

University of Bologna

School of Engineering and Architecture

Master Degree in Automation Engineering

AUTOMATION SOFTWARE AND DESIGN PATTERNS

IMPLEMENTATION OF AN AUTOMATIC MACHINE DEMO WITH B&R'S ACOPOS6D

Candidate:

Rambaldi Nicola

University Tutor:

Prof. Eng. Gianluca Palli

Company Tutors:

Eng. Andrea Guglielmi

Eng. Luca Dallari

Session III

Academic Year 2021/22

To my family, my friends and Federica, my girlfriend, for the counseling and support of these months. A special thanks to my professor, for the precious help provided during the writing of this paper. A final acknowledgement to B&R Industrial Automation and all its employees for the opportunity that I had to know more in depth the automation world and in particular to Luca and Andrea, always available for assistance and supervision during this journey that helped me a lot to grow both personally and professionally.

Abstract

The world of industrial automation has changed a lot in the last years, with machine-builders that always look for innovative technologies that are able to simplify the production process by also introducing new degrees of flexibility, which is the focus of the newest trend in this sector, in particular the possibility to change on the fly the size of batches to produce products different in size and shape in order to meet the high production standards that rise continuously nowadays.

This thesis project is the summary of a journey in the B&R Industrial Automation company, one of the world leader automation suppliers, with a focus on its newest and most revolutionary technology: ACOPOS6D. The first part is dedicated to a description of the company, its traditional products and Automation Studio, the software development tool on which every automation application can be easily implemented, with a focus on the possibility to manage a very wide range of different devices. The next part of this thesis highlights the three transportation systems developed by B&R that exploit the magnetic force in order to grant an unprecedented flexibility and compactness to an automatic machine, with a focus on their advantages with respect to traditional transportation systems like conveyor belts and the difference between the use of magnetic attraction and magnetic levitation. The last part of the project, instead, is focused on an application developed on the ACOPOS6D system during the internship in order to highlight the potentialities of the system, extract some useful data on its functioning and show that an actual automation process is implementable with the system, with high degrees of robustness and long term reliable functioning. The implemented application is developed in the deeply analyzed Template framework, a new standard for automatic machines development proposed by B&R.

Contents

Abstract	v
List of Figures	ix
Introduction	1
1 Magnetic levitation applications	3
1.1 Magnetic levitation principle	3
1.2 Magnetic levitation applications	4
1.2.1 Magnetic levitation trains	4
1.2.2 Rocket launch	4
1.2.3 Magnetic levitation fans	5
1.2.4 Heart pumps	6
1.2.5 Food quality analysis	6
1.3 Magnetic levitation in industrial automation	7
1.3.1 Semiconductors manufacturing	7
1.3.2 Flexible assembly automation	8
1.3.3 Magnetic levitation transport system for the OLED display evaporation process under vacuum	9
1.3.4 Current state of the technology	10
2 B&R Overview	13
2.1 Automation Studio	14
2.2 Automation Runtime	15
2.3 POWERLINK	16
2.4 mapp Technology	17
2.4.1 mapp View	17
2.4.2 mapp Services	20
2.5 Motion Control	23

3	ACOPOSTrak and SuperTrak overview	27
3.1	ACOPOSTrak	27
3.1.1	Technology overview	28
3.1.2	High speed Diverters	29
3.1.3	Washdown and cooled designs	31
3.2	SuperTrak	31
3.3	Trak technologies overview	32
4	ACOPOS6D	35
4.1	Hardware	35
4.1.1	PLC X20CP3685	36
4.1.2	Planar Motor controller	37
4.1.3	Segments	38
4.1.4	Shuttles and motion profiles	39
4.2	Application fields and advantages	41
4.3	Software: mapp 6D	43
5	B&R Template and its advantages	45
5.1	Modular structure	45
5.2	Recipes and configuration files handling	47
5.3	Alarms handling	48
5.4	HMI and other tools	49
5.4.1	CreateFilePath tool	49
5.4.2	RenameFull tool	50
5.4.3	HMI VC and mapp View	51
5.4.4	Adding a device	54
6	Demo overview: ACOPOS6D device	57
6.1	mapp 6D function blocks	58
6.2	Initialization	61
6.3	Phasing	63
6.4	Stations concept	64
6.5	Shuttle recognition function block	65
6.5.1	Inputs	66
6.5.2	Working principle and outputs	66
6.6	Working stations	67
6.6.1	Load station	69
6.6.2	Buffer station	72
6.6.3	Labeling station	73
6.6.4	Shaker station	74
6.6.5	Index station	75

6.6.6	Capping station	76
6.6.7	Unloading station	77
6.6.8	Direction station	77
6.7	Configuration and recipe files	78
6.8	Scene Viewer simulation	79
6.9	Experimental results	81
	Conclusions and future developments	85
	Bibliography	89

List of Figures

1.1	SCMaglev train.	4
1.2	Design for the Maglev rocket accelerator.	5
1.3	Magnetic levitation fan.	5
1.4	Magnetic levitation heart pump.	6
1.5	Magnetic levitation fluid density sensor.	7
1.6	Traditional RCC.	8
1.7	MEISTER project and prototype	9
1.8	Components of the Maglev transport sytem.	10
2.1	B&R products	13
2.2	Generic project in Automation Studio	14
2.3	Different task classes	15
2.4	Master and slaves in POWERLINK	16
2.5	mapp Technology packages	17
2.6	mapp View application example	18
2.7	mapp View application MainContent development	19
2.8	Edge alarm behaviour	21
2.9	Persistent alarm behaviour	22
2.10	PLC-drive-motor scheme	23
2.11	Cascade control scheme	24
2.12	Motion control loop scheme	25
2.13	ACOPOSmulti and ACOPOS P3	25
2.14	ACOPOSmicro and ACOPOSmotor	26
2.15	ACOPOSInverter	26
3.1	Shuttles and stator in ACOPOSTrak	28
3.2	Stator segments in ACOPOSTrak	29
3.3	Diverters in ACOPOSTrak	29
3.4	ACOPOSTrak application simulated in SceneViewer	30

3.5	SuperTrak example	32
4.1	ACOPOS6D topology.	36
4.2	ACOPOS6D example layout with two segments and one shuttle.	36
4.3	PLC X20CP3685	37
4.4	Planar Motor controller	38
4.5	ACOPOS6D segments front and rear view.	38
4.6	ACOPOS6D example layout in Scene Viewer.	39
4.7	ACOPOS6D shuttle.	39
4.8	ACOPOS6D motion buffers.	40
4.9	Zone and fence for safe shuttle removal.	41
4.10	Pharmaceutical and food and beverage sectors in which ACO- POS6D can be applied	42
4.11	mapp 6D layout creation page	43
5.1	Template's modular structure.	46
5.2	Recipes and configuration handling structure.	48
5.3	Alarms handling structure.	49
5.4	CreateFilePath tool.	50
5.5	RenameFull tool.	50
5.6	Main page of the Template HMI.	51
5.7	Debug page of the Template HMI.	52
5.8	Shift register and setup pages.	53
5.9	Recipe handling page.	53
5.10	Alarm page.	54
6.1	Cyclic.st action of the Empty Device	57
6.2	MC_BR_AsmGetShuttle_Acp6D function block	58
6.3	MC_BR_ShReadInfo_Acp6D function block	58
6.4	MC_BR_AsmPowerOn_Acp6D function block	59
6.5	MC_BR_AsmStop_Acp6D function block	59
6.6	MC_BR_MoveInPlaneAsync_Acp6D function block	60
6.7	MC_BR_MoveInPlane_Acp6D function block	60
6.8	MC_BR_RotaryMotion_Acp6D function block	60
6.9	MC_BR_MoveArc_Acp6D function block	61
6.10	Shuttle recognition algorithm.	61
6.11	Shuttle order algorithm.	62
6.12	Initial positions of the 28 shuttles.	63
6.13	Station structure.	64
6.14	Acp6DStat function block.	65
6.15	ShuttleInfoType and Position structures.	66

6.16	Function block's code.	67
6.17	Shuttle deadlock.	67
6.18	Shuttles in obstacle state.	68
6.19	Working stations: load (red), labeling (blue), shaker (pink), capping (green) and unloading (white) and routing stations (yel- low).	68
6.20	Loading station's recognition and processing step.	69
6.21	Shuttles' routing in the loading station.	70
6.22	Loading station's routing step.	70
6.23	Loading station's routing step part 2.	71
6.24	Loading station's routing steps part 3.	71
6.25	Loading station's ending code.	72
6.26	Shuttles' routing in the buffer station.	72
6.27	Possible routings of shuttles in the labeling station.	73
6.28	Shuttles' routing in the shaker station.	74
6.29	Shuttles' routing in the index.	75
6.30	Routing for the capping station.	76
6.31	Routing for the unloading station.	77
6.32	Routing for the direction station.	78
6.33	Configuration file.	78
6.34	Recipe file example.	79
6.35	Scene Viewer simulation of the demo.	80
6.36	Position and color bindings for the first shuttle.	81
6.37	Physical layout after phasing.	81
6.38	New positions for the loading station.	83
6.39	System with 32 shuttles.	83

Introduction

The case study in this thesis project is focused on the development of an automatic machine with the B&R's ACOPOS6D technology. This technology aims to revolutionize the sector of transportation systems in automation industries. The project shows an automation system that takes groups of four bottles containing some liquid as input and then processes the bottles singularly or in groups of two depending on the application. Then the machine groups the bottles in a pattern of eight in order to facilitate the extraction via a robotic gripper.

ACOPOS6D is a system that exploits magnetic levitation in order to move shuttles, on which products are transported, at high speeds with extremely precise control, over a layout. This technology implements a friction-free contactless transport system that is able to move products in any direction on a plane and not simply in one specified direction like for conventional transport systems. Every shuttle's trajectory on the layout is dependent only on the implemented code logic of the machine and it is obviously reprogrammable without having to change anything in the mechanic of the machine, allowing unprecedented degrees of flexibility and reprogrammability. Since this device implements a magnetic levitation, a brief summary of what this technology is and how it has been developed and exploited up to date, especially in the industrial automation field, is presented at the beginning of the thesis. It is clear, after some researches, that in 2022 magnetic levitation is applied in very specific fields (robot's grippers for example) and most of the developed applications is at the prototype stage. ACOPOS6D, on the other hand, is adapt for a lot of different sectors and so it is the first general purpose magnetic levitation transportation system that is ready to be used in the construction of actual automatic machines and tested for long time operation. The aim of this thesis work is actually this: implement a software to control an application that is close to the high dynamics standards requested by today automation. The

technology allows, thanks to its six degrees of freedom, to reduce dramatically the footprint of the machine and is maintenance free because of its contactless moving parts. ACOPOS6D is not the only B&R's mechatronic system that make use of magnetism, so in the central part of this thesis ACOPOSTrak and SuperTrak technologies are presented, with a focus on their differences with ACOPOS6D that are highlighted in order to understand why the magnetic levitation system is an evolution of these products, that exploit magnetic attraction instead of magnetic levitation.

The developed project is presented in Chapter 6, while the central part of this work focuses on the company and what instruments have been provided in order to realize the developed application. B&R is an automation supplier that provides the whole package to its machine constructor customers, from hardware to software and from drives to mechatronic systems. The project has been developed on Automation Studio: B&R's software development tool that allows to program and manage the entire hardware that composes the machine (B&R's and third party devices) easily, so it is presented as well as the B&R's Real Time Operating System, Automation Runtime. The project is also composed of an HMI, an alarm handler and a recipes and configuration files' management system, with Chapter 2 that focuses on the company and its tools that are able to handle these things.

The project is developed in the B&R's Template project framework, a new standard in the construction of the software configuration of an automatic machine proposed by B&R and in particular by Bologna's office, where this thesis project has been realized. B&R's Template project provides the skeleton of an automatic machine, with a Main Logic already implemented that is able to activate and control the mechatronic units of the machine. These units are called devices that the user has to program in order to create its custom application. The Template provides also other interesting functionalities that aim to reduce the effort that the customer has to produce in order to configure and synchronize the devices, leaving the possibility to entirely focus on the code and the personalization of the machine. The project developed in this thesis aims to create a device fully compatible with the Template in order to show how ACOPOS6D is entirely integrable with the B&R ecosystem.

Magnetic levitation applications

This thesis project is based on a magnetic levitation system used in the automation industry. The first part of this thesis will focus on the magnetic levitation technology and its applications in mankind history, focusing on the most used industrial automation applications [1].

1.1 Magnetic levitation principle

Magnetic levitation is a technique that allows an object to be suspended in air with no support other than magnetic fields. Magnetic fields in this case are used in order to overcome the gravitational force that pulls the object to the ground and tuned in such a way that the object can move in space without the need of wheels, axles and transmission.

There are two different kinds of magnetic levitations: the diamagnetic one and the one typical of superconductors. The diamagnetic levitation necessitates of strong magnetic fields and is able to lift only small and light objects, while the one actually useful for practical and useful applications is the one obtained in superconductors thanks to the Meissner effect. The Meissner effect brings the magnetic field's lines to be expelled from the superconductor, allowing the magnet to levitate, spin and bounce without losing its control.

1.2 Magnetic levitation applications

1.2.1 Magnetic levitation trains

The most famous and well known application of magnetic levitation is the operation of magnetically levitated trains, the first fundamental innovation in railroad technology since the invention of railways itself. Maglev trains are powered by noncontact magnetic levitation, guidance and propulsions systems and, at no time, there is contact between the vehicle and the guideway. The possibility to have the classical wheel-on-rail technology replaced by wear-free electronics allows to achieve speeds of motion never reached before with a record, that has been achieved by the japanese SCMaglev, of 603 km/h .

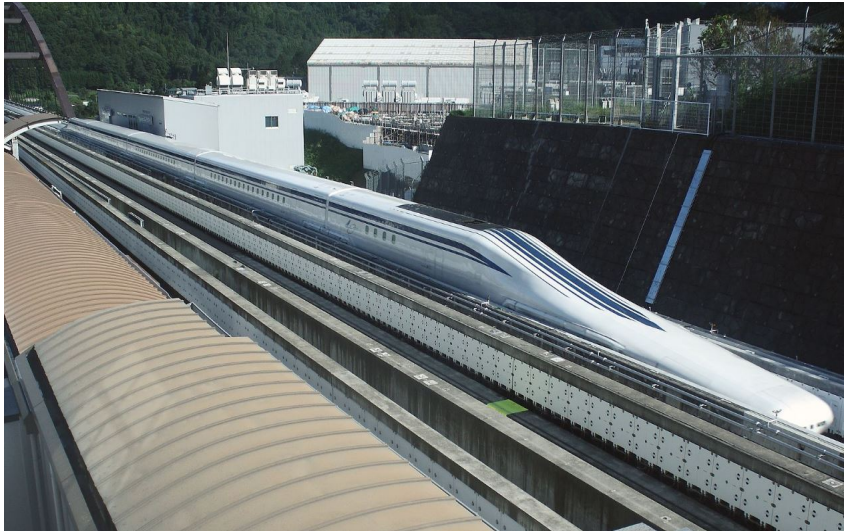


Figure 1.1: SCMaglev train.

1.2.2 Rocket launch

Another application field for magnetic levitation technologies is the one of the propulsion systems for space rockets. A magnetic levitation track is up and running at NASA's Marshall Space Flight Center in Huntsville, Ala, USA. The idea that scientists are developing would be very useful for the initial phase of the launch, where the rocket has to break free from Earth's gravity. Some advantages that are provided by this particular choice are the speed, that would reach up to almost 1000 km/h and the convenience of electricity with respect to rocket fuel, since electricity does not need to be added to the rocket, increasing its weight, and its cheaper, reducing the cost of the rocket up to 20%.



Figure 1.2: Design for the Maglev rocket accelerator.

1.2.3 Magnetic levitation fans

The possibility to not have contact between the moving and the standing parts of a magnetic levitation system allows this technology to be exploitable for the realization of fans that provide superior performance, low noise and longer life due to the zero friction present between shaft and bearing. Very important feature for this fans is the possibility to stabilize the rotative motion in order to reduce shuddering and vibrations, a typical problem for classical fans whose rotation trajectory is not controlled. The possibility to have no friction allows to avoid the long term abrasions that the shaft causes on the bearings of traditional fans, allowing, as said before, to increase the fan's life.



Figure 1.3: Magnetic levitation fan.

1.2.4 Heart pumps

Artificial heart pumps are a fundamental medical tool that require small structure, low energy consumption and a long life. These characteristics all belongs to a magnetic levitation system, that also grants the possibility to avoid blood pollution due to the rolling and sliding of the bearings that enter in contact with blood cells.

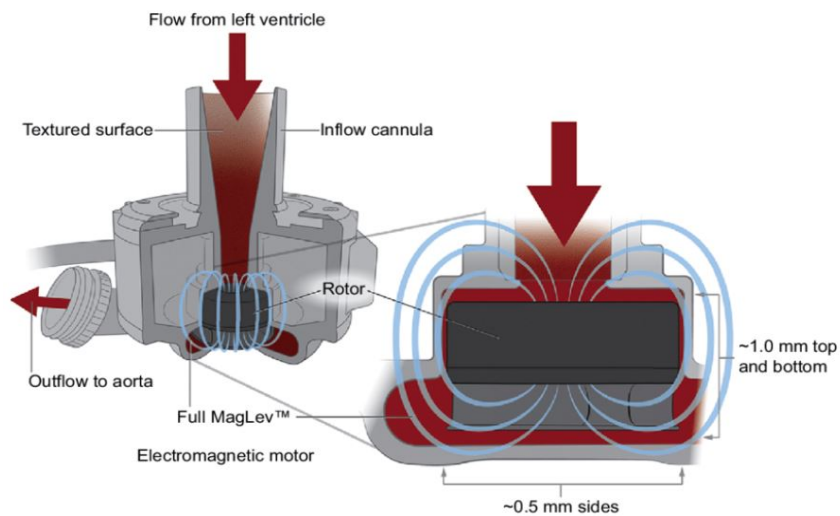


Figure 1.4: Magnetic levitation heart pump.

1.2.5 Food quality analysis

A sensor based on magnetic levitation can be very useful for the food industry, in particular for the analysis of food and beverages. In these applications it is important to know the density of fluids (content of sugar in soft drinks, amount of alcohol in wine, amount of salt in irrigation water and so on and so forth) and this can be done by using a sensor made of a fluid-filled container with magnets at each end in which samples of different materials can be placed. The distance that these samples travel in the fluid can provide a measure of their density.

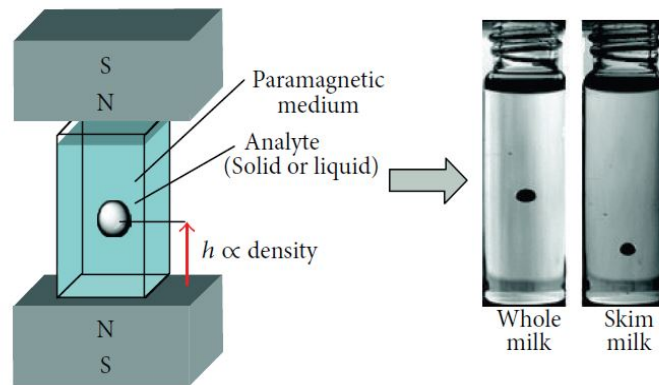


Figure 1.5: Magnetic levitation fluid density sensor.

1.3 Magnetic levitation in industrial automation

Magnetic levitation technology is something that can be, and is, very useful for industrial automation applications because they grant the possibility of a very high position precision and no contact forces are in place, granting to avoid friction and wear of the components and making these systems very adapt to applications in clean rooms where friction would cause the production of particles and in harsh environments too, because they can be completely coated in order to withstand high temperatures and low pressures.

1.3.1 Semiconductors manufacturing

The manufacturing of semiconductors is a critical task that requires specific characteristics of the involved machinery in order to be realizable. The required accuracy and repeatability is dictated by the size of the features presents in the produced semiconductor and this are in the micron range, requiring high precision positioning that a levitated manipulator can provide.

In 1990 Ilene Busch-Vishniac [2] proposed a series of tasks that are crucial in the manufacturing of semiconductors and demonstrated how a rather simple system is able to perform these tasks in a satisfactory and efficient way. Usually in semiconductors manufacturing it is required to transport materials over large distances with high precision, resulting in a motion profile in which the scale of the three dimensional workspace is several orders of magnitude larger than the required resolution, with conventional technologies that struggle to satisfy this requirement while magnetic levitation allows this possibility. Up to that point, semiconductor wafers were transported using belts or large multiple

links robots, which had some drawbacks and in particular they are hard to reprogram (robots) or substitute (belts) whenever the factory line needs to be changed or modified. Another key problem in this task is that a great speed is required during the motion phase and very gentle and delicate forces are needed during the loading and unloading phases, all perks that are provided by magnetic levitation systems, that are able to move wafers very quickly without inflicting damage on them.

Another important task that has to be carried out during semiconductors manufacturing is the testing of bonds between different parts. The traditional bonds testers were off-line, destructive testers that broke the bond in order to understand the force at which it was irreparably damaged, or non-destructive testers that still work off-line, slowing down the production. The proposed solution is to use magnetic levitation in order to create an on-line bond tester that is able to position accurately and quickly on the line and deliver a known force in order to test the bond without slowing down the production.

1.3.2 Flexible assembly automation

In 2016 Masahiro Tsuda, Toshiro Higuci and Shigeki Fujiwara proposed an application of magnetic levitation to assembly automation in order to achieve actuation and force sensing, and by distinguish with respect to the classical uses of magnetic levitations that are suspension-only systems [3]. The idea is based on the design of a magnetic levitation system that is innested between the wrist and the end effector of the manipulator, allowing to levitate and control the end effector and obtaining a friction-free, backlash-free mechanism for a multi-DOF motion. The main advantages of this approach are: programmable compliance and viscous damping directly by software, precision actuation that can be obtained by simply modifying the magnetic forces that act on the end effector and multi-axis force sensing without the use of any additional sensory equipments.

