FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of robotic manipulators for scalable production lines

Paulo Diogo Carvalho Ribeiro

Integrated Master in Mechanical Engineering

Supervisor: Prof. Dr. Germano Veiga

Second Supervisor: Pedro Malaca

Third Supervisor: Luís Rocha

September 3, 2018

Development of robotic manipulators for scalable production lines

Paulo Diogo Carvalho Ribeiro

Integrated Master in Mechanical Engineering

September 3, 2018

Abstract

Unpredictable market changes constantly force companies to adapt their production capacity (scalability) and improve their overall flexibility. This dissertation was done within the scope of the project H2020 ScalABLE 4.0 that attempts to tackle these issues.

The work done began with the participation in the development of a mobile manipulator and subsequently branched into the independent mechanical and electrical design of a modular robotic system capable of performing two other tasks, namely, the tugging of heavy structures and the lifting and transporting of heavy loads. Furthermore, two software communication interfaces were developed for the system.

The system was divided into modules, according to its main functions, and their mechanical, electrical and software interfaces were defined and designed. This design approach resulted in a flexible solution that could be adapted to a changing production system.

The motion control of mobile robots and the use of Bézier curves as paths were studied, in order to develop a controller for the system. A path generating algorithm and a series of PID based controllers were developed and simulated in the MATLAB environment. The typical go-to-goal and line following controllers were studied and implemented, along with an experimental target following controller.

Acknowledgements

For helping me reach this milestone in my life, I would like to express my deepest gratitude to:

My supervisors Professor Germano Veiga, Pedro Malaca and Luís Rocha for their help, availability and patience.

The team at SARKKIS Robotics, Luís Ruas, Daniel Marques, João Silva and Pedro Tavares for the great work environment and always being ready to help.

My father, for his innumerable teachings.

My mother, for her care.

My sister, for her amazing laptop.

My girlfriend, for enduring me and for her constant support and motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope and motivation

Many manufacturing companies struggle with frequent and unpredictable changes in market demand, along with an ever-increasing need for product customization. To remain competitive, they are forced to increase the scalability (the ability to adapt their production capacity by adding, subtracting or modifying manufacturing resources) and the overall flexibility of their manufacturing systems.

The European project H2020 ScalABLE 4.0 attempts to tackle these issues by developing an open scalable production system framework (OSPS) that enables the optimization and maintenance of production lines 'on the fly', through visualization and virtualization of the line itself.[1]

This dissertation was done within the scope of this project, focusing on the development of a modular robotic system that could be easily integrated into existing manufacturing lines. This system will be used to evaluate the efficiency, effectiveness and generality of the framework.[1]

1.2 Collaborators

The author integrated a multidisciplinary team from two institutions of the project's consortium, INESC-TEC and SARKKIS Robotics. INESC-TEC is a private non-profit research institution dedicated to scientific research and technological development, technology transfer, advanced consulting and training, and pre-incubation of new technology-based companies. SARKKIS is a spin-off company of the University of Coimbra that develops solutions on mechatronics software.

Two other members of the consortium, the manufacturing companies PSA and Simoldes Plásticos, provided two use cases to define the requirements for the robotic system.

[1] http://criis.inesctec.pt/index.php/criis-projects/scalable-4-0/

1.3 Objectives

The desired modular robotic system was a mobile manipulator that could be autonomous or simply movable. As a way to further explore the possibilities of a modular system two other alternative functions were proposed. These were a tugging and a lifting system, to move loads between different locations. Furthermore, two software communication interfaces had to be developed and integrated in the system.

Additionally, the development of a motion controller for the system was proposed. The controller would generate paths based on Bézier curves, to achieve a smooth profile, and make the robot follow them.

The objectives for this dissertation were the following:

- Participation in the development of a mobile manipulator system;
- Further development of the mobile manipulator into a modular robotic system, with two other alternative functions (mechanical and electrical design);
- Stress analysis of the relevant mechanical components;
- Development of two software communication interfaces for the system;
- Study and design of a motion controller for the system.

1.4 Methodology

The development process was approximately structured as follows:

1. Definition of the requirements for each system;
2. Selection and design of the required components;
3. Arrangement of such components into modules;
4. Definition and design of the mechanical and electrical interfaces between the modules;
5. Analytical and numerical stress analysis of the relevant mechanical components;
6. Assembly of two submodules of the system;
7. Development of two software communication interfaces for those submodules;
8. Study of the motion control of mobile robots;
9. Study of Bézier curves as paths for mobile robots;

10. Design of a path generator and a series of motion controllers;
11. Simulation of the designed controllers and selection of the most advantageous.

1.5 Structure

The report is structured as follows:

- **State of the art review** — The technologies currently used to implement the different systems are reviewed along with the relevant safety standards;
- **Problem description** — The requirements of each problem are laid out as a first step towards their solution;
- **Proposed solutions** — The proposed solutions are detailed and discussed along with the methods used to achieve them;
- **Conclusion** — A final review of the most relevant aspects of the project is given, along with a possible direction for future work.

Chapter 2

State of the art review

In this chapter, the technologies currently used to implement the different systems are reviewed along with the relevant safety standards.

2.1 Modularity

The modularity of a given system is the degree to which that system can be divided into cooperative and self-contained subsystems called modules [1]. This allows the creation of complex systems using a set of simpler parts. Modularity can be found in nature, such as in the structure of DNA, and is being increasingly applied with advantages in multiple fields of engineering, especially in the context of design (e.g. software development, product development, machine design, manufacturing systems, architecture, etc.).

The increasing demand for product customization forces companies to take new approaches to product development, such as the development of modular products. A modular product is composed of well defined building blocks called modules that are connected to each other through standard interfaces. To develop these modules designers need to take the components required to achieve the intended functions and carefully arrange them into groups with well defined interfaces between them. However, a good understanding of the ideal module is needed in order to be successful in this process of modularisation.

The ideal module performs a single major function of the product, with the minimum possible number of shared functions with other modules [2]. It should be self-contained, with the ability to be tested separately from the rest of the system [3], as well as to be removed non-destructively from it [4]. It should cooperate with other modules with which it is combined to achieve the required product functions [3].

One important aspect of a modular design are the interfaces between the modules [1]. These are links between them that are used to transfer information, energy, or material [5]. They should be

as simple as possible to maximize the functional flexibility of the product and standardized so that any modules developed in the future can easily be integrated. This means that once specified they must not suffer any changes for the rest of the product development process [6].

Modular product development can have several advantages:

- Simplified development
  - It simplifies the development process of very complex products, as different people can focus on different modules without having to know much about the rest of them [7, 8].
  - If all modules are self-contained then the design of one module can be changed without interfering with the others [9].
  - Designers can wait for more information regarding a particular aspect before taking a decision, without delaying the whole design process [1].
  - Development times can be reduced if new products can use modules that have already been designed [2, 10].
  - Increased product variety can be achieved by simply searching for new combinations and configurations of existing modules [1, 11].
- Lower costs
  - Malfunctions can quickly be corrected and updates easily implemented by replacing one or a small number of modules [10].
  - If the same modules are used across different products, cost effectiveness can be achieved due to economies of scale [12].
  - Each module can be developed and tested separately from the others before being integrated in the final product [10]. This can reduce the time and costs of manufacturing [13] and can have benefits in terms of logistics, for example, when components of very large dimensions are used.
- Higher customization
  - It allows greater customization of the final product, possibly done by the end-user, by combining the desired modules together to achieve the most convenient configuration [7].
  - With interchangeable modules, customers may be able to customize, upgrade, repair and reuse their products.

However, it can also bring some disadvantages:

- Lack of innovation due to the reusability of modules can lead to a static product architecture and excessive similarity between products of the same family [2, 10].
- Lack of internal function sharing prevents product optimization and can lead to an excessive number of components and thus a higher cost [2, 10].
- Can result in higher maintenance costs if a whole module is replaced instead of a single malfunctioning component.
- Modular products can be reverse engineered more easily which can increase competition [2].

2.2 Mobile manipulators

A mobile manipulator is the result of the integration of a robotic manipulator arm in a robotic mobile platform, which results in a manipulator with an horizontal workspace only limited by its surroundings. These systems have been used in the past for applications such as space exploration and military operations but only now are they beginning to be introduced in the industry [14, 15]. Figure 2.1 shows some examples of currently available industrial mobile manipulators.

(a) Fetch by Fetch Robotics

(b) TORU by Magazino

(c) Swift by IAM Robotics

(d) KMR QUANTEC by KUKA

Figure 2.1: Examples of mobile manipulators

2.3 Mobile robots

A mobile robot is a vehicle or platform with more or less complexity and functionalities, that is capable of navigating through a certain environment with minimum human intervention. To achieve this, a typical mobile robot requires a power source, a drive system and a guiding system.

When applied in the industry, mobile robots have specific functions, typically related to material handling tasks, such as: lifting and carrying heavy loads (e.g. forklift robots), tugging movable structures and automatic loading and unloading of crates through integrated conveyors.

2.3.1 Power source

Although there are some mobile robots powered by inductive power transfer through trails on the floor, the vast majority are battery powered. These batteries can be charged or swapped either

manually or in automatic docking/swapping stations, when the opportunity arises or a certain level is reached.

Table 2.1 lists the most common battery chemistries along with their properties.

Table 2.1: A comparison of the most commonly used batteries.[1]

Specifications	Lead Acid	NiCd	NiMH	Li-ion Cobalt	Li-ion Manganese	Li-ion Phosphate
Specific energy (Wh/kg)	30–50	45–80	60–120	150–250	100–150	90–120
Internal resistance	Very Low	Very low	Low	Moderate	Low	Very low
Cycle life (80% DoD)	200–300	1,000	300–500	500–1,000	500–1,000	1,000–2,000
Charge time	8–16h	1–2h	2–4h	2–4h	1–2h	1–2h
Overcharge tolerance	High	Moderate	Low	Low. No trickle charge		
Self-discharge/ month (room temp)	5%	20%	30%	<5% Protection circuit consumes 3%/month		
Cell voltage (nominal)	2V	1.2V	1.2V	3.6V	3.7V	3.2–3.3V
Charge cutoff voltage (V/cell)	2.40 Float 2.25	Full charge detection by voltage signature		4.20 typical Some go to higher V		3.60
Discharge cutoff voltage (V/cell, 1C)	1.75V	1.00V		2.50–3.00V		2.50V
Peak load current Best result	5C 0.2C	20C 1C	5C 0.5C	2C <1C	>30C <10C	>30C <10C
Charge temperature	–20 to 50°C (–4 to 122°F)	0 to 45°C (32 to 113°F)		0 to 45°C (32 to 113°F)		
Discharge temperature	–20 to 50°C (–4 to °F)	–20 to 65°C (–4 to 49°F)		–20 to 60°C (–4 to 140°F)		
Maintenance requirement	3–6 months (toping chg.)	Full discharge every 90 days when in full use		Maintenance-free		
Safety requirements	Thermally stable	Thermally stable, fuse protection		Protection circuit mandatory		
In use since	Late 1800s	1950	1990	1991	1996	1999
Toxicity	Very high	Very high	Low	Low		
Coulombic efficiency	~90%	~70% slow charge ~90% fast charge		99%		
Cost	Low	Moderate		High[1]		

For some mobile robots, batteries with a large weight can actually be desired, to be placed close to the floor and thus lower their center of mass and increase their stability. However, the numerous advantages of modern Li-ion batteries are making them the most common choice nowadays.

2.3.2 Drive system

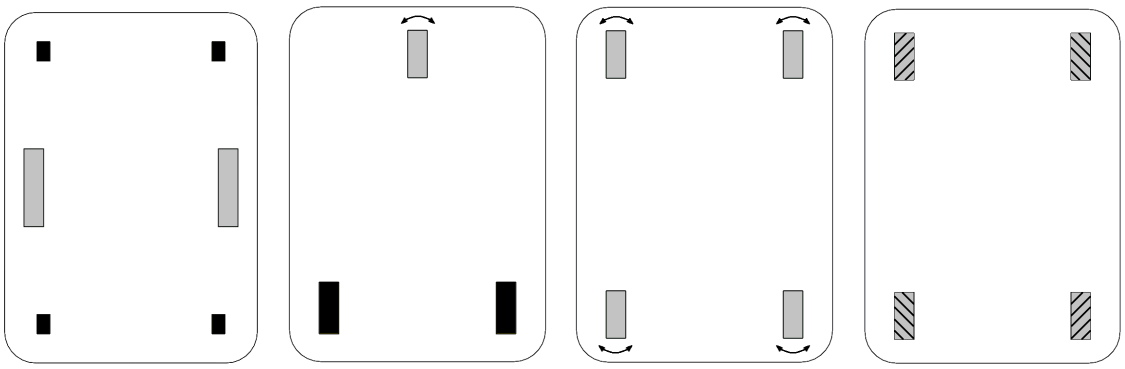
Mobile robots are usually supported by a combination of drive and caster wheels. Drive wheels impose forward/backward and/or steering movement and caster wheels, rigid or swivel casters, serve only as support.

[1] https://batteryuniversity.com/index.php/learn/article/secondary_batteries

The motion of mobile robots is usually achieved by attaching one electric motor (or two if both forward/backward and steering movements are desired) to one or more drive wheels. A gearbox may be used, depending on the selected motors.

The maneuverability of a mobile robot is greatly influenced by its steering mechanism. There are many possible geometries for these mechanisms but the two most common ones are the differential drive and the tricycle.

- The **differential** drive system (2.2a) uses two drive wheels side by side for locomotion and two or more caster wheels for support. This system achieves a curvilinear motion by imposing different velocities to each of the drive wheels. It can rotate around its center by rotating the drive wheels at the same speed and in opposite directions.
- The **tricycle** configuration (2.2b) consists of a single steerable drive wheel and two supporting caster wheels. This system achieves a more accurate motion than the differential drive and is used in most forklift AGVs [2].
- **Omnidirectional** robots are mobile robots with three degrees of freedom in the horizontal plane, making them especially useful in tight space situations [16]. To achieve omnidirectional movement, two approaches are commonly used. One of them simply consists in adding a vertical axis of rotation to each of the four wheels (2.2c). The other method consists in using omnidirectional wheels (2.2d), such as the Omni wheels. However, these types of wheel have a relatively low carrying capacity and are sensitive to dirt and other small particles lying on the ground [16].



(a) (b) (c) (d)

Figure 2.2: Examples of drive systems

[2] https://www.transbotics.com/learning-center/drive-steering

2.3.3 Navigation system

For a mobile robot to be autonomous, it needs some sort of guiding system to keep track of its position relative to its environment and use that information to determine the path it should take. This requires both sensors to obtain information from the environment and a navigation system to process that information and plan and control the movement of the robot.

Odometry consists in using the information from the motor encoders to estimate the robots displacement. However, when the covering of long distances with a high precision is required, this method is not reliable enough by itself and so it is typically used in conjunction with other techniques, the most common being:

- **Wire / Tape** – This solution consists in embedding a wire or laying a tape on the floor, which the robot senses and follows, correcting its position along the way. If modifications need to be made, the tape is more flexible than the wire, however, it is also more susceptible to damage.
- **Laser triangulation** – Similarly to the global positioning system (GPS), a mobile robot can be equipped with a rotating laser emitter and detector that emits a beam of light which is reflected from landmarks located in known positions. By measuring the angles/distances, formed between its reference frame and these landmarks, the robot is then able to triangulate/trilaterate its position.
- **Fiducial markers** - Easily recognizable markers can be placed around a facility so that robots can detect and use them to estimate their relative position. When no markers can be sighted, the robots rely on odometry and inertial measurement units (IMUs) to maintain their path. The most commonly used markers are matrix barcodes [17, 18].
- **Natural features** – A camera or laser can be fitted in a mobile robot and be used to scan a facility along the paths the robot is allowed to follow, while recording unique and easily identifiable features of the environment. These features can then be sensed by the robot during normal operation and be used as references for navigation. No modifications to the facility are needed when implementing this system, simplifying its installation. Furthermore, the robot can plan optimal trajectories 'on the fly', which can be advantageous, especially if obstacle circumvention is desired.

2.4 Robotic manipulators

Robotic manipulators now have a strong presence in industry applications such as welding, pick and place, assembly, etc. These are machines composed by a series of controllable angular or linear joints, rigidly linked to each other. They exist in several different configurations (Figure 2.3), the most common being the serial revolute that emulates the shape of a human arm.

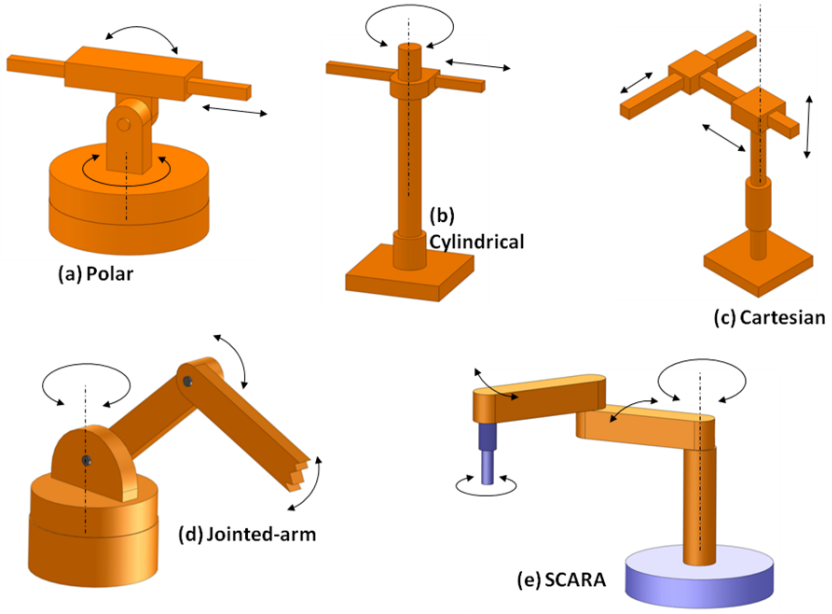


Figure 2.3: Different configurations of robotic manipulators.[3]

These robots provide several benefits when used as a replacement for human workers. They can produce forces that far exceed human strength, allowing them to manipulate heavier objects. They can guarantee a higher precision and reliability in repetitive tasks and are not susceptible to tiredness or boredom, allowing them to maintain a high productivity indefinitely. On the other hand, they are very expensive, require some time for integration and setup, and are still unable to accomplish many complex tasks.

