Retrofitting Industrial Robotics: An Embedded System Approach to Controller Enhancement

Vinicius Ferreira da Silva Bianchi Grilo

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Electrical and Computers Engineering.

Work oriented by:

Professor PhD José Luís Sousa de Magalhães Lima

Professor PhD Fabiano Drumond Chaves

Bragança

2024

Retrofitting Industrial Robotics: An Embedded System Approach to Controller Enhancement

Vinicius Ferreira da Silva Bianchi Grilo

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Electrical and Computers Engineering.
Work developed under the Dual Degree Program between the Polytechnic Institute of Bragança (IPB) and the Federal Center for Technological Education of Minas Gerais (CEFET-MG).

Work oriented by:
Professor PhD José Luís Sousa de Magalhães Lima
Professor PhD Fabiano Drumond Chaves

Bragança
2024

Acknowledgement

Firstly, I would like to express my gratitude to my mother Sílvia and father Wilson for their whole support throughout my life. No one has believed in my potential as much as you both, and your investment in my education has made all the difference, allowing me to reach where I am today. I would also like to thank my sister, Tamyris, for the moments of joy she gave me in life. I am grateful for love you have always given me.

Additionally, I extend my appreciation to all my friends in Brazil, as well as my friends from the exchange program, expecially the members of Bairro Nobre: Melo, Edilson, William, Leonardo, Bruno and Guigaz.

A special thanks to my girlfriend, Júlia, for her support and companionship during this period. Your presence was essential to this achievement.

Thanks to my supervisors, Professor PhD José Luís Sousa de Magalhães Lima of Instituto Politécnico de Bragança (IPB), I sincerely value the support and trust given to me. Furthermore, I would like to thank my advisor, Professor PhD Fabiano Drumond Chaves, from Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG). Throughout my academic journey, both during college and my master's degree, his guidance and assistance really helped me with my professional development.

I would like to thank CEFET-MG for providing support throughout my relocation to Portugal. Thanks to IPB for giving me the chance of doing this double degree. This incredible opportunity has had a profound impact on my personal and professional development.

Last, but no less important, I would like to thank the CeDRI (UIDB/05757/2020 and UIDP/05757/2020) for the support provided during the development of this work.

Abstract

The increase in productivity and efficiency caused by the presence of robots in industries and in automation processes, whether they are mobile robots or manipulator robots, encourages research and makes the area of robotics widely studied in academic scope around the world. The development methodology began with a study of the functioning of the internal modules of the original FANUC RJ Controller model, including the main controllers, circuits responsible for driving the motors and emergency systems, in order to assess which modules could be reused in the new architecture proposed for the controller. As a result, in addition to the robot's mechanical unit, the FANUC Servo Amplifiers, the EMG Board, the Power Supply Unit, the Dynamic Brake Unit and the Main AC Power Distribution were reused in the development of the proposed controller, and a detailed study of these components was carried out. The proposed controller architecture includes the modularization of the motor position controllers, where the Dual Axis Control Board was developed, a system capable of controlling two PMSM motors via the FANUC Servo Amplifier, requiring three groups of these systems to control the robot's six motors. To calculate the kinematics and trajectory, an ESP32-S3 was used, which receives the user's commands, calculates the trajectory and angles of each joint of the robot and sends the angular references so that the motor controllers can act to move the robot. To integrate the systems, the Main Board Controller was developed, responsible for ensuring communication between the ESP32-S3 and the Dual Axis Control Boards, as well as providing auxiliary circuits for the emergency signals sent and received from the EMG Board. In order to improve interaction with the user and visualization of the trajectories executed

by the robot, a graphical interface was developed capable of receiving positioning commands from the user using its own programming language and displaying the robot's positioning using a 3D model of the FANUC S-420FD developed and incorporated into the system. Tests and validations carried out with the circuits developed showed that the signal processing and conditioning achieved the expected results, proving that the PCBs work. The algorithms developed to perform the kinematics and trajectory calculations were also validated, with high accuracy and processing times in line with the rest of the system. It can therefore be concluded that the results presented make it possible to achieve the objectives initially proposed.

Keywords: Controller; Industrial robot; Kinematics; Trajectory control; Graphical user interface.

Resumo

O aumento de produtividade e eficiência causada pela presença de robôs das indústrias e em processos de automação, sejam estes robôs móveis ou robôs manipuladores, encorajam a investigação e faz com que a área de robótica seja amplamente estudada no âmbito académico em todo o mundo. O presente trabalho aborda diversos assuntos que suportam o estudo da robótica, como eletrónica, controle de motores, cálculos geométricos, cálculo de trajetórias, interação com o utilizador e programação de microcontroladores, sendo o objetivo principal o desenvolvimento de um controlador para o robô industrial FANUC S-420FD de seis graus de liberdade, motivado pelo mau funcionamento do controlador original do sistema robótico. A metodologia de desenvolvimento iniciou-se com o estudo do funcionamento dos módulos internos do controlador original modelo FANUC RJ Controller, incluindo os controladores principais, circuitos responsáveis pelo acionamento dos motores e sistemas de emergência, visando avaliar quais módulos seriam reaproveitados na nova arquitetura proposta para o controlador. Com isso, além da unidade mecânica do robô, os FANUC Servo Amplifiers, a EMG Board, a Power Supply Unit, o Dynamic Brake Unit e o Main AC Power Distribution foram reutilizados no desenvolvimento do controlador proposto, sendo realizado um estudo detalhado destes componentes. A arquitetura proposta para o controlador inclui a modularização dos controladores de posição dos motores, onde foi desenvolvido a Dual Axis Control Board, um sistema capaz de controlar dois motores PMSM através do FANUC Servo Amplifier, sendo necessário três conjuntos destes sistemas para controlar os seis motores do robô. Para o cálculo de cinemática e trajetória foi utilizado um ESP32-S3, que recebe os comandos do utilizador, calcula a trajetória e os ângulos de cada junta do robô e envia as referências angulares para que

os controladores dos motores atuem movimentando o robô. Para realizar a integração entre os sistemas, foi desenvolvido a Main Board Controller, responsável por garantir a comunicação entre o ESP32-S3 e as Dual Axis Control Boards, além de oferecer circuitos auxiliares para os sinais de emergência enviados e recebidos da EMG Board. Visando melhorar a interação com o utilizador e a visualização das trajetórias executadas pelo robô, foi desenvolvido uma interface gráfica capaz de receber comandos de posicionamento do utilizador utilizando uma linguagem de programação própria e exibir o posicionamento do robô por meio de um modelo 3D do FANUC S-420FD desenvolvido e incorporado ao sistema. Testes e validações realizados com os circuitos desenvolvidos mostraram que o processamento e condicionamento dos sinais obtiveram os resultados esperados, comprovando o funcionamento das PCBs. Os algoritmos desenvolvidos para realizar os cálculos de cinemática e trajetória também foram validados, apresentando acurácia elevada e tempo de processamento condizente com o restante do sistema. Sendo assim, pode-se concluir que os resultados apresentados permitem alcançar os objetivos propostos inicialmente.

Palavras-chave: Controlador; Robô industrial; Cinemática; Controle de trajetória; Interface gráfica do utilizador.

Contents

List of Tables

List of Figures

Acronyms

3D three dimensional

CAD/CAM Computer-Aided Design and Computer-Aided Manufacturing

CNC Computer Numerical Control

DoF Degrees of Freedom

EMG Board Emergency Board

FK Forward Kinematics

FOC Field Oriented Control

G-Code Geometric Code

GPIO General Purpose Input/Output

GUI Graphical User Interface

IK Inverse Kinematics

J1 Joint 1

J2 Joint 2

J3 Joint 3

J4 Joint 4

J5 Joint 5

J6 Joint 6

MCSDK Motor Control Software Development Kit

PC Personal Computer

PCB Printed Circuit Board

PID Proportional, Integral and Derivative

PMSM Permanent Magnet Synchronous Motor

PSU Power Supply Unit

PWM Pulse Width Modulation

TP Teach Pendant

UART Universal Asynchronous Receiver/Transmitter

USB-OTG USB On-The-Go

Chapter 1

Introduction

Industrial robots assume a significant importance on the industrial scenario, increasing productivity in various production sectors [1]. These programmable machines, designed to perform specific tasks with efficiency and precision [2], [3], are constantly being improved since their conception, incorporating advanced technologies such as artificial intelligence, computer vision and machine learning to enhance their capabilities and operational flexibility.

The growing demand for efficiency, quality and speed in production has favoured the proliferation of industrial robots in a variety of sectors, from automotive manufacturing to electronics and healthcare [4], [5]. These automated machines offer significant benefits, such as reduced errors, increased productivity, lower production costs and the ability to perform repetitive tasks consistently. In addition, industrial robots play a crucial role in optimising workspaces, allowing humans to focus on more complex and creative tasks [6].

In university academic environments, the study of industrial robotics contributes to the formation of professionals capable of solving the challenges proposed by industry, as well as developing new solutions to improve and optimise robotic processes [7]. The development of projects involving industrial robots allows students to apply theoretical knowledge in practice, fostering the development of skills in areas such as electronics, mechanics, programming and automation [8].

In general, an industrial robotic unit is composed of two primary components: a

robotic manipulator and a controller [9]. The robotic manipulator is responsible for executing the tasks, and it is comprised of a mechanical structure, motors, and sensors. The controller is the system's central processing unit, interpreting the instructions received and issuing the necessary commands to the manipulator [10]. In many cases, the controller is provided with a user interface system, which allows for interaction between the operator and the robot. This system provides information on the state of the system and allows tasks to be programmed [1].

This study deals with issues related to the controller of an industrial robot and its revitalization. Revitalization is understood as the modernization of a system in order to improve its functionality, correct faults, and extend its useful life [11].

1.1 Context and motivation

This work approaches the more primary study of robotics, discussing topics related to the inner workings of an industrial robot controller, including electronics, motor control, emergency actions, geometric calculations of the robot and interaction with the operator.

The object of study used was a FANUC S-420FD industrial robot, dating from 1995, which was in operation in the automotive industry until 2012. This unit is currently located at CEFET-MG in Leopoldina, MG, Brazil, and presents irreversible malfunctions in its original controller, which motivated the development of this project. More details about the robotic system used will be covered in later chapters.

1.2 Objectives

This Section presents the general and specific objectives of this work.

General objective

The general objective of this work is to develop a new controller for the FANUC S-420FD robot, replacing the boards responsible for the trajectory calculations, kinematics,

user interface and motor control system, as well as integrating the proposed solution with other essential components of the robot's original controller.

Specific objectives

The specific objectives of this work are:

I. Study the main components of the robot's original controller.

II. Analyze the internal operations of the FANUC Servo Amplifier present in the robot's original controller.

III. Analyze the internal operations of the Emergency Board present in the original robot controller.

IV. Analyze the internal operations of the Power Supply Unit present in the robot's original controller.

V. Develop the architecture of the proposed controller.

VI. Develop the motor position control system.

VII. Develop the robot's forward and inverse kinematics and incorporate them into the developed controller.

VIII. Develop the trajectory control and incorporate it into the developed controller.

IX. Develop the printed circuit boards according to the proposed structure.

X. Develop a graphical interface that allows interaction with the user and monitoring of the robot's movement.

XI. Carry out performance tests on the systems developed.

1.3 Document structure

This document is divided into seven chapters, where Chapter 1 (Introduction) provides a brief contextualization and presents the desired general and specific objectives of the work, Chapter 2 (State of Art) discusses theoretical issues that were highlighted in the development of the project, as well as a brief literature review of related works, Chapter 3 (Development) describes all the methods and tools used to conduct the work, Chapter 4 (Results) presents the experimental results of the solutions developed and discussed in Chapter 3, Chapter 5 (Conclusion and Future work) analyzes the results presented, taking into account the proposed objectives and suggests work and implementations to improve the results obtained.

Chapter 2

State of art and Study of tools

2.1 Theoretical background

This section will detail the main theoretical concepts of greatest importance to this work.

2.1.1 Robotic manipulators

Robotic manipulators are mechatronic devices designed to perform specific manipulation tasks, whether on industrial production lines, in hazardous environments or even in more delicate applications such as healthcare [1]. One of the main characteristics of these systems is their ability to perform repetitive tasks with high precision and speed, which contributes to efficiency and productivity in various sectors [12].

An important feature of a robotic manipulator is the number of degrees of freedom present in its construction, which refers to the number of independent movements the system can perform. Each joint or articulation in a manipulator contributes to one Degrees of Freedom (DoF), allowing the robot to move in a specific direction [1]. For example, a manipulator with six degrees of freedom can perform movements along six different axes. Another important issue is that in order for the system to respect the number of degrees of freedom in the design, the movements must be independent, i.e. each DoF must be

controlled by a specific actuator.

In order for manipulator robots to perform the programmed movements, actuators are needed to move each part in synchronization. In industrial robots, electric actuators are used in most cases. In view of energy efficiency, power density and control accuracy, Permanent Magnet Synchronous Motors are often used [1], [13].

2.1.2 Permanent Magnet Synchronous Motor (PMSM)

Permanent Magnet Synchronous Motor (PMSM) have a significant presence in the electrical industry due to their superior efficiency, power density and precise control characteristics [14]. Compared to traditional asynchronous motors, PMSMs have notable energy efficiency, making them more applicable in situations that require high performance and low energy consumption [15]. This class of motor is characterized by the strategic use of permanent magnets in its rotor, which contributes to a more stable and efficient magnetic field [16].

PMSMs find application in a variety of areas, from electric propulsion systems in automotive vehicles to advanced industrial automation systems [17]. The ability of these motors to provide precise speed and position control makes them ideal for applications that require dynamic response, such as Computer Numerical Control (CNC) machines, robotic manipulators and positioning systems in general [16]. In addition, their low-maintenance profile and long service life make them the preferred choice in sectors where reliability and durability are critical [18].

The construction characteristics of PMSMs have an important impact on their efficiency and performance [18]. The inclusion of permanent magnets in the rotor results in a more stable fixed magnetic field, contributing to more efficient operation. However, it is important to recognize the technological challenges associated with this construction, such as the cost related to the production of rare earth permanent magnets. Dependence on these materials can impact the economic viability of PMSMs in certain applications, affecting their final cost [16].

Despite the advantages related to efficiency and the possibility of precise speed and position control, the control system and electronics required to achieve these results are more complex when compared to other simpler motor models, which require electronics equipped with a three-phase inverter and current sensors that allow the application of precise control algorithms [18].

2.1.3 Field Oriented Control (FOC)

Field Oriented Control (FOC), also known as Vector Control, is an advanced control technique used in electrical systems, especially in alternating current electric motors. This approach aims to optimize system performance and efficiency by controlling system currents in order to obtain the highest torque using the least energy. FOC is particularly applicable in situations where it is crucial to achieve high levels of precision, dynamic response and energy efficiency, making it a preferred choice in many industrial applications [19].

The main feature of the FOC is its ability to independently handle the magnetic and current components of the system. This is achieved through Clarke and Park's mathematical transformation, which transforms the readings of three-phase currents (i_a, i_b and i_c) with sinusoidal characteristics into continuous components (i_d and i_q) representing flux and torque. The continuous characteristic of the current signals after applying the Clarke and Park transformation makes it possible to use conventional control methods such as Proportional, Integral and Derivative (PID) [19].

The Clarke and Park transform is the combination of two transforms applied in sequence. The first to be applied is the Clarke transform, where the three-phase currents i_a, i_b and i_c are transformed with i_α and i_β as detailed in Equation 2.1 and illustrated in Figure 2.1 [20], [21].

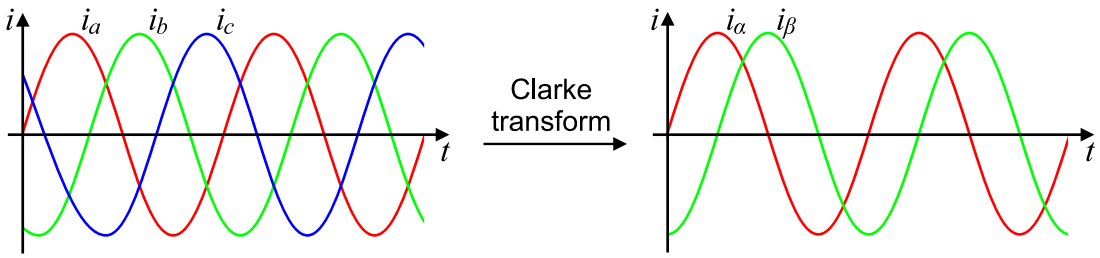


Figure 2.1: Clarke transformation.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.1}$$

The Park transform works with the result of the Clarke transform and transforms i_α and i_β into i_d and i_q as detailed in Equation 2.2 and illustrated in Figure 2.2 [21].

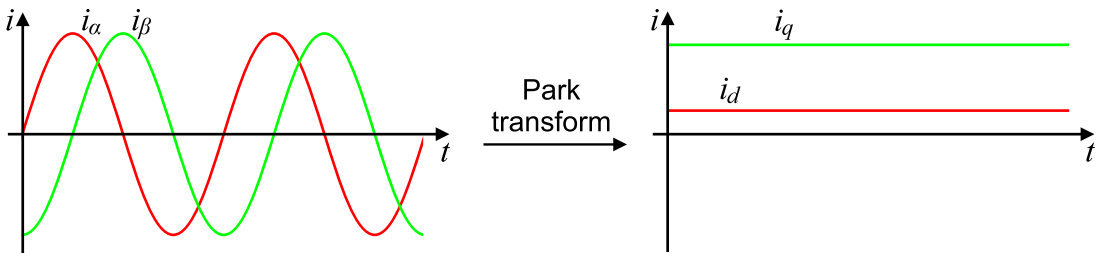


Figure 2.2: Park transformation.

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{2.2}$$

The Clarke and Park inverse transforms can also be applied and are used in the FOC, in this case the voltages v_d and v_q are transformed to the three-phase format v_a, v_b and v_c.

Figure 2.3 shows a simplified diagram of Field Oriented Control using the Clarke and Park transforms. The i_a, i_b and i_c currents are obtained using current sensors connected to the three phases of the motor, which are inserted into the Clarke and Park transforms to obtain the i_d and i_q currents. The i_d and i_q currents are used as current feedback for PI controllers, which are used to adjust the v_d and v_q voltages according to the torque and flux references applied to the controller. The inverse Park transform is used to obtain

v_α and v_β, which are then modulated in Pulse Width Modulation (PWM) and drive the inverter connected to the motor [22].

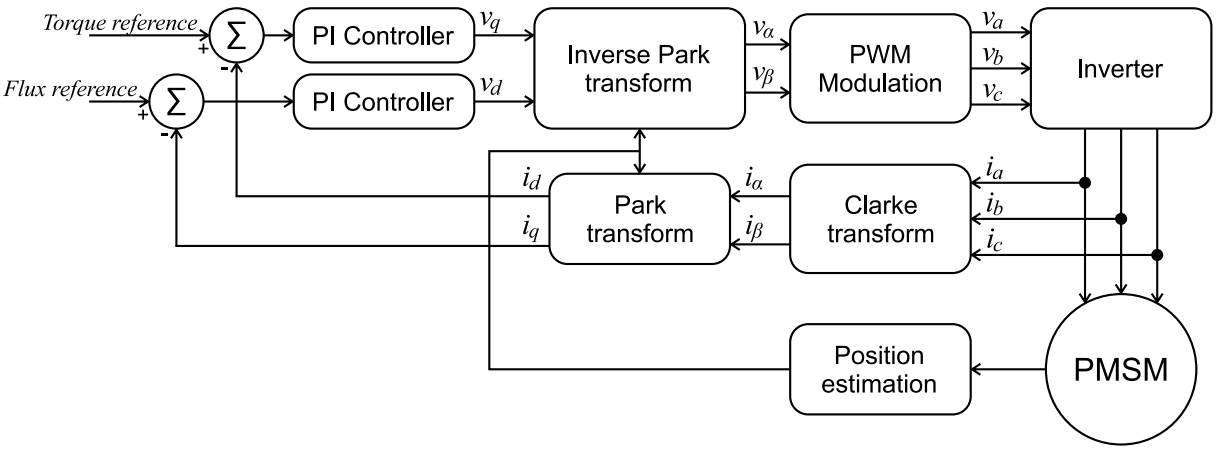


Figure 2.3: Field Oriented Control simplified diagram.

In order to realize Field Oriented Control based speed or position controllers, it is necessary to add controllers (usually PID controllers) to generate the torque and flux references for the respective controllers and drive the motor as desired [19].

2.1.4 STM32 Motor Control Software Development Kit

The Motor Control Software Development Kit (MCSDK) [23] is a tool developed by STMicroelectronics [24] to facilitate the design of motor control software for devices based on ST microcontrollers. This tool is specially developed to control the torque, speed or position of up to two Permanent Magnet Synchronous Motors using Field Oriented Control and has a Graphical User Interface for the system to be configured and customized, shown in Figure 2.4 and called MCSDK Workbench.

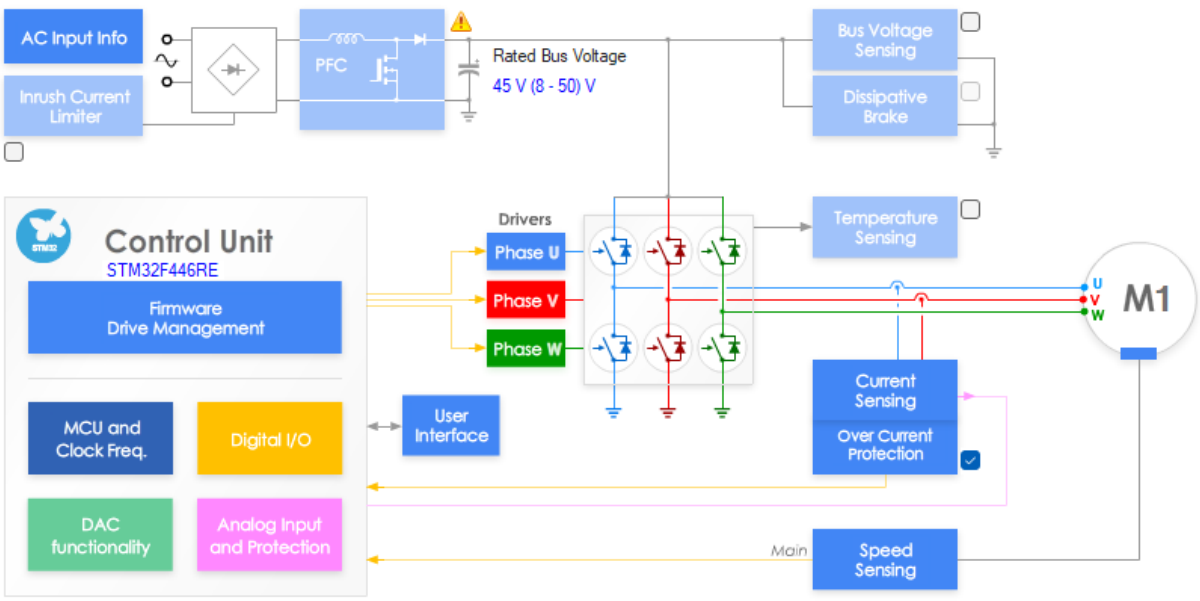


Figure 2.4: Motor Control Software Development Kit Workbench.

Using the MCSDK Workbench, it is possible to configure the parameters of the desired control system, such as motor parameters, current sensor characteristics, speed/position sensor characteristics, PWM frequency, controller configuration, microcontroller model used, etc. The code project is generated for STM32CubeIDE [25], provided by STMicroeletronics for programming the microcontrollers developed by the company, where it is possible to make changes to the system's operation and add new features, as well as allowing the code to be loaded into the microcontroller.

The MCSDK also has its own serial communication protocol for sending commands via external devices. Details of the protocol can be found at *"<installation folder>\html\motor-control-protocol-suite.html"*, available after installation of the tool.

When configured for position control, two possible commands are available via the serial protocol: *"Move Command"* and *"Follow Command"*. The *"Move Command"* sends the target position in radians and the time the movement will take in seconds, and the system performs all the control to ensure that the parameters are respected. In the *"Follow Command"* only the target position in radians is sent and through a sequence of commands of this type, the controller performs the control to follow the desired positions.

2.1.5 Jerk

The study of movement in physics covers various quantities, including position, velocity and acceleration. One parameter of great importance, especially in areas such as mechanical engineering, robotics and biomechanics, is *"Jerk"*, which represents the rate of change of acceleration in relation to time and is commonly referred to by the letter "J" [26]. The analysis of jerk is crucial in these fields, where abrupt changes in acceleration can impact the efficiency and safety of systems.

The main characteristic of a movement that uses Jerk is that the acceleration does not remain constant and behaves as a ramp in certain phases of the movement. Figure 2.5 shows the characteristic curves of a movement using Jerk, where the Jerk applied is constant, the acceleration is on a ramp, the velocity follows a second order curve and the position behaves like a third order curve [27], [28], [29].

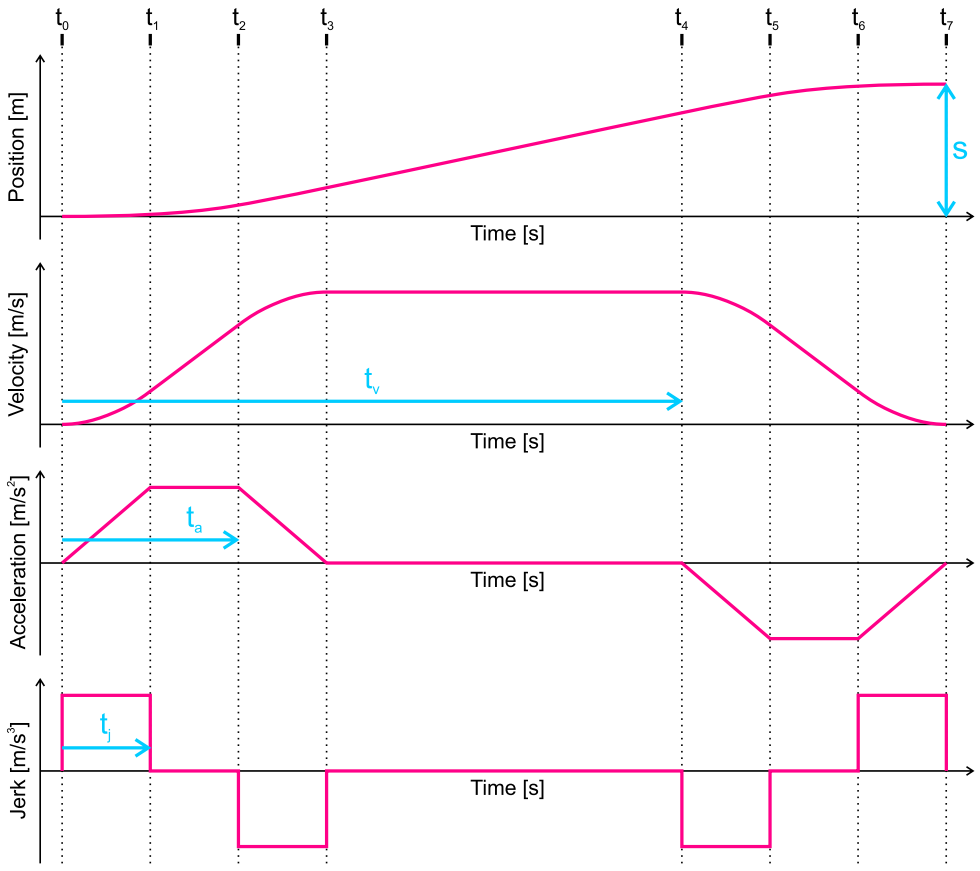


Figure 2.5: Characteristic curves of a movement using Jerk [27].

To define the equations of the curves shown in Figure 2.5, it is necessary to divide the movement into seven phases, defined by the time markers t_0 to t_7 in Figure 2.5 [27], [28], [29]. It is also important to note that the movement can follow six different shapes depending on the desired final position "s" and the time in which the movement is to occur. For example, there may be cases where there is no constant acceleration phase, and therefore $t_1 = t_2$.

The definitions of v_a, s_a and s_v shown in Equations 2.3, 2.4 and 2.5 support the conditions for defining which of the six shapes the movement conforms to, taking into account the maximum velocity v_{max}, the maximum acceleration a_{max} and the maximum Jerk j_{max}, defined previously [27].

$$v_a = \frac{{a_{max}}^2}{j_{max}} \tag{2.3}$$

$$s_a = \frac{2.{a_{max}}^3}{{j_{max}}^2} \tag{2.4}$$

$$\begin{gathered} s_v = v_{max}\left[M\left(2.\sqrt{\frac{v_{max}}{j_{max}}}\right) + N\left(\frac{v_{max}}{a_{max}} + \frac{a_{max}}{j_{max}}\right)\right] \\ \text{if } v_{max}.j_{max} < {a_{max}}^2 \ , \ M = 1 \text{ and } N = 0 \\ \text{if } v_{max}.j_{max} \geq {a_{max}}^2 \ , \ M = 0 \text{ and } N = 1 \end{gathered} \tag{2.5}$$

The Equation 2.6 defines the conditions for defining which of the six shapes the movement conforms to, considering v_{max} and s in relation to the defined constants v_a, s_a and s_v [27], [28], [29].

$$\begin{cases} \text{if } (v_{max} < v_a) \text{ and } (s > s_a) \text{ , shape I} \\ \text{if } (v_{max} > v_a) \text{ and } (s < s_a) \text{ , shape II} \\ \text{if } (v_{max} < v_a) \text{ and } (s < s_a) \text{ and } (s > s_v) \text{ , shape III} \\ \text{if } (v_{max} < v_a) \text{ and } (s < s_a) \text{ and } (s < s_v) \text{ , shape IV} \\ \text{if } (v_{max} > v_a) \text{ and } (s > s_a) \text{ and } (s > s_v) \text{ , shape V} \\ \text{if } (v_{max} > v_a) \text{ and } (s > s_a) \text{ and } (s > s_v) \text{ , shape VI} \end{cases} \tag{2.6}$$

Once the shape of the movement has been defined, Table 2.1 defines t_j, t_a and t_v shown in Figure 2.5 according to shapes I to VI [27], [28], [29].

Shape / Constant	**I**	**II**	**III**	**IV**	**V**	**VI**
$\boldsymbol{t_j}$	$\sqrt{\frac{v_{max}}{j_{max}}}$	$\sqrt[3]{\frac{s}{2.j_{max}}}$	$\sqrt{\frac{v_{max}}{j_{max}}}$	$\sqrt[3]{\frac{s}{2.j_{max}}}$	$\frac{a_{max}}{j_{max}}$	$\frac{a_{max}}{j_{max}}$
$\boldsymbol{t_a}$	t_j	t_j	t_j	t_j	$\frac{v_{max}}{j_{max}}$	$\frac{1}{2}\left(\sqrt{\frac{4.s.{j_{max}}^2+{a_{max}}^3}{a_{max}.{j_{max}}^2}} - \frac{a_{max}}{j_{max}}\right)$
$\boldsymbol{t_v}$	$\frac{s}{v_{max}}$	$2t_j$	$\frac{s}{j_{max}}$	$2t_j$	$\frac{s}{v_{max}}$	$t_a + t_j$

Table 2.1: Definition of t_j, t_a and t_v for each movement shape [29].

Analysing Figure 2.5, the values from t_1 to t_6 can be defined according to Equation 2.7, where t_0 is defined by the time the movement starts [27], [28], [29].

$$\begin{cases} t_1 = t_j \\ t_2 = t_a \\ t_3 = t_a + t_j \\ t_4 = t_v \\ t_5 = t_v + t_j \\ t_6 = t_v + t_a \\ t_6 = t_v + t_j + t_a \end{cases} \tag{2.7}$$

Finally, the equations used to determine the behaviour of the jerk, acceleration, velocity and position in each time interval of the movement are defined in Table 2.2 .

	Jerk	**Acceleration**	**Velocity**	**Position**
$\mathbf{t_0...t_1}$	j_{max}	$j_{max}.(t-t_0)$	$\frac{1}{2}.j_{max}.(t-t_0)^2$	$\frac{1}{6}.j_{max}.(t-t_0)^3$
$\mathbf{t_1...t_2}$	0	$a_1 = a_2$	$v_1 + a_1.(t-t_1)$	$p_1 + v_1.(t-t_1) + \frac{1}{2}.a_1.(t-t_1)^2$
$\mathbf{t_2...t_3}$	$-j_{max}$	$a_2 - j_{max}.(t-t_2)$	$v_2 + a_2.(t-t_2) - \frac{1}{2}.j_{max}.(t-t_2)^2$	$p_2 + v_2.(t-t_2) + \frac{1}{2}.a_2.(t-t_2)^2 - \frac{1}{6}.j_{max}.(t-t_2)^3$
$\mathbf{t_3...t_4}$	0	0	$v_3 = v_4$	$p_3 + v_3.(t-t_3)$
$\mathbf{t_4...t_5}$	$-j_{max}$	$-j_{max}.(t-t_4)$	$v_4 + \frac{1}{2}. - j_{max}.(t-t_4)^2$	$p_4 + v_4.(t-t_4) - \frac{1}{6}.j_{max}.(t-t_4)^3$
$\mathbf{t_5...t_6}$	0	$a_5 = a_6$	$v_5 - a_{max}.(t-t_5)$	$p_5 + v_5.(t-t_5) + \frac{1}{2}.a_5.(t-t_5)^2$
$\mathbf{t_6...t_7}$	j_{max}	$a_6 + j_{max}.(t-t_6)$	$v_6 + a_6.(t-t_6) + \frac{1}{2}.j_{max}.(t-t_6)^2$	$p_6 + v_6.(t-t_6) + \frac{1}{2}.a_6.(t-t_6)^2 - \frac{1}{6}.j_{max}.(t-t_6)^3$

Table 2.2: Equations that describe the movement over time intervals [29].

