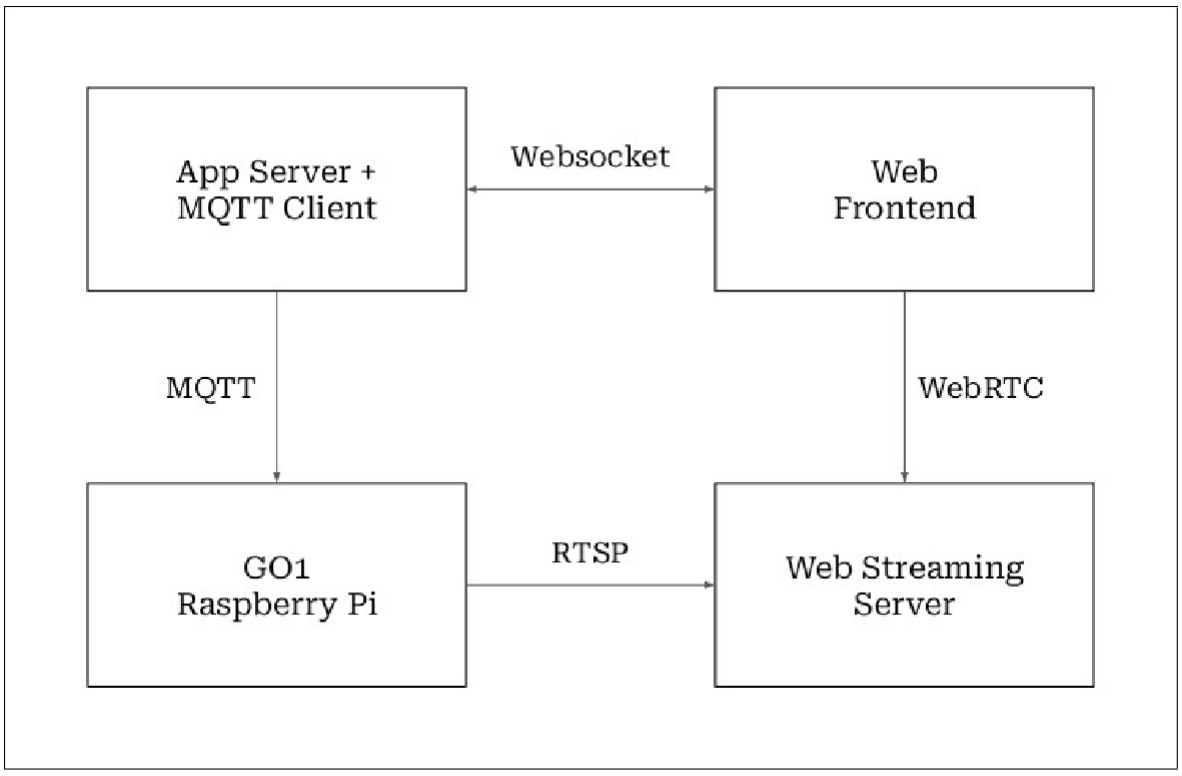

Figure 30: Diagram of the components of an exemplary remote control software

30 shown must be in the same network or be able to communicate with each other across several networks. The preparations from the Wifi and Mobile radio chapters can be helpful for this.

6 Conclusion

This chapter concludes and summarizes the findings of this thesis. At the beginning, the results of this work are summarized and evaluated. Then the approach and the tool in the form of Go1 will be evaluated. At the same time, a conclusion will be drawn on its integration into a university ecosystem. Finally, an outlook will be drawn up, which will outline the potential next steps in the work on the robot.

6.1 Review

The main aim of this work was to provide a basis for working with the Go1 so that it could then be used in a university environment in research, teaching and other areas. For this purpose, the robot was categorized in a simplified field of service robots, whereby it was also explained which other characteristics are applicable. The structure and scope of the Go1 was then documented in detail in the Edu version so that this part can be used as a reference for familiarization with the robot. In the architecture section, the internal components were analyzed in detail. This also includes the exact characteristics of the installed computing units, their functions and, in particular, the communication between the units.