Due to their potential for causing injuries, these robots have typically been used in full isolation from human-beings. This approach can limit the overall productivity of manufacturing systems because while robotic manipulators guarantee a high efficiency in repetitive tasks, they lack the flexibility required for many other tasks that only a human worker can currently provide.

2.4.1 Collaborative robots

With the increasing demand for customization, flexibility is becoming ever more important. This lead to the development of collaborative robot systems. These systems can safely share their workspace with human workers and collaborate with them in many manufacturing tasks [19]. The result is an adjustable balance between the flexibility provided by the workers and the high efficiency inherent to the robots [15]. Work related injuries due to repetitive tasks, such as tendinitis, are also reduced as most of these tasks can be delegated to the robot.

[3] https://nptel.ac.in/courses/112103174/module7/lec5/3.html

According to the sensor manufacturing company, SICK, human-robot interactions can be divided into three categories[4]:

- **Coexistence**, the lowest level, where they work on the same process but in separate workspaces and the workpiece is transferred between them.
- **Cooperation**, where they share one workspace but alternate operations, rarely being inside it concurrently.
- **Collaboration**, where the workspace is shared and both parties are free to carry out operations on the workpiece at the same time.

2.5 Motion control

Mobile robots typically use a motion controller to read the current state or state progression of the robot, compare it to the desired state or progression and compute the control action required to cancel the difference between the two, if it exists. This state is usually its pose (i.e. its position and orientation) and/or its velocity.

When the robot has a significant mass (e.g. an electric car) its dynamics cannot be neglected and a dynamic model and control approach has to be used. When the mass of the robot is low and its center of gravity is not very high, its dynamics can be neglected and a kinematic control approach can be taken.

2.5.1 Unicycle model

To be able to design a controller for a system, first a model of that system has to be defined. A model is an approximation of the real system that allows us to understand how it behaves and how it should be conditioned, in order to move it closer towards the desired state. In the case of small mobile robots, a kinematic model is enough, most of the time. A commonly used kinematic model for mobile robots is the unicycle model, which is depicted in figure 2.4.

[4] https://www.sick.com/us/en/-/human-robot-collaboration/w/human-robot-collaboration/

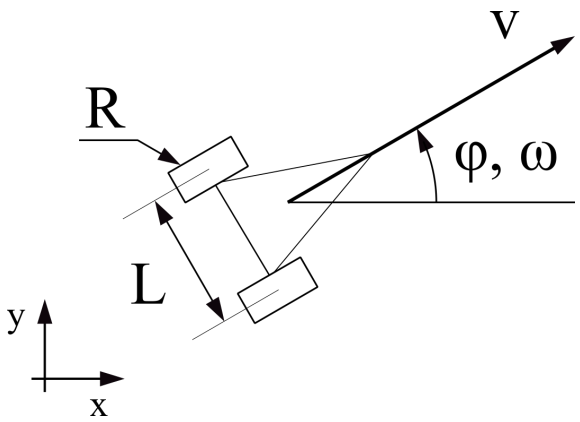


Figure 2.4: The unicycle model for mobile robots

This model is widely used due to its simplicity, which facilitates the controller design, and the fact that it can be applied to a wide range of robots.

Its behaviour is defined by the following equations:

$$\dot{x} = v \cdot \cos(\phi), \quad \dot{y} = v \cdot \sin(\phi), \quad \dot{\phi} = \omega,$$

where v and ω are the imposed linear and angular velocities. Therefore, in this case, the goal of the controller is to compute the velocities required to move the robot to the desired state. After obtaining them, the following equations can be used to calculate the corresponding angular velocities for each of the wheels.

$$\omega_L = \frac{2v - \omega \cdot L}{2R}, \quad \omega_R = \frac{2v + \omega \cdot L}{2R}.$$

One way to keep track of the position of the robot is by monitoring the values output by the encoders on its motors. These increment or decrement as the motor rotates and are commonly referred to as the ticks of the encoder. The change in the position of the robot can be derived from the differences between the subsequent values, for each of the wheels. As a starting point, the distance traveled by one of the wheels is given by:

$$D_i = 2 \cdot \pi \cdot R \cdot \frac{\Delta ticks_i}{N \cdot i},$$

where:

N is the number of ticks per motor revolution and

i is the reduction ratio of the gearbox, if one is being used.

Consequently, the distance traveled by the robot is given by:

$$D_c = \frac{D_l + D_r}{2}.$$

Finally, the estimate for the new state of the robot will be:

$$x' = x + D_c \cdot \cos(\phi), \quad y' = y + D_c \cdot \sin(\phi), \quad \phi' = \phi + \frac{D_r - D_l}{L}.$$

2.5.2 Path following

Path following consists in making the robot follow a given path, usually using a motion controller. When there is also a time constraint, the problem becomes that of trajectory tracking, which will not be addressed in this dissertation.

Before attempting to make the robot follow a path, the path itself must be defined. For some operations, a simple curve such as the quadratic Bézier curve may be enough. This is a parametric curve defined by three points:

$$P(t) = P_1 \cdot (1-t)^2 + P_2 \cdot (1-t) \cdot t + P_3 \cdot t^2, \quad 0 \leq t \leq 1.$$

2.5.3 Controllers

PID controllers are the most commonly used controllers in the industry for process control. They typically compute a control action given by:

$$u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)\, d\tau + K_D \cdot \frac{de(t)}{dt},$$

where e is the error (i.e. the difference between the desired and current values) of a given variable that is to be controlled. Kp, Ki and Kd are the three adjustable gains. However, for many applications, mobile robots can also be controlled using these controllers.

A simple go-to-goal controller can be implemented by taking the angle between the direction of the goal and the heading of the robot, as the error input to a PID controller, which in turn computes the angular velocity required to cancel it.

A line following controller can also be implemented by using an estimate of the distance to the desired path as the error and, similarly, using a PID controller to compute the angular velocity required to cancel it.

Chapter 3

Problem description

In this chapter, the requirements of each problem are laid out as a first step towards their solution.

3.1 Modular robotic system

One of the objectives for this dissertation was to design a modular robotic system with three alternative functions, depending on the modules used. These functions were:

- Mobile manipulator – A mobile manipulator that could be either autonomous or simply transportable by a human worker and that could be easily integrated in existing production lines. The manufacturing companies PSA and Simoldes Plásticos provided a use case each, resulting in the need for two different versions of the system, whose requirements will be further detailed.
- Mobile tugger – A mobile tugging system capable of automatically attaching itself to a movable platform carrying up to 1000 kg and tugging it between given locations.
- Mobile lifter – A mobile lifting platform system aimed at lifting and transporting up to 500 kg loads between different locations.

The minimum height of both the mobile tugger and the mobile lifter had to be as low as possible so that the system could fit below more structures. The height of the mobile manipulator depended on each use case.

A stress analysis had to be performed for the mechanical components more susceptible to failure. For the elements with a simpler design, an analytical approach could be taken. However, for the more complex components, numerical methods had to be used.

The robotic system had to be developed with the following general requirements in mind:

- Modularity – Each main function of the system had to be contained inside a module and the connections between them had to be as simple as possible.
- Flexibility – The system had to be as flexible as possible to allow future modifications and experimentation.
- Standard components – The use of components already available on the market was preferred to designing and manufacturing customized components.

The system would be controlled using the Robot Operating System (ROS), a collection of frameworks for robot software development. However, being an open source project, its functionality was limited in some areas. One such area was the implementation of up-to-date fieldbus networks communication protocols. Because these would be required to communicate with the controllers of the motors responsible for the mobility of the system and the battery management system (BMS) that were selected, an alternative path had to be taken.

On the other hand, the Codesys environment provided high-level and up-to-date packages to easily establish communications with most fieldbus networks. Furthermore, a bridge between the ROS and Codesys applications had already been developed by INESC-TEC. Therefore, two software interfaces capable of serving as intermediaries between the ROS and the aforementioned components had to be developed. An object-oriented programming approach had to be taken to guarantee their modularity, which could be done by designing IEC 61131 function blocks to handle these interactions.

Additionally, a path following motion controller for mobile robots had to be developed for the mobile manipulator system. The paths between the initial and desired final poses would be generated using Bézier curves.

3.2 Mobile manipulators

One of the goals of this dissertation was to participate in the development of a mobile manipulator, to be deployed in existing manufacturing lines, alongside human workers. The manufacturing companies PSA and Simoldes Plásticos provided a use case each and so, two different configurations where needed for this system. Their requirements will be detailed in this section.

Both systems required laser sensors to maintain a monitored safety area around them and allow the manipulator to work at higher speeds when no human workers were nearby.

3.2.1 PSA

In the PSA use case, the mobile manipulator would be deployed in an engine assembly line and had to be autonomous. It would perform the following tasks:

- Inserting a piston into an engine – Two types of similar engines, with two or three pistons, are fed through a conveyor. The manipulator should proceed to introduce the third piston when necessary.
- Coupling a connecting rod to a piston – This task requires two manipulators working in parallel.

The requirements that were considered for the PSA use case were the need to integrate two manipulators, along with their separate controllers, in the same system.

3.2.2 Simoldes Plásticos

For Simoldes Plásticos, the mobile manipulator would be moved manually and so there was no need for it to be autonomous. Any connections to the workstation could also be done by a human worker. The system would perform the following tasks:

- Picking and placing of parts into boxes – The manipulator should detect parts moving on a conveyor and proceed to pick and place them inside a box, carried by the robot itself;
- Assembly of a car door handle – This task was not considered for this dissertation.

The Simoldes use case resulted in the following list of requirements:

1. Part sensor – A sensor was required for the robot to detect the parts moving along the conveyor.
2. Pneumatic circuit – In order to pick up the parts, a pneumatic area gripper had already been chosen due to the larger range of parts that it could grasp. Thus a pneumatic circuit had to be integrated on the system. Because the system for this use case was non-autonomous, the pneumatic connection to an external source could be done by the operator, along with an electrical connection whenever the batteries required charging. Otherwise, an automatic connector would be desired.
3. Part repositioning – When the manipulator picked up a part from the conveyor, it had no way of knowing its exact position and orientation. Therefore, the ability to reposition the parts relative to a reference point known by the robot was needed.
4. Box support – The system also had to carry two boxes with varying dimensions, up to 600x400x300mm.
5. Reachability – Some of the larger parts had to be placed vertically inside the boxes. If only a standard area gripper was used, the wrist of the robot would have to enter them, risking its collision with their walls (figure 3.1).

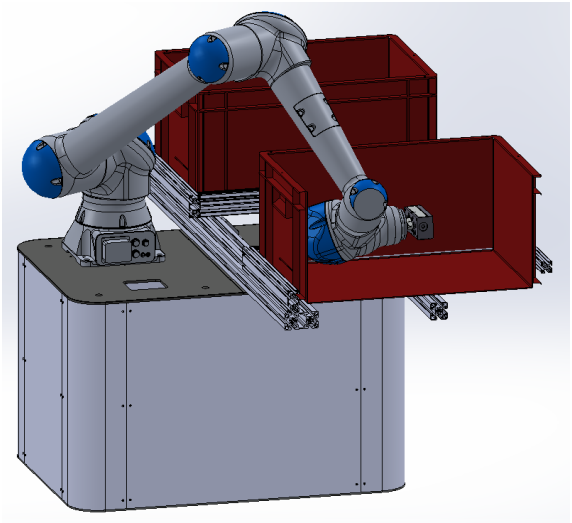


Figure 3.1: Low reachability inside the boxes

3.3 Motion controller

High-level navigation software can be used on mobile robots to map a facility, detect obstacles and plan optimal trajectories in real time. In this case, the motion controller may be integrated in this software, run as a separate software package on the same hardware components or run on different hardware altogether. The latter can increase the safety of the overall system by preventing the robot from moving in a dangerous manner, in case the high-level software malfunctions (e.g. due to an unknown bug).

Most of the time, when the mobile manipulator approaches or leaves its workstation, its path needs to be smooth, without the need to rotate around its center when they finish or start the movement, respectively. The same happens when the mobile tugger or the mobile lifter approach or leave their target load, respectively. This means that their path must be a curve, beginning and ending with the same orientation as the initial and desired final poses of the robot. This can be achieved by defining the path as a Bézier curve based on these poses.

The final objective of this dissertation was the development of a motion controller that would compute paths, based on Bézier curves, between subsequent target poses, provided by the high-level software, and guarantee that the system follows them with sufficient accuracy (i.e. enough to accomplish the required operations).

Chapter 4

Proposed solutions

In this chapter, the proposed solutions are detailed and discussed along with the methods used to achieve them.

4.1 Modularity

To obtain a modular system, each of the desired main functions had to be mapped to a single module. Four main modules were defined and the two different use cases for the mobile manipulator system resulted in two specific configurations for two of the modules.

- Mobile module
  - Autonomous mobile module (AMM)
  - Non-autonomous mobile module (NAMM)
- Manipulator module
  - PSA manipulator module (PMM)
  - Simoldes manipulator module (SMM)
- Tugger module (TM)
- Lifter module (LM)

To obtain the desired end systems they would be combined as follows:

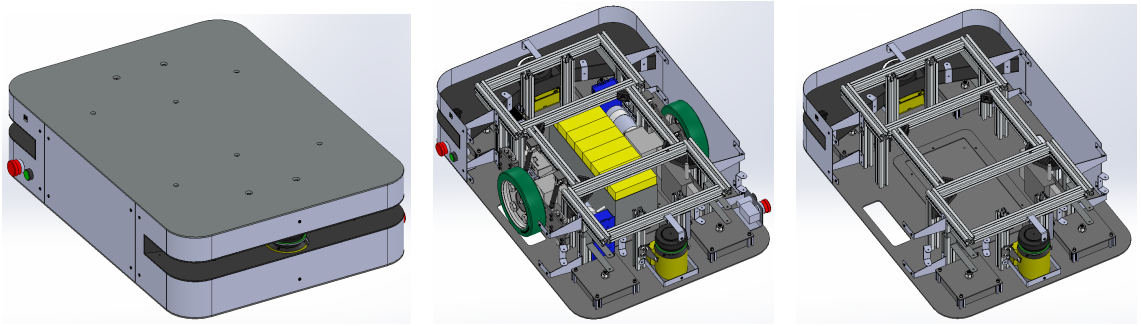
- AMM + PMM $\rightarrow$ PSA autonomous mobile manipulator
- NAMM + SMM $\rightarrow$ Simoldes transportable mobile manipulator

- AMM + TM → Mobile tugger
- AMM + LM → Mobile lifter

Both the mobile and manipulator modules were developed in cooperation with the team of the ScalABLE project while the tugger and lifter modules were developed apart from the project. Two software interfaces were developed, to handle the internal communication of the mobile module.

4.2 Mobile module

The mechanical and electrical design of the mobile module was mostly done by the rest of the team, while the author handled the development of two software interfaces for two of its submodules, along with the assembly of the latter. The rest of the mobile module had to be studied in order to guarantee that the remaining modules would be compatible with it. In this section, the module is briefly described in its two different configurations, shown in figure 4.1.



(a) Mobile module exterior (b) AMM interior (c) NAMM interior

Figure 4.1: The mobile module in its two configurations

Four submodules were defined for this module:

- Actuator submodule – Drive wheel, gearbox, motor, controller and suspension.
- Sensor submodule – Safety lasers.
- Power submodule – Batteries with management system and power converter.
- Signal submodule – Industrial PC, PLC and WAGO modules.
- Structural submodule – Structure and caster wheels.

Despite being manually moved around, in the NAMM configuration, the mobile module would still require the safety lasers, to be able to signal the manipulator to stop all movement, in case someone entered its working area. On the other hand, it would not need the batteries, as the worker

that moved the system could also plug it into an external power source. Therefore, while the AMM configuration included all of the submodules, the NAMM required only the structural, sensor and signal submodules.

The author handled the integration of the actuator and power submodules (figure 4.2). This included their assembly and the development of two software interfaces, which will be presented in section 4.3, for each of them to communicate with the PC.

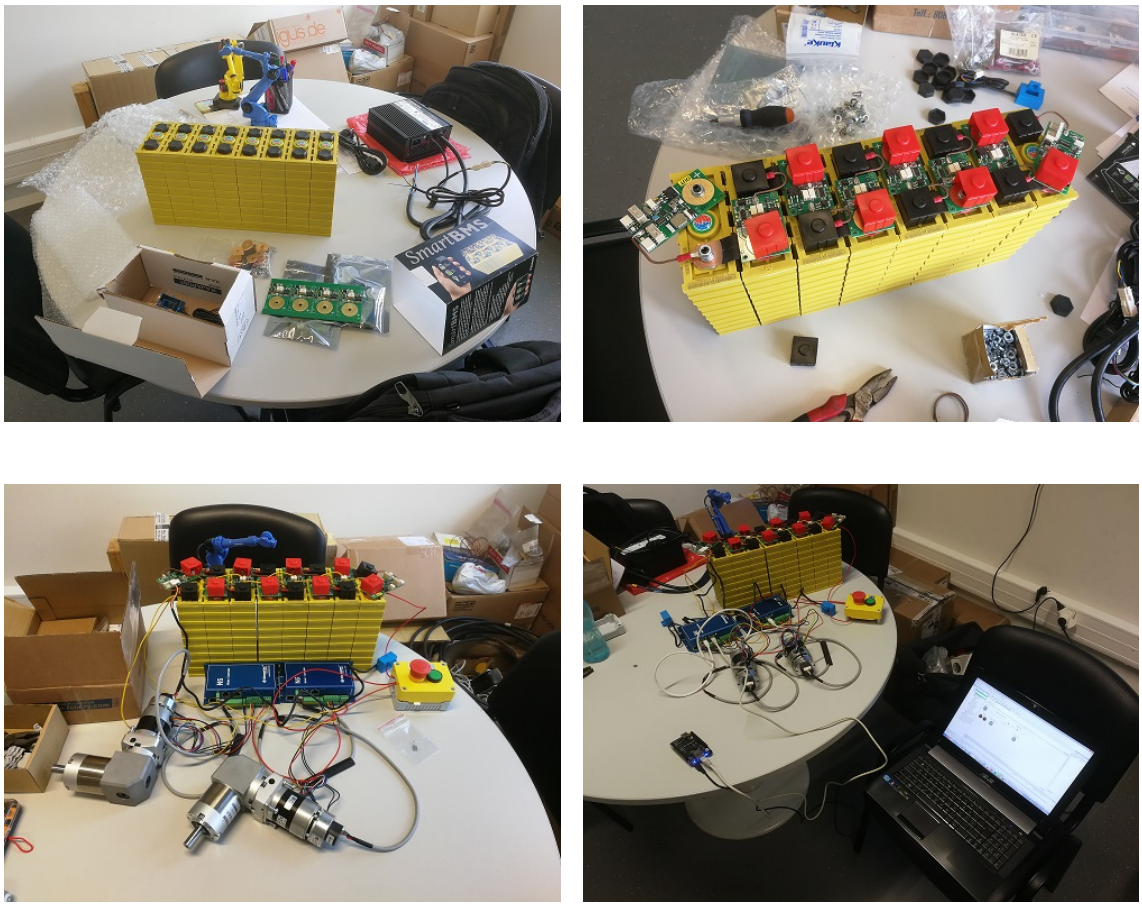


Figure 4.2: Assembly of the submodules and development of the respective software interfaces.

