UNIVERSIDADE D
COIMBRA

Floyd Wayne Bernard D'Souza

DESENVOLVIMENTO DE UMA SOLUÇÃO QUE PERMITE ADICIONAR UM ROBÔ COLABORATIVO A UM AGV INDUSTRIAL

ASPETOS RELACIONADOS COM O AGV E SOFTWARE DE COMANDO

VOLUME 1

Dissertação no âmbito do Mestrado Integrado em Engenharia Mecânica, na especialização de Produção e Projeto orientada pelo Professor Doutor Joaquim Norberto Pires e pelo Engenheiro Abel Mendes da Active Space Automation e apresentada ao Departamento de Engenharia Mecânica, da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2019

1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Development of a solution that allows integration of a collaborative robot onto an industrial AGV – AGV and GUI related aspects

Submitted in Partial Fulfilment of the Requirements for the Degree of Master in Mechanical Engineering in the speciality of Production and Project

Desenvolvimento de uma solução que permite adicionar um robô colaborativo a um AGV industrial – aspetos relacionados com o AGV e software de comando

Author

Floyd Wayne Bernard D'Souza

Advisors

Professor Doutor Joaquim Norberto Cardoso Pires da Silva

Engenheiro Abel Mendes

Jury

President	**Professor Doutor Miguel Panão** **Professor Auxiliar da Universidade de Coimbra**
Vowels	**Engenheiro Ricardo Patrício** **CEO na empresa Active Space Automation** **Professor Carlos Xavier Pais Viegas** **Investigador Auxiliar da Universidade de Coimbra**
Advisor	**Professor Doutor Joaquim Norberto Cardoso Pires da Silva** **Professor Associado c/ Agregação da Universidade de Coimbra**

Institutional Collaboration

Active Space Automation

Roboplan – Robotics Experts

Coimbra, September, 2019

“The only way to do great work is to love what you do. If you haven’t found it yet, keep looking. Don’t settle.”

Steve Jobs

ACKNOWLEDGEMENTS

The present dissertation represents all the hard work put to conclude the course of Mechanical Engineering during these 5 years. 5 years that went so fast and couldn't have been done without the help from many people, who I'll cherish for the rest of my life and I would like to give thanks to, nevertheless.

Firstly, I would like to give thanks to my professor Dr. Joaquim Norberto Pires for the advices given to aid me in my Thesis and for the classes of Industrial Robotics, with all the tools I needed that helped me overcome the obstacles in this Dissertation.

To the Engineer Abel Mendes from Active Space Automation and to Roboplan, for also helping on this thesis, by providing the AGV and the collaborative robot needed for this work.

To my colleague and friend João Costa, my partner in this project, for your knowledge and advices given throughout this work and without your help, this project wouldn't be possible.

To my other colleague and friend Carlos Ye Zhu, that without his expertise in programming, I wouldn't be able to develop the necessary software needed for this project.

To my friends in FAN-Farra Académica de Coimbra, without you guys and the music we play together, Coimbra would just be another city to me. Now, my greatest memories of Coimbra come from you and no words can describe how much it meant to me.

To all my friends, who helped create many wonderful memories within these past 5 years and who I've had the pleasure to work with.

To Margarida, who without her advices and her friendship, it wouldn't have been possible to complete this dissertation.

To my dad, my mum, my sister and Pooch, the ones that I'll cherish and love the most and the reason I'm here. For the past 23 years they've supported me with unconditional love and they gave me everything in the world, with no charge, obviously. I hope that with this thesis, I can make you proud of me and whatever happens in the future, I know you'll be there for me, once again, with no charge.

Abstract

The Industry 4.0 brought the smart and autonomous systems, creating possible ways to automate processes that would otherwise be unpractical or tedious for any human being. This dissertation will focus on autonomous robots and the internet of things, which are two of the main pillars of this idea that has become a reality in this century. The goals set for this work are to create an interface with the collaborative robot, guarantee the connection with the automated guided vehicle and create a graphical user interface. The reason for this project is simple: to aid workers on their day-to-day, since performing tedious tasks, like repetitive motions, can cause injuries and mistakes that can cost a company.

With the aid of various software like the Microsoft Visual Studio 2019, the Arduino IDE and the software involved with the collaborative robot, the solution was more than possible and made easier to build the electronic part of the interface, where the Arduino mkr1000 served as the brain of the operation, the cobot as the arm and the automated guided vehicle the legs. The collaborative robot interface and the connection with the AGV were established with the mkr1000 microcontroller and the Arduino IDE. The motive for the utilization of the mkr1000 is due to the Wi-Fi compatibility, allowing for remote control via the internet. The graphical user interface was developed with the Microsoft Visual Studio 2019. The AGV is an ActiveOne from Active Space Automation and the cobot is the KR810 from Kassow Robots. This work created one application and a possible solution to the automated pick-and-place operation to help many industries easily integrate into the Factory of the Future with the utilization of a simple microcontroller.

Although the work has performed on all tasks and it is possible to have a mobile manipulator, there can be possible improvements that can help make the solution more human friendly and learn to recognize objects with an incorporated vision system.

Keywords Collaborative Robot, Industrial AGV, IoT, Mobile Cobot, Palletizing, Industry 4.0.

Resumo

A Indústria 4.0 trouxe os sistemas inteligentes e autónomos, criando possíveis maneiras de automatizar processos que seriam pouco práticos ou monótonos para um ser humano. Esta dissertação tem como foco os robôs autónomos e a *internet* das coisas, os pilares principais desta ideia que se tornou numa realidade neste século. Os objetivos deste trabalho são criar uma interface ao robô colaborativo, estabelecer uma ligação com o AGV e criar um *software* de comandos. A razão para a realização deste trabalho é simples: ajudar trabalhadores no seu quotidiano, já que os trabalhos monótonos ou repetitivos podem causar lesões graves e gerar erros que podem prejudicar uma empresa na sua totalidade.

Com a utilização de vários *softwares* como o *Microsoft Visual Studio 2019*, o *Arduino IDE* e o *software* utilizado para controlar o robô colaborativo, a solução era mais que possível e a interface elétrica tornou-se mais acessível, em que um *Arduino* mkr1000 foi o cérebro da operação, o robô colaborativo o braço e o AGV as pernas. A interface do robô manipulador e a ligação com o AGV foram realizadas através de um microcontrolador *Arduino* mkr1000 e com o *software Arduino IDE*. O motivo por se utilizar tal microcontrolador é devido à funcionalidade Wi-Fi que permite controlo remoto via *internet*. O *software* de comandos foi desenvolvido a partir do *Microsoft Visual Studio 2019*. O AGV é um *ActiveOne* da *Active Space Automation* e o robô colaborativo é um KR810 da *Kassow Robots*. Este trabalho gerou uma aplicação e uma possível solução para a automatização da operação de carga e descarga para ajudar indústrias a integrarem com maior facilidade a Fábrica do Futuro, graças a um simples microcontrolador.

Apesar dos objetivos terem sido cumpridos e é possível ter um robô manipulador móvel, é muito provável existirem futuros trabalhos que ajudem a melhorar esta solução, de maneira a que esta seja mais interativa com os humanos e aprender a reconhecer objetos com um sistema de visão instalado.

Palavras-chave: Robô Colaborativo, AGV Industrial, IoT, Cobot Móvel, Paletização, Indústria 4.0.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF SIMBOLS AND ACRONYMS/ ABBREVIATIONS

List of Symbols

CAGR – Compound Annual Growth Rate [%]

H – Height [mm]

L – Length [mm]

W – Width [mm]

Acronyms/Abbreviations

AGV – Automated Guided Vehicle

ASCII – American Standard Code for Information Interchange

CLI – Command-Line Interface

Cobot – Collaborative Robot

GUI – Graphical User Interface

I/O – Input/Output

IFToMM – International Federation for the Promotion of Mechanism and Machine Science

IoT – Internet of Things

IP – Internet Protocol

IPv4 – Internet Protocol version 4

OEM – Original Equipment Manufacturer

OS – Operating System

PLC – Programmable Logic Controller

SMEs – Small and Medium-sized Enterprises

TCP – Transmission Control Protocol

TMR – Transparency Market Research

USB – Universal Serial Bus

VCC – Voltage Common Collector

Wi-Fi – Wireless Fidelity

1. INTRODUCTION

1.1. Dissertation's structure

This dissertation was the culmination of a semester of work to develop a solution to add a collaborative robot onto an automated guided vehicle. It took into consideration the assumptions made for the actual simulation to execute pick-and-place operations and show the results in a real-time model, in this case, in the Active Space Automation company in Taveiro, Coimbra. The following is an overview of what each chapter in this thesis will talk about:

- **Chapter 1**: Motivation for this project, the forecast of this mobile units and the objectives set for this work;
- **Chapter 2**: The state of knowledge;
- **Chapter 3**: The description of each software used for this work;
- **Chapter 4**: The setup for the communication with the collaborative robot;
- **Chapter 5**: The setup for the communication with the Automated Guided Vehicle (AGV);
- **Chapter 6:** The graphical user interface used to communicate with both the cobot and the AGV;
- **Chapter 7**: Conclusions regarding the development of the solution and what can be done in the future to improve this project.

1.2. Motivation

With the help of the Third Industrial Revolution, contributing with the appearance of computers and automation, the idea of an "Industry 4.0" became more than a dream: it became a reality. This term is very complex to explain, but, in short, it is the Third Industrial Revolution with the addition of smart and autonomous systems, including fully digitalized systems and artificial intelligence. That is why "Industry 4.0" (I4.0) is also known as "Smart Factory".

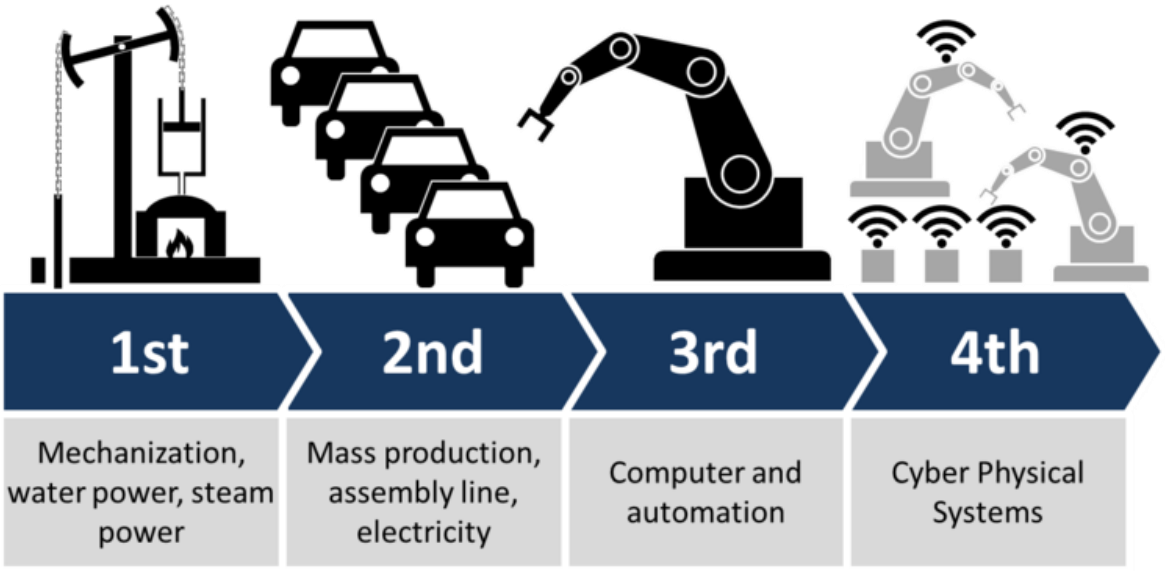


Figure 1.1. – The Industrial Revolutions. **SOURCE**: https://www.nordson.com/en/divisions/dage/about/blog/industry-4-and-its-relevance-to-inspections-solutions.

According to Erboz (2018), the main pillars of the Factory of the Future are:

- Big data;
- Autonomous robots;
- Simulation;
- Horizontal and vertical system integration;
- Internet of Things (IoT);
- Cloud computing;
- Additive manufacturing;
- Augmented Reality;
- Cyber security.

This dissertation will mainly focus on autonomous robots and IoT. Autonomous robots, as the name suggests, are robots that have a certain level of autonomy. Used in manufacturing industries to perform tasks that would either be too difficult or too monotonous for a human being to execute. Although it has autonomy, it needs the knowledge to work in a certain station and that is where the human factor is important. The robots can learn from their human counterparts and start working immediately.