2.1.6 Denavit-Hartenberg parameters and Kinematics

One method for expressing a robot in geometric and mathematical terms is to utilize the Denavit-Hartenberg parameters. This approach involves the representation of the geometry and configuration of the robot's joints, which facilitates the analysis of the robot's movement [1], [30].

The Denavit-Hartenberg parameters consider the analysis of each link of the robot in

relation to the previous one and can be represented as shown in Table 2.3, which summarizes the Denavit-Hartenberg parameters for a 6-axis manipulator, where θ_i represents the joint angle, α_i the twist angle, d_i the joint offset and a_i the link length [31], [32].

$axis$	θ_i	$\alpha_i[^\circ]$	d_i[mm]	a_i[mm]
1	θ_1	α_1	d_1	a_1
2	θ_2	α_2	d_2	a_2
3	θ_3	α_3	d_3	a_3
4	θ_4	α_4	d_4	a_4
5	θ_5	α_5	d_5	a_5
6	θ_6	α_6	d_6	a_6

Table 2.3: Denavit-Hartenberg parameters for a 6-axis manipulator [31], [32].

The kinematics of a robotic manipulator refers to the study of the movements and positions of the system, without taking into account the forces involved. It describes the geometric relationship between the different parts of the manipulator, specifically the joints and links, allowing the positioning and orientation of the end effector to be analyzed in relation to a coordinate system [33].

Forward kinematics involves determining the position and orientation of the end effector based on the values of the joints, while inverse kinematics involves calculating the positions and orientations of the joints required to position the end effector at a given position in space [34].

Forward kinematics

The robot kinematics model proposed by Denavit and Hartenberg is based on the D-H coordination system [30], and illustrates the connection between adjacent links in terms of translation and rotation. By substituting the parameters listed in the parameter table, the transformations between consecutive joints can be expressed using a matrix denoted as A_n (Equation 2.8), with the subscript n representing the respective joint [35], [36].

$$A_n = \begin{bmatrix} cos(\theta_n) & -sin(\theta_n)cos(\alpha_n) & sin(\theta_n)sin(\alpha_n) & a_n cos(\theta_n) \\ sin(\theta_n) & cos(\theta_n)cos(\alpha_n) & cos(\theta_n)sin(\alpha_n) & a_n sin(\theta_n) \\ 0 & sin(\alpha_n) & cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

The Forward Kinematics is obtained by multiplying matrices of joints of the robot, according to the Equation 3.5, resulting in the homogeneous transformation matrix of the end effector in relation to the base of the robot [35], [36].

$$T_f = \prod_{n=1}^{nJoints} A_n \tag{2.9}$$

Inverse kinematics

The Inverse Kinematics is the process of determining the joint angles required to position the end effector at a specific point in space. The solution to this problem is not always unique, and in some cases, it may not be possible to find a solution. The IK problem can be solved using geometric methods [37] or analytical methods [38].

The angles of each joint is calculated considering the desired translation and orientation for the end effector and can be represented as in Equation 2.10. The translation part is shown in the first three positions of the matrix and the orientation part is shown in the last three positions of the matrix [35], [36].

$$T_e = \begin{bmatrix} p_x \\ p_y \\ p_z \\ Z_e \\ Y'_e \\ Z''_e \end{bmatrix} \tag{2.10}$$

Considering the abbreviations presented below, the matrix presented in Equation 2.11 represents the homogeneous transformation matrix that describes the desired translation and orientation for the end effector.

$$\alpha = Z_e \qquad \beta = Y'_e \qquad \gamma = Z''_e$$
$$C_1 = cos(\alpha) \qquad C_2 = cos(\beta) \qquad C_3 = cos(\gamma)$$
$$S_1 = sin(\alpha) \qquad S_2 = sin(\beta) \qquad S_3 = sin(\gamma)$$

$$T = \begin{bmatrix} C_1C_2C_3 - S_1S_3 & -C_3S_1 - C_1C_2S_3 & C_1S_2 & p_x \\ C_1S_3 - C_2C_3S_1 & C_1C_3 - C_2C_1S_3 & S_1S_2 & p_y \\ -C_3S_2 & S_2S_3 & C_2 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

The process for determining the angles of each joint, given a desired translation and orientation for the end effector, can be divided into two steps:

1. Calculate the angles of the first three joints (θ_1, θ_2 and θ_3);

2. Calculate the angles of the last three joints (θ_4, θ_5 and θ_6), represented by spherical joints, using the calculated angles of the first three joints.

The detailed calculation of the angle of each joint will be revisited in Chapter 3 and applied to the context and the robot used in this work.

2.2 Related works

Robotics is widely studied around the world and research can be found in various areas. In order to better understand the extent to which the key words of this work are covered in published works, a search was performed on Web of Science [1], Scopus[2] and IEEE Xplore[3]. These platforms bring together various databases of academic publications with the intention of facilitating the search for scientific articles.

[1]https://www.webofscience.com
[2]https://www.scopus.com
[3]https://ieeexplore.ieee.org

Table 2.4 shows the keywords used and the results obtained. The first term searched was *"Robot controller"* and a considerable number of results were obtained on both platforms, including not only work related to robotic manipulators, but also mobile robots. Adding the keyword *"Kinematics"*, a subject also discussed in this work, reduced the number of results to around 5% on Web of Science, 8% on Scopus and 22% on IEEE Xplore. As more keywords from topics related to this work were added, such as *"Graphical user interface"*, *"Trajectory control"* and *"FANUC"*, the results were increasingly reduced, reaching zero results when all the terms were searched together on Web of Science and Scopus and only one result on IEEE Xplore.

Key words	**Web of Science**	**Scopus**	**IEEE Xplore**
Robot controller	55190	2845	35212
Robot controller Kinematics	2951	228	8041
Robot controller Kinematics Trajectory control	967	4	2710
Robot controller Kinematics Graphical user interface	10	1	93
Robot controller Kinematics Trajectory control Graphical user interface	5	0	18
Robot controller Kinematics Trajectory control Graphical user interface FANUC	0	0	1

Table 2.4: Terms searched and number of results obtained in different databases.

The result obtained on IEEE Xplore using all the key words mensioned was the project realised in [39], which used the mechanical unit of a FANUC S-420F industrial robot as

the object of study for the development of an FPGA-based open architecture industrial robot controller. The development methodology did not use any FANUC equipment other than that contained in the robot's mechanical unit. For this reason, six AMC servo amplifiers were used to carry out individual torque control for each motor, and an FPGA was responsible for generating the torque references for joint position control. A Personal Computer (PC) was used to perform the mathematical solutions for kinematics and trajectory, communicating directly with the FPGA. In addition, a Graphical User Interface was developed to interact with the user and be executed by the PC. Despite the differences in the architecture of the controller proposed in [39], it is similar to the work covered in this document in the use of a FANUC industrial robot as the object of study, but differs in not using other FANUC equipment present in the robot's original controller.

In [40] is proposed an open design DSP-based control system for industrial robot, using a Texas Instruments TMS320LF2407A model DSP as the system's main controller, supported by circuits based on the LM628 integrated circuit to process encoder signals and regulate the motion control feedback. The system uses an Industrial Personal Computer (IPC) communicating with the DSP to process user commands. The gains of the axis position PID controller were adjusted using fuzzy auto-tuning techniques and showed superior results to traditional methods. The authors validated the proposed system using a SCARA robot with four degrees of freedom and have achieved a good level of control accuracy and dynamic performance.

The work discussed in [41] proposes a methodology for remanufacturing industrial robotics manipulators and used the ASEA IRB6-S2 robot to validate the new controller approach. The methodology proposed starts analysing if the mechanical structure is in use conditions followed by the analysis of the condition of the control system. If the control system is in working condition, the necessary minor repairs are carried out and the robotic system is available for use, otherwise a new controller must be developed. The case study presented in [41] follows the case where the entire controller must be developed. Experimental tests were carried out on the platform developed using a welding tool as the robot's actuator and it was concluded that the kinematics and trajectory

calculations and the control system developed obtained satisfactory results. In addition, the authors considered replacing the MACH3 and MATLAB software with solutions using microcontrollers, considering that these softwares are proprietary and require a specific licence to operate, which makes the project unviable in some contexts.

Chapter 3

Development

This Chapter describes all the development stages of this work, including the analysis of the original FANUC S-420FD robotic system and the development of the proposed controller systems and algorithms.

3.1 FANUC S-420FD original system

The FANUC S-420FD robot is a complete robotic system, consisting of the robot's mechanical unit and its controller model FANUC RJ Controller. This robot model was designed in 1995, has six axes, a maximum payload of 120kg and a maximum horizontal reach of 2.4 meters [42] [43].

This Section describes the main parts of the FANUC S-420FD and FANUC RJ Controller that are relevant to this work.

3.1.1 Mechanical unit

The mechanical unit of the FANUC S-420FD, showed in Figure 3.1, is composed predominantly of metal parts, tota a total weight of 1700kg [42].

The robot's base supports its entire weight and has mounting holes for attachment to specific platforms or to the ground.

Each one of the six joints is driven by an independent motor. The two motors of Joint 2 (J2) and Joint 3 (J3) are exposed and can be easily seen, but the other motors are protected by metal plates or mounted inside the robot.

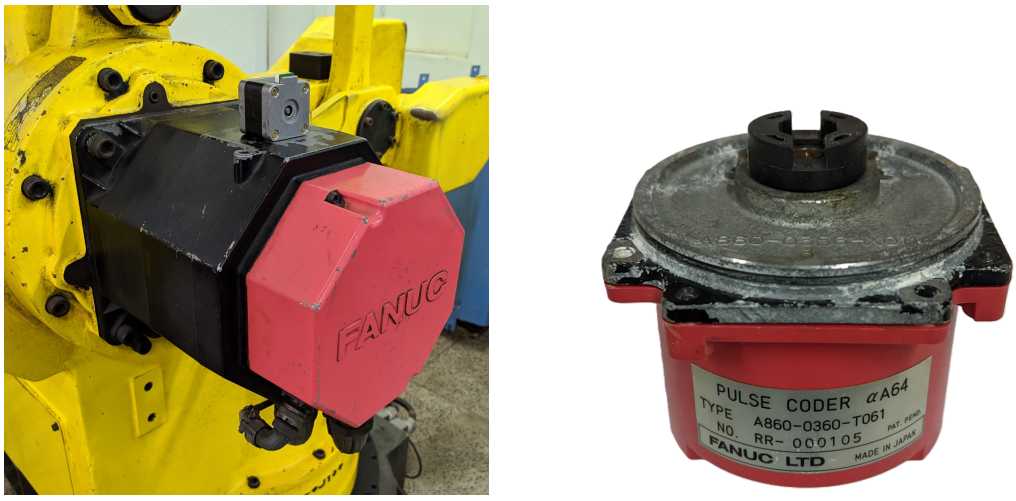
All cables are routed through the inside of the robot and are accessed via connectors at the bottom rear of the base.

Despite the age of the robot, the mechanical unit is in good condition, with motors and mechanical joints working properly.

Figure 3.1: FANUC S-420FD.

3.1.2 Robot motors

The motors of the FANUC S-420FD robot are fundamental components of the system and are responsible for moving the robot's parts. The models and specifications of the motors vary according to the axis on which the motor is responsible for moving, but all six motors are 3-phase PMSM motors with mechanical brakes and incremental encoders. All the motors are driven by the same model of FANUC Servo Amplifier, which is described in Section 3.1.3.1. The Figure 3.2(a) shows the motor that controls J2 compared to a Nema 17 stepper motor and Figure 3.2(b) shows the encoder of the Joint 6 (J6) axis, model FANUC Pulse Coder $\alpha A64$.



(a) (b)

Figure 3.2: FANUC S-420FD parts. **(a)** J2 motor. **(b)** J6 encoder.

The specifications of the motor of each axis are described in Table 3.1.

Axis	Part number	Model	Stall torque	Poles	RPM	Stall current	Voltage
J1	A06B-0502-B755	20S/2000	23 Nm	8	2000	20 A	146 V
J2	A06B-0356-B755	20F/2000	24 Nm	8	2000	20 A	140 V
J3	A06B-0357-B755	30F/2000	30 Nm	8	2000	24 A	148 V

J4	A06B-0501-B755	10/2000	12 Nm	8	2000	11 A	144 V
J5	A06B-0348-B356	8F/2000	12 Nm	8	2000	9 A	165 V
J6	A06B-0346-B356	5F/3000	7 Nm	8	3000	7.6 A	145 V

Table 3.1: FANUC S-420FD axis motor specifications [42].

3.1.3 FANUC S-420FD original controller

The original controller for the FANUC S-420FD robot is the FANUC RJ Controller model. Figure 3.3 shows two controllers of this model available for the work in question.

Figure 3.3: FANUC RJ Controller.

To provide a better understanding of the robot's original operation and assess which parts could be reused, an analysis of its original FANUC RJ Controller was performed. It is important to mention that a lot of information about the controller was found in the equipment manuals, but technical details, such as the internal workings of the components, are not available in the maintenance manuals. Therefore, extensive reverse engineering was required to obtain the information described in this Section.

The FANUC RJ Controller's Operator Panel is one of the controller's interfaces with the user and provides important functions for the system, such as turning the controller on and off, as well as an emergency button. Figure 3.4 shows the Operator Panel in more detail.

Figure 3.4: FANUC RJ Controller Operator Panel.

Another important interface between the controller and the user is the Teach Pendant (TP) (Figure 3.5). This device allows the user to control, configure and program the robot, as well as offering two primary emergency mechanisms: an emergency button and the Deadman Switch. The Deadman Switch is a pair of buttons located on the back of the TP and must be pressed throughout the robot's manual operation, otherwise the motors are deactivated and the brakes are activated.

Figure 3.5: FANUC RJ Controller Teach Pendant.

Figure 3.6 shows the main parts of the controller, namely the Dynamic Brake Unit, Main Controller Modules, Modular I/O, Main AC Power Distribution and FANUC Servo Amplifiers [43]. Although Figure 3.6 shows two FANUC Servo Amplifiers, the controller has three units, with the leftmost slot being empty at the time of capture.

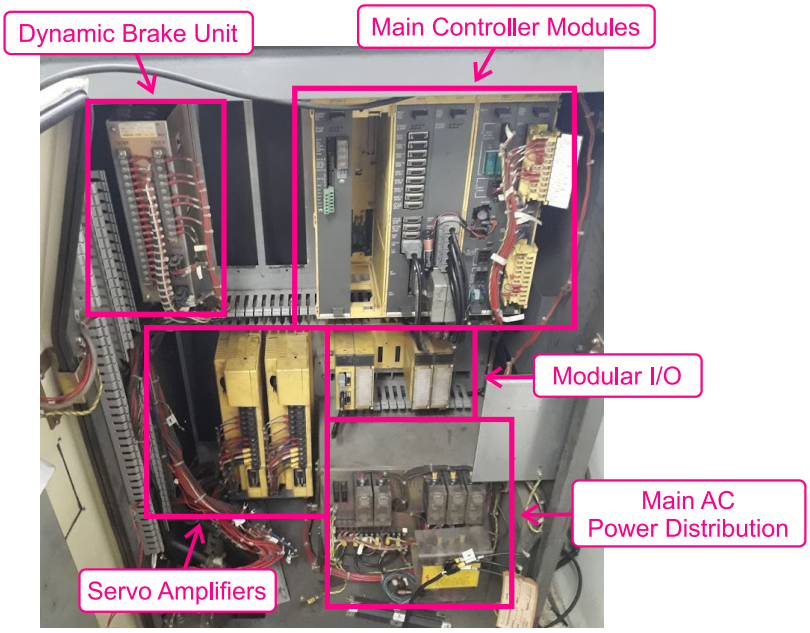


Figure 3.6: FANUC RJ Controller parts.

The block indicated as Main Controller Modules in Figure 3.6 in divided in modules. The Figure 3.7 shows this part in more detail, indicating the Optional Slots, Main CPU, Axis Control Board, Power Supply Unit (PSU), Emergency Board (EMG Board) and Backplane, which is responsible for connecting the modules and ensuring communication between them [43].

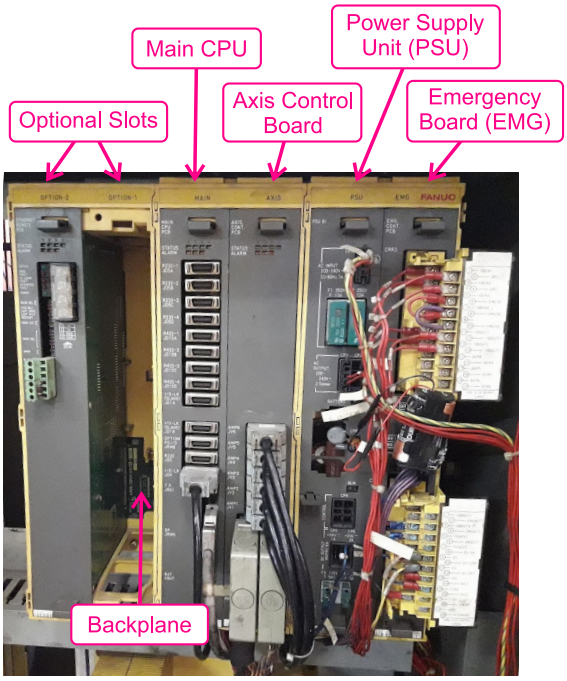


Figure 3.7: Main Controller Modules parts.

According to the *SYSTEM R-J Controller Series Electrical Connection and Maintenance Manual* [43], the electrical connections between the controller's components are as specified in Figure 3.8. Analysing Figure 3.8 from right to left, each motor has an encoder attached to provide the position information for the axes. The motors are driven by three FANUC Servo Amplifiers, each unit responsible for controlling two motors. The Axis Control Board receives the position information from the encoders and sends the control signal to the FANUC Servo Amplifiers, as well as using the Dynamic Brake Unit to help brake the motors when necessary. The Main CPU is responsible for calculating the kinematics, trajectory control, user interface, as well as other important functions, such as communicating with other equipment via Module I/O and providing the reference angles for each axis via the Backplane for the Axis Control Board. The EMG Board is responsible for the system's emergency actions and can act on all the components, including switching off the FANUC Servo Amplifiers' main contactor and shutting down the motors as a result. In addition, the EMG Board controls the controller's main contactor located in the Main AC Power Distribution which supplies AC voltage to the entire

system, including the PSU which supplies DC voltage to the other boards.

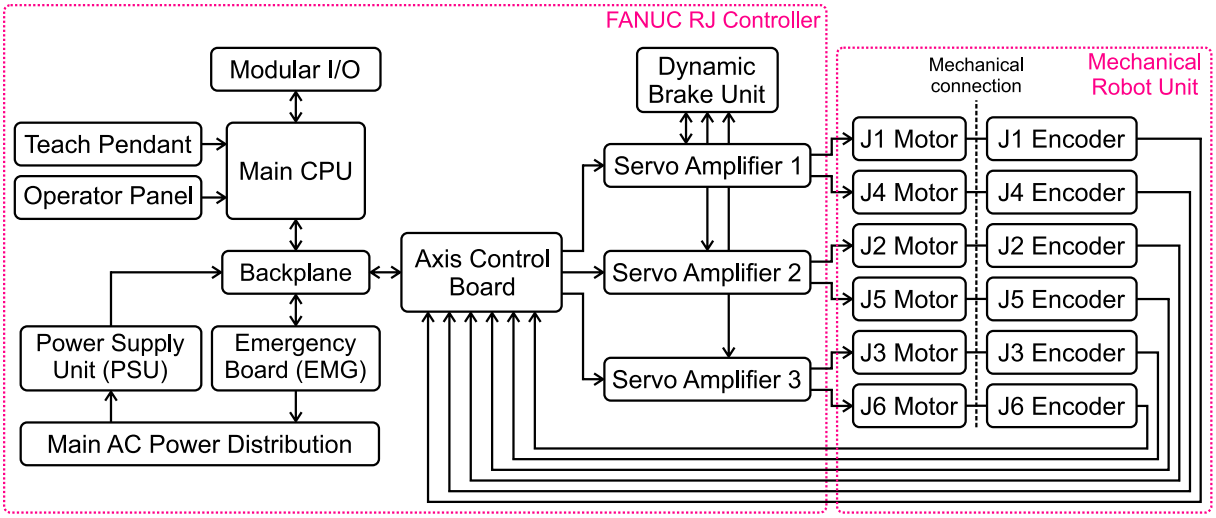


Figure 3.8: Simplified electrical connections of the FANUC RJ Controller parts.

After analysis, it was concluded that the reused parts of the original controller would be the Main AC Power Distribution, the three FANUC Servo Amplifiers, Dynamic Brake Unit, Power Supply Unit and Emergency Board, as well as the robot's mechanical unit and some functions of the Operator Panel.

The main CPU plays an essential role in the system and is responsible for interfacing with the user via the Teach Pendant, communicating with the external environment via Module I/O, calculating kinematics and trajectory control, receiving user commands, as well as other fundamental controller functions. However, this module malfunctions, making it impossible to reuse. The same applies to the Axis Control Board, which performs the fundamental role of controlling the position of the motors, but its communication with the Main CPU via the Backplane is not simple and would require very detailed reverse engineering, so it was decided not to reuse it.

3.1.3.1 FANUC Servo amplifiers

The three FANUC Servo Amplifiers contained in the FANUC RJ Controller are the FANUC AC Servo Amplifier C Series model with part number A06B-6066-H291, capable of controlling two motors via two channels L and M, with the L channel capable of up

to 80A and the M channel up to 40A. This device is responsible for supplying voltage to the motors, corresponding to the power part of a conventional Servo Drive and does not contain any switching or current control, which is performed by the Axis Control Board.

In order to control two motors, most of the components in the FANUC Servo Amplifier are duplicated, such as the gate drive circuit, the three-phase inverter and the current sensor. The components shared between the channels are the main contactor, the three-phase rectifier, the filter capacitors and the emergency and monitoring circuits.

Figure 3.9 shows the internal circuits and arrangement of the FANUC Servo Amplifier's main components. The Figure 3.10 shows the three-phase inverter modules, the three-phase rectifier and the discharge resistor, all attached to the aluminum heat sink.

The emergency module shown in Figure 3.9 monitors the supply voltages and the temperature of the heat sink. This module switches off the main contactor if anything is out of specification.

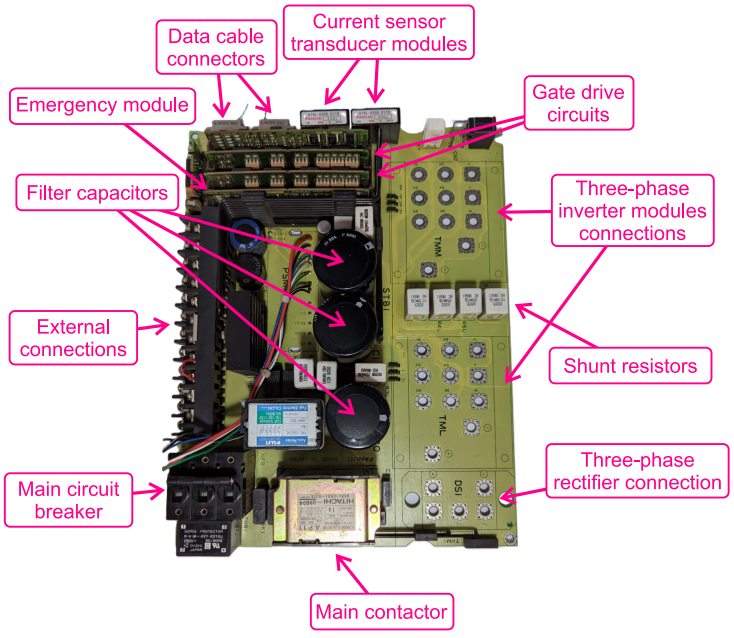


Figure 3.9: Arrangement of the FANUC Servo Amplifier's main components.

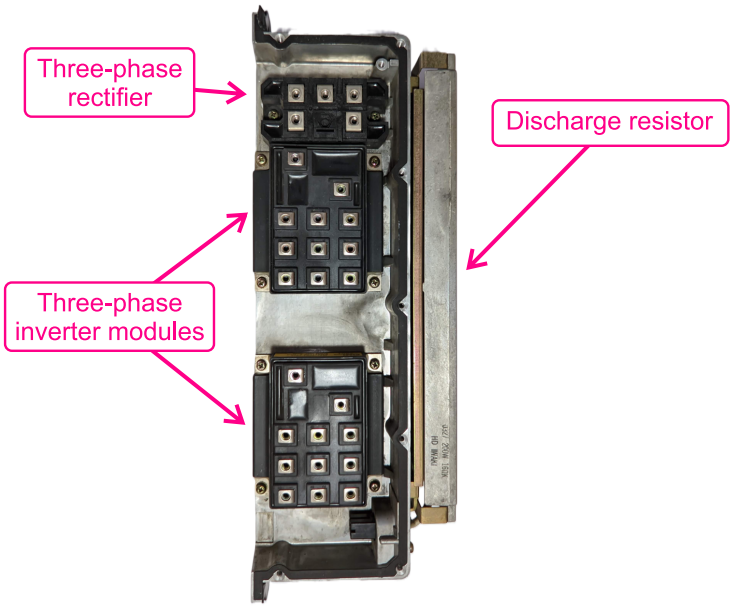


Figure 3.10: Power components attached to the heat sink.

The Servo Amplifier receives the control signals via the two data cables, corresponding to each channel L and M. Figure 3.11 shows the FANUC Servo Amplifier's data cable connector disassembled with the wires showing and Table 3.2 the description of each signal, where the "Pin" column corresponds to the connector's pin number.

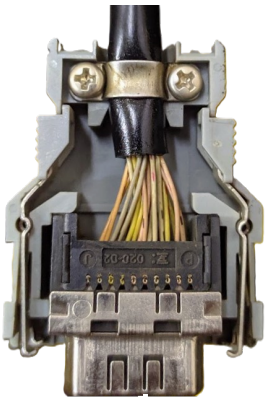


Figure 3.11: FANUC Servo Amplifier data cable connector.

Signal name	Pin	Description
PWM A	3	PWM A signal for switching the three-phase inverter.

PWM B	5	PWM B signal for switching the three-phase inverter.
PWM C	7	PWM C signal for switching the three-phase inverter.
PWM D	13	PWM D signal for switching the three-phase inverter.
PWM E	15	PWM E signal for switching the three-phase inverter.
PWM F	17	PWM F signal for switching the three-phase inverter.
IR	1	Motor phase U current signal.
IS	11	Motor phase V current signal
MCON	10	Signal received by the FANUC Servo Amplifier to switch on the main contactor.
DRDY	20	Signal sent by the FANUC Servo Amplifier to inform that it is ready.

Table 3.2: FANUC Servo Amplifier data signals description.

3.1.3.2 Current sensors of FANUC Servo Amplifiers

The FANUC Servo Amplifiers have two current sensors for each channel for the U and V phases of the motor outputs. This is done via two shunt resistors and the signal is isolated and instrumented by the Current Sensor Transducer Module, as shown in Figure 3.12.

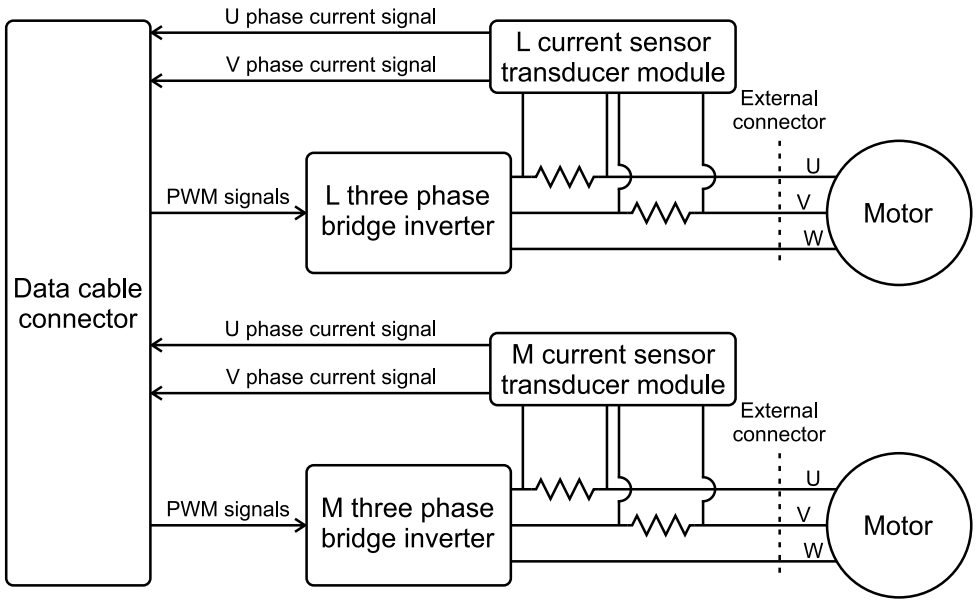


Figure 3.12: Simplified current sensing diagram of the FANUC Servo Amplifier.

The Current Sensor Transducer Module is located on the FANUC Servo Amplifier main board, as shown in Figure 3.9, and has Part Number A76L-0300-0170. Figure 3.13 shows the module in more detail and its connections. In the capture, the module was unsoldered from the board in order to carry out the experiments described in this Section.

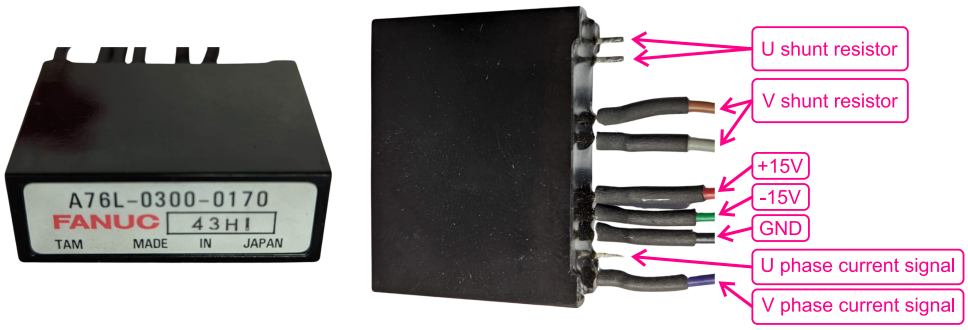


Figure 3.13: FANUC Current Sensor Transducer Module.

The shunt resistors are also located on the FANUC Servo Amplifier's main board, as shown in Figure 3.9, and Figure 3.14 shows them in more detail. As can be seen, the shunt resistor pair for channels L and M have resistance values of 25mΩ and 12.5mΩ respectively.

Figure 3.14: Shunt resistors of the current sensing of FANUC Servo Amplifier.

Considering that the L and M channels have a current capacity of up to 80A and 40A, respectively, Equations 3.2 and 3.3 show the calculation of the maximum voltage drop across the shunt resistors and, consequently, the maximum voltage level that is applied to the inputs of the Current Sensor Transducer Modules, according to Ohm's Law (Equation 3.1).

$$V = R.I \tag{3.1}$$

$$V_{L_{max}} = 25m\Omega.80A = 2V \tag{3.2}$$

$$V_{M_{max}} = 12.5m\Omega.40A = 2V \tag{3.3}$$

It can therefore be concluded that the maximum voltage level applied to the inputs of the Current Sensor Transducer Modules is the same for both channels.

To draw conclusions about the Current Sensor Transducer Module's output signal, a function generator was connected to the module's input, simulating the signal coming from the shunt resistor, and the output signal was measured using an oscilloscope. Figure 3.15 shows the results, where the blue curve represents the input signal and the pink curve represents the module's output signal. The tests were carried out with sinusoidal input signals with different amplitudes ($500mV_{max}$, $1V_{max}$ and $2V_{max}$) and different frequencies ($10Hz$, $100Hz$ and $1000Hz$).