4.2.1 Actuator submodule

The differential drive mechanism was chosen due to its ability to rotate the system around its center and to its simplicity when compared to the omnidirectional systems.

A top speed of 1 m/s and a 2 s velocity ramp were defined, which resulted in an acceleration of 0.5 m/s^2. For a total weight of 500 kg, the required combined force would be of 250 N. The selected drive wheels had a radius of 100 mm, thus each gearbox would have to provide a torque of 12.5

Nm. On the other hand, they would have to rotate at a nominal speed of around 95 rpm, resulting in a power output of 125 W.

Two brushless DC motors were selected with 0.37 Nm at 3500 rpm, along with two gearboxes with a reduction ratio of 35. Due to the limited space available inside the module and the desire to place the power submodule in the middle of the platform, bevel gearboxes were used because of their 90º angle shaft axis shift. The motor controllers that were selected communicated through the EtherCAT fieldbus system, which motivated the development of the first software interface.

A suspension with a pair of springs was coupled to each drive wheel to impose a downwards force on them and guarantee permanent contact with the ground despite any floor irregularities.

The submodule is represented in figure 4.3.

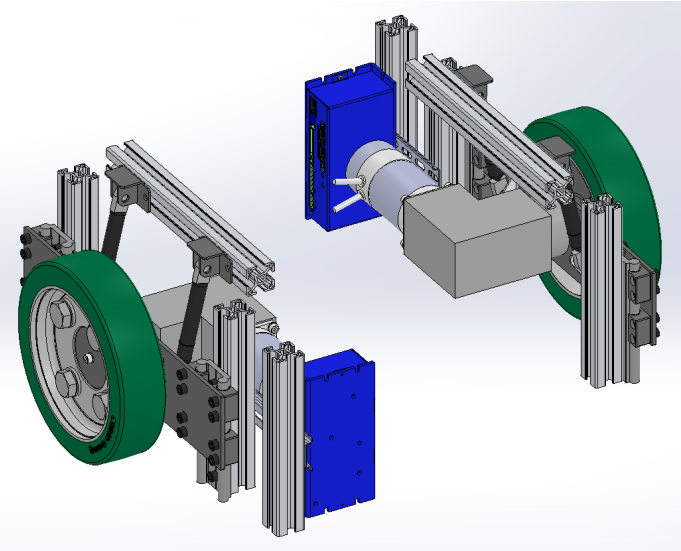


Figure 4.3: Actuator submodule of the mobile module

4.2.2 Sensor submodule

Two safety lasers were used, each centered on the front and back of the robot. These communicated with the PC through a RS422 connection. To guarantee a monitored safety area that surrounded the whole platform, two horizontal gaps were built into the enclosure of the module to maximize their field of vision.

4.2.3 Power submodule

Lithium batteries, more specifically, LiFePO4 batteries, were chosen as the power source due to their higher energy density, longer service life, and stable chemistry. The battery capacity was estimated for an autonomy of 5 h, with the motors consuming half of their nominal current of 8 A. Therefore, eight 40 Ah batteries with nominal and maximum discharge currents of 20A and

400A, respectively, were used and connected in series, obtaining a total nominal voltage of 25.6V. Due to their weight, these were placed in the center of the module, to maximize its stability and the accuracy of its movement.

A battery management system was used to monitor the state of the batteries, which motivated the development of the second software interface.

The motors had a peak current of 24 A each, thus a 50 A circuit breaker and a 100 A fast blow fuse were selected to be placed in series with the battery.

A DC/DC converter was used as a 12V power supply for the PC.

A set of emergency and reset switches was placed on each side of the module.

4.2.4 Signal submodule

An industrial PC was used to run the navigation software.

A WAGO modular I/O system was selected to handle the I/Os between the mobile and the remaining modules.

An Ethernet switch was used to connect together the components that required so, such as the PC and the WAGO system.

An Ethernet connector was placed on the side of the module to allow an external communication to the PC to be made, without having to disassemble the former.

The components of this submodule had to be repositioned in order to accommodate the tugger and lifter modules.

4.2.5 Structural submodule

The module was designed in a common rectangle shape with rounded corners, to eliminate sharp edges and minimize the damage in case of impact. Its internal structure was built from T-slot extruded aluminium profiles due to their convenience for prototyping, by allowing subsequent modifications to be easily made. This frame was then screwed to two steel plates, one below and one above it, forming a rigid structure.

A plastic enclosure was designed to hide and protect the internal components of the module.

Four swivel caster wheels where selected and placed one on each corner.

DIN rails were screwed to the aluminium frame and used to hold the components of the signal submodule.

The structural submodule can be seen in figure 4.4.

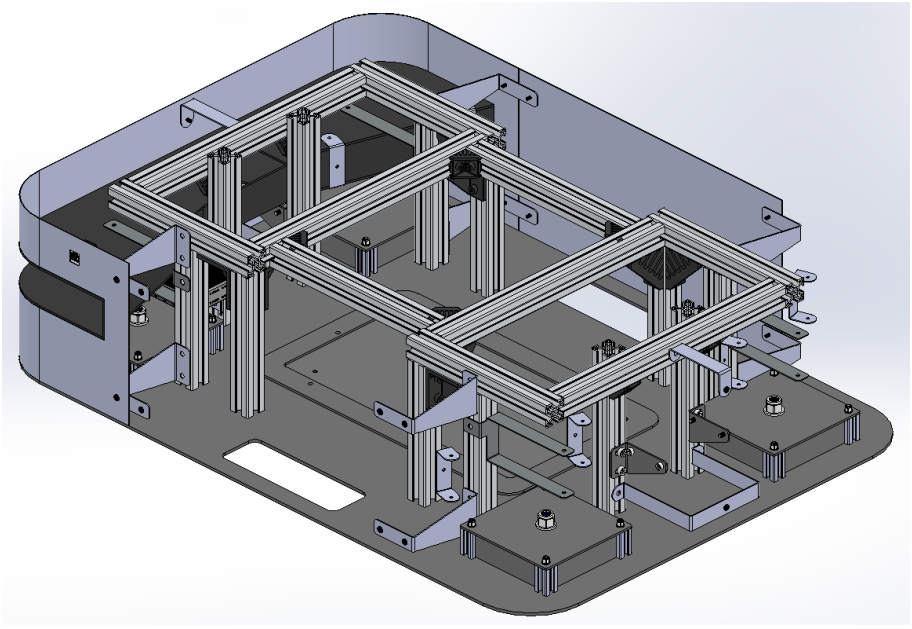


Figure 4.4: Structural submodule of the mobile module

4.3 Software interfaces for the mobile module

Two software interfaces were developed to handle the communication between high-level navigation software developed with the ROS and both the motor controllers and the BMS.

These were developed using the Codesys environment and an object-oriented approach was used, by creating function blocks that encapsulate specific properties and functions of the real world components (i.e. the controllers and the management system). These function blocks can now be used on any Codesys program without having to know and understand the code that is inside of them, allowing for a modular programming strategy to be employed.

The ROS was Linux based but the Codesys environment required either a Microsoft Windows based machine or some specific lower level systems, such as the Raspberry Pi or the BeagleBone Black (BBB). The latter was chosen due to its higher reliability for industrial applications.

4.3.1 Motor controllers

A software interface between the ROS and the motor controllers was implemented. This interface abstracted the communication with the latter by conveying encoder readings, and other relevant status information, from the controllers to the ROS, and velocity setpoints from the ROS to the controllers. This increased the flexibility of the platform because if the communication protocol with the motors ever has to be changed, no software development will be required.

The controllers chosen were the Nanotec N5-2-1, which were prepared for the EtherCAT communication protocol. The advantage of the Codesys environment was obvious when attempting to setup this communication. A readily available "EtherCAT Master" block, was added, requiring only the selection of the Ethernet interface of the BBB. The "EtherCAT Slave" blocks were provided by Nanotec and the only requirements were the selection and mapping of the desired inputs and outputs to the appropriate variables (figure 4.5).

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.PLC_PRG.driver1.submode		Motor drive submode select	~~%QD0~~	UDINT		Motor drive submode select
Application.PLC_PRG.driver1.operation		Modes of operation	~~%QB4~~	SINT		Modes of operation
		Target Position	%QD2	DINT		Target Position
Application.PLC_PRG.driver1.targetVelocity		Target velocity	~~%QD3~~	DINT		Target velocity
Application.PLC_PRG.driver1.controlWord		Controlword	~~%QW8~~	UINT		Controlword
Application.PLC_PRG.driver1.currentOperation		Modes of operation display	~~%IB0~~	SINT		Modes of operation display
Application.PLC_PRG.driver1.currentPosition		Position actual value	~~%ID1~~	DINT		Position actual value
Application.PLC_PRG.driver1.currentVelocity		Velocity actual value	~~%ID2~~	DINT		Velocity actual value
Application.PLC_PRG.driver1.statusWord		Statusword	~~%IW6~~	UINT		Statusword

Figure 4.5: Codesys EtherCAT variable mapping

A function block was developed to handle the interactions with the controllers, based on the information available on their manual. To switch the controllers to an operational state, a state machine (4.6) had to be run through, as defined in the CANopen standard CiA 402. This was done by setting a "control word", in the controller, to subsequent values and checking for the state transition by reading a "status word", each step of the way. The final "Operation enabled" state imposed torque on the motor, as well as the movement pattern defined by the selected mode of operation. If a fault was detected by the controller, it would stop the motor and go into a safe fault state, until a reset was issued, after which the controller would go back to the "Switch on disabled" state. If a quick stop was issued, the controller would go back to that same state.

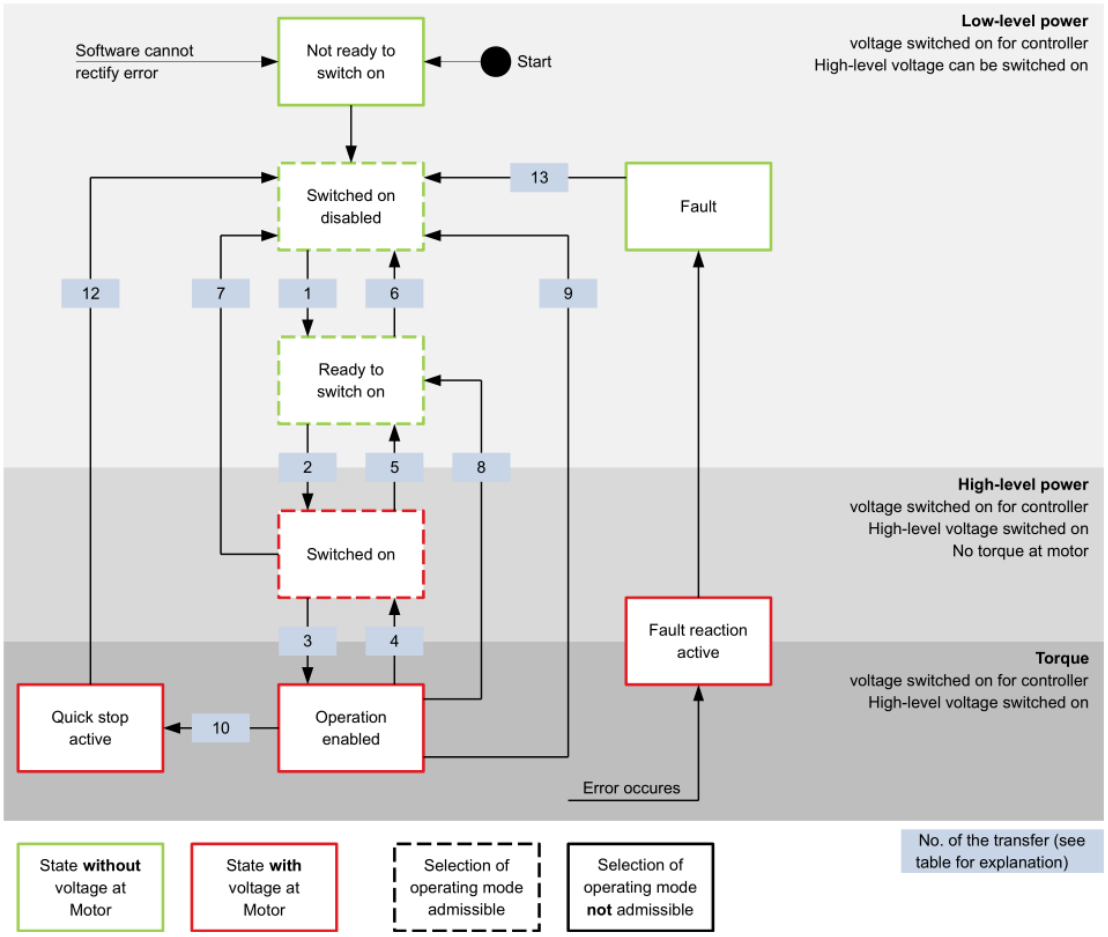


Figure 4.6: CiA 402 power state machine

The interface was integrated into a function block that abstracted both this process and the direct interaction with the parameters of the controller which was done through bitwise operations and required consulting the manual. Therefore, five public methods, whose source codes can be found on appendix B, where implemented to allow the user to control the motor:

- setVelocity(VEL) – Set the target velocity of the motor to VEL.
- stop() – Stop the motor and transition to the “Ready to switch on” state.
- quickStop() – Stop the motor through a quick stop.
- getEncoderTicks() – Read the value output by the encoder.
- faultReset() – Attempt to reset the controller back from a fault state.

The main body of the function block, which can also be found in appendix B, detected errors and handled the requests to start the movement of the motor by transitioning up until the “Enable

operation" state on the state machine. The function block had only one output that indicated if the controller was in a fault state.

4.3.2 Battery management system

The state of charge of the batteries of the mobile module had to be monitored to avoid running out of power during a task. Instead, the platform would automatically go to a charging station when required and available. The instantaneous current consumption also had to be monitored in order to detect abnormally high currents that could reveal system malfunctions and to avoid draining the batteries too quickly. Other parameters such as individual cell voltage levels and daily energy consumption were also relevant and desired.

The manufacturer of the batteries that were selected for the mobile module provided a battery management system (BMS) that could be used to manually inspect these and other parameters, with the use of a smartphone app, via Bluetooth Low Energy. The system also had an UART serial connection available, meant to be connected to another module, sold separately, that could then communicate via CAN bus with other devices. However, there was a possibility that this serial connection could also be read directly by the BBB, which could then decode the data and send it to the ROS. Both an electrical and software interface were prototyped to test this possibility.

The manufacturer provided the electrical schematic of a circuit (figure 4.7), based on an opto-coupler, that both isolated and inverted the signal output by the management system, returning a signal that could be read by the BBB. A document with the communication protocol used was also provided, containing the communication settings and the structure of the string of data that was output by the system (appendix A).

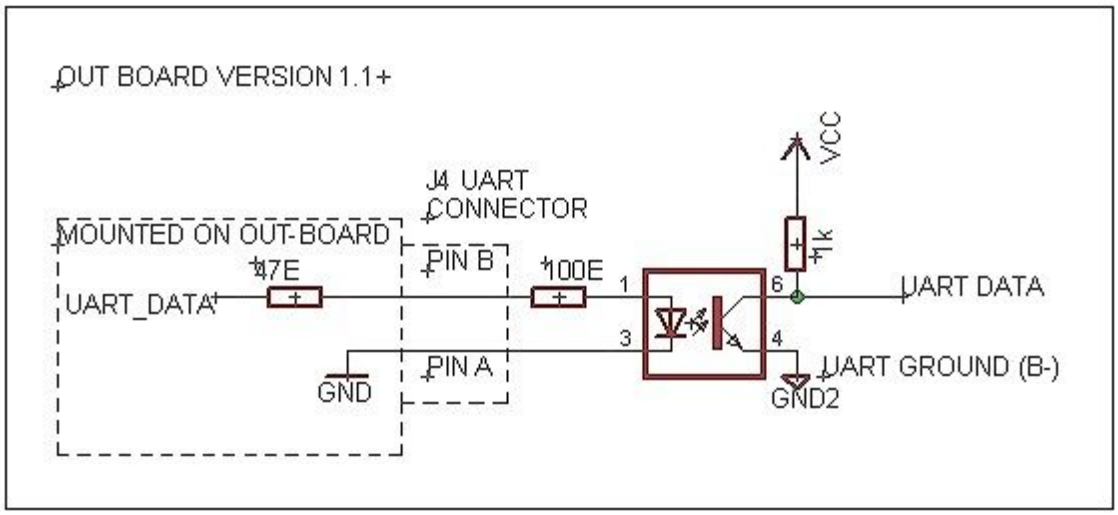


Figure 4.7: Electrical interface provided by the manufacturer

Again, an interface was implemented as a function block to read these parameters through the serial port of the BBB and provide them through a Modbus slave. A state machine was used to alternate the function block between three states:

- HALT – Does nothing.
- OPEN – Opens the serial connection of the BBB.
- LISTEN – Reads incoming data.

The SysCom library was used to open and listen to the serial port. The communication settings provided by the manufacturer were input as the parameters of the SysComOpen(2) function which returned a handle to the opened port. This handle was used, along with the address and size of a data buffer, as the parameters of the SysComRead function to read and store the incoming data.