Figure 1.2. – Example of a robot learning from a human user. **SOURCE**: https://www.kassowrobots.com/products/download-center/.

They are very important in areas such as production, logistics, distribution activities and, to establish a more effective human-robot interface, they can be remotely controlled by humans (Erboz, 2018). IoT is a way that humans can communicate with any robot, by utilizing standard protocols through a network (Vaidya *et al*., 2018).

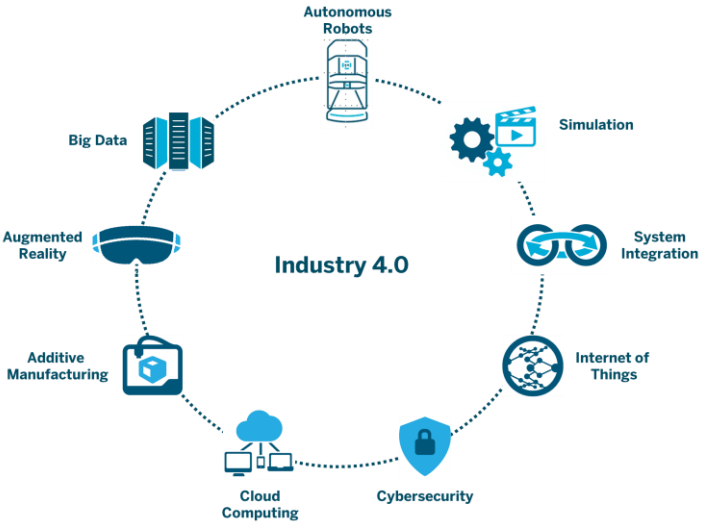


Figure 1.3. – The main pillars of the Industry 4.0. **SOURCE**: https://aethon.com/mobile-robots-and-industry4-0/.

The objective of this work is to find a solution to communicate with a collaborative robot (cobot) that is installed on an automated guided vehicle (AGV), utilizing

a communication protocol provided by Active Space Automation. In other words, create a mobile cobot that does simple pick-and-place tasks.

Mobile cobots or mobile manipulators combine the advantages of a simple cobot with the added mobility. Such advantages are, for example, through teach-by-demonstration it is simple to program a simple robot to perform specific chores; they are safe to work with humans thanks to the use of sensors which can slow down or stop the cobot when it comes across a force too big for it; and because of the dimensions of the cobot compared to an industrial one, they can be mobilized easily and implanted into various production processes. The AGV can navigate through any kind of spaces through radio waves, vision cameras, magnets or lasers.

To conclude, having an AGV and a cobot combined into a mobile manipulating machine can be very beneficial for manufacturing industries and smart factories due to the strong synergy between these two devices.

1.3. The impact on the market

In a report made by TMR, the adoption of mobile cobots by industries has been significant. Specially, in the automotive sector where there is a considerable amount of customization of cars demanded by customers, such as spraying, painting and assembling different parts.

The TMR also reports that the market for this mobile manipulation due to the increased variability of the product is expected to reach a CAGR (Compound Annual Growth Rate) of 33,28% from 2017 to 2026, starting at US$ 336,98 million by the end of 2017. In other words, each year, the market revenue increases exponentially by 33,28% of the revenue from the year before, reaching an astounding US$ 4 472,53 million by the end of 2026.

2017	336.98	Mn US$		**2017**	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**2026**	
2026	4472.53	Mn US$		336.98	449.13	598.60	797.81	1063.32	1417.19	1888.84	2517.44	3355.24	4471.87	[Mn US$]
CAGR =	**33.28%**			**CAGR =**	((end point/start point)^(1/number of years)) - 1					**=**	((2026/2017)^(1/9)) - 1			

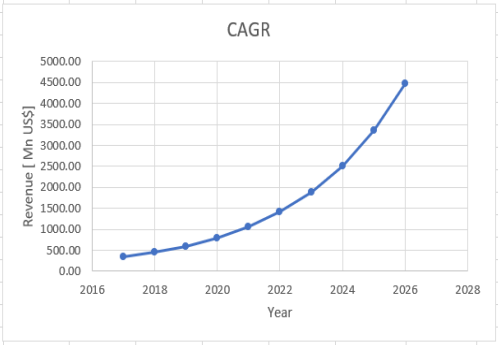


Figure 1.4. – CAGR of the collaborative robot market from 2016 to 2026.

As it can be seen in **Figure 1.4**, it is simple to conclude that there will be an increase in the need of mobile cobots not only in automotive sectors as referred before, but in other manufacturing industries also.

Nowadays, cobots can hold up to 10 kg of payload, within a reach of 1300 mm. TMR also states that several market players are moving forward with the idea of mobile manipulation because it helps achieve maximum return of investment. The most prominent players in this market are ABB, Aubo Robotics, Fanuc Corporation, Techman Robots, Universal Robots and so on.

1.4. Objectives

It is imperative that the following objectives be accomplished:

- Develop an interface for the cobot;
- Guarantee the connection with the AGV;
- Create a graphical user interface (GUI).

Not only this project has to accomplish these specific objectives, but also can be taken into consideration for future works or projects, as the development of autonomous robots is something helpful for industries to get a better return of investment and help smaller industries grow even further to easily integrate into the Factory of the Future. It is important

to keep in mind that this work is part of the final project which is developing a solution to integrate a cobot onto an AGV. The other part was done by Costa *et al.* (2019), where the objectives of the work is to also develop an application for the cobot, collaborate on the interface to the AGV and execute functions like loading, unloading and item manipulation.

2. STATE OF KNOWLEDGE

This chapter will refer to what is known about mobile manipulation and how much it has evolved over the years, mentioning some examples of collaborative robots and automated guided vehicles and possible fusion of these two from two separate companies.

Chebab *et al*. (2015) stated that there are robots that do cooperative or collaborative tasks. The ones that do cooperative tasks imply that they can work together without any human intervention, while the collaborative type involve interactions between human and robots to perform certain tasks. With inspiration from what Chebab *et al*. (2015) wrote, this thesis will focus on mobile robots, manipulation robots and mobile manipulators.

Mobile robots are sorted into various criteria. The first one is the environment where they will be put into and supposedly evolve. They can be aerial, terrestrial, marine or submarine. Then, they are subdivided into their locomotion system. These locomotion systems can be wheeled, tracked or crawling. This thesis will focus more on terrestrial mobile robots. As discussed by Chebab *et al*. (2015), most of the mobile manipulators are equipped with rolling mobile bases, as they are used with the intention of working on regular grounds and because it is more energy efficient this way. The mobile bases can be omnidirectional or not, where omnidirectional implicates that the robot has no constraint on its movement. Finally, to be located on the ground, it is necessary that the robot has 3 degrees of freedom.

A manipulation robot is, by the IFToMM, "a device for gripping and the controlled movement of objects". It usually provides 6 degrees of freedom to complete their task, such as moving heavy objects to guarantee safety for human workers. More than one of these robots can be put in serial, parallel or even a hybrid of both (Chebab *et al*., 2015).

A mobile manipulator is the addition of a manipulation robot to a mobile robot. They are mostly used in areas such as military and security applications, research platforms or in construction. Chebab *et al*. (2015) said that the size of mobile manipulators can differ from different types of environments. For example, large mobile manipulators are usually applied in processing surfaces of aircrafts or ships, while small mobile manipulators are usually used in automatic picking and transporting small parts in assembly lines. In this case,

the study will be focused on small mobile manipulators to perform pick-and-place of objects from one point to another.

The following subchapters will analyse some of the robots utilized nowadays and that have special features that were relevant for this work, indicating some collaborative robots, some mobile robots and a fusion of these two.

2.1. Collaborative robot

The collaborative robots that will be presented below are from models that were considered throughout the semester due to their unique features. One of them is from Kassow Robots: the KR810 with its 7 degrees of freedom and another from Techman Robots: the TM5 700 with the built-in vision system.

2.1.1. Techman Robots – TM5-700

Techman is part of Quanta Computer Inc., the world's largest industry for notebook computer OEM (Original Equipment Manufacturer), as a subsidiary company. What makes the robots from Techman unique is the built-in intelligent visual system. The company is currently the second largest collaborative robot brand since its opening in 2016.

Figure 2.1. – TM5-700 collaborative robot. **SOURCE**: https://tm-robot.com/en/regular-payload/.

According to Techman Robots, it has a payload of 6 kg and a reach of 700 mm. With 6 degrees of freedom, it weighs 22,1 kg and has a repeatability of +/- 0,05 mm. The robot comes with a control box and a robot stick, as shown in Figure 2.2.

Figure 2.2. – Control box and the robot stick of the TM5-700 cobot. **SOURCE:** https://tm-robot.com/en/wpdmdownload/tm5-hardware-installation-manual-hw3-00-rev1-03/.

The control box has 16 digital inputs, 16 digital outputs and the software that is utilized to control this cobot is a software exclusive to the company which is TMflow. This software is like the one in Kassow Robots, a simple GUI with the objective of making it easy to program the cobot to perform tasks.

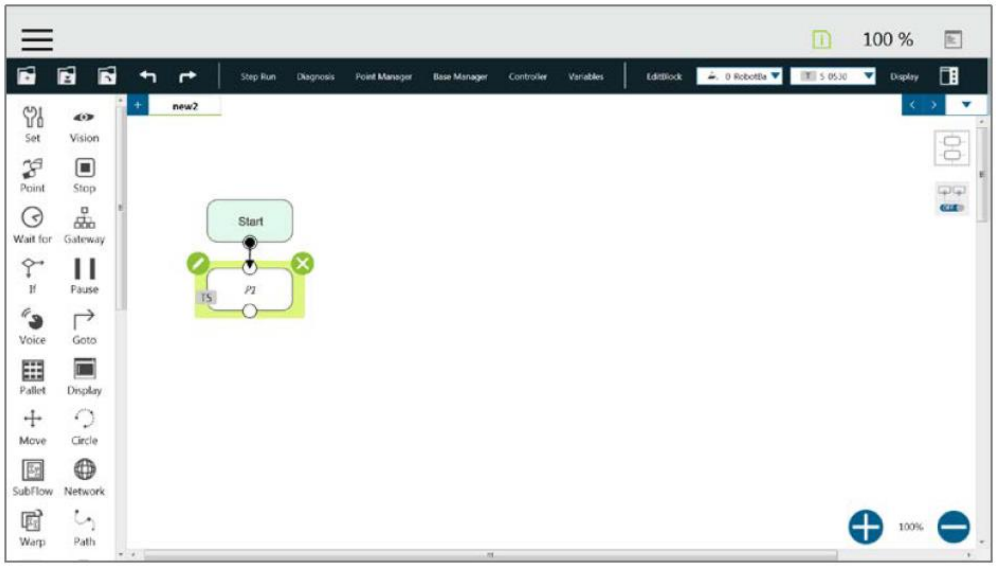


Figure 2.3. – Graphical User Interface of a TM robot. **SOURCE:** https://tm-robot.com/en/wpdmdownload/software-manual-tmflow-sw1-68-rev1-03/.

The built-in vision system is what makes it unique. It has 1,2M pixels and it is a coloured camera, able to recognize landmarks of its own company, controlled by the TMflow application.

Figure 2.4. – TM Vision of the TM5-700 cobot. **SOURCE:** https://tm-robot.com/en/wpdmdownload/software-manual-tmvision-sw1-68-rev1-00-2/.

2.1.2. Kassow Robot – KR810

Founded in Copenhagen, Denmark, Kassow Robots is a company that produces 7-axis lightweight robots with the intention to offer high speed and power, permitting long reaches. Because of the robot's user-friendliness, it allows SMEs to obtain complex automation and programming without ever needing robotics specialists. This cobot is compact enough for confined spaces.

Figure 2.5. – KR810 collaborative robot. **SOURCE:** https://www.coboticsworld.com/portfolio-items/kassow-robots-kr810/.

According to Kassow Robots, this KR810, made from anodized aluminium, has 7 degrees of freedom and it can hold up to 10 kg of payload and has a reach of 850 mm. It comes with a teach pendant that contains the application to control the robot and a portable electrical cabinet. In Chapter 3, which talks about the software used for this project, it will be said in detail how the software of the teach pendant to control the cobot works.

Figure 2.6. – Graphical User Interface of the KR810 collaborative robot. **SOURCE:** https://www.kassowrobots.com/products/download-center/.