Based on the knowledge gained about the robot, it was possible to start analyzing the functions. After documenting the scope of delivery and the various commissioning options, various basic functions as well as some advanced functions of the Go1 were documented and evaluated. Although many of these functions are advertised by the manufacturer, they are unfortunately not documented, which is why the work at this point can be regarded as detailed and particularly comprehensive alternative documentation. It should be noted here the high level of detail of the investigations for functions that are not or only insufficiently documented or not fully functional. As not all of the robot's functions were tested, a list of the most important unused functions was then drawn up, which also contains references to another work that deals with large parts of these functions. Finally, based on the knowledge of the structure, components and functions of the Go1, a series of new functions and extensions were developed, which generally all pursue the goal of being able to control the robot remotely and thus make it much more flexible to use. A further aim of the functions is to provide the basis for a

autonomous use of the robot by enabling the controlling software to access the robot's data without being directly connected.

Finally, the work is evaluated and an outlook is created, which is developed in the following sections.

6.2 Assessment

Working on the Go1 is usually rewarding, but often laborious. On the one hand, the precision of the components, especially the mechanical components of the motors and joint structures, the quality of the components used, such as the three NVIDIA Jetsons and the Raspberry Pi, as well as the range of functions of parts of the supplied software are impressive, especially in relation to the robot's entry-level price. However, the features and implementation of the Edu model of the Go1 in particular show clear deficits, possibly due to the short development time, which can be estimated at approximately two to three years based on the open-source software used and the announcement of the sale. This model is also considerably more expensive to purchase, which significantly increases initial expectations.

On a positive note, all systems are open for expansion with new functions, the sufficient computing power of the built-in single-board computers and the extensive range of sensors. In particular, the use of open source components such as the operating systems, native libraries and drivers for built-in additional components make it easier to examine and expand the functions that the Go1 already provides ex works.

On the other hand, the inadequate documentation of the robots, especially the unitree libraries, and the concealment of the implementations through the exclusive use of binary code for native functions are negative aspects. This makes it almost impossible t o identify the causes, especially when searching for errors in functions to be tested, without examining the existing software in detail and sometimes resorting to reverse engineering.

A simple integration of Go1 into a university ecosystem is therefore not possible without reasonable effort. The initial documentation of the functions and the testing and evaluation of these must be seen as the basis of this work. In order to be able to use the robot in a real situation at the university, it is then possible to build on the knowledge gained. On the other hand, the robot is a valuable asset for integration into teaching. The testing of the functions and the expansion of these through own developments can be carried out in different levels of knowledge of robotics. For example, this robot can be used with familiar components such as NVIDIA cards and

of the Raspberry Pi provides a solid platform for beginners in the field. However, working more intensively with the device requires a wide range of knowledge, including in the areas of basic computer science, hardware-related programming and programming in general, network technology, physics, electrical engineering, system administration and especially AI. Used correctly, the Go1, as well as similar devices, is a sensible investment in a university and its teaching.

6.3 Outlook

Finally, this work should serve as a reference for future projects on the Go1. The scope of the robot's functions can be developed in several steps, which increase in complexity and build on each other. The aim of the extensions is to be able to use the robot as autonomously as possible, i.e. to assign it a task that it can then perform in all respects without human assistance. The first step is to make the robot permanently accessible to developers and external servers, regardless of its location. For this purpose, the findings from chapters Wifi and 5.1 can be used to connect the robot to a server via VPN, via It is now permanently accessible regardless of the type of connection to the Internet.

In the second step, this connection can be used to implement a type of exchange of logs and current data on telemetry, the location and the BMS between the robot and a permanently installed server. This is not necessarily trivial, as the format of the data, the interval of exchange and the type of data must be defined. In addition, the protocol for communication and the format of the data to be stored must be worked out. Particular attention must be paid to the latency of the transmission, as this often represented a problem in the efficiency of the functions used in the course of this work.

Once the relevant data has been exchanged in an efficient, robust and defined format, work can begin on the manual remote control of the system. Here too, the focus must be on the transmission speed of the data and the reliability of the connection. As part of the remote control, work can also be done on efficient video transmission, as this is absolutely necessary for safe control of the robot without visual contact with it. The use of ultrasonic data can also be considered here.

