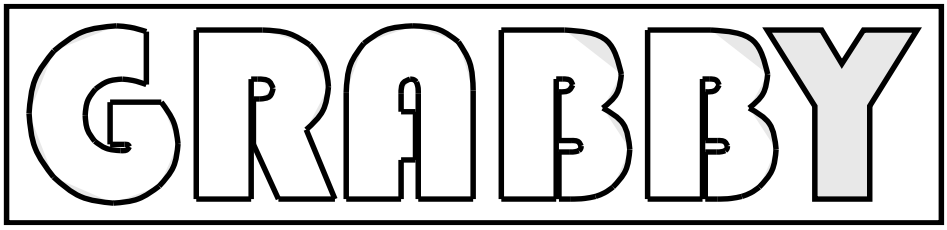
R&D Project



X

Final Project Report

Grabby

Students	Jeremy Hendrikse Julius Ortstadt
Supervisors	M. Frédéric JUAN M. Frédéric RALLO M. Sébastien ROTHHUT
Year	2025-2026
Class	ROBO5

Table of contents

Special thanks 3
1/ Introduction 4
1.1/ Overview 4
1.2/ GitHub 5
1.3/ Objectives 5
2/ Hardware development 6
2.1/ Chassis 6
2.2/ Computing 7
2.3/ Modular top interface 7
2.4/ Drive system 8
2.5/ Power system 8
3/ Software development 10
3.1/ Operating System 10
3.2/ ROS Distribution 10
3.3/ Mapping 11
3.4/ Navigation 12
3.5/ Computer vision 13
3.5.1/ Depth Map 13
3.5.2/ QR Code detection 14
3.5.3/ Triangulation 14
4/ Documentation 15
5/ Future improvement and changes 15
6/ Conclusion 16
6.1/ Costs 16
6.2/ Project 17

Table of Figures

Figure 1 - Grabby V2 4
Figure 2 - New chassis (V4) 6
Figure 3 - Disassembled drive system 8
Figure 4 - Drive system control deck 8
Figure 5 - Power schematic 9
Figure 6 - Power panel 9
Figure 7 - Mapping procedure 12
Figure 8 - Autonomous navigation 13
Figure 9 - Depth Map 14
Figure 10 - Real-Life Scene of Figure 9 14

Special thanks

We would like to sincerely thank M. Frédéric Juan for his continued help and support throughout the development of this project. His expertise, insights, and technical advice were key to the successful completion of the work.

We also extend a special thank you to M. Guillaume Ducard, whose lectures on sensor fusion played a central role in both the understanding and implementation of the project's localization, sensor fusion, and SLAM algorithms.

We thank M. Pascal Masson for his guidance and electronics lectures in ROBO3, which laid the technical foundation for the project.

Additionally, we would like to thank M. Frédéric Rallo and M. Sébastien Rothhut for their support and advice at various stages of the project. Their feedback often encouraged us to reassess design choices and improve our overall approach.

Finally, we would like to thank Polytech Nice-Sophia for making this project possible by providing the necessary funding, equipment and academic environment.

1/ Introduction

1.1/ Overview

Grabby is designed as an autonomous warehouse robot prototype intended as a proof of concept and a research platform for investigating cost-effective and efficient automation in warehouse environments. Rather than targeting immediate industrial deployment, the system serves as a foundational basis for exploring how autonomous mobile robots could be applied to logistical tasks under realistic constraints.

To support this objective, the robot integrates a range of sensors and software algorithms that enable autonomous navigation and task execution within structured indoor environments. The platform allows for experimentation with perception, localization, and motion planning approaches relevant to warehouse automation. Insights gained from this work may contribute to reducing logistical errors and improving operational efficiency in future warehouse systems.

The robot architecture is composed of two primary subsystems. The first subsystem is the mobile base, which houses the onboard computer, drive motors, power distribution system, control electronics, and sensors required for locomotion and environmental perception.

The second subsystem is the lifting mechanism, which incorporates a vertical lifting and grabbing system designed to manipulate packages at varying heights. A stereo camera mounted at the top of the structure supports navigation-related tasks and provides visual data for package localization and experimental perception pipelines.