A data structure representing the battery and containing all the parameters read was created, which the function block would output to the user.

Initially, the number of received bytes was not being checked and the result was that only about 10% of the received messages were passing the checksum. Eventually it was ascertained that this was due to the small cycle interval of the BBB, which was setup at 4 ms. The program would read the data before the string was complete and thus the checksum was incorrect most of the time and the data corrupted.

To prevent this from happening, the program had to be modified. The received bytes were successively appended to the data buffer until the full-sized string was obtained. This was done by saving the number of bytes read and using it to increment the address of the data buffer, each time more data was read. When the complete string was obtained, the checksum was verified and if it was correct, the battery parameters were extracted from the string and the data structure was updated through the private method, update(). After implementing this modification, the checksum was correct almost 100% of the times. Due to its large size, the main code of the function block can be seen in appendix C.

Two methods, start() and stop(), were used to put the system in the OPEN and HALT state, respectively. The update() method (figure 4.8) used another private method, parseStr(), to extract individual values from the buffer string, according to the structure specified on the protocol. The parseStr() method required four parameters:

- firstByte – The first byte of the value (from 1 to 58, as per the protocol);
- length – The number of bytes that make up the value;
- scaleFactor – A scaling factor used to obtain the real value from an integer;
- signByte – A boolean flag that indicated if the value contained a sign byte.

```
METHOD PRIVATE update

battery.voltTotal := parseStr(firstByte := 1, length := 3,
    scaleFactor := 0.005, signByte := FALSE);
battery.currentIn := parseStr(5, 2, 0.125, TRUE);
...
battery.voltBypassSetting := parseStr(56, 2, 0.005, FALSE);
battery.statusByte := str[30];
battery.socNotCalibrated := (battery.statusByte AND 128) = 128;
battery.exceedTempMax := (battery.statusByte AND 64) = 64;
...
```

Figure 4.8: Part of the update() method

The parseStr() method (figure 4.9) was designed to start at byte 1, following the numbering convention from the table in appendix A. First, it computes the position of the last byte, based on the first byte and length that were given as input. Then it converts the data bytes into an integer number by multiplying each byte by 256 to the power of its position, according to the big-endian format. It then multiplies the obtained integer by a scaling factor specified in the aforementioned table, to get the real value. And finally, it inverts the value if there is a negative sign byte preceding the first byte.

```
METHOD PRIVATE parseStr : REAL
VAR_INPUT
    firstByte, length   : USINT;
    scaleFactor         : REAL;
    signByte            : BOOL := FALSE;
END_VAR
VAR
    ii, last            : USINT;
END_VAR

parseStr := 0;
last := firstByte + length - 2;
FOR ii := firstByte - 1 TO last DO
    parseStr := parseStr + str[ii] * LREAL_TO_REAL(EXPT(256,
        last - ii));
END_FOR
parseStr := parseStr * scaleFactor;
IF signByte AND str[firstByte - 2] = 45 THEN
    parseStr := -parseStr;
END_IF
```

Figure 4.9: The parseStr() method

The electronic diagram provided, shown in figure 4.7, was used to assemble a circuit on a breadboard and test the interface.

After successfully being able to read the state of the batteries with the BBB, a C# based application was developed on Visual Studio to retrieve this information from the BBB and display it on a Windows based tablet. The Modbus TCP/IP protocol was used to implement the communication.

On the BBB side, a Modbus slave was setup, leveraging the Ethernet and Modbus communication interfaces available on the Codesys system. On the appplication, the NModbus library was used to implement a Modbus master on top of a TCP/IP client and connect to the slave to retrieve the battery information.

4.4 Manipulator module

The manipulator module was developed in conjunction with the team of the ScalABLE project. The author handled the design of the structural components of the module and the development of a series of submodules and features to solve problems related to the Simoldes use case. A simple

electrical interface was also designed, to connect to the batteries and communicate with the PC of the mobile module. This section describes the manipulator module in its different configurations.

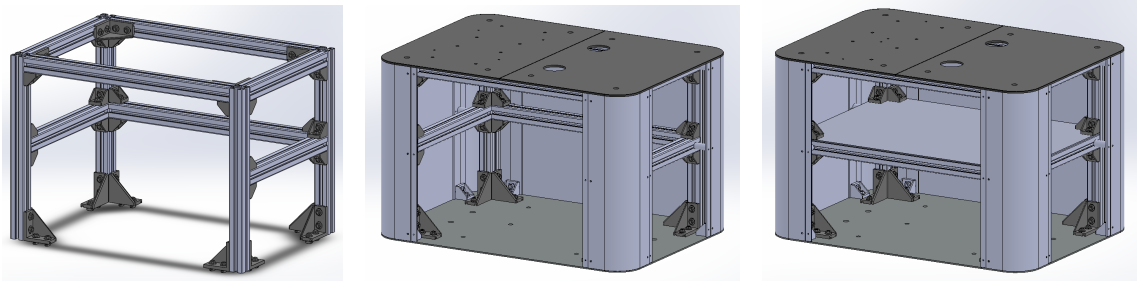
4.4.1 Structural components

Similarly to the mobile module, the internal structure of the manipulator module, shown in figure 4.10a, was built from T-slot extruded aluminium profiles. It was fixed to a lower steel plate that replaced the top plate of the mobile module, when the two were assembled together. The profiles had a square shape with 45 mm of side length and their selection was based on existing systems. Thus no structural analysis was done.

Two top plates were designed, one to support additional submodules and another to support the manipulator. The latter served as the mechanical interface between the module and the actual manipulator and was specifically designed for the desired model.

A plastic cover was designed to hide and protect the internal components of the module.

Three horizontal reinforcement profiles were used to increase the rigidity of the overall structure. By adding one last horizontal reinforcement profile along with another plate, a second level could be easily made (figure 4.10c). This was required for the PSA use case, to accommodate a second controller. It could also be useful to store extra batteries if the robot were to be deployed in a large facility, where it would need to cover long distances, or if there were a large number of robots for a small number of charging stations.



(a) (b) (c)

Figure 4.10: Structural components of the manipulator module

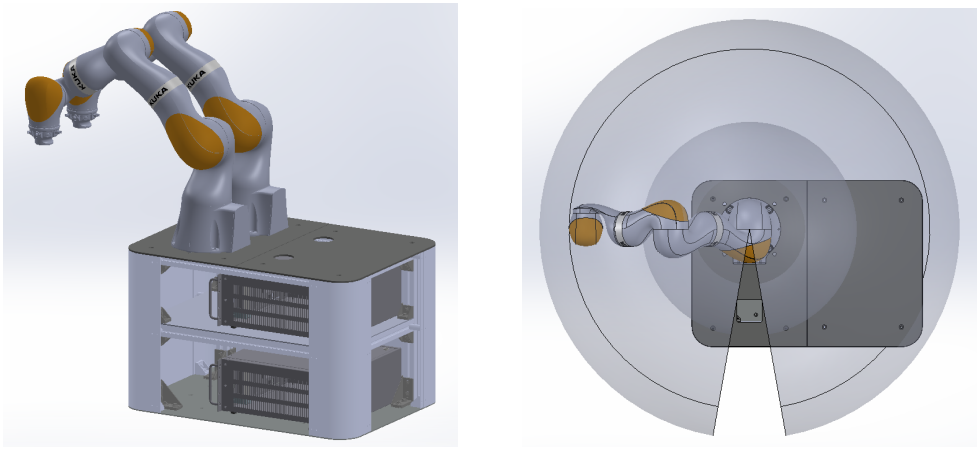
4.4.2 Electrical components

An electrical interface was designed into the alternative plate. It contained an Ethernet connector to allow the robot controllers to communicate with the PC inside the mobile module, as well as some electrical terminals to connect to the batteries and to the 12V converter.

Both the KUKA and the Yaskawa robot controllers had a rated power input of 1 kVA. Therefore, a 2 and 1 kW power inverters were selected for the PSA and the Simoldes Plásticos use cases, respectively.

4.4.3 PSA

The main particularity of the manipulator module in the PSA use case was the capability to integrate two manipulators. Depending on the operation, one or two KUKA LBR iiwa 14 R820 collaborative robotic manipulators would be used. These would be fixed on the front top plate and its cables would be guided through two holes on the back top plate, as seen in figure 4.11a.



(a) (b)

Figure 4.11: The PSA mobile manipulator in its two configurations.

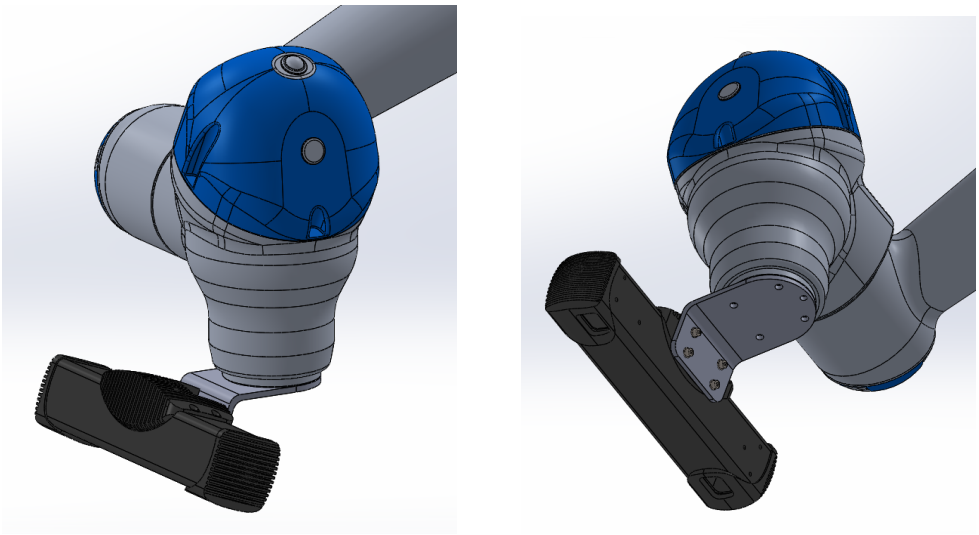
Because the movement of the base joint of these manipulators was limited to +/-170^{o}, the robot would have limited access to the back plate area. An alternative configuration was made possible, when using a single manipulator, by adding two more guiding pin holes and another hole for the cables to go through. This positioned the base of the manipulator at a 90^{o} angle, relatively to the first configuration, giving it full access to the back top plate, as well as freeing up space in that plate (figure 4.11b).

4.4.4 Simoldes Plásticos

For the Simoldes use case, a Yaskawa HC10 collaborative robotic manipulator was used in the same configuration as in the PSA module with a single manipulator. A handle was designed to be attached to the top of the module and facilitate its maneuverability by the worker.

The solutions found for the previously stated requirements were the following:

1. Part sensor – A structured light 3D scanner was selected to detect the parts moving on the conveyor. This sensor was coupled to the wrist of the manipulator using a custom tool interface that could hold the necessary tool, simultaneously (figure 4.12).

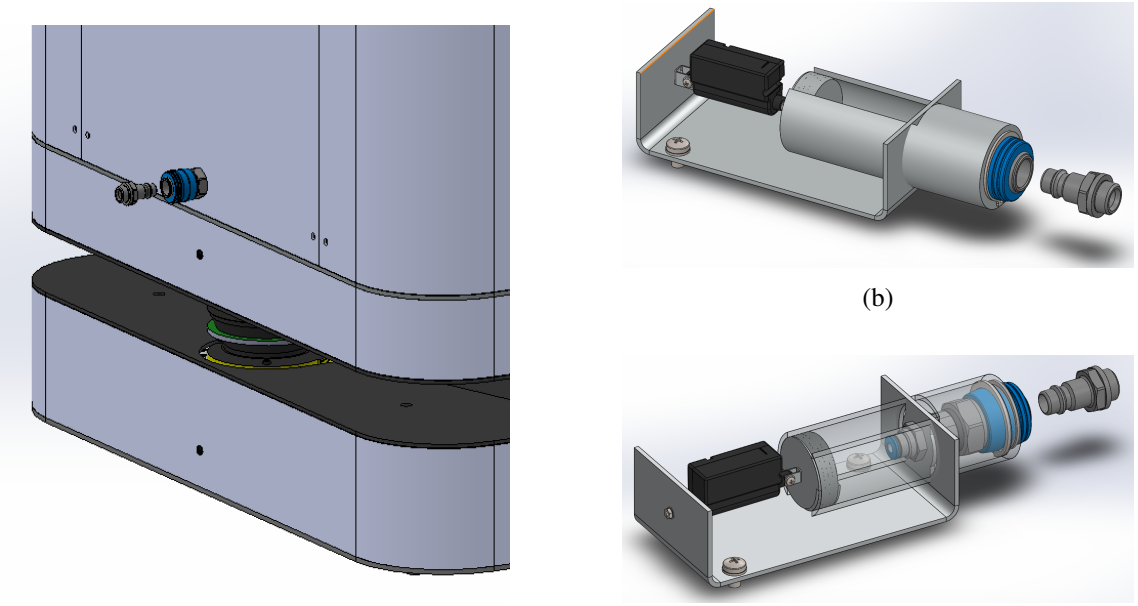


(a) (b)

Figure 4.12: 3D scanner and manipulator wrist interface

2. Pneumatic circuit – A quick coupling pneumatic connector was placed on the front of the bottom plate of the module (figure 4.13a) and a pneumatic circuit would be guided through the structure up until the robot plate.

   Despite not being needed for the current state of the factory, an automatic connection mechanism was designed using the same quick coupling pneumatic connector and a small linear actuator (figures 4.13b and 4.13c). This submodule would be fixed to the bottom plate of the robot module and would allow the system to automatically connect to a pneumatic source when it reached a workstation with a compatible connector. For this use case, however, the mobile module was non-autonomous and thus this was not possible.



(a) (b) (c)

Figure 4.13: The quick coupling pneumatic connector and the automatic connector mechanism.

3. Part repositioning – A submodule to reposition the parts that required so was designed to be fitted between the boxes (figure 4.14). The robot could drop the parts there, which would then slide down to a known position, due to the shape of the structure and the force of gravity. The robot would then repick them and, knowing their relative position, could proceed to place them inside the box, appropriately.

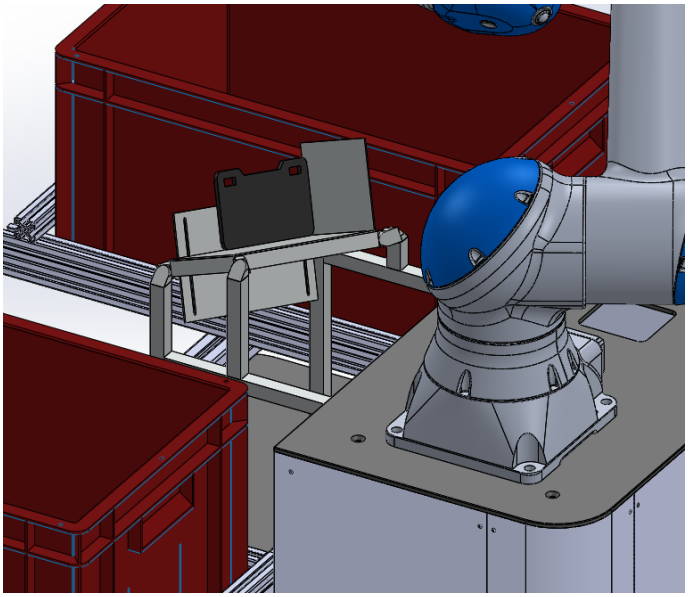


Figure 4.14: An adaptable submodule to reposition the picked parts.

4. Box support – To support the two boxes, an easily adaptable structure was designed using T slot extruded aluminium profiles, which is shown in figure 4.15.

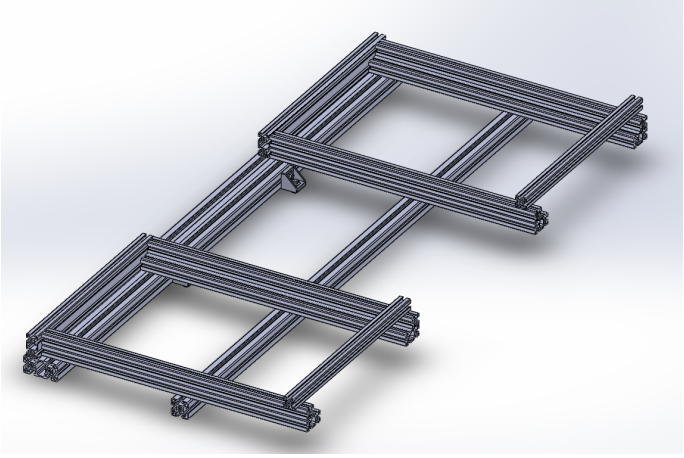


Figure 4.15: The adaptable support for the boxes.

5. Reachability – To increase the reachability of the manipulator inside the boxes, four different approaches were studied:

   (a) Manipulator elevation.

   One solution would be to elevate the base of the manipulator relatively to the boxes. This was accomplished by designing an additional pedestal to be fixed on top of the base structure of the module as shown in figure 4.16.

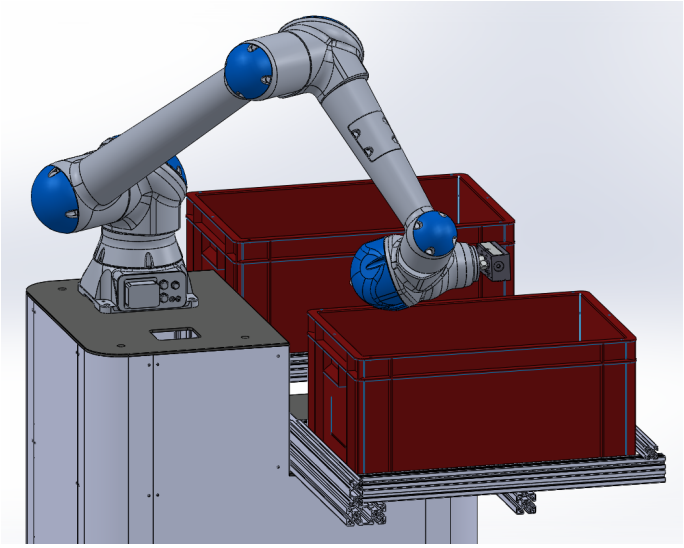


Figure 4.16: The pedestal used to elevate the manipulator.