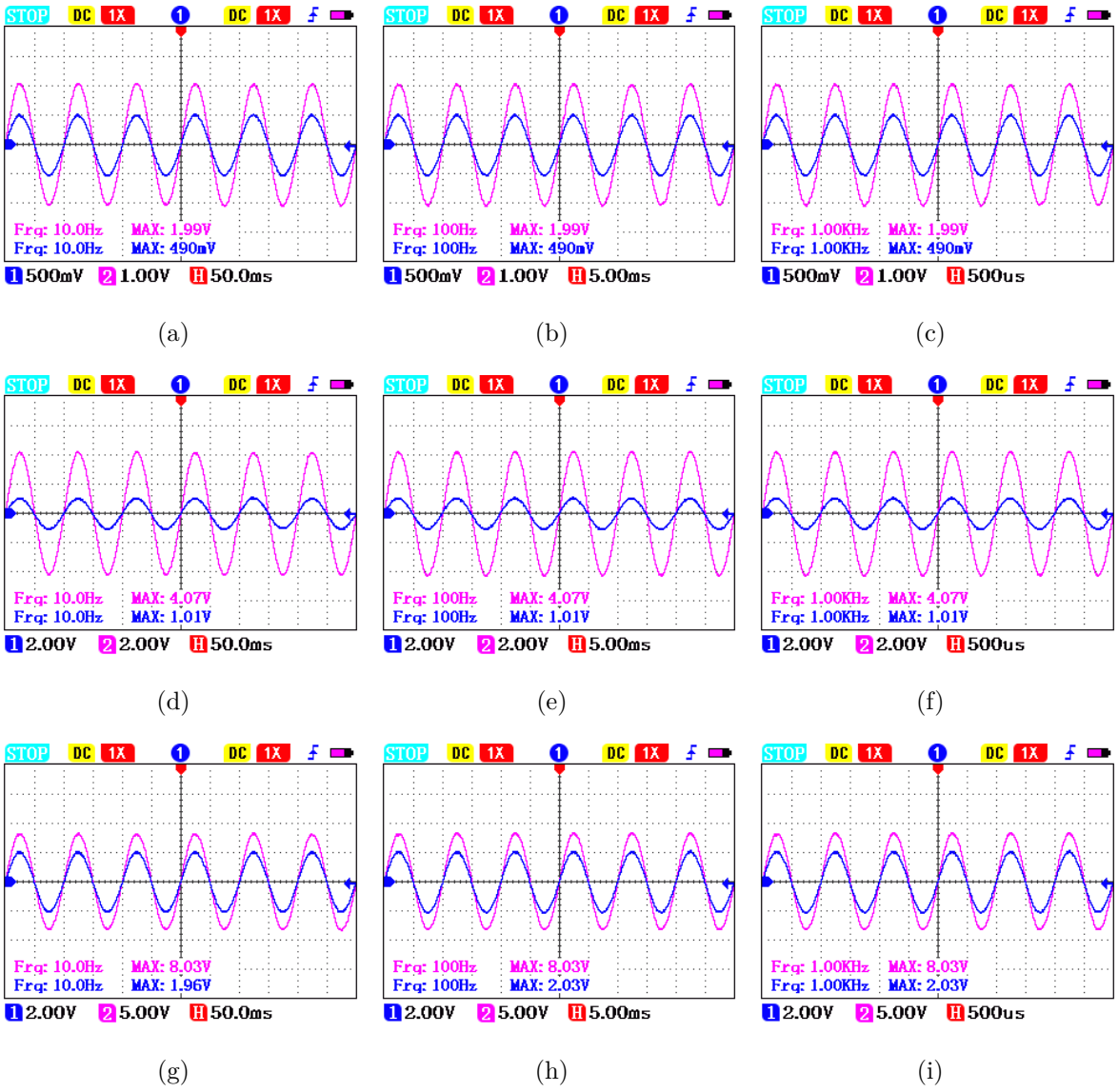


Figure 3.15: Testing the Current Sensor Transducer Module with different input signals. **(a)** $500mV_{max}$ and $10Hz$. **(b)** $500mV_{max}$ and $100Hz$. **(c)** $500mV_{max}$ and $1000Hz$. **(d)** $1V_{max}$ and $10Hz$. **(e)** $1V_{max}$ and $100Hz$. **(f)** $1V_{max}$ and $1000Hz$. **(g)** $2V_{max}$ and $10Hz$. **(h)** $2V_{max}$ and $100Hz$. **(i)** $2V_{max}$ and $1000Hz$.

As a result, it was possible to conclude that the Current Sensor Transducer Module amplifies the input signal four times, as well as isolating the signal. In this way, it is possible to calculate the relationship between the output signal level and the actual current being conducted through the motor phases.

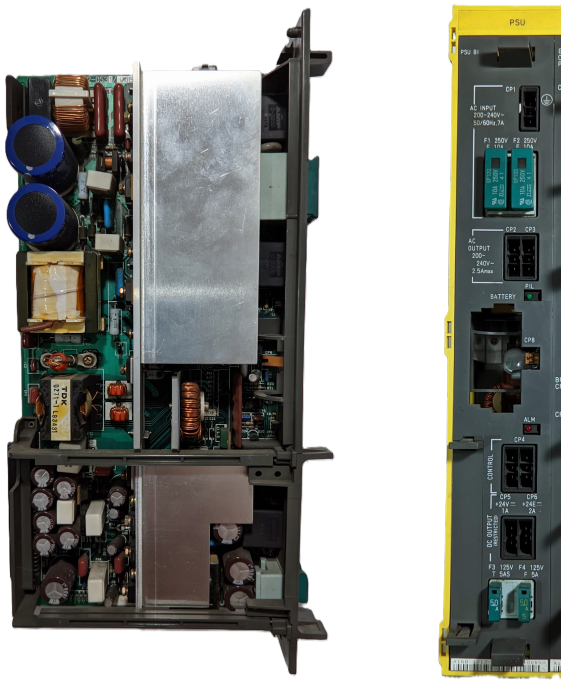
The L channel has a maximum of $80A$ and a maximum voltage of 8V at the output

of the current sensor, resulting in a ratio of $0.1V/A$.

The M channel has a maximum of $40A$ and a maximum voltage of 8V at the output of the current sensor, resulting in a ratio of $0.2V/A$.

3.1.3.3 Power Supply Unit (PSU)

The Power Supply Unit (Figure 3.16) is one of the modules in the Main Controller, shown in Figure 3.7. This unit is responsible for supplying DC voltages to the rest of the system. The voltages supplied by the PSU are: $24V$, $15V$, $-15V$ and $5V$. It is connected to the rest of the system via the Backplane and only receives AC voltage from the front of the module.



(a) (b)

Figure 3.16: FANUC Power Supply Unit.
(a) Board view. **(b)** Front view.

Another important function performed by the PSU is to power up the system. The signals ON, OFF and COM originate from the EMG Board, access the PSU via the

Backplane and cause the PSU to switch on or off. In this way, the Main Controller Modules are energized with DC voltages, the main contactor is switched on and the entire system is supplied with AC voltage. The connections with the EMG Board are described in more detail in Figure 3.21.

The connection between the PSU and the Backplane is made via a 120-pin SCSI Centronics male connector and, given the difficulty in finding this part available, the connection to the system developed in this work will be made by soldering wires directly to the connector contacts on the board, which was also considered an effective and viable solution, given the context of the project. For this purpose, it was necessary to map the connection signals with the Backplane. Figure 3.17 shows the identification and pin numbering of the PSU connector with the Backplane, where the power and non-connected signals are identified in the image and the pins marked in cyan are described in more detail in Table 3.3.

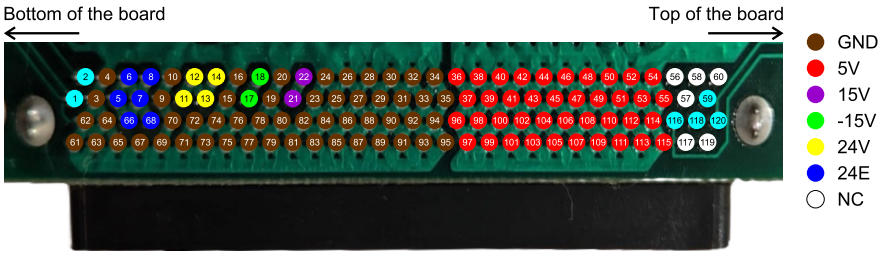


Figure 3.17: Bottom view and pins identification of the Power Supply Unit connector.

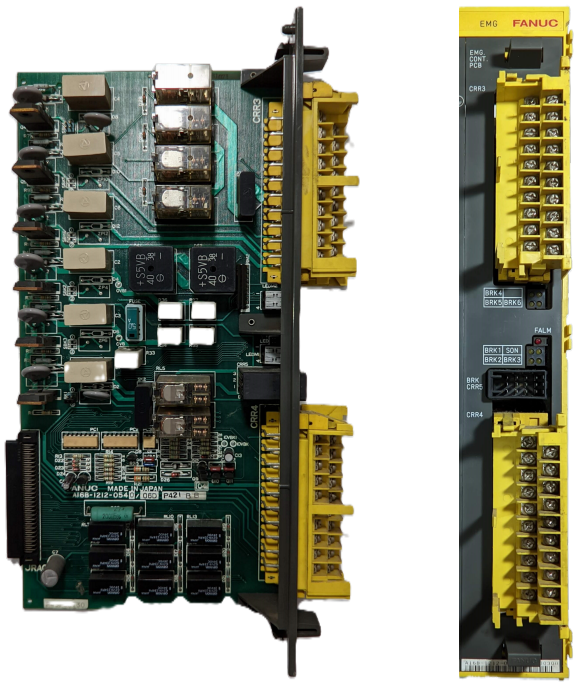
Signal name	Pin	Description
BAT+	1	Positive signal from backup battery.
BAT-	2	Negative signal from backup battery.
ENABLE	59	PSU signal to inform the system that the voltages are being supplied.
COM	116	Control signal to assist in switching the Power Supply Unit on and off.
ON	118	Control signal to switch on the Power Supply Unit.

OFF	120	Control signal to switch off the Power Supply Unit.

Table 3.3: Description and pin numbering of the PSU connector with the Backplane.

3.1.3.4 Emergency Board (EMG Board)

The Emergency Board (Figure 3.18) is one of the modules in the Main Controller, shown in Figure 3.7. This unit is responsible for all the system's primary emergency actions, which include switching off the power to the robot's motors and actuating the mechanical brakes. The board is connected with the Backplane to communicate with the other modules and have two frontal connectors, to send and receive signals from other parts of the controller. In general, the circuit present in the EMG Board is composed of simple components, such as relays, MOSFETs and logic gates, in order for the emergency signals to be actuated without failures or delays.



(a) (b)

Figure 3.18: FANUC Emergency Board.
(a) Board view. **(b)** Front view.

The connection between the EMG Board and the Backplane is made via a 100-pin SCSI Centronics male connector and, given the difficulty in finding this part available, the connection to the system developed in this work will be made by soldering wires directly to the connector contacts on the board, which was also considered an effective and viable solution, given the context of the project. For this purpose, it was necessary to map the connection signals with the Backplane. Figure 3.19 shows the identification and pin numbering of the EMG Board connector with the Backplane, where the power and non-connected signals are identified in the image and the pins marked in cyan are described in more detail in Table 3.4.

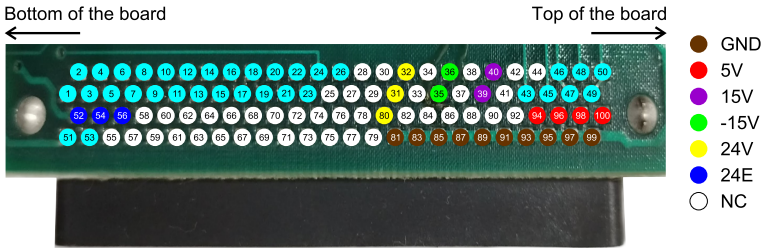


Figure 3.19: Bottom view and pins identification of the Emergency Board connector.

Signal name	Pin	Description
J1 brake solenoid	1	Control signal to actuate the J1 brake solenoid.
10VDC	2	Power supply to assist the brake solenoid control signals.
Auxiliary axis brake	3	Control signal to actuate auxiliary axis brake solenoids.
J2 e J3 brake solenoid	4	Control signal to actuate the J2 and J3 brake solenoids.
J4, J5 e J6 brake solenoid	5	Control signal to actuate the J4, J5 and J6 brake solenoids.
Auxiliary axis brake	6	Control signal to actuate auxiliary axis brake solenoids.

Auxiliary axis brake	7	Control signal to actuate auxiliary axis brake solenoids.
Auxiliary axis brake	8	Control signal to actuate auxiliary axis brake solenoids.
*ROT	9	Signal from the robot's limit switches.
*BRKE	10	Control signal to actuate the relay that supplies the brake solenoid circuit.
*HBKD	11	Signal to report an emergency in the robot's final actuator.
TPREL	12	Signal from an optional button on the Operator Panel to deactivate the system's Teach Pendant functions.
ESPTP	13	Emergency Board signal to inform the system that there is no operational emergency button on the Teach Pendant.
OTREL	14	Control signal to move the robot in an overtravel condition.
O.P. E-STOP	15	Signal from the emergency button of the Operator Panel.
FUSE ALARM	16	Emergency Board signal to inform the system that there are problems with the fuses.
EMGDM	17	Signal from the Teach Pendant's Deadman Switch.
EMGB1	18	Signal from the emergency button of the Teach Pendant.
EMGEN	19	Signal from the ON/OFF button of the Teach Pendant.

EMGB2	20	Signal from the emergency button of the Teach Pendant.
TPDISC	21	Signal to inform if the Teach Pendant is disconnected from the controller.
EMGTP	22	Emergency Board signal to inform the Teach Pendant if there are any active emergencies.
*EMG	23	Emergency Board signal to inform the system that there are no active emergencies.
*FENCE	24	Emergency Board signal to inform the system that the fence switch has been opened.
SOFF	26	Control signal to switch off the power supply of the FANUC Servo Amplifiers.
BT OFF 1	43	Signal from the OFF button of the Operator Panel.
BT OFF 2	45	Signal from the OFF button of the Operator Panel.
COM	46	Emergency Board signal to assist in switching the Power Supply Unit on and off.
BT ON 1	47	Signal from the ON button of the Operator Panel.
OFF	48	Emergency Board signal to switch off the Power Supply Unit.
BT ON 2	49	Signal from the ON button of the Operator Panel.
ON	50	Emergency Board signal to switch on the Power Supply Unit.
24T	51	24V power supply for the Teach Pendant.
24T	53	24V power supply for the Teach Pendant.

Table 3.4: Description and pin numbering of the EMG Board connector with the Backplane.

On the front, the EMG Board has three connectors: CRR3, CRR4 and CRR5. The

CRR5 connector is used to connect the motors' mechanical brakes to the emergency circuit. The CRR4 and CRR5 connectors link the EMG Board to other system components and their respective contacts are shown in Figure 3.20 and their internal functionalities are shown in Figure 3.21.

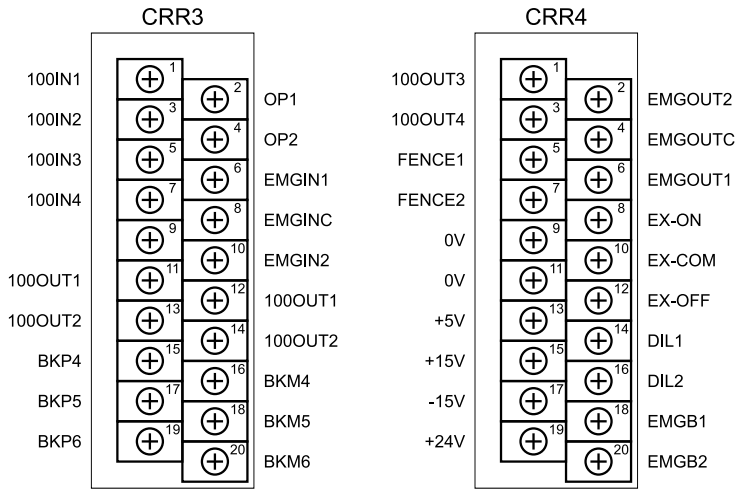


Figure 3.20: Emergency Board CRR3 and CRR4 frontal connectors.

The emergency circuit of the robot is described in Figure 3.21. It is important to note that the TP will not be reused, so the signals coming from it are marked in pink and their states will not be changed during operation.

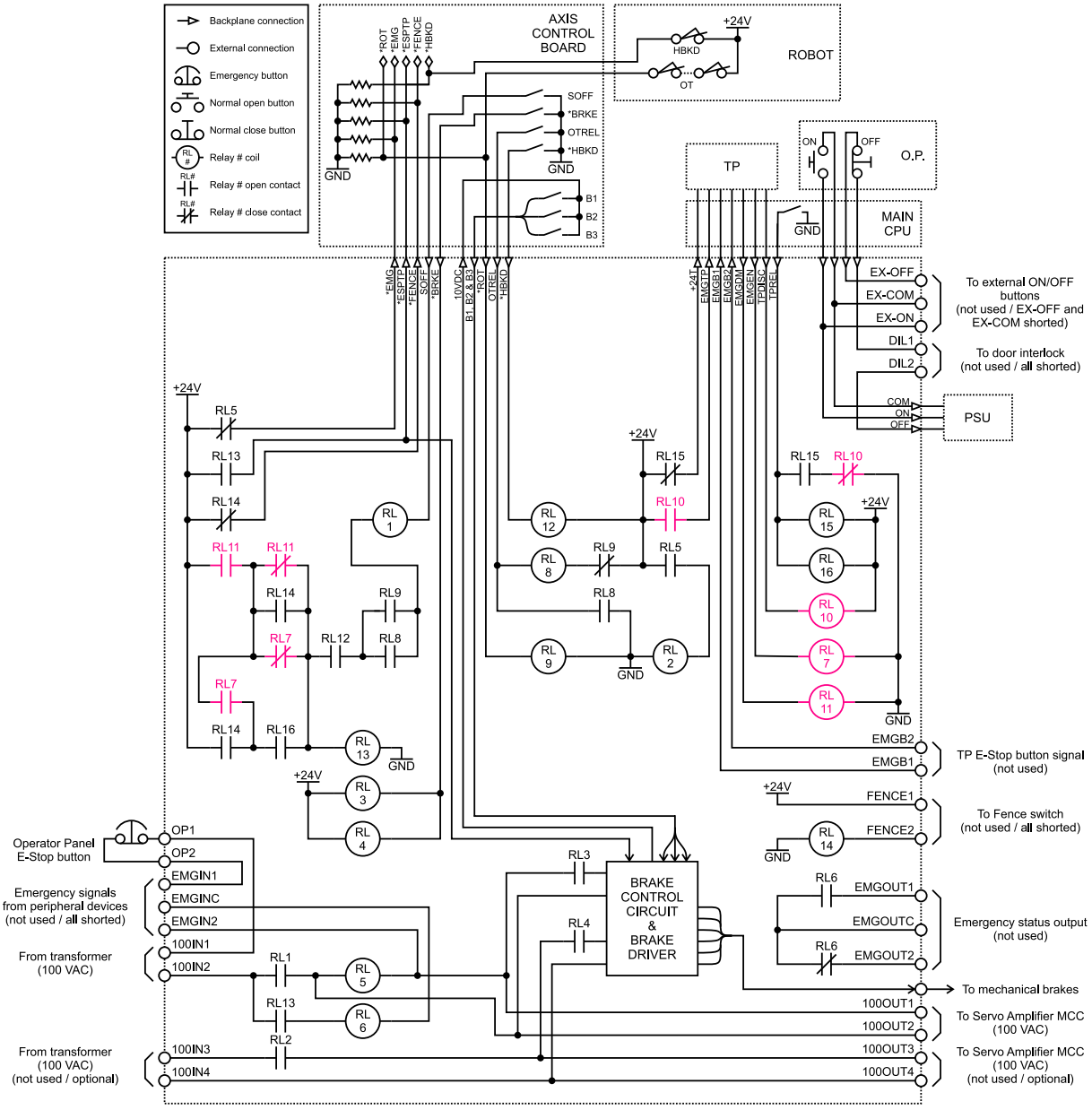


Figure 3.21: Emergency Board simplified circuit.

The TP has important functions in the emergency system, such as the Deadman Switch and an E-Stop button, and in normal situations the robot cannot operate if these signals are not correct or the TP is disconnected from the system. To get around this, there is an optional button on the Operator Panel called Teach Pendant Release, which operates on the TPREL signal and switches on relays RL15 and RL16. The coils of the two relays

are connected in seal with the normal open contact of relay RL15 and is deactivated by the normal closed contact of relay RL10, which is triggered when the TP is physically connected to the system. Therefore, a pulse on the TPREL signal is enough to activate relays RL15 and RL16 until the system is switched off.

The two primary emergency actions are to control the mechanical brakes and supply the main magnetic contactor of FANUC Servo Amplifiers with AC voltage. The conditions that enable these actions are described below.

- **Control the mechanical brakes**

  The brakes are directly powered by the Brake Control Circuit, but a series of conditions must be satisfied before it can be activated.

  The circuit is supplied with 100 VAC via external connections 100IN1 and 100IN2, but this requires the Operator Panel's emergency button to be released and relays RL1 and RL3 to be activated.

  RL1 is activated by the SOFF signal. But for this, relays RL14, RL16, RL12 and RL8 or RL9 must also be activated. This means that the Fence, TPREL, *HBKD and *ROT or OTREL signals must be on.

  Relay RL3 is activated by the *BRKE signal.

  In addition to the 100 VAC supply, the Brake Control Circuit needs a control signal from relay RL13, which is triggered by the Fence (RL14) and TPREL (RL16) signals and the signals B1, B2 e B3 connected with 10VDC, that effectively turns on the MOSFETs in the Brake Control Circuit to drive the brakes coils.

- **Supply the main magnetic contactor of FANUC Servo Amplifiers with AC voltage**

  The EMG Board is connected to the main magnetic contactor of the FANUC Servo Amplifier via external connections 100OUT1 and 100OUT2 and supplies a voltage of 100 VAC.

This power supply originates from external connections 100IN1 and 100IN2, conditioned by the Operator Panel's emergency button and the activation of RL1.

The *HBKD signal originates from the robot's mechanical unit and is forwarded to the EMG Board as a fundamental signal for the system's operation. Although these two signals are named equally, the original signal received by the Axis Control Board is processed and passed on to the EMG Board as programmed in the system. In other words, when an emergency is identified via the *HBKD signal, the Axis Control Board decides if the system will enter an emergency condition or not.

In conclusion, for the robot to operate normally, the SOFF, Fence, TPREL, *HBKD, *ROT and *BRKE signals must be in agreement and the E-Stop button on the Operator Panel must be released. If one of these conditions is not satisfied, the motors are switched off and the mechanical brakes are applied.

3.2 Proposed architecture for the controller

Considering that the Teach Pendant, Main CPU and Axis Control Board will not be reused in the new architecture proposed for the robot controller, as explained in Section 3.1.3, the functionalities performed by these components have been replaced by systems developed throughout this work.

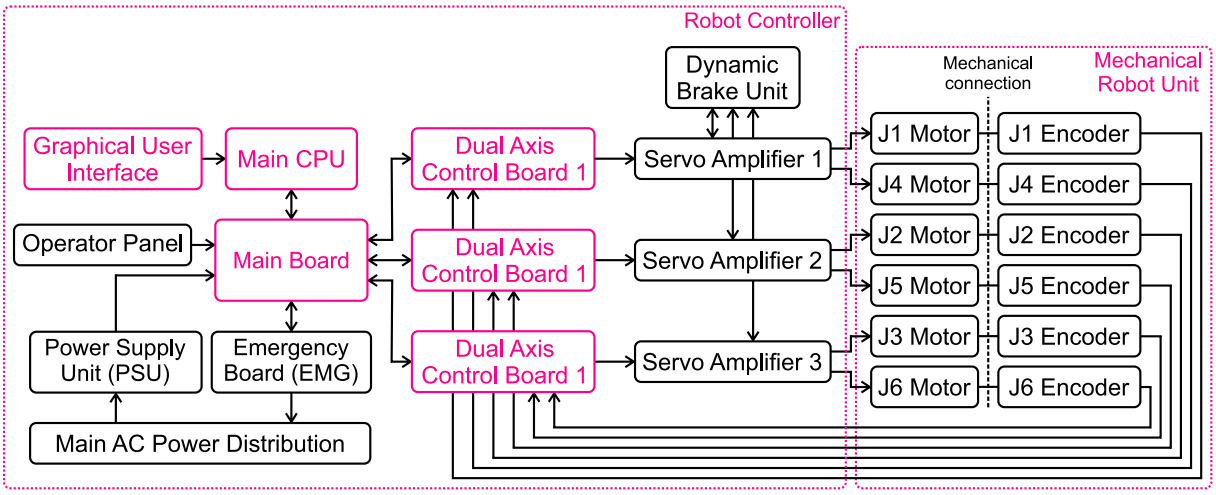


Figure 3.22: Simplified electrical connections of the proposed controller parts.

Figure 3.22 shows the proposed architecture for the new robot controller. Comparing Figure 3.22 with the architecture of the original robot controller (Figure 3.8), it can be seen that new components have been added or replaced and are highlighted in pink. Firstly, the Dual Axis Control Boards have been modulated, and an exclusive board has been proposed to control each FANUC Servo Amplifier and its pair of motors, detailed in Section 3.4. Instead of the Backplane, a "Main Board" was proposed to make the connection between the components, as well as performing other functions that will be detailed Section 3.5. Although the "Main CPU" component has been retained from the original architecture, it is not the same system. This was developed over the course of this work and will be detailed in Section 3.3. The Teach Pendant performed an important user interaction function and this component has been replaced by a Graphical User Interface, detailed in Section 3.7.

3.3 Main CPU controller

The Main CPU mentioned in Figure 3.22 is responsible for the system's trajectory and kinematic calculations, as well as communicating with the Graphical User Interface and the Dual Axis Control Boards. It is also responsible for receiving and sending emergency signals to the Emergency Board.

The microcontroller used in the Main CPU was the ESP32-S3-WROOM-1 mounted on the ESP32-S3-DevKitC-1 development platform (Figure 3.23). This microcontroller has two 32-bit LX7 Xtensa processing cores that work at 240MHz by default, 16MB of flash memory, 8MB of PSRAM memory, 45 General Purpose Input/Output (GPIO), 3 Universal Asynchronous Receiver/Transmitter (UART) communications and a native USB On-The-Go (USB-OTG) communication, as well as other peripherals not used in this work. The supply voltage of this microcontroller is $3.3V$ and its GPIOs also work with this voltage level.

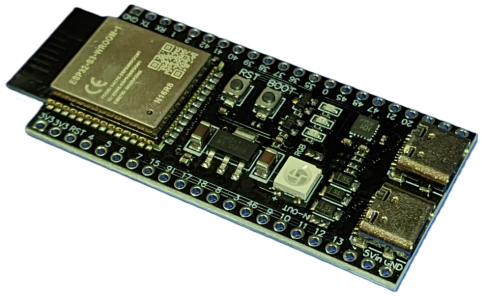


Figure 3.23: ESP32-S3-DevKitC-1 development platform.

The specifications of the microcontroller version used were sufficient, considering the performance shown in Chapter 4 and the number of communication ports and GPIOs provided. Figure 3.24 shows the peripherals used in the communication between the Main CPU and the other components of the system. Serial communications 0, 1 and 2 are used to communicate with the Dual Axis Control Boards via the UART hardware peripherals and USB-OTG is used in serial communication mode to communicate with the GUI. All the communications mentioned are made using cables and physical connections.

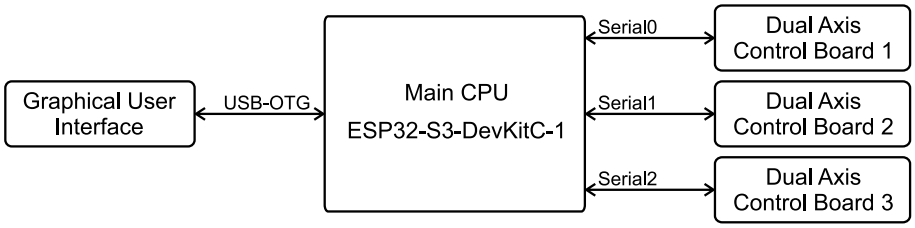
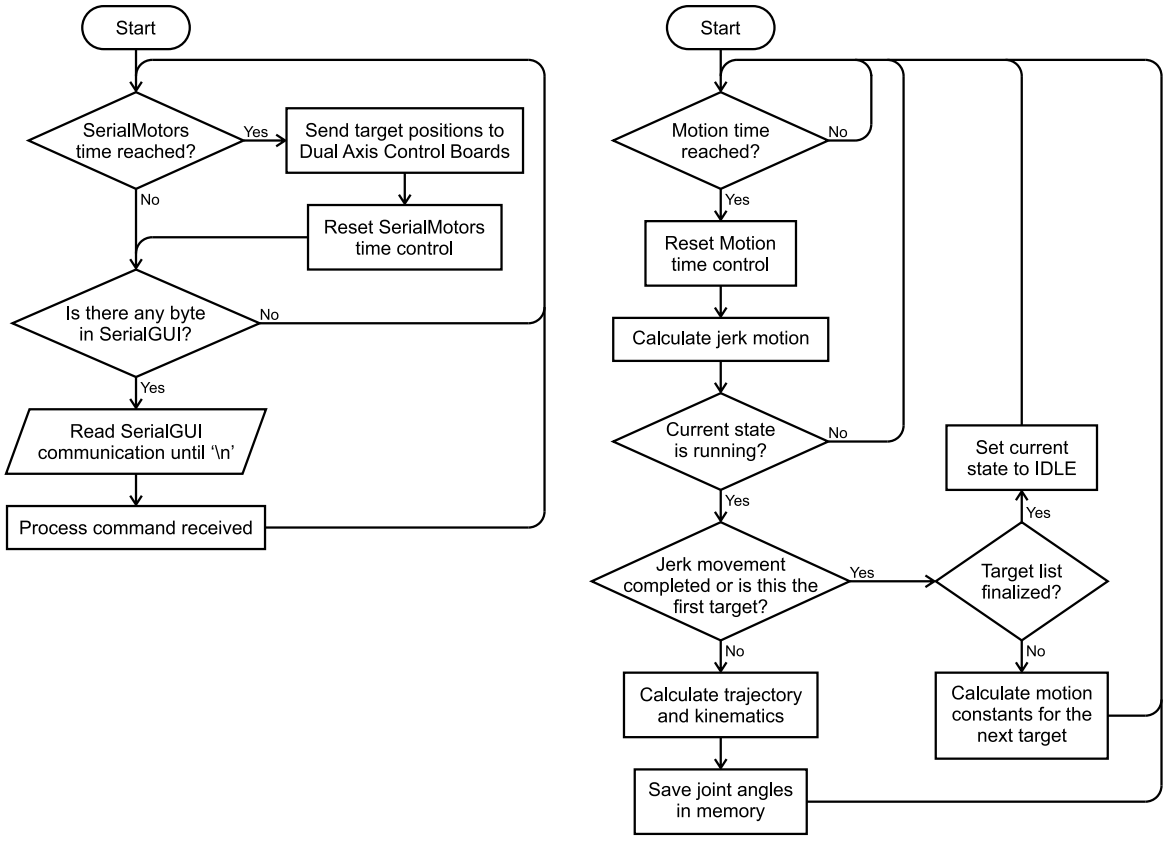


Figure 3.24: Peripherals used in Main CPU's communications.

The algorithm developed to be executed on the Main CPU is divided into two main parts (Figure 3.25), which are executed on different cores of the microprocessor, i.e. are executed simultaneously and in parallel by the microprocessor.



(a) (b)

Figure 3.25: Main tasks performed by the Main CPU's microcontroller. **(a)** Task performed by core 1. **(b)** Task performed by core 0.

Figure 3.25(a) shows the task performed by core 1, which corresponds to the communications involved in controlling the system. Initially, it checks if the elapsed time is greater than or equal to the period defined for sending the angular references of the robot's joints to the Dual Axis Control Boards via serial communication. Next, it is checked if there is any information in the buffer of the serial communication used with the GUI and then the message is read and processed according to the specified in the Section 3.3.1. This process is repeated indefinitely.

Figure 3.25(b) shows the task performed by core 0, which corresponds to the calculations related to controlling the robot's movement. Initially, it checks that the elapsed time is greater than or equal to the period set to perform the calculations related to the

robot's movement and then calculates the jerk motion according to what was discussed in Section X. Next, it checks if the robot's current state is "Running". If not, execution returns to waiting for the next period of algorithm execution. If yes, it continues by checking if the jerk trajectory for the current target of the movement has been completed or if the current target is the first to be executed of the programmed movement. If these conditions are met, the program proceeds to calculate the trajectory, interpolate the axes and calculate the kinematics, resulting in the angle of the robot's joints for that instant of movement. If the conditions do not match, it is checked if the programmed movement has been completed. If yes, the robot's current state is set to "IDLE". If not, new movement constants are calculated, and the execution is returned to check the algorithm's execution period for both. This process is repeated indefinitely.

All temporal control performed by the Main CPU is based on the execution time provided by the microcontroller in microseconds and the execution times set for the tasks were $2000\mu s$ for the task performed by core 1 and $200\mu s$ for the task performed by core 0.

3.3.1 Graphical User Interface serial communication

The Graphical User Interface is described in more detail in Section 3.7, but with regard to its communication, Figure 3.24 shows that USB-OTG communication in serial mode is used to communicate with the GUI. Communication between the Main CPU and the GUI follows a defined protocol inspired by Geometric Code (G-Code), where the GUI sends a command to the Main CPU and it responds with the requested information, preventing the Main CPU from sending spontaneous messages.

Messages sent by the GUI are always in text format, however the response in the Main CPU may contain messages in text format when they are sporadic messages or binary messages when they are messages sent at higher frequencies. When messages are in binary format, a protocol is used where the first byte is the message code, followed by the size in bytes of the message and the information to be transmitted. Figure 3.26 shows an example byte sequence for this protocol.

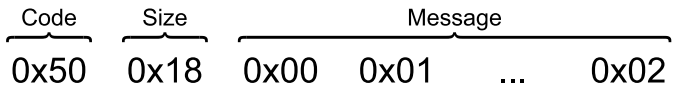


Figure 3.26: Example byte sequence for fast messages.