A key design consideration of the Grabby platform is modularity. The robot is structured such that the upper subsystem can be replaced or reconfigured with minimal impact on the mobile base. Communication between the upper and lower sections is limited primarily to standardized data (USB) and power interfaces (XT60), allowing alternative payloads, sensor configurations, or manipulation systems to be integrated without requiring significant changes.

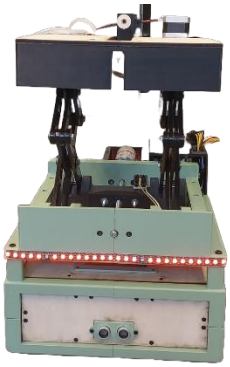


Figure 1 - Grabby V2

1.2/ GitHub

The GitHub repository for this project can be found with the following link:

https://github.com/JeremyHendr/GrabbyBotics

This repository holds all the necessary files and information to properly hand over or continue the project.

Note that all steps for setting up the RaspberryPi as well as the development environment are provided in their respective markdown files present in the repository.

Also, CAD files are provided in either or both .3mf and fusion files. Additionally, the code and the relevant dependencies are also available in the corresponding directories.

1.3/ Objectives

This year's objectives focus on solving problems which were encountered last year but also on implementing the necessary changes to turn Grabby into a more operational prototype.

To summarize, the goals were:

- Upgrading to a RaspberryPi 5[1]
- Designing custom PCBs
- Implementing a modular system architecture based on a plug-and-play approach
- Increasing the physical robustness of the robot
- Installing a 9-DoF IMU
- Redesigning the power system
- Optimizing the drive system

Additional goals set for this year were:

- Achieving reliable and optimized mapping
- Achieving reliable and optimized autonomous navigation
- Improving the overview of the system
- Creating detailed documentation of the robot
- Achieving accurate depth map using a stereo camera
- Localization and positioning of QR code in space

Multiple of these goals were achieved as will be detailed in this report.

[1] From here on the RaspberryPi 5 will be referred to as Pi or RaspberryPi 5.

2/ Hardware development

2.1/ Chassis

The chassis and frame were fully redesigned to improve structural integrity, accessibility, and modularity while maintaining a similar overall footprint to the previous version of the robot. The new design is based on 20*20mm aluminum profiles and carbon-reinforced 3D-printed components, providing a rigid structure while allowing flexible mounting options. This, however, led to the total weight of the robot increasing.

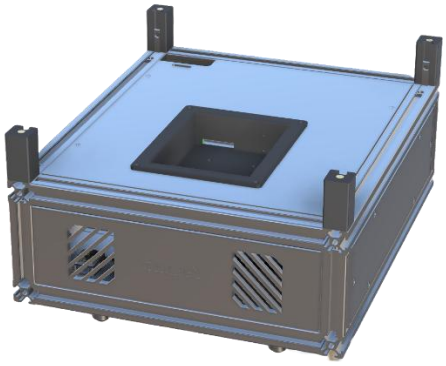


Figure 2 - New chassis (V4)

The architecture emphasizes ease of assembly and disassembly. Each component is mounted using dedicated 3D-printed adapter brackets rather than being fixed directly to the structural plates. This approach allows the mounting hole patterns in the plates to remain standardized, while only the adapter needs to be redesigned when a component is changed or replaced.

A flush outer wall design increases usable internal mounting space and simplifies the integration of additional components. Cable management was also significantly improved through the introduction of double-sided internal floors. This separation allows power and data cables to be routed independently, resulting in a cleaner internal layout and reducing the risk of interference. To further improve usability, a magnetically mounted quick-access panel in the front was added to allow fast access to the battery.

Dedicated ventilation openings combined with a rear-mounted fan create a controlled airflow path from back to front, directing air over critical components such as the Raspberry Pi and the battery to prevent high temperatures and thermal throttling.

2.2/ Computing

The main computing platform was upgraded from an NVIDIA Jetson Nano to a Raspberry Pi 5. The Jetson was originally used as part of the lectures given during the robotics classes which presented the advanced GPU capabilities. However, these capabilities were not required for the intended navigation and control workloads. The Pi offers significantly improved CPU performance, a large variety of expansion boards, better long-term software support and broader ecosystem compatibility while taking up less space and consuming less power; a critical consideration when building an autonomous robot. In addition, the platform allows future integration of an AI HAT if hardware acceleration becomes necessary.

