ResQBot – P.e.D.r.O.

André Monteiro Cocco, David Björklund & Tobias Sjöblom
Informationsteknik, Mechanical and Sustainable Engineering & Process- och Materialteknik

Robotics
23.4.2024

Abstract:

Robotics are becoming increasingly incorporated into society which makes it an important fundamental in engineering education. And this is why the purpose of this project is to learn basic robotics by creating of a fully autonomous robotic system. Using a TurtleBot3 burger kit as a framework which is a robot operating systems (ROS) based robot designed for students, researchers, and hobbyists. With an objective to navigate around spaces using light detection and ranging technology (LiDAR). Which is a technology that uses a laser to measure distance by calculating the time it takes for the light to bounce back to the sensor. And by rotating it measurement can be obtained in a full 360 degrees. By utilizing simultaneous localisation and mapping (SLAM) technique the robot can utilize LiDAR sensor data to map its surroundings and localise itself in an area and use this information to effectively navigate and avoid obstacles. To deepen its functionality, it is equipped with a camera and object recognition program to allow it some working functions while navigating.

The educational purpose of the project is to prepare students for broader engineering challenges. With a wide range of robotics applications and sensors and gadgets that can be integrated makes the options available almost unlimited. Which is excellent for project management and as most robotic applications include electrical, mechanical as well as programming making it a perfect theme for getting practical, hands-on experiences in three different fields even while keeping it simple.

Acknowledgements:

We want to thank Dennis Biström for all the help and Arcada University of Applied Sciences for providing the TurtleBot3 Burger kit and tools.

Contents

1. Introduction

A group effort centred around project management with an objective to plan, design, build and program a functional robot. Some different pathways were given such as Niryo Ned2 simulation programming and Nvidia Jetson Nano platform. But showing interest in LiDAR technology our group were offered an alternative, an TurtleBot3 Burger. This robot platform consists out of OpenCR control module which is an open-source robot controller with an ARM Cortex M7 MCU (OpenCR1.0 n.d.) as well as a Raspberry Pi and USB2LDS board (Features n.d). This makes it a powerful setup where sensors and electronics such as LiDAR, camera and motors can be connected.

The first concept was a rescue robot that would be able to make its way through the rubble of a fallen building in search of life, hence the name ResQBot. Which is something that could have important impact in real-life scenarios. The world is overwhelmed by war and natural disasters like earthquakes, tsunamis and volcanic eruptions which makes robotic rescues very important so you can find sign of life before you send rescuers into a dangerous situation. A robot like this would need to be tough enough to handle some falling debris as well as being as watertight as possible. It would also require excellent mobility and traction while maintaining the smallest possible size. Equipping it with a light source, camera, and thermography to give it the ability to find people and identify their life sign.

As the group consists of a blend of members from mechanical and information technology backgrounds the interest in a physical robot was the most logical choice. So, the options were between the Nvidia Jetson Nano and TurtleBot3 Burger. And the robot platform of choice were the TurtleBot3 as LiDAR and autonomous navigation have many similarities with self-driven vehicles which could be important knowledge for engineering careers.

One aspect of the plan was to make the robot able to drive both upside up as well as upside down to reduce the possibility of getting stuck if fallen over. For maximum traction and climbing ability different wheel designs as well as so called tank tracks concepts were investigated. And to be able to produce as much as possible of the robot parts by ourselves the concept of 3D printing the wheels or tracks out of thermoplastic polyurethane (TPU) were consider and as a second alternative was to 3d print moulds to produce them out of silicone as silicon has more rubbery consistency which would result in better traction.

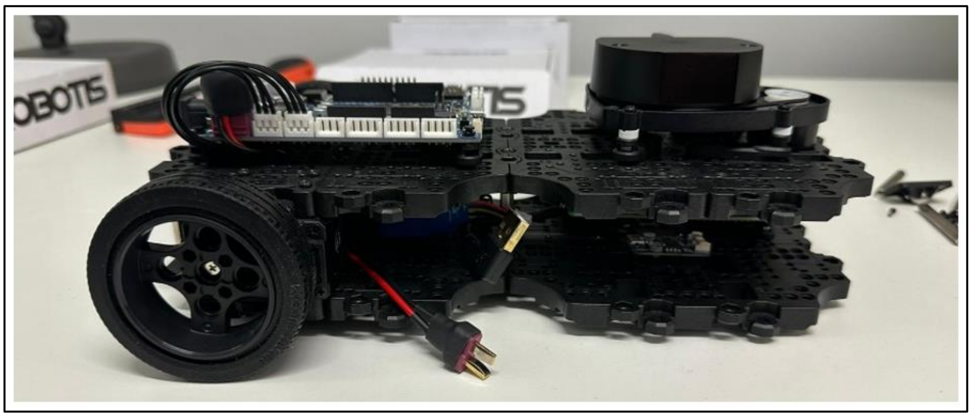


Figure 1. Configuration of TurtleBot3 to be used as a tank design.

But as this would require the purchase of materials that we lacked experience using which would result in many tests both for materials and designs. But as this started to get quite far beyond the scope of the course and lacking time and funds the decision to simplify was done. This meant the homemade wheel and tank tracks concept were replace with the original wheels of the TurtleBot3 and the rescue robot concept were replaced with SLAM navigation and object recognition.

After changing tactics for a more simplified approach the first step was to rebuild the TurtleBot3 Burger to its original setup.

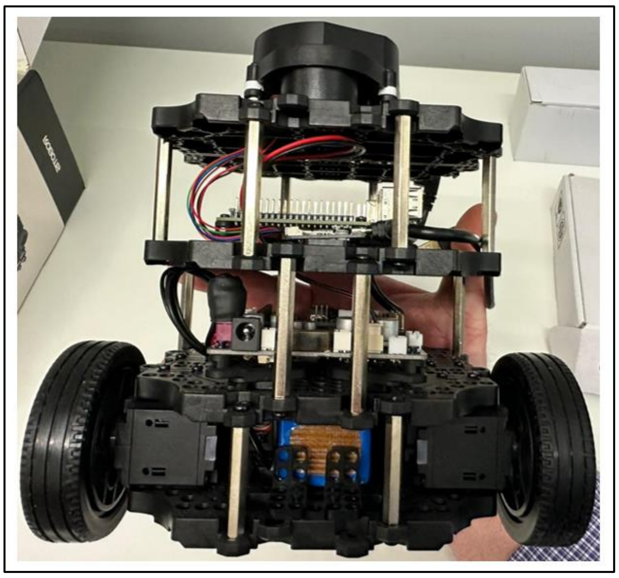


Figure 2. TurtleBot3 assembled back to Burger design.

The next step was to secure a connection with the robot and start to familiarize ourselves with the SLAM programming. The expected outcome of the project is to have the TurtleBot3 Burger to be able to map and navigate an area without running into obstacles, autonomously, and to recognise humans and objects. This would be a good base that can be modified and improved upon.

2. Methods

Programming and robot hardware

1.1.1 Introduction

The programming part of the robot consisted of researching topics, choosing implementation angle and developing an MVP (Minimum Viable Product) system of a robot that can navigate using slam, recognize people and alert the driver/operator of people in a hazardous environment in real time. With undefined resources in the beginning, the project had a long research period. After focusing on the Turtlebot3 development kit the software research and development could start. We did not have a starting point other than the hardware.