Other important characteristics are, for example, its repeatability (the precision of which the robot returns to a programmed position) is +/- 0,1 mm, weighing 23,5 kg. In terms of the controller, it has 8 digital output that go from 0 V to 24 V, 4 relay outputs, 16 digital inputs that vary between 3 V and 30 V. This will be important in Chapter 4.

Figure 2.7. – Control box and the teach pendant of the KR810 cobot. **SOURCE:** https://www.kassowrobots.com/products/benefits/.

This will be the robot used for the current project, as it's a prototype, providing an opportunity to analyse and develop something different for the respective company.

2.2. Automated guided vehicle

The following AGVs have the objective of supporting the items on top of the platform, the cobot and the cobot's control box. One of these AGVs is from SSI Schaefer, the Weasel, which can handle up to 35 kg and another from Active Space Automation, the ActiveOne, that can withstand up to 800 kg.

2.2.1. SSI Schaefer – Weasel

Founded in 1937 by Fritz Schaefer, it's known to provide solutions for logistics system and products, being one of the world's best. Thanks to the purchase of Motum in 2015, a manufacturer of auto-guided transport systems, it was possible to develop AGVs. In this case, the AGV to analyse from SSI Schaefer is the Weasel model.

Figure 2.8. – SSI Schaefer-Weasel AGV. **SOURCE:** https://www.ssi-schaefer.com/pt-pt/produtos/conveying-transport/veículos-automaticamente-guiados-/fahrerloses-transportsystem-weasel-187760.

The dimensions are up to 810 mm × 420 mm × 180 mm (L × W × H), with the capability of supporting weights up to 35 kg, it can travel up to 1000 mm/s. Because of its relatively small dimensions, it can transport goods to inaccessible areas. In other words, a mix of flexibility with versatility that follows an optical guiding line, making it cost effective on installation. In terms of energy supply, it can be done manually or automatically. By

manual means, it is through a charging cabinet. Or it can be automatically charged via ground-contact charging stations. With either energy supplies, it has an autonomy of 16 hours with one battery charge.

2.2.2. Active Space Automation – ActiveOne

Active Space Automation is a spin-off of Active Space Technology with the intention of creating and deploying AGV and robotics for a wide range of applications, specially, high-throughput logistics and intra-logistic operations. One of their products, that was important for this work, is the ActiveOne AGV.

Figure 2.9. – Active Space Automation's ActiveONE. **SOURCE:** https://www.activespaceautomation.com/agv/activeone/.

It is a bi-directional AGV with a lifting capacity of 800 kg, being able to lift to a height of 90 mm with the ability to navigate through a magnetic tape or through natural navigation. Even though its dimensions are 1500 × 500 × 190 mm (L × W × H), the ActiveOne has the capability of manoeuvring in small places like the Weasel. It is capable of communication via Wi-Fi, making it easy to communicate with it and control it via even an application for a phone with internet connection. Able to achieve speeds up to 1500 mm/s, making it faster than the Weasel. It has an opportunity charger option, meaning that it can charge itself via a charging station after it has no task to resolve, according to Kickham (2018).

In the end, the ActiveOne is the AGV used for this thesis, as it is a product of Active Space Automation which is based near the city where the research was done.

2.3. Cobot + AGV

Throughout the years, companies thought of the possibility of making cobots mobile and it was not long until they started existing. Companies like KUKA developed a combination of a collaborative robot with an automated guided vehicle: the fusion of a KMR iiwa and a LBR iiwa that combines the strengths of the robot with those of the mobile, autonomous platform. With this combination of two robots from the same company, it became possible to develop solutions that incorporated this idea, but between companies. In other words, using a mobile robot from one company and a manipulation robot from another. The following is a mobile cobot developed by EVO, using the AGV from EVO and a robot from Universal Robots.

2.3.1. EVO – Cobot 1

EVO was founded in 1960 by Ettore Beretta with the intention of developing and producing first-rate automated solutions. Constantly growing, they offer a wide variety of AGV to offer efficient solutions. The product being analysed is the EVO Cobot 1.

Figure 2.10. – EVO-cobot 1 with a UR10e cobot from Universal Robots. **SOURCE:** https://www.oppent-evo.com/en/prodotto/evo-cobot-1-2/.

With the dimensions of 1400 × 660 × 700 mm (L × W × H), it can withstand a payload of 500 kg and travel up to 1000 mm/s. The AGV has an autonomy of 20 hours and

is omnidirectional. It has a UR10e cobot that can hold up to 10 kg of payload and a reach of 1300 mm, weighing in a 33,5 kg.

It is possible to say that joining two products into one can bring the best out of each of them. That is why this thesis is important, because it adds a solution that could be practical and simple to achieve and help manufacturing industries achieve automation in a matter of time. While the ActiveOne is relatively big and robust with good speed and can be communicated with via wireless communication and the cobot has a good reach, with 7 joints and can be controlled by a teach pendant, it is easy to say that combining these two will bring only advantages for any kind of manufacturing industry.

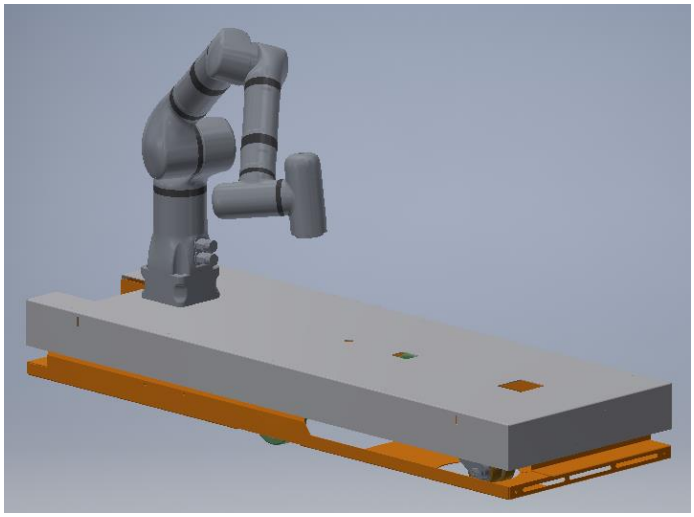


Figure 2.11. – The possible solution of adding a cobot onto an AGV.

3. SOFTWARES

This chapter will essentially give a brief overview of the software used in this project. To achieve full communication between the robots and the user, several were the software used to guarantee the Human-Machine interface. Whether it was in the simulations or in the real-life model, all the software mentioned below had an essential role for this thesis.

Firstly, the computer-aided design (CAD) software Autodesk Inventor Professional 2019 will be mentioned and it had an important function, which was help create the element that connects the gripper with the cobot. Afterwards, the Arduino IDE (Integrated Development Environment) used for both the simulation and the real-life event. Then, to make the communication between the human user and the robots easier, the next software to explain will be the Microsoft Visual Studio C#. Lastly, to teach the robot to do the pick-and-place tasks, the software that is in the teach pendant of the Kassow KR810 will also be explained into detail here.

3.1. Autodesk Inventor Professional 2019

Developed by Autodesk in 1999, it is a Windows' exclusive. Inventor is a CAD software that is used for 3D mechanical design, simulation, visualization and documentation. It aids in the creation of a virtual simulation of how an item would look like, establishing dimensions to the item and validating the form before it is built. It is capable of multi-CAD compatibility. In other words, opening a part that was created via Solidworks can be opened on Inventor and it is able to make a 2D drawing of an individual part or an assembly of parts.

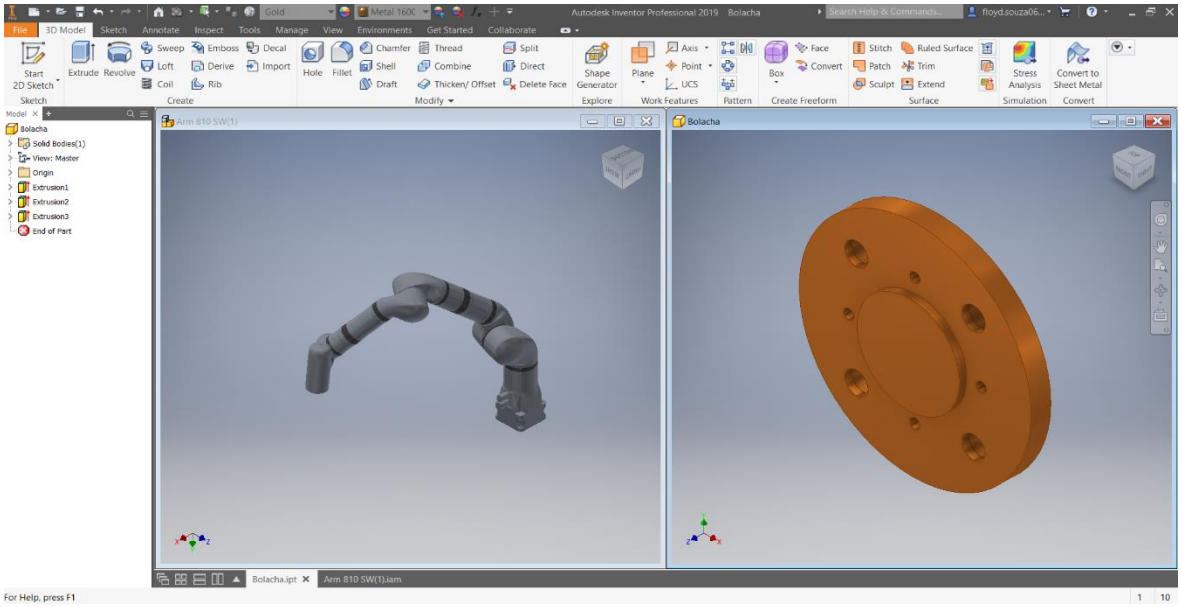


Figure 3.1. – Autodesk Inventor Professional 2019 desktop.

3.2. Arduino IDE

The Arduino IDE is an application that can be used in operating systems like Windows, macOS and Linux. It is utilized to write and upload programs to an Arduino compatible board, like the Arduino Uno, and third-party cores. It has two basic functions: a setup function which only runs once every time a program is uploaded; and a loop function that keeps on running anything inside the loop. It offers the possibility to work through the desktop or through the official site of Arduino, but the advantage of working through the desktop is the possibility to work from anywhere without depending on a network or Wi-Fi.

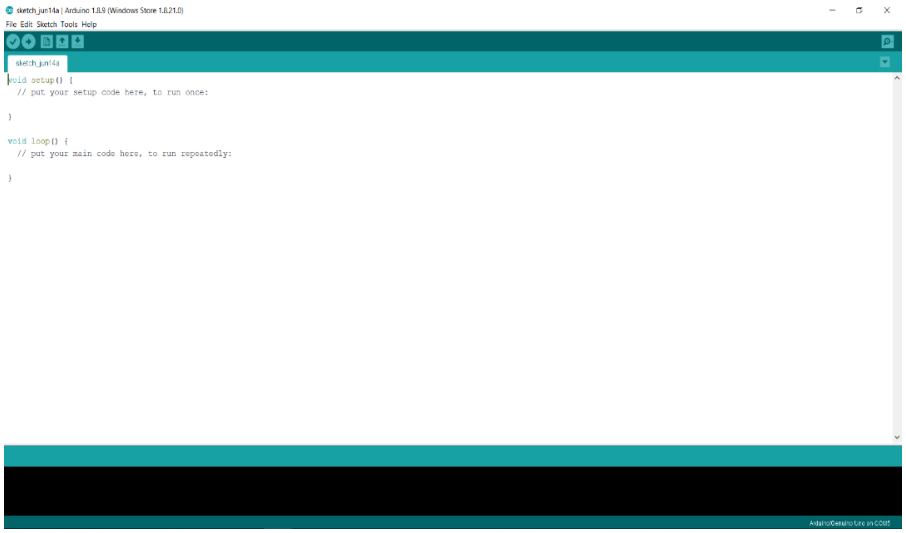


Figure 3.2. – Arduino IDE.

3.3. Microsoft Visual Studio 2019

This software is an integrated development environment created by Microsoft and written in C++ and C#. Used to develop computer programs, it is a Windows' exclusive IDE with an integrated debugger that allows testing of the program for errors before running it, supporting several different programming languages, including C, C++, C#, Visual Basic, etc.