Following this change, the microcontroller architecture was also revised. Instead of relying on a single Arduino Mega to handle all low-level tasks from both the mobile base and the upper subsystem, multiple dedicated Arduino Nano boards are used, each responsible for a specific task in the robot. This distributed approach improves modularity, simplifies debugging and reduces interdependencies between subsystems. The Nano was chosen due to its compact size, low power consumption, and adequate performance. However, as will be seen later it also came with limitations regarding the software development of the robot.

2.3/ Modular top interface

The redesigned top interface focused on modularity. The structural pillars that mount the two systems together are removable and can be replaced if design changes are required. The use of standardized aluminum profiles ensures that new components can be designed and integrated without mechanical constraints.

The communication and power interface was also simplified. The previous version had all the data lines going directly to the Arduino Mega in the mobile base and relied on bulky MOLEX connectors. The new version uses two XT60 connectors for 12V and 5V power, sharing a common GND, along with a single USB connection to the Pi.

This USB link allows the heavy processing to be made on the Pi, while the basic tasks are outsourced to a small microcontroller placed on the other subsystem. As a result, system complexity and interdependency are reduced, and the total number of interconnecting cables has been reduced from approximately twenty to three.

2.4/ Drive system

In the previous drive system, the motors were mounted directly onto the robot's base plate, which also supported the load of all other components. This design proved suboptimal, as continuous mechanical stress and heat exposure led to material fatigue and deformation over time. The redesigned system leverages the increased structural strength provided by the aluminum profiles and carbon-based 3D printed components.

The updated drive module is fully modular and can be removed together with its motor drivers and controllers using only six screws. This modularity simplifies maintenance and allows the motors to be easily replaced with alternative variants if required. The new layout also results in a significantly cleaner integration, as only power and USB connections are routed between the drive unit and the rest of the robot.

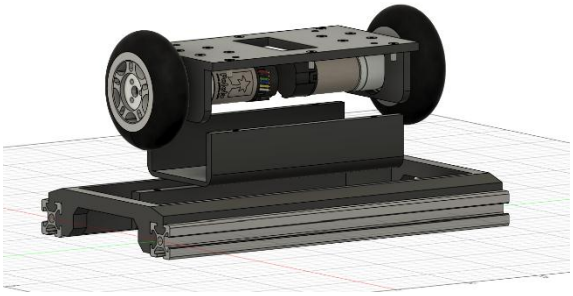


Figure 3 - Disassembled drive system

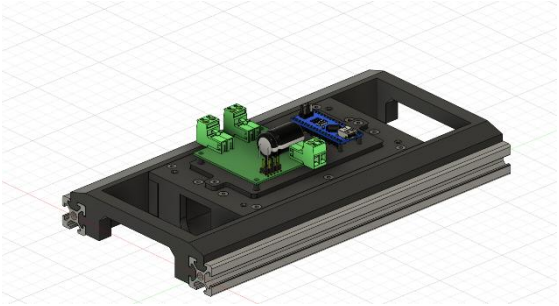


Figure 4 - Drive system control deck

Stability was further improved by redesigning the caster wheel configuration. The previous design used two centrally mounted caster wheels at the front and rear, which caused instability near the edges of the robot. The new system incorporates four caster wheels, one at each corner, each mounted with a TPU-based suspension. This configuration improves overall stability and reduces system rigidity, enabling the robot to better handle minor surface irregularities. This addresses issues observed in earlier versions, where uneven terrain could cause the robot to become immobilized.

2.5/ Power system

The previous version of the robot supported only an external power supply via a barrel jack connector. The upgraded design adds support for both an integrated LiPo battery and an external power source. The battery, however, must be removed from the robot for charging.

The introduction of multiple power sources required a redesign of the power distribution system. By using a double-layer floor structure, power cables are largely separated from data lines, resulting in cleaner routing and simplified power

management. The onboard battery provides 12V for the drive system, which is stepped down using two buck converters: one supplying 5V@3A via USB-C for the Pi, and a second delivering up to 5V@10A for auxiliary electronics like sensors. This separation increases system robustness by electrically isolating the main computing unit from other subsystems, improving resilience against short circuits and power faults.