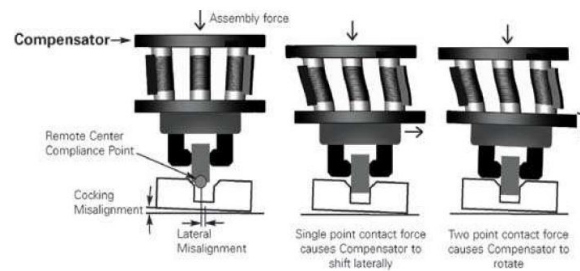


Figure 1.6: Traditional RCC.

The idea is to create a Remote Center of Compliance (RCC, Figure 1.6), usually used in order to decouple torques and forces that will respectively generate simply rotations and translations of the object in contact with the end effector [4], but this one will exploit magnetic levitation. The fact that magnetic levitation is used allows also to increase the life of the end effector, that will not be ruined by friction. This project aimed to develop something different with respect to the system described in Section 1.3.1, that not only levitated with high precision but also moved for long distances with open-loop control during the transfer phases of the wafers. In Figure 1.7 it is shown the prototype realized, the MEISTER.

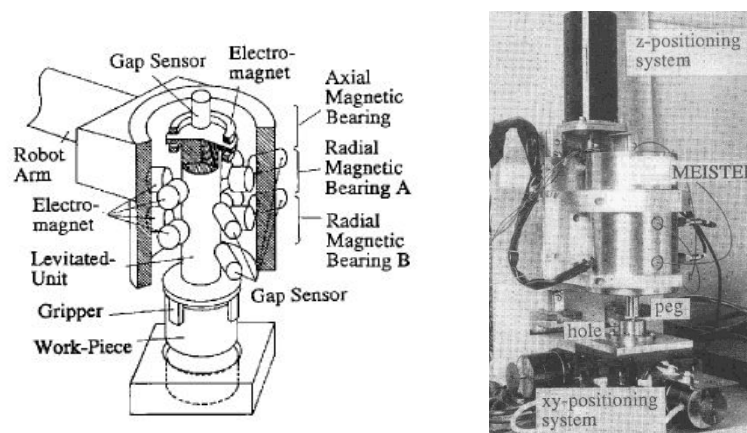


Figure 1.7: MEISTER project and prototype.

1.3.3 Magnetic levitation transport system for the OLED display evaporation process under vacuum

The manufacturing process of OLED displays requires a high accuracy and a highly clean transport system because during the evaporation process, organic materials are inherently sensitive to contamination and uniform deposition of an organic layer is important. Roller-based transfer systems are not suitable for OLED evaporation processes because the roller-based system has a poor driving performance due to nonlinear factors such as friction and generates a large amount of dust. In 2018 a magnetic levitation transport system for this application has been proposed by Chang-Wan Ha, Chang-Hyun Kim and Jaewon Lim [5]. The proposed system is able to transport the display with high precision while not producing any dust that cannot be tolerated during the evaporation system.

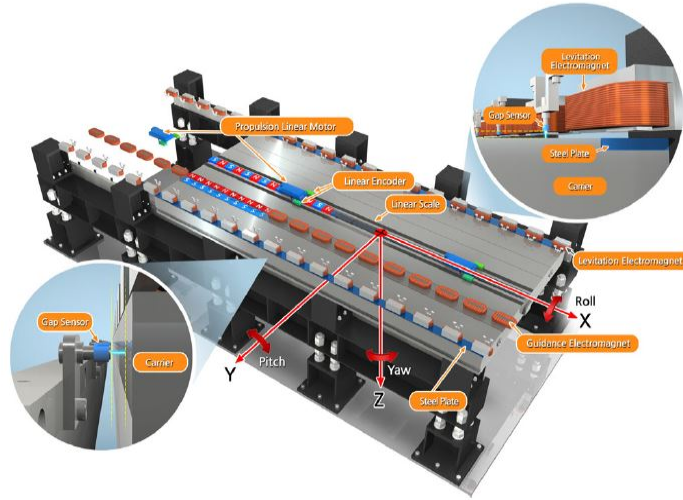


Figure 1.8: Components of the Maglev transport system.

Every device that contains a power supply is anchored to the fixed frame in Figure 1.8 in order to minimize the thermal expansion of the carrier during the evaporation process. At each time the carrier faces 28 levitation electromagnets that share the load brought by the levitation of the carrier's weight. Some guidance electromagnets are placed on the other side with respect to the carrier and, applying a current in different directions will result in the carrier, that mounts a permanent magnet, to move left or right. The system was tested for 12 hours and granted good control performances during time span, allowing to consider this a technology capable of, in the future, increase the production of OLED displays and reduce their prices due to the lower number of defects with respect to the currently used transportation systems.

1.3.4 Current state of the technology

In 2022, Lei Zhou and Jingjie Wu [6] wrote a review of the developments in the application of magnetic levitation to precision motion systems and their current state. There are a lot of different applications of these systems that have been proposed, especially in the last twenty years due to the contactless, frictionless and cleanness of this technology, together to the possibility to have high precision in positioning in all possible 6 degrees of freedom, and the possibility to accurately sense the force applied on the system in each one of these degrees of freedom. These applications, however, are mostly prototypes and ideas that are not yet applied in large scale automation production, but aim to be in the foreseeable future. In this thesis project the first system that deploys magnetic levitation for general purpose automation is highlighted:

B&R's ACOPOS6D, a system already tested on automation applications that will revolutionize the concept of transportation system, allowing an unprecedented degree of flexibility in industrial automation applications.

B&R Overview

B&R Industrial Automation is a company belonging to the ABB group, founded in 1979 in Eggelsberg, Austria [7]. It is leader in the production of a wide range of automation products with a particular focus on PLCs and drives and serves customers on a very wide range of sectors: from packaging to medical applications, from ceramics to metal handling, up to purely robotic applications. The company is based on the concept of integration: every B&R product is fully integrated in Automation Studio, an environment that B&R provides to its customers in order to simplify the procedures of configuring and setting up every device, leaving the programmer free to focus on the coding and logical parts of the project. In picture 2.1 the wide range of different products offered by the company is shown:



Figure 2.1: B&R products.

B&R products ranges from the programming tool Automation Studio to the operating system that runs in every B&R controller, from motion control systems such as drives, from I/O components to vision systems and then also PLCs, industrial PCs and power panels that combine an HMI interface with

an integrated controller. The company’s portfolio covers all the needs of a machine builder and, from the firmware of hardware modules to the development tool, allows the full control of devices granting the highest performances. In this chapter the most relevant products for this thesis project will be presented.

2.1 Automation Studio

Automation Studio is a software environment fully compatible with every B&R product and allows also to introduce in the project third party devices. This guarantees to the customer the highest number of possibilities during the machine development. Automation Studio’s *Logical View* is the place in which the entire logic of the machine can be implemented since it is compliant with all the main IEC programming languages (ST, SFC, Ladder diagrams and more like ANSI C and C++). The code can be organized in tasks (see more at Section 2.2) that can use different languages but are perfectly able to function together since every programming language in Automation Studio is able to access the same data types, libraries and variables. With Automation Studio it is possible to follow the machine in all its phases, from the actual development up to the commissioning and then the diagnostic in the end-user factory, without the necessity to implement control interfaces from scratch.

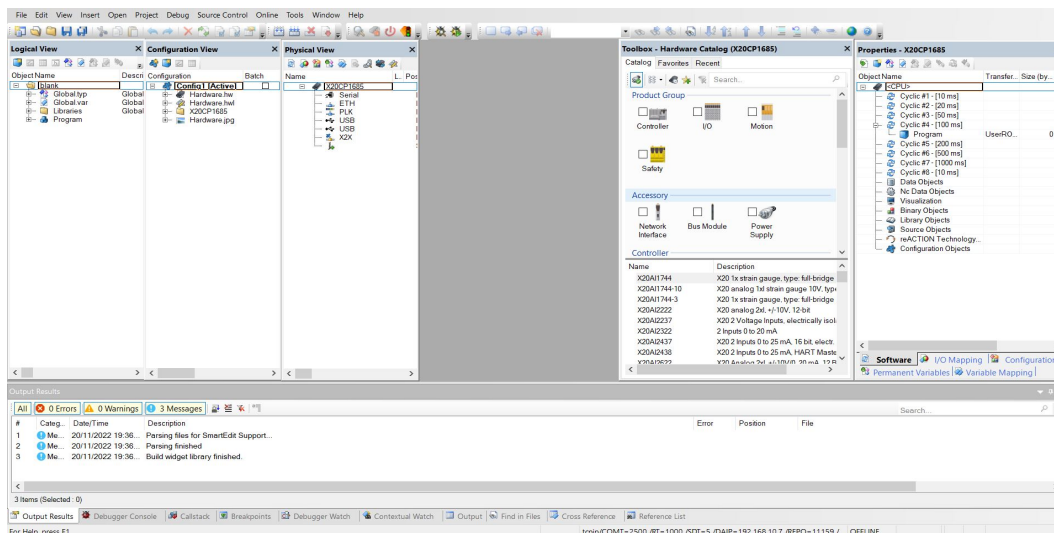


Figure 2.2: Generic project in Automation Studio.

Automation Studio allows to design the entire hardware configuration used in the project in order to simulate it even without the presence of the actual hardware, and this is possible with any B&R device. The *Configuration View* section also allows the creation of more than one configuration for the same

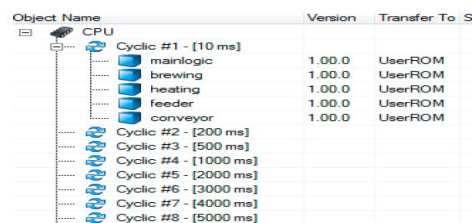
project on which the tasks to be executed on each configurations can be decided every time, allowing a smart and fast approach that can be used for example when the customer has more than one machine with the same logic but different hardware components.

In Figure 2.2 the five main areas of Automation Studio are shown. As just said, the *Configuration View* is used to select which tasks, libraries and hardware configuration will be used in the project, but also to collect all the configuration files for the many mapp technology packages (software packages that cover functionalities like motion, recipes, alarms, users and more) that B&R provides to the customers (Chapter 2.4 for further informations about this technology). The other two main areas in Automation Studio are the *Logical View* and the *Physical View* that, respectively, are used to access to tasks, code and variables, and to build the actual hardware that composes the project. The *Toolbox* is used to insert in the project any kind of hardware, software or coding component that the customer can use, while the *Property Window* allows the configuration of the specific properties that each element is provided with.[8]

2.2 Automation Runtime

Automation Runtime is a deterministic real-time operating system that serves as the software platform for B&R's entire product range. Automation Runtime provides services for freely configuring and troubleshooting the target system and for executing programs and it is fully integrated into all B&R target systems, supports all hardware platforms and allows the customer to avoid managing low-level memory and allows to separate the choice of the controller with respect to the code development.

Automation Runtime is able to meet high demands in terms of deterministic behaviour and speed and so, in order to exploit this performance advantage in the application, an abstraction layer exists over the real-time operating system, ensuring the possibility to change OS version without having to make adjustments in the application.



Object Name	Version	Transfer To	S
CPU			
Cyclic #1 - [10 ms]			
mainlogic	1.00.0	UserROM	
brewing	1.00.0	UserROM	
heating	1.00.0	UserROM	
feeder	1.00.0	UserROM	
conveyor	1.00.0	UserROM	
Cyclic #2 - [200 ms]			
Cyclic #3 - [500 ms]			
Cyclic #4 - [1000 ms]			
Cyclic #5 - [2000 ms]			
Cyclic #6 - [3000 ms]			
Cyclic #7 - [4000 ms]			
Cyclic #8 - [5000 ms]			

Figure 2.3: Different task classes.

In figure 2.3 the eight task classes to which each task can be assigned are shown. As already pointed out, Automation Runtime is real-time, meaning that tasks are executed cyclically and must be completed inside specified time bounds, otherwise the OS will trigger a cycle time violation. The cycle times of each task class can be configured entirely and these classes have different priority levels: there are eight priority levels, one for each task class (increasing from eight to one) and this means that tasks can be organized in order to make the most important tasks interrupt the least important ones and be executed first at each time cycle.[9]

2.3 POWERLINK

POWERLINK is an evolution of the classic Fieldbus technology based on the Ethernet technology with incredible performances and real-time functionings used to transfer process data in automation systems. The transmission speed is up to 100 Mbit/s and this allows to combine in a unique network the most complex activities. POWERLINK is an open and IEC standard technology around the globe and it is used by B&R as the primary way to exchange data between its products.

POWERLINK uses a request/response, or polling approach: one master, or managing node controls the access to the network while the slaves, or controlled nodes follow its commands and only transmit data upon request by the managing node. In this way POWERLINK grants a deterministic transmission of data in the bus, avoiding collisions of said data, without introducing delay in the transmission. This structure can be seen in Figure 2.4.

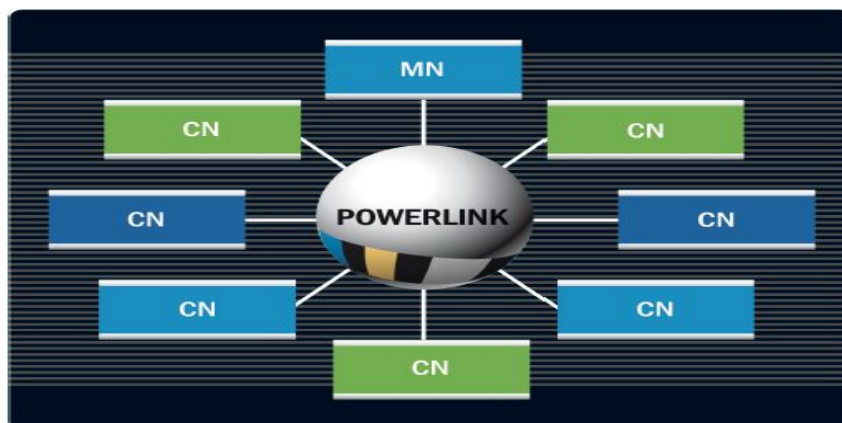


Figure 2.4: Master and slaves in POWERLINK.

A powerlink managing node allows up to 239 controlled nodes, with node 240

that is always the master, the minimum cycle time is 200 μs , with each node that can process up to 1490 bytes.

The POWERLINK cycle during which there is the exchange of data is divided into two distinct phases: the isochronous and asynchronous phase. During the isochronous phase, process data from the network nodes is transferred cyclically, while in the asynchronous phase two nodes are allowed to exchange non-cyclic data between themselves.[10]

2.4 mapp Technology

mapp (modular application) Technology is a collection of services provided by B&R to its customers that is aimed to simplify to the maximum degree the life of the programmer that will have to focus solely on the writing of the machine logic without having to write code for the configuration of recipes, alarms, the HMI (Human Machine Interface) and things of that sort, but allowing to shift all the focus on the creation of the machine's logic. mapp Technology is composed of a large number of plug-in packages for Automation Studio that touch every possible aspect of an automation project, the ones that are of interest for the application developed in this thesis will be briefly described in this chapter.

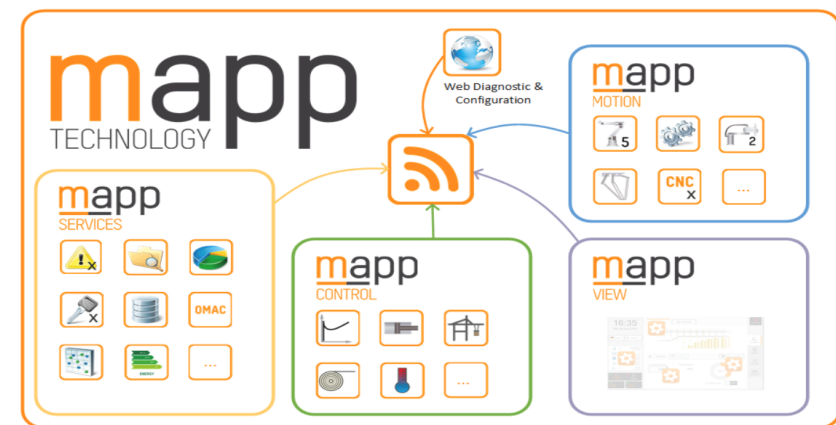


Figure 2.5: mapp Technology packages.

2.4.1 mapp View

mapp View is the mapp package that manages the implementation of an HMI for an automatic machine. It is based on HTML 5, the go to between web development programming languages, and for this reason, the HMI created with *mapp View* can be easily accessed via web browser (HMI client) where a *mapp*

View server will publish it at the link `<target IP address:port number/index.html?visuId= visualization ID>` (the visualization ID is specified in a file `.vis` created in the Configuration View). This grants the possibility to access the HMI (and control the machine) without the use of any particular visual component and via any kind of device provided with a web browser (a PC but also a smartphone or a tablet). This allows also to exploit the multitouch technology with which these devices are provided and also to access multimedial files like videos, PDFs (for example this grants the possibility to include in the HMI some useful manuals about some machine components) or the B&R main diagnostic tools: the System Diagnostic Manager and the B&R Cockpit, directly from the HMI.

The base components of an HMI developed with *mapp View* are the widgets: a wide range of html components that are provided by B&R and allow the software developer to design a complex and pleasing to the eye HMI without the need to write a single line of html code and by simply selecting the desired widgets, configuring some intuitive and simple properties and linking the desired process variables to the widgets that have to react to their behaviour in time.

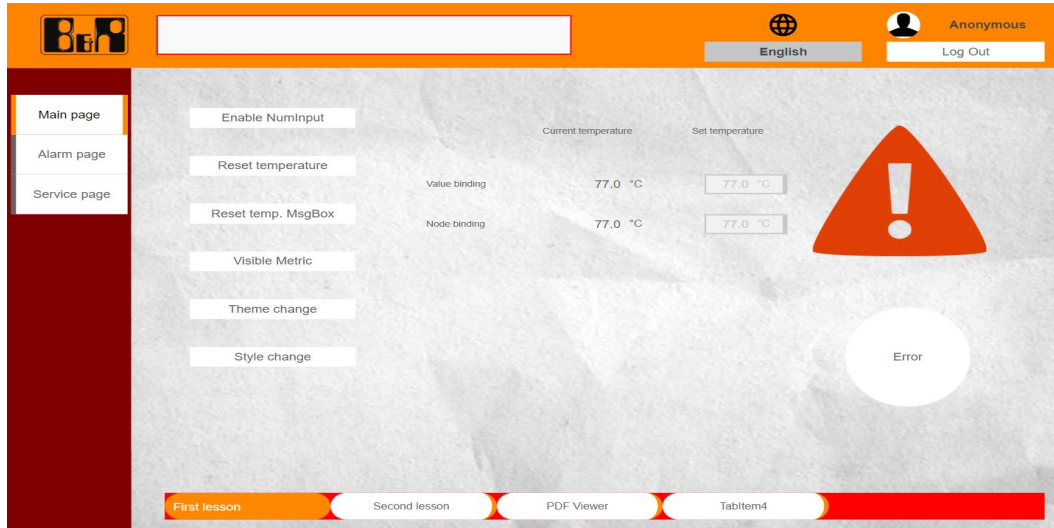


Figure 2.6: mapp View application example.

An HMI created with *mapp View* is made of a series of pages that refer to one or different layouts created in the Logical View. Each layout can be split in different areas and each area will be binded to a content that actually defines what the HMI will show and contains all the widgets that are used in the application. In Figure 2.6 we can see an example of HMI developed with *mapp*

View composed of three pages where each page is divided in three areas binded to a top, left and main content. This example contains only a few of the many widgets that are provided by *mapp View* like for example a language selector, a log-in widget (top content), a navigation bar that allows the switch between the different pages and, in the main page, some toggle buttons, numeric inputs and outputs and also simple images that can be visualized in the HMI (B&R logo and warning signal).

A strenght of *mapp View*, as said before, is that the development of the HMI is of the *what you see is what you get* kind, as we can see in Figure 2.7 where it is shown the development page in Autmation Studio of the main content, absolutely identical to what will be visualized in the end, and completely void of html programming:

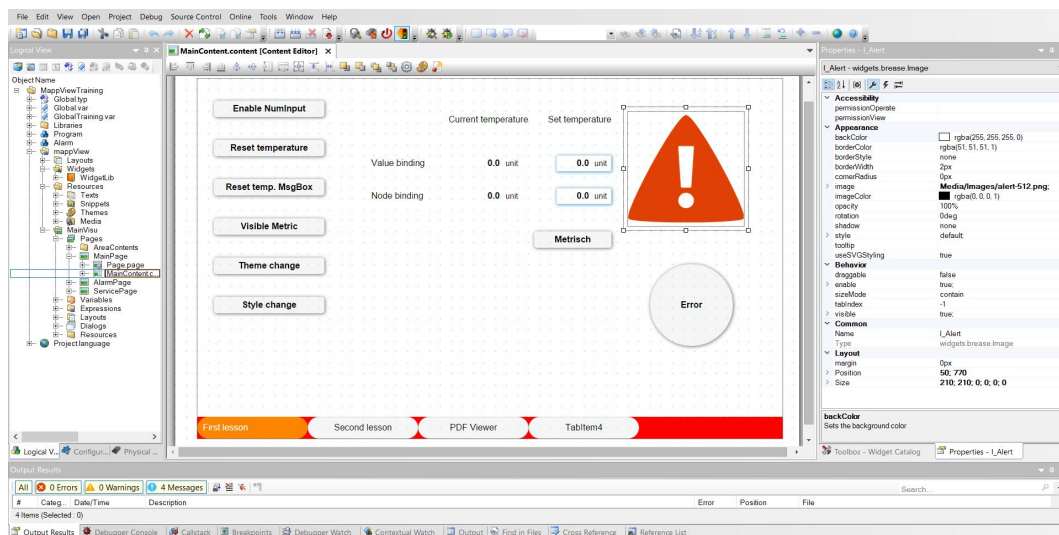


Figure 2.7: mapp View application MainContent development.