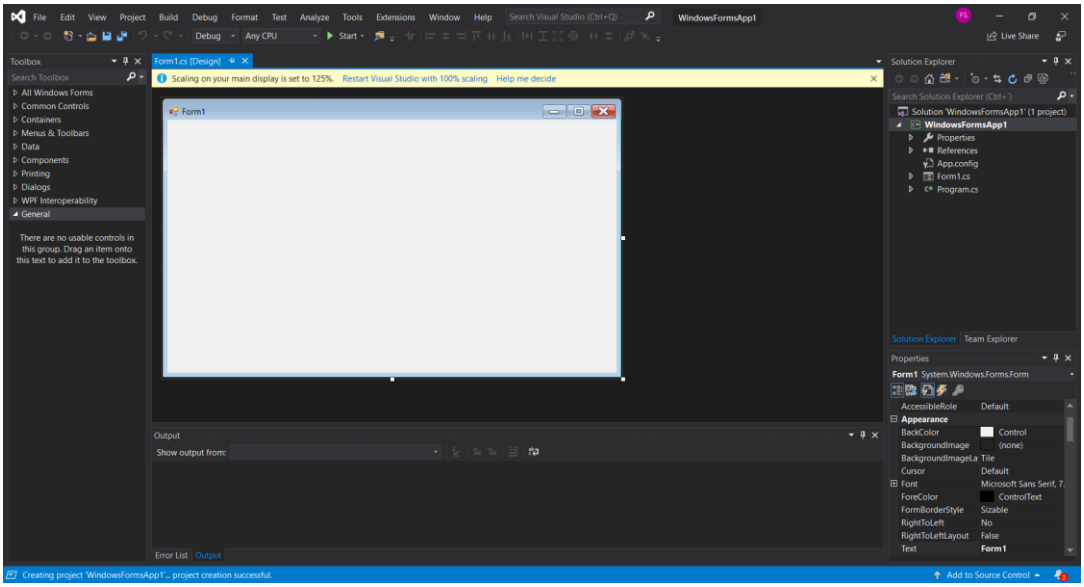


Figure 3.3. – Microsoft Visual Studio 2019 IDE.

3.4. Teach pendant application

A teach pendant, according to PC Magazine Encyclopedia, is a device that allows the user to control the motions of the respective robot. Also known as "teach box", the pendant is used to control the robot step-by-step. They can be handheld devices and can be wired or wireless.

In this work, the teach pendant is basically a touchpad that uses Android as its operating system (OS) and the application used is exclusive to the robot. As the figure shows, it is a graphical user interface where the person that controls the robot issues various commands through the application, using the available functions. It has, by default, 10 different commands: SEQUENCE, IF, LOOP, FOR, MOVE, STOP, RESUME, SET, WAIT and TF:

- **Sequence**: defines the beginning of the execution of the commands;
- **If**: controls the flow of the execution. It can support ELSE IF commands too;
- **Loop**: repeats block of commands until the argument is met;
- **For**: repeats block of commands until a specific number of iterations;
- **Move**: makes robot move to a specific position;
- **Stop**: interrupts the move being executed;
- **Resume**: resumes move that was interrupted by the stop command;
- **Set**: assigns a new value to a variable;
- **Wait**: makes the program go to sleep after a specific time or until a condition is met;
- **TF**: Transforms a pose and saves the new one. It can be a move pose to a new reference, transform a pose or convert pose to a new reference.

It is important to state that this application can only support data variables number, pose and value. The number can be an integer type, float type or Boolean.

Figure 3.4. – Teach pendant's graphical user interface. **SOURCE:** https://www.kassowrobots.com/products/benefits/.

4. COMMUNICATION WITH COBOT

This chapter will talk about how the communication with the cobot was established. To communicate with the robot, a simple microcontroller was used that feeds the manipulation robot commands and, depending on the command, the robot would do the respective task. This microcontroller is an Arduino/Genuino mkr1000 with Wi-Fi functionality.

In the subchapters below, there will be mentioning of this Arduino board and its technical specifications, part of the protocol of the Active Space Automation for the commands to issue and how it was possible to guarantee the communication between these two pieces of hardware, including a schematic with the setup.

4.1. The communication protocols

It is important to state that the following protocol was provided by the company Active Space Automation, based in Taveiro, Coimbra.

The protocol's purpose is to create a possible connection between the ActiveOne and any manipulation robot that needs to be mobilized. According to it, there are two types of interfaces that needs to be done:

- **Electric Interface**: Explained in both this chapter and **Chapter 5 (p. 29)**. Essentially, the AGV must send an electric signal to the robot to say that it is safe for it to start working. Afterwards, the AGV stays still until the cobot has completed its tasks. When it is done, it sends a signal to the AGV stating that it is finished and in a safe position, making the AGV proceed to its next position;
- **Message Interface**: Mentioned in this chapter and in further detail at **Chapter 6 (p. 35)**. Through TCP/IP sockets, in which the client is the AGV and the server is the cobot, the objective is that the client sends a message to the server and the server responds with another message.

The communication protocol can be better understood through **APPENDIX A (p. 51)** which corresponds to the flowchart of the communication protocol between the AGV and the manipulation robot. An example of simulation for this work is a pallet that can hold 9 objects (3 rows and 3 columns) and should be on top of the AGV and on the delivery station. The goal was to command the cobot to pick and place a random object from one location to another specific location.

This was one of the earlier stages of this thesis and there had to be some modifications in the said protocol. So, instead of the AGV being the client, now the Arduino board gives the commands to both the AGV and the cobot. Anatomically speaking, the Arduino board was the "brain" of this operation, the AGV the "legs" and the cobot the "arm". As said before, one of the objectives of this chapter was creating the message interface and it was initially done in the Arduino IDE and then developed in the Microsoft Visual Studio C# IDE, adding the TCP/IP protocol, seen in **Chapter 6**.

4.2. Arduino mkr1000 and the setup

For people that want to do IoT projects at home or anywhere else, the Arduino mkr1000 can be a possible pick for that person. It is a board that allows connection through Wi-Fi and is cost effective for the consumer. It can be programmed via the Arduino IDE (software indicated in **Chapter 3**), but to fully use this microcontroller for this thesis, it was necessary the installation of some libraries, so it could be possible to work through the Arduino desktop IDE. According to the Arduino official website, it required the installation of the SAMD core, by going to the "Boards Manager" in the IDE. And then, to add the Wi-Fi functionality in the Arduino, it was also required to add the library "WiFi101". In terms of technical specifications that were important for this work, the board runs at 3,3 V, meaning that by applying voltages above this value to the digital pins could possibly damage the Arduino board. Other than that, it needs a power supply of 5 V via USB and it has 15 digital I/O pins.

Figure 4.1. – Arduino mkr1000 board. **SOURCE:** https://store.arduino.cc/arduino-mkr1000-with-headers-mounted.

About the setup, the connections made between the Arduino and the cobot were purely electric. Thus, when it sends a command to the robot, it sends electric signals to the robot and starts working when the command is recognized. Meaning that, according to the **APPENDIX B (p. 53)**, the Arduino must first check if the robot is currently busy or not by examining the signal that was sent by the cobot to the board. The only problem with this method is that the KR810 sends a signal from 0 V to 24 V and when the robot sends a 24 V signal to the mkr1000 it can damage the board, as the microcontroller is only capable of withstanding 3,3 V. Therefore, to eliminate this problem, a voltage divider was implemented. So, when a 24 V signal is sent to the board, the signal goes to the Arduino with a signal of 3,3 V, because of the voltage divider.

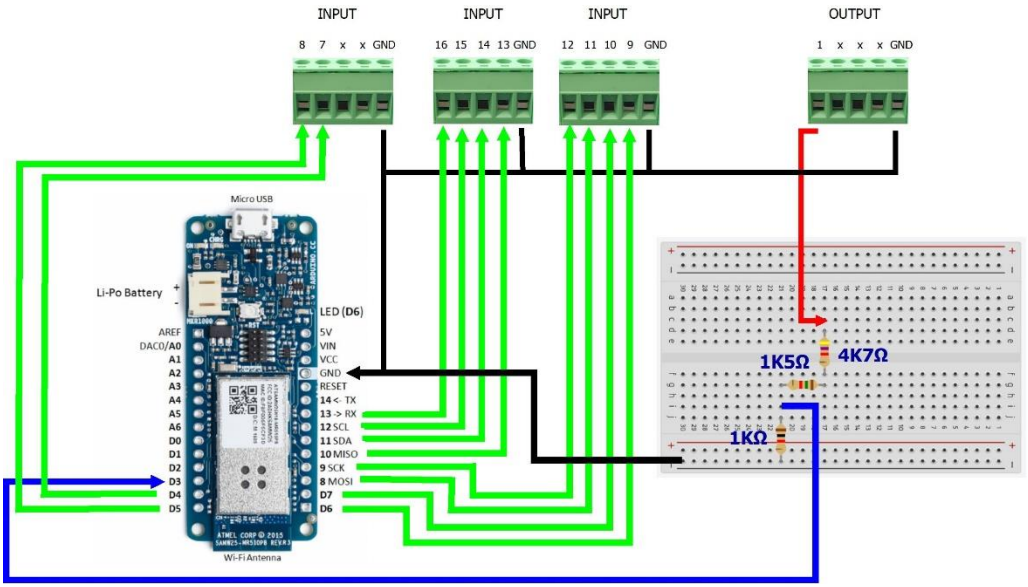


Figure 4.2. – Setup of the cobot's I/O pins with the Arduino's mkr1000 board and the voltage divider.

This subchapter referred to how dangerous it can be receiving a signal from the cobot, considering it sends 24 V of voltage to the mkr1000 and how the setup was made for the Arduino-Cobot communication. The following subchapter is about how the Arduino communicates with the cobot through the electric signal sent by the board and the Arduino IDE.

4.3. The Arduino-Cobot communication

In a first phase, it was decided that the human should issue commands to the robot by introducing a string variable and if the robot could recognize that variable, then it would execute the task that matched the string. The only problem with this approach is that the robot does not currently recognize string variables and to solve this issue, it was considered the use of digital I/O pins from the mkr1000 and the cobot to communicate with each other (as shown in **Figure 4.2**), as the robot is able to recognize numerical Boolean variables (0 corresponding to FALSE and 1 corresponding to TRUE).

Before testing the TCP/IP protocol, the serial communicator of the Arduino IDE was utilized to check if the commands can be read by the microcontroller and afterwards the cobot could execute the task. In the Active Space Automation's protocol, it is stated how the message should look like and what should the answer be to that command.

The format of the message is the following:

- **string_1#string_2#string_3#string_4#string_5#string_6**

Where "string_1" is the command to issue, and from "string_2" to "string_6" are the parameters of the respective command.

The format of the answer is the following:

- **string_1#string_2#string_3**

Where "string_1" is the error code (0 for command accepted, -1 for invalid command, -4 for timeout), "string_2" is the answer to the command and "string_3" is the message to say that the command was received. Also separated by spaces (" ").

Active Space Automation gave an example of how the mobile manipulator should work. The example states that the mobile manipulator should have a pallet mounted

on it and the pallet should contain 9 objects, 3 in a row and 3 in a column, making a 3 by 3 matrix. It also says that the AGV should be told to place a random object onto a determined location. Some of the commands that were considered important were:

- **The cobot going to home position**:
  - **Command**: go_home speed;
  - **Speed**: fast, normal, slow.
- **The robot going to the object's position in the pallet**:
  - **Command**: go_to line column;
  - **Line**: the position in a row;
  - **Column**: the position in a column.
- **Release robot from its duties**:
  - **Command**: end_work;
  - It is important to say that if the robot is not in "home" position, that the command should not be executed.
- **Check robot state**:
  - **Command**: check_robot.
- **Check AGV state:**
  - **Command**: check_agv.

Basically, when a command is given by the user, the Arduino interprets the command and, depending on it, sends an electric signal into the respective digital inputs of the robot. To distinguish the commands, it was necessary the use of the concept of bit. A bit can take the form of a 0 or a 1. 0 meaning that the signal will not be sent and 1 meaning that the signal is sent. For example, when there are 2 bits, 1 bit can take the form of 2 numbers and the other can take the form of 2 numbers too, making it 4 possible combination of bits, when there are 3 bits, it spawns 8 possibilities, and when there are 4 bits, it generates 16 possibilities and so on. In short, the number of possibilities for n bits is 2^n. In this thesis, the command was limited to 4 bits, the first, second and third parameters to 2 bits.

