Mobile Manipulator Robot: Omni 3 Wheels Manipulator Robot

Youssef Mestiri

Work performed under the supervision of

Prof. Dr. José Gonçalves

Prof. Dr. José Lima

Dr. Mohamed Aymen Slim

Master in Industrial Engineering

2020-2021

Mobile Manipulator Robot: Omni 3 Wheels Manipulator Robot

Master in Industrial EngineeringEscola Superior de Tecnologia e Gestão

Youssef Mestiri

2020-2021

To my dear father Afif and my dear mother Monia ,for their love, sacrifices and support in the most difficult moments, which are at the origin of my success, may God keep and protect them.

To my dear brother Elyes, for his continuous encouragement and support since my birth, I wish you a life full of happiness and success.

To all the people who, actively or not, participated and helped in the accomplishment of this work. It is with great pleasure, I dedicate this modest work to you

Acknowledgements

First of all, I would like to thank Allah, the Almighty and the Merciful, who has given me the strength and patience to accomplish this humble work.

With the most sincere gratitude, I would like to thank my supervisors Prof. Dr. José Gonçalves, Prof. Dr. José Lima and Dr. Mohamed Aymen Slim, for all their efforts, their constant help and the trust they give me on a daily basis, without their active participation and encouragement this work would not have been done. You have been an undeniable source of motivation and working with you has been a real pleasure.

I would like to express my deepest gratitude to the members of the scientific committee for their kindness in reading this study and their interest in reviewing this document and enriching it with their proposals.

I would like to thank Dr. Yassine Mkadem, for all his efforts and for contributing to the success of the partnership between ULT and the IPB, which allowed us to carry out this work in the IPB premises. I also take this opportunity to thank all the staff and teaching team either at ULT and the IPB.
And finally a big thank you to my friends, Ekram Trabelsi, Seif Lazghab, Arezki Abderrahim Chellal, Khalil Fraj, Yosri Tayechi , Majd Chellal, Hamza Khadraoui, Ameni Nafkha, Issam Khorchani, Mohamed Abdel waheb Sethom and many others, for always being around.

Abstract

Robots are electromechanical machines having ability to perform tasks or actions on some given electronic programming. While Omni directional mobile robots have been popularly used in several applications since they can respond more quickly and it would be capable of more sophistication. A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. This work proposes to design and create a model of a three-Wheeled Omnidirectional manipulator robot that can move faster and transport materials and placed on a processing machine by combine the two type of robots. Using an electrical, mechanical and power supply model controlled by PID control and serial communication between two microcontrollers.

Keywords: three-Wheeled Omnidirectional manipulator robot, PID control, power supply, serial communication

Resumo

Os robôs são máquinas electromecânicas com capacidade para executar tarefas ou acções em alguma programação electrónica dada. O robôs móveis Omni direccionais têm sido popularmente empregados em várias aplicações porque podem responder mais rapidamente e seriam capazes de ser mais sofisticados. Também um braço robótico é um tipo de braço mecânico, geralmente programável, com funções semelhantes às de um braço humano; o braço pode ser a soma total do mecanismo ou pode fazer parte de um robô mais complexo. Este trabalho propõe-se conceber e criar um modelo de robô manipulador Omnidireccional de três rodas que pode mover-se mais rapidamente e transportar materiais e ser colocado numa máquina de processamento através da fusão dos dois tipos de robôs. Utilizando um modelo eléctrico, mecânico e de alimentação controlado por controlo PID e comunicação em série entre dois microcontroladores.

Palavras-chave: robô manipulador omnidireccional de três rodas, controlo PID , fonte de alimentação , comunicação em série

Contents

List of Tables

List of Figures

Acronyms

3D	3 Dimensions. 2, 24
A	Ampere. 38
AC	Alternate Current. 29
CAD	Computer Aided Design. 17, 26
CAE	Computer-Aided Engineering. 26
CAM	Computer-Aided Manufacturing. 26
CATIA	Computer-Aided Three-Dimensional Interactive Application. 26
DC	Direct Current. 30
EOD	Explosive Ordenance Disposal. 4
GaAs	Gallium arsenide. 34
PCB	Printed Circuit Board. 27
PID	Proportional Integral Derivative. 20
PLM	Product Management Lifecycle. 26
PN	P-type cilicon, N-type cilicon. 34

ROV Remotely Operated Vehicles. 6

TWOMR Three-Wheeled Omnidirectional Mobile Robot. 10

U.S. United States. 4

UART Universal Asynchronous Receiver/Transmitter. 24

V Volt. 18, 24

Chapter 1

Introduction

1.1 Theoretical Framework

Mobile manipulation is nowadays a widespread term to refer to robot systems built from a robotic manipulator arm mounted on a mobile platform. Such systems combine the advantages of mobile platforms and robotic manipulator arms and reduce their drawbacks. For instance, the mobile platform extends the workspace of the arm, whereas an arm offers several operational functionalities [1]. A mobile manipulation system offers a dual advantage of mobility offered by a mobile platform and dexterity offered by the manipulator. The mobile platform offers almost workspace to the manipulator. The extra degrees of freedom of the mobile platform also provide user with more choices. However, the operation of such a system is challenging because of the many degrees of freedom and the unstructured environment that it performs in. General system composition:

- Mobile platform
- Robot manipulator
- Sensors
- Tooling

1.1.1 Objectives

The work presented in this thesis has several objectives mainly:

- design a mechanical, electrical and power supply to mobile manipulator robot
- Create the mechanical parts
- Print with the mechanical parts 3 Dimensions (3D) printer
- Assemble robot parts (motors, sensors, motor griver.)
- Coding the robot usig PID controller, serial Communication and interruption
- Testing

All the work and the test were made at LCAR laboratory using measuring, electronics and mechanical devices.

1.2 Document Structure

This document is divided into 5 sections, using the following structure. Section 2 promotes the work already done in this field and highlights the objectives intended throught this work. Section 3 describes the system design and technical study with the relation between all parts. Section 4 present all equipment used and a financial study. Section 5 provides the results and some discussion. Finally, Section 6 draws together the main ideas described in this document and outlines future work to further improve this work.

Chapter 2

State of the Art

Robots are developing rapidly due to technological advances and the increased need for mobility. They are taking on increasingly complex forms, from huge stationary robots to small devices that can move around, perform many difficult tasks, and move freely in factories, offices and even homes. A mobile manipulator is a robotic system that consists of a robotic manipulator arm attached to a movable platform. These systems combine the benefits of mobile platforms and robotic manipulator arms while trying to minimize their disadvantages. The mobile platform, for example, expands the arm's workspace, whereas an arm has numerous operational functions. This chapter gives a general overview on the history of robotics in the world. Then, it presents the new techniques used and identifies the tasks to be performed in the functional analysis part. Finally, it explains the issues that led to the elaboration of this project as well as the specifications.

2.1 The Evolution of Robotics in the Industry Sector

While productivity, accuracy, repetition and speed have been at focus of research in robotics for years, other criteria based on industrial transformation have become of equal importance. Modern robots must meet criteria such as flexibility, adaptability, precision, and autonomy of action in practically every industry. These characteristics reveal mobile robots and related automation technology play a significant role in the development and advancement of industry. However, it is not only the industrial sector that benefits from these developments [2], being shown in the figure below that the world supply chain also improves with the use of industrial robots

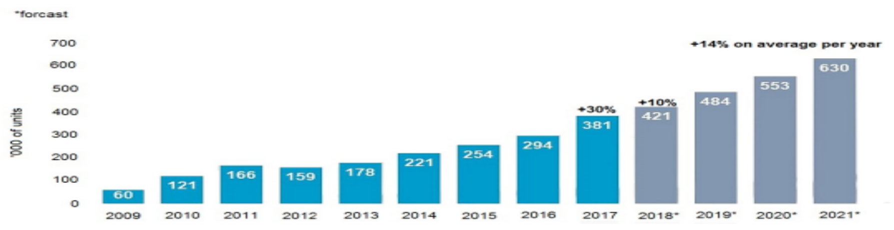


Figure 2.1: The Robotics Evolution in The Industrial sector [3]

2.2 Description of Mobile Manipulation

A mobile manipulation system benefits from the mobility of a mobile platform and the dexterity of the manipulator. The manipulator has increased workspace on the mobile platform. The additional degrees of freedom offered by the mobile platform provide users more options. However, due of the multiple degrees of freedom and the unstructured environment in which it operates, the operation of such a system is difficult.

2.3 Evolution of Mobile Manipulation

Since the 1960's, robotic arms have been transforming the way companies do business. Although they were big, bulky, and highly specialized, early industrial robotic arms were able to automate repetitive tasks at a rate of three times the speed of humans[4]. Several decades later, robotic arms are not only smaller and sleeker, but also able to mimic the way humans think, react, and adapt due to the improved controlling algorithms. Although mobile robots have existed for decades, it was the United States (U.S.) Military that widely adopted and deployed unmanned ground vehicles with manipulation capability to keep troops safe while performing Explosive Ordenance Disposal (EOD) missions during the wars in Iraq and Afghanistan[5]. Considered cutting-edge at the time, these simple robotic arms with 6 degrees of freedom would be considered crude by today's standards, especially since modern robot arms can offer more dexterity than a human arm [6].

2.4 Classification of Mobile Manipulator

The concept of the mobile manipulation consists of associating, in the same system, one or more means of locomotion with one or more means of manipulation. The means of locomotion provide the system with a working space limited mainly by its energy autonomy. The means of manipulation ensure that the system is able to move and manipulation capabilities. Among the systems that currently exist we can mention three main families.

2.4.1 Multiped

As the name suggests, multi-legged mobile manipulators can have one or more legs. Humanoids are the most popular representatives of this type of manipulator as it can be seen in Figure 2.2, both with the general public and with researchers, because of the challenge they represent. From the point of view of manipulation, their possible uses are limited from an industrial point of view and their main outlet is service robotics.

Figure 2.2: Multiped Robot [7]

2.4.2 Underwater Mobile Manipulators

Underwater mobile manipulators are nowadays the most widely used mobile manipulators for work purposes. Often remotely operated, they are called Remotely Operated Vehicles (ROV) and allow access to maritime areas not accessible to these provide sampling, manipulation, measurement and data acquisition capabilities that can be adapted to the missions required missions. An example is the Victor 6000 as it can be seen in Figure 2.3

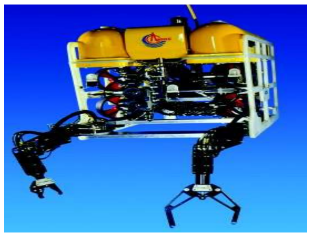


Figure 2.3: Underwater Mobile Manipulators [8]

2.4.3 Mobile Wheel Manipulators

Wheeled mobile manipulators figure [7] are more common than those presented. This is due particularly due to two facts:

- Their relatively simple mechanical design
- The natural suitability of their means of locomotion to a wide range of terrains.

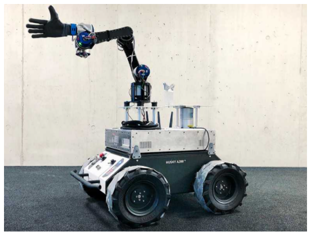


Figure 2.4: Mobile Wheel Manipulators [9]

2.5 General System Composition

Mobile manipulator is generally composed by 4 parts, as it can be seen in Figure 2.5:

- Mobile platform is a machine controlled by software that use sensors and other technology to identify its surroundings and move around its environment.

- Robot manipulator is a device used to manipulate materials without direct physical contact by the operator. They have been used in diverse range of applications including welding automation, robotic surgery and in space. It is an arm-like mechanism that consists of a series of segments, usually sliding or jointed, which grasp and move objects with a number of degrees of freedom.

- Vision system is a subarea of robotics science intended to give robots sensing capabilities, so that robots are more human-like. Robotic sensing mainly gives robots the ability to see, touch, hear and move and uses algorithms that require environmental feedback.

- Tooling aims to manipulate objects and is designed according to the needs of manipulation.

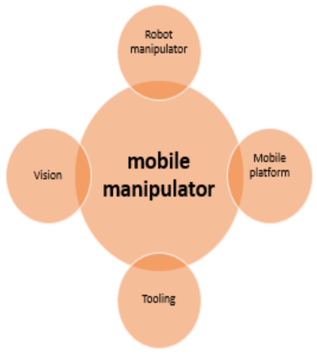


Figure 2.5: System Composition

2.5.1 Mobile Platform

A mobile robot is a mechanical, electronic and computer system that physically acts on its environment to achieve an assigned objective. This machine is versatile and capable of adapting to certain variations in its operating conditions. It has equipment that enable some functions of perception, decision and action. Thus, the robot should be able to perform multiple tasks in different ways and perform its task correctly, even if it encounters new and unexpected situations [10].
The name Mobile Robot includes all types of robots that have the ability to move, which is the common characteristic between them. The difference lies in the locomotion system, which depends on the area of use of the robot, by which the robot will achieve this ability of movement. Wheeld traction system is the most commonly applied mechanical structure. Depending on the arrangement and dimensions of the wheels, this technique ensures movement in all directions with high acceleration and speed.

The Characteristics of mobile robotics

- Perception of the environment thanks to sensors
- Capacity to adapt in the event of changes in the environment by overcomes obstacles
- Autonomous navigation, planning and action
- Task-based software/task-based programming

2.5.2 Robot Manipulator

The mechanical structure of robot manipulator consists of a sequence of rigid bodies (links) interconnected by means of articulations (joints); a manipulator is characterized by an arm that ensures mobility and an end-effector that performs the task required of the robot. The fundamental structure of a manipulator is the serial or open kinematic chain. From a topoligical viewpoint, a kinematic chain is termed open when there is only one sequence of links connecting the two ends of the chain. Alternatively, a manipulator contains a closed kinematic chain when a sequence of links forms a loop.

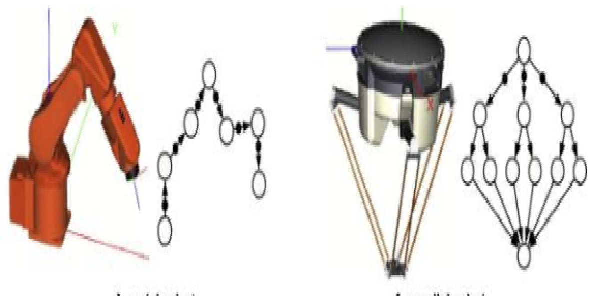


Figure 2.6: Serial and Open Kinematic Chain. [11]

2.5.3 Sensors

Sensors are the sensory parts of a robot. Some are fragile and must be protected, others on the contrary must be able to absorb shocks. The simpler ones can be directly connected to the control center, like switches. The other types require a small adapter interface, such as infrared or ultrasonic sensors. Other more sophisticated types require a special card, such as cameras. There are two types of sensors:

• The external sensors: These are exteroceptive sensors, delivering information related to the environment or to the interactions between the robot and its environment.Such as distance sensors.

• The internal sensors : These are the sensors that provide information on the internal state of the robot: wheel position or speed and battery charge sensors.

2.5.4 Tools

Final actuators whose purpose is to perform work on a part instead of picking up on them.

2.6 Three-Wheeled Omnidirectional Robot

Three-Wheeled Omnidirectional Mobile Robot (TWOMR), is a holonomic robot that can move in translation and rotation simultaneously and independently [12]. The robot is equipped with three omni-wheels that are evenly spaced around its circumference at 120 degrees. Each omni-wheel is directly connected to its motor shaft, allowing the motor and the wheel to rotate on the same axle. The robot is able to make in translation and rotation simultaneously and independently that robots on standard wheels cannot due to individual control of each motor's speed. Figure 2.7 shows a representation for a 3 wheels robot.

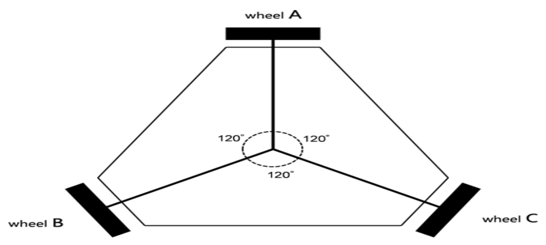


Figure 2.7: The robot configuration [13]

Figure 2.8 shows an example of an omni-wheel where the wheel is equipped with many rollers to enable it to move sideways perpendicular to the normal direction of rolling. An omnidirectional drive system requires a minimum of three omni-wheels.