The current state of open software resources for search and rescue robots are limited. One student project that inspired this projects code is from a repository where the robot could navigate, detect objects, and track objects. It also had an arm that could grab simple objects. (maggielovedd)

The programming contribution to the project focuses on demonstrating the viability of a cost-effective search and rescue robot with existing software. The Turtlebot3 development kit, augmented with a web camera and a speaker, provides a base. We utilized open-source libraries like ROS and OpenCV to implement SLAM navigation and real-time person detection using the pre-trained object detection machine learning model MobileNetV2. While older hardware may have limitations, we prioritized affordability and adaptability over raw processing power. This of course constrains how powerful the product will be.

1.1.2 Research robot ecosystem

1.1.2.1 Machine learning

The project employed machine learning techniques to enable robotic navigation and object recognition. I selected the MobileNetV2 convolutional neural network (CNN) model due to its balance of computational efficiency and accuracy. The model was pre-trained on the COCO dataset, providing a robust foundation for person detection, a critical requirement for the search and rescue context.

1.1.2.2 NVIDIA Jetson Nano

During the initial research phase, we considered the NVIDIA Jetson Nano development kit. This platform offers an ARM CPU with higher theoretical performance than the Raspberry Pi, making it an attractive option for robotics applications. It has a dedicated GPU but lacks wireless Lan.

1.1.2.3 TurtleBot3

The TurtleBot3 development kit, popular among hobbyists, researchers, and students, is compatible with both Raspberry Pi and Jetson Nano single-board computers. It features a control board for motor communication and comes equipped with a LIDAR sensor for navigation. While the TurtleBot3 has its own libraries, it fundamentally relies on the Robot Operating System (ROS). The ROS version we used (Humble Hawksbill) imposed limitations on compatible libraries, particularly those for autonomous driving and machine learning. Maintaining this complex ecosystem requires up-to-date open-source libraries, which can create challenges when using older hardware or if tutorials have not been updated to match software dependencies.

1.1.2.4 LiDAR

A lidar is a light detection and ranging system. It consists of a laser detector and a laser that targets a surface and measures the time for the light to reach back to the detector. The laser emitter often rotates or uses a system of mirrors to scan a wide field of view. By calculating the time of flight for numerous laser pulses, the lidar builds a detailed point cloud representation of its environment.

1.1.3 Implementation

1.1.3.1 Hardware and Software

The project hardware centred around the TurtleBot3 Burger developer kit, which includes a single-board computer and motor control board. We followed the official online manual (Robotis, 2024) for initial setup and configuration. For computer vision, the robot was equipped with a standard webcam. Audio output was handled by a compact speaker connected via a 3.5mm cable.

The web camera was a Microsoft Lifecam NX-6000 and the speaker a Nokia MD-9. Both over 15 years old at time of writing.

1.1.3.2 ROS

The Robot Operating System (ROS) is a powerful open-source framework providing essential tools and libraries for robotics development. (Woodall, 2022) Its core strengths include hardware abstraction, device drivers, communication protocols, and extensive visualization tools. ROS's wide adoption across the robotics field makes it a valuable asset. (Woodall, 2022) To ensure compatibility with our chosen operating system, Ubuntu 22.04, we selected the ROS Humble Hawksbill distribution. This decision ensures access to the specific libraries and features supported by this ROS version.

1.1.3.3 Navigation

Robots offer multiple navigation modes. Manual control can be achieved via network or radio connections, allowing for direct user input. For this project, we leveraged existing libraries within the TurtleBot3 and ROS frameworks to enable navigation. This involves using sensors (e.g., LIDAR) to map the environment and algorithms to determine the optimal route. Specifically, we used RViz, a ROS visualizer, to set navigation goals for the TurtleBot3.

1.1.3.4 SLAM

Simultaneous Localization and Mapping (SLAM) is a computational technique enabling robots to concurrently build a map of an unknown environment while tracking their location within it. In this project, we employed a SLAM implementation that utilized the TurtleBot3 development kit's onboard LIDAR sensor. This approach facilitates accurate mapping of rooms and environments, allowing for robust navigation of the robot without relying solely on external visual cues.

1.1.3.5 Visualisation

Various approaches exist for real-time robot visualization. (Alsallakh, 2022) In this project, we opted for a straightforward solution using a standard webcam and an onboard web server. This setup provides a live video feed accessible to the user, offering remote situational awareness. We selected Python as the programming language for this task due to its ease of use, extensive libraries for image processing, and efficient web server frameworks. We utilized the server framework Flask for our project.

1.1.4 Operating system

1.1.4.1 Raspberry Pi

The Raspberry Pi, a versatile single-board computer, is widely used for development projects due to its affordability, strong community support, and ease of prototyping. For this project, we utilized the Raspberry Pi Model 3 B+. This model, released in February 2016, our approach was that it offers enough processing power and compatibility with the necessary hardware and software components. (raspberry).

1.1.4.2 OpenCR1.0

OpenCR1.0 is a control board for robot projects that can orchestrate the commands issued by the software to the hardware, such as wheels, motors, and other electrically operated devices. (OpenCRgit)

1.1.4.3 Ubuntu

Ubuntu is a Linux Operating system based on the GNU/Linux Debian operating system. It is an operating system used in large projects and small projects. We chose the operating system according to the manual from TurtleBot3. (Robotis, 2024)

1.1.5 Computer vision

For our project requirements we had person recognition. For that we need computer vision. CV is the way computers can see and interact with our world. It is similar to how the human vision works because of how we interact with computers.
In our project we used a Python library called OpenCV. The library is a Open Source Computer Vision library and has many tools to develop real-time computer vision applications. The software library can be used in many different programming languages. For our purpose, the OpenCV has module called Deep Neural Networks. It can be used to run classification, segmentation, and detection models. (opencv)

1.1.5.1 Google Coral TPU

Google Coral USB TPU inference accelerator was released in 2019 and used as a development platform for machine learning. It uses an edge TPU, a small coprocessor that a can be used by systems for accelerated inference from different machine learning models. Our initial theory was that we could improve the machine learning inference by using an accelerator such as the Coral TPU. Google has not officially discontinued the device but there has been less development in recent years to support newer technology and software dependencies. (google)

1.1.5.2 Python

Python is a programming language used in many machine-learning and data processing tasks. Both the ROS, Coral TPU and Raspberry Pi support python. We were prepared to use python in our project because of the many libraries that we could use in both programming and machine learning.
For our operator view we chose a python library called Flask. Flask is a framework can be used to build a web application. It has the tools to generate the content that we planned.

For our project we wanted:
Visual interaction. See what the robot does.
Sound. Stream the sound from the robot to the operator.
Communication. Send voice or sounds to the potential person in need of assistance.

For the webserver we forked a small test project from a developer EbenKouao. He has built a Raspberry Pi camera stream server with Flask. (EbenKouao, 2020). The fork is located at GitHub and can be freely viewed and used as it is with the same license as the original repository. (davidrepo) For object detection we based the code from the tutorial written by Saumya Shovan Roy. (Saumya Shovan Roy) He has also made a repository on GitHub with the code in question.

1.1.5.3 TFlite

Tensorflow Lite is a smaller version of Google's machine learning framework, TensorFlow. The primary goal is on-device inference on resource-constrained environments. TFLite is optimized to run machine learning models on devices like:
Smartphones (Android and iOS)
Embedded systems (Raspberry Pi and similar)
Microcontrollers (Tiny, low-power devices)

For our project we chose to use a pre-trained model called MobilenetV2. Mobilenets are useful for less powerful computers. (Andrew G. Howard, 2017) The model can recognize objects such as people, planes, dogs and other objects.