The Table 3.5 shows the commands received by the Main CPU from the GUI to request some information from the system and the response description, including the code, size and description of the response sent back to the GUI, according to the format described in the Figure 3.26.

Command received	**Response**		
	Code	**Size**	**Message description**
M114	"P" or 0x50	25 bytes	Six 4 bytes float numbers representing the actual cartesian position of the robot and one byte number representing the status of the system.
M115	"A" or 0x41	25 bytes	Six 4 bytes float numbers representing the angles of each joint of the robot and one byte number representing the status of the system.
M117	"T" or 0x54	25 bytes	Six 4 bytes float numbers representing the frame of the actual end effector configured and one byte number representing the status of the system

Table 3.5: Commands to request information from the system and description of the response.

The Table 3.6 shows simple commands received by the Main CPU and their description. The response sent back to the GUI is "ok" to inform if the command has been processed or "no" if it is an invalid command.

Command received	Description
R1	Command start the trajectory configured.
R0	Command stop the current trajectory performed by the system.
M116	Command erases the configured targets.

Table 3.6: Simple commands accepted by the Main CPU.

The Table 3.7 shows commands used to add new targets for the trajectory to be executed by the robot. The response sent back to the GUI is "ok" to inform if the command has been processed or "no" if it is an invalid command.

Command received	Example	Description
G0	G0 X1820 Y0 Z2170 A0 B90 C180	Command to add a new joint-to-joint movement to the current trajectory.
G1	G1 X1500 Y500 Z1750 A0 B0 C90	Command to add a new linear movement to the current trajectory.

Table 3.7: Commands to add new targets for the trajectory.

3.3.2 Dual Axis Control Board serial communication

Figure 3.24 shows that three serial communications are used to send and receive commands from the Dual Axis Control Boards, i.e. communication between the Main CPU and the Dual Axis Control Boards occurs individually for each of them.

The algorithm incorporated into the Dual Axis Control Board will be described in more detail in Section 2.1.4, but it includes a suite of communication protocols with features to drive and monitor the motor control system.

The serial communication present on the Dual Axis Control Board respects the UART protocol and follows an architecture where the device connected to it requests information or sends commands and it responds with the requested information or a confirmation of the command sent. A more detailed description of the format of the messages received and sent by the Dual Axis Control Board can be found in "*<installation folder>\Documentation\html\motor-control-protocol-suite.html*", available after installation of the tool described in Section 2.1.4.

3.4 Dual Axis Control Board

The Dual Axis Control Board is an interface board between the Main Board, the STM32 NUCLEO board, the FANUC Servo Amplifier and the motor encoders.

To interact with the Main Board, a 24-pin connector was used, where the signals are specified according to Figure 3.27, which are summarized as power supply signals ($+5V$, $+15V$ and $-15V$) and UART communication (RX and TX pins).

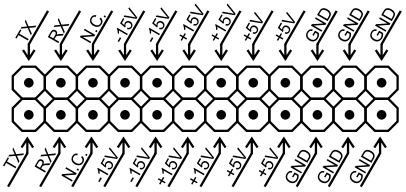
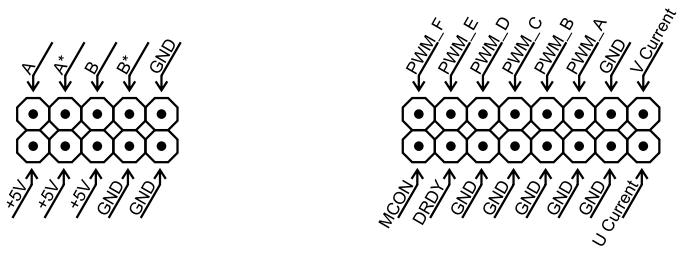


Figure 3.27: Dual Axis Control Board connector with the Main Board.

The STM32 NUCLEO board is embedded with the MCSDK specified in Section 2.1.4 and connected directly to the Dual Axis Control Board, sharing all its pins.

The FANUC Servo Amplifiers and motor encoders are connected to the system via DB15 and DB9 connectors respectively, but the signals are routed to the Dual Axis Control Board via a flat cable with a DB9/DB15 connector and a female pin header to connect to the board. The connectors are duplicated for control the L and M channels of the FANUC Servo Amplifier and the pins on the Dual Axis Control Board are specified in Figure 3.28.



(a) (b)

Figure 3.28: Dual Axis Control Board connectors.
(a) Encoder connector. **(b)** FANUC Servo Amplifier connector.

As explained in Section 3.1.3.1 and detailed in Table 3.2, the interface with the FANUC Servo Amplifier is via two channels that control two motors independently. These channels receive six PWM signals and a control signal to drive the device's main contactor and send two current signals and a signal to inform it that it is ready to perform.

All these signals are processed by the STM32 NUCLEO board. In order to protect the microcontroller from possible signal interference that could damage it, the digital signals are isolated by optocouplers. The MCON and DRDY signals are slow signals, which means that the system does not need to act immediately. Therefore, PC817 optocouplers were used for these signals. Figure 3.29 shows the interface with the MCON signal transmitted to the FANUC Servo Amplifier using the isolation circuit described.

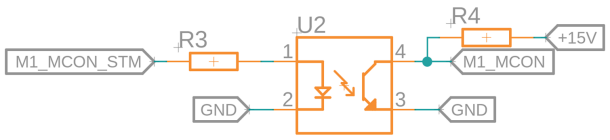


Figure 3.29: Low-frequency signal optocoupler circuit using PC817.

PWM signals are fast signals, in the order of tens of kilohertz. For these, VO2630 optocouplers were used, a model commonly used in telecommunications, where the signals are of a high-frequency nature. The standard operation of this model of optocoupler inverts the level of the output signal when compared to the input signal. To avoid this

problem, the signal from the microcontroller was connected to the cathode of the component's input, so that the output signal remains in the same phase as the input signal and avoiding possible control problems. This circuit is shown in Figure 3.30.

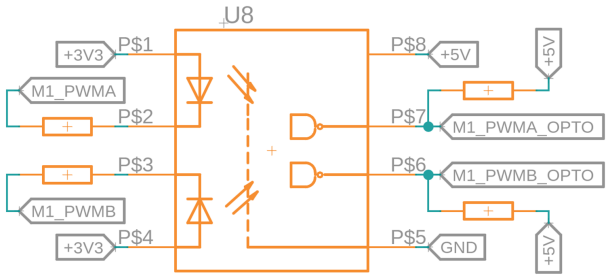


Figure 3.30: High-frequency signal optocoupler circuit using VO2630.

The microcontroller used in the Dual Axis Control Board is the STM32F446RE model mounted on the STM32 NUCLEO platform. This microcontroller works with signals from $0V$ to $3.3V$ on its analog inputs, so it was necessary to instrument the current signals coming from the FANUC Servo Amplifiers. As mentioned in Section 3.1.3.2, the current signals are limited to between $-8V$ and $+8V$ and must therefore be treated to behave between $0V$ and $3.3V$. To do this, the circuit shown in Figure 3.31 was developed, where the signal is modified using a dual LM358 operational amplifier.

The "Vin" signal mentioned in Figure 3.31 refers to the current signal coming from the FANUC Servo Amplifier and this is directed to the inverting input of one of the operational amplifier channels, where the amplitude of the signal is altered according to the setting of the variable resistor referred to as "TR1". The signal is then directed to the inverting input of the other channel of the operational amplifier, where it is shifted to positive voltages according to the setting of the variable resistor referred to as "TR2".

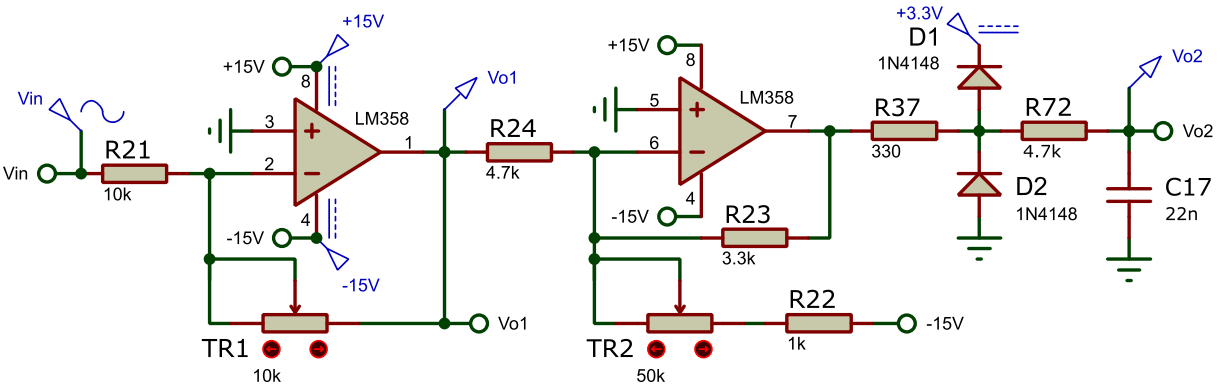


Figure 3.31: Current signal processing circuit of the FANUC Servo Amplifier.

The circuit described was simulated and validated using the software Proteus and the graphical result of the signal manipulation is shown in Figure 3.32, where the green signal "Vin" represents the signal coming from the FANUC Servo Amplifier varying in its maximum amplitude of $-8V$ and $+8V$, the red signal "Vo1" represents the signal with reduced amplitude at the output of the first operational amplifier and the blue signal "Vo2" represents the signal at positive levels ready to be read by the microcontroller.

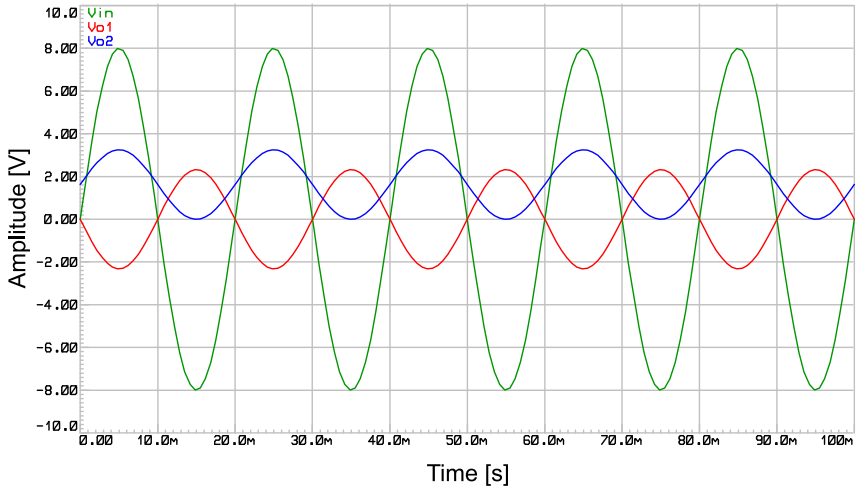


Figure 3.32: Result of the current signal processing circuit in normal operation.

An extra safety circuit has also been added using two 1N4148 fast-acting diodes to prevent the voltage levels from exceeding the specified level. A low-pass filter with a cut-off frequency of $1.5kHz$ was also added to filter out any unwanted noise in the signal, following the microcontroller manufacturer's recommendations [44]. An undesirable simulated

situation is shown in Figure 3.33, in which the signal amplitude exceeds that supported by the microcontroller, but the circuit mentioned maintains voltage levels between -0.7V and +4V, which saturates the microcontroller's analog inputs, but is not enough to damage them.

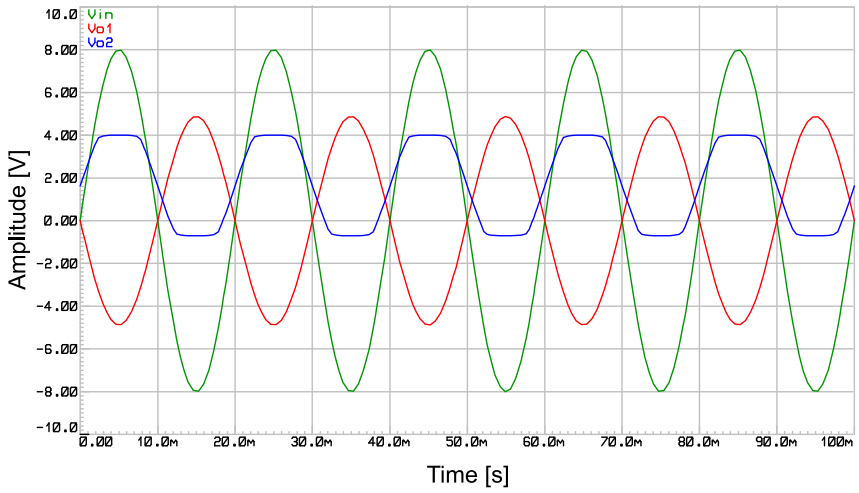


Figure 3.33: Result of the current signal processing circuit with saturated amplitude.

The signals expected from the encoder are of type AB and their $\bar{A}\bar{B}$ complements. This signal configuration is commonly used to filter out any noise that might affect the encoder signals. For this purpose, the MAX9122 integrated circuit was used to receive the differential signals, manipulate them to remove unwanted noise and supply the AB signals which are connected directly to the microcontroller.

This integrated circuit operates with $3.3V$ signals, but the voltage level of the encoders is $5V$, so a voltage divider with $10k\Omega$ and $20k\Omega$ was used to reduce the voltage level of each encoder signal. The Figure 3.34 shows the circuit described.

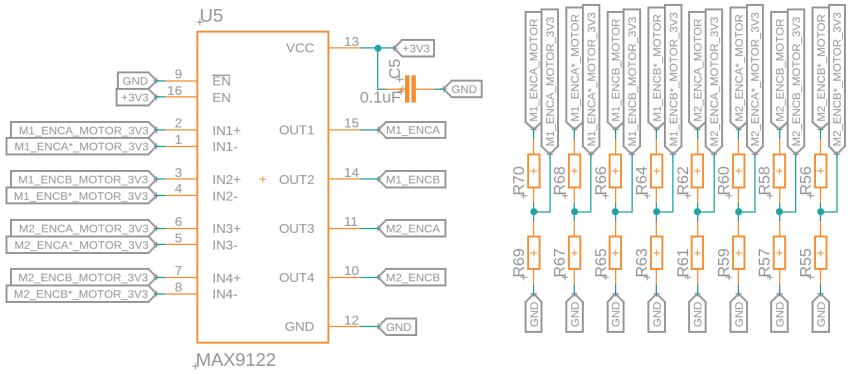
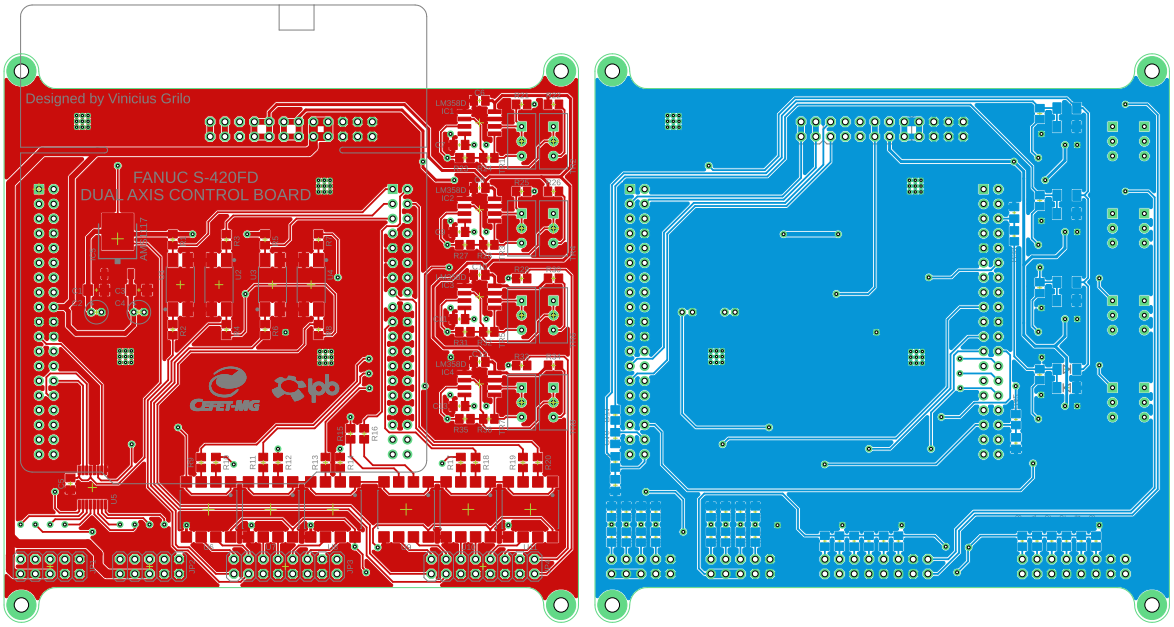


Figure 3.34: High-frequency signal optocoupler circuit using VO2630.

The design of the Dual Axis Control Board's Printed Circuit Board (PCB) was developed to accommodate the components in a favorable way for assembly with the rest of the system, with smaller dimensions and ease of use. The PCB design can be seen in Figure 3.35 with dimensions of 99mm x 98mm and the three dimensional (3D) project in Figure 3.36.



(a) (b)

Figure 3.35: Dual Axis Control Board. **(a)** Top. **(b)** Bottom.

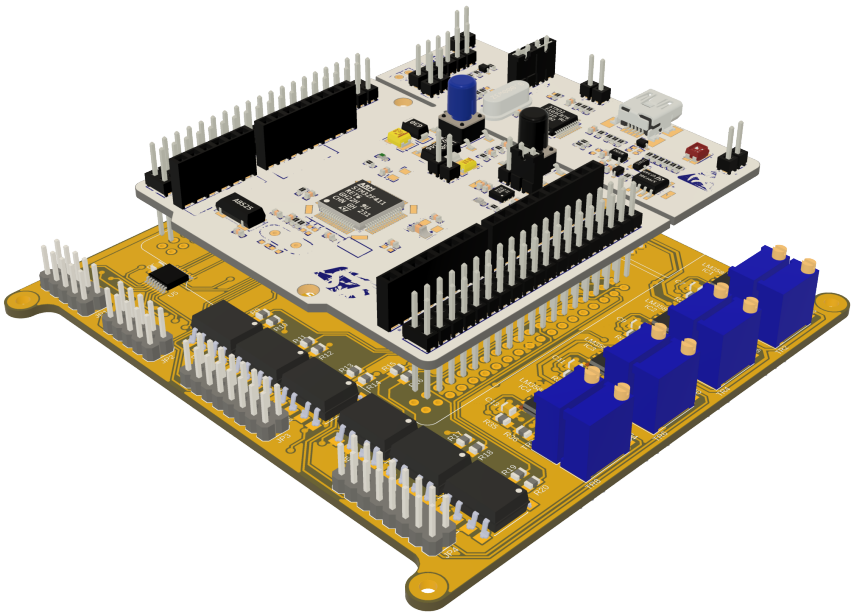


Figure 3.36: Dual Axis Control Board 3D project.

The complete Dual Axis Control Board circuit can be found in Appendix B.1.

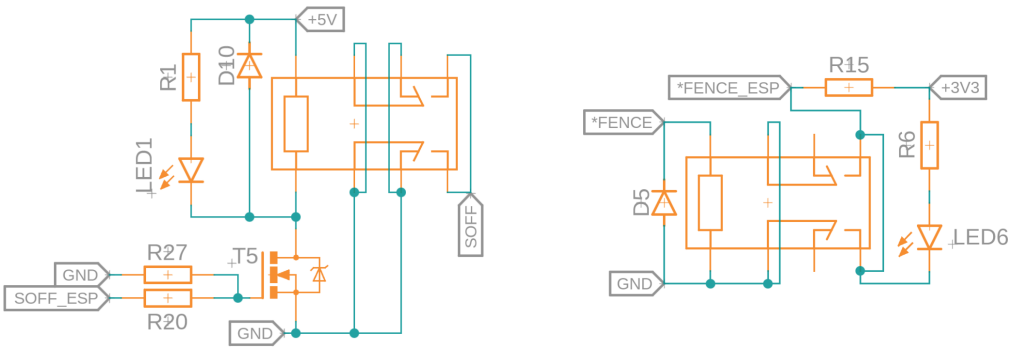
3.5 Controller Main Board

The Controller Main Board is an interface board between the three Dual Axis Control Boards, the Main CPU, the Power Supply Unit, the Emergency Board, receiving signals from the Operator Panel and the robot's emergency signals. Figure 3.37(a).

The interface with the three Dual Axis Control Boards is via three connectors as illustrated in Figure 3.27, in addition to providing mechanical supports for the boards.

As mentioned in Section 3.3, the microcontroller used as the system's main controller works with voltages of $3.3V$, but the signals originating from the robot, the EMG Board and the PSU are 24V. Relays were used to make the signals compatible. Figure 3.37(a) shows the use of a mosfet to drive a 5V relay with a $3.3V$ control signal, where its normally open contacts are connected to the 24V signal to be manipulated. Figure 3.37(b) shows the use of a 24V relay triggered by a signal from the EMG Board, where its normally open contacts are directed to the microcontroller's digital inputs. Both circuits have an

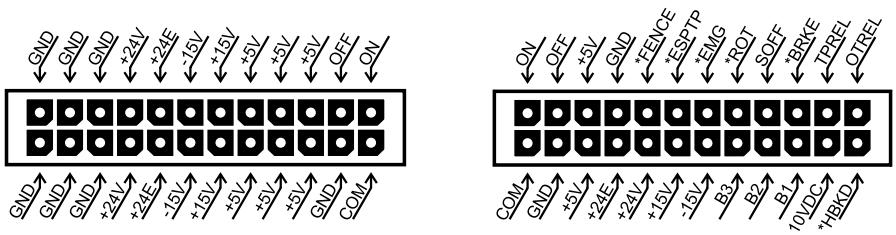
LED to indicate when the respective relays are activated. More details on the 24V control signals can be seen in Figure 3.21, which shows the signals coming from other parts of the system and which are manipulated with replicas of the circuits mentioned. The Main Board Controller also has a $3.3V$ voltage regulator to power the LEDs and other parts of the circuit.



(a) (b)

Figure 3.37: Relay drive circuit. **(a)** 5V relay circuit. **(b)** 24V relay circuit.

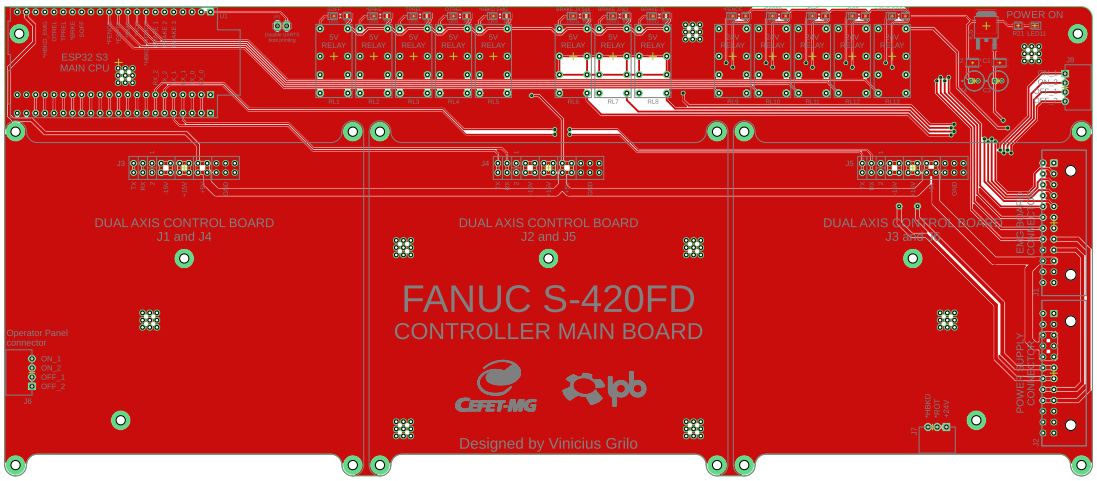
The physical connection between the Main Board Controller and the PSU and EMG Board is made via Molex Micro-Fit 43045-2400 [45] connectors and the signal configuration can be seen in Figure 3.38 in frontal view of the connectors. The Main Board Controller also has three more connectors, two of which are duplicated to transmit the signals from the ON and OFF buttons on the Operator Panel to the EMG Board, and another connector to transmit the *ROT and *HDBK signals coming from the robot to the EMG Board. These connectors are the HX 90 degree model [46] with 4 and 3 pins respectively.



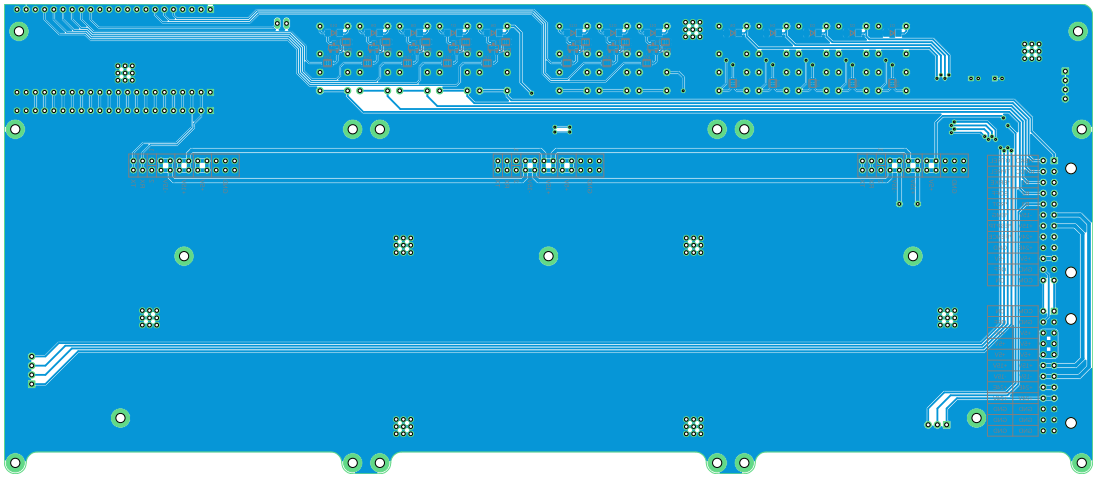
(a) (b)

Figure 3.38: Main Board Controller connectors.
(a) Power Supply Unit connector. **(b)** Emergency Board connector.

The circuit design of the Controller Main Board has been developed to accommodate the three Dual Axis Control Boards, the Main CPU, the interface relays and the physical connections in a that is favorable for assembly and use and can be seen in Figure 3.39 with dimensions of 300mm x 129.5mm, where Figure 3.39(a) shows the top circuit and Figure 3.39(b) shows the bottom circuit.



(a)



(b)

Figure 3.39: Controller Main Board. **(a)** Top. **(b)** Bottom.

The Figure 3.40 shows the 3D project of the Controller Main Board and the Figure 3.41 shows the Controller Main Board with the Dual Axis Control Boards connected in the 3D project.

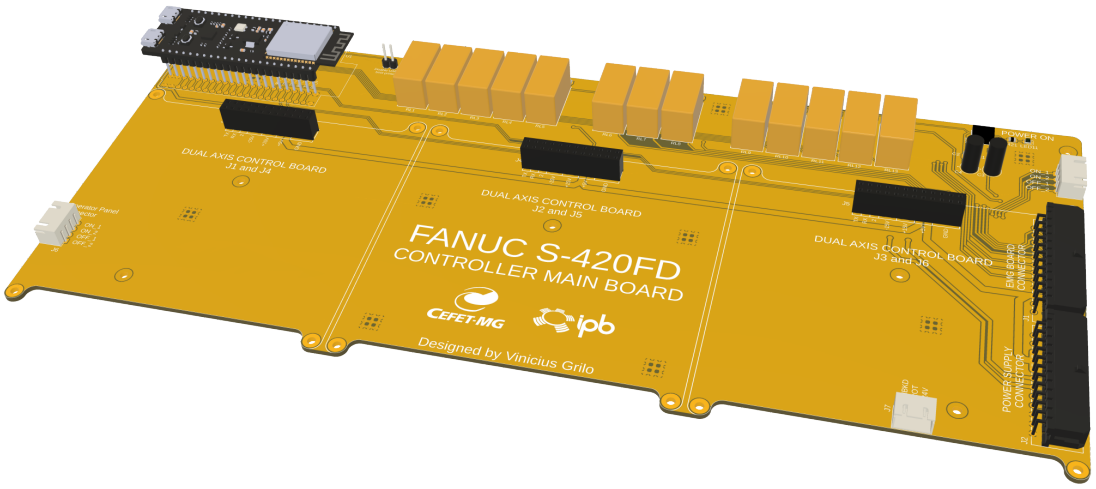


Figure 3.40: Controller Main Board 3D project.

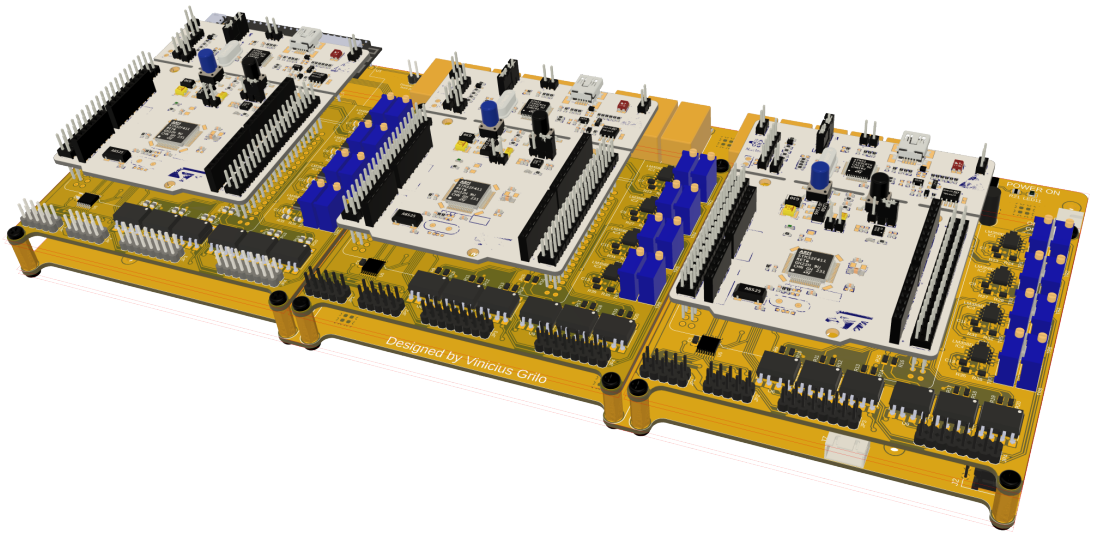


Figure 3.41: Controller Main Board 3D project with the Dual Axis Control Boards attached.

Signal name	GPIO	Function	Description
RX_1	1	Output	RX signal of the serial communication with the Dual Axis Control Board 1.
TX_1	2	Input	TX signal of the serial communication with the Dual Axis Control Board 1.

B3_ESP	5	Output	Control signal to actuate the J4, J5 and J6 brake solenoids.
B2_ESP	6	Output	Control signal to actuate the J2 and J3 brake solenoids.
B1_ESP	7	Output	Control signal to actuate the J1 brake solenoid.
*FENCE_ESP	8	Input	Signal indicating a Fence emergency.
SOFF_ESP	9	Output	Control signal to actuate the main contactor of the FANUC Servo Amplifiers.
*BRKE_ESP	10	Output	Control signal to actuate the power supply for the Brake Control Circuit.
TPREL_ESP	11	Output	Control signal to deactivate the system's Teach Pendant functions.
OTREL_ESP	12	Output	Control signal to move the robot in an overtravel condition.
*HBKD_EMG_ESP	13	Output	Control signal to report an emergency in the robot's final actuator.
*HBKD_ROBOT_ESP	15	Input	Signal from robot to report an emergency in the robot's final actuator.
*ROT_ESP	16	Input	Signal from the robot's limit switches.
*EMG_ESP	17	Input	Signal from Emergency Board to inform that there are no active emergencies.
*ESPTP_ESP	18	Input	Signal from Emergency Board to inform that there is no operational emergency button on the Teach Pendant.

RX_2	41	Output	RX signal of the serial communication with the Dual Axis Control Board 2.
TX_2	42	Input	TX signal of the serial communication with the Dual Axis Control Board 2.
RX_0	43	Output	RX signal of the serial communication with the Dual Axis Control Board 0.
TX_0	44	Input	TX signal of the serial communication with the Dual Axis Control Board 0.
JUMPER_PRINT	46	Input	Configuration pin to deactivate the ROM messages on the serial communication.

Table 3.8: Description and pin numbering of ESP32 emergency signals.

During normal system operation the conditions described in Section 3.1.3.4 relating to EMG Board emergency signals must be respected. Otherwise, the EMG Board will actuate the mechanical brakes and disconnect the power supply to the FANUC Servo Amplifiers. In addition, the Main CPU must also monitor the emergency signals coming from the emergency board and the robot in order to switch off the operation of the motors and send the emergency information and robot status to the GUI.

Another important emergency-related function assigned to the Main CPU is during system start-up, where it is necessary to work in collaboration with the Dual Axis Control Board. During system start-up, emergencies can occur in relation to the FANUC Servo Amplifier even if there are no active emergencies on the EMG Board, since the main power supply to this device is switched off via its main contactor and will only be activated at this moment, making this phase critical for the system.