   (b) The use of two separate tools.

   If a mechanical finger gripper was used to store the last parts of the box, the robot could avoid the walls. Because an area gripper had to be used to pick up all the

parts from the conveyor, the robot had to change tools after placing the part on the repositioning station. In addition to requiring a tool changing station, this procedure would substantially delay the rest of the operation. One solution would be to create a buffer to store the last, problematic, parts and switch tools only when all of these parts were already in the platform and in a known position. However, this would still slow down the operation considerably and so this solution was excluded.

(c) The use of a custom tool interface to hold both tools in the manipulator, simultaneously.

By using a tool interface capable of holding both tools at the same time (figure 4.17, the tool changing procedure could be reduced to a 180^{o} rotation of the robot's wrist, which would greatly reduce the delay introduced in the operation. However, the shape of the end effector would increase the chances of its collision with other objects, such as the walls of the boxes.

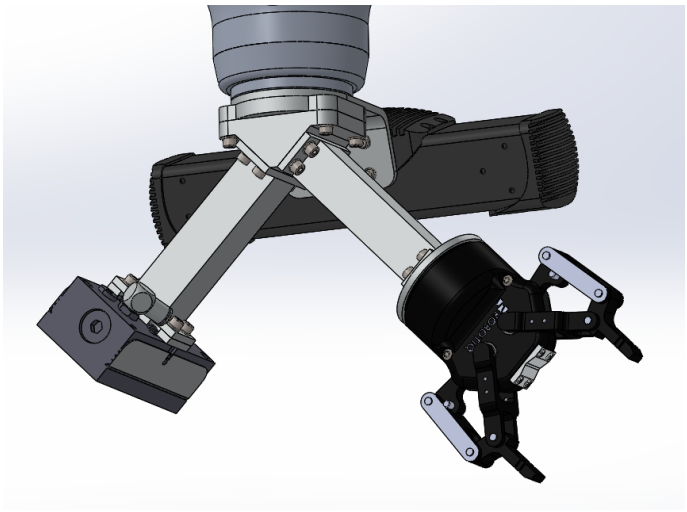
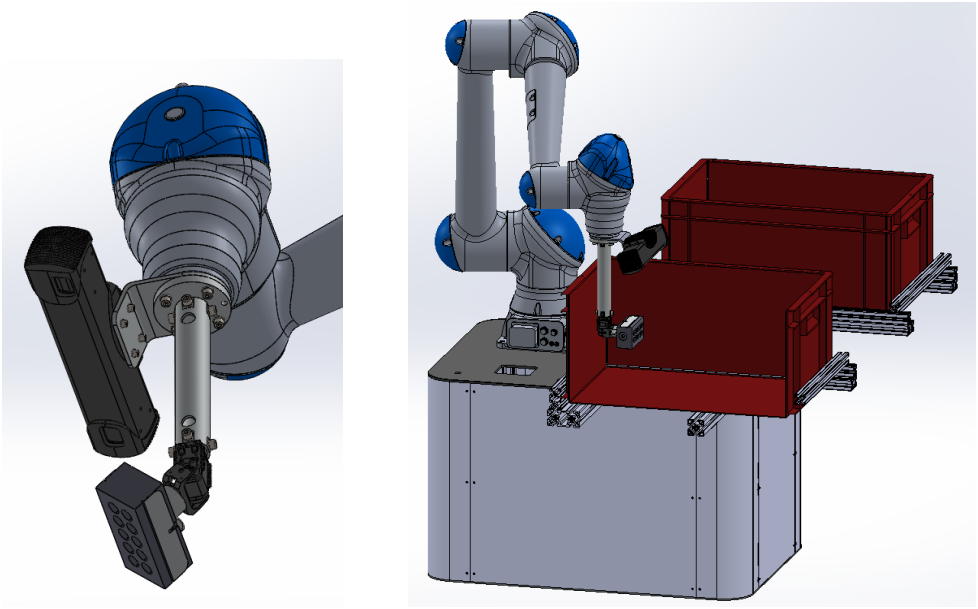


Figure 4.17: Custom interface to hold both tools

(d) The use of an extended custom tool interface, with the area gripper at the end.

By designing a tool interface with an extension, holding the area gripper at the end, using a small servo actuator to grant an additional degree of freedom (figure 4.18), the robot's wrist would barely need to enter the box at all, eliminating the possibility of collision while at the same time increasing the robot's movement flexibility. This extension of the position of the end effector was only possible because the parts were all made of plastic, and thus were light enough not to exceed the maximum payload of the robot.



(a) (b)

Figure 4.18: Custom interface with an extension to hold the area gripper

The last solution was considered the most advantageous as it would not delay the operation and would not increase the possibility of collision as much as the second alternative. Furthermore, increasing the height of the manipulator would decrease the stability of the system.

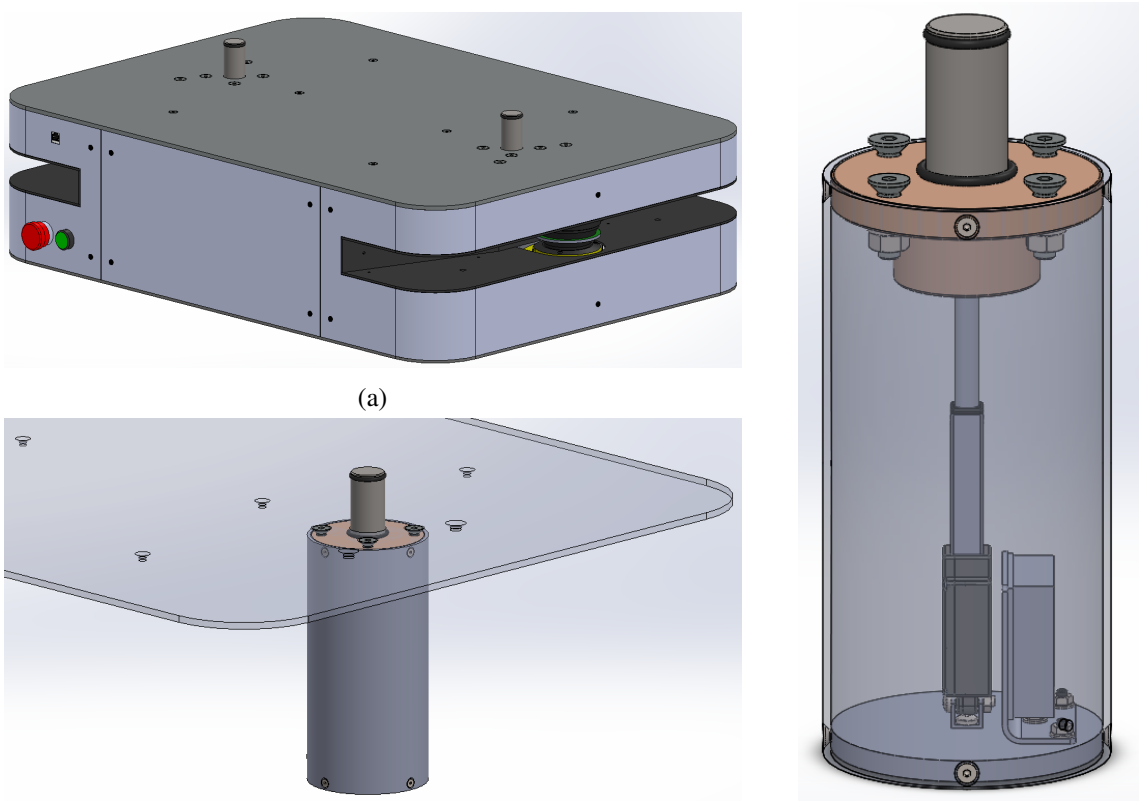
4.5 Tugger module

The tugger module was designed independently of the ScalABLE project. This section will detail the development process of this module.

Two different design mechanisms were studied:

- Below – In this case, one or two linearly actuated pins are fitted inside the mobile module. These are retracted when the robot is moving without the structure and are inserted into compatible slots when the robot gets below the structure, coupling the two. When two pins are used, the orientation of the towed structure can be precisely controlled.
- Back – In this approach, the module couples itself to the structure from behind, near the floor level, and pulls it, with the help of the mobile module. This approach is preferred when the mobile module does not fit below the structure but is substantially more complex to implement. Another disadvantage is that the back laser of the mobile module will be mostly covered, which can hinder safety, depending on the facility.

Due to its simplicity, the first approach was chosen for the design of the module, which can be seen in figure 4.19b.



(a) (b) (c)

Figure 4.19: The mobile tugger system and the tugger module in detail.

Two round medium carbon steel shafts with 30 mm of diameter were used for the pins, each moved by a small linear actuator. The selected actuators were the Actuonix L16-P with a 50 mm stroke and a maximum force of 50 N, which would be enough, as the pins would weigh around 2 kg. The manufacturer also provided a controller that allowed to control the position of the actuator through a 4-20 mA current loop. The WAGO 750-554 module, with two analog current outputs, was selected to provide this signal.

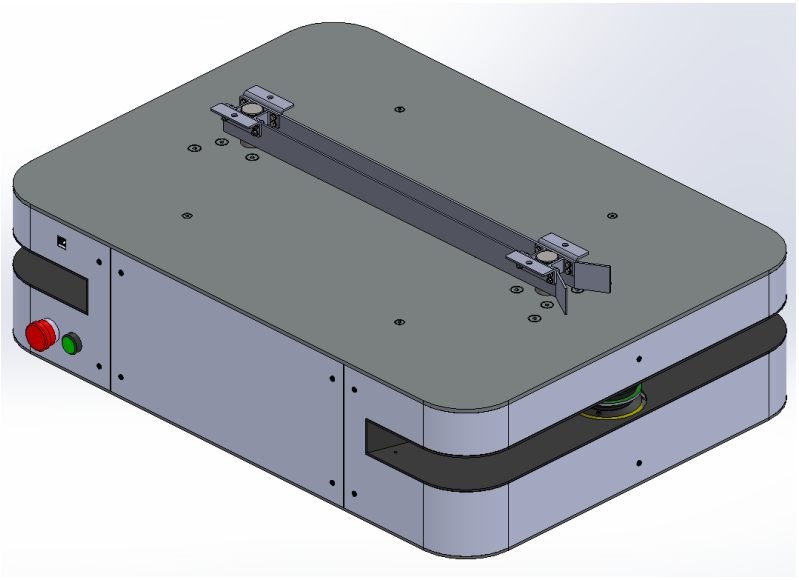
The mechanism was integrated into the mobile module using an alternative top plate, as it was done for the MM , with a thickness of 6 mm. The plate included a hole for each of the pins to go through and a bronze bushing, attached to it by four M8 screws, for the pins to slide through, minimizing wear. Some kind of seal, such as an o-ring, would have to be used to prevent debris from entering the bushing. To accommodate it, a groove would have to be made in either the bushing or the top plate. Because the plate would not be subjected to stresses around the pin hole, it was concluded that it would be the better option.

An aluminium tube and disc were used to hold the controller and the other side of the actuator. These also served as a cover that would be fixed to the bushing and would avoid having to make any connections to the bottom plate of the mobile module when installing the module.

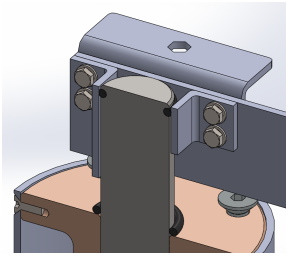
Because the bronze bushing would be subjected to alternating bending moments it would be susceptible to failure by fatigue. This meant that the edge between the body of the bushing and its flange had to be filleted to minimize the stress concentration. Furthermore, its size was limited to be able to fit inside the mobile module. Due to these constraints, an appropriate bushing could not be found on the market and so it was designed, to be machined.

The possibility of thermal expansion or contraction of the system, relatively to the interface on the structure to be towed, could cause a misalignment of the pins with the slots. The thermal expansion coefficients of steel and aluminium are 12 and $25 \cdot 10^{-6}$ m/mK, approximately. However, in plastics, it can go up to $120 \cdot 10^{-6}$ m/mK. For a 20 °C temperature difference and a distance of 0.6 m, the dilation would be of 1.44 mm. To allow for a plastic interface to be used, this would have to be taken into account by leaving a clearance of around 0.75 mm between the pins and the walls of the slots.

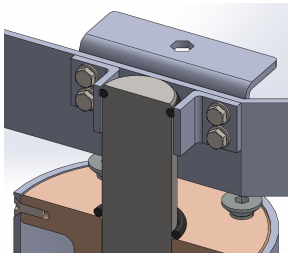
An interface was designed, as an example, to be attached to the towed structure (figure 4.20). An o-ring was placed on the pin to prevent vibrations due to the aforementioned clearance. For this interface, to couple itself to the structure, the system would first lift the pins to half their stroke and advance until the front pin hit the front of the interface. Then the system would fully extend the pins and the connection would be made.



(a) Whole interface



(b) Front in detail



(c) Back in detail

Figure 4.20: An example interface to be attached to the load to be moved.

4.5.1 Stress verification

The parts susceptible to failure would be the pin, the bushing, the plate and the screws.

The solicitations on the pin would result from the inertia of the load and the critical situation would be when a system with a single pin configuration was moving at its maximum speed and had to do an emergency stop. A top speed of 1 m/s and a maximum stopping distance of 0.5 m with constant deceleration were assumed. Thus, this deceleration would be given by:

$$a = -\frac{\dot{y}_{max}}{\Delta t_s}. \tag{4.1}$$

and the average velocity of the system depended on delta Δt_s by:

$$\begin{aligned} \dot{y}_{avg} = \frac{\Delta x_s}{\Delta t_s} = \frac{\dot{y}_{max}}{2} \\ \Delta t_s = \frac{2 \cdot \Delta x_s}{\dot{y}_{max}}. \end{aligned} \tag{4.2}$$

Replacing 4.2 in 4.1 we get:

$$a = -\frac{\dot{y}_{max}^2}{2\Delta x_s}. \tag{4.3}$$

For the assumed values we get an acceleration of 1 $\mathrm{m/s^2}$ and for a load m of 1000 kg, the force on the pin would be $F = ma = 1000$ N.

Assuming that the pin was fixed to the bushing and that a 1 kN force was applied at its tip, at a distance of 60 mm, the maximum bending moment would be of 60 Nm. The selected material for the pin was the low carbon steel DIN St37-2K, with a minimum yield strength of 260 MPa. For the pin diameter of 30 mm, a safety factor of 11.5 was obtained.

The endurance limit of the pin was calculated and a safety factor of 6.7 was obtained, meaning that, theoretically, it would never fail due to fatigue.

A finite element analysis was done on half of the system (i.e. one pin, one bushing, four screws and half of the top plate), as the stresses in the plate, distant from the pin, were negligible. The fasteners between the bushing and the plate where simplified, to minimize the required processing power. The coefficients of friction used were 0.15 for the lubricated contact between the pin and the bushing, 0.35 for the dry contact between the bushing and the plate and 0.2 for the dry contact between the fasteners and the bushing and plate. A preload of 5 kN was used for the screws.

The materials chosen for the bushing and the plate were the bronze DIN 1705 CuSn7ZnPb and the steel DIN QStE 420 TM, with minimum yield strengths of 120 and 420 MPa, respectively. The percent elongations of the pin, bushing and plate were of at least 10, 12 and 21%, respectively. Therefore, the chosen failure criterion had to be appropriate for ductile materials. The maximum distortion energy (or von-Mises) criterion was used and the safety factors of 10.8, 2.2 and 3.0 were obtained for the pin, bushing and plate, respectively.

For the screws, the 9.8 material class from the EN ISO 898 standard was chosen, which guaranteed a minimum yield strength of 720 MPa. Being made of a medium carbon steel, these would present a brittle failure behavior. Therefore, the maximum principal stress criterion was used and a safety factor of 2.4 was estimated.

These safety factors remained approximately constant, independently of the direction that the force was applied, and were considered sufficient, as the critical scenario of an emergency brake should not happen often.

4.6 Lifter module

Similarly to the tugger module, the lifter module was designed independently of the ScalABLE project. This section will detail the development process of this module.

Three different lifting mechanisms where studied for the lifter module:

- One or more scissor platforms coupled to one or more electric motors.
- A pair of scissor car jacks with integrated electric motors and gearboxes.
- Two or four linear actuators.

4.6.1 General

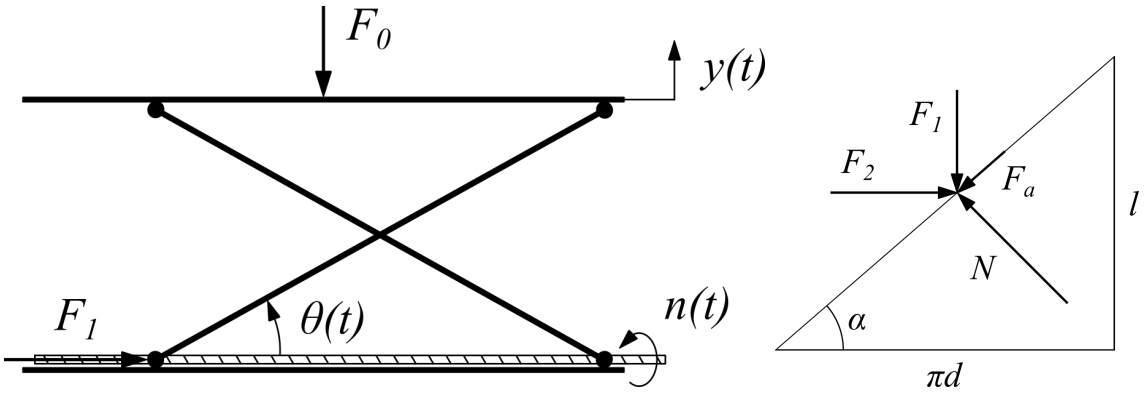
The scissor platforms were wide enough so that no guidance was needed but the forces output by both the car jacks and the linear actuators were imposed on small areas and both systems had almost no resistance to horizontal forces or bending moments by themselves. To be able to withstand the solicitations resulting from decentered loads as well as from the acceleration generated by the mobile module, a guiding mechanism would be required. With that in mind, the only advantage of the car jacks would be their lower cost when compared to the linear actuators.