Of particular interest for an automatic machine HMI are bindings and event bindings. A binding is essentially the link of a process variable, that changes value according to the machine logic, to a widget. In particular, in the example shown before, it can be seen with the numeric input and numeric output widgets that allow to, respectively, set and visualize the value of a temperature variable that is defined in the Logical View, and this is simply achievable by binding the widgets to the process variables of interest from the properties of said widgets. In B&R HMI applications, variables are shared with the HMI using the OPC-UA standard between the logic of the machine and the *mapp View* server, and this allows also the binding with variables that are defined in PLCs different from the one that contains the *mapp View* server, increasing

again the flexibility provided by *mapp View*. It is allowed to define two different kinds of binding: a value binding, in which the HMI will receive only the value of the variable, or a node binding, that is a lot more useful in case the variable represents some physical quantity with a unit measure. In said case the node binding will also perform the conversion of the variable's value in the HMI if the unit measure changes.

Another important concept used in *mapp View* is the one of event bindings, that allows to link an event to an action like, in the previous example, the warning sign widget that is made visible only if the current temperature is above 45°, in order to signal to the operator something wrong in the machine. Other actions that can be done are, for example, the set of a variable value to zero, the change of theme of the HMI or the style of a widget at the click of a toggle button (event). Event-bindings are what actually make the HMI dynamical and interactive and they are managed together with bindings from two files provided by *mapp View* in the Configuration View in order, again, to simplify the life of the developer that will interact with intuitive and graphical interfaces: a *.vis* file and a *.mappviewcfg* file. The former is a file that contains the reference to all pages, contents, dialog boxes and themes used in the HMI, while the latter is the configuration file of the *mapp View* server that allows to configure the communication protocol used, the port number and the maximum number of clients that can access simultaneously to the HMI. Besides event bindings, *mapp View* presents other two advantages with respect to the previous visualization tool deployed by B&R: the possibility of the automatic resize of the HMI with the browser window and the possibility to have multiple clients connected to the system at the same time, wich make this technology future poof in the auotomation world.

2.4.2 *mapp Services*

mapp Services is a collection of modular software components that handle the basic functions of machinery and equipment. *mapp* components are created with the possibility to speak within themselves, increasing their functionality proportionally with the number of them that is used. In particular, in this thesis project, the two components used are *mapp AlarmX* and *mapp Recipe*. How these components have been used will be highlighted in Chapter 5, here a brief description of their general functioning is reported.

mapp AlarmX

Alarms are crucial during the functioning of an automatic machine in order to signal in real time warnings and expecially errors occurred, to the opera-

tor. *mapp AlarmX* is a modular, full-featured alarm system. The controller registers alarms with millisecond accuracy and can forward them on to other systems. Alarms can be linked with actions and properties that can trigger actions such as opening a PDF file, playing an instructional video or displaying a virtual model of the machine with the location of the fault highlighted. These options all accelerate troubleshooting and boost the productivity. Alarms in general represent the diagnostic of the machine and grants the possibility to have different reactions to the problem found (for example the machine can be immediately stopped if a fatal error occurs, stopped in phase if some materials required for the processing need to be recharged or a simple warning can be visualized without stopping the machine).

mapp AlarmX provides a configuration file *.mpalarmxcore* where all alarms can be configured by simply defining the name of the alarm, the message that must be visualized when the alarm is triggered, the severity (priority) and the behaviour of the alarm. The possible predefined behaviours of an alarm are six, two that require the use of the function block *MpAlarmXCore* and are more suitable for time-critical alarms and four that can be entirely configured from this *mapp AlarmX* file.

Edge alarms are alarms that when triggered deliver the alarm message and then are deactivated, they can be triggered using the function *MpAlarmXSet* and acknowledged using *MpAlarmXAcknowledge* or the specific *mapp View* widget in order to remove it from a list of alarms (a structure that can be created using the function block *MpAlarmXListUI*), this is usually done when the operator has completed the procedure that deals with the alarm.

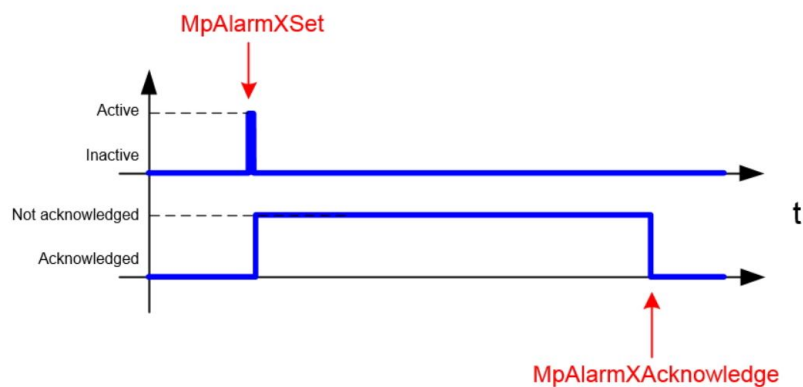


Figure 2.8: Edge alarm behaviour.

Persistent alarms remain activated until they are manually reset by calling the function *MpAlarmXReset*, usually after a certain condition has been met (for example if the temperature value that triggered the alarm is back to a

tollerable value). The acknowledgment of the alarm is independent from the fact that the alarm is still active or not and can be done as for edge alarms.

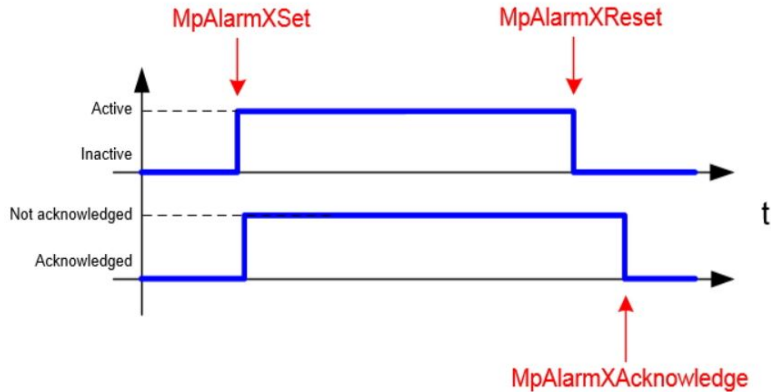


Figure 2.9: Persistent alarm behaviour.

The last four behaviours for alarms do not require any code and are: level monitoring alarms that monitor a process variable and are activated if the variable's value exits from a predefined range, deviation monitoring alarms that monitor the standard deviation of a process variable and rate of change alarms and discrete value monitoring alarms that are triggered by other limits that can be defined for the value of a process variable. *mapp AlarmX* is perfectly integrated with *mapp View* in order to be able to visualize, directly on the HMI, the alarms in the system and this is done through the apposite widget that must be binded with the alarm files provided by *mapp AlarmX*.

mapp Recipe

In the automation world a recipe is a file that contains some machine parameters that, if modified, influence the shape or frequency of the products generated by the machine, in the B&R framework, these files can be in *.xml* or *csv* format. Recipes are intrinsically linked to the flexibility of a machine cause they allow to modify these parameters at runtime, store them and then change the production output of the machine on the fly without stopping it, a recipe allows to create different products by simply loading in the machine different parameter files. *mapp Recipe* allows a smart and intuitive management of recipes through the use of one single file *.mprecipe.xml* that is defined in the Configuration View and defines the core that will manage the *.xml* file that contains the recipe. This file will be generated in a directory that can be specified in the Physical View in the configuration of the CPU.

The management of recipes is done using mainly two function blocks provided by *mapp Recipe*: *MprecipeRegPar* that links the process variables that will

contain the parameters specified in the *.xml* file with the *.mprecipexml* file that manages recipes, and *MprecipeXML* that, given the links to the core, the file device (directory in which the recipe file is present in the target) and the XML file, is able to save the recipe in said file if this exists, or create it if not. If the file is modified in the HMI or using external tools, the updated values of the recipe can be load in Automation Studio using the *load* input of this function block.

2.5 Motion Control

The constant increase of flexibility and performances required by machines has been satisfied in recent history with the substitution of a lot of mechanical pieces with electric motors, so the motion control is something crucial and composes the main focus of B&R products. A classical motion control scheme is composed by three or four different parts:

- Drive
- Motor
- Gearbox if necessary
- Encoder

The purpose of the drive system is to set some set-points for the motor in position or velocity and to make the motor reach these set points as fast and accurately as possible, with the controller that needs a feedback from the motor to be able to complete this task. This feedback is given by the encoder that is a measuring element that measures position and speed of the motor in real time. [11]

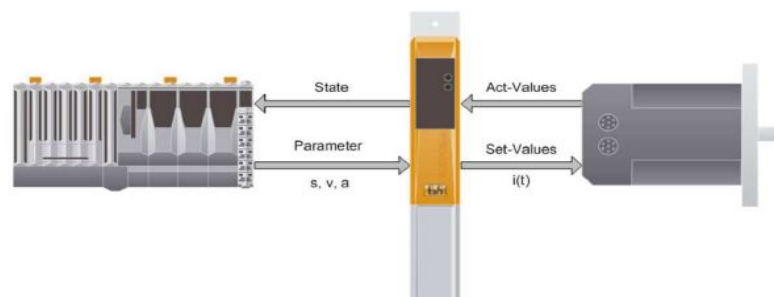


Figure 2.10: PLC-drive-motor scheme.

In Figure 2.10, the decentralized control approach chosen by B&R can be seen. A centralized approach would be with one single controller that controls

different drives and not only generates the set-points but also closes the control loops, with the computational load of the PLC that becomes very high also for the control of a few axis. The decentralized control, on the other hand, has the CPU that only defines master-slave relationships and executes diagnostics during the motion and then the drive is in charge of defining the set points, computing the motion profile and closing the control loops, allowing the use of a small sized PLC also for the control of a large number of axis without losing performances, thanks to B&R smart drives.

The motion controller is a cascade closed loop controller with a structure that can be seen in Figure 2.11:

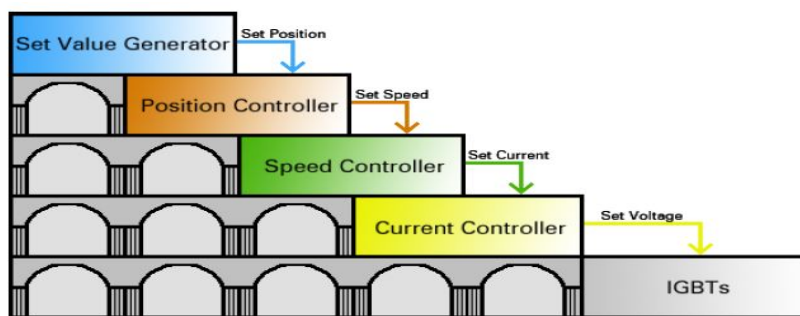


Figure 2.11: Cascade control scheme.

The drive contains four different control loops: the position control loop receives the set-point in position generated by the drive and confronts it with the actual position of the motor provided by the encoder, generating the reference speed that will be passed to the speed controller. The speed and current controllers behave in a similar way and this control structure converts the manipulated variable at higher level in the reference variable for the next control loop, since the voltages for the IGBTs are generated in order to drive the motor.

Each one of these controllers is a PI controller and for the position and speed controller both the Proportional and Integral gains can be set manually in Automation Studio or automatically tuned by the system. The current controller's gains, on the other hand, are always automatically calculated using the motor parameters. The overall control scheme can be seen in Figure 2.12 and it presents also a feed-forward controller that can improve the current set-point generated by the speed controller if the load torque, the speed dependent torque losses, the negative and positive Coulomb torques and the motor inertia are known. B&R provides also a specific autotune that is able to make the drive estimate automatically these parameters.

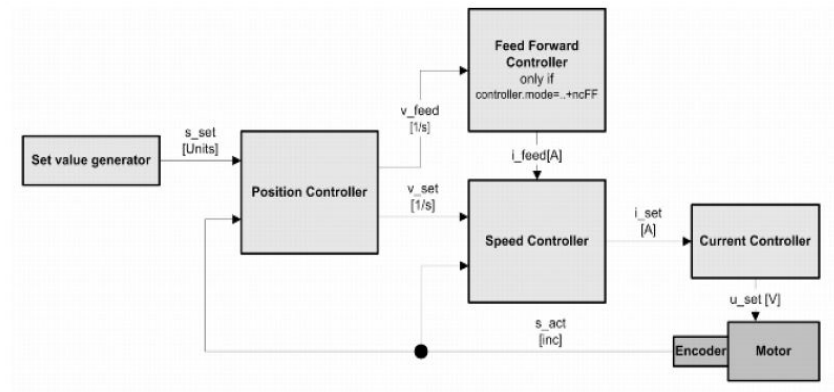


Figure 2.12: Motion control loop scheme.

The B&R drives belong to different ACOPOS families: ACOPOSmicro, ACOPOS P3, ACOPOSmulti and ACOPOSmotor. The ACOPOSmotor is a compact system that has the drive directly mounted on the motor in order to reduce the space occupied by the cabinet in the machine, while ACOPOSmicro is a small drive that can be used again to save space in the plant for applications that do not require high performances, while existing in the version adapt to drive stepper motors. ACOPOSmulti is the classic B&R system of drives that offer high performances (cycle time up to $400 \mu s$ for the position control loop, $200 \mu s$ for the speed controller and $100\text{-}50 \mu s$ for the current controller) and it is a drive that can be combined with other ACOPOSmulti drives in order to create a rack of drives, ideal for applications with a large number of axes. All these drives are connected to the same power supply that can be active, allowing to use the braking energy developed by one drive to feed another drive, or passive and this require a braking resistor in order to dissipate the energy developed.



Figure 2.13: ACOPOSmulti and ACOPOS P3.

ACOPOS P3 is B&R's cutting edge in this sector and grants better performances with respect to ACOPOSmulti, granting a cycle time of $50 \mu s$ for each one of the three control loops and it is able to control one, two or three axes, depending on the chosen unit.[11]



Figure 2.14: ACOPOSmicro and ACOPOSmotor.

As previously said, B&R makes of motion control its main focus, so, in order to complete the portfolio of its products, it provides also inverters and gears, by selling also the ACOPOSInverters that are very adapt to drive synchronous and three-phase induction motors with a compact solution that reduces the machine size as well as its cost.



Figure 2.15: ACOPOSInverter.

ACOPOSTrak and SuperTrak overview

This chapter is a brief overview on the first two technologies developed by B&R in order to exploit the magnetic force as a way to move products around the production plant. These technologies are not simply something that can replace the classical conveyors but allow to completely rethink the automatic machine concept by reducing encumbrances and the cost of the format change of the product, thanks to the independent shuttles. In particular these two technologies use magnetic attraction instead of magnetic levitation and so they differ a bit from the ACOPOS6D technology on which the thesis project is based. ACOPOS6D is based on magnetic levitation so there is no mechanical friction and it is able to move on a plane while ACOPOSTrak and SuperTrak move on rails in specified directions with friction with the guides. ACOPOSTrak is a B&R original technology while SuperTrak is the result of a collaboration between B&R and the canadian company ATS Automation [7].

3.1 ACOPOSTrak

ACOPOSTrak is a technology born to face the new trends in the automation market and in order to satisfy the newest machine requirements:

- Easy and fast change of format
- Adaptability to the future
- High product density per square meter (high productivity in small encumbrances)

- Efficiency on small batches of products

3.1.1 Technology overview

ACOPOSTrak is made by independent shuttles with motion laws totally autonomous that can be set from Automation Studio. The shuttles are made of permanent magnets with alternate poles that are attracted by a stator made by a series of windings placed one next to the other forming a Long Stator Linear Motor that potentially extends endlessly. The windings in the stators attract the shuttles and make them also translate in the motion direction (parallel to the stator) by generating a magnetic field that can be controlled independently for each set of windings, this being the key to the independence of the shuttles one with respect to the other.

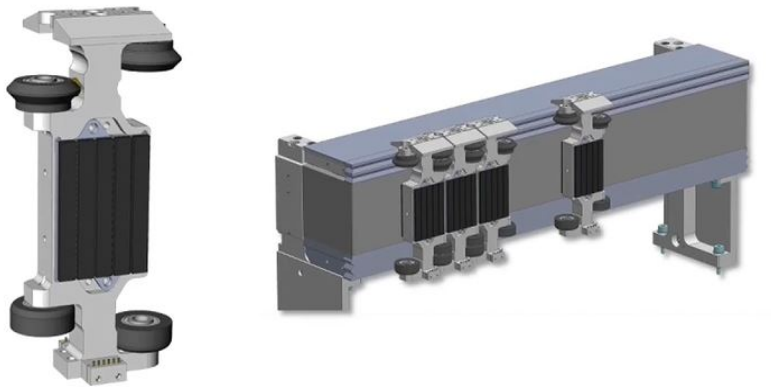


Figure 3.1: Shuttles and stator in ACOPOSTrak.

ACOPOSTrak's shuttles are able to withstand payloads of around 2 kg with a positioning repeatability of $150 \mu m$ and is targetized for the packaging industry in particular. The shuttles are anchored to the segments of the stator by two different guides, one in the upper part where two different planes of wheels move in two parallel channels in order to guarantee the correct resistance in the vertical direction during motion, while in the lower part of the shuttles, they simply move because of the rolling without slipping of the wheels on the planar surface of the segments.

The segments (stators) come in different fashions: straight segment, arc segment with 45° of curvature and round segment (both in left and right directions) with 22.5° of curvature and this allows the creation of very different and complex layouts in order to satisfy any customer's needs. In particular, these segments present other two advantages: every segment, no matter the

curvature, fits inside a standard dimension grid so it is easy for the costumer to understand what the footprint of the machine will be before starting the project, and these segments, as well as ACOPOS6D ones present the possibility to be mounted vertically in order to create even the most particular layouts.



Figure 3.2: Stator segments in ACOPOSTrak.

3.1.2 High speed Diverters

The main feature of ACOPOSTrak, that guarantees the maximum possible flexibility for the creation of the layout, is the presence of high speed Diverters that gives the possibility to separate or combine the flow of the shuttles in any point of the trak. This solution is totally electronic and is implemented without the use of any mechanical device, making the maintenance a topic of not primary importance.

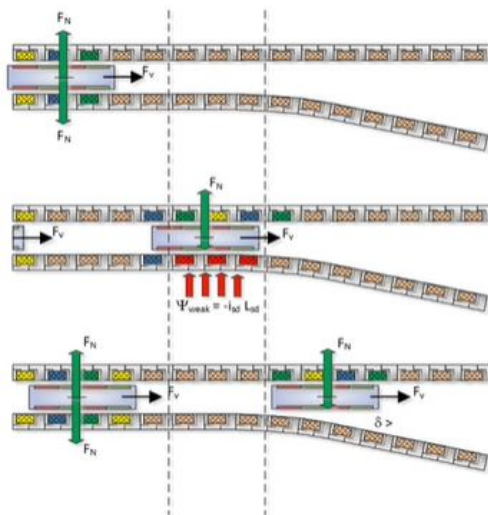


Figure 3.3: Diverters in ACOPOSTrak.

As visible in Figure 3.3 the deviation of the shuttles from one guide to the other is obtained by reducing the magnetic field on one side, making the shuttle more incline to be trapped from the magnetic field of the other segment and guaranteeing the change of segment. The possibily of dividing the product

flow can be very useful for the quality control, that can easily be set up using a B&R camera, while the possibility of converging two or more product flows in a single one can be really useful in order to create specific batches of different products and changing the format of these batches on the fly.

Since the shuttles are kept in place only by the stator's magnetic field, they can be swapped in and out of the trak without having to stop and restart the machine, with the anti-collision functionality that will prevent any impact between shuttles and the system that will be able to recognize on the fly new shuttles added to the layout.

The configuration of the layout is made directly in Automation Studio by defining the segments with their start and end points and defining some relevant points (process points) on the segments. The code is then usually executed by defining some working stations that will acknowledge the shuttles, perform some operations on them, start their movement towards the next station and then acknowledge the next shuttle. It is not necessary to know a priori the next station in which the shuttles will be sent because of the process points: these points will recognize the shuttles when they acknowledge them and allow the programmer to change on the fly the destination of the shuttles, allowing unprecedented flexibility to the application.

Automation Studio will then generate some files that can be opened with SceneViewer, a 3D simulation of the kinematics of the application. This reduces the time to market being able to develop a machine prototype without having the full hardware setup. In Figure 3.4 an example application can be seen, that also includes two other machines and a robot that work together with the Trak.

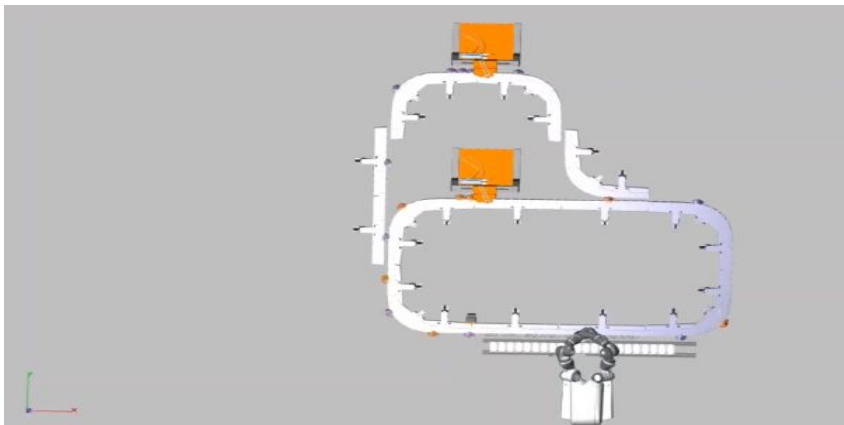


Figure 3.4: ACOPOSTrak application simulated in SceneViewer.

3.1.3 Washdown and cooled designs

ACOPOSTrak is available with IP69K protections in the washdown design, in order to give the customers the possibility to apply high pressure steam jets with temperatures up to 80° and industrial cleansing liquids without any damage to shuttles and segments. This is possible because of the stainless steel cover all segments are provided with and because of the sealed holes and the welded connectors that avoid possible corrosion of the internal parts of the segments. This is especially useful for pharmaceutical and food and beverage applications.