Therefore, for the system to be fully initialized and for the motors to be energized, the Main CPU must first send a command to the three Dual Axis Control Boards to activate the MCON signal of each channel of the FANUC Servo Amplifiers for the main contactors

to be activated. After this, the DRDY signals from the FANUC Servo Amplifiers should be read, confirming the normal operation of the units. If the DRDY signals do not remain turned on after the MCON has been activated, this means that there has been an internal emergency in the FANUC Servo Amplifier. With this, the system must abort the initialization and alert the user to check for possible emergencies in the FANUC Servo Amplifiers. The handling of MCON and DRDY signals is not included in MCSDK, so it had to be implemented in addition to the original implementation of the tool. The signals received and sent by the FANUC Servo Amplifiers are described in more detail in Section 3.1.3.1.

The complete Controller Main Board circuit can be found in Appendix B.2.

3.6 Kinematics and trajectory control

Kinematics and trajectory control perform an important role in the controller, since these calculations allow the robot to behave according to what has been programmed by the user.

Both topics refer to calculations involving the geometric analysis of the robot, including the configuration and distance between joints. A method of representing the robot's geometry in mathematical terms is the use of the Denavit-Hatemberg method, which is discussed in this section and facilitates the kinematic analyses referred to in this chapter.

3.6.1 Denavit-Hartenberg parameters

The Denavit-Hatemberg parameters, discussed in more detail in Section 2.1.6, refer to a way of representing the geometry and configuration of the robot's joints in mathematical terms.

Figure 3.42 shows the configuration of the robot's joints, where each joint is represented by a frame in which the Z axis is aligned with the rotation axis of the joint.

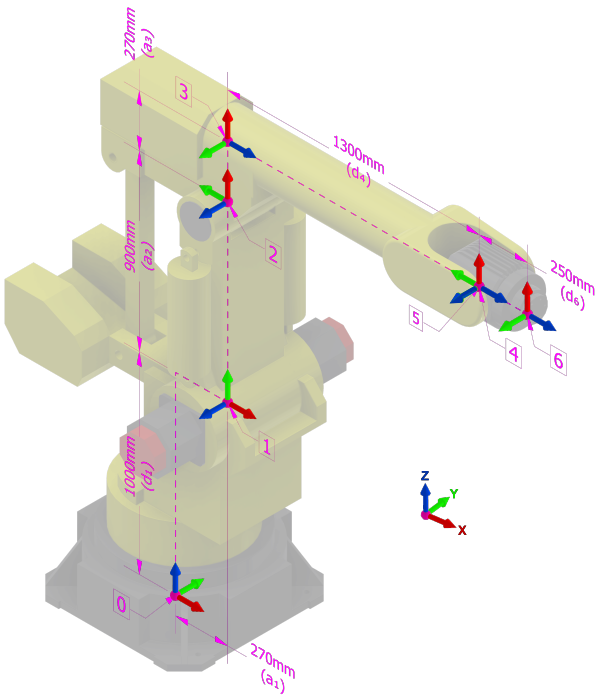


Figure 3.42: Main dimensions and frame configuration of FANUC S-420FD.

Figure 3.43 shows the measurements between the joints, where the notation d is used when the displacement is in the direction of the Z axis of the joint in question and the notation a is used when the displacement is perpendicular to the axis of rotation.

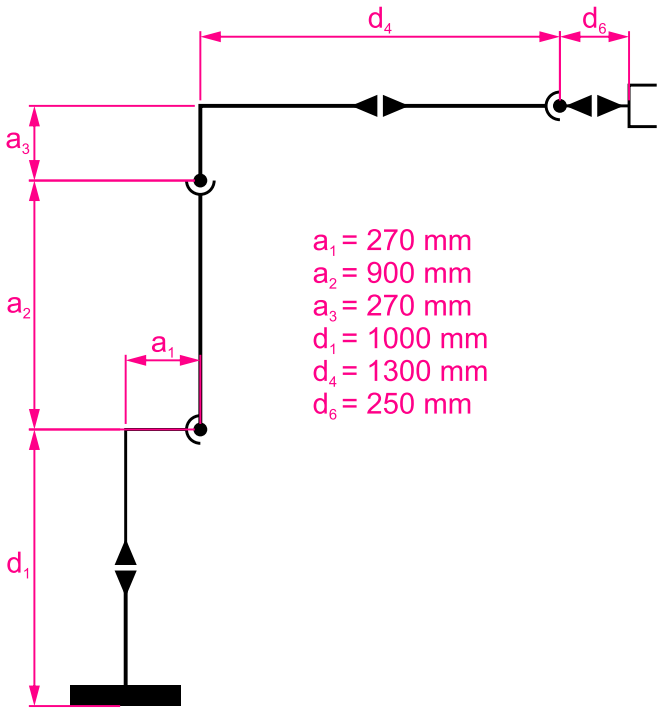


Figure 3.43: Main dimensions representation of FANUC S-420FD.

With the information shown in Figures 3.42 and 3.43, it is possible to develop the matrix shown in Table 3.9 containing the Denavit-Hatemberg parameters and the angular limit of each joint. These parameters and notations will be used to develop the kinematic equations shown in later sections.

$axis$	θ_i	α_i	d_i[mm]	a_i[mm]	$range_i$
1	θ_1	90°	1000	270	330°(±165°)
2	θ_2+90°	0°	0	900	115°(−50°/ + 65°)
3	θ_3	90°	0	270	130°(−100°/ + 30°)
4	θ_4	−90°	1300	0	480°(±140°)
5	θ_5	90°	0	0	260°(±130°)
6	θ_6	0°	250	0	720°(±360°)

Table 3.9: Denavit-Hartenberg parameters of FANUC S-420FD.

3.6.2 Forward kinematics

Following this concept presented by Jacques Denavit and Richard S. Hartenberg [30], the matrices presented in Equation 3.4 represent the homogeneous transformation matrices of each joint of the robot, following the parameters presented in Chapter 2, Table 3.9.

$$A_1 = \begin{bmatrix} cos(\theta_1) & 0 & sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & -cos(\theta_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 0 & a_2cos(\theta_2) \\ sin(\theta_2) & cos(\theta_2) & 0 & a_2sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} cos(\theta_3) & 0 & sin(\theta_3) & a_3cos(\theta_3) \\ sin(\theta_3) & 0 & -cos(\theta_3) & a_3sin(\theta_3) \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4 = \begin{bmatrix} cos(\theta_4) & 0 & -sin(\theta_4) & 0 \\ sin(\theta_4) & 0 & cos(\theta_4) & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} cos(\theta_5) & 0 & sin(\theta_5) & 0 \\ sin(\theta_5) & 0 & -cos(\theta_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6 = \begin{bmatrix} cos(\theta_6) & -sin(\theta_6) & 0 & 0 \\ sin(\theta_6) & cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The geometric representation of the matrices presented in Equation 3.4 is shown in Figure 3.44, where each frame shown in the figure represents an A_n matrix.

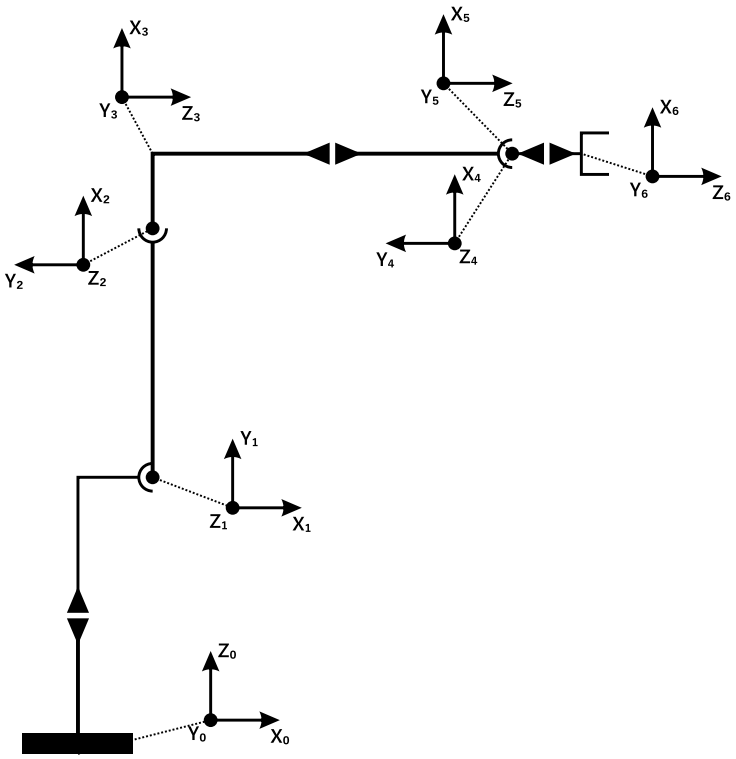


Figure 3.44: Frames representation of FANUC S-420FD.

The Forward Kinematics using this method is obtained by multiplying the homogeneous transformation matrices represented by each joint of the robot, according to the Equation 3.5, resulting in the homogeneous transformation matrix of the end effector in relation to the base of the robot.

$$T_f = \prod_{n=1}^{6} A_n = A_1 A_2 A_3 A_4 A_5 A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

To simplify the representation of the solution to Equation 3.5, the notation shown in Equations 3.6 and 3.7 were used, where $cos(\theta_n)$ and $sin(\theta_n)$ are simplified to C_n and S_n, respectively, and C_{ab} and S_{ab} are the simplifications of $C_a C_b - S_a S_b$ and $C_a S_b - S_a C_b$, respectively.

$$C_n = cos(\theta_n) \qquad S_n = sin(\theta_n) \tag{3.6}$$

$$C_{ab} = C_a C_b - S_a S_b \qquad S_{ab} = C_a S_b - S_a C_b \tag{3.7}$$

Equations 3.8 to 3.19 show the solution of the homogeneous transformation matrix of the end effector in relation to the base of the robot considering the angle of each joint presented in Equation 3.5 using the notations presented in the Equations 3.6 and 3.7.

$$n_x = C_6(C_5(C_1C_{23}C_4 + S_1S_4) - C_1S_{23}S_5) + S_6(S_1C_4 - C_1C_{23}S_4) \tag{3.8}$$

$$n_y = C_6(C_5(S_1C_{23}C_4 + C_1S_4) - S_1S_{23}S_5) - S_6(C_1C_4 + S_1C_{23}S_4) \tag{3.9}$$

$$n_z = C_6(C_{23}S_5 + S_{23}C_4C_5) - S_{23}S_4S_6 \tag{3.10}$$

$$o_x = S_6(C_1S_{23}S_5 - C_5(S_1C_{23}C_4 + S_1S_4)) + C_6(S_1C_4 - C_1C_{23}S_4) \tag{3.11}$$

$$o_y = S_6(S_1S_{23}S_5 - C_5(S_1C_{23}C_4 + C_1S_4)) - C_6(C_1C_4 + S_1C_{23}S_4) \tag{3.12}$$

$$o_z = -S_6(C_{23}S_5 + S_{23}C_4C_5) - S_{23}S_4C_6 \tag{3.13}$$

$$a_x = S_5(C_1C_{23}C_4 + S_1S_4) + C_1S_{23}C_5 \tag{3.14}$$

$$a_y = S_5(S_1C_{23}C_4 - C_1S_4) + S_1S_{23}C_5 \tag{3.15}$$

$$a_z = S_{23}C_4S_5 - C_{23}C_5 \tag{3.16}$$

$$p_x = d_6(S_5(C_1C_{23}C_4 + S_1S_4) + C_1S_{23}C_5) + C_1(a_1 + a_2C_2 + a_3C_{23} + d_4S_{23}) \tag{3.17}$$

$$p_y = d_6(S_5(S_1C_{23}C_4 + C_1S_4) + S_1S_{23}C_5) + S_1(a_1 + a_2C_2 + a_3C_{23} + d_4S_{23}) \tag{3.18}$$

$$p_z = a_2S2f + d_1 + a_3S_{23} - d_4C_{23} + d_6(S_{23}C_4S_5 - C_{23}C_5) \tag{3.19}$$

All the calculations mentioned in this section have been implemented in C++ and embedded in an ESP32-S3. The codes related to the Forward Kinematics calculation can be found in Appendix A.1.

3.6.3 Inverse kinematics

Inverse Kinematics in robotic manipulators is a crucial concept, where the goal is to determine the angles of the robot's joints based on a desired position and orientation for

its end effector. In other words, inverse kinematics allows to calculate the configurations that the robot must assume in order to reach a specific point in the workspace or perform a specific task [35], [36].

As mentioned in Chapter 2, the process for determining the angles of each joint, given a desired translation and orientation for the end effector, can be divided into two steps:

1. Calculate the angles of the first three joints (θ_1, θ_2 and θ_3);

2. Calculate the angles of the last three joints (θ_4, θ_5 and θ_6), represented by spherical joints, using the calculated angles of the first three joints.

Theta 1 (θ_1)

The Figure 3.45 shows a graphical representation to calculate the angle of Joint 1 (J1) (θ_1), showing the front and top views of the pose demonstrated by the robot. The definition of the vectors present in Figure 3.45 are presented in the Equations 3.20 and 3.21.

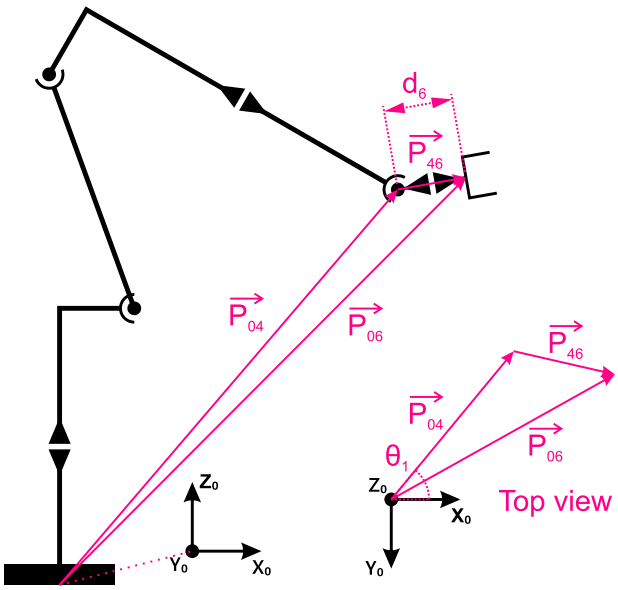


Figure 3.45: Geometric representation of the calculation of angle $\theta 1$ in inverse kinematics [35], [36].

$$\overrightarrow{P_{04}} = \overrightarrow{P_{06}} - \overrightarrow{P_{46}} \quad , where \quad \overrightarrow{P_{06}} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad , \quad \overrightarrow{P_{46}} = d_6 \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{3.20}$$

$$\overrightarrow{P_{04}} = \begin{bmatrix} x_{05} \\ y_{05} \\ z_{05} \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} - d_6 \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} p_x - d_6.a_x \\ p_y - d_6.a_y \\ p_z - d_6.a_z \end{bmatrix} \tag{3.21}$$

As can be seen in the top view displayed in Figure 3.45, θ_1 is described by the angle of vector $\overrightarrow{P_{04}}$ with respect to the X axis in the XY plane. Therefore, the angle θ_1 can be calculated according to Equation 3.22.

$$\theta_1 = arctan\left(\frac{p_y - d_6.a_y}{p_x - d_6.a_x}\right) \tag{3.22}$$

Theta 3 (θ_3)

The Figure 3.46 shows a graphical representation to calculate the angle of J3 (θ_3). The definition of the vectors present in Figure 3.46 are presented in the Equations 3.23.

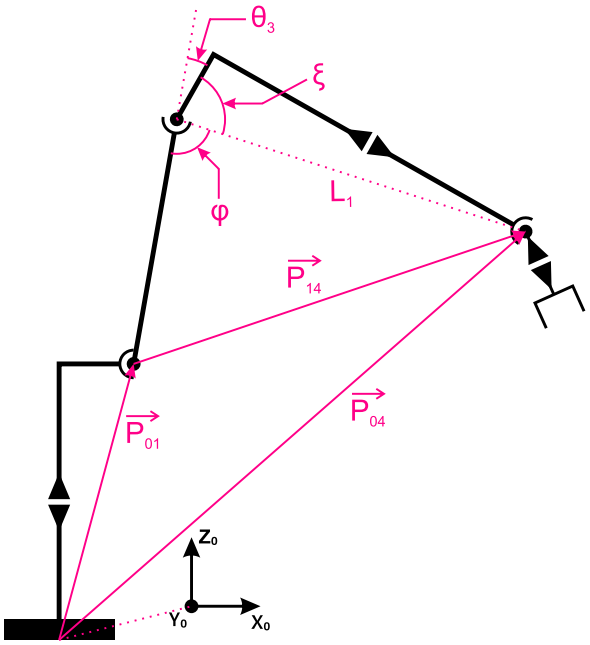


Figure 3.46: Geometric representation of the calculation of angle $\theta 3$ in inverse kinematics [35], [36].

$$\overrightarrow{P_{02}} = \begin{bmatrix} x_{02} \\ y_{02} \\ z_{02} \end{bmatrix} = \begin{bmatrix} a_1.C_1 \\ a_1.S_1 \\ d_1 \end{bmatrix} \quad , \quad \overrightarrow{P_{04}} = \begin{bmatrix} x_{04} \\ y_{04} \\ z_{04} \end{bmatrix} = \begin{bmatrix} p_x - d_6.a_x \\ p_y - d_6.a_y \\ p_z - d_6.a_z \end{bmatrix} \tag{3.23}$$

To define the vector $\overrightarrow{P_{14}}$ simply subtract $\overrightarrow{P_{04}} - \overrightarrow{P_{02}}$, as shown in Equation 3.24.

$$\overrightarrow{P_{14}} = \begin{bmatrix} x_{14} \\ y_{14} \\ z_{14} \end{bmatrix} = \overrightarrow{P_{04}} - \overrightarrow{P_{02}} = \begin{bmatrix} p_x - d_6.a_x - a_1.C_1 \\ p_y - d_6.a_y - a_1.S_1 \\ p_z - d_6.a_z - d_1 \end{bmatrix} \tag{3.24}$$

It is possible to calculate the angle φ shown in Figure 3.46 using the cosine law, but for this it is necessary to define the length of L_1 and the vector $\overrightarrow{P_{14}}$. These definitions are shown in Equation 3.25, where the subscript L represents the length of the vector.

$$P_{14L} = \sqrt{{x_{14}}^2 + {y_{14}}^2 + {z_{14}}^2} \quad , \quad L_1 = \sqrt{{a_3}^2 + {d_4}^2} \tag{3.25}$$

Therefore, the equations presented in 3.26 show the calculation of the angle φ through the law of cosines and the angle ε through the relationship between the measurements d_4 and a_3.

$$\varphi = arccos\left(\frac{{L_1}^2 + {a_2}^2 - {P_{14L}}^2}{2.L_1.a_2}\right) \quad , \quad \varepsilon = arctan\left(\frac{d_4}{a_3}\right) \tag{3.26}$$

Finally, the angle θ_3 is defined according to Equation 3.27.

$$\theta_3 = \varphi + \varepsilon - \pi \tag{3.27}$$

Theta 2 (θ_2)

The Figure 3.47 shows a graphical representation to calculate the angle of J2 (θ_2).

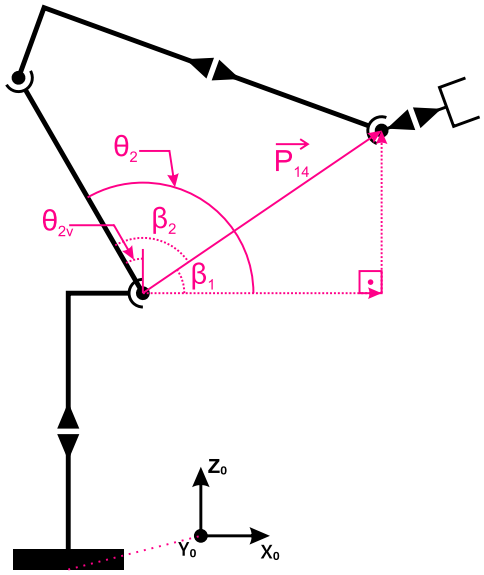


Figure 3.47: Geometric representation of the calculation of angle $\theta 2$ in inverse kinematics [35], [36].

The angle β_1 represents the angle between the vector $\overrightarrow{P_{14}}$ and the X axis in XZ plane. This calculation is shown in Equation 3.28.

$$\beta_1 = arctan\left(\frac{z_{14}}{\sqrt{{x_{14}}^2 + {y_{14}}^2}}\right) \tag{3.28}$$

The angle β_2 can be calculated using the law of cosines, as shown in Equation 3.29.

$$\beta_2 = arccos\left(\frac{{a_2}^2 + {P_{14L}}^2 - {L_1}^2}{2.a_2.P_{14L}}\right) \tag{3.29}$$

As shown in Figure 3.47, the angle θ_2 is defined by the sum of β_1 and β_2. This result is shown in Equation 3.30.

$$\theta_2 = \theta_{2v} + \frac{\pi}{2} = \beta_1 + \beta_2 \tag{3.30}$$

Theta 5 (θ_5)

For the calculation of the angle of Joint 5 (J5) (θ_5) it is necessary to assume that the angle of Joint 4 (J4) (θ_4) is equal to zero. Thus, θ_5 is defined by the angle between vectors

$\overrightarrow{R_{04}}$ and $\overrightarrow{R_{06}}$ shown in Figure 3.48.

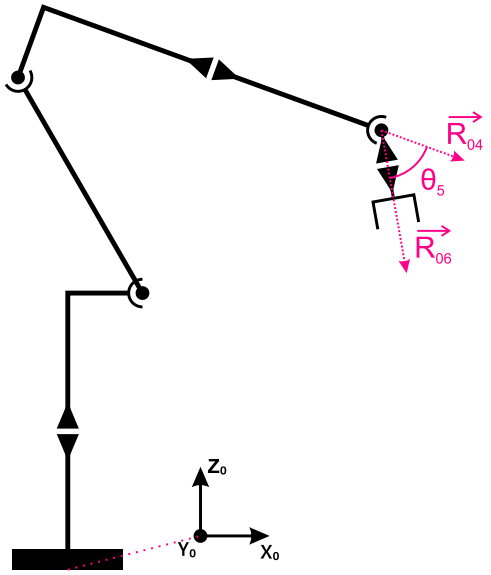


Figure 3.48: Geometric representation of the calculation of angle $\theta 5$ in inverse kinematics [35], [36].

Considering A_1, A_2 and A_3 presented in the Equations 3.4, the homogeneous transformation matrix for J4/J5 is defined in Equation 3.31.

$$A_{04} = A_1.A_2.A_3 = \begin{bmatrix} C_1.C_{23} & S_1 & C_1.C_{23} & C_1.(a_1 + a_2.C_2 + a_3.C_{23}) \\ S_1.C_{23} & -C_1 & S_1.C_{23} & S_1.(a_1 + a_2.C_2 + a_3.C_{23} \\ S_{23} & 0 & -C_{23} & a_2.S_2 + d_1 + a_3.S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

The orientation of J6, and consequently of vector $\overrightarrow{R_{06}}$ present in Figure 3.48, is equal to the desired orientation for the end effector. According to the orientations defined for the reference frames of the robot joints, only the rotation around the Z axis will be considered for the calculation. These two definitions are shown in Equation 3.32.

$$\overrightarrow{R_6} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \qquad \overrightarrow{R_{4z}} = \begin{bmatrix} C_1.C_{23} \\ S_1.C_{23} \\ -C_{23} \end{bmatrix} \tag{3.32}$$

With this, the angle θ_5 of J5 is found through Equation 3.33.

$$\theta_5 = arccos(\overrightarrow{R_{6z}}.\overrightarrow{R_{3z}}) \tag{3.33}$$

Theta 4 (θ_4) **and theta 6** (θ_6)

The angles of J4 and J6 are calculated using the same method. To do this, the matrix A_{46} must be defined according to Equation 3.34.

$$A_{46} = A_4.A_5.A_6 = \begin{bmatrix} E_x & F_x & G_x & H_x \\ E_y & F_y & G_y & H_y \\ E_z & F_z & G_z & H_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

$$\begin{array}{llll} E_x = C_4C_5C_6 - S_4S_6 & F_x = -(C_4C_5S_6) & G_x = C_4S_5 & H_x = C_4S_5d_6 \\ E_y = S_4C_5C_6 - C_4S_6 & F_y = -S_4C_5S_6 + C_4C_6 & G_y = S_4S_5 & H_y = S_4S_5d_6 \\ E_z = -S_5C_6 & F_z = S_5S_6 & G_z = C_5 & H_z = C_5d_6 + d_4 \end{array}$$

Therefore, the angles θ_4 and θ_6 are defined in Equation 3.35.

$$\theta_4 = arctan\left(\frac{G_y}{G_x}\right) \qquad \theta_6 = arctan\left(\frac{F_z}{-E_z}\right) \tag{3.35}$$

All the calculations mentioned in this section have been implemented in C++ and embedded in an ESP32-S3. The codes related to the Inverse Kinematics calculation can be found in Appendix A.2.

3.6.4 Trajectory control

Trajectory control in robotic systems refers to the ability of a robot to follow a desired trajectory in a precise and controlled path. This involves defining a specific path for the robot to follow and implementing algorithms and control techniques to ensure that the robot follows this path as expected.

Two trajectory control approaches were developed for this work. Linear trajectories, where the end effector performs the rectilinear trajectory from the start point to the end point, and joint-to-joint trajectories, where the robot performs the trajectory from the start point to the end point by interpolating the axes.

Linear trajectory

To calculate the linear trajectory between two points, the origin and destination frames are considered, as shown in Figure 3.49, where frame A is the origin target and frame B is the destination target, both with defined coordinates and orientations defined according to Equation 3.36.

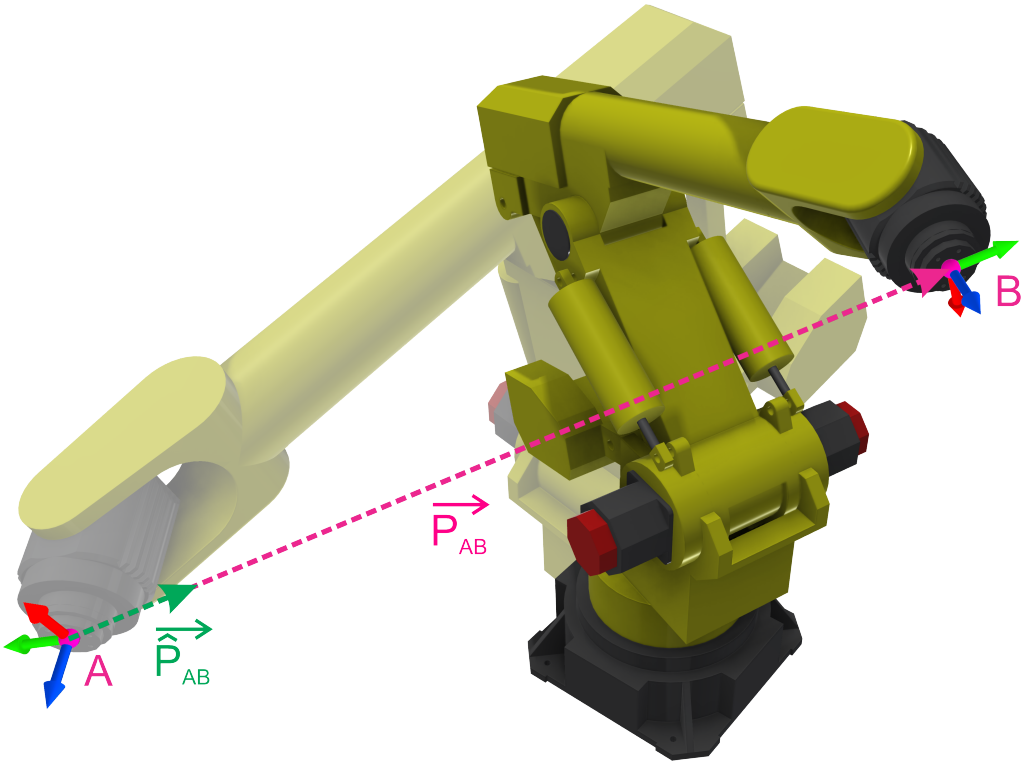


Figure 3.49: Linear trajectory representation.

$$\overrightarrow{P_A} = \begin{bmatrix} x_A \\ y_A \\ z_A \\ R_{x_A} \\ R_{y_A} \\ R_{z_A} \end{bmatrix}, \qquad \overrightarrow{P_B} = \begin{bmatrix} x_B \\ y_B \\ z_B \\ R_{x_B} \\ R_{y_B} \\ R_{z_B} \end{bmatrix} \tag{3.36}$$

The vector $\overrightarrow{P_{AB}}$ represents the vector between frame A and B, considering their respective coordinates and orientations, and is defined by subtracting $\overrightarrow{P_B} - \overrightarrow{P_A}$, according to Equation 3.37.

$$\overrightarrow{P_{AB}} = \overrightarrow{P_B} - \overrightarrow{P_A} = \begin{bmatrix} x_B - x_A \\ y_B - y_A \\ z_B - z_A \\ R_{x_B} - R_{x_A} \\ R_{y_B} - R_{y_A} \\ R_{z_B} - R_{z_A} \end{bmatrix} \tag{3.37}$$

The $\overrightarrow{\hat{P}_{AB}}$ vector represents a unit vector with the same direction as the $\overrightarrow{P_{AB}}$ vector and can be calculated according to Equation 3.38, where the modulus of the $\overrightarrow{P_{AB}}$ vector can be calculated according to Equation 3.39.

$$\overrightarrow{\hat{P}_{AB}} = \frac{\overrightarrow{P_{AB}}}{|\overrightarrow{P_{AB}}|} \tag{3.38}$$

$$|\overrightarrow{P_{AB}}| = \sqrt{x_{AB}{}^2 + y_{AB}{}^2 + z_{AB}{}^2 + R_{x_{AB}}{}^2 + R_{y_{AB}}{}^2 + R_{z_{AB}}{}^2} \tag{3.39}$$

The Equation 3.40 shows the calculation of the linear trajectory from frame A to frame B, where the coordinates of frame A are added to the coordinates of the unit vector $\overrightarrow{\hat{P}_{AB}}$ multiplied by the constant K. Varying the constant K from 0 to the modulus of the vector $\overrightarrow{P_{AB}}$ (Equation 3.39), the result of Equation 3.40 are the intermediate points between frame A and B, forming a linear trajectory.

$$\begin{cases} X = x_A + K.x_{\hat{P}_{AB}} \\ Y = y_A + K.y_{\hat{P}_{AB}} \\ Z = z_A + K.z_{\hat{P}_{AB}} \\ R_x = R_{x_A} + K.R_{x_{\hat{P}_{AB}}} \\ R_y = R_{y_A} + K.R_{y_{\hat{P}_{AB}}} \\ R_z = R_{z_A} + K.R_{z_{\hat{P}_{AB}}} \end{cases} \tag{3.40}$$

Considering the constant K as the modulus of the vector formed between the start and end points of the movement, this value has a direct physical relationship with the movement because it is represented as a unit of measurement of distance. Therefore, for the robot's trajectory to follow a motion shape with Jerk, it is only necessary to consider the constant K as the representation of the position s in the equations presented in Chapter 2, Table 2.2.

3.6.5 Tool configuration

The tool configuration refers to the tools connected to the flange of the robot's end effector, which can have linear and rotational offsets in relation to the standard end effector. In this way, the tool configuration is defined by a homogeneous transformation matrix representing the linear and rotational offsets of the new end effector in relation to the robot flange.

The kinematics calculations described in Sections 3.6.2 and 3.6.3 are designed to calculate the angles of the joints and the robot's flange point in relation to the base. Therefore, the methodology adopted was to calculate the point that the robot flange must be, in order for the tool attached to the robot to be in the desired position.

Equation 3.41 shows the approached homogeneous transformation defined by the matrix representing the point to be reached by the tool (T_{BT}) multiplied by the homogeneous transformation matrix of the flange in relation to the tool attached (T_{TF}).

$$T_{BF} = T_{BT}.T_{TF} \tag{3.41}$$

Considering that the tool is represented by the homogeneous transformation matrix that defines the measurements of the tool in relation to the flange (T_{FT}), the relationship required for the calculation must be obtained by applying the inverse of this matrix, where $T_{TF} = (T_{FT})^{-1}$. Therefore, the appropriate calculation is shown in Equation 3.42.

$$T_{BF} = T_{BT}.(T_{FT})^{-1} \tag{3.42}$$

All the calculations mentioned in this section have been implemented in C++ and embedded in an ESP32-S3. The codes related to the Trajectory Control calculation can be found in Appendix A.3.