1.1.5.4 Oak-D

Oak-D is the name of a camera line built by Luxonis. The cameras are equipped with cameras and also computer vision capabilities because of the built-in processor. One can use the python library depth-ai from Luxonis to develop computer vision applications. We were given the Oak-D Lite to test if it could fit our purpose with robot vision. The theory was that we could use the inbuilt stereo camera to recognize people and offload the raspberry pi somewhat because of the required processing power to run object detection locally on the computer.

3. Shell Design – CAD

The design concept had different influences. It began with the decision to forgo the initial tank model ideation, due to time and resources constraints, and from that moment onwards it was decided amongst the group participants that the brand image of the robot would be approach with fun. A brainstorm session was held, and since the original assembly's name of the TurtleBot3 is Burger, the group chose to use the namesake food as the robot's skin design, a cheeseburger. Because the project foundation was to be used as a rescue assistance robot, the choice to characterise the cheeseburger as a firefighter was made, with features that evoke the American movies' image on firefighters - thick eyebrows and thick moustache.

The design was developed in SolidWorks® by using different parts to create the elements that form the complete skin. The top and bottom bun shapes were inspired by a model found in Thingiverse® (Thorburn, 2019), but nevertheless, the sketches, construction and final shape are original. The constrain in the shell's inner diameter was determined by measuring the perimeter of the TurtleBot3's waffles (⌀140 mm) and adding an offset of 20 mm radius to ensure that the skin would not touch the robot (total ⌀ 180 mm), and that the hardware would have airflow allowing the cooldown while in operation. The height was determined by measuring the distance between the base of the LiDAR® that rests on top of the TurtleBot3, and 10 mm before reaching the intersection of robot tires/ground (approximately 158 mm), and that would be the same height constrain for the eyebrows to be set and not to interfere with the LiDAR® readings.

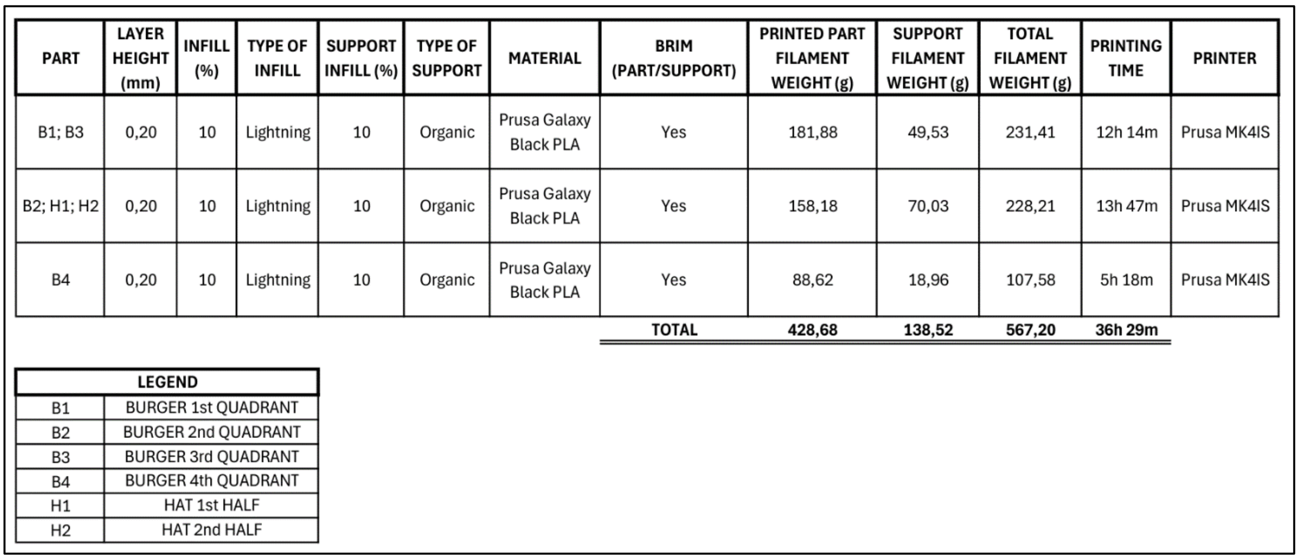
The top bun, patty, and lower bun were sketched as a profile and extruded together as a shell using the command "Revolved Boss/Base". On top of the patty and perpendicularly, new planes were created to design the melting cheese by using "Lofted Boss/Base" from sketches in different planes; after done, a new auxiliary plane was created above the top bun, and the sesame seeds were sketched, then “Wrapped” into the top-bun’s surface creating a small cut on each seed’s surface, which were then extruded using “Dome”, individually.

The tomatoes, and pickles were sketched, and extruded with “Extruded Boss/Base”. The eyebrows and moustache were created by sketching the desired shape and then creating a dome off each shape and mirroring the results. After all parts were done, an assembly was created, and the elements were carefully positioned to create a cheeseburger with eyes, eyebrows and moustache, and then the assembly file was saved as a part, to decrease the file size and enable a quicker processing response from the computer while editing the final details. The eyes, eyebrows, and moustache were purposely set out of alignment to create a unique result, and as a reminder that aesthetic perfection is not necessarily a goal.

Holes were to be made in the design to accommodate the LiDAR®, the wheels, the camera, and the speaker, and to accomplish such, a full CAD (STL) model of the TurtleBot3 (Pyo, 2017) was used by adding it to an assembly file together with the final shell. From the intersection of both models, the recess for the wheels/tires, LiDAR® and speaker were sketched and cut. The camera slot was created by measuring the position on the physical robot and sketching and cutting it on the digital model. For the cheeseburger shell to be adhered to the robot, pins were created by projecting the slots on the TurtleBot3 waffle into the inner top surface of the shell and extruding the projected sketched circles into pins. To accompany the firefighter cheeseburger, a hardhat (printable_models, 2018) was downloaded from Free3D®, edited, and then the logo of the Helsinki’s fire department was created in SolidWorks® by tracing out an internet image of said logo, projected and extruded into the modified hat.

The 3D printing of the model was done using a Prusa® MK4IS printer, and the parameters used are described in Table 1.
Due to size and printing time limits, the shell was divided into four quadrants and the hat was divided in two; by doing so, shorter printing times were achieved, and the geometries to be printed were less challenging for the printer, decreasing the printing error probabilities.

Table 1. 3D printing parameters used in the project.



PART	LAYER HEIGHT (mm)	INFILL (%)	TYPE OF INFILL	SUPPORT INFILL (%)	TYPE OF SUPPORT	MATERIAL	BRIM (PART/SUPPORT)	PRINTED PART FILAMENT WEIGHT (g)	SUPPORT FILAMENT WEIGHT (g)	TOTAL FILAMENT WEIGHT (g)	PRINTING TIME	PRINTER
B1; B3	0,20	10	Lightning	10	Organic	Prusa Galaxy Black PLA	Yes	181,88	49,53	231,41	12h 14m	Prusa MK4IS
B2; H1; H2	0,20	10	Lightning	10	Organic	Prusa Galaxy Black PLA	Yes	158,18	70,03	228,21	13h 47m	Prusa MK4IS
B4	0,20	10	Lightning	10	Organic	Prusa Galaxy Black PLA	Yes	88,62	18,96	107,58	5h 18m	Prusa MK4IS
							TOTAL	**428,68**	**138,52**	**567,20**	**36h 29m**	

LEGEND	
B1	BURGER 1st QUADRANT
B2	BURGER 2nd QUADRANT
B3	BURGER 3rd QUADRANT
B4	BURGER 4th QUADRANT
H1	HAT 1st HALF
H2	HAT 2nd HALF