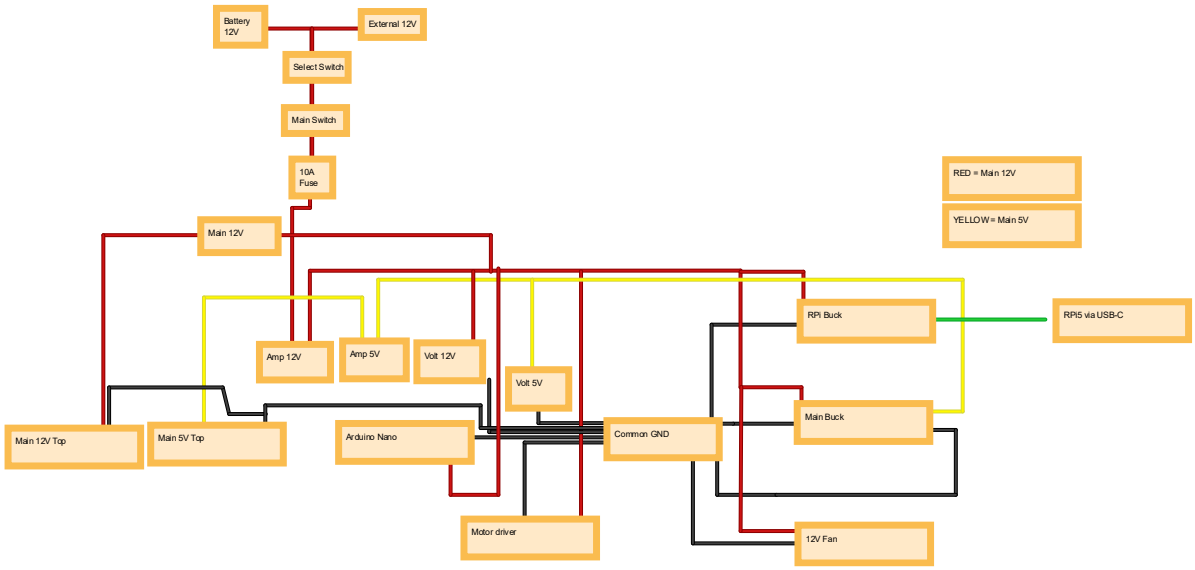


Figure 5 - Power schematic

Also, to monitor all the power statistics, the rear of the robot features current and voltage sensors and a small display managed using a dedicated pre-programmed Arduino Nano.

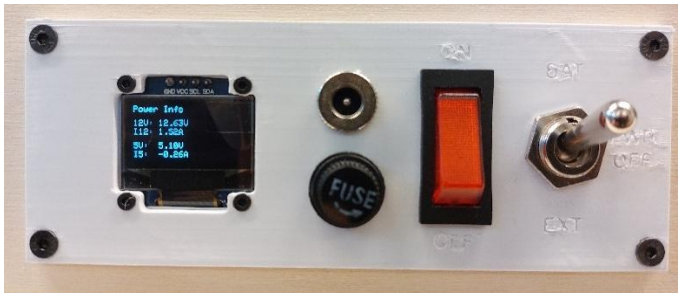


Figure 6 - Power panel

3/ Software development

3.1/ Operating System

The mobile base is built around a Raspberry Pi 5, which serves as the main onboard computing unit. This platform was selected to provide sufficient computational performance while maintaining a compact form factor and low power consumption suitable for mobile robotics applications.

A new software stack was implemented on the mobile base, based on Ubuntu Server 24.04 LTS. To facilitate development and operation, a lightweight desktop environment was added alongside comprehensive system configuration, as detailed in the accompanying documentation. The main advantage of using Ubuntu server with a later installed desktop environment is that it is lightweight but it can also be directly configured as headless during imaging of the SD card; something that is not possible with the Ubuntu Desktop. The latter requires a monitor to be connected during first boot. This wasn’t an option as the Pi comes equipped with mini-HDMI making the whole process more complicated.