The final step towards autonomy is to work on the control system. Two scenarios are conceivable here, which are particularly dependent on the findings on the quality of the network connection from the previous steps. If the data transmission is unreliable,

the transmission volume should be reduced and the robot should be controlled via embedded software. The focus here should be on the efficiency of the implementation, as the robot's computing units can only provide limited performance. If the transmission rate is sufficient, the control of the robot can be outsourced to an external system. The focus here is on the transfer of data.

In conclusion, the Go1 can be explored for potential uses based on the findings of this thesis and the possible extensions to make the transition from academic use to commercial use that potentially benefits society, which is the ultimate goal of all academic work, including this thesis entitled *Integration of a Unitree Go1 Quadruped Robot into a University Ecosystem*.

Appendix A Listings

/home/pi/UnitreeUpgrade/start.sh

```
#!/ bin / bash
cd / home / pi/ Unitree Upgrade
python 3 ./ startup_manager .
py&

python3 ./ startup_manager .
py&
./ update . sh &
```

/home/pi/autostart/Unitree/configNetwork/configNetwork.sh

```
#!/ bin / bash
eval echo "[ config Network ] starting ... " $to
Startlog # configure 4 G
sudo ifconfig wlan0 down
sudo ifconfig wlan1 down
sudo ifconfig wlan2 down
sudo ifconfig wwan0 down

_EC25 =$( lsusb | sed - n '/ EC/ P' | wc -
c)
if [[ _EC25 - gt 10 ]]; then                detected " $to
  eval echo "[ config Network ] 4 G         Startlog
  module sudo ifconfig eth0 down
  sudo ifconfig eth1 down sudo
  ./ ppp / quectel - pppd . sh&
  _ppp 0 Count
  =0 sleep 1
  while [[ $_ppp 0 Count - lt 6
  ]] do
    sleep 1 (( _ppp 0
    Count ++) )                             for ppp0 " $to
    eval echo "[ config Network ] waiting Startlog
    _ppp0 =' ifconfig | grep ' ppp0 ' | wc-c'
    sleep 1
    if [[ $_ppp 0 - gt 10 ]]; then
      eval echo "[ config Network ] ppp0 obtained " $to
      Startlog break
    fi
    sleep 1
  done
  sleep 1
  sudo ifconfig eth0 up
fi sudo ifconfig eth1 up

# configure WIFI
sudo ifconfig wlan1 192 . 168 . 12 . 1 / 24
sudo ifconfig wlan1 up
sudo hostapd / etc / hostapd / hostapd . conf &
eval echo "[ config Network ] config WIFI " $to
     Startlog
```

```
sleep 5

sudo ./ connect Wifi 2 . sh&
```

hostapd.conf

```
interface =
wlan1
driver = nl
80211 hw_mode
=a ieee 80211 n
=1 ieee 80211
ac =1 ieee
80211 d =1 ieee
80211 h =1
require_ht =1
require_vht =1
wmm_enabled =1
vht_oper_chwidth =1
channel =165
vht_oper_centr_freq_seg 0 _idx
=155
ht_capab =[ HT40 -][ HT40 +][ SHORT - GI - 40][
DSSS_CCK -40]

wpa =2 wpa_key_mgmt
=WPA - PSK
rsn_pairwise = CCMP
ssid = Unitree_Go 501075 A
wpa_passphrase = 00000000
```