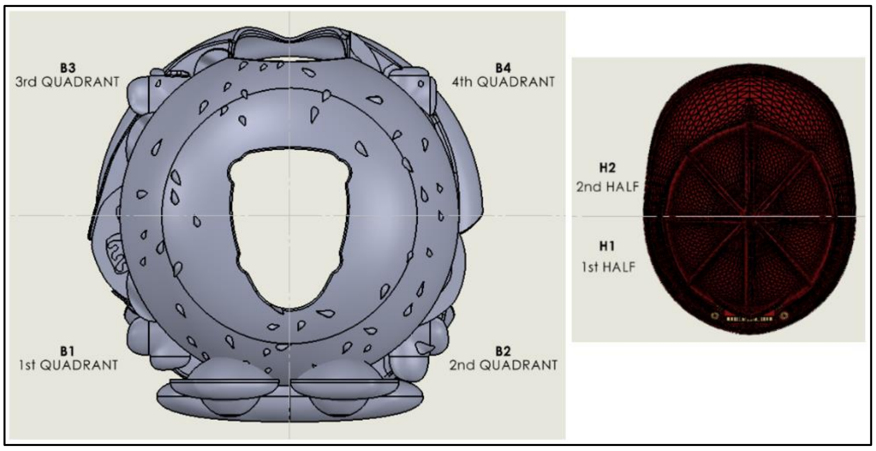


Figure 3. Quadrants division for printing.

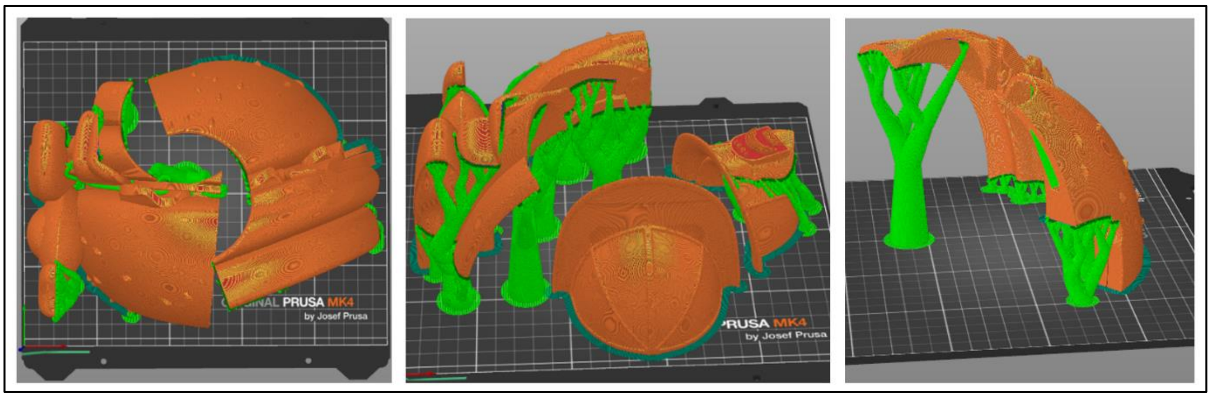


Figure 4. Screenshot of the Prusa® slicer for the three print portions.

The results of the prints were mostly positive, the only issue that had to be delt with was an error that caused a layer shift, most likely caused by the low density of the supports (which was done to save filament), which printed the top of the B2 part 20 mm off its original place;

nonetheless, the shifted parts were cut, adhered with super glue, and finally the shift generated seams were filled and covered with acrylic plaster (Kintsuglue®).

After the prints were done all supports were removed, and the parts B1-B2, and B3-B4 were adhered with superglue and internally filled with acrylic plaster in their inner seams, followed by painting each part with acrylic paint, and making the moustache and eyebrows hairs from cotton thread, which were set in place with super glue. The hat received a coat of golden paint to highlight the fire department logo, and to portray a worn-down aesthetic; leather straps were glued to the inside of the hat, and magnets were adhered to the end of the bands to ensure that the piece would remain in place while the robot is roaming. For the manufacturing of the shell, from the SolidWorks® designing, 3D printing, painting, to glueing down cotton threads, a total of 65 hours were used.

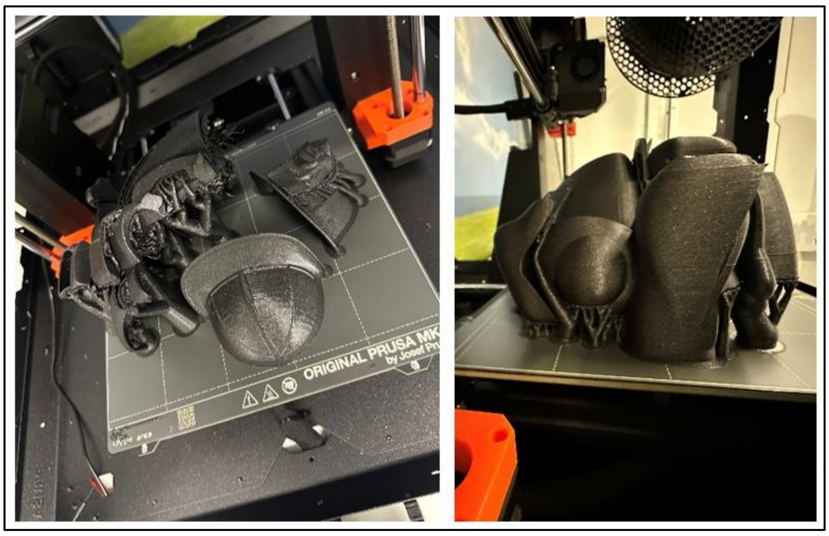


Figure 5. Result of the first printing portion. (B2, H1, and H2)

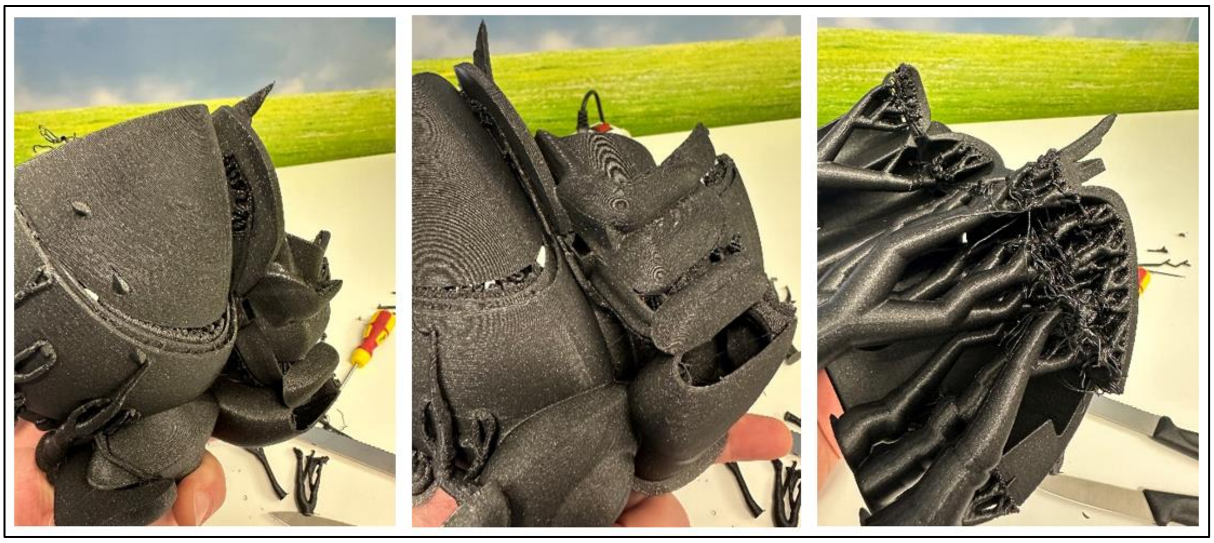


Figure 6. Layer shift error. (B2)