It would be advantageous if the lifting mechanism could fit inside the mobile module to thus reduce the minimum total height of the platform. However, both the scissor platforms and the car jacks where too large for this and only the linear actuators could, be partially placed inside the mobile module.

Despite its lower cost, the car jacks solution was discarded due to the need for a guiding system and the larger required height of the module, as well as their lower reliability. The scissor platform solution was the first to be studied in more detail.

4.6.2 Scissor platforms

The scissor platforms considered included a lead screw to convert a manually applied rotational motion to the linear motion required to move its bars. The two mechanisms were modeled as shown in figures 4.21a and 4.21b.



(a) Scissor platform (b) Lead screw

Figure 4.21: Models of the scissor platform and the lead screw

A trapezoidal velocity profile was assumed for the load and, through a kinematic analysis, it was concluded that the rotational speed n of the lead screw was related to the vertical speed $\dot{y}$ of the load by:

$$n = \frac{60}{l} \cdot \tan(\theta) \cdot \dot{y},$$

where:

l is the lead of the lead screw.

Consequently, n would be maximum at the beginning of the deceleration ramp. This also meant that for small values of theta, when the mechanism was almost closed, y. was greatly amplified, relatively to the rotational speed imposed on the lead screw.

The force F_0 that needed to be applied to the load was given by:

$$F_0 = m \cdot (g + \ddot{y}),$$

where:

m is the mass of the load,

g is the gravitational acceleration and

$\ddot{y}$ is the imposed vertical acceleration.

A static analysis of the scissor platform and the car jacks revealed that the horizontal force F_1 that had to be applied by the lead screw was given by:

$$F_1 = \frac{F_0}{\tan(\theta)}.$$

This meant that for small values of theta, the force F_0 imposed on the load was greatly attenuated, relatively to the force generated by the lead screw, which would be useful if the platforms were to work around their fully open position. However, that was not possible, due to the need to minimize the total height of the module and the fact that the platforms did not fit inside of the mobile module at all.

Therefore, the platforms working range had to be around their fully closed position and thus the torque required to lift the load (T) would be substantially amplified in comparison with F_0. On top of that, the lead screw that converted the rotational motion from the motor into linear motion, implied significant losses to prevent back driving. The torque on the lead screw was given by:

$$T = \frac{d}{2} \cdot \tan(\alpha + \phi) \cdot F_1 = \frac{d}{2} \cdot \frac{\tan(\alpha + \phi)}{\tan(\theta)} \cdot F_0, \quad \phi = \arctan(\mu), \quad \alpha = \arctan\left(\frac{\pi \cdot d}{L}\right),$$

where:

d is the diameter of the lead screw and

μ is the coefficient of friction between the lead screw and the nut.

Because the torque imposed on the lead screw decreased with theta, its maximum value would be at the beginning of the lifting movement. Finally, the necessary input power on the lead screw was given by:

$$P_2 = n \cdot \frac{2\pi}{60} \cdot T = F_0 \cdot y \cdot \frac{\tan(\alpha + \phi)}{\tan(\alpha)}.$$

Which was simply the power P_0 imposed on the load, multiplied by the efficiency of the lead screw. Because this efficiency was constant, the maximum power required for the motor could be found by minimizing P_0 which *a priori* would be maximum at the end of the acceleration ramp, when the load reached its maximum velocity while still being accelerated.

P_0 depended on the acceleration and deceleration time intervals. It was concluded that, for a reliable range of values for these intervals, that is, for large enough intervals, P_0 would increase with their increase.

Estimates of the maximum required rotational speed n, torque T and power P_2 were calculated by defining the following parameters:

- Vertical displacement of the platform, $\Delta y = 30$ mm;

- Total time for the lifting movement, $\Delta t_{tot} = 8$ s;
- Acceleration/deceleration time, $\Delta t_a = \Delta t_d = 2$ s;
- Maximum mass of the load, m = 500 kg.

Additionally:

- The lead screw had 10 mm of diameter with a lead of 1.5 mm;
- The minimum angle theta formed by the bars was measured as 9.23 ^{o};
- A coefficient of friction of 0.16 was assumed for lubricated contact between steel surfaces;
- Again, a trapezoidal velocity profile was assumed.

The following values were calculated from these parameters:

- Average velocity of the platform, $\dot{y}_{med} = \frac{\Delta y}{\Delta t} = 3.75$ mm/s;
- Maximum velocity of the platform, $\dot{y}_{max} = \frac{\dot{y}_{med}}{1-\Delta t_a/\Delta t_{tot}} = 5$ mm/s;
- Acceleration imposed on the load, $\ddot{y}_a = \frac{\dot{y}_{max}}{\Delta t_a} = 0.0025$ m/s^2;
- Maximum rotational speed of the lead screw, $n_{2max} = 45.6$ rpm;
- Maximum torque imposed on the lead screw, $T_{2max} = 48.6$ Nm;
- Maximum power input on the lead screw, $P_{2max} = 112$ W.

The rotational speed, torque and power were also plotted along the time for the lifting movement.

Two slightly different shapes of scissor platforms available on the market were found and different configurations were studied:

- The first configuration, with a single platform, would be the simplest and cheapest one to implement but would provide the smallest support area.
- The second configuration, with a pair of platforms inline with each other, would increase the longitudinal support area but would provide a limited transversal support area.
- The third configuration, with the platforms parallel to each other and oriented perpendicularly relatively to the structure of the module, would provide the largest support area. However, because of their shape, the resistance of the resulting structure to longitudinal forces, due to an emergency stop, for example, would be too low. This meant that this configuration, would be the best option, if the environment in which the module would be deployed was controlled enough so that no unexpected obstacles could appear.

- Finally, the fourth configuration, with the platforms parallel to each other and to the structure of the module, would provide the second largest support area, while still maintaining a high structural resistance to longitudinal forces.
- Alternatively, a customized scissor platform could be designed and manufactured with the desired area. However, this would go against the requirement of using components already available on the market.

The fourth configuration was considered the most advantageous and was thus chosen.

The first approach that was studied consisted in coupling the two platforms to a single motor and gearbox through a chain drive mechanism. This guaranteed that the torque supplied by the motor would be automatically directed to the platform with the most weight on, in case the load was off-center. This would lower the cost of the module when compared to using two independent actuators.

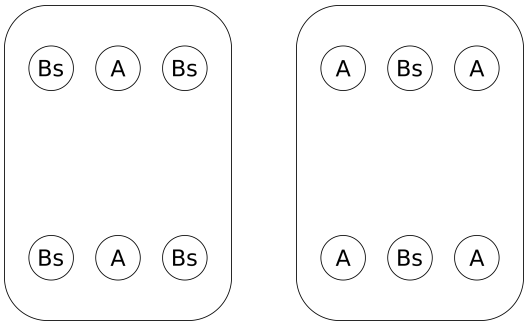
However, the large torques involved would result in a large radial force on the shaft of the platform, which was certainly not predicted in its design and would probably cause its failure. To solve this problem, an intermediate shaft supported by two bearings would have to be used for each platform, which would not only make the solution too complex but would also occupy too much space on the module. The conclusion was that two motors had indeed to be used.

However, to guarantee that the system could hold a 500 kg load, independently of the position of its center of gravity, relatively to the platform, both motors and gearboxes would have to be dimensioned to lift 500 kg each.

The limited space available on the platform difficulted the development of a viable solution.

4.6.3 Linear actuators

For the linear actuators solution, a guiding system, such as a set of ball splines, would be required. Two similar configurations were studied. Either two actuators were used and placed on the front and back of the module, along with four vertical ball splines, one on each corner (figure 4.22a). Or the opposite, four actuators were placed one on each corner, along with two ball splines, on the front and back of the module (figure 4.22b). Due to the lower cost of the ball splines and the higher rigidity that they provided, relatively to the linear actuators, the first approach was chosen.



(a) (b)

Figure 4.22: Alternative configurations for the linear actuators and the ball splines

The actuators would have to be capable of lifting 500 kg each but, in a critical situation, where the system would be running at full speed and would have to do an emergency stop, they would have to handle higher loads. An estimate of these and the loads on the ball splines was obtained using the model shown in figure 4.23. The point C is the center of mass of the load and the fixed supports represent the top plate of the mobile module. The forces arising from the load would be $F_g = mg$ and $F_a = ma$, where m is the mass of the load and g and a are the gravitational acceleration and the deceleration resulting from the braking, respectively.

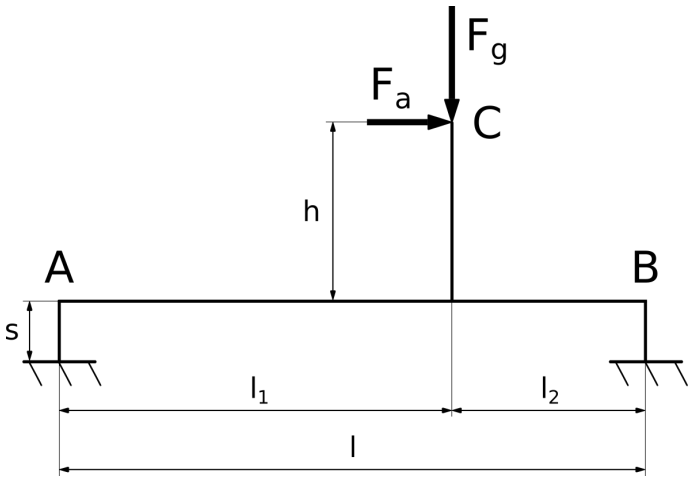


Figure 4.23: Model of the lifter module

This model provided a conservative estimate as in a real situation, the load would be distributed along part of the plate. A vertical reaction force and a bending moment were considered for each actuator and pair of ball splines, respectively, and the previous model was further simplified into the free-body diagram shown in figure 4.24.

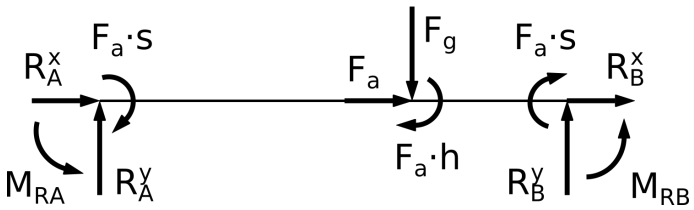


Figure 4.24: Free-body diagram of the model in figure 4.23

A static analysis of the latter resulted in the following equations for the forces and bending moments on the actuators and ball splines, respectively:

$$R_A^y = F_g \cdot \beta^2 \cdot (3 - 2\beta) - \frac{6 \cdot F_a \cdot h \cdot \beta \cdot (1 - \beta)}{l},$$

$$R_B^y = F_g \cdot \alpha^2 \cdot (3 - 2\alpha) + \frac{6 \cdot F_a \cdot h \cdot \alpha \cdot (1 - \alpha)}{l},$$

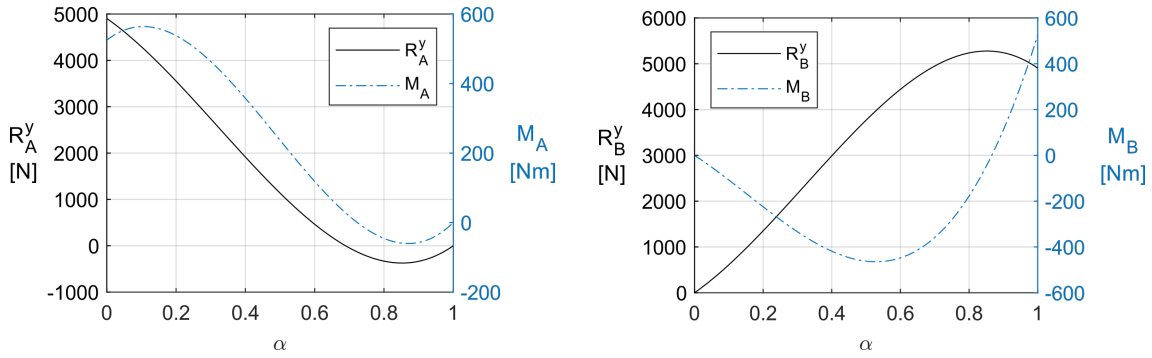
$$M_A = F_g \cdot l \cdot \beta^2 \cdot (1 - \beta) + F_a \cdot h \cdot \beta \cdot (3\beta - 2) + F_a \cdot s \cdot \beta,$$

$$M_B = -F_g \cdot l \cdot \alpha^2 \cdot (1 - \alpha) + F_a \cdot h \cdot \alpha \cdot (3\alpha - 2) + F_a \cdot s \cdot \alpha,$$

$$\alpha = \frac{l_1}{l}, \quad \beta = \frac{l_2}{l}.$$

These equations were then used to find the critical point at which the maximum reaction force and bending moment would appear. The following parameters were defined and a numerical approximation, shown in figure 4.25, was obtained using the software MATLAB:

- A length l = 570 mm;
- A maximum h = 1 m — A load with a higher center of gravity would be susceptible to tipping;
- A stroke s = 50 mm;
- The same top speed, stopping distance and thus deceleration a = 1 m/s^2 that were defined in the previous subsection;
- A mass m = 500 kg.



(a) Support A

(b) Support B

Figure 4.25: Reaction forces and bending moments as functions of α

The maximum forces and moments obtained were the following:

- $R^y_A = 4905$ N at $\alpha = 0$;
- $R^y_B = 5278$ N at $\alpha = 0.852$;
- $M_A = 564$ Nm at $\alpha = 0.106$;
- $M_B = 525$ Nm at $\alpha = 1$.

Assuming that the bending moments would be equally absorbed by the ball splines of each pair, at each end of the plate, the maximum bending moment per ball spline would be $M = 282$ Nm.

4.6.4 Selected solution

The fact that the scissor platforms could not be fitted inside of the mobile module, associated with the limited space available on its top plate, difficulted the development of a viable solution. Furthermore, these provided a lower rigidity, relatively to the ball splines. Lastly, the configuration chosen for the scissor platforms allowed only about two thirds of the total area of the mobile module to be used for support, while the solution with the linear actuators encompassed the full area. For these reasons, the latter solution was chosen.

Despite not being able to find the price for the H-track actuators, together with the ball splines they would be substantially more expensive than the scissor platforms along with motors and gearboxes. Therefore, if the total cost of the system weighed more than the aforementioned advantages, the scissor platforms solution would have been selected.

The H-Track linear actuators from Warner Linear were chosen for their exceptionally compact design, which minimized the height of the system, and high force capability. These are hydraulic actuators, with an integrated hydraulic system, that only require the supply of electrical power. The actuators chosen were the H2C-24-1AB32-A-02, with a stroke of 50 mm and capable of providing

5560 N, when extending, and 22241 N, when being compressed while stationary. Each contained a 24V, 250 W motor that had to be supplied with a current of up to 25 A. Contrarily to the actuators on the tugger module, these would have to be screwed to the bottom plate of the mobile module.

A meter-out valve was specified, to prevent shattering when the actuators were retracting and the load was being lowered. Two pairs of back-to-back 30V Zener diodes were selected to be placed across the terminals of the actuators, as per their manual, to avoid damage to the system (e.g. voltage spikes) during its start and stop.

The direction of rotation of the motors had to be controlled and their acceleration and deceleration had to be limited, to prevent damage to the system. Therefore, two KBBC-24M controllers from KB Electronics were selected. These allowed the control of the direction by closing one of two contacts and the acceleration/deceleration could be limited using built in trimmer potentiometers.

Two pairs of inductive sensors from SICK were selected to detect when the platform was fully lowered or raised and power off the motors. Each pair would be placed near each actuator to guarantee that they would stop only when they reached their maximum position. Otherwise, if one of the actuators moved slower (e.g. due to an unequal distribution of the load), it could stop before fully extending and the platform would end up inclined. The sensors would be used in the configuration shown in figure 4.26, allowing the stroke to be easily adapted. The two sensors were positioned at slightly different heights as a safety measure, so that when the platform reached one of its limit positions, only one of the sensors would detect the part. If for any reason the actuators do not stop (e.g. the sensor malfunctions), the second sensor would eventually detect the part and the system would respond by stopping and entering in an alarm state.

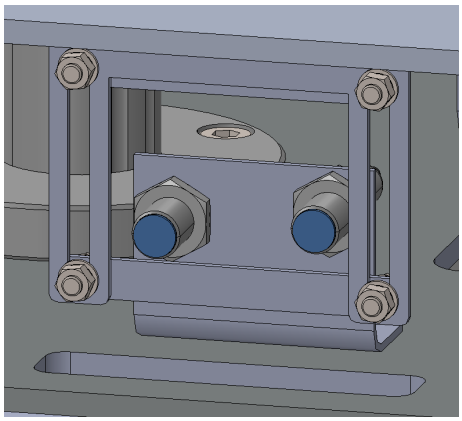


Figure 4.26: Inductive sensors as limit switches with adaptable stroke.

Five WAGO 788-312 and two WAGO 858-304 relay modules, with two an four changeover contacts, respectively, were selected to be used in the configuration shown in figure 4.27. The WAGO 750-432 module, with four digital inputs, was used to monitor the inductive sensors and the WAGO 750-531 module, with four digital outputs, was used to switch the contacts of the controllers.