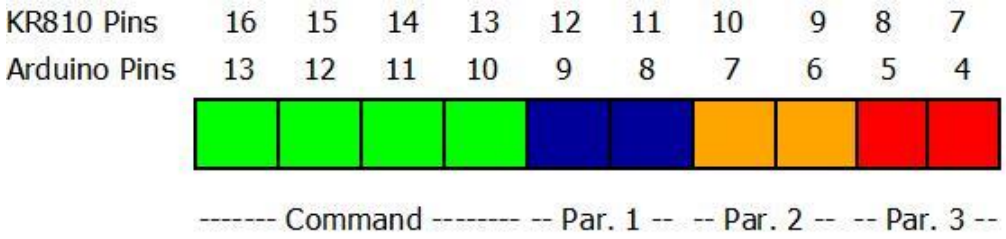


Figure 4.3. – The division of the commands in bits.

Since the commands are going to be sent by bits to the cobot, the following table shows examples of the important commands that will be recognized by the Arduino:

Table 4.1. – Example of commands in bits.

Command	Code (10 bits)	Function
go_to 1 1	0101101000	Goes to position (1,1)
go_to 2 3	0101011100	Goes to position (2,3)
go_home normal	1101010000	Goes home at normal speed
go_home fast	1101100000	Goes home fast
check_robot	xxxxxxxxxx	Checks robot's state

So, when a pin of the board has the number "1", that means that the Arduino will send an electric signal of the highest voltage to the robot's respective pin. After that was done, the cobot must translate the signal into a decimal number, seeing that the cobot can recognize a numerical variable. So, because of this, translating the bits into a decimal number was crucial. The way that this was executed was by summing 2 by the power of n, where n corresponds to the bit position. In other words, $\sum 2^n$, where n is between 0 and 3 for the command's I/O and between 0 and 1 for the 3 parameters' I/O.

For instance, if the command is "go_to", pin 15 and pin 13 are active, which corresponds to the bits 1 and 3, respectively. Thus, the corresponding decimal number of this command to the cobot is equal to the sum of 2^1 with 2^3, which is equal to 10. Afterwards, the cobot must find the command that has the numerical value of 10 and enters in that condition. When the first parameter is the position "1", for the same command for example,

pin 12 is active and the bit corresponding to that pin is bit 0. Therefore, the decimal number to this parameter is 2^0 which is equal to 1. So, when the cobot recognizes the numerical value 1 in the first parameter variable of the cobot, then it will execute the task correspondent to that value.

It is easy to add commands to the Arduino sketch, as the number of possibilities is equal to 16 possibilities, such if the 3 parameters do not exceed the 4 possibilities each. This is an efficient way to communicate with the robot, as it helps organize the commands into a list that can be easily interpreted by either the Arduino or the KR810. In comparison to what the protocol stated, the changes made were done on the electric interface of the project. Basically, everything is communicated by the Arduino and when a user sends a message to the Arduino (message interface) the microcontroller sends (or receives) electric signals to (or from) both the AGV and the collaborative robot (electric interface).

5. COMMUNICATION WITH AGV

The ActiveOne AGV of Active Space Automation is, in its core, a PLC that can be programmed to perform various tasks. According to Bolton (2015), a programmable logic controller is a controller that "uses programmable memory to store instructions" and "implement functions such as logic, sequencing, timing, counting, and arithmetic in order to control machines and processes". Bolton (2015) also states that the reason it's a programmable logic is because the controller uses logic in its processing. For instance, if a certain event happens, then turn on a certain switch. In other words, a PLC is a program that needs inputs and afterwards converts them into outputs. So, inputs like sensors can turn on several output devices (like motors, for example).

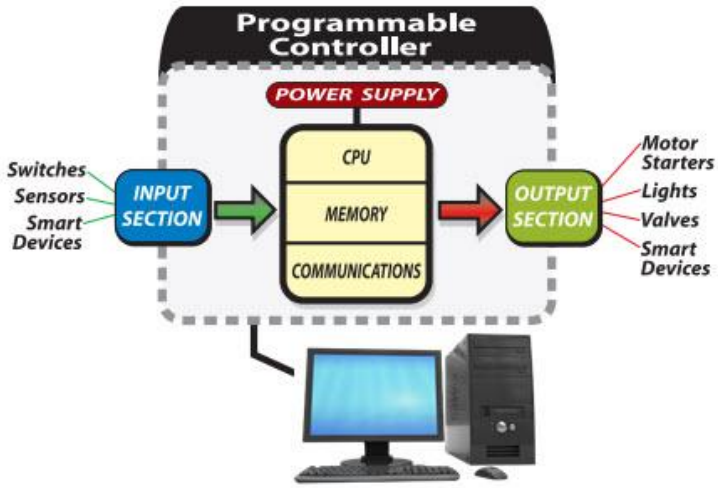


Figure 5.1. – Definition of a PLC. **SOURCE:** https://library.automationdirect.com/what-is-a-plc/.

This chapter will explain how a connection with the AGV was established with the Arduino mkr1000 and how did this help the cobot understand when to start working. It is important to state that the collaborators from Active Space Automation were responsible for the coding of the AGV's PLC.

5.1. The logic of a PLC

As said before, the ActiveOne is basically a PLC, meaning that if a certain event happens, then through logical functions, the PLC will trigger another certain event. So, a PLC is a device that can listen to several inputs and give several outputs.

Much like an Arduino microcontroller, it has various I/O pins that can be used through coding. Therefore, to simulate the behavior of an automated guided vehicle like the ActiveOne, an Arduino Uno microcontroller board was used, because it can receive and send electric signals through its I/O pins just like the AGV. The programming of said Arduino board was based on the information Active Space Automation provided to establish a connection between an AGV and a cobot, but instead of making this connection directly as stated by the communication protocol provided by the same company, it will be the AGV communicating with the mkr1000 and then the mkr1000 giving permission to the cobot to work or not to work.

Throughout the subsequent chapters, the AGV's state in this project will be simulated by the referred Arduino Uno microcontroller and then applied later to the ActiveOne, in the company.

5.2. The setup of the AGV

The setup of the AGV was very similar to the one used between the cobot and the mkr1000. It required a voltage divider for each output of the AGV to safely use with the Arduino microcontroller, as the AGV sends a high signal of 24V and a low signal of 0V. And the connection of the input signal of the AGV (signal from the Arduino board to the AGV) was connected directly, because the 3,3V signal doesn't harm the AGV's input. As said before in the introduction to this chapter, it was considered throughout the extent of this work that an Arduino Uno microcontroller board would take the place of the AGV. The Uno sends a high signal of 5V through its output pins. Because of safety precautions and to get a better approximation of the reality, another voltage divider was used on the Uno board, just like between the KR810 and the mkr1000, converting its 5V signal to a 3,3V one. Obviously, when used with the AGV instead of the Uno, a different voltage divider must be used so that when a 24V signal goes to the Arduino mkr1000, the voltage divider can split the voltage and send 3,3V instead.

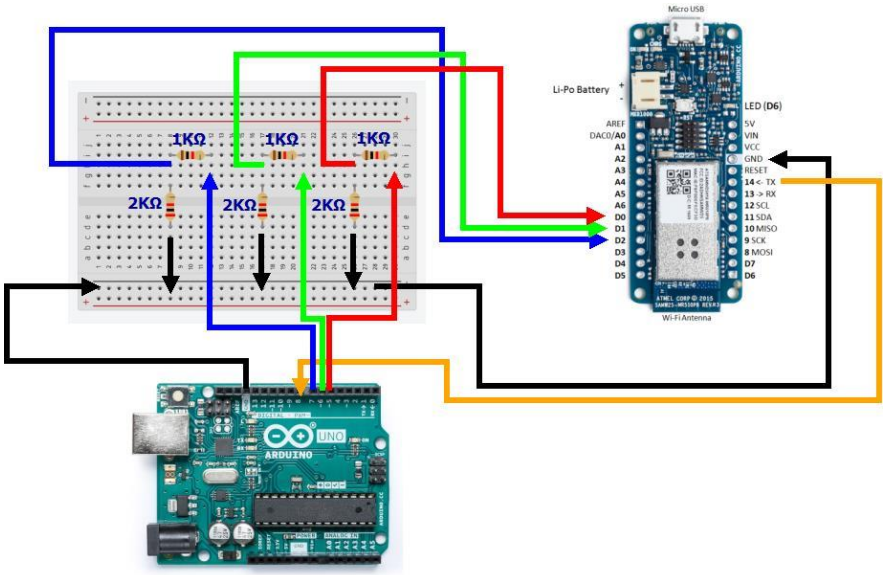


Figure 5.2. – The setup of the AGV (Arduino Uno microcontroller) with the mkr1000.

This figure shows the setup of the AGV (Arduino Uno) with the mkr1000 and the pinout here was used, considering how many digital pins could be spared after setting up the cobot with the same board. Basically, pins D0, D1 and D2 were the outputs of the AGV while pin 14 was the input of the AGV, where the mkr1000 sends a signal to the ActiveOne. When the Uno arrived, supposedly, at a position, a message is sent to the Uno microcontroller and then is recognized by the same board. When it found the adequate answer, it turns on the respective output signals to the mkr1000, as it will be explained in the next subchapter.

5.3. The Arduino-AGV communication

The communication protocol for this part of the process, shown in **APPENDIX C (p. 55)** was relatively simple. It was settled with Active Space Automation that the AGV would receive from the mkr1000 one input signal and output 3 signals to the same microcontroller. The AGV was programmed to reach 3 locations: The home position, the picking position and the placing (or dropping) position, always in this sequence. Depending on the place it is on, each output of the AGV triggers the correspondent input pins of the mkr1000 and by interpreting those signals either send or not a signal to the input of the AGV.

To demonstrate, when the AGV is in picking position, it sends signal to the mkr1000 and, because it is in this position, the microcontroller sends a signal to the ActiveOne, ordering it to stay still while the cobot is busy completing tasks. When the cobot is done, the Arduino board later sends a signal to the input of the AGV to say that it can move to the next point which is, in this case, the "Dropping" position. The following table represents the signals that can be sent to the mkr1000 from the AGV's point of view to signal the position of the mobile robot:

Table 5.1. – The communication protocol between AGV and Robot.

Position	Output0	Output1	Output2	Input0
AGV's Home	Off	On	On	Off
Picking	On	Off	On	On
Dropping	On	On	On	On
Transition	Off	Off	Off	Off

In a step-by-step process, the AGV initially stays in Home position and only starts to work when a human user pushes a button on the AGV. When that happens, the AGV sends signal that it is in transit to the mkr1000. When the AGV is at the first position, which is the picking, it sends a signal to the I/O pins of the microcontroller, as shown in the table above, giving total mobility to the cobot to perform its tasks by turning the Input0 ON. After that is done, the mkr1000 disables the input signal (turns Input0 from ON to OFF), which the ActiveOne recognizes and, consequentially, disables all the output signals, making the AGV move to the next point. After it reaches the dropping position, it sends, once again, a completely different signal to the Arduino board saying that it has arrived and once again the mkr1000 turns the input signal on, permitting the collaborative robot to move. After it is done, the microcontroller disables the input signal, leaving it to the AGV to understand that it can move again towards home position. The state of the AGV can be seen by using the command "check_agv", referred in **Chapter 4**. And this cycle keeps on going.

Utilizing this solution brings this work much closer to a possible working mobile robot with an embarked collaborative and item manipulating robot. The only subject left to discuss is the graphical user interface used which has a TCP/IP socket. So that way, the

communication could be done via Wi-Fi and the parameters to define in the interface are the IP address of each component and their respective port. The next chapter will be focused on the GUI and it will take into consideration all that has been explained in the previous chapters.

6. GRAPHICAL USER-INTERFACE (GUI)

Any kind of robot, in order to perform its tasks, needs to learn how to do them and the human's objective is to guide it throughout its chores. But for a more user-friendly communication, the use of software to achieve such thing can considerably make an impact. That is why this chapter is very important, as it explains how to help human and machine work together to create a powerful synergy for any industry. A GUI is a computer application that allows a person to communicate with a computer via graphical elements. Besides the GUI, another option to talk with computers is the command-line interface (CLI) which allows the user to talk with an electronic device using commands, but due to the steep learning curve of the CLI and the flexibility of the GUI, it was easier to use a GUI.

This chapter will explain how the connection of the AGV and the cobot was established with the command software through TCP/IP and how it was developed for this dissertation, considering the devices that communicate with it, plus the features put into the software that help make this interface much more user-friendly. The application was created with the aid of the software Microsoft Visual Studio 2019, with C# as the default programming language.