Furthermore, remote access to the system is enabled through NoMachine and a standard SSH connection, allowing reliable interaction with the robot over both Ethernet and Wi-Fi connections. This setup supports headless operation and simplifies debugging, monitoring, and software deployment without requiring direct physical access to the robot. Here, another advantage of the Pi can be seen, namely that the integrated Wi-Fi is way more reliable than the one on the NVIDIA Jetson Nano which offered a lower bandwidth and had a more complicated configuration and often failed making prototyping unnecessarily difficult at times.

3.2/ ROS Distribution

ROS1 is no longer officially supported on Ubuntu 24.04 LTS, which introduced significant compatibility constraints for the software stack. To solve this problem, development and deployment were conducted within a Docker container. This approach ensured software reproducibility and allowed legacy ROS1 components to run on a modern operating system. It was, however, challenging to find a complete desktop build of ROS1 Melodic for the ARM64 architecture. Eventually a functional base image was found, but it required some additional configuration to make it ready for implementation on the robot.

An evaluation of ROS2 was also performed; however, several critical limitations made the decision easier to stay on ROS1. The current implementation uses rosserial for communication between the main computer (previously NVIDIA Jetson Nano) and the Arduino. This communication was easy to set up and allowed the Arduino to directly publish on topics within the running ROS environment. Under

ROS2, rosserial has been replaced by Micro-ROS which isn't supported by the Arduino. This created the need to create custom serial communication between the Pi and the Arduino under ROS2 to publish and receive the necessary data; a very complicated implementation prone to errors and instability.

Furthermore, many mature SLAM and navigation packages commonly used in ROS1, such as Cartographer or HECTOR SLAM, are no longer directly supported or maintained within the ROS2 ecosystem.

Finally, native ROS 2 desktop distributions for ARM64 on the selected Ubuntu version remain limited. This would have required the use of containerized deployments similar to ROS 1. While ROS 2 offers long-term advantages, these constraints made ROS 1 the more viable option for the current implementation, at least to test the new hardware and obtain a working prototype.

3.3/ Mapping

The teleoperation system for mapping was reworked to improve reliability and usability. The original approach meant using a SSH connection from a mobile device to control the robot using the keyboard keys to move the robot. However, since the Pi has integrated Bluetooth, it was now possible to use a Bluetooth controller, like the DualShock 4, to move the robot. This was achieved using the joystick package and a custom code to translate the controller readings to velocity commands later used by the Arduino.

The mapping pipeline was also revised to enhance localization accuracy. This included the integration of an improved 9-DoF IMU sensor and refined state estimation through more appropriate configuration of the mapping and sensor fusion parameters, leading to more consistent and reliable map generation. The previously used algorithm, Google Cartographer, was kept as it offers robust and reliable mapping capabilities with a wide range of parameters for configuration. This includes the data sources: LiDAR, IMU, wheel odometry and visual odometry. Having most of these components installed on the robot, it was possible to tune the algorithm to use the best combination of those to achieve the best performance.