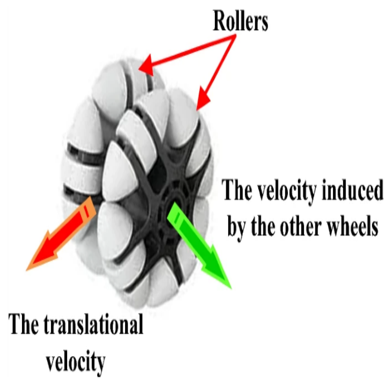

Rollers
The velocity induced by the other wheels
The translational velocity

Figure 2.8: Omni-Wheel [12]

2.6.1 Coordinates and Symbols

Figure 2.9 depicts the three-wheel robot's configuration, as well as all of the system's axes, pertinent forces, and velocities. A 120-degree angle separates the wheels.

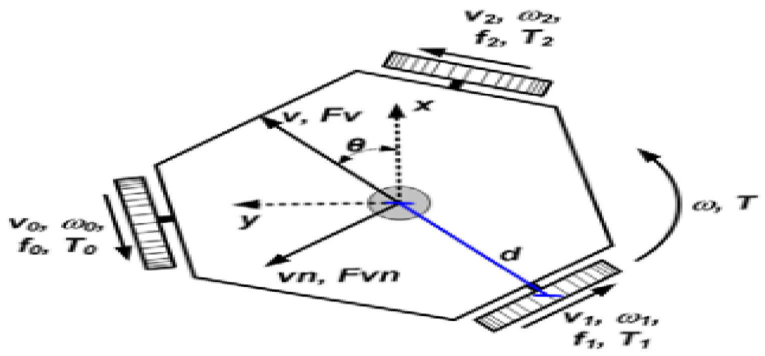

v_2, ω_2, f_2, T_2
v, Fv
x
θ
y
ω, T
v_0, ω_0, f_0, T_0
d
vn, Fvn
v_1, ω_1, f_1, T_1

Figure 2.9: Coordinate Frames
[14]

notation used

$x, y,$ - Robot's position (x,y) and angle to the defined front of robot

d [m] - Distance between wheels and center robot

$v0, v1, v2$ [m/s] - Wheels linear velocity

$0, 1, 2$ [rad/s] - Wheels angular velocity

$f0, f1, f2$ [N] - Wheels traction force

$T0, T1, T2$ [N $\cdot$ m] - Wheels traction torque

v, vn[m/s] - Robot linear velocity

[rad/s] - Robot angular velocity

Fv, Fvn [N] - Robot traction force along v and vn

T [N $\cdot$ m] - Robot torque

2.7 Three-Wheeled Omnidirectional Kinematics

Vx(t)=dx(t)/dt, vy(t)=dy(t)/dt, and (t)=d(t)/dt are the kinematic models of an omnidirectional robot positioned at (x, y,) [14] Figure 2.9 .The linearvelocities vxand vyon the static axis may be converted to linearvelocitiesv and vn on the robot local axis using Equation 1.

$$Xr \quad = \quad \begin{bmatrix} v(t) \\ Vn(t) \\ W(t) \end{bmatrix} \tag{2.1}$$

$$X0 \quad = \quad \begin{bmatrix} vx(t) \\ Vy(t) \\ W(t) \end{bmatrix} \tag{2.2}$$

The relationship between the wheel velocities and therobot velocities is:

$$\begin{bmatrix} v0(t) \\ V1(t) \\ V2(t) \end{bmatrix} \quad = \quad \begin{bmatrix} -sin(\pi/3) & cos(\pi/3) & d \\ 0 & -1 & d \\ sin(\pi/3) & cos(\pi/3) & d \end{bmatrix} \quad \cdot \quad \begin{bmatrix} v(t) \\ Vn(t) \\ \omega(t) \end{bmatrix} \tag{2.3}$$

2.8 The TinkerKit Braccio

The TinkerKit Braccio is a arm assembly kit composed of plastic parts and 6 servomotors, which are controlled by a Shield designed to overlap on an Arduino development board. The current estimated price is currently about 199 € r.

2.8.1 Characteristic

One of the most notable features is that its structure is not unique, so it can be adapted to different applications as shown in this Figure 2.10

Concerning the more technical specifications, they are summarized in table 2.1, which presents the most general characteristics, and then in table 2.2 with the details of the

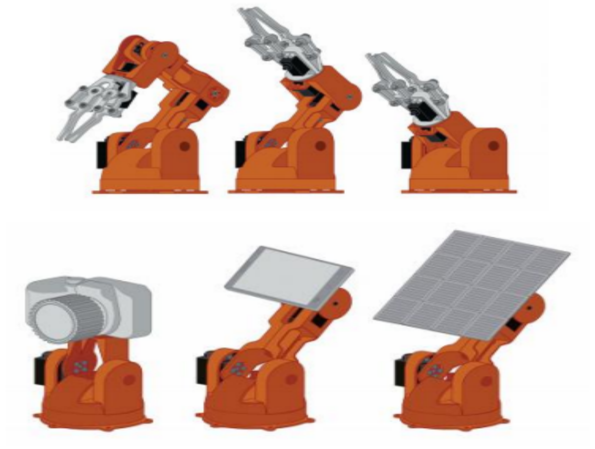


Figure 2.10: Different Cpnfigurations of the Tinkerkit Braccio Robot [15]

servomotors available in the Tinkerkit Arduino Robot kit. This servo is controlled via

Characteristics	Details
Weight	0.792 Kg
Operating distance	80 cm
Maximum Height	52 cm
Base width	14 cm
Clamp width	9cm
Cable length	40 cm
Load capacity	0.15 kg / 32cm
Load capacity in minimum configuration	0.4
power supply	5V / 5A
Shield consumption	0.02W
Maximum Shield Current	From M1 to M4 1.1A From M5 to M6 0.75A
Servomotors	2x SR 311 + 4x SR431
PCB Shield Measurements	6,858x5,334 cm

Table 2.1: Technical Specifications [16]

analog signal(PWM), the range of signal impulse:from 0.5ms to 2.5ms. A Rotation range :from 0to 180

Characteristics	Servomotor SR 311	servomotor SR 431-Dual
Control Signal	Analog PWM	Analog PWM
Torsional stress	4,8V:3,1kg.cm 6v :3,8 kg.cm	4,8V:12,2kg.cm 6v :14,5 kg.cm
Velocitat	4,8V:0,14s/60º 6v :0,12s/60º	4,8V:0,20s/60º 6v :0,18s/60º
Rang of rotation	180º	180º
Dimensions	31,13 x16,5x28,6 mm	42,0x20,5x39,5
weight	0,027kg	0,062kg
Speed	0.14 sec/60	0.20 sec/60
Torque	3kg.cm	12.2kg.cm

Table 2.2: Technical Characteristics of Servomotor SR 311 and SR 431-Dual Output Servo [8]

Figure 2.11: Servomotor SR 311 and SR 431-Dual

2.8.2 Degrees of Freedom

The number of joints of a manipulator arm is also referred to as the degree of freedom. Braccio Robot have between 4 and 5 degrees of freedom as shown in Figure 2.12.

2.9 Robot Programming Language

Intelligent robots must be capable of action in reasonably complicated domains with some degree of autonomy. This requires adaptivity to a dynamic environment, ability to plan and also speed of execution. In the case of helper robots, or domestic robots, the ability to adapt to the special needs of their users is crucial. The problem addressed here is one of how a user could instruct the robot to perform tasks which manufacturers cannot

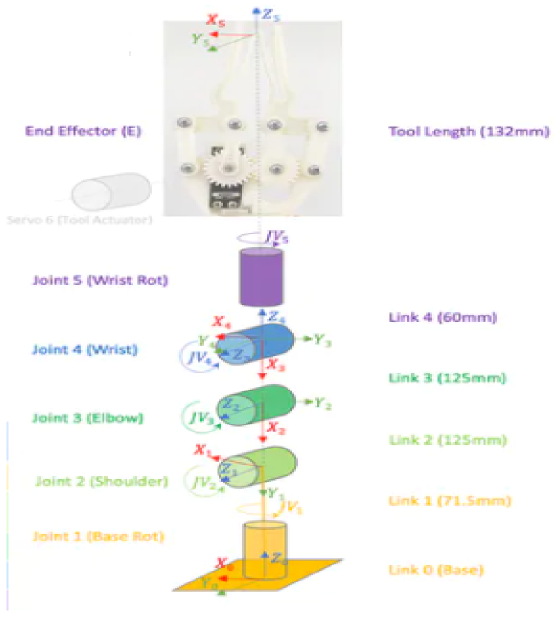


Figure 2.12: Degree of Freedom
[17]

completely program in advance. [6].

1. C: is an imperative programming language designed for system programming. Invented in the early 1970s with UNIX, C has become one of the most widely used languages. Many modern read-only languages such as C++, Java and PHP take up aspects of C. However, professionals place C language at the top of the list for several reasons: it is simple and powerful.

2. C++: in the 80's B. Stroustrup proposed to call C++ a new language, designed not as a replacement but as an improvement of the C language. Like C, C++ takes a very machine-like view. It was primarily intended for writing operating systems but its characteristics have opened up other perspectives. It consists of very explicit, short instructions, whose execution time can be predicted in advance, when the program is written.

3. Java: Java is a high-level computer programming language. High-level languages, such as Java, allow programmers to write instructions using commands in English. Each instruction in a high-level language corresponds to many instructions in the language of

the machine.

4. The assembler: The assembler language is very close to the machine language (i.e. the language used by the computer: information in binary, i.e. 0s and 1s). It therefore depends strongly on the type of processor. Thus there is not one assembler language, but one assembler language per type of processor [18].

2.10 Conclusion

After the presentation of the history of robotics in the world, the Description of Mobile Manipulation and the techniques used, in the next chapter, it will presented the design of the system, the technical study and the equipment used for the proposed robot.

Chapter 3

System Design and Technical Study

The technical study of a project is a design expertise that serves as a preliminary study for the production of a prototype that meets the set expectations and budget. It mostly comprises of displaying the robot's information and technical qualities. The design of a product is a complex task that requires the use of different skills from various areas. The three-wheel omnidirectional manipulator robot is the name of our project. This chapter aims to describe first the operating principle adopted for the system, then to describe the relationships between the various electronic, and finally the mechanical Computer Aided Design (CAD) .

3.1 Robot Design

The robot is equipped with three omni-wheels that are evenly spaced around its circumference at 120°. In the center of the robot there is a robotic arm with a certain number of degrees of freedom, which is connected to the moving part. The Figure 3.1 attempts to show the different relationships between the components of the robot.

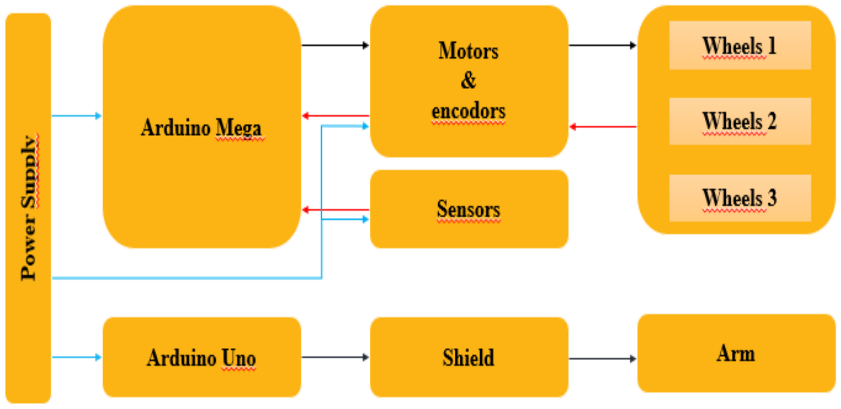


Figure 3.1: Robot Design

3.2 Power Supply Design

To power the main control system of power supply and motor drive power supply figure 3.2, the moving platform of choice was 12 Volt (V) lithium-ion battery as a power source, main control board needs 5V and for the sensors 3.3V power supply, and the power needed is small, so to reduce power consumption, the Buck step-down circuit is powered down to 5V and 3.3V. Because the motor drive requires a 12 V power source and a high amount of power.

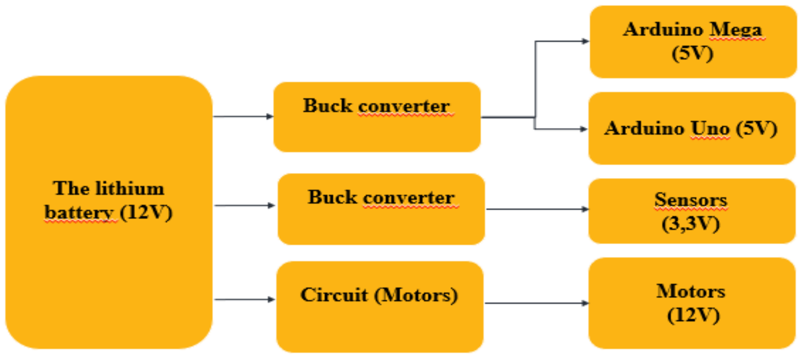


Figure 3.2: Power Supply Design

3.3 Microcontroller

The two microcontrollers Arduino Mega and Arduino Uno can be programmed to analyze and produce electrical signals, so as to perform a wide range of tasks such as home automation (control of domestic appliances, lighting, heating ...), the control of a robot and embedded computing. It is a platform based on a input/output interface. The master microcontroller is the central element of the system, it is one of the most important because it contains the code to be executed, but also it depends on all the other components like the sensors to receive the external information

3.4 Braccio Shield

The included Braccio shield connects to an Arduino or compatible board and has TinkerKit connectors for easy connection of servo motors.

3.4.1 Battery Dimensioning

This model's total storable power varies depending on the capacity of the cells fitted. The use of three 2200 mAh lithium cells provides the system a battery capacity of 6600 mAh. The greatest current that can be given is 2 A, allowing for a power output of up to 24 W, which is adequate to power the robot.

3.4.2 Buck Converter

Buck converters transform a greater input voltage into a stabilized lower output voltage figure [19], which is rather impressive3.3. On the other hand, correct implementation of the controller, which is extremely difficult to parameterize, is required for this regulator. Incorrect control can result in interference, radio frequency interference, and harmonics, all of which can impair the proper operation of electronic components and shorten their service life.

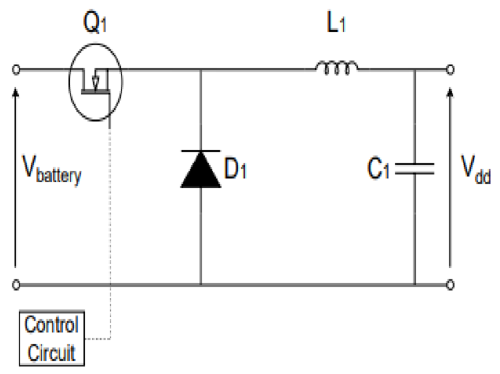


Figure 3.3: Buck Converter Schema
[20]

3.5 Motor Control

The key to ensuring that the omnidirectional mobile platform moves in the desired direction is precise control of the motor speed. Motor control can be divided into two categories:

- The open-loop control means that the main controller is provided direction and speed without requiring to know the motor variables state.

- Closed-loop systems are considered as fully automatic control system because it is designed in a way that the achieved output is automatically compared with the reference input to have the required output, so we must use a closed-loop control method for the motor output speed before using the Proportional Integral Derivative (PID) algorithm to adjust the input speed to the actual predicted speed.

3.5.1 PID Algorithm

The input of PID controller figure3.4 is the error e(t) between the given output value r(t) and the actual output value y(t), and the input error e(t).The input error can be expressed as:

$$e(t) = r(t) - y(t) \tag{3.1}$$

The mathematical expression of PID algorithm is:

$$u(t) = K_p \cdot \ e(t) + K_i \cdot \int t \cdot e(t) + K_d \cdot \frac{de(t)}{dt} \tag{3.2}$$

where,

- K_p :Represent the proportion coefficient
- K_i :Represent the integral coefficient
- K_d :Represent the differential coefficient

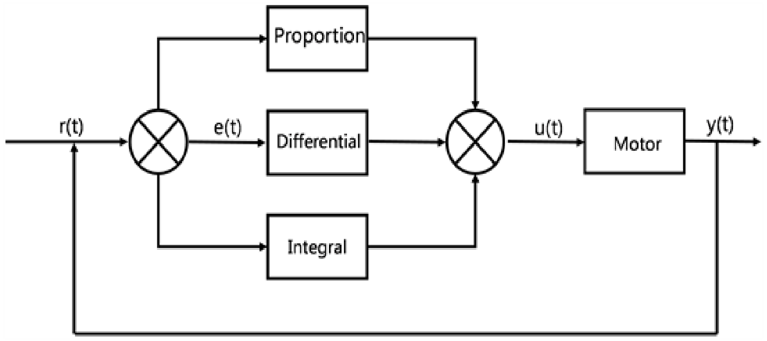


Figure 3.4: PID Control Principle

A discrete PID algorithm formula needs to be obtained:

$$u[k] = u[k-1] + K_p \cdot [e[k] - e[k-1]] + K_i \cdot e[k] + K_d \cdot [e[k] - 2 \cdot e[k-1] + e[k-2]] \tag{3.3}$$

where,

- u[k] : represents the output value at the first k sampling time.

- u[k-1] : represents the output value at the first k-1 sampling time.
- e[k] : the error values of input and output at the current sampling time.
- e[k-1] : the error value of input and output at the last sampling time.

3.6 H-Bridge DC Motor Control

For controlling the rotation direction, we just need to invert the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements figure 3.5, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.

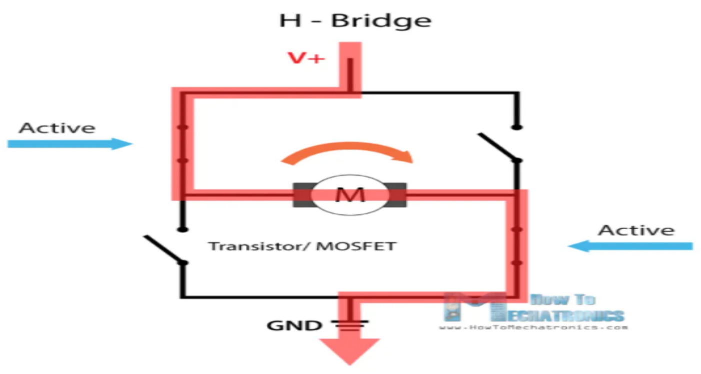


Figure 3.5: H-Bridge[21]

3.7 Programing with Interruption

Interrupts are useful in microcontroller applications for making things happen automatically and for resolving timing issues. Reading a rotary encoder or monitoring user input are two examples of interrupt-worthy operations. To ensure that a program always captured the pulses from a rotary encoder and never missed a pulse, the software would have to constantly poll the sensor lines for the encoder in order to catch pulses as they occurred, making it very difficult to design a program to do anything else. Other sensors, such as a sound sensor attempting to detect a click or an infrared slot sensor (photo-interrupter) attempting to detect a coin drop, have a similar interface dynamic. In all of these cases, using an interrupt allows the microcontroller to focus on other tasks while still receiving input [22].

3.7.1 Type of Interruption

Interrupts significantly improve the use of microcontrollers. Interrupts cause programs to react to the microcontrollers' hardware, which could be a reaction from the circuit environment outside of the microcontroller. An interrupt is a circumstance that forces the microprocessor to operate on a different task for a short period of time before returning to its original duty.

Internal interruption

An internal interrupt is a sort of interrupt that occurs as a result of a specific event within the processor, such as a division by zero error, which causes an internal interrupt known as the divide by a zero interrupt. In arduino, an interrupt is a signal that tells the processor to halt what it's doing right now and handle some high-priority work. Any void function can be used as an interrupt handler. It will be invoked anytime the interrupt signal is triggered if it is written one and attached to an interrupt.

External interruption

The arduino mega 2560 has six external interrupts for serving external devices. Both of these interrupts have a low active level. An external interrupt alerts the microcontroller to the need for routine service on an external device. External interrupt is a process in which the Arduino suspends its regular work or loops and switches to the interrupt function to perform the task assigned to it. Digital pins 2, 3, 18, 19, 20, and 21 are the ones to look for [19]. This interrupt model has been used quite a few times in our project.

3.8 Serial Communication Arduino Mega & Arduino Uno

In the Arduino ecosystem, serial communication is the most commonly utilized way. It's also known as Universal Asynchronous Receiver/Transmitter (UART), and it's an asynchronous link, as the name suggests. It can be used to connect an Arduino to a computer or other electronic devices. When the two devices are connected, the data is also shown on the computer via this port.from the Arduino Mega is connected with the Vin pin of the Arduino Uno. In the serial communication we will power up the Arduino Uno using the Arduino Mega's 5 V and the grounds of both the Arduino boards are connected together.

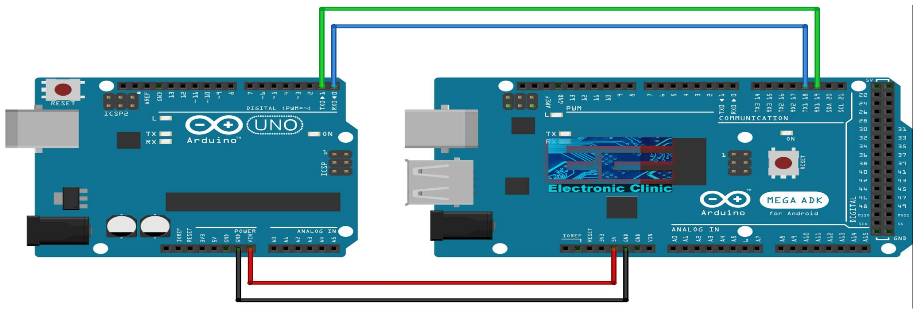


Figure 3.6: Serial Communication Arduino Mega & Arduino Uno [23]

3.9 Platform and Support Design

After having known the operating system and our system components. We made the design of our mobile part by respecting the standards then we printed it in 3D.

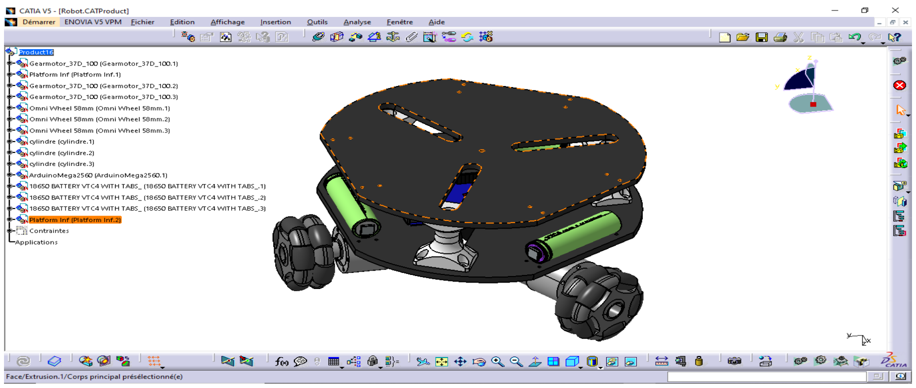

CATIA V5 - [Robot.CATProduct]
Démarrer
ENOVIA V5 VPM
Fichier
Edition
Affichage
Insertion
Outils
Analyse
Fenêtre
Aide
Product16
Gearmotor_37D_100 (Gearmotor_37D_100.1)
Platform Inf (Platform Inf.1)
Gearmotor_37D_100 (Gearmotor_37D_100.2)
Gearmotor_37D_100 (Gearmotor_37D_100.3)
Omni Wheel 58mm (Omni Wheel 58mm.1)
Omni Wheel 58mm (Omni Wheel 58mm.2)
Omni Wheel 58mm (Omni Wheel 58mm.3)
cylindre (cylindre.1)
cylindre (cylindre.2)
cylindre (cylindre.3)
ArduinoMega2560 (ArduinoMega2560.1)
18650 BATTERY VTC4 WITH TABS_ (18650 BATTERY VTC4 WITH TABS_.1)
18650 BATTERY VTC4 WITH TABS_ (18650 BATTERY VTC4 WITH TABS_.2)
18650 BATTERY VTC4 WITH TABS_ (18650 BATTERY VTC4 WITH TABS_.3)
Platform Inf (Platform Inf.2)
Contraintes
Applications
Face/Extrusion.1/Corps principal présélectionné(e)

Figure 3.7: Mobile Robot 3D Design

3.9.1 Platform Design

The platform has well determined dimensions so that we can print it in plastic 2 times for the 2 parts (base part and top part) with 3D printer to support the weight of all the components. One for Motors, Motor Driver and Sensors the second for the arm.

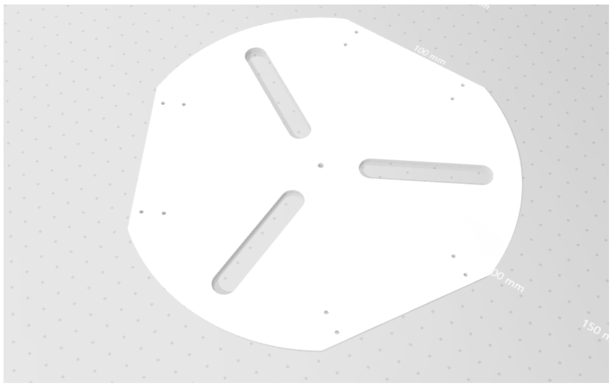


Figure 3.8: Platform Design

3.9.2 Support Design

The supports figure3.9 are the most delicate parts because we are limited to a very specific space and they have to support the weight of our robotic arm more than that Regarding the top, we should avoid to have planar surfaces and It would be better if the convection between the pillar and the top should be more smooth.

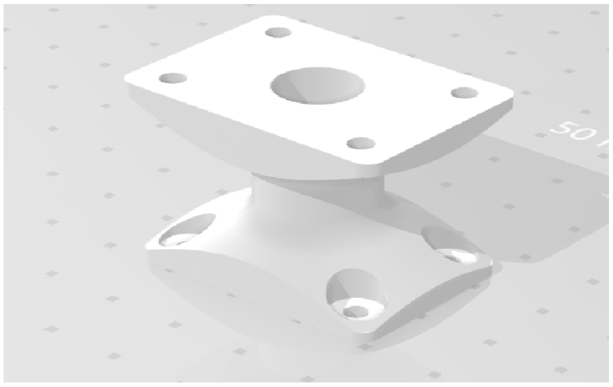


Figure 3.9: Support Design

3.10 Software and Programming Tools

Two software packages are mainly used for this project. CATIA V5R18, where the mechanical design was done in 2D and 3D, and ARDUINO IDE to write and compile the algorithm via an Arduino board.

3.10.1 CATIA V5R18

Computer-Aided Three-Dimensional Interactive Application (CATIA) is a multi-platform software suite for CAD, Computer-Aided Manufacturing (CAM), Computer-Aided Engineering (CAE), Product Management Lifecycle (PLM) and 3D, developed by the French company Dassault Systèmes.

3.10.2 ARDUINO IDE

The Arduino IDE is an open source software for Arduino, which uses an abbreviation of the C/C++ language, with many libraries already implemented or that can be implemented, it allows easy writing of codes and implementation [24]. This software can be used in any Arduino board.

3.10.3 Schematic and PCB Design Software

Various software packages provide the possibility of designing electronic circuits, such as Fritzing, ISIS Proteuse, EASY EDA or Eagle, all of which have a large library containing hundreds or thousands of components. It is possible to make the equivalent circuit first, simulate it according to the software, and then make a Printed Circuit Board (PCB) equivalent to the designed circuit. Making a schematic is very useful because it allows you to detect possible problems in the design and to think more clearly about other alternatives. As the project progressed, a schematic was developed to meet the requirements of our project.

3.11 Conclusion

The methods presented in this chapter are the most used in our system and in our programming code because we have guaranteed a well determined power supply system, a motor control in a closed loop and it was applied an interruption.
The master microcontroller, which is the central element of the system, is one of the most important, but it depends on all the other components, meaning that if one of them fails, the whole system is at risk. Each electronic component, when mounted in this robot circuit, according to a defined installation principle, performs a specific function.

Chapter 4

Equipement Used and Financial Study

In this chapter we are going to describe the different characteristics of the materials used in our project, the instructions provided by the manufacturer for each of them and an economic and financial analysis of our robot is made at the end.

4.1 The Controller

A controller is a system that looks like a computer: it has a memory, a processor. Microcontrollers have reduced performance, but are small in size and consume little power, making them indispensable in any embedded electronics solution.

4.1.1 Arduino Mega 2560

The ATmega2560 is the basis for the Arduino Mega 2560 microcontroller board. It contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power connector, an ICSP header, and a reset button. It comes with everything you need to get started with the microcontroller; simply plug it into a computer with a USB connection or power it with an Alternate

Controller	Atmega 2560
Operating Voltage	5V
Input Voltage	7V-12V
USB Port	Yes
DC Power Jack	Yes
Current Rating Per I/O Pin	20mA
Current Draw from Chip	50mA
Analog Pins	16 (Out of Digital I/O Pins)
Flash Memory	256KB
SRAM	8KB
EEPROM	4KB
Crystal Oscillator	16MHz
LED	Yes/Attached with Digital Pin 13
Wi-Fi	No
Shield Compatibility	Yes

Table 4.1: Arduino Mega 2560 Specification
[25]

Current (AC) to Direct Current (DC) converter or battery to get started. Most shields created for the Uno and previous boards Duemilanove or Diecimila are compatible with the Mega 2560 board.

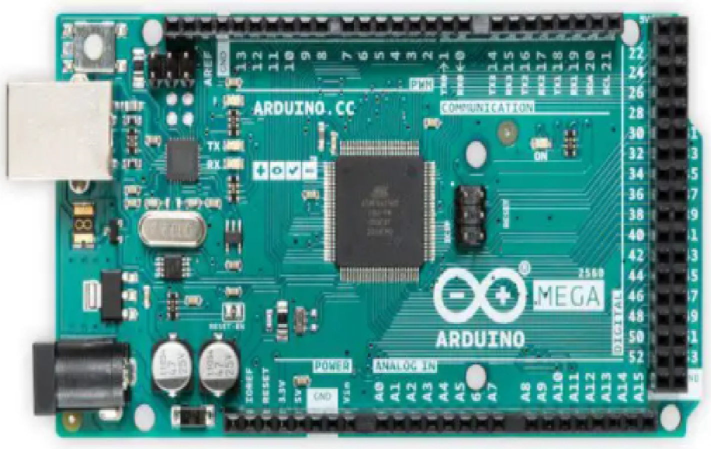


Figure 4.1: Arduino Uno

4.1.2 Arduino Uno

The Arduino Uno is a microcontroller board that uses the ATmega328P microprocessor. There are 14 digital input/output pins (six of which may be used as PWM outputs), six analog inputs, a 16 MHz ceramic resonator, a USB connection, a power connector, an ICSP header, and a reset button on the board. It comes with everything you'll need to get started with the microcontroller. It has the same specifications as the PWM Digital I/O, but fewer analog input pins [25]

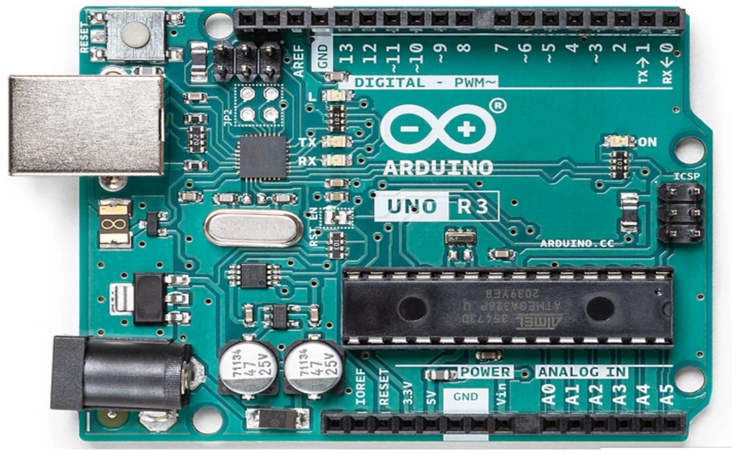


Figure 4.2: Arduino Uno

4.2 EMG30 Motor

The EMG30 assembles a 12 V motor with an encoders with 30:1 reduction gearbox. It is excellent for small or medium robotic application, providing cost-effective drive and feedback for the user. It also has a conventional noise suppression capacitor across the motor windings. The EMG30 is supplied with a six-way JST connector, the connectionsare given in table4.2. These motors provide a wheel revolution 200 rpm offload when supplied 12 V power supply. About input and output of motors, the input, is voltage given by the L298N motor driver and the output is shaft angular velocity of motor. A DC motor

Red	Motor power +
Black	Motor power -
Green	GND encoder
Brown	Hall sensor Vcc
Blue	A output encoder
Purple	B encoder output
Shield Compatibility	Yes

Table 4.2: EMG30 Color Connection

converts direct electrical power into mechanical power, to do this it has a mechanical part, the rotor, and an electrical part, the armature.

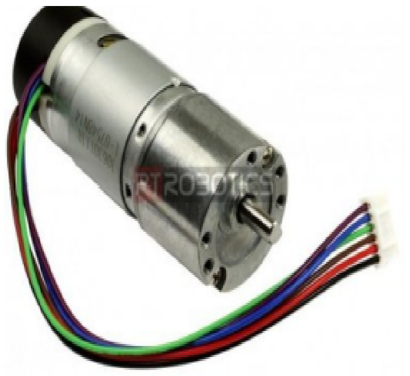


Figure 4.3: EMG30 Motor

4.3 The Motor Driver L298N

The L298N is a dual H-Bridge motor driver that allows for simultaneous speed and direction control of two DC motors. DC motors with voltages ranging from 5 to 35V and peak currents of up to 2A can be powered by the module. Two screw terminal blocks for motors A and B, as well as a screw terminal block for the Ground pin, motor VCC, and a 5V pin that may be used as an input or output, are included in the module figure4.4. This is determined by the voltage applied to the motor's VCC. The module has an integrated 5V regulator that can be turned on or off using a jumper. If the motor supply voltage is more than 12V, the 5V regulator can be enabled, and the 5V pin can be utilized as an output.

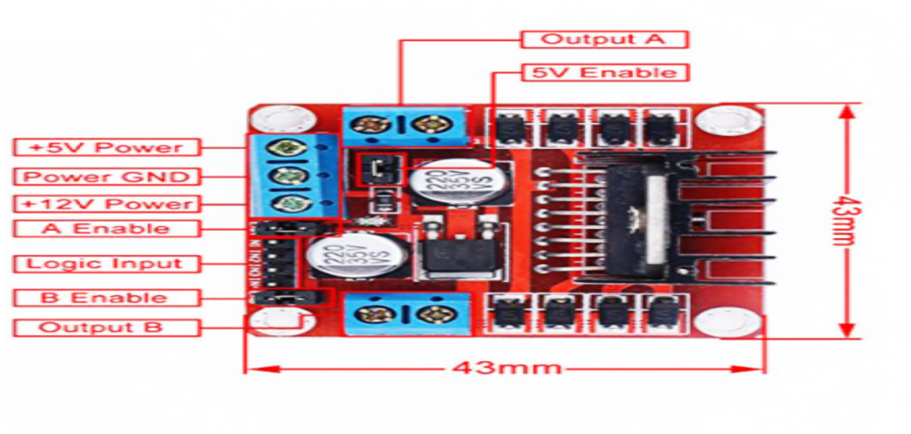


Figure 4.4: Motor Driver L298N
[26]

4.4 Sensors

A sensor is a device, module, machine, or subsystem that detects events or changes in its surroundings and transmits the data to other electronics, most commonly a computer processor. Sensors are always used in conjunction with other electronics.

4.4.1 TCS3200 Color Sensor

The functional block diagram of TCS3200D is shown in the figure 4.5.
TCS3200D has four different types of filters: red, green, blue, and clear (no filter). The types of filters employed by the device (blue, green, red, or clear) may be chosen by two logic inputs, S2 and S3[24], when the sensor is irradiated by a beam of light [27]. The table 4.3 illustrates the relationship among S2, S3 and filter type:

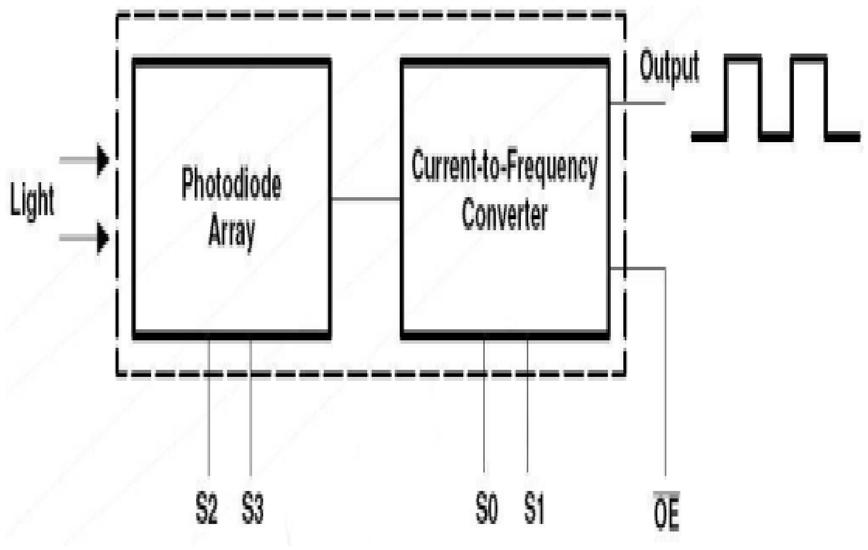


Figure 4.5: The Functional Block Diagram

S2	S3	Filter type
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Table 4.3: Relationship Among S2, S3 and Filter Type

4.4.2 Infrared Reflective Sensor

There are two pieces to an infrared sensor: an infrared transmitter and an infrared receiver. The infrared transmitter is made up of a luminophor made up of an infrared LED array and a P-type cilicon, N-type cilicon (PN) junction made up of a particular material with high infrared radiation efficiency, often Gallium arsenide (GaAs)). When a current is injected into the PN junction, a forward bias voltage can activate a source of infrared light with a center wavelength range of 830nm-950nm. The strength of the infrared light stimulated 4.6 is determined by the current injected. If the injected current exceeds the maximum rating, however, the strength of the infrared light may decrease as the current rises. Semiconductors that transform infrared light impulses into electrical signals are known as infrared receivers. The heart of it is a PN junction made of a particular material. It has a different PN junction than a general-purpose diode, which allows it to receive more infrared light. As the intensity of the infrared light rises, more current may be generated [28].

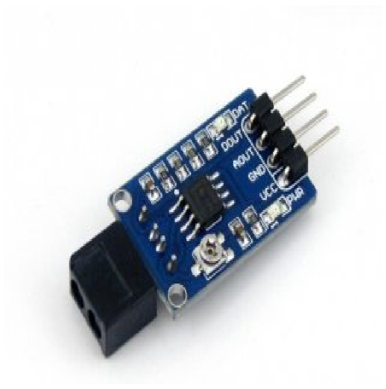


Figure 4.6: Infrared Reflective Sensor

4.4.3 QTR-8RC Sensor

With 8 IR LED/photo-transistor pairs installed on a 0.375" pitch figure4.7, this sensor module is ideal for a line-following robot figure. A MOSFET allows the LEDs to be turned off for extra sensing or power-savings choices, and pairs of LEDs are placed in series to cut current usage in half. Each sensor has a digital I/O output that can be measured.

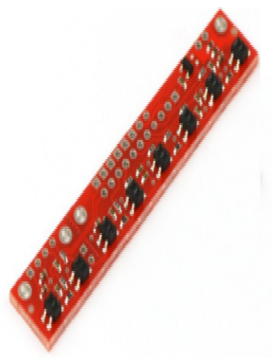


Figure 4.7: QTR-8RC Sensor[29]

4.5 Mechanical Parts

4.5.1 Omni Wheels

Omni wheels are used by several robots to allow them to move in all directions. Omni wheels are also used as powered casters for differential drive robots to help them turn more quickly. This design, however, is not widely utilized since it causes a vehicle handling issue. Omni wheels are based on the idea of a regular wheel with the capacity to 'slip' or roll sideways figure4.8. So, even though there is no drive in the lateral direction, the wheel can nevertheless move in that direction. Many smaller wheels or cylinders are placed on the edges of the main wheel to achieve this. The wheels are also known as Swedish 90 Wheels because of the angle of the smaller wheels in relation to the main wheel. Two wheels are frequently coupled to make a wheel with a more complete surface to avoid uncomfortable circumstances where a roller may not be in the right location and to minimize friction in lateral movement [30]. Omni-directional wheels are unique, since they can freely roll in both directions. Depending on the situation, they can either roll like normal wheels or sideways. It is possible to transform a non-holonomic robot into a holonomic robot using omni-directional wheels.

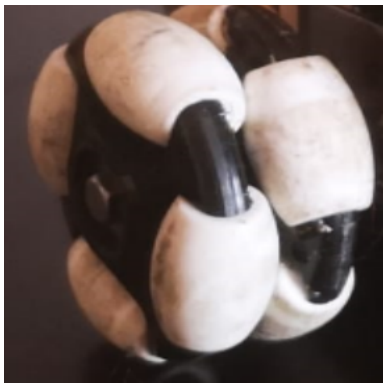


Figure 4.8: Omni Wheels

Specifications	Details
Diameter of the wheel	58 mm
Diameter of a bearing	13 mm
Load capacity	3Kg
Material	Nylon and plastic center

Table 4.4: Specifications of Omni Wheels

4.5.2 Motor Support

The motor support is shown in the figure4.9 providing easy mounting of the EMG30 to the robot. The bracket is made from a 2 mm thick strong aluminum and finished in blue color .

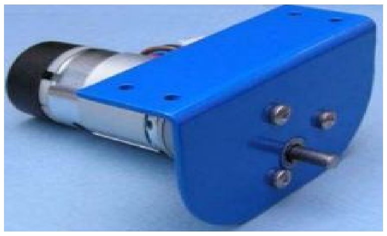


Figure 4.9: Motor Support

4.6 Power Supply

4.6.1 Lithium Batteries

Lithium batteries feature a metallic lithium anode and are used as primary batteries. Lithium-metal batteries are another name for these types of batteries. Their high charge density and expensive cost per unit set them apart from other batteries. Lithium cells can produce voltages ranging from 1.5 V to 3.7 V, depending on the design and chemical compounds utilized, in this project we used cylindrical lithium 5.2 cells because its ideal for robotics applications where space is limited and weight is in overall performance.

Figure 4.10: Lithium Batteries

4.6.2 Buck Convert Step-Down

The LM2596 DC-DC buck convert step-down 4.11 power module features a high-precision potentiometer and is capable of driving a load up to 3A with great efficiency. It can be used with the aduino UNO, as well as other mainboards and basic modules.

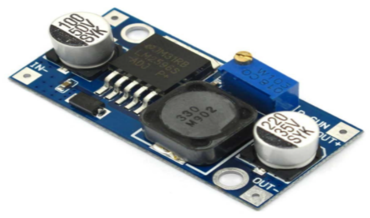


Figure 4.11: Buck Convert Step-Down

- Input voltage: 4.5 - 35V
- Output voltage: 1.5 - 35V
- Output current: Rated current is 2Ampere (A), maximum 3A.

4.7 Financial Study

The overall cost of the project is assessed in this study. It includes the cost of acquiring the various electronic components but excludes the cost of delivery. This table summarises the cost of the different components used. A thorough financial analysis will reveal whether the proposal is feasible and marketable, as well as whether it can be offered to potential

Components	Unity Cost (EUR/Unit)	Qty (Units)	Price (EUR)
Arduino Mega	35	1	35
Arduino Uno	20	1	20
EMG30	40.53	3	121.53
Motors Driver L298N	8.85	2	17.7
Color Sensor	8.9	1	8.9
Infrared Reflective Sensor	3.5	1	3.5
QTR-8RC Sensor	11.32	1	11.32
Omni Wheels	17.9	3	53.97
Motor Support	4.57	3	13.71
Lithium batteries	5.29	3	15.87
Buck Convert	2.46	1	2.46
TINKERKIT BRACCIO	247.23	1	247.23

Table 4.5: Robot Components Costs

investors in order to raise further funds for the prototype's industrial production. The overall bill for Robot components is estimated to be 551.19 euros. For a prototype this price is very high but we can reduce it by changing the kit braccio for another arm made by a 3D printer in the laboratory that allows us to lower more than 200 euros.

4.8 Conclusion

Knowing the various characteristics of these electronic components allows one to determine, on the one hand, the tension and current thresholds that they can support, but also, more importantly, their power consumption. Indeed, knowing the total consumption of the product is an important aspect of prototype design that should not be overlooked, and allows one to improve it later. According to the financial analysis, the design price is 551.19 Euros, which is now a high price in comparison to similar products.

Chapter 5

Results and Discussion

A crucial part of project design is implementation and realization. This step entails the concretization of the developed conceptual model, allowing for model confirmation on the one hand, as well as the discovery of previously undetected issues.

5.1 Power Supply Test

The power supply role is to provide the voltage for all the electrical element but the robot need two types of voltage to supply the circuit (12V for the motors and 5V for the Arduino cards and sensors). On the other hand, our battery provides a voltage of 12V so we used the buck step-down to lower the voltage to 5V. Figure 5.1 shows the result of the voltage after using the Buck converter

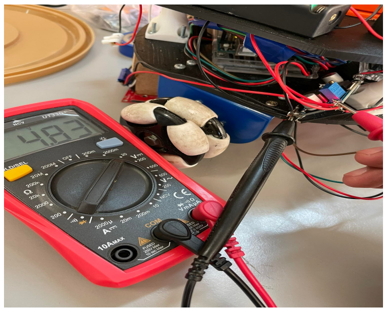


Figure 5.1: Buck Convert Test

5.2 Final Structure

For this robot, we have made the mechanical structure in plastic for all the parts of the mechanical structure. This material has some interesting properties: on the one hand, it is light and has an acceptable density and rigidity. On the other hand, it is available and has the ability to be machined well. The prototyping of the parts and their assembly were carried out manually. The figure 5.2 present the final result of robot.

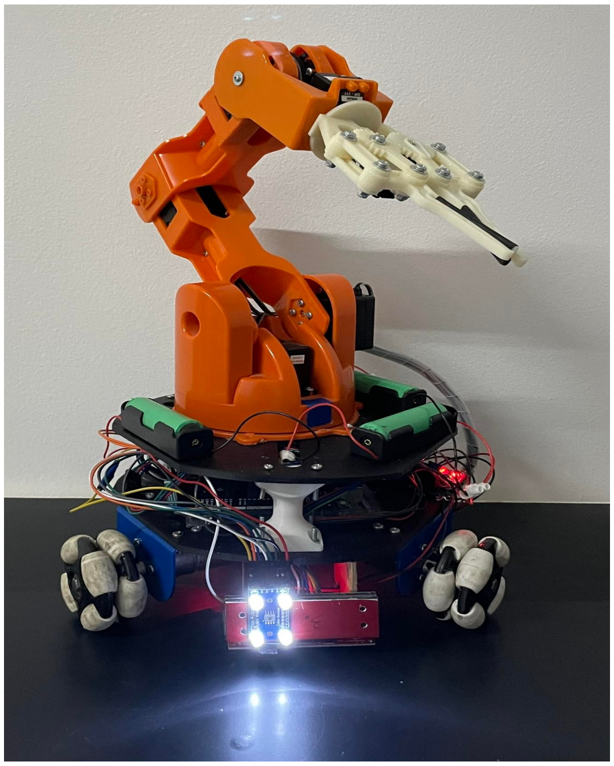


Figure 5.2: Robot Structure

5.3 Sensors Code and Practical Test

5.3.1 QTR Sensor

To follow the line with the stable motion of the robot, it is important to adjust the right KP,ki and kd value for the PID controller of the robot as indicated in the algorithm 1.The right value found with sampling in many real test of the robot motion on the black line and calibration of the QTR sensor.

Algorithm 1: QTR Sensor void Loop

```
                                        /*Main Line Follower Code with PID*/
void loop()
{
// read calibrated sensor values
// To get raw sensor values
// qtrrc.read(sensorValues); instead of unsigned int position =
  qtrrc.readLineBlack(sensor);
Position = qtrRC.readLineBlack(sensors);
int error_QTR = -3500 + Position;

int motorSpeed = $K_p$_QTR + $K_d$_QTR + (error_QTR - lastError_QTR);
lastError_QTR = error_QTR;

int rightMotorSpeed = 100 - motorSpeed;
int leftMotorSpeed = 100 + motorSpeed;

vitesseDemandeeM1 = constrain(rightMotorSpeed,0,200);
vitesseDEmandeeM2 = constrain(leftMotorSpeed, 0 ,200);
}
```

5.3.2 Color Sensor

TCS3200D outputs a square wave corresponding to light intensity and color, and the frequency is directly proportional to light intensity. Using the oscilloscope we can detect the frequency of the 3 color (red, blue and green) as we can see in this figure 5.3
The following code, shown in algorithm 2 is applied to read all the three colors. The algorithm collects, through the function ReadRGB(), the equivalent color in RGB format of the piece that is in front of the sensor, and identify through the functions ReadRed(), ReadGreen() and ReadBlue() the specific color frequency.

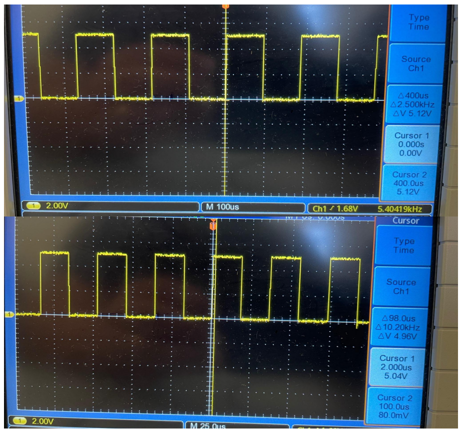


Figure 5.3: Square Wave Corresponding to Red and Blue Color

5.3.3 Infrared Reflective Sensor

The existence of an item within a certain range is detected using an infrared reflective sensor. An IR LED and a photosensor pair make up the sensor. Any item put in front of the sensor reflects the light generated by the IR LED, and the photosensor detects this reflection. Any surface that is white reflects more light than a surface that is black. The detected result can be checked by a signal indicator on the module, which will turn on the signal indicator, when the sensor is close to a barrier. It will turn off afterwards the signal, when the sensor will be away from the barrier as we can see in figure5.4.

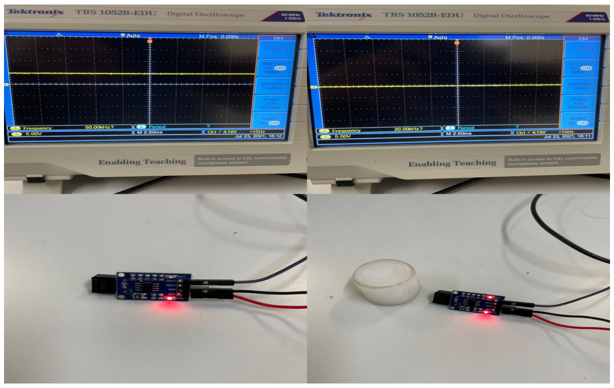


Figure 5.4: Test of Infrared Reflective

Algorithm 2: Color sensor code

```
void ReadRGB()
{
RGB[0] = ReadRed();
RGB[1] = ReadGreen();
RGB[2] = ReadBlue();
}
unsigned int ReadRed()
{
// Setting Red (R) filtered photodiodes to be read digitalWrite(s2, LOW);
digitalWrite(s3, LOW);
// Reading the output frequency unsigned int f = pulseIn(out,LOW);
// Remaping the value of the RED (R) frequency from 0 to 255 f =
 map(f,15,180,255,0);
return f;
}
unsigned int ReadGreen()
{
// Setting Green (G) filtered photodiodes to be read
digitalWrite(s2, HIGH);
digitalWrite(s3, HIGH);
// Reading the output frequency
unsigned int f = pulseIn(out,LOW);
// Remaping the value of the Green (G) frequency from 0 to 255
f = map(f,25,180,255,0);
return f;
}
unsigned int ReadBlue()
{
// Setting Bleu (B) filtered photodiodes to be read
digitalWrite(s2, LOW);
digitalWrite(s3, HIGH);
// Reading the output frequency
unsigned int f = pulseIn(out,LOW);
// Remaping the value of the Blue (B) frequency from 0 to 255
f = map(f,17,180,255,0);
return f;
}
```

5.4 EMG30 Control Code

The code represent This function will made the motor speed is exactly as we want
This figure 5.5 and 5.6 shows 2 curves Vt the target speed in blue and V current speed in red after increasing Kp and decreasing Ki the error was reduced so the response became faster when the motor get started and shutdown. The controller is able to reach zero speed and the target spped in 100 rpm.

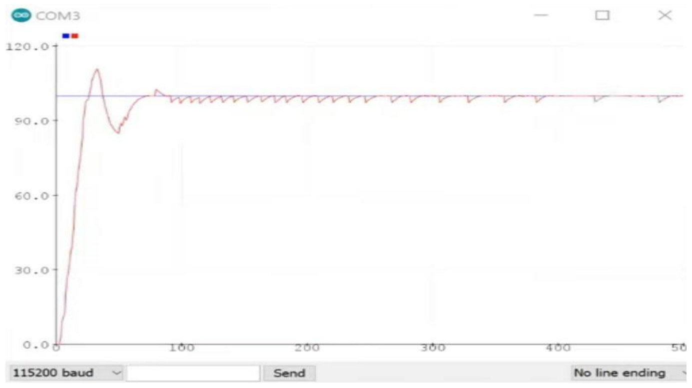


Figure 5.5: PID Measurment When The Motor Get Started

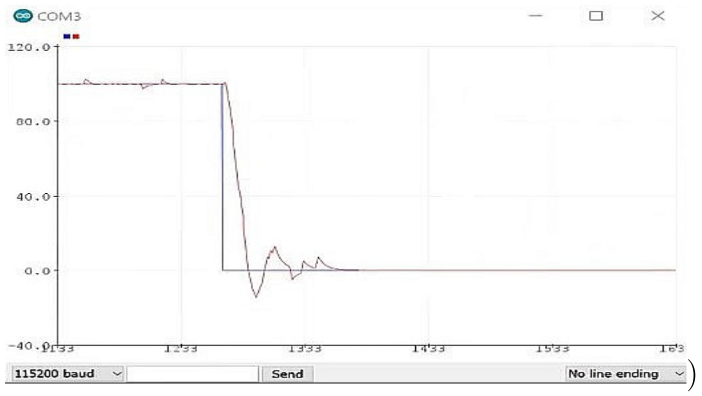


Figure 5.6: PID Measurment When The Motor Shutdown.

This algorithm has shown the method used to get the measurement by :
-Set a target and define Kp and Ki, the determination of these parameters can directly be obtained through experience and empirical experiments[31].
-Compute the error then the control signal u.

-Set the motor speed and direction and finally Serial plotter.

Algorithm 3: EMG30 Control Measurment Code

```
// Set a target
float vt = 100*(sin(currT/1e6)>0);
// Compute the control signal u
float kp = 5;
float ki = 10;
float e = vt-v1Filt;
eintegral = eintegral + e*deltaT;
float u = kp*e + ki*eintegral;
// Set the motor speed and direction
int dir = 1;
if (u<0){
dir = -1;
}
int pwr = (int) fabs(u);
if(pwr > 255){
pwr = 255;
}
setMotor(dir,pwr,ENA,IN1,IN2); Serial.print(vt);
Serial.print(" ");
Serial.print(v1Filt);
Serial.println();
delay(1);
}
}
```

5.5 Code

5.5.1 Library Used

In this project we have used 5 libraries in the programming to help us to make the simulations :

- digitalWriteFast.h To read faster the GPIO pins (needed for boath of encoder's chanels)

- FlexiTimer2.h This is the library of the timer2 that used to calculate periodically the motor's angular speed
- QTRSensors.h This is the library of the QTR sensors
- Servo.h This is the library used to control servo motors
- Braccio.h this library used to control the arm braccio

Algorithm 4: Arduino Library

```
#include<digitalWriteFast.h>//To read faster the GPIO pins (needed for boath
  of encoder's chanels)
#include<FlexiTimer2.h>// This is the library of the timer2 that we use to
  calculate periodically the motor's angular speed
#include<QTRSensors.h>// This is the library of the QTR sensors
```

5.5.2 Interruption

An interrupt occurs when a microprocessor is forced to do a different task for a brief amount of time before returning to its original function. In this part of the work we show the interruption declaration for 3 channesl of all motor's encoder for rising edge

Algorithm 5: Code for Interruption

```
// Interupt declarations
  attachInterrupt(digitalPinToInterrupt(Motor1SensorA),CallBack1,RISING);
attachInterrupt(digitalPinToInterrupt(Motor2SensorA),CallBack2,RISING);
attachInterrupt(digitalPinToInterrupt(Motor3SensorA),CallBack3,RISING);
attachInterrupt(digitalPinToInterrupt(SensorPin),CallBack3,FALLING);
```

5.5.3 Serial Communication

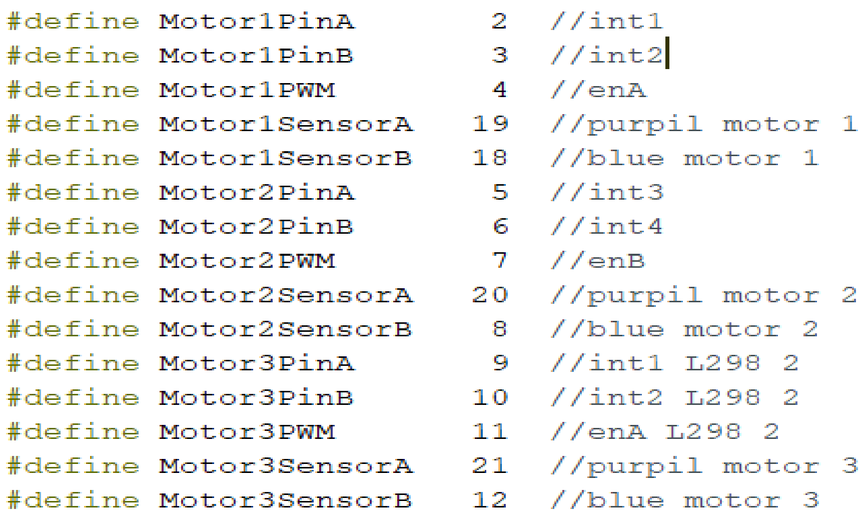
The serial.begin used to communicate with the other Arduino (Boath of them must have the same baudrate) by send or resive one lettre

Algorithm 6: Code for Serial Communication

```
void setup() {
Serial.begin(9600); Braccio.begin(); }
void loop() {
if (Serial.available() > 0) { frame = Serial.read();
GoToAssociatedPosition(DetectColor(frame));
Serial.println(DetectColor(frame));
frame = "";
}}
```

5.5.4 Pins Declaration

In this part of the code we have made the I/O pins declarations for all the electrical elements of our robot. This figure shows the pind declaration for the 3 motors



```
#define Motor1PinA        2   //int1
#define Motor1PinB        3   //int2
#define Motor1PWM         4   //enA
#define Motor1SensorA    19   //purpil motor 1
#define Motor1SensorB    18   //blue motor 1
#define Motor2PinA        5   //int3
#define Motor2PinB        6   //int4
#define Motor2PWM         7   //enB
#define Motor2SensorA    20   //purpil motor 2
#define Motor2SensorB     8   //blue motor 2
#define Motor3PinA        9   //int1 L298 2
#define Motor3PinB       10   //int2 L298 2
#define Motor3PWM        11   //enA L298 2
#define Motor3SensorA    21   //purpil motor 3
#define Motor3SensorB    12   //blue motor 3
```

Figure 5.7: Pins Declaration

5.6 Electronic Circuit

5.6.1 Electrical Wiring

The test plate is a perforated insulated plastic plate with many holes. These holes are 2.54 mm apart, which is the industry standard for the electronic components used in the assembly. This is an excellent method for testing a solderless assembly, adjusting what has to be altered, and correcting various design and sizing issues. Figure 5.8 shows the robot prototype wiring.

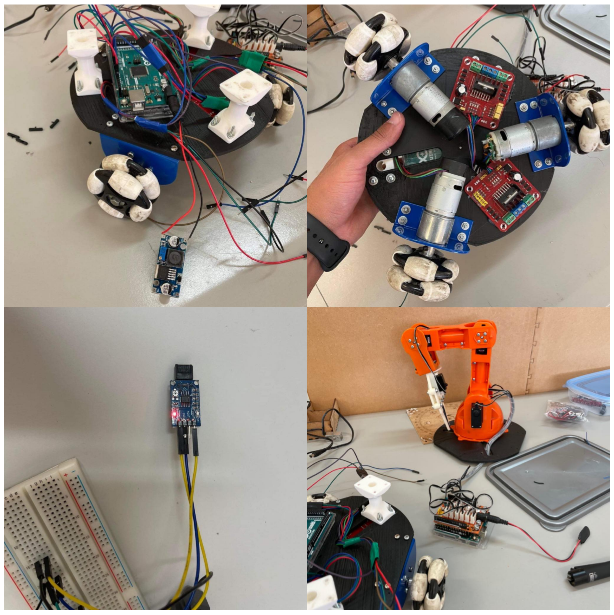


Figure 5.8: Electrical Wiring

5.6.2 Printed Circuit Design

Various software packages such as Eagle, EasyEDA, ARES provide access to a PCB development interface, It is important that the various components are positioned so as to minimise the occupied space. Some software offer the possibility automatic routing feature that allows the different components to be connected in a few seconds, but it is much more efficient to do it manually, which is what was chosen.

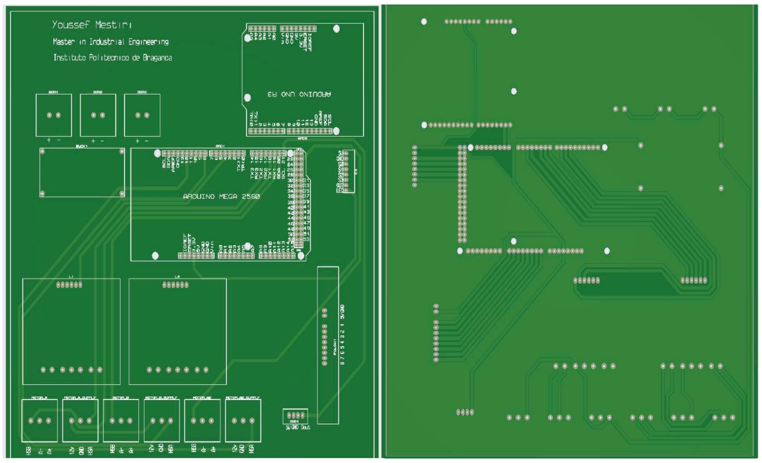

Youssef Mestiri
Master in Industrial Engineering
Instituto Politecnico de Braganca
ARDUINO UNO R3
ARDUINO MEGA 2560

Figure 5.9: Robot Structure

5.7 Conclusion

One important phase of the project design is the code and test. This phase consists of the concretization of the created conceptual model and allows a confirmation of the model on the one hand, but also to find various problems which were not detected.

Chapter 6

General Conclusion and Future Work

The work presented in this project is part of a design and imptementation of an omnidirectional 3 wheeled manipulator robot. This robot can make various movements and this thanks to the use of arduino module.
To reach the end of our work, we have organized our project in 4 chapters: The first chapter covers broad aspects of robots as well as the constituent pieces of these robots based on their mechanical construction. In general, a mobile manipulator robot can be thought of as a generator of movements and efforts in multiple directions in space. The second chapter explains System Design and Technical Analysis. The latter is a set of models represented by a schema that depicts and describes our system's operation. It is a set of up of various parts, the most important of which are:Power supply,PID control,serial communication and programing with interruption. The third chapter depicts all of the electrical and mechanical elements employed in our robot in order to comprehend their function, mode of operation, and characteristics. A cost report was also included at the end of this chapter. The fourth chapter is a practical study that explains the design, the realization of our robot. On the basis of the knowledge and information of the previous chapters, For the controlled part, in this work and the elaboration of the model :

- The realization of the mechanical structure.
- Programming with the arduino module

Future Work

The future work will try to improve our project and make them more efficient and usable by :

- The addition of more sophisticated sensors such as a camera for example can be considered for a visual control of the robot.

- Planning of the different trajectories for this manipulator arm for the different operations

- Design a specific controller other than the UMAC interface or the implementation of virtual interface allowing to control the robot and to know its position in real time.

Bibliography

[1] Hisour. (Consulted on august 2021). Mobile manipulator, [Online]. Available: `https://www.hisour.com/mobile-manipulator-42888/`.

[2] Mckinsey. (consulted on the 10 mars 2021). Automation, robotics, and the factory of the future, [Online]. Available: `https://www.mckinsey.com/business-functions/operations/our-insights/automation-robotics-and-the-factory-of-the-future`.

[3] I. F. of Robotics, *Global industrial robot sales doubled over the past five years*, `https://ifr.org/news/global-industrial-robot-sales-doubled-over-the-past-five-years/`, 2018.

[4] R. Robotics, *The Five Keys to Mobile Manipulation.* RE2 Robotics, 2020.

[5] I. Spectrum. (23 septembre). How the u.s. army is turning robots into team players, [Online]. Available: `https://spectrum.ieee.org/ai-army-robots`.

[6] R. Robotics, *The Five Keys to Mobile Manipulation.* RE2 Robotics, 2020.

[7] Pinterest. (consulted on the 10 mars 2021). Robobuilder: Build multiple robots with one kit, [Online]. Available: `https://www.pinterest.com/pin/313140980313471127/`.

[8] R. Costanzi, F. Fanelli, N. Monni, A. Ridolfi, and B. Allotta, “An attitude estimation algorithm for mobile robots under unknown magnetic disturbances”, *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 4, pp. 1900–1911, 2016.

[9] ETHzurich. (consulted on the 30 mars 2021). Robotic systems lab, [Online]. Available: `https://rsl.ethz.ch/research/researchtopics/mobile-manipulation.html`.

[10] L. S. Lopes and J. H. Connell, "Semisentient robots: Routes to integrated intelligence", *IEEE Intelligent Systems*, vol. 16, no. 5, pp. 10–14, 2001.

[11] B. Kelly, J. Padayachee, and G. Bright, "Quasi-serial manipulator for advanced manufacturing systems.", in *ICINCO (2)*, 2019, pp. 300–305.

[12] N. Hacene and B. Mendil, "Motion analysis and control of three-wheeled omnidirectional mobile robot", *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 2, pp. 194–213, 2019.

[13] S. Hu *et al.*, "Triangular omnidirectional wheel motion control system", *Open Access Library Journal*, vol. 7, no. 08, p. 1, 2020.

[14] J. Gonçalves, J. Lima, H. Oliveira, and P. Costa, "Sensor and actuator modeling of a realistic wheeled mobile robot simulator", in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, IEEE, 2008, pp. 980–985.

[15] Arduino, *Braccio quick start guide*, Arduino, 2013.

[16] ——, *Tinkerkit*, Arduino, 2013.

[17] Rowan. (August 29, 2021). Arduino (tinkerkit) braccio robot arm + kinematics, [Online]. Available: `https://www.hackster.io/rpatterson/arduino-tinkerkit-braccio-robot-arm-kinematics-1a8303`.

[18] H. drive SE. (consulted on the 19 mars 2021). 'la robotique mobile', [Online]. Available: `https://harmonicdrive.de/fr/glossaire/la-robotique-mobile`.

[19] S. Roberts, *DC/DC BOOK OF KNOWLEDGE*. RECOM Engineering GmbH & Co KG, 2020.

[20] S. R. M. B.Sc, *DC/DC BOOK OF KNOWLEDGE Practical tips for the User*, 1st. Münzfeld 35, 4810 Gmunden, Austria.

[21] D. Nedelkovski. (consulted on April 2021). L298n motor driver – arduino interface, how it works, codes, schematics, [Online]. Available: `https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/`.

[22] Arduino. (consulted on April 2021). Attachinterrupt(), [Online]. Available: `https://fr.overleaf.com/learn/latex/List_of_Greek_letters_and_math_symbols`.

[23] Electroniclinic. (consulted on April 2021). Serial communication between two arduino boards, [Online]. Available: `https://www.electroniclinic.com/serial-communication-between-two-arduino-boards/`.

[24] N. Dunbar, "Alternatives to the arduino ide", in *Arduino Software Internals*, Springer, 2020, pp. 273–340.

[25] Arduino. (consulted on April 2021). Arduino mega 2560, [Online]. Available: `https://store.arduino.cc/products/arduino-mega-2560-rev3`.

[26] Electroya. (consulted on 7 April 2021). Contrôleur / contrôleur double pour moteurs pas à pas et à courant continu - l298n, [Online]. Available: `https://www.electroya.com/pt/produto/driver-de-driver-duplo-para-motores-de-passo-e-dc-l298n/`.

[27] TAOS, *Color sensor user manual*, The LUMENOLOGY Company, JULY 2009.

[28] ——, *Infrared reflective sensor user manual*, nagasm, JULY 2009.

[29] PTROBOTICS. (consulted on the 10 mars 2021). Qtr-src sensor, [Online]. Available: `https://www.ptrobotics.com/sensores-opticos/1224-qtr-8a-reflectance-sensor-array.html?gclid=EAIaIQobChMIspXyis_g9AIVCYjVCh2QEQMwEAQYAiABEgIoZ_D_BwE`.

[30] J. H. S. Soni T. Mistry, "Experimental analysis of mecanum wheel and omni wheel", in *International Journal of Innovative Science, Engineering Technology*, 2014, pp. 292–295.

[31] K. Tan, T. H. Lee, and H. X. Zhou, “Micro-positioning of linear-piezoelectric motors based on a learning nonlinear pid controller”, *IEEE/ASME transactions on mechatronics*, vol. 6, no. 4, pp. 428–436, 2001.

Appendix A

Code Arduino Uno

```
#include <Servo.h>
#include <Braccio.h>

#define delayBetweenTasks 1000

Servo base;                                                    // 0°to 180°
Servo shoulder;                                               // 15°to 165°
Servo elbow;                                                   // 0°to 180°
Servo wrist_ver;                                               // 0°to 180°
Servo wrist_rot;                                               // 0°to 180°
Servo gripper;                        // 10°to 73°(10°for Open 73°for Close)

unsigned char M1 = 90, M2 = 45, M3 = 180, M4 = 180, M5 = 90, M6 = 10;
unsigned char frame = "", lastFrame = "";

typedef enum Color

RED = 0,
GREEN,
BLUE,
UNKOWN,
;

void setup()

Serial.begin(9600);
Braccio.begin();


void loop()

if (Serial.available() > 0)

frame = Serial.read();
GoToAssociatedPosition(DetectColor(frame));
Serial.println(DetectColor(frame));

frame = "";



void DoRedJob()

// Go to the initial box position
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Maintain the box
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 73);
```

```
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Turn to the Red destination
Braccio.ServoMovement(20, 180, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Move down the arm
Braccio.ServoMovement(20, 180, 60, 130, 160, 0, 73);
delay(delayBetweenTasks);
// Release the Box
Braccio.ServoMovement(20, 180, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 10);
delay(delayBetweenTasks);

void DoGreenJob()

// Go to the initial box position
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Maintain the box
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 73);
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Turn to the Green destination
```

```
Braccio.ServoMovement(20, 135, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Move down the arm
Braccio.ServoMovement(20, 135, 60, 130, 160, 0, 73);
delay(delayBetweenTasks);
// Release the Box
Braccio.ServoMovement(20, 135, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 10);
delay(delayBetweenTasks);

void DoBlueJob()

// Go to the initial box position
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Maintain the box
Braccio.ServoMovement(20, 0, 60, 130, 160, 0, 73);
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Turn to the Blue destination
Braccio.ServoMovement(20, 90, 90, 90, 160, 0, 73);
delay(delayBetweenTasks);
// Move down the arm
Braccio.ServoMovement(20, 90, 60, 130, 160, 0, 73);
delay(delayBetweenTasks);
```

```
// Release the Box
Braccio.ServoMovement(20, 90, 60, 130, 160, 0, 10);
delay(delayBetweenTasks);
// Move up the arm
Braccio.ServoMovement(20, 0, 90, 90, 160, 0, 10);
delay(delayBetweenTasks);



Color DetectColor(unsigned char s)

return s == 'R' ? RED :
(s == 'B' ? BLUE :
(s == 'G' ? GREEN : UNKOWN));



void GoToAssociatedPosition(Color C)

switch (C)

case RED:
DoRedJob();
break;
case GREEN:
DoGreenJob();
break;
case BLUE:
DoBlueJob();
break;
default:
```

```
break;

void GoToInitPosition()

Braccio.ServoMovement(20, 0, 60, 130, 160, 90, 10);
```

Appendix B

Code Arduino Mega 2560

```
#include <digitalWriteFast.h>
#include <FlexiTimer2.h>
#include <QTRSensors.h>
#define SensorPin 21
#define s0 25
#define s1 27
#define s2 29
#define s3 31
#define out 33
// These defines are for enabling the appropriate test
#define TEST_MOTORS
#define testQTR
——————-QTR——————-
#define NUM_SENSORS 8
QTRSensors qtrRC;
unsigned int sensors[NUM_SENSORS];
int Position;
int lastError_QTR = 0;
double Kp_QTR = 0.1;
double Kd_QTR = 0.01;
———- MOTOR PINS DECLARATION———
#define Motor1PinA  2                                   //int1
#define Motor1PinB  3                                   //int2
#define Motor1PWM  4                                    //enA
#define Motor1SensorA  19                    //purpil motor 1
#define Motor1SensorB  18                      //blue motor 1
#define Motor2PinA  5                                   //int3
#define Motor2PinB  6                                   //int4
#define Motor2PWM  7                                    //enB
#define Motor2SensorA  20                    //purpil motor 2
#define Motor2SensorB  8                       //blue motor 2
#define Motor3PinA  9                             //int1 L298 2
#define Motor3PinB  10                            //int2 L298 2
#define Motor3PWM  11                             //enA L298 2
#define Motor3SensorA  21                    //purpil motor 3
#define Motor3SensorB  12                      //blue motor 3
———— Motors  PID  related  variables ————
#define diametre_roue 60
#define resolution_encodeur 90
#define pi 3.141592
#define TSDATA 100
#define cadenceMs 10
volatile long Sensor1Counter = 0;
volatile long Sensor2Counter = 0;
———- ENUM  DECLARATION ———
enum  Direction
{
ALL_BLACK,
ALL_WHITE,
GO_FORWARD,
SLIGHT_LEFT,
LEFT,
URGENT_LEFT,
SLIGHT_RIGHT,
RIGHT,
URGENT_RIGHT,
UNKOWN,
};
——- STRUCT  DECLARATION ——-
typedef  struct  threshold
{
unsigned char R;
unsigned char G;
unsigned char B;
} T;
typedef struct Color
{
threshold Threshold;
} C;
——-GLOBAL VARIABLE——-
int frequency = 0;
Color Red, Green, Blue;
unsigned int RGB[3];
unsigned char frame = "";
bool DoAction = false;
volatile long c1 = 0;
volatile long c2 = 0;
```

```
double tensionBatterie = 12.0;
volatile double vitesseDemandeeM1 = 0 ;
volatile double vitesseDemandeeM2 = 0 ;
unsigned int PWM3 = 0;
int sensMotor1 = -1;
int sensMotor2 = 1;
int sensMotor3 = -1;
unsigned long tempsDernierEnvoi = 0;
unsigned long tempsCourant = 0;
volatile double dt = cadenceMs / 1000.;
volatile double temps = -cadenceMs / 1000.;
volatile double omega1, omega2;
volatile double commande1 = 0., commande2 = 0.;
———- PID  VARIABLE———-
volatile double Kp = 0.3;
volatile double Ki = 9.6;
volatile double P = 0.;
volatile double I1 = 0., I2 = 0., I3 = 0.;
volatile double erreur = 0.;
volatile double v;
volatile uint8_t timer100 = 0;
———- SETUP FUNCTION ———-
void setup()
‘{

Serial.begin(9600);


Serial3.end();
Initialization();
AdjustThresholds();
qtrRC.setTypeRC();
qtrRC.setSensorPins((const uint8_t[])
{
A0, A1, A2, A3, A4, A5, A6, A7

}, NUM_SENSORS);                                    // Declaring the QTR used pins

// Interrupt declarations

attachInterrupt(digitalPinToInterrupt(Motor1SensorA), CallBack1, RISING);
attachInterrupt(digitalPinToInterrupt(Motor2SensorA), CallBack2, RISING);
attachInterrupt(digitalPinToInterrupt(SensorPin), CallBack3, FALLING);
// Timer declaration and starter (the period is 20ms)
FlexiTimer2::set(20, AsservissementVitesse);
FlexiTimer2::start();
delay(2000);
// Calibrate the QTR
for (int i = 0; i < 100; i++)
{
qtrRC.calibrate();
delay(20);
}
vitesseDemandeeM1 = 50 ;
vitesseDemandeeM2 = 50 ;
// We incomment the Test function if we want to test or calibrate our robot
//Test();
}
———- INFINIT FUNCTION ———-
```

```
void loop()
{
// ecritureData();
———-Main Line Follower Code with PID———-
Position = qtrRC.readLineBlack(sensors);
int error_QTR = - 3500 + Position;
int motorSpeed = Kp_QTR * error_QTR + Kd_QTR * (error_QTR - lastError_QTR);
lastError_QTR = error_QTR;
int rightMotorSpeed = 100 - motorSpeed;
int leftMotorSpeed = 100 + motorSpeed;
vitesseDemandeeM1 = constrain(rightMotorSpeed , 0, 200);
vitesseDemandeeM2 = constrain(leftMotorSpeed , 0, 200);
} ———- FUNCTION DECLARATION ———-
// Pin State declaration function
void Initialization() // Setting the mode of each used pin
{
pinMode(Motor1PinA, OUTPUT);

pinMode(Motor1PinB, OUTPUT);

pinMode(Motor1PWM, OUTPUT);

pinMode(Motor2PinA, OUTPUT);

pinMode(Motor2PinB, OUTPUT);

pinMode(Motor2PWM, OUTPUT);

pinMode(Motor3PinA, OUTPUT);
pinMode(Motor3PinB, OUTPUT);

pinMode(Motor3PWM, OUTPUT);

pinMode(Motor1SensorA, INPUT_PULLUP);

pinMode(Motor1SensorB, INPUT_PULLUP);

pinMode(Motor2SensorA, INPUT_PULLUP);

pinMode(Motor2SensorB, INPUT_PULLUP);

pinMode(Motor3SensorA, INPUT_PULLUP);

pinMode(Motor3SensorB, INPUT_PULLUP);

pinMode(IRsensor1, INPUT);

pinMode(IRsensor2, INPUT);

pinMode(IRsensor3, INPUT);

pinMode(IRsensor4, INPUT);

pinMode(IRsensor5, INPUT);

pinMode(s0, OUTPUT);
pinMode(s1, OUTPUT);

pinMode(s2, OUTPUT);

pinMode(s3, OUTPUT);

pinMode(out, INPUT);

pinMode(SensorPin, INPUT_PULLUP);

// To use properly the color sensor, we put S0 High ans S1 LOW

digitalWrite(s0, HIGH);

digitalWrite(s1, LOW);

}
// Test Function
void Test()
{
#ifdef testQTR
while (1)
{
Position = qtrRC.readLineBlack(sensors);
for (int i = 0; i < NUM_SENSORS; i++)
{
Serial.print(sensors[i]);
if (i == NUM_SENSORS - 1) Serial.println(" | ");
else
{
Serial.print("(");
Serial.print(i);
Serial.print(") | ");
}
delay(20);
}
Serial.println(Position);
}
#endif
#ifdef TEST_MOTORS
while (true)
{
MoveMotor1(255);
delay(1000);
MoveMotor1(-255);
delay(1000);
StopMotor1();
MoveMotor2(255);
delay(1000);
MoveMotor2(-255);
delay(1000);
StopMotor2();
MoveMotor3(255);
delay(1000);
MoveMotor3(-255);
delay(1000);
StopMotor3();
}
```

```
#endif
}
// Call Back for Motor1 interruption
void CallBack1()
{
// If the logic level of the two channels are equal so the sens in 1 and we increase the
accimulated variable, else the motor sens is 2 and we decrise that variable
Sensor1Counter -=
(digitalReadFast(Motor1SensorA) == digitalReadFast(Motor1SensorB)) ?
1 : (-1);
c1 -=
(digitalReadFast(Motor1SensorA) == digitalReadFast(Motor1SensorB)) ?
1 : (-1);
}
// Call Back for Motor2 interruption
void CallBack2()
{
Sensor2Counter -=
(digitalReadFast(Motor2SensorA) == digitalReadFast(Motor2SensorB)) ?
1 : (-1);
c2 -=
(digitalReadFast(Motor2SensorA) == digitalReadFast(Motor2SensorB)) ?
1 : (-1);
}
void CallBack3()
{
DoAction = true;
}
// Periodic function to read color
```

```
void ReadColor()
{
if (DoAction)
{
DoAction = false;
ReadRGB();
frame = WhatIsTheColor(RGB);
Serial.write(frame);
Serial.write("\n");
Serial1.write(frame);
//Serial1.write("\n");
RGB[0] = 0;
RGB[1] = 0;
RGB[2] = 0;
frame = "";
}
}
// Detect action needed based on direction order
void PerformMove(Direction dir)
{
switch (dir)
{
case ALL_BLACK :
Serial.println("ALL_BLACK");
break;
case ALL_WHITE :
Serial.println("ALL_WHITE");
break;
case GO_FORWARD :
```

```
Serial.println("GO_FORWARD");
GoForward();
break;
case SLIGHT_LEFT:
Serial.println("SLIGHT_LEFT");
GoSlightLeft();
break;
case LEFT:
Serial.println("LEFT");
GoLeft();
break;
case URGENT_LEFT:
Serial.println("URGENT_LEFT");
GoUrgentLeft();
break;
case SLIGHT_RIGHT:
Serial.println("SLIGHT_RIGHT");
GoSlightRight();
break;
case RIGHT:
Serial.println("RIGHT");
GoRight();
break;
case URGENT_RIGHT:
Serial.println("URGENT_RIGHT");
GoUrgentRight();
break;
default:
break;
```

```
}
}
// Detect all direction orders from sensors values
Direction DecodeDirection(bool *SensorsState)
{ return (AllBlack(SensorsState) ? ALL_BLACK :
(AllWhite(SensorsState) ? ALL_WHITE :
(ForwardOrder(SensorsState) ? GO_FORWARD :
(SlightLeft(SensorsState) ? SLIGHT_LEFT :
(Left(SensorsState) ? LEFT :
(UrgentLeft(SensorsState) ? URGENT_LEFT :
(SlightRight(SensorsState) ? SLIGHT_RIGHT :
(Right(SensorsState) ? RIGHT :
(UrgentRight(SensorsState) ? URGENT_RIGHT : UNKOWN)))))))));
}
//Read Sensor states and store then in an array
void ReadSensorsStates()
{ Sensors[0] = digitalRead(IRsensor1);
Sensors[1] = digitalRead(IRsensor2);
Sensors[2] = digitalRead(IRsensor3);
Sensors[3] = digitalRead(IRsensor4);
Sensors[4] = digitalRead(IRsensor5);
}
bool AllBlack(bool *SensorsState)
{
// [1] [1] [1] [1] [1]
return (!SensorsState[0] && !SensorsState[1]
&& !SensorsState[2] && !SensorsState[3] && !SensorsState[4]);
}
bool AllWhite(bool *SensorsState)
```

```
{
// [0] [0] [0] [0] [0]
return (SensorsState[0] && SensorsState[1]
&& SensorsState[2] && SensorsState[3] && SensorsState[4]);
}
bool ForwardOrder(bool *SensorsState)
{
// [1] [1] [0] [1] [1]
// [1] [0] [0] [0] [1]
return (SensorsState[0] && !SensorsState[2] && SensorsState[4]);
}
bool SlightLeft(bool *SensorsState)
{
// [1] [0] [0] [1] [1]
// [1] [0] [1] [1] [1]
return (SensorsState[0] && !SensorsState[1]
&& SensorsState[3] && SensorsState[4]);
}
bool Left(bool *SensorsState)
{
// [0] [0] [1] [1] [1]
// [0] [0] [0] [1] [1]
return (!SensorsState[0] && !SensorsState[1]
&& SensorsState[3] && SensorsState[4]);
}
bool UrgentLeft(bool *SensorsState)
{
// [0] [1] [1] [1] [1]
return (!SensorsState[0] && SensorsState[1]
&& SensorsState[2] && SensorsState[3] && SensorsState[4]);
}
bool SlightRight(bool *SensorsState)
{
// [1] [1] [0] [0] [1]
// [1] [1] [1] [0] [1]
return (SensorsState[0] && SensorsState[1]
&& !SensorsState[3] && SensorsState[4]);
}
bool Right(bool *SensorsState)
{
// [1] [1] [1] [0] [0]
// [1] [1] [0] [0] [0]
return (SensorsState[0] && SensorsState[1]
&& !SensorsState[3] && !SensorsState[4]);
}
bool UrgentRight(bool *SensorsState)
{
// [1] [1] [1] [1] [0]
return (SensorsState[0] && SensorsState[1]
&& SensorsState[2] && SensorsState[3] && !SensorsState[4]);
}
void MoveMotor1(int val)
{
constrain(val, -255, 255); // Limit the value of PWM between -255 and 255
if (val >= 0) // if the value is positif so the sens in 1
{
digitalWrite(Motor1PinA, HIGH);
digitalWrite(Motor1PinB, LOW);
```

```
analogWrite(Motor1PWM, val);
}
else // if the PWM value is negatif so the sens is 2
{
digitalWrite(Motor1PinA, LOW);
digitalWrite(Motor1PinB, HIGH);
analogWrite(Motor1PWM, abs(val));
}
}
void MoveMotor2(int val)
{
constrain(val, -255, 255);
if (val >= 0)
{
digitalWrite(Motor2PinA, HIGH);
digitalWrite(Motor2PinB, LOW);
analogWrite(Motor2PWM, val);
}
else
{
digitalWrite(Motor2PinA, LOW);
digitalWrite(Motor2PinB, HIGH);
analogWrite(Motor2PWM, abs(val));
}
}
void MoveMotor3(int val)
{
constrain(val, -255, 255);
if (val >= 0)
```

```
{
digitalWrite(Motor3PinA, HIGH);
digitalWrite(Motor3PinB, LOW);
analogWrite(Motor3PWM, val);
}
else
{
digitalWrite(Motor3PinA, LOW);
digitalWrite(Motor3PinB, HIGH);
analogWrite(Motor3PWM, abs(val));
}
}
void StopMotor1()
{
digitalWrite(Motor1PinA, LOW);
digitalWrite(Motor1PinB, LOW);
analogWrite(Motor1PWM, 0);
}
void StopMotor2()
{
digitalWrite(Motor2PinA, LOW);
digitalWrite(Motor2PinB, LOW);
analogWrite(Motor2PWM, 0);
}
void StopMotor3()
{
digitalWrite(Motor3PinA, LOW);
digitalWrite(Motor3PinB, LOW);
analogWrite(Motor3PWM, 0);
```

```
}
void StopRobot()
{
StopMotor1();
StopMotor2();
StopMotor3();
}

void GoForward(uint8_t vitesse)
{
MoveMotor1(vitesse);
MoveMotor2((-1) * vitesse);
}
void GoBack(uint8_t vitesse)
{
MoveMotor1((-1) * vitesse);
MoveMotor2(vitesse);
}
void GoSlightLeft(uint8_t vitesse)
{
MoveMotor1(vitesse + 20);
MoveMotor2((-1) * vitesse);
}
void GoLeft(uint8_t vitesse)
{
MoveMotor1(vitesse);
MoveMotor2((-1) * (vitesse + 50));
}
void GoUrgentLeft(uint8_t vitesse)
```

```
{
MoveMotor3(vitesse);
}
void GoSlightRight()
{
MoveMotor1(vitesse);
MoveMotor2((-1) * (vitesse + 20));
}
void GoRight()
{
MoveMotor1(vitesse + 50);
MoveMotor2((-1) * vitesse);
}
void GoUrgentRight()
{
MoveMotor3((-1) * vitesse);
}
void AsservissementVitesse(void)
{
int codeurDeltaPos1, codeurDeltaPos2, codeurDeltaPos3;
codeurDeltaPos1 = c1;
c1 = 0;
omega1 = ((2 * pi * ((double)codeurDeltaPos1)) / resolution_encodeur) / dt;
//convertion from round/min to rad/s
v = (vitesseDemandeeM1 * pi) / diametre_roue;
// PID Correction
erreur = v - omega1;
P = Kp * erreur;
commande1 = P + I1;
```

```
I1 = I1 + Ki * dt * erreur;
CommandeMoteur(1, commande1, tensionBatterie);
codeurDeltaPos2 = c2;
c2 = 0;
omega2 = ((2 * pi * ((double)codeurDeltaPos2)) / resolution_encodeur) / dt;
//convertion from round/min to rad/s
v = (vitesseDemandeeM2 * pi) / diametre_roue;
//Correction PID
erreur = v - omega2;
P = Kp * erreur;
commande2 = P + I2;
I2 = I2 + Ki * dt * erreur;
CommandeMoteur(2, commande2, tensionBatterie);

   MoveMotor3(sensMotor3 * PWM3);
temps += dt;

   if (timer100++ >= 5)
timer100 = 0;
{
ReadColor();
}
}
void ecritureData(void)
{
double o = 0;
o = (diametre_roue * omega1) / pi;
tempsCourant = millis();
if (tempsCourant - tempsDernierEnvoi > TSDATA)
```

```
{
Serial.print(temps);

Serial.print(" , ");

Serial.print(omega1);

Serial.print("\r");

Serial.print("\n");

tempsDernierEnvoi = tempsCourant;
} } void CommandeMoteur(int moteur, double tension, double tensionBatterie)
{
int tensionPWM;
tensionPWM = (int)(255 * (tension / tensionBatterie));
constrain(tensionPWM, -255, 255);
if (moteur == 1) MoveMotor1(sensMotor1 * tensionPWM);
else if (moteur == 2) MoveMotor2(sensMotor2 * tensionPWM);
}
void PrintRGB()
{
Serial.print("R:"); Serial.print(RGB[0]);
Serial.print(" G:"); Serial.print(RGB[1]);
Serial.print(" B:"); Serial.println(RGB[2]);
}
void AdjustThresholds() // Adjust filter for Red, Green and Blue colors
{
RedFilter();
```

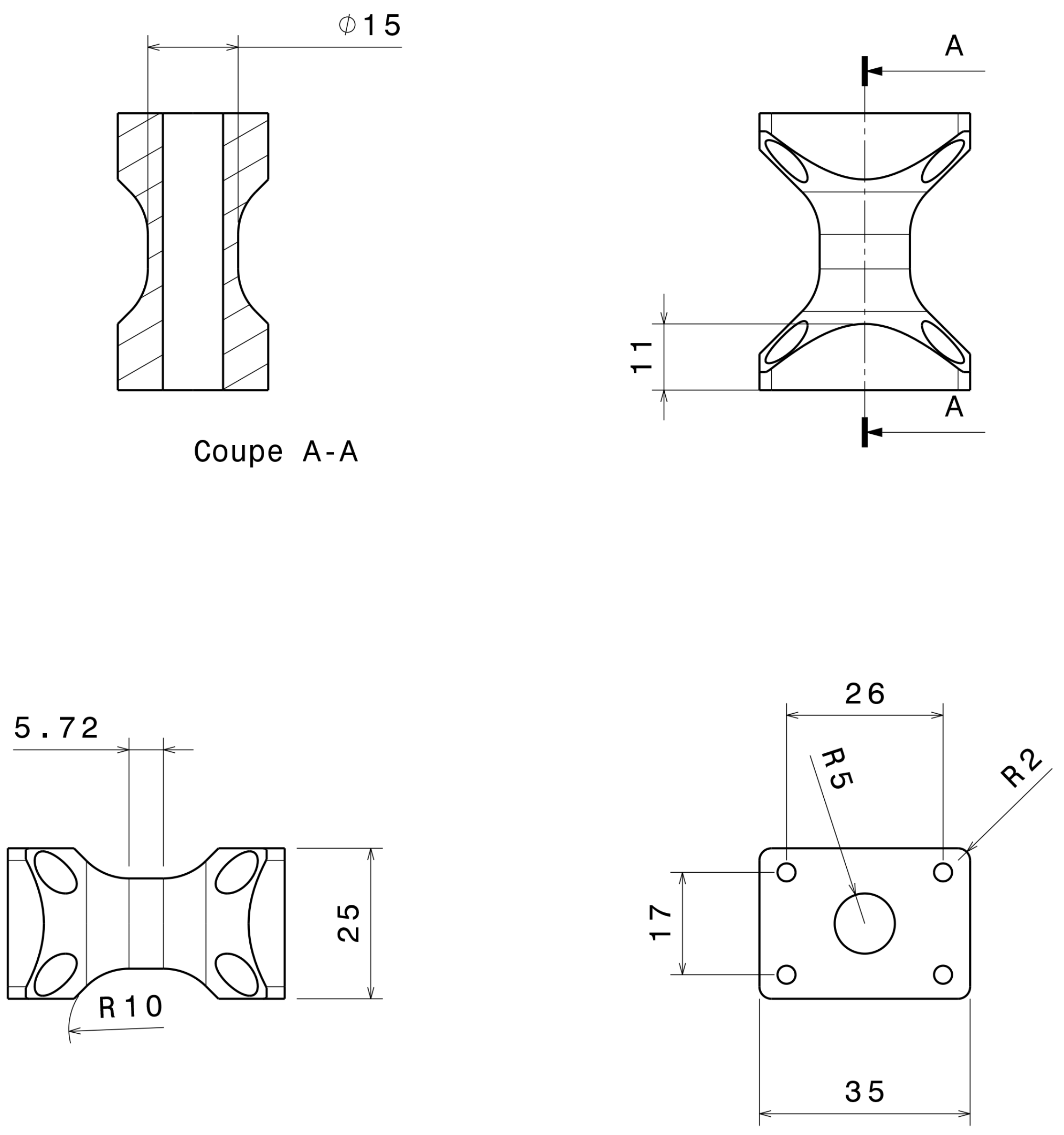
```
GreenFilter();
BlueFilter();
}
void RedFilter()
{
Red.Threshold.R = 240;
Red.Threshold.G = 220;
Red.Threshold.B = 222;
}
void GreenFilter()
{
Green.Threshold.R = 235;
Green.Threshold.G = 240;
Green.Threshold.B = 230;
}
void BlueFilter()
{
Blue.Threshold.R = 220;
Blue.Threshold.G = 235;
Blue.Threshold.B = 235;
}

void ReadRGB()
{
RGB[0] = ReadRed();
RGB[1] = ReadGreen();
RGB[2] = ReadBlue();
}
unsigned int ReadRed()
```

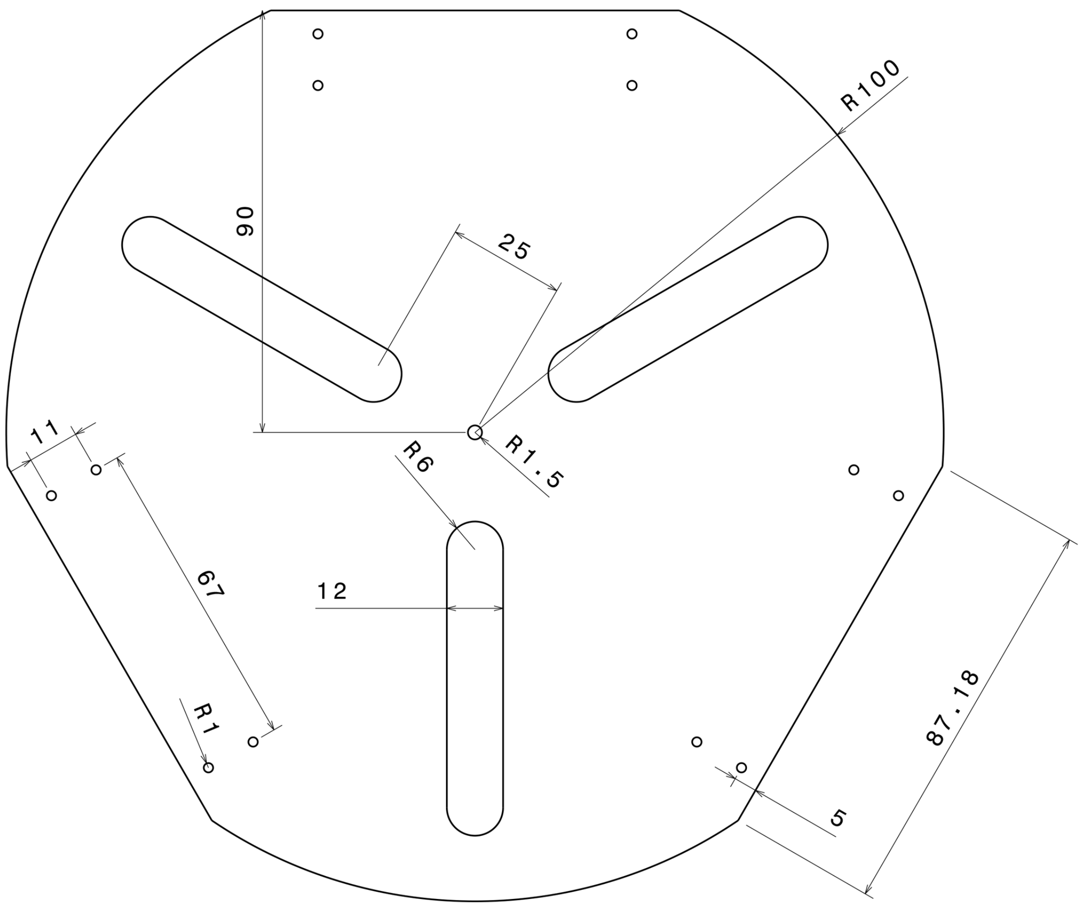
```
{
digitalWrite(s2, LOW);
digitalWrite(s3, LOW);
unsigned int f = pulseIn(out, LOW);
f = map(f, 15, 180, 255, 0);
return f;
}
unsigned int ReadGreen()
{
digitalWrite(s2, HIGH);
digitalWrite(s3, HIGH);
unsigned int f = pulseIn(out, LOW);
f = map(f, 25, 180, 255, 0);
return f;
}
unsigned int ReadBlue()
{
digitalWrite(s2, LOW);
digitalWrite(s3, HIGH);
unsigned int f = pulseIn(out, LOW);
f = map(f, 17, 180, 255, 0);
return f;
}
unsigned char WhatIsTheColor(unsigned int *c) // Determinate the color R for Red, B
for Blue G for Green and X for unknown color
{
return ColorIsRed(c) ? 'R' :
(ColorIsGreen(c) ? 'G' :
(ColorIsBlue(c) ? 'B'
: 'X'));
}
bool ColorIsRed(unsigned int *c) // Returns true if the color is Red
{
return c[0] >= Red.Threshold.R &&
c[1] < Red.Threshold.G &&
c[2] < Red.Threshold.B;
}
bool ColorIsGreen(unsigned int *c) // Returns true if the color is Green
{
return c[0] < Green.Threshold.R &&
c[1] >= Green.Threshold.G &&
c[2] < Green.Threshold.B;
}
bool ColorIsBlue(unsigned int *c) // Returns true if the color is Blue
{
return c[0] < Blue.Threshold.R &&
c[1] >= Blue.Threshold.G &&
c[2] >= Blue.Threshold.B;
}
```

Appendix C

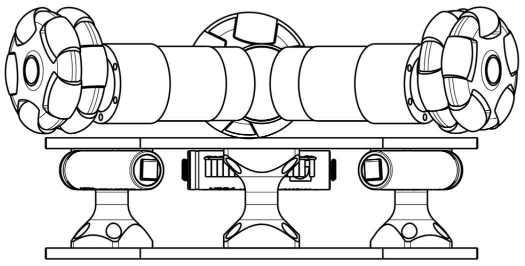
Technical Design

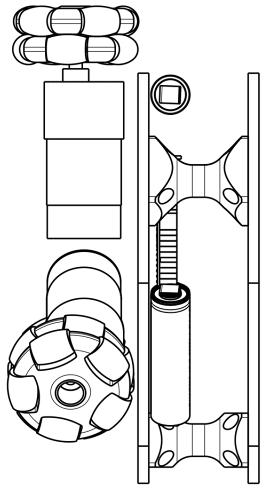


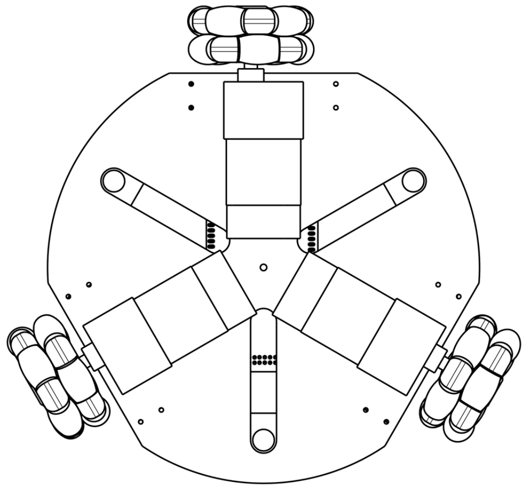
Material : Aluminium Alloy	Support x3		Designed by : Youssef Mestiri
Treatment : 2	Parts : Support	Project : Omni robot	Verified and approved by :
Qty : 4 / Weight :	General linear tolerance: 0.1 mm	General angular tolerance: 1°	Date: 17/10/2022
Scale : 1:1	INSTITUTO POLITÉCNICO DE BRAGANÇA		Page: 1 / Format: A4



Material : Aluminium Alloy	Platform		Designed by : Youssef Mestiri
Treatment : 2	Parts : Support	Project : Omni robot	Verified and approved by :
Qty : 4	Weight :	General linear tolerance: 0.1 mm	General angular tolerance: 1°	Date: 17/10/2022
Scale : 1:1	INSTITUTO POLITÉCNICO DE BRAGANÇA		Page: 1	Format: A4







Material : Aluminium Alloy			Designed by : Youssef Mestiri
Treatment : 2	Parts : Support	Project : Omni robot	Verified and approved by :
Qty : 4	Weight :	General linear tolerance: 0.1 mm	General angular tolerance: 1°	Date: 17/10/2022
	Scale : 1 : 3	INSTITUTO POLITÉCNICO DE BRAGANÇA	Page: 1	Format: A4

C4

Appendix D

Wiring Scheme

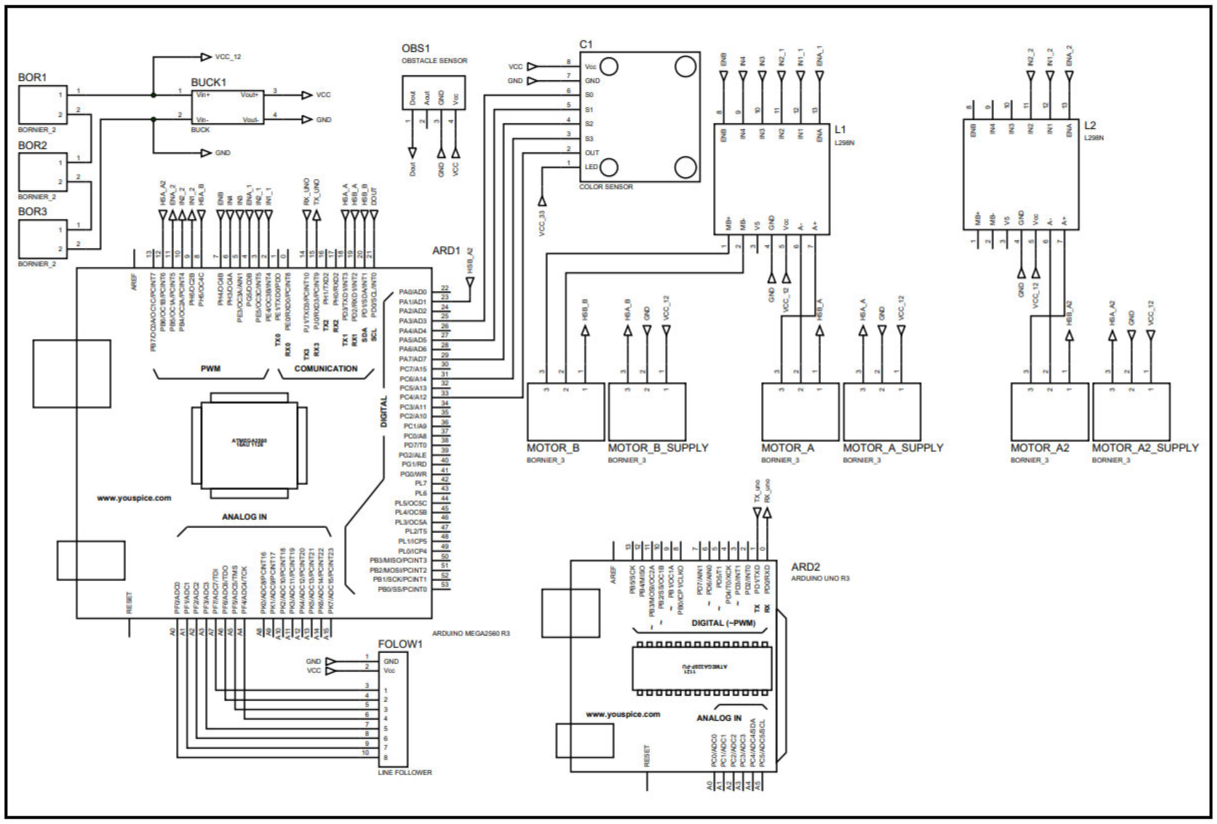


Figure D.1: Schematic diagram of the electronic circuit

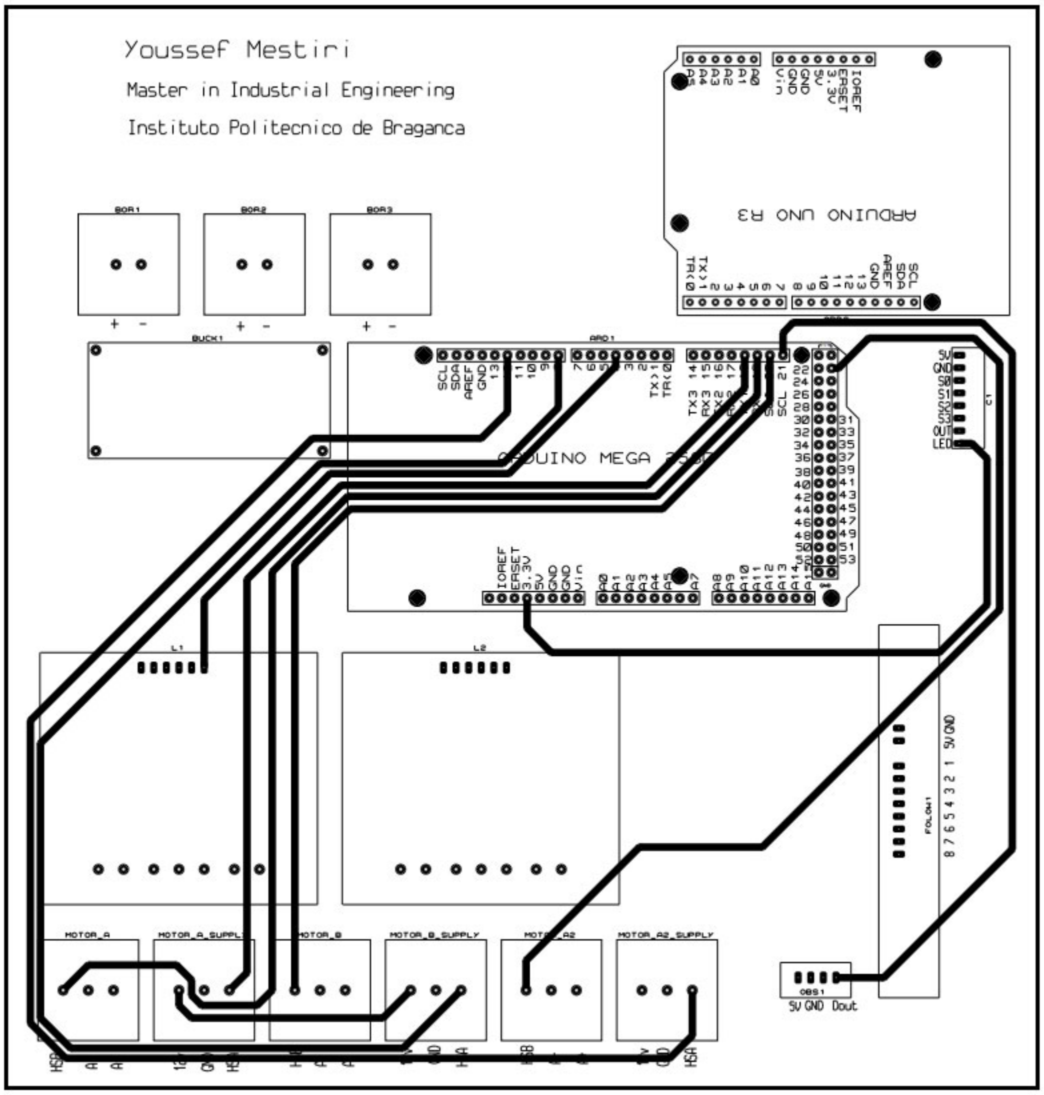


Figure D.2: PCB Design Diagram - Top Layer

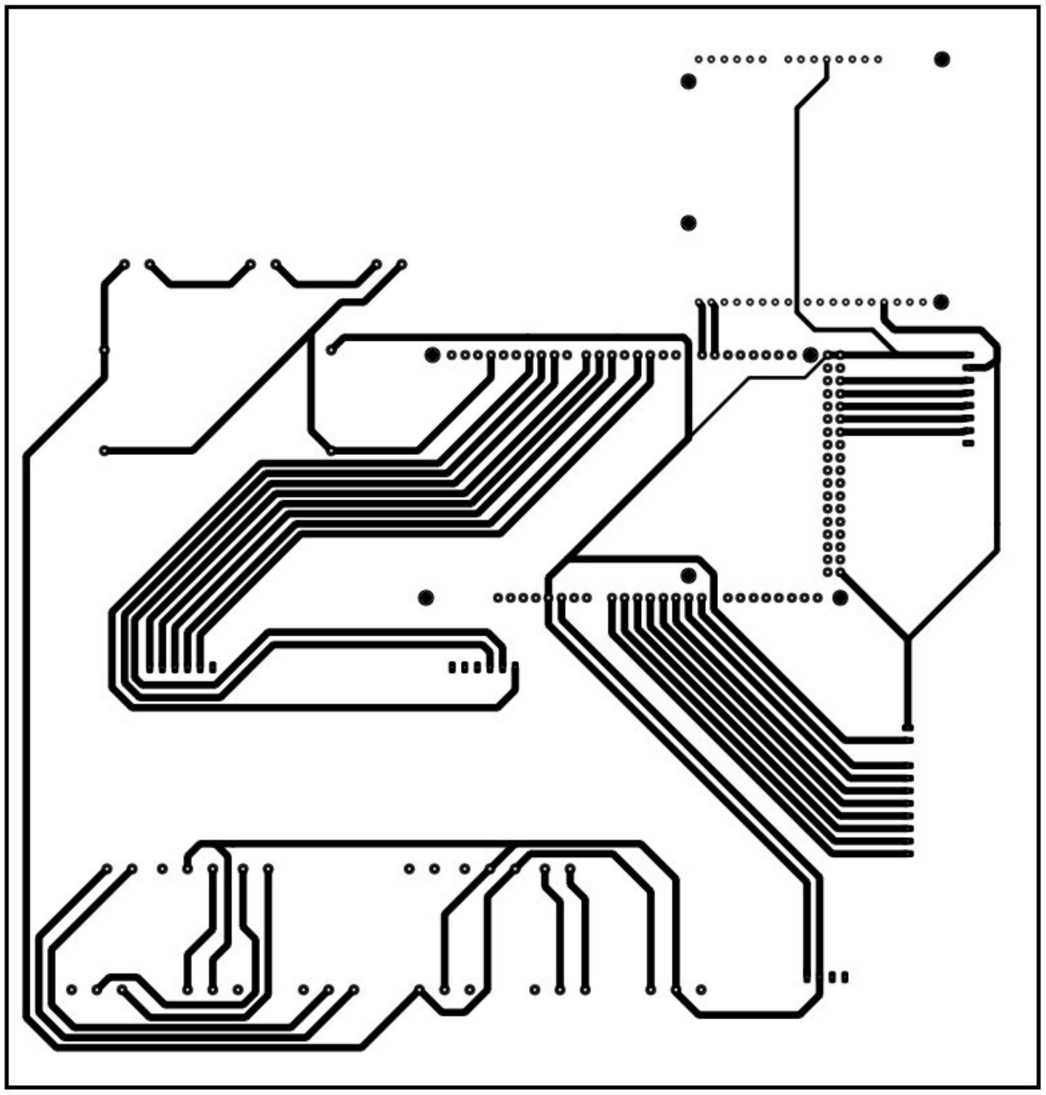


Figure D.3: PCB Design Diagram - Bottom Layer