6.1. The Human-Machine connection

TCP/IP protocol, according to Kessler (2019), is a protocol used by the host to communicate with any server through a network. It is responsible to guarantee that the message sent by the client gets to the respective destination, thanks to an IP address. While the TCP divides the message into packets and those are transmitted to the network and received by the server, the IP part of the TCP/IP protocol makes sure that the packets are sent to the correct destination in the respective network where the server and the host are connected to at the same time.

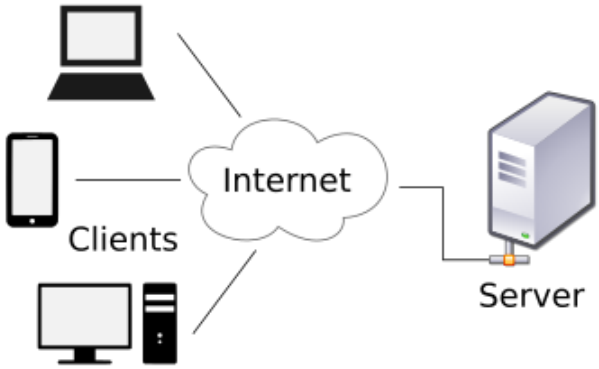


Figure 6.1. – TCP/IP schematic. **SOURCE:** https://en.wikipedia.org/wiki/Client%E2%80%93server_model.

The device that was essential to communicate with was the mkr1000. So, the microcontroller must be capable of establishing a connection with the said application. The communication was made through a TCP/IP socket and that was established by first adding the adequate libraries, like ***System.IO***, which consists of IO related classes, structures, etc., ***System.Net***, which provides many of the protocols used on networks today, and ***System.Net.Sockets***, that developers can use to tightly control access to the network.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Net.Sockets;
```

Figure 6.2. – Libraries to add with the application.

The next step for this connection was defining the type of socket to open, specifying the ***Address Family***, in this case, IPv4, the ***Socket Type***, which creates a TCP connection and the ***Protocol Type***, in this case is an Internet Protocol.

```
//defines the type of socket to open, namely, family, type and data protocol
client_robot_socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
```

Figure 6.3. – Definition of the socket for TCP/IP.

Next, the application must generate a proper IP address structure and serialize the remote port of the device, converting a string representing the IP and the port into a 32-bit signed integer to begin transmitting the orders and that is basically how the app can communicate with the Arduino mkr1000.

```
//fetch the IP of the robot
if (select_kassow_robot.Checked == true)
msg = txt_IP1.Text;
else
   {
    robot_connected = false;
    return;
}

//serializes the IP of the robot and generates a proper IPAdress structure
IPAddress robot_remote_IP = IPAddress.Parse(msg);

//serializes the remote port, making it possible to be transmited
if (select_kassow_robot.Checked == true)
    port = int.Parse(txt_Port1.Text);
else
{
    robot_connected = false;
    return;
}
```

Figure 6.4. – Conversion of the IP address and remote port into a comprehensible structure for application.

Afterwards, it was required to create the bridge that connects the application with the mkr1000. That was done by creating an IP endpoint that needs the IP address and the remote port created in the previous step. And finally connect to that IP endpoint, using the defined structures.

```
//creates a proper IPEndpoint structure required by socket_connect
IPEndPoint IP_Endpoint_remote = new IPEndPoint(robot_remote_IP, port);

//try to connect to robot using the defined structures
client_robot_socket.Connect(IP_Endpoint_remote);
if (client_robot_socket.Connected == true)
    robot_connected = true;
else robot_connected = false;
```

Figure 6.5. – Creation of the IP endpoint and the attempt to connect.

6.2. The Human-Machine interface

The interface between human and machine can be compared to a person writing a letter to another person. There is a mean of communication, which is the letter, a message to send and an answer to that message through the same mean of communication. Same as between a human and a machine, where there must be a mean to send a message and obtain an answer between these two parts. In this project, the use of the Microsoft Visual Studio 2019 helped make such interface possible, using buttons and other graphical functions. In this subchapter, the main topics are: how machines communicate with each other and the features that are in the actual interface created for this project.

6.2.1. Robots' mean of communication

For the application to send the proper commands to the Arduino mkr1000, the app must first convert the command into a proper string so that when it reaches the Arduino board, it can recognize the command and send the respectful output to either the cobot or the AGV. The board then sends a string to the application with a reply that the command was executed or not. So, there must be a common language that both the microcontroller and the app must understand to help the human observe if the job is done. That said language is the ASCII (7-bit) character set. The 7-bit ASCII provides 128 possible characters through binary code (0s and 1s) that represent texts and other type of characters in computers and other electronic devices.

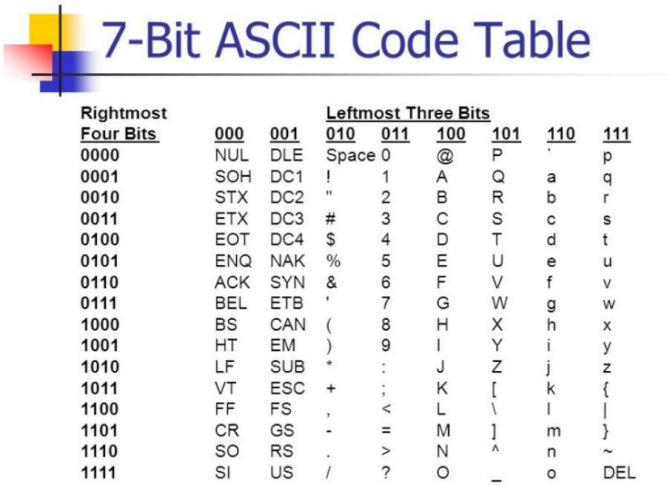


Figure 6.6. – 7-bit ASCII table. **SOURCE:** https://regilanj.wordpress.com/2017/12/04/computer-computations/.

First, the application verifies the connection. When the board is connected, the user can send commands to the board via the app. Before a command can be sent, it goes through a conversion process, encoding the message with a proper ASCII table and then is sent to the server (the Arduino mkr1000 microcontroller). After the server gets the message, it sends the answer to the application and the app reads the answer, writing it on a text window readable to the human. And, finally, the application closes the socket connection and releases all associated resources.

```
//fetch message to send frm the input window
msg = kassow_msg_sent.Text;

//serializes and encodes using a proper ASCII table
sent_data = System.Text.ASCIIEncoding.ASCII.GetBytes(msg);

//send message over open socket
client_robot_socket.Send(sent_data);

//read the answer
number_of_received_bytes = client_robot_socket.Receive(received_data);

//get data and make it writable on a text windows
msg = System.Text.Encoding.ASCII.GetString(received_data, 0, number_of_received_bytes);

//write message on text window
kassow_msg_recv.Text = msg;