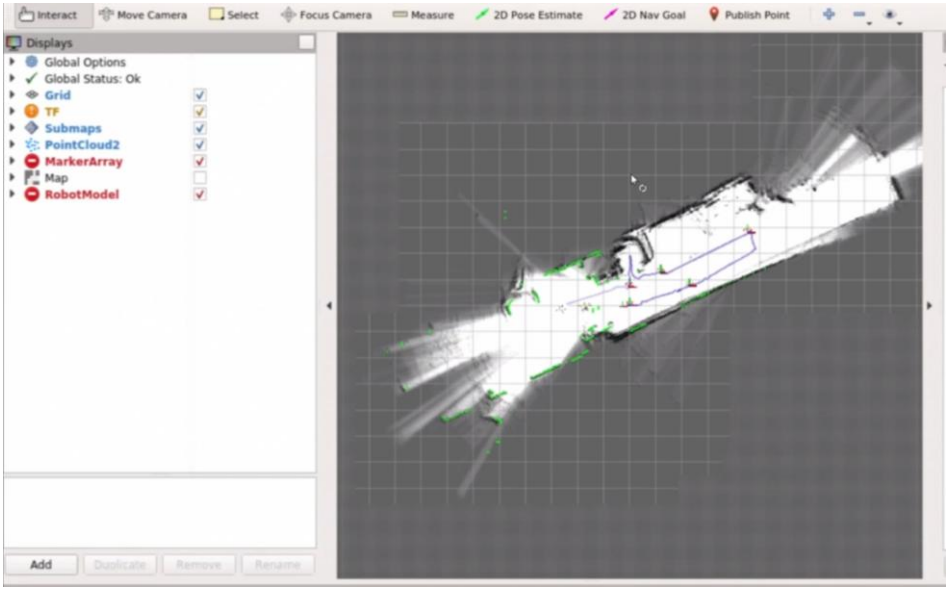


Figure 7 - Mapping procedure

3.4/ Navigation

The navigation stack was reworked to address multiple reliability issues observed during autonomous operation. Improvements to the IMU integration and corrections to the TF tree configuration resolved orientation problems which were already observed during last year's testing. These previously caused incorrect interpretation of sensor data. With these changes, the alignment between the robot base, odometry, and sensor frames is now consistent. Additionally, the navigation stack was fine-tuned to enable more accurate pose estimation and navigation. This was achieved by adapting the code for the base controller and reworking the costmap configurations.

Despite these improvements, the navigation behavior remains unstable under certain conditions. It was possible to make the robot navigate accurately under perfect conditions, like using the same starting point as the map and no change to the environment. However, this isn't practical as achieving these conditions is practically impossible in real-world environments. In most cases, the robot exhibits hesitant motion and oscillatory behavior during goal execution, largely attributed to limitations in odometry accuracy and the ROS1 navigation stack's sensitivity to localization uncertainty. In RViz, this manifested as correct LaserScan updates without corresponding updates to the robot's pose, highlighting remaining deficiencies in pose estimation and frame synchronization.

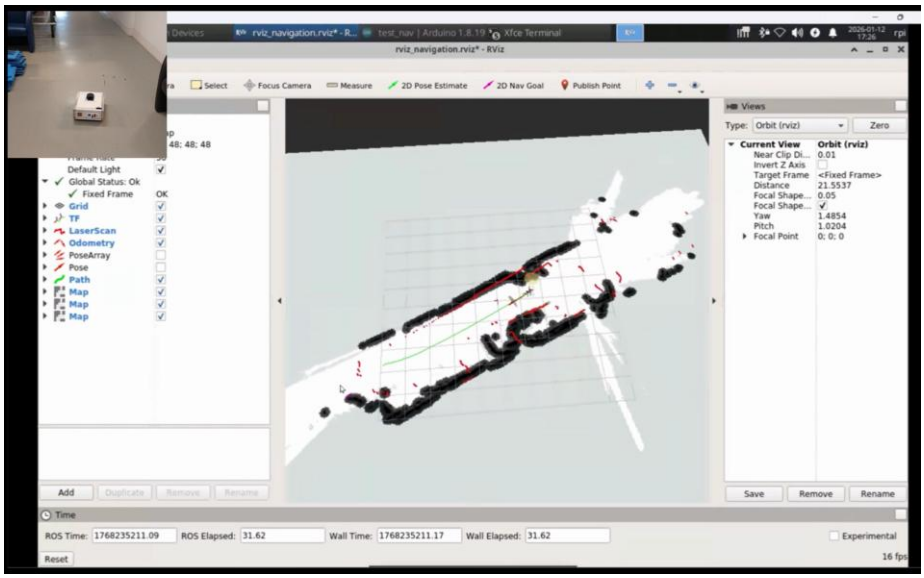


Figure 8 - Autonomous navigation

A potential future improvement would be a migration to ROS 2, which offers more modern navigation frameworks, improved middleware performance, and better support for real-time nodes. However, such a transition would require significant refactoring of the existing hardware interface and software architecture.

3.5/ Computer vision

3.5.1/ Depth Map

Using the OpenCV library, the project focuses on stereo vision to correlate object positions with depth information. The first and most critical step is camera calibration, which is required to obtain accurate geometric and metric measurements from the stereo setup.