webpack://../bmyReceivers.ts

```
// webpack :/// src / plugins / mqtt / receivers / bms
Receivers . ts
import { Mqtt Data } from "@/ plugins / mqtt / data ";
import { Bms Sub Topic } from "@/ plugins / mqtt / topics
";

type Receivers = {
    [ key in Bms Sub Topic ]:
        ( data : MqttData ,
        message : Buffer ,
        data View : Data
        View
    ) => void ;
};

const receivers : Receivers = {
    " bms / state ": ( data , message , data View ) =>
        { const uint 8 s = new Uint 8 Array ( message
        );
        data . bms . version = uint 8 s [0] + "." + uint 8
        s [1]; data . bms . status = uint 8 s [2];
        data . bms . soc = uint 8 s [3];
        data . bms . current = data View . get Int 32 (4 ,
        true ); data . bms . cycle = data View . get Uint
        16 (8 , true );
```

```
22          data . bms . temps = [ uint 8 s [10] , uint 8 s [11] , uint 8 s
            [12] ,
23              '→uint 8 s [ 13 ]];
24          for ( let i = 0; i < 10; i ++) {
                data . bms . cell Voltages [ i] = data View . get Uint 16
25              (14 + i
26                  '→* 2 , true );
            }
27      }   data . bms . voltage = data . bms . cell Voltages . reduce ((
28  };      a, c) =>
29              '→  a + c);
30  export default receivers ;
```

/home/pi/autostart/Unitree/configNetwork/connectWlan2.sh

```
1  #!/ bin / bash
2  # / home / pi/ Unitree / autostart / config Network / connect
   Wlan 2 . sh
3  sleep 1
4  echo "[ connect Wlan 2 ] Start initializing wlan2 interface "
5      '→$to start log
   sudo ifconfig wlan2 up
6  sudo rm / var / run / wpa_supplicant / wlan2
7  echo "[ connect Wlan 2 ] Connecting wlan2 interface " $to Startlog
8  sudo wpa_supplicant - B - i wlan2 - c / etc / wpa_supplicant /
9  config . d/
10     '→wpa_wlan 2 . conf
   echo "[ connect Wlan 2 ] Done connecting wlan2 interface via
11     '→wpa_supplicant " $to Startlog
   sleep 3
```

/home/pi/autostart/Unitree/configNetwork/connectWwan0.sh

```
1  #!/ bin / bash
2  # / home / pi/ Unitree / autostart / config Network / connect
   Wwan 0 . sh
3  sleep 1
4  echo "[ connect Wwan 0 ] Preparing environment for wwan0 interface "
5      '→$to start log
   sudo systemctl stop Modem Manager
6  echo "[ connect Wwan 0 ] Start initializing wwan0 interface "
7      '→$to start log
8  sudo ifconfig wwan2 up
9  sudo qmicli - p - d / dev / cdc - wdm0 -- device - open - net =' net -
   raw - ip|net
       '→- no - qos - header ' -- wds - start - network =" apn =' internet
10     ', ip - type
11     '→=4" -- client - no - release - cid
12 echo "[ connect Wwan 0 ] Getting IP for wwan0 interface " $to
13 Startlog sudo udhcpc - q - f - i wwan0
   echo "[ connect Wwan 0 ] Connected wwan0 to internet " $to Startlog
   sleep 3
```

/home/pi/Extensions/go1-bmsMonitor/install.sh

```
#!/ bin /
bash
DIR = "/ home / pi/ Unitree / autostart /"
STARTUP_FILE =". startlist . sh"
SCRIPT_DIR =$( cd -- " $( dirname -- " ${ BASH_SOURCE [    )" & > /
0]}"                                                          dev
    '-> / null && pwd )
if ! command - v python & > / dev /
null
then
  echo  " python could not be found , exiting ..."
  exit  1
fi

if ! [  - d " $DIR "
then    ];
  echo  " $DIR was not found , exiting ..."
  exit  1
fi

if [ - f " $STARTUP_FILE " ];
then
  echo " $STARTUP_FILE was not found , exiting
  ..." exit 1
fi

if grep - q " bms Monitor " " $DIR$STARTUP_FILE "
; then
  echo " bms Monitor already in $STARTUP_FILE ";
else
  echo " bms Monitor " >> " $DIR$STARTUP_FILE "
fi


# Delete folder
rm - r " $DIR / bms Monitor " & > / dev / null
# Create symbolic links to the repository cp -
rs " $SCRIPT_DIR / bms Monitor " " $DIR "
# Copy script to get paths
cp - f " $SCRIPT_DIR / bms Monitor / bms Monitor . template .
sh" " $DIR /
    '-> bms Monitor / bms Monitor .
sh" # Make $DIR/bms Monitor executable
chmod +x " $DIR / bms Monitor / bms Monitor . sh" " $SCRIPT_DIR / bms
Monitor /
    '-> run . sh"
```