3.7 Graphical User Interface (GUI)

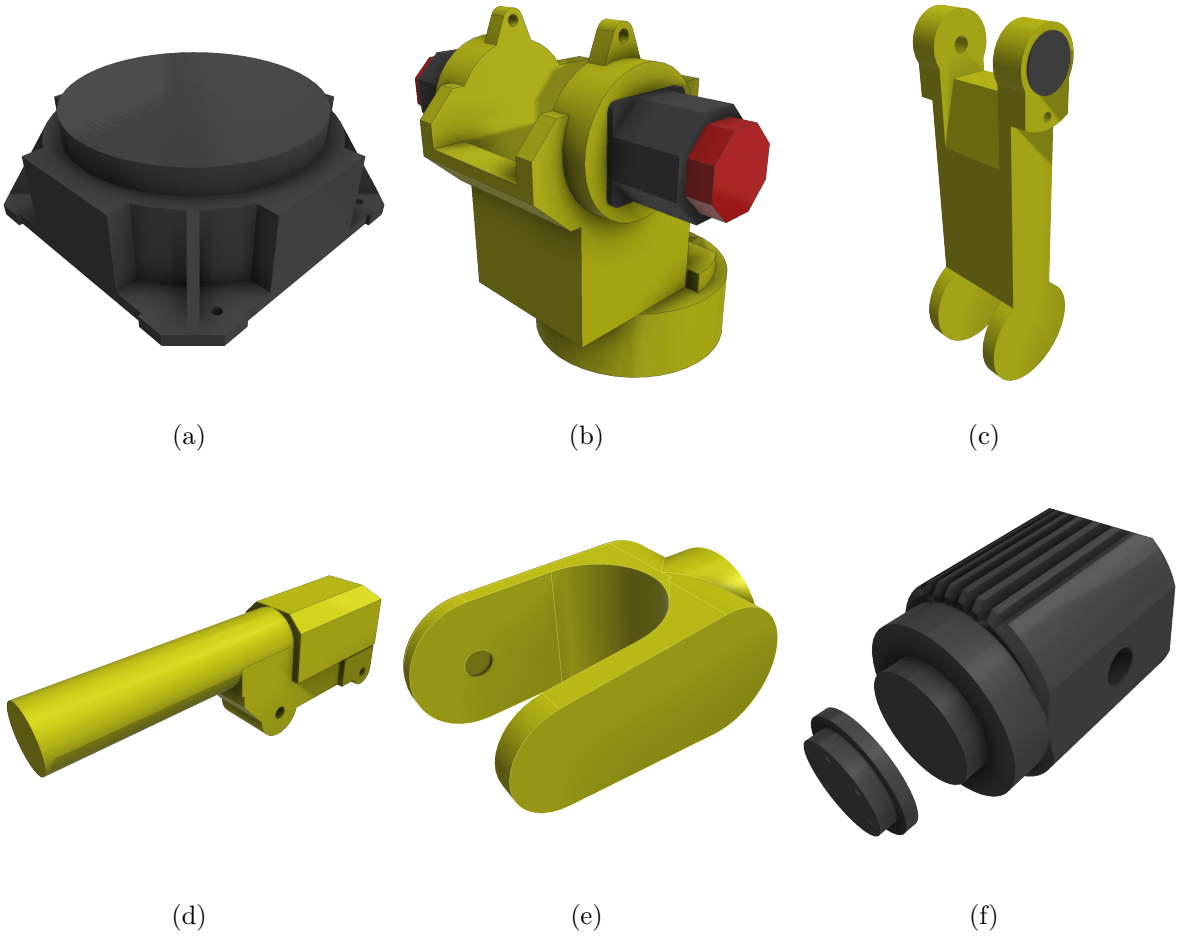
A Graphical User Interface was developed to send and receive commands from the Main CPU in order to program and send commands to the robot. It was developed mainly using the Python programming language [47], which means that the GUI can be run on most computer systems, requiring only support for this programming language.

The main function envisaged for the GUI is to display robot information, such as the position of the end effector and angle of each joint, to make it possible to program the robot and to display a 3D model of the FANUC S-420FD in order to improve visualization of the trajectory performed by the real robot.

This Section will describe the features implemented in the GUI and its development.

3.7.1 FANUC S-420FD 3D model

To improve visual understanding of the robot's operational tasks, a 3D model of the FANUC S-420FD was developed. For this, the robot's individual components were developed using Autodesk Inventor software [48] and to maintain accuracy in relation to the real robot, all the relevant measurements were taken from the robot's manuals [42]. Figure 3.50 shows the main parts of the robot in 3D modeling.

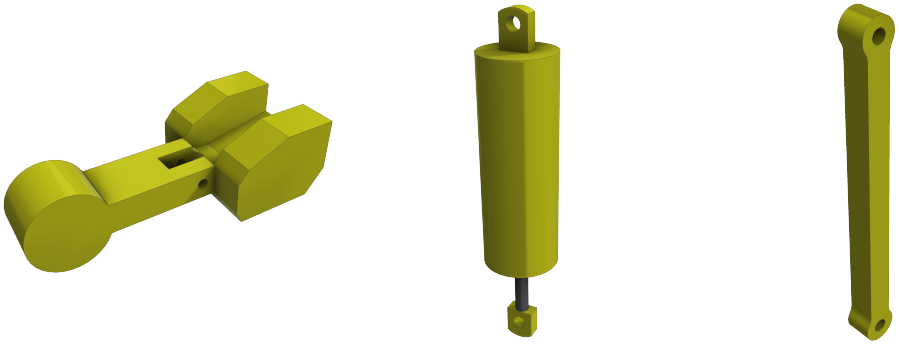


(a) (b) (c)

(d) (e) (f)

Figure 3.50: 3D model of the FANUC S-420FD's main parts.
(a) Base of robot. **(b)** Link between J1 and J2. **(c)** Link between J2 and J3. **(d)** Link between J3 and J4. **(e)** Link between J4 and J5. **(f)** J5 and J6.

Extra parts of the robot were also designed, which in the real robot perform important functions, but in the 3D model are just aesthetics, such as counterweights and auxiliary links. These parts are shown in Figure 3.51.



(a) (b) (c)

Figure 3.51: 3D model of the FANUC S-420FD's auxiliary parts.
(a) Counterweight. **(b)** Spring counterweight. **(c)** Auxiliary J3 link.

The assembly of the parts in the simulated environment will be described in later sections, but the virtual model with the parts positioned in Autodesk Inventor [48] is shown in Figure 3.52.

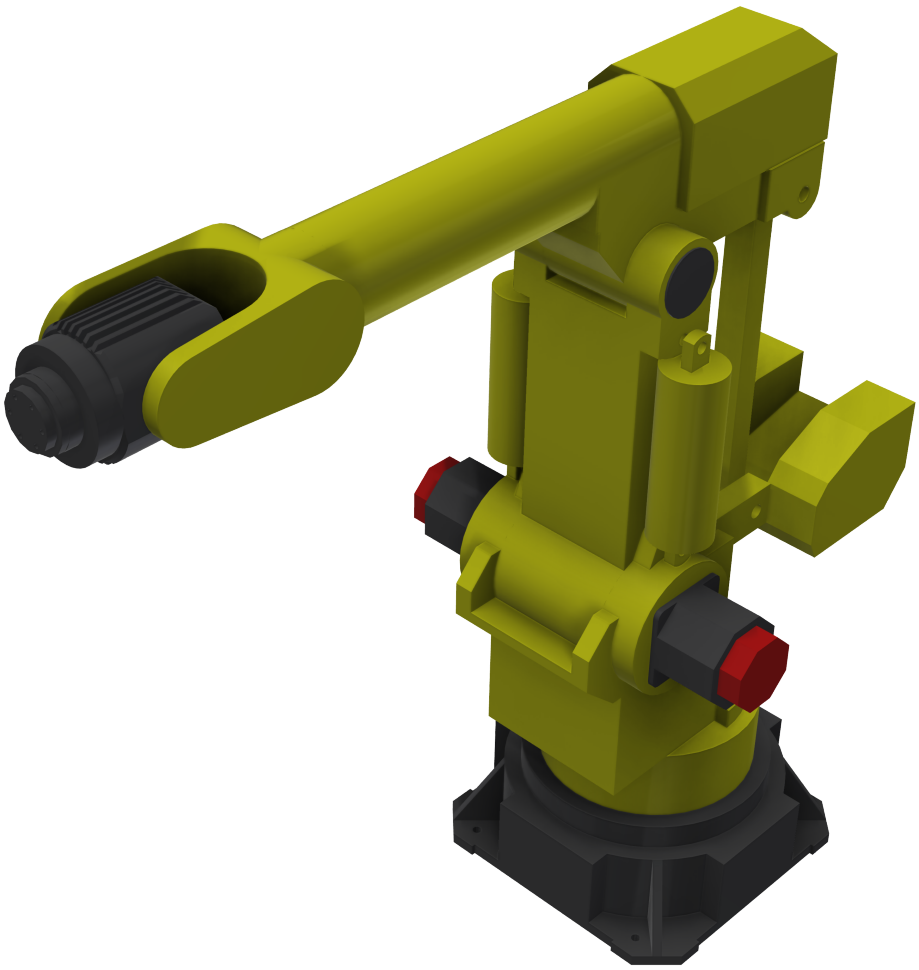


Figure 3.52: Assembly with all the robot parts.

3.7.2 Graphical User Interface description

The Graphical User Interface was constructed using Python programming language [47], QT Designer [49], and the Python OpenGL library [50] and is displayed in Figure 3.53. The models shown in Figure 3.50 were added to the 3D environment, each placed according to the robot's geometry. Furthermore, functionalities enabling joint-to-joint movements of the robot were incorporated to assist the interface operation.

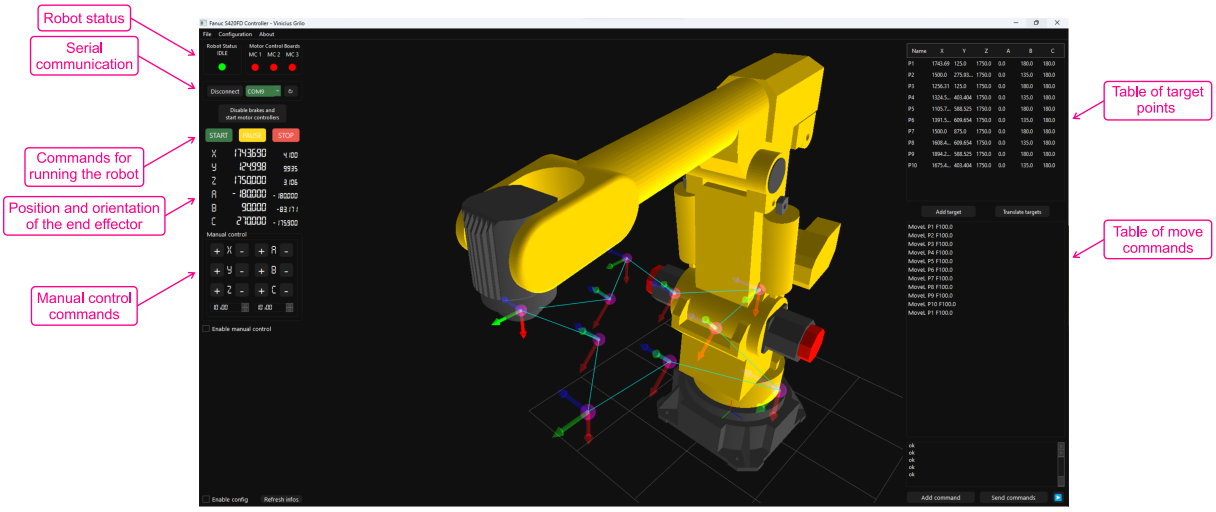


Figure 3.53: Graphical User Interface developed.

The central part of the GUI presents the 3D environment, exhibiting the robot 3D model, its executed path, and the defined target points.

On the left side, functionalities were integrated to establish a connection to the robot controller via serial communication. Additionally, buttons to initiate or stop operations, visualization tools indicating the end effector's position and orientation, joint angle displays, and controls for manual robot movement.

On the right, a table of target points and a table of move commands have been added, as well as a button for adding new target points, a button for adding new move commands, and a button for effectively sending the commands to the robot. As points are added to the table, they are displayed in the 3D environment with frame representations [1], indicating the position and orientation of each target point.

The programming language for movement commands is simplified and is intended for improvement in future versions. It's summarized as *"MoveL"* for linear movements and *"MoveJ"* for joint movements, followed by the target point and the maximum linear speed in mm/s.

G-Code support

In order to improve the compatibility with Computer-Aided Design and Computer-Aided Manufacturing (CAD/CAM) software, a G-Code file import function has been implemented. In this way it is possible to configure the motion parameters in specialized design and manufacturing software, export them in G-Code format, import the document into the developed GUI and then execute the motion in the robot controller.

This functionality can be accessed via the top menu *File* → *Open G-Code file*, where a window appears for selecting the desired file. The algorithm reads all the lines from the file and appends the movement points to the table of target points displayed on the right of the interface, in addition to incorporating movement commands into the table of move commands. The configured target points and the movement commands are shown in the 3D environment in the center of the GUI.

It is important to note that this feature does not offer complete support for all G-Code commands, requiring manual adjustment of the files exported by the CAD/CAM software in most cases.

The file must be in plain text and only the *G0* and *G1* commands are supported, representing the *MoveJ* and *MoveL* commands respectively. The algorithm expects seven parameters per command, indicating the robot's six DoF represented by the letters *X, Y, Z, A, B* and *C* followed by the value in millimeters, and the maximum movement speed represented by the letter *F* followed by the value in millimeters per second. The following example of G-Code is designed to result in a square-shaped movement that is analogous to the one depicted in Figure 3.55.

```
1 G0 X1200 Y500 Z1500 A0 B90 C180 F200
2 G1 X1200 Y1000 Z1500 A0 B180 C180 F200
3 G1 X1700 Y1000 Z1500 A0 B90 C180 F200
4 G1 X1700 Y500 Z1500 A0 B135 C180 F200
5 G1 X1200 Y500 Z1500 A0 B90 C180 F200
```

Tool configuration

The tool attached to the robot can be configured via the window shown in Figure 3.54. This window can be accessed via the *"Configuration"* menu at the top of the interface and allows the configuration of the tool's measurements in relation to the robot's flange, select and configure the 3D model of the tool to be added to the main window, add a name to the configured tool and select previously saved tools.

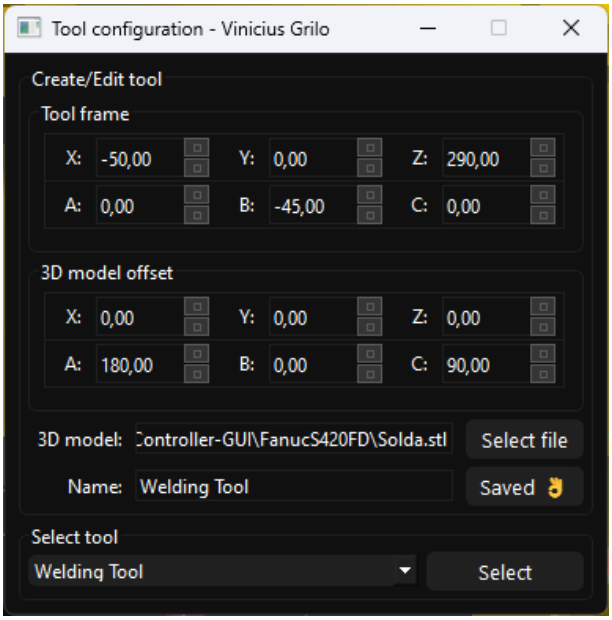


Figure 3.54: Tool configuration window.

After the user selects the desired tool, the selected measurements are sent to the Main CPU in order for the robot's position calculations to take the new tool into consideration. Figure 3.55 shows an example of a trajectory considering a tool attached to the robot.

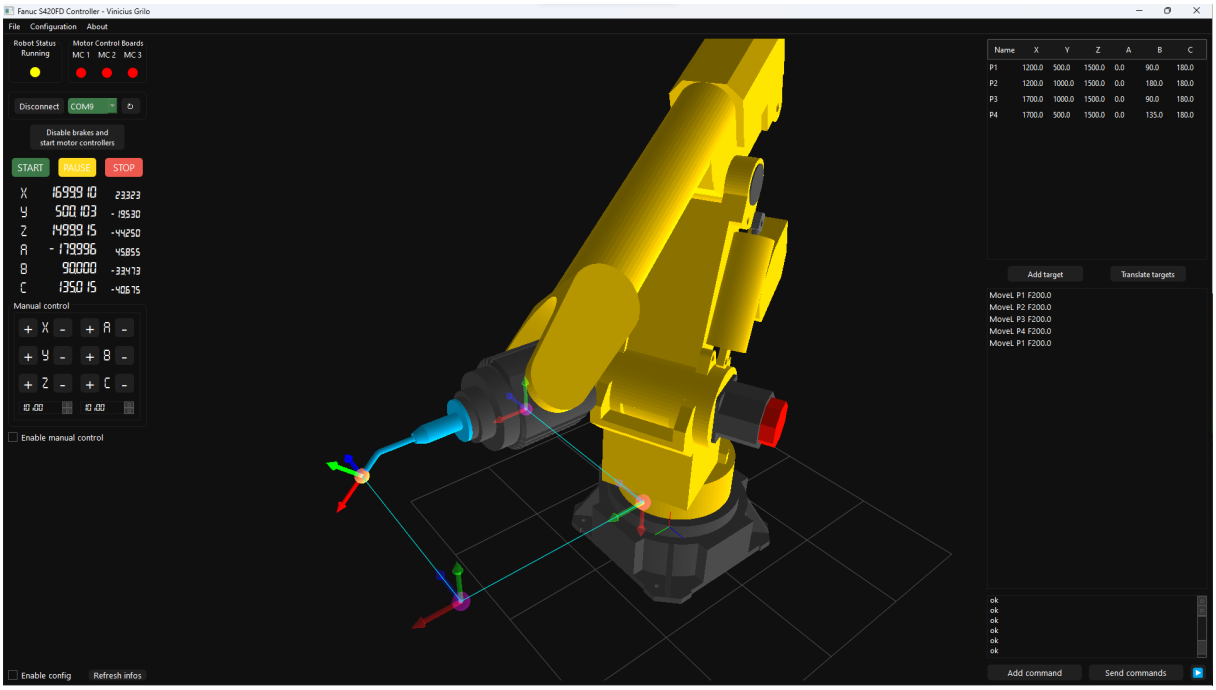


Figure 3.55: Performed trajectory considering the tool attached to the robot.

Chapter 4

Results

This Chapter discusses the results obtained from the developed circuits and algorithms presented in Chapter 3, including the methodology adopted and the equipment used.

4.1 Tests platform

In order to collect results and improve the architecture proposed (Section 3.2), the development of the Main CPU algorithm (Section 3.3), the kinematics and trajectory (Section 3.6), the motor control system and the operation of the GUI (Section 3.7), a test platform was developed containing a voltage supply, a PMSM three-phase motor, a test inverter, an STM32 NUCLEO board and an ESP32 microcontroller. Figure 4.1 shows the components mentioned in addition to the mounting base, the voltage indicator of the power supply and the encoder connection.

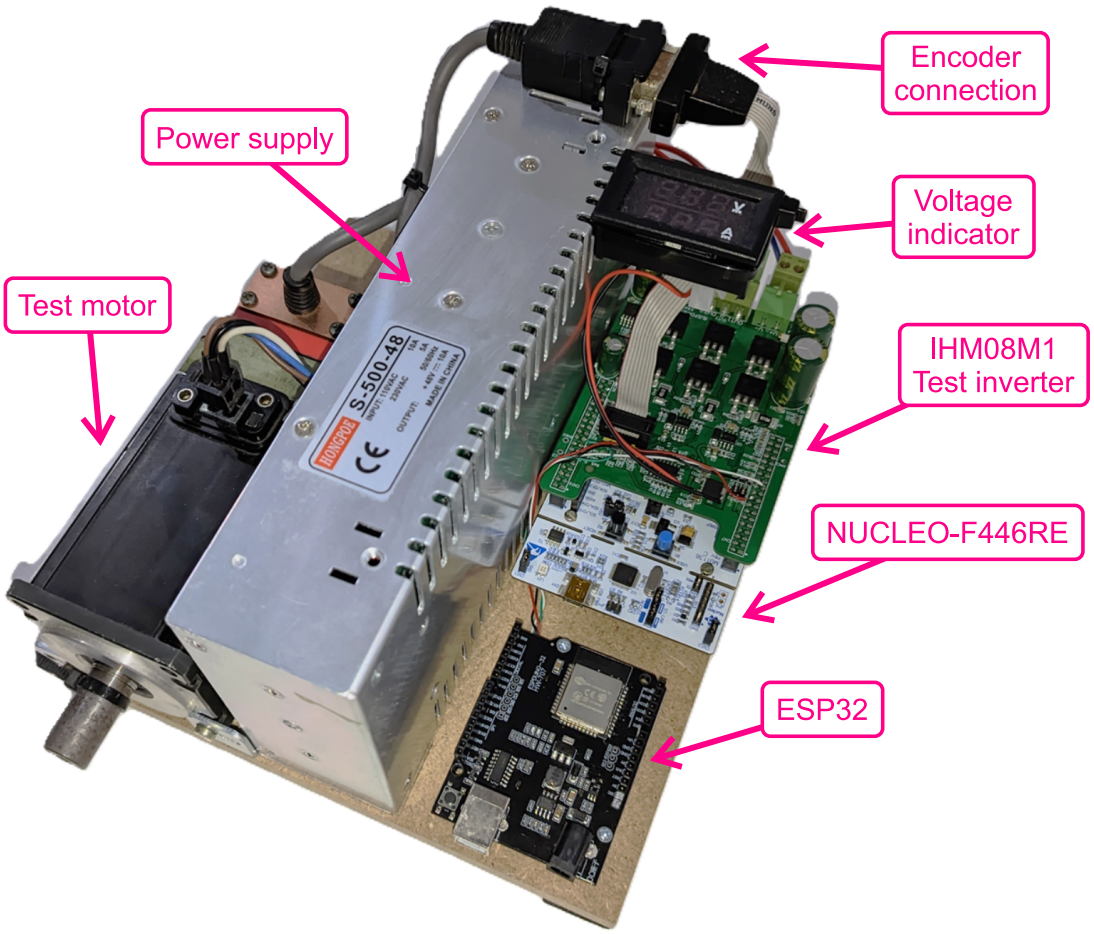


Figure 4.1: Test platform.

4.1.1 Test motor

The motor used to perform the initial system tests was the FANUC $\beta\iota S$ 1/6000 motor (part number A06B-0116-B103) shown in Figure 4.2. This motor was obtained with a defect in its original encoder, making it necessary to adapt a new encoder and for this was used a Nemicon model OVW2-25-2MD encoder with resolution of 2500 pulses per revolution and ABZ and their complements $\bar{A}\bar{B}\bar{Z}$ line drive outputs.

The motor used is a three-phase PMSM motor, very similar to the original robot motors described in Section 3.1.2. The characteristics of this motor are shown in Table 4.1 and it can be seen that the main differences when compared to the robot's original motors are in the stall torque, stall current, maximum speed and operating voltage. Despite the

differences, testing using this motor validates the use of the system with the robot's real motors, since the differences mentioned are compensated for by adjustments to the controllers.

Figure 4.2: FANUC $\beta\iota S$ 1/6000 test motor.

Model	FANUC $\beta\iota S$ 1/6000
Part number	A06B-0116-B103
Stall torque	1.2 Nm
Poles	8
RPM	6000
Stall current	2.6 A
Voltage	172 V

Table 4.1: FANUC $\beta\iota S$ 1/6000 characteristics.

4.1.2 Test inverter

The FANUC Servo Amplifier present in the original FANUC S-420FD controller receives six PWM signals and provides two current signals per channel, as well as other control signals, as described in Section 3.1.3.1. In order for the test platform to be as close as possible to the real conditions provided by the FANUC Servo Amplifier, an IHM08M1 inverter was used (Figure 4.3), which also receives six PWM signals to switch the MOSFETs present in the inverter bridge and provides the current signals for the motor's three phases.

The main difference between the FANUC Servo Amplifier and the test inverter used is in the current sensors. While the FANUC device has two current sensors to sense two

phases of the motor directly, the IHM08M1 has three shut-type current sensors connected between the source pin and GND of the lower MOSFETs of the three branches of the inverter bridge, which results in the current of the three phases of the motor indirectly.

Despite the differences in the current sensor specified, the model of the test inverter used validates the proposed solution, since both current sensor models are supported by the MCSDK, as specified in Section 2.1.4.

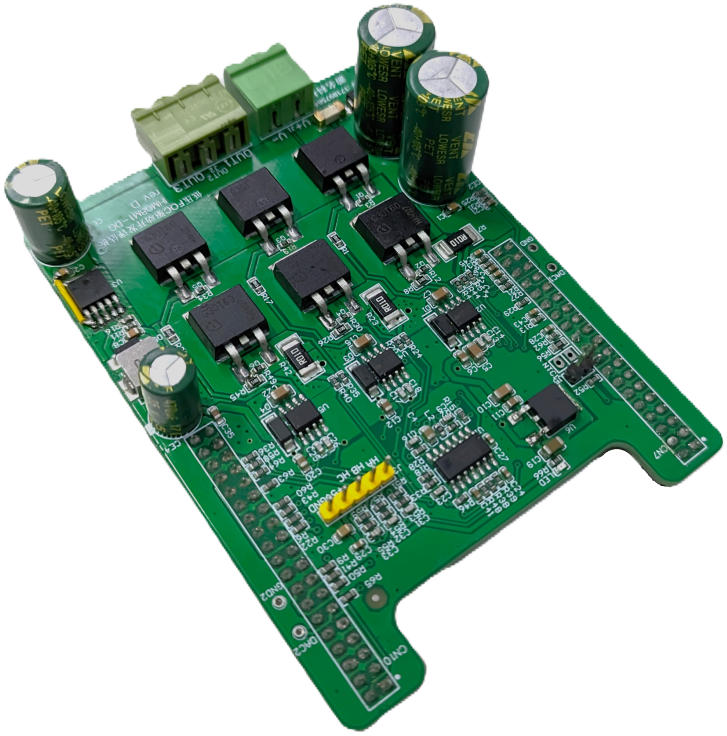


Figure 4.3: Test inverter model IHM08M1.

4.2 Dual Axis Control Board circuits

In accordance with the Printed Circuit Board design presented in Section 3.4, Figure 4.4 shows the PCB manufactured by a specialized company with components and circuits soldered on subsequently.

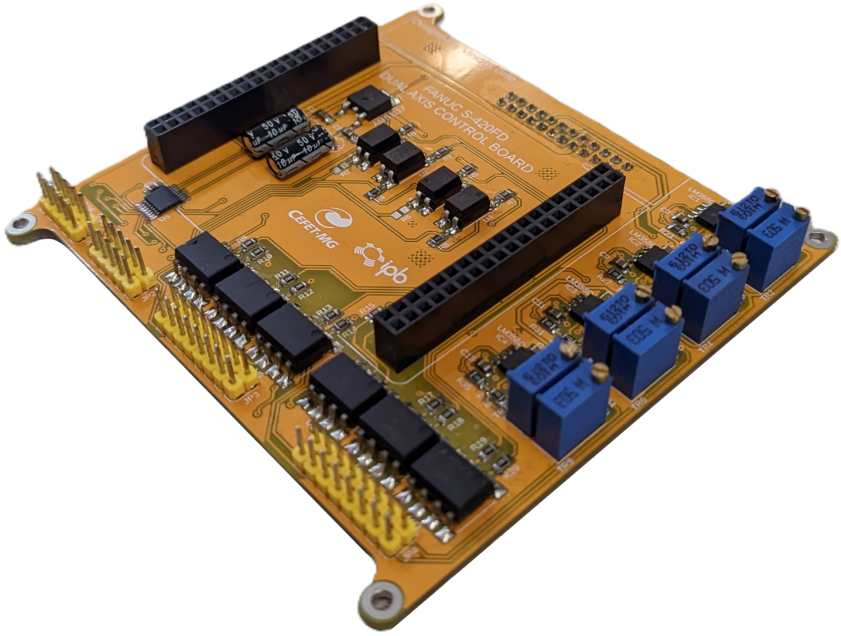
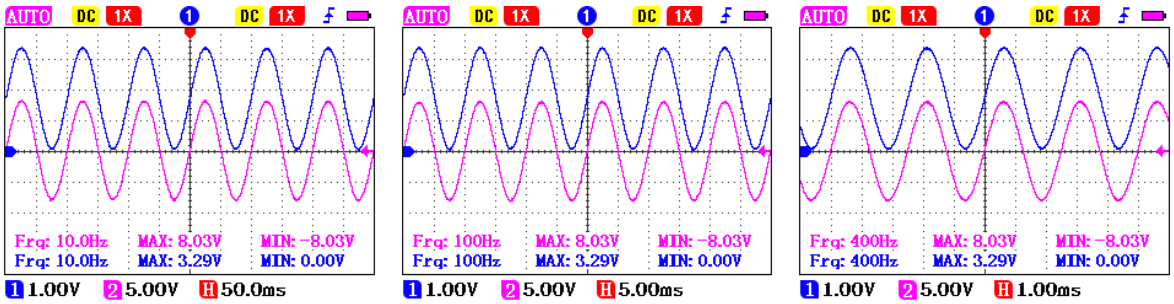


Figure 4.4: Dual Axis Control Board PCB manufactured.

Current sensing

The current signals coming from the FANUC Servo Amplifiers are limited to between $-8V$ and $+8V$ when they are supplying their respective maximum currents to the motors, as discussed in Section 3.1.3.2. The circuit presented in Figure 3.31 was developed to adapt these signal levels to be processed by the STM32F446RE microcontroller used in the Dual Axis Control Board.

Figure 4.5 shows the experimental results of the mentioned circuit, where sinusoidal signals with amplitude between $-8V$ and $+8V$ and frequencies of $10Hz$, $100Hz$ and $400Hz$ were applied (pink singal) and the signal routed to the microcontroller was measured, which, after adjustments to the circuit's calibration, remained between $0V$ and $3.3V$ (blue signal). The maximum frequency of $400Hz$ was determined by the plate information of the motors present in the robot, indicating this value as the standard operating frequency of the equipment.



(a) (b) (c)

Figure 4.5: Experimental tests of the current signal adjustment circuit. **(a)** $10Hz$. **(b)** $100Hz$. **(c)** $400Hz$.

Figure 4.6 shows an experimental test of the added protection circuit, where the signal level is limited to avoid damaging the microcontroller. In the case presented, the input signal remained at voltage levels between $-8V$ and $+8V$, but the circuit settings were purposely altered to present voltage levels outside the microcontroller's operating limits. The output signal remained saturated within the limits of $-784mV$ and $3.99V$, as expected.

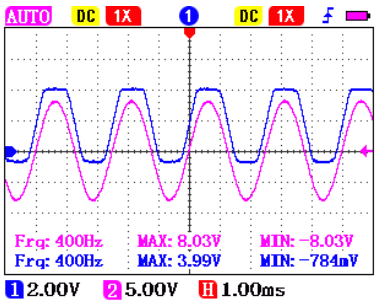


Figure 4.6: Experimental tests of the current signal protection circuit.

Optocouplers circuit

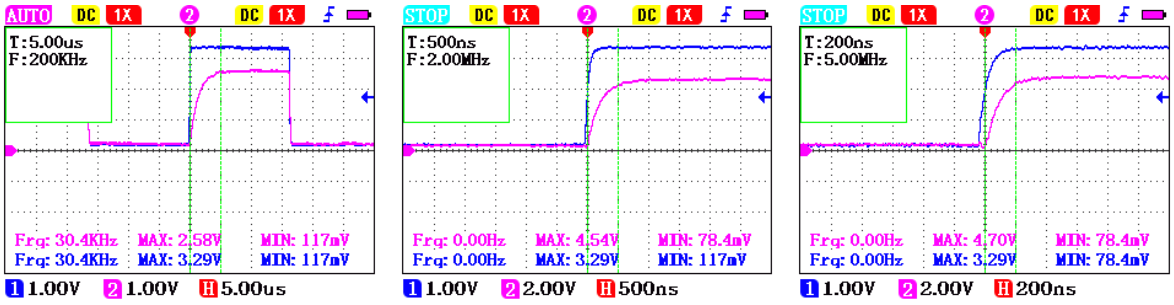
The PWM signal isolator circuits using model VO2630 optocouplers, discussed in more detail in Section 3.4, are critical parts of the system, since the output signals from the optocouplers are used to switch the motor phases via the FANUC Servo Amplifiers.

Considering that PWM signals are high-frequency signals, the rise time and fall time

of the optocouplers' output are fundamental characteristics to ensure that the original signal is faithfully reproduced at its output.

According to the datasheet of the optocoupler model used, the pull-up resistors connected to the outputs of the components directly influence the reaction times of the signal. The document recommends using resistors between 330Ω and $4k\Omega$, but tests were carried out with other resistor values.

Figure 4.7 shows the rise time for three pull-up resistor values ($10k\Omega$, $1k\Omega$ and 330Ω), where the rise times were $5\mu s$, $500ns$ and $200ns$, respectively, where the input signal is represented by the blue curve and the output signal by the pink curve. The result obtained with a pull resistor of 330Ω was more favorable, reproducing the original signal with greater fidelity.



(a) (b) (c)

Figure 4.7: Testing the rise time for different values of pull up resistor.
(a) $10k\Omega$ pull up resistor. **(b)** $1k\Omega$ pull up resistor. **(c)** 330Ω pull up resistor.

Figure 4.8 shows a $30kHz$ signal being directed to the input of one of the optoacolators (blue curve) and the reading of its respective output signal (pink curve) with a 330Ω pull up resistor. It can be seen that the voltage level is changed from $3.3V$ to $5V$, as expected, and the frequency and phase characteristics of the original signal are appropriately maintained.