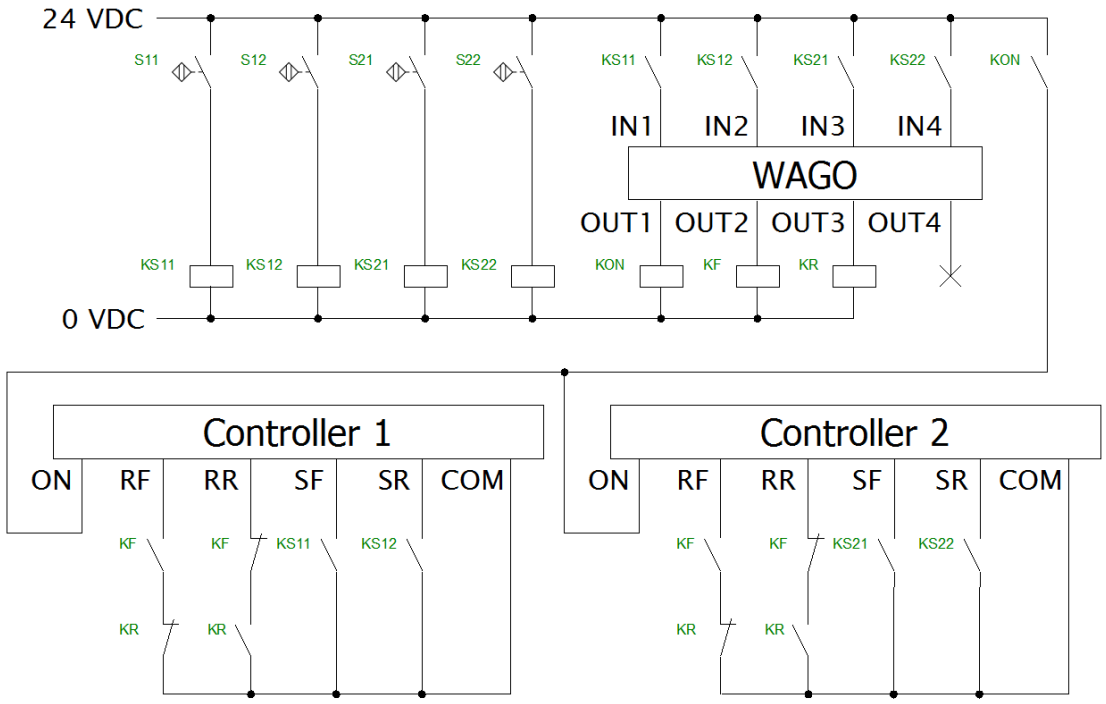


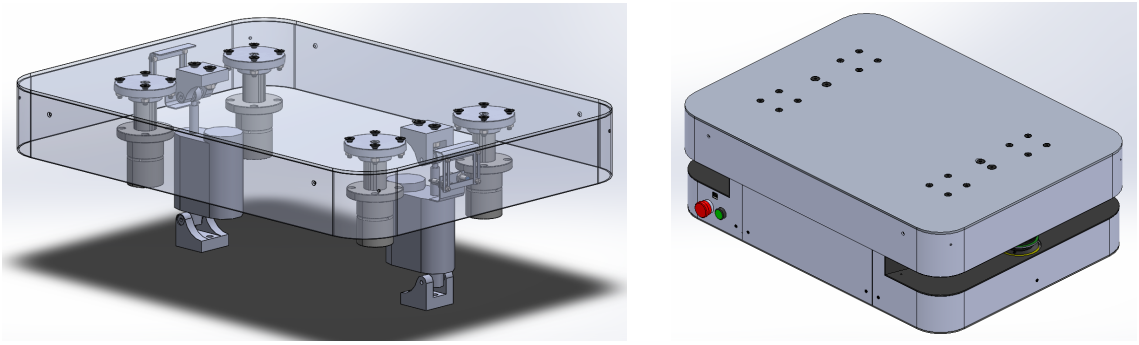
Figure 4.27: Electrical circuit used to control the movement of the linear actuators

The contacts of the relays KF (forward) and KR (reverse) make an exclusive or, allowing only one of the signals to pass. When the system reaches one of its limit positions, the contacts of either the KS11 and KS21 or of the KS12 and KS22 are closed, sending a stop signal to the controllers and pulling up the respective inputs on the WAGO module.

The LF ball splines from THK, with a flanged bushing and a spline shaft with 40 mm of diameter, were selected. These were capable of withstanding a bending moment of up to 687 Nm, according to their datasheet, which would be more than enough. The next smaller ball spline, with 30 mm of diameter, would hold 290 Nm, which was considered too close to the maximum estimated torque.

Similarly to what was done for the other modules, an alternative top plate with a thickness of 6 mm was designed to replace the one on the mobile module. Six holes were designed into the plate for the two actuators and the ball splines to go through. Furthermore a lifting plate with the same thickness was also designed to hold the load. These two plates were validated with finite element analysis approximations.

Figure 4.28a shows the lifter module by itself and figure 4.28b shows it assembled on the mobile module.



(a) (b)

Figure 4.28: Lifter module and mobile lifter system

4.7 Motion controller

A motion controller was developed in the Codesys environment for the mobile module. In order to speed up the development, a MATLAB based simulator was setup. The Sim.I.Am MATLAB simulator for mobile robots, developed by the Georgia Robotics and Intelligent Systems (GRITS) Lab, was simplified, to require less processing power, and was used to test the designed controllers. When the result was satisfying, the controller was ported to Codesys and a Modbus TCP/IP interface was setup between the motion controller and the simulator. Because the mobile module only became operational near the end of this dissertation, the controller could not be tested on the real system.

For this dissertation, it was assumed that only odometry would be used to keep track of the position of the robot. This is unreliable for distances of more than a few meters, when a high precision is required. In the future, in order to make the system accurate enough for real applications, the data from the safety lasers will have to be used as well.

Quadratic Bézier curves were used as the paths between the pose of the robot and the desired goal. The unicycle model was used to design three PID based controllers:

- A go-to-goal controller.
- A target following controller.
- A line following controller.

4.7.1 Go-to-goal

The goal to goal controller only moved the robot to a given goal position, without the ability to follow a path. The angle between the direction of the desired goal (ϕ_d) and the heading of the robot (ϕ) was used as the error input (e) to a regular PID controller.

$$e = \phi_d - \phi, \quad \phi_d = \arctan\left(\frac{G_y - R_y}{G_x - R_x}\right).$$

The atan2() function from MATLAB was used to guarantee that both ϕ_d and e where between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. The controller used the error at each iteration (e_k) to compute the angular velocity (ω_k) to be imposed. A constant sampling time interval (Δt_s) was assumed.

$$\omega_k = K_P \cdot e_k + K_I \cdot E_k + K_d \cdot \frac{(e_k - e_{k-1})}{\Delta t_s}, \quad E_k = E_{k-1} + e_k \cdot \Delta t_s,$$

4.7.2 Path generation

The path for the robot to follow was defined using a quadratic Bézier curve defined by:

$$b(t) = R \cdot (1-t)^2 + P \cdot (1-t) \cdot t + G \cdot t^2, \quad 0 \leq t \leq 1,$$

being R, P and G the current position of the robot, an intermediate point and the goal, respectively. By defining the desired orientation for the robot when it reaches the goal, the intermediate point P could be obtained by computing the intersection of the directions corresponding to the initial and desired final poses (figure 4.29).

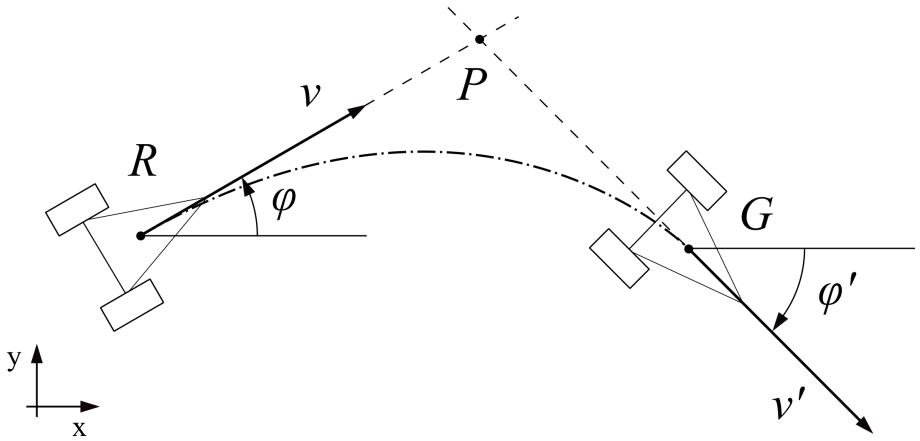


Figure 4.29: Intersection between the directions of the current and the desired poses

4.7.3 Target following

This controller generated subsequent targets along the defined path, that were fed to the go-to-goal controller presented in subsection 4.7.1. The targets had to be points along the path, a small distance ahead of the robot. However, the Bézier curve was a parametric curve and function of t. Therefore, to obtain a point along the path given a certain distance, its total length had to be computed, which was done using the following line integral:

$$S = \int_a^b f(x,y)\,ds = \int_0^1 1 \cdot \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\,dt = \int_0^1 \sqrt{a \cdot t^2 + b \cdot t + c}\,dt,$$

$$a = 4 \cdot \left(d_x^2 + d_y^2\right), \quad b = 4 \cdot (d_x \cdot e_x + d_y \cdot e_y), \quad c = e_x^2 + e_y^2,$$

$$d_i = R_i - 2P_i + G_i, \quad e_i = 2 \cdot (P_i - R_i) \quad i = \{x,y\}.$$

The distance of the target along the path (d_p) was updated at each iteration by incrementing a small predefined value (d_d), taking into account the distance from the robot to the current target position (d_t).

$$d_p = d_p + max(\Delta d_{min}, d_d - d_t), \quad d_t = \sqrt{(T_x - R_x)^2 + (T_y - R_y)^2}, \quad \Delta d_{min} \geq 0,$$

where d_{min} is the minimum distance increment, to prevent backtracking, and T is the position of the current target. After updating the distance of the target along the path, the corresponding value of the parameter t could then be obtained $\left(t = \frac{d_p}{S}\right)$ and the new target position computed, using equation 4.7.2. This position was then fed to the go-to-goal controller described in 4.7.1.

4.7.4 Line following

The line following controller computed the distance from the center of the robot to the desired path and used this value as the error input to a PID controller, to calculate the angular velocity to be imposed.

In order to compute the distance from the robot to the path, the point on the path closest to the robot (P_{pcr}) had to be found. The golden-section search algorithm was used for this. This algorithm finds the extremum of a strictly unimodal function, in this case the minimum distance from the robot to a point on the Bézier curve, by successively shortening the range of values that contain it. To guarantee that the function was strictly unimodal, the initial range of values on each iteration was confined to a small region around the estimated position of the robot. Otherwise, a path such as the one shown in figure 4.30 could be problematic, as the robot could skip a portion of the curve or go back to a previous position. The imposed limits were the previous position of the robot and that same position plus the maximum possible displacement, based on the linear velocity of the system and the sampling time. This restriction also minimized the time complexity of the algorithm.

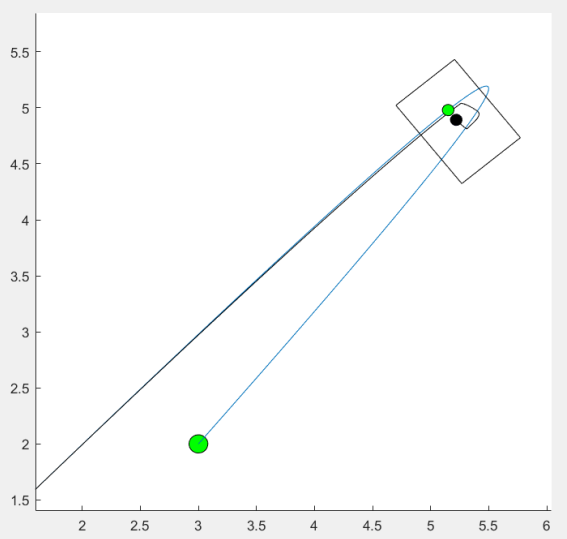
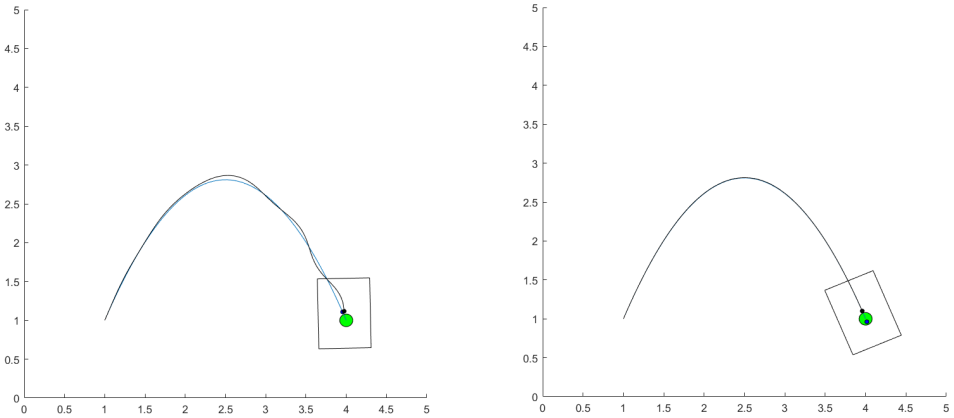


Figure 4.30: Problematic path [m]

It was concluded that the reference point of the robot could not be its center of rotation but a point a small distance ahead of it, otherwise the controller would not provide sufficient accuracy (figure 4.31).



(a) Center of rotation

(b) Point ahead of center of rotation

Figure 4.31: Lack of accuracy when using the center of rotation as the reference point. [m]

The obtained distance was then fed to a regular PID controller, to get the required angular velocity. The following procedure was used to ascertain if this velocity had to be in the positive or negative direction and figure 4.32 shows the relevant vectors.

1. Compute the vector $\mathbf{u_1}$ that goes from the previously obtained point P_{pcr} to a point along the path, a very small distance (e.g. 1 mm) ahead of it;
2. Compute the vector $\mathbf{u_1'}$ by rotating $\mathbf{u_1}$ by -90°;

3. Compute the vector $\mathbf{u_2}$ that goes from the point P_{pcr} to the reference point of the robot;

4. Compute the scalar product of the vectors $\mathbf{u_1}$ and $\mathbf{u_2}$.

The direction of the angular velocity would have to be equal to the sign of the result of the scalar product, as the scalar product of two two-dimensional vectors is positive or negative if the angle formed between them is less or more than 90°, respectively. The procedure can be summarized as the expression:

$$\left(\mathbf{R}\left(-\frac{\pi}{2}\right)\cdot\mathbf{u_1}\right)^{\mathsf{T}}\cdot\mathbf{u_2} = \left(\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\cdot\mathbf{u_1}\right)^{\mathsf{T}}\cdot\mathbf{u_2}.$$

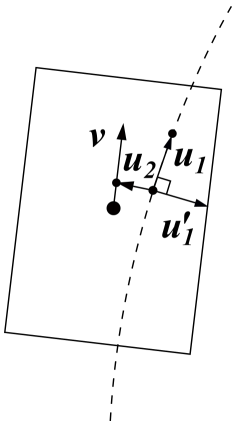


Figure 4.32: Vectors used to compute the direction for the angular velocity

4.7.5 Simulation results

Two different paths were simulated, one with a soft turn and another with a sharp turn. The integral and derivative gains were set to zero on both controllers. The proportional gain and the distance to the target or the reference point were manually optimized through trial and error. A constant linear velocity was imposed and the measurement used for comparison was the maximum deviation from the desired path.

The target following controller achieved the best performance in the both paths with a maximum error of 0.75 and 25 mm on the soft and sharp turns, respectively. The line following controller deviated from the path by a maximum of 1.6 and 40 mm, respectively. Furthermore, the latter was more susceptible to oscillations. Figures 4.33 to 4.36 show the results of the simulations.