/home/pi/Extensions/go1-bmsMonitor/bmsMonitor/run.sh

```
#!/ bin / bash
my Now =$( date +"% T")
eval echo " $my Now [ diagnosis ] starting ... " $to Startlog
sleep 30
my Now =$( date +"% T")
eval echo " $my Now [ diagnosis ] starting the monitor " $to Startlog

```

```
python 3 - m pip install - r / home / pi/ unitree / autostart / bms monitor /
    '→ requirements . txt
python 3 / home / pi/ unitree / autostart / bms monitor / snif_bms . py
```

/home/pi/Extensions/go1-bmsMonitor/bmsMonitor/sniff_bms.py

```
import struct

import paho . mqtt . client as
mqtt
from constants import HOSTNAME , PORT , TIMEOUT , BMS_STATE_TOPIC
,
    '→ CONTROLLER_ACTION_TOPIC
from led_control import alert_led , const_led

def on_connect ( c, userdata , flags ,
    rc):
    c. subscribe ( BMS_STATE_TOPIC )

def on_message ( c, userdata , msg ):
    print (" message ")
    [ ver0 , ver1 , status , soc , current , cycle , temp0 ,
     temp1 , temp2 , temp3 , cell_v0 , cell_v1 , cell_v2 ,
     cell_v3 , cell_v4 , cell_v5 , cell_v6 , cell_v7 , cell_v8 ,
     cell_v 9 ] \
        = struct . unpack (' BBBBi HBBBBBBHHHHHHHHHHHH ', msg
    . payload ) if not bms_responding ( msg . payload ):
        10 ,
    elif 5 >= soc : alert_led
        ( c, 255 , 0 ,
        lay_down ()

def bms_responding ( payload
    ):
    # if payload only zeros
    return 0 != int . from_bytes ( payload , byteorder ='
    big ')

def lay_down ():
    client . publish ( CONTROLLER_ACTION_TOPIC , payload =" stand
    Down ") client . publish ( CONTROLLER_ACTION_TOPIC , payload ="
    damping ")

client = mqtt . Client (" Stick Data ")
client . on_connect =
on_connect client . on_message
= on_message

client . connect ( HOSTNAME , PORT ,

TIMEOUT ) try :
    client . loop_forever ()
except :
    client . disconnect ()
```

/home/pi/Extensions/go1-bmsMonitor/bmsMonitor/led_control.py

```
import struct
import time

from constants import FACE_LIGHT_TOPIC , QOS


def alert_led ( client , r, g, b):
 client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB r, g, b),
                                                        ',
        '→  qos = QOS )
    time . sleep (1)
 client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB 0 , 0 , 0) ,
                                                        ',
        '→  qos = QOS )
    time . sleep (1)


def const_led ( client , r, g, b):
 client . publish ( FACE_LIGHT_TOPIC , struct . pack (' BBB r, g, b),
                                                        ',
        '→  qos = QOS )
    time . sleep (1)
```

/home/pi/Extensions/go1-bmsMonitor/bmsMonitor/constants.py

```
HOSTNAME = " 192 . 168 . 12 . 1 "
PORT = 1883
TIMEOUT = 60
FACE_LIGHT_TOPIC = " face_light / color "
BMS_STATE_TOPIC = " bms / state "
CONTROLLER_ACTION_TOPIC = " controller /
action "
QOS = 2  # exactly once
```

Excerpt from webpack:///src/views/Vision.vue

```
const  handle Stick Data = ( evt : {
    lx: number ;
    ly: number ;
    rx: number ;
    ry: number ;
 }) => {
    const  floats = new  Float 32 Array (4) ;
    floats [0] = evt . lx;
    floats [1] = evt . rx;
    floats [2] = evt . ry;
    floats [3] = evt . ly;
    mqtt . publish (" controller / stick ", floats .    { qos : 0 });
       buffer ,
};
```

Literature

[1] Evan Ackerman. "Boston Dynamics' Spot Robot Dog Now Available for $74,500. For the price of a luxury car, you can now get a very smart, very capable, very yellow robotic dog". In: *IEEE Spectrum* (June 16, 2020).