//close the socket
client_robot_socket.Close();
```

Figure 6.7. – The process of communication between client and server.

6.2.2. The main features

The main features of the application are the ones that execute the main tasks of the pick-and-place process, like going to a certain position on the AGV or on the picking/dropping station, opening the robotic hand and closing it, making the cobot go to home position and checking the status either of the mobile robot or the collaborative robot.

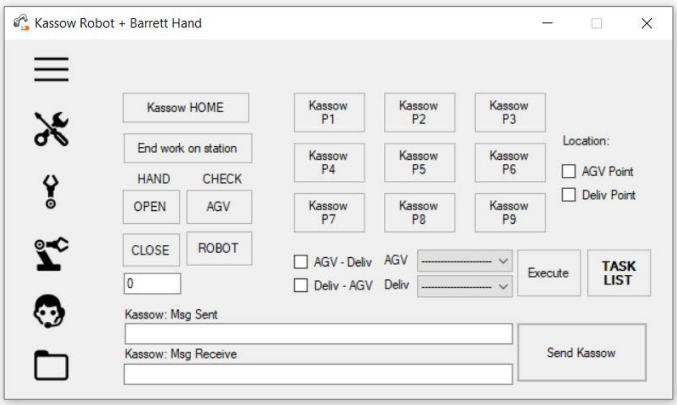


Figure 6.8. –The main interface of the communication with the AGV and the cobot.

With the application referred in the above figure, the human user can:

- Command the cobot to go to Home position (defined by the human user as well as through the teach pendant);
- End work on a specific station, permitting the AGV to move;
- Open and close the hand that is installed to the cobot;
- Check the status of both AGV and cobot;
- Make the cobot move to a specific position;
- Request the manipulation robot to do a specific sequence of pick-and-place;
- Create a task list to make furthermore pick and place sequences.

This part of the app is very intuitive and reliable for any person with or without programming knowledge. In general, all these commands are sent to the Arduino mkr1000 and executed, according to what was explained in **6.2.1**. The tab presented on **Figure 6.8** simulates how a robot will look like when it reaches a certain position, resulting as a sort of a simulation tab to check if the cobot can do the position right and, if in need of adjusting use the teach pendant of the cobot to do so. But when it comes to picking and placing objects automatically in any industry, it was necessary the use of a task list so that a person can pick objects from the picking position and automatically deliver those same objects to the dropping position.

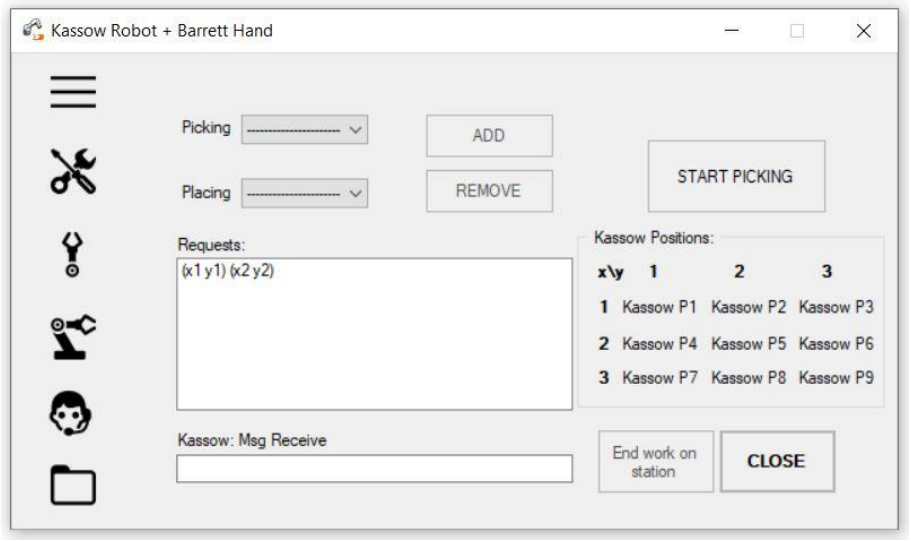


Figure 6.9. – The task list for pick and place chores.

As for the task list, as shown in **Figure 6.9**,when the AGV reaches the picking position, the human user can select the desired positions of picking and placing, adding as many requests (up to one thousand requests) and when the person is done he or she can start the picking process and the last position (x2 and y2) of each request is stored for use in the delivery position. Then, a message box appears asking if the person wants to execute more requests. If the response is no, then the cobot goes automatically to home position and the user can end the work on the respective station and move to the next one, which is the dropping position. In this project, it was assumed that when in the first position of the cycle, x2 and y2 are the default coordinates for x1 and y1 and x2 and y2 of the dropping position (the second position of the cycle). Meaning that the placing position of the first point the AGV must go to will be the picking and placing positions when the AGV reaches the second point of the cycle. After executing the said requests, the person has the option of continuing the dropping tasks or terminate, making the cobot return to home position. Afterwards, when the work at the station is done, the AGV moves to home position and the cycle repeats itself as many times the user wants.

Besides the main features of this software, there are additional ones that complement the main tasks and make the interface more human-friendly for anyone that has or not experience in the field of programming.

6.2.3. Additional features

The additional features put into the software are something that can be interesting to use to further develop a more responsive interaction between the robot and the human user. Nowadays, using graphical function like buttons and check boxes, is something very common between programmers to develop the referred interaction between human and machine. But, to make it more interactive, more modern approaches are used like voice commands, as it is done with, for instance, the Siri of the Apple OS. In this case, a tab was created to add this feature where the user issues a command and the robot recognizes such command and talks to the human.

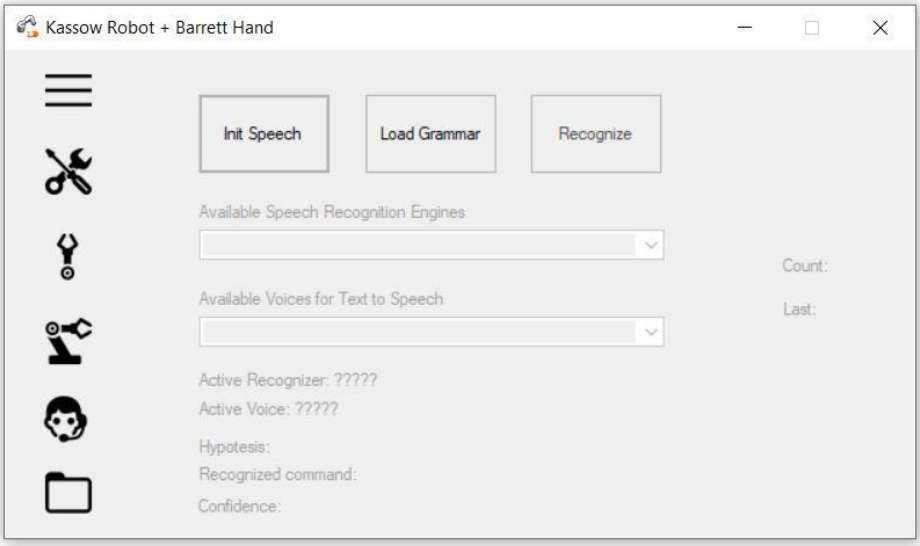


Figure 6.10. – Voice command tab.

The person simply must add the adequate grammar so that the robot can recognize the commands and only then can initiate speech with the interface, supporting languages like English and Portuguese.

Another feature is checking the settings for the gripper. In this case, a Barrett Hand, model BH8-282, was used to manipulate the items for the cobot.

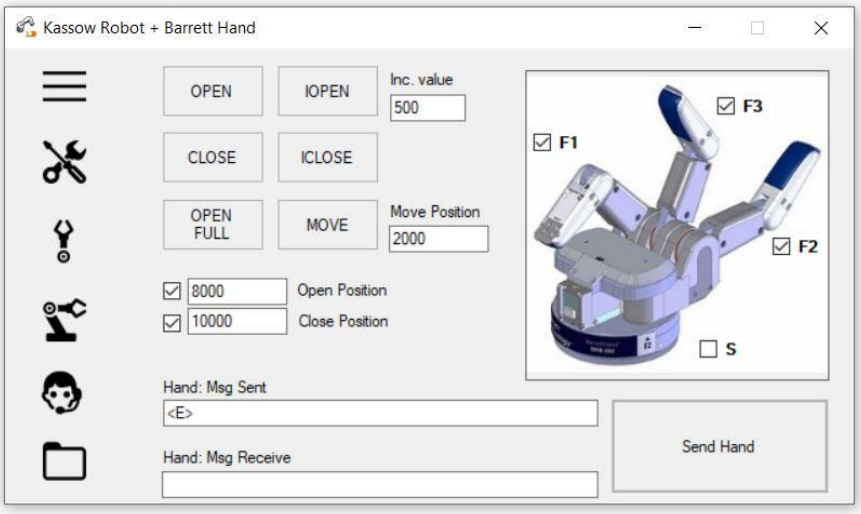


Figure 6.11. – Settings for the Barrett hand tab.

In this tab, it is possible to define the opening and closing positions, move the fingers of the hand to certain position, open and close in increments of a certain value and move either all three or two or one finger of the said Barrett hand.

The additional features, as explained before, complement the software and help develop a more communicative robot that can respond not only through text, but by giving it a voice so it can talk to the user when it has completed a task or is in a certain position.

7. FINAL DISCUSSION

The present dissertation spawned a solution for the mobile manipulator robot dilemma and an application that allows full human control over the same robot. The tasks set for this work have been completed and the human user can work easily with the said solution with an intuitive app and easy-to-control robot.

The **APPENDIX D (p. 57)** shows the total setup of the electronic interface made in the actual simulation in the Mechanical Engineering Department of the Coimbra University. When it came to the use of the GUI, with the TCP/IP socket, the results were positive, where a message is sent to the Uno to say that when it reaches a position, it sends signals to the mkr1000 stating that it is in that position. When the cobot is done executing tasks, the user sends a command that states that it is done on that station and the cobot goes to home position and the user has the option to end work on station enabled. When the user issues the "end_work" command, the mkr1000 turns off the input of the AGV, making it move to the next station. After performing the simulation successfully, it was time to experiment with the real-time model in the Active Space Automation facility with a working AGV and a software that sends a message to the microcontroller that it has reached a specific place.