One main issue that can be encountered using ACOPOSTrak is that, especially in some applications where there are a lot of shuttles that accelerates and decelerates together in specific points of the layout, a lot of heat can be generated in said regions. In order to solve this problem B&R produces motor segments with embedded built-in liquid cooling system, without any additional installation, that can be combined modularly with segments without this cooling system, allowing the customer to minimize the cooling costs by limiting cooling to the segments that actually need it. The cooling system pumps cooling water through a circuit that absorbs heat generated by components and releases it into the surrounding air via a heat exchanger.

The *mapp Trak* technology, that is specifically designed to manage and set up the Trak application, allows the customer to calculate exactly where the track system's power requirements are the highest and determine how much heat will be generated in each track segment, showing in simulation which segments require to be cooled.

3.2 SuperTrak

SuperTrak is a technology characterized by some differences with respect to ACOPOSTrak: the payload that each shuttle can sustain (10 kg with respect to the 2 of ACOPOSTrak) and a different position repeatability ($10\mu m$). These two technologies have also different target sectors, with the machines realized using SuperTrak that are suitable for transport automation and assembly lines, where a lower topological flexibility is required.

The segments are linear or curved with a 180° angle with motors and power electronics placed in the lower part for the straight segment and in the center of curvature for the curved one, SuperTrak does not have the Diverters and this means that the typical topology of this technology is an oval shape. In order to remove a shuttle from the layout, in this case, a mechanical tool is

needed in order to tilt the shuttle and defeat the magnetic force that keeps the shuttle anchored to the segment, still providing a fast and easy way to complete the operation.

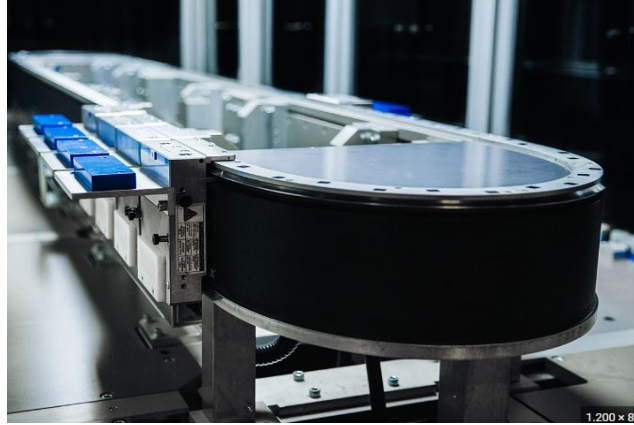


Figure 3.5: SuperTrak example.

3.3 Trak technologies overview

One peculiar feature of the Trak technology (both for ACOPOSTrak and SuperTrak) is the anti-sloshing, provided by a specific B&R library that tunes the motion control of the shuttles on which liquid is carried in such a way that the surface of the liquid is kept as horizontal as possible even during high acceleration and deceleration applications in order to avoid overflowing of the liquid during said high-dinamic applications.

The main drawback of these two innovative technologies is the presence of mechanical friction. The problem is given by the fact that both ACOPOSTrak and SuperTrak are based on magnetic attraction that keeps the shuttles in contact with the segments and this, even if the use of wheels allows a reduction of the friction during motion, brings the shuttles to consumption and to be worn out with time during functioning, making it necessary to substitute the shuttles once in a while. In order to overcome these limitation and to answer some costumer's needs to not only manage and move products in one direction (provided by the segments' guides) but on a plane, B&R developed a new product, that is the subject of this thesis and will be presented in the next chapter: ACOPOS6D. Some aspects that are enhanced by ACOPOS6D are a vaster and vaster number of flexible solutions that were not possible with these two technology, as well as an higher positioning precision. ACOPOS6D is also able to reduce the already small footprint that machine have in production

sites with ACOPOSTrak and SuperTrak [7].

ACOPOS6D

ACOPOS6D is the newest technology applied by B&R to the automation world in order to revolutionize the concepts of flexibility and fast adaptation to different kinds of products that a machine has to produce. It is based on magnetic levitation and it is one of the first tested and available for production systems that exploit this technology in industrial automation. As highlighted in Chapter 3, ACOPOSTrak and SuperTrak have already differentiated a lot with respect to classical conveyor belts by allowing the customer to change, without any mechanical modification to the machine (only a different logic is needed), the number of products in a batch but also the order in which these products are put in said batch. This concept of enhanced flexibility has been brought to a whole new level by ACOPOS6D that allows these changes in shape and order of batches of products to be done on a plane (both x and y directions) and not on a straight line, as it was for the previous technologies.

4.1 Hardware

The hardware of which an ACOPOS6D application comprehends a PLC or Automation PC that allows to fully integrate it in the B&R ecosystem of products, a 6D controller that is connected to segments able to generate a suitable magnetic field that controls the motion and levitation of some shuttles. The controller is connected in POWERLINK to the B&R controller while the segments communicate with the controller through HDMI cables for data exchange.

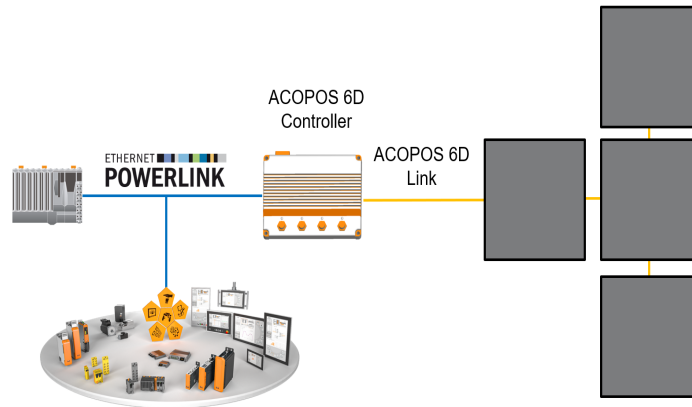


Figure 4.1: ACOPOS6D topology.

The hardware used to develop this thesis project is showed in the next Figure, its components will be then analyzed in this chapter:

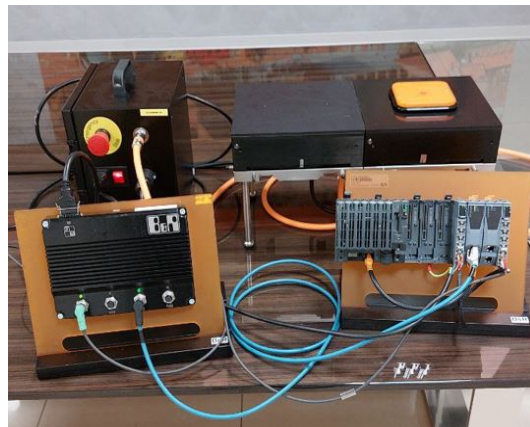


Figure 4.2: ACOPOS6D example layout with two segments and one shuttle.

4.1.1 PLC X20CP3685

The PLC used in this thesis project belongs to the X20 family, and in particular it is the unit X20CP3685. The X20 PLCs are certified for a protection IP20 that grants an assurance of protection against solid objects with more than 1.2 *mm* in diameter and are fanless with working temperature that go from -25° to 60° and this makes these family of PLCs perfect for automation applications. This design has been thought and realized in order to place the PLC in a cabinet. This PLC presents two USB, one Ethernet and one POWERLINK ports and one slot for a CompactFlash storage device, the storage devices used by B&R for its PLCs, as well as three slots for I/O modules. A CompactFlash can reach up to 256 GB of storage space and usually is partitioned in a SAFE

way: a C-SYSTEM partition, two DATA partitions that contain the same data (user ROM for programs and libraries and system ROM for the configuration objects) in order to avoid the loss of data due to the corruption of one of the two partitions and one F-USER partition for recipes and configuration files (see Chapter 5). When the system is started all data contained in the ROMs is copied to the DRAM memory for higher performances and when the system is shut down, retain variables are copied to the SRAM memory for the next power up of the system.



Figure 4.3: PLC X20CP3685.

The X20CP3685 module is able to manage cycle times up to $200 \mu s$ and in particular mounts an $0.8 GHz$ (middle sized) Atom processor and is provided with $512 MB$ of DRAM and $1 MB$ of SRAM. It is not the most powerful CPU in the X20 family but can still be used in a computational demanding application with ACOPOS6D because the computational burden for this particular technology is split between other two components: the Planar Motor controller and the segments [12].

4.1.2 Planar Motor controller

The Planar Motor controller is a POWERLINK node and this means that it can be fully integrated with any other B&R hardware component without any particular procedure to be actuated. It is connected to the PLC in POWERLINK and to one of the segments (master) via HDMI cable for fast exchange of data. The purpose of this controller is to manage the integrated anti-collision algorithm that stops one shuttle with a controlled deceleration when the algorithm sees an obstacle (another shuttle) on its path and is in charge of the computation of the trajectories that the shuttles have to travel on the layout. Since the computational burden is mainly on the Planar Motor controller and on the segments this allows also to use small size PLCs to operate the system even in very complex and performance demanding applications, as said before.



Figure 4.4: Planar Motor controller.

4.1.3 Segments

The segments are what forms the base of an ACOPOS6D application, also called layout, and have a central role because they generate the magnetic field that brings the shuttles to levitate and move. They also grant some computational power, decentralized with respect to the controller in order to ease its computational burden. In particular, the control loops are closed in the segments and not in the controller. Segments come in standard dimensions of $240 \times 240 \times 70$ mm (70 mm being the height) and can be combined in order to create the most different and complex layouts in order to satisfy every customer's need.



Figure 4.5: ACOPOS6D segments front and rear view.

Each segment is provided with four HDMI ports that allow the connection for the exchange of data with the controller and other segments, the ports for the low voltage (48-58 V) power supply and holes that allow to connect each segment with a liquid cooling system in order to overcome the high temperature generated on segments by high speed and acceleration's movements of shuttles with possibility to cool only selected segments as it was for ACOPOSTrak. The connection between segments is realized in daisy chain in order to allow a very high topological flexibility without having to connect each segment to the controller and allowing to disconnect easily some segments from the layout without having to change the other segments' cabling, as well as the possibility to enlarge the layout significantly in a very comfortable and easy way.

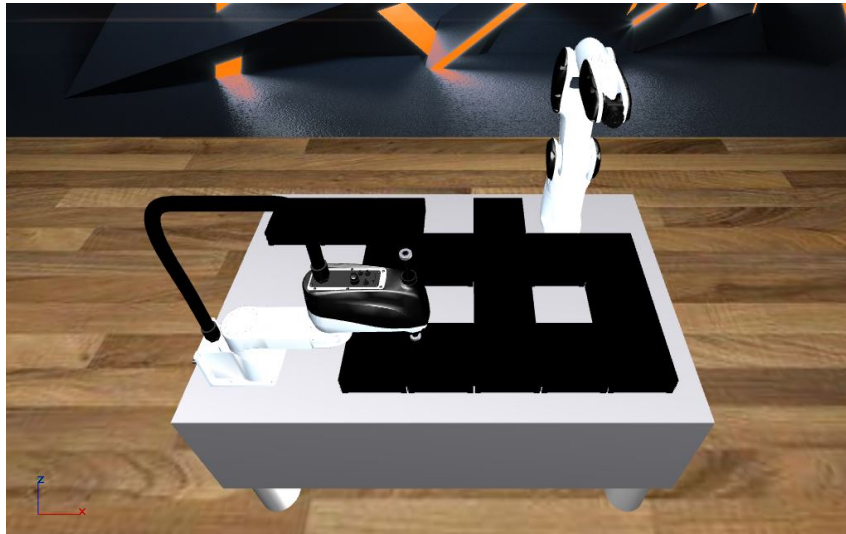


Figure 4.6: ACOPOS6D example layout in Scene Viewer.

Each segment can be connected to other segments on each of the four sides, allowing a very high topological flexibility in order to fit any workspace and integrate with any kind of robots and other machine groups that have to perform some manipulations on the products brought by the shuttles.

4.1.4 Shuttles and motion profiles

The shuttles are ACOPOS6D's passive elements, in particular they are squared permanent magnets without any kind of computational power whom's motion is controlled by the magnetic field generated by the segments. Shuttles come in different dimensions in order to satisfy every possible need of the customer and being able to transport products of very different width and weight. The standard shuttles that have been used in this thesis project are 120x120x10 *mm* shuttles as the one showed in Figure 4.7, the smallest possible shuttles:



Figure 4.7: ACOPOS6D shuttle.

Shuttles measure up to 450x450x16 *mm*, with forces and torques sustainable that increase with its dimensions as well as the power consumption during operation.

Motion profiles

Shuttles can move with a speed up to 2 *m/s* and with 20 *m/s²* acceleration on the layout, levitate up to 4 *mm* in height and rotate freely in the layout with very high precision in every motion. The motion profiles realizable with ACOPOS6D are vast and this allows an extreme flexibility of the machine. Shuttles can translate on the layout but also rotate freely for a 360° angle around their center in specific layout points (edges and centers of segments) or rotate of about 10 degrees in both directions in any layout point. As the name of the system suggests, they are also able to tilt in the *Rx* and *Ry* directions, achieving a complete six dimensional motion. It can be created a shuttle group that makes each shuttle of the group replicate the motion of any of them, creating a fleet of shuttles that move together. An important function that is present in the system is the possibility to create a master-slave relation between two or more shuttles where the master can then move freely with slaves that replicate exactly its motion (sun-planet behaviour). Another perk granted by the system is the possibility to seamlessly synchronize the motion of shuttles with other machines and robots present in the line: the position of a motion axis can be cyclically given to the shuttles as a set-point in order to create an actual cam profile or, for example, a fictional CNC can be simulated in the PLC and then a shuttle can move exactly as the end-effector of this CNC. This allows to perform the same operations that would have required a robot simply with a static tool and a shuttle.

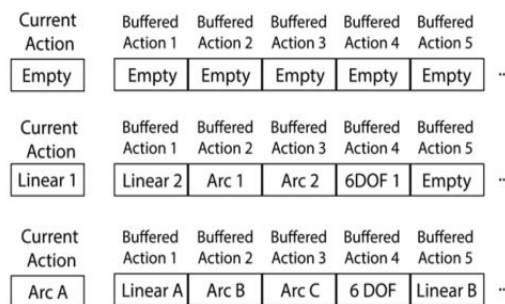


Figure 4.8: ACOPOS6D motion buffers.

Another interesting concept proposed by ACOPOS6D is the possibility to bufferize commands: if a command given to a shuttle is acknowledged by the cor-

responding function block, another command can be sent to the same shuttle even if the previous movement has not been executed yet, so the motion buffer of that shuttle will store all the received commands and execute them in succession. This concept allows also the possibility to use motion macros in order to bufferize commands and then execute them at will, macros can also be referred to other macros in order to obtain an endless loop of motions for a shuttle.

Other features

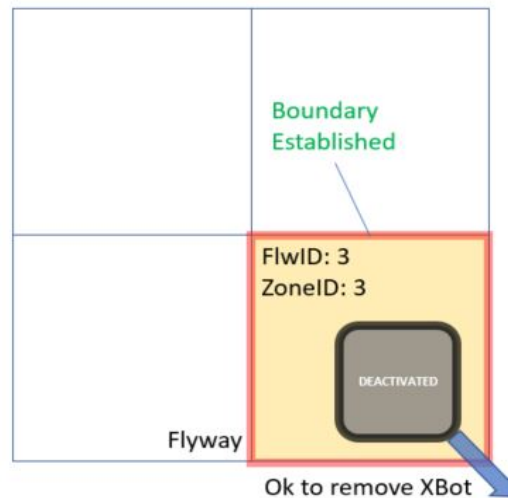


Figure 4.9: Zone and fence for safe shuttle removal.

Discharge zones can be defined in the layout in order to safely remove shuttles from the machine without stopping the production: once the zone is activated, all shuttles that are placed on it will be disabled and a fence is created to deny to other shuttles the possibility to enter the zone. Another interesting possibility offered by ACOPOS6D is the weighing function, with which shuttles are able, on the fly, to measure the weight of objects placed on them and, in general, the forces and torques applied to the shuttle in each of the six directions.

4.2 Application fields and advantages

ACOPOS6D is perfect for applications in a vast variety of automation fields and is a strong candidate to become the go to transportation system in any automation application. In particular, specific sectors in which ACOPOS6D could be extremely useful are the food and beverage sector and the pharmaceutical one. This is due to the IP67 protection of which segments are provided:

an IP67 system is completely covered against solid bodies penetration (even dust and sand) but is also able to be plunged in water up to 1 *m* for 30 minutes without presenting any internal contamination. It is clear that this characteristic makes the system adapt to food and beverage applications in which pieces of products or even liquids could end up on the segments without damage the magnetic field generation and the control of shuttles.

For what concerns the pharmaceutical field but also the previous one, an important perk provided by B&R is the possibility to have hygienic shuttles completely covered in stainless steel, in order to avoid the possibility of product contamination and granting the possibility to wash the system repeatedly as it is requested in applications belonging to this field.



Figure 4.10: Pharmaceutical and food and beverage sectors in which ACOPOS6D can be applied.

Due to the extreme flexibility in the composition of products' batches, the high speed and accelerations that the system is able to reach ACOPOS6D is in general perfect for any kind of packaging application and it also grants the possibility to reduce a lot the encumbrance in production plants occupied by the transportation system: a plant that, with traditional automation systems, would have been as big as an entire room can be reduced to the dimensions of a kitchen table by using ACOPOS6D. In a real application, the encumbrance of a machine constructed with traditional automation components can be reduced by 1/3 with ACOPOSTrak while this encumbrance can also reach 1/10 of the original machine with the use of ACOPOS6D. It is not necessary to use a different conveyor for every kind of product that the system can produce, but it is sufficient to modify the trajectory covered by shuttles (coding procedure without having to substitute mechanical pieces) on the layout based on the kind of product that they are moving around.

4.3 Software: mapp 6D

mapp 6D is the mapp Technology package developed by B&R in order to control and configure the whole ACOPOS6D system. mapp Technology changes the way applications are built in the automation industry; it is based on modern modular software principles and offers an interconnected framework of software components that satisfy the most common use cases for the industry. In the Configuration View two files are provided for the setup of the system: a *.layout6d* file and a *.assembly6d* file. The former is a file that grants, through the use of a very intuitive graphical interface (Figure 4.11), the possibility to recreate the layout by selecting the number of segments (active in green and inactive in grey, in order to create also layouts with holes inbetween segments) and the master segment.

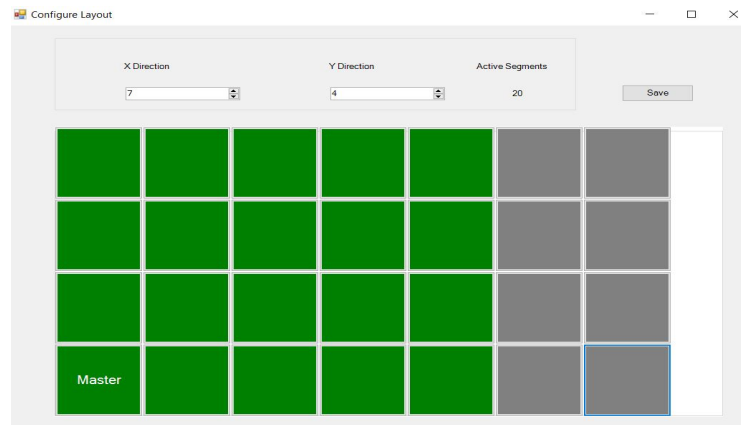


Figure 4.11: mapp 6D layout creation page.

The latter is another file that allows to easily complete the configuration of the system by selecting the maximum number of shuttles that will move on the layout, if the application is simulated or not, the default position in Z , R_x , R_y and R_z directions for each shuttle when they are activated, the velocity and acceleration limits that shuttles must respect in the application and the POWERLINK node number of the controller. In this file a reference to a *.layout6d* file must be defined because in the system more than one of these files can be defined but only one at a time can be used by the *.assembly6d* file. Another interesting option that can be configured here is the kind of ID mode desired for shuttles identification: Real-Time ID makes the system assign one random ID to each shuttle at every power up, while Absolute ID allows to use the same ID for the same shuttle even in case of shut down of the system (this makes the activation procedure last longer cause the system has to move shuttles around the layout in order to recognize each one of them).[7]

mapp 6D comes with the *McAcp6D* library provided by B&R that is composed of a series of function blocks that can be used by the customer to program an ACOPOS6D application, the function blocks used in this thesis project will be highlighted in Chapter 6.

B&R Template and its advantages

B&R Template is a project developed by Bologna's B&R office that has been realized in order to face the increasing requests coming from customers to help them develop an automatic machine starting from scratch. The Template project aims to provide to all these customers a robust, modular and reliable framework with a machine structure already integrated in it and a lot of useful functionalities at the disposal of the client. The use of the Template allows to exploit as much as possible all the mapp Technology packages that compose the B&R portfolio because this project has been explicitly developed in order to maximize the exploitation of these products by B&R's engineers with years and years of experience in the automation field and by the developers of the same mapp packages. The B&R Template is also fully available in the PackML (Packaging Machine Language) version, a standard for the control of packaging automatic machines that grants common states and operational flows to the machines that compose a packaging line.

5.1 Modular structure

Every automatic machine is composed of distinct mechatronic units, also called devices, that have to work together and cooperate in order to assure the correct functioning of the machine. The Template project uses an object oriented approach for the programming of the machine that has as its first step the definition of these units. The coordination between the different devices is done by the Main Logic, that will interface with each singular device with a

Command/Status structure in which the Main Logic will simply command to each device the state that it has to reach and then wait for a feedback from the device. The Main Logic is completely unaware of what the device will do and its code is completely independent from the device's one, making it easy to add and remove devices from the machine without having to change the code of the Main Logic. The default states (user-defined states can be easily added during coding) in which it is possible to bring each device are:

- INIT
- WAIT
- MANUAL
- PHASING
- PRODUCTION
- EMERGENCY

The Main Logic first of all calls a Startup action that brings every device in the INIT state and usually checks the correct loading of the recipes and configuration files and if the device is present and ready and then starts its nominal working phase. It is composed of three different tasks that work concurrently: *L_Main* for the management of the devices, *A_Main* for alarms management (Section 5.2) and *F_Main* for the management of recipes and configuration files (Section 5.3). It is important to note that each device will also have its *A_Device* and *F_Device* tasks in order to make it completely independent from the Main Logic code.