[2] Defense Advanced Research Projects Agency. *Big Dog*. URL: urhttps://www.darpa.mil/about-us/timeline/big-dog (visited on 07. 08. 2023).

[3] Defense Advanced Research Projects Agency. *Maximum Mobility and Manipulation (M3)*. URL: https://www.darpa.mil/program/maximum-mobility-and-manipulation (visited on 07. 08. 2023).

[4] Alsa-Project.org. *Advanced Linux Sound Architecture (ALSA) project homepage*. Oct. 20, 2020. URL: https://www.alsa-project.org/wiki/Main_Page (visited on Aug. 18, 2023).

[5] Gerardo Bledt et al. "MIT Cheetah 3: Design and Control of a Robust, Dynamic Qua- druped Robot". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2245-2252. DOI: 10.1109/IROS.2018.8593885.

[6] MDN contributors. *DataView*. Aug. 21, 2013. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/DataView (visited on Aug. 22, 2023).

[7] MDN contributors. *Uint8Array*. Aug. 14, 2013. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Uint8Array (visited on 22. 08. 2023).

[8] NVIDIA Corporation. *Developer Kit and Modules for Embedded Systems*. 2023. URL: urhttps://www.nvidia.com/de-de/autonomous-machines/embedded-system s/.

[9] *Development and use of Go1 binocular fisheye camera*. Unitree Robotics.

[10] *Development and use of Go1 ultrasonic module*. Unitree Robotics.

[11] Boston Dynamics. *Photo of a BigDog*. URL: https://www.bostondynamics.com/wp-content/uploads/2023/07/bigdog-1-1.jpg (visited on 28. 08. 2023).

[12] Boston Dynamics. *Photo of a spot*. URL: https://bostondynamics.com/wp-content/uploads/2023/05/spot-explorer-web-2-2048x1152.jpg (visited on 28. 08. 2023).

[13] Boston Dynamics. *Legacy Robots*. URL: https://bostondynamics.com/legacy/ (visited on 08. 08. 2023).

[14] Boston Dynamics. "Robotics' Role in Public Safety. How robots like Boston Dynamics' Spot are keeping people safe." In: (2023).

[15] Jason Falconer. "MIT Cheetah Robot Runs Fast, and Efficiently. It's now the second fastest legged robot in the world". In: *IEEE Spectrum* (May 14, 2013).

[16] Sophie Fischer. *Robotics - Market data analysis & forecasts*. Statista Technology Market Outlook. Statista, Aug. 2022. URL: https://de.statista.com/statistik/study/id/116785/document/robotics-report/ (visited on 04. 07. 2023).

[17] FreeDesktop.org. *Autostart Of Applications During Startup*. URL: https://specifications.freedesktop.org/autostart-spec/0.5/ar01s02.html (visited on 17. 08. 2023).

[18] Srijeet Halder et al. "Real-Time and Remote Construction Progress Monitoring with a Quadruped Robot Using Augmented Reality". In: *Buildings* (Dec. 2022). URL: https://doi.org/10.3390/buildings12112027.

[19] Devan Joseph. "MIT reveals how its military-funded Cheetah robot can now jump over obstacles on its own". In: *Business Insider* (June 3, 2015).

[20] Jonas Kemnitzer. "Development of an intelligent lidar-based 3D navigation system for the Unitree GO1". Master thesis. Hof University of Applied Sciences, 2023.

[21] Noah Lehmann. *go1-bmsMonitor*. July 1, 2023. URL: https://github.com/noahlehmann/go1-bmsMonitor (visited on 24. 08. 2023).

[22] Jouni Malinen. *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. Jan. 12, 2013. URL: https://w1.fi/hostapd/ (visited Aug. 18, 2023).