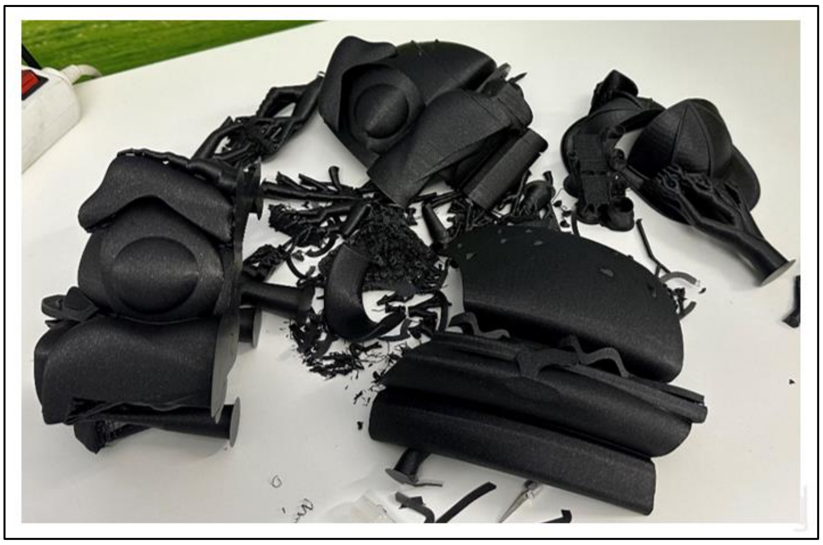


Figure 7. 3D printing results. (B1, B2, H1, and H2)

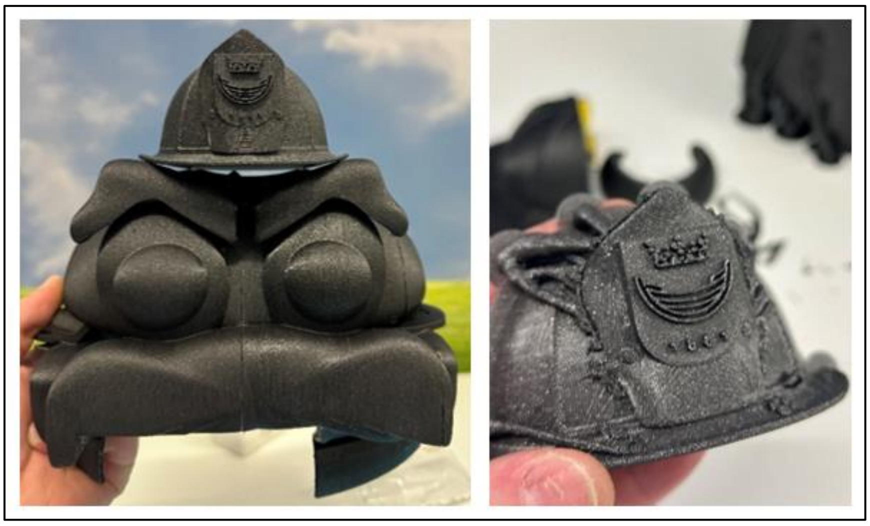


Figure 8. Part B2 fixed and adhered to part B1; hat print glued and logo detail.

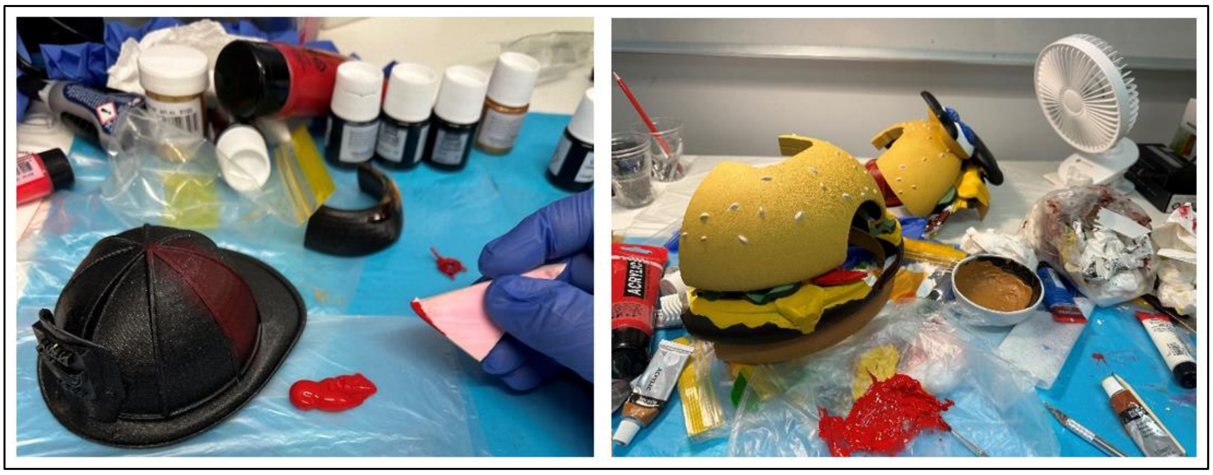


Figure 9. Painting process.

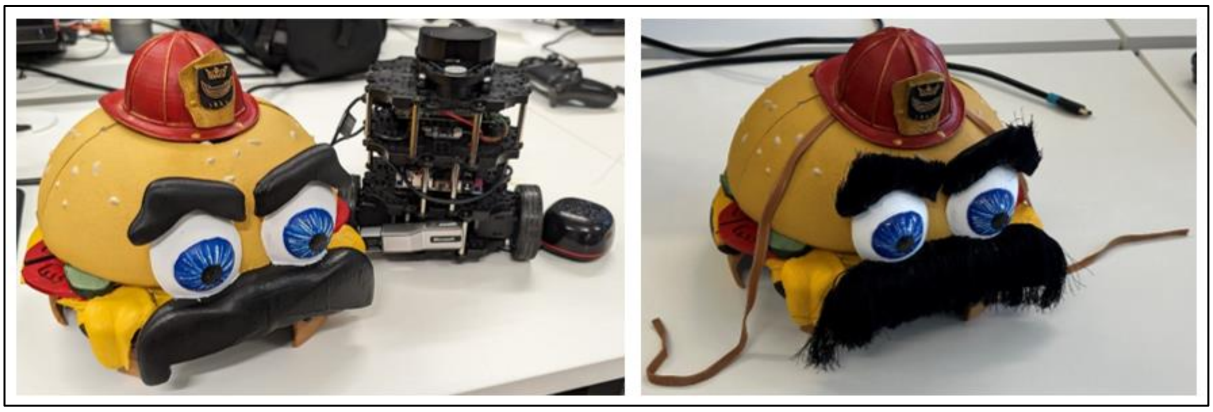


Figure 10. Finalised design prior to the addition of "hair" and leather straps; finalised design.