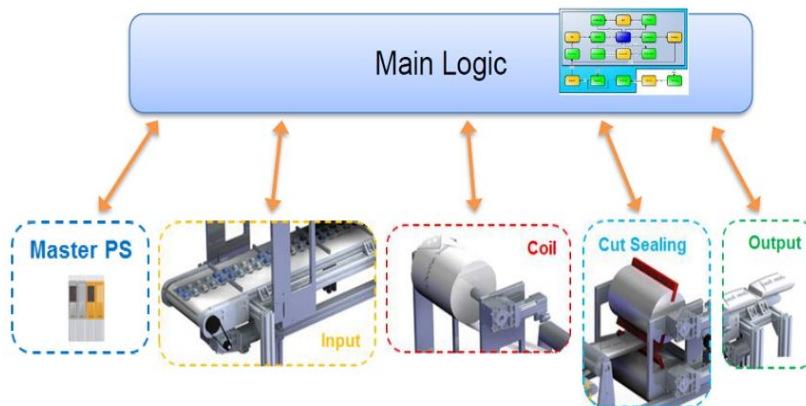


Figure 5.1: Template's modular structure.

The *L_Main* task now can send different commands to each device: in the

PHASING state each device usually executes the initial operations that bring it in the correct initial position in order to start the nominal operations, the WAIT state is where the device, after the initialization, waits for commands from the Main Logic, the PRODUCTION state usually is the state in which the machine works automatically and represent its nominal working state, it also contains usually the machine jog and the device jog; the device jog is a unique feature of the Template, allowed by the presence of a virtual master in each device that refers to the device Master. This specific kind of jog allows to move at a controlled velocity one single device with the execution of all cams and gears contained in it and is a very powerful diagnostic tool. Another advantage granted by the presence of this virtual master is the possibility to create a different cam for each device with respect to the Master device, allowing incredible flexibility to the machine. A particular state in which a device can be brought is the MANUAL state, that allows to perform a jog of single axis in the machine or to perform manual operations like opening valves or activate digital outputs.

The structure of a device differs slightly with respect to the structure of the Main Logic: in this case what controls the behaviour of the device and the eventual motion part is the *M_Device* task that is composed as a state machine that changes states between the ones described before, while the *L_Device* is a task that manages I/Os and the shift register that can be used to simulate and track the products on the line that will be exchanged between the devices. It is usually used to monitor the production quality (number of discarded products) and in general for datalogging.

5.2 Recipes and configuration files handling

In this dissertation the concept of recipe has already been discussed in Chapter 5.3 but it is necessary to define the difference between a recipe and a configuration file: both come as *.xml* files that contain name, type and value of variables that are used in the machine code but the configuration file contains variables that must be loaded at the startup of the machine and can be changed only when the machine is shut down cause they refer for example to some mechanical parameters of the machine (for example the length of the arm of a robot), so they vary from machine to machine but not from working process to working process. The recipe's variables, instead, can be modified at runtime and rely on the flexibility of the machine to, for example, change the speed of operation or the positioning of some tools in order to modify at runtime the products. Both these files are loaded in the user partition of the PLC memory where the

program can find them and load the dedicated variables values.

These files are managed by the *F_Device* tasks that will call two function blocks of the *MpRecipeRegPar* type for the management of recipes and configuration files per device and these are local structures that will be recognized by the main handler, making it possible to remove or add any device without any problem in the managing of recipes. All variables belonging to different devices are collected in a single *.xml* file, so it is necessary to load a single recipe file in the system in order to parametrize all devices. The Template project presents also two additional tasks *ConfigMan* and *RecipeMan* that allows to load in the system a *Factory settings* recipe in case no recipe has been loaded and to keep loaded the last recipe at each reboot of the system. These tasks are also the ones that actuate the HMI commands that can be execute in order to save, load and activate the recipes (Section 5.4.3).

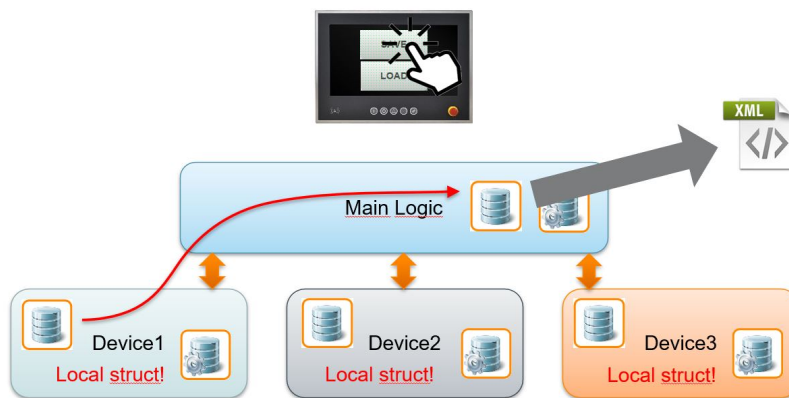


Figure 5.2: Recipes and configuration handling structure.

5.3 Alarms handling

The structure used for the handling of alarms is similar to the one used for recipes: each device has a *mapp AlarmX* core that manages its alarms and these are local structures that are independent from the ones of other devices and the Main Logic. Each device's core is provided with the escalation of alarms that allows all of them to be managed by the Main Logic core that is placed at an higher hierarchy level than the ones of the devices:

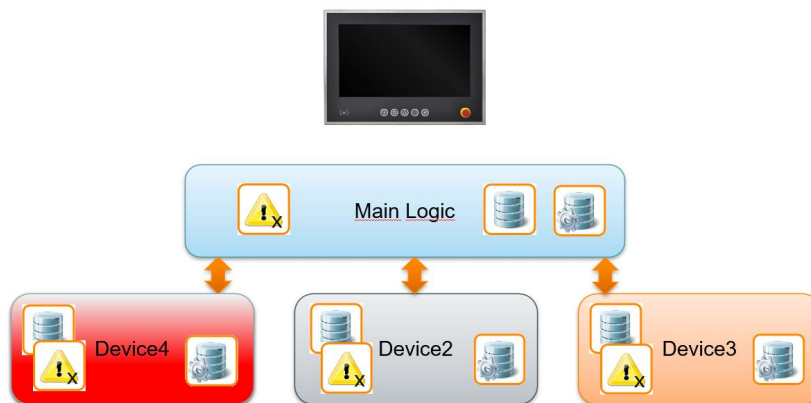


Figure 5.3: Alarms handling structure.

The hierarchy levels are created using the grouping of alarms and making each device core a child of the parent core belonging to the Main Logic in order to be able to escalate all alarms from the devices (this can be simply done by creating a folder hierarchy in the *Configuration View*). This is done in order to keep the modularity a strenght of the Template and allowing to remove a device without having to cancel its alarms one by one from the global list of alarms but simply deleting the file containing its list of alarms with the device from the system. Also the files that contain all device's alarms in all languages are contained in the folder of the single device in order to be added or removed easily from the system.

5.4 HMI and other tools

The advantages that come with the use of the Template project as framework for an automatic machine do not end with the coding structure described before, that was only the start of the useful tools already developed and ready to use in this project.

5.4.1 CreateFilePath tool

The first important tool that comes with the Template project is able to create with a simple click the necessary file devices (for the handling of recipes and configurations) in a specified path and it is also able to detect if a suitable directories already exist or not in the system and to suggest one if necessary:

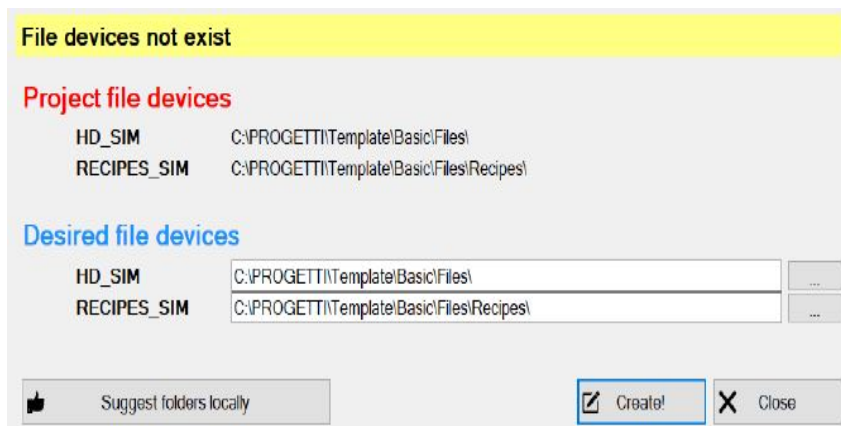


Figure 5.4: CreateFilePath tool.

5.4.2 RenameFull tool

It can happen in a machine that two different devices are of the same identical kind but have to operate in different points of the line. In this case it can be very useful this tool that allows to fully rename a device imported in the project: it is as easy as it gets since it is necessary only to import one device, fully rename each single file of the device to a new device name and then import again the same device without having conflicts in the code or having to manually rename every single variable in the device. Once the path to the project is specified, this tool can search for any file in the project that contains a defined word and then rename this word instantaneously and automatically in each file or only specified ones.

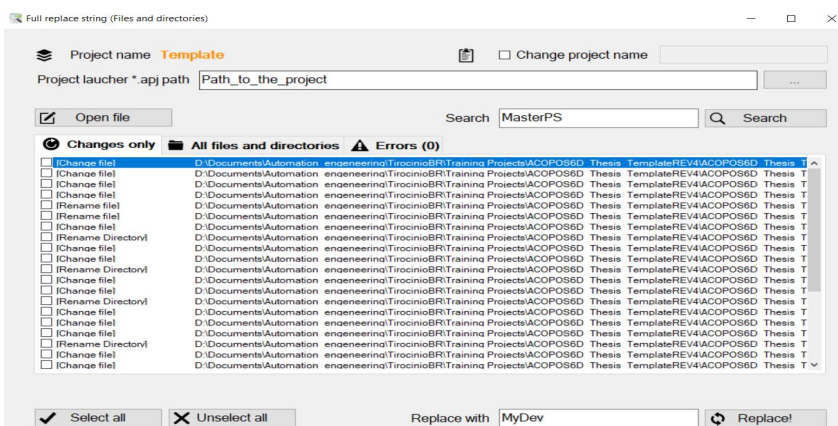


Figure 5.5: RenameFull tool.

5.4.3 HMI VC and mapp View

The Template project is provided to customers with also a built-in HMI available both in *mapp View* and *Virtual Components* (the other visualization tool used by B&R previously to *mapp View*) that is already very useful for debugging the code and understanding what is happening in the machine. In this thesis work, only the *mapp View* HMI will be analyzed cause it is future proof because of the HTML 5 in which it is developed, as well as its possibility to be visualized on every device.

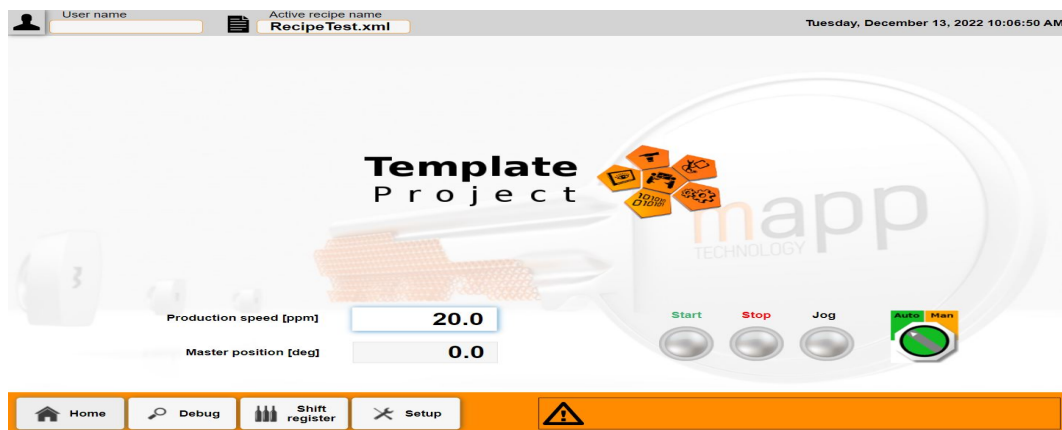


Figure 5.6: Main page of the Template HMI.

The HMI is composed of six pages with a lot of functionalities implemented with shared top and lower content. The top content contains a log-in widget that allows to log as one of the users that are defined using *mapp Services*, in order to be able to restrict the access and visibility to part of the HMI to only selected users, the active recipe used in the machine, a PackML interface if the standard is used and the timestamp. The lower content presents the navigation buttons that allow to navigate between the different pages and a widget that shows the most recent alarm active in the system. In Figure 5.6 the main page is showed: in this page the master position is visible and it is already implemented the possibility to regulate the production speed of the machine in products per minute (this is a global variable that can be used in the logic of each device in order to regulate the production speed to be the desired one). The main page presents also four buttons that can be used in order to operate the machine: by pressing the start button for three seconds the main logic will execute the phasing of each device and then start the production that can be interrupted by pressing the stop button. The jog button can be pressed in the auto mode to perform a machine jog and in the manual mode to perform an axis jog. When the manual selector is activated the operations written in the

MANUAL action are performed.

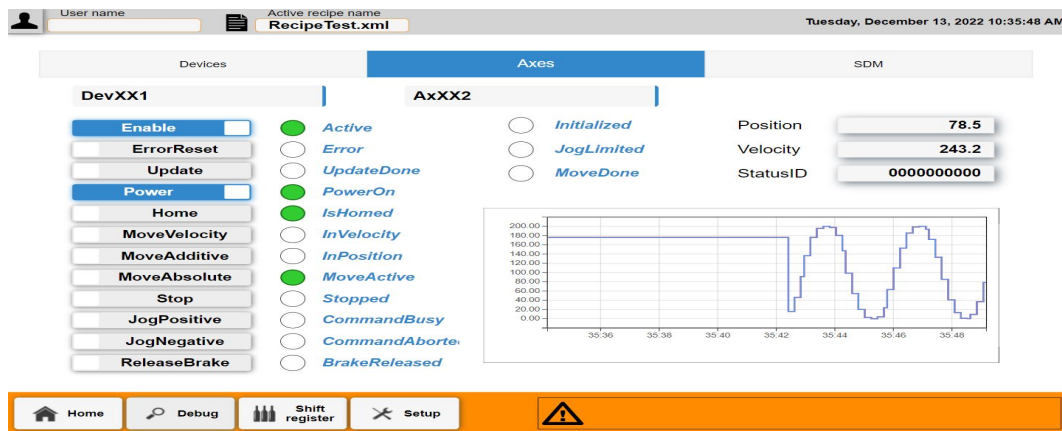


Figure 5.7: Debug page of the Template HMI.

The debug page of the HMI presents three different sections: the Devices section allows the user to see the state of each device in the machine while the SDM section allows to directly consult the System Diagnostic Manager in order to facilitate the debugging of the application. This is a particularly useful tool provided by B&R that allows to perform some debugging of the system even without the need of installing Automation Studio from every browser. The Axes section is showed in Figure 5.7 and shows the state of every axes present in every device while also showing its position and velocity. It is also already implemented a widget that visualizes the position of the currently selected axes as function of time.

In the Shift Register page it is possible to follow the flow of products or also a single product in the machine and to visualize if a product has been rejected and in what point of the machine. The Setup page is provided with the possibility to change in real time the language of the HMI with already implemented the translations in english, italian, german and french and obviously the integrated possibility to introduce other user defined languages from Automation Studio. The main feature of this page is the possibility to save the machine configuration and this is done in order to avoid the need for the customer to write manually the *.xml* file that constitutes the configuration of the machine. The programmer will simply have to define for each device and for the Main Logic the configuration structure types that will contain the variables that need to be loaded in the machine when it is turned up and then, by pressing the button visible in Figure 5.8, the system will automatically create the corresponding file in the path specified in the file devices. This is very useful also when a device is added or removed from the machine because then

it will simply be necessary to save the configuration and the file will adjust automatically to the new machine.

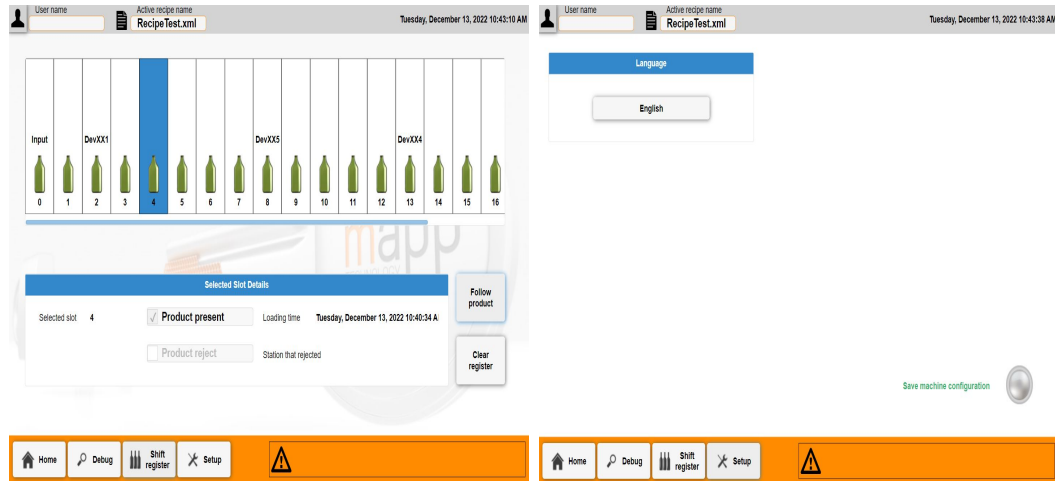


Figure 5.8: Shift register and setup pages.

The recipe handling page allows to load a new recipe in the system between the ones that are present in the folder specified in the file devices by pressing Activate and then Load, while it also allows to directly save the recipe file form the HMI if the recipe structure type is modified in Automation Studio (as for the machine configuration). This page also allows to create a new recipe and rename or delete an existing one. If the Offline Recipe Handling is activated the system allows also to visualize a different recipe with respect to the active one without having to activate it.

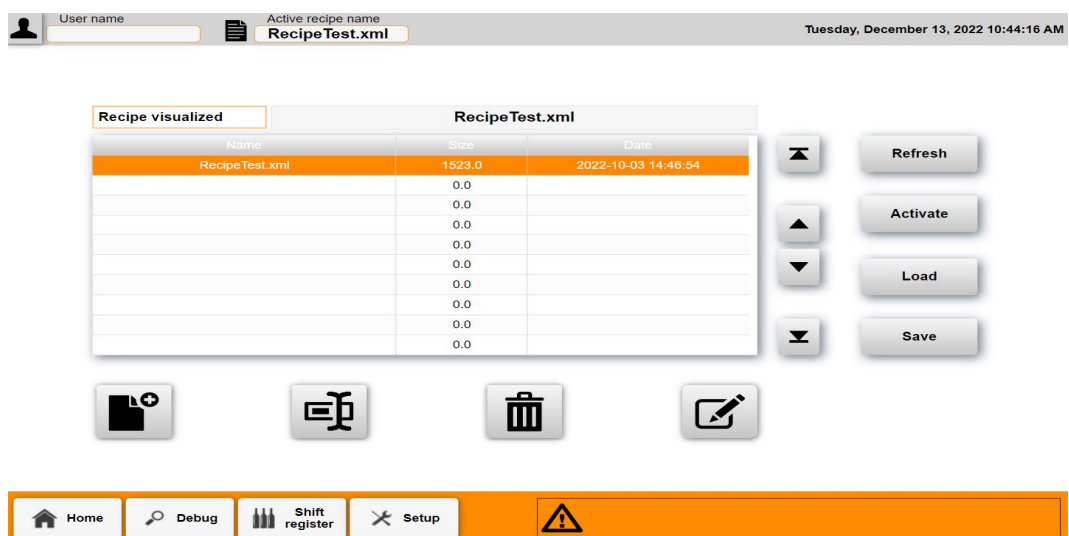


Figure 5.9: Recipe handling page.

In Figure 5.10 the alarm page is showed and this page contains a list of all active alarms in the system with timestamp, message, severity level and state of the alarm and also allows to acknowledge one or all active alarms.

Date and time	Message	Severity	State
Tuesday, December 13, 2022 10:43:04 AM	Device DevXX5 phasing	10	Active
Tuesday, December 13, 2022 10:43:00 AM	Device DevXX4 phasing	10	Active
Tuesday, December 13, 2022 10:42:54 AM	Device DevXX1 phasing	10	Active

Figure 5.10: Alarm page.

This HMI is provided to the customer not as a finished product but as a sort of guide containing all the most useful elements for a machine's HMI in order to show some possible usage of that elements that can be used as a starting point for the customized HMI that almost every customer requires nowadays.

5.4.4 Adding a device

In order to highlight the extreme modularity and flexibility of the Template project, the simple procedure to be actuated in order to add and remove a device will be presented. A device can be built from scratch by the customer but also imported from the vastly furnished set of examples provided by B&R (empty device, master with two slaves, cams, gears and also an ACOPOSTrak device) and then modified in order to meet the customer's needs.

The necessary steps to import a device in the Template are:

1. Import the device from the toolbox from the custom solutions' folder (defined in the options of Automation Studio)
2. Place the four tasks in the suitable timing classes in order to maximize the performances (usually *M_Device* in cyclic number 1, *A_Device* in cyclic number 3, *L_Device* in cyclic number 4 and *F_Device* in cyclic number 8) by placing in high priority classes the most time critical tasks

3. In the Text System folder in the *Configuration View* add the text file for the alarms of the new device
4. Rename the device with the *RenameFull* tool and adjust the hardware configuration in the *Physical View* if order to replicate the machine hardware

This steps are only a few and very fast to be completed in order to obtain a device ready to work and to be plugged in the machine, they can be found in the Documentation folder already included in the Template project.