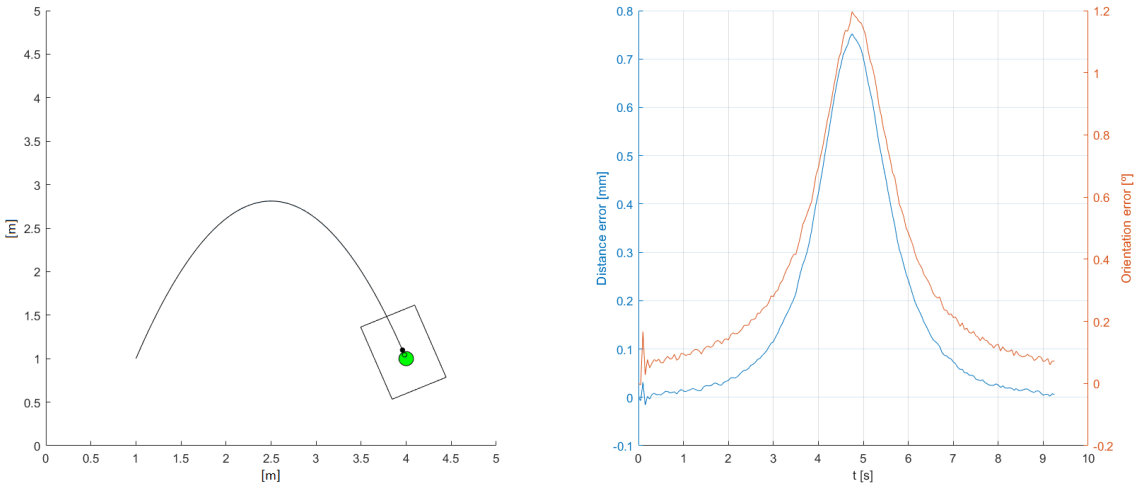


Figure 4.33: Target following controller on the soft turn

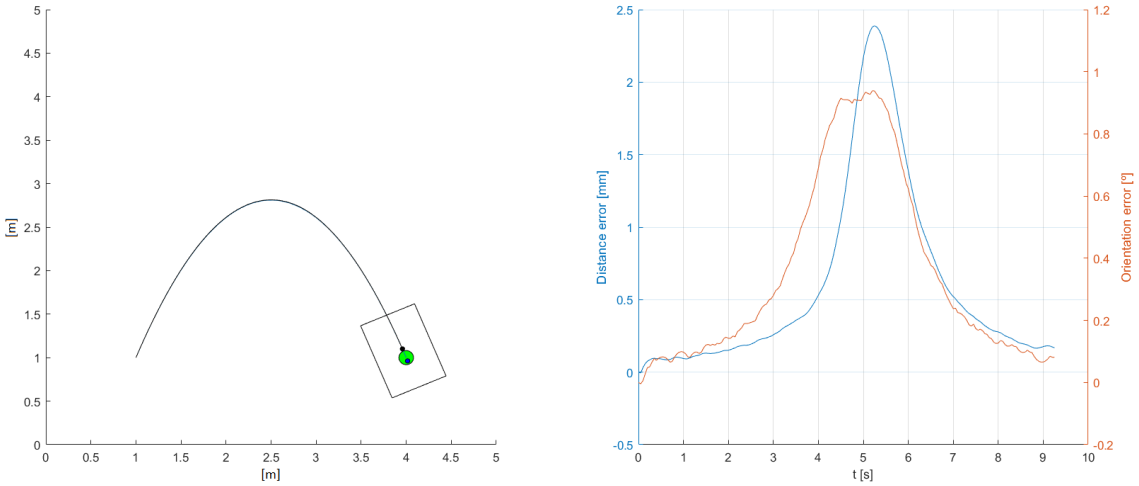


Figure 4.34: Line following controller on the soft turn

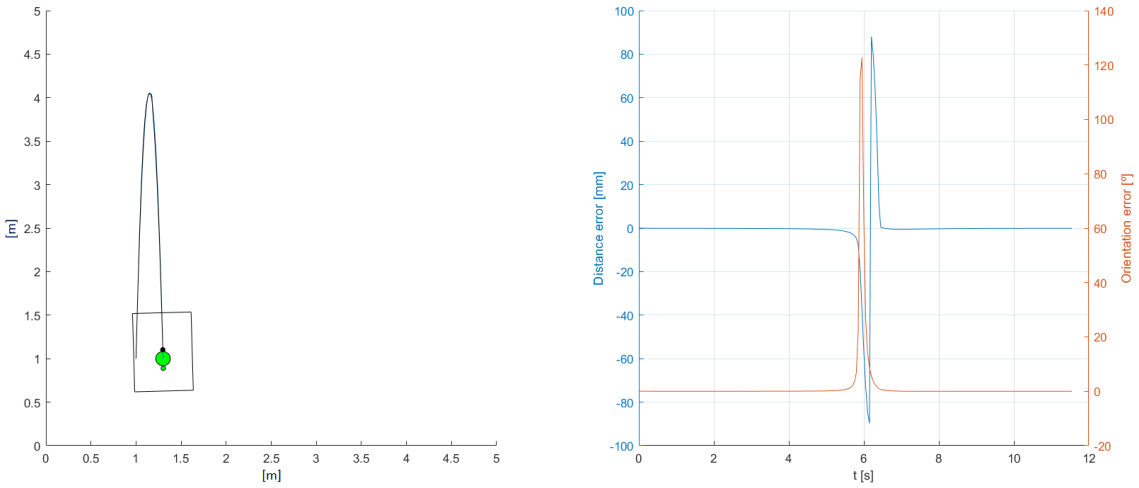


Figure 4.35: Target following controller on the soft turn

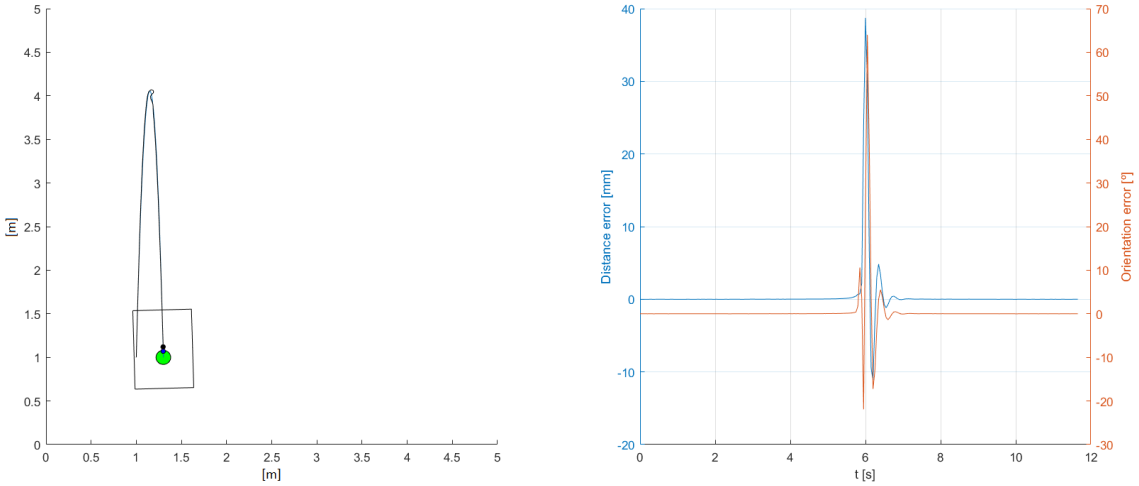


Figure 4.36: Line following controller on the soft turn

Chapter 5

Conclusions and future work

In this chapter, a final review of the most relevant aspects of the project is given, along with a possible direction for future work.

This dissertation was aimed at the participation in the development of a mobile manipulator as well as the extension of its design into that of a modular robotic system, capable of performing two other alternative tasks, namely, the tugging of heavy structures and the lifting and transporting of heavy loads. Besides the mechanical and electrical design of the defined modules, this entailed the development of two software interfaces using the Codesys environment. These handled the communication between an industrial PC and both the motor controllers and the battery management system of the mobile module.

Furthermore, the motion control of mobile robots, using Bézier curves as paths, was studied in order to design a controller for the system. Multiple controllers based on the PID control loop feedback mechanism were designed and tested, along with a path generating algorithm. The selected controller would be integrated into the software interface that was designed to handle the communication between the navigation system and the actuator submodule of the mobile module. However, the controller could not be tested on the real system as the mobile module only became operational near the end of the dissertation when time was short.

Despite one of the advantages of a modular design being the possibility to develop each module independently, it was not the case for this project. This was due to the fact that the system was built from scratch and thus, the interactions between the modules changed as their design evolved. The modules had to be developed in parallel and iteratively, as some changes to one module would affect the others. This was especially true in terms of the spatial distribution of their subcomponents as the space available was very limited.

This work will serve as an example of some of the aspects that need to be taken into consideration when designing modular robotic systems. On the other hand it provides a flexible solution to problems related to the use of mobile robots for material handling tasks, which are being increasingly explored in the industry.

The following tasks would be desired to complement the work done up to this point:

- Redesign of the mobile module to increase its compatibility with the other modules;
- Further exploration of the alternative solutions for the other modules, such as the use of an integral scissor platform for the lifter module;
- Development of the software required to control the designed modules;
- Development of more modules to perform other functions;
- Test of the motion controller on the real system;
- Design of a new motion controller capable of leveraging the data from the safety lasers.

References

[1] J. K. Gershenson, G. J. Prasad, and Y. Zhang. Product modularity: Definitions and benefits. *Journal of Engineering Design*, 14(3):295–313, August 2010.

[2] Matti Eiden. Modular product development literature review and case study. May 2013.

[3] R. Marshall, P. G. Leaney, and P. Botterell. Enhanced product realisation through modular design: an example of product/process integration. *Journal of Integrated Design and Process Technology*, 3:143–150, July 1998.

[4] K. R. Allen and S. Carlson Skalak. Defining product architecture during conceptual design. *Proceedings of the 1998 ASME Design Engineering Technical Conference*, 1998.

[5] T. U. Pimmler and S. D. Eppinger. Integration analysis of product decompositions. *Proceedings of the 1994 ASME Design Engineering Technical Conferences—6th International Conference on Design Theory and Methodology*, September 1994.

[6] R. Sanchez. Strategic product creation: managing new interactions of technology, markets, and organizations. *European Management Journal*, 14(2):121–138, 1996.

[7] C. Y. Baldwin and K. B. Clark. Managing in an age of modularity. *Harvard Business Review*, 75:84–93, September 1997.

[8] D. W. He and A. Kusiak. Performance analysis of modular products. *International Journal of Product Research*, 34:253–272, 1996.

[9] K. Ulrich and S.D. Eppinger. *Product Design and Development*. 1995.

[10] K. Ulrich and K. Tung. Fundamentals of product modularity. *Proceedings of the 1991 ASME Design Engineering Technical Conferences—Conference on Design/Manufacture Integration*, 1991.

[11] S. Sosale, M. Hashemian, and P. G U. Product modularization for reuse and recycling. *Concurrent Product Design and Environmentally Conscious Manufacturing*, 94:195–206, January 1997.

[12] C.-C. Huang and A. Kusiak. Modularity in design of products and systems. *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, 28:66–77, January 1998.

[13] M. Carey. Modularity times three. *Sea Power*, April 1997.

[14] Ole Madsen, Simon Bøgh, Casper Schou, Rasmus Andersen, Jens Damgaard Skov, Mikkel Rath Pedersen, and Volker Krüger. Integration of mobile manipulators in an industrial production. *Industrial Robot: An International Journal*, January 2015.

[15] Simon Bøgh, Mads Hvilshøj, Morten Kristiansen, and Ole Madsen. Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (aimm). *The International Journal of Advanced Manufacturing Technology*, 2012.

[16] Suyang Yu, Changlong Ye, Hongjun Liu, and Jun Chen. Development of an omnidirectional automated guided vehicle with my3 wheels. *Perspectives in Science*, 2016.

[17] Charles B. Owen, Fan Xiao, and Paul Middlin. What is the best fiducial? *ART 2002 - 1st IEEE International Augmented Reality Toolkit Workshop, Proceedings*, 2002.

[18] Y. Li, YT. Wang, and Y. J. Liu. Fiducial marker based on projective invariant for augmented reality. *Journal of Computer Science and Technology*, November 2007.

[19] Robots and robotic devices – Collaborative robots. Standard, International Organization for Standardization, February 2016.

Appendix A

BMS communication protocol

The battery management system sends out the string described in A.1 every 0.5 or 1 s, depending on the position of an onboard switch, using the following communication settings:

- Format: big-endian
- Baud rate: 9600
- Parity bit: None
- Start bit: 1
- Stop bit: 1

Table A.1: Contents of data string

Byte	Information	Step	Size	Example
1,2,3	Total Voltage	0,005 Volt/bit	24 bit	0x0105FF = 335,4 Volt
4	Sign byte I1	Ascii: + , -, X	8 bit	0x2B = +, 0x2D = - , 0x58 = X
5,6	Current IN	0,125 Amp/bit	16 bit	0x0100 = 32 Amp
7	Sign byte I2	Ascii: + , -, X	8 bit	0x2B = +, 0x2D = - , 0x58 = X
8,9	I2	0,125 Amp/bit	16 bit	0x0100 = 32 Amp
10	Sign byte I3	Ascii: + , -, X	8 bit	0x2B = +, 0x2D = - , 0x58 = X
11,12	I3	0,125 Amp/bit	16 bit	0x0100 = 32 Amp
13,14	Vmin	0,005 Volt/bit	24 bit	0x0230 = 2,80 Volt
15	Cell Vmin	Cell Nr/bit	8 bit	0x32 = Cell number 50
16,17	Vmax	0,005 Volt/bit	24 bit	0x0230 = 2,80 Volt
18	Cell Vmax	Cell Nr/bit	8 bit	0x32 = Cell number 50
19/20	Tmin	1 Deg / bit + ofset	16 bit	0x0114 = 0 deg Celcius, 0x0128 = 20 deg Celcius
21	Cell Tmin	Cell Nr/bit	8 bit	0x32 = Cell number 50
22,23	Tmax	1 Deg / bit + ofset	16 bit	0x0114 = 0 deg Celcius, 0x0128 = 20 deg Celcius
24	Cell Tmax	Cell Nr/bit	8 bit	0x32 = Cell number 50
25	Cell nr info	Cell Nr/bit	8 bit	0x32 = Cell number 50
26	Nr off cells	Cell Nr/bit	8 bit	0xFF = Cell number 255
27,28	Cell Voltage	0,005 Volt/bit	24 bit	0x0230 = 2,80 Volt
29/30	Cell Temp	1 Deg / bit + ofset	16 bit	0x0114 = 0 deg Celcius, 0x0128 = 20 deg Celcius
31	Status	*See info below*	8 bit	*See info below*
32,33,34	TodayEnergy collected	Wh/bit	24 bit	0x000064 = 100 Wh
35,36,37	Energy stored	Wh/bit	24 bit	0x00F221 = 61,985 kWh
38,39,40	Today Energy consumed	Wh/bit	24 bit	0x000064 = 100 Wh
41	SOC %	1%/bit	8 bit	0x32 = 50%
42,43,44	Total collected	kWh/bit	24 bit	0x00640000 = 6.553.600 kWh
45,46,47	Total consumed	kWh/bit	24 bit	0x00640000 = 6.553.600 kWh
48,49	Device time MM:SS	H/bit, M/bit	16 bit	0x1620 = 22:32
50,51	Battery capacity	0,1 kWh/bit	16 bit	0x00A0 = 16,0 Kwh
52,53	V-MIN Setting	0,005 Volt/bit	24 bit	0x15FF = 56,31 Volt
54,55	V-MAX Setting	0,005 Volt/bit	24 bit	0x15FF = 56,31 Volt
56,57	V-Bypass Setting	0,005 Volt/bit	24 bit	0x15FF = 56,31 Volt
58	Checksum	0x34	8 bit	Lowest 8 bits of an addition of all bytes before

Table A.2: Contents of status byte

Status Byte		
MSB	7	SOC not calibrated
	6	Exceed T-MAX
	5	Exceed T-MIN
	4	Exceed V-MAX
	3	Exceed V-MIN
	2	Communication error
	1	Allow to discharge
LSB	0	Allow to charge

Appendix B

Motor controllers interface

Figures B.1 and B.2 contain the main code and the public methods, respectively, of the function block of the interface developed to communicate with the motor controllers of the mobile module.

```
FUNCTION_BLOCK PUBLIC DRIVER
VAR_OUTPUT
    faultState : BOOL := FALSE;
END_VAR
VAR
    submode : UDINT := 69;
    operation : SINT;
    currentOperation : SINT;
    currentInternalPosition : DINT;
    targetVelocity : DINT;
    controlWord, statusWord : UINT;
    iState : SINT := 0;
    flagResetFault : BOOL := FALSE;
END_VAR

// Error detection
faultState := (statusWord AND 8) = 8;
IF faultState THEN
    iState := -1;
END_IF
// State machine
CASE iState OF
    -1: // Error handling
        IF NOT faultState THEN
```

```
            flagResetFault := FALSE;
            iState := 0;
        ELSIF (statusWord AND 1) = 1 OR ((controlWord AND 128) =
            128 AND NOT flagResetFault) THEN
            controlWord := 0;
        END_IF
    0: // Stop
        IF (statusWord AND 1) = 1 AND controlWord <> 0 THEN
            controlWord := 0;
        END_IF
    1: // Begin enabling sequence (1-4)
        IF currentOperation = operation THEN
            controlWord := 6;
            iState := iState + 1;
        END_IF
    2: // Ready to switch on
        IF (statusWord AND 545) = 545 THEN
            controlWord := 7;
            iState := iState + 1;
        END_IF
    3: // Switched on
        IF (statusWord AND 563) = 563 THEN
            controlWord := 15;
            iState := iState + 1;
        END_IF
    4: // Operation enabled
        IF (statusWord AND 567) = 567 THEN
            iState := iState + 1;
        END_IF
    5: // Moving | TODO: Add position error correction
        IF NOT (statusWord AND 567) = 567 THEN
            iState := 0;
        END_IF
END_CASE
```

Figure B.1: Main code of the function block

```
METHOD PUBLIC setVelocity
VAR_INPUT
    VEL : INT := 200;
END_VAR

IF iState = 5 AND operation = 9 THEN
    targetVelocity := VEL;
ELSIF iState = 0 THEN
    operation := 9;
    targetVelocity := VEL;
    iState := 1;
END_IF
```

(a) setVelocity() method

```
METHOD PUBLIC stop

IF iState > 0 THEN
    iState := 0;
END_IF
```

(b) stop() method

```
METHOD PUBLIC quickStop

controlWord := (controlWord AND 2#1101111011) OR 2#10;
```

(c) quickStop() method

```
METHOD PUBLIC getEncoderTicks : DINT

getEncoderTicks := currentInternalPosition;
```

(d) getEncoderTicks() method

```
METHOD PUBLIC faultReset

IF iState = -1 THEN
    controlWord := 128;
    flagResetFault := TRUE;
END_IF
```

(e) faultReset() method

Figure B.2: Public methods of the function block

Appendix C

BMS interface

Figure C.1 contains the main code of the function block of the interface developed to communicate with the battery management system of the mobile module.

```
FUNCTION_BLOCK PUBLIC BMS123Smart_SerialReceiver
VAR_OUTPUT
    battery : BMS123Smart_BatteryStats;
    error : RTS_IEC_RESULT;
END_VAR
VAR
    iState : USINT := OPEN;
    hCom : RTS_IEC_HANDLE := RTS_INVALID_HANDLE;
    bytesRead : UDINT;
    checksum : USINT;
    str : ARRAY[0..(SZ_STR - 1)] OF BYTE;
    i : UINT;
END_VAR
VAR CONSTANT
    HALT    : USINT := 0;
    OPEN    : USINT := 10;
    LISTEN  : USINT := 20;
    SZ_STR  : USINT := 58;
    CS : COM_Settings := (
        sPort := SYS_COMPORT2, // BeagleBone Black UART1_RXD =
            SYS_COMPORT2; UART1 unavailable?
        byStopBits := SYS_ONESTOPBIT,
        byParity := SYS_NOPARITY,
        ulBaudrate := SYS_BR_9600,
```

```
        ulTimeout := 10,
        ulBufferSize := 0
        );
END_VAR

CASE iState OF
    HALT:

    OPEN:
        hCom := SysComOpen2(pSettings := ADR(CS), pSettingsEx :=
            0, pResult := ADR(error));
        IF hCom <> RTS_INVALID_HANDLE THEN
            iState := LISTEN;
        ELSE
            iState := HALT;
        END_IF

    LISTEN:
        bytesRead := bytesRead + SysComRead(hCom:= hCom,
           pbyBuffer:= ADR(str)+bytesRead, ulSize:= SIZEOF(str)-
           BYTEsRead, ulTimeout:= 10, pResult:= ADR(error));
        IF bytesRead = SZ_STR THEN
            checksum := 0;
            FOR i:= 0 TO SZ_STR - 2 DO
                checksum := checksum + str[i];
            END_FOR
            checksum := checksum AND 255;
            IF checksum = str[SZ_STR - 1] THEN
                update();
            END_IF
            FOR i:= 0 TO SZ_STR - 1 DO
                str[i] := 0;
            END_FOR
            bytesRead := 0;
        END_IF
END_CASE
```

Figure C.1: Main code of the function block