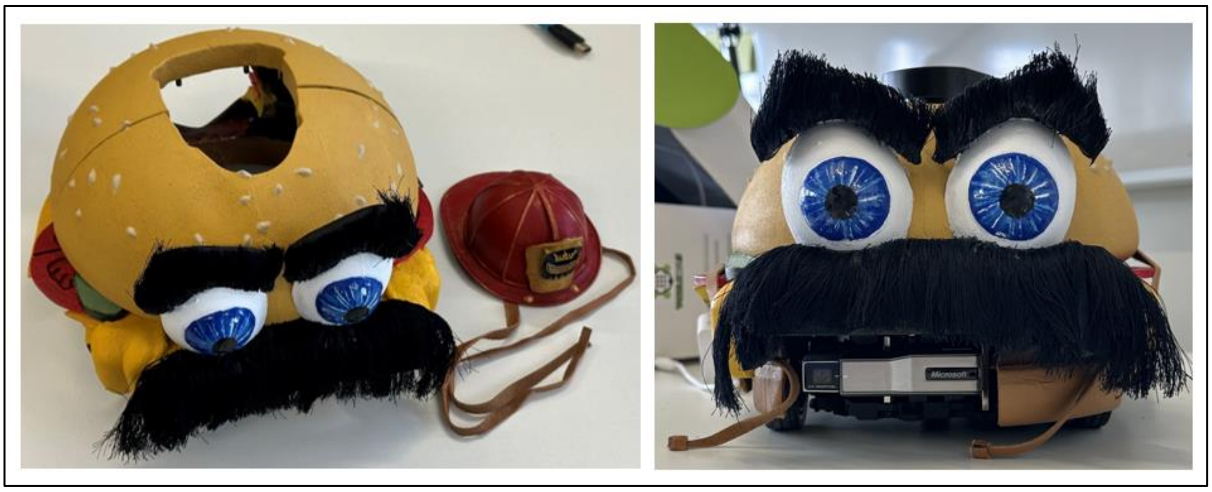


Figure 11. Lidar recess detail; camera recess detail.

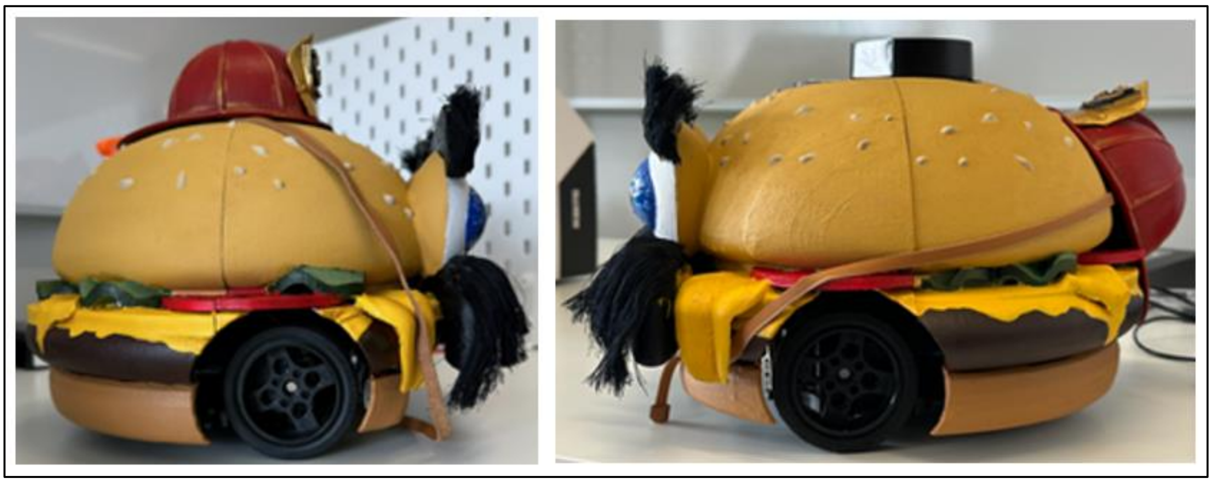


Figure 12. Wheels/tires recess detailed view.

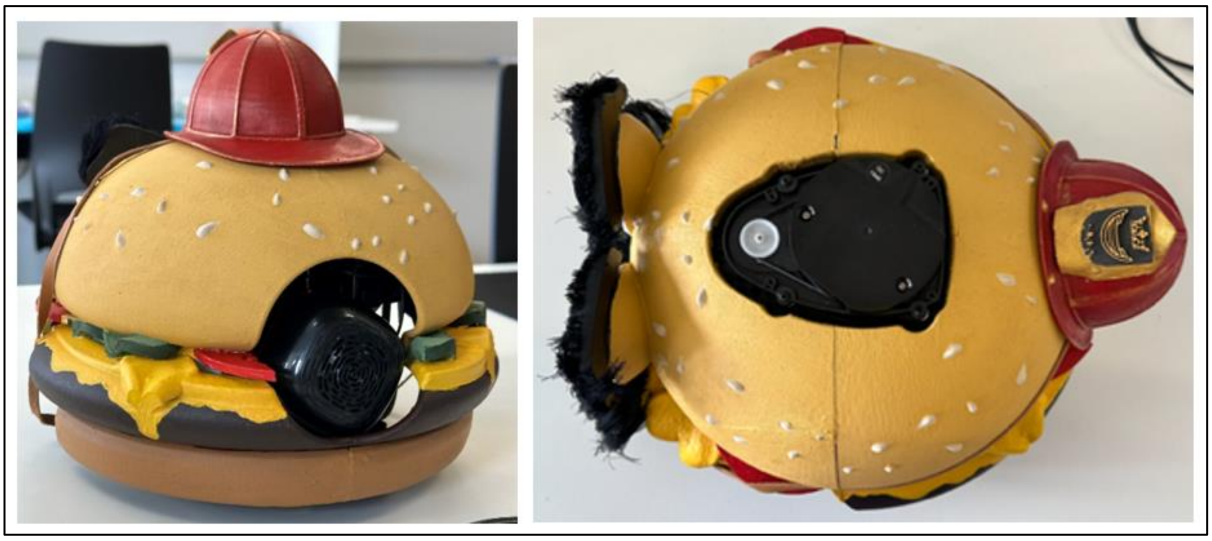


Figure 13. Speaker recess detail; top view of LiDAR® recess with LiDAR in place.

4. Results

1.2 Programming Results

The robot we had in the end was a robot that could navigate from one point to another. Stop when an obstacle was detected and identify persons. The operator could also manually drive the robot and communicate a simple command if needed. For the readers convenience a repository for following along with the installation and operation of the PEDRO (People Detection Robot Operations) robot. (Dodisbeaver)

In the research phase we were given the option to use the Jetson Nano or Turtlebot3. After some consideration we chose the Turtlebot3. The main reason for not using the Nano was that it had complex instructions and we did not find as new software as for the raspberry pi. It was potentially more performant but according to our competitor team it had many obsolete dependencies and problems because of obsolete software.

In the research phase we started with a simulation of the TurtleBot3 in the Gazebo program. A user can install the ROS framework and run a simulation of a robot on their computer. This was a success and showed us the capabilities of the TurtleBot3. This led us to research about the TurtleBot3 more. Initially it was complex because it was unclear to the user which operating system to use with which ROS framework version and TurtleBot3 library to build.

1.2.1.1 MobileNetV2

In the object detection research, we looked at what other developers had used with similar equipment. Many of the tutorials and repositories used the MobilenetV2 model. After testing it on a PC with ubuntu installed it showed performant and accurate results that would fit our needs. The MobilenetV2 model that was trained on the COCO image dataset became the model we used in the end. We had to modify and tune the input size of the inference frame from 300x300 pixels to 150x150 pixels because the frame rate became lower with the higher size.