Calibration and stereo processing are performed using a single high-resolution stereo camera and a printed checkerboard. The camera provides a combined frame that is cropped into left and right images. Single-camera calibration is applied first, where image pairs in which the checkerboard corners are not detected are automatically discarded. Once sufficient valid images are collected, stereo calibration is computed. When calibration data is available, rectification maps are applied to align epipolar lines between the two views. Depth information is then obtained using a tunable Semi-Global Block Matching (SGBM) algorithm. The resulting disparity map is converted into metric depth using the focal length and baseline obtained during calibration.

To visually validate calibration and depth coherence, a manual point cloud generation tool was implemented. From a single stereo frame, a colored 3D point cloud is reconstructed, allowing direct inspection of depth consistency and geometric

correctness. This step proved useful to ensure that calibration, rectification, and stereo matching were functioning as expected before moving to higher-level detection tasks.

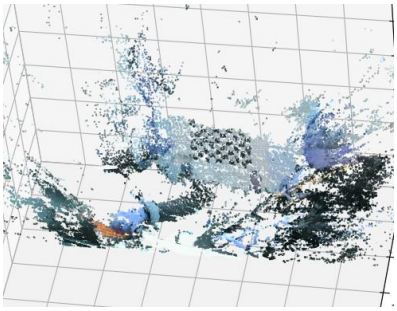


Figure 9 - Depth Map

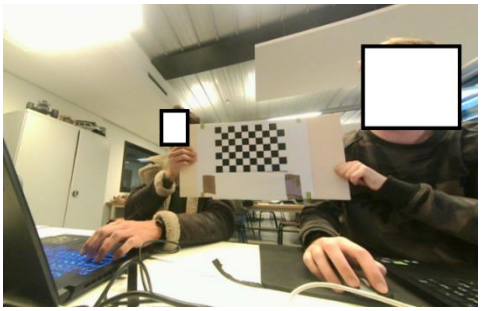


Figure 10 - Real-Life Scene of Figure 9

3.5.2/ QR Code detection

Once the stereo pipeline was validated, the system was extended to QR code detection. Detection is performed using the *pyzbar* library, which directly decodes QR codes from raw image data without requiring explicit preprocessing or handcrafted pattern detection.

For each incoming frame, QR codes are independently detected in the left and right images then matched using two criteria: identical decoded content and spatial consistency. Specifically, the vertical positions (y-coordinates) of the QR code centers must be within a small tolerance, typically 10-12 pixels, which leverages the epipolar constraint of rectified stereo images. Only QR codes satisfying both conditions are considered valid stereo pairs, ensuring that the same physical marker is observed simultaneously by both cameras.

3.5.3/ Triangulation

After matching, stereo triangulation is used to compute the 3D position of each QR code. The horizontal disparity is calculated as the difference between the x-coordinates of the QR code center in the left and right images. Distance is obtained by using the standard stereo relation.

With the depth known, image coordinates are projected back into metric space using the camera intrinsic matrix. The X and Y coordinates are computed from the QR code image position, focal length, principal point, and depth, yielding a full 3D position (X, Y, Z) expressed in millimeters, with the origin centered between the two cameras. In parallel, an independent depth estimate is extracted from the SGBM disparity map at the corresponding image region. The Euclidean difference between the triangulated depth and the depth-map estimate is used as a confidence metric, quantifying the consistency between the two methods.

Finally, the detected QR code position in image space directly provides the X and Y coordinates, while the associated region in the depth map is used to estimate distance. This depth estimate is validated against the triangulated depth. The two values are combined to produce a final Z measurement, along with a confidence score derived from their difference. This approach improves robustness while providing a quantitative measure of depth reliability for each detected QR code.

4/ Documentation

The project features extensive documentation to ensure reproducibility and facilitate further development. This documentation includes a detailed description of the Ubuntu installation process on the Pi, covering system configuration, desktop environment setup and remote access configuration.

In addition, information is also provided on the deployment of ROS1 Melodic within a Docker environment, including image selection, required modifications and container configuration to support the ARM64 architecture.

The documentation further details the setup and configuration of both the mapping and navigation workspaces, including package dependencies, launch configurations, TF structure, and sensor integration. A dedicated troubleshooting section in each document references all the encountered issues and potential solutions that worked.

These files also contain the bibliography of the used and necessary resources and sources for the project.