Devices can be removed from the project with a few steps but can also be simply disabled using the Enable parameter that each device is provided with, allowing the machine to work as if the device was not present.

Demo overview: ACOPOS6D device

The objective of this thesis project is to develop a device fully compatible with the Template project that is able to control an ACOPOS6D system with 28 segments and 28 shuttles in order to create a transportation system for the processing of bottles containing some liquid to be mixed in a solution. The device has been implemented starting from the "Empty Device", one of the standard devices developed internally by B&R, that presents some basic alarms implemented, an empty configuration and recipe files and a motion structure that simply handles the states in which the device can be sent by the Main Logic:

```
PROGRAM _CYCLIC
CASE Device[x].Status.State OF
STATE_INIT:
  Initialization;
STATE_WAIT:
  IF Device[x].Cmd.Emergency THEN
    Device[x].Status.State := STATE_EMERGENCY;
  ELSIF Device[x].Cmd.Phasing THEN
    Device[x].Status.State := STATE_PHASING;
  ELSIF Device[x].Cmd.Production THEN
    Device[x].Status.State := STATE_PRODUCTION;
  ELSIF Device[x].Cmd.Manual THEN
    Device[x].Status.State := STATE_MANUAL;
  END_IF;
STATE_MANUAL:
  Manual;
STATE_PHASING:
  Phasing;
STATE_PRODUCTION:
  Production;
STATE_EMERGENCY:
  Emergency;
END_CASE
// Actions call
MotionReset;
MotionAlarms;
MotionHmi;
END_PROGRAM
```

Figure 6.1: Cyclic.st action of the Empty Device.

As it is visible in Figure 6.1, the device starts in the INIT state and then, after the initialization procedure has been completed, the device will wait in the WAIT state for commands from the Main Logic. The device developed in this thesis project has been realized by starting from this footprint and is able to perform the working cycle described before, remaining completely independent from the rest of the machine.

6.1 mapp 6D function blocks

A function block can be seen as an entity that, given some inputs is able to provide some desired outputs after some internal calculations and manipulations of the inputs. B&R libraries are mainly composed of function blocks for the most various automation tasks and the library *McAcp6D* is no exception. In this section the main function blocks used in this thesis project in order to recognize, power and move the shuttles will be highlighted.

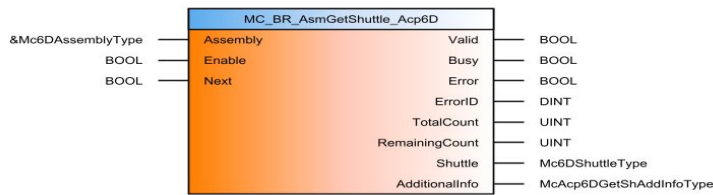


Figure 6.2: MC_BR_AsmGetShuttle_Acp6D function block.

MC_BR_AsmGetShuttle_Acp6D is the function block used to recognize the shuttles on the layout, essential to get the address of each shuttle and then be able to use it in order to give commands to them. It needs as input the reference to the layout (address) defined in the *Configuration View* and the Enable then gives as output the reference to one random shuttle and its ID, together with the total number of sensed shuttles. If the Next input is activated the reference to the shuttle will change and the FB will update the number of shuttles not already acknowledged by the system (RemainingCount).

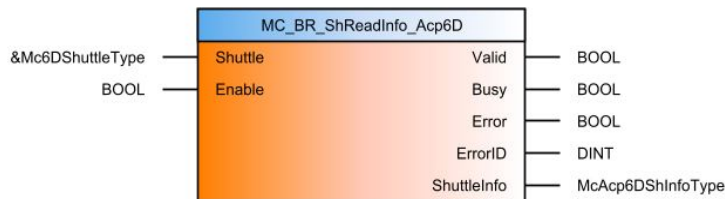


Figure 6.3: MC_BR_ShReadInfo_Acp6D function block.

MC_BR_ShReadInfo_Acp6D takes as input a reference to a shuttle and, when enabled, gives as output the structure *ShuttleInfo* of the *McAcp6D ShInfoType* that contains the ID, state and position in all six degrees of freedom for one shuttle (*Sh* refers to one single shuttle while *Asm* refers to the entire layout). This function block can be called cyclically in order to provide these informations at every PLC cycle and in this project, an array of these function blocks has been defined in order to check the position of all 28 shuttles.

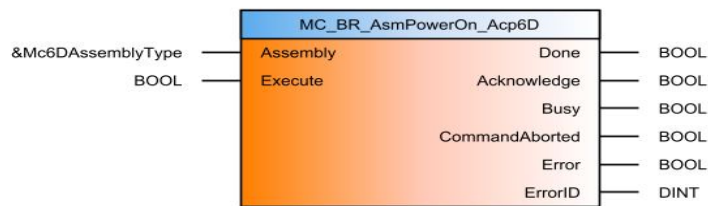


Figure 6.4: *MC_BR_AsmPowerOn_Acp6D* function block.

MC_BR_AsmPowerOn_Acp6D is used in order to bring all shuttles on the layout to the default levitation height defined in the *mapp 6D* configuration files. In this case the input is an Execute and not an Enable so this is meant to be activated and then deactivated when the command has been executed. The Acknowledge output signals that the controller has taken the PowerOn command in memory and the Done output signals when everything has been executed. Once shuttles are in levitation the system is ready to work.



Figure 6.5: *MC_BR_AsmStop_Acp6D* function block.

If a shuttle receives more than one consecutive command it is able to store that commands in its motion buffer and then execute them one after the other, in order to clean the motion buffer, if necessary, the *MC_BR_AsmStop_Acp6D* function block can be used. This will clean all shuttles' motion buffers and break all grouping of shuttles and master-slave relations active on the layout. This function block is available also for a single shuttle and not the entire layout. Another important thing to note is that the stop command triggers a controlled breaking that allows to stop all shuttles in motion in a safe way.

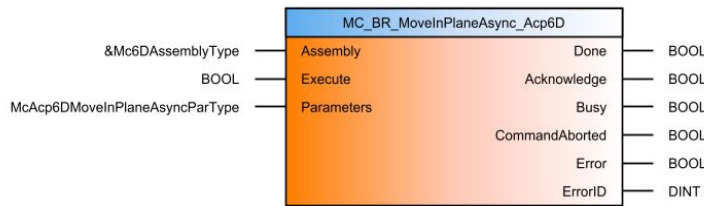


Figure 6.6: MC_BR_MoveInPlaneAsync_Acp6D function block.

MC_BR_MoveInPlaneAsync_Acp6D is a function block that, given a reference to the layout and the `Execute` is able to move all shuttles to the desired locations. These locations are specified as x and y positions for each shuttle in the `Parameters` structure of the *McAcp6DMoveInPlaneAsyncParType* type, as well as the maximum speed and acceleration of transfer. The system will automatically generate the trajectories that bring every shuttle in the desired position avoiding collisions.



Figure 6.7: MC_BR_MoveInPlane_Acp6D function block.

MC_BR_MoveInPlane_Acp6D is the core of each ACOPOS6D application and allows basically to move a shuttle from its initial position to an (x, y) point specified in the `Parameters` structure. In this structure also the speed, acceleration and ending speed of the motion can be specified, with this last parameter that is very useful to concatenate movements in order to increase the system's performances. The motion type can be absolute in the layout or relative to the initial position of the shuttle and it can be a direct motion or decomposed in the two coordinates.

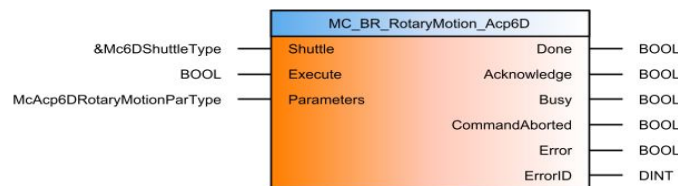


Figure 6.8: MC_BR_RotaryMotion_Acp6D function block.

This function block can be used to make a shuttle rotate with respect to its center in both directions and it can be specified the angle that the shuttle has to reach and velocity and accelerations desired. Shuttles can rotate in this way only in specific regions of the layout and cannot meet obstacle during the rotative motion, so typically a whole segment is reserved for the rotation of each shuttle.



Figure 6.9: MC_BR_MoveArc_Acp6D function block.

The last library function block exploited in this project is used in order to make a shuttle execute an arc motion and in this case it is sufficient to specify the center of the circle on which the shuttle has to move together with the angle that it has to reach.

6.2 Initialization

The initialization sequence is called by the startup of the Main Logic and it is signaled to the operator through the alarm "Device Acp6D initialization" for the whole duration of the process, in four possible different languages (Section 5.1). This procedure is necessary at the startup of the system in order to being able to recognize all shuttles and reference their addresses in the memory.

```

14:
MC_BR_AsmGetShuttle_Acp6D_0.Enable := TRUE;
Device[x].Status.Step := 15;

15:
IF (MC_BR_AsmGetShuttle_Acp6D_0.Valid) THEN
  ShuttleRef[ShcCnt] := MC_BR_AsmGetShuttle_Acp6D_0.Shuttle;
  MC_BR_ShReadInfo_Acp6D_0[ShcCnt].Shuttle := ADDR[ShuttleRef[ShcCnt]
  MC_BR_ShReadInfo_Acp6D_0[ShcCnt].Enable := TRUE;
  ShcCnt := ShcCnt+1;
  MC_BR_AsmGetShuttle_Acp6D_0.Next := FALSE;
  Device[x].Status.Step := 16;
END_IF;

16:
IF (MC_BR_AsmGetShuttle_Acp6D_0.RemainingCount > 0) THEN
  MC_BR_AsmGetShuttle_Acp6D_0.Next := TRUE;
  Device[x].Status.Step := 15;
ELSE
  Device[x].Status.Step := 17;
END_IF;

17:
MC_BR_AsmGetShuttle_Acp6D_0.Enable := FALSE;
MC_BR_AsmGetShuttle_Acp6D_0.Next := FALSE;
Device[x].Status.Step := 18;

```

Figure 6.10: Shuttle recognition algorithm.

The first part of the procedure consists in the cyclic call of two states in which the *MC_BR_AsmGetShuttle_Acp6D* function block is called and an array of addresses is populated. An array of function blocks of the *MC_BR_ShReadInfo_Acp6D* type is also enabled here for each shuttle that is found. Once this procedure has been completed for one shuttle the next state checks if there are other shuttles on the layout that have not been acknowledged and, in that case, goes back to the previous state until the RemainingCount output gets to zero.

```

18:
FOR i:=0 TO MAX_SHUTTLE_NUMBER_ARRAY DO
  IF MC_BR_ShReadInfo_Acp6D_0[i].ShuttleInfo.ShuttleID = j+1 THEN
    ShuttleRefOrd[j].controlif := ShuttleRef[i].controlif;
    j := j+1;
  END_IF;
END_FOR;
IF j = MAX_SHUTTLE_NUMBER THEN
  Device[x].Status.Step := 19;
  j := 0;
END_IF;

19:
FOR i:=0 TO MAX_SHUTTLE_NUMBER_ARRAY DO
  MC_BR_ShReadInfo_Acp6D_0[i].Enable := FALSE;
END_FOR;
Device[x].Status.Step := 20;

20:
FOR i:=0 TO MAX_SHUTTLE_NUMBER_ARRAY DO
  MC_BR_ShReadInfo_Acp6D_0[i].Shuttle := ADR(ShuttleRefOrd[i]);
  MC_BR_ShReadInfo_Acp6D_0[i].Enable := TRUE;
END_FOR;
Device[x].Status.Step := 100;

100: // Init Done
Device[x].Status.Init           := CMD_DONE;
Device[x].Status.Step          := 1000;

```

Figure 6.11: Shuttle order algorithm.

The next phase of the Initialization procedure is aimed to order the array of shuttles references populated before, that are detected randomly by the system at every startup. The two arrays of shuttle references and *MC_BR_ShReadInfo_Acp6D* function blocks have been populated in a way such that in the same position in both arrays there is the reference to the same shuttle. Exploiting this fact the algorithm scans the array of function blocks and, in order of ID, populates another array of shuttle references, this time ordered from shuttle 1 to, in this case, shuttle 28. The procedure is general since it is sufficient to change the constant `MAX_SHUTTLE_NUMBER` once in the device and it is automatically able to handle a different number of shuttles without modifying anything in the code. The last part of the procedure simply consists in the disabling of the ReadInfo function blocks in the array and enabling them again in order to synchronize the index of this array with the ordered references array. This procedure is not strictly needed for the functioning of the application but has been done to allow an easier diagnostic of the project during its development.

6.3 Phasing

In an automatic machine, usually, the phasing procedure is carried out before starting the nominal production in order to power all axes and components of the machine and to bring everyone of them in the correct initial position. In this case the operations that must be carried out in order to correctly recover the system are the powering of the shuttles and the movement of all shuttles in the station positions. The powering of the system is done using the *MC_BR_AsmPowerOn_Acp6D* function block while the positioning of each shuttle in a determined station is done by using the *MC_BR_MoveInPlaneAsync_Acp6D* function block and simply specifying the end position for each shuttle, as well as giving the correct destination to each shuttle in order to make it recognizable by the station.

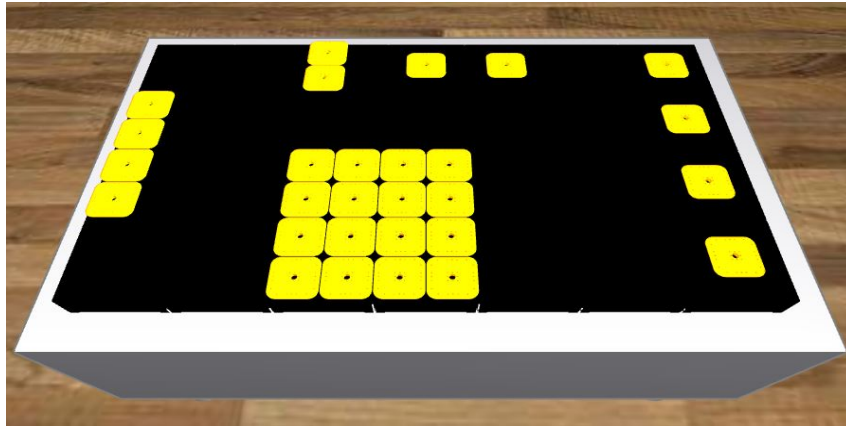


Figure 6.12: Initial positions of the 28 shuttles.

In this thesis project 28 shuttles have been considered, and they will be directed to the stations slots positions by the phasing procedure, in order to start the production. If the number of shuttles is higher than the number of stations it is sufficient, in the phasing procedure, to command to these shuttles to go in a station, that will be occupied, and so the shuttle will move in that direction and enter in the OBSTACLE state, creating a queue of shuttles that will complete their movement once the station will be freed.

The implemented phasing procedure is able to manage a stop of the machine: if the stop button is pressed during the production phase, the device will stop its execution and, at the restart of the machine, the state machine of the phasing procedure, will check if shuttles are still powered and, in this case, will skip the powering phase and execute the *MC_BR_MoveInPlane Async_Acp6D* directly. The implemented procedure is not able to restart in-phase and so brings to the discharge of all products present on the machine during the

stop operation, a possible future improvement could be to implement a recovery procedure able to restart from the stop position without necessitating of a rephasing step.

6.4 Stations concept

The main idea behind this thesis project, is that the control is not shuttle-oriented but station-oriented: the layout has been divided in five working stations that have to interface themselves with the loading and unloading robots, the labeler machine and the capping machine, two stations that are used to create the correct pattern of shuttles and one buffer station. These stations are represented by structured text actions called cyclically by the ACOPOS6D device during the production state of the machine and are composed of three main steps:

- Recognition of the shuttles present in the station
- Processing of the products brought by the shuttles
- Sending the shuttles to the next station

These three steps are repeated cyclically by the stations that are defined in the form of finite states machines. A finite state machine is composed of states in which a certain sequence of operations is executed and a series of transitions between these states that are triggered by some conditions. A station, from the ACOPOS6D point of view, is a set of positions in the layout in which shuttles have to bring products in order to be processed. As it is visible in the previous section, all function blocks that command some movements to shuttles require a reference to the shuttle, represented by an address to a memory cell, so something that is able to recognize which shuttles are in position at every time is required. Each station is characterized by a structure of the *StationType* and this structure type is interesting because allows to increase the repeatability of a station, allowing the creation of a new station to be faster, while also generating more readable and coherent code:

Component	Data Type	Checkbox 1	Checkbox 2	Checkbox 3	Checkbox 4	Checkbox 5	Checkbox 6	Checkbox 7	Checkbox 8
StationType									<input checked="" type="checkbox"/>
StationStep	INT								<input checked="" type="checkbox"/>
Error	BOOL								<input checked="" type="checkbox"/>
ErrorStep	INT								<input checked="" type="checkbox"/>
IDInPos	UINT[0..10]								<input checked="" type="checkbox"/>
ShtInPos	Mc6DShuttleT...								<input checked="" type="checkbox"/>
MC_BR_MoveInPlane_Acp6D...	MC_BR_Mov...								<input checked="" type="checkbox"/>
MC_BR_MoveInPlane_Acp6D...	MC_BR_Mov...								<input checked="" type="checkbox"/>
MC_BR_MoveArc_Acp6D_0	MC_BR_Mov...								<input checked="" type="checkbox"/>
MC_BR_MoveArc_Acp6D_1	MC_BR_Mov...								<input checked="" type="checkbox"/>
Acp6DStat_0	Acp6DStat								<input checked="" type="checkbox"/>

Figure 6.13: Station structure.

The structure is provided with a `Step` variable that allows the implementation of the state machine and an `Error` and `ErrorStep` variables that can be used to signal if an error in the code occurred and in which step of the state machine. The `IDInPos` and `ShtInPos` arrays are used to temporarily save the references to the shuttles that the station is processing, being able to give commands to the shuttle even if it moves from the station position and is not seen anymore in position, these arrays are emptied at each iteration of the station after the shuttles have been sent to the next station. This is done in order to be able to recognise on the fly a new shuttle that reached the station position. In addition to these variables each station structure is already provided with four function blocks that can be used to move the shuttles in the layout (Section 6.1) and a customized function block developed in this thesis project and described in the next Section, that is used to recognize shuttles in a station.

6.5 Shuttle recognition function block

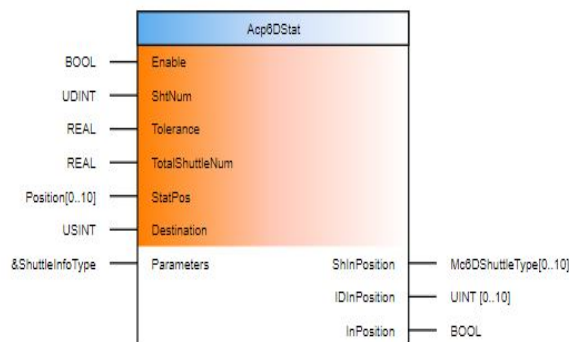


Figure 6.14: Acp6DStat function block.

In order to carry out the recognition of shuttles in each station in real time, the *Acp6DStat* library has been created. A library in Automation Studio includes four files:

- A file with the definitions of all structures that will be used by the library, in this case a *Position* type ((x, y) coordinates) and *ShuttleInfoType* have been created.
- A file with the definitions of constants that can be useful for the library.
- The definition of all function blocks and functions contained in the libraries with inputs, outputs and internal variables for function blocks.
- The code of said function blocks and functions.

6.5.1 Inputs

The *Acp6DStat* function block is the core of this library and it has been thought as the "brain" of a station. This function block has been set up to be general and abstract with respect to the application since it is possible to configure as inputs the number of shuttles in the system, the number of shuttles in the specific station, the positions of each slot in the station and the tolerance that must be used for the acknowledgement of a shuttle in position. The activation is of the Enable kind, so the function block, once enabled, runs endlessly and its outputs can be read at each instant of time.

Name	Type	Reference	Replicable
Position			<input checked="" type="checkbox"/>
X	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Y	REAL	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ShuttleInfoType			<input checked="" type="checkbox"/>
Shuttle	Mc6DShuttleT...	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ID	UINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Destination	USINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Color	USINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Position	McAcp6DShP...	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 6.15: ShuttleInfoType and Position structures.

The Parameters structure that is passed as input to the function block must be updated at every PLC cycle and contains the address, ID, position, destination and color of each shuttle. It is populated in the *cyclic.st* of the device by taking the outputs of the *MC_BR_ShReadInfo_Acp6D* function block and the ordered addresses of shuttles.

6.5.2 Working principle and outputs

Once enabled, the function block resets its outputs with a FOR cycle in order to delete old data that could lead to localize shuttles that are not actually in the station. At this point two nested FOR cycles are called in order to scan all slots in the station and all shuttles' positions on the layout. A shuttle is considered in position only if it is in the position of the slot (+/- some tolerance that is used since the position is relevated with an uncertainty in the order of the μm) and if its destination coincides with the destination of the station, in order to avoid to process multiple times the same shuttle once it has been recognized in position. Once one shuttle has been found inside the Tolerance and with the right Destination, its ID and reference are saved in two internal variables and the internal FOR cycle is interrupted, going on to search a shuttle in the next slot of the station. Once all shuttles in the station have been found, their ID and addresses are forwarded as outputs and the function

block signals that through the InPosition output.

```

FUNCTION_BLOCK AcpeDStat
IF Enable THEN
FOR j := 0 TO ShtNum-1 DO
IDinPosition[j] := 0;
ShInPosition[j] := Null;
InPosition := FALSE;
END_FOR

FOR j := 0 TO ShtNum-1 DO
FOR i := 0 TO TotalShuttleNum-1 DO
IF (Parameters[i].Position.X < StatPos[j].X + Tolerance) AND (Parameters[i].Position.X > StatPos[j].X - Tolerance)
AND (Parameters[i].Position.Y < StatPos[j].Y + Tolerance) AND (Parameters[i].Position.Y > StatPos[j].Y - Tolerance)
AND Destination = Parameters[i].Destination THEN
IDinPositionTemp[j] := Parameters[i].ID;
ShInPositionTemp[j].controlif := Parameters[i].Shuttle.controlif;
EXIT;
END_IF
END_FOR
IF i > TotalShuttleNum-1 THEN
InPosition := FALSE;
EXIT;
END_IF;
END_FOR;

IF j > ShtNum-1 THEN
FOR k := 0 TO ShtNum-1 DO
IDinPosition[k] := IDinPositionTemp[k];
ShInPosition[k].controlif := ShInPositionTemp[k].controlif;
END_FOR
InPosition := TRUE;
END_IF

IF InPosition = FALSE THEN
FOR k := 0 TO ShtNum-1 DO
IDinPosition[k] := 0;
ShInPosition[k] := Null;
END_FOR
END_IF
END_IF
END_FUNCTION_BLOCK

```

Figure 6.16: Function block's code.