1.2.1.2 Google Coral Tpu

We initially wanted to use the Google Coral TPU in our project because it could potentially accelerate the inference for the object detection and offload some of the workload of the raspberry pi’ (printable_models, 2018) (Pyo, 2017) (Thorburn, 2019)s CPU. Unfortunately, the TPU is not supported by newer operating systems such as Ubuntu 22.04 and is dependent on python older than the initially installed version in the system. Google has not updated the version dependencies as of this writing. Some users have shown interest in updating it themself and has forked the repository. But this is not a simple task and requires some knowledge of how to build a package from scratch. We were unsuccessful in using the accelerator in our project, but if installed on older operating systems the accelerator can be used. In our research and test phase we used an image from a Home Assistant project where one of the required hardware (Aiy Project). The accelerator has potential to speed up the inference if the packages can be updated accordingly.

1.2.1.3 Computer Vision

The tutorial located at the PEDRO GitHub repository also has the object detection script we used in the end. For testing we used different cameras with varying degree of success. The first camera used in testing phase was Creative Life Cam Sync HD. We initially used a Raspberry Pi 2. This computer could stream and use object detection but at the cost of many frames per second. We continued our testing on PC:s and a Raspberry Pi 3 B+ similar to the Raspberry located in the Turtlebot3 kit. Continuing our testing with other cameras included a Microsoft Lifecam Studio with no success on the Raspberry Pi 3. Later we tested a Logitech HD camera with some success but with artifacts. This led to debugging why some cameras did not work and some did. The theory is that the usb bandwidth is not enough when using open-cv with python 3.11 as some of the cameras worked with 3.10 but not with 3.11. After finding an old Microsoft Lifecam NX-6000 we found that it had enough of resolution to use in our project.

We got to test the Oak-D Lite and Oak-D Pro W in our research. The cameras are powerful and can inference on device because of the onboard AI processor. They have stereo vision and can measure the distance of objects because of this. The Oak-D Pro W also has infrared lights and a laser for more accurate distance measuring.
The Oak-D lite was initially a potential candidate for our computer vision but because of time constraints from getting the robot working we did not have enough time to implement it in our project. With more time this would have been our primary camera candidate because it could offload the raspberry pi from inferencing the object detection.
The maker of the cameras Luxonis has an extensive library of tutorials. The camera shows much potential for future use cases where computer vision and inference are required with edge devices such as the raspberry pi. One other bonus aspect that we found is that the inference is done on the camera and such potential data leaks and frames that can be viewed by a third party can be mitigated.

For the object detection we used OpenCv:s dnn moduel (Deep Neural Network). It has the convenience of having the capability to load many of the machine learning models. It can feed images into the models and produce classification or object detection. For our use case it fit because of the possibility to quickly create a program that can detect objects. When used by python there is a trade-off of performance, in the research we came to the conclusion that there is a possibility to use opencv with a faster programming language such as C++ that has a faster compilation time.

5. Discussion and Conclusion

The project began with challenges beyond the groups' control; by having five lecturers, information was unevenly distributed, and on several occasions, when asking a question, the response would be different depending on which lecturer one would ask. This happened mostly throughout the first month of the course, while waiting for a robot to be assigned to each group, there was no direct group conversation available to post questions. These instances caused wear, tear, and loss of momentum in the group.

A different problem was the type of feedback given after the pitch submission, where the group received as feedback "be careful not to promise a space trip and end up with a water rocket (…) you don't have the capability to deliver what you are promising", with the link of video of a person launching a pressurised PET water bottle.

The 3D printing was also an issue and set the group back off its proposed schedule for over five weeks. The CAD design was done, but there was no permission to print the parts in the laboratory B225. When asked why the print unavailability, the laboratory B225's chief engineer (Harri Anukka) said that no one asked him for approval to print the project. After waiting for five weeks to print in B225 (and four days before the final submission), the group was able to print the project thanks to Dennis Biström, who provided access to his 3D printer.

The group delivered what was promised in the pitch, and achieved the goals proposed of autonomous navigation, object/human recognition, SLAM, and obstacle avoidance.

It would be of great use for a next group to begin from where the final development of this project has finished, and build on the developed systems, rather than beginning from zero. Overall, it was a great project, with too many (idle) lecturers.

References

Networks for Mobile Vision Applications. Hämtat från arXiv.org: https://arxiv.org/abs/1704.04861

EbenKouao. (den 9 October 2020). *pi-camera-stream-flask.* Hämtat från https://github.com/: https://github.com/EbenKouao/pi-camera-stream-flask

printable_models. (den 27 September 2018). *Free3D Models, NYC Fireman Hat v1 printable, low poly model.* Hämtat från Free3D: https://free3d.com/3d-model/nyc-fireman-hat-v1--151920.html

Pyo. (December 2017). *Onshape, ROBOTIS TurtleBot3 Burger*. Hämtat från Onshape: https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/9ae9841864e78c02c4966c5e

Robotis. (den 24 4 2024). *Turtlebot3 e-Manual.* Hämtat från emanual robotis: https://emanual.robotis.com/docs/en/platform/turtlebot3/features

Thorburn, B. (den 01 July 2019). *Burger Man*. Hämtat från Ultimakler Thingiverse: https://www.thingiverse.com/thing:3723251

Woodall, S. M. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, eabm6074.

Features. (N.d.). ROBOTIS. https://emanual.robotis.com/docs/en/platform/turtlebot3/features/#components