5/ Future improvement and changes

If the project is continued, several improvements and extensions could be explored to enhance the capabilities and maintainability of the system. One potential direction is total migration from ROS1 to ROS2 in order to take advantage of newer packages, improved real-time support, and enhanced communication mechanisms. However, it is important to note that many challenges will occur as ROS2 is fairly new and the documentation is still rather incomplete depending on the use case. This is one of the reasons why development still continued under ROS1 this year.

In support of ROS2 adoption, the current Arduino Nano-based motor controller could be replaced with a microcontroller or embedded board capable of running micro-ROS as rosserial was replaced by it. This would allow tighter integration of low-level motor control with the rest of the robotic system and improved communication reliability.

Another valuable improvement would be the use of further containerization to simplify the deployment and system configuration. By introducing a Docker Compose setup, the full software stack could be launched without requiring manual installation and

configuration of individual components. This approach would improve reproducibility, reduce setup time and make the system more accessible for future development.

The mobility system could be upgraded by replacing the differential drive configuration with a holonomic drive system. A holonomic base would allow omnidirectional movement, resulting in more accurate positioning and smoother navigation in constrained environments.

Finally, the upper subsystem which features the box detection as well as the grabbing mechanism itself, can be improved by revising the structure to prevent bending due to motor torque. Also, the communication interface with the mobile base will have to be reworked as part of the modularity upgrades.

6/ Conclusion

6.1/ Costs

The component costs for the project are summarized in the table below:

Components needed to build V4 of Grabby			
Component	Link	Quantity	Unit price (€)
RaspberryPi			
RaspberryPi 5 (16Gb)	Amazon	1	139,89 €
SanDisk 128 Go Ultra microSDXC	Amazon	1	12,55 €
Active Cooler for Raspberry Pi 5	Amazon	1	8,99 €
Bloc d'alimentation Officiel Raspbe	Amazon	1	14,95 €
Battery and Power			
XT60 Connecteurs	Amazon	1	6,49 €
Convertisseur DC-DC 60 W DFR10	GoTronic	1	11,20 €
Honseadek Convertisseurs DC-DC	Amazon	1	8,99 €
Structure			
Aluminium Profile 2020 mm 1000 r	Amazon	2	28,02 €
10PCS Interior Angle Slot 6 EU-202	Amazon	2	9,91 €
2020 Series Black Aluminum Corne	Amazon	1	11,99 €
Other			
Aimant Neodyme	Amazon	1	5,39 €
Attaches de Câble	Amazon	1	5,99 €
Module ICM20948 9 DoF PIM448	Gotronic	1	16,90 €
AZDelivery 3 x ACS712 20A Module	Amazon	1	9,99 €
Gebildet 3PCS Métal on/Off/on Inte	Amazon	1	11,99 €
Total	341,17 €		

The filament costs also need to be considered. Mixing the different types of filaments used (PLA, PETG, GreenTech Pro Carbon) and the total weight, the cost estimate is about 50€.

Considering an average salary of 38.000€ per year for 1600 work hours, this means that the total of 320 invested hours of work totals to a salary of 7.600€ for this development year.

This means that the project has cost about 7.991,17€ this year.

Over the three years that this project has been developed, this raises the total costs (components and work hours included) to 29.091,91€.

6.2/ Project

This project led to the development of **Grabby**, an autonomous mobile robot designed as a proof of concept for warehouse automation and research. The work focused on building a modular, reliable, and extensible platform capable of supporting autonomous navigation, mapping, and localization in structured indoor environments. The redesigned mechanical architecture significantly improved robustness, maintainability, and cable management, while preserving a compact footprint.

On the software side, a complete autonomy stack was implemented using ROS1 integrating teleoperation, SLAM, localization, mapping and navigation as well as a computer vision stack. Despite challenges related to hardware architecture limitations and ROS version compatibility, a mostly functional prototype was achieved. The use of containerization simplified development by keeping the software environment independent from the underlying system, while future improvements - such as a migration to ROS 2 - could further enhance autonomy and long-term maintainability.

Overall, this project establishes a solid technical foundation for further experimentation and development. The modular hardware design and well-documented software architecture enable future work to focus on improving navigation performance, integrating additional sensing or manipulation capabilities, and transitioning toward more modern robotic frameworks.