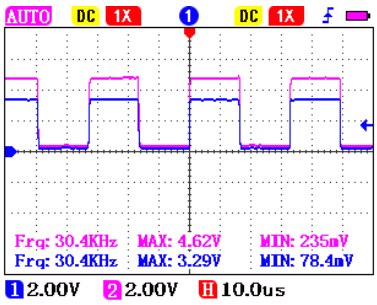


Figure 4.8: Input and output signal of one of the optocoupler circuits at a frequency of $30kHz$.

Tests with FANUC Servo Amplifier

To ensure that the signal characteristics of the circuits developed were consistent with the FANUC Servo Amplifier, the experiment illustrated in Figure 4.9 was performed. The figure shows an experimental version of the Dual Axis Control Board connected to the FANUC Servo Amplifier, assisted by auxiliary equipment such as power supplies and the supply voltage for the main contactor, provided by the Power Supply Unit and EMG Board under normal conditions.

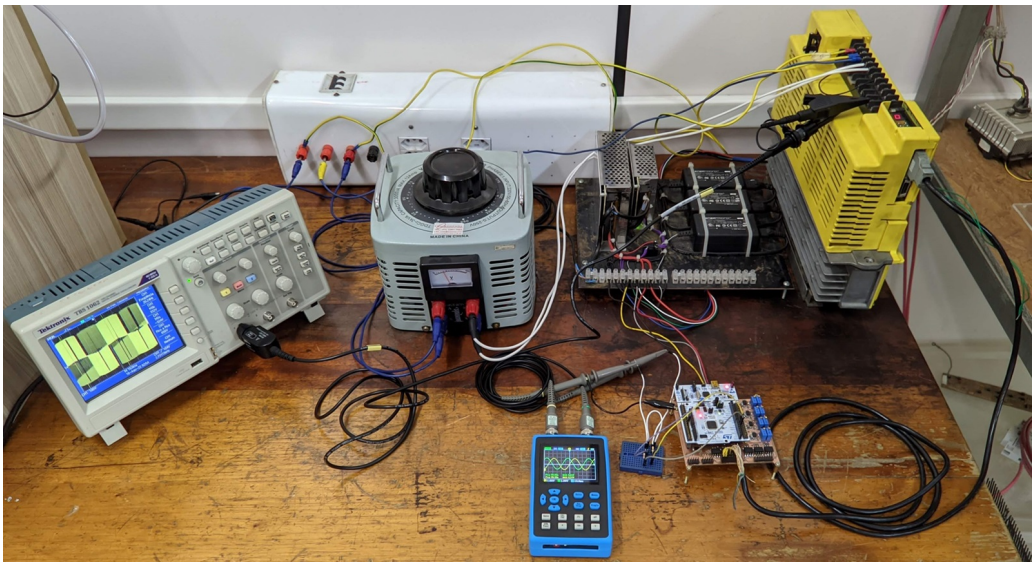


Figure 4.9: Experiment conducted with the FANUC Servo Amplifier.

The experiment consisted of applying a PWM modulated three-phase sinusoidal signal

with $20Hz$ to the inputs of the FANUC Servo Amplifier and reading the amplified output of the equipment. Figure 4.10 shows the reading of two of the three phases of the original signal applied to the FANUC Servo Amplifier.

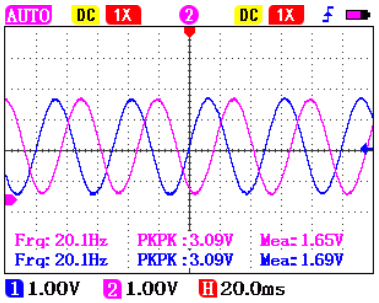
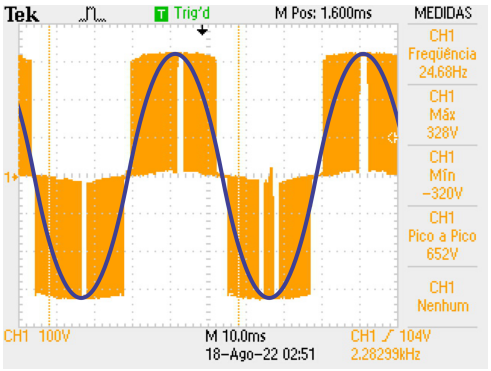
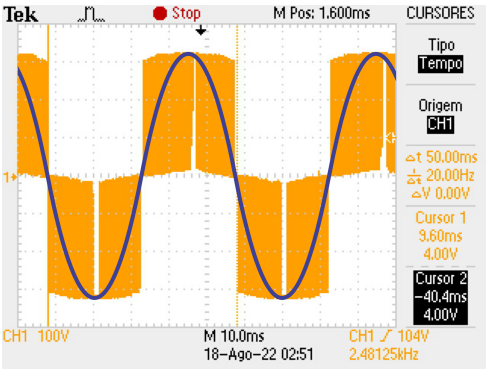


Figure 4.10: Sinusoidal signal used in the experiment with the FANUC Servo Amplifier.

The reading of the amplified output of the FANUC equipment is shown in Figure 4.11. Figure 4.11(a) shows the amplitude reading of the signal, with a peak-to-peak of $652V$, and Figure 4.11(b) shows the frequency of the signal. Although the signal has modulated characteristics, it is possible to see that the frequency of the original signal has been maintained, which can be seen by the signal in blue overlaid on both figures.



(a)



(b)

Figure 4.11: Reading of the amplified outputs during the experiment conducted with the FANUC Servo Amplifier.
(a) Reading of the signal amplitude. **(b)** Signal frequency reading.

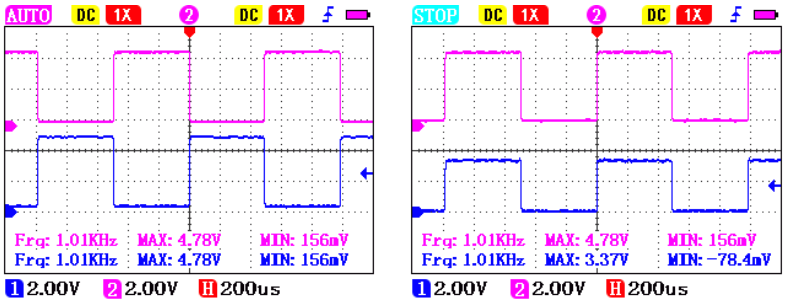
In addition to verifying that the outputs of the FANUC Servo Amplifier were manipulated satisfactorily, the experiment was also useful for verifying that the main contactor of

the equipment was activated via the MCON signal, detailed in Section 3.1.3.1, and that the FANUC equipment operated with the signals generated by the Dual Axis Control Board without internal errors, validating the entire development.

Encoders circuit

The main purpose of the circuit added to receive and process the encoder signals is to filter out possible unwanted signal noise, as discussed in Section 3.4. To do this, the MAX9122 signal driver was used, which has four channels capable of receiving two signals inverted from each other in order to process them and generate the filtered signals at its output.

Figure 4.12 shows the experimental tests of the encoder circuit, where Figure 4.12(a) shows a pair of complementary signals coming from the encoder and Figure 4.12(b) shows one of the original signals (pink curve) and its respective output (blue curve) with the same frequency and phase characteristics, but with a reduced signal, as expected.



(a) (b)

Figure 4.12: Test of the circuit developed to receive the encoder signals.
(a) Original complementary signals from encoder.
(b) Comparison between the original signal and the circuit's output signal.

4.3 Main Board circuits

In accordance with the Printed Circuit Board design presented in Section 3.5, Figure 4.13 shows the PCB manufactured.

Figure 4.13: Main Board PCB.

Figure 4.14 and 4.15 show the Main Board PCB with the three Dual Axis Control Boards attached and also with a NUCLEO-F446RE board attached to the second slot.

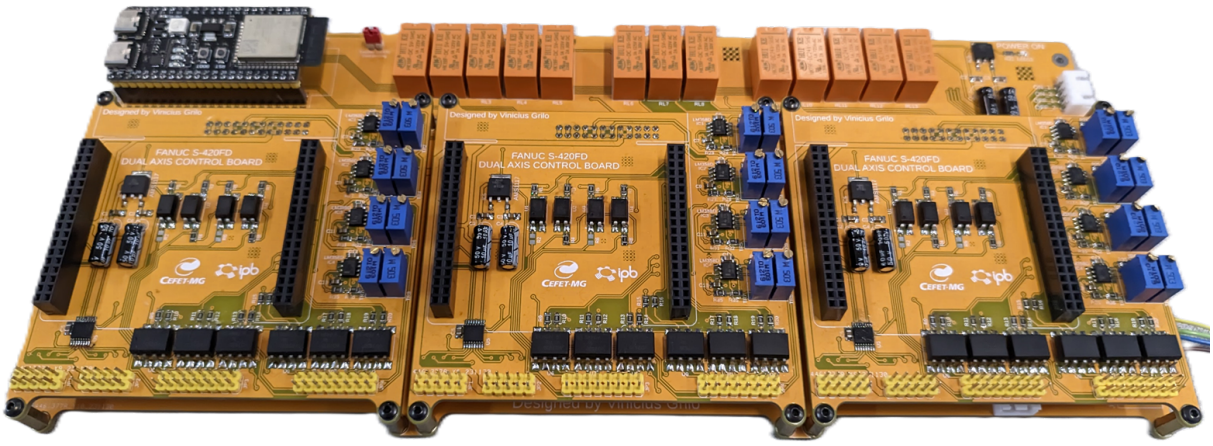


Figure 4.14: Main Board PCB with three Dual Axis Control Boards attached.

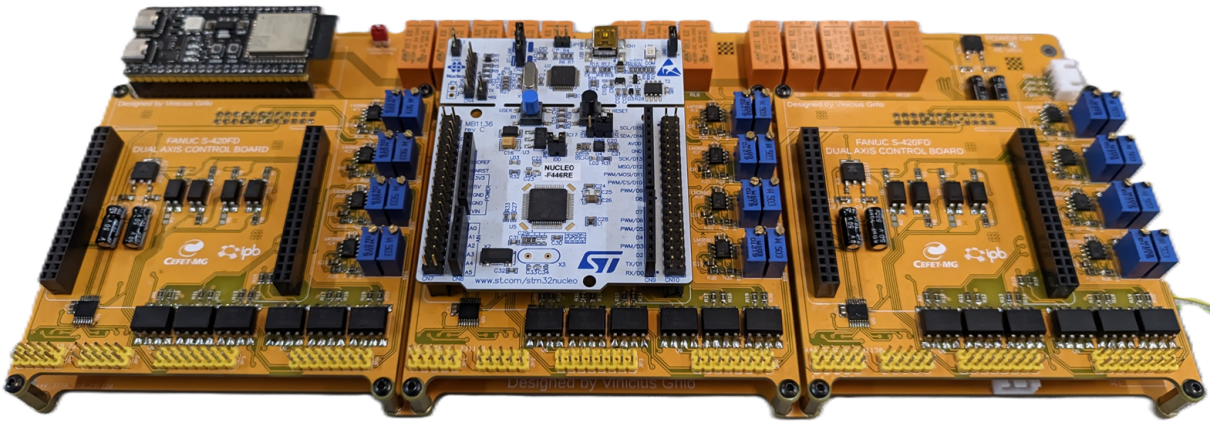


Figure 4.15: Main Board PCB with three Dual Axis Control Boards attached with a NUCLEO-F446RE attached to the second slot.

4.4 Main Board and Dual Axis Control Board communication

With the base software of the Dual Axis Control Board being the MCSDK running on a NUCLEO-F446RE board operating the position control of the motors in *Follow Mode*, as detailed in Sections 2.1.4, 3.3.2 and 3.4, it is necessary that the ESP32-S3 sends the position commands to the Dual Axis Control Boards via the Main Board at a fixed frequency, in order for the position controllers to operate as expected.

The period for sending position commands must take into consideration the speed of serial communication between the Main CPU and the Dual Axis Control Board, the processing capacity of the ESP32-S3 (Main CPU), the processing capacity of the STM32F446RE used in the Dual Axis Control Board and the period of the position control loop defined in the MCSDK Workbench.

The default setting for the control loop period in MCSDK Workbench is $0.5ms$. Therefore, the period for sending position commands was defined as $5ms$, which means that the position control loop operates ten times for each command received.

One of the tests performed was to verify if the STM32F446RE was receiving commands at the correct frequency. To check this, the state of one of the microcontroller's GPIOs

was inverted for each command received and was read using an oscilloscope. The result is shown in Figure 4.16 where it is confirmed that the position command was received every $5ms$, as expected.

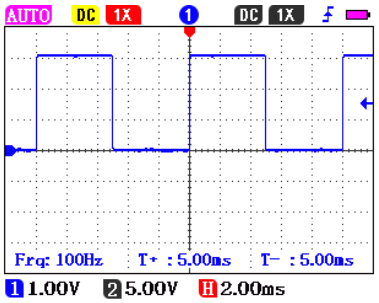


Figure 4.16: Experimental test of the position command sending period.

4.5 Forward and inverse kinematics

To test the performance and accuracy of the Forward Kinematics and Inverse Kinematics implemented, the tests described in this Section were performed.

Accuracy test

The first test performed was to evaluate the accuracy of the algorithms, where the angles of all the joints were varied from 0 to 90 degrees at a step of 0.1 degree and the Forward Kinematics of all the values were calculated, resulting in 900 operations. All the results of the Forward Kinematics were used as input for calculating the Inverse Kinematics.

In this way, the accuracy of the calculation was obtained by comparing the initial angles, used to calculate the Forward Kinematics, and the angles resulting from the Inverse Kinematics. All 900 results obtained were consistent using two decimal places, giving the algorithm an accuracy of 100% in this situation.

Performance tests

The tests conducted to evaluate the time taken to perform the kinematics calculations were supported by an oscilloscope, using the on/off state of a GPIO on the ESP32-S3 to indicate when an action is being performed or not.

The first test in this context was carried out using the same strategy used in the accuracy tests, where the Forward Kinematics were calculated for joint angles from 0 to 90 degrees with a step of 0.1 degree and applying the resulting point to calculate the Inverse Kinematics. Figure 4.17 shows the reading of the resulting signal, where the high signal represents the FK calculation and the low signal represents the IK calculation and it can be concluded that the calculation time remains constant regardless of the values involved.

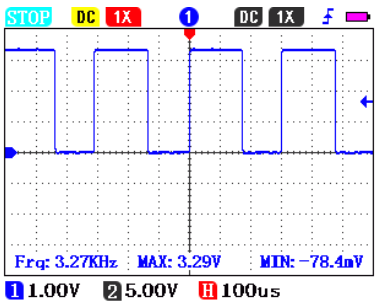
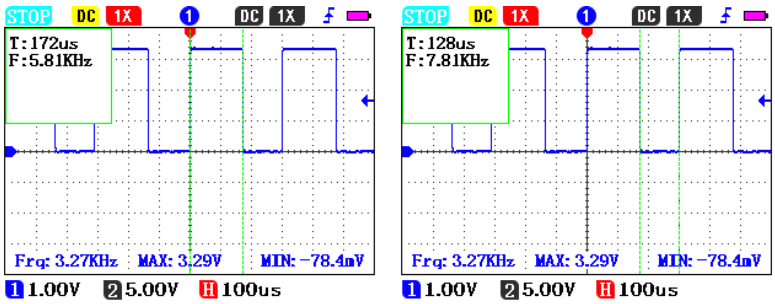


Figure 4.17: Inverse and Forward kinematics calculation time.

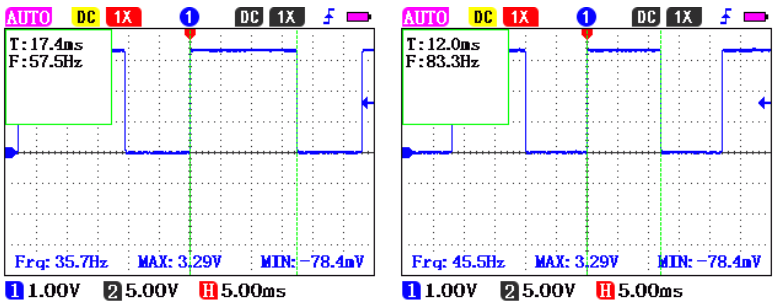
Figure 4.18 shows the calculation time for both kinematics, where a time of 172μs was obtained for the FK calculation and 128μs for the Inverse Kinematics calculation.



(a) (b)

Figure 4.18: Inverse and Forward kinematics calculation time.
(a) Forward Kinematics. **(b)** Inverse Kinematics.

In order to make the result more precise, the same test was carried out, but for the calculation of 100 values. Figure 4.19 shows the result, where the high signal represents the calculation of 100 kinematics followed by a low signal of $10ms$. Figure 4.19 shows the result of $17.4ms$ and $12.0ms$ for the calculation of 100 forward and reverse kinematics, respectively, resulting in an average of $174\mu s$ and $120\mu s$ for the single calculations of forward and reverse kinematics, respectively, which is similar to the result presented earlier.



(a) (b)

Figure 4.19: Multiple Inverse and Forward kinematics calculation time.
(a) Forward Kinematics. **(b)** Inverse Kinematics.

4.6 Trajectory control

The trajectory control implemented in the Main CPU includes calculating the intermediate points on a given trajectory, discussed in more detail in Section 3.6.4, and calculating the kinematics for each point on the specified trajectory.

The higher the calculation frequency, the greater the number of intermediate points calculated and, consequently, the smoother the trajectory generated. Therefore, considering the longer calculation time of the kinematics discussed in Section 4.5, the calculation period of the intermediate points of the trajectory control was set to $200\mu s$.

To check that the defined period was respected, one of the ESP32-S3's GPIOs was programmed to invert its state at each calculated point in the trajectory. The result is shown in Figure 4.20, confirming the calculation period of $200\mu s$.

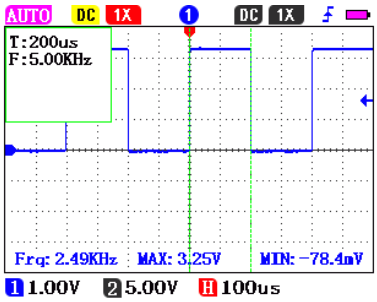


Figure 4.20: Experimental test of the trajectory control period.

To demonstrate the techniques and algorithms implemented detailed in Section 3.6.4, the linear trajectory shown in Figure 4.21 will be exemplified, where the movement runs along the Y axis from point $A = \{1200, -600, 1500, 0, 90, 180\}$ to point $B = \{1200, 600, 1500, 0, 90, 180\}$. In this way, the vector $\overrightarrow{P_{AB}}$ is defined as $P_B - P_A = \{0, 1200, 0, 0, 0, 0\}$, its modulus $|\overrightarrow{P_{AB}}| = 1200mm$ and the unit vector $\overrightarrow{\hat{P}_{AB}} = \{0, 1, 0, 0, 0, 0\}$.

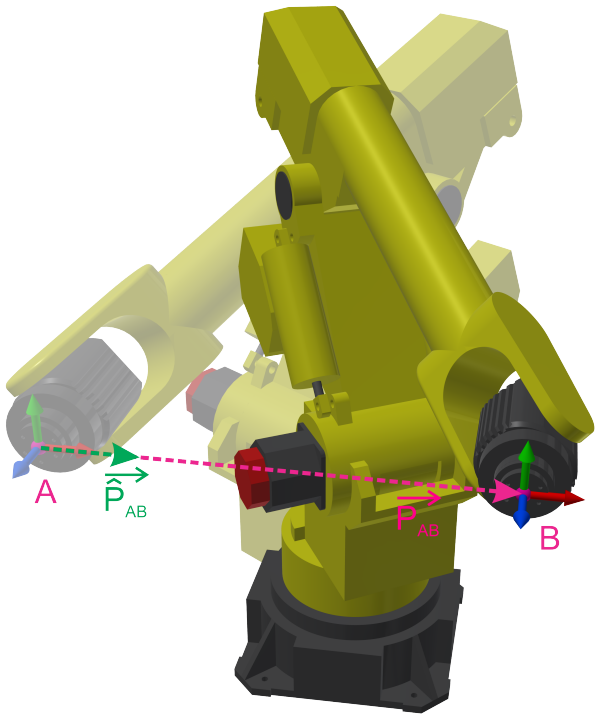


Figure 4.21: Linear trajectory between points A and B.

The linear trajectory between the two points respects Equation 3.40, where the constant K follows the Jerk profile, varying from 0 to the modulus of the $\overrightarrow{P_{AB}}$ vector (1200mm), to ensure that all movement follows the same profile. The trajectory was configured with the parameters $j_{max} = 1000mm/s^3$, $a_{max} = 500mm/s^s$ and $v_{max} = 400mm/s$. The Jerk profile mentioned can be visualised in Figure 4.22.

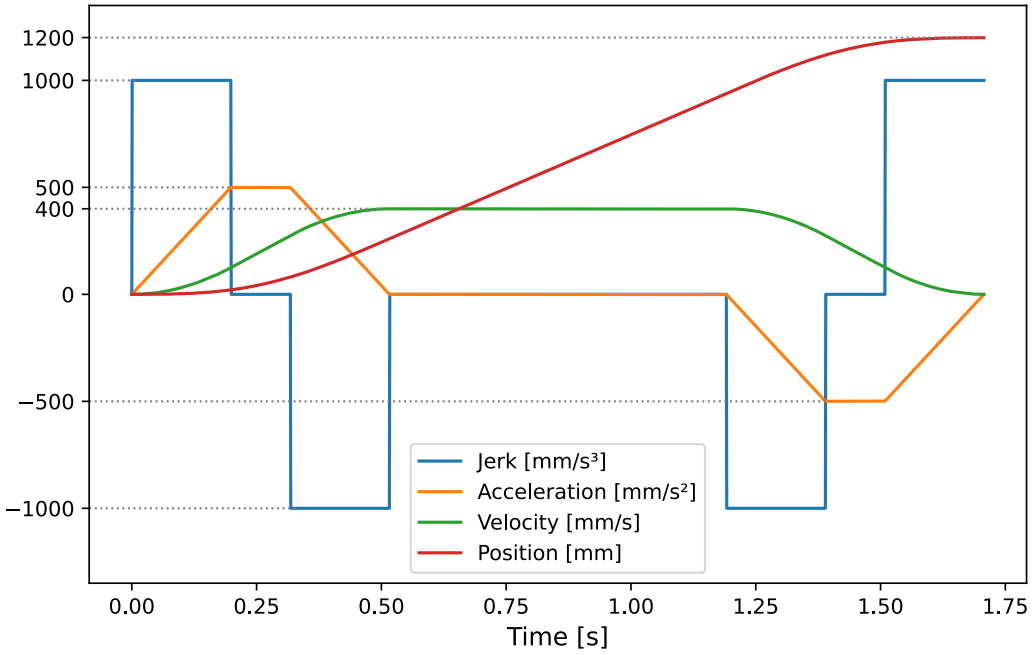


Figure 4.22: Jerk, acceleration, velocity and position along the trajectory.

The individual angular trajectories of each robot joint along the path shown in Figure 4.21 are shown in Figure 4.23.

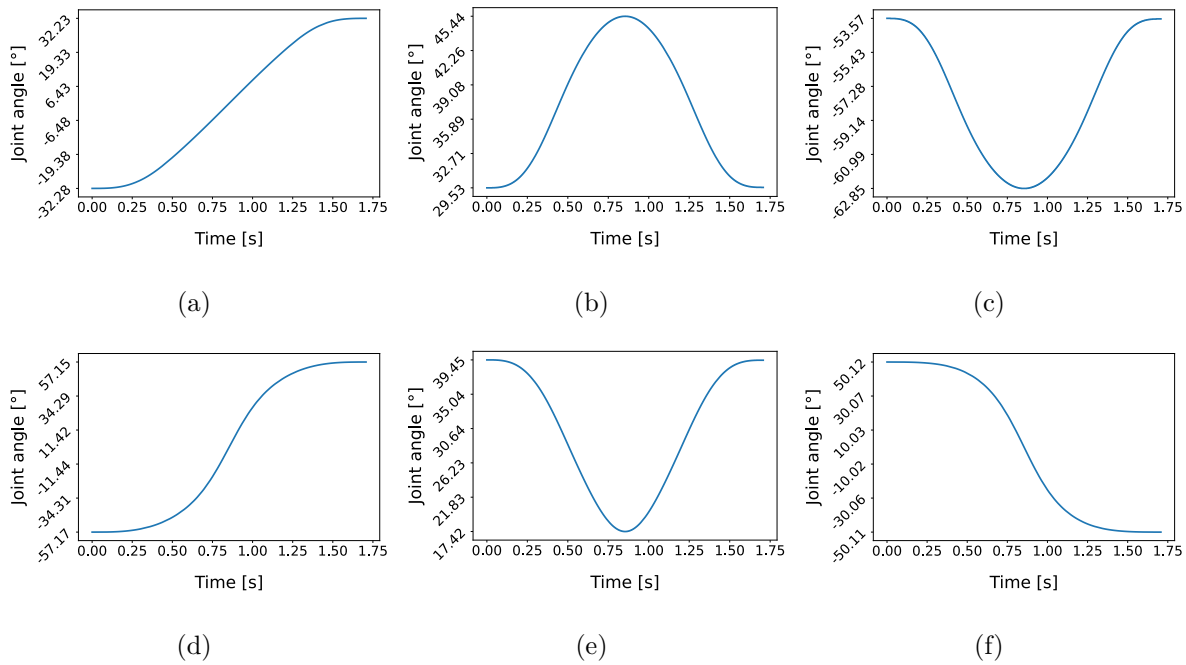


Figure 4.23: Joint angular trajectories. **(a)** J1. **(b)** J2. **(c)** J3. **(d)** J4. **(e)** J5. **(f)** J6.

All the curves and graphs shown were calculated point by point by the ESP32-S3 during trajectory control and saved to be rendered later.

4.7 Graphical User Interface (GUI)

One of the functions implemented in the GUI is the visualization of the robot's movement through the implemented 3D model. The GUI receives the current angles of each joint of the robot and reproduces them in the three-dimensional environment, so the position and orientation of the end effector displayed in the graphical interface is the real position of the frame representing the end effector within the 3D environment.

With this, the 3D environment implicitly realizes the direct kinematics of the robot by reproducing the angles received from the controller.

Figure 4.24 shows the comparison between the X, Y and Z coordinates calculated by the controller (blue curve) and those reproduced by the GUI (orange curve) along a trajectory three meters long.

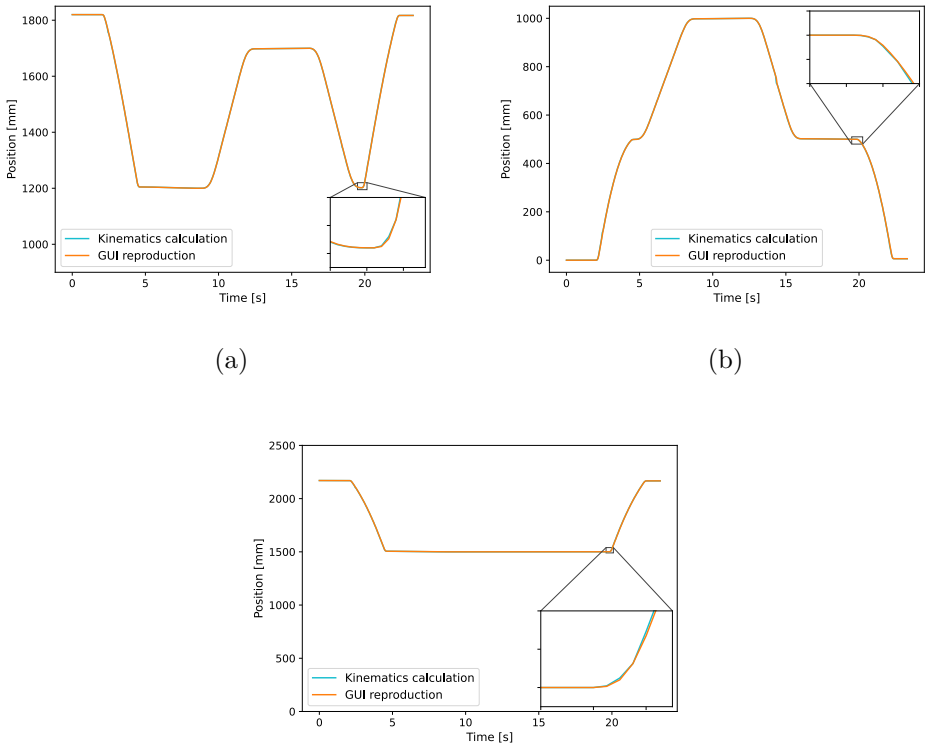


Figure 4.24: Comparison between the coordinates calculated by kinematics and the GUI reproduction. **(a)** X coordinate. **(b)** Y coordinate. **(c)** Z coordinate.

The trajectory used for the tests is shown in Figure 3.55 and it can be seen that the curves overlap almost the entire course, except for the highlighted parts. The Mean Squared Error (MSE) [51] calculated for the three coordinates was $0.894mm$ for the X coordinate, $0.764mm$ for the Y coordinate and $0.209mm$ for the Z coordinate, which is good for the purpose of visualizing the robot's trajectory.

The operation of the GUI combined with the trajectory and motor position controllers can be seen in the demonstration video at the link: `https://cloud.ipb.pt/f/aef3ce5c50304b86a976/`. In the video, the robot's six joints were emulated using the test platform and recorded separately. For demonstration purposes, simulated reduction ratios were also used for each joint.

Chapter 5

Conclusion and Future Work

This work was developed through a cooperation between the Federal Centre for Technological Education of Minas Gerais (CEFET-MG), located in Leopoldina - MG, Brazil, and the Polytechnic Institute of Bragança (IPB), located in Bragança, Portugal, through the Double Degree programme.

The general objective of this Master Thesis was to develop a new controller for the FANUC S-420FD robot inspired by the malfunctioning of the original controller, in which the methodologies used, the development procedures and the results achieved were described throughout this document.

The development of the specific objective of studying the main components of the original robot controller was described in Section 3.1 and was considered by the authors to be crucial for the rest of the development, since the new controller proposed in Section 3.2 reuses many parts of the original controller, as well as having a structure inspired by the FANUC controller.

The detailed analysis of the specific components of the FANUC RJ Controller played an important role in the development of the work, since most of the information acquired during development is not contained in any public documents or manuals.

The detailed analysis of the FANUC Servo Amplifier discussed in Sections 3.1.3.1 and 3.1.3.2 helped to understand the internal workings of the equipment and how it could be operated. The analyses of the Power Supply Unit and Emergency Board described in

Sections 3.1.3.3 and 3.1.3.4 also elucidated the internal workings of the controller and the actions and emergency signals required for the robot to function properly.

The modularization of the motor controllers by developing three Dual Axis Control Boards, described in Section 3.4, to communicate with each FANUC Servo Amplifier was consistent with the project, as it followed the original structure of the controller, where each pair of motors is controlled by a FANUC Servo Amplifier and, consequently, by a Dual Axis Control Board. The decision to use the MCSDK tool was also appropriate, as it offered all the necessary resources for controlling the PMSMs present in the mechanical part of the robot and saved time developing a new motor control solution.

Section 3.6 describes the development of the direct kinematics, inverse kinematics and trajectory control and, according to the results presented in Section 4.5, showed high performance for the proposed solution. The calculation time for each intermediate point along the trajectory was $200\mu S$, which means calculating 5000 points on a one-meter trajectory at a speed of $1m/S$, which was considered sufficient to keep the robot's trajectory consistent.

The circuits of the Main Board and Dual Axis Control Board developed and described in Sections 3.5 and 3.4, and their results presented in Sections 4.3 and 4.2, showed sufficient performance for the project proposal, where they ensured communication between the Main CPU and the motor controllers, as well as properly processing and conditioning the signals coming from the FANUC Servo Amplifiers, EMG Board and PSU.

The Graphical User Interface described in Section 3.7 was initially intended to allow interaction between the user and the controller developed, as well as to monitor the robot's movements. The 3D model of the FANUC S-420FD robot developed and discussed in Section 3.7.1 played a fundamental role in achieving this objective. The three-dimensional environment developed made it possible to verify the kinematics developed and visualize the trajectory control without the need to test it with the real robot, which increased the safety and agility of the development. The results in terms of accuracy in reproducing the movements presented in Section 4.7 were considered good, especially for the purpose of visualizing the trajectory performed by the robot.

One question that was not clarified during the development of this work was related to the robot's original encoders. The model of the encoders is shown in Section 3.1.2, and it was assumed that they would output $A\bar{A}B\bar{B}$ signals phased by 90 degrees, as is usual with incremental encoders, but it was later realized that communication between the encoders and the robot's original controller is done via RS422 serial communications using a proprietary protocol developed by FANUC. Possible solutions to this impasse are discussed in the Future Work section.

Although the general and specific objectives proposed for the development of this work have been achieved, it was not possible to perform tests with the real mechanical unit of the robot due to geographical limitations between the authors and the robot in question. Despite this, it can be concluded that the work achieved good results within what was initially proposed.

Future Work

This work explores a variety of different engineering and robotics themes, involving not only development projects but also practical and empirical work. In projects of this nature, it is inherent that details remain to be improved.

The most fundamental future suggestion is to mount the developed controller in the enclosure of the original controller, in conjunction with the robot's mechanical unit, performing all the necessary assembly and electrical connections.