[23] Lee Milburn, Juan Gamba and Claudio Semini. *Towards Computer-Vision Based Vineyard Navigation for Quadruped Robots*. Dynamic Legged Systems Lab - Istituto Italiano di Tecnologia Genova, Italy, Jan. 2, 2023.

[24] MIT. *Photo of a Cheetah 3* . URL: https://news.mit.edu/sites/default/files/styles/news_article image_gallery/public/images/201903/MIT-Mini-Cheetah-01_0.jpg (visited on 28. 08. 2023).

[25] Jeremy Moses and Geoffrey Ford. *The Rise of the Robot Quadrupeds*. Dec. 11, 2020. URL: https://mappinglaws.net/rise-robot-quadrupeds.html (visited on 07. 08. 2023).

[26] VDI Society for Production Engineering. *VDI 2860/ Assembly and handling technology. Handling functions, handling equipment; terms, definitions, symbols*. Beuthe publishing house, 1990.

[27] Marc Raibert et al. *BigDog, the Rough-Terrain Quaduped Robot*. Boston Dynamics, Apr. 8, 2008.

[28] Unitree Robotics. *About Unitree Robotics*. URL: https://www.unitree.com/en/ab out/ (visited on 28. 08. 2023).

[29] Unitree Robotics. *GO-M8010-6 Motor - Unitree Robotics*. URL: https://shop.unit ree.com/products/go1-motor (visited 07. 07. 2023).

[30] Unitree Robotics. *Go1 APP Download*. URL: uhttps://www.unitree.com/app/ (visited on 17. 08. 2023).

[31] Unitree Robotics. *GO1 Manuals - GO1 Tutorials 1.0.0 documentation*. URL: https://www.docs.quadruped.de/projects/go1/html/index.html (visited on 08. 08. 2023).

[32] Unitree Robotics. *Unitree Go1*. URL: https://shop.unitree.com/cdn/shop/products/75_540x.jpg?v=1668073774 (visited Aug. 28, 2023).

[33] Spectrum. *Encyclopedia of Biology*. Spektrum Akademischer Verlag Heidelberg, 1999.

Explanation

I hereby declare that I have written this thesis independently and without the use of aids other than those specified; ideas taken directly or indirectly from external sources are identified as such. To the best of my knowledge, this thesis has not been submitted in the same or a similar form to any other examination authority and has not yet been published.

Hof, September 15, 2023

Noah Lehmann

Abstract

The thesis *Integration of a Unitree Go1 quadruped robot into a university ecosystem* deals with the intensive functional analysis of the *Unitree Robotics Go1 Edu* and the classification of its potential application in a university environment. After a thematic classification of the work, the basics for the classification and categorization of the Go1 in the field of robotics are developed.

The structure of the robot is described and documented in detail below. First, the external shape is described and the components are named. The internally installed components are then documented in detail as a further reference. Communication and the structure of the internal network are described based on the structure of the internals.

In the following analysis section, the use is documented by first recording the scope of delivery and the external accessories, whereupon two forms of commissioning are shown, the intended battery-powered commissioning and the commissioning of the robot with an external power supply. The scope of functions of the mobile applications supplied with the Go1 are described.

The following section describes the functional scope of the robot itself, after which extensions to the scope are developed with the aim of autonomous use. The functions described are the software autostart function, remote control, the local network, the use of the audio interface, control of the head lighting, transmission of camera images and battery management.

Finally, the autonomous use of the Go1 is prepared in four steps by expanding the range of functions. First, the robot's connectivity is expanded and made more fail-safe. For this purpose, the local connection of the robot with WLAN access points is configured and the use of a mobile radio card for mobile connection is explained and worked out in detail. A weakness of the robot in use is remedied by extending the battery management system to include safe protection of the device when the battery is low. In preparation for remote control of the robot, a video streaming option is shown using common protocols, after which the remote control itself is implemented. This is based on the stable connection of the robot via a network and builds on the findings from the analysis chapters.

Finally, the Unitree Robotics Go1 Edu is evaluated and an extension of the robot is presented based on the findings documented in the thesis. The extensions to be developed in several steps and individual projects combine to create an autonomously functioning system for which ultimately only sensible applications need to be researched.