OpenCR1.0. (N.d.). ROBOTIS. https://www.robotis.us/opencr1-0/

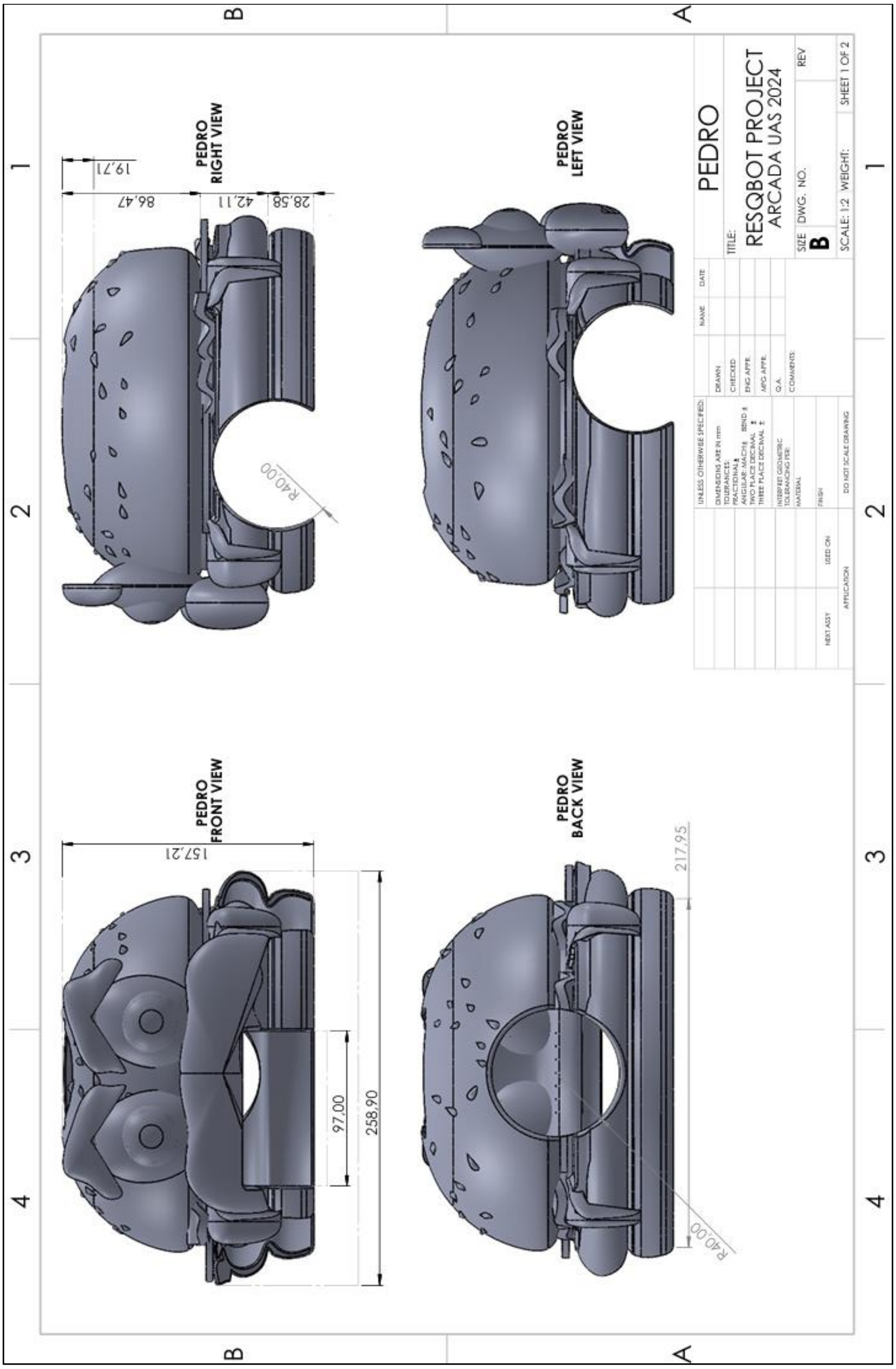
Alsallakh, B., Bhattacharya, P., Feng, V., Kokhlikyan, N., Reblitz-Richardson, O., Rajan, R., & Yan, D. (2022). A Tour of Visualization Techniques for Computer Vision Datasets. *arXiv preprint arXiv:2204.08601*.

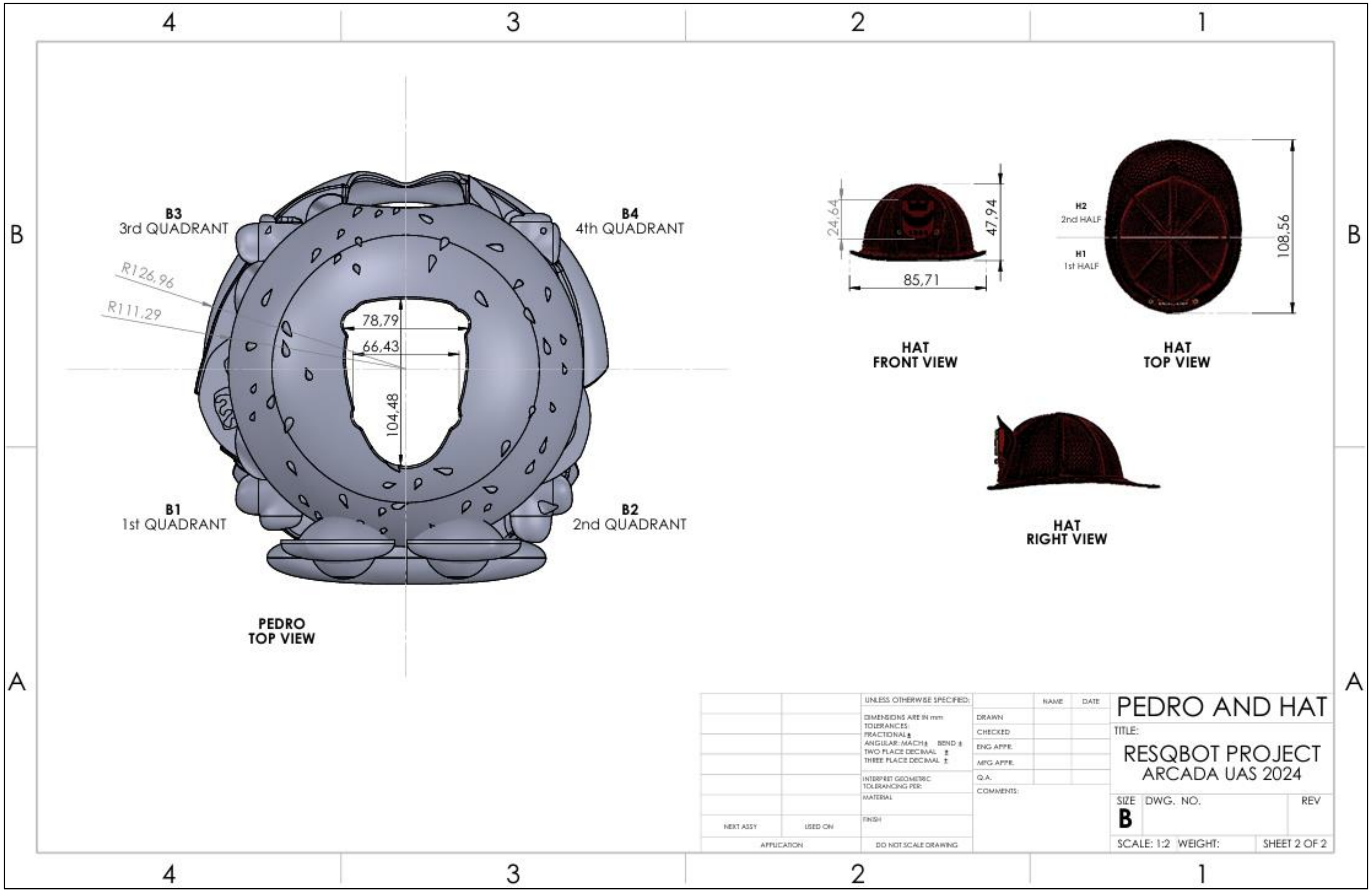
https://github.com/rdeepc/ExploreOpencvDnn

https://github.com/Dodisbeaver/PEDRO

https://github.com/maggielovedd/fyp-rescue-robot

Appendix


PEDRO
RIGHT VIEW
PEDRO
LEFT VIEW
PEDRO
FRONT VIEW
PEDRO
BACK VIEW
19,71
86,47
42,11
28,58
R40,00
157,21
97,00
258,90
217,95
R40,00
PEDRO
TITLE:
RESQBOT PROJECT
ARCADA UAS 2024
SIZE
B
DWG. NO.
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 2
UNLESS OTHERWISE SPECIFIED:
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
A
B
1
2
3
4


B3
3rd QUADRANT
B4
4th QUADRANT
B1
1st QUADRANT
B2
2nd QUADRANT
R126,96
R111,29
78,79
66,43
104,48
PEDRO
TOP VIEW
24,64
47,94
85,71
HAT
FRONT VIEW
H2
2nd HALF
H1
1st HALF
108,56
HAT
TOP VIEW
HAT
RIGHT VIEW
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN mm
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
PEDRO AND HAT
TITLE:
RESQBOT PROJECT
ARCADA UAS 2024
SIZE B
DWG. NO.
REV
SCALE: 1:2
WEIGHT:
SHEET 2 OF 2