6.6 Working stations

Five working stations have been identified in the system, as well as three routing stations that are used in order to manage the shuttle's traffic and avoid deadlocks. A deadlock happens when two shuttles collide and each one is trying to move in the opposite direction of the other shuttle:



Figure 6.17: Shuttle deadlock.

This situation must be avoided cause it would cause the stall of the application, that is a very difficult situation to come out of while restarting the production

in-phase. An interesting characteristic of the control that is actuated by segments is that shuttles can collide with each other, entering the OBSTACLE state, and then complete their movement once the path becomes free, allowing the possibility to create queues of shuttles that wait to enter the same station without any problem for the functioning of the machine.

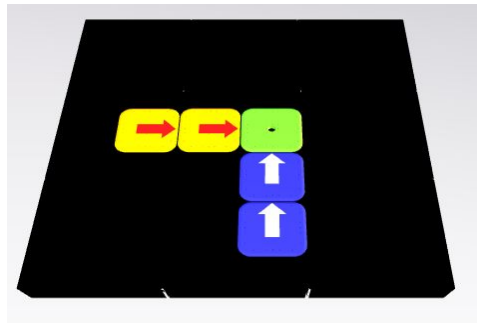


Figure 6.18: Shuttles in obstacle state.

In the example in Figure 6.18, the yellow and blue shuttles are trying to move to the green shuttle's position, and, if the code is written well and the green shuttles move in one of the free directions, a yellow or blue shuttle will take its place and the others will, again, form the same queue without presenting any errors. This characteristic of the motion profile is the key component for all the code developed in this project, since it is the possibility to create queues that allows the structure of stations be the one described in Section 6.4.

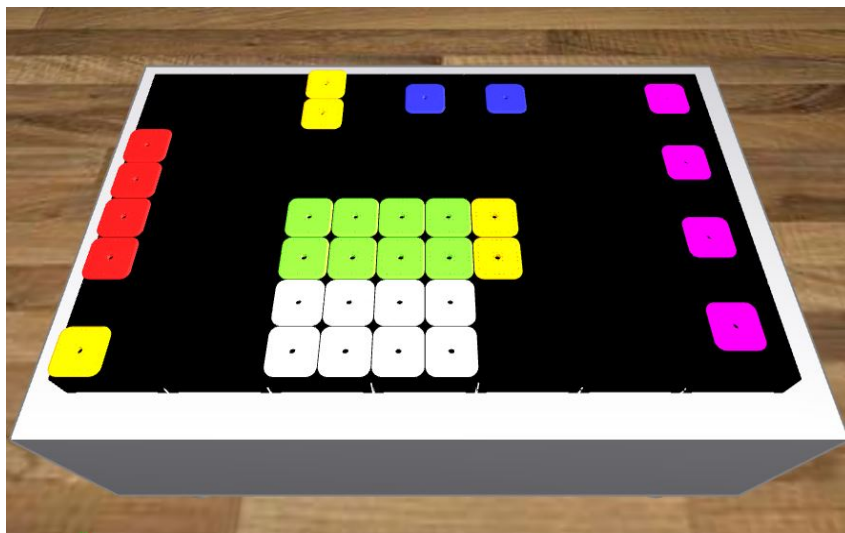


Figure 6.19: Working stations: load (red), labeling (blue), shaker (pink), capping (green), unloading (white) and routing stations (yellow).

In Figure 6.19 the stations defined on the layout are showed, and the next paragraphs of this thesis will describe in depth their goals and behaviour.

6.6.1 Load station

The red one is the station where the bottles are loaded on the shuttles by an external robot and are kept in place by means of a product holder, fixed to each shuttle before the startup of the machine. A product holder is a mechanic component designed with a CAD by the mechanical department of the machine builder costumer. This component has not been physically realised for this theis project because the aim was to obtain a proof of concept of the actual possibility to implement this kind of systems with this technology, for this reason the working cycle implemented has been realized without products on the machine.



Figure 6.20: Loading station's recognition and processing step.

The *Ac6DStat* FB is enabled after setting the suitable inputs for this station. In the examined case the loading robot loads a batch of 4x1 products on the shuttles but if, for example, the robot loads a batch of 4x2 products at each motion, it is sufficient to pass the eight positions to the function block, as well as the number of shuttles that must be acknowledged and the station starts working seamlessly with the new batch of shuttles. After checking the presence of all shuttles in the station, it saves in suitable arrays the ID and reference of the found shuttles, changes their color for the simulation (Section 6.8) and

then waits for the robot to load the products. In this case the loading robot has been simulated by simply waiting a specified time, that is taken from a recipe file, allowing to change at runtime the loading time and see how this affects the production speed.

The next part of the code for this station sends the processed shuttles to the next one. Since the labeling station performs a rotation of the shuttles in order to apply a label on the bottles, the shuttles cannot be directly sent there because they would enter in deadlock with the rotating ones, but making them wait for the end of the rotation in the loading station would decrease the productivity of the machine, so shuttles are sent in a buffer station where they can wait for the rotation to end and, since shuttles are kept still, here queues can be created, allowing twelve shuttles to be loaded while the labeling station complete its work, instead of just four, as it is visible in the next Figure:

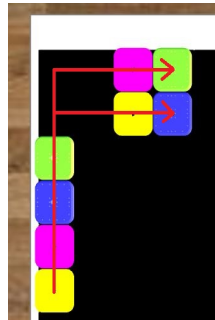


Figure 6.21: Shuttles's routing in the loading station.

```

3:
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Shuttle := ADR(LoadStation.ShtInPos[3]);
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Velocity := RecipeAcp6D.TransferSpeed;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.EndVelocity := 0;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Acceleration := RecipeAcp6D.TransferAcceleration;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Mode := 0;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Path := mcACP6D_PATH_Y_THEN_X;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Position.X := ConfigAcp6D.Stations[4].Position.X;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Parameters.Position.Y := ConfigAcp6D.Stations[4].Position.Y;
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := TRUE;

1
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Shuttle := ADR(LoadStation.ShtInPos[2]);
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Velocity := RecipeAcp6D.TransferSpeed;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.EndVelocity := 0;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Acceleration := RecipeAcp6D.TransferAcceleration;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Mode := 0;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Path := mcACP6D_PATH_Y_THEN_X;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Position.X := ConfigAcp6D.Stations[5].Position.X;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Parameters.Position.Y := ConfigAcp6D.Stations[5].Position.Y;
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := TRUE;

FOR i := 0 TO MAX_SHUTTLE_NUMBER_ARRAY DO
  FOR j := 0 TO 10 DO
    IF ShuttleInfo[i].ID = LoadStation.IDInPos[j] THEN
      ShuttleInfo[i].Destination := 2;
    END_IF
  END_FOR
END_FOR

FOR i := 0 TO MAX_SHUTTLE_NUMBER_ARRAY DO
  FOR j := 0 TO 3 DO
    IF ShuttleInfo[i].ID = LoadStation.IDInPos[j] THEN
      ShuttleInfo[i].Color := 6;
    END_IF
  END_FOR
END_FOR

LoadStation.StationStep := 4;

IF NOT (Device[X].Cmd.Production) THEN
  LoadStation.Acp6DStat_0.Enable := FALSE;
  LoadStation.StationStep := 0;
END_IF

```

Figure 6.22: Loading station's routing step.

The movements are generated such that the green and blue shuttles (shuttles 3 and 2 in Figure 6.22) will be sent with two *MC_BR_MoveInPlane_Acp6D* function blocks to the buffer station's positions and this is done in a single movement cause it is exploited the possibility to decompose the movements in the two motion directions: *x* and *y* (parameter *mcACP6D_PATH_X_THEN_Y*). After the commands are sent the Destination of the shuttles recognized by the station is changed as well as their color.

```

4:
IF LoadStation.MC_BR_MoveInPlane_Acp6D_0.Acknowledge AND LoadStation.MC_BR_MoveInPlane_Acp6D_1.Acknowledge THEN
  LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := FALSE;
  LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := FALSE;
  LoadStation.StationStep := 5;
ELSIF LoadStation.MC_BR_MoveInPlane_Acp6D_0.Error AND LoadStation.MC_BR_MoveInPlane_Acp6D_1.Error THEN
  LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := FALSE;
  LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := FALSE;
  LoadStation.Error := TRUE;
  LoadStation.ErrorStep := 4;
END_IF

IF NOT(Device[x].Cmd.Production) THEN
  LoadStation.Acp6DStat_0.Enable := FALSE;
  LoadStation.StationStep := 0;
END_IF

5:
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Shuttle := ADR(LoadStation.ShtInPos[1]);
LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := TRUE;

LoadStation.MC_BR_MoveInPlane_Acp6D_1.Shuttle := ADR(LoadStation.ShtInPos[0]);
LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := TRUE;

LoadStation.StationStep := 6;

IF NOT(Device[x].Cmd.Production) THEN
  LoadStation.Acp6DStat_0.Enable := FALSE;
  LoadStation.StationStep := 0;
END_IF

```

Figure 6.23: Loading station's routing step part 2.

The next step in the state machine waits for the Acknowledge output of the function blocks in order to being able to continue the station working process even if the two movements are not completed right away, because of queues in the buffer station and then the movement commands are sent again to the next two shuttles in position in the station. It is not necessary to specify again the motion parameters because the two function blocks used are the same and they already contain the parameters defined for the previous couple of shuttles.

```

6:
IF LoadStation.MC_BR_MoveInPlane_Acp6D_0.Acknowledge AND LoadStation.MC_BR_MoveInPlane_Acp6D_1.Acknowledge THEN
  LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := FALSE;
  LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := FALSE;
  LoadStation.StationStep := 7;
ELSIF LoadStation.MC_BR_MoveInPlane_Acp6D_0.Error AND LoadStation.MC_BR_MoveInPlane_Acp6D_1.Error THEN
  LoadStation.MC_BR_MoveInPlane_Acp6D_0.Execute := FALSE;
  LoadStation.MC_BR_MoveInPlane_Acp6D_1.Execute := FALSE;
  LoadStation.Error := TRUE;
  LoadStation.ErrorStep := 6;
END_IF

IF NOT(Device[x].Cmd.Production) THEN
  LoadStation.Acp6DStat_0.Enable := FALSE;
  LoadStation.StationStep := 0;
END_IF

```

Figure 6.24: Loading station's routing step part 3.

The last part of the station's code is dedicated to resetting the arrays with the addresses and IDs of the recognized shuttles in order to go back to the state where the station waits for the InPosition pin of the function block, as visible in Figure 6.25.

```

7:
  FOR i := 0 TO 3 DO
    brsmemset(ADR(LoadStation.ShtInPos[i]),0, sizeof(LoadStation.ShtInPos[i]));
    LoadStation.IDInPos[i] := 0;
  END_FOR

  LoadStation.StationStep := 2;

  IF NOT(Device[x].Cmd.Production) THEN
    LoadStation.Acp6DStat_0.Enable := FALSE;
    LoadStation.StationStep := 0;
  END_IF

END_CASE

LoadStation.Acp6DStat_0();
LoadStation.MC_BR_MoveInPlane_Acp6D_0();
LoadStation.MC_BR_MoveInPlane_Acp6D_1();
TON>Loading();

END_ACTION

```

Figure 6.25: Loading station's ending code.

6.6.2 Buffer station

The buffer station has been created in order to allow shuttles to wait for the labeling station to free itself up and so it is not a processing station: after the initialization of the function block with the correct positions of the two slots belonging to this station, the station waits for the *Acp6DStat* function block to give the InPosition output but also for a global boolean variable (for the device, not for the whole system, in order to maintain the modularity of the Template project) to become FALSE in order to check for the labeling station to be free. Once these conditions have been met, the two shuttles are sent to the labeling station by making them move first in the *x* coordinate and then in the *y* as it can be seen in the next Figure, and the global variable is put at TRUE in order to stop the next two shuttles from moving to the labeling station and causing a deadlock.

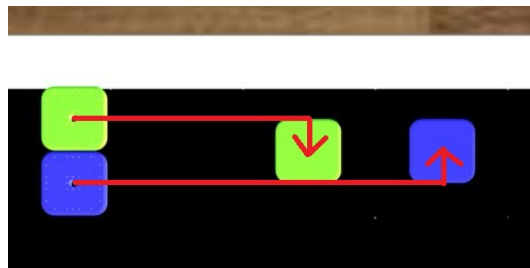


Figure 6.26: Shuttles's routing in the buffer station.

6.6.3 Labeling station

In the labeling stations (blue shuttles in Figure 6.19) the two shuttles perform a 360° rotation around the z axis in order to allow an external labeler to print a specified label on each side of the bottles, thanks to the `MC_BR_RotaryMotion_Acp6D` function block. The rotation has been computed as four consecutive discrete counterclockwise rotations of 90° but, if needed, this can be also done in a single continuous rotation. Rotations are performed as soon as the two shuttles are acknowledged by the function block and then the station checks if the two upper slots of the shaker station are free and sends the shuttles there, otherwise the two shuttles are sent to the lower two slots with a different routing.

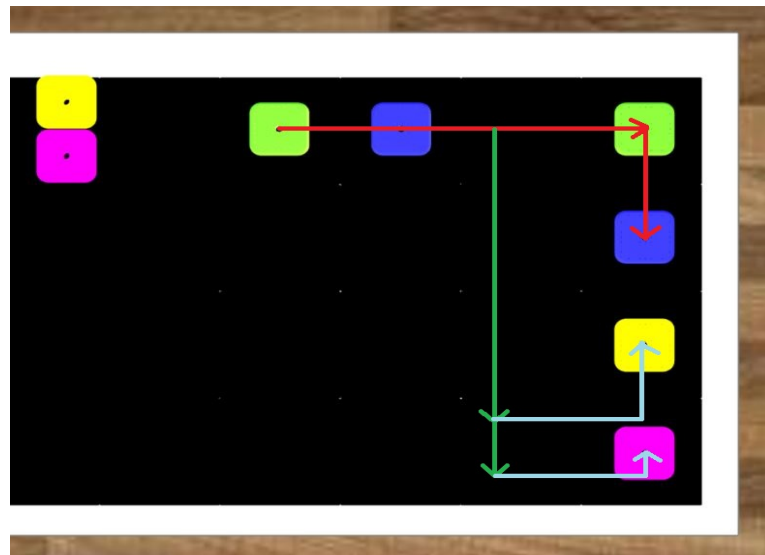


Figure 6.27: Possible routings of shuttles in the labeling station.

Once the shuttles are sent to the shaker station, it is signaled that the labeling station is free and so two new shuttles can be acknowledged in order to be processed. In Figure 6.27 the two possible routings are showed. The red arrows represent the movement that shuttles have to perform in order to reach the upper two slots of the shaker station and this movement is obtained with one single function block. The more interesting routing is the one developed in order to send the shuttles to the other two slots. These movements have been obtained by calling two `MC_BR_MoveInPlane_Acp6D` function blocks for each shuttle, one after the other, by exploiting the motion buffer discussed in Chapter 4.1.4.

Once the first function block obtains the `Execute`, the step machine goes to the next state and wait for the `Acknowledge` output, not the `Done` one, and

then the same function block can be used to give a different command to the shuttle without having to wait for the first movement to end. This principle is used in general in each station, even if one single movement is performed: after obtaining the acknowledge, the station can go back to the state in which it waits for the new shuttles to be recognized, allowing to speed up the production because it does not have to wait for each shuttle to reach the next station and can directly start to process the next ones.

6.6.4 Shaker station

The shaker station has been thought to outline one of the most peculiar motion profiles that the ACOPOS6D's shuttles are able to perform. The *MC_BR_MoveArc_Acp6D* function block is usually used in order to decrease the traveling time when a shuttle has to compute a motion that is not on a straight line, allowing to smooth the profile of curves and changes of directions. If the arc motion is performed with a sufficiently small radius and high speed, the shuttles will perform a motion that is able to mix liquids contained in the products brought by the shuttles, shakering them in order to obtain a solution. More than one consecutive shaker motion can be obtained by defining as target angle a multiple of 6.28 rad that corresponds to a 360° cycle. In this application the number of cycles is defined in the recipe file and then it is multiply by 6.28 in order to obtain the proper target angle.

The shaker station starts to work once all four shuttles have been recognized by it and, after the process has been completed, shuttles are sent to the index station with two different routings for the upper and lower couple of shuttles:

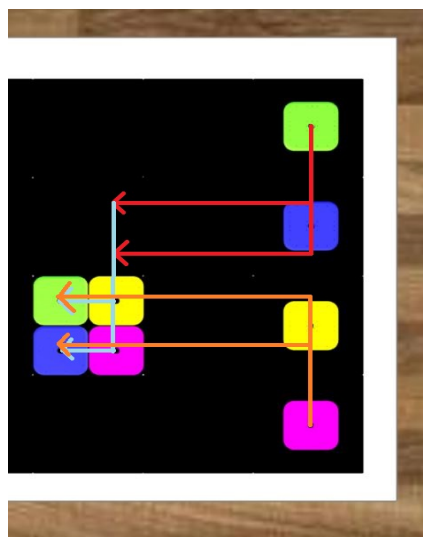


Figure 6.28: Shuttles' routing in the shaker station.

The routing of the shuttles has been thought in order to manage the traffic that is very high in this specific portion of the layout, considering that during the movement of the four shuttles towards the index station, two shuttles will move from the labeling station to two of the slots belonging to the shaker station, that now are free. The three different motions that can be seen in Figure 6.28 are executed almost simultaneously and allow a correct routing of the shuttles, and they are still executed using the `MC_BR_MoveInPlane_Acp6D` function blocks and decomposing the motion coordinates. Since the shuttles in output from the shaker station are more than the ones that the index station can allocate, the first two shuttles will actually reach the station while the next two will create a queue and wait for the station to get free.

6.6.5 Index station

The index station is not visible during the nominal working process of the machine because it is a routing station where decisions are taken on the fly. Its aim is to create the eight shuttle's pattern that is then processed by the capping station. Once two shuttles are recognized by the station, the state machine can move to four different states where shuttles are then sent to form the first, second, third or fourth column of the pattern, and this is realized with an index that is changed at the end of each of these states so that, at the next execution cycle of the station, the next couple of shuttles will form the next column of the pattern. The pattern is formed starting from the left and going to the right, in order to avoid deadlock problems.

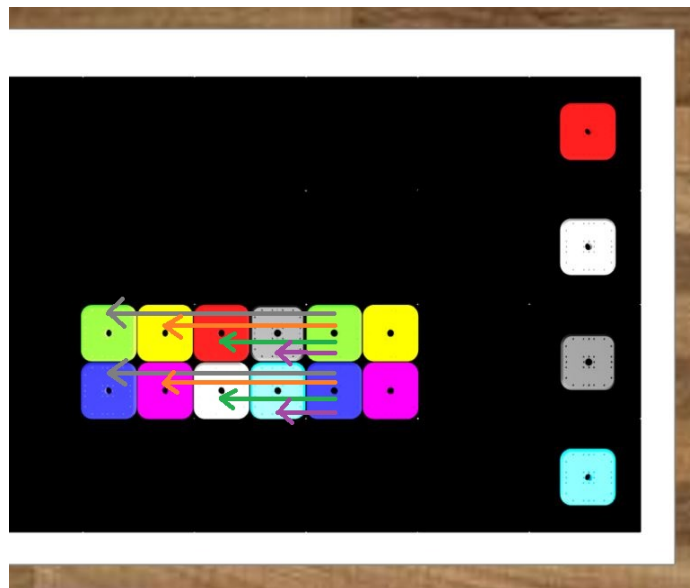


Figure 6.29: Shuttles' routing in the index station.

The sequence of motions can be seen clearly in Figure 6.29: the first two shuttles that are processed by the index station perform the grey movement, then the index is changed in order to execute the orange movement for the next couple of shuttles and so on and so forth up to the violet movement, after which the index is reset because a new pattern has to be created.

6.6.6 Capping station

In this station, bottles are capped by a robot placed over the layout. In order to simulate this, again, a timer has been used and its value can be changed at runtime since it is passed to the station by the recipe file. Since the pattern is composed by eight shuttles as the one processed by the next station, the shuttles' movements have been "concatenated" using the grouping option. The grouping of shuttles allows the possibility to command the same movement to a series of shuttles by simply passing the command to one single shuttle in the group. The grouping procedure is done in a series of consecutive steps:

- Create the group by means of the *MC_BR_ShGroupCreate_Acp6D* function block and save it in an appropriate structure of the *MC6DSShuttleGroupType*
- Add the desired (eight in this specific case) shuttles references to the group by means of the *MC_BR_ShGroupAddShuttle_Acp6D* function block
- Couple the shuttles in the group (if the coupling is not executed shuttles cannot be moved as a group even if they have been added to one) with the *MC_BR_ShGroupCoupleCtrl_Acp6D* function block

And after the creation of the group the upper left shuttle of the pattern is moved to the corresponding position in the Unloading station. Since shuttles are coupled they will move only once all of them can perform the desired movement, so only after the at least a line of the unloading station's pattern is free.

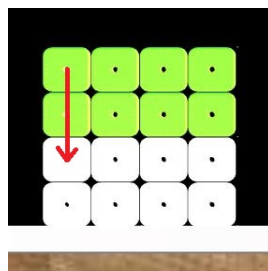


Figure 6.30: Routing for the capping station.

The last part of the code is aimed to decouple the shuttles in the group by means of the already used *MC_BR_ShGroupCoupleCtrl_Acp6D* function block and then deleting the group in order to avoid problems in the control of the shuttles in the next steps with the *MC_BR_ShGroupDelete_Acp6D* function block.

6.6.7 Unloading station

In the unloading station a robot discharges batches of eight products from the shuttles and this is, again, simulated with a timer that can be configured from the recipe file. After the unload of the products, shuttles must be routed in order to restart the working cycle and so they must be sent to the loading stations positions. Since the loading pattern is made of four shuttles, an intermediate station has been created.



Figure 6.31: Routing for the unloading station.

The routing from the unloading station to the direction station is done by executing, in order, the blue, violet, red and green movements visible in Figure 6.31 in order to send each shuttle to the direction station.

6.6.8 Direction station

The direction station is needed in order to create the pattern of four shuttles from which the working cycle then starts. This is necessary because, since all commands that send the eight shuttles from the unloading station to the direction one are given in sequence to the shuttles and it is not guaranteed that they will be executed exactly in the specified order. It can happen that a shuttle belonging to the bottom row is blocked by the shuttles in the upper row and this denies the possibility to directly send the shuttles from the unloading station to the loading one.



Figure 6.32: Routing for the direction station.

The direction station works similarly to the index station and, acknowledged one shuttle in position, sends it to one the four positions belonging to the loading station with four states that work in parallel in the state machine. The index changes every time one shuttle has been sent in the pattern and signals the next motion to be executed, it is reset once four shuttles have been sent to the unloading station and then the next ones will create a queue waiting to reach their final positions.

6.7 Configuration and recipe files

```
<?xml version="1.0" encoding="UTF-8" ?>
<DATA>
  <Element Name="ConfigAcp6D" Type="PvParameter">
    <Group ID="ConfigAcp6D">
      <Property ID="Enable" DataType="BOOL" Value="true" />
      <Property ID="SlotProdReg" DataType="UDINT" Value="1" />
      <Group ID="Stations">
        <Group ID="[0]">
          <Group ID="Position">
            <Property ID="X" DataType="REAL" Value="0.06" />
            <Property ID="Y" DataType="REAL" Value="0.3" />
          </Group>
        </Group>
        <Group ID="[1]">
          <Group ID="Position">
            <Property ID="X" DataType="REAL" Value="0.06" />
            <Property ID="Y" DataType="REAL" Value="0.42" />
          </Group>
        </Group>
        <Group ID="[2]">
          <Group ID="Position">
            <Property ID="X" DataType="REAL" Value="0.06" />
            <Property ID="Y" DataType="REAL" Value="0.54" />
          </Group>
        </Group>
      </Group>
    </Element>
  </DATA>
```

Figure 6.33: Configuration file.

The configuration file needed for the device to work contains the Enable of the device (if the Enable is set at FALSE the Main Logic will not acknowledge the device at the startup and will be able to work as if it was not present) and the slot in the product register, if used, that the device will occupy, together with the position of each shuttle in each station in the layout, as it is visible in Figure 6.33 for the first three slots. This file is loaded in the machine at the startup and can be modified only before that operation, allowing to change easily the position of the stations in the layout.

```
<?xml version="1.0" encoding="UTF-8"?>
<DATA>
  <Element Name="F_Acp6D:HmiRecipeAcp6D" Type="FvParameter">
    <Group ID="F_Acp6D:HmiRecipeAcp6D">
      <Property ID="Parameter1" DataType="REAL" Value="0" />
      <Property ID="T_Load" DataType="TIME" Value="3000" />
      <Property ID="T_Cap" DataType="TIME" Value="5000" />
      <Property ID="T_Unload" DataType="TIME" Value="5000" />
      <Property ID="RotationSpeed" DataType="REAL" Value="100" />
      <Property ID="RotationAcceleration" DataType="REAL" Value="100" />
      <Property ID="ShakerSpeed" DataType="REAL" Value="50" />
      <Property ID="ShakerAcceleration" DataType="REAL" Value="100" />
      <Property ID="ShakerCyclesNumber" DataType="INT" Value="8" />
      <Property ID="TransferSpeed" DataType="REAL" Value="2" />
      <Property ID="TransferAcceleration" DataType="REAL" Value="20" />
    </Group>
  </Element>
</DATA>
```

Figure 6.34: Recipe file example.

The recipe file, instead, contains all variables that can be changed at runtime in order to modify the working cycle of the machine: the three times that simulate the robot operations in the loading, capping and unloading stations, the speed and acceleration of the rotation in the labeling station, the speed and acceleration of motion of all shuttles when they are traveling from a station to another and some parameters for the shaker station. The ShakerCyclesNumber is then multiplied for 6.28 rad (360°) in the shaker station and passed as the target angle for the *MC_BR_MoveArc_Acp6D* function block, in order to realize the desired number of cycles.

6.8 Scene Viewer simulation

In order to visualize the results of this thesis project without having to always rely on the real hardware, a simulation has been set up on the B&R's software Scene Viewer. It is a visualization tool that allows to link variables to 3D objects in order to visualize their evolution in time and it's fully integrated in the B&R portfolio, allowing to integrate robots, and also representation of shuttles and segments in order to simulate ACOPOSTrak, SuperTrak and ACOPOS6D's applications. The 3D objects that can be added in Scene Viewer can be the predefined ones offered by the highly furnished catalog or directly by user developed CAD models. The variables that can be binded to these

objects are shared using OPC-UA from Automation Studio and then the Scene Viewer client receives these variables that can be used to animate the simulations developed on it.

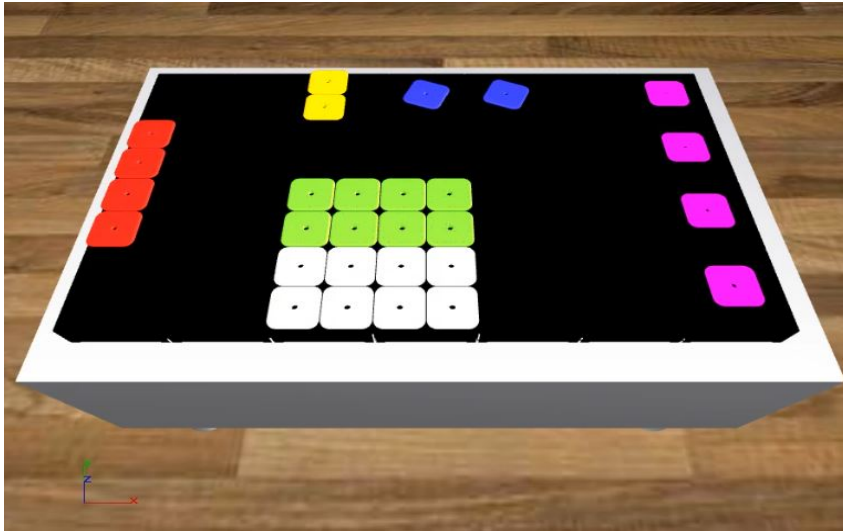
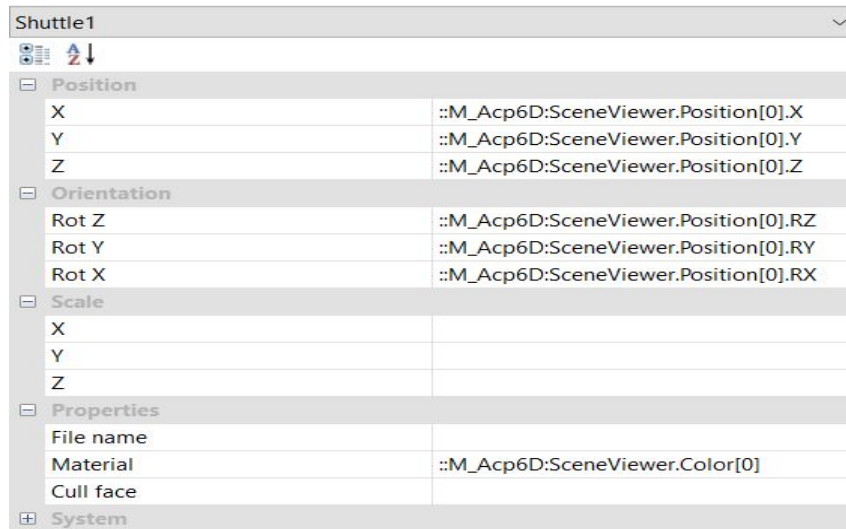


Figure 6.35: Scene Viewer simulation of the demo.

In Figure 6.35 the simulation developed is visible: the 28 segments used in the demo have been placed in known coordinates that start from $(0,0)$ on the lower left corner of the lower left segment. The 28 shuttles have been linked to seven variables each: their position in the six directions are shared cyclically by a variable called *SceneViewer* belonging to the *SceneViewerType* structure (composed by seven arrays) that is populated in the *cyclic* program of the device with position and orientation provided by the function block *MC_BR_ShReadInfo_Acp6D* and with an array of integers called *color*. This variable is binded to the material of each shuttle in Scene Viewer and it is handled by each station in order to change the color of each shuttle that is undergoing some operations in the station. The position in x , y and z directions is provided by Automation Studio in meters while in Scene Viewer millimeters are used so the appropriate conversions are computed before passing the value to the *SceneViewer* variable. The same thing is done for orientation that is expressed in radians in Automation Studio and in degrees in Scene Viewer.



Shuttle1	
Position	
X	::M_Acp6D:SceneViewer.Position[0].X
Y	::M_Acp6D:SceneViewer.Position[0].Y
Z	::M_Acp6D:SceneViewer.Position[0].Z
Orientation	
Rot Z	::M_Acp6D:SceneViewer.Position[0].RZ
Rot Y	::M_Acp6D:SceneViewer.Position[0].RY
Rot X	::M_Acp6D:SceneViewer.Position[0].RX
Scale	
X	
Y	
Z	
Properties	
File name	
Material	::M_Acp6D:SceneViewer.Color[0]
Cull face	
System	

Figure 6.36: Position and color bindings for the first shuttle.

6.9 Experimental results

The developed project has not only been tested in simulation but also on a real layout made of 28 segments with 28 shuttles:



Figure 6.37: Physical layout after phasing.

What has been developed is an automatic machine under every aspect and it has been tested on the physical layout for 24 hours without presenting any alarm or stop of production. The developed device is completely compatible with a base Template project and can be fully integrated in it following the steps explained in Chapter 5.4.4, as any other custom solution offered by

B&R. It presents a fully integrated HMI on which it is possible to visualize the device's alarms for the initialization and phasing steps and from which it is possible to fully control the device itself. By pressing the start button the shuttles reach their initial positions and the working cycle, that starts automatically, can be interrupted in any point with the pression of the stop button. The machine can easily be restarted from the phasing positions again pressing the start button.

The working cycle implemented has been thought to show the potentialities of the ACOPOS6D system: the number of shuttles processed in each station changes continuously in order to show the flexibility of the system, that is able to manage the movements of shuttles with very high dynamics: the system can reach up to 2 m/s in speed with 20 m/s^2 of acceleration for what concerns the motions between stations without losing precision and repeatability in the positioning of shuttles. The interesting part of this is that the system is able to reach these speeds but also to work at lower speeds in order to adapt to any kind of products and their specific dynamical needs. The working cycle is completely customizable at runtime thanks to the recipe file that allows to modify each speed and acceleration in the system, allowing to change the production speed and the output of products produced by the machine. Another important feature of the system is the possibility to change the positions of the stations and even single slots in the stations from the configuration file and the machine will automatically send the shuttles to the new positions. The creation of a new station is something that does not create any problem to the system and it also a very fast and easy process for the programmer, since the stations' code is standardized and one station can be easily copied and then adapted to the new process that needs to be implemented. That is possible because stations are independent one with respect to the other and do not communicate, with the exception of the position in which shuttles are sent. It is important to note that the new stations positions must be thought carefully in order to not interfere with the routings of shuttles and generate deadlocks.

One example of station position that can be changed is the loading one that can be moved on the right as it can be seen in Figure 6.38. In this case, since each movement is thought in order to reach the specific position of the next station, the system will automatically handle the routing of shuttles and the system can start the working cycle without needing any modification in the code, another example is the possibility to move the labeling station one segment on the right.

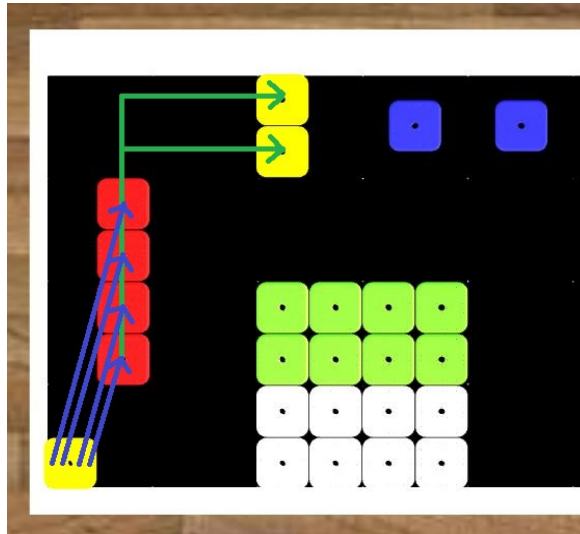


Figure 6.38: New positions for the loading station.

In Figure 6.39 another feature of the system is showed: if the initial position of each shuttle is calibrated correctly, it is able to handle any number of shuttles that can be changed easily by specifying the new value in a mapp 6D's configuration file and modifying the global constant that is defined in the device's logic.

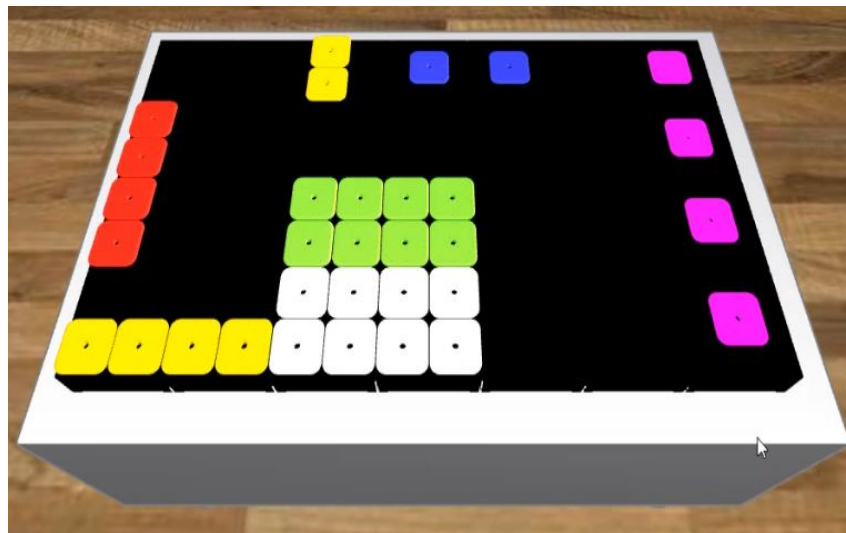


Figure 6.39: System with 32 shuttles.

This system can work with a minimum of eight shuttles because the largest station needs eight of them in order to start the procedures, but it can also work with more than 28 shuttles: the only thing that must be taken into consideration is that, in the phasing procedure, it is necessary to send the newly

define shuttles to a free station with the *MC_BR_MoveInPlaneAsync_Acp6D* function block or, if the number of shuttles is greater than the number of stations, it is sufficient to send with a *MC_BR_MoveInPlane_Acp6D* function block the shuttle to a station that allows queuing (all but the labeling and shaker stations) and the shuttles will wait for the station to become free. The working cycle is not affected by the number of shuttles in the system and will be executed seamlessly.

Conclusions and future developments

The aim of this thesis work was to show the potentialities of an innovative and revolutionary technology applied to industrial automation. The first part of the dissertation is about the magnetic levitation principle and how it has been applied, especially in the industrial automation field, during history. The central part of the document focuses on the B&R Industrial Automation company and on its most relevant products for this application: the developer tool Automation Studio, the Real Time Operating System Automation Runtime, the communication protocol between all B&R's products POWERLINK and the mapp Technology packages utilized in this project. This work is based on a mechatronic device that utilizes magnetism in order to create an efficient, contactless transportation systems so, again in the central part of the thesis, the two other B&R solutions that exploit these technologies are highlighted. The last part of this work is dedicated to the analysis of ACOPOS6D and its various advantages and potentialities, introducing the actual project of this thesis: the realization of an automatic machine application with the device, that is able to implement an automation process with high dynamics, an high concentration of products on the machine and for a long period of time, in a robust way. The major challenge in this thesis project has been to think about an innovative way of programming in order to approach this new technology that presents important differences with respect to traditional automation but also to the other mechatronic solutions developed by B&R. The objective was not only to create a demo able to perform the desired working cycle but also to implement a smart and reusable way of programming that will allow, in the future, to reduce the time to market of future solutions required by customers, making possible an easier commissioning of future applications.

The potential future developments for this paper are multiple and would allow to use in more depth the potentialities of ACOPOS6D. Up to now the stations positions are configurable freely from the configuration file, with the code that is able to reprogram shuttles' routing for some variation of the positions but not all of them. A possible future development could be to implement the stations positions in the recipe file instead of the configuration file and make the code more prone to handle a change in the positions. Up to now the function block in every station is able to handle a change in the stations positions, and the final destination of each shuttle after their processing in the stations is also adaptable to these changes during operation. What would have to be handled is the routing of shuttles, especially when there is more than one movement to be executed in order to move from a station to the next one. At the current state some deadlock could happen so some shuttles' routing should be rethought.

Another upgrade that can be made on the project is the independence of stations: currently the function block signals which shuttle is in each position of a station only if all of them are found in position, forcing the stations to start the processing of shuttles only when the InPosition output pin is TRUE. In general, this behaviour should be implemented for some stations while, for other ones, it could be a great improvement the possibility to recognize shuttles as soon as they are in position, without needing to have all of them in the station. This upgrade would allow to start the processing of shuttles earlier granting an higher production speed. This is a software modification that could be done for the labeling and shaker stations, where a state machine with the only task to recognize shuttles in position could communicate to another state machine the shuttle to process, decoupling the recognition of shuttles from the processing and moving that each station does. This modification would require also to restructure the code of the function block in order to communicate in real time the presence of each shuttle.

At the current state of the project, there is no way to remove products that have not been processed correctly by the machine during the working cycle, as well as a maintenance procedure to hot-swap the shuttles out of the machine while the production is not stopped is not present up to now. These are very important procedures that most automatic machines need in order to hit the market. Besides this, also a procedure to restart the production after a stop without needing the pausing procedure to be executed, could be useful for the realization of the actual machine.

As mentioned before, the machine is able to work with different speeds and accelerations, so once the physical product is defined, it would be very useful to conduct tests about the control stability of shuttles while they are bringing the

product and are subject to its load. Other upgrades that could be implemented are: a less standard HMI that allows to change recipe values from the web browser without Automation Studio, the implementation of other alarms and in general the interconnection of the device with other devices that would actually provide the loading, labeling, capping and unloading of the products, instead of simulating them with a customizable timer.

In general the project has been thought in order to work in the packaging sector but this approach could be validated also in other fields, like the food and beverage or the semiconductors' production , a field in which magnetic levitation can be exploited at its full potential (see Chapter 1.3.1 for more details).

This journey in B&R brought me to know more in depth the automation world because of the wide range of products that B&R disposes of, that focus on every part of different automatic machines and automation sectors. This internship has also allowed me to understand new ways of thinking and programming, like for the station-oriented programming of shuttles and for other tests that have been ran on ACOPOS6D during this experience.

Bibliography

- [1] Hamid Yaghoubi. *The Most Important Maglev Applications*. [Hindawi Publishing Corporation Journal of Engineering Volume 2013]
- [2] Ilene Busch-Vishniac. *Applications of Magnetic Levitation-Based Micro-Automation in Semiconductor Manufacturing*. [IEEE Transactions on Semiconductor Manufacturing · September 1990]
- [3] Masahiro Tsuda, Toshiro Higuchi, Shigeki Fujiwara. *Magnetic Levitation Servo for Flexible Assembly Automation*. [The International Journal of Robotics Research]
- [4] Claudio Melchiorri. *Force Control*. [Industrial Robotics].
- [5] Chang-Wan Ha, Chang-Hyun Kim, Jaewon Lim. *Experimental Verification of a Magnetic Levitation Transport System for the OLED Display Evaporation Process Under Vacuum*. [IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 3, NO. 4, OCTOBER 2018]
- [6] Lei Zhou, Jingjie Wu. *Magnetic Levitation Technology for Precision Motion Systems: A Review and Future Perspectives*. [Department of Mechanical Engineering, The University of Texas at Austin 204 E Dean Keeton Street, Austin, Texas 78712, USA.]
- [7] BR website.
<https://www.br-automation.com/it-it/>
- [8] Automation Studio training manual.
<https://www.br-automation.com/it-it/downloads/automation-academy/training-modules/control-technology/tm210-working-with-automation-studio/>

- [9] Automation Runtime training manual.
<https://www.br-automation.com/it-it/downloads/automation-academy/training-modules/control-technology/tm213-automation-runtime/>
- [10] POWERLINK training manual.
<https://www.br-automation.com/it-it/downloads/automation-academy/training-modules/powerlink-opensafety-and-opc-ua/tm950-powerlink-configuration-and-diagnostics/>
- [11] Motion control training manual.
<https://www.br-automation.com/it-it/downloads/automation-academy/training-modules/motion-control/tm400-introduction-to-motion-control/>
- [12] X20CP3685 data sheet.
<https://www.br-automation.com/it-it/downloads/control-and-io-systems/x20-system/cpus/x20cp3687x/data-sheet-x20cpx68xx/>