In the Active Space Automation, a track was created for this work, where its first position is the dropping station, then the home position and then the picking place. After it has left the picking position, it inverts its trajectory, reversing to dropping position. When the AGV reaches home after leaving the dropping position, it waits two seconds and then unlocks its brakes to move to the picking place, giving it the sense of moving automatically or start from a remote position. When moving in reverse, the AGV doesn't stop at home position, but skips it to move to the dropping point. Everything about the setup of the track is on **APPENDIX E (p. 59)**. About the electric setup, for the AGV to stop when it reaches either picking or dropping position, it requires a 24V signal to its input and to achieve that, instead of the mkr1000 supplying voltage to the AGV, the cobot now outputs the signal to the AGV's input, directly. But to control this signal, another input of the cobot was used to communicate with the mkr1000. So, when the mkr1000 recognizes a position by reading the

outputs of the AGV, it sends a signal to the input of the cobot and when this is on, the cobot then sends a 24V signal to the AGV directly. And for the outputs of the mobile robot, instead of using voltage dividers, relays were used in its place and when a relay activates, it outsources 3,3V to the input pins of the microcontroller. Such source came from the VCC pin of the mkr1000.

With these changes, the mobile manipulator robot was successful in performing the pick-and-place tasks, as shown in both videos from YouTube by D'Souza *et. al* (2019), accomplishing the objectives that were established for this dissertation. An article was written regarding this final project by Costa *et al.* (2019).

8. CONCLUSION

The solution to embark a robot manipulator on a mobile robot involved the use of a microcontroller that can connect to the Wi-Fi and establish the concept of the Internet of Things. This way, the human user can remote control both robots. With the created application and a proper protocol to communicate with the electronic device, it was a matter of time until everything would be operational for the simulation and the real-time event. All of this to obtain a solution to automate pick-and-place tasks and contribute to the Industry 4.0 reality that defines itself with autonomous systems.

Although all objectives were accomplished throughout this semester, there were still opportunities to improve and add features to the solution, but with the limited time that was provided, there was no chance of executing such improvements and adding the new features. In terms of improvements, a different gripper could have been used, instead of the Barrett Hand, since the size of its fingers are excessive for lifting smaller objects, for instance, and the amount of force it uses could damage an object. The amount of equipment that is on top of the AGV can be reduced to decrease the total weight on top of the AGV. For example, the computer can be replaced with a simple and smaller one like a Raspberry Pi single board computer. Regarding added features, another sense that can be added to the cobot should be the sense of sight, recognizing objects from their shapes and open or close the hand in different positions, according to that shape of the object.

BIBLIOGRAPHY

Analysis, T. M. R. (n.d.). Global Mobile Cobots Market - Snapshot. Accessed May 13, 2019, from https://www.transparencymarketresearch.com/mobile-cobots-market.html

Arduino. (n.d.). Arduino - MKR1000. Accessed June 24, 2019, from https://www.arduino.cc/en/Guide/MKR1000

Arduino. (n.d.). Arduino MKR1000 WiFi. Acessed June 15, 2019, from https://store.arduino.cc/arduino-mkr1000

Automation, A. S. (n.d.). ActiveONE - Active Space Automation. Accessed May 20, 2019, from https://www.activespaceautomation.com/agv/activeone/

Bolton, W. (2015). *Programmable Logic Controllers*. Retrieved from https://books.google.pt/books?id=sDqnBQAAQBAJ&lpg=PP1&dq=plc%20programmable%20logic%20principle&lr&pg=PP1#v=onepage&q=plc%20programmable%20logic%20principle&f=false

Chebab, Z.-E., Fauroux, J.-C., Bouton, N., Mezouar, Y., & Sabourin, L. (2015). Autonomous Collaborative Mobile Manipulators : State of the Art. *TrC-IFToMM Symposium on Theory of Machines and Mechanisms , Izmir, Turkey*. Accessed from https://www.researchgate.net/publication/330600823_Autonomous_Collaborative_Mobile_Manipulators_State_of_the_Art

Costa, J., D'Souza, F., & Pires, J.N. (2019). Development of a mobile manipulator system for order picking tasks. Industrial Robot. Emerald Publishing, London.

D'Souza, F., Costa, J., & Pires, J. N. (2019). Kassow Cobot & ActiveSpace AGV1 - Demo APP. Accessed September 2, 2019, from https://www.youtube.com/watch?v=o1zTrdTdqPQ%0D

D'Souza, F., Costa, J., & Pires, J. N. (2019). Demo APP 2 - Kassow Cobot & ActiveSpace AGV1. Accessed September 6, 2019 from https://www.youtube.com/watch?v=dJdVCu7W46M&fbclid=IwAR2L41UugBTsfMw4G9LDoMPzGlY2TMz06QhhiIBvJt5v7lmnDCMXapf49LE

Erboz, G. (2018). *HOW TO DEFINE INDUSTRY 4 . 0 : The Main Pillars of Industry 4 . 0*. (July). Accessed from https://www.researchgate.net/publication/326557388_How_To_Define_Industry_40_Main_Pillars_Of_Industry_40

EVO. (n.d.). AGV Robot | EVO cobot 1. Accessed June 18, 2019, from https://www.oppent-evo.com/en/prodotto/evo-cobot-1-2/

Kessler, G. C. (2019). An Overview of TCP/IP Protocols and the Internet. Accessed June 26, 2019, from https://www.garykessler.net/library/tcpip.html#what

Kickham, V. (2018). Start me up: Opportunity charging or fast charging? - DC Velocity. Accessed June 22, 2019, from https://www.dcvelocity.com/articles/20180803-start-me-up--opportunity-charging-or-fast-charging-/

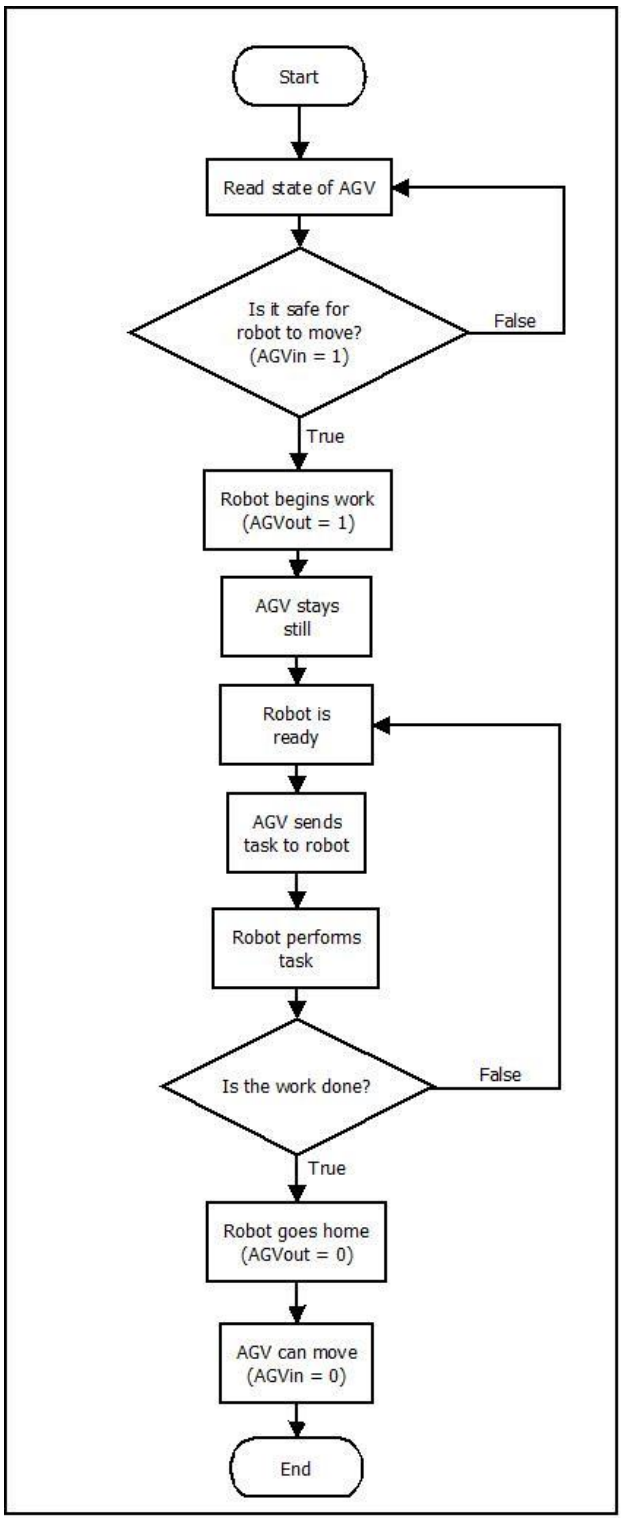
Robotics, S. (n.d.). About Repeatability. Accessed June 19, 2019, from http://strobotics.com/repeat.htm

Robots, K. (n.d.). *Kassow Robots: Controller manual*.

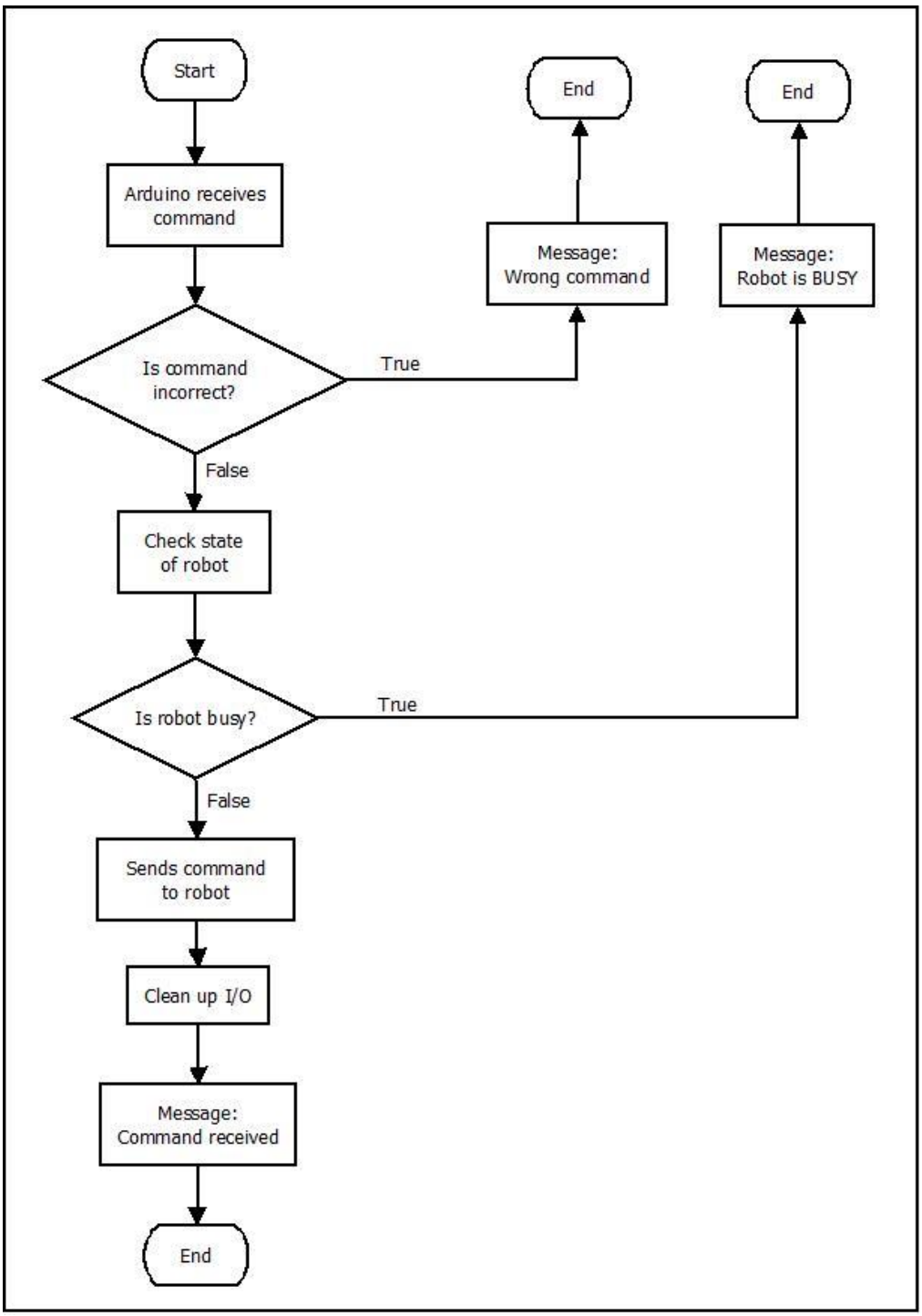
Robots, K. (n.d.). *Kassow Robots: Instruction Manual*.

Robots, K. (n.d.). *Kassow Robots: Software Manual*.
Robots, K. (n.d.). KR810 Cobot - Kassow Robots. Accessed May 14, 2019, from https://www.kassowrobots.com/products/kr810/
Robots, U. (n.d.). UR10 Collaborative industrial robotic arm - Payload up to 10 kg. Accessed June 22, 2019, from https://www.universal-robots.com/products/ur10-robot/
Schaeffer, S. *WEASEL ® AUTOMATED GUIDED VEHICLE*. Accessed from https://www.ssi-schaefer.com/en-de/products/conveying-transport/automated-guided-vehicles/fahrerloses-transportsystem-weasel-1918
Vaidya, S., Ambad, P., & Bhosle, S. (2018). Industry 4 . 0 – A Glimpse. *Procedia Manufacturing*, *20*, 233–238. Accessed from: https://www.sciencedirect.com/science/article/pii/S2351978918300672?via%3Dihub

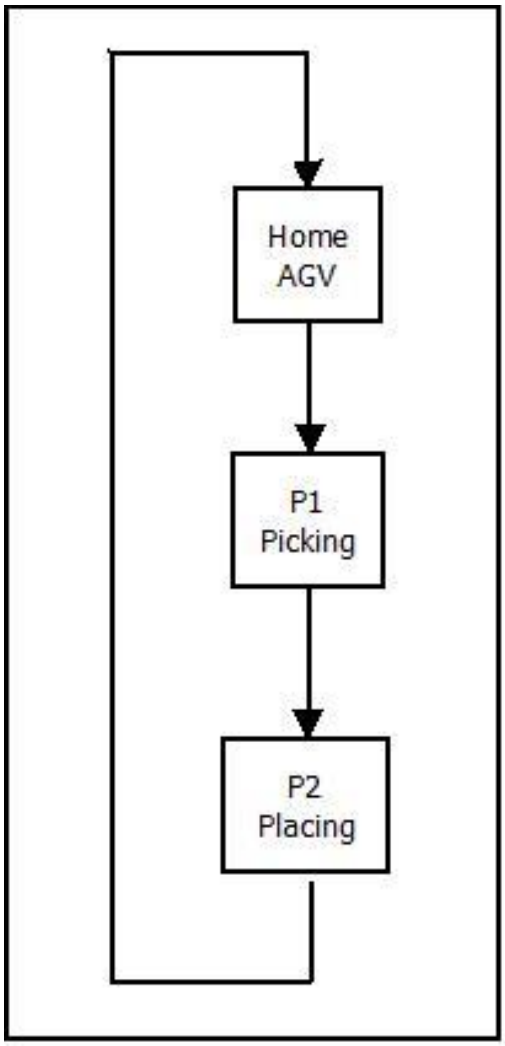
APPENDIX A


Start
Read state of AGV
Is it safe for robot to move? (AGVin = 1)
False
True
Robot begins work (AGVout = 1)
AGV stays still
Robot is ready
AGV sends task to robot
Robot performs task
Is the work done?
False
True
Robot goes home (AGVout = 0)
AGV can move (AGVin = 0)
End

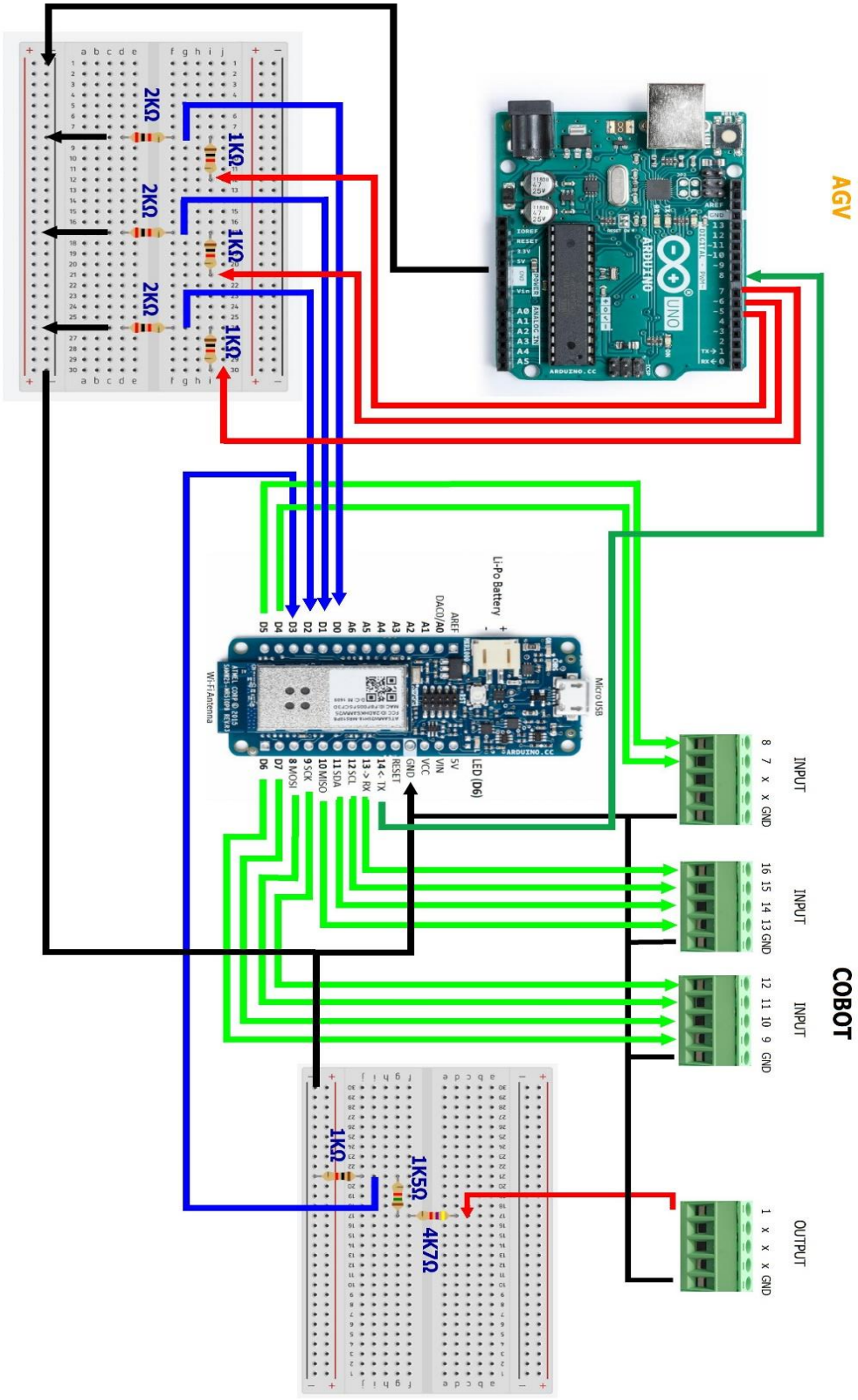
APPENDIX B


Start
Arduino receives command
Is command incorrect?
True
Message: Wrong command
End
False
Check state of robot
Is robot busy?
True
Message: Robot is BUSY
End
False
Sends command to robot
Clean up I/O
Message: Command received
End

APPENDIX C



APPENDIX D


AGV
COBOT
2KΩ
1KΩ
2KΩ
1KΩ
2KΩ
1KΩ
Li-Po Battery
Micro USB
LED (D6)
INPUT
INPUT
INPUT
OUTPUT
1KΩ
1K5Ω
4K7Ω

APPENDIX E