The motor controllers must be tuned individually for each of the six motors in the robot. For the current controllers, it is sufficient to enter the DC bus voltage present in the FANUC Servo Amplifier, the inductance and resistance of the motors into the MCSDK Workbench, where the controller gains are adjusted according to the current curve caused by these characteristics. For the position controllers, some method of tuning the PID controllers available in the literature must be applied, assessing which one is better suited to the context of the controller.

As discussed in Section 4, the robot's original encoders are not compatible with the

solution developed in its standard configuration. To resolve this issue, the code embedded in the STM32F446RE can be adjusted to work with the encoder's original protocol or electrical modifications can be made to the encoder's connections to access the $A\bar{A}B\bar{B}$ signals.

The study of the robot's mechanical characteristics is also required for the proper functioning of the proposed controller. Although the Main CPU implementation already anticipates the necessity of this parameter, the relationship of the mechanical reductions that connect the motor shaft to the robot's moving parts needs to be studied.

The analysis and dynamic implementation of the robot, including the torque generated by the weight of the mechanical parts and tools attached to the robot and the sensing of external forces acting on it, can provide new functionalities to the robot discussed in modern robotics.

Bibliography

[1] J. J. Craig, *Introduction to robotics*. Pearson Educacion, 2006.

[2] D. Meike, M. Pellicciari, G. Berselli, A. Vergnano, and L. Ribickis, "Increasing the energy efficiency of multi-robot production lines in the automotive industry," in *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, 2012, pp. 700–705.

[3] D. Meike and L. Ribickis, "Energy efficient use of robotics in the automobile industry," in *2011 15th international conference on advanced robotics (ICAR)*, IEEE, 2011, pp. 507–511.

[4] M. Bartoš, V. Bulej, M. Bohušík, J. Stanček, V. Ivanov, and P. Macek, "An overview of robot applications in automotive industry," *Transportation Research Procedia*, vol. 55, pp. 837–844, 2021.

[5] J. Holland, L. Kingston, C. McCarthy, *et al.*, "Service robots in the healthcare sector," *Robotics*, vol. 10, no. 1, p. 47, 2021.

[6] R. Goel and P. Gupta, "Robotics and industry 4.0," *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*, pp. 157–169, 2020.

[7] N. Spolaôr and F. B. V. Benitti, "Robotics applications grounded in learning theories on tertiary education: A systematic review," *Computers & Education*, vol. 112, pp. 97–107, 2017.

[8] A. Eguchi, "Robotics as a learning tool for educational transformation," in *Proceedings of 4th international workshop teaching robotics, teaching with robotics & 5th international conference robotics in education*, vol. 18, 2014, pp. 27–34.

[9] M. Hägele, K. Nilsson, J. N. Pires, and R. Bischoff, "Industrial robotics," *Springer handbook of robotics*, pp. 1385–1422, 2016.

[10] G. Ferretti, G. Magnani, P. Putz, and P. Rocco, "The structured design of an industrial robot controller," *Control Engineering Practice*, vol. 4, no. 2, pp. 239–249, 1996.

[11] D. H. Arjoni, F. S. Madani, G. Ikeda, *et al.*, "Manufacture equipment retrofit to allow usage in the industry 4.0," in *2017 2nd international conference on Cybernetics, Robotics and Control (CRC)*, IEEE, 2017, pp. 155–161.

[12] J. Hu, C. Li, Z. Chen, and B. Yao, "Precision motion control of a 6-dofs industrial robot with accurate payload estimation," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1821–1829, 2020.

[13] T. Yuan, D. Wang, X. Wang, X. Wang, and Z. Sun, "High-precision servo control of industrial robot driven by pmsm-dtc utilizing composite active vectors," *IEEE Access*, vol. 7, pp. 7577–7587, 2019.

[14] V. M. Bida, D. V. Samokhvalov, and F. S. Al-Mahturi, "Pmsm vector control techniques — a survey," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018, pp. 577–581. DOI: 10.1109/EIConRus.2018.8317164.

[15] J. Vittek, B. Ftorek, *et al.*, "Energy efficient speed and position control of electric drives with pmsm," *Communications-Scientific Letters of the University of Zilina*, vol. 16, no. 1, pp. 64–71, 2014.

[16] J. Puranen *et al.*, "Induction motor versus permanent magnet synchronous motor in motion control applications: A comparative study," 2006.

[17] A. Loganayaki and R. B. Kumar, "Permanent magnet synchronous motor for electric vehicle applications," in *2019 5th international conference on advanced computing & communication systems (ICACCS)*, IEEE, 2019, pp. 1064–1069.

[18] S. Sakunthala, R. Kiranmayi, and P. N. Mandadi, "A study on industrial motor drives: Comparison and applications of pmsm and bldc motor drives," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, IEEE, 2017, pp. 537–540.

[19] A. Samar, P. Saedin, A. I. Tajudin, and N. Adni, "The implementation of field oriented control for pmsm drive based on tms320f2808 dsp controller," in *2012 IEEE international conference on control system, computing and engineering*, IEEE, 2012, pp. 612–616.

[20] K. Jash, P. K. Saha, and G. K. Panda, "Vector control of permanent magnet synchronous motor based on sinusoidal pulse width modulated inverter with proportional integral controller," *International Journal of Engineering Research and Applications*, vol. 3, no. 5, pp. 913–917, 2013.

[21] S. Chattopadhyay, M. Mitra, S. Sengupta, S. Chattopadhyay, M. Mitra, and S. Sengupta, "Clarke and park transform," *Electric Power Quality*, pp. 89–96, 2011.

[22] B. Adhavan, A. Kuppuswamy, G. Jayabaskaran, and V. Jagannathan, "Field oriented control of permanent magnet synchronous motor (pmsm) using fuzzy logic controller," in *2011 IEEE Recent Advances in Intelligent Computational Systems*, IEEE, 2011, pp. 587–592.

[23] STMicroelectronics, *Stm32 motor control software development kit (mcsdk)*, STMicroelectronics, Ed., Accessed: 15-12-2023. [Online]. Available: `https://www.st.com/en/embedded-software/x-cube-mcsdk.html`.

[24] STMicroelectronics, *Stmicroelectronics website*, STMicroelectronics, Ed., Accessed: 15-12-2023. [Online]. Available: `https://www.st.com/content/st_com/en.html`.

[25] STMicroelectronics, *Stm32cube ecosystem*, STMicroelectronics, Ed., Accessed: 15-12-2023. [Online]. Available: `https://www.st.com/content/st_com/en/ecosystems/stm32cube-ecosystem.html`.

[26] H. J. Yoon, S. Y. Chung, H. S. Kang, and M. J. Hwang, "Trapezoidal motion profile to suppress residual vibration of flexible object moved by robot," *Electronics*, vol. 8, no. 1, p. 30, 2019.

[27] H. Mu, Y. Zhou, S. Yan, and A. Han, "Third-order trajectory planning for high accuracy point-to-point motion," *Frontiers of Electrical and Electronic Engineering in China*, vol. 4, pp. 83–87, 2009.

[28] H. Mu and Y. Zhou, "Profile generation algorithm and implementation for high accuracy motion," in *2006 IEEE International Conference on Robotics and Biomimetics*, IEEE, 2006, pp. 549–554.

[29] H. Mu, "Third order point-to-point motion-profile," Aug. 2009.

[30] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[31] A. A. Hayat, R. G. Chittawadigi, A. D. Udai, and S. K. Saha, "Identification of denavit-hartenberg parameters of an industrial robot," in *Proceedings of conference on advances in robotics*, 2013, pp. 1–6.

[32] P. I. Corke, "A simple and systematic approach to assigning denavit–hartenberg parameters," *IEEE transactions on robotics*, vol. 23, no. 3, pp. 590–594, 2007.

[33] S. Kucuk and Z. Bingul, *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

[34] R. Singh, V. Kukshal, and V. S. Yadav, "A review on forward and inverse kinematics of classical serial manipulators," *Advances in Engineering Design: Select Proceedings of ICOIED 2020*, pp. 417–428, 2021.

[35] S. DİKMENLİ, "Forward & inverse kinematics solution of 6-dof robots those have offset & spherical wrists," *Eurasian Journal of Science Engineering and Technology*, vol. 3, no. 1, pp. 14–28, 2022.

[36] A. Khatamian, "Solving kinematics problems of a 6-dof robot manipulator," in *International conference scientific computing*, vol. 2, 2015.

[37] C. Lee and M. Ziegler, "Geometric approach in solving inverse kinematics of puma robots," *IEEE Transactions on Aerospace and Electronic Systems*, no. 6, pp. 695–706, 1984.

[38] J.-D. Sun, G.-Z. Cao, W.-B. Li, Y.-X. Liang, and S.-D. Huang, "Analytical inverse kinematic solution using the dh method for a 6-dof robot," in *2017 14th international conference on ubiquitous robots and ambient intelligence (URAI)*, IEEE, 2017, pp. 714–716.

[39] M.-A. Martínez-Prado, J. Rodríguez-Reséndiz, R.-A. Gómez-Loenzo, G. Herrera-Ruiz, and L.-A. Franco-Gasca, "An fpga-based open architecture industrial robot controller," *IEEE Access*, vol. 6, pp. 13 407–13 417, 2018. DOI: 10.1109/ACCESS.2018.2797803.

[40] B. You, D. Li, and S. Liu, "Design of dsp-based open control system for industrial robot," in *2007 IEEE International Conference on Automation and Logistics*, 2007, pp. 1585–1590. DOI: 10.1109/ICAL.2007.4338825.

[41] M. H. Bomfim, R. A. Gontijo, A. Q. Bracarense, and E. J. Lima, "Methodology for remanufacturing of industrial robotic manipulators using open control architecture," in *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, 2013, pp. 47–52. DOI: 10.1109/ICICIP.2013.6568038.

[42] FANUC, *FANUC Robotics R-J Controller S-420FD, S-420FD Long Arm Mechanical Maintenance*. FANUC Robotics America, 2003.

[43] FANUC, *SYSTEM R-J Controller Series Electrical Connection and Maintenance Manual*. FANUC Robotics North America, 2000.

[44] S. Microelectronics, "An2834 - how to optimize the adc accuracy in the stm32 mcus," *ST Microelectronics, USA*, 2023.

[45] Molex, *Molex 430452400 part detail*, Accessed: 15-12-2023. [Online]. Available: `https://www.molex.com/en-us/products/part-detail/430452400`.

[46] Z. H. E. Co, *Hx connector part detail*. [Online]. Available: `https://www.hxdy.com/pro_bigpic.asp?id=5&Big_Class=1&Small_Class=3`.

[47] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[48] C. Waguespack, *Mastering Autodesk Inventor 2014 and Autodesk Inventor LT 2014: Autodesk Official Press*. John Wiley & Sons, 2013.

[49] J. M. Willman, "Creating guis with qt designer," in *Beginning PyQt: A Hands-on Approach to GUI Programming with PyQt6*, Springer, 2022, pp. 217–258.

[50] L. Stemkoski and M. Pascale, *Developing Graphics Frameworks with Python and OpenGL*. Taylor & Francis, 2021.

[51] T. O. Hodson, T. M. Over, and S. S. Foks, "Mean squared error, deconstructed," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 12, 2021.

Appendix A

Codes

A.1 Forward Kinematics function

```
void calcForward(float robotAngles[6], float robotPosition[6]) {
  std::vector<Frame> DH_Frames;
  DH_Frames.push_back(workFrame);

  float finalAngles[6];
  for (int i = 0; i < 6; i++)
    finalAngles[i] = radians(initialAngles[i] + robotAngles[i]);

  for (int i = 0; i < 6; i++) {
    float matrixAux[4][4] = {
      {
        cos(finalAngles[i]),
        -sin(finalAngles[i]) * cos(DH_Parameters[i][2]),
        sin(finalAngles[i]) * sin(DH_Parameters[i][2]),
        DH_Parameters[i][1] * cos(finalAngles[i]),
      },
      {
        sin(finalAngles[i]),
        cos(DH_Parameters[i][2]) * cos(finalAngles[i]),
        -cos(finalAngles[i]) * sin(DH_Parameters[i][2]),
        DH_Parameters[i][1] * sin(finalAngles[i]),
      },
      {0, sin(DH_Parameters[i][2]), cos(DH_Parameters[i][2]), DH_Parameters[i][0]},
      {0, 0, 0, 1},
    };
    DH_Frames.push_back(Frame(matrixAux));
  }
```

```
28
29    DH_Frames.push_back(toolFrame);
30
31    std::vector<Frame> transformationMatrix;
32
33    float aux[4][4];
34    Matrix.Multiply((float*)DH_Frames[0].matrix, (float*)DH_Frames[1].matrix, 4, 4, 4,
      ↪  (float*)aux);
35    transformationMatrix.push_back(Frame(aux));
36
37    for (int i = 0; i < DH_Frames.size() - 2; i++) {
38      Matrix.Multiply((float*)transformationMatrix[i].matrix, (float*)DH_Frames[i +
        ↪  2].matrix, 4, 4, 4, (float*)aux);
39      transformationMatrix.push_back(Frame(aux));
40    }
41
42    robotPosition[0] = transformationMatrix[6].matrix[0][3];
43    robotPosition[1] = transformationMatrix[6].matrix[1][3];
44    robotPosition[2] = transformationMatrix[6].matrix[2][3];
45
46    robotPosition[3] = degrees(atan2(transformationMatrix[6].matrix[1][2],
      ↪  transformationMatrix[6].matrix[0][2]));
47    robotPosition[4] = degrees(atan2(sqrt(1 - pow(transformationMatrix[6].matrix[2][2],
      ↪  2)), transformationMatrix[6].matrix[2][2]));
48    robotPosition[5] = degrees(atan2(transformationMatrix[6].matrix[2][1],
      ↪  -transformationMatrix[6].matrix[2][0]));
49  }
```

A.2 Inverse Kinematics function

```
1  void calcInverse(float robotPosition[6], float robotAngles[6]) {
2    float alpha = radians(robotPosition[3]);
3    float beta = radians(robotPosition[4]);
4    float gamma = radians(robotPosition[5]);
5    float c1 = cos(alpha);
6    float c2 = cos(beta);
7    float c3 = cos(gamma);
8    float s1 = sin(alpha);
9    float s2 = sin(beta);
10   float s3 = sin(gamma);
11
12   float positionMatrix_Tool2Base[4][4] = {
13     {(c1 * c2 * c3) - (s1 * s3), (-c3 * s1) - (c1 * c2 * s3), c1 * s2,
       ↪  robotPosition[0]},
14     {(c1 * s3) + (c2 * c3 * s1), (c1 * c3) - (c2 * s1 * s3), s1 * s2,
       ↪  robotPosition[1]},
15     {-c3 * s2, s2 * s3, c2, robotPosition[2]},
16     {0, 0, 0, 1} };
17
18   float positionMatrix_EE2Base[4][4];
19   Matrix.Multiply((float*)positionMatrix_Tool2Base, (float*)float_toolFrame_inverted,
     ↪  4, 4, 4, (float*)positionMatrix_EE2Base);
20
21   robotPosition[0] = positionMatrix_EE2Base[0][3];
22   robotPosition[1] = positionMatrix_EE2Base[1][3];
23   robotPosition[2] = positionMatrix_EE2Base[2][3];
24
25   /*-------------- Theta 1 --------------*/
26   robotAngles[0] = atan2(robotPosition[1] - (DH_Parameters[5][0] *
     ↪  positionMatrix_EE2Base[1][2]), robotPosition[0] - (DH_Parameters[5][0] *
     ↪  positionMatrix_EE2Base[0][2]));
27   /*-------------- Theta 1 --------------*/
28
29   /*-------------- Theta 3 --------------*/
30   float P14[3][1] = {
31     {robotPosition[0] - (DH_Parameters[5][0] * positionMatrix_EE2Base[0][2]) -
       ↪  (DH_Parameters[0][1] * cos(robotAngles[0]))},
32     {robotPosition[1] - (DH_Parameters[5][0] * positionMatrix_EE2Base[1][2]) -
       ↪  (DH_Parameters[0][1] * sin(robotAngles[0]))},
33     {robotPosition[2] - (DH_Parameters[5][0] * positionMatrix_EE2Base[2][2]) -
       ↪  DH_Parameters[0][0]} };
34
35   float P14L = sqrt(P14[0][0] * P14[0][0] + P14[1][0] * P14[1][0] + P14[2][0] *
     ↪  P14[2][0]);
```

```
36
37    float L1 = sqrt(pow(DH_Parameters[2][1], 2) + pow(DH_Parameters[3][0], 2));
38    float fi = acos((pow(L1, 2) + pow(DH_Parameters[1][1], 2) - pow(P14L, 2)) / (2 * L1
      ↪  * DH_Parameters[1][1]));
39    float epsilon = atan2(DH_Parameters[3][0], DH_Parameters[2][1]);
40    robotAngles[2] = fi + epsilon - PI;
41    /*-------------- Theta 3 --------------*/
42
43    /*-------------- Theta 2 --------------*/
44    float beta_1 = atan2(P14[2][0], sqrt(pow(P14[0][0], 2) + pow(P14[1][0], 2)));
45    float beta_2 = acos((pow(DH_Parameters[1][1], 2) + pow(P14L, 2) - pow(L1, 2)) / (2 *
      ↪  DH_Parameters[1][1] * P14L));
46    robotAngles[1] = beta_1 + beta_2 - PI / 2;
47    /*-------------- Theta 2 --------------*/
48
49    /*-------------- Theta 5 --------------*/
50    float R16_z[3] = { positionMatrix_EE2Base[0][2], positionMatrix_EE2Base[1][2],
      ↪  positionMatrix_EE2Base[2][2] };
51    float C1 = cos(robotAngles[0]);
52    float C2 = cos(robotAngles[1] + PI / 2);
53    float C3 = cos(robotAngles[2]);
54    float S1 = sin(robotAngles[0]);
55    float S2 = sin(robotAngles[1] + PI / 2);
56    float S3 = sin(robotAngles[2]);
57    float C23 = C2 * C3 - S2 * S3;
58    float S23 = C2 * S3 + S2 * C3;
59    float R14_z[3] = { C1 * S23, S1 * S23, -C23 };
60    float R16_z_MULT_R14_z[1];
61    Matrix.Multiply((float*)R16_z, (float*)R14_z, 1, 3, 1, (float*)R16_z_MULT_R14_z);
62    robotAngles[4] = acos(R16_z_MULT_R14_z[0]);
63    /*-------------- Theta 5 --------------*/
64
65    /*-------------- Theta 4 --------------*/
66    float A_13[4][4] = { {C1 * C23, S1, C1 * S23, C1 * (DH_Parameters[0][1] +
      ↪  (DH_Parameters[1][1] * C2) + DH_Parameters[2][1] * C23)},
67      {S1 * C23, -C1, S1 * S23, S1 * (DH_Parameters[0][1] + (DH_Parameters[1][1] * C2) +
        ↪  DH_Parameters[2][1] * C23)},
68      {S23, 0, -C23, (DH_Parameters[1][1] * S2) + DH_Parameters[0][0] +
        ↪  DH_Parameters[2][1] * (C3 * S2 + S3 * C2)},
69      {0, 0, 0, 1} };
70    Matrix.Invert((float*)A_13, 4);
71    float A_46[4][4];
72    Matrix.Multiply((float*)A_13, (float*)positionMatrix_EE2Base, 4, 4, 4,
      ↪  (float*)A_46);
73    robotAngles[3] = atan2(A_46[1][2], A_46[0][2]);
74    /*-------------- Theta 4 --------------*/
75
```

```
76    /*-------------- Theta 6 --------------*/
77    robotAngles[5] = atan2(A_46[2][1], -A_46[2][0]);
78    /*-------------- Theta 6 --------------*/
79  }
```

A.3 Trajectory calculation functions

```
void TrajectoryControl::calcMotionConstants(float _jerkMax, float _accelMax, float
↪  _velMax, float _pos) {
  jerkMax = _jerkMax;
  accelMax = _accelMax;
  velMax = _velMax;
  pos = _pos;

  byte tShape = 0;

  float va = (pow(accelMax, 2) / jerkMax);
  float sa = (2 * (pow(accelMax, 3) / pow(jerkMax, 2)));

  float M, N;
  if ((velMax * jerkMax) < pow(accelMax, 2)) {
    M = 1;
    N = 0;
  }
  else {
    M = 0;
    N = 1;
  }
  float sv = velMax * ((M * 2 * sqrt(velMax / jerkMax)) + (N * ((velMax / accelMax) +
  ↪  (accelMax / jerkMax))));

  if ((velMax < va) && (pos >= sa))
    tShape = 1;
  else if ((velMax >= va) && (pos < sa))
    tShape = 2;
  else if ((velMax < va) && (pos < sa) && (pos >= sv))
    tShape = 3;
  else if ((velMax < va) && (pos < sa) && (pos < sv))
    tShape = 4;
  else if ((velMax >= va) && (pos >= sa) && (pos >= sv))
    tShape = 5;
  else if ((velMax >= va) && (pos >= sa) && (pos < sv))
    tShape = 6;

  float tj, ta, tv;
  if (tShape == 1 || tShape == 3) {
    tj = sqrt(velMax / jerkMax);
    ta = tj;
    tv = pos / velMax;
  }
  else if (tShape == 2 || tShape == 4) {
```

```
43      tj = pow(pos / (2 * jerkMax), 1.0 / 3.0);
44      ta = tj;
45      tv = 2 * tj;
46    }
47    else if (tShape == 5) {
48      tj = accelMax / jerkMax;
49      ta = velMax / accelMax;
50      tv = pos / velMax;
51    }
52    else if (tShape == 6) {
53      tj = accelMax / jerkMax;
54      ta = 0.5 * (sqrt(((4 * pos * pow(jerkMax, 2)) + pow(accelMax, 3)) / (accelMax *
        ↪  pow(jerkMax, 2))) - (accelMax / jerkMax));
55      tv = ta + tj;
56    }
57
58    t[0] = tj;
59    t[1] = ta;
60    t[2] = ta + tj;
61    t[3] = tv;
62    t[4] = tv + tj;
63    t[5] = tv + ta;
64    t[6] = tv + tj + ta;
65  }
66
67  bool TrajectoryControl::calcMotion() {
68    bool isMotionCompleted_return = false;
69
70    if (startTrajectoryFlag && comfirmTOFlag) {
71      T = (micros() - t0) / 1000000.0;
72
73      if ((T < t[0])) {
74        j[0] = jerkMax;
75        a[0] = jerkMax * T;
76        v[0] = 0.5 * jerkMax * pow(T, 2);
77        p[0] = (1.0 / 6.0) * jerkMax * pow(T, 3);
78        tControlPhase = 0;
79      }
80      else if ((T >= t[0]) && (T < t[1])) {
81        j[1] = 0;
82        a[1] = a[0];
83        v[1] = v[0] + a[0] * (T - t[0]);
84        p[1] = p[0] + v[0] * (T - t[0]) + 0.5 * a[0] * pow((T - t[0]), 2);
85        tControlPhase = 1;
86      }
87      else if ((T >= t[1]) && (T < t[2])) {
88        j[2] = -jerkMax;
```

```
89        a[2] = a[1] - (jerkMax * (T - t[1]));
90        v[2] = v[1] + a[1] * (T - t[1]) + 0.5 * -jerkMax * pow((T - t[1]), 2);
91        p[2] = p[1] + v[1] * (T - t[1]) + 0.5 * a[1] * pow((T - t[1]), 2) + (1.0 / 6.0)
          ↪  * -jerkMax * pow((T - t[1]), 3);
92        tControlPhase = 2;
93      }
94      else if ((T >= t[2]) && (T < t[3])) {
95        j[3] = 0;
96        a[3] = 0;
97        v[3] = v[2];
98        p[3] = p[2] + v[2] * (T - t[2]);
99        tControlPhase = 3;
100     }
101     else if ((T >= t[3]) && (T < t[4])) {
102       j[4] = -jerkMax;
103       a[4] = -jerkMax * (T - t[3]);
104       v[4] = v[3] + 0.5 * -jerkMax * pow((T - t[3]), 2);
105       p[4] = p[3] + v[3] * (T - t[3]) + (1.0 / 6.0) * -jerkMax * pow((T - t[3]), 3);
106       tControlPhase = 4;
107     }
108     else if ((T >= t[4]) && (T < t[5])) {
109       j[5] = 0;
110       a[5] = a[4];
111       v[5] = v[4] + a[4] * (T - t[4]);
112       p[5] = p[4] + v[4] * (T - t[4]) + 0.5 * a[4] * pow((T - t[4]), 2);
113       tControlPhase = 5;
114     }
115     else if ((T >= t[5]) && (T < t[6])) {
116       j[6] = jerkMax;
117       a[6] = a[5] + (jerkMax * (T - t[5]));
118       v[6] = v[5] + a[5] * (T - t[5]) + 0.5 * jerkMax * pow((T - t[5]), 2);
119       p[6] = p[5] + v[5] * (T - t[5]) + 0.5 * a[5] * pow((T - t[5]), 2) + (1.0 / 6.0)
          ↪  * jerkMax * pow((T - t[5]), 3);
120       tControlPhase = 6;
121     }
122     else if (T >= t[6]) {
123       isMotionCompleted_return = true;
124       startTrajectoryFlag = 0;
125     }
126
127     for (int i = 0; i < 6; i++) {
128       if (t[i] == t[i + 1]) {
129         j[i + 1] = j[i];
130         a[i + 1] = a[i];
131         v[i + 1] = v[i];
132         p[i + 1] = p[i];
133       }
```

```
134       }
135
136       tControlJerk = j[tControlPhase];
137       tControlAccel = a[tControlPhase];
138       tControlVelocity = v[tControlPhase];
139       tControlPosition = p[tControlPhase];
140     }
141     else {
142       t0 = micros();
143       comfirmT0Flag = 1;
144     }
145
146     return isMotionCompleted_return;
147   }
```

Appendix B

Circuit Sheets

B.1 Dual Axis Control Board circuit

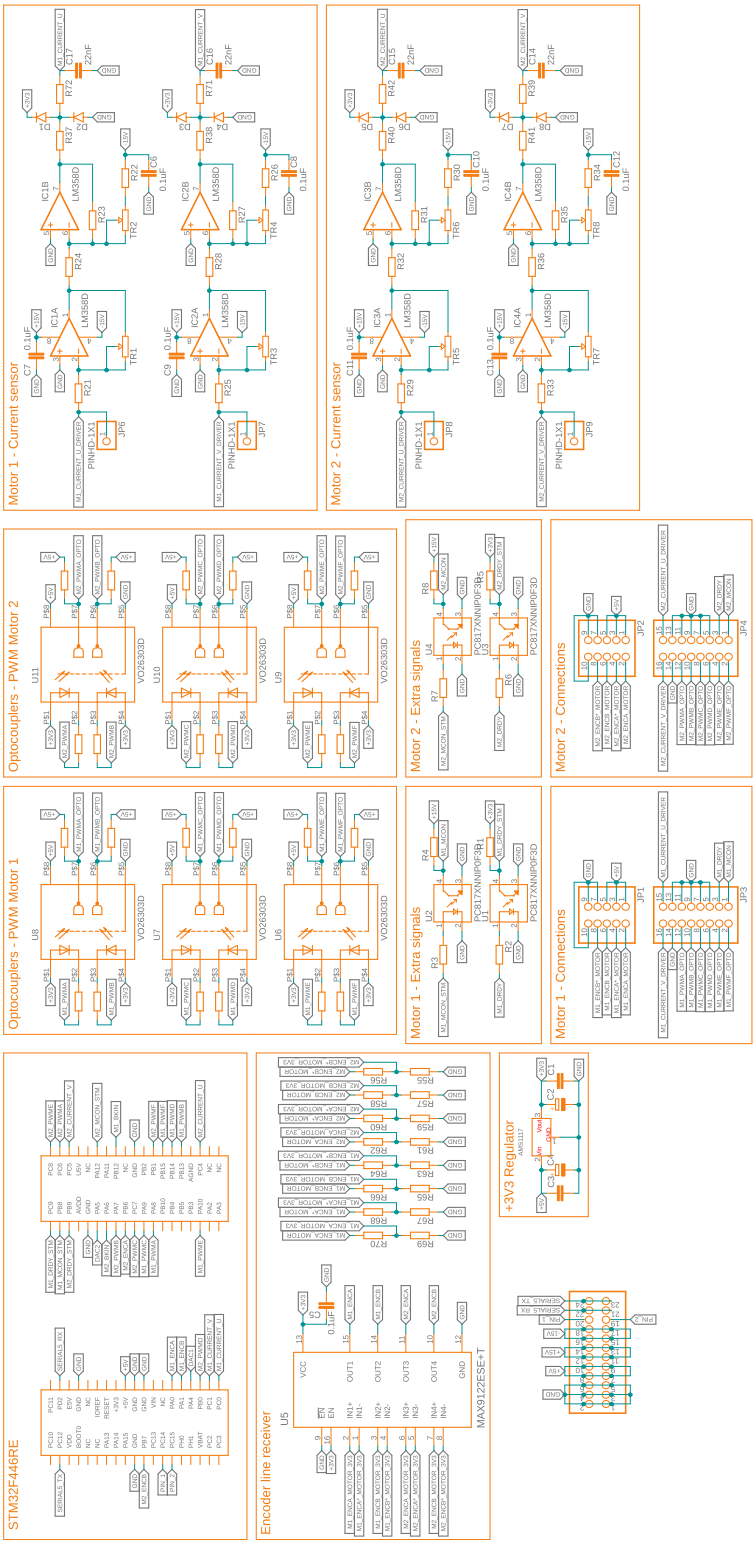


Figure B.1: Dual Axis Control Board circuit diagram.

B.2 Main Board circuit

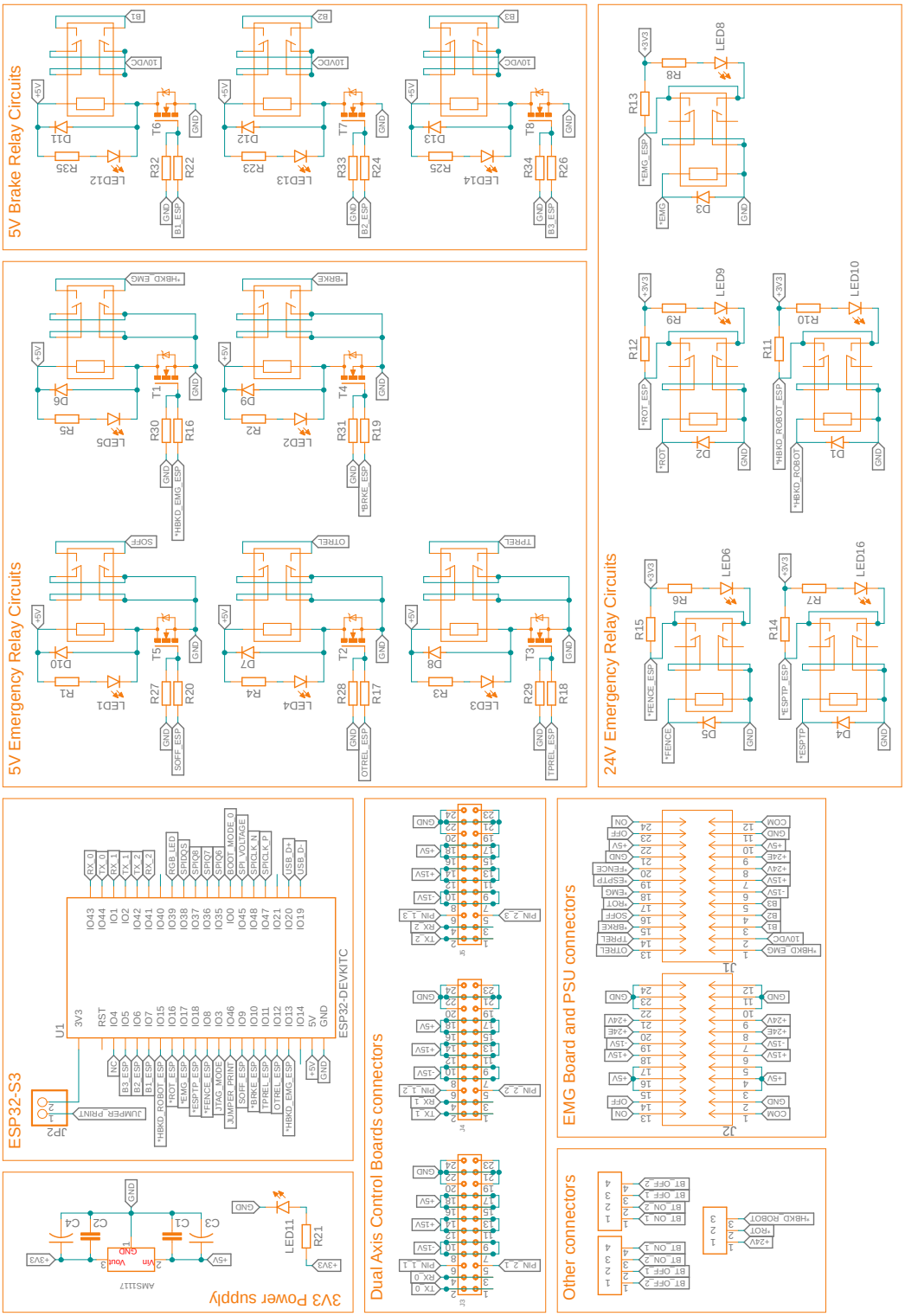


Figure B.2: Main Board